亚马逊云科技

用户指南

Amazon CloudFormation



API 版本 2010-05-15

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon CloudFormation: 用户指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅 中国的 Amazon Web Services 服务入门 (PDF)。

Table of Contents

什么是 Amazon CloudFormation？ 1
简化基础设施管理 1
快速复制您的基础设施 1
轻松控制和跟踪对您的基础设施所做的更改 1
CloudFormation 入门 2
相关信息 2
开始使用 3
CloudFormation 的工作原理 3
重要概念 3
CloudFormation 的工作原理 7
CloudFormation 入门方式 10
注册 Amazon Web Services 账户 10
注册 Amazon Web Services 账户 10
保护 IAM 用户 11
创建第一个堆栈 11
先决条件 12
使用控制台创建 CloudFormation 堆栈 12
监控堆栈创建 15
测试 Web 服务器 16
问题排查 17
清理 18
后续步骤 18
最佳实践 20
缩短反馈循环以提高开发速度 21
按生命周期和所有权组织您的堆栈 22
使用跨堆栈引用在堆栈之间共享值 22
使用 Amazon CloudFormation StackSets 进行多账户和多区域部署 23
验证所有资源类型的配额 23
重复使用模板以在多个环境中复制堆栈 23
使用模块重复使用资源配置 23
采用基础设施即代码实践 24
不要在模板中嵌入凭证 24
使用 Amazon 特定的参数类型 24
使用参数约束 25

使用伪参数来提高可移植性 25
使用 AWS::CloudFormation::Init 在 Amazon EC2 实例上部署软件应用程序 26
使用最新的帮助程序脚本 26
在使用模板前先验证模板 26
验证模板的组织策略合规性 27
使用 YAML 或 JSON 进行模板编写 27
实施全面的标记策略 28
利用模板宏进行高级转换 28
通过 CloudFormation 管理所有堆栈资源 28
更新堆栈之前创建更改集 28
使用堆栈策略保护资源 29
使用 Amazon CloudTrail 记录 CloudFormation 调用 29
使用代码审查和修订控制来管理您的模板 29
定期更新您的 Amazon EC2 实例 29
定期使用偏移检测 30
配置回滚触发器以实现自动恢复 30
实施有效的堆栈重构策略 30
使用 CloudFormation Hooks 进行生命周期管理 30
使用 IaC 生成器从现有资源创建模板 31
使用模板工作室直观地编辑模板 31
考虑对复杂的基础设施使用 Amazon Cloud Development Kit (Amazon CDK) 32
使用 IAM 控制访问 32
采用最低权限原则 32
保护敏感参数 32
通过 Guard 实施策略即代码 33
使用模板 34
模板的存储位置 35
验证模板 35
与人工智能编程代理配合使用 36
了解验证范围 36
开始使用模板 2
计划使用 CloudFormation 模板引用 36
示例模板 37
模板格式 37
模板结构 37
评论 39

规格 40
了解更多 40
使用正则表达式 41
模板部分 42
Resources 43
Parameters 51
Outputs 63
Mappings 68
Metadata 75
Rules 81
Conditions 101
Transform 112
格式版本 113
Description 113
CloudFormation 语言服务器 114
IDE 与 CloudFormation 语言服务器集成 114
开始使用 115
在 IDE 中初始化 CloudFormation 项目（仅限 VS Code） 120
需要帮助？ 121
CloudFormation Linter 121
安装 cfn-lint 122
运行 cfn-lint 122
了解结果 123
配置检查 124
添加自定义规则 125
向您的工作流添加 cfn-lint 126
CloudFormation 验证 127
验证的工作原理 127
选择验证方法 128
安装命令行工具 128
运行验证 128
了解结果 129
筛选结果 129
添加自定义规则 130
验证新的资源类型和属性 131
嵌入验证库 131

将 与 结合使用Amazon CDK 133
CloudFormation Guard 133
安装 Guard 134
了解 Guard 语言 134
编写规则 135
验证模板 136
测试规则 137
从模板生成规则 138
向您的工作流添加 Guard 138
模板工作室 139
注意事项 139
模板工作室所需的权限 140
打开模板工作室 140
使用模板和示意图 141
在 YAML 和 JSON 之间进行转换 142
键入时出现语法问题 142
验证模板 143
将模板保存到 Amazon S3 143
复制或下载模板或示意图 144
IaC 生成器 144
注意事项 145
权限 146
常用命令 146
将模板迁移到 Amazon CDK 147
启动资源扫描 147
查看扫描摘要 151
根据扫描的资源创建模板 152
根据扫描的资源创建堆栈 157
解决只写属性问题 158
获取存储在其他服务中的值 162
一般注意事项 162
获取 Systems Manager 纯文本值 163
获取 Systems Manager 安全字符串 167
获取 Secrets Manager 值 170
获取 Amazon 值 174
语法 174

可用伪参数 175
示例 176
获取堆栈输出 180
导出堆栈输出值与使用嵌套堆栈 180
注意事项 181
列出导出的输出值 181
列出导入已导出输出值的堆栈 182
在不导出的情况下引用输出 183
在运行时指定现有资源 184
概述 184
示例 185
注意事项 188
支持的特定于 Amazon 的参数类型 189
支持的 Systems Manager 参数类型 191
不支持的 Systems Manager 参数类型 192
演练 192
引用其他 CloudFormation 堆栈中的资源输出 192
在 Amazon EC2 上部署应用程序 197
更新堆栈 206
创建已扩展且负载均衡的应用程序 221
与另一个账户中的 VPC 对等 234
执行 ECS 蓝/绿部署 244
模板代码段 267
General 268
自动扩缩 278
Amazon 账单控制台 309
Amazon CloudFormation 312
CloudFront 319
CloudWatch 331
CloudWatch Logs 337
DynamoDB 370
Amazon EC2 375
Amazon ECS 410
Amazon EFS 433
Elastic Beanstalk 458
Elastic Load Balancing 463

IAM ... 468
Amazon Lambda ... 485
Amazon Redshift ... 492
Amazon RDS ... 501
Route 53 ... 510
Amazon S3 ... 518
Amazon SNS ... 526
Amazon SQS ... 527
Amazon Timestream ... 528
基于 Windows 的堆栈 ... 556
引导 Windows 堆栈 ... 558
使用 CloudFormation 提供的资源类型 ... 564
自定义资源 ... 565
模板宏 ... 606
嵌套堆栈 ... 622
等待条件 ... 628
使用模块创建可重复使用的资源配置 ... 634
使用模块时的注意事项 ... 635
了解模块版本控制 ... 635
在模板中使用模块 ... 637
使用参数指定模块值 ... 638
引用模块资源 ... 640
创建和管理堆栈 ... 642
用于管理堆栈的界面 ... 643
代理设置指南 ... 644
与您的编码代理协作 ... 644
先决条件 ... 644
安装内容 ... 645
Claude Code ... 645
Codex ... 646
Cursor ... 646
Kiro ... 647
GitHub Copilot ... 647
Windsurf ... 648
OpenCode ... 648
适用于与代理技能和 MCP 服务器配置兼容的所有其他代理 ... 649

代理可以为您做些什么 650
IAM 权限 650
相关资源 651
创建堆栈 651
创建堆栈 651
配置堆栈选项 654
预览堆栈的配置 656
查看堆栈信息 657
更新堆栈模板 659
理解堆栈资源的更新行为 661
使用更改集更新堆栈 662
创建更改集 664
查看更改集 669
偏差感知更改集 677
执行更改集 682
删除更改集 684
示例更改集 685
嵌套堆栈的更改集 697
验证堆栈部署 703
部署前验证的工作原理 704
注意事项 704
先决条件 705
禁用部署前验证 705
验证堆栈部署（控制台） 706
验证堆栈部署（Amazon CLI） 707
验证类型 709
资源限制 710
直接更新堆栈 724
取消堆栈更新 728
取消堆栈更新（控制台） 728
取消堆栈更新（Amazon CLI） 728
删除堆栈 728
相关资源 730
查看已删除的堆栈 731
监控堆栈进度 731
查看堆栈事件 732

按操作查看堆栈事件 ... 739
查看堆栈部署时间线图 ... 743
了解堆栈创建事件 ... 746
监控堆栈更新 ... 748
继续回滚更新 ... 750
确定堆栈失败的原因 ... 755
堆栈故障选项 ... 756
在突破警报阈值时回滚堆栈 ... 765
在堆栈创建或更新期间添加回滚触发器 ... 766
将回滚触发器添加到更改集 ... 767
查看堆栈的回滚触发器 ... 767
查看更改集的回滚触发器 ... 767
检测堆栈和资源的非托管配置更改 ... 767
什么是偏差？ ... 768
偏差检测状态代码 ... 769
检测偏差时的注意事项 ... 771
在整个 CloudFormation 堆栈上检测偏差 ... 773
在单个堆栈资源上检测偏差 ... 778
通过导入操作消除偏差 ... 782
导入 Amazon 资源 ... 791
手动导入 Amazon 资源 ... 792
自动导入 Amazon 资源 ... 827
恢复导入操作 ... 854
堆栈重构 ... 860
堆栈重构的工作原理 ... 860
堆栈重构注意事项 ... 861
先决条件 ... 861
重构堆栈（控制台） ... 861
重构堆栈（Amazon CLI） ... 863
资源限制 ... 867
资源类型支持 ... 871
使用快速创建链接创建 CloudFormation 堆栈 ... 951
URL 格式 ... 951
示例 ... 952
使用快速创建链接创建堆栈 ... 952
Express Mode ... 953

何时使用 Express Mode 954
Express Mode 的工作原理 954
CloudFormation 在 Express Mode下如何处理依赖关系 955
Express Mode 入门 955
Express Mode 下的回滚行为 957
Express Mode 和嵌套堆栈 957
资源就绪性注意事项 957
不支持的功能和限制 958
与 Amazon CDK 热交换部署的对比 958
相关资源 959
了解堆栈部署时间 959
CloudFormation 如何预置堆栈 959
哪些因素影响部署时间 960
缩短部署时间的方法 960
相关资源 960
Amazon CLI 和 PowerShell 的命令示例 961
CancelUpdateStack 961
ContinueUpdateRollback 962
CreateStack 963
部署 967
DeleteStack 968
DescribeStackEvents 969
DescribeStackResource 971
DescribeStackResources 972
DescribeStacks 975
GetTemplate 979
ListStackResources 980
ListStacks 982
UpdateStack 984
ValidateTemplate 988
将本地构件上传到 S3 存储桶 990
使用 StackSets 管理堆栈 993
StackSets 概念 994
管理员账户和目标账户 994
Amazon CloudFormation 堆栈集 995
StackSet 的权限模型 995

堆栈实例 996
StackSet 操作 996
StackSet 操作选项 997
标签 999
StackSets 状态代码 999
堆栈实例状态代码 1000
先决条件 1001
启用默认禁用的 Amazon Web Services 区域 1002
授予自行管理的权限 1002
激活可信访问权限 1017
使用示例模板开始使用。 1021
先决条件 1022
使用控制台中的示例模板创建 StackSet 1023
监控 StackSet 创建 1025
查看 StackSet 结果 1025
更新 StackSet 1026
将堆栈添加到 StackSet 1026
清理 1027
后续步骤 1028
创建 StackSets（自行管理权限） 1029
创建具有自行管理权限的 StackSet（控制台） 1029
创建具有自行管理权限的 StackSet（Amazon CLI） 1031
创建 StackSets（服务托管权限） 1032
注意事项 1032
创建具有服务管理权限的 StackSet（控制台） 1033
创建具有服务管理权限的 StackSet（Amazon CLI） 1035
启用和禁用自动部署 1039
自动部署的工作原理 1039
注意事项 1039
启用或禁用自动部署（控制台） 1040
启用或禁用自动部署（Amazon CLI） 1040
更新 StackSets 1041
更新 StackSet（控制台） 1041
更新 StackSet（Amazon CLI） 1043
将堆栈添加到 StackSets 1044
将堆栈添加到 StackSet（控制台） 1044

将堆栈添加到 StackSet（Amazon CLI） 1046
覆盖堆栈参数 1047
覆盖堆栈参数（控制台） 1047
覆盖堆栈参数（Amazon CLI） 1049
从堆栈集中删除堆栈 1050
从 StackSet 中删除堆栈（控制台） 1050
从 StackSet 中删除堆栈（Amazon CLI） 1051
删除 StackSets 1053
删除 StackSet（控制台） 1053
删除 StackSet（Amazon CLI） 1053
删除服务角色 (可选) 1054
目标账户门 1055
要求 1055
用于创建 Lambda 函数的 CloudFormation 模板 1056
选择并发模式 1056
每种并发模式的工作原理 1056
根据部署速度在“严格容错”和“软容错”之间进行选择 1059
选择并发模式（控制台） 1060
选择并发模式（Amazon CLI） 1060
检测 StackSets 上的偏差 1061
CloudFormation 如何对 StackSet 执行偏移检测 1061
检测 StackSet 上的偏移（控制台） 1062
检测 StackSet 上的偏移（Amazon CLI） 1063
停止对 StackSet 的偏移检测 1070
将堆栈导入 StackSet 1070
自行管理堆栈的导入 1071
服务托管堆栈的导入 1074
恢复堆栈导入 1079
最佳实践 1079
定义模板 1080
创建堆栈或将堆栈添加到 StackSet 1080
更新 StackSet 中的堆栈 1080
示例 模板 1080
问题排查 1082
堆栈操作失败的常见原因 1082
重试失败的堆栈创建或更新操作 1083

堆栈实例删除操作失败 1084
堆栈导入操作失败 1084
StackSets 操作的堆栈实例失败计数 1084
将堆栈与 Git 源代码同步 1091
Git 同步的工作原理 1092
Git 同步的工作原理 1092
关于拉取请求的注释 1093
堆栈部署文件 1093
CloudFormation 模板文件 1094
模板定义存储库 1094
先决条件 1094
Git 存储库 1095
CloudFormation 模板 1095
Git 同步服务角色 1095
控制台用户的 IAM 权限 1098
从存储库源代码创建堆栈 1101
从存储库源代码创建堆栈 1101
从 Git 存储库更新堆栈 1104
启用拉取请求注释 1104
状态控制面板 1105
Git 同步状态 1105
最新同步事件 1106
支持的堆栈状态 1106
通过 CloudFormation 注册表管理扩展 1110
相关文档 1110
概念 1111
扩展类型 1111
公有扩展 1112
已激活扩展 1112
查看可用和已激活的扩展 1113
公有扩展 1113
IAM 角色 1114
自动使用新版本的扩展 1116
使用别名以引用扩展 1117
处理公有扩展的常用 Amazon CLI 命令 1117
激活公有扩展 1117

更新公有扩展 1121
停用公有扩展 1122
私有扩展 1123
IAM 权限 1124
处理私有扩展的常用 Amazon CLI 命令 1124
注册私有扩展 1124
取消注册私有扩展 1127
编辑扩展的配置数据 1128
使用动态引用所需的权限 1128
编辑扩展的配置数据（控制台） 1129
编辑扩展的配置数据（Amazon CLI） 1130
在 Amazon Config 中记录资源类型 1130
防止在配置项目中记录敏感属性 1131
持续交付 1132
演练：为测试和生产堆栈构建管道 1132
先决条件 1132
演练概述 1133
步骤 1：编辑构件并将构件上传到 S3 存储桶 1133
步骤 2：创建管道堆栈 1135
步骤 3：查看 WordPress 堆栈 1140
步骤 4：清理资源 1140
另请参阅 1141
配置属性参考 1142
配置属性 (控制台) 1142
配置属性 (JSON 对象) 1144
另请参阅 1148
Amazon CloudFormation 构件 1149
堆栈模板文件 1149
模板配置文件 1150
另请参阅 1151
将参数覆盖函数用于 CodePipeline 管道 1151
Fn::GetArtifactAtt 1151
Fn::GetParam 1153
另请参阅 1155
模板参考指南 1156
安全性 1157

堆栈删除保护 1158
控制谁可以在堆栈上更改终止保护 1159
防止更新堆栈资源 1160
示例堆栈策略 1160
定义堆栈策略 1161
设置堆栈策略 1165
更新受保护资源 1166
修改堆栈策略 1169
更多示例堆栈策略 1170
数据保护 1173
静态加密 1174
传输中加密 1174
互联网络流量隐私 1174
使用 IAM 控制访问权限 1174
为 CloudFormation 定义基于 IAM 身份的策略 1175
确认 CloudFormation 模板中的 IAM 资源 1183
管理 Amazon EC2 实例上运行的应用程序的凭证 1184
授予临时访问权限（联合访问） 1184
基于身份的策略示例 1185
Amazon 托管策略 1193
Amazon CloudFormation 服务角色 1195
防止跨服务混淆代理 1196
FAS 请求和权限评估 1197
记录 API 调用 1200
CloudTrail 中的 CloudFormation 信息 1200
了解 CloudFormation 日志文件条目 1201
基础结构安全性 1205
故障恢复能力 1205
合规性验证 1206
配置和漏洞分析 1206
安全最佳实践 1206
使用 IAM 控制访问 1206
请勿将凭证嵌入您的模板 1207
使用 Amazon CloudTrail 记录 CloudFormation 调用 1207
Amazon PrivateLink 1207
Amazon CloudFormation VPC 端点注意事项 1208

为 Amazon CloudFormation 创建接口 VPC 端点 1209
为 Amazon CloudFormation创建 VPC 端点策略 1209
使用 EventBridge 监控 1211
CloudFormation 和 Git 同步事件概述 1211
权限 1213
创建自定义事件模式 1213
事件详细信息参考 1214
资源状态更改 1215
堆栈状态更改 1218
偏差检测状态更改 1220
StackSet 状态更改 1222
StackSet 堆栈实例状态更改 1223
StackSet 操作状态更改 1226
存储库同步状态更改 1228
资源同步状态更改 1230
限额 1234
功能可用性 1239
StackSet 和宏 1240
问题排查 1241
故障排除指南 1241
错误故障排除 1242
删除堆栈失败 1242
依赖关系错误 1243
Amazon Config 和 Amazon Systems Manager 冲突 1243
传递列表时解析参数出错 1244
IAM 权限不足 1244
无效的值或不受支持的资源属性 1244
超出配额 1245
嵌套堆栈回滚失败 1245
无需执行更新 1245
资源在创建、更新或删除堆栈操作期间无法稳定工作 1246
VPC 中不存在安全组 1246
更新回滚失败 1246
等待条件未收到来自 Amazon EC2 实例的所需数目的信号 1248
资源已从堆栈中移除但未删除 1248
联系支持人员 1249

部署前验证错误 1249
Express Mode 完成行为 1250
使用 Amazon Q 进行故障排除 1252
功能 1252
工作原理 1252
定价 1253
区域可用性 1253
限制 1253
文档历史记录 1254
归档更新 1271

什么是 Amazon CloudFormation？

Amazon CloudFormation 是一项服务，可帮助您对 Amazon 资源进行建模和设置，以便能花较少的时间管理这些资源，而将更多的时间花在运行于 Amazon 中的应用程序上。您创建一个描述您所需的所有 Amazon 资源（如 Amazon EC2 实例或 Amazon RDS 数据库实例）的模板，并且 CloudFormation 将负责为您预置和配置这些资源。您无需单独创建和配置 Amazon 资源并了解 what; CloudFormation 句柄处理该工作时所依赖的内容。以下方案演示 CloudFormation 如何提供帮助。

简化基础设施管理

对于还包括后端数据库的可扩展 Web 应用程序，您可使用自动扩缩组、Elastic Load Balancing 负载均衡器和 Amazon Relational Database Service 数据库实例。您可以使用每个单独的服务来预置这些资源, 创建资源后, 您需要将它们进行配置从而一起生效。在您的应用程序启动并正常运行之前，所有这些任务会增加复杂性和时间。

相反，您可创建或修改现有 CloudFormation 模板。模板描述了您的所有资源及其属性。当您使用该模板创建 CloudFormation 堆栈时，CloudFormation 将为您预置自动扩缩组、负载均衡器和数据库。成功创建堆栈之后，您的 Amazon 资源将正常运行。您可以轻松删除堆栈，这将删除堆栈中的所有资源。通过使用 CloudFormation，您可以轻松地将一组资源作为一个单元进行管理。

快速复制您的基础设施

如果您的应用程序需要其他可用性，您可在多个区域中复制它，以便在一个区域变得不可用的情况下，您的用户仍可在其他区域中使用您的应用程序。复制应用程序的难点在于它还需要您复制您的资源。您不仅需要记录您的应用程序所需的所有资源，还必须在每个区域中设置和配置这些资源。

重复使用 CloudFormation 模板，从而以一致且可重复的方式创建资源。要重复使用模板，仅描述您的资源一次，然后在多个区域中反复配置相同的资源。

轻松控制和跟踪对您的基础设施所做的更改

在某些情况下，您可能拥有增量升级所需的基础资源。例如，您可能在 Auto Scaling 启动配置中更改为更高的执行实例类型，以便您能减小自动扩缩组中的最大实例数。如果完成更新后出现问题，您可能需要将基础设施回滚到原始设置。要手动执行此操作，您不仅必须记住已发生更改的资源，还必须知道原始设置是什么。

当您使用 CloudFormation 配置基础设施时，CloudFormation 模板准确描述了所配置的资源及其设置。由于这些模板是文本文件，因此您只需跟踪模板中的区别即可跟踪对基础设施所做的更改，其方式类似于开发人员控制对源代码所做的修订的方式。例如，您可使用将版本控制系统用于模板，以便准确了解所做的更改、执行更改的人员和时间。如果您在任何时候需要撤消基础设施所做的更改，则可使用模板的上一个版本。

CloudFormation 入门

CloudFormation 可通过 CloudFormation 控制台、API、Amazon CLI、、Amazon SDK 以及通过多种集成来使用。

有关 CloudFormation 的简介，请参阅 CloudFormation 的工作原理。

要开始使用 CloudFormation，请参阅创建第一个堆栈。

相关信息

您可以在本用户指南以及以下资源中了解有关 CloudFormation 的更多信息:

- 有关产品详细信息和常见问题解答，请参阅 Amazon CloudFormation 产品页面。
- 有关定价信息，请参阅 Amazon CloudFormation 定价。

CloudFormation 入门

您可以通过 Amazon Web Services 管理控制台，使用示例模板创建一个堆栈来学习使用 CloudFormation，这有助您了解堆栈创建的基础知识。模板是一个用于定义堆栈中所有资源的文本文件。堆栈是 CloudFormation 模板的部署。您可以从单个模板创建多个堆栈。每个堆栈都包含可作为单个单元管理的 Amazon 资源集合。

CloudFormation 是一种免费服务；但是，您需要按各种资源的现行价格，为堆栈使用的 Amazon 资源付费。有关 Amazon 定价的更多信息，请转至 http://aws.amazon.com 上每种产品的详细信息页面。

主题

- CloudFormation 的工作原理
- 注册 Amazon Web Services 账户
- 创建第一个堆栈

CloudFormation 的工作原理

本主题说明了 CloudFormation 的工作原理，并介绍了在使用此服务时需要了解的关键概念。

主题

- 重要概念
- CloudFormation 的工作原理
- CloudFormation 入门方式

重要概念

在使用 CloudFormation 时，您将使用模板和堆栈。您创建模板来描述 Amazon 资源及其属性。当您创建堆栈时，CloudFormation 会配置模板中描述的资源。

主题

- 模板
- 堆栈
- 更改集

模板

CloudFormation 模板是一个 JSON 或 YAML 格式的文本文件。您可使用任何扩展名（如 `.yaml`、`.json`、`.template` 或 `.txt`）保存这些文件。CloudFormation 使用这些模板作为构建 Amazon 资源的蓝图。例如，在模板中，您可描述 Amazon EC2 实例，如实例类型、AMI ID、块储存设备映射和其 Amazon EC2 密钥对名称。当您创建堆栈时，还可以指定 CloudFormation 用来创建模板中描述的任何项的模板。

例如，如果您使用以下模板创建堆栈，则 CloudFormation 将使用 `ami-0ff8a91507f77f867` AMI ID、`t2.micro` 实例类型、`testkey` 密钥对名称和 Amazon EBS 卷来配置实例。

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "A sample template",
    "Resources": {
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0ff8a91507f77f867",
                "InstanceType": "t2.micro",
                "KeyName": "testkey",
```

```
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sdm",
                        "Ebs": {
                            "VolumeType": "io1",
                            "Iops": 200,
                            "DeleteOnTermination": false,
                            "VolumeSize": 20
                        }
                    }
                ]
            }
        }
    }
}
```

您还可以在单个模板中指定多种资源并将这些资源配置为结合使用。例如，您可以修改上一个模板来包含一个弹性 IP 地址（EIP）并将其与 Amazon EC2 实例相关联，如以下示例所示：

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
  MyEIP:
    Type: AWS::EC2::EIP
    Properties:
      InstanceId: !Ref MyEC2Instance
```

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "A sample template",
    "Resources": {
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0ff8a91507f77f867",
                "InstanceType": "t2.micro",
                "KeyName": "testkey",
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sdm",
                        "Ebs": {
                            "VolumeType": "io1",
                            "Iops": 200,
                            "DeleteOnTermination": false,
                            "VolumeSize": 20
                        }
                    }
                ]
            }
        },
        "MyEIP": {
            "Type": "AWS::EC2::EIP",
            "Properties": {
                "InstanceId": {
                    "Ref": "MyEC2Instance"
                }
            }
        }
    }
}
```

之前的模板以 Amazon EC2 实例为中心；但 CloudFormation 模板还具有其他功能，可利用这些功能来构建复杂的资源集并在许多环境中重新使用这些模板。例如，您可添加输入参数，其值是在创建 CloudFormation 堆栈时指定的。换句话说，您可在创建堆栈而不是创建模板时指定一个值 (如实例类型)，以便在不同的情况下更轻松地重新使用模板。

堆栈

在您使用 CloudFormation 时，可将相关资源作为一个称为堆栈的单元进行管理。您可通过创建、更新和删除堆栈来创建、更新和删除一组资源。堆栈中的所有资源均由堆栈的 CloudFormation 模板定义。假设您创建了一个模板，它包括自动扩缩组、Elastic Load Balancing 负载均衡器和 Amazon Relational Database Service (Amazon RDS) 数据库实例。要创建这些资源，您可通过提交已创建的模板来创建堆栈，CloudFormation 将会为您配置所有这些资源。

更改集

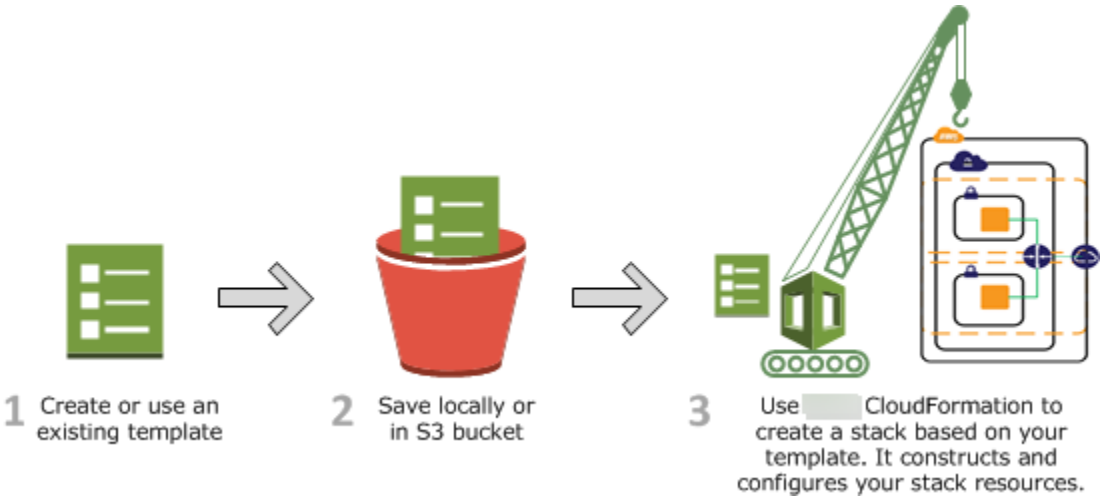
如果您需要更改堆栈中运行的资源，则可更新堆栈。在更改资源之前，您可以生成一个更改集，这是建议进行的更改的摘要。利用更改集，您可以在实施更改之前，了解更改可能会对运行的资源 (特别是关键资源) 造成的影响。

例如，如果您更改 Amazon RDS 数据库实例的名称，则 CloudFormation 将创建新数据库并删除旧数据库。除非您已经对旧数据库中的数据进行备份，否则您将丢失该数据。如果您生成了更改集，则将了解更改会导致数据库被替换，而您可以先做出相应的计划，然后再更新堆栈。

CloudFormation 的工作原理

使用 CloudFormation 创建堆栈时，CloudFormation 会向 Amazon 发出基础服务调用以预置和配置模板中描述的资源。您需要有创建这些资源的权限。例如，要使用 CloudFormation 创建 EC2 实例，您需要具有创建实例的权限。您可以使用 Amazon Identity and Access Management（IAM）来管理这些权限。

CloudFormation 进行的调用全部由您的模板声明。例如，假设您有一个描述带 `t2.micro` 实例类型的 EC2 实例的模板。当您使用该模板创建堆栈时，CloudFormation 将调用 Amazon EC2 创建实例 API 并将该实例类型指定为 `t2.micro`。以下示意图归纳了用于创建堆栈的 CloudFormation 工作流程。



创建堆栈

1. 使用文本编辑器创建 JSON 或 YAML 格式的 CloudFormation 模板。CloudFormation 模板描述了您所需的资源及其设置。使用 模板工作室 可视化显示和验证模板。这有助您确保模板结构正确，没有语法错误。有关更多信息，请参阅 使用 CloudFormation 模板。
2. 您可将模板保存在本地或 Amazon S3 存储桶中。
3. 通过指定模板文件的位置（例如，本地计算机上的路径或 Amazon S3 URL）来创建 CloudFormation 堆栈。如果模板包含参数，则可在创建堆栈时指定输入值。利用参数，您可以将值传入模板，以便能在创建堆栈时自定义资源。

> **Note**
>
> 如果您指定本地存储的模板文件，CloudFormation 会将其上传到您 Amazon 账户中的 S3 桶。CloudFormation 为您上传模板文件的每个区域创建一个桶。具有您 Amazon 账户中 Amazon Simple Storage Service（Amazon S3）权限的任何人均可访问桶。如果 CloudFormation 创建的存储桶已存在，则将模板添加到该存储桶。
> 您可通过手动将模板上传到 Amazon S3 来使用您自己的存储桶并管理其权限。之后，当您创建或更新堆栈时，请指定模板文件的 Amazon S3 URL。

在配置资源之前，CloudFormation 会对您的模板进行部署前验证。这将检查常见错误，例如属性语法无效和资源名称冲突。如果验证失败，操作将在任何资源被创建之前停止。验证会在所有“创建堆栈”、“更新堆栈”和“创建更改集”操作中自动运行。要跳过验证，请使用 `DisableValidation` 参数。有关更多信息，请参阅 验证堆栈部署。

所有资源创建完毕后，CloudFormation 会报告已创建您的堆栈。然后，您可以开始使用堆栈中的资源。如果堆栈创建失败，则 CloudFormation 会通过删除已创建的资源来回滚您的更改。

Deployment time (部署时间)

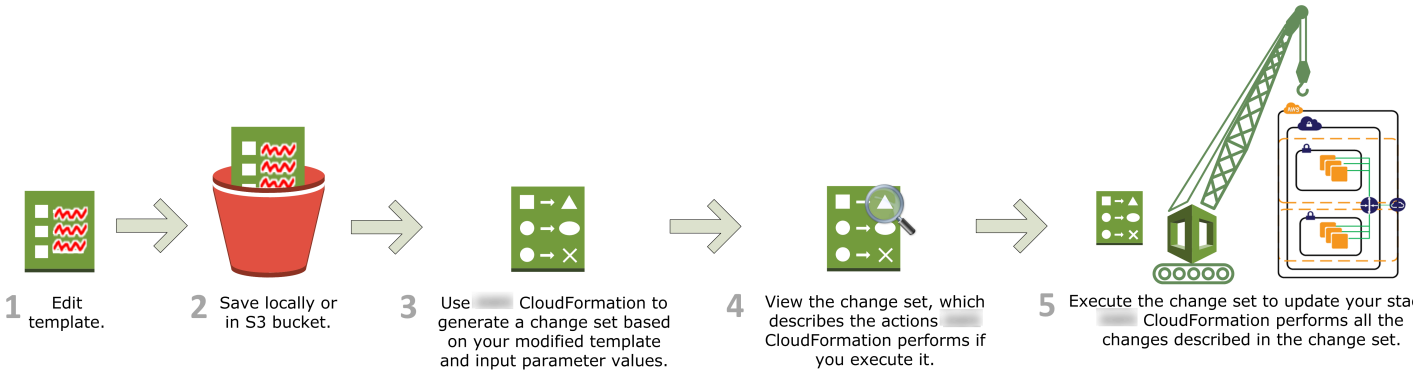
堆栈操作所需的时间取决于堆栈中的资源。CloudFormation 并行预置独立的资源，并在每个资源准备就绪后立即继续后续操作。在包含数据库实例、负载均衡器或 NAT 网关等资源的工作负载中，这些资源需要较长时间才能准备就绪、能够处理流量，并占用了大部分部署时间。有关更多信息，请参阅 了解堆栈部署时间。

Express Mode

要在开发过程中更快地执行堆栈操作，请使用 Express Mode。Express Mode 会在 CloudFormation 应用资源配置后立即完成堆栈操作，而无需等待资源完全稳定。有关更多信息，请参阅 Express Mode。

使用更改集更新堆栈

在需要更新堆栈的资源时，您可以修改堆栈的模板。您不需要创建新堆栈和删除旧堆栈。要更新堆栈，请提交修改的原始堆栈模板版本和/或不同的输入参数值以创建一个更改集。CloudFormation 将修改的模板与原始模板进行比较并生成一个更改集。更改集列出了建议的更改。在审核更改后，您可以启动更改集以更新堆栈，也可以创建新的更改集。以下示意图概述了用于更新堆栈的工作流程。



使用更改集更新堆栈

1. 您可以使用模板工作室或文本编辑器来修改 CloudFormation 堆栈模板。有关更多信息，请参阅 更新堆栈模板。

   更新模板时，应注意更新可能会导致中断。根据您所更新的资源和属性，更新可能会中断或者甚至替换现有资源。有关更多信息，请参阅 理解堆栈资源的更新行为。

2. 您可在本地或在 S3 存储桶中保存 CloudFormation 模板。

3. 通过指定要更新的堆栈和修改后模板的位置 (例如本地计算机上的路径或 Amazon S3 URL) 来创建更改集。有关创建更改集的更多信息，请参阅使用更改集更新 CloudFormation 堆栈。

   > **Note**
   >
   > 如果您指定存储在本地计算机上的模板，则 CloudFormation 自动将模板上传到 Amazon Web Services 账户 中的 S3 桶。

4. 查看更改集以检查 CloudFormation 是否将执行预期更改。例如，检查 CloudFormation 是否将替换任何关键堆栈资源。您可以创建所需数量的更改集，直到您包含所需的更改。

> **Important**
>
> 更改集并不指示您的堆栈更新是否将成功。例如，更改集不会检查是否将超出账户配额、是否在更新不支持更新的资源或者是否权限不足而无法修改资源，这些都可能导致堆栈更新失败。

5. 启动要应用于堆栈的更改集。CloudFormation 仅更新您修改的资源以更新堆栈，并发出已成功更新堆栈的信号。如果堆栈更新失败，则 CloudFormation 将回滚更改以将堆栈还原到上一个已知工作状态。

CloudFormation 入门方式

要使用控制台创建 hello world CloudFormation 堆栈，请参阅创建第一个堆栈。

如需指导学习，请尝试 Amazon CloudFormation 入门讲习会，其提供模板开发的实践经验。

注册 Amazon Web Services 账户

注册 Amazon Web Services 账户

如果您还没有，Amazon Web Services 账户请完成以下步骤来创建一个。

注册 Amazon Web Services 账户

1. 打开 https://signin.aws.amazon.com/signup?request_type=register。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个。Amazon Web Services 账户根用户根用户有权访问该账户中的所有 Amazon Web Services 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

Amazon注册过程完成后， 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅《IAM 用户指南》中的 为 IAM 用户启用虚拟 MFA 设备（控制台）。

要授予其他用户访问您的 Amazon Web Services 账户资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅《IAM 用户指南》中的以下主题：

- 在您的 Amazon Web Services 账户中创建 IAM 用户
- 适用于 Amazon 资源的 Access Management
- 基于 IAM 身份的策略示例

> **Note**
>
> 有关如何管理谁有权访问哪些功能的更多信息，请参阅 使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限。

创建第一个堆栈

本文演示使用 Amazon Web Services 管理控制台 创建第一个 CloudFormation 堆栈的过程。学习本教程后，您将会了解如何配置基础 Amazon 资源、监控堆栈事件和生成输出。

在本示例中，CloudFormation 模板采用 YAML 格式编写。YAML 是一种人类可解读的格式，广泛用于将基础架构定义为代码。您对 CloudFormation 的了解越来越多后，可能还会遇到其他 JSON 格式的模板，但是本教程选择使用 YAML，因为它的可读性好。

> **Note**
>
> CloudFormation 使用免费，但您需要为自己创建的 Amazon EC2 和 Amazon S3 资源付费。如果您不熟悉 Amazon，可以利用免费套餐来最大限度地降低或避免在学习过程产生费用。

主题

- 先决条件
- 使用控制台创建 CloudFormation 堆栈

- 监控堆栈创建
- 测试 Web 服务器
- 问题排查
- 清理
- 后续步骤

先决条件

- 您必须具有对 Amazon Web Services 账户的访问权限，且该账户需要拥有有权使用 Amazon EC2、Amazon S3 和 CloudFormation 的 IAM 用户或角色，或者拥有管理用户访问权限。
- 必须拥有可访问互联网的虚拟私有云（VPC）。本演练模板需要默认 VPC，新版本 Amazon Web Services 账户自动附带该默认 VPC。如果您没有默认 VPC，或者已将其删除，请参阅本主题中的疑难解答章节，了解其他解决方案。

使用控制台创建 CloudFormation 堆栈

使用控制台创建 Hello world CloudFormation 堆栈

1. 打开 CloudFormation 控制台。
2. 选择创建堆栈。
3. 在创建堆栈页面上，选择使用模板工作室构建。模板将在模板工作室中打开。
4. 要创建和验证示例模板，请执行以下操作：

    a. 将以下 CloudFormation 模板复制粘贴到模板编辑器中：

```
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for WebServer with Security Group and EC2
 Instance

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
```

```
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.

  MyIP:
    Description: Your IP address in CIDR format (e.g. 203.0.113.1/32).
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access via my IP address
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP

  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData: !Base64 |
        #!/bin/bash
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html

Outputs:
```

```
  WebsiteURL:
    Value: !Join
      - ''
      - - http://
        - !GetAtt WebServer.PublicDnsName
    Description: Website URL
```

在继续下一步之前，我们花些时间来了解一下模板和一些关键的 CloudFormation 概念。

- **Parameters** 部分声明在创建堆栈时可传递给模板的值。稍后在模板中指定的资源会引用这些值并使用这些数据。如果要指定不想存储在模板内的信息，使用参数就是非常有效的办法。这种方法也可以用于指定对正在部署的特定应用程序或配置可能具有唯一性的信息。
- 模板会定义以下参数：
  - **LatestAmiId** – 从 Amazon Systems Manager Parameter Store 检索最新的 Amazon Linux 2 AMI ID。
  - **InstanceType** – 允许选择 EC2 实例类型（默认：`t2.micro`，允许：`t3.micro`、`t2.micro`）。
  - **MyIP** – 指定 HTTP 访问权限的 IP 地址范围（默认：0.0.0.0/0，允许来自任意 IP 的访问）。
- **Resources** 部分包含要使用该模板创建的 Amazon 资源的定义。资源声明是立即指定这些所有配置设置的一种有效方法。如果您将资源声明放入模板，您将可以使用该模板创建和配置所有的声明资源，从而创建堆栈。您也可以使用相同模板创建新的堆栈，来启动相同的资源配置。
- 此模板创建以下资源：
  - **WebServerSecurityGroup** – EC2 安全组，允许来自指定 IP 范围的端口 80 上的入站 HTTP 流量。
  - **WebServer** – 具有以下配置的 EC2 实例：
    - 使用最新的 Amazon Linux 2 AMI
    - 应用选定的实例类型
    - 将 `WebServerSecurityGroup` 添加到 `SecurityGroupIds`属性
    - 包括用于安装 Apache HTTP 服务器的用户数据脚本
- 在每个资源和参数声明的开头都指定逻辑名称。例如，`WebServerSecurityGroup` 是分配给 EC2 安全组资源的逻辑名称。然后，系统会使用 `Ref` 函数来按资源和参数在模板其他部分中的逻辑名称来引用资源和参数。如果一个资源引用另一个资源，两者之间会形成依赖关系。

- **Outputs** 部分定义堆栈创建后返回的自定义值。您可以使用输出值在堆栈中返回来自资源的信息，例如资源标识符或 URL。
- 该模板定义了一个输出：
  - **WebsiteURL** – 已部署 Web 服务器的 URL（使用 EC2 实例的公有 DNS 名称构建）。Join 函数可以将固定 http:// 和可变 PublicDnsName 组合成一个字符串，从而轻松输出 Web 服务器的完整 URL。

b. 选择验证以确保 YAML 代码有效，然后再上传模板。

c. 选择保存并继续将模板保存到 S3 存储桶。

d. 在开启的对话框中，记下 S3 存储桶的名称，之后便可删除。然后，选择确认并继续。这会将您返回至 CloudFormation 控制台，其中已指定了您模板的 S3 路径。

5. 在创建堆栈主页上，选择下一步。
6. 在指定堆栈详细信息页面的堆栈名称字段中，输入名称。堆栈名中不得含有空格。对于本示例，请使用 **MyTestStack**。
7. 在参数下，指定参数值，如下所示：

- LatestAmiId：此值默认设定为最新的 Amazon Linux 2 AMI。
- InstanceType：选择 t2.micro 或 t3.micro 作为 EC2 实例类型。

> **Note**
>
> 如果您是 Amazon 新用户，您可以使用免费套餐在 12 个月内免费启动和使用 t2.micro 实例（在 t2.micro 不可用的区域，您可以通过免费套餐使用 t3.micro 实例）。

- MyIP：使用 /32 后缀指定实际公有 IP 地址。在 CIDR 表示法中使用 /32 后缀来指定允许使用单个 IP 地址。这基本上意味着仅允许流入和流出此特定 IP 地址的流量，而不允许其他 IP 地址的流量。

8. 选择下一步两次，前往检查和创建页面。在本教程中，您可以保留配置堆栈选项页面的默认值。
9. 审查堆栈信息。如果对设置满意，请选择 Submit (提交)。

监控堆栈创建

选择提交后，CloudFormation 将开始创建模板中指定的资源。您的新堆栈 (**MyTestStack**) 将显示在 CloudFormation 控制台顶部的列表中。状态应为 CREATE_IN_PROGRESS。您可以通过查看事件了解堆栈的详细状态。

想要查看堆栈的事件

1. 在 CloudFormation 控制台上，选择列表中的堆栈 **MyTestStack**。
2. 在堆栈详细信息窗格中，选择 Events (事件) 选项卡。

   控制台每 60 秒自动使用最新事件刷新事件列表。

Events (事件) 选项卡显示堆栈创建过程中的每个重要步骤（按每个事件的时间排序，最新事件位于最上面）。

第一个事件 (在事件列表最底部) 为堆栈创建过程的开始：

```
2024-12-23 18:54 UTC-7 MyTestStack CREATE_IN_PROGRESS User initiated
```

下面是标志所有资源创建开始和完成的事件。例如，创建 EC2 实例后会产生以下条目：

```
2024-12-23 18:59 UTC-7 WebServer CREATE_COMPLETE

2024-12-23 18:54 UTC-7 WebServer CREATE_IN_PROGRESS Resource creation
initiated
```

当 CloudFormation 报告它已开始创建资源时，将记录 CREATE_IN_PROGRESS 事件。在资源创建成功时，将记录 CREATE_COMPLETE 事件。

当 CloudFormation 成功创建了堆栈时，您将会在 Event (事件) 选项卡顶部看到下列事件：

```
2024-12-23 19:17 UTC-7 MyTestStack CREATE_COMPLETE
```

如果 CloudFormation 无法创建资源，它将报告 CREATE_FAILED 事件，并（在默认情况下）回滚堆栈、删除已创建的所有资源。状态原因列显示已导致失败的问题。

创建堆栈后，您可以前往资源选项卡查看创建的 EC2 实例和安全组。

测试 Web 服务器

成功创建堆栈后，在 CloudFormation 控制台中前往输出选项卡。找到 WebsiteURL 字段。字段中会包含 EC2 实例的公有 URL。

打开浏览器并前往 WebsiteURL 下列出的 URL。您会在浏览器中看到一条简单的 "Hello World!" 消息。

这表示您的 EC2 实例正在运行 Apache HTTP 服务器并提供基本网页。

问题排查

如果您在创建堆栈期间遇到回滚，这可能是缺少 VPC 所致。解决办法如下。

没有可用的默认 VPC

本演练中的模板需要默认 VPC。如果因为 VPC 或子网可用性错误而出现堆栈创建失败的情况，这可能是因为账户中没有默认 VPC。您有以下选项：

- 创建新的默认 VPC：可以通过 Amazon VPC 控制台创建新的默认 VPC。有关说明，请参阅《Amazon VPC 用户指南》中的创建默认 VPC。
- 修改模板以指定子网：如果您有非默认 VPC，则可以修改模板以明确指定 VPC 和子网 ID。将以下参数添加到模板：

```
  SubnetId:
    Description: The subnet ID to launch the instance into
    Type: AWS::EC2::Subnet::Id
```

然后，更新 `WebServer` 资源以包含子网 ID：

```
  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      SubnetId: !Ref SubnetId
      UserData: !Base64 |
        #!/bin/bash
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html
```

创建堆栈时，您需要指定一个可以访问互联网的子网才能确保 Web 服务器可供访问。

清理

为了保证您不为不需要的服务承担费用，您可以通过删除堆栈和它的资源进行清理。您也可以删除存储堆栈模板的 S3 存储桶。

想要删除堆栈和它的资源

1. 打开 CloudFormation 控制台。
2. 在堆栈页面上，选择您创建的堆栈名称旁的选项（**MyTestStack**），然后选择删除。
3. 当系统提示进行确认时，选择 Delete（删除）。
4. 在事件选项卡上监控堆栈删除过程的进度。**MyTestStack** 的状态更改为 DELETE_IN_PROGRESS。当 CloudFormation 完成删除堆栈后，它会将从列表中移堆栈除。

如果您已完成此示例模板的使用，不再需要 Amazon S3 存储桶，请将其删除。您必须先清空存储桶，然后才能删除存储桶。清空存储桶将删除存储桶中的所有对象。

清空和删除 Amazon S3 存储桶

1. 打开 Amazon S3 控制台。
2. 在控制台左侧的导航窗格中，选择 Buckets（存储桶）。
3. 在存储桶列表中，选择要为本教程创建的存储桶名称旁的选项，然后选择清空。
4. 在清空存储桶页面上，通过在文本字段中键入 **permanently delete** 来确认要清空存储桶，然后选择清空。
5. 在清空存储桶：状态页面上监控存储桶清空过程的进度。
6. 要返回到存储桶列表，请选择退出。
7. 选择存储桶名称旁的选项，然后选择删除。
8. 在系统提示进行确认时，键入存储桶的名称，然后选择删除存储桶。
9. 在存储桶清单上监控存储桶删除过程的进度。Amazon S3 完成存储桶删除后，会将存储桶从列表中移除。

后续步骤

恭喜您！您已成功创建了堆栈、监控了堆栈的创建并使用了堆栈的输出。

继续学习：

- 详细了解模板以便自己创建模板。有关更多信息，请参阅 使用 CloudFormation 模板。
- 试试参加 Amazon CloudFormation 入门研习会，获得更多模板创建动手实践。
- 有关 Amazon CloudFormation 入门的简短版本，请参阅在 Amazon EC2 上部署应用程序。本主题同样描述了使用 CloudFormation 帮助程序脚本 `cfn-init` 的情境，但是用于引导 Amazon EC2 实例。

CloudFormation 最佳实践

最佳实践是可以帮助您更有效地使用 CloudFormation 并在整个工作流中采用安全实践的建议。了解如何执行以下操作：计划和组织您的堆栈，创建描述您的资源以及在这些资源上运行的软件应用程序的模板，以及管理您的堆栈及其资源。以下最佳实践基于来自当前 CloudFormation 客户的实际经验。

计划和组织

- 缩短反馈循环以提高开发速度
- 按生命周期和所有权组织您的堆栈
- 使用跨堆栈引用在堆栈之间共享值
- 使用 Amazon CloudFormation StackSets 进行多账户和多区域部署
- 重复使用模板以在多个环境中复制堆栈
- 验证所有资源类型的配额
- 使用模块重复使用资源配置
- 采用基础设施即代码实践

创建模板

- 不要在模板中嵌入凭证
- 使用 Amazon 特定的参数类型
- 使用参数约束
- 使用伪参数来提高可移植性
- 使用 `AWS::CloudFormation::Init` 在 Amazon EC2 实例上部署软件应用程序
- 使用最新的帮助程序脚本
- 在使用模板前先验证模板
- 使用 YAML 或 JSON 进行模板编写
- 实施全面的标记策略
- 利用模板宏进行高级转换

管理堆栈

- 通过 CloudFormation 管理所有堆栈资源
- 更新堆栈之前创建更改集
- 使用堆栈策略保护资源
- 使用 Amazon CloudTrail 记录 CloudFormation 调用

- 使用代码审查和修订控制来管理您的模板
- 定期更新您的 Amazon EC2 实例
- 定期使用偏移检测
- 配置回滚触发器以实现自动恢复
- 实施有效的堆栈重构策略
- 使用 CloudFormation Hooks 进行生命周期管理

编写工具

- 使用 IaC 生成器从现有资源创建模板
- 使用模板工作室直观地编辑模板
- 考虑对复杂的基础设施使用 Amazon Cloud Development Kit (Amazon CDK)

安全性与合规性

- 使用 IAM 控制访问
- 采用最低权限原则
- 保护敏感参数
- 通过 Guard 实施策略即代码

缩短反馈循环以提高开发速度

采用实践和工具，以帮助您缩短使用 CloudFormation 模板描述的基础架构的反馈循环。这包括在工作站中对模板执行早期检查和测试；执行此操作时，甚至在向源代码存储库提交贡献之前，您就有机会发现潜在的语法和配置问题。及早发现此类问题有助于防止这些问题进入正式的生命周期环境，例如开发、质量保证和生产。这种早期测试、快速失效机制方法的好处是减少返工等待时间、减少潜在影响区域，并提高您对成功进行配置操作的信心。

帮助您进行快速失效机制实践的工具选择包括 Amazon CloudFormation Linter（`cfn-lint`）和 TaskCat 命令行工具。`cfn-lint` 工具让您能够根据 Amazon CloudFormation 资源规范验证 CloudFormation 模板。这包括检查资源属性的有效值以及最佳实践。`cfn-lint` 的插件可用于许多代码编辑器；这让您能够在编辑器中可视化问题并获得直接的 linter 反馈。您也可以选择将 `cfn-lint` 集成到源代码存储库的配置中，以便您可以在提交贡献时执行模板验证。有关更多信息，请参阅 Git pre-commit validation of Amazon CloudFormation templates with `cfn-lint`。执行初始检查并修复了 `cfn-lint` 可能出现的任何问题后，您就可以使用 TaskCat 在选择的 Amazon Web Services 区域中以编程方式创建堆栈来测试模板。TaskCat 还会生成报告，其中包含您选择的每个区域的通过/未通过等级。

有关如何使用这两种工具缩短反馈循环的分步实作演练，请参阅 Amazon CloudFormation Workshop 中的 Linting and Testing lab。

Amazon CloudFormation 部署前验证会在任何资源被配置之前自动检测出常见错误。这些错误包括属性语法无效和资源名称冲突。默认情况下，部署前验证会在所有“创建堆栈”、“更新堆栈”和“创建更改集”操作中运行。无需进行任何配置。对于 CDK 项目，`cdk validate` 会对您合成的模板运行 CloudFormation 部署前验证。

按生命周期和所有权组织您的堆栈

使用 Amazon 资源的生命周期和所有权帮助您决定每个堆栈应包含的资源。最初，您可将所有资源置于一个堆栈中，但当您堆栈的规模增大和范围扩大时，管理一个堆栈将是一项麻烦且耗时的工作。通过使用常见的生命周期和所有权对资源进行分组，所有者可使用自己的流程和计划来更改其资源集而不会影响其他资源。

例如，假设开发人员和工程师团队拥有一个托管于负载均衡器背后的 Amazon EC2 Auto Scaling 实例的网站。由于该网站具有自己的生命周期并由网站团队维护，因此您可以为网站及其资源创建一个堆栈。现在假设该网站还使用后端数据库，其中数据库位于由数据库管理员所有和维护的单独堆栈中。当网站团队或数据库团队需要更新其资源时，他们可以这样做而不会影响彼此的堆栈。如果所有资源位于一个堆栈中，则协调和传达更新会很难。

有关组织堆栈的其他指导，您可以使用两个常见框架：多层架构和服务导向型架构 (SOA)。

分层架构将堆栈组织到相互堆放的多个水平层上，其中每个层都依赖于其正下方的层。每个层可包含一个或多个堆栈，但在每个层中，您的堆栈应包含具有类似生命周期和所有权的 Amazon 资源。

利用服务导向型架构，可以将大型业务问题组织到可管理的部分中。每个部分均为一个服务，该服务具有清楚定义的用途并代表一个独立的功能单元。您可以将这些服务映射到一个堆栈，其中每个堆栈均拥有自己的生命周期和所有者。可将这些服务（堆栈）关联在一起，使其能够互相交互。

使用跨堆栈引用在堆栈之间共享值

在根据生命周期和所有权安排您的 Amazon 资源时，建议您构建堆栈来使用其他堆栈中的资源。您可以对值进行硬编码，或者使用输入参数传递资源名称和 ID。但是，这些方法使得模板难以重复使用，或者会增加堆栈的运行开销。相反，使用跨堆栈引用在堆栈之间共享值。CloudFormation 提供了两种方法：

- `Fn::ImportValue` - 导入另一个堆栈已显式导出的值。此方法会在同一账户和区域内创建强引用。当其他堆栈依赖于某个导出值时，CloudFormation 会阻止您删除导出该值的堆栈。

- `Fn::GetStackOutput` - 直接引用任意堆栈的输出，包括来自其他 Amazon 账户或区域中堆栈的输出，无需显式导出。此方法会创建弱引用，该引用在创建或更新堆栈时解析。

有关更多信息，请参阅 获取从已部署的 CloudFormation 堆栈导出的输出 和 Fn::GetStackOutput。

使用 Amazon CloudFormation StackSets 进行多账户和多区域部署

Amazon CloudFormation StackSets 通过让您在单个操作中跨多个账户和区域创建、更新或删除堆栈来扩展堆栈的功能。使用 StackSets 在整个组织内部署通用基础设施组件、合规性控制或共享服务。

使用 StackSets 时，请通过 Amazon Organizations 实现服务托管权限，以简化权限管理。通过这种方法，您可以将 StackSets 部署到组织内的账户，而无需在每个账户中手动配置 IAM 角色。

有关 StackSets 的更多信息，请参阅 StackSets 概念。

验证所有资源类型的配额

在启动堆栈前，请确保您能创建所需的所有资源而不会达到 Amazon 账户限制。如果您达到限制，则 CloudFormation 将无法成功创建堆栈，直到您增加配额或删除超额资源。每项服务均可具有不同的限制，您在启动堆栈前应了解这些限制。例如，默认情况下，您只能在 Amazon Web Services 账户中为每个区域启动 2000 个 CloudFormation 堆栈。有关限制以及如何增加默认限制的更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon service quotas。

重复使用模板以在多个环境中复制堆栈

在设置您的堆栈和资源后，您可以重复使用您的模板以便在多个环境中复制您的基础结构。例如，您可以创建用于开发、测试和生产的环境，以便在应用更改前先对其进行测试。要使模板可重用，可使用参数、映射和条件部分，以便能在创建堆栈时对其进行自定义。例如，对于您的开发环境，您可以指定相对于生产环境来说成本更低的实例类型，但所有其他配置和设置保持不变。有关参数、映射和条件的更多信息，请参阅CloudFormation 模板部分。

使用模块重复使用资源配置

随着您的基础设施的发展，常见模式可合并以便声明每个模板中的相同组件。模块是一种供您以透明、易管理和可重复的方式打包资源配置以便跨堆栈模板实现包含的方法。模块可以将常见服务配置和最佳实践封装为模块化、可自定义的构建基块，供您包含在堆栈模板中。

这些构建块可以用于单个资源，例如定义 Amazon Elastic Compute Cloud (Amazon EC2) 实例的最佳实践，也可以用于多个资源，以定义应用程序架构的通用模式。这些构建块可以嵌套到其他模块中，因此您可以将最佳实践堆叠到更高级别的构建块中。CloudFormation 模块在 CloudFormation 注册表中可用，因此您可以像使用本机资源一样使用它们。使用 CloudFormation 模块时，模块模板会扩展到使用模板中，这样您就可以使用 Ref 或 Fn::GetAtt 来访问模块内的资源。有关更多信息，请参阅 使用 CloudFormation 模块创建可跨模板包含的可重复使用的资源配置。

采用基础设施即代码实践

通过实施基础设施即代码 (IaC) 实践，将 CloudFormation 模板视为代码。将模板存储在版本控制系统中，实施代码审查，并使用自动化测试来验证更改。此方法可确保一致性、改善协作，并为基础设施更改提供审计跟踪记录。

考虑为基础设施代码实施 CI/CD 管道，以自动测试和部署 CloudFormation 模板。Amazon CodePipeline、Amazon CodeBuild 和 Amazon CodeDeploy 等工具可用于为基础设施部署创建自动化工作流。

有关实施 IaC 最佳实践的更多信息，请参阅 Using Amazon CloudFormation as an IaC tool。

有关使用 CloudFormation 进行持续交付的更多信息，请参阅利用 CodePipeline 进行持续交付。

不要在模板中嵌入凭证

我们建议您在堆栈模板中使用动态引用，而不是在 CloudFormation 模板中嵌入敏感信息。

动态引用提供了一种简洁、功能强大的方法，用于引用在其他服务（例如 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager）中存储和管理的外部值。当您使用动态引用时，CloudFormation 会在堆栈和更改集合操作期间根据需要检索指定引用的值，并将值传递到相应的资源。但是，CloudFormation 从不存储实际引用值。有关更多信息，请参阅使用动态引用以指定模板值。

Amazon Secrets Manager 帮助您安全地加密、存储和检索数据库和其他服务的凭证。Amazon Systems Manager Parameter Store 提供安全的分层存储，用于配置数据管理。

有关定义模板参数的信息，请参阅CloudFormation 模板 Parameters 语法。

使用 Amazon 特定的参数类型

如果您的模板需要输入现有的 Amazon 特定的值（例如，现有的 Amazon Virtual Private Cloud ID 或 Amazon EC2 密钥对名称），请使用 Amazon 特定的参数类型。例如，您可以将参数指定为类型

AWS::EC2::KeyPair::KeyName，这将使用位于 Amazon Web Services 账户和从中创建堆栈的区域中的现有密钥对名称。CloudFormation 可在创建堆栈之前快速验证 Amazon 特定参数类型的值。此外，如果您使用 CloudFormation 控制台，则 CloudFormation 会显示有效值的下拉列表，因此您无需查找或记住正确的 VPC ID 或密钥对名称。有关更多信息，请参阅 使用 CloudFormation 提供的参数类型在运行时指定现有资源。

使用参数约束

利用约束，您可以描述允许的输入值，以便 CloudFormation 在创建堆栈之前捕获任何无效值。可以设置约束，例如最小长度、最大长度和允许的模式。例如，您可以对数据库用户名值设置约束，使其最小长度必须为 8 个字符且仅包含字母数字字符。有关更多信息，请参阅 CloudFormation 模板 Parameters 语法。

使用伪参数来提高可移植性

您可以在模板中使用伪参数作为内部函数的参数，例如 Ref 和 Fn::Sub。伪参数是 CloudFormation 预定义的参数。请您不要在您的模板中声明它们。在内置函数中使用伪参数可提高堆栈模板在区域和账户之间的可移植性。

例如，假设您想要创建一个模板，在该模板中，您需要为给定的资源属性指定另一个现有资源的 Amazon 资源名称（ARN）。在这种情况下，现有资源是具有以下 ARN 的 Amazon Systems Manager Parameter Store 资源：arn:aws:ssm:us-east-1:123456789012:parameter/MySampleParameter。您将需要根据您的目标 Amazon 分区、区域和账户 ID 调整 ARN 格式。您可以使用 AWS::Partition、AWS::Region 和 AWS::AccountId 伪参数来提高模板的可移植性，而不用对这些值进行硬编码。在这种情况下，以下示例向您展示如何将 ARN 中的元素与 CloudFormation 连接：!Sub 'arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter/MySampleParameter。

另举一例，假设您想要跨多个堆栈共享资源或配置。在此示例中，假设您已为您的 VPC 创建一个子网，然后导出其 ID 以便与相同 Amazon Web Services 账户和区域中的其他堆栈一起使用。在另一个堆栈中，您在描述 Amazon EC2 实例时引用子网 ID 的导出值。有关使用 Export 输出字段和 Fn::ImportValue 内置函数的详细示例，请参阅 引用其他 CloudFormation 堆栈中的资源输出。

对于每个账户和区域，堆栈导出必须是唯一的。因此，在这种情况下，您可以使用 AWS::StackName 伪参数为您的导出创建前缀。由于每个账户和区域的堆栈名称也必须是唯一的，因此使用此伪参数作为前缀可以提高具有唯一导出名称的可能性，同时还推广了在导出值的堆栈之间采用可重复使用的方法。或者，您可以使用您自己选择的前缀。

使用 AWS::CloudFormation::Init 在 Amazon EC2 实例上部署软件应用程序

在启动堆栈时，您可以在 Amazon EC2 实例上安装和配置软件应用程序，方式是使用 cfn-init 帮助程序脚本和 AWS::CloudFormation::Init 资源。通过使用 AWS::CloudFormation::Init，您可以描述所需的配置而不是为程序步骤编写脚本。您还可以更新配置，而无需重新创建实例。如果您的配置出现问题，CloudFormation 会生成可用于调查问题的日志。

在您的模板中，在 AWS::CloudFormation::Init 资源中指定安装和配置状态。有关说明如何使用 cfn-init 和 AWS::CloudFormation::Init 的演练，请参阅在 Amazon EC2 上部署应用程序。

使用最新的帮助程序脚本

CloudFormation 帮助程序脚本将定期进行更新。在调用帮助程序脚本之前，请务必在您的模板的 UserData 属性中包括以下命令，以确保您的启动实例获得最新的帮助程序脚本：

```
yum install -y aws-cfn-bootstrap
```

有关获取最新帮助程序脚本的更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 CloudFormation 帮助程序脚本参考。

在使用模板前先验证模板

在使用模板创建或更新堆栈之前，可以先使用 CloudFormation 验证模板。在 CloudFormation 创建任何资源之前，验证模板可帮助您捕获语法错误和一些语义错误，例如循环依赖性。如果您使用 CloudFormation 控制台，该控制台会在您指定输入参数后自动验证模板。对于 Amazon CLI 或 CloudFormation API，请使用 validate-template CLI 命令或 ValidateTemplate API 操作。

在验证期间，CloudFormation 首先检查模板是否是有效的 JSON。如果不是，CloudFormation 会检查模板是否是有效的 YAML。如果两种检查都失败，CloudFormation 会返回模板验证错误。

为了在部署前发现更多问题，您还可以检查计算机上的模板。有关安装和使用 cfn-lint、cloudformation-validate 和 CloudFormation 语言服务器的更多信息，请参阅验证模板。要了解本地验证的限制，请参阅了解验证范围。

验证模板的组织策略合规性

您还可以验证模板是否符合组织策略准则。Amazon CloudFormation Guard（`cfn-guard`）是一种命令行界面（CLI）工具，它提供了策略即代码语言来定义规则，可以检查必需和禁止的资源配置。然后，您可以根据这些规则验证模板。例如，管理员可以创建规则以确保用户始终创建加密的 Amazon S3 存储桶。

您可以在编辑模板时本地使用 `cfn-guard`，也可以自动将其用作 CI/CD 管道的一部分，以停止非合规资源的部署。

此外，`cfn-guard` 还包括功能 `rulegen`，使您能够从现有兼容的 CloudFormation 模板中提取规则。

有关安装和使用示例，请参阅使用 Guard 验证模板。

使用 YAML 或 JSON 进行模板编写

CloudFormation 支持 YAML 和 JSON 格式的模板。每种都有其优点，具体选择取决于特定需求：

在以下情况下使用 YAML

- 优先考虑可读性和可维护性
- 希望包括注释来记录模板
- 正在处理带有嵌套结构的复杂模板
- 希望使用 YAML 特有的功能（如锚点和别名）来减少重复

在以下情况下使用 JSON：

- 需要与支持 JSON 格式的工具或系统集成
- 正在使用编程模板生成或操作
- 需要严格的数据验证

由于 YAML 的可读性和注释支持，通常建议用于手动模板编写。它对于复杂的模板特别有用，其中基于缩进的结构有助于可视化资源层次结构。JSON 在自动化工作流中或使用需要 JSON 输入的 API 时具有优势。当需要确保严格遵守特定结构时，它也是有益的。无论您选择哪种格式，都应专注于创建结构良好、有据可查且易于维护的模板。如果使用 YAML，请利用其锚点和别名等功能来减少重复并提高可维护性。

实施全面的标记策略

为 CloudFormation 模板创建的所有资源实施一致的标记策略。标签有助于资源组织、成本分配、访问控制以及自动化。考虑包括环境、所有者、成本中心、应用程序和目的的标签。

使用 `AWS::CloudFormation::Stack` 资源的 `Tags` 属性将标签应用于堆栈中所有支持的资源。您还可以使用许多资源类型可用的 `TagSpecifications` 属性在资源创建期间应用标签。

有关标记的更多信息，请参阅资源标记。

利用模板宏进行高级转换

CloudFormation 宏使您能够对模板执行自定义处理，从查找和替换操作等简单操作到生成额外资源的复杂转换。使用宏来扩展 CloudFormation 模板的功能并在整个组织中实施可重复使用的模式。

Amazon Serverless Application Model是一个简化无服务器应用程序开发的宏示例。请考虑为组织特定的模式和要求创建自定义宏。

有关在模板中使用宏的更多信息，请参阅 CloudFormation 宏概述。

通过 CloudFormation 管理所有堆栈资源

在启动堆栈后，使用 CloudFormation 控制台、API 或 Amazon CLI 更新您堆栈中的资源。请勿在 CloudFormation 外部更改堆栈资源。这样做会使您的堆栈模板和堆栈资源的当前状态不匹配，从而在您更新或删除堆栈时导致出现错误。这就是所谓的偏差。如果对 CloudFormation 模板之外的资源进行了更改，并且您更新了堆栈，则直接对该资源所做的更改将被丢弃，并且资源配置将恢复为模板中的配置。

有关偏移的更多信息，请参阅什么是偏差？。

有关更新堆栈的更多信息，请参阅更新 CloudFormation 堆栈。

更新堆栈之前创建更改集

更改集允许您在实施之前查看对堆栈提议的更改可能会如何影响正在运行的资源。在您运行更改集之前，CloudFormation 不会对您的堆栈进行任何更改，从而允许您决定是继续进行提议的更改还是创建另一个更改集。

使用更改集来检查更改可能对您正在运行的资源造成的影响，特别是关键资源。例如，如果您更改 Amazon RDS 数据库实例的名称，CloudFormation 将创建新的数据库并删除旧数据库；除非您已进行

备份，否则，将会丢失旧数据库中的数据。如果您生成更改集，就可以看到更改将替换您的数据库。这可以帮助您在更新堆栈之前进行规划。有关更多信息，请参阅 使用更改集更新 CloudFormation 堆栈。

> **Note**
>
> 如果您使用直接堆栈更新而非更改集，部署前验证仍会自动对您的模板进行验证。不过，更改集具有其他优势。在执行更新之前，您可以使用更改集来预览哪些资源将被添加、修改或替换。

使用堆栈策略保护资源

堆栈策略可帮助避免关键堆栈资源进行可能导致资源中断甚至被替换的非有意更新。堆栈策略是一个 JSON 文档，该文档描述可对指定资源执行哪些更新操作。只要创建具有重要资源的堆栈，就要指定堆栈策略。

在堆栈更新过程中，您必须显式指定要更新的受保护的资源；否则，不会更改受保护的资源。有关更多信息，请参阅 防止更新堆栈资源。

使用 Amazon CloudTrail 记录 CloudFormation 调用

Amazon CloudTrail 跟踪在您的 中进行 CloudFormation API 调用的任何人。Amazon Web Services 账户当任何人使用 CloudFormation API、CloudFormation 控制台、后端控制台或 CloudFormation Amazon CLI 命令时，系统将会记录 API 调用。启用日志记录并指定用于存储日志的 Amazon S3 存储桶。这样一来，在需要时，您可以审核在您的账户中发起 CloudFormation 调用的人员。

有关更多信息，请参阅 使用 Amazon CloudTrail 记录 Amazon CloudFormation API 调用。

使用代码审查和修订控制来管理您的模板

您的堆栈模板描述您的 Amazon 资源的配置，例如它们的属性值。要查看更改并保留资源的准确历史记录，请使用代码审查和修订控制。这些方法可帮助您跟踪不同版本的模板之间的更改，以帮助您跟踪对您的堆栈资源所做的更改。此外，通过保留历史记录，您始终能将堆栈恢复为特定版本的模板。

定期更新您的 Amazon EC2 实例

在使用 CloudFormation 创建的所有 Amazon EC2 Windows 实例和 Amazon EC2 Linux 实例上，可以定期运行 `yum update` 命令以更新 RPM 程序包。这可确保您获得最新的修复和安全更新。

定期使用偏移检测

定期使用 CloudFormation 偏移检测功能来识别在 CloudFormation 管理范围之外已修改的资源。检测和解决偏移有助于维护基础设施即代码方法的完整性，并确保模板准确反映已部署资源的状态。

考虑将自动偏移检测作为运营流程的一部分实施。您可以使用由 Amazon EventBridge 规则触发的 Amazon Lambda 函数定期检查偏移，并在检测到差异时通知您的团队。

有关偏移的更多信息，请参阅使用偏移检测功能检测堆栈和资源的非托管配置更改。

配置回滚触发器以实现自动恢复

使用回滚触发器指定 CloudFormation 在堆栈创建和更新操作期间应监控的 Amazon CloudWatch 警报。如果任何指定的警报进入 ALARM 状态，CloudFormation 将自动回滚整个堆栈操作，从而帮助确保基础设施保持稳定状态。

为关键指标（例如应用程序错误率、系统资源利用率或指示应用程序和基础设施运行状况的自定义业务指标）配置回滚触发器。

有关回滚触发器的更多信息，请参阅在突破警报阈值时回滚堆栈。

实施有效的堆栈重构策略

随着基础设施的发展，您可能需要重构 CloudFormation 堆栈来提高可维护性、降低复杂性或适应不断变化的需求。堆栈重构涉及重构模板和资源，同时保留其外部行为和功能。堆栈重构通过以下方式与 CloudFormation 一起使用是有益的：

- 拆分整体堆栈：将大型、复杂的堆栈分解为按生命周期或所有权组织的更小、更易于管理的堆栈
- 整合相关资源：将来自多个堆栈的相关资源组合成一个单一的、有凝聚力的堆栈，以简化管理
- 提取可重用组件：将常见模式移入模块或嵌套堆栈，以促进重用和一致性
- 改进资源组织：重构堆栈内的资源，以更好地反映它们之间的关系和依赖项

有关重构 CloudFormation 堆栈的更多信息，请参阅堆栈重构。

使用 CloudFormation Hooks 进行生命周期管理

CloudFormation Hooks 提供的代码可在预置之前主动检查 Amazon 资源配置，并执行复杂的验证检查。Hooks 会检查资源、堆栈和更改集是否符合组织的安全、运营和成本优化需求。它们会在资源

预置之前发出警告，或者根据配置方式导致操作失败并完全停止。违规行为和警告会记录在 Amazon CloudWatch 中，以便您了解不合规的部署。

有关 Hooks 的这些最佳实践的更多信息，请参阅 Amazon CloudFormation Hooks concepts。

有关 Hooks 可以为 CloudFormation 资源做什么的更多信息，请参阅 What are Amazon CloudFormation Hooks?

使用 IaC 生成器从现有资源创建模板

CloudFormation IaC（基础设施即代码）生成器可帮助您利用现有 Amazon 资源创建 CloudFormation 模板。当您需要复制现有基础设施、记录手动创建的资源或将以前未管理的资源纳入 CloudFormation 管理时，此功能特别有用。IaC 生成器可通过以下方式创建 CloudFormation 模板：

- 加速模板创建：从现有资源生成模板，而不是从头开始编写
- 一致的基础设施：使用生成的模板作为起点，确保新环境与现有环境匹配
- 迁移到基础设施即代码：逐步将手动创建的资源纳入 CloudFormation 管理
- 文档：以模板形式创建现有基础设施的记录

有关 IaC 生成器的更多信息，请参阅使用 IaC 生成器为现有资源生成模板。

使用模板工作室直观地编辑模板

模板工作室让您能够在编辑 CloudFormation 模板的同时查看其资源及其关系的图表。请考虑使用模板工作室完成下面的任务：

- 模板创作：编辑 YAML 或 JSON 并查看图表更新以匹配模板。
- 模板现代化：可视化现有模板，了解其结构并发现改进机会。
- 培训和入职：帮助团队成员理解 CloudFormation 概念和 Amazon 服务关系。
- 审查：使用资源示意图来支持架构、安全性和合规性审查。

有关更多信息，请参阅 模板工作室。

考虑对复杂的基础设施使用 Amazon Cloud Development Kit (Amazon CDK)

对于复杂的基础设施需求，请考虑使用 CDK，通过熟悉的编程语言（如 TypeScript、Python、Java 和 .NET）来定义云资源。Amazon CDK 会根据您的代码生成 CloudFormation 模板，让您能够充分利用 CloudFormation 的全部功能，同时使用首选语言的抽象和编程结构。

Amazon CDK 提供高级构造，封装最佳实践并简化常见基础设施模式的定义。这可以显著减少定义基础设施所需的代码量，同时确保遵循最佳实践。

有关 CDK 的更多信息，请参阅 Amazon Cloud Development Kit (Amazon CDK)。

使用 IAM 控制访问

IAM 是一项可用于管理 Amazon 中的用户及其权限的 Amazon 服务。您可以将 IAM 与 CloudFormation 结合使用来指定用户可以执行哪些 CloudFormation 操作，例如查看堆栈模板、创建堆栈或删除堆栈。此外，管理 CloudFormation 堆栈的任何人都需要具备对这些堆栈中的资源执行操作的权限。例如，如果用户想使用 CloudFormation 启动、更新或终止 Amazon EC2 实例，则他们必须拥有调用相关 Amazon EC2 操作的权限。

在大多数情况下，用户需要完全访问权限来管理模板中的所有资源。CloudFormation 可代表用户发出调用，来创建、修改和删除这些资源。要分隔用户和 CloudFormation 服务之间的权限，请使用服务角色。CloudFormation 使用服务角色的策略发出调用，而不是通过用户的策略。有关更多信息，请参阅 Amazon CloudFormation 服务角色。

采用最低权限原则

在为 CloudFormation 服务角色或模板创建的资源配置 IAM 角色时，请始终遵循最低权限原则。仅授予预期功能所需的权限，并尽可能避免使用通配符权限。

使用 IAM Access Analyzer 查看向 CloudFormation 服务角色授予的权限，并确定可移除的未使用权限。定期审查和更新 IAM 策略，以确保它们符合您的安全要求。

保护敏感参数

对于密码、API 密钥和其他机密等敏感信息，请使用 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager，而不是将其直接嵌入模板中。请在模板中使用动态引用，以便在堆栈操作期间安全地检索这些值。

在模板中使用参数时，请将敏感参数的 NoEcho 属性设置为 `true`，以防止其值显示在控制台、API 响应或 CLI 输出中。请注意，如果将值传递给可能记录该值的其他服务或资源，NoEcho 不会阻止记录该值。

有关将 Amazon Systems Manager Parameter Store 与 CloudFormation 结合使用的更多信息，请参阅获取 Amazon Systems Manager Parameter Store 中的纯文本值。

有关使用 NoEcho 属性的更多信息，请参阅 CloudFormation 模板 Parameters 语法。

有关将 Amazon Secrets Manager 与 CloudFormation 结合使用的更多信息，请参阅 Create Amazon Secrets Manager secrets in Amazon CloudFormation。

通过 Guard 实施策略即代码

Amazon CloudFormation Guard（`cfn-guard`）是一款策略即代码工具，可让您为 CloudFormation 模板定义和强制执行规则。使用 `cfn-guard` 可确保模板符合组织策略、安全最佳实践和治理要求。

将 `cfn-guard` 集成到 CI/CD 管道中，以便在部署之前根据策略规则自动验证模板。这有助于防止不合规资源部署到环境中，并向开发人员提供有关策略违规行为的早期反馈。

有关设置和使用示例，请参阅使用 Guard 验证模板。

使用 CloudFormation 模板

Amazon CloudFormation 模板定义了您要在堆栈中创建、更新或删除的 Amazon 资源。模板由多个部分组成，但唯一必须具备的是 Resources 部分，其中必须至少声明一种资源。

您可以使用以下方法创建模板：

- 模板工作室：编辑模板并通过可视化图表查看其资源。
- 文本编辑器 – 直接使用 JSON 或 YAML 语法编写模板。
- IaC 生成器 – 利用您账户中已预置但当前不由 CloudFormation 管理的资源生成模板。IaC 生成器适用于云管控 API 在您所在区域支持的多种资源类型。

本部分提供有关如何使用 CloudFormation 模板的不同部分以及如何开始创建堆栈模板的全面指南。其涵盖以下主题：

主题

- 模板的存储位置
- 验证模板
- 开始使用模板
- 示例模板
- CloudFormation 模板格式
- CloudFormation 模板部分
- CloudFormation 语言服务器
- 使用 cfn-lint 验证模板
- 使用 cloudformation-validate 验证模板
- 使用 Guard 验证模板
- 使用模板工作室直观地编辑模板
- 使用 IaC 生成器为现有资源生成模板
- 使用动态引用获取存储在其他服务中的值
- 使用伪参数获取 Amazon 值
- 获取从已部署的 CloudFormation 堆栈导出的输出
- 使用 CloudFormation 提供的参数类型在运行时指定现有资源

- CloudFormation 演练
- CloudFormation 模板代码段
- 使用 CloudFormation 部署基于 Windows 的堆栈
- 使用 CloudFormation 提供的资源类型扩展模板的功能
- 使用 CloudFormation 模块创建可跨模板包含的可重复使用的资源配置

模板的存储位置

Amazon S3 存储桶

您可以将 CloudFormation 模板存储在 Amazon S3 存储桶中。创建或更新堆栈时，您可以指定模板的 S3 URL，而不是直接上传模板。

如果您通过 Amazon Web Services 管理控制台 或 Amazon CLI 直接上传模板，系统将自动为您创建一个 S3 存储桶。有关更多信息，请参阅 通过 CloudFormation 控制台创建堆栈。

Git 存储库

使用 Git sync，您可以将模板存储在 Git 存储库中。创建或更新堆栈时，您可以指定 Git 存储库位置和包含模板的分支，而不是直接上传模板或引用 S3 URL。CloudFormation 会自动监控指定的存储库和分支以查找模板更改。有关更多信息，请参阅 使用 Git 同步从存储库源代码创建堆栈。

验证模板

在部署之前验证模板，以便尽早发现并修复问题。CloudFormation 服务和本节中的工具检查模板的不同部分。您可以将它们配合使用。

您可以使用 validate-template CLI 命令或在 Amazon Web Services 管理控制台 中指定模板来检查模板是否使用了有效的 JSON 或 YAML。控制台会自动执行此检查。这些服务端检查不会运行由以下工具提供的其他属性、安全性或最佳实践检查：

- CloudFormation 语言服务器：在编辑器中编写模板时，获取建议、文档和验证反馈。
- CloudFormation Linter：通过命令行、编辑器或自动构建检查资源属性、允许的值和常见问题。
- CloudFormation 验证：通过命令行或库运行本地离线检查，并以公共表达式语言（CEL）、Rego 或 Guard 添加自定义规则。Amazon CDK 会在合成模板后自动使用此验证器。
- CloudFormation Guard：根据贵组织的安全性、合规性和治理要求编写策略规则并检查模板。

与人工智能编程代理配合使用

AI 编程代理可帮助您使用 GitHub 上的 Agent Toolkit for Amazon 中的 `aws-cloudformation` 技能来创作、验证 CloudFormation 模板并排查其问题。有关 安装说明，请参阅代理设置指南。

了解验证范围

本地工具会检查您提供的模板和规则。它们无法保证在 Amazon Web Services 账户 和 Amazon Web Services 区域 中使用当前资源、权限和配额部署一定会成功。部署之前，请查看更改集并修复 CloudFormation 服务报告的所有问题。

开始使用模板

要开始创建 CloudFormation 模板，请遵循以下步骤：

1. 选择资源 – 确定要包含在堆栈中的 Amazon 资源，例如 EC2 实例、VPC、安全组等。
2. 编写模板 – 以 JSON 或 YAML 格式编写模板，定义资源及其属性。
3. 保存模板：使用以下文件扩展名在本地保存模板：`.json`、`.yaml` 或 `.txt`。
4. 验证模板 – 使用 验证模板 部分中描述的方法验证模板。
5. 创建堆栈 – 使用验证后的模板创建堆栈。

计划使用 CloudFormation 模板引用

在编写模板时，可以在 Amazon 资源和属性类型参考中找到不同资源类型详细语法的文档。

通常，您的堆栈模板需要内置函数来分配直到运行时才可用的属性值，并需要特殊属性来控制资源的行为。在编写模板时，请参阅以下资源以获取指导：

- 内置函数参考：一些常用的内置函数包括：
  - `Ref` – 检索资源的参数值或物理 ID。
  - `Sub` – 用实际值替换字符串中的占位符。
  - `GetAtt` – 返回模板中的资源的属性值。
  - `Join` – 将一组值连接成单个字符串。
- 资源属性参考：一些常用的特殊属性包括：
  - `DependsOn` – 此属性用于指定必须在一个资源创建之后才能创建另一个资源。
  - `DeletionPolicy` – 此属性用于指定 CloudFormation 如何处理资源删除的方式。

示例模板

CloudFormation 提供了您可以开始使用的开源堆栈模板。有关更多信息，请参阅 GitHub 上的 Amazon CloudFormation 示例模板。

请记住，这些模板并不意味着已做好生产的准备。您应该花些时间了解模板的工作方式，根据需要进行调整，并确保它们符合自己公司的合规标准。

该存储库中的每个模板都通过了 CloudFormation Linter (cfn-lint) 检查，以及一组基于 Center for Internet Security (CIS) 前 20 项控制的基本 Amazon CloudFormation Guard 规则，但示例应侧重于单个用例的某些规则除外。

CloudFormation 模板格式

可使用 JSON 或 YAML 格式创作 CloudFormation 模板。这两种格式的目的相同，但在可读性和复杂性方面各有明显的优势。

- JSON – JSON 是一种轻量级的数据交换格式，非常便于机器解析和生成。但对于人类来说，读写可能会变得很麻烦，配置复杂时尤其如此。在 JSON 中，模板使用嵌套的大括号 {} 和方括号 [] 来定义资源、参数和其他组件。其语法要求显式声明每个元素，因此可能会使模板变得冗长，不过可以确保严格遵守结构化的格式。
- YAML – 与 JSON 相比，YAML 的设计更便于人类阅读且不那么冗长。YAML 使用缩进而不是大括号和方括号来表示嵌套，因此更便于直观地显示资源和参数的层次结构。YAML 因清晰的结构和方便易用而备受青睐，尤其是在处理更复杂的模板时。但由于 YAML 依赖缩进，如果间距不一致则可能会导致错误，因此需要仔细注意以保持准确性。

模板结构

CloudFormation 模板可分为不同的部分，每个部分都专用于保存特定类型的信息。有些部分必须按特定顺序声明，而对于其他部分，顺序则无关紧要。但在构建模板时，使用以下示例中显示的逻辑顺序可能非常实用，因为一个部分中的值可能会引用前一个部分中的值。

创建模板时，不要使用重复的主要部分，例如 `Resources` 部分。尽管 CloudFormation 可能会接受该模板，但在处理该模板时将会出现未定义的行为，并且可能会错误地预置资源，或者返回无法解释的错误。

JSON

以下示例演示了 JSON 格式的模板的结构，其中包含所有可用部分。

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Rules" : {
    set of rules
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

YAML

以下示例演示了 YAML 格式的模板的结构，其中包含所有可用部分。

```
---
```

```
AWSTemplateFormatVersion: version date

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Rules:
  set of rules

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

评论

JSON 格式的模板不支持注释。JSON 在设计上未包含注释语法，因此无法直接在 JSON 结构中添加注释。但如果您需要包含解释性的说明或文档，则可以考虑添加元数据。有关更多信息，请参阅 Metadata 属性。

在 YAML 格式的模板中，您可以使用 # 符号包含内联注释。

以下示例介绍一个包含内联注释的 YAML 模板。

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample CloudFormation template with YAML comments.
# Resources section
Resources:
```

```
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      # Linux AMI
      ImageId: ami-1234567890abcdef0
      InstanceType: t2.micro
      KeyName: MyKey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

规格

CloudFormation 支持以下 JSON 和 YAML 规范：

JSON

CloudFormation 遵循 ECMA-404 JSON 标准。有关 JSON 格式的更多信息，请访问 http://www.json.org。

YAML

CloudFormation 支持 YAML 版本 1.1 规范，但有一些例外。CloudFormation 不支持以下功能：

- `binary`、`omap`、`pairs`、`set` 和 `timestamp` 标签
- 别名
- 哈希合并

有关 YAML 的更多信息，请参阅 https://yaml.org/。

了解更多

对于您在模板中指定的每个资源，可以使用 JSON 或 YAML 的特定语法规则定义其属性和值。有关适用于每种格式的模板语法的更多信息，请参阅CloudFormation 模板部分。

在 CloudFormation 模板中使用正则表达式

您可以在 CloudFormation 模板中的多个位置使用正则表达式（通常称为 Regex），例如，在创建模板参数时为 `AllowedPattern` 属性指定正则表达式。

CloudFormation 中的所有正则表达式符合 Java 正则表达式语法。有关 Java 正则表达式语法及其构造的全面说明，请参阅 java.util.regex.Pattern。

如果您使用 JSON 语法编写 CloudFormation 模板，则对于正则表达式中的任何反斜杠字符（`\`），您必须再添加一个反斜杠进行转义。这是因为 JSON 将反斜杠解释为转义字符，因此您需要对反斜杠进行转义，以确保在正则表达式中将其视为反斜杠字符。

例如，如果要与数字字符匹配的正则表达式包含 `\d`，则需要在 JSON 模板中将其写为 `\\d`。

在以下示例中，`AllowedPattern` 属性指定了一个匹配连续四个数字字符（`\d{4}`）的正则表达式。但是，由于正则表达式是在 JSON 模板中定义的，因此需要添加一个额外的反斜杠（`\\d`）来对反斜杠字符进行转义。

```
{
  "Parameters": {
    "MyParameter": {
      "Type": "String",
      "AllowedPattern": "\\d{4}"
    }
  }
}
```

如果您使用 YAML 语法编写 CloudFormation 模板，则必须用单引号（'）将正则表达式括起来。不需要额外的转义。

```
Parameters:
  MyParameter:
    Type: String
    AllowedPattern: '\d{4}'
```

> **Note**
>
> CloudFormation 中的正则表达式仅在特定上下文（如 `AllowedPattern`）中支持用于验证目的。不支持在 CloudFormation 内置函数（如 `Fn::Equals`）中作为模式匹配操作使用，这些函数仅执行精确字符串比较，不进行模式匹配。

CloudFormation 模板部分

每个 CloudFormation 模板都由一个或多个部分组成，每个部分都有特定的用途。

每个 CloudFormation 模板都需要有 Resources 部分，这是模板的核心。这一部分用于指定堆栈资源及其属性，如 Amazon EC2 实例或 Amazon S3 存储桶。每个资源都使用唯一的逻辑 ID、类型和特定的配置详细信息进行定义。

Parameters 部分尽管是可选的，但对于提高模板的灵活性十分重要。这一部分让用户能够在创建或更新堆栈时传递运行时的值。这些参数可以在 `Resources` 和 `Outputs` 部分中引用，从而无需更改模板本身即可进行自定义。例如，您可以使用参数来指定不同部署的实例类型或环境设置。

Outputs 部分也是可选的，用于定义查看堆栈属性时返回的值。输出提供了可用于操作目的或与其他堆栈集成的实用信息，例如资源标识符或 URL 等。这一部分有助于用户检索和使用与模板所创建资源有关的重要详细信息。

其他可选部分包括 Mappings，其功能类似于查找表，用于管理条件值。使用映射，您可以定义键值对，并将其与 `Resources` 和 `Outputs` 部分中的 `Fn::FindInMap` 内置函数结合使用。这对于需要根据条件（例如 Amazon Web Services 区域 或环境）调整配置的场景非常实用。

Metadata 和 Rules 部分虽然不太常用，但提供了额外的功能。`Metadata` 可以包含有关模板的更多信息，而 `Rules` 会在堆栈创建或更新期间验证参数或参数组合，确保符合特定标准。Conditions 部分根据环境类型等条件来控制是否创建特定的资源或者属性分配值，从而进一步提高灵活性。

最后是 Transform 部分，用于在处理模板期间应用宏。对于无服务器应用程序（也称为 Lambda 应用程序），这一部分会指定要使用的 Amazon Serverless Application Model（Amazon SAM）版本。当您指定转换时，可以使用 Amazon SAM 语法声明您的模板中的资源。此模型定义您可使用的语法及其处理方式。您也可以使用 `AWS::Include` 转换来包含与主 CloudFormation 模板分开存储的模板代码片段。

以下主题提供了有关如何使用每个部分的更多信息和示例。

主题

- CloudFormation 模板 Resources 语法
- CloudFormation 模板 Parameters 语法
- CloudFormation 模板 Outputs 语法
- CloudFormation 模板 Mappings 语法
- CloudFormation 模板 Metadata 语法
- CloudFormation 模板 Rules 语法

- CloudFormation 模板 Conditions 语法
- CloudFormation 模板 Transform 部分
- CloudFormation 模板格式版本语法
- CloudFormation 模板 Description 语法

CloudFormation 模板 Resources 语法

Resources 部分是 CloudFormation 模板中必需具备的一个顶级部分，用于声明您希望 CloudFormation 在堆栈中预置和配置的 Amazon 资源。

语法

Resources 部分使用以下语法：

JSON

```
"Resources" : {
    "LogicalResourceName1" : {
        "Type" : "AWS::ServiceName::ResourceType",
        "Properties" : {
            "PropertyName1" : "PropertyValue1",
            ...
        }
    },

    "LogicalResourceName2" : {
        "Type" : "AWS::ServiceName::ResourceType",
        "Properties" : {
            "PropertyName1" : "PropertyValue1",
            ...
        }
    }
}
```

YAML

```
Resources:
  LogicalResourceName1:
    Type: AWS::ServiceName::ResourceType
    Properties:
      PropertyName1: PropertyValue1
```

```
      ...

  LogicalResourceName2:
    Type: AWS::ServiceName::ResourceType
    Properties:
      PropertyName1: PropertyValue1
      ...
```

逻辑 ID（也称为逻辑名称）

在 CloudFormation 模板中，资源由其逻辑资源名称来标识。这些名称必须为字母数字字符（A-Za-z0-9），并且在模板中必须唯一。逻辑名称用于引用模板其他部分中的资源。

资源类型

每个资源必须有一个 Type 属性，用于定义 Amazon 资源的类型。Type 属性的格式为 `AWS::ServiceName::ResourceType`。例如，Amazon S3 存储桶的 Type 属性为 `AWS::S3::Bucket`。

有关支持的资源类型的完整列表，请参阅 Amazon 资源和属性类型参考。

资源属性

资源属性是可以指定的附加选项，用于定义特定资源类型的详细配置。某些属性是必需的，而另一些属性则是可选的。有些属性具有默认值，因此指定这些属性是可选的。

有关每种资源类型支持的属性的详细信息，请参阅 Amazon 资源和属性类型参考中的主题。

属性值可以是文本字符串、字符串列表、布尔值、动态引用、参数引用、伪引用或者函数返回的值。以下示例演示了如何声明不同的属性值类型：

JSON

```
"Properties" : {
    "String" : "A string value",
    "Number" : 123,
    "LiteralList" : [ "first-value", "second-value" ],
    "Boolean" : true
}
```

YAML

```
Properties:
```

```
String: A string value
Number: 123
LiteralList:
  - first-value
  - second-value
Boolean: true
```

物理 ID

除了逻辑 ID 外，某些资源还有物理 ID，这是资源的实际分配名称，如 EC2 实例 ID 或 S3 存储桶名称。使用物理 ID 来标识 CloudFormation 模板外部的资源，但只能在创建了资源之后进行。例如，假设您为一个 EC2 实例资源指定 `MyEC2Instance` 的逻辑 ID。当 CloudFormation 创建实例时，CloudFormation 会自动生成物理 ID（如 `i-1234567890abcdef0`）并将其分配给该实例。您可以使用该物理 ID 来标识实例，可以使用 Amazon EC2 控制台查看其属性 (如 DNS 名称)。

对于 Amazon S3 存储桶和许多其他资源，如果您没有明确指定实体名称，CloudFormation 会自动为该资源生成唯一的实体名称。该实体名称基于 CloudFormation 堆栈的名称、在 CloudFormation 模板中指定的资源逻辑名称和唯一 ID 的组合。例如，假设您在名为 `MyStack` 的堆栈中有一个逻辑名称为 `MyBucket` 的 Amazon S3 存储桶，CloudFormation 可能会使用如下物理名称 `MyStack-MyBucket-abcdefghijk1` 来命名该存储桶。

对于支持自定义名称的资源，您可以分配自己的物理名称来帮助快速标识资源。例如，您可以将存储日志的 S3 存储桶命名为 `MyPerformanceLogs`。有关更多信息，请参阅名称类型。

引用资源

您经常会需要根据一个资源的名称或属性来设置另一个资源的属性。例如，您可以创建一个使用 EC2 安全组的 EC2 实例，或由 S3 存储桶支持的 CloudFront 分配。所有这些资源都可在同一 CloudFormation 模板中创建。

CloudFormation 提供了内置函数，您可以使用这些函数来引用其他资源及其属性。这些函数让您可以在资源之间创建依赖关系，并将一个资源的值传递到另一个资源。

`Ref` 函数

`Ref` 函数通常用于检索在同一 CloudFormation 模板中定义的资源的标识属性。具体返回的值取决于资源的类型。对于大多数资源，该函数会返回资源的实体名称。不过对于某些资源类型，该函数可能会返回其他的值，例如 `AWS::EC2::EIP` 资源的 IP 地址或 Amazon SNS 主题的 Amazon 资源名称（ARN）。

以下示例演示了如何在属性中使用 Ref 函数。在以下的每个示例中，Ref 函数都将返回模板中其他部分声明的 LogicalResourceName 资源的实际名称。YAML 示例中的 !Ref 语法示例只是编写 Ref 函数的较简单方法。

JSON

```
"Properties" : {
    "PropertyName" : { "Ref" : "LogicalResourceName" }
}
```

YAML

```
Properties:
  PropertyName1:
    Ref: LogicalResourceName
  PropertyName2: !Ref LogicalResourceName
```

有关使用 Ref 函数的更多详细信息，请参阅 Ref。

Fn::GetAtt 函数

如果参数或为资源返回的值正是您需要的，则 Ref 函数会非常实用。不过，您可能需要资源的其他属性。例如，如果您想创建一个具有 S3 起始地址的 CloudFront 分配，就需要使用 DNS 样式的地址来指定存储桶的位置。很多资源都具有附加属性，您可以在模板中使用这些属性的值。要获取这些属性，请使用 Fn::GetAtt 函数。

以下示例演示了如何在属性中使用 GetAtt 函数。Fn::GetAtt 函数有两个参数，分别是资源的逻辑名称和要检索的属性名称。YAML 示例中的 !GetAtt 语法示例只是编写 GetAtt 函数的较简单方法。

JSON

```
"Properties" : {
    "PropertyName" : {
        "Fn::GetAtt" : [ "LogicalResourceName", "AttributeName" ]
    }
}
```

YAML

```
Properties:
  PropertyName1:
    Fn::GetAtt:
```

```
      - LogicalResourceName
      - AttributeName
  PropertyName2: !GetAtt LogicalResourceName.AttributeName
```

有关使用 GetAtt 函数的更多详细信息，请参阅 Fn::GetAtt。

示例

以下示例演示了如何声明资源以及 CloudFormation 模板会如何引用在同一模板中定义的其他资源以及现有的 Amazon 资源。

主题

- 使用自定义名称声明单个资源
- 使用 Ref 函数引用其他资源
- 使用 Fn::GetAtt 函数引用资源属性

使用自定义名称声明单个资源

以下示例声明了 AWS::S3::Bucket 类型的单个资源，逻辑名称为 MyBucket。BucketName 属性设置为 *amzn-s3-demo-bucket*，因此应替换为您的 S3 存储桶所需的名称。

如果您使用此资源声明创建堆栈，CloudFormation 将使用默认设置创建 Amazon S3 存储桶。对于 Amazon EC2 实例或自动扩缩组等其他资源，CloudFormation 需要更多信息。

JSON

```
{
    "Resources": {
        "MyBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": "amzn-s3-demo-bucket"
            }
        }
    }
}
```

YAML

```
Resources:
  MyBucket:
```

```
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
```

使用 **Ref** 函数引用其他资源

以下示例演示的资源声明定义了一个 EC2 实例和安全组。Ec2Instance 资源使用 Ref 函数引用 InstanceSecurityGroup 资源，以作为其 SecurityGroupIds 属性的一部分。此外还包括未在模板中声明的一个现有安全组 (sg-12a4c434)。您可以使用文字字符串来指代现有 Amazon 资源。

JSON

```
{
    "Resources": {
        "Ec2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "SecurityGroupIds": [
                    {
                        "Ref": "InstanceSecurityGroup"
                    },
                    "sg-12a4c434"
                ],
                "KeyName": "MyKey",
                "ImageId": "ami-1234567890abcdef0"
            }
        },
        "InstanceSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Enable SSH access via port 22",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 22,
                        "ToPort": 22,
                        "CidrIp": "0.0.0.0/0"
                    }
                ]
            }
        }
    }
}
```

YAML

```
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
        - sg-12a4c434
      KeyName: MyKey
      ImageId: ami-1234567890abcdef0
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
```

使用 **Fn::GetAtt** 函数引用资源属性

以下示例演示的资源声明定义了一个 CloudFront 分配资源和一个 S3 存储桶。MyDistribution 资源指定了 MyBucket 资源的 DNS 名称，后者使用 Fn::GetAtt 函数来获取存储桶的 DomainName 属性。Fn::GetAtt 函数会在一个数组中列出这两个参数。对于采用多个参数的函数，可以使用数组来指定其参数。

JSON

```
{
  "Resources": {
    "MyBucket": {
      "Type": "AWS::S3::Bucket"
    },
    "MyDistribution": {
      "Type": "AWS::CloudFront::Distribution",
      "Properties": {
        "DistributionConfig": {
          "Origins": [
            {
              "DomainName": {
                "Fn::GetAtt": [
```

```
                "MyBucket",
                "DomainName"
              ]
            },
            "Id": "MyS3Origin",
            "S3OriginConfig": {}
          }
        ],
        "Enabled": "true",
        "DefaultCacheBehavior": {
          "TargetOriginId": "MyS3Origin",
          "ForwardedValues": {
            "QueryString": "false"
          },
          "ViewerProtocolPolicy": "allow-all"
        }
      }
    }
  }
}
```

YAML

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
  MyDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
          - DomainName: !GetAtt
              - MyBucket
              - DomainName
            Id: MyS3Origin
            S3OriginConfig: {}
        Enabled: 'true'
        DefaultCacheBehavior:
          TargetOriginId: MyS3Origin
          ForwardedValues:
            QueryString: 'false'
          ViewerProtocolPolicy: allow-all
```

CloudFormation 模板 Parameters 语法

使用可选的 Parameters 部分来自定义模板。利用参数，您能够在每次创建或更新堆栈时在模板中输入自定义值。通过在模板中使用参数，您可以构建可重复使用的灵活模板，以便根据具体场景进行自定义。

通过定义适当类型的参数，您可以在使用控制台创建堆栈时从现有资源的标识符列表中进行选择。有关更多信息，请参阅 使用 CloudFormation 提供的参数类型在运行时指定现有资源。

参数是指定堆栈资源属性值的一种常用方法。但是，可能会有一些因区域而异的设置，或是一些因其他条件或依赖关系而需要用户确定的复杂设置。在这些情况下，您可能需要将一些逻辑放入模板本身中，以便用户可以指定简单值（或不指定任何值）来获得期望的结果，例如通过使用映射。有关更多信息，请参阅 CloudFormation 模板 Mappings 语法。

语法

您需要在模板的 Parameters 部分声明参数，该部分会使用以下通用语法：

JSON

```
"Parameters" : {
  "ParameterLogicalID" : {
    "Description": "Information about the parameter",
    "Type" : "DataType",
    "Default" : "value",
    "AllowedValues" : ["value1", "value2"]
  }
}
```

YAML

```
Parameters:
  ParameterLogicalID:
    Description: Information about the parameter
    Type: DataType
    Default: value
    AllowedValues:
      - value1
      - value2
```

一个参数包含一系列属性，这些属性定义了其值但也限制了其值。唯一必需的属性是 Type，该属性可以是 String、Number 或 CloudFormation 提供的参数类型。您还可以添加 Description 属性来描述要指定的值的类型。在创建堆栈向导中使用该模板时，参数的名称和描述会在指定参数页面中显示。

> **Note**
>
> 默认情况下，CloudFormation 控制台按照输入参数的逻辑 ID 的字母顺序列出这些参数。要覆盖此默认排序并将相关参数组合在一起，您可以在模板中使用 AWS::CloudFormation::Interface 元数据键。有关更多信息，请参阅 使用 AWS::CloudFormation::Interface 元数据整理 CloudFormation 参数。

对于具有默认值的参数，CloudFormation 将使用默认值，除非用户指定其他值。如果省略默认属性，则用户将需要指定该参数的值。但是，要求用户输入值并不能确保该值有效。要验证参数的值，您可以声明一些约束或指定某个特定于 Amazon 的参数类型。

对于没有默认值的参数，用户必须在创建堆栈时指定一个键名称值。否则，CloudFormation 将无法创建堆栈并引发异常：

```
Parameters: [KeyName] must have values
```

属性

`AllowedPattern`

一个正则表达式，表示要允许 String 和 CommaDelimitedList 类型使用的模式。应用于类型为 String 的参数时，模式必须与提供的整个参数值匹配。应用于类型为 CommaDelimitedList 的参数时，模式必须与列表中的每个值匹配。

必需：否

`AllowedValues`

包含参数允许值列表的阵列。应用于类型为 String 的参数时，该参数值必须是允许的值之一。应用于类型为 CommaDelimitedList 的参数时，列表中的每个值都必须是指定的允许的值之一。

必需：否

> **Note**
>
> 如果您使用 YAML，并且想要在 AllowedValues 中使用 Yes 和 No 字符串，请使用单引号来防止 YAML 解析器考虑这些布尔值。

ConstraintDescription

用于在违反约束时说明该约束的字符串。例如，在没有约束条件描述的情况下，具有允许的 [A-Za-z0-9]+ 模式的参数会在用户指定无效值时显示以下错误消息：

```
Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+
```

通过添加约束描述（如 must only contain letters (uppercase and lowercase) and numbers），您可以显示以下自定义的错误消息：

```
Malformed input-Parameter MyParameter must only contain uppercase and
lowercase letters and numbers
```

必需：否

Default

模板适当类型的值，用于在创建堆栈时未指定值的情况下。如果您定义参数的约束，则必须指定一个符合这些约束的值。

必需：否

Description

用于描述参数的长度最多为 4000 个字符的字符串。

必需：否

MaxLength

一个整数值，确定要允许 String 类型使用的字符的最大数目。

必需：否

MaxValue

一个数字值，确定要允许 Number 类型使用的最大数字值。

必需：否

`MinLength`

一个整数值，确定要允许 `String` 类型使用的字符的最小数目。

必需：否

`MinValue`

一个数字值，确定要允许 `Number` 类型使用的最小数字值。

必需：否

`NoEcho`

是否遮蔽参数值以防止在控制台、命令行或 API 中显示。如果您将 `NoEcho` 属性设置为 `true`，则对于描述堆栈或堆栈事件的任何调用，CloudFormation 返回使用星号 (*****) 遮蔽的参数值，但存储在下面指定位置的信息除外。

必需：否

> **Important**
>
> 使用 `NoEcho` 属性不会遮蔽在以下各区段中存储的任何信息：
>
> - `Metadata` 模板区段。CloudFormation 不会转换、修改或编辑您在 `Metadata` 部分中包含的任何信息。有关更多信息，请参阅 Metadata。
> - `Outputs` 模板区段。有关更多信息，请参阅 Outputs。
> - 资源定义的 `Metadata` 属性。有关更多信息，请参阅 `Metadata` 属性。
>
> 强烈建议您不要使用这些机制来包含敏感信息，例如密码。

> **Important**
>
> 我们建议不要将敏感信息直接嵌入 CloudFormation 模板中，而应使用堆栈模板中的动态参数来引用在 CloudFormation 外部存储和管理的敏感信息，例如 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager 中的敏感信息。
> 有关更多信息，请参阅请勿将凭证嵌入您的模板最佳实践。

> **Important**
>
> 我们建议不要在作为资源主标识符一部分的资源属性中包含 NoEcho 参数或任何敏感数据。
> 当 NoEcho 参数包含在构成主资源标识符的属性中时，CloudFormation 可能会在主资源标识符中使用实际明文值。此资源 ID 可能出现在任何派生输出或目标中。
> 要确定哪些资源属性构成了资源类型的主标识符，请参阅 Amazon 资源和属性类型参考中该资源的资源参考文档。在 Return values（返回值）部分，Ref 函数返回值，表示构成资源类型主标识符的资源属性。

`Type`

参数 (DataType) 的数据类型。

是否必需：是

CloudFormation 支持以下参数类型：

`String`

一个文字字符串。您可以使用以下属性来声明约束：MinLength、MaxLength、Default、AllowedValues 和 AllowedPattern。

例如，用户可指定 "MyUserName"。

`Number`

整型或浮点型。CloudFormation 将参数值作为数字进行验证；但当您在模板中的其他位置使用参数时（例如，通过使用 Ref 内置函数），该参数值将变为字符串。

您可以使用以下属性来声明约束：MinValue、MaxValue、Default 和 AllowedValues。

例如，用户可指定 "8888"。

`List<Number>`

以逗号分隔的整型或浮点型数组。CloudFormation 将参数值作为数字进行验证；但当您在模板中的其他位置使用参数时（例如，通过使用 Ref 内置函数），该参数值将变为字符串列表。

例如，用户可指定 "80,20"，并且 Ref 将生成 ["80","20"]。

`CommaDelimitedList`

一组用逗号分隔的文本字符串。字符串的总数应比逗号总数多 1。此外，会对每个成员字符串进行空间修剪。

例如，用户可指定 `"test,dev,prod"`，并且 `Ref` 将生成 `["test","dev","prod"]`。

特定于 Amazon 的参数类型

Amazon 值，例如 Amazon EC2 密钥对名称和 VPC ID。有关更多信息，请参阅 在运行时指定现有资源。

Systems Manager 参数类型

与 Systems Manager Parameter Store 中的现有参数对应的参数。您指定 Systems Manager 参数键作为 Systems Manager 参数类型的值，然后 CloudFormation 从 Parameter Store 中检索最新的值来用于堆栈。有关更多信息，请参阅 在运行时指定现有资源。

参数的一般要求

使用参数时存在以下要求：

- 一个 CloudFormation 模板中最多可包含 200 个参数。
- 必须为每个参数提供一个逻辑名称（也称为逻辑 ID），该名称必须是字母数字，并且在模板内的所有逻辑名称中必须唯一。
- 必须向每个参数分配一个 CloudFormation 支持的参数类型。有关更多信息，请参阅类型。
- 必须向每个参数分配一个运行时的值，这样 CloudFormation 才能成功预置堆栈。您可以选择为要使用的 CloudFormation 指定默认值，除非提供有其他值。
- 必须在同一模板内声明和引用参数。您可以引用模板的 `Resources` 和 `Outputs` 部分中的参数。

示例

主题

- 简单字符串参数
- 密码参数
- 引用参数
- 逗号分隔列表参数
- 从逗号分隔列表参数返回值

简单字符串参数

以下示例声明了一个名为 InstanceTypeParameter，类型为 String 的参数。利用此参数，您可以为堆栈指定 Amazon EC2 实例类型。如果在创建或更新堆栈期间未提供任何值，CloudFormation 将使用 t2.micro 的默认值。

JSON

```
"Parameters" : {
  "InstanceTypeParameter" : {
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro.",
    "Type" : "String",
    "Default" : "t2.micro",
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"]
  }
}
```

YAML

```
Parameters:
  InstanceTypeParameter:
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
```

密码参数

以下示例声明了一个名为 DBPwd，类型为 String 且没有默认值的参数。将 NoEcho 属性设置为 true 是为了防止在堆栈描述中显示参数值。可指定的最小长度为 1，可指定的最大长度为 41。该模式允许小写和大写字母字符和数字。此示例还演示了 AllowedPattern 属性的正则表达式用法。

JSON

```
"Parameters" : {
  "DBPwd" : {
    "NoEcho" : "true",
    "Description" : "The database admin account password",
    "Type" : "String",
```

```
    "MinLength" : "1",
    "MaxLength" : "41",
    "AllowedPattern" : "^[a-zA-Z0-9]*$"
  }
}
```

YAML

```
Parameters:
  DBPwd:
    NoEcho: true
    Description: The database admin account password
    Type: String
    MinLength: 1
    MaxLength: 41
    AllowedPattern: ^[a-zA-Z0-9]*$
```

引用参数

您可以使用 Ref 内置函数来引用某个参数，然后 CloudFormation 将使用该参数的值来预置堆栈。您可以引用同一模板的 Resources 和 Outputs 部分中的参数。

在以下示例中，EC2 实例资源的 InstanceType 属性引用了 InstanceTypeParameter 参数值：

JSON

```
"Ec2Instance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "InstanceType" : { "Ref" : "InstanceTypeParameter" },
    "ImageId" : "ami-0ff8a91507f77f867"
  }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType:
      Ref: InstanceTypeParameter
    ImageId: ami-0ff8a91507f77f867
```

逗号分隔列表参数

需要为单个属性提供多个值时，CommaDelimitedList 参数类型可能非常实用。以下示例声明了一个名为 DbSubnetIpBlocks 的参数，其默认值为用逗号分隔的三个 CIDR 块。

JSON

```
"Parameters" : {
  "DbSubnetIpBlocks": {
    "Description": "Comma-delimited list of three CIDR blocks",
    "Type": "CommaDelimitedList",
    "Default": "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
  }
}
```

YAML

```
Parameters:
  DbSubnetIpBlocks:
    Description: "Comma-delimited list of three CIDR blocks"
    Type: CommaDelimitedList
    Default: "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
```

从逗号分隔列表参数返回值

要引用以逗号分隔的参数列表中的特定值，请在模板的 Resources 部分中使用 Fn::Select 内置函数。可传递所需对象的索引值以及对象列表，如以下示例所示。

JSON

```
{
    "Parameters": {
        "VPC": {
            "Type": "String",
            "Default": "vpc-123456"
        },
        "VpcAzs": {
            "Type": "CommaDelimitedList",
            "Default": "us-west-2a, us-west-2b, us-west-2c"
        },
        "DbSubnetIpBlocks": {
            "Type": "CommaDelimitedList",
            "Default": "172.16.0.0/26, 172.16.0.64/26, 172.16.0.128/26"
```

```
        }
    },
    "Resources": {
        "DbSubnet1": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "AvailabilityZone": {
                    "Fn::Select": [
                      0,
                      {
                        "Ref": "VpcAzs"
                      }
                   ]
                },
                "VpcId": {
                    "Ref": "VPC"
                },
                "CidrBlock": {
                    "Fn::Select": [
                        0,
                        { "Ref": "DbSubnetIpBlocks" }
                    ]
                }
            }
        },
        "DbSubnet2": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "AvailabilityZone": {
                    "Fn::Sub": [
                        "${AWS::Region}${AZ}",
                        {
                            "AZ": {
                                "Fn::Select": [
                                    1,
                                    { "Ref": "VpcAzs" }
                                ]
                            }
                        }
                    ]
                },
                "VpcId": {
                    "Ref": "VPC"
                },
```

```
                "CidrBlock": {
                    "Fn::Select": [
                        1,
                        { "Ref": "DbSubnetIpBlocks" }
                    ]
                }
            }
        },
        "DbSubnet3": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "AvailabilityZone": {
                    "Fn::Sub": [
                        "${AWS::Region}${AZ}",
                        {
                            "AZ": {
                                "Fn::Select": [
                                    2,
                                    { "Ref": "VpcAzs" }
                                ]
                            }
                        }
                    ]
                },
                "VpcId": {
                    "Ref": "VPC"
                },
                "CidrBlock": {
                    "Fn::Select": [
                        2,
                        { "Ref": "DbSubnetIpBlocks" }
                    ]
                }
            }
        }
    }
}
```

YAML

```
Parameters:
  VPC:
    Type: String
```

```
    Default: vpc-123456
  VpcAzs:
    Type: CommaDelimitedList
    Default: us-west-2a, us-west-2b, us-west-2c
  DbSubnetIpBlocks:
    Type: CommaDelimitedList
    Default: 172.16.0.0/26, 172.16.0.64/26, 172.16.0.128/26
Resources:
  DbSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Select
        - 0
        - !Ref VpcAzs
      VpcId: !Ref VPC
      CidrBlock: !Select
        - 0
        - !Ref DbSubnetIpBlocks
  DbSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Sub
        - ${AWS::Region}${AZ}
        - AZ: !Select
            - 1
            - !Ref VpcAzs
      VpcId: !Ref VPC
      CidrBlock: !Select
        - 1
        - !Ref DbSubnetIpBlocks
  DbSubnet3:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Sub
        - ${AWS::Region}${AZ}
        - AZ: !Select
            - 2
            - !Ref VpcAzs
      VpcId: !Ref VPC
      CidrBlock: !Select
        - 2
        - !Ref DbSubnetIpBlocks
```

相关资源

CloudFormation 还支持使用动态引用来动态指定属性值。例如，您可能需要引用存储在 Systems Manager Parameter Store 中的安全字符串。有关更多信息，请参阅 使用动态引用获取存储在其他服务中的值。

您也可以在 Ref 或 Sub 函数中使用伪参数来动态填充值。有关更多信息，请参阅 使用伪参数获取 Amazon 值。

CloudFormation 模板 Outputs 语法

可选的 Outputs 部分用于声明堆栈的输出值。这些输出值可以通过多种方式使用：

- 捕获与资源有关的重要信息 – 通过输出可以方便地捕获与资源有关的重要信息。例如，您可以输出堆栈的 S3 存储桶名称以使该存储桶更容易找到。可以在 CloudFormation 控制台的输出选项卡中或使用 describe-stacks CLI 命令查看输出值。
- 跨堆栈引用 – 您可以将输出值导入其他堆栈，以便在堆栈之间创建引用。当您需要跨多个堆栈共享资源或配置时，这会非常实用。
- 跨账户和跨区域引用 - 您可以使用 Fn::GetStackOutput 函数引用其他 Amazon 账户或区域堆栈中的输出值，无需显式导出声明。有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 Fn::GetStackOutput。

> **Important**
>
> CloudFormation 不会对 Outputs 部分中包含的任何信息进行编辑或模糊处理。我们强烈建议您不要使用此部分输出敏感信息，例如密码或密钥。
> 堆栈操作完成后，输出值可用。当堆栈状态处于任何 IN_PROGRESS 状态时，堆栈输出值都不可用。我们不建议在服务运行时和堆栈输出值之间建立依赖关系，因为输出值可能并非始终都可用。

语法

Outputs 部分包括键名称 Outputs。您最多可在一个模板中声明 200 个输出。

以下示例演示了 Outputs 部分的结构。

JSON

使用括号将所有输出声明括起来。使用逗号分隔多个输出。

```
"Outputs" : {
  "OutputLogicalID" : {
    "Description" : "Information about the value",
    "Value" : "Value to return",
    "Export" : {
      "Name" : "Name of resource to export"
    }
  }
}
```

YAML

```
Outputs:
  OutputLogicalID:
    Description: Information about the value
    Value: Value to return
    Export:
      Name: Name of resource to export
```

输出字段

Outputs 部分包含以下字段。

逻辑 ID（也称为逻辑名称）

当前输出的标识符。逻辑 ID 必须为字母数字（a-z、A-Z 和 0-9），并且在模板中具有唯一性。

Description（可选）

一个描述输出类型的 String 类型。描述声明的值必须是长度介于 0 和 1024 个字节之间的文字字符串。您无法使用参数或函数来指定描述。

Value（必需）

describe-stacks 命令返回的属性值。输出值可以包括文字、参数引用、伪参数、映射值或内置函数。

Export（可选）

要为跨堆栈引用导出的资源输出的名称。

您可使用内部函数自定义导出的 Name 值。

有关更多信息，请参阅 获取从已部署的 CloudFormation 堆栈导出的输出。

要将条件与输出关联，请在模板的 Conditions 部分中定义条件。

示例

以下示例说明了堆栈输出的工作原理。

主题

- 堆栈输出
- 使用 Fn::Sub 自定义导出名称
- 使用 Fn::Join 自定义导出名称
- 返回使用 Fn::Join 构造的 URL

堆栈输出

在以下示例中，仅当 BackupLoadBalancerDNSName 条件为 true 时，名为 BackupLoadBalancer 的输出才返回逻辑 ID 为 CreateProdResources 的资源的 DNS 名称。名为 InstanceID 的输出会返回逻辑 ID 为 EC2Instance 的 EC2 实例的 ID。

JSON

```
"Outputs" : {
  "BackupLoadBalancerDNSName" : {
    "Description": "The DNSName of the backup load balancer",
    "Value" : { "Fn::GetAtt" : [ "BackupLoadBalancer", "DNSName" ]},
    "Condition" : "CreateProdResources"
  },
  "InstanceID" : {
    "Description": "The Instance ID",
    "Value" : { "Ref" : "EC2Instance" }
  }
}
```

YAML

```
Outputs:
```

```
  BackupLoadBalancerDNSName:
    Description: The DNSName of the backup load balancer
    Value: !GetAtt BackupLoadBalancer.DNSName
    Condition: CreateProdResources
  InstanceID:
    Description: The Instance ID
    Value: !Ref EC2Instance
```

使用 **Fn::Sub** 自定义导出名称

在以下示例中，名为 StackVPC 的输出返回 VPC 的 ID，然后在堆栈名称中附加 VPCID 导出跨堆栈引用的值。

JSON

```
"Outputs" : {
  "StackVPC" : {
    "Description" : "The ID of the VPC",
    "Value" : { "Ref" : "MyVPC" },
    "Export" : {
      "Name" : {"Fn::Sub": "${AWS::StackName}-VPCID" }
    }
  }
}
```

YAML

```
Outputs:
  StackVPC:
    Description: The ID of the VPC
    Value: !Ref MyVPC
    Export:
      Name: !Sub "${AWS::StackName}-VPCID"
```

有关 Fn::Sub 函数的更多信息，请参阅Fn::Sub。

使用 **Fn::Join** 自定义导出名称

还可以使用 Fn::Join 函数以根据参数、资源属性和其他字符串来构造值。

以下示例使用 Fn::Join 函数而不是 Fn::Sub 函数，自定义导出名称。示例 Fn::Join 函数将冒号用作分隔符，连接堆栈名称与名称 VPCID。

JSON

```
"Outputs" : {
  "StackVPC" : {
    "Description" : "The ID of the VPC",
    "Value" : { "Ref" : "MyVPC" },
    "Export" : {
      "Name" : { "Fn::Join" : [ ":", [ { "Ref" : "AWS::StackName" }, "VPCID" ] ] }
    }
  }
}
```

YAML

```
Outputs:
  StackVPC:
    Description: The ID of the VPC
    Value: !Ref MyVPC
    Export:
      Name: !Join [ ":", [ !Ref "AWS::StackName", VPCID ] ]
```

有关 `Fn::Join` 函数的更多信息，请参阅Fn::Join。

返回使用 **`Fn::Join`** 构造的 URL

在以下用于创建 WordPress 网站的模板示例中，`InstallURL` 是 `Fn::Join` 函数调用返回的字符串，该字符串将 `http://`、资源 `ElasticLoadBalancer` 的 DNS 名称和 `/wp-admin/install.php` 连接起来。该输出值可能会与以下值相似：

```
http://mywptests-elasticl-1gb51l6sl8y5v-206169572.aws-region.elb.amazonaws.com/wp-
admin/install.php
```

JSON

```
{
    "Outputs": {
        "InstallURL": {
            "Value": {
                "Fn::Join": [
                    "",
                    [
                        "http://",
```

```
                        {
                            "Fn::GetAtt": [
                                "ElasticLoadBalancer",
                                "DNSName"
                            ]
                        },
                        "/wp-admin/install.php"
                    ]
                ]
            },
            "Description": "Installation URL of the WordPress website"
        }
    }
}
```

YAML

```
Outputs:
  InstallURL:
    Value: !Join
      - ''
      - - 'http://'
        - !GetAtt
          - ElasticLoadBalancer
          - DNSName
        - /wp-admin/install.php
    Description: Installation URL of the WordPress website
```

有关 Fn::Join 函数的更多信息，请参阅Fn::Join。

CloudFormation 模板 Mappings 语法

您可以借助可选的 Mappings 部分来创建键值对，从而可以根据特定条件或依赖项指定值。

Mappings 部分的一个常见应用场景是根据部署堆栈的 Amazon Web Services 区域来设置值。这可以通过使用 AWS::Region 伪参数来实现。AWS::Region 伪参数是 CloudFormation 用来解析到创建堆栈所在区域的值。在创建堆栈时，CloudFormation 解析了这些伪参数。

要检索映射中的值，您可以在模板的 Resources 部分中使用 Fn::FindInMap 内置函数。

语法

Mappings 部分使用以下语法：

JSON

```
"Mappings" : {
  "MappingLogicalName" : {
    "Key1" : {
      "Name" : "Value1"
    },
    "Key2" : {
      "Name" : "Value2"
    },
    "Key3" : {
      "Name" : "Value3"
    }
  }
}
```

YAML

```
Mappings:
  MappingLogicalName:
    Key1:
      Name: Value1
    Key2:
      Name: Value2
    Key3:
      Name: Value3
```

- MappingLogicalName 是映射的逻辑名称。
- 在映射中，每个映射是一个键，后跟另一个映射。
- 键必须是名称/值对映射，且在映射中必须唯一。
- 名值对是一个标签，该值是用来映射的。您可以通过给值命名以将多组值映射到密钥中。
- 映射中的键必须为文字字符串。
- 值可以是 String 或 List 类型。

> **Note**
>
> 不得在 Mappings 部分包含参数、伪参数或内部函数。

如果堆栈目前未使用映射中的值，则无法单独更新映射。您必须包含添加、修改或删除资源的更改。

示例

主题

- 基本映射
- 带有多个值的映射
- 从映射返回值
- 输入参数和 Fn::FindInMap

基本映射

以下示例显示的是带映射 Mappings 的 RegionToInstanceType 部分，该映射包含五个映射到含单字符串值的名称-值对的密钥。密钥为区域名称。每个名称值对都是该键所表示区域内的 T 系列中可用的一种实例类型。名称值对中有一个名称（示例中的 InstanceType）和一个值。

JSON

```
"Mappings" : {
  "RegionToInstanceType" : {
    "us-east-1"      : { "InstanceType" : "t2.micro" },
    "us-west-1"      : { "InstanceType" : "t2.micro" },
    "eu-west-1"      : { "InstanceType" : "t2.micro" },
    "eu-north-1"     : { "InstanceType" : "t3.micro" },
    "me-south-1"     : { "InstanceType" : "t3.micro" }
  }
}
```

YAML

```
Mappings:
  RegionToInstanceType:
    us-east-1:
      InstanceType: t2.micro
    us-west-1:
      InstanceType: t2.micro
```

```
    eu-west-1:
      InstanceType: t2.micro
    eu-north-1:
      InstanceType: t3.micro
    me-south-1:
      InstanceType: t3.micro
```

带有多个值的映射

以下示例中的区域键映射到两组值：一组名为 MyAMI1，另一组为 MyAMI2。

> **Note**
>
> 这些示例中显示的 AMI ID 是演示用的占位符。应尽量考虑使用 Amazon Systems Manager 参数的动态引用，作为 Mappings 部分的替代方案。要避免每次要使用的 AMI 发生更改时都使用新 ID 来更新所有模板，请在创建或更新堆栈时使用 Amazon Systems Manager 参数来检索最新 AMI ID。常用 AMI 的最新版本也可作为 Systems Manager 中的公有参数提供。有关更多信息，请参阅使用动态引用获取存储在其他服务中的值。

JSON

```
"Mappings" : {
  "RegionToAMI" : {
    "us-east-1"        : { "MyAMI1" : "ami-12345678901234567", "MyAMI2" :
 "ami-23456789012345678" },
    "us-west-1"        : { "MyAMI1" : "ami-34567890123456789", "MyAMI2" :
 "ami-45678901234567890" },
    "eu-west-1"        : { "MyAMI1" : "ami-56789012345678901", "MyAMI2" :
 "ami-67890123456789012" },
    "ap-southeast-1"   : { "MyAMI1" : "ami-78901234567890123", "MyAMI2" :
 "ami-89012345678901234" },
    "ap-northeast-1"   : { "MyAMI1" : "ami-90123456789012345", "MyAMI2" :
 "ami-01234567890123456" }
  }
}
```

YAML

```
Mappings:
  RegionToAMI:
```

```
    us-east-1:
      MyAMI1: ami-12345678901234567
      MyAMI2: ami-23456789012345678
    us-west-1:
      MyAMI1: ami-34567890123456789
      MyAMI2: ami-45678901234567890
    eu-west-1:
      MyAMI1: ami-56789012345678901
      MyAMI2: ami-67890123456789012
    ap-southeast-1:
      MyAMI1: ami-78901234567890123
      MyAMI2: ami-89012345678901234
    ap-northeast-1:
      MyAMI1: ami-90123456789012345
      MyAMI2: ami-01234567890123456
```

从映射返回值

您可以使用 Fn::FindInMap 函数根据指定的密钥返回命名的值。以下示例模板包含由 FindInMap 函数分配其 InstanceType 属性的 Amazon EC2 资源。FindInMap 函数将键指定为创建堆栈的 Amazon Web Services 区域（使用 AWS::Region 伪参数），并将 InstanceType 指定为要映射到的值的名称。ImageId 使用 Systems Manager 参数动态检索最新的 Amazon Linux 2 AMI。有关伪参数的更多信息，请参阅使用伪参数获取 Amazon 值。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Mappings" : {
    "RegionToInstanceType" : {
      "us-east-1"      : { "InstanceType" : "t2.micro" },
      "us-west-1"      : { "InstanceType" : "t2.micro" },
      "eu-west-1"      : { "InstanceType" : "t2.micro" },
      "eu-north-1"     : { "InstanceType" : "t3.micro" },
      "me-south-1"     : { "InstanceType" : "t3.micro" }
    }
  },
  "Resources" : {
    "myEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
```

```
        "InstanceType" : { "Fn::FindInMap" : [ "RegionToInstanceType", { "Ref" :
 "AWS::Region" }, "InstanceType" ]}
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Mappings:
  RegionToInstanceType:
    us-east-1:
      InstanceType: t2.micro
    us-west-1:
      InstanceType: t2.micro
    eu-west-1:
      InstanceType: t2.micro
    eu-north-1:
      InstanceType: t3.micro
    me-south-1:
      InstanceType: t3.micro
Resources:
  myEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      InstanceType: !FindInMap [RegionToInstanceType, !Ref 'AWS::Region', InstanceType]
```

输入参数和 **Fn::FindInMap**

以下示例模板显示如何使用多个映射来创建 EC2 实例。该模板使用嵌套映射，根据目标 Amazon Web Services 区域 和环境类型（`Dev` 或 `Prod`）自动选择相应的实例类型和安全组。模板还使用 Systems Manager 参数来动态检索最新的 Amazon Linux 2 AMI。

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Parameters" : {
    "EnvironmentType" : {
```

```
      "Description" : "The environment type (Dev or Prod)",
      "Type" : "String",
      "Default" : "Dev",
      "AllowedValues" : [ "Dev", "Prod" ]
    }
  },
  "Mappings" : {
    "RegionAndEnvironmentToInstanceType" : {
      "us-east-1"        : { "Dev" : "t3.micro", "Prod" : "c5.large" },
      "us-west-1"        : { "Dev" : "t2.micro", "Prod" : "m5.large" }
    },
    "RegionAndEnvironmentToSecurityGroup" : {
      "us-east-1"        : { "Dev" : "sg-12345678", "Prod" : "sg-abcdef01" },
      "us-west-1"        : { "Dev" : "sg-ghijkl23", "Prod" : "sg-45678abc" }
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
        "InstanceType" : { "Fn::FindInMap": [ "RegionAndEnvironmentToInstanceType",
 { "Ref": "AWS::Region" }, { "Ref": "EnvironmentType" } ]},
        "SecurityGroupIds" : [{ "Fn::FindInMap" :
 [ "RegionAndEnvironmentToSecurityGroup", { "Ref" : "AWS::Region" }, { "Ref" :
 "EnvironmentType" } ]}]
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvironmentType:
    Description: The environment type (Dev or Prod)
    Type: String
    Default: Dev
    AllowedValues:
      - Dev
      - Prod
```

```
Mappings:
  RegionAndEnvironmentToInstanceType:
    us-east-1:
      Dev: t3.micro
      Prod: c5.large
    us-west-1:
      Dev: t2.micro
      Prod: m5.large
  RegionAndEnvironmentToSecurityGroup:
    us-east-1:
      Dev: sg-12345678
      Prod: sg-abcdef01
    us-west-1:
      Dev: sg-ghijkl23
      Prod: sg-45678abc
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      InstanceType: !FindInMap [RegionAndEnvironmentToInstanceType, !Ref
 'AWS::Region', !Ref EnvironmentType]
      SecurityGroupIds:
        - !FindInMap [RegionAndEnvironmentToSecurityGroup, !Ref 'AWS::Region', !Ref
 EnvironmentType]
```

相关资源

在开发使用 Fn::FindInMap 函数的模板时，这些相关主题可能对您有所帮助。

- Fn::FindInMap
- Fn::FindInMap 增强功能
- Fn::Sub

CloudFormation 模板 Metadata 语法

Metadata 使用 JSON 或 YAML 对象存储其他信息。您可以在模板中使用的模板级元数据类型包括：

自定义元数据

存储用户定义的键值对。例如，您可以提供其他信息，这些信息不影响资源创建，但可以提供有关基础设施、团队或部署细节的其他背景信息。

`AWS::CloudFormation::Interface`

定义在 CloudFormation 控制台中显示输入参数时的分组和排序。默认情况下，CloudFormation 控制台按照参数的逻辑 ID 的字母顺序排序。

`AWS::CloudFormation::Designer`

Amazon CloudFormation Designer (Designer) 已于 2025 年 2 月 5 日终止使用。

> **Important**
>
> 堆栈更新期间，您无法更新 `Metadata` 部分本身。您只能在包括添加、修改或删除资源的更改时更新它。
>
> CloudFormation 不会转换、修改或编辑您在 `Metadata` 部分中包含的任何信息。因此，我们强烈建议您不要使用此部分存储敏感信息，例如密码或密钥。

语法

要在 CloudFormation 模板中声明自定义元数据，请使用以下语法：

JSON

```
"Metadata" : {
  "Instances" : {"Description" : "Information about the instances"},
  "Databases" : {"Description" : "Information about the databases"}
}
```

YAML

```
Metadata:
  Instances:
    Description: "Information about the instances"
  Databases:
```

```
    Description: "Information about the databases"
```

有关 AWS::CloudFormation::Interface 的语法，请参阅 使用 AWS::CloudFormation::Interface 元数据整理 CloudFormation 参数。

使用 AWS::CloudFormation::Interface 元数据整理 CloudFormation 参数

AWS::CloudFormation::Interface 是一种元数据键，用于定义参数在 CloudFormation 控制台中的分组和排序方式。在控制台中创建或更新堆栈时，默认情况下控制台会依照输入参数的逻辑 ID 的字母顺序将其列出。您可以借助此键定义自己的参数分组和排序方法，以便用户高效地指定参数值。例如，您可以将所有 EC2 相关的参数放到一个组中、将所有 VPC 相关的参数放到另一个组中。

在元数据键中，您可以指定要创建的组、每个组包含的参数以及控制台在其分组中显示各个参数时的顺序。

您还可以定义参数标签。标签是控制台显示的易记名称或描述，以代替参数的逻辑 ID。标签可帮助用户了解为每个参数指定的值。例如，您可以将 KeyPair 参数标记为 Select an EC2 key pair。

在元数据键中引用的所有参数都必须在模板的 Parameters 部分中声明。

> **Note**
>
> 仅 CloudFormation 控制台使用 AWS::CloudFormation::Interface 元数据键。Amazon CLI 和 API 调用不使用此键。

语法

要在 CloudFormation 模板中声明此实体，请使用以下语法：

JSON

```
"Metadata" : {
  "AWS::CloudFormation::Interface" : {
    "ParameterGroups": [
      {
        "Label": {
          "default": "Group Label"
        },
        "Parameters": [
```

```
          "Parameter1",
          "Parameter2"
        ]
      }
    ],
    "ParameterLabels": {
      "Parameter1": {
        "default": "Friendly Name for Parameter1"
      }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: Group Label
        Parameters:
          - Parameter1
          - Parameter2
    ParameterLabels:
      Parameter1:
        default: Friendly Name for Parameter1
```

属性

`ParameterGroups`

参数组类型的列表、指定组名称的地方、每个组中的参数以及参数的显示顺序。

必需：否

`Label`

参数组的名称。

必需：否

`default`

CloudFormation 控制台用于命名参数组的默认标签。

必需：否

类型：字符串

Parameters

要包含在组中的参数逻辑 ID (区分大小写) 的列表。参数必须已在模板的 Parameters 部分中定义。参数只能包含在一个参数组中。

控制台在 Other parameters 组中按字母顺序列出未与参数组关联的参数。

必需：否

类型：字符串值列表

ParameterLabels

创建或更新堆栈时，CloudFormation 控制台显示的参数与易记名称的映射。

必需：否

参数标签

参数的标签。标签定义在创建或更新堆栈时 CloudFormation 控制台在 Specify Parameters 页面上显示的易记名称或描述。参数标签必须为已在模板的 Parameters 部分中声明的有效参数的逻辑 ID（区分大小写）。

必需：否

default

CloudFormation 控制台用于命名参数的默认标签。

必需：否

类型：字符串

示例

以下示例定义了两个参数组：Network Configuration 和 Amazon EC2 Configuration。Network Configuration 组包含 VPCID、SubnetId 和 SecurityGroupID 参数 (在模板的 Parameters 部分中定义，未显示)。参数的列出顺序定义了控制台显示这些参数的顺序，以 VPCID 参数开始。以下示例以相似方式分组和排序 Amazon EC2 Configuration 参数。

该示例还为 VPCID 参数定义了一个标签。控制台将显示应将此元素部署到哪个 VPC？，而不是参数的逻辑 ID (VPCID)。

JSON

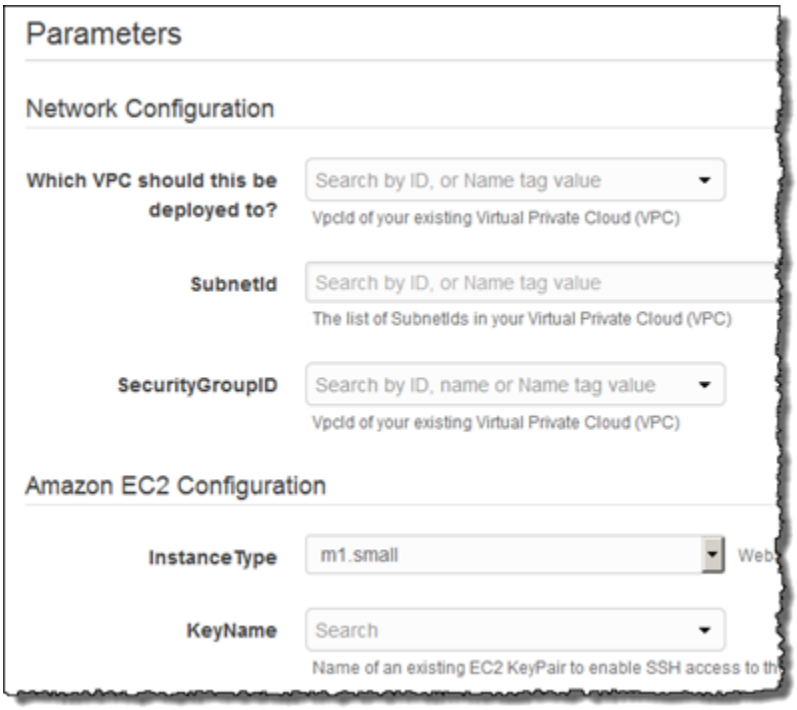
```
"Metadata" : {
  "AWS::CloudFormation::Interface" : {
    "ParameterGroups" : [
      {
        "Label" : { "default" : "Network Configuration" },
        "Parameters" : [ "VPCID", "SubnetId", "SecurityGroupID" ]
      },
      {
        "Label" : { "default":"Amazon EC2 Configuration" },
        "Parameters" : [ "InstanceType", "KeyName" ]
      }
    ],
    "ParameterLabels" : {
      "VPCID" : { "default" : "Which VPC should this be deployed to?" }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: "Network Configuration"
        Parameters:
          - VPCID
          - SubnetId
          - SecurityGroupID
      - Label:
          default: "Amazon EC2 Configuration"
        Parameters:
          - InstanceType
          - KeyName
    ParameterLabels:
      VPCID:
        default: "Which VPC should this be deployed to?"
```

控制台中的参数组

下图展示了使用本示例中的元数据键后，在创建或更新堆栈时控制台如何显示参数组：控制台中的参数组



CloudFormation 模板 Rules 语法

Rules 部分是 CloudFormation 模板的可选部分，可启用自定义验证逻辑。如果包含，此部分包含在 CloudFormation 创建或更新任何资源之前验证参数值的规则函数。

当标准参数约束不足时，规则非常有用。例如，如果启用了 SSL，则必须同时提供证书和域名。规则可以确保满足这些依赖关系。

语法

Rules 部分使用以下语法：

JSON

模板的Rules 部分由后跟冒号的密钥名称 Rules 组成。必须使用括号将所有规则声明括起来。如果您声明多个规则，则可用逗号将它们分隔开。对于每个规则，您必须声明一个用引号引起来的逻辑名称，后跟冒号以及将规则条件和断言括起来的括号。

```
{
```

```
    "Rules": {
        "LogicalRuleName1": {
            "RuleCondition": {
                "rule-specific intrinsic function": "Value"
            },
            "Assertions": [
                {
                    "Assert": {
                        "rule-specific intrinsic function": "Value"
                    },
                    "AssertDescription": "Information about this assert"
                },
                {
                    "Assert": {
                        "rule-specific intrinsic function": "Value"
                    },
                    "AssertDescription": "Information about this assert"
                }
            ]
        },
        "LogicalRuleName2": {
            "Assertions": [
                {
                    "Assert": {
                        "rule-specific intrinsic function": "Value"
                    },
                    "AssertDescription": "Information about this assert"
                }
            ]
        }
    }
}
```

YAML

```
Rules:
  LogicalRuleName1:
    RuleCondition:
      rule-specific intrinsic function: Value
    Assertions:
      - Assert:
          rule-specific intrinsic function: Value
        AssertDescription: Information about this assert
```

```
      - Assert:
          rule-specific intrinsic function: Value
        AssertDescription: Information about this assert
  LogicalRuleName2:
    Assertions:
      - Assert:
          rule-specific intrinsic function: Value
        AssertDescription: Information about this assert
```

规则字段

Rules 部分包含以下字段。

逻辑 ID（也称为逻辑名称）

每条规则的唯一标识符。

RuleCondition（可选）

确定规则生效时间的属性。如果您未定义规则条件，则规则的断言始终生效。对于每个规则，您只能定义一个规则条件。

Assertions（必需）

指定特定参数可接受值的一条或多条语句。

Assert

必须计算为 true 的条件。

AssertDescription

断言失败时显示的消息。

特定于规则的内部函数

要定义规则，您必须使用特定于规则的函数，只能在模板的 Rules 部分中使用这些函数。虽然可以嵌套这些函数，但规则条件或断言的最终结果必须为 true 或 false。

以下规则函数可用：

- Fn::And
- Fn::Contains
- Fn::EachMemberEquals

- Fn::EachMemberIn
- Fn::Equals
- Fn::If
- Fn::Not
- Fn::Or
- Fn::RefAll
- Fn::ValueOf
- Fn::ValueOfAll

这些函数在规则的条件或断言中使用。条件属性确定 CloudFormation 是否应用断言。如果条件的计算结果为 `true`，则 CloudFormation 将评估断言以验证在创建或更新预配置产品时参数值是否有效。如果参数值无效，则 CloudFormation 不会创建或更新堆栈。如果条件的计算结果为 `false`，则 CloudFormation 不会检查参数值并继续堆栈操作。

示例

主题

- 按条件验证参数值
- 交叉参数验证

按条件验证参数值

在以下示例中，两个规则检查 `InstanceType` 参数的值。根据环境参数（`test` 或 `prod`）的值，用户必须为 `InstanceType` 参数指定 `t3.medium` 或 `t3.large`。必须在同一模板的 `InstanceType` 部分中声明 `Environment` 和 `Parameters` 参数。

JSON

```
{
  "Rules": {
    "testInstanceType": {
      "RuleCondition": {
        "Fn::Equals": [
          {"Ref": "Environment"},
          "test"
        ]
      },
```

```
      "Assertions": [
        {
          "Assert": {
            "Fn::Contains": [
              ["t3.medium"],
              {"Ref": "InstanceType"}
            ]
          },
          "AssertDescription": "For a test environment, the instance type must be
 t3.medium"
        }
      ]
    },
    "prodInstanceType": {
      "RuleCondition": {
        "Fn::Equals": [
          {"Ref": "Environment"},
          "prod"
        ]
      },
      "Assertions": [
        {
          "Assert": {
            "Fn::Contains": [
              ["t3.large"],
              {"Ref": "InstanceType"}
            ]
          },
          "AssertDescription": "For a production environment, the instance type must be
 t3.large"
        }
      ]
    }
  }
}
```

YAML

```
Rules:
  testInstanceType:
    RuleCondition: !Equals
      - !Ref Environment
      - test
```

```
    Assertions:
      - Assert:
          'Fn::Contains':
            - - t3.medium
            - !Ref InstanceType
        AssertDescription: 'For a test environment, the instance type must be
 t3.medium'
  prodInstanceType:
    RuleCondition: !Equals
      - !Ref Environment
      - prod
    Assertions:
      - Assert:
          'Fn::Contains':
            - - t3.large
            - !Ref InstanceType
        AssertDescription: 'For a production environment, the instance type must be
 t3.large'
```

交叉参数验证

以下示例模板演示如何使用规则进行跨参数验证。其创建了一个在负载均衡器背后自动扩缩组上运行的示例网站。根据输入参数，该网站可在端口 80 或 443 上使用。可以将自动扩缩组中的实例配置为在任何端口上侦听（默认为 8888）。

此模板中的规则会在创建堆栈之前验证输入参数。其验证所有子网是否属于指定的 VPC，并确保将 UseSSL 参数设置为 Yes 时，同时提供 SSL 证书 ARN 和托管区域名称。

> **Note**
>
> 如果您通过本模板创建堆栈，则需为使用的 Amazon 资源支付相应费用。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "VpcId": {
      "Type": "AWS::EC2::VPC::Id",
      "Description": "VpcId of your existing Virtual Private Cloud (VPC)",
```

```
      "ConstraintDescription": "must be the VPC Id of an existing Virtual Private
Cloud."
    },
    "Subnets": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "The list of SubnetIds in your Virtual Private Cloud (VPC)",
      "ConstraintDescription": "must be a list of at least two existing subnets
associated with at least two different availability zones."
    },
    "InstanceType": {
      "Description": "WebServer EC2 instance type",
      "Type": "String",
      "Default": "t2.micro",
      "AllowedValues": ["t2.micro", "t3.micro"],
      "ConstraintDescription": "must be a valid EC2 instance type."
    },
    "KeyName": {
      "Description": "Name of an existing EC2 KeyPair to enable SSH access to the
instances",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
    },
    "SSHLocation": {
      "Description": "The IP address range that can be used to SSH to the EC2
instances",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "UseSSL": {
      "AllowedValues": ["Yes", "No"],
      "Default": "No",
      "Description": "Select \"Yes\" to implement SSL, \"No\" to skip (default).",
      "Type": "String"
    },
    "ALBSSLCertificateARN": {
      "Default": "",
      "Description": "[Optional] The ARN of the SSL certificate to be used for the
Application Load Balancer",
      "Type": "String"
    },
```

```
    "HostedZoneName": {
      "AllowedPattern": "^$|(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\\-]*[a-zA-Z0-9])\
\.)*([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9])$",
      "Default": "",
      "Description": "[Optional] The domain name of a valid Hosted Zone on AWS.",
      "Type": "String"
    }
  },
  "Conditions": {
    "UseALBSSL": {"Fn::Equals": [{"Ref": "UseSSL"}, "Yes"]}
  },
  "Rules": {
    "SubnetsInVPC": {
      "Assertions": [
        {
          "Assert": {"Fn::EachMemberEquals": [{"Fn::ValueOf": ["Subnets", "VpcId"]},
 {"Ref": "VpcId"}]},
          "AssertDescription": "All subnets must be in the VPC"
        }
      ]
    },
    "ValidateHostedZone": {
      "RuleCondition": {"Fn::Equals": [{"Ref": "UseSSL"}, "Yes"]},
      "Assertions": [
        {
          "Assert": {"Fn::Not": [{"Fn::Equals": [{"Ref": "ALBSSLCertificateARN"},
 ""]}]},
          "AssertDescription": "ACM Certificate value cannot be empty if SSL is
 required"
        },
        {
          "Assert": {"Fn::Not": [{"Fn::Equals": [{"Ref": "HostedZoneName"}, ""]}]},
          "AssertDescription": "Route53 Hosted Zone Name is mandatory when SSL is
 required"
        }
      ]
    }
  },
  "Resources": {
    "WebServerGroup": {
      "Type": "AWS::AutoScaling::AutoScalingGroup",
      "Properties": {
        "VPCZoneIdentifier": {"Ref": "Subnets"},
        "LaunchTemplate": {
```

```
          "LaunchTemplateId": {"Ref": "LaunchTemplate"},
          "Version": {"Fn::GetAtt": ["LaunchTemplate","LatestVersionNumber"]}
        },
        "MinSize": "2",
        "MaxSize": "2",
        "TargetGroupARNs": [{"Ref": "ALBTargetGroup"}]
      },
      "CreationPolicy": {
        "ResourceSignal": {"Timeout": "PT15M"}
      },
      "UpdatePolicy": {
        "AutoScalingRollingUpdate": {
          "MinInstancesInService": "1",
          "MaxBatchSize": "1",
          "PauseTime": "PT15M",
          "WaitOnResourceSignals": true
        }
      }
    },
    "LaunchTemplate": {
      "Type": "AWS::EC2::LaunchTemplate",
      "Metadata": {
        "Comment": "Install a simple application",
        "AWS::CloudFormation::Init": {
          "config": {
            "packages": {"yum": {"httpd": []}},
            "files": {
              "/var/www/html/index.html": {
                "content": {"Fn::Join": ["\n", ["<h1>Congratulations, you have
successfully launched the AWS CloudFormation sample.</h1>"]]},
                "mode": "000644",
                "owner": "root",
                "group": "root"
              },
              "/etc/cfn/cfn-hup.conf": {
                "content": {"Fn::Join": ["", [
                  "[main]\n",
                  "stack=", {"Ref": "AWS::StackId"}, "\n",
                  "region=", {"Ref": "AWS::Region"}, "\n"
                ]]},
                "mode": "000400",
                "owner": "root",
                "group": "root"
              },
```

```
              "/etc/cfn/hooks.d/cfn-auto-reloader.conf": {
                "content": {"Fn::Join": ["", [
                  "[cfn-auto-reloader-hook]\n",
                  "triggers=post.update\n",
                  "path=Resources.LaunchTemplate.Metadata.AWS::CloudFormation::Init\n",
                  "action=/opt/aws/bin/cfn-init -v ",
                  "         --stack ", {"Ref": "AWS::StackName"},
                  "         --resource LaunchTemplate ",
                  "         --region ", {"Ref": "AWS::Region"}, "\n",
                  "runas=root\n"
                ]]},
                "mode": "000400",
                "owner": "root",
                "group": "root"
              }
            },
            "services": {
              "sysvinit": {
                "httpd": {
                  "enabled": "true",
                  "ensureRunning": "true"
                },
                "cfn-hup": {
                  "enabled": "true",
                  "ensureRunning": "true",
                  "files": [
                    "/etc/cfn/cfn-hup.conf",
                    "/etc/cfn/hooks.d/cfn-auto-reloader.conf"
                  ]
                }
              }
            }
          }
        },
        "Properties": {
          "LaunchTemplateName": {"Fn::Sub": "${AWS::StackName}-launch-template"},
          "LaunchTemplateData": {
            "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
            "SecurityGroupIds": [{"Ref": "InstanceSecurityGroup"}],
            "InstanceType": {"Ref": "InstanceType"},
            "KeyName": {"Ref": "KeyName"},
            "UserData": {
```

```
            "Fn::Base64": {"Fn::Join": ["", [
              "#!/bin/bash\n",
              "yum install -y aws-cfn-bootstrap\n",
              "/opt/aws/bin/cfn-init -v ",
              "         --stack ", {"Ref": "AWS::StackName"},
              "         --resource LaunchTemplate ",
              "         --region ", {"Ref": "AWS::Region"}, "\n",
              "/opt/aws/bin/cfn-signal -e $? ",
              "         --stack ", {"Ref": "AWS::StackName"},
              "         --resource WebServerGroup ",
              "         --region ", {"Ref": "AWS::Region"}, "\n"
            ]]}
          }
        }
      }
    },
    "ELBSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Allow access to the ELB",
        "VpcId": {"Ref": "VpcId"},
        "SecurityGroupIngress": [{
          "Fn::If": [
            "UseALBSSL",
            {
              "IpProtocol": "tcp",
              "FromPort": 443,
              "ToPort": 443,
              "CidrIp": "0.0.0.0/0"
            },
            {
              "IpProtocol": "tcp",
              "FromPort": 80,
              "ToPort": 80,
              "CidrIp": "0.0.0.0/0"
            }
          ]
        }]
      }
    },
    "ApplicationLoadBalancer": {
      "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties": {
        "Subnets": {"Ref": "Subnets"},
```

```
        "SecurityGroups": [{"Ref": "ELBSecurityGroup"}]
      }
    },
    "ALBListener": {
      "Type": "AWS::ElasticLoadBalancingV2::Listener",
      "Properties": {
        "DefaultActions": [{
          "Type": "forward",
          "TargetGroupArn": {"Ref": "ALBTargetGroup"}
        }],
        "LoadBalancerArn": {"Ref": "ApplicationLoadBalancer"},
        "Port": {"Fn::If": ["UseALBSSL", 443, 80]},
        "Protocol": {"Fn::If": ["UseALBSSL", "HTTPS", "HTTP"]},
        "Certificates": [{
          "Fn::If": [
            "UseALBSSL",
            {"CertificateArn": {"Ref": "ALBSSLCertificateARN"}},
            {"Ref": "AWS::NoValue"}
          ]
        }]
      }
    },
    "ALBTargetGroup": {
      "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties": {
        "HealthCheckIntervalSeconds": 30,
        "HealthCheckTimeoutSeconds": 5,
        "HealthyThresholdCount": 3,
        "Port": 80,
        "Protocol": "HTTP",
        "UnhealthyThresholdCount": 5,
        "VpcId": {"Ref": "VpcId"}
      }
    },
    "InstanceSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Enable SSH access and HTTP access on the inbound port",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
```

```
            "SourceSecurityGroupId": {"Fn::Select": [0, {"Fn::GetAtt":
["ApplicationLoadBalancer", "SecurityGroups"]}]}
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIp": {"Ref": "SSHLocation"}
          }
        ],
        "VpcId": {"Ref": "VpcId"}
      }
    },
    "RecordSet": {
      "Type": "AWS::Route53::RecordSetGroup",
      "Condition": "UseALBSSL",
      "Properties": {
        "HostedZoneName": {"Fn::Join": ["", [{"Ref": "HostedZoneName"}, "."]]},
        "RecordSets": [{
          "Name": {"Fn::Join": ["", [
            {"Fn::Select": ["0", {"Fn::Split": [".", {"Fn::GetAtt":
["ApplicationLoadBalancer", "DNSName"]}]}]},
            ".",
            {"Ref": "HostedZoneName"},
            "."
          ]]},
          "Type": "A",
          "AliasTarget": {
            "DNSName": {"Fn::GetAtt": ["ApplicationLoadBalancer", "DNSName"]},
            "EvaluateTargetHealth": true,
            "HostedZoneId": {"Fn::GetAtt": ["ApplicationLoadBalancer",
"CanonicalHostedZoneID"]}
          }
        }]
      }
    }
  },
  "Outputs": {
    "URL": {
      "Description": "URL of the website",
      "Value": {"Fn::Join": ["", [
        {"Fn::If": [
          "UseALBSSL",
          {"Fn::Join": ["", [
```

```
            "https://",
            {"Fn::Join": ["", [
              {"Fn::Select": ["0", {"Fn::Split": [".", {"Fn::GetAtt":
 ["ApplicationLoadBalancer", "DNSName"]}]}]},
              ".",
              {"Ref": "HostedZoneName"},
              "."
            ]]}
          ]]},
          {"Fn::Join": ["", [
            "http://",
            {"Fn::GetAtt": ["ApplicationLoadBalancer", "DNSName"]}
          ]]}
        ]}
      ]]}
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: VpcId of your existing Virtual Private Cloud (VPC)
    ConstraintDescription: must be the VPC Id of an existing Virtual Private Cloud.
  Subnets:
    Type: List<AWS::EC2::Subnet::Id>
    Description: The list of SubnetIds in your Virtual Private Cloud (VPC)
    ConstraintDescription: >-
      must be a list of at least two existing subnets associated with at least
      two different availability zones. They should be residing in the selected
      Virtual Private Cloud.
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t3.micro
    ConstraintDescription: must be a valid EC2 instance type.
  KeyName:
```

```
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  UseSSL:
    AllowedValues:
      - 'Yes'
      - 'No'
    ConstraintDescription: Select Yes to create a HTTPS Listener
    Default: 'No'
    Description: 'Select "Yes" to implement SSL, "No" to skip (default).'
    Type: String
  ALBSSLCertificateARN:
    Default: ''
    Description: >-
      [Optional] The ARN of the SSL certificate to be used for the Application
      Load Balancer
    Type: String
  HostedZoneName:
    AllowedPattern: >-
      ^$|(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-]*[a-zA-Z0-9])\.)*([A-Za-z0-9]|[A-Za-
z0-9][A-Za-z0-9\-]*[A-Za-z0-9])$
    Default: ''
    Description: '[Optional] The domain name of a valid Hosted Zone on AWS.'
    Type: String
Conditions:
  UseALBSSL: !Equals
    - !Ref UseSSL
    - 'Yes'
Rules:
  SubnetsInVPC:
    Assertions:
      - Assert:
          'Fn::EachMemberEquals':
            - 'Fn::ValueOf':
                - Subnets
                - VpcId
```

```
            - Ref: VpcId
        AssertDescription: All subnets must be in the VPC
  ValidateHostedZone:
    RuleCondition: !Equals
      - !Ref UseSSL
      - 'Yes'
    Assertions:
      - Assert: !Not
          - !Equals
            - !Ref ALBSSLCertificateARN
            - ''
        AssertDescription: ACM Certificate value cannot be empty if SSL is required
      - Assert: !Not
          - !Equals
            - !Ref HostedZoneName
            - ''
        AssertDescription: Route53 Hosted Zone Name is mandatory when SSL is required
Resources:
  WebServerGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier: !Ref Subnets
      LaunchTemplate:
        LaunchTemplateId: !Ref LaunchTemplate
        Version: !GetAtt LaunchTemplate.LatestVersionNumber
      MinSize: '2'
      MaxSize: '2'
      TargetGroupARNs:
        - !Ref ALBTargetGroup
    CreationPolicy:
      ResourceSignal:
        Timeout: PT15M
    UpdatePolicy:
      AutoScalingRollingUpdate:
        MinInstancesInService: '1'
        MaxBatchSize: '1'
        PauseTime: PT15M
        WaitOnResourceSignals: 'true'
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Metadata:
      Comment: Install a simple application
      AWS::CloudFormation::Init:
        config:
```

```
          packages:
            yum:
              httpd: []
          files:
            /var/www/html/index.html:
              content: !Join
                - |+
                - - >-
                    <h1>Congratulations, you have successfully launched the AWS
                    CloudFormation sample.</h1>
              mode: '000644'
              owner: root
              group: root
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |-
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.LaunchTemplate.Metadata.AWS::CloudFormation::Init
                action=/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
LaunchTemplate --region ${AWS::Region}
                runas=root
              mode: '000400'
              owner: root
              group: root
          services:
            sysvinit:
              httpd:
                enabled: 'true'
                ensureRunning: 'true'
              cfn-hup:
                enabled: 'true'
                ensureRunning: 'true'
                files:
                  - /etc/cfn/cfn-hup.conf
                  - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    Properties:
```

```
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
        SecurityGroupIds:
        - !Ref InstanceSecurityGroup
        InstanceType: !Ref InstanceType
        KeyName: !Ref KeyName
        UserData: !Base64
          Fn::Sub: |
            #!/bin/bash
            yum install -y aws-cfn-bootstrap
            /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 LaunchTemplate --region ${AWS::Region}
            /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerGroup --region ${AWS::Region}
  ELBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow access to the ELB
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - !If
          - UseALBSSL
          - IpProtocol: tcp
            FromPort: 443
            ToPort: 443
            CidrIp: 0.0.0.0/0
          - IpProtocol: tcp
            FromPort: 80
            ToPort: 80
            CidrIp: 0.0.0.0/0
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Subnets: !Ref Subnets
      SecurityGroups:
        - !Ref ELBSecurityGroup
  ALBListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref ALBTargetGroup
```

```
      LoadBalancerArn: !Ref ApplicationLoadBalancer
      Port: !If
        - UseALBSSL
        - 443
        - 80
      Protocol: !If
        - UseALBSSL
        - HTTPS
        - HTTP
      Certificates:
        - !If
          - UseALBSSL
          - CertificateArn: !Ref ALBSSLCertificateARN
          - !Ref 'AWS::NoValue'
  ALBTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 30
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 3
      Port: 80
      Protocol: HTTP
      UnhealthyThresholdCount: 5
      VpcId: !Ref VpcId
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH access and HTTP access on the inbound port
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          SourceSecurityGroupId: !Select
            - 0
            - !GetAtt
              - ApplicationLoadBalancer
              - SecurityGroups
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref SSHLocation
      VpcId: !Ref VpcId
  RecordSet:
    Type: AWS::Route53::RecordSetGroup
```

```
    Condition: UseALBSSL
    Properties:
      HostedZoneName: !Join
        - ''
        - - !Ref HostedZoneName
          - .
      RecordSets:
        - Name: !Join
            - ''
            - - !Select
                - '0'
                - !Split
                  - .
                  - !GetAtt
                    - ApplicationLoadBalancer
                    - DNSName
              - .
              - !Ref HostedZoneName
              - .
          Type: A
          AliasTarget:
            DNSName: !GetAtt
              - ApplicationLoadBalancer
              - DNSName
            EvaluateTargetHealth: true
            HostedZoneId: !GetAtt
              - ApplicationLoadBalancer
              - CanonicalHostedZoneID
Outputs:
  URL:
    Description: URL of the website
    Value: !Join
      - ''
      - - !If
          - UseALBSSL
          - !Join
            - ''
            - - 'https://'
              - !Join
                - ''
                - - !Select
                    - '0'
                    - !Split
                      - .
```

```
                    - !GetAtt
                      - ApplicationLoadBalancer
                      - DNSName
                - .
                - !Ref HostedZoneName
                - .
          - !Join
            - ''
            - - 'http://'
              - !GetAtt
                - ApplicationLoadBalancer
                - DNSName
```

CloudFormation 模板 Conditions 语法

可选的 Conditions 部分包含一些声明，以定义在哪些情况下创建或配置实体。例如，您可以创建一个条件，然后将其与某个资源或输出关联，以便 CloudFormation 仅在该条件成立时创建该资源或输出。同样，您可以将条件与一个属性关联，以便 CloudFormation 仅在该条件为 true 时将该属性设置为特定的值。如果该条件为 false，则 CloudFormation 会将该属性设置为您指定的其他值。

当您需要重新使用模板在不同环境（如测试环境与生产环境）中创建资源时，可以使用条件。例如，在模板中，您可以添加 EnvironmentType 输入参数，它接受 prod 或 test 以作为输入。对于 prod 环境，您可以采用带特定功能的 EC2 实例；但对于 test 环境，您可以使用更少功能，以便节约资金。通过定义条件，您可以定义为每个环境类型创建哪些资源以及如何配置资源。

语法

Conditions 部分包括键名称 Conditions。每个条件声明均包括一个逻辑 ID 和多个内置函数。

JSON

```
"Conditions": {
  "LogicalConditionName1": {
    "Intrinsic function": ...[
  },

  "LogicalConditionName2": {
    "Intrinsic function": ...
  }
}
```

YAML

```
Conditions:
  LogicalConditionName1:
    Intrinsic function:
      ...

  LogicalConditionName2:
    Intrinsic function:
      ...
```

条件的工作原理

要使用条件，请按照下列步骤操作：

1. 添加参数定义：定义条件在模板 `Parameters` 部分中进行评估的输入。根据这些输入参数的值，条件评估结果为 true 或 false。请注意，伪参数是自动可用的，不需要在 `Parameters` 部分中进行明确定义。有关伪参数的更多信息，请参阅使用伪参数获取 Amazon 值。
2. 添加条件定义：使用内置函数（例如 `Fn::If` 或 `Fn::Equals`）在 `Conditions` 部分中定义条件。这些条件将决定 CloudFormation 何时创建关联的资源。条件可以基于：
   - 输入或伪参数值
   - 其他条件
   - 映射值

   但是，您不能在条件中引用资源逻辑 ID 或其属性。
3. 将条件与资源或输出关联：使用 `Condition` 键和条件的逻辑 ID 在资源或输出中引用条件。或者，也可以在模板的其他部分（例如属性值）中使用 `Fn::If` 以根据条件设置值。有关更多信息，请参阅 使用 `Condition` 密钥。

在创建或更新堆栈时，CloudFormation 会评估条件。CloudFormation 创建与成立的条件关联的实体，并忽略与不成立的条件关联的实体。在每个堆栈更新期间，CloudFormation 还会在修改任何资源之前重新评估这些条件。与 true 条件关联的实体将会更新，与 false 条件关联的实体将会删除。

> **Important**
>
> 堆栈更新期间，您无法更新条件本身。您只能在包含添加、修改或删除资源的更改时更新条件。

条件内部函数

您可以使用以下内部函数定义条件：

- Fn::And
- Fn::Equals
- Fn::ForEach
- Fn::If
- Fn::Not
- Fn::Or

> **Note**
>
> 仅在模板的 Resources 和 Outputs 部分中的元数据属性、更新策略属性和属性值中支持 Fn::If。

使用 **Condition** 密钥

定义条件后，您可以使用 Condition 密钥将其应用于模板中的多个位置，例如 Resources 和 Outputs。Condition 密钥引用条件的逻辑名称并返回指定条件的评估结果。

主题

- 将条件与资源关联
- 将条件与输出关联
- 其他条件中的引用条件
- 使用 Fn::If 有条件地返回属性值

将条件与资源关联

要有条件地创建资源，请添加 Condition 密钥，并将条件的逻辑 ID 作为属性添加到该资源中。仅当条件评估为 true 时，CloudFormation 才会创建 资源。

JSON

```
"NewVolume" : {
```

```
  "Type" : "AWS::EC2::Volume",
  "Condition" : "IsProduction",
  "Properties" : {
     "Size" : "100",
     "AvailabilityZone" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone" ]}
  }
}
```

YAML

```
NewVolume:
  Type: AWS::EC2::Volume
  Condition: IsProduction
  Properties:
    Size: 100
    AvailabilityZone: !GetAtt EC2Instance.AvailabilityZone
```

将条件与输出关联

您还可以将条件与输出关联。仅当关联条件的评估结果为 true 时，CloudFormation 才会创建输出。

JSON

```
"Outputs" : {
  "VolumeId" : {
    "Condition" : "IsProduction",
    "Value" : { "Ref" : "NewVolume" }
  }
}
```

YAML

```
Outputs:
  VolumeId:
    Condition: IsProduction
    Value: !Ref NewVolume
```

其他条件中的引用条件

在 Conditions 部分中定义条件时，可以使用 Condition 密钥引用其他条件。这使您能够通过组合多个条件来创建更复杂的条件逻辑。

在以下示例中，仅当 IsProduction 和 IsProdAndFeatureEnabled 条件的评估结果为 true 时，IsFeatureEnabled 条件才会评估为 true。

JSON

```
"Conditions": {
  "IsProduction" : {"Fn::Equals" : [{"Ref" : "Environment"}, "prod"]},
  "IsFeatureEnabled" : { "Fn::Equals" : [{"Ref" : "FeatureFlag"}, "enabled"]},
  "IsProdAndFeatureEnabled" : {
    "Fn::And" : [
      {"Condition" : "IsProduction"},
      {"Condition" : "IsFeatureEnabled"}
    ]
  }
}
```

YAML

```
Conditions:
  IsProduction:
    !Equals [!Ref Environment, "prod"]
  IsFeatureEnabled:
    !Equals [!Ref FeatureFlag, "enabled"]
  IsProdAndFeatureEnabled: !And
    - !Condition IsProduction
    - !Condition IsFeatureEnabled
```

使用 **Fn::If** 有条件地返回属性值

要进行更精细的控制，您可以使用 Fn::If 内置函数有条件地返回资源或输出中的两个属性值之一。该函数会对条件进行评估，如果条件为 true，则返回一个值；如果条件为 false，则返回另一个值。

条件属性值

以下示例演示了如何根据环境条件设置 EC2 实例类型。如果 IsProduction 条件的评估结果为 true，则实例类型设置为 c5.xlarge。否则，该值将设置为 t3.small。

JSON

```
"Properties" : {
  "InstanceType" : {
```

```
    "Fn::If" : [
      "IsProduction",
      "c5.xlarge",
      "t3.small"
    ]
  }
}
```

YAML

```
Properties:
  InstanceType: !If
    - IsProduction
    - c5.xlarge
    - t3.small
```

条件属性移除

您也可以使用 AWS::NoValue 伪参数名称】作为返回值，在条件为 false 时移除对应的属性。

JSON

```
"DBSnapshotIdentifier" : {
  "Fn::If" : [
    "UseDBSnapshot",
    {"Ref" : "DBSnapshotName"},
    {"Ref" : "AWS::NoValue"}
  ]
}
```

YAML

```
DBSnapshotIdentifier: !If
  - UseDBSnapshot
  - !Ref DBSnapshotName
  - !Ref "AWS::NoValue"
```

示例

主题

- 基于环境创建资源

- 多条件资源预置

基于环境创建资源

以下示例将预置一个 EC2 实例，并且仅当环境类型为 prod 时，才会有条件地创建和附加新的 EBS 卷。如果环境是 test，示例则只会创建不带额外卷的 EC2 实例。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "EnvType": {
            "Description": "Environment type",
            "Default": "test",
            "Type": "String",
            "AllowedValues": [
                "prod",
                "test"
            ],
            "ConstraintDescription": "must specify prod or test"
        }
    },
    "Conditions": {
        "IsProduction": {
            "Fn::Equals": [
                {
                    "Ref": "EnvType"
                },
                "prod"
            ]
        }
    },
    "Resources": {
        "EC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-1234567890abcdef0",
                "InstanceType": "c5.xlarge"
            }
        },
        "MountPoint": {
            "Type": "AWS::EC2::VolumeAttachment",
```

```
            "Condition": "IsProduction",
            "Properties": {
                "InstanceId": {
                    "Ref": "EC2Instance"
                },
                "VolumeId": {
                    "Ref": "NewVolume"
                },
                "Device": "/dev/sdh"
            }
        },
        "NewVolume": {
            "Type": "AWS::EC2::Volume",
            "Condition": "IsProduction",
            "Properties": {
                "Size": 100,
                "AvailabilityZone": {
                    "Fn::GetAtt": [
                        "EC2Instance",
                        "AvailabilityZone"
                    ]
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvType:
    Description: Environment type
    Default: test
    Type: String
    AllowedValues:
      - prod
      - test
    ConstraintDescription: must specify prod or test
Conditions:
  IsProduction: !Equals
    - !Ref EnvType
    - prod
```

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
      InstanceType: c5.xlarge
  MountPoint:
    Type: AWS::EC2::VolumeAttachment
    Condition: IsProduction
    Properties:
      InstanceId: !Ref EC2Instance
      VolumeId: !Ref NewVolume
      Device: /dev/sdh
  NewVolume:
    Type: AWS::EC2::Volume
    Condition: IsProduction
    Properties:
      Size: 100
      AvailabilityZone: !GetAtt
        - EC2Instance
        - AvailabilityZone
```

多条件资源预置

如果提供了存储桶名称，则以下示例会有条件地创建 S3 存储桶，并且仅当环境设置为 `prod` 时，才会附加存储桶策略。如果未提供存储桶名称或环境为 `test`，则不会创建任何资源。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "EnvType": {
            "Type": "String",
            "AllowedValues": [
                "prod",
                "test"
            ]
        },
        "BucketName": {
            "Default": "",
            "Type": "String"
        }
```

```
    },
    "Conditions": {
        "IsProduction": {
            "Fn::Equals": [
                {
                    "Ref": "EnvType"
                },
                "prod"
            ]
        },
        "CreateBucket": {
            "Fn::Not": [
                {
                    "Fn::Equals": [
                        {
                            "Ref": "BucketName"
                        },
                        ""
                    ]
                }
            ]
        },
        "CreateBucketPolicy": {
            "Fn::And": [
                {
                    "Condition": "IsProduction"
                },
                {
                    "Condition": "CreateBucket"
                }
            ]
        }
    },
    "Resources": {
        "Bucket": {
            "Type": "AWS::S3::Bucket",
            "Condition": "CreateBucket",
            "Properties": {
                "BucketName": {
                    "Ref": "BucketName"
                }
            }
        },
        "Policy": {
```

```
            "Type": "AWS::S3::BucketPolicy",
            "Condition": "CreateBucketPolicy",
            "Properties": {
                "Bucket": {
                    "Ref": "Bucket"
                },
                "PolicyDocument": { ... }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvType:
    Type: String
    AllowedValues:
      - prod
      - test
  BucketName:
    Default: ''
    Type: String
Conditions:
  IsProduction: !Equals
    - !Ref EnvType
    - prod
  CreateBucket: !Not
    - !Equals
      - !Ref BucketName
      - ''
  CreateBucketPolicy: !And
    - !Condition IsProduction
    - !Condition CreateBucket
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Condition: CreateBucket
    Properties:
      BucketName: !Ref BucketName
  Policy:
    Type: AWS::S3::BucketPolicy
```

```
    Condition: CreateBucketPolicy
    Properties:
      Bucket: !Ref Bucket
      PolicyDocument: ...
```

在此示例中，CreateBucketPolicy 条件演示了如何使用 Condition 密钥引用其他条件。只有当 IsProduction 和 CreateBucket 条件的评估结果都为 true 时，才会创建策略。

> **Note**
>
> 有关使用条件的更复杂的示例，请参阅《Amazon CloudFormation模板参考指南》中的Condition 属性主题。

CloudFormation 模板 Transform 部分

可选的 Transform 部分指定 CloudFormation 用来以某种方式处理模板的一个或多个宏。

宏可以执行查找和替换文本等简单任务，也可以对整个模板进行更广泛的转换。CloudFormation 按照指定的顺序执行宏。当您创建更改集时，CloudFormation 会生成一个包含已处理模板内容的更改集。然后，您可以查看更改并执行更改集。有关宏工作原理的更多信息，请参阅使用模板宏对 CloudFormation 模板执行自定义处理。

CloudFormation 还支持转换，后者是由 CloudFormation 托管的宏。在执行顺序和范围方面，CloudFormation 以相同的方法处理转换以及您创建的任何宏。有关更多信息，请参阅转换参考。

要声明多个宏，请使用列表格式并指定一个或多个宏。

例如，在下面的模板示例中，CloudFormation 会首先评估 MyMacro，然后再评估 AWS::Serverless，并且两者都可以处理整个模板的内容，因为它们包含在 Transform 部分中。

```
# Start of processable content for MyMacro and AWS::Serverless
Transform:
  - MyMacro
  - 'AWS::Serverless'
Resources:
  WaitCondition:
    Type: AWS::CloudFormation::WaitCondition
  MyBucket:
    Type: AWS::S3::Bucket
    Properties:
```

```
      BucketName: amzn-s3-demo-bucket
      Tags: [{"key":"value"}]
      CorsConfiguration: []
  MyEc2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
# End of processable content for MyMacro and AWS::Serverless
```

CloudFormation 模板格式版本语法

AWSTemplateFormatVersion 部分（可选）用于标识模板要符合的模板格式版本。最新的模板格式版本是 2010-09-09，并且它是目前唯一的有效值。

模板格式版本与 API 版本不同。模板格式版本可独立于 API 版本进行更改。

模板格式版本声明的值必须是文字字符串。您无法使用参数或函数来指定模板格式版本。如果您没有指定任何值，则 CloudFormation 会假定为最新的模板格式版本。以下代码段是有效的模板格式版本声明的示例：

JSON

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
```

CloudFormation 模板 Description 语法

Description 部分（可选）可用来包含描述模板的文本字符串。此部分必须始终紧随模板格式版本部分之后。

描述声明的值必须是长度介于 0 和 1024 个字节之间的文字字符串。您无法使用参数或函数来指定描述。以下代码段是描述声明的示例：

> **Important**
>
> 堆栈更新期间，您无法更新 Description 部分本身。您只能在包括添加、修改或删除资源的更改时更新它。

JSON

```
"Description" : "Here are some details about the template."
```

YAML

```
Description: > Here are some details about the template.
```

CloudFormation 语言服务器

Amazon CloudFormation 语言服务器可帮助您在编辑器中编写 CloudFormation 模板。在您键入时，它会建议资源类型和属性、显示文档、查找错误、允许您跳转到定义，并针对某些问题提供修复建议。它使用语言服务器协议（LSP），使该服务器能够与不同的编辑器配合使用。启动和使用语言服务器来实现这些本地创作功能不需要 Amazon Web Services 账户 或 Amazon 凭证。

Amazon Toolkit 集成还使用语言服务器来处理调用 Amazon 服务的在线工作流。这些工作流可以浏览资源和堆栈。它们还可以运行服务端模板验证、部署模板、显示更改集差异以及监控堆栈事件。这些在线工作流是唯一需要活跃 Amazon Web Services 账户 和有效 Amazon 凭证的功能。

有关不同编辑器的发布说明和设置示例，请参阅 GitHub 上的 CloudFormation 语言服务器。

主题

- IDE 与 CloudFormation 语言服务器集成
- 开始使用
- 在 IDE 中初始化 CloudFormation 项目（仅限 VS Code）
- 需要帮助？

IDE 与 CloudFormation 语言服务器集成

Amazon 通过 Amazon 工具包为以下 IDE 提供与 CloudFormation 语言服务器的现成集成：

- Visual Studio Code
- JetBrains IDE（版本 2025.3 或更高版本），包括 IntelliJ IDEA、WebStorm 和 PyCharm

以下 IDE 还支持 CloudFormation 语言服务器：

- Kiro：
- Cursor
- 大多数 VS Code 分叉和分配

对于 Visual Studio Code 和 JetBrains IDE，请安装 Amazon Toolkit。Toolkit 会包括并启动语言服务器，因此您无需单独下载该服务器。

对于支持 LSP 的其他编辑器，请从 GitHub 上的最新版本下载适用于您计算机的文件，将其解压缩，然后配置编辑器以运行下面的命令：

```
node /path/to/install-location/cfn-lsp-server-standalone.js --stdio
```

有关支持的 Node.js 版本、可用下载、文件验证和不同编辑器的设置示例，请参阅独立安装指南。

开始使用

下面的步骤展示了如何通过 Amazon Toolkit for Visual Studio Code 和 JetBrains IDE 使用语言服务器。对于其他编辑器，请按照独立安装指南中的设置示例进行操作。

主题

- 先决条件
- 第 1 步：安装或升级 Amazon 工具包
- 第 2 步：在 Amazon 工具包中访问 CloudFormation
- 第 3 步：验证、测试和优化您的模板
- 第 4 步：浏览模板
- 第 5 步：验证和部署

先决条件

在您开始之前，请确保：

- 您正在 macOS、Windows 或 Linux 上使用支持的 IDE。
- 对于 Visual Studio Code 或 JetBrains IDE，请安装或更新 Amazon Toolkit。
- 对于其他编辑器，请安装支持的 Node.js 版本，并下载适用于计算机的语言服务器文件。独立安装指南列出了支持的版本和文件。

无需 Amazon Web Services 账户 或 Amazon 凭证即可启动语言服务器或使用其本地创作和模板检查功能。只有当您使用浏览资源或堆栈、调用 CloudFormation 服务或部署模板的在线工作流时，才需要在 Toolkit 中配置有效的 Amazon 凭证。

第 1 步：安装或升级 Amazon 工具包

从 IDE 的扩展程序或插件管理器中安装或更新到最新版本的 Amazon 工具包，然后重新启动 IDE。

安装后，Amazon 工具包会自动启用 CloudFormation IDE 支持。首次安装或升级带有 CloudFormation 语言服务器的 Amazon Toolkit 时，系统会提示您授予 Amazon 收集匿名使用数据的权限。这些数据有助于 Amazon 改进 CloudFormation 语言服务器并增强撰写体验。不会收集任何敏感信息，Amazon 也不会记录或存储模板内容、资源配置或任何可识别身份的客户数据。您可以随时从 IDE 设置中更改遥测首选项。重新启动 IDE 以使更改生效。所收集的使用数据仅涵盖功能交互和性能指标。这些见解有助于 Amazon 识别并确定需要改进方面优先顺序，例如加快验证速度、增强自动补全功能以及改进错误诊断机制。

第 2 步：在 Amazon 工具包中访问 CloudFormation

安装 Amazon 工具包后，在 IDE 中打开 CloudFormation 面板。在 VS Code 中，从活动栏打开 Amazon 工具包面板，然后选择 CLOUDFORMATION。在 JetBrains IDE 中，从侧栏打开 Amazon 工具包工具窗口，然后选择 CloudFormation 选项卡。

CloudFormation 面板包含以下部分：

- 区域：显示当前 Amazon Web Services 区域。在 VS Code 中，您可以通过选择区域名称或使用命令面板中的 Amazon CloudFormation: Select Region 命令来更改区域。在 JetBrains IDE 中，区域是通过 Amazon 工具包连接设置配置的。
- 堆栈：显示您的账户中 CloudFormation 堆栈的分页列表。展开堆栈，以便查看其更改集节点，其中列出了与该堆栈关联的更改集。使用“查看堆栈详细信息”操作打开堆栈详细信息视图，该视图显示堆栈概述、事件、输出和资源。
- 资源：添加资源类型后，面板将显示您账户中该类型的 Amazon 资源。您可以查看、刷新、复制或将它们导入到您的模板中。

在 JetBrains IDE 中，树上方的工具栏可快速访问常见操作，包括验证和部署、重新运行、验证和部署、添加资源类型和刷新。也可以通过右键单击树节点上的上下文菜单来执行操作。

第 3 步：验证、测试和优化您的模板

在您编写 CloudFormation 模板时，CloudFormation 语言服务器会在后台运行，并在无需 Amazon 凭证的情况下提供本地创作辅助：

- 代码完成：根据 CloudFormation 架构来建议资源类型、参数和属性。
- 提取到参数：当光标位于模板中的字面值（例如，类似 `t2.micro` 的字符串）上时，IDE 会提供重构操作，用于将该值提取到 `Parameters` 部分中，并将字面值替换为指向新参数的 `!Ref`。如果在多个位置出现相同的字面值，则可以选择一次提取所有出现的字面值。

Toolkit 还提供了一个在线工作流，用于向模板添加现有 Amazon 资源。此工作流需要有效的 Amazon 凭证，并使用 Amazon 云端控制 API（CCAPI）检索资源的实时配置和属性，从而帮助您在模板中克隆或重复使用现有基础设施。

要向模板添加资源

- 添加资源类型：在 Amazon 工具包 CloudFormation 面板的资源下，添加要浏览的资源类型。在 VS Code 中，单击添加 + 图标或使用命令面板中的 Amazon CloudFormation: Add Resource Types 命令。在 JetBrains 中，单击工具栏中的添加资源类型按钮或右键单击资源节点。
- 搜索资源类型：在搜索对话框中，键入要添加的 Amazon 资源类型。示例：
  - `AWS::S3::Bucket`
  - `AWS::Lambda::Function`
- 浏览资源：在资源部分，将显示您的账户中检测到的 Amazon 资源的分页列表。如果您拥有大量资源，则只会显示第一页。使用导航控件来浏览其他页面并查看所有可用资源。
- 选择要包含在您的模板中的资源。
- 您可以通过两种方式将资源插入到您的模板中，具体方式取决于您的目标：
  - 克隆现有资源：使用现有 Amazon 资源的实时配置和属性在模板中创建新资源。
  - 导入现有资源：通过将资源的实时状态添加到模板中，将其插入到您的堆栈中。

提示

- 您可以随时刷新资源部分，查看您的账户或区域中可用的最新资源列表。
- 如果您正在导入资源，请不要在同一个账户中添加属于现有 CloudFormation 堆栈的已存在资源。
- 要确认资源是否已由 CloudFormation 管理，请使用资源旁边的信息操作。在 VS Code 中，单击 i 图标。在 JetBrains IDE 中，右键单击资源并选择获取堆栈管理信息。

添加相关资源

在 VS Code 中，从命令面板运行 Amazon CloudFormation：按类型添加相关资源命令。从模板中已定义的资源类型中选择一种。IDE 会列出所选类型可通过其属性引用的资源类型。例如，如果您选择 `AWS::EC2::Instance`，则 IDE 可能会建议 `AWS::EC2::SecurityGroup` 或 `AWS::EC2::Subnet`。选择一个或多个相关资源类型，然后选择是创建新的资源定义还是导入现有资源。此功能目前在 JetBrains IDE 中不受支持。

静态验证

CloudFormation 语言服务器提供由 CloudFormation Linter 和 CloudFormation Guard 提供支持的内置静态验证。这些检查在后台本地运行，不需要 Amazon 凭证。它们可帮助您在部署前识别语法错误、合规性差距以及最佳实践问题。

静态验证概述

您将在 IDE 中看到两种类型的实时静态验证：

- CloudFormation Linter（`cfn-lint`）：根据 CloudFormation 资源规范和架构规则验证您的模板。
- Guard（`cfn-guard`）：根据合规性规则和组织策略包验证您的模板。

CloudFormation Linter（cfn-lint）

CloudFormation Linter 已集成到 IDE 中，以在您键入时自动检查模板的语法和结构。

- 架构验证：检测语法和架构错误，以确保您的模板符合 CloudFormation 资源架构。
- 错误突出显示：在出现问题时，会显示内联标记，这些标记代表部署拦截器或警告信息。
- 悬停获取帮助：将鼠标悬停在错误上方时，IDE 会显示与该问题相关的诊断消息。如果有快速修复方案可用，也会提供。

Guard 集成

Guard 会根据定义合规性和安全策略的规则集来验证您的模板。IDE 通过 CloudFormation 语言服务器实时运行 Guard 验证，以在您撰写模板时为您提供即时反馈。

- 默认规则包：IDE 包含了一套预先注册好的 Guard 规则，这些规则侧重于资源安全和配置维护方面的基本最佳实践。要了解更多信息，请参阅 GitHub 上的 Amazon Guard Rules Registry。
- 添加规则包：要添加或修改规则集，请打开“IDE 设置”，导航到“Guard 配置”部分，然后选择或上传其他 Guard 规则包。

提示：了解诊断指示器

- 蓝色指示器：最佳实践提示或优化建议。
- 黄色指示器：关于非拦截问题的警告（例如，缺少标签或参数）。
- 红色指示器：部署拦截器，例如无效的属性名称、缺少必填字段或架构不匹配。

第 4 步：浏览模板

IDE 提供 CloudFormation 模板的结构化分层视图，该视图分为 `Parameters`、`Resources`、`Outputs` 和 `Mappings` 等部分，显示了每种资源类型和逻辑 ID。这使得在大型模板中能够轻松快速地定位并导航到特定的资源或参数。在 VS Code 中，探索器侧栏中的大纲面板显示此结构。在 JetBrains IDE 中，打开结构工具窗口，查看当前打开的文件的模板结构。

您可以将转到定义用于 `GetAtt` 和 `Ref` 之类的内置函数，从而使您可以直接跳转到模板中引用的资源或参数。这可以帮助您跟踪依赖项、了解资源关系并更高效地进行编辑。

第 5 步：验证和部署

验证和部署是一个在线工作流，它调用 CloudFormation 服务，并且需要有效的 Amazon 凭证。当您准备好部署模板时，请使用此功能来创建更改集。IDE 会验证您的模板，如果未发现任何拦截错误，则会继续创建偏差感知更改集。然后 IDE 会显示差异视图，以便您可以在执行更改集之前查看所有提议的更改。

在 VS Code 中，打开命令面板并运行 Amazon CloudFormation: Validate and Deploy。命令面板将引导您逐步选择模板、堆栈名称、参数、功能和其他部署选项。在 JetBrains IDE 中，使用验证和部署工具栏按钮，在编辑器中右键单击模板文件或右键单击树中的堆栈。JetBrains 会显示一个向导对话框，您可以在其中配置所有部署选项，包括模板选择、堆栈名称、参数、功能、标签和高级选项。

验证的工作原理

IDE 会在部署前自动执行验证检查，并根据常见的故障原因对模板进行验证，包括：

- 无效的属性语法或架构不匹配：这些问题通常在撰写过程中被 `cfn-lint` 检测出来，但如果您在未解决这些问题的情况下继续进行部署，那么在创建或更新堆栈时，CloudFormation 的部署时验证也会显示出同样的错误。
- 资源名称与您账户中的现有资源冲突。
- 特定于服务的约束，例如 S3 存储桶名称冲突或缺少加密。

如果验证检测到错误，IDE 会直接在模板中突出显示问题，并在诊断面板中列出错误。每个问题都包括导致故障的特定属性或资源，并附上相应的解决建议。如果没有阻止错误，则可以继续进入部署阶段。

如果发现警告（非阻止问题），会出现一个对话框，让您可以选择继续进行部署操作，或者取消操作并进行修正。

IDE 将打开一个偏差感知更改集，其中显示当前模板与已部署堆栈配置之间的任何差异。这使您能够在执行之前查看、确认或取消更改集。取消部署会删除更改集。

偏差感知更改集能够增强 CloudFormation 的部署流程，使您能够安全地处理堆栈的偏差问题。当您的资源的实际状态与 CloudFormation 模板中所定义的内容不一致时，就会出现堆栈偏差，这种情况通常因通过 Amazon Web Services 管理控制台、CLI 或 SDK 进行的手动更改导致。CloudFormation 偏差感知更改集将处理后的堆栈配置与实时资源状态进行比较，IDE 会显示这些差异，这样您就可以在部署之前使资源恢复合规性。

查看堆栈事件

部署开始后，您可以从 CloudFormation 面板实时监控进度。在堆栈事件下，您将看到部署期间执行的操作列表。每个事件都包含详细信息，例如：

- 时间戳：事件发生的时间
- 资源：正在创建、更新或删除的特定 Amazon 资源
- 状态：操作的当前状态（例如 `CREATE_IN_PROGRESS`、`UPDATE_COMPLETE` 或 `ROLLBACK_IN_PROGRESS`）
- 原因：其他背景信息或错误消息（如果适用）

您也可以通过此面板查看堆栈的资源和输出。堆栈事件视图有助于您跟踪部署进度、识别潜在问题，并确认您的堆栈是否已成功完成部署。

在 IDE 中初始化 CloudFormation 项目（仅限 VS Code）

在 IDE 中初始化 CloudFormation 项目可帮助您使用正确的文件夹和环境配置设置结构化工作区，以便进行服务端验证和部署。这些在线工作流需要 Amazon 凭证，但是语言服务器的本地创作和静态验证功能不需要。您可以直接从 IDE 来初始化一个新的 CloudFormation 项目，从而创建此建议设置。此功能目前仅在 VS Code 中可用，在 JetBrains IDE 中不受支持。

要初始化 CloudFormation 项目，请执行以下操作：

- 打开命令面板

  - 从 VS Code，打开命令面板（macOS 上的 `Ctrl+Shift+P` 或 `Cmd+Shift+P`）。
  - 选择 Amazon CloudFormation：CFN Init：初始化项目。
- 选择项目目录
  - 默认情况下，IDE 将使用您当前的工作目录。
  - 您可以将此路径更改为您希望存储 CloudFormation 模板所在的任何文件夹。
- 选择您的 Amazon 凭证配置文件
  - 系统将提示您选择 Amazon 凭证配置文件。所选配置文件用于环境检测、服务端验证和部署。
- 设置环境
  - 系统将提示您创建或选择环境。
  - 环境定义了您的模板在何处以及以何种方式进行部署或验证（例如，开发环境、测试环境或生产环境）。您可以使用 Amazon CloudFormation：CFN Init：添加环境来选择或更改您的环境。
  - 您可以使用 Amazon CloudFormation: CFN Init: Remove Environment 移除您选择的环境。
- （可选）导入参数文件
  - 如果您已经有现有的参数文件，那么该 IDE 会在初始化时允许您导入这些文件。
  - IDE 会自动检测兼容的文件，并将它们链接到您的项目中，以便在模板验证和部署过程中使用。
- 命名并最终确定项目
  - 提供项目名称，例如 beta-environment，然后完成设置。
  - IDE 将为您创建初始项目结构和配置文件。

您可以直接从 IDE 中进行验证、预览部署操作，或者在不同环境中进行切换。

需要帮助？

在 Amazon Web Services re:Post 上试用 CloudFormation 社区。

使用 `cfn-lint` 验证模板

Amazon CloudFormation Linter（`cfn-lint`）根据 CloudFormation 资源提供程序架构和其他规则检查 JSON 和 YAML 模板。它可以发现无效的资源属性、不支持的属性值、损坏的引用以及常见的最佳实践问题。在将模板发送给 CloudFormation 之前，请使用它来发现这些问题。如果模板使用 Amazon Serverless Application Model（Amazon SAM）转换，则 `cfn-lint` 会在检查模板之前应用该转换。

有关发布说明、完整的规则列表和所有命令选项，请参阅 GitHub 上的 `cfn-lint`。

有关本地验证的限制以及部署前需采取的步骤，请参阅了解验证范围。

主题

- 安装 cfn-lint
- 运行 cfn-lint
- 了解结果
- 配置检查
- 添加自定义规则
- 向您的工作流添加 cfn-lint

安装 cfn-lint

从 Python 包索引（PyPI）安装 `cfn-lint`：

```
python3 -m pip install cfn-lint
```

在 macOS 上，您也可以使用 Homebrew 来安装它：

```
brew install cfn-lint
```

要确认安装，请输出已安装的版本：

```
cfn-lint --version
```

有关 Docker 映像和其他安装方法，请参阅 GitHub 上的安装说明。

运行 cfn-lint

向命令传递一个或多个模板路径：

```
cfn-lint template.yaml
cfn-lint template-one.yaml template-two.json
```

您还可以使用通配符来检查目录中的所有模板。使用 ** 来包含子目录：

```
cfn-lint templates/*.yaml
```

```
cfn-lint templates/**/*.yaml
```

> **Note**
>
> 在 bash 中，在使用 ** 之前先运行 shopt -s globstar，使其与子目录匹配。在 zsh 中，** 默认情况下匹配子目录。

要从标准输入中读取模板，请将 - 作为模板路径传递：

```
cat template.yaml | cfn-lint -
```

默认情况下，cfn-lint 会根据 us-east-1 的架构检查模板。要检查其他 Amazon Web Services 区域的模板，请使用 --regions 选项。由于此选项接受多个值，因此请在模板路径之前添加 --，或使用 -t 来标识模板：

```
cfn-lint --regions us-east-1 us-west-2 -- template.yaml
cfn-lint --regions us-east-1 us-west-2 -t template.yaml
```

某些检查取决于参数值。要检查具有特定值的模板，请通过 --parameters 传递它们，或通过 --parameter-files 提供参数文件：

```
cfn-lint --parameters InstanceType=t3.micro Environment=prod -- template.yaml
cfn-lint --parameter-files params.json -- template.yaml
```

要列出 cfn-lint 可应用的所有规则，请运行以下命令：

```
cfn-lint --list-rules
```

了解结果

对于每个调查发现，cfn-lint 会输出规则 ID、一条消息以及问题的文件、行和列。有关规则的完整列表及其各自的检查内容，请参阅 GitHub 上的 cfn-lint 规则。

要更改输出格式，请将 --format 与quiet、parseable、json、junit、pretty 或 sarif 配合使用：

```
cfn-lint --format json template.yaml
```

cfn-lint 未发现任何问题时返回退出代码 0，否则返回非零退出代码。要控制哪些调查发现导致非零退出代码，请使用 --non-zero-exit-code。当模板未通过您选择的检查时，您可以使用退出代码停止自动化构建。

配置检查

您可以使用命令行选项、配置文件或模板中的元数据来配置 cfn-lint。当同一设置出现在多处时，命令行选项会覆盖模板元数据，而模板元数据会覆盖配置文件。

选择并配置规则

要跳过规则，请使用 --ignore-checks 并指定一个或多个规则 ID 或规则 ID 前缀。要运行默认关闭的规则，请使用 --include-checks。要在规则与忽略设置匹配的情况下运行该规则，请使用 --mandatory-checks。

```
cfn-lint --ignore-checks RULE_ID -- template.yaml
```

某些规则接受设置。使用 --configure-rule 并采用格式 *RULE_ID*:*key*=*value*：

```
cfn-lint --configure-rule RULE_ID:key=value -- template.yaml
```

使用配置文件

要保留模板的设置，请在运行 cfn-lint 的目录中创建 .cfnlintrc、.cfnlintrc.yaml 或 .cfnlintrc.yml 文件。您还可以在主目录中创建 ~/.cfnlintrc 文件，以获取适用于所有地方的设置。以下示例选择了要检查的模板，跳过了其中一个模板，设置了区域，并跳过了一条规则：

```
templates:
  - templates/**/*.yaml
ignore_templates:
  - templates/legacy.yaml
regions:
  - us-east-1
  - us-west-2
ignore_checks:
  - RULE_ID
```

当配置文件选择模板时，您可以不带参数地运行 cfn-lint 来检查所有模板。有关所有设置，请参阅 GitHub 上的 Config file。

在模板中配置检查

要为一个模板配置 cfn-lint，请在顶层 Metadata 部分添加一个 cfn-lint 键：

```
Metadata:
  cfn-lint:
    config:
      regions:
        - us-east-1
        - us-east-2
      ignore_checks:
        - RULE_ID
```

要跳过单个资源的规则，请在该资源的 Metadata 属性中添加相同的键。以下示例仅跳过一个实例的规则：

```
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Metadata:
      cfn-lint:
        config:
          ignore_checks:
            - RULE_ID
    Properties:
      InstanceType: t3.micro
      ImageId: ami-abc1234
```

添加自定义规则

您可以在不编写 Python 的情况下添加自己的检查。自定义规则文件每行包含一条规则，格式如下：

```
ResourceType Property Operator Value [ErrorLevel] [Message]
```

例如，当模板使用昂贵的实例类型时，custom_rules.txt 中的以下规则会报告警告：

```
AWS::EC2::Instance InstanceType NOT_EQUALS "m4.16xlarge" WARN "This is an expensive
 instance type, don't use it"
```

通过 --custom-rules（或 -z）传递文件，或者在配置文件中添加 custom_rules 设置：

```
cfn-lint --custom-rules custom_rules.txt -- template.yaml
```

有关支持的运算符，请参阅 GitHub 上的 Custom rules。对于更复杂的检查，您可以用 Python 编写规则并使用 `--append-rules` 加载它们。要更改 `cfn-lint` 认为有效的内容，例如要求特定标签，您可以通过 `--override-spec` 提供一个规范覆盖文件。有关更多信息，请参阅 GitHub 上的 Getting started with rules 和 Customize specifications。

向您的工作流添加 `cfn-lint`

您可以在以下位置运行 `cfn-lint`：

- 在您的编辑器中：CloudFormation 语言服务器使用 `cfn-lint` 在您键入时显示问题。有关设置，请参阅CloudFormation 语言服务器。其他编辑器（包括 Vim、Emacs 和 Sublime Text）也有相应的插件可用。
- Git 提交之前：将 `cfn-lint` 添加到 `.pre-commit-config.yaml` 文件中，使其在每次提交之前在更改的模板上运行。请将 `rev` 值替换为要使用的版本。

```
repos:
  - repo: https://github.com/aws-cloudformation/cfn-lint
    rev: v1.56.0
    hooks:
      - id: cfn-lint
        files: templates/.*\.(json|yml|yaml)$
```

- 在自动化构建中：将 `cfn-lint` 作为构建步骤运行，并使用退出代码使构建失败。此外，还有一个适用于 `cfn-lint` 的 GitHub 操作。
- 在您自己的 Python 代码中：导入 `cfnlint.api` 并调用 `lint` 或 `lint_all` 来检查模板字符串，并在应用程序中处理结果。

```
from cfnlint.api import lint, ManualArgs

with open("template.yaml") as f:
    template = f.read()

matches = lint(template, config=ManualArgs(regions=["us-east-1"]))
for match in matches:
    print(match)
```

有关编辑器插件、GitHub 操作和更多集成示例，请参阅 GitHub 上的 cfn-lint。

使用 cloudformation-validate 验证模板

Amazon CloudFormation 验证（cloudformation-validate）可帮助您在部署之前发现模板问题。它会在本地检查 JSON 和 YAML 模板，识别无效的模板结构、损坏的引用、安全问题以及最佳实践问题。每项调查发现都包括稳定的规则 ID、严重性、模板中的行和列、受影响的资源，以及建议的修复（如果有）。

该工具包括所需的规则和 CloudFormation 资源架构。因此，在安装完成后，它无需网络访问或 Amazon 凭证即可运行。验证一个典型模板所需的时间不到一秒。您既可以通过命令行运行验证，也可以将其嵌入到 Rust、Node.js、Python、Go 或 JVM 应用程序中。您也可以让 Amazon CDK 在合成之后运行它。所有这些方法产生的结果都相同。

有关发布说明和内置规则的完整列表，请参阅 GitHub 上的 cloudformation-validate。

有关本地验证的限制以及部署前需采取的步骤，请参阅了解验证范围。

主题

- 验证的工作原理
- 选择验证方法
- 安装命令行工具
- 运行验证
- 了解结果
- 筛选结果
- 添加自定义规则
- 验证新的资源类型和属性
- 嵌入验证库
- 将 与 结合使用Amazon CDK

验证的工作原理

验证模板时，cloudformation-validate 会解析内置函数（例如 Ref、Fn::GetAtt、Fn::Sub 和 Fn::If），然后运行以下检查：

- 架构检查：将每个资源与 CloudFormation 资源提供程序架构进行比较，并报告类型不匹配、缺失必需属性、枚举值无效以及模式或约束失败。
- 规则：运行内置规则和您提供的任何自定义规则。这些规则可以发现语义错误、损坏的跨资源引用、安全风险和最佳实践问题。
- 状态机检查：验证 `AWS::StepFunctions::StateMachine` 资源的定义，包括状态类型、`StartAt` 和 `Next` 引用以及必填字段。

选择验证方法

- 命令行：运行 `cfn-validate` 以从终端或自动构建检查一个模板或目录中的每个模板。请参阅安装命令行工具。
- 库：向 Rust、Node.js、Python、Go 或 JVM 应用程序添加验证器，然后在代码中处理结果。请参阅嵌入验证库。
- Amazon CDK：在 Amazon CDK 合成模板后自动验证模板。请参阅将 与 结合使用Amazon CDK。

无论采用哪种方法，您都可以添加使用公共表达式语言（CEL）、Rego 或 Guard 规则语言编写的自定义规则。请参阅添加自定义规则。

安装命令行工具

`cfn-validate` 命令是一个没有依赖项的单个文件。请在 GitHub 上打开最新版本，然后下载适用于您的操作系统和处理器的文件。文件适用于 Linux（x86-64 和 ARM64）、macOS（Apple silicon 和 Intel）以及 Windows（x86-64 和 ARM64）。

在 Linux 或 macOS 上，将文件设为可执行文件，将其重命名为 `cfn-validate`，然后将其移动到您的 PATH 中的目录。以下示例使用 Linux x86-64 文件。请将 *VERSION* 替换为您下载的版本。

```
chmod +x cfn-validate-VERSION-linux-x64
sudo mv cfn-validate-VERSION-linux-x64 /usr/local/bin/cfn-validate
```

在 Windows 上，将文件重命名为 `cfn-validate.exe`，然后将其移动到您的 PATH 中的目录。

每个版本还包括每次下载的签名文件以及用于对其进行签名的公有密钥。要在使用下载之前对其进行验证，请参阅 GitHub 上的 Verify a downloaded release asset。

运行验证

该命令使用以下语法：

```
cfn-validate TEMPLATE_OR_DIRECTORY [OPTIONS]
```

检查一个模板：

```
cfn-validate template.yaml
```

传递一个目录，以递归方式检查每个 .yaml、.yml 和 .json 文件：

```
cfn-validate ./templates/
```

某些检查取决于 Amazon Web Services 区域 或参数值。请使用 --region 设置 AWS::Region 伪参数，使用 --parameter 设置模板参数，使用 --pseudo-parameter 设置其他伪参数（例如 AWS::AccountId 或 AWS::Partition）。您可以重复 --parameter 和 --pseudo-parameter：

```
cfn-validate template.yaml --region us-west-2 --parameter Environment=prod --parameter
 InstanceType=t3.micro
```

要将警告视为错误（例如在自动化构建中），请添加--strict。要列出所有内置规则，请运行 cfn-validate --list-rules。

了解结果

该命令将 JSON 报告写入标准输出。报告中的每项调查发现都包括规则 ID、严重性、消息、源位置（行和列）、资源或其他模板实体以及建议的修复（如果有）。默认 --format detailed 报告还包括规则描述和相关上下文。请使用 --format standard 生成省略这些字段的精简报告。

要将报告限制为更严重的调查发现，请使用 --level 并设置最低严重性以包括：

```
cfn-validate template.yaml --level error
```

该命令在未发现错误时会返回退出代码 0，在发现错误时会返回 1，在出现使用或初始化错误（例如无效选项或文件不存在）时会返回 2。使用 --strict，警告也会导致退出代码 1。

筛选结果

您可以按规则 ID、规则 ID 范围、类别、资源类型、服务前缀或模板实体的逻辑 ID 来包含或排除调查发现。每个筛选条件都有一个 --include- 和一个 --exclude- 形式。以下示例展示了常见的筛选条件：

```
# Suppress specific rules everywhere
cfn-validate template.yaml --exclude-ids RULE_ID,RULE_ID

# Report only security findings
cfn-validate template.yaml --include-categories Security

# Suppress one rule for one resource only
cfn-validate template.yaml --exclude-logical-id MyBucket=RULE_ID

# Suppress every rule for a resource type
cfn-validate template.yaml --exclude-resource-type AWS::EC2::Instance

# Suppress every rule for a service
cfn-validate template.yaml --exclude-service AWS::AutoScaling
```

以资源类型、服务或逻辑 ID 为目标的筛选条件接受可选 =*RULE_ID* 后缀，将筛选条件限制为一条规则。如果没有后缀，则筛选条件将应用于该目标的每条规则。有关所有筛选条件选项，请参阅 GitHub 上的 cfn-validate CLI 参考。

添加自定义规则

您可以采用以下任一格式添加您自己的检查。每种格式都假设用户已基本了解其规则语言。默认情况下，除了内置规则外，还会运行自定义规则。要仅运行自定义规则，请添加 `--disable-builtin-rules`。

- CEL（`.json`）：JSON 文件中以 CEL 表达式编写的属性和数据驱动的检查。通过 `--rule-source` 加载。
- Rego（`.rego`）：以 Rego 策略语言编写的复杂跨资源逻辑。通过 `--rule-source` 加载。
- Guard（`.guard`）：Guard 规则语言中的声明式合规性规则。使用 `--guard-rule-source` 加载文件或目录。验证器支持部分 Guard 语言。如果规则使用了不受支持的构造，验证器会在加载该规则时将其拒绝，而不是忽略这些构造。有关规则语法，请参阅使用 Guard 验证模板。

您可以重复任一选项以加载多个规则源：

```
cfn-validate template.yaml --rule-source ./rules/checks.json --rule-source ./rules/
network.rego
cfn-validate template.yaml --guard-rule-source ./guard-rules/
```

checks.json 中的以下 CEL 示例会针对每个未加密的 Amazon S3 存储桶报告一项调查发现。resource_type 字段会使规则针对该类型的每个资源各运行一次，而消息中的 {name} 会被替换为资源的逻辑 ID。

```
{
  "rules": [
    {
      "rule_id": "CUSTOM001",
      "severity": "ERROR",
      "resource_type": "AWS::S3::Bucket",
      "expression": "!has(properties.BucketEncryption)",
      "message": "S3 bucket {name} must have encryption configured",
      "prop_path": "Properties.BucketEncryption",
      "suggested_fix": "Add BucketEncryption with SSEAlgorithm: aws:kms"
    }
  ]
}
```

有关规则可用数据、支持的 CEL 函数以及编写 Rego 规则的更多信息，请参阅 GitHub 上的使用 Guard 验证模板。

验证新的资源类型和属性

捆绑的架构与发布时的 CloudFormation 注册表相匹配。如果某模板使用 CloudFormation 在该版本之后发布的资源类型、属性或允许的值，或者使用您自己的注册表中的私有资源类型，请使用 --additional-schema 传递资源提供程序架构。验证器使用它来扩展现有类型的捆绑架构，或对新类型进行验证。您可以传递一个 .json 文件或架构文件目录，然后重复以下选项：

```
cfn-validate template.yaml --additional-schema ./schemas/
```

附加架构永远不会移除捆绑的架构中的约束。如果无法读取或应用架构，则命令将退出并显示代码 2，而不是忽略它。

嵌入验证库

安装适用于所用编程语言的程序包，以便从您的应用程序运行相同的离线检查。创建一次验证引擎，将其重复用于多个模板，然后处理针对每个模板返回的结构化诊断。您还可以通过库 API 配置自定义 CEL、Rego 或 Guard 规则以及其他架构。

语言	程序包	安装	文档
Rust	cloudformation-validate crates.io 上的	`cargo add cloudformation-validate`	API 和示例
Node.js	@aws/cloudformation-validate npm 上的	`npm install @aws/cloudformation-validate`	API 和示例
Python	cloudformation-validate PyPI 上的	`python3 -m pip install cloudformation-validate`	API 和示例
Go	pkg.go.dev 上的 cloudformation-validate/src/bindings-go/go	`go get github.com/aws-cloudformation/cloudformation-validate/src/bindings-go/go@latest`	API 和示例
JVM（Java 或 Kotlin）	software.amazon.cloudformation:cloudformation-validate Maven Central 上的	`implementation("software.amazon.cloudformation:cloudformation-validate:VERSION")`（Gradle）或等效的 Maven 依赖项	API 和示例

每个库都公开了一种方法来验证模板并返回具有与命令行工具相同的诊断结果的报告。以下示例使用默认引擎 `CompositeEngine`，它运行内置规则和您配置的任何自定义 CEL、Rego 或 Guard 规则。每个示例都会验证一个模板并输出每个调查发现。

Python：

```
from cloudformation_validate import CompositeEngine

engine = CompositeEngine()
report = engine.validate_template("template.yaml")
for d in report.diagnostics:
    print(f"[{d.severity.name}] {d.rule_id}: {d.message}")
```

Kotlin：

```
import software.amazon.cloudformation.validate.CompositeEngine
import java.io.File

fun main() {
    val engine = CompositeEngine()
    val report = engine.validateTemplate(File("template.yaml"))
    for (d in report.diagnostics) {
        println("[${d.severity}] ${d.ruleId}: ${d.message}")
    }
}
```

有关支持的语言运行时、版本固定、预发布渠道以及 Maven 和 Gradle 语法，请参阅 GitHub 上的语言绑定安装指南。有关每种语言的 API 参考和完整示例，请参阅 GitHub 上的 Embedding as a library。

将 与 结合使用Amazon CDK

如果您使用 Amazon CDK，则无需单独安装验证器。Amazon CDK 构造库包含一个默认 `CloudFormationValidatePlugin`，其会在合成 CloudFormation 模板后自动运行相同的检查。

```
cdk synth
```

Amazon CDK 会在其验证结果中报告可能的部署失败和最佳实践调查发现。当您想要添加自定义 Rego 或 Guard 规则时，可以显式配置 `CloudFormationValidatePlugin`。

有关插件配置、确认和验证报告，请参阅《Amazon CDK API 参考》中的 Template and Policy Validation。

使用 Guard 验证模板

Amazon CloudFormation Guard（`cfn-guard`）是一种策略即代码工具。您可以使用 Guard 语言编写描述必需或禁止配置的规则，然后根据这些规则检查 JSON 或 YAML 数据。例如，您可以要求模板中的每个 Amazon S3 存储桶都使用加密，或者屏蔽向互联网开放端口 22 的安全组规则。

Guard 适用于任何 JSON 或 YAML 数据。除了 CloudFormation 模板外，您还可以查看 CloudFormation 更改集和 Amazon Config 配置项目。本主题侧重于检查模板。

有关发布说明和完整的命令参考，请参阅 GitHub 上的 Guard。有关完整的语言参考，请参阅《Guard 用户指南》。

有关本地验证的限制以及部署前需采取的步骤，请参阅了解验证范围。

主题

- 安装 Guard
- 了解 Guard 语言
- 编写规则
- 验证模板
- 测试规则
- 从模板生成规则
- 向您的工作流添加 Guard

安装 Guard

在 macOS 上，使用 Homebrew 安装 Guard：

```
brew install cloudformation-guard
```

在 Linux 或 macOS 上，您可以改为运行安装脚本。该脚本将最新版本安装到 `~/.guard/bin`。脚本完成后，请将该目录添加到您的 PATH。

```
curl --proto '=https' --tlsv1.2 -sSf https://raw.githubusercontent.com/aws-cloudformation/cloudformation-guard/main/install-guard.sh | sh
```

如果您有 Rust 工具链，也可以通过 Cargo 安装 Guard：

```
cargo install cfn-guard
```

要确认安装，请输出已安装的版本：

```
cfn-guard --version
```

有关 Windows 安装、Docker 映像和版本验证的信息，请参阅 Setting up Guard。

了解 Guard 语言

现在 Guard 已安装完毕，您可以开始编写规则了。Guard 规则由以下部分构成：

- 查询：使用虚线路径从数据中选择值，例如 `Resources.MyBucket.Properties.BucketName`。使用 `*` 选择映射中的每个值，使用 `[*]` 选择列表中的每个项目。在方括号中添加筛选条件以缩小选择范围。例如，`Resources.*[ Type == 'AWS::S3::Bucket' ]` 选择每个 Amazon S3 存储桶。
- 子句：做出判定为真或假的断言。子句将查询与运算符（例如 `==`、`!=`、`>=`、`in`、`exists`、`empty` 或 `is_string`）组合在一起。位于不同行上的子句必须全部通过。要接受两个子句中的任何一个，请将其与 `or` 联接。
- 变量：使用 `let` 存储查询结果，并在以后使用 `%` 前缀引用它。变量允许您一次性选择一组资源，并在多条规则中重复使用它。
- 命名规则：使用名称对 `rule` 块中的子句进行分组。添加一个 `when` 条件，以便在规则不适用时将其跳过，例如，当模板没有 Amazon S3 存储桶时。规则可以引用其他规则，因此您可以从小部分构建复杂的策略。
- 自定义消息：在子句后的 `<<` 和 `>>` 之间添加文本，以解释输出中的失败。

有关完整语法，请参阅 Writing Guard rules。

编写规则

将规则保存为扩展名为 `.guard` 的文件。`rules.guard` 中的以下示例要求模板中的每个 Amazon S3 存储桶都必须指定存储桶加密并屏蔽公共访问权限：

```
let s3_buckets = Resources.*[ Type == 'AWS::S3::Bucket' ]

rule S3_BUCKET_ENCRYPTED when %s3_buckets !empty {
  %s3_buckets.Properties.BucketEncryption exists
  <<
    Violation: S3 buckets must specify BucketEncryption.
    Fix: Add a BucketEncryption property with ServerSideEncryptionConfiguration.
  >>
}

rule S3_BUCKET_PUBLIC_ACCESS_BLOCKED when %s3_buckets !empty {
  %s3_buckets.Properties.PublicAccessBlockConfiguration {
    BlockPublicAcls == true
    BlockPublicPolicy == true
    IgnorePublicAcls == true
    RestrictPublicBuckets == true
  }
```

```
}
```

第一行将每个 Amazon S3 存储桶存储在 s3_buckets 变量中。仅当模板包含至少一个存储桶时，每条规则才会运行。第一条规则检查 BucketEncryption 属性是否存在，并包含一条在检查失败时由 Guard 输出的消息。第二条规则检查 PublicAccessBlockConfiguration 内部的四个属性。由于子句位于不同的行中，因此所有四个子句均须通过。

您可以将多条规则保存在同一个文件中，也可以按主题将规则拆分到不同的文件中，例如，将存储规则存入一个文件，将网络规则存入另一个文件。当您将目录传递给命令时，Guard 会评估该目录中的每个 .guard 文件。

验证模板

将规则文件传递给 --rules，将模板传递给 --data。添加 --type CFNTemplate 以便 Guard 按资源逻辑 ID 报告调查发现。

```
cfn-guard validate --rules rules.guard --data template.yaml --type CFNTemplate
```

模板通过后，Guard 会输出 PASS 状态并返回退出代码 0。当规则未通过时，Guard 会列出未通过的规则；对于每个不合规的资源，它还会列出未通过的检查、属性路径以及原因。对于存储桶缺失 BucketEncryption 的模板，其输出以如下内容开头：

```
template.yaml Status = FAIL
FAILED rules
rules.guard/S3_BUCKET_ENCRYPTED    FAIL
---
Evaluating data template.yaml against rules rules.guard
Number of non-compliant resources 1
Resource = MyBucket {
  Type      = AWS::S3::Bucket
  Rule = S3_BUCKET_ENCRYPTED {
    ...
```

当一条或多条规则未通过时，Guard 会返回退出代码 19；当无法解析规则或数据文件时，则返回退出代码 5。您可以使用这些退出代码来停止自动化构建。

在验证模板时，以下选项非常有用：

- 多次传递 --rules 或 --data，或传递一个目录，以在一次运行中检查多个规则文件或模板。传递到 --data 的目录必须仅包含数据文件，而传递到 --rules 的目录必须仅包含规则文件。

- 将 `--show-summary` 与 `pass`、`fail`、`skip`、`all` 或 `none` 配合使用以控制摘要表中显示哪些规则。默认值为 `fail`。
- 将 `--output-format` 与 `json`、`yaml`、`junit` 或 `sarif` 配合使用，为其他工具生成输出。`junit` 和 `sarif` 格式适用于大多数构建系统和代码扫描工具。
- 使用 `--input-parameters` 传递包含规则所引用的值的 JSON 或 YAML 文件，例如已批准的安全组 ID 列表。这使得环境特定的值不会出现在您的规则中。

有关所有选项，请参阅《Guard 用户指南》中的 validate。

测试规则

在自动化工作流中使用规则之前，请利用内置的单元测试支持对其进行测试。测试文件列出了示例输入以及针对每条规则所预期的结果：PASS、FAIL 或 SKIP。`rules_tests.yaml` 中的以下示例测试了上一个示例中的 `S3_BUCKET_ENCRYPTED` 规则：

```
---
- input:
    Resources: {}
  expectations:
    rules:
      S3_BUCKET_ENCRYPTED: SKIP
- input:
    Resources:
      MyBucket:
        Type: AWS::S3::Bucket
        Properties: {}
  expectations:
    rules:
      S3_BUCKET_ENCRYPTED: FAIL
- input:
    Resources:
      MyBucket:
        Type: AWS::S3::Bucket
        Properties:
          BucketEncryption:
            ServerSideEncryptionConfiguration:
              - ServerSideEncryptionByDefault:
                  SSEAlgorithm: aws:kms
  expectations:
    rules:
```

```
    S3_BUCKET_ENCRYPTED: PASS
```

使用 `test` 命令运行测试：

```
cfn-guard test --rules-file rules.guard --test-data rules_tests.yaml
```

Guard 为每个期望值输出 PASS 或 FAIL，并在任何期望失败时返回退出代码 7。有关更多信息，请参阅 Testing Guard rules。

从模板生成规则

若想快速上手，您可以根据已满足您要求的模板生成规则。`rulegen` 命令会读取模板，并为需要模板中的属性值的每种资源类型编写规则：

```
cfn-guard rulegen --template template.yaml --output generated.guard
```

检查生成的文件，只保留代表您策略的检查。生成的规则是一个起点，通常需要源模板中的精确值，这对于其他模板可能过于严苛。

向您的工作流添加 Guard

您可以在以下位置运行 Guard：

- 在您的编辑器中：CloudFormation 语言服务器会在您编辑模板时运行 Guard 规则包，并以内联方式显示失败。有关设置，请参阅CloudFormation 语言服务器。
- 在 Git 提交之前：Guard 提供了一个 `pre-commit` 钩子，可以在更改的模板上运行 `validate` 或在更改的规则上运行 `test`。请将以下内容添加到 `.pre-commit-config.yaml` 文件中，并将路径替换为您自己的路径：

```
repos:
  - repo: https://github.com/aws-cloudformation/cloudformation-guard
    rev: pre-commit-v0.0.2
    hooks:
      - id: cfn-guard
        args:
          - --operation=validate
          - --rules=rules/
        files: ^templates/.*
```

- 在自动化构建中：将 `cfn-guard validate` 作为构建步骤运行，并使用退出代码使构建失败。使用 `--output-format junit` 或 `--output-format sarif` 发布结果。此外还提供了一个 GitHub 操作和一个容器映像。请从 `public.ecr.aws/aws-cloudformation/cloudformation-guard` 提取映像。
- 使用 **`cloudformation-validate`**：`cfn-validate` 命令和 Amazon CDK 验证插件可以运行支持的 Guard 规则子集及其内置检查。有关更多信息，请参阅 使用 `cloudformation-validate` 验证模板。
- 堆栈操作期间：要在 CloudFormation 或 Cloud Control API 创建、更新或删除资源时强制执行规则，请使用 Guard 钩子。有关更多信息，请参阅 Guard Hooks。

有关涵盖常见安全和合规控制的即用型规则，请参阅 GitHub 上的 Amazon Guard Rules Registry。

使用模板工作室直观地编辑模板

模板工作室是 CloudFormation 控制台中的一款模板编辑器。它以文本形式展示模板，并在旁边呈现该模板所声明资源的示意图。在您编辑模板时，示意图会重新绘制以匹配。您无需离开控制台，即可验证模板、在 YAML 和 JSON 格式之间进行转换，并将其保存至 Amazon S3。

模板文本具有权威性。示意图是对该文本的视觉呈现，而非一个可独立编辑的区域。您可以在模板中进行所有更改。

注意事项

在模板工作室中工作时，请牢记以下几点：

- 您是在模板中创作的，而不是在示意图上。示意图展示您的模板所声明的资源以及这些资源之间的相互引用关系；您无法通过编辑示意图来添加或配置资源。
- 您的编辑不会自动保存。如果您在存在未保存更改的情况下尝试离开页面，模板工作室会要求您进行确认。如果您离开或刷新页面，则未保存的更改会被丢弃。在离开之前，请保存到 Amazon S3，或者复制或下载模板。
- 将模板从 YAML 转换为 JSON 会丢弃其注释，因为 JSON 没有相应的语法。转换回 YAML 并不能恢复它们。
- 在模板工作室中，对于大小超过 51,200 字节（50 KiB）的草稿，目前无法进行 CloudFormation 验证。当前的“验证”操作通过 `TemplateBody` 参数发送草稿，该参数存在此项限制。这并非模板的整体大小限制。您仍然可以保存该模板并将其用于堆栈操作，但需受 CloudFormation 配额限制。有关更多信息，请参阅验证模板。

- 保存操作会将模板写入您账户中的存储桶，因此您所保存的模板将按照标准的 Amazon S3 费率计费。有关更多信息，请参阅 Amazon S3 定价。
- 仅当画布以画布或两者视图显示时，您才能下载资源示意图。

模板工作室所需的权限

您所需的权限取决于模板来源以及您所执行的操作：

`cloudformation:GetTemplate`

打开现有堆栈的模板时需要。授予对您想要编辑其模板的堆栈的访问权限。

`s3:GetObject`

使用您的凭证从 Amazon S3 读取模板时需要。授予对源模板对象的访问权限。预签名 URL 使用该 URL 中包含的授权信息。

`cloudformation:ValidateTemplate`

选择验证时需要。

`s3:PutObject`

保存模板时需要。例如，将此权限的范围限定为目标存储桶中的对象 `arn:aws:s3:::destination-bucket/*`，而不是所有资源。

`s3:CreateBucket`

仅在控制台创建缺失的目标存储桶时才需要。这可以是控制台管理的存储桶，也可以是您指定的存储桶名称。重复使用现有存储桶无需此权限。

`s3:ListBucket`

当保存至您指定的现有存储桶时需要。模板工作室在上传前会使用 `HeadBucket` 来验证访问权限、预期的存储桶所有者以及存储桶 Amazon Web Services 区域。

对于控制台管理的存储桶，此权限是可选的。若没有它，保存对话框仍会显示存储桶名称。保存需要 `s3:PutObject`，如果必须创建存储桶，则需要 `s3:CreateBucket`。

打开模板工作室

打开模板工作室的位置决定了模板工作室的打开方式以及保存后会发生什么。

启动新模板

1. 打开 CloudFormation 控制台。
2. 在屏幕顶部的导航栏中，选择您想要部署的 Amazon Web Services 区域。
3. 在导航窗格中，选择模板工作室。

在创建堆栈时构建模板

1. 在创建堆栈页面上，选择使用模板工作室构建。
2. 构建模板，然后选择保存并继续。
3. 堆栈向导会继续使用您已保存的模板，因此您无需另行上传。

编辑现有堆栈的模板

1. 在堆栈页面上，选择堆栈的名称。
2. 既可以直接更新堆栈，也可以创建更改集。
3. 在准备模板页面上，选择在模板工作室中打开。堆栈的当前模板随即打开。
4. 选择保存并继续即可返回至包含您所编辑模板的更新或更改集。

在控制台的其他位置，凡是显示模板的地方，您都可以选择在模板工作室中打开来处理该模板。

使用模板和示意图

模板工作室打开，模板和示意图并排显示。使用视图控件来选择屏幕上显示的内容：

模板 –

仅模板，全宽显示。使用它来读取或写入没有示意图的模板。

画布。

仅示意图，全宽显示。将其用于示意图需要全宽的大型模板。

二者

模板与示意图并列显示。拖动两者之间的分隔线，即可为其中一侧留出更多空间。

示意图将相关资源归为一组，若某资源通过 Ref、Fn::GetAtt、Fn::Sub 或 DependsOn 引用了另一资源，示意图便会绘制相应的连接。未被任何其他资源引用的资源会单独显示，这使得人们能够轻松发现那些意外断开连接的资源。

> **Note**
>
> 例如，三个函数可以代入相同的 IAM 角色。示意图展示每个函数内部的角色，但模板仅声明一个角色。查看模板，了解其中存在多少资源。

在 YAML 和 JSON 之间进行转换

您保存的格式决定了模板工作室写入 Amazon S3 的对象的文件名和内容类型。

转换您正在编辑的模板

1. 选择格式控件，它会显示您当前所处格式的名称。
2. 选择其他格式。
3. 将 YAML 转换为 JSON 时，系统会要求您进行确认，因为注释无法被保留。若要继续，请选择转换。

> **Important**
>
> JSON 没有用于注释的语法。当您将包含注释的 YAML 模板转换为 JSON 时，这些注释会被移除，即便再将其转换回 YAML，这些注释也无法恢复。在转换之前，请将您想要保留的内容复制下来。

当模板无法读取时，格式控件不可用，因为没有足够完整的内容可以转换。请先修复模板。

键入时出现语法问题

编辑器会在您编写模板时检查其语法，并为无法读取的内容加下划线，例如未封闭的方括号、放错位置的缩进或重复的键。将鼠标悬停在下划线上即可查看问题。如果某一行存在多个问题，这些问题会进行编号，以便您加以区分。

此操作仅检查模板的语法，而非其内容。编辑器无法获知存在哪些资源类型，也不了解这些资源类型所包含的属性；因此，即便模板的 YAML 格式完全正确，仍可能存在其他问题，这些问题只有 CloudFormation 才能识别。为此，请选择验证。

验证模板

在将模板用于堆栈操作之前，验证过程会通过编辑器检查模板的语法，并利用 CloudFormation 对模板进行核查。这两项检查均使用当前的草稿，包括您尚未保存的更改。即使 CloudFormation 检查通过，也无法覆盖编辑器所报告的语法错误。

> **Note**
>
> 模板工作室使用 `TemplateBody` 参数将草稿发送到 `ValidateTemplate` API。此参数的限制为 51,200 字节（50 KiB）。
> 如果草稿超过此限制，模板工作室不会发送 CloudFormation 请求。CloudFormation 检查不可用，但这并不意味着模板无效。编辑器仍然可以报告语法错误。
> 同一 API 通过 `TemplateURL` 支持最大 1 MB 的模板。模板工作室不会上传已编辑的草稿进行基于 URL 的验证。您可以将模板保存至 Amazon S3，并通过堆栈操作对其进行验证，但需受 CloudFormation 配额限制。

验证您正在编辑的模板

1. 选择验证。
2. 通过两项检查的模板被报告为有效。
3. 如果任一检查发现错误，请在验证消息中查看这些错误。编辑器错误包括行号和列号。请在编辑器中更正这些错误，然后再次进行验证。

将模板保存到 Amazon S3

CloudFormation 会从 Amazon S3 读取用于堆栈操作的模板，因此，保存操作能让您的模板在控制台的其他部分中可用。

在保存的模板中，应使用动态引用来指向存储于 Secrets Manager 中的密钥或 Systems Manager Parameter Store 中的安全字符串，而非直接在模板内包含明文机密密钥。有关更多信息，请参阅 使用动态引用获取存储在其他服务中的值。

保存您正在编辑的模板

1. 选择 保存并继续。
2. 检查目的地。默认情况下，模板将转到 CloudFormation 为当前 Amazon Web Services 区域中上传的模板管理的 Amazon S3 存储桶。如果该存储桶尚不存在，CloudFormation 会在您保存时创建它，并将其重新用于您稍后保存的模板。
   - 若要自行选择目的地，请选择使用其他存储桶并输入存储桶名称。如果存储桶尚不存在，则会创建该存储桶。
3. 选择确认并继续。

> **Note**
>
> 存储在 Amazon S3 中的模板将按照 Amazon S3 的标准费率计费（无论其存放在哪个存储桶中）。有关更多信息，请参阅 Amazon S3 定价。

保存操作无需验证结果通过。若验证发现错误，模板工作室会要求您确认是否仍要保存。保存操作并不会修复这些错误；在将该模板用于堆栈操作之前，请先检查并更正这些错误。

复制或下载模板或示意图

选择模板操作以将模板或示意图从控制台中导出：

复制模板

将编辑器中当前显示的模板复制到剪贴板。

下载模板

将模板保存为文件，所用格式与您当前编辑时所用的格式一致。

下载资源示意图

将示意图另存为 PNG 图像。由于图像是示意图的捕获，因此仅当示意图显示在屏幕上时才可用。

使用 IaC 生成器为现有资源生成模板

借助 CloudFormation 基础设施即代码生成器（IaC 生成器），您可以使用账户中已预置但尚未由 CloudFormation 管理的 Amazon 资源生成模板。

IaC 生成器的优点如下：

- 将整个应用程序置于 CloudFormation 管理之下，或者将其迁移到 Amazon CDK 应用程序中。
- 生成模板时无需逐个描述资源属性，然后将其转换为 JSON 或 YAML 语法。
- 使用该模板在新账户或区域中复制资源。

IaC 生成过程包括以下步骤：

1. 扫描资源 – 第一步是启动资源扫描。该扫描为全区域扫描，持续 30 天。在此期间，您可以根据同一扫描创建多个模板。
2. 创建模板 – 您可以通过两个选项来创建模板：
   - 从头开始创建新模板并将扫描的资源和相关资源添加到模板中。
   - 以现有的 CloudFormation 堆栈作为起点，并将扫描的资源和相关资源添加到其模板中。
3. 导入资源 – 使用您的模板，将资源作为 CloudFormation 堆栈导入或将其迁移到 Amazon CDK 应用程序中。

IaC 生成器功能适用于所有商业区域，并支持许多常见的 Amazon 资源类型。有关受支持资源的完整列表，请参阅资源类型支持。

主题

- 注意事项
- 扫描资源所需的 IAM 权限
- 用于模板生成、管理和删除的常用命令
- 将模板迁移到 Amazon CDK
- 使用 CloudFormation IaC 生成器启动资源扫描
- 在 CloudFormation 控制台中查看扫描摘要
- 使用 IaC 生成器根据扫描的资源创建 CloudFormation 模板
- 根据扫描的资源创建 CloudFormation 堆栈
- 解决只写属性问题

注意事项

您可以为您具有读取权限的 Amazon 资源生成 JSON 或 YAML 模板。IaC 生成器功能模板晋江可靠、快速地对云资源进行建模，而无需按属性描述资源属性。

下表列出了 IaC 生成功能的可用限额。

名称	完全扫描	部分扫描
一次扫描可以处理的最大资源数	100000	100000
每天的扫描次数（适用于少于 10,000 个资源的扫描）	10	10
每天的扫描次数（适用于超过 10,000 个资源的扫描）	1	1
每个账户生成的并发模板数量	5	5
为一次模板生成建模的并发资源数量	5	5
可以在一个模板中建模的资源总数	500	500
每账户最大生成的模板数	1000	1000

> **Important**
>
> IaC 生成器仅支持您所在区域的 Cloud Control API 支持的 Amazon 资源。有关更多信息，请参阅 资源类型支持。

扫描资源所需的 IAM 权限

要使用 IaC 生成器扫描资源，您的 IAM 主体（用户、角色或组）必须具备：

- CloudFormation 扫描权限
- 目标 Amazon 服务的读取权限

扫描范围仅限于您有读取权限的资源。缺少权限不会导致扫描失败，但会排除那些资源。

有关授予扫描和模板管理权限的 IAM 策略示例，请参阅 允许所有 IaC 生成器操作。

用于模板生成、管理和删除的常用命令

使用 IaC 生成器的常用命令包括：

- start-resource-scan，用于启动扫描该账户在 Amazon Web Services 区域中的资源。
- describe-resource-scan，用于监控资源扫描的进度。
- list-resource-scans，用于列出 Amazon Web Services 区域中的已扫描资源。
- list-resource-scan-resources，用于列出在资源扫描期间找到的资源。
- list-resource-scan-related-resources，用于列出与您扫描的资源相关的资源。
- create-generated-template，用于根据一组扫描的资源生成 CloudFormation 模板。
- update-generated-template，用于更新生成的模板。
- describe-generated-template，用于返回有关已生成模板的信息。
- list-generated-templates，用于列出您的账户和当前区域中所有生成的模板。
- delete-generated-template，用于删除生成的模板。

将模板迁移到 Amazon CDK

Amazon Cloud Development Kit (Amazon CDK) 是一个开源软件开发框架，您可以通过该框架使用流行的编程语言开发、管理和部署 CloudFormation 资源。

Amazon CDK CLI 提供了与 IaC 生成器的集成。使用 Amazon CDK CLI `cdk migrate` 命令生成 CloudFormation 模板并创建包含您的资源的新 CDK 应用程序。然后，您可以使用 Amazon CDK 管理您的资源并部署到 CloudFormation。

有关更多信息，请参阅《Amazon Cloud Development Kit (Amazon CDK) 开发人员指南》中的迁移到 Amazon CDK。

使用 CloudFormation IaC 生成器启动资源扫描

在从现有资源创建模板之前，您必须首先启动资源扫描以发现当前资源及其关系。

您可以使用以下选项之一启动资源扫描。对于首次使用 IaC 生成器的用户，我们推荐第一个选项。

- 扫描所有资源（完全扫描）- 扫描当前账户和区域中的所有现有资源。对于 1,000 个资源，此扫描过程最多可能需要 10 分钟。
- 扫描特定资源（部分扫描）– 在当前账户和区域中手动选择要扫描的资源类型。此选项提供更快、更集中的扫描过程，使其成为迭代模板开发的理想选择。

扫描完成后，您可以选择生成模板时要包括哪些资源及其相关资源。使用部分扫描时，相关资源仅在以下任一情况下在模板生成期间可用：

- 您在开始扫描之前特意选择了它们，或者
- 它们是发现所选资源类型所必需的。

例如，如果您选择 AWS::EKS::Nodegroup 而不选择 AWS::EKS::Cluster，IaC 生成器会自动在扫描中包括 AWS::EKS::Cluster 资源，因为发现节点组需要先发现集群。在所有其他情况下，扫描将仅包含您专门选择的资源。

> **Note**
>
> 在继续操作之前，请确认您拥有使用 IaC 生成器所需的权限。有关更多信息，请参阅 扫描资源所需的 IAM 权限。

主题

- 启动资源扫描（控制台）
- 启动资源扫描（Amazon CLI）

启动资源扫描（控制台）

启动所有资源类型的资源扫描（完全扫描）

1. 打开 CloudFormation 控制台的 IaC 生成器页面。
2. 在屏幕顶部的导航栏上，选择包含要扫描的资源的 Amazon Web Services 区域。
3. 从扫描面板中，选择启动新扫描，然后选择扫描所有资源。

启动特定资源类型的资源扫描（部分扫描）

1. 打开 CloudFormation 控制台的 IaC 生成器页面。
2. 在屏幕顶部的导航栏上，选择包含要扫描的资源的 Amazon Web Services 区域。
3. 从扫描面板中，选择开始新扫描，然后选择扫描特定资源。
4. 在启动部分扫描对话框中，最多选择 100 种资源类型，然后选择启动扫描。

启动资源扫描（Amazon CLI）

启动所有资源类型的资源扫描（完全扫描）

使用以下 start-resource-scan 命令。将 *us-east-1* 替换为包含要扫描的资源的 Amazon Web Services 区域。

```
aws cloudformation start-resource-scan --region us-east-1
```

如果成功，该命令会返回该扫描的 ARN。记下 ResourceScanId 属性中的 ARN。创建模板将会需要该数据。

```
{
    "ResourceScanId":
      "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60"
}
```

启动特定资源类型的资源扫描（部分扫描）

1. 使用以下 cat 命令将要扫描的资源类型存储在主目录中名为 config.json 的 JSON 文件中。下面是扫描 Amazon EC2 实例、安全组和所有 Amazon S3 资源的示例扫描配置。

   ```
   $ cat > config.json
   [
     {
       "Types":[
         "AWS::EC2::Instance",
         "AWS::EC2::SecurityGroup",
         "AWS::S3::*"
       ]
     }
   ]
   ```

2. 使用带有 --scan-filters 选项的 start-resource-scan 命令以及您创建 config.json 的文件来启动部分扫描。将 *us-east-1* 替换为包含要扫描的资源的 Amazon Web Services 区域。

   ```
   aws cloudformation start-resource-scan --scan-filters file://config.json --region us-east-1
   ```

   如果成功，该命令会返回该扫描的 ARN。记下 ResourceScanId 属性中的 ARN。创建模板将会需要该数据。

   ```
   {
   ```

```
    "ResourceScanId":
      "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-
a3ef-3e6ecfa5da60"
}
```

监控资源扫描的进度

使用 describe-resource-scan 命令。对于 `--resource-scan-id` 选项，请将示例 ARN 替换为实际的 ARN。

```
aws cloudformation describe-resource-scan --region us-east-1 \
  --resource-scan-id arn:aws:cloudformation:us-
east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60
```

如果成功，该命令返回类似以下内容的输出：

```
{
    "ResourceScanId": "arn:aws:cloudformation:region:account-
id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60",
    "Status": "COMPLETE",
    "StartTime": "2023-08-21T03:10:38.485000+00:00",
    "EndTime": "2023-08-21T03:20:28.485000+00:00",
    "PercentageCompleted": 100.0,
    "ResourceTypes": [
        "AWS::CloudFront::CachePolicy",
        "AWS::CloudFront::OriginRequestPolicy",
        "AWS::EC2::DHCPOptions",
        "AWS::EC2::InternetGateway",
        "AWS::EC2::KeyPair",
        "AWS::EC2::NetworkAcl",
        "AWS::EC2::NetworkInsightsPath",
        "AWS::EC2::NetworkInterface",
        "AWS::EC2::PlacementGroup",
        "AWS::EC2::Route",
        "AWS::EC2::RouteTable",
        "AWS::EC2::SecurityGroup",
        "AWS::EC2::Subnet",
        "AWS::EC2::SubnetCidrBlock",
        "AWS::EC2::SubnetNetworkAclAssociation",
        "AWS::EC2::SubnetRouteTableAssociation",
        ...
    ],
```

```
    "ResourcesRead": 676
}
```

对于部分扫描，输出将类似于以下内容：

```
{
    "ResourceScanId": "arn:aws:cloudformation:region:account-
id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60",
    "Status": "COMPLETE",
    "StartTime": "2025-03-06T18:24:19.542000+00:00",
    "EndTime": "2025-03-06T18:25:23.142000+00:00",
    "PercentageCompleted": 100.0,
    "ResourceTypes": [
        "AWS::EC2::Instance",
        "AWS::EC2::SecurityGroup",
        "AWS::S3::Bucket",
        "AWS::S3::BucketPolicy"
    ],
    "ResourcesRead": 65,
    "ScanFilters": [
        {
            "Types": [
                "AWS::EC2::Instance",
                "AWS::EC2::SecurityGroup",
                "AWS::S3::*"
            ]
        }
    ]
}
```

有关输出中字段的描述，请参阅《Amazon CloudFormation API 参考》中的 DescribeResourceScan。

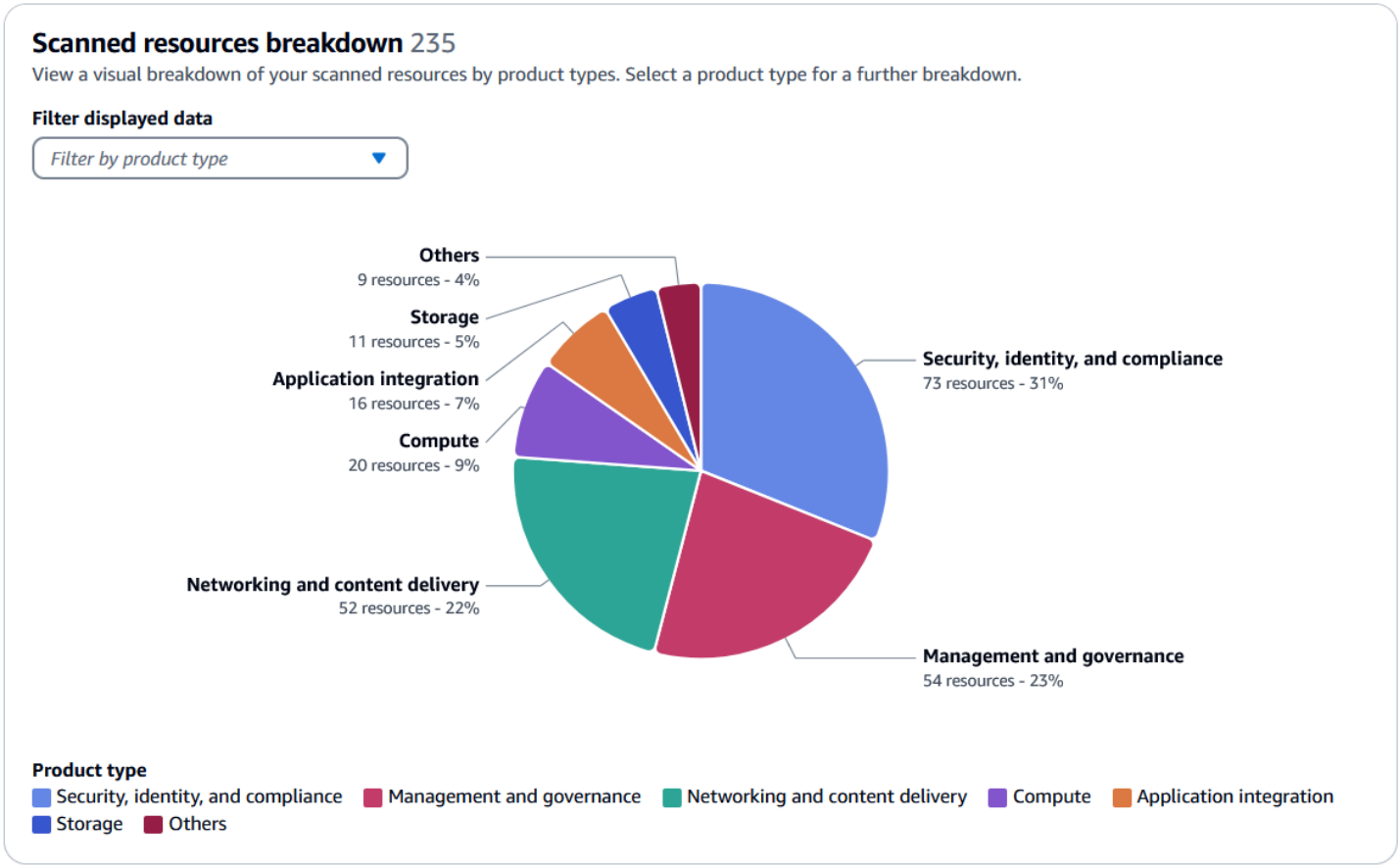
在 CloudFormation 控制台中查看扫描摘要

扫描完成后，您可以通过可视化界面查看扫描期间发现的资源，以帮助确定不同产品类型的资源集中情况。

查看有关扫描期间所发现资源的信息

1. 打开 CloudFormation 控制台的 IaC 生成器页面。
2. 在屏幕顶部的导航栏上，选择包含要查看的资源扫描的 Amazon Web Services 区域。
3. 从导航窗格中，选择 IaC 生成器。

4. 在已扫描资源明细下，您可以通过可视化界面查看按产品类型（例如计算和存储）划分的已扫描资源明细。
5. 要自定义显示的产品类型数量，请选择筛选显示的数据。这有助您自定义可视化效果，从而聚焦您最感兴趣的产品类型。
6. 页面右侧为扫描摘要详细信息面板。要打开该面板，请选择打开面板图标。



使用 IaC 生成器根据扫描的资源创建 CloudFormation 模板

本主题介绍了如何使用 IaC 生成器功能，根据扫描的资源创建模板。

根据扫描的资源创建模板（控制台）

根据扫描的资源创建堆栈模板

1. 打开 CloudFormation 控制台的 IaC 生成器页面。
2. 在屏幕顶部的导航栏上，选择包含已扫描资源的 Amazon Web Services 区域。
3. 从模板部分中，选择创建模板。

4. 选择从新模板开始。

   a. 对于模板名称，请提供模板的名称。

   b. （可选）配置您的删除策略和更新替换策略。

   c. 选择下一步将扫描的资源添加到模板中。

5. 对于添加扫描的资源，浏览扫描的资源列表，然后选择要添加到模板中的资源。您可以按资源标识符、资源类型或标签筛选资源。筛选器是互相包容的。

6. 将所有需要的资源添加到模板后，选择下一步以退出添加扫描的资源页面，然后进入添加相关资源页面。

7. 查看推荐的相关资源列表。诸如 Amazon EC2 实例和安全组之类的相关资源是相互依赖的，通常属于同一个工作负载。选择要包含在生成的模板中的相关资源。

   > **Note**
   >
   > 建议您将所有相关资源添加到此模板中。

8. 查看模板详细信息、扫描的资源和相关资源。

9. 选择创建模板退出查看并创建页面并创建模板。

根据扫描的资源创建模板（Amazon CLI）

根据扫描的资源创建堆栈模板

1. 使用 list-resource-scan-resources 命令来列出在扫描期间找到的资源，还可选择指定 `--resource-identifier` 选项来限制输出范围。对于 `--resource-scan-id` 选项，请将示例 ARN 替换为实际的 ARN。

   ```
   aws cloudformation list-resource-scan-resources \
     --resource-scan-id arn:aws:cloudformation:us-
   east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60 \
     --resource-identifier MyApp
   ```

   示例响应如下，其中 `ManagedByStack` 指示 CloudFormation 是否已管理该资源。复制输出。您在下一个步骤中需要用到它。

   ```
   {
       "Resources": [
   ```

```
        {
            "ResourceType": "AWS::EKS::Cluster",
            "ResourceIdentifier": {
                "ClusterName": "MyAppClusterName"
            },
            "ManagedByStack": false
        },
        {
            "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
            "ResourceIdentifier": {
                "AutoScalingGroupName": "MyAppASGName"
            },
            "ManagedByStack": false
        }
    ]
}
```

有关输出中字段的描述，请参阅《Amazon CloudFormation API 参考》中的 ScannedResource。

2. 使用 `cat` 命令将资源类型和标识符存储到您主目录中名为 `resources.json` 的 JSON 文件中。以下是基于上一步中示例输出的示例 JSON。

```
$ cat > resources.json
[
    {
        "ResourceType": "AWS::EKS::Cluster",
        "ResourceIdentifier": {
            "ClusterName": "MyAppClusterName"
        }
    },
    {
        "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
        "ResourceIdentifier": {
            "AutoScalingGroupName": "MyAppASGName"
        }
    }
]
```

3. 使用 list-resource-scan-related-resources 命令以及您创建的 `resources.json` 文件来列出与您扫描的资源相关的资源。

```
aws cloudformation list-resource-scan-related-resources \
```

```
  --resource-scan-id arn:aws:cloudformation:us-
east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60 \
  --resources file://resources.json
```

示例响应如下，其中 ManagedByStack 指示 CloudFormation 是否已管理该资源。将这些资源添加到您在上一步中创建的 JSON 文件中。您需要这些输出来创建模板。

```
{
    "RelatedResources": [
        {
            "ResourceType": "AWS::EKS::Nodegroup",
            "ResourceIdentifier": {
                "NodegroupName": "MyAppNodegroupName"
            },
            "ManagedByStack": false
        },
        {
            "ResourceType": "AWS::IAM::Role",
            "ResourceIdentifier": {
                "RoleId": "arn:aws::iam::account-id:role/MyAppIAMRole"
            },
            "ManagedByStack": false
        }
    ]
}
```

有关输出中字段的描述，请参阅《Amazon CloudFormation API 参考》中的 ScannedResource。

> **Note**
>
> 资源输入列表的长度不能超过 100。如果要列出的资源数超过了 100 个，请按照一个批次 100 个资源的方式，来运行 list-resource-scan-related-resources 命令并合并结果。请注意，输出列表中可能包含重复的资源。

4. 使用 create-generated-template 命令来创建新的堆栈模板，并进行下列修改：

   - 将 *us-east-1* 替换为包含已扫描资源的 Amazon Web Services 区域。
   - 将 *MyTemplate* 替换为要创建的模板的名称。

```
aws cloudformation create-generated-template --region us-east-1 \
```

```
 --generated-template-name MyTemplate \
  --resources file://resources.json
```

下面是一个 resources.json 示例文件。

```
[
    {
        "ResourceType": "AWS::EKS::Cluster",
        "LogicalResourceId":"MyCluster",
        "ResourceIdentifier": {
            "ClusterName": "MyAppClusterName"
        }
    },
    {
        "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
        "LogicalResourceId":"MyASG",
        "ResourceIdentifier": {
            "AutoScalingGroupName": "MyAppASGName"
        }
    },
    {
        "ResourceType": "AWS::EKS::Nodegroup",
        "LogicalResourceId":"MyNodegroup",
        "ResourceIdentifier": {
            "NodegroupName": "MyAppNodegroupName"
        }
    },
    {
        "ResourceType": "AWS::IAM::Role",
        "LogicalResourceId":"MyRole",
        "ResourceIdentifier": {
            "RoleId": "arn:aws::iam::account-id:role/MyAppIAMRole"
        }
    }
]
```

如果成功，该命令会返回以下响应。

```
{
  "Arn":
    "arn:aws:cloudformation:region:account-id:generatedtemplate/7fc8512c-d8cb-4e02-
b266-d39c48344e48",
```

```
  "Name": "MyTemplate"
}
```

根据扫描的资源创建 CloudFormation 堆栈

创建模板后，请使用画布视图预览其中的资源及其关系，然后再创建堆栈并导入已扫描的资源。

创建堆栈并导入已扫描资源

1. 打开 CloudFormation 控制台的 IaC 生成器页面。
2. 在屏幕顶部的导航栏中，选择模板所在的 Amazon Web Services 区域。
3. 选择模板选项卡，然后选择模板名称以查看更多信息。
4. 在模板定义选项卡中模板部分的顶部，您可以根据自己的喜好将模板从 YAML 语法切换为 JSON 语法。
5. 检查模板的详细信息，以确保所有设置都正确无误。为更便于检查和理解模板，您可以从默认的代码视图切换到图形视图来查看模板中描述的基础设施。要执行此操作，请在模板下选择画布，而不是模板。

   画布操作

   - 要聚焦模板中特定资源的详细信息，请双击卡片以打开资源属性面板。
   - 要在画布上直观地排列和整理卡片，请在画布的左上角选择排列。
   - 要放大和缩小画布，请使用画布右下角的缩放控件。
6. 要在控制台中查看特定资源，请选择模板资源选项卡，然后选择要查看的资源的物理 ID。这时您将进入该特定资源的控制台。您也可以通过模板资源选项卡在模板定义中添加、移除和重新同步资源。
7. 在模板定义选项卡中，IaC 生成器可能会对包含只写属性的资源发出警告。查看警告后，您可以下载生成的模板并进行任何必要的更改。有关更多信息，请参阅 解决只写属性问题。
8. 如果模板定义符合要求，请在模板定义选项卡中选择导入到堆栈，然后选择下一步。
9. 在指定堆栈详细信息页面的指定堆栈面板上输入堆栈名称，然后选择下一步。
10. 查看并输入堆栈的参数。选择下一步。
11. 在查看更改页面上查看您的选项，然后选择下一步。
12. 在查看和导入页面上查看您的详细信息，然后选择导入资源。

解决只写属性问题

使用 CloudFormation IaC 生成器，您可以使用账户中已预置但尚未由 CloudFormation 管理的资源生成模板。但是，某些资源属性被指定为只写，这意味着 CloudFormation 可以写入但不能读取这些属性，例如数据库密码。

利用现有资源生成 CloudFormation 模板时，只写属性会带来挑战。通常，CloudFormation 会在生成的模板中将这些属性转换为参数。这允许您在导入操作期间输入属性作为参数值。但在某些情况下无法实现这种转换，而且 CloudFormation 对这些情况的处理方式有所不同。

互斥属性

一些资源有多组互斥属性，其中至少有一些是只写属性。对于这些情况，IaC 生成器无法确定在创建期间对资源应用了哪组互斥属性。例如，您可以使用其中一组属性提供 AWS::Lambda::Function 代码。

- `Code/S3Bucket`、`Code/S3Key` 以及可选的 `Code/S3ObjectVersion`
- `Code/ImageUri`
- `Code/ZipFile`

所有这些属性均为只写属性。IaC 生成器会选择其中一组独占属性并将其添加到生成的模板中。为每个只写属性添加参数。参数名称包括 `OneOf`，参数说明表示相应的属性可以替换为其他独占属性。IaC 生成器为包含的属性设置的警告类型为 `MUTUALLY_EXCLUSIVE_PROPERTIES`。

互斥类型

在某些情况下，只写属性可以为多种数据类型。例如，AWS::ApiGateway::RestApi 的 `Body` 属性可以是 `object` 或 `string`。遇到这种情况时，IaC 生成器会使用 `string` 类型在生成的模板中包含该属性，并将警告类型设置为 `MUTUALLY_EXCLUSIVE_TYPES`。

Array 属性

如果只写属性的类型为 `array`，则 IaC 生成器无法将其包含在生成的模板中，因为参数只能是标量值。在这种情况下，模板中会忽略该属性，并设置 `UNSUPPORTED_PROPERTIES` 警告类型。

可选属性

对于可选的只写属性，IaC 生成器无法检测在设置资源时是否使用过该属性。在这种情况下，生成的模板中会忽略该属性，并设置 `UNSUPPORTED_PROPERTIES` 警告类型。

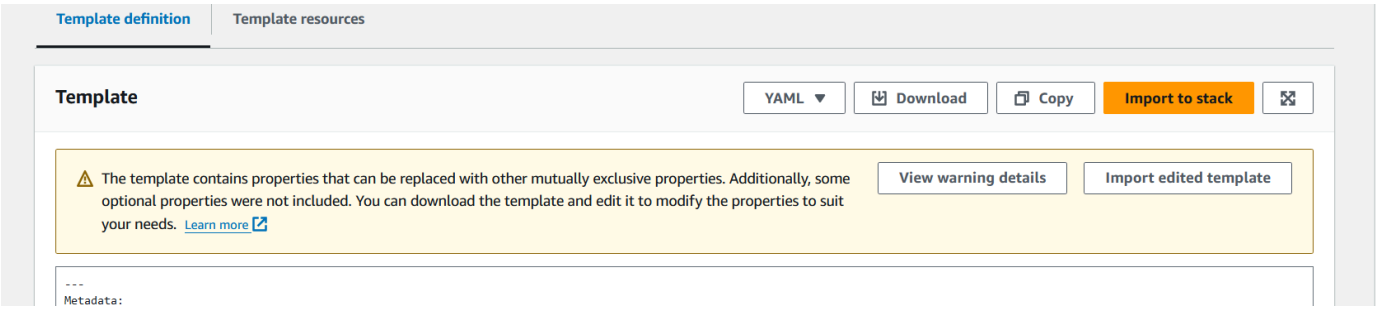
警告和后续步骤

要确定哪些属性为只写属性，您必须查看 IaC 生成器控制台返回的警告。Amazon 资源和属性类型参考并未指示某个属性是否为只写属性，也未指示其是否支持多种类型。

您也可从资源提供程序架构中查看哪些属性为只写属性。要下载资源提供程序架构，请参阅 CloudFormation 资源提供程序架构。

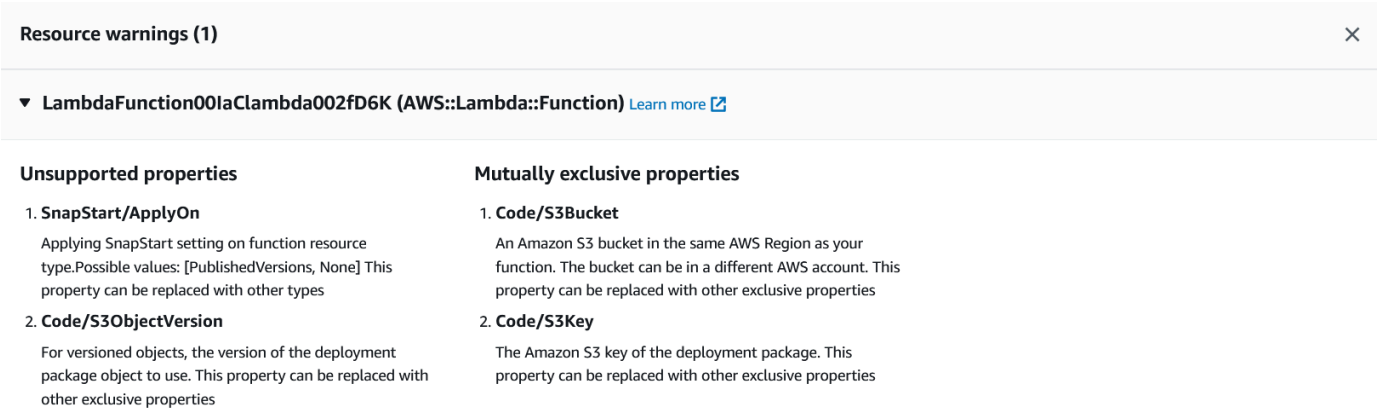
解决只写属性问题

1. 打开 CloudFormation 控制台的 IaC 生成器页面。
2. 在屏幕顶部的导航栏中，选择模板所在的 Amazon Web Services 区域。
3. 选择模板选项卡，然后选择您创建的模板的名称。
4. 在模板定义选项卡中，当生成的模板包含具有只写属性的资源时，IaC 生成器控制台会显示一条警告，其中包含问题类型的摘要。例如：



5. 选择查看警告详细信息以了解更多详情。具有只写属性的资源由生成的模板中使用的逻辑 ID 和资源类型标识。

   使用警告列表来识别具有只写属性的资源，并查看每个资源以确定需要对生成的模板进行哪些更改（如果有）。



6. 如果必须更新模板才能解决只写属性问题，请执行以下操作：

a. 选择下载以下载模板的副本。
b. 编辑模板。
c. 完成更改后，您可以选择导入已编辑的模板按钮以继续完成导入过程。

如何解决 AWS::ApiGateway::RestAPI 资源中的只写属性问题

本主题介绍如何解决在使用 IaC 生成器时的 AWS::ApiGateway::RestApi 资源只写属性问题。

问题

当生成的模板包含 AWS::ApiGateway::RestApi 资源时，会生成警告，指明 Body、BodyS3Location 和 CloneFrom 属性将标识为 UNSUPPORTED_PROPERTIES。这是因为这些是可选的只写属性。IaC 生成器不知道这些属性是否曾应用于资源，因此生成的模板中会忽略这些属性。

解决方案

要为您的 REST API 设置 Body 属性，请更新生成的模板。

1. 使用 Amazon API Gateway GetExport API 操作下载 API。例如，通过使用 aws apigateway get-export Amazon CLI 命令。有关更多信息，请参阅《API Gateway 开发人员指南》中的从 API Gateway 导出 REST API。
2. 从 GetExport API 操作的响应中检索 Body 属性。将其上传到 Amazon S3 存储桶。
3. 下载生成的模板。
4. 将 BodyS3Location/Bucket 和 BodyS3Location/Key 属性添加到模板中，指定存储 Body 的桶名称和密钥。
5. 在 IaC 生成器控制台中打开生成的模板，然后选择导入编辑的模板。

如何解决 AWS::Lambda::Function 资源中的只写属性问题

本主题介绍如何解决在使用 IaC 生成器时的 AWS::Lambda::Function 资源只写属性问题。

问题

AWS::Lambda::Function 资源具有三组互斥属性，用于指定 Lambda 代码：

- Code/S3Bucket 和 Code/S3Key 属性，以及可选的 Code/S3ObjectVersion 属性

- Code/ImageUri 属性
- Code/ZipFile 属性

任何给定的 AWS::Lambda::Function 资源只能使用其中的一组属性。

IaC 生成器无法确定使用了哪组互斥只写属性来创建或更新资源，因此，生成的模板中仅包含第一组属性。Code/ImageUri 和 Code/ZipFile 属性将被忽略。

此外，IaC 生成器会发出以下警告：

- **MUTUALLY_EXCLUSIVE_PROPERTIES** – 警告 Code/S3Bucket 和 Code/S3Key 被标识为互斥属性。
- **UNSUPPORTED_PROPERTIES** – 警告 Code/S3ObjectVersion 属性不受支持。

要在生成的模板中包含 AWS::Lambda::Function 资源，您必须下载模板并使用正确的代码属性更新该模板。

解决方案

如果您将 Lambda 代码存储在 Amazon S3 存储桶中并且不使用 **S3ObjectVersion** 属性，则无需任何修改即可导入生成的模板。在导入操作期间，IaC 生成器将要求您提供 Amazon S3 存储桶和密钥作为模板参数。

如果将 Lambda 代码存储为 Amazon ECR 存储库，则您可以按照以下说明更新您的模板：

1. 下载生成的模板。
2. 从生成的模板中删除 Code/S3Bucket 和 Code/S3Key 属性的属性和相应参数。
3. 将生成的模板中已删除的属性替换为 Code/ImageUri 属性，指定 Amazon ECR 存储库的 URL。
4. 在 IaC 生成器控制台中打开生成的模板，然后选择导入编辑的模板按钮。

如果您将 Lambda 代码存储为 zip 文件，则可以按照以下说明更新您的模板：

1. 下载生成的模板。
2. 从生成的模板中删除 Code/S3Bucket 和 Code/S3Key 属性的属性和相应参数。
3. 将生成的模板中已删除的属性替换为 Code/ZipFile 属性。

4. 在 IaC 生成器控制台中打开生成的模板，然后选择导入编辑的模板按钮。

如果没有 Lambda 代码的副本，则您可以按照以下说明更新您的模板：

1. 使用 Amazon Lambda `GetFunction` API 操作（例如，通过使用 `aws lambda get-function` Amazon CLI 命令）。
2. 在响应中，如果代码位于 Amazon S3 存储桶中，则 `RepositoryType` 参数为 `S3`；如果代码位于 Amazon ECR 存储库中，则该参数为 `ECR`。
3. 在响应中，`Location` 参数包含一个预签名 URL，您可以用其下载部署包 10 分钟。下载代码。
4. 将代码上传到 Amazon S3 存储桶。
5. 使用生成的模板运行导入操作，并提供桶名称和密钥作为参数值。

使用动态引用获取存储在其他服务中的值

动态引用是一种十分便捷的方法，让您可以轻松指定在其他服务中存储和管理的外部值，并将敏感信息与基础设施即代码模板分离。在堆栈和更改集操作期间，CloudFormation 会在必要时检索指定引用的值。

借助动态引用，您可以：

- 使用安全字符串 – 对于敏感数据，始终应使用 Amazon Systems Manager Parameter Store 中的安全字符串参数或 Amazon Secrets Manager 中的密钥来保证数据的静态加密。
- 限制访问权限 – 仅允许获得授权的主体和角色访问 Parameter Store 参数或 Secrets Manager 密钥。
- 轮换凭证 – 定期轮换存储在 Parameter Store 或 Secrets Manager 中的敏感数据，以确保高度的安全性。
- 自动轮换 – 利用 Secrets Manager 的自动轮换功能定期更新和分配各种应用程序和环境中的敏感数据。

一般注意事项

以下是在 CloudFormation 模板中指定动态引用之前的一般注意事项：

- 不要在资源主标识符所包含资源属性中包含动态引用或任何敏感数据。CloudFormation 可能会在主资源标识符中使用实际的纯文本值，这可能存在安全风险。此资源 ID 可能出现在任何派生输出或目标中。

要确定哪些资源属性构成了资源类型的主标识符，请参阅 Amazon 资源和属性类型参考中该资源的资源参考文档。在 Return values（返回值）部分，Ref 函数返回值，表示构成资源类型主标识符的资源属性。

- 您最多可以在堆栈模板中包含 60 个动态引用。
- 如果您使用的是转换（如 AWS::Include 或 AWS::Serverless），则 CloudFormation 在应用转换之前不会解析动态引用，而是将动态引用的明文字符串传递给转换，并在使用模板执行更改集时解析引用。
- 动态引用不能用于自定义资源中的安全值（例如存储在 Parameter Store 或 Secrets Manager 中的值）。
- AWS::CloudFormation::Init 元数据和 Amazon EC2 UserData 属性也不支持动态引用。
- 不要创建以反斜杠（\）结尾的动态引用。CloudFormation 无法解析此类引用，这将导致堆栈操作失败。

以下主题提供了有关使用动态引用的信息和其他注意事项。

主题

- 获取 Systems Manager Parameter Store 中的纯文本值
- 获取 Systems Manager Parameter Store 中的安全字符串值
- 获取 Secrets Manager 中的密钥或密钥值

获取 Systems Manager Parameter Store 中的纯文本值

在创建 CloudFormation 模板时，可能需要使用存储在 Parameter Store 中的纯文本值。Parameter Store 是 Amazon Systems Manager 的一项功能。有关 Parameter Store 的介绍，请参阅《Amazon Systems Manager 用户指南》中的 Amazon Systems Manager Parameter Store。

要在模板中使用 Parameter Store 中的明文值，请使用 ssm 动态引用。此引用让您能够访问 Parameter Store 中 String 或 StringList 类型的参数的值。

要验证堆栈操作中将使用哪个版本的 ssm 动态引用，请为堆栈操作创建更改集。然后，在模板选项卡上查看已处理的模板。有关更多信息，请参阅 创建 CloudFormation 堆栈的更改集。

使用 ssm 动态引用时，需要注意几个重要的事项：

- CloudFormation 不支持对动态引用进行偏差检测。对于尚未指定参数版本的 `ssm` 动态引用，建议在 Systems Manager 中更新参数版本时，还要对包含 `ssm` 动态引用的任何堆栈执行堆栈更新操作，以获取最新的参数版本。
- 要在 CloudFormation 模板的 `Parameters` 部分中使用 `ssm` 动态引用，必须包含版本号。CloudFormation 不允许在这一部分中引用没有版本号的 Parameter Store 值。您也可以在模板中将您的参数定义为某个 Systems Manager 参数类型。执行此操作时，您可以将某个 Systems Manager 参数键指定为参数的默认值。然后，CloudFormation 将从 Parameter Store 中检索最新版本的参数值，而无需您指定版本号。这可以使您的模板更简洁、更易于维护。有关更多信息，请参阅使用 CloudFormation 提供的参数类型在运行时指定现有资源。
- 对于自定义资源，CloudFormation 在将请求发送到自定义资源之前解析 `ssm` 动态引用。
- CloudFormation 不支持使用动态引用来引用另一个 Amazon Web Services 账户共享的参数。
- CloudFormation 不支持在动态引用中使用 Systems Manager 参数标签。

权限

要指定存储在 Systems Manager Parameter Store 中的参数，您必须有权为指定的参数调用 GetParameters。要了解如何创建提供对特定 Systems Manager 参数的访问权限的 IAM 策略，请参阅《Amazon Systems Manager 用户指南》中的使用 IAM 策略限制对 Parameter Store 参数的访问。

引用模式

要在 CloudFormation 模板中引用 Systems Manager Parameter Store 中存储的明文值，请使用以下 `ssm` 引用模式。

```
{{resolve:ssm:parameter-name:version}}
```

您的引用必须遵循以下参数名称和版本的正则表达式模式：

```
{{resolve:ssm:[a-zA-Z0-9_.\-/]+(:\d+)?}}
```

`parameter-name`

Parameter Store 中的参数的名称。参数名称区分大小写。

必需。

`version`

一个整数，指定要使用的参数的版本。如果您没有指定确切的版本，CloudFormation 将在创建或更新堆栈时使用参数的最新版本。有关更多信息，请参阅《Amazon Systems Manager 用户指南》中的使用参数版本。

可选。

示例

主题

- 公用 AMI ID 参数
- 自定义 AMI ID 参数

公用 AMI ID 参数

以下示例创建的 EC2 实例引用了一个公用 AMI 参数。动态引用从公有参数中检索最新版 Amazon Linux 2023 AMI ID。有关公有参数的更多信息，请参阅《Amazon Systems Manager 用户指南》中的发现 Parameter Store 中的公有参数。

JSON

```
{
    "Resources": {
        "MyInstance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64}}",
                "InstanceType": "t2.micro"
            }
        }
    }
}
```

YAML

```
Resources:
  MyInstance:
```

```
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64}}'
      InstanceType: t2.micro
```

自定义 AMI ID 参数

以下示例创建了一个 EC2 启动模板，其中引用了存储在 Parameter Store 中的自定义 AMI ID。每次从启动模板启动实例时，动态引用都会从 *golden-ami* 参数的版本 *2* 中检索 AMI ID。

JSON

```
{
    "Resources": {
        "MyLaunchTemplate": {
            "Type": "AWS::EC2::LaunchTemplate",
            "Properties": {
                "LaunchTemplateName": {
                    "Fn::Sub": "${AWS::StackName}-launch-template"
                },
                "LaunchTemplateData": {
                    "ImageId": "{{resolve:ssm:golden-ami:2}}",
                    "InstanceType": "t2.micro"
                }
            }
        }
    }
}
```

YAML

```
Resources:
  MyLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: '{{resolve:ssm:golden-ami:2}}'
        InstanceType: t2.micro
```

获取 Systems Manager Parameter Store 中的安全字符串值

在 CloudFormation 中，您可以将密码和许可证密钥等敏感数据作为“安全字符串”存储在 Amazon Systems Manager Parameter Store 中，从而在不直接在模板中直接公开的情况下使用这些数据。有关 Parameter Store 的介绍，请参阅《Amazon Systems Manager 用户指南》中的 Amazon Systems Manager Parameter Store。

要在模板中使用 Parameter Store 安全字符串，请使用 `ssm-secure` 动态引用。CloudFormation 永远不会存储实际的安全字符串值，而是仅存储包含安全字符串的纯文本参数名称的明文动态引用。

在创建或更新堆栈期间，CloudFormation 会根据需要访问安全字符串值，而不会暴露实际值。安全字符串只能用于支持 `ssm-secure` 动态引用模式的资源属性。有关更多信息，请参阅 对安全字符串支持动态参数模式的资源。

CloudFormation 不会在任何 API 调用中返回安全字符串的实际参数值，而是仅返回明文动态引用。在使用更改集比较更改时，CloudFormation 仅会比较明文动态引用字符串，而不会解析并比较实际的安全字符串值。

使用 `ssm-secure` 动态引用时，需要注意几个重要的事项：

- CloudFormation 无法访问来自其他 Amazon Web Services 账户的 Parameter Store 值。
- CloudFormation 不支持在动态引用中使用 Systems Manager 参数标签或公有参数。
- 在 `cn-north-1` 和 `cn-northwest-1` 区域，Systems Manager 不支持安全字符串。
- 自定义资源中目前不支持对安全值的动态引用，例如 `ssm-secure`。
- 如果 CloudFormation 需要回滚堆栈更新，同时先前指定的安全字符串参数版本不再可用，则回滚操作将会失败。在这种情况下，您有两个选择：
  - 使用 `CONTINUE_UPDATE_ROLLBACK` 跳过资源。
  - 在 Systems Manager Parameter Store 中重新创建安全字符串参数，并更新它，直到参数版本达到模板中使用的版本。然后使用 `CONTINUE_UPDATE_ROLLBACK` 而不跳过该资源。

对安全字符串支持动态参数模式的资源

支持 `ssm-secure` 动态引用模式的资源包括：

资源	属性类型	属性
AWS::DirectoryService::MicrosoftAD		Password
AWS::DirectoryService::SimpleAD		Password
AWS::ElastiCache::ReplicationGroup		AuthToken
AWS::IAM::User	LoginProfile	Password
AWS::KinesisFirehose::DeliveryStream	RedshiftDestinationConfiguration	Password
AWS::OpsWorks::App	源	Password
AWS::OpsWorks::Stack	CustomCookbooksSource	Password
AWS::OpsWorks::Stack	RdsDbInstances	DbPassword
AWS::RDS::DBCluster		MasterUserPassword
AWS::RDS::DBInstance		MasterUserPassword
AWS::Redshift::Cluster		MasterUserPassword

引用模式

要在 CloudFormation 模板中引用 Systems Manager 参数存储中的安全字符串值，请使用以下 `ssm-secure` 引用模式。

```
{{resolve:ssm-secure:parameter-name:version}}
```

您的引用必须遵循以下参数名称和版本的正则表达式模式：

```
{{resolve:ssm-secure:[a-zA-Z0-9_.\-/]+(:\d+)?}}
```

`parameter-name`

Parameter Store 中的参数的名称。参数名称区分大小写。

必需。

`version`

一个整数，指定要使用的参数的版本。如果您没有指定确切的版本，CloudFormation 将在创建或更新堆栈时使用参数的最新版本。有关更多信息，请参阅《Amazon Systems Manager 用户指南》中的使用参数版本。

可选。

示例

以下示例使用 `ssm-secure` 动态引用将 IAM 用户的密码设置为存储在 Parameter Store 中的安全字符串。如指定的那样，CloudFormation 将使用 *`IAMUserPassword`* 参数的版本 *`10`* 进行堆栈和更改集操作。

JSON

```
"MyIAMUser": {
  "Type": "AWS::IAM::User",
  "Properties": {
    "UserName": "MyUserName",
    "LoginProfile": {
      "Password": "{{resolve:ssm-secure:IAMUserPassword:10}}"
    }
  }
}
```

YAML

```
MyIAMUser:
  Type: AWS::IAM::User
  Properties:
    UserName: 'MyUserName'
    LoginProfile:
      Password: '{{resolve:ssm-secure:IAMUserPassword:10}}'
```

获取 Secrets Manager 中的密钥或密钥值

Secrets Manager 是一项便于您安全存储和管理密钥的服务，例如数据库凭证、密码和第三方 API 密钥等。使用 Secrets Manager，您可以集中存储和控制对这些密钥的访问，以便使用对 Secrets Manager 的 API 调用来替换代码中的硬编码凭证（包括密码），以编程方式检索密钥。有关更多信息，请参阅《Amazon Secrets Manager 用户指南》中的什么是 Amazon Secrets Manager？。

要在 CloudFormation 模板中使用 Secrets Manager 中存储的全部密钥或密钥值，请使用 `secretsmanager` 动态引用。

最佳实践

在 CloudFormation 模板中使用 Secrets Manager 动态引用时，请遵循以下最佳实践：

- 对 CloudFormation 模板使用无版本引用：将凭证存储在 Secrets Manager 中并使用动态引用而不指定 `version-stage` 或 `version-id` 参数，以支持正确的密钥轮换工作流程。
- 利用自动轮换：使用 Secrets Manager 的自动轮换功能，结合无版本动态引用进行凭证管理。这可以确保您的凭证得到定期更新，而无需更改模板。有关更多信息，请参阅轮换 Amazon Secrets Manager 密钥。
- 谨慎使用带版本引用：仅为测试或回滚等特定场景指定显式的 `version-stage` 或 `version-id` 参数。

注意事项

使用 `secretsmanager` 动态引用时，需要记住以下重要注意事项：

- CloudFormation 不会追踪之前的部署中使用了哪个版本的密钥。在实现动态引用之前，请认真规划您的密钥管理策略。尽量使用无版本引用，以利用自动密钥轮换功能。在更改动态引用配置时（例如从无版本动态引用改换为带版本动态引用，反之亦然）时，监控和验证资源更新。
- 仅更新 Secrets Manager 中的密钥值不会自动触发 CloudFormation 检索新值。CloudFormation 仅在创建或更新资源修改包含动态引用的资源时检索密钥值。

  例如，假设您的模板包含 AWS::RDS::DBInstance 资源，其中 `MasterPassword` 属性设置为 Secrets Manager 动态引用。使用此模板创建堆栈后，需要在 Secrets Manager 中更新密钥值。但是，`MasterPassword` 属性保留了旧密码值。

  要应用新密钥值，需修改 CloudFormation 模板中的 `AWS::RDS::DBInstance` 资源并执行堆栈更新。

为避免将来执行这种手动过程，可考虑使用 Secrets Manager 自动轮换密钥。

- 自定义资源中目前不支持对安全值的动态引用，例如 `secretsmanager`。
- `secretsmanager` 动态引用可用于所有资源属性。使用 `secretsmanager` 动态引用表示 Secrets Manager 和 CloudFormation 日志均不应保留任何已解析的密钥值。但是，密钥值可能会显示在正在其资源中使用的服务中。检查您的使用情况，以避免泄露密钥数据。

权限

要指定存储在 Secrets Manager 中的密钥，您必须有权为该密钥调用 GetSecretValue。

引用模式

要在 CloudFormation 模板中引用 Secrets Manager 密钥，请使用以下 `secretsmanager` 引用模式。

```
{{resolve:secretsmanager:secret-id:secret-string:json-key:version-stage:version-id}}
```

`secret-id`

密钥的名称或 ARN。

要访问您的 Amazon Web Services 账户中的密钥，只需指定密钥名称即可。要访问其他 Amazon Web Services 账户中的密钥，请指定密钥的完整 ARN。

必需。

`secret-string`

`SecretString` 是唯一受支持的值。默认值为 `SecretString`。

`json-key`

要检索其值的键值对的键名称。如果您不指定 `json-key`，CloudFormation 会检索整个密钥文本。

此分段不得包含冒号字符 (:)。

`version-stage`

要使用的密钥版本的暂存标签。Secrets Manager 在轮换过程中使用暂存标注来跟踪不同的版本。如果您使用 `version-stage`，则不要指定 `version-id`。如果您既未指定 `version-stage`，也未指定 `version-id`，则原定设置将为 `AWSCURRENT` 版本。

此分段不得包含冒号字符 (:)。

`version-id`

要使用的密钥版本的唯一标识符。如果指定 `version-id`，则不要指定 `version-stage`。如果您既未指定 `version-stage`，也未指定 `version-id`，则原定设置将为 `AWSCURRENT` 版本。

此分段不得包含冒号字符 (:)。

示例

主题

- 从密钥中检索用户名和密码值
- 检索整个 SecretString
- 从密钥的特定版本中检索值
- 从另一个 Amazon Web Services 账户检索密钥

从密钥中检索用户名和密码值

以下 AWS::RDS::DBInstance 示例检索在 *MySecret* 密钥中存储的用户名和密码值。此示例展示了无版本动态引用的推荐模式，该模式自动使用 `AWSCURRENT` 版本且支持 Secrets Manager 轮换工作流程，而无需更改模板。

JSON

```
{
    "MyRDSInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "DBName": "MyRDSInstance",
            "AllocatedStorage": "20",
            "DBInstanceClass": "db.t2.micro",
            "Engine": "mysql",
            "MasterUsername":
 "{{resolve:secretsmanager:MySecret:SecretString:username}}",
            "MasterUserPassword":
 "{{resolve:secretsmanager:MySecret:SecretString:password}}"
        }
    }
```

```
}
```

YAML

```
  MyRDSInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      DBName: MyRDSInstance
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MySecret:SecretString:username}}'
      MasterUserPassword: '{{resolve:secretsmanager:MySecret:SecretString:password}}'
```

检索整个 SecretString

以下动态引用检索 *MySecret* 的 SecretString。

```
{{resolve:secretsmanager:MySecret}}
```

或者：

```
{{resolve:secretsmanager:MySecret::::}}
```

从密钥的特定版本中检索值

以下动态引用检索 *MySecret* 的 *AWSPREVIOUS* 版本的 *password* 值。

```
{{resolve:secretsmanager:MySecret:SecretString:password:AWSPREVIOUS}}
```

从另一个 Amazon Web Services 账户检索密钥

以下动态引用检索另一个 Amazon Web Services 账户中的 *MySecret* 的 SecretString。您必须指定完整的密钥 ARN 才能访问其他 Amazon Web Services 账户中的密钥。

```
{{resolve:secretsmanager:arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret}}
```

以下动态引用检索另一个 Amazon Web Services 账户中的 *MySecret* 的 *password* 值。您必须指定完整的密钥 ARN 才能访问其他 Amazon Web Services 账户中的密钥。

```
{{resolve:secretsmanager:arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret:SecretString:password}}
```

使用伪参数获取 Amazon 值

伪参数是内置变量，用于提供对重要 Amazon 环境信息的访问权限，例如可能因部署或环境的不同而发生变化的账户 ID、区域名称和堆栈详细信息等。

您可以使用伪参数代替硬编码值，以此确保模板更便于移植，更易于跨不同 Amazon Web Services 账户和区域重复使用。

语法

您可以使用 `Ref` 内置函数或 `Fn::Sub` 内部函数来引用伪参数。

Ref

`Ref` 内置函数使用以下常规语法。有关更多信息，请参阅 Ref。

JSON

```
{ "Ref" : "AWS::PseudoParameter" }
```

YAML

```
!Ref AWS::PseudoParameter
```

Fn::Sub

`Fn::Sub` 内置函数使用不同的格式，其中包括伪参数周围的 `${}` 语法。有关更多信息，请参阅 Fn::Sub。

JSON

```
{ "Fn::Sub" : "${AWS::PseudoParameter}" }
```

YAML

```
!Sub '${AWS::PseudoParameter}'
```

可用伪参数

AWS::AccountId

返回在其中创建堆栈的账户的 Amazon Web Services 账户 ID，如 123456789012。

此伪参数通常用于定义 IAM 角色、策略和其他涉及账户特定 ARN 的资源策略。

AWS::NotificationARNs

返回接收堆栈事件通知的 Amazon SNS 主题的 Amazon 资源名称（ARN）列表。在创建或更新堆栈时，您可以通过 Amazon CLI 中的 `--notification-arns` 选项或通过控制台指定这些 ARN。

与返回单个值的其他伪参数不同，`AWS::NotificationARNs` 会返回 ARN 列表。要访问列表中的特定 ARN，使用 `Fn::Select` 内置函数。有关更多信息，请参阅 Fn::Select。

AWS::NoValue

如果在 `Fn::If` 内部函数中指定为返回值，则删除相应的资源属性。有关更多信息，请参阅 Fn::If。

如果需要创建只有在特定条件下才会被包含的条件资源属性，此伪参数特别有用。

AWS::Partition

返回资源所处的分区。对于标准 Amazon Web Services 区域，分区是 `aws`。对于位于其他分区中的资源，则分区是 `aws-`*`partitionname`*。例如，中国（北京和宁夏）区域中的资源的分区为 `aws-cn`，Amazon GovCloud（US-West）区域中的资源的分区为 `aws-us-gov`。

分区是资源 ARN 的一部分。使用 `AWS::Partition` 可确保模板在不同的 Amazon 分区中正常运行。

AWS::Region

返回代表在其中创建包容性资源的区域的字符串，如 `us-west-2`。

这是最常用的伪参数之一，因为它能让模板适应不同的 Amazon Web Services 区域，无需另行修改。

AWS::StackId

返回堆栈 ID（ARN），例如 `arn:aws:cloudformation:us-west-2:123456789012:stack/teststack/51af3dc0-da77-11e4-872e-1234567db123`。

AWS::StackName

返回堆栈名称，例如 teststack。

堆栈名称通常用于创建唯一资源名称，方便轻松识别出相应堆栈。

AWS::URLSuffix

返回堆栈部署所在的 Amazon Web Services 区域中的 Amazon 域后缀。后缀通常为 amazonaws.com，但中国（北京）区域的后缀为 amazonaws.com.cn。

在为 Amazon 服务端点构建 URL 时，此参数特别有用。

示例

主题

- 基本用法
- 使用 AWS::NotificationARNs
- AWS::NoValue 条件属性

基本用法

以下示例创建了两个资源：一个 Amazon SNS 主题和一个向该主题发送通知的 CloudWatch 警报。它们使用 AWS::StackName、AWS::Region 和 AWS::AccountId 将堆栈名称、当前 Amazon Web Services 区域和账户 ID 动态插入到资源名称、描述和 ARN 中。

JSON

```
{
    "Resources": {
        "MyNotificationTopic": {
            "Type": "AWS::SNS::Topic",
            "Properties": {
                "DisplayName": { "Fn::Sub": "Notifications for ${AWS::StackName}" }
            }
        },
        "CPUAlarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": { "Fn::Sub": "Alarm for high CPU in
 ${AWS::Region}" },
```

```
                "AlarmName": { "Fn::Sub": "${AWS::StackName}-HighCPUAlarm" },
                "MetricName": "CPUUtilization",
                "Namespace": "AWS/EC2",
                "Statistic": "Average",
                "Period": 300,
                "EvaluationPeriods": 1,
                "Threshold": 80,
                "ComparisonOperator": "GreaterThanThreshold",
                "AlarmActions": [{ "Fn::Sub": "arn:aws:sns:${AWS::Region}:
${AWS::AccountId}:${MyNotificationTopic}" }]
            }
        }
    }
}
```

YAML

```
Resources:
  MyNotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: !Sub Notifications for ${AWS::StackName}
  CPUAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: !Sub Alarm for high CPU in ${AWS::Region}
      AlarmName: !Sub ${AWS::StackName}-HighCPUAlarm
      MetricName: CPUUtilization
      Namespace: AWS/EC2
      Statistic: Average
      Period: 300
      EvaluationPeriods: 1
      Threshold: 80
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - !Sub arn:aws:sns:${AWS::Region}:${AWS::AccountId}:${MyNotificationTopic}
```

使用 AWS::NotificationARNs

以下示例配置了一个自动扩缩组来发送有关实例启动事件和启动错误的通知。该配置使用 AWS::NotificationARNs 伪参数，提供堆栈创建期间指定的 Amazon SNS 主题 ARN 的列表。Fn::Select 函数会从该列表中选择第一个 ARN。

JSON

```
"myASG": {
   "Type": "AWS::AutoScaling::AutoScalingGroup",
   "Properties": {
      "LaunchTemplate": {
         "LaunchTemplateId": { "Ref": "myLaunchTemplate" },
         "Version": { "Fn::GetAtt": [ "myLaunchTemplate", "LatestVersionNumber" ] }
       },
       "MaxSize": "1",
       "MinSize": "1",
       "VPCZoneIdentifier": [
          "subnetIdAz1",
          "subnetIdAz2",
          "subnetIdAz3"
      ],
      "NotificationConfigurations" : [{
         "TopicARN" : { "Fn::Select" : [ "0", { "Ref" : "AWS::NotificationARNs" } ] },
         "NotificationTypes" : [ "autoscaling:EC2_INSTANCE_LAUNCH",
 "autoscaling:EC2_INSTANCE_LAUNCH_ERROR" ]
      }]
   }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MinSize: '1'
    MaxSize: '1'
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    NotificationConfigurations:
      - TopicARN:
          Fn::Select:
          - '0'
          - Ref: AWS::NotificationARNs
```

```
        NotificationTypes:
        - autoscaling:EC2_INSTANCE_LAUNCH
        - autoscaling:EC2_INSTANCE_LAUNCH_ERROR
```

AWS::NoValue 条件属性

以下示例创建了一个 Amazon RDS 数据库实例，仅当提供了快照 ID 时，该数据库实例才会使用快照。如果 `UseDBSnapshot` 条件计算为 true，则 CloudFormation 对 `DBSnapshotIdentifier` 属性使用 `DBSnapshotName` 参数值。如果条件计算为 false，则 CloudFormation 删除 `DBSnapshotIdentifier` 属性。

JSON

```
"MyDB" : {
  "Type" : "AWS::RDS::DBInstance",
  "Properties" : {
    "AllocatedStorage" : "5",
    "DBInstanceClass" : "db.t2.small",
    "Engine" : "MySQL",
    "EngineVersion" : "5.5",
    "MasterUsername" : { "Ref" : "DBUser" },
    "MasterUserPassword" : { "Ref" : "DBPassword" },
    "DBParameterGroupName" : { "Ref" : "MyRDSParamGroup" },
    "DBSnapshotIdentifier" : {
      "Fn::If" : [
        "UseDBSnapshot",
        {"Ref" : "DBSnapshotName"},
        {"Ref" : "AWS::NoValue"}
      ]
    }
  }
}
```

YAML

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: MySQL
    EngineVersion: '5.5'
```

```
    MasterUsername:
      Ref: DBUser
    MasterUserPassword:
      Ref: DBPassword
    DBParameterGroupName:
      Ref: MyRDSParamGroup
    DBSnapshotIdentifier:
      Fn::If:
        - UseDBSnapshot
        - Ref: DBSnapshotName
        - Ref: AWS::NoValue
```

获取从已部署的 CloudFormation 堆栈导出的输出

如果在同一 Amazon Web Services 账户和区域中有多个堆栈，您可能需要在这些堆栈之间共享信息。当一个堆栈需要使用另一个堆栈创建的资源时，此功能非常实用。

例如，您可能有一个堆栈为您的 Web 服务器创建子网和安全组等网络资源。然后，创建实际 Web 服务器的其他堆栈可以使用第一个堆栈创建的网络资源。您无需对堆栈的模板中的资源 ID 进行硬编码或将 ID 作为输入参数传递。

要在堆栈之间共享信息，请从一个堆栈中导出输出值，然后将这些值导入另一个堆栈。下面将介绍操作方式：

1. 在第一个堆栈的模板（例如，联网堆栈）中，您可以使用 `Outputs` 部分中的 `Export` 字段来定义要导出的特定值。有关更多信息，请参阅 CloudFormation 模板 Outputs 语法。
2. 当您创建或更新该堆栈时，CloudFormation 会导出输出值，使其可供同一 Amazon Web Services 账户和区域中的其他堆栈使用。
3. 在另一个堆栈的模板中，您可以使用 Fn::ImportValue 函数导入从第一个堆栈导出的值。
4. 创建或更新第二个堆栈（例如，Web 服务器堆栈）时，CloudFormation 会自动从第一个堆栈中检索并使用导出的值。

有关演练和示例模板，请参阅引用其他 CloudFormation 堆栈中的资源输出。

导出堆栈输出值与使用嵌套堆栈

嵌套堆栈是您使用 `AWS::CloudFormation::Stack` 资源在其他堆栈中创建的堆栈。利用嵌套堆栈，您可从一个堆栈部署和管理所有资源。您可将来自嵌套堆栈组中的一个堆栈的输出用作该组中的另一个堆栈的输入。这不同于导出值。

如果您要将信息共享隔离在嵌套堆栈组中，建议您使用嵌套堆栈。要与其他堆栈 (不只是在嵌套堆栈组中) 共享信息，请导出值。例如，您可使用某个子网创建一个堆栈，然后导出其 ID。其他堆栈可通过导入子网的 ID 来使用该子网。每个堆栈都不需要创建自己的子网。只要堆栈导入子网 ID，您就无法更改或删除它。

有关嵌套堆栈的更多信息，请参阅使用嵌套堆栈将模板拆分为可重复使用的部分。

注意事项

以下限制适用于跨堆栈引用：

- 对于每个 Amazon Web Services 账户，`Export` 名称在区域内必须是唯一的。
- 使用 `Export` 和 `Fn::ImportValue` 时，跨堆栈引用仅限于相同的账户和区域。要跨账户或区域引用堆栈输出，请使用 `Fn::GetStackOutput`。
- 对于输出，`Export` 的 `Name` 属性值无法使用依赖于资源的 `Ref` 或 `GetAtt` 函数。

  同样，`ImportValue` 函数无法包含依赖于资源的 `Ref` 或 `GetAtt` 函数。
- 在另一个堆栈导入输出值后，您将无法删除正在导出输出值的堆栈或修改已导出的输入值。必须先删除所有导入，然后才能删除导出堆栈或修改输出值。

列出导出的输出值

如果您需要查看从堆栈导出的输出值，则请使用以下方法之一：

列出已导出输出值（控制台）

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 从左侧导航窗格中选择导出。

列出已导出输出值 (Amazon CLI)

使用以下 list-exports 命令。将 *us-east-1* 替换为您的 Amazon Web Services 区域。

```
aws cloudformation list-exports --region us-east-1
```

下面是示例输出。

```
{
    "Exports": [
        {
            "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
            "Name": "private-vpc-subnet-a",
            "Value": "subnet-07b410xmplddcfa03"
        },
        {
            "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
            "Name": "private-vpc-subnet-b",
            "Value": "subnet-075ed3xmplebd2fb1"
        },
        {
            "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
            "Name": "private-vpc-vpcid",
            "Value": "vpc-011d7xmpl100e9841"
        }
    ]
}
```

CloudFormation 会显示当前区域已导出输出的名称和值，以及导出这些输出的来源堆栈。要在其他堆栈的模板中使用导出的输出值，请使用导出名称和 `Fn::ImportValue` 函数来引用该输出值。

列出导入已导出输出值的堆栈

要删除或更改导出的输出值，必须首先确定要导入这些输出值的堆栈。

要查看导入导出的输出值的堆栈，请使用以下方法之一：

列出导入已导出的输出值的堆栈 (控制台)

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 从左侧导航窗格中选择导出。
3. 要查看哪些堆栈导入给定的导出值，请选择该导出值的 Export Name（导出名称）。CloudFormation 会显示导出详细信息页面，其中列出了要导入值的所有堆栈。

列出导入已导出的输出值的堆栈 (Amazon CLI)

使用 list-imports 命令。将 *us-east-1* 替换为您的 Amazon Web Services 区域，并将 *private-vpc-vpcid* 替换为所导出输出值的名称。

```
aws cloudformation list-imports --region us-east-1 \
    --export-name private-vpc-vpcid
```

CloudFormation 将返回正在导入该值的堆栈的列表。

```
{
    "Imports": [
        "my-app-stack"
    ]
}
```

确定要导入特定导出值的堆栈后，您需要修改这些堆栈以移除引用相关输出值的 `Fn::ImportValue` 函数。您必须先删除引用了已导出输出值的所有导入，然后才能删除或修改已导出的输出值。

在不导出的情况下引用输出

除了使用 `Export` 和 `Fn::ImportValue` 进行跨堆栈引用外，您还可以使用 `Fn::GetStackOutput` 直接引用堆栈输出。这种方法不需要引用的堆栈声明导出，并且支持跨账户和跨区域引用。

下表比较了这两个方法：

导出方式（Fn::ImportValue）与 Fn::GetStackOutput 的对比

能力	导出+ **Fn::ImportValue**	**Fn::GetStackOutput**
同一个账户，同一个区域	支持	支持
跨账户	不支持	支持
跨区域	不支持	支持
需要显式导出	是	否
参考类型	强引用（阻止删除导出堆栈）	弱引用（在部署时解析）
引用完整性	强制实施	未强制实施

何时使用每个方法：

- 当您需要强引用完整性时，请使用 `Export` 和 `Fn::ImportValue`。采用此方法时，CloudFormation 会阻止您删除那些导出值被其他堆栈所使用的堆栈，从而防止意外破坏依赖关系。
- 当您需要在不同账户或不同区域之间引用输出时，或者您不希望管理显式导出时，请使用 `Fn::GetStackOutput`。请注意，`Fn::GetStackOutput` 创建的是弱引用。如果所引用的堆栈或其输出被删除，使用该引用的堆栈不会收到任何通知，后续任何需要重新解析该引用的操作都将失败。

有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 Fn::GetStackOutput。

使用 CloudFormation 提供的参数类型在运行时指定现有资源

创建模板时，您可以使用 CloudFormation 提供的专用参数类型创建要求用户输入现有 Amazon 资源标识符或 Systems Manager 参数的参数。

主题

- 概述
- 示例
- 注意事项
- 支持的特定于 Amazon 的参数类型
- 支持的 Systems Manager 参数类型
- 不支持的 Systems Manager 参数类型

概述

在 CloudFormation 中，您可以通过在堆栈创建或更新期间提供输入值，从而使用参数来自定义堆栈。此功能使您的模板可以在不同的场景中重复使用并提高灵活性。

参数是在 CloudFormation 模板的 `Parameters` 部分中定义的。每个参数都有一个名称和一个类型，此外还可以配置其他设置，例如默认值和允许值。有关更多信息，请参阅 CloudFormation 模板 Parameters 语法。

参数类型决定了参数可以接受的输入值种类。例如，`Number` 只接受数值，而 `String` 可接受文本输入。

CloudFormation 提供了几种额外的参数类型，您可以在模板中使用这些参数类型来引用现有 Amazon 资源和 Systems Manager 参数。

这些参数类型分为两类：

- Amazon 特定参数类型：CloudFormation 提供了一组参数类型，可帮助在创建或更新堆栈时捕获无效值。当您使用这些参数类型时，使用您模板的任何人都必须指定其创建堆栈的 Amazon Web Services 账户和区域中的有效值。

  如果他们使用 Amazon Web Services 管理控制台，则 CloudFormation 会预先填充其账户和区域现有值的列表。这样一来，用户便无需记住和正确键入特定的名称或 ID，相反，他们只需从下拉列表中选择值。在某些情况下，他们甚至还可以按 ID、名称或 Name 标签值来搜索值。
- Systems Manager 参数类型：CloudFormation 还提供与 Systems Manager Parameter Store 中现有参数相对应的参数类型。当您使用这些参数类型时，使用您的模板的任何人都必须指定 Parameter Store 键作为 Systems Manager 参数类型的值，然后 CloudFormation 从 Parameter Store 中检索最新值以在其堆栈中使用。当您需要使用新亚马逊机器映像（AMI）ID 等新属性值来频繁更新应用程序时，这可能非常实用。有关 Parameter Store 的更多信息，请参阅 Systems Manager Parameter Store。

在 `Parameters` 部分中定义参数后，您可以使用 `Ref` 函数在整个 CloudFormation 模板中引用这些参数值。

示例

以下示例展示一个使用以下参数类型的模板。

- `AWS::EC2::VPC::Id`
- `AWS::EC2::Subnet::Id`
- `AWS::EC2::KeyPair::KeyName`
- `AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>`

要通过此模板创建堆栈，您必须指定账户中的现有 VPC ID、子网 ID 和密钥对名称。您也可以指定引用所需 AMI ID 的现有 Parameter Store 键或保留 */aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2* 的默认值。此公有参数是最新 Amazon Linux 2 AMI 的区域 AMI ID 的别名。有关公有参数的更多信息，请参阅《Amazon Systems Manager 用户指南》中的发现 Parameter Store 中的公有参数。

JSON

```
{
    "Parameters": {
        "VpcId": {
            "Description": "ID of an existing Virtual Private Cloud (VPC).",
            "Type": "AWS::EC2::VPC::Id"
        },
        "PublicSubnetId": {
            "Description": "ID of an existing public subnet within the specified VPC.",
            "Type": "AWS::EC2::Subnet::Id"
        },
        "KeyName": {
            "Description": "Name of an existing EC2 key pair to enable SSH access to
 the instance.",
            "Type": "AWS::EC2::KeyPair::KeyName"
        },
        "AMIId": {
            "Description": "Name of a Parameter Store parameter that stores the ID of
 the Amazon Machine Image (AMI).",
            "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
            "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
        }
    },
    "Resources": {
        "InstanceSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Enable SSH access via port 22",
                "VpcId": { "Ref": "VpcId" },
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 22,
                        "ToPort": 22,
                        "CidrIp": "0.0.0.0/0"
                    }
                ]
            }
        },
        "Ec2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
```

```
                "KeyName": { "Ref": "KeyName" },
                "ImageId": { "Ref": "AMIId" },
                "NetworkInterfaces": [
                    {
                        "AssociatePublicIpAddress": "true",
                        "DeviceIndex": "0",
                        "SubnetId": { "Ref": "PublicSubnetId" },
                        "GroupSet": [{ "Ref": "InstanceSecurityGroup" }]
                    }
                ]
            }
        }
    },
    "Outputs": {
        "InstanceId": {
            "Value": { "Ref": "Ec2Instance" }
        }
    }
}
```

YAML

```
Parameters:
  VpcId:
    Description: ID of an existing Virtual Private Cloud (VPC).
    Type: AWS::EC2::VPC::Id
  PublicSubnetId:
    Description: ID of an existing public subnet within the specified VPC.
    Type: AWS::EC2::Subnet::Id
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance.
    Type: AWS::EC2::KeyPair::KeyName
  AMIId:
    Description: Name of a Parameter Store parameter that stores the ID of the Amazon
 Machine Image (AMI).
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH access via port 22
      VpcId: !Ref VpcId
```

```
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      ImageId: !Ref AMIId
      NetworkInterfaces:
        - AssociatePublicIpAddress: "true"
          DeviceIndex: "0"
          SubnetId: !Ref PublicSubnetId
          GroupSet:
            - !Ref InstanceSecurityGroup
Outputs:
  InstanceId:
    Value: !Ref Ec2Instance
```

用于创建堆栈的 Amazon CLI 命令

以下 create-stack 命令基于示例模板创建堆栈。

```
aws cloudformation create-stack --stack-name MyStack \
  --template-body file://sampletemplate.json \
  --parameters \
ParameterKey="VpcId",ParameterValue="vpc-a123baa3" \
ParameterKey="PublicSubnetId",ParameterValue="subnet-123a351e" \
ParameterKey="KeyName",ParameterValue="MyKeyName" \
ParameterKey="AMIId",ParameterValue="MyParameterKey"
```

要使用接受字符串列表的参数类型（例如 `List<AWS::EC2::Subnet::Id>`），必须使用双反斜杠对 `ParameterValue` 中的逗号进行转义，如以下示例所示。

```
--parameters ParameterKey="SubnetIDs",ParameterValue="subnet-5ea0c127\\,subnet-6194ea3b
\\,subnet-c87f2be0"
```

注意事项

强烈建议使用动态引用来限制对敏感配置定义（例如第三方凭证）的访问。有关更多信息，请参阅 使用动态引用获取存储在其他服务中的值。

如果要允许模板用户指定不同 Amazon Web Services 账户的值，则请不要使用 Amazon 特定参数类型。而是应定义 `String` 或 `CommaDelimitedList` 类型的参数。

对于 Systems Manager 参数类型，需要注意以下几点：

- 您可以在控制台中或通过运行 describe-stacks 或 describe-change-set 来查看堆栈的参数选项卡上的已解析参数值。请记住，这些值是在创建或更新堆栈时设置的，因此它们可能与 Parameter Store 中的最新值不同。
- 对于堆栈更新，当您使用使用现有值选项（或将 `UsePreviousValue` 设置为 true）时，这意味着您要继续使用相同的 Parameter Store 键，而不是其值。CloudFormation 始终检索最新值。
- 如果您指定了任何允许的值或其他约束，则 CloudFormation 会根据您指定的参数键对其进行验证，但不会验证其值。您应该在 Parameter Store 本身中验证这些值。
- 创建或更新堆栈并创建更改集时，CloudFormation 会使用当时 Parameter Store 中存在的任何值。如果某个指定的参数在调用方的 Amazon Web Services 账户下的 Parameter Store 中不存在，CloudFormation 会返回验证错误。
- 执行更改集时，CloudFormation 将使用在更改集中指定的值。您应在执行更改集前检查这些值，因为在您创建更改集和运行更改集之间的时间内，它们可能会在 Parameter Store 中发生更改。
- 对于存储在同一 Amazon Web Services 账户中的 Parameter Store 参数，必须提供参数名称。对于由其他 Amazon Web Services 账户共享的 Parameter Store 参数，必须提供完整的参数 ARN。

支持的特定于 Amazon 的参数类型

CloudFormation 支持以下 Amazon 特定的类型：

`AWS::EC2::AvailabilityZone::Name`

可用区，如 `us-west-2a`。

`AWS::EC2::Image::Id`

Amazon EC2 映像 ID，如 `ami-0ff8a91507f77f867`。请注意，CloudFormation 控制台不会显示此参数类型的值的下拉列表。

`AWS::EC2::Instance::Id`

Amazon EC2 实例 ID，如 `i-1e731a32`。

`AWS::EC2::KeyPair::KeyName`

Amazon EC2 密钥对名称。

`AWS::EC2::SecurityGroup::GroupName`

默认 VPC 安全组名称，例如 `my-sg-abc`。

`AWS::EC2::SecurityGroup::Id`

安全组 ID，如 `sg-a123fd85`。

`AWS::EC2::Subnet::Id`

子网 ID，如 `subnet-123a351e`。

`AWS::EC2::Volume::Id`

Amazon EBS 卷 ID，如 `vol-3cdd3f56`。

`AWS::EC2::VPC::Id`

VPC ID，如 `vpc-a123baa3`。

`AWS::Route53::HostedZone::Id`

Amazon Route 53 托管区域 ID，如 `Z23YXV4OVPL04A`。

`List<AWS::EC2::AvailabilityZone::Name>`

针对某个区域的一组可用区，如 `us-west-2a, us-west-2b`。

`List<AWS::EC2::Image::Id>`

一组 Amazon EC2 映像 ID，如 `ami-0ff8a91507f77f867, ami-0a584ac55a7631c0c`。请注意，CloudFormation 控制台不会显示此参数类型的值的下拉列表。

`List<AWS::EC2::Instance::Id>`

一组 Amazon EC2 实例 ID，如 `i-1e731a32, i-1e731a34`。

`List<AWS::EC2::SecurityGroup::GroupName>`

一组默认 VPC 安全组名称，例如 `my-sg-abc, my-sg-def`。

`List<AWS::EC2::SecurityGroup::Id>`

一组安全组 ID，如 `sg-a123fd85, sg-b456fd85`。

`List<AWS::EC2::Subnet::Id>`

一组子网 ID，如 `subnet-123a351e, subnet-456b351e`。

`List<AWS::EC2::Volume::Id>`

一组 Amazon EBS 卷 ID，如 `vol-3cdd3f56, vol-4cdd3f56`。

`List<AWS::EC2::VPC::Id>`

一组 VPC ID，如 `vpc-a123baa3, vpc-b456baa3`。

`List<AWS::Route53::HostedZone::Id>`

一组 Amazon Route 53 托管区域 ID，如 `Z23YXV4OVPL04A, Z23YXV4OVPL04B`。

支持的 Systems Manager 参数类型

CloudFormation 支持以下 Systems Manager 参数类型：

`AWS::SSM::Parameter::Name`

Systems Manager 参数键的名称。仅使用此参数类型来检查所需的参数是否存在。CloudFormation 不会检索与该参数关联的实际值。

`AWS::SSM::Parameter::Value<String>`

其值是字符串的 Systems Manager 参数。这与 Parameter Store 中的 `String` 参数对应。

`AWS::SSM::Parameter::Value<List<String>>` 或
`AWS::SSM::Parameter::Value<CommaDelimitedList>`

其值是字符串列表的 Systems Manager 参数。这与 Parameter Store 中的 `StringList` 参数对应。

`AWS::SSM::Parameter::Value<`*`AWS-specific parameter type`*`>`

其值是 Amazon 特定参数类型的 Systems Manager 参数。

例如，下面指定了 `AWS::EC2::KeyPair::KeyName` 类型：

- `AWS::SSM::Parameter::Value<AWS::EC2::KeyPair::KeyName>`

`AWS::SSM::Parameter::Value<List<`*`AWS-specific parameter type`*`>>`

其值是 Amazon 特定参数类型列表的 Systems Manager 参数。

例如，下面指定了 `AWS::EC2::KeyPair::KeyName` 类型的列表：

- `AWS::SSM::Parameter::Value<List<AWS::EC2::KeyPair::KeyName>>`

不支持的 Systems Manager 参数类型

CloudFormation 不支持以下 Systems Manager 参数类型：

- Systems Manager 参数类型的列表，例如：`List<AWS::SSM::Parameter::Value<String>>`

此外，CloudFormation 不支持将模板参数定义为 `SecureString` Systems Manager 参数类型。不过对于某些资源，您可以将安全字符串指定为参数值。有关更多信息，请参阅 使用动态引用获取存储在其他服务中的值。

CloudFormation 演练

本文档提供了旨在让您动手实践堆栈部署的一系列演练。

- 引用其他 CloudFormation 堆栈中的资源输出 – 本演练演示了如何在一个堆栈中引用另一个 CloudFormation 堆栈的输出。您可以在单独的堆栈中创建相关的 Amazon 资源，从而创建更加模块化和可重复使用的模板，而不是在单个堆栈中包含所有资源。
- 在 Amazon EC2 上部署应用程序 – 了解如何使用 CloudFormation 在 Amazon EC2 实例上自动安装、配置和启动应用程序。这样能让您轻松复制部署和更新现有安装，而无需直接连接到实例。
- 更新 CloudFormation 堆栈 – 演示使用 CloudFormation 更新正在运行的堆栈的简单过程。
- 创建已扩展且负载均衡的应用程序 – 了解如何使用 CloudFormation 创建可扩展的负载均衡应用程序。本演练介绍如何创建自动扩缩组、负载均衡器和其他相关资源，以确保应用程序能够处理不断变化的流量负载并保持高可用性。
- 与其他 Amazon Web Services 账户中的 VPC 建立对等连接 – 本演练将引导您在不同 中的两个 VPC 之间，完成创建虚拟私有云（VPC）对等连接的过程。Amazon Web Services 账户VPC 对等连接有助您在 VPC 之间路由流量，并像这些资源位于同一网络中一样访问资源。
- 通过 CodeDeploy 使用 CloudFormation 执行 ECS 蓝/绿部署：了解如何使用 CloudFormation 在 Amazon ECS 上执行 Amazon CodeDeploy 蓝绿部署。蓝绿部署是一种在最短停机时间内更新应用程序或服务的方法。

引用其他 CloudFormation 堆栈中的资源输出

本演练演示了如何在一个堆栈中引用另一个 CloudFormation 堆栈的输出，以创建更多模块化和可重复使用的模板。

您可以在单独的堆栈中创建相关的 Amazon 资源，而不是在单个堆栈中包含所有资源。然后，您可以引用其他堆栈中的所需资源输出。通过限制对输出的跨堆栈引用，您可以控制由其他堆栈引用的堆栈部分。

例如，您可以将一个带 VPC、安全组和子网的网络堆栈用于公有 Web 应用程序，并且有一个单独的公有 Web 应用程序堆栈。为确保 Web 应用程序能使用网络堆栈中的安全组和子网，可以创建一个使 Web 应用程序堆栈能够引用网络堆栈中的资源输出的跨堆栈引用。借助跨堆栈引用，Web 应用程序堆栈的所有者无需创建或维护联网规则或资产。

要创建跨堆栈引用，可使用 `Export` 输出字段标记要导出的资源输出的值。然后，使用 `Fn::ImportValue` 内置函数导入该值。有关更多信息，请参阅 获取从已部署的 CloudFormation 堆栈导出的输出。

> **Note**
>
> CloudFormation 是一项免费服务。但是，对于堆栈中包含的 Amazon 资源，您需要按各自的现有费率付费。有关 Amazon 定价的更多信息，请参阅每种产品的详细信息页。

主题

- 使用示例模板创建网络堆栈
- 使用示例模板创建 Web 应用程序堆栈
- 验证堆栈的运行是否符合设计
- 排查 AMI 映射问题
- 清除资源

使用示例模板创建网络堆栈

在开始此演练之前，请确认您具有使用以下所有服务的 IAM 权限：Amazon VPC、Amazon EC2 和 CloudFormation。

网络堆栈包含您将在 Web 应用程序堆栈中使用的 VPC、安全组和子网。除了这些资源之外，网络堆栈还将创建互联网网关和路由表来实现公共访问。

您必须先创建此堆栈，然后才能创建 Web 应用程序堆栈。如果先创建 Web 应用程序堆栈，则它没有安全组或子网。

该堆栈模板可通过以下 URL 获取：https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleNetworkCrossStack.template。要查看堆栈将创建的资源，请选择该链接，这将打开模板。在 `Outputs` 部分中，您可看到示例模板导出的联网资源。如果您从其他堆栈导出联网资源，则已导出资源的名称将以堆栈的名称为前缀。当用户导入联网资源时，他们可指定从中导入资源的堆栈。

创建网络堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面，选择右上角的堆栈，然后选择使用新资源（标准）。
3. 选择模板已准备就绪，然后在指定模板部分选择 Amazon S3 URL。
4. 在 Amazon S3 URL 中，粘贴以下 URL：**`https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleNetworkCrossStack.template`**。
5. 选择下一步。
6. 对于 Stack name (堆栈名称)，键入 **`SampleNetworkCrossStack`**，然后选择 Next (下一步)。

   > **Note**
   >
   > 记录此堆栈的名称。您在启动 Web 应用程序堆栈时需要此堆栈名称。

7. 选择下一步。在本演练中，您无需添加标记或指定高级设置。
8. 确保堆栈名称和模板 URL 正确，然后选择 Create stack (创建堆栈)。

   CloudFormation 可能需要几分钟的时间创建堆栈。请耐心等待，直到成功创建所有资源，然后再继续创建 Web 应用程序堆栈。
9. 要监控进度，可查看堆栈事件。有关更多信息，请参阅 监控堆栈进度。

使用示例模板创建 Web 应用程序堆栈

Web 应用程序堆栈将创建一个使用网络堆栈中的安全组和子网的 EC2 实例。

您必须在网络堆栈所在的同一 Amazon Web Services 区域中创建该堆栈。

该堆栈模板可通过以下 URL 获取：https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template。要查看堆栈将创建的资源，请选择该链接，这样

会打开模板。在 Resources 部分中，查看 EC2 实例的属性。您可查看如何使用 Fn::ImportValue 函数从其他堆栈导入联网资源。

创建 Web 应用程序堆栈

1. 在堆栈页面，选择右上角的创建堆栈，然后选择使用新资源（标准）。
2. 选择模板已准备就绪，然后在指定模板部分选择 Amazon S3 URL。
3. 在 Amazon S3 URL 中，粘贴以下 URL：**https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template**。
4. 选择下一步。
5. 对于 Stack name，键入 **SampleWebAppCrossStack**。在 Parameters 部分中，让 NetworkStackName 参数使用默认值，然后选择 Next。

   示例模板使用参数值指定要从中导入值的堆栈。
6. 选择下一步。在本演练中，您无需添加标记或指定高级设置。
7. 确保堆栈名称和模板 URL 正确，然后选择 Create stack (创建堆栈)。

   CloudFormation 可能需要几分钟的时间创建堆栈。

验证堆栈的运行是否符合设计

创建该堆栈后，查看其资源并记下实例 ID。有关查看堆栈资源的更多信息，请参阅从 CloudFormation 控制台查看堆栈信息。

要验证实例的安全组和子网，请在 Amazon EC2 控制台中查看实例的属性。如果实例使用 SampleNetworkCrossStack 堆栈中的安全组和子网，则表示您已成功创建跨堆栈引用。

使用控制台可查看堆栈输出和示例网站 URL 以验证 Web 应用程序是否正在运行。有关更多信息，请参阅 从 CloudFormation 控制台查看堆栈信息。

排查 AMI 映射问题

如果您收到 Template error: Unable to get mapping for AWSRegionArch2AMI::[region]::HVM64 错误，则该模板不包含您在 Amazon Web Services 区域的 AMI 映射。我们建议不要更新映射，而是使用 Systems Manager 公共参数动态引用最新的 AMI：

1. 将 SampleWebAppCrossStack 模板下载到您的本地计算机，下载网址为：https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template。

2. 删除整个 AWSRegionArch2AMI 映射部分。
3. 添加以下 Systems Manager 参数：

```
"LatestAmiId": {
  "Description": "The latest Amazon Linux 2 AMI from the Parameter Store",
    "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
    "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
  }
```

4. 替换现有的 ImageId 引用：

```
"ImageId": { "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" } ,
 "HVM64" ] },
```

   借助:

```
"ImageId": { "Ref": "LatestAmiId" },
```

   此参数会自动解析为堆栈部署区域的最新 Amazon Linux 2 AMI。

   对于其他 Linux 发行版，请使用相应的参数路径。有关公共参数的更多信息，请参阅《Amazon Systems Manager 用户指南》中的发现 Parameter Store 中的公有参数。

5. 将修改后的模板上传到您账户中的 S3 存储桶：

```
aws s3 cp SampleWebAppCrossStack.template s3://amzn-s3-demo-bucket/
```

6. 创建堆栈时，请指定您的 S3 模板 URL，不要使用示例 URL。

清除资源

为了确保不因任何不必要的服务而产生费用，请删除堆栈。

删除堆栈

1. 在 CloudFormation 控制台中，选择 SampleWebAppCrossStack 堆栈。
2. 选择 Actions)（操作），然后选择 Delete Stack（删除堆栈）。
3. 在确认消息中，选择 Delete (删除)。
4. 在删除堆栈后，对 SampleNetworkCrossStack 堆栈重复相同的步骤。

> **Note**
>
> 等到 CloudFormation 完全删除 `SampleWebAppCrossStack` 堆栈。如果 EC2 实例仍在 VPC 中运行，则 CloudFormation 不会删除 `SampleNetworkCrossStack` 堆栈中的 VPC。

在 Amazon EC2 上部署应用程序

您可以使用 CloudFormation 在 Amazon EC2 实例上自动安装、配置和启动应用程序。这样能让您轻松复制部署和更新现有安装而无需直接连接到该实例，从而为您节省大量时间和工作量。

CloudFormation 包括一组基于 `cloud-init` 的帮助程序脚本（`cfn-init`、`cfn-signal`、`cfn-get-metadata` 和 `cfn-hup`）。您可从 CloudFormation 模板中调用这些帮助程序脚本来在使用相同模板的 Amazon EC2 实例上安装、配置和更新应用程序。有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 CloudFormation 帮助程序脚本参考。

在入门教程中，您使用 `UserData` 和基础 bash 脚本创建了一个简单的 Web 服务器。虽然顺利创建了“Hello World”这样的简单页面，但实际应用程序通常需要更复杂的配置，包括：

- 按照正确的顺序安装多个软件包。
- 使用特定内容创建复杂的配置文件。
- 启动服务并将其配置为自动运行。
- 错误处理和验证设置过程。

与 `UserData` 中的基础 bash 脚本相比，CloudFormation 的帮助程序脚本提供了一种更稳健、更易于维护的方式来配置 EC2 实例。`cfn-init` 帮助程序脚本可以从模板的元数据中读取配置数据，并将其系统地应用于您的实例。

在本教程中，您将学习如何使用 `cfn-init` 帮助程序脚本以及如何监控引导过程。

> **Note**
>
> CloudFormation 使用免费，但您需要为自己创建的 Amazon EC2 资源付费。如果您不熟悉 Amazon，可以利用免费套餐来最大限度地降低或避免在学习过程产生费用。

主题

- 先决条件
- 了解引导的概念
- 从简单的引导示例开始
- 添加文件和命令
- 增强网络安全
- 完整的引导模板
- 使用控制台创建堆栈。
- 监控引导过程
- 测试引导的 Web 服务器
- 引导问题排查
- 清理 资源
- 后续步骤

先决条件

- 您必须已完成 创建第一个堆栈 教程或具有 CloudFormation 基础知识的同等经验。
- 您必须具有对 Amazon Web Services 账户的访问权限，且该账户需要拥有有权使用 Amazon EC2 和 CloudFormation 的 IAM 用户或角色，或者拥有管理用户访问权限。
- 必须拥有可访问互联网的虚拟私有云（VPC）。本教程模板需要默认 VPC，新版本 Amazon Web Services 账户自动附带该默认 VPC。如果您没有默认 VPC，或者已将其删除，请参阅“创建第一个堆栈”教程中的疑难解答部分，了解其他解决方案。

了解引导的概念

在创建模板之前，我们先了解一下有助于引导顺利完成的关键概念。

`cfn-init` 帮助程序脚本

CloudFormation 提供 Python 帮助程序脚本，可用于在 Amazon EC2 实例中安装软件和启动服务。`cfn-init` 脚本可以从模板的元数据中读取配置数据，并将其应用于实例。

过程如下所述：

1. 在 EC2 资源的 `Metadata` 部分中定义配置。

2. 从 UserData 脚本中调用 cfn-init。
3. cfn-init 读取元数据并应用配置。
4. 根据规范配置实例。

元数据结构

该配置是在 EC2 实例中的特定结构中进行定义。

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Metadata:                       # Metadata section for the resource
      AWS::CloudFormation::Init:    # Required key that cfn-init looks for
        config:                     # Configuration name (you can have multiple)
          packages:                 # Install packages
          files:                    # Create files
          commands:                 # Run commands
          services:                 # Start/stop services
```

cfn-init 脚本将以特定顺序处理这些部分：
packages、groups、users、sources、files、commands，然后是 services。

从简单的引导示例开始

我们先从最简单的引导示例开始，该示例仅安装和启动 Apache。

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:                 # Install Apache web server
            yum:
              httpd: []
          services:                 # Start Apache and enable it to start on boot
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
    Properties:
```

```
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      UserData: !Base64             # Script that runs when instance starts
        Fn::Sub: |
          #!/bin/bash
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource EC2Instance --
region ${AWS::Region}
```

这个简单的示例演示了核心概念：

- packages 部分使用 yum 安装 httpd 软件包。这也适用于 Amazon Linux 和其他使用 yum 的 Linux 发行版。
- services 部分用于确保 httpd 自动启动和运行。
- UserData 会安装最新的引导工具并调用 cfn-init。

添加文件和命令

现在，我们来改进示例，在 EC2 实例的 /var/log 目录中添加自定义网页和日志文件。

创建文件

files 部分可让您在实例上创建包含特定内容的文件。直竖线 (|) 允许您将文字文本块（HTML 代码）作为文件内容 (/var/www/html/index.html) 传递。

```
files:
  /var/www/html/index.html:
    content: |
      <body>
        <h1>Congratulations, you have successfully launched the AWS CloudFormation
 sample.</h1>
      </body>
```

运行命令

commands 部分可让您在引导过程中运行 shell 命令。此命令会在 EC2 实例的 /var/log/welcome.txt 位置创建日志文件。要查看该文件，您需要 Amazon EC2 密钥对以进行 SSH 访问，以及 IP 地址范围以便以 SSH 方式连接实例（此处未介绍）。

```
commands:
  createWelcomeLog:
```

```
    command: "echo 'cfn-init ran successfully!' > /var/log/welcome.txt"
```

增强网络安全

由于我们正在设置 Web 服务器，因此需要允许 Web 流量（HTTP）传输到 EC2 实例。为此，我们将创建一个安全组，以便来自您的 IP 地址的入站流量通过端口 80。EC2 实例还需要将流量发送到互联网，以便实现软件包更新安装等目的。默认情况下，安全组会允许所有出站流量。然后，我们使用 SecurityGroupIds 属性将此安全组与 EC2 实例关联。

```
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow HTTP access from my IP address
    SecurityGroupIngress:
      - IpProtocol: tcp
        Description: HTTP
        FromPort: 80
        ToPort: 80
        CidrIp: !Ref MyIP
```

完整的引导模板

现在，我们把所有部分合在一起。这就是之前介绍的所有概念整合而成的完整模板。

```
AWSTemplateFormatVersion: 2010-09-09
Description: Bootstrap an EC2 instance with Apache web server using cfn-init

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.
```

```
  MyIP:
    Description: Your IP address in CIDR format (e.g. 203.0.113.1/32)
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access from my IP address
      SecurityGroupIngress:
        - IpProtocol: tcp
          Description: HTTP
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP

  WebServer:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:
            yum:
              httpd: []
          files:
            /var/www/html/index.html:
              content: |
                <body>
                  <h1>Congratulations, you have successfully launched the AWS
 CloudFormation sample.</h1>
                </body>
          commands:
            createWelcomeLog:
              command: "echo 'cfn-init ran successfully!' > /var/log/welcome.txt"
          services:
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
```

```
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData: !Base64
        Fn::Sub: |
          #!/bin/bash
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource WebServer --
region ${AWS::Region}
      Tags:
        - Key: Name
          Value: Bootstrap Tutorial Web Server

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServer.PublicDnsName}'
    Description: EC2 instance public DNS name
```

使用控制台创建堆栈。

以下过程涉及从文件上传示例堆栈模板。在本地计算机上打开文本编辑器并添加模板。使用文件名 `samplelinux2stack.template` 保存该文件。

启动堆栈模板

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 依次选择创建堆栈和使用新资源（标准）。
3. 在指定模板下，选择上传模板文件，然后选择选择文件，上传 `samplelinux2stack.template` 文件。
4. 选择下一步。
5. 在指定堆栈详细信息页面上，将堆栈命名为 **`BootstrapTutorialStack`**。
6. 在参数下，执行以下操作。
   - LatestAmiId：保留默认值。
   - InstanceType：选择 t2.micro 或 t3.micro 作为 EC2 实例类型。
   - MyIP：输入带 `/32` 后缀的公有 IP 地址。

7. 选择两次下一步，然后选择提交即可创建堆栈。

监控引导过程

引导过程比简单的 EC2 启动需要更长的时间，因为引导过程中需要安装和配置其他软件。

监控引导进度

1. 在 CloudFormation 控制台中，选择堆栈并打开事件选项卡。
2. 注意 `WebServer CREATE_IN_PROGRESS` 事件。引导过程在实例启动后开始。
3. 引导过程通常需要花费几分钟的时间。完成时，您可以看到 `WebServer CREATE_COMPLETE`。

如果想知道引导过程中发生了什么，可以查看实例日志。

查看引导日志（可选）

1. 打开 EC2 控制台，找到您的实例。
2. 选择实例，然后依次选择操作、监控和故障排除、获取系统日志，即可查看引导过程。
3. 如果日志没有即刻出现，请稍等片刻，然后刷新页面。

测试引导的 Web 服务器

当堆栈显示 `CREATE_COMPLETE` 时，请测试 Web 服务器。

测试 Web 服务器

1. 在 CloudFormation 控制台中，转到您堆栈的输出选项卡。
2. 单击 WebsiteURL 值，在新选项卡中打开 Web 服务器。
3. 您应看到自定义网页和消息 `Congratulations, you have successfully launched the AWS CloudFormation sample`。

> **Note**
>
> 如果页面没有立即加载，请稍等片刻，然后重试。即使堆栈已显示 `CREATE_COMPLETE`，引导过程可能仍在完成。

引导问题排查

如果引导过程失败或您的 Web 服务器无法运行，以下是常见问题和解决方案。

常见问题

- 堆栈创建失败：查看事件选项卡以获取相应的错误消息。
- 无法访问 Web 服务器：验证 `MyIP` 参数中您的 IP 地址是否正确。别忘了在末尾加上 `/32`。
- 引导过程失败：实例可能会启动但 `cfn-init` 失败。按照“监控”章节所述查看系统日志。

清理 资源

为了避免持续产生费用，您可以删除堆栈及其资源来进行清除。

想要删除堆栈和它的资源

1. 打开 CloudFormation 控制台。
2. 在堆栈页面上，选择您创建的堆栈名称旁的选项（**`BootstrapTutorialStack`**），然后选择删除。
3. 当系统提示进行确认时，选择 Delete（删除）。
4. 在事件选项卡上监控堆栈删除过程的进度。**`BootstrapTutorialStack`** 的状态更改为 `DELETE_IN_PROGRESS`。当 CloudFormation 完成删除堆栈后，它会将从列表中移堆栈除。

后续步骤

恭喜您！现在您已经成功地学会如何使用 CloudFormation 引导 EC2 实例。您现在已经知道：

- 如何使用 `cfn-init` 帮助程序脚本
- 如何构建用于引导的元数据
- 如何安装软件包、创建文件、运行命令和管理服务
- 如何监控引导问题

继续学习：

- 了解如何更新正在运行的堆栈并使用 `cfn-hup` 帮助程序脚本。有关更多信息，请参阅 更新 CloudFormation 堆栈。

- 了解如何引导 Windows 堆栈。有关更多信息，请参阅 引导基于 Windows 的 CloudFormation 堆栈。
- 使用多个配置集探索更复杂的引导情境。有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 cfn-init 和 AWS::CloudFormation::Init。
- 了解如何使用 `cfn-signal` 报告引导完成状态。有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 cfn-signal。

更新 CloudFormation 堆栈

> **Note**
>
> 本教程基于 在 Amazon EC2 上部署应用程序 教程中的概念生成。如果您尚未完成该教程，我们建议您先完成该教程，以了解使用 CloudFormation 进行的 EC2 引导的过程。

本主题演示了正在运行的堆栈的简单更新进展。我们将带您逐步了解以下步骤：

1. 创建起始堆栈：使用基础 Amazon Linux 2 AMI 创建堆栈，从而使用 CloudFormation 帮助程序脚本安装 Apache Web Server 和简单的 PHP 应用程序。
2. 更新应用程序：使用 CloudFormation 更新应用程序中的一个文件并部署软件。
3. 添加密钥对：向实例中添加 Amazon EC2 密钥对，然后更新安全组以允许 SSH 访问实例。
4. 更新实例类型：更改底层 Amazon EC2 实例的实例类型。
5. 更新 AMI：在您的堆栈中更改 Amazon EC2 实例的亚马逊机器映像（AMI）。

> **Note**
>
> CloudFormation 使用免费，但您需要为自己创建的 Amazon EC2 资源付费。如果您不熟悉 Amazon，可以利用免费套餐来最大限度地降低或避免在学习过程产生费用。

主题

- 第 1 步：创建起始堆栈
- 第 2 步：更新应用程序
- 第 3 步：添加使用密钥对的 SSH 访问权限
- 第 4 步：更新实例类型

- 第 5 步：更新 AMI
- 可用性和影响注意事项
- 相关资源

第 1 步：创建起始堆栈

我们将从创建可在本主题剩下的所有内容中使用的堆栈开始。我们已提供了一个简单的模板来启动在 Apache Web Server 中托管并在 Amazon Linux 2 AMI 上运行的简单实例 PHP Web 应用程序。

Apache Web Server、PHP 和简单的 PHP 应用程序全部都由默认安装在 Amazon Linux 2 AMI 上的 CloudFormation 帮助程序脚本进行安装。以下模板代码段显示描述待安装软件包和文件的元数据，此情况下为 Amazon Linux 2 AMI 的 Yum 存储库中的 Apache Web Server 和 PHP 基础设施。代码段还显示 `Services` 部分，以确保 Apache Web Server 处于运行状态。

```
WebServerInstance:
  Type: AWS::EC2::Instance
  Metadata:
    AWS::CloudFormation::Init:
      config:
        packages:
          yum:
            httpd: []
            php: []
        files:
          /var/www/html/index.php:
            content: |
              <?php
              echo '<h1>Hello World!</h1>';
              ?>
            mode: '000644'
            owner: apache
            group: apache
        services:
          systemd:
            httpd:
              enabled: true
              ensureRunning: true
```

应用程序本身是“Hello World”示例，在模板中进行了完整定义。对于现实工作中的应用程序，文件可存储在 Amazon S3、GitHub 或另一个存储库中，并通过模板进行参考。CloudFormation 可下载软件包

（如 RPM 或 RubyGem），并能引用单个的文件以及展开 .zip 和 .tar 文件，以在 Amazon EC2 实例上创建应用程序项目。

模板用于启用和配置 cfn-hup 进程守护程序以侦听 Amazon EC2 实例元数据中所定义配置的更改。您可以使用 cfn-hup 进程守护程序更新应用程序软件，如 Apache 或 PHP 的版本，也可通过 CloudFormation 更新 PHP 应用程序文件。来自该模板中同一 Amazon EC2 资源的以下代码片段展示了配置 cfn-hup 所需的必要部分，即每两分钟调用 cfn-init 一次以通知并应用元数据的更新。否则，cfn-init 只能在启动时运行一次。

```
files:
  /etc/cfn/cfn-hup.conf:
    content: !Sub |
      [main]
      stack=${AWS::StackId}
      region=${AWS::Region}
      # The interval used to check for changes to the resource metadata in minutes.
 Default is 15
      interval=2
    mode: '000400'
    owner: root
    group: root
  /etc/cfn/hooks.d/cfn-auto-reloader.conf:
    content: !Sub |
      [cfn-auto-reloader-hook]
      triggers=post.update
      path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
      action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --region
 ${AWS::Region}
      runas=root
services:
  systemd:
    cfn-hup:
      enabled: true
      ensureRunning: true
      files:
        - /etc/cfn/cfn-hup.conf
        - /etc/cfn/hooks.d/cfn-auto-reloader.conf
```

为了完成堆栈，在 Amazon EC2 实例定义中的 Properties 部分，UserData 属性包含调用 cfn-init 以安装软件包和文件的 cloud-init 脚本。有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 CloudFormation 帮助程序脚本参考。该模板还创建了一个 Amazon EC2 安全组。

```
AWSTemplateFormatVersion: 2010-09-09

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues:
      - t3.nano
      - t3.micro
      - t3.small
      - t3.medium
      - t3a.nano
      - t3a.micro
      - t3a.small
      - t3a.medium
      - m5.large
      - m5.xlarge
      - m5.2xlarge
      - m5a.large
      - m5a.xlarge
      - m5a.2xlarge
      - c5.large
      - c5.xlarge
      - c5.2xlarge
      - r5.large
      - r5.xlarge
      - r5.2xlarge
      - r5a.large
      - r5a.xlarge
      - r5a.2xlarge
    ConstraintDescription: must be a valid EC2 instance type.

Resources:
  WebServerInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
```

```
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          # Get the latest CloudFormation package
          yum update -y aws-cfn-bootstrap
          # Run cfn-init
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
WebServerInstance --region ${AWS::Region} || error_exit 'Failed to run cfn-init'

          # Start up the cfn-hup daemon to listen for changes to the EC2 instance
metadata
          /opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'
          # Signal success or failure
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
WebServerInstance --region ${AWS::Region}
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:
            yum:
              httpd: []
              php: []
          files:
            /var/www/html/index.php:
              content: |
                <?php
                echo "<h1>Hello World!</h1>";
                ?>
              mode: '000644'
              owner: apache
              group: apache
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
                # The interval used to check for changes to the resource metadata in
minutes. Default is 15
                interval=2
              mode: '000400'
              owner: root
```

```
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
                action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --
region ${AWS::Region}
                runas=root
          services:
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
              cfn-hup:
                enabled: true
                ensureRunning: true
                files:
                  - /etc/cfn/cfn-hup.conf
                  - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    CreationPolicy:
      ResourceSignal:
        Timeout: PT5M

  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServerInstance.PublicDnsName}'
    Description: URL of the web application
```

要使用此模板启动堆栈

1. 复制该模板并将其作为文本文件本地保存到您的系统中。注意保存位置，因为您需要在后续步骤中使用此文件。

2. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
3. 依次选择创建堆栈、使用新资源（标准）。
4. 选择选择现有模板。
5. 在指定模板下，选择上传模板文件，浏览到您在第一步中创建的文件，然后选择下一步。
6. 在指定堆栈详细信息页面上，输入 **UpdateTutorial** 作为堆栈名称。
7. 在参数下，保持所有参数不变，然后选择两次下一步。
8. 在审核并创建屏幕上，选择提交。

当您的堆栈状态变成 CREATE_COMPLETE 后，输出选项卡会显示网站的 URL。如果您选择 WebsiteURL 的输出值，您将看到新 PHP 应用程序在工作。

第 2 步：更新应用程序

现在您已部署好堆栈，接下来更新应用程序吧。我们将对应用程序所打印出的文本进行简单更改。要执行此操作，我们将添加回显命令至 index.php 文件，如此模板代码段所示：

```
files:
  /var/www/html/index.php:
    content: |
      <?php
      echo "<h1>Hello World!</h1>";
      echo "<p>This is an updated version of our application.</p>";
      ?>
    mode: '000644'
    owner: apache
    group: apache
```

使用文本编辑器手动编辑您保存在本地的模板文件。

现在，更新堆栈。

要使用更新后的模板更新堆栈

1. 在 CloudFormation 控制台中，选择您的 **UpdateTutorial** 堆栈。
2. 选择更新、直接更新。
3. 选择替换现有模板。
4. 在指定模板下，选择上传模板文件，上传修改的模板文件模板，然后选择下一步。

5. 在指定堆栈详细信息页面上，保持所有参数相同，然后选择两次下一步。
6. 在审核页面上，审核您的更改。在更改下，您应该会看到 CloudFormation 将更新 `WebServerInstance` 资源。
7. 选择提交。

当您的堆栈处于 `UPDATE_COMPLETE` 状态时，您可以再次选择 `WebsiteURL` 输出值以验证应用程序的更改已生效。`cfn-hup` 进程守护程序每 2 分钟运行一次，因此最多能花 2 分钟在堆栈更新后更改应用程序。

要查看已更新资源集，请转至 CloudFormation 控制台。在事件选项卡上，查看堆栈事件。在此特别示例中，Amazon EC2 实例 `WebServerInstance` 的元数据已更新，这导致 CloudFormation 也会重新评估其他资源（`WebServerSecurityGroup`）以确保没有其他更改。其他堆栈资源都未修改。CloudFormation 将只更新堆栈中受堆栈的任何更改影响的资源。此类更改可直接进行，如属性或元数据更改，也可由依赖项或 `Ref`、`GetAtt` 中的数据流或其他内部模板函数导致。有关更多信息，请参阅内置函数参考。

这一简单更新对此过程进行了阐述。但是，您可以对您的 Amazon EC2 实例中所部署文件和软件包进行更复杂的更改。例如，您可能会确定需要将 MySQL 与 MySQL 的 PHP 支持一起添加到实例中。要执行此操作，只需要将附加软件包和文件与任何附加服务一起添加到配置中，然后更新堆栈以部署更改。

```
packages:
  yum:
    httpd: []
    php: []
    mysql: []
    php-mysql: []
    mysql-server: []
    mysql-libs: []

  ...

services:
  systemd:
    httpd:
      enabled: true
      ensureRunning: true
    cfn-hup:
      enabled: true
      ensureRunning: true
```

```
      files:
        - /etc/cfn/cfn-hup.conf
        - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    mysqld:
      enabled: true
      ensureRunning: true
```

您可以更新 CloudFormation 元数据，将应用程序所使用的软件包更新到新版本。前述示例中，每个软件包的版本属性都为空，这表示 `cfn-init` 应安装最新版的软件包。

```
packages:
  yum:
    httpd: []
    php: []
```

您可以视需要指定软件包的版本字符串。如果您在后续更新堆栈调用中更改版本字符串，则会部署新版软件包。此处显示 RubyGems 软件包版本号的使用示例。支持版本化的任何软件包都可以有特定版本。

```
packages:
  rubygems:
    mysql: []
    rubygems-update:
      - "1.6.2"
    rake:
      - "0.8.7"
    rails:
      - "2.3.11"
```

第 3 步：添加使用密钥对的 SSH 访问权限

您还可以更新模板中的资源，以添加原先未在模板中指定的属性。为了阐明上述操作，我们将会添加 Amazon EC2 密钥对到现有 EC2 实例中，然后在 Amazon EC2 安全组中打开端口 22，从而使您可以使用 Secure Shell（SSH）访问实例。

向现有 Amazon EC2 实例添加 SSH 访问权限

1. 向模板额外添加两个参数，从而以现有 Amazon EC2 密钥对和 SSH 位置的名称进行传递。

   ```
   Parameters:
     KeyName:
   ```

```
    Description: Name of an existing EC2 KeyPair to enable SSH access to the
 instance
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.

  SSHLocation:
    Description: The IP address that can be used to SSH to the EC2 instances in
 CIDR format (e.g. 203.0.113.1/32)
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
```

2. 将 KeyName 属性添加到 Amazon EC2 实例。

```
WebServerInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref LatestAmiId
    InstanceType: !Ref InstanceType
    KeyName: !Ref KeyName
    SecurityGroupIds:
      - !Ref WebServerSecurityGroup
```

3. 将端口 22 和 SSH 位置添加到 Amazon EC2 安全组的入口规则。

```
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref SSHLocation
```

4. 使用与 第 2 步：更新应用程序 中所述的相同步骤更新堆栈。

第 4 步：更新实例类型

现在，让我们演示一下如何通过更改实例类型来更新底层基础设施。

我们到目前为止所建立的堆栈使用 t3.micro Amazon EC2 实例。假设您新建的网站获取的流量比 t3.micro 实例能处理的流量多，且您现在想移动到 m5.large Amazon EC2 实例类型中。如果实例类型的架构发生变化，则必须使用不同的 AMI 创建实例。但是，t3.micro 和 m5.large 都使用相同的 CPU 架构并运行 Amazon Linux 2（x86_64）AMI。有关更多信息，请参阅《Amazon EC2 用户指南》中的更改实例类型的兼容性。

让我们使用在上一步中进行修改的模板更改实例类型。由于 `InstanceType` 是模板的输入参数，我们不需要修改模板；我们能在指定堆栈详细信息页面上更改参数值。

要使用新的参数值更新堆栈

1. 在 CloudFormation 控制台中，选择您的 **`UpdateTutorial`** 堆栈。
2. 选择更新、直接更新。
3. 选择使用当前模板，然后选择下一步。
4. 在指定堆栈详细信息页面上，将文本框中 InstanceType 的值从 `t3.micro` 更改为 `m5.large`。然后，选择两次下一步。
5. 在审核页面上，审核您的更改。在更改下，您应该会看到 CloudFormation 将更新 `WebServerInstance` 资源。
6. 选择提交。

对于 EBS 支持的 Amazon EC2 实例，可以通过启动和停止实例来动态更改实例类型。CloudFormation 会尝试通过更新实例类型并重启实例来优化更改，因此实例 ID 不会更改。但是，实例重启时，实例的公用 IP 地址会更改。为了确保在更改后正确绑定弹性 IP 地址，CloudFormation 还会更新弹性 IP 地址。您可以在事件选项卡上的 CloudFormation 控制台中看到更改。

要通过 Amazon Web Services 管理控制台 检查实例类型，请打开 Amazon EC2 控制台并在其中查找您的实例。

第 5 步：更新 AMI

现在，让我们更新堆栈以使用下一代 Amazon Linux，即 Amazon Linux 2023。

更新 AMI 是一项重大更改，需要更换实例。我们不能只通过启动和停止实例来修改 AMI；CloudFormation 会将此视为对资源不可变属性的更改。要更改不可变属性进，CloudFormation 必须启动替代资源，在此例中是运行新 AMI 的新 Amazon EC2 实例。

让我们来看看如何更新堆栈模板以使用 Amazon Linux 2023。关键更改包括更新 AMI 参数和从 yum 更改为 dnf 程序包管理器。

```
AWSTemplateFormatVersion: 2010-09-09

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2023 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues:
      - t3.nano
      - t3.micro
      - t3.small
      - t3.medium
      - t3a.nano
      - t3a.micro
      - t3a.small
      - t3a.medium
      - m5.large
      - m5.xlarge
      - m5.2xlarge
      - m5a.large
      - m5a.xlarge
      - m5a.2xlarge
      - c5.large
      - c5.xlarge
      - c5.2xlarge
      - r5.large
      - r5.xlarge
      - r5.2xlarge
      - r5a.large
      - r5a.xlarge
      - r5a.2xlarge
    ConstraintDescription: must be a valid EC2 instance type.

  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance
```

```
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.

  SSHLocation:
    Description: The IP address that can be used to SSH to the EC2 instances in CIDR
 format (e.g. 203.0.113.1/32)
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      KeyName: !Ref KeyName
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          # Get the latest CloudFormation package
          dnf update -y aws-cfn-bootstrap
          # Run cfn-init
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 WebServerInstance --region ${AWS::Region} || error_exit 'Failed to run cfn-init'

          # Start up the cfn-hup daemon to listen for changes to the EC2 instance
 metadata
          /opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'
          # Signal success or failure
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerInstance --region ${AWS::Region}
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:
            dnf:
              httpd: []
              php: []
```

```
          files:
            /var/www/html/index.php:
              content: |
                <?php
                echo "<h1>Hello World!</h1>";
                echo "<p>This is an updated version of our application.</p>";
                echo "<p>Running on Amazon Linux 2023!</p>";
                ?>
              mode: '000644'
              owner: apache
              group: apache
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
                # The interval used to check for changes to the resource metadata in
 minutes. Default is 15
                interval=2
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
                action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --
region ${AWS::Region}
                runas=root
          services:
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
              cfn-hup:
                enabled: true
                ensureRunning: true
                files:
                  - /etc/cfn/cfn-hup.conf
                  - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    CreationPolicy:
      ResourceSignal:
        Timeout: PT5M
```

```
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref SSHLocation

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServerInstance.PublicDnsName}'
    Description: URL of the web application
```

使用与 第 2 步：更新应用程序 中所述的相同步骤更新堆栈。

在新实例运行之后，CloudFormation 将更新堆栈中的其他资源以指向新资源。创建所有新资源并删除旧资源的过程称为 UPDATE_CLEANUP。此时，您将注意到堆栈中实例的实例 ID 和应用程序 URL 已随着更新而更改。事件表中的事件包含描述“Requested update has a change to an immutable property and hence creating a new physical resource”，以指示资源已被替代。

或者：如果您已将应用程序代码写入您想更新的 AMI 中，您可以使用同一堆栈更新机制更新 AMI 以加载您的新应用程序。

要使用自定义应用程序代码更新 AMI

1. 创建含有应用程序或操作系统更改的新 AMI。有关更多信息，请参阅《Amazon EC2 用户指南》中的创建 Amazon EBS-backed AMI。
2. 更新您的模板以合并新 AMI ID。
3. 使用与 第 2 步：更新应用程序 中所述的相同步骤更新堆栈。

在您更新堆栈时，CloudFormation 检测到 AMI ID 已更改，然后用我们启动前一个更新所使用的方法触发堆栈更新。

可用性和影响注意事项

不同的属性会对堆栈中的资源造成不同的影响。您可以使用 CloudFormation 更新任何属性，但是您应该在进行任何更改之前考虑以下问题：

1. 更新会如何影响资源本身？ 例如，更新警报阈值会使警报在更新期间处于非活动状态。正如我们所见，更改实例类型时需要停止并重启实例。CloudFormation 使用底层资源的更新或修改操作来对资源进行更改。要了解更改的影响，您应该查看特定资源的文档。
2. 更改可变还是不可变？ 对资源属性的某些更改，如更改 Amazon EC2 实例上的 AMI，不受基础服务的支持。如果更改可变，CloudFormation 将使用适用于基础资源的“Update”或“Modify”类型 API。对于不可变的属性更改，CloudFormation 将用更新后的属性创建新资源，然后再删除旧资源之前将此资源链接至堆栈。虽然 CloudFormation 尝试减少堆栈资源的停机时间，但替代资源是一个多步骤过程，需要时间。重新配置堆栈期间，您的应用程序不能全面运行。例如，它可能不能为请求提供服务或访问数据库。

相关资源

有关使用 CloudFormation 启动应用程序的更多信息以及集成其他配置与 Puppet 和 Opscode Chef 等部署服务的更多信息，请参阅以下白皮书：

- 通过 CloudFormation 启动应用程序
- 将 CloudFormation 与 Opscode Chef 集成
- 将 CloudFormation 与 Puppet 集成

创建已扩展且负载均衡的应用程序

在本演练中，您将创建一个堆栈，该堆栈可帮助您设置已扩展且负载平衡的应用程序。此演练为您提供将用于创建堆栈的示例模板。此示例模板预置了自动扩缩组、应用程序负载均衡器、控制负载均衡器和自动扩缩组流量的安全组，以及用于发布有关扩展活动的通知的 Amazon SNS 通知配置。

本模板将创建一个或多个 Amazon EC2 实例和一个应用程序负载均衡器。如果您通过本模板创建堆栈，则需为使用的 Amazon 资源支付相应费用。

完整堆栈模板

我们从使用模板开始。

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  InstanceType:
    Description: The EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues:
      - t3.micro
      - t3.small
      - t3.medium
  KeyName:
    Description: Name of an existing EC2 key pair to allow SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'
  OperatorEmail:
    Description: The email address to notify when there are any scaling activities
    Type: String
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  Subnets:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: At least two public subnets in different Availability Zones in the
 selected VPC
  VPC:
    Type: AWS::EC2::VPC::Id
    Description: A virtual private cloud (VPC) that enables resources in public subnets
 to connect to the internet
Resources:
  ELBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: ELB Security Group
      VpcId: !Ref VPC
      SecurityGroupIngress:
      - IpProtocol: tcp
```

```
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
  EC2SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: EC2 Security Group
      VpcId: !Ref VPC
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        SourceSecurityGroupId:
          Fn::GetAtt:
          - ELBSecurityGroup
          - GroupId
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref SSHLocation
  EC2TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 30
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 15
      HealthyThresholdCount: 5
      Matcher:
        HttpCode: '200'
      Name: EC2TargetGroup
      Port: 80
      Protocol: HTTP
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: '20'
      UnhealthyThresholdCount: 3
      VpcId: !Ref VPC
  ALBListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref EC2TargetGroup
      LoadBalancerArn: !Ref ApplicationLoadBalancer
```

```
      Port: 80
      Protocol: HTTP
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      Subnets: !Ref Subnets
      SecurityGroups:
        - !GetAtt ELBSecurityGroup.GroupId
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: !Ref LatestAmiId
        InstanceType: !Ref InstanceType
        KeyName: !Ref KeyName
        SecurityGroupIds:
          - !Ref EC2SecurityGroup
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            yum update -y
            yum install -y httpd
            systemctl start httpd
            systemctl enable httpd
            echo "<h1>Hello World!</h1>" > /var/www/html/index.html
  NotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !Ref OperatorEmail
          Protocol: email
  WebServerGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref LaunchTemplate
        Version: !GetAtt LaunchTemplate.LatestVersionNumber
      MaxSize: '3'
      MinSize: '1'
      NotificationConfigurations:
        - TopicARN: !Ref NotificationTopic
```

```
          NotificationTypes: ['autoscaling:EC2_INSTANCE_LAUNCH',
 'autoscaling:EC2_INSTANCE_LAUNCH_ERROR', 'autoscaling:EC2_INSTANCE_TERMINATE',
 'autoscaling:EC2_INSTANCE_TERMINATE_ERROR']
      TargetGroupARNs:
        - !Ref EC2TargetGroup
      VPCZoneIdentifier: !Ref Subnets
```

JSON

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Parameters":{
    "InstanceType":{
      "Description":"The EC2 instance type",
      "Type":"String",
      "Default":"t3.micro",
      "AllowedValues":[
        "t3.micro",
        "t3.small",
        "t3.medium"
      ]
    },
    "KeyName":{
      "Description":"Name of an existing EC2 key pair to allow SSH access to the
 instances",
      "Type":"AWS::EC2::KeyPair::KeyName"
    },
    "LatestAmiId":{
      "Description":"The latest Amazon Linux 2 AMI from the Parameter Store",
      "Type":"AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
      "Default":"/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
    },
    "OperatorEmail":{
      "Description":"The email address to notify when there are any scaling
 activities",
      "Type":"String"
    },
    "SSHLocation":{
      "Description":"The IP address range that can be used to SSH to the EC2
 instances",
      "Type":"String",
      "MinLength":9,
      "MaxLength":18,
```

```
      "Default":"0.0.0.0/0",
      "ConstraintDescription":"Must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "Subnets":{
      "Type":"List<AWS::EC2::Subnet::Id>",
      "Description":"At least two public subnets in different Availability Zones in the
selected VPC"
    },
    "VPC":{
      "Type":"AWS::EC2::VPC::Id",
      "Description":"A virtual private cloud (VPC) that enables resources in public
subnets to connect to the internet"
    }
  },
  "Resources":{
    "ELBSecurityGroup":{
      "Type":"AWS::EC2::SecurityGroup",
      "Properties":{
        "GroupDescription":"ELB Security Group",
        "VpcId":{
          "Ref":"VPC"
        },
        "SecurityGroupIngress":[
          {
            "IpProtocol":"tcp",
            "FromPort":80,
            "ToPort":80,
            "CidrIp":"0.0.0.0/0"
          }
        ]
      }
    },
    "EC2SecurityGroup":{
      "Type":"AWS::EC2::SecurityGroup",
      "Properties":{
        "GroupDescription":"EC2 Security Group",
        "VpcId":{
          "Ref":"VPC"
        },
        "SecurityGroupIngress":[
          {
            "IpProtocol":"tcp",
            "FromPort":80,
            "ToPort":80,
```

```
            "SourceSecurityGroupId":{
              "Fn::GetAtt":[
                "ELBSecurityGroup",
                "GroupId"
              ]
            }
          },
          {
            "IpProtocol":"tcp",
            "FromPort":22,
            "ToPort":22,
            "CidrIp":{
              "Ref":"SSHLocation"
            }
          }
        ]
      }
    },
    "EC2TargetGroup":{
      "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties":{
        "HealthCheckIntervalSeconds":30,
        "HealthCheckProtocol":"HTTP",
        "HealthCheckTimeoutSeconds":15,
        "HealthyThresholdCount":5,
        "Matcher":{
          "HttpCode":"200"
        },
        "Name":"EC2TargetGroup",
        "Port":80,
        "Protocol":"HTTP",
        "TargetGroupAttributes":[
          {
            "Key":"deregistration_delay.timeout_seconds",
            "Value":"20"
          }
        ],
        "UnhealthyThresholdCount":3,
        "VpcId":{
          "Ref":"VPC"
        }
      }
    },
    "ALBListener":{
```

```
      "Type":"AWS::ElasticLoadBalancingV2::Listener",
      "Properties":{
        "DefaultActions":[
          {
            "Type":"forward",
            "TargetGroupArn":{
              "Ref":"EC2TargetGroup"
            }
          }
        ],
        "LoadBalancerArn":{
          "Ref":"ApplicationLoadBalancer"
        },
        "Port":80,
        "Protocol":"HTTP"
      }
    },
    "ApplicationLoadBalancer":{
      "Type":"AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties":{
        "Scheme":"internet-facing",
        "Subnets":{
          "Ref":"Subnets"
        },
        "SecurityGroups":[
          {
            "Fn::GetAtt":[
              "ELBSecurityGroup",
              "GroupId"
            ]
          }
        ]
      }
    },
    "LaunchTemplate":{
      "Type":"AWS::EC2::LaunchTemplate",
      "Properties":{
        "LaunchTemplateName":{
          "Fn::Sub":"${AWS::StackName}-launch-template"
        },
        "LaunchTemplateData":{
          "ImageId":{
            "Ref":"LatestAmiId"
          },
```

```
        "InstanceType":{
          "Ref":"InstanceType"
        },
        "KeyName":{
          "Ref":"KeyName"
        },
        "SecurityGroupIds":[
          {
            "Ref":"EC2SecurityGroup"
          }
        ],
        "UserData":{
          "Fn::Base64":{
            "Fn::Join":[
              "",
              [
                "#!/bin/bash\n",
                "yum update -y\n",
                "yum install -y httpd\n",
                "systemctl start httpd\n",
                "systemctl enable httpd\n",
                "echo \"<h1>Hello World!</h1>\" > /var/www/html/index.html"
              ]
            ]
          }
        }
      }
    },
    "NotificationTopic":{
      "Type":"AWS::SNS::Topic",
      "Properties":{
        "Subscription":[
          {
            "Endpoint":{
              "Ref":"OperatorEmail"
            },
            "Protocol":"email"
          }
        ]
      }
    },
    "WebServerGroup":{
      "Type":"AWS::AutoScaling::AutoScalingGroup",
```

```
      "Properties":{
        "LaunchTemplate":{
          "LaunchTemplateId":{
            "Ref":"LaunchTemplate"
          },
          "Version":{
            "Fn::GetAtt":[
              "LaunchTemplate",
              "LatestVersionNumber"
            ]
          }
        },
        "MaxSize":"3",
        "MinSize":"1",
        "NotificationConfigurations":[
          {
            "TopicARN":{
              "Ref":"NotificationTopic"
            },
            "NotificationTypes":[
              "autoscaling:EC2_INSTANCE_LAUNCH",
              "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
              "autoscaling:EC2_INSTANCE_TERMINATE",
              "autoscaling:EC2_INSTANCE_TERMINATE_ERROR"
            ]
          }
        ],
        "TargetGroupARNs":[
          {
            "Ref":"EC2TargetGroup"
          }
        ],
        "VPCZoneIdentifier":{
          "Ref":"Subnets"
        }
      }
    }
  }
}
```

模板演练

此模板的第一部分指定了 Parameters。必须向每个参数分配一个运行时的值，使 Amazon CloudFormation 能够成功预置堆栈。稍后在模板中指定的资源会引用这些值并使用这些数据。

- InstanceType：Amazon EC2 Auto Scaling 预置的 EC2 实例类型。如果未指定，则默认使用 t3.micro。
- KeyName：用于允许 SSH 访问实例的现有 EC2 密钥对。
- LatestAmiId：实例的亚马逊机器映像（AMI）。如果未指定，则您的实例将使用由 Amazon 维护的 Amazon Systems Manager 公有参数，通过 Amazon Linux 2 AMI 启动。有关更多信息，请参阅 Amazon Systems Manager User Guide 中的 Finding public parameters。
- OperatorEmail：您要发送扩展活动通知的电子邮件地址。
- SSHLocation：可用于通过 SSH 连接到实例的 IP 地址范围。
- Subnets：位于不同可用区的至少两个公有子网。
- VPC：您账户中的虚拟私有云（VPC），其允许公有子网中的资源连接到互联网。

> **Note**
>
> 您可以使用默认 VPC 和默认子网来允许实例访问互联网。如果使用您自己的 VPC，请确保它拥有映射到您工作时所在区域的每个可用区的子网。您至少必须具有两个公有子网，且这些子网可用于创建负载均衡器。

此模板的下一个部分指定了 Resources。本部分指定堆栈资源及其属性。

AWS::EC2::SecurityGroup 资源 ELBSecurityGroup

- SecurityGroupIngress 包含一个 TCP 入口规则，该规则允许来自端口 80 上的所有 IP 地址（"CidrIp" : "0.0.0.0/0"）的访问。

AWS::EC2::SecurityGroup 资源 EC2SecurityGroup

- SecurityGroupIngress 包含两个入口规则：1) 允许来自您为 SSHLocation 输入参数提供的 IP 地址范围内的 SSH 访问（端口 22）的 TCP 入口规则；2) 允许通过指定负载均衡器的安全组实现的来自负载均衡器的访问的 TCP 入口规则。GetAtt 函数用于获取具有逻辑名称 ELBSecurityGroup 的安全组 ID。

AWS::ElasticLoadBalancingV2::TargetGroup 资源 EC2TargetGroup

- Port、Protocol 和 HealthCheckProtocol 指定 ApplicationLoadBalancer 要向其中路由流量以及 Elastic Load Balancing 用于检查 EC2 实例运行状况的 EC2 实例端口（80）和协议（HTTP）。
- HealthCheckIntervalSeconds 指定 EC2 实例的每次运行状况检查之间有 30 秒的时间间隔。HealthCheckTimeoutSeconds 定义为 Elastic Load Balancing 等待来自运行状况检查目标响应的时间（本示例中为 15 秒）。超时时间超过之后，Elastic Load Balancing 将标记 EC2 实例运行状况检查为“运行状况不佳”。当 EC2 实例连续三次未通过运行状况检查 (UnhealthyThresholdCount) 时，Elastic Load Balancing 会停止将流量路由到该 EC2 实例，直到该实例连续五次运行状况检查均正常 (HealthyThresholdCount)。此时，Elastic Load Balancing 认为实例运行状况良好，并再次开始将流量路由到该实例。
- TargetGroupAttributes 将目标组的取消注册延迟值更新为 20 秒。默认情况下，Elastic Load Balancing 会等待 300 秒，才会完成注销过程。

AWS::ElasticLoadBalancingV2::Listener 资源 ALBListener

- DefaultActions 指定负载均衡器侦听的端口、负载均衡器转发请求的目标组以及用于路由请求的协议。

AWS::ElasticLoadBalancingV2::LoadBalancer 资源 ApplicationLoadBalancer

- Subnets 将 Subnets 输入参数的值作为要在其中创建负载均衡器节点的公有子网列表。
- SecurityGroup 获取安全组的 ID，该安全组充当虚拟防火墙，以便负载均衡器节点控制传入流量。GetAtt 函数用于获取具有逻辑名称 ELBSecurityGroup 的安全组 ID。

AWS::EC2::LaunchTemplate 资源 LaunchTemplate

- ImageId 将 LatestAmiId 输入参数的值作为要使用的 AMI。
- KeyName 将 KeyName 输入参数的值作为要使用的 EC2 密钥对。
- SecurityGroupIds 获取逻辑名称为 EC2SecurityGroup 的安全组的 ID，该安全组充当虚拟防火墙，以便 EC2 实例控制传入流量。
- UserData 是在实例启动并运行之后运行的配置脚本。在此示例中，脚本安装 Apache 并创建 index.html 文件。

AWS::SNS::Topic 资源 NotificationTopic

- 当有任何扩展活动时，Subscription 将 OperatorEmail 输入参数的值作为通知收件人的电子邮件地址。

AWS::AutoScaling::AutoScalingGroup 资源 WebServerGroup

- MinSize 和 MaxSize 设置自动扩缩组中的 EC2 实例的最小和最大数量。
- TargetGroupARNs 使用逻辑名称为 EC2TargetGroup 的目标组的 ARN。随着此自动扩缩组的扩展，它会自动向该目标组注册和注销实例。
- VPCZoneIdentifier 将 Subnets 输入参数的值作为要在其中创建 EC2 实例的公有子网列表。

步骤 1：启动堆栈

在启动堆栈之前，请检查您是否拥有可以使用以下所有服务的 Amazon Identity and Access Management（IAM）权限：Amazon EC2、Amazon EC2 Auto Scaling、Amazon Systems Manager、Elastic Load Balancing、Amazon SNS 和 Amazon CloudFormation。

以下过程涉及从文件上传示例堆栈模板。在本地计算机上打开文本编辑器并添加其中一个模板。使用文件名 sampleloadbalancedappstack.template 保存该文件。

启动堆栈模板

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 依次选择创建堆栈和使用新资源（标准）。
3. 在指定模板下，选择上传模板文件，然后选择选择文件，上传 sampleloadbalancedappstack.template 文件。
4. 选择下一步。
5. 在指定堆栈详细信息页面上，键入堆栈的名称（例如 **SampleLoadBalancedAppStack**）。
6. 在参数下，查看堆栈的参数并为没有默认值的所有参数提供值，包括 OperatorEmail、SSHLocation、KeyName、VPC 和 Subnets。
7. 选择下一步两次。
8. 在审核页面上，审核并确认设置。
9. 选择提交。

您可以在 Amazon CloudFormation 控制台的状态列中查看堆栈的状态。Amazon CloudFormation 成功创建堆栈后，您将收到 CREATE_COMPLETE 状态。

> **Note**
>
> 创建堆栈后，必须先确认订阅，电子邮件地址才能开始接收通知。有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的自动扩缩组扩展时获取 Amazon SNS 通知。

步骤 2：清除示例资源

为确保您无需为未使用的示例资源付费，请删除堆栈。

删除堆栈

1. 在 Amazon CloudFormation 控制台中，选择 SampleLoadBalancedAppStack 堆栈。
2. 选择删除。
3. 在确认消息中，选择删除堆栈。

   SampleLoadBalancedAppStack 的状态更改为 DELETE_IN_PROGRESS。当 Amazon CloudFormation 完成删除堆栈后，它会将从列表中移堆栈除。

使用本演练中的示例模板构建您自己的堆栈模板。有关更多信息，请参阅 Amazon EC2 Auto Scaling User Guide 中的 Tutorial: Set up a scaled and load-balanced application。

与其他 Amazon Web Services 账户中的 VPC 建立对等连接

您可以使用 AWS::EC2::VPCPeeringConnection 与其他 Amazon Web Services 账户中的虚拟私有云（VPC）建立对等连接。这会在两个 VPC 之间创建网络连接，使您能够在它们之间路由流量，它们可以像在同一网络中那样进行通信。VPC 对等连接有助于简化数据访问和数据传输。

要建立 VPC 对等连接，您需要为单个 CloudFormation 堆栈中的两个单独 Amazon Web Services 账户授权。

有关 VPC 对等连接及其限制的更多信息，请参阅 Amazon VPC 对等连接指南。

先决条件

1. 您需要为对等连接提供对等 VPC ID、对等 Amazon Web Services 账户 ID 以及跨账户访问角色。

   > **Note**
   >
   > 本演练涉及两个账户：第一个是允许跨账户对等的账户 (接受方账户)。第二个是请求对等连接的账户 (请求者账户)。

2. 要接受 VPC 对等连接，您必须能够担任跨账户访问角色。该资源的行为方式与同一账户中的 VPC 对等连接资源相同。要了解 IAM 管理员如何授予代入跨账户角色的权限，请参阅 IAM 用户指南中的向用户授予切换角色的权限。

步骤 1：创建 VPC 和跨账户角色

在该步骤中，您需要在接受方账户 中创建 VPC 和角色。

创建 VPC 和跨账户访问角色

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面，选择右上角的创建堆栈，然后选择使用新资源（标准）。
3. 在先决条件 – 准备模板中，选择选择现有模板，然后依次选择上传模板文件 > 选择文件。
4. 在本地计算机上打开文本编辑器并添加以下模板之一。保存文件并返回控制台，将其选择为模板文件。

   Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a VPC and an assumable role for cross account VPC
 peering.",
  "Parameters": {
    "PeerRequesterAccountId": {
      "Type": "String"
    }
  },
  "Resources": {
    "vpc": {
```

```
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "10.1.0.0/16",
        "EnableDnsSupport": false,
        "EnableDnsHostnames": false,
        "InstanceTenancy": "default"
      }
    },
    "peerRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Principal": {
                "AWS": {
                  "Ref": "PeerRequesterAccountId"
                }
              },
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow"
            }
          ]
        },
        "Path": "/",
        "Policies": [
          {
            "PolicyName": "root",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": "ec2:AcceptVpcPeeringConnection",
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    }
```

```
  },
  "Outputs": {
    "VPCId": {
      "Value": {
        "Ref": "vpc"
      }
    },
    "RoleARN": {
      "Value": {
        "Fn::GetAtt": [
          "peerRole",
          "Arn"
        ]
      }
    }
  }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and an assumable role for cross account VPC peering.
Parameters:
  PeerRequesterAccountId:
    Type: String
Resources:
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.1.0.0/16
      EnableDnsSupport: false
      EnableDnsHostnames: false
      InstanceTenancy: default
  peerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Principal:
              AWS: !Ref PeerRequesterAccountId
            Action:
              - 'sts:AssumeRole'
            Effect: Allow
```

```
      Path: /
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action: 'ec2:AcceptVpcPeeringConnection'
                Resource: '*'
Outputs:
  VPCId:
    Value: !Ref vpc
  RoleARN:
    Value: !GetAtt
      - peerRole
      - Arn
```

5. 选择下一步。
6. 为堆栈提供一个名称（例如 **VPC-owner**），然后在 PeerRequesterAccountId 字段中输入请求者账户的 Amazon Web Services 账户 ID。
7. 接受默认值，然后选择 Next。
8. 选择我确认 Amazon CloudFormation 可能创建 IAM 资源，然后选择创建堆栈。

步骤 2：创建包含 **AWS::EC2::VPCPeeringConnection** 的模板

现在，您已创建 VPC 和跨账户角色，您可以与使用另一个 Amazon Web Services 账户（请求者账户）的 VPC 进行对等。

创建包含 AWS::EC2::VPCPeeringConnection 资源的模板

1. 返回 Amazon CloudFormation 控制台主页。
2. 在堆栈页面，选择右上角的创建堆栈，然后选择使用新资源（标准）。
3. 在先决条件 – 准备模板中，选择选择现有模板，然后依次选择上传模板文件 > 选择文件。
4. 在本地计算机上打开文本编辑器并添加以下模板之一。保存文件并返回控制台，将其选择为模板文件。

   Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
```

```
 "Description": "Create a VPC and a VPC Peering connection using the PeerRole to
accept.",
 "Parameters": {
   "PeerVPCAccountId": {
     "Type": "String"
   },
   "PeerVPCId": {
     "Type": "String"
   },
   "PeerRoleArn": {
     "Type": "String"
   }
 },
 "Resources": {
   "vpc": {
     "Type": "AWS::EC2::VPC",
     "Properties": {
       "CidrBlock": "10.2.0.0/16",
       "EnableDnsSupport": false,
       "EnableDnsHostnames": false,
       "InstanceTenancy": "default"
     }
   },
   "vpcPeeringConnection": {
     "Type": "AWS::EC2::VPCPeeringConnection",
     "Properties": {
       "VpcId": {
         "Ref": "vpc"
       },
       "PeerVpcId": {
         "Ref": "PeerVPCId"
       },
       "PeerOwnerId": {
         "Ref": "PeerVPCAccountId"
       },
       "PeerRoleArn": {
         "Ref": "PeerRoleArn"
       }
     }
   }
 },
 "Outputs": {
   "VPCId": {
     "Value": {
```

```
        "Ref": "vpc"
      }
    },
    "VPCPeeringConnectionId": {
      "Value": {
        "Ref": "vpcPeeringConnection"
      }
    }
  }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and a VPC Peering connection using the PeerRole to
 accept.
Parameters:
  PeerVPCAccountId:
    Type: String
  PeerVPCId:
    Type: String
  PeerRoleArn:
    Type: String
Resources:
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.2.0.0/16
      EnableDnsSupport: false
      EnableDnsHostnames: false
      InstanceTenancy: default
  vpcPeeringConnection:
    Type: AWS::EC2::VPCPeeringConnection
    Properties:
      VpcId: !Ref vpc
      PeerVpcId: !Ref PeerVPCId
      PeerOwnerId: !Ref PeerVPCAccountId
      PeerRoleArn: !Ref PeerRoleArn
Outputs:
  VPCId:
    Value: !Ref vpc
  VPCPeeringConnectionId:
    Value: !Ref vpcPeeringConnection
```

5. 选择下一步。
6. 为堆栈提供一个名称（例如，**VPC-peering-connection**）。
7. 接受默认值，然后选择 Next。
8. 选择我确认 Amazon CloudFormation 可能创建 IAM 资源，然后选择创建堆栈。

创建具有高度限制策略的模板

在将您的 VPC 与另一个 Amazon Web Services 账户 对等时，您可能需要创建一个高度限制的策略。

以下示例模板显示如何更改 VPC 对等所有者模板 (上面步骤 1 中创建的接受方账户)，以加强对它的限制。

Example JSON

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Description":"Create a VPC and an assumable role for cross account VPC peering.",
  "Parameters":{
    "PeerRequesterAccountId":{
      "Type":"String"
    }
  },
  "Resources":{
    "peerRole":{
      "Type":"AWS::IAM::Role",
      "Properties":{
        "AssumeRolePolicyDocument":{
          "Statement":[
            {
              "Action":[
                "sts:AssumeRole"
              ],
              "Effect":"Allow",
              "Principal":{
                "AWS":{
                  "Ref":"PeerRequesterAccountId"
                }
              }
            }
          ]
        },
        "Path":"/",
```

```
        "Policies":[
          {
            "PolicyDocument":{
              "Statement":[
                {
                  "Action":"ec2:acceptVpcPeeringConnection",
                  "Effect":"Allow",
                  "Resource":{
                    "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}"
                  }
                },
                {
                  "Action":"ec2:acceptVpcPeeringConnection",
                  "Condition":{
                    "StringEquals":{
                      "ec2:AccepterVpc":{
                        "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/
${vpc}"
                      }
                    }
                  },
                  "Effect":"Allow",
                  "Resource":{
                    "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc-
peering-connection/*"
                  }
                }
              ],
              "Version":"2012-10-17"
            },
            "PolicyName":"root"
          }
        ]
      }
    },
    "vpc":{
      "Type":"AWS::EC2::VPC",
      "Properties":{
        "CidrBlock":"10.1.0.0/16",
        "EnableDnsHostnames":false,
        "EnableDnsSupport":false,
        "InstanceTenancy":"default"
      }
    }
```

```
  },
  "Outputs":{
    "RoleARN":{
      "Value":{
        "Fn::GetAtt":[
          "peerRole",
          "Arn"
        ]
      }
    },
    "VPCId":{
      "Value":{
        "Ref":"vpc"
      }
    }
  }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and an assumable role for cross account VPC peering.
Parameters:
  PeerRequesterAccountId:
    Type: String
Resources:
  peerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              AWS:
                Ref: PeerRequesterAccountId
      Path: /
      Policies:
        - PolicyDocument:
            Statement:
              - Action: 'ec2:acceptVpcPeeringConnection'
                Effect: Allow
```

```
                Resource:
                  'Fn::Sub': 'arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}'
              - Action: 'ec2:acceptVpcPeeringConnection'
                Condition:
                  StringEquals:
                    'ec2:AccepterVpc':
                      'Fn::Sub': 'arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/
${vpc}'
                Effect: Allow
                Resource:
                  'Fn::Sub': >-
                    arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc-peering-
connection/*
            Version: 2012-10-17
          PolicyName: root
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.1.0.0/16
      EnableDnsHostnames: false
      EnableDnsSupport: false
      InstanceTenancy: default
Outputs:
  RoleARN:
    Value:
      'Fn::GetAtt':
        - peerRole
        - Arn
  VPCId:
    Value:
      Ref: vpc
```

要访问 VPC，您可以使用上面步骤 2 中使用的请求者模板。

有关更多信息，请参阅《Amazon VPC 对等连接指南》中的VPC 对等连接的身份和访问权限管理。

通过 CodeDeploy 使用 CloudFormation 执行 ECS 蓝/绿部署

要更新在 Amazon Elastic Container Service（Amazon ECS）上运行的应用程序，您可以使用 CodeDeploy 蓝绿部署策略。此策略有助于尽可能减少因更改应用程序版本造成的中断。

在蓝绿部署中，您可以在当前的实时环境（称为蓝色）之外创建一个新的应用程序环境（称为绿色）。这使您可以先监控和测试绿色环境，然后再将实时流量从蓝色环境路由到绿色环境。在绿色环境开始为实时流量提供服务后，您可以安全地终止蓝色环境。

要使用 CloudFormation 在 ECS 上执行 CodeDeploy 蓝绿部署，请在堆栈模板中包括以下信息：

- 描述 `AWS::CodeDeploy::BlueGreen` 钩子的 `Hooks` 部分。
- 指定 `AWS::CodeDeployBlueGreen` 变换的 `Transform` 部分。

以下主题提供了有关为 ECS 上的蓝绿部署设置 CloudFormation 模板的指导。

主题

- 关于蓝绿部署
- 使用 CloudFormation 管理 ECS 蓝/绿部署时的注意事项
- AWS::CodeDeploy::BlueGreen 钩子语法
- 蓝绿部署模板示例

关于蓝绿部署

本主题概述了如何使用 CloudFormation 执行蓝绿部署。此外还介绍了如何准备用于蓝绿部署的 CloudFormation 模板。

主题

- 工作原理
- 会启动绿色部署的资源更新
- 准备模板以执行 ECS 蓝/绿部署
- 使用 CloudFormation 资源对蓝/绿部署建模
- 更改集
- 监控堆栈事件
- 蓝绿部署所需的 IAM 权限

工作原理

使用 CloudFormation 通过 CodeDeploy 执行 ECS 蓝绿部署时，首先创建一个堆栈模板，用于定义蓝色和绿色应用程序环境的资源，包括指定要使用的流量路由和稳定设置。然后基于该模板创建一个堆

栈。这会生成蓝色（当前）应用程序。CloudFormation 仅在堆栈创建过程中创建蓝色资源。绿色部署的资源仅在需要时才会创建。

然后，如果在将来的堆栈更新中更新蓝色应用程序中的任务定义或任务集资源，CloudFormation 会执行以下操作：

- 生成所有必需的绿色应用程序环境资源
- 根据指定的流量路由参数转移流量
- 删除蓝色资源

如果在绿色部署成功和完成之前的任何时刻发生错误，CloudFormation 会将堆栈回滚到启动整个绿色部署之前的状态。

会启动绿色部署的资源更新

执行将会更新特定 ECS 资源的特定属性的堆栈更新时，CloudFormation 将会启动绿色部署过程。将会启动此过程的资源有：

- AWS::ECS::TaskDefinition
- AWS::ECS::TaskSet

但是，如果对这些资源的更新不涉及需要替换的属性更改，则不会启动绿色部署。有关更多信息，请参阅 理解堆栈资源的更新行为。

请注意，不能在同一堆栈更新操作中同时包括对上述资源的更新以及对其他资源的更新。如果需要更新列出的资源以及同一堆栈中的其他资源，则有两种选择：

- 执行两个单独的堆栈更新操作：一个仅包括对上述资源的更新，另一个单独的堆栈更新包括对任何其他资源的更改。
- 从模板中删除 `Transform` 和 `Hooks` 部分，然后执行堆栈更新。在这种情况下，CloudFormation 不会执行绿色部署。

准备模板以执行 ECS 蓝/绿部署

要在堆栈上启用蓝/绿部署，请在执行堆栈更新之前在堆栈模板中包含以下部分。

- 将 `AWS::CodeDeployBlueGreen` 转换的引用添加到模板中：

```
"Transform": [
  "AWS::CodeDeployBlueGreen"
],
```

- 添加调用 AWS::CodeDeploy::BlueGreen 挂钩并指定部署属性的 Hooks 部分。有关更多信息，请参阅 AWS::CodeDeploy::BlueGreen 钩子语法。
- 在 Resources 部分中，定义部署的蓝色和绿色资源。

您可以在第一次创建模板时（即创建堆栈本身之前）添加这些部分，也可以在执行堆栈更新之前将它们添加到现有模板中。如果是为新堆栈指定蓝/绿部署，则 CloudFormation 只在堆栈创建期间创建蓝色资源 - 仅当在堆栈更新期间需要绿色部署的资源时才会创建它们。

使用 CloudFormation 资源对蓝/绿部署建模

要在 ECS 上执行 CodeDeploy 蓝绿部署，CloudFormation 模板需要包含为部署建模的资源，例如 Amazon ECS 服务和负载均衡器。如需更详细地了解这些资源代表了什么，请参阅《Amazon CodeDeploy 用户指南》中的开始 Amazon ECS 部署之前。

要求	资源	必需/可选	如果替换则会启动蓝绿部署？
Amazon ECS 集群	AWS::ECS::Cluster	可选。可以使用默认集群。	否
Amazon ECS 服务	AWS::ECS::Service	必需。	否
应用程序或网络负载均衡器	AWS::ECS::Service LoadBalancer	必需。	否
生产侦听器	AWS::ElasticLoadBalancingV2::Listener	必需。	否
测试侦听器	AWS::ElasticLoadBalancingV2::Listener	可选。	否
两个目标组	AWS::ElasticLoadBalancingV2::TargetGroup	必需。	否

要求	资源	必需/可选	如果替换则会启动蓝绿部署？
Amazon ECS 任务定义	AWS::ECS::TaskDefinition	必需。	是
您的 Amazon ECS 应用程序的容器	AWS::ECS::TaskDefinition Container Definition Name	必需。	否
您的替换任务集的端口	AWS::ECS::TaskDefinition PortMapping ContainerPort	必需。	否

更改集

我们强烈建议您在执行会启动绿色部署的堆栈更新之前创建更改集。这样便于您在执行堆栈更新之前查看对堆栈作出的实际更改。请注意，资源更改可能不会按堆栈更新期间的执行顺序列出。有关更多信息，请参阅 使用更改集更新 CloudFormation 堆栈。

无论您在蓝绿部署中使用更改集还是直接堆栈操作，CloudFormation 都会在部署前自动执行验证，以便在任何资源被配置之前捕获常见的模板错误。这包括对所有堆栈操作进行的属性语法验证和资源名称冲突检测，以及在创建更改集时以警告形式提供的其他检查（服务配额限制、记录器冲突以及 ECR 存储库删除就绪性）。

监控堆栈事件

您可以在 Stack（堆栈）页面的 Events（事件）选项卡上，使用 Amazon CLI 查看 ECS 部署的每个步骤生成的堆栈事件。有关更多信息，请参阅 监控堆栈进度。

蓝绿部署所需的 IAM 权限

为了使 CloudFormation 成功执行蓝绿部署，您必须具有以下 CodeDeploy 权限：

- `codedeploy:Get*`
- `codedeploy:CreateCloudFormationDeployment`

有关更多信息，请参阅《服务授权参考》中的 Actions, resources, and condition keys for CodeDeploy。

使用 CloudFormation 管理 ECS 蓝/绿部署时的注意事项

使用 CloudFormation 来通过 CodeDeploy 执行 ECS 蓝绿部署的过程，不同于仅使用 CodeDeploy 的标准 Amazon ECS 部署过程。要详细了解这些区别，请参阅《Amazon CodeDeploy 用户指南》中的通过 CodeDeploy 和 Amazon CloudFormation 进行的 Amazon ECS 蓝绿部署的区别。

使用 CloudFormation 管理蓝绿部署时，需要记住某些限制和注意事项：

- 只有对特定资源的更新才会启动绿色部署。有关更多信息，请参阅 会启动绿色部署的资源更新。
- 不能在同一堆栈更新中包括启动绿色部署的资源更新和对其他资源的更新。有关更多信息，请参阅会启动绿色部署的资源更新。
- 您只能将单个 ECS 服务指定为部署目标。
- 如果参数的值由 CloudFormation 模糊处理，则 CodeDeploy 无法在绿色部署期间更新该参数，并且会导致错误和堆栈更新失败。这些方法包括：
  - 使用 NoEcho 属性定义的参数。
  - 使用动态引用从外部服务检索其值的参数。有关动态引用的更多信息，请参阅使用动态引用获取存储在其他服务中的值。
- 要取消仍在进行的绿色部署，请取消 CloudFormation 中的堆栈更新，而不是 CodeDeploy 或 ECS。有关更多信息，请参阅 取消堆栈更新。更新完成之后，将无法取消它。但是，您可以利用任何先前的设置再次更新堆栈。
- 定义 ECS 蓝绿部署的模板目前不支持以下 CloudFormation 功能：
  - 声明 outputs 或使用 Fn::ImportValue 导入其他堆栈中的值。
  - 导入资源。有关资源导入的更多信息，请参阅将 Amazon 资源导入 CloudFormation 堆栈。
  - 您不能在包含嵌套堆栈资源的模板中使用 AWS::CodeDeploy::BlueGreen 钩子。有关嵌套堆栈的更多信息，请参阅使用嵌套堆栈将模板拆分为可重复使用的部分。
  - 在嵌套堆栈中使用 AWS::CodeDeploy::BlueGreen 钩子。

AWS::CodeDeploy::BlueGreen 钩子语法

以下语法描述了 ECS 蓝绿部署的 AWS::CodeDeploy::BlueGreen 钩子结构。

语法

```
"Hooks": {
  "Logical ID": {
    "Type": "AWS::CodeDeploy::BlueGreen",
    "Properties": {
```

```
      "TrafficRoutingConfig": {
        "Type": "Traffic routing type",
        "TimeBasedCanary": {
          "StepPercentage": Integer,
          "BakeTimeMins": Integer
        },
        "TimeBasedLinear": {
          "StepPercentage": Integer,
          "BakeTimeMins": Integer
        }
      },
      "AdditionalOptions": {"TerminationWaitTimeInMinutes": Integer},
      "LifecycleEventHooks": {
        "BeforeInstall": "FunctionName",
        "AfterInstall": "FunctionName",
        "AfterAllowTestTraffic": "FunctionName",
        "BeforeAllowTraffic": "FunctionName",
        "AfterAllowTraffic": "FunctionName"
      },
      "ServiceRole": "CodeDeployServiceRoleName",
      "Applications": [
        {
          "Target": {
            "Type": "AWS::ECS::Service",
            "LogicalID": "Logical ID of AWS::ECS::Service"
          },
          "ECSAttributes": {
            "TaskDefinitions": [
              "Logical ID of AWS::ECS::TaskDefinition (Blue)",
              "Logical ID of AWS::ECS::TaskDefinition (Green)"
            ],
            "TaskSets": [
              "Logical ID of AWS::ECS::TaskSet (Blue)",
              "Logical ID of AWS::ECS::TaskSet (Green)"
            ],
            "TrafficRouting": {
              "ProdTrafficRoute": {
                "Type": "AWS::ElasticLoadBalancingV2::Listener",
                "LogicalID": "Logical ID of AWS::ElasticLoadBalancingV2::Listener
(Production)"
              },
              "TestTrafficRoute": {
                "Type": "AWS::ElasticLoadBalancingV2::Listener",
```

```
                "LogicalID": "Logical ID of AWS::ElasticLoadBalancingV2::Listener
 (Test)"
              },
              "TargetGroups": [
                "Logical ID of AWS::ElasticLoadBalancingV2::TargetGroup (Blue)",
                "Logical ID of AWS::ElasticLoadBalancingV2::TargetGroup (Green)"
              ]
            }
          }
        }
      ]
    }
  }
}
```

属性

逻辑 ID（也称为逻辑名称）

在模板的 Hooks 部分中声明的钩子。逻辑 ID 必须为字母数字 (A-Za-z0-9)，并且在模板中具有唯一性。

是否必需：是

Type

挂钩的类型。AWS::CodeDeploy::BlueGreen

是否必需：是

Properties

挂钩的属性。

是否必需：是

TrafficRoutingConfig

流量路由配置设置。

必需：否

默认配置为基于时间的 canary 流量转移，具有 15% 的步骤百分比和五分钟的稳定时间。

Type

部署配置使用的流量转移类型。

有效值：AllAtOnce | TimeBasedCanary | TimeBasedLinear

是否必需：是

TimeBasedCanary

指定一个配置，以两个增量将流量从部署的一个版本转移到另一个版本。

必需：如果指定 TimeBasedCanary 作为流量路由类型，则必须包含 TimeBasedCanary 参数。

StepPercentage

在 TimeBasedCanary 部署的第一个增量中要转移的流量百分比。步骤百分比必须为 14% 或更大。

必需：否

BakeTimeMins

TimeBasedCanary 部署的第一次和第二次流量转移之间的分钟数。

必需：否

TimeBasedLinear

指定一个配置，以相等的增量将流量从部署的一个版本转移到另一个版本，每个增量之间的分钟数相等。

必需：如果指定 TimeBasedLinear 作为流量路由类型，则必须包含 TimeBasedLinear 参数。

StepPercentage

在 TimeBasedLinear 部署的每个增量开始时转移的流量百分比。步骤百分比必须为 14% 或更大。

必需：否

BakeTimeMins

TimeBasedLinear 部署的每次增量流量转移之间的分钟数。

必需：否

AdditionalOptions

蓝/绿部署的其他选项。

必需：否

`TerminationWaitTimeInMinutes`

指定终止蓝色资源之前等待的时间（以分钟为单位）。

必需：否

`LifecycleEventHooks`

使用生命周期事件挂钩指定 CodeDeploy 可以调用的用于验证部署的 Lambda 函数。对于部署生命周期事件，您可以使用相同函数或不同函数。验证测试完成后，Lambda `AfterAllowTraffic` 函数会回调 CodeDeploy 并提供 `Succeeded` 或 `Failed` 结果。有关更多信息，请参阅《Amazon CodeDeploy 用户指南》中的 AppSpec 的“hooks”部分。

必需：否

`BeforeInstall`

用于在创建替换任务集之前运行任务的函数。

必需：否

`AfterInstall`

用于在创建替换任务集并且其中一个目标组与之关联后运行任务的函数。

必需：否

`AfterAllowTestTraffic`

用于在测试侦听器为替换任务集提供流量后运行任务的函数。

必需：否

`BeforeAllowTraffic`

用于在第二个目标组与替换任务集关联之后但在流量转移到替换任务集之前运行任务的函数。

必需：否

`AfterAllowTraffic`

用于在第二个目标组为替换任务集提供流量后运行任务的函数。

必需：否

`ServiceRole`

CloudFormation 用于执行蓝绿部署的执行角色。有关必要权限的列表，请参阅 蓝绿部署所需的 IAM 权限。

必需：否

`Applications`

指定 Amazon ECS 应用程序的属性。

是否必需：是

`Target`

是否必需：是

`Type`

资源的类型。

是否必需：是

`LogicalID`

资源的逻辑 ID。

是否必需：是

`ECSAttributes`

代表 Amazon ECS 应用程序部署的各种需求的资源。

是否必需：是

`TaskDefinitions`

运行包含 Amazon ECS 应用程序的 Docker 容器所用的 AWS::ECS::TaskDefinition 资源的逻辑 ID。

是否必需：是

`TaskSets`

用作应用程序任务集的 AWS::ECS::TaskSet 资源的逻辑 ID。

是否必需：是

TrafficRouting

指定用于流量路由的资源。

是否必需：是

ProdTrafficRoute

负载均衡器将流量定向到目标组所用的侦听器。

是否必需：是

Type

资源的类型。AWS::ElasticLoadBalancingV2::Listener

是否必需：是

LogicalID

资源的逻辑 ID。

是否必需：是

TestTrafficRoute

负载均衡器将流量定向到目标组所用的侦听器。

是否必需：是

Type

资源的类型。AWS::ElasticLoadBalancingV2::Listener

是否必需：是

LogicalID

资源的逻辑 ID。

必需：否

TargetGroups

用作目标组以将流量传送到注册目标的资源的逻辑 ID。

是否必需：是

蓝绿部署模板示例

以下示例模板设置了 ECS 上的 CodeDeploy 蓝绿部署，流量路由进度为每个步骤 15%，每个步骤之间的稳定期为 5 分钟。

使用模板创建堆栈将会预置部署的初始配置。如果您随后对 `BlueTaskSet` 资源中需要替换的属性进行了任何更改，则 CloudFormation 会在堆栈更新过程中启动绿色部署。

JSON

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Parameters":{
    "Vpc":{ "Type":"AWS::EC2::VPC::Id" },
    "Subnet1":{ "Type":"AWS::EC2::Subnet::Id" },
    "Subnet2":{ "Type":"AWS::EC2::Subnet::Id" }
  },
  "Transform":[ "AWS::CodeDeployBlueGreen" ],
  "Hooks":{
    "CodeDeployBlueGreenHook":{
      "Type":"AWS::CodeDeploy::BlueGreen",
      "Properties":{
        "TrafficRoutingConfig":{
          "Type":"TimeBasedCanary",
          "TimeBasedCanary":{
            "StepPercentage":15,
            "BakeTimeMins":5
          }
        },
        "Applications":[
          {
            "Target":{
              "Type":"AWS::ECS::Service",
              "LogicalID":"ECSDemoService"
            },
            "ECSAttributes":{
              "TaskDefinitions":[ "BlueTaskDefinition","GreenTaskDefinition" ],
              "TaskSets":[ "BlueTaskSet","GreenTaskSet" ],
              "TrafficRouting":{
                "ProdTrafficRoute":{
                  "Type":"AWS::ElasticLoadBalancingV2::Listener",
                  "LogicalID":"ALBListenerProdTraffic"
                },
```

```
                "TargetGroups":[ "ALBTargetGroupBlue","ALBTargetGroupGreen" ]
              }
            }
          }
        ]
      }
    }
  },
  "Resources":{
    "ExampleSecurityGroup":{
      "Type":"AWS::EC2::SecurityGroup",
      "Properties":{
        "GroupDescription":"Security group for ec2 access",
        "VpcId":{ "Ref":"Vpc" },
        "SecurityGroupIngress":[
          {
            "IpProtocol":"tcp",
            "FromPort":80,
            "ToPort":80,
            "CidrIp":"0.0.0.0/0"
          },
          {
            "IpProtocol":"tcp",
            "FromPort":8080,
            "ToPort":8080,
            "CidrIp":"0.0.0.0/0"
          },
          {
            "IpProtocol":"tcp",
            "FromPort":22,
            "ToPort":22,
            "CidrIp":"0.0.0.0/0"
          }
        ]
      }
    },
    "ALBTargetGroupBlue":{
      "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties":{
        "HealthCheckIntervalSeconds":5,
        "HealthCheckPath":"/",
        "HealthCheckPort":"80",
        "HealthCheckProtocol":"HTTP",
        "HealthCheckTimeoutSeconds":2,
```

```
        "HealthyThresholdCount":2,
        "Matcher":{ "HttpCode":"200" },
        "Port":80,
        "Protocol":"HTTP",
        "Tags":[{ "Key":"Group","Value":"Example" }],
        "TargetType":"ip",
        "UnhealthyThresholdCount":4,
        "VpcId":{ "Ref":"Vpc" }
      }
    },
    "ALBTargetGroupGreen":{
      "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties":{
        "HealthCheckIntervalSeconds":5,
        "HealthCheckPath":"/",
        "HealthCheckPort":"80",
        "HealthCheckProtocol":"HTTP",
        "HealthCheckTimeoutSeconds":2,
        "HealthyThresholdCount":2,
        "Matcher":{ "HttpCode":"200" },
        "Port":80,
        "Protocol":"HTTP",
        "Tags":[{ "Key":"Group","Value":"Example" }],
        "TargetType":"ip",
        "UnhealthyThresholdCount":4,
        "VpcId":{ "Ref":"Vpc" }
      }
    },
    "ExampleALB":{
      "Type":"AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties":{
        "Scheme":"internet-facing",
        "SecurityGroups":[{ "Ref":"ExampleSecurityGroup" }],
        "Subnets":[{ "Ref":"Subnet1" },{ "Ref":"Subnet2" }],
        "Tags":[{ "Key":"Group","Value":"Example" }],
        "Type":"application",
        "IpAddressType":"ipv4"
      }
    },
    "ALBListenerProdTraffic":{
      "Type":"AWS::ElasticLoadBalancingV2::Listener",
      "Properties":{
        "DefaultActions":[
          {
```

```
            "Type":"forward",
            "ForwardConfig":{
              "TargetGroups":[
                {
                  "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" },
                  "Weight":1
                }
              ]
            }
          }
        ],
        "LoadBalancerArn":{ "Ref":"ExampleALB" },
        "Port":80,
        "Protocol":"HTTP"
      }
    },
    "ALBListenerProdRule":{
      "Type":"AWS::ElasticLoadBalancingV2::ListenerRule",
      "Properties":{
        "Actions":[
          {
            "Type":"forward",
            "ForwardConfig":{
              "TargetGroups":[
                {
                  "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" },
                  "Weight":1
                }
              ]
            }
          }
        ],
        "Conditions":[
          {
            "Field":"http-header",
            "HttpHeaderConfig":{
              "HttpHeaderName":"User-Agent",
              "Values":[ "Mozilla" ]
            }
          }
        ],
        "ListenerArn":{ "Ref":"ALBListenerProdTraffic" },
        "Priority":1
      }
```

```
    },
    "ECSTaskExecutionRole":{
      "Type":"AWS::IAM::Role",
      "Properties":{
        "AssumeRolePolicyDocument":{
          "Version": "2012-10-17",
          "Statement":[
            {
              "Sid":"",
              "Effect":"Allow",
              "Principal":{
                "Service":"ecs-tasks.amazonaws.com"
              },
              "Action":"sts:AssumeRole"
            }
          ]
        },
        "ManagedPolicyArns":[ "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" ]
      }
    },
    "BlueTaskDefinition":{
      "Type":"AWS::ECS::TaskDefinition",
      "Properties":{
        "ExecutionRoleArn":{
          "Fn::GetAtt":[ "ECSTaskExecutionRole","Arn" ]
        },
        "ContainerDefinitions":[
          {
            "Name":"DemoApp",
            "Image":"nginxdemos/hello:latest",
            "Essential":true,
            "PortMappings":[
              {
                "HostPort":80,
                "Protocol":"tcp",
                "ContainerPort":80
              }
            ]
          }
        ],
        "RequiresCompatibilities":[ "FARGATE" ],
        "NetworkMode":"awsvpc",
        "Cpu":"256",
```

```
        "Memory":"512",
        "Family":"ecs-demo"
      }
    },
    "ECSDemoCluster":{
      "Type":"AWS::ECS::Cluster",
      "Properties":{}
    },
    "ECSDemoService":{
      "Type":"AWS::ECS::Service",
      "Properties":{
        "Cluster":{ "Ref":"ECSDemoCluster" },
        "DesiredCount":1,
        "DeploymentController":{ "Type":"EXTERNAL" }
      }
    },
    "BlueTaskSet":{
      "Type":"AWS::ECS::TaskSet",
      "Properties":{
        "Cluster":{ "Ref":"ECSDemoCluster" },
        "LaunchType":"FARGATE",
        "NetworkConfiguration":{
          "AwsVpcConfiguration":{
            "AssignPublicIp":"ENABLED",
            "SecurityGroups":[{ "Ref":"ExampleSecurityGroup" }],
            "Subnets":[{ "Ref":"Subnet1" },{ "Ref":"Subnet2" }]
          }
        },
        "PlatformVersion":"1.4.0",
        "Scale":{
          "Unit":"PERCENT",
          "Value":100
        },
        "Service":{ "Ref":"ECSDemoService"},
        "TaskDefinition":{ "Ref":"BlueTaskDefinition" },
        "LoadBalancers":[
          {
            "ContainerName":"DemoApp",
            "ContainerPort":80,
            "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" }
          }
        ]
      }
    },
```

```
    "PrimaryTaskSet":{
      "Type":"AWS::ECS::PrimaryTaskSet",
      "Properties":{
        "Cluster":{ "Ref":"ECSDemoCluster" },
        "Service":{ "Ref":"ECSDemoService" },
        "TaskSetId":{ "Fn::GetAtt":[ "BlueTaskSet","Id" ]
        }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  Vpc:
    Type: AWS::EC2::VPC::Id
  Subnet1:
    Type: AWS::EC2::Subnet::Id
  Subnet2:
    Type: AWS::EC2::Subnet::Id
Transform:
  - 'AWS::CodeDeployBlueGreen'
Hooks:
  CodeDeployBlueGreenHook:
    Type: AWS::CodeDeploy::BlueGreen
    Properties:
      TrafficRoutingConfig:
        Type: TimeBasedCanary
        TimeBasedCanary:
          StepPercentage: 15
          BakeTimeMins: 5
      Applications:
        - Target:
            Type: AWS::ECS::Service
            LogicalID: ECSDemoService
          ECSAttributes:
            TaskDefinitions:
              - BlueTaskDefinition
              - GreenTaskDefinition
            TaskSets:
              - BlueTaskSet
```

```
              - GreenTaskSet
            TrafficRouting:
              ProdTrafficRoute:
                Type: AWS::ElasticLoadBalancingV2::Listener
                LogicalID: ALBListenerProdTraffic
              TargetGroups:
                - ALBTargetGroupBlue
                - ALBTargetGroupGreen
Resources:
  ExampleSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for ec2 access
      VpcId: !Ref Vpc
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 8080
          ToPort: 8080
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
  ALBTargetGroupBlue:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 5
      HealthCheckPath: /
      HealthCheckPort: '80'
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 2
      HealthyThresholdCount: 2
      Matcher:
        HttpCode: '200'
      Port: 80
      Protocol: HTTP
      Tags:
        - Key: Group
          Value: Example
      TargetType: ip
```

```
      UnhealthyThresholdCount: 4
      VpcId: !Ref Vpc
  ALBTargetGroupGreen:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 5
      HealthCheckPath: /
      HealthCheckPort: '80'
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 2
      HealthyThresholdCount: 2
      Matcher:
        HttpCode: '200'
      Port: 80
      Protocol: HTTP
      Tags:
        - Key: Group
          Value: Example
      TargetType: ip
      UnhealthyThresholdCount: 4
      VpcId: !Ref Vpc
  ExampleALB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      SecurityGroups:
        - !Ref ExampleSecurityGroup
      Subnets:
        - !Ref Subnet1
        - !Ref Subnet2
      Tags:
        - Key: Group
          Value: Example
      Type: application
      IpAddressType: ipv4
  ALBListenerProdTraffic:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          ForwardConfig:
            TargetGroups:
              - TargetGroupArn: !Ref ALBTargetGroupBlue
                Weight: 1
```

```
      LoadBalancerArn: !Ref ExampleALB
      Port: 80
      Protocol: HTTP
  ALBListenerProdRule:
    Type: AWS::ElasticLoadBalancingV2::ListenerRule
    Properties:
      Actions:
        - Type: forward
          ForwardConfig:
            TargetGroups:
              - TargetGroupArn: !Ref ALBTargetGroupBlue
                Weight: 1
      Conditions:
        - Field: http-header
          HttpHeaderConfig:
            HttpHeaderName: User-Agent
            Values:
              - Mozilla
      ListenerArn: !Ref ALBListenerProdTraffic
      Priority: 1
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Sid: ''
            Effect: Allow
            Principal:
              Service: ecs-tasks.amazonaws.com
            Action: 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'
  BlueTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ExecutionRoleArn: !GetAtt
        - ECSTaskExecutionRole
        - Arn
      ContainerDefinitions:
        - Name: DemoApp
          Image: 'nginxdemos/hello:latest'
          Essential: true
          PortMappings:
```

```
            - HostPort: 80
              Protocol: tcp
              ContainerPort: 80
      RequiresCompatibilities:
        - FARGATE
      NetworkMode: awsvpc
      Cpu: '256'
      Memory: '512'
      Family: ecs-demo
  ECSDemoCluster:
    Type: AWS::ECS::Cluster
    Properties: {}
  ECSDemoService:
    Type: AWS::ECS::Service
    Properties:
      Cluster: !Ref ECSDemoCluster
      DesiredCount: 1
      DeploymentController:
        Type: EXTERNAL
  BlueTaskSet:
    Type: AWS::ECS::TaskSet
    Properties:
      Cluster: !Ref ECSDemoCluster
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsVpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - !Ref ExampleSecurityGroup
          Subnets:
            - !Ref Subnet1
            - !Ref Subnet2
      PlatformVersion: 1.4.0
      Scale:
        Unit: PERCENT
        Value: 100
      Service: !Ref ECSDemoService
      TaskDefinition: !Ref BlueTaskDefinition
      LoadBalancers:
        - ContainerName: DemoApp
          ContainerPort: 80
          TargetGroupArn: !Ref ALBTargetGroupBlue
  PrimaryTaskSet:
    Type: AWS::ECS::PrimaryTaskSet
```

```
    Properties:
      Cluster: !Ref ECSDemoCluster
      Service: !Ref ECSDemoService
      TaskSetId: !GetAtt
        - BlueTaskSet
        - Id
```

CloudFormation 模板代码段

本部分提供了很多示例应用场景，您可以用其了解如何声明多种 Amazon CloudFormation 模板部件。您还可以将代码段用作您的自定义模板的起始部分。

> **Note**
>
> 由于 Amazon CloudFormation 模板必须符合 JSON，所以没有对行继续符进行预配置。如果行的长度大于 68 个字符，则本文档中的代码片段可随机换行。

主题

- 通用模板代码段
- 自动扩缩 CloudFormation 模板代码段
- Amazon 账单控制台模板代码段
- Amazon CloudFormation 模板代码段
- Amazon CloudFront 模板代码段
- Amazon CloudWatch 模板代码段
- Amazon CloudWatch Logs 模板代码段
- Amazon DynamoDB 模板代码段
- Amazon EC2 CloudFormation 模板代码段
- Amazon Elastic Container Service 示例模板
- Amazon Elastic File System 示例模板
- Elastic Beanstalk 模板代码段
- Elastic Load Balancing 模板代码段
- Amazon Identity and Access Management 模板代码段
- Amazon Lambda 模板
- Amazon Redshift 模板代码段

- Amazon RDS 模板代码段
- Route 53 模板代码段
- Amazon S3 模板代码段
- Amazon SNS 模板代码段
- Amazon SQS 模板代码段
- Amazon Timestream 模板代码片段

通用模板代码段

以下示例介绍了并非特定于某个 Amazon 服务的不同 CloudFormation 模板功能。

主题

- Base64 编码的 UserData 属性
- 使用 Base64 编码的 UserData 属性，具有 AccessKey 和 SecretKey
- 含一个文字字符串参数的 Parameters 部分
- 含有带正则表达式约束的字符串参数的 Parameters 部分
- 含有数字参数的 Parameters 部分，带有 MinValue 和 MaxValue 约束
- 含有数字参数的 Parameters 部分，带有 AllowedValues 约束
- 含一个文字 CommaDelimitedList 参数的 Parameters 部分
- 包含基于伪参数的参数值的 Parameters 部分
- 带有三个映射的 Mapping 部分
- 基于文字字符串的 Description
- 带有一个文件字符串输出的 Outputs 部分
- 包含一个资源引用和一个伪参数输出的 Outputs 部分
- 包含一个基于函数的输出、一个文字字符串、一个引用和一个伪参数的 Outputs 部分
- 模板格式版本
- Amazon Tags 属性

Base64 编码的 UserData 属性

此示例演示了使用 `Fn::Base64` 和 `Fn::Join` 函数的 `UserData` 属性集合。引用 `MyValue` 和 `MyName` 是必须要在模板的 `Parameters` 部分定义的参数。文字字符串 `Hello World` 只是此示例传输作为 `UserData` 的一部分的另一个值。

JSON

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ ",", [
            { "Ref" : "MyValue" },
            { "Ref" : "MyName" },
            "Hello World" ] ]
    }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     Ref: MyValue
     Ref: MyName
     Hello World
```

使用 Base64 编码的 UserData 属性，具有 AccessKey 和 SecretKey

此示例演示了使用 Fn::Base64 和 Fn::Join 函数的 UserData 属性集合。它包含 AccessKey 和 SecretKey 信息。参考 AccessKey 和 SecretKey 是必须要在模板的 Parameters 部分中定义的参数。

JSON

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ "", [
            "ACCESS_KEY=", { "Ref" : "AccessKey" },
            "SECRET_KEY=", { "Ref" : "SecretKey" } ]
        ]
    }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     ACCESS_KEY=${AccessKey}
```

```
     SECRET_KEY=${SecretKey}
```

含一个文字字符串参数的 Parameters 部分

以下示例描述了有效的 Parameters 部分声明，其中声明有单个 `String` 类型参数。

JSON

```
"Parameters" : {
    "UserName" : {
        "Type" : "String",
        "Default" : "nonadmin",
        "Description" : "Assume a vanilla user if no command-line spec provided"
    }
}
```

YAML

```
Parameters:
  UserName:
    Type: String
    Default: nonadmin
    Description: Assume a vanilla user if no command-line spec provided
```

含有带正则表达式约束的字符串参数的 Parameters 部分

以下示例描述了有效的 Parameters 部分声明，其中声明有单个 `String` 类型参数。`AdminUserAccount` 参数的默认值为 `admin`。参数值的最小长度必须为 1，最大长度必须为 16，且其中包含字母字符和数字，但必须以字母字符开头。

JSON

```
"Parameters" : {
    "AdminUserAccount": {
      "Default": "admin",
      "NoEcho": "true",
      "Description" : "The admin account user name",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
```

```
    }
}
```

YAML

```
Parameters:
  AdminUserAccount:
    Default: admin
    NoEcho: true
    Description: The admin account user name
    Type: String
    MinLength: 1
    MaxLength: 16
    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
```

含有数字参数的 Parameters 部分，带有 MinValue 和 MaxValue 约束

以下示例描述了有效的 Parameters 部分声明，其中声明有单个 Number 类型参数。WebServerPort 参数的默认值为 80，最小值为 1，最大值为 65535。

JSON

```
"Parameters" : {
    "WebServerPort": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "MinValue": "1",
      "MaxValue": "65535"
    }
}
```

YAML

```
Parameters:
  WebServerPort:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
    MinValue: 1
    MaxValue: 65535
```

含有数字参数的 Parameters 部分，带有 AllowedValues 约束

以下示例描述了有效的 Parameters 部分声明，其中声明有单个 Number 类型参数。WebServerPort 参数的默认值为 80，且只允许值 80 和 8888。

JSON

```
"Parameters" : {
    "WebServerPortLimited": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "AllowedValues" : ["80", "8888"]
    }
}
```

YAML

```
Parameters:
  WebServerPortLimited:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
    AllowedValues:
    - 80
    - 8888
```

含一个文字 CommaDelimitedList 参数的 Parameters 部分

以下示例描述了有效的 Parameters 部分声明，其中声明了单个 CommaDelimitedList 类型参数。NoEcho 属性设置为 TRUE，该属性将使用 describe-stacks 输出中的星号（*****）掩盖其值，但存储在下面指定位置的信息除外。

> **Important**
>
> 使用 NoEcho 属性不会遮蔽在以下各区段中存储的任何信息：
>
> - Metadata 模板区段。CloudFormation 不会转换、修改或编辑您在 Metadata 部分中包含的任何信息。有关更多信息，请参阅 Metadata。
> - Outputs 模板区段。有关更多信息，请参阅 Outputs。

- 资源定义的 Metadata 属性。有关更多信息，请参阅 Metadata 属性。

强烈建议您不要使用这些机制来包含敏感信息，例如密码。

> **Important**
>
> 我们建议不要将敏感信息直接嵌入 CloudFormation 模板中，而应使用堆栈模板中的动态参数来引用在 CloudFormation 外部存储和管理的敏感信息，例如 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager 中的敏感信息。
> 有关更多信息，请参阅请勿将凭证嵌入您的模板最佳实践。

JSON

```
"Parameters" : {
    "UserRoles" : {
        "Type" : "CommaDelimitedList",
        "Default" : "guest,newhire",
        "NoEcho" : "TRUE"
    }
}
```

YAML

```
Parameters:
  UserRoles:
    Type: CommaDelimitedList
    Default: "guest,newhire"
    NoEcho: true
```

包含基于伪参数的参数值的 Parameters 部分

以下示例说明 EC2 用户数据中使用伪参数 AWS::StackName 和 AWS::Region 的命令。有关伪参数的更多信息，请参阅使用伪参数获取 Amazon 值。

JSON

```
          "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
```

```
            "#!/bin/bash -xe\n",
            "yum install -y aws-cfn-bootstrap\n",

            "/opt/aws/bin/cfn-init -v ",
            "         --stack ", { "Ref" : "AWS::StackName" },
            "         --resource LaunchConfig ",
            "         --region ", { "Ref" : "AWS::Region" }, "\n",

            "/opt/aws/bin/cfn-signal -e $? ",
            "         --stack ", { "Ref" : "AWS::StackName" },
            "         --resource WebServerGroup ",
            "         --region ", { "Ref" : "AWS::Region" }, "\n"
        ]]}}
      }
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     #!/bin/bash -xe
     yum update -y aws-cfn-bootstrap
     /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig --
region ${AWS::Region}
     /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource WebServerGroup
 --region ${AWS::Region}
```

带有三个映射的 Mapping 部分

以下示例描述了其中包含三个映射的有效 Mapping 部分声明。映射在与 Stop、SlowDown 或 Go 映射密钥匹配时提供分配至相应 RGBColor 属性的 RGB 值。

JSON

```
"Mappings" : {
    "LightColor" : {
        "Stop" : {
            "Description" : "red",
            "RGBColor" : "RED 255 GREEN 0 BLUE 0"
        },
        "SlowDown" : {
            "Description" : "yellow",
            "RGBColor" : "RED 255 GREEN 255 BLUE 0"
        },
```

```
        "Go" : {
            "Description" : "green",
            "RGBColor" : "RED 0 GREEN 128 BLUE 0"
        }
    }
}
```

YAML

```
Mappings:
  LightColor:
    Stop:
      Description: red
      RGBColor: "RED 255 GREEN 0 BLUE 0"
    SlowDown:
      Description: yellow
      RGBColor: "RED 255 GREEN 255 BLUE 0"
    Go:
      Description: green
      RGBColor: "RED 0 GREEN 128 BLUE 0"
```

基于文字字符串的 Description

以下示例描述了有效的 Description 部分声明，其中的值基于文字字符串。此代码段可用于模板、参数、资源、属性或输出。

JSON

```
"Description" : "Replace this value"
```

YAML

```
Description: "Replace this value"
```

带有一个文件字符串输出的 Outputs 部分

此示例显示的是基于文字字符串的输出分配。

JSON

```
"Outputs" : {
```

```
    "MyPhone" : {
        "Value" : "Please call 555-5555",
        "Description" : "A random message for aws cloudformation describe-stacks"
    }
}
```

YAML

```
Outputs:
  MyPhone:
    Value: Please call 555-5555
    Description: A random message for aws cloudformation describe-stacks
```

包含一个资源引用和一个伪参数输出的 Outputs 部分

此示例显示的是含有两个输出分配的 Outputs 部分。一个基于资源，另一个基于伪引用。

JSON

```
"Outputs" : {
   "SNSTopic" : { "Value" : { "Ref" : "MyNotificationTopic" } },
   "StackName" : { "Value" : { "Ref" : "AWS::StackName" } }
}
```

YAML

```
Outputs:
  SNSTopic:
    Value: !Ref MyNotificationTopic
  StackName:
    Value: !Ref AWS::StackName
```

包含一个基于函数的输出、一个文字字符串、一个引用和一个伪参数的 Outputs 部分

此示例显示的是带有一个输出分配的输出部分。Join 函数用于连接值，将百分比符号用作分隔符。

JSON

```
"Outputs" : {
    "MyOutput" : {
```

```
        "Value" : { "Fn::Join" :
            [ "%", [ "A-string", {"Ref" : "AWS::StackName" } ] ]
        }
    }
}
```

YAML

```
Outputs:
  MyOutput:
    Value: !Join [ %, [ 'A-string', !Ref 'AWS::StackName' ]]
```

模板格式版本

以下代码段描述了有效的 AWSTemplateFormatVersion 部分声明。

JSON

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
```

Amazon Tags 属性

本示例将展示 Amazon Tags 属性。您将在资源的 Properties 部分中执行该属性。资源被创建后，会标上您声明的标签。

JSON

```
"Tags" : [
      {
        "Key" : "keyname1",
        "Value" : "value1"
      },
      {
        "Key" : "keyname2",
        "Value" : "value2"
      }
```

```
    ]
```

YAML

```
Tags:
  -
    Key: "keyname1"
    Value: "value1"
  -
    Key: "keyname2"
    Value: "value2"
```

自动扩缩 CloudFormation 模板代码段

凭借 Amazon EC2 Auto Scaling，您可以通过扩缩策略或计划扩缩自动扩缩 Amazon EC2 实例。自动扩缩组是 Amazon EC2 实例的集合，这些实例支持自动扩缩和实例集管理功能，例如例如扩缩策略、计划操作、运行状况检查、生命周期挂钩和负载平衡。

Application Auto Scaling 通过扩展策略或计划扩展，可以自动扩展 Amazon EC2 以外的其他资源。

您可以使用 Amazon CloudFormation 模板创建和配置自动扩缩组、扩缩策略、计划操作以及其他自动扩缩资源，将其作为基础架构的一部分。模板可以让您轻松地以可重复且一致的方式管理和自动部署自动扩缩资源。

以下示例模板代码段描述了 Amazon EC2 Auto Scaling 和 Application Auto Scaling 的 Amazon CloudFormation 资源或组件。这些代码段旨在集成到模板中，不打算独立运行。

代码段类别

- 使用 Amazon CloudFormation 配置 Amazon EC2 Auto Scaling 资源
- 使用 Amazon CloudFormation 配置 Application Auto Scaling 资源

使用 Amazon CloudFormation 配置 Amazon EC2 Auto Scaling 资源

以下示例演示了要包含在模板中的不同代码段，以便与 Amazon EC2 Auto Scaling 一起使用。

代码段类别

- 创建单个实例自动扩缩组
- 创建具有附加负载均衡器的自动扩缩组

- 创建具有通知的自动扩缩组
- 创建使用 CreationPolicy 和 UpdatePolicy 的自动扩缩组
- 创建分步扩缩策略
- 混合实例组示例
- 启动配置示例

创建单个实例自动扩缩组

此示例显示具有单个实例的 AWS::AutoScaling::AutoScalingGroup 资源以帮助您开始使用。自动扩缩组的 VPCZoneIdentifier 属性指定三个不同可用区中的一系列现有子网。在创建堆栈之前，必须先从您的账户指定适用的子网 ID。LaunchTemplate 属性引用具有在模板其他位置中定义的逻辑名称 myLaunchTemplate 的 AWS::EC2::LaunchTemplate 资源。

> **Note**
>
> 有关启动模板的示例，请参阅 Amazon EC2 代码段部分中的 使用 CloudFormation 创建启动模板 和 AWS::EC2::LaunchTemplate 资源中的示例部分。

JSON

```
"myASG" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
      "LaunchTemplate" : {
        "LaunchTemplateId" : {
          "Ref" : "myLaunchTemplate"
        },
        "Version" : {
          "Fn::GetAtt" : [
            "myLaunchTemplate",
            "LatestVersionNumber"
          ]
        }
      },
      "MaxSize" : "1",
      "MinSize" : "1"
   }
```

```
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MaxSize: '1'
    MinSize: '1'
```

创建具有附加负载均衡器的自动扩缩组

此示例显示了用于在多个服务器之间实现负载均衡的 AWS::AutoScaling::AutoScalingGroup 资源。它指定在同一模板中的其他位置声明的 Amazon 资源的逻辑名称。

1. VPCZoneIdentifier 属性指定将在其中创建自动扩缩组的 EC2 实例的两个 AWS::EC2::Subnet 资源的逻辑名称：myPublicSubnet1 和 myPublicSubnet2。
2. LaunchTemplate 属性指定具有逻辑名称 myLaunchTemplate 的 AWS::EC2::LaunchTemplate 资源。
3. TargetGroupARNs 属性列出用于将流量路由到自动扩缩组的应用程序负载均衡器或网络负载均衡器的目标组。在此示例中，指定了一个目标组，即具有逻辑名称 myTargetGroup 的 AWS::ElasticLoadBalancingV2::TargetGroup 资源。

JSON

```
"myServerGroup" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "VPCZoneIdentifier" : [ { "Ref" : "myPublicSubnet1" }, { "Ref" :
 "myPublicSubnet2" } ],
      "LaunchTemplate" : {
        "LaunchTemplateId" : {
```

```
          "Ref" : "myLaunchTemplate"
        },
        "Version" : {
          "Fn::GetAtt" : [
            "myLaunchTemplate",
            "LatestVersionNumber"
          ]
        }
      },
      "MaxSize" : "5",
      "MinSize" : "1",
      "TargetGroupARNs" : [ { "Ref" : "myTargetGroup" } ]
    }
}
```

YAML

```
myServerGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref myPublicSubnet1
      - !Ref myPublicSubnet2
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MaxSize: '5'
    MinSize: '1'
    TargetGroupARNs:
      - !Ref myTargetGroup
```

另请参阅

有关根据 ALBRequestCountPerTarget 预定义指标为应用程序负载均衡器创建具有目标跟踪扩展策略的自动扩缩组的详细示例，请参阅 AWS::AutoScaling::ScalingPolicy 资源中的示例部分。

创建具有通知的自动扩缩组

此示例显示的是在指定事件发生时发送 Amazon SNS 通知的 AWS::AutoScaling::AutoScalingGroup 资源。NotificationConfigurations 属性指定 Amazon CloudFormation 在其中发送通知的 SNS 主题和将导致 Amazon CloudFormation 发送通知的事件。当 NotificationTypes 指定的事件发生时，Amazon CloudFormation 会发送一条通知至

TopicARN 指定的 SNS 主题中。启动堆栈时，Amazon CloudFormation 会创建在同一模板中声明的 AWS::SNS::Subscription 资源（snsTopicForAutoScalingGroup）。

自动扩缩组的 VPCZoneIdentifier 属性指定三个不同可用区中的一系列现有子网。在创建堆栈之前，必须先从您的账户指定适用的子网 ID。LaunchTemplate 属性引用在同一模板中的其他位置声明的 AWS::EC2::LaunchTemplate 资源的逻辑名称。

JSON

```
"myASG" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "DependsOn": [
    "snsTopicForAutoScalingGroup"
  ],
  "Properties" : {
    "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
    "LaunchTemplate" : {
      "LaunchTemplateId" : {
        "Ref" : "logicalName"
      },
      "Version" : {
        "Fn::GetAtt" : [
          "logicalName",
          "LatestVersionNumber"
        ]
      }
    },
    "MaxSize" : "5",
    "MinSize" : "1",
    "NotificationConfigurations" : [
      {
        "TopicARN" : { "Ref" : "snsTopicForAutoScalingGroup" },
        "NotificationTypes" : [
          "autoscaling:EC2_INSTANCE_LAUNCH",
          "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
          "autoscaling:EC2_INSTANCE_TERMINATE",
          "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
          "autoscaling:TEST_NOTIFICATION"
        ]
      }
    ]
  }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  DependsOn:
    - snsTopicForAutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    LaunchTemplate:
      LaunchTemplateId: !Ref logicalName
      Version: !GetAtt logicalName.LatestVersionNumber
    MaxSize: '5'
    MinSize: '1'
    NotificationConfigurations:
      - TopicARN: !Ref snsTopicForAutoScalingGroup
        NotificationTypes:
          - autoscaling:EC2_INSTANCE_LAUNCH
          - autoscaling:EC2_INSTANCE_LAUNCH_ERROR
          - autoscaling:EC2_INSTANCE_TERMINATE
          - autoscaling:EC2_INSTANCE_TERMINATE_ERROR
          - autoscaling:TEST_NOTIFICATION
```

创建使用 **CreationPolicy** 和 **UpdatePolicy** 的自动扩缩组

以下示例演示如何将 CreationPolicy 和 UpdatePolicy 属性添加到 AWS::AutoScaling::AutoScalingGroup 资源。

示例创建策略会阻止自动扩缩组达到 CREATE_COMPLETE 状态，直到 Amazon CloudFormation 在该组准备就绪后收到 Count 个成功信号。要发出自动扩缩组已就绪的信号，在实例上运行添加到启动模板用户数据（未显示）的 cfn-signal 帮助程序脚本。如果实例未在规定的 Timeout 内发送信号，则 CloudFormation 假设未创建实例、资源创建失败和 CloudFormation 回滚堆栈。

示例更新策略指示 CloudFormation 使用 AutoScalingRollingUpdate 属性执行滚动更新。滚动更新根据 MaxBatchSize 和基于 PauseTime 的更新批次之间的暂停时间以小批量的方式更改自动扩缩组（在本示例中，为逐个实例更改）。MinInstancesInService 属性指定当 CloudFormation 更新旧实例时，自动扩缩组内必须处于服务状态的实例的最小数量。

WaitOnResourceSignals 属性设置为 true。CloudFormation 必须在指定的 PauseTime 内收到每个新实例的信号，才能继续更新。在执行堆栈更新时，以下 EC2 Auto Scaling 进程

会暂停：HealthCheck、ReplaceUnhealthy、AZRebalance、AlarmNotification 和 ScheduledActions。注意：不要暂停 Launch、Terminate 或 AddToLoadBalancer（如果自动扩缩组正在与 Elastic Load Balancing 一起使用）进程类型，因为这样做会使滚动更新无法正常工作。

自动扩缩组的 VPCZoneIdentifier 属性指定三个不同可用区中的一系列现有子网。在创建堆栈之前，必须先从您的账户指定适用的子网 ID。LaunchTemplate 属性引用在同一模板中的其他位置声明的 AWS::EC2::LaunchTemplate 资源的逻辑名称。

有关 CreationPolicy 和 UpdatePolicy 属性的更多信息，请参阅资源属性引用。

JSON

```
{
  "Resources":{
    "myASG":{
      "CreationPolicy":{
        "ResourceSignal":{
          "Count":"3",
          "Timeout":"PT15M"
        }
      },
      "UpdatePolicy":{
        "AutoScalingRollingUpdate":{
          "MinInstancesInService":"3",
          "MaxBatchSize":"1",
          "PauseTime":"PT12M5S",
          "WaitOnResourceSignals":"true",
          "SuspendProcesses":[
            "HealthCheck",
            "ReplaceUnhealthy",
            "AZRebalance",
            "AlarmNotification",
            "ScheduledActions",
            "InstanceRefresh"
          ]
        }
      },
      "Type":"AWS::AutoScaling::AutoScalingGroup",
      "Properties":{
        "VPCZoneIdentifier":[ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
        "LaunchTemplate":{
          "LaunchTemplateId":{
            "Ref":"logicalName"
```

```
          },
          "Version":{
            "Fn::GetAtt":[
              "logicalName",
              "LatestVersionNumber"
            ]
          }
        },
        "MaxSize":"5",
        "MinSize":"3"
      }
    }
  }
}
```

YAML

```
---
Resources:
  myASG:
    CreationPolicy:
      ResourceSignal:
        Count: '3'
        Timeout: PT15M
    UpdatePolicy:
      AutoScalingRollingUpdate:
        MinInstancesInService: '3'
        MaxBatchSize: '1'
        PauseTime: PT12M5S
        WaitOnResourceSignals: true
        SuspendProcesses:
          - HealthCheck
          - ReplaceUnhealthy
          - AZRebalance
          - AlarmNotification
          - ScheduledActions
          - InstanceRefresh
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
        - subnetIdAz1
        - subnetIdAz2
        - subnetIdAz3
```

```
      LaunchTemplate:
        LaunchTemplateId: !Ref logicalName
        Version: !GetAtt logicalName.LatestVersionNumber
      MaxSize: '5'
      MinSize: '3'
```

创建分步扩缩策略

此示例显示了使用分步扩缩策略横向扩展自动扩缩组的 AWS::AutoScaling::ScalingPolicy 资源。AdjustmentType 属性指定 ChangeInCapacity，这表示 ScalingAdjustment 代表待添加（如果 ScalingAdjustment 为正值）或待删除（如果它为负值）实例的数量。在此示例中，ScalingAdjustment 为 1；因此，当超出警报阈值时，策略会将组中 EC2 实例的数量增加 1。

AWS::CloudWatch::Alarm 资源 CPUAlarmHigh 指定扩缩策略 ASGScalingPolicyHigh 作为警报处于 ALARM 状态时要运行的操作（AlarmActions）。Dimensions 属性引用在同一模板中的其他位置声明的 AWS::AutoScaling::AutoScalingGroup 资源的逻辑名称。

JSON

```
{
  "Resources":{
    "ASGScalingPolicyHigh":{
      "Type":"AWS::AutoScaling::ScalingPolicy",
      "Properties":{
        "AutoScalingGroupName":{ "Ref":"logicalName" },
        "PolicyType":"StepScaling",
        "AdjustmentType":"ChangeInCapacity",
        "StepAdjustments":[
          {
            "MetricIntervalLowerBound":0,
            "ScalingAdjustment":1
          }
        ]
      }
    },
    "CPUAlarmHigh":{
      "Type":"AWS::CloudWatch::Alarm",
      "Properties":{
        "EvaluationPeriods":"2",
        "Statistic":"Average",
        "Threshold":"90",
        "AlarmDescription":"Scale out if CPU > 90% for 2 minutes",
```

```
        "Period":"60",
        "AlarmActions":[ { "Ref":"ASGScalingPolicyHigh" } ],
        "Namespace":"AWS/EC2",
        "Dimensions":[
          {
            "Name":"AutoScalingGroupName",
            "Value":{ "Ref":"logicalName" }
          }
        ],
        "ComparisonOperator":"GreaterThanThreshold",
        "MetricName":"CPUUtilization"
      }
    }
  }
}
```

YAML

```
---
Resources:
  ASGScalingPolicyHigh:
    Type: AWS::AutoScaling::ScalingPolicy
    Properties:
      AutoScalingGroupName: !Ref logicalName
      PolicyType: StepScaling
      AdjustmentType: ChangeInCapacity
      StepAdjustments:
        - MetricIntervalLowerBound: 0
          ScalingAdjustment: 1
  CPUAlarmHigh:
    Type: AWS::CloudWatch::Alarm
    Properties:
      EvaluationPeriods: 2
      Statistic: Average
      Threshold: 90
      AlarmDescription: 'Scale out if CPU > 90% for 2 minutes'
      Period: 60
      AlarmActions:
        - !Ref ASGScalingPolicyHigh
      Namespace: AWS/EC2
      Dimensions:
        - Name: AutoScalingGroupName
          Value:
```

```
            !Ref logicalName
      ComparisonOperator: GreaterThanThreshold
      MetricName: CPUUtilization
```

另请参阅

有关扩展策略的更多示例模板，请参阅 AWS::AutoScaling::ScalingPolicy 资源中的示例部分。

混合实例组示例

使用基于属性的实例类型选择创建自动扩缩组

此示例显示了 AWS::AutoScaling::AutoScalingGroup 资源，其中包含使用基于属性的实例类型选择启动混合实例组的信息。您指定了 VCpuCount 属性的最小值和最大值，以及 MemoryMiB 属性的最小值。自动扩缩组使用的所有实例类型都必须与您所需实例的属性匹配。

自动扩缩组的 VPCZoneIdentifier 属性指定三个不同可用区中的一系列现有子网。在创建堆栈之前，必须先从您的账户指定适用的子网 ID。LaunchTemplate 属性引用在同一模板中的其他位置声明的 AWS::EC2::LaunchTemplate 资源的逻辑名称。

JSON

```
{
  "Resources":{
    "myASG":{
      "Type":"AWS::AutoScaling::AutoScalingGroup",
      "Properties":{
        "VPCZoneIdentifier":[
          "subnetIdAz1",
          "subnetIdAz2",
          "subnetIdAz3"
        ],
        "MixedInstancesPolicy":{
          "LaunchTemplate":{
            "LaunchTemplateSpecification":{
              "LaunchTemplateId":{
                "Ref":"logicalName"
              },
              "Version":{
                "Fn::GetAtt":[
                  "logicalName",
                  "LatestVersionNumber"
```

```
              ]
            }
          },
          "Overrides":[
            {
              "InstanceRequirements":{
                "VCpuCount":{
                  "Min":2,
                  "Max":4
                },
                "MemoryMiB":{
                  "Min":2048
                }
              }
            }
          ]
        }
      },
      "MaxSize":"5",
      "MinSize":"1"
    }
  }
}
```

YAML

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
        - subnetIdAz1
        - subnetIdAz2
        - subnetIdAz3
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref logicalName
            Version: !GetAtt logicalName.LatestVersionNumber
          Overrides:
            - InstanceRequirements:
```

```
                VCpuCount:
                  Min: 2
                  Max: 4
                MemoryMiB:
                  Min: 2048
      MaxSize: '5'
      MinSize: '1'
```

启动配置示例

创建启动配置

此示例显示了自动扩缩组的 AWS::AutoScaling::LaunchConfiguration 资源，您可以在其中指定 ImageId、InstanceType 和 SecurityGroups 属性的值。SecurityGroups 属性指定在模板中其他位置指定的 AWS::EC2::SecurityGroup 资源的逻辑名称，以及名为 myExistingEC2SecurityGroup 的现有 EC2 安全组。

JSON

```
"mySimpleConfig" : {
   "Type" : "AWS::AutoScaling::LaunchConfiguration",
   "Properties" : {
      "ImageId" : "ami-02354e95b3example",
      "InstanceType" : "t3.micro",
      "SecurityGroups" : [ { "Ref" : "logicalName" }, "myExistingEC2SecurityGroup" ]
   }
}
```

YAML

```
mySimpleConfig:
  Type: AWS::AutoScaling::LaunchConfiguration
  Properties:
    ImageId: ami-02354e95b3example
    InstanceType: t3.micro
    SecurityGroups:
      - !Ref logicalName
      - myExistingEC2SecurityGroup
```

使用启动配置创建自动扩缩组

此示例显示具有单个实例的 AWS::AutoScaling::AutoScalingGroup 资源。自动扩缩组的 VPCZoneIdentifier 属性指定三个不同可用区中的一系列现有子网。在创建堆栈之前，必须先从您的账户指定适用的子网 ID。LaunchConfigurationName 属性引用具有在模板中定义的逻辑名称 mySimpleConfig 的 AWS::AutoScaling::LaunchConfiguration 资源。

JSON

```
"myASG" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
      "LaunchConfigurationName" : { "Ref" : "mySimpleConfig" },
      "MaxSize" : "1",
      "MinSize" : "1"
   }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    LaunchConfigurationName: !Ref mySimpleConfig
    MaxSize: '1'
    MinSize: '1'
```

使用 Amazon CloudFormation 配置 Application Auto Scaling 资源

本节为不同 Amazon 资源的 Application Auto Scaling 扩展策略和计划操作提供了 Amazon CloudFormation 模板示例。

> **Important**
>
> 当模板中包含 Application Auto Scaling 代码段时，您可能需要使用 DependsOn 属性声明对通过模板创建的特定可扩展资源的依赖关系。这将覆盖默认并行度，并指示 Amazon

CloudFormation 按指定的顺序对资源进行操作。否则，可能会在完全设置资源之前应用扩展配置。

有关更多信息，请参阅 DependsOn 属性。

代码段类别

- 创建 AppStream 实例集的扩缩策略
- 为 Aurora 数据库集群创建扩缩策略
- 创建 DynamoDB 表的扩缩策略
- 创建 Amazon ECS 服务的扩缩策略（指标：平均 CPU 和内存）
- 创建 Amazon ECS 服务的扩缩策略（指标：每个目标的平均请求数）
- 使用 Lambda 函数的 cron 表达式创建计划操作
- 使用竞价型实例集的 at 表达式创建计划操作

创建 AppStream 实例集的扩缩策略

此代码段说明如何创建策略并将其应用于使用 AWS::ApplicationAutoScaling::ScalingPolicy 资源的 AWS::AppStream::Fleet 资源。AWS::ApplicationAutoScaling::ScalableTarget 资源声明一个应用此策略的可扩展目标。Application Auto Scaling 可以扩展队列实例数，最少为 1 个实例，最多为 20 个实例。该策略将实例集的平均容量利用率保持在 75%，横向扩展和横向缩减冷却时间为 300 秒（5 分钟）。

此示例利用 Fn::Join 和 Rev 内置函数，使用在同一模板中指定的 AWS::AppStream::Fleet 资源的逻辑名称来构造 ResourceId 属性。有关更多信息，请参阅内置函数参考。

JSON

```
{
  "Resources" : {
    "ScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 20,
        "MinCapacity" : 1,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet" },
        "ServiceNamespace" : "appstream",
```

```
        "ScalableDimension" : "appstream:fleet:DesiredCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "fleet",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        }
      }
    },
    "ScalingPolicyAppStreamFleet" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu75" },
        "PolicyType" : "TargetTrackingScaling",
        "ServiceNamespace" : "appstream",
        "ScalableDimension" : "appstream:fleet:DesiredCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "fleet",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 75,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "AppStreamAverageCapacityUtilization"
          },
          "ScaleInCooldown" : 300,
          "ScaleOutCooldown" : 300
        }
      }
    }
  }
```

```
}
```

YAML

```
---
Resources:
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 20
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet'
      ServiceNamespace: appstream
      ScalableDimension: appstream:fleet:DesiredCapacity
      ResourceId: !Join
        - /
        - - fleet
          - !Ref logicalName
  ScalingPolicyAppStreamFleet:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu75
      PolicyType: TargetTrackingScaling
      ServiceNamespace: appstream
      ScalableDimension: appstream:fleet:DesiredCapacity
      ResourceId: !Join
        - /
        - - fleet
          - !Ref logicalName
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 75
        PredefinedMetricSpecification:
          PredefinedMetricType: AppStreamAverageCapacityUtilization
        ScaleInCooldown: 300
        ScaleOutCooldown: 300
```

为 Aurora 数据库集群创建扩缩策略

在此代码段中，您注册了 AWS::RDS::DBCluster 资源。AWS::ApplicationAutoScaling::ScalableTarget 资源表示应动态扩展数据库集群以具

有 1-8 个 Aurora 副本。您还可以使用 AWS::ApplicationAutoScaling::ScalingPolicy 资源将目标跟踪扩缩策略应用到集群。

在此配置中，RDSReaderAverageCPUUtilization 预定义指标用于根据 Aurora 数据库集群中所有 Aurora 副本上 40% 的平均 CPU 利用率来调整该 Aurora 数据库集群。该配置将缩减冷却时间指定为 10 分钟，并将扩展冷却时间指定为 5 分钟。

此示例利用 Fn::Sub 内置函数，使用在同一模板中指定的 AWS::RDS::DBCluster 资源的逻辑名称来构造 ResourceId 属性。有关更多信息，请参阅内置函数参考。

JSON

```
{
  "Resources" : {
    "ScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 8,
        "MinCapacity" : 1,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/rds.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_RDSCluster" },
        "ServiceNamespace" : "rds",
        "ScalableDimension" : "rds:cluster:ReadReplicaCount",
        "ResourceId" : { "Fn::Sub" : "cluster:${logicalName}" }
      }
    },
    "ScalingPolicyDBCluster" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu40" },
        "PolicyType" : "TargetTrackingScaling",
        "ServiceNamespace" : "rds",
        "ScalableDimension" : "rds:cluster:ReadReplicaCount",
        "ResourceId" : { "Fn::Sub" : "cluster:${logicalName}" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 40,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "RDSReaderAverageCPUUtilization"
          },
          "ScaleInCooldown" : 600,
          "ScaleOutCooldown" : 300
        }
```

```
      }
    }
  }
}
```

YAML

```
---
Resources:
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 8
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/rds.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster'
      ServiceNamespace: rds
      ScalableDimension: rds:cluster:ReadReplicaCount
      ResourceId: !Sub cluster:${logicalName}
  ScalingPolicyDBCluster:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu40
      PolicyType: TargetTrackingScaling
      ServiceNamespace: rds
      ScalableDimension: rds:cluster:ReadReplicaCount
      ResourceId: !Sub cluster:${logicalName}
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 40
        PredefinedMetricSpecification:
          PredefinedMetricType: RDSReaderAverageCPUUtilization
        ScaleInCooldown: 600
        ScaleOutCooldown: 300
```

创建 DynamoDB 表的扩缩策略

此代码段说明如何使用 TargetTrackingScaling 策略类型创建策略并将其应用于使用 AWS::ApplicationAutoScaling::ScalingPolicy 资源的 AWS::DynamoDB::Table 资源。AWS::ApplicationAutoScaling::ScalableTarget 资源声明一个应用此策略的可扩展目标，其最小写入容量单位为 5，最大写入容量单位为 15。扩展策略会扩展表的写入容量吞吐量，以 DynamoDBWriteCapacityUtilization 预定义指标将目标利用率维持在 50%。

此示例利用 Fn::Join 和 Ref 内置函数，使用在同一模板中指定的 AWS::DynamoDB::Table 资源的逻辑名称来构造 ResourceId 属性。有关更多信息，请参阅内置函数参考。

> **Note**
>
> 有关如何为 DynamoDB 资源创建 Amazon CloudFormation 模板的更多信息，请参阅 Amazon 数据库博客上的博客文章如何使用 Amazon CloudFormation 配置 Amazon DynamoDB 表和索引的自动伸缩。

JSON

```
{
  "Resources" : {
    "WriteCapacityScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 15,
        "MinCapacity" : 5,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable" },
        "ServiceNamespace" : "dynamodb",
        "ScalableDimension" : "dynamodb:table:WriteCapacityUnits",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "table",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        }
      }
    },
    "WriteScalingPolicy" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : "WriteScalingPolicy",
        "PolicyType" : "TargetTrackingScaling",
```

```
        "ScalingTargetId" : { "Ref" : "WriteCapacityScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 50.0,
          "ScaleInCooldown" : 60,
          "ScaleOutCooldown" : 60,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "DynamoDBWriteCapacityUtilization"
          }
        }
      }
    }
  }
}
```

YAML

```
---
Resources:
  WriteCapacityScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 15
      MinCapacity: 5
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable'
      ServiceNamespace: dynamodb
      ScalableDimension: dynamodb:table:WriteCapacityUnits
      ResourceId: !Join
        - /
        - - table
          - !Ref logicalName
  WriteScalingPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: WriteScalingPolicy
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref WriteCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 50.0
        ScaleInCooldown: 60
        ScaleOutCooldown: 60
```

```
        PredefinedMetricSpecification:
          PredefinedMetricType: DynamoDBWriteCapacityUtilization
```

创建 Amazon ECS 服务的扩缩策略（指标：平均 CPU 和内存）

此代码段说明如何创建策略并将其应用于使用 AWS::ApplicationAutoScaling::ScalingPolicy 资源的 AWS::ECS::Service 资源。AWS::ApplicationAutoScaling::ScalableTarget 资源声明一个应用此策略的可扩展目标。Application Auto Scaling 可扩展任务的数量，最少 1 个任务，最多 6 个任务。

此示例创建两个 TargetTrackingScaling 策略类型的扩展策略。这些策略用于根据服务的平均 CPU 和内存使用率扩展 ECS 服务。此示例利用 Fn::Join 和 Ref 内置函数，使用在同一模板中指定的 AWS::ECS::Cluster（myContainerCluster）和 AWS::ECS::Service（myService）资源的逻辑名称来构造 ResourceId 属性。有关更多信息，请参阅内置函数参考。

JSON

```
{
  "Resources" : {
    "ECSScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : "6",
        "MinCapacity" : "1",
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService" },
        "ServiceNamespace" : "ecs",
        "ScalableDimension" : "ecs:service:DesiredCount",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "service",
              {
                "Ref" : "myContainerCluster"
              },
              {
                "Fn::GetAtt" : [
                  "myService",
                  "Name"
                ]
              }
```

```
            ]
          ]
        }
      }
    },
    "ServiceScalingPolicyCPU" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu70" },
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 70.0,
          "ScaleInCooldown" : 180,
          "ScaleOutCooldown" : 60,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "ECSServiceAverageCPUUtilization"
          }
        }
      }
    },
    "ServiceScalingPolicyMem" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-mem90" },
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 90.0,
          "ScaleInCooldown" : 180,
          "ScaleOutCooldown" : 60,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "ECSServiceAverageMemoryUtilization"
          }
        }
      }
    }
  }
}
```

YAML

```
---
```

```
Resources:
  ECSScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 6
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ecs.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService'
      ServiceNamespace: ecs
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId: !Join
        - /
        - - service
          - !Ref myContainerCluster
          - !GetAtt myService.Name
  ServiceScalingPolicyCPU:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu70
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 70.0
        ScaleInCooldown: 180
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: ECSServiceAverageCPUUtilization
  ServiceScalingPolicyMem:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-mem90
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 90.0
        ScaleInCooldown: 180
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: ECSServiceAverageMemoryUtilization
```

创建 Amazon ECS 服务的扩缩策略（指标：每个目标的平均请求数）

以下示例将具有 ALBRequestCountPerTarget 预定义指标的目标跟踪扩展策略应用到 ECS 服务。该策略用于在每个目标的请求计数（每分钟）超过目标值时向 ECS 服务添加容量。由于 DisableScaleIn 的值设置为 true，因此，目标跟踪策略不会从可扩展目标中删除容量。

此示例利用 Fn::Join 和 Fn::GetAtt 内置函数，使用在同一模板中指定的 AWS::ElasticLoadBalancingV2::LoadBalancer（myLoadBalancer）和 AWS::ElasticLoadBalancingV2::TargetGroup（myTargetGroup）资源的逻辑名称来构造 ResourceLabel 属性。有关更多信息，请参阅内置函数参考。

可扩展目标的 MaxCapacity 和 MinCapacity 属性以及扩展策略的 TargetValue 属性引用您在创建或更新堆栈时传递给模板的参数值。

JSON

```
{
  "Resources" : {
    "ECSScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : { "Ref" : "MaxCount" },
        "MinCapacity" : { "Ref" : "MinCount" },
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService" },
        "ServiceNamespace" : "ecs",
        "ScalableDimension" : "ecs:service:DesiredCount",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "service",
              {
                "Ref" : "myContainerCluster"
              },
              {
                "Fn::GetAtt" : [
                  "myService",
                  "Name"
                ]
              }
            ]
```

```
          ]
        }
      }
    },
    "ServiceScalingPolicyALB" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : "alb-requests-per-target-per-minute",
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : { "Ref" : "ALBPolicyTargetValue" },
          "ScaleInCooldown" : 180,
          "ScaleOutCooldown" : 30,
          "DisableScaleIn" : true,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "ALBRequestCountPerTarget",
            "ResourceLabel" : {
              "Fn::Join" : [
                "/",
                [
                  {
                    "Fn::GetAtt" : [
                      "myLoadBalancer",
                      "LoadBalancerFullName"
                    ]
                  },
                  {
                    "Fn::GetAtt" : [
                      "myTargetGroup",
                      "TargetGroupFullName"
                    ]
                  }
                ]
              ]
            }
          }
        }
      }
    }
  }
}
```

YAML

```
---
Resources:
  ECSScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: !Ref MaxCount
      MinCapacity: !Ref MinCount
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ecs.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService'
      ServiceNamespace: ecs
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId: !Join
        - /
        - - service
          - !Ref myContainerCluster
          - !GetAtt myService.Name
  ServiceScalingPolicyALB:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: alb-requests-per-target-per-minute
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: !Ref ALBPolicyTargetValue
        ScaleInCooldown: 180
        ScaleOutCooldown: 30
        DisableScaleIn: true
        PredefinedMetricSpecification:
          PredefinedMetricType: ALBRequestCountPerTarget
          ResourceLabel: !Join
            - '/'
            - - !GetAtt myLoadBalancer.LoadBalancerFullName
              - !GetAtt myTargetGroup.TargetGroupFullName
```

使用 Lambda 函数的 cron 表达式创建计划操作

此代码段使用 AWS::ApplicationAutoScaling::ScalableTarget 资源注册名为 BLUE 的函数别名（AWS::Lambda::Alias）的预置并发。它还使用 cron 表达式创建具有重复计划的计划操作。定期计划的时区为 UTC。

它在 RoleARN 属性中使用 Fn::Join 和 Ref 内置函数以指定服务相关角色的 ARN。此示例利用 Fn::Sub 内置函数，使用在同一模板中指定的 AWS::Lambda::Function 或 AWS::Serverless::Function 资源的逻辑名称来构造 ResourceId 属性。有关更多信息，请参阅内置函数参考。

> **Note**
>
> 您无法在指向未发布版本（$LATEST）的别名上分配预置并发。
> 有关如何为 Lambda 资源创建 Amazon CloudFormation 模板的更多信息，请参阅 Amazon 计算博客上的博客文章计划 Amazon Lambda 预置并发性以实现重复使用峰值。

JSON

```
{
  "ScalableTarget" : {
    "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
    "Properties" : {
      "MaxCapacity" : 250,
      "MinCapacity" : 0,
      "RoleARN" : {
        "Fn::Join" : [
          ":",
          [
            "arn:aws:iam:",
            {
              "Ref" : "AWS::AccountId"
            },
            "role/aws-service-role/lambda.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency"
          ]
        ]
      },
      "ServiceNamespace" : "lambda",
      "ScalableDimension" : "lambda:function:ProvisionedConcurrency",
      "ResourceId" : { "Fn::Sub" : "function:${logicalName}:BLUE" },
      "ScheduledActions" : [
        {
          "ScalableTargetAction" : {
            "MinCapacity" : "250"
          },
          "ScheduledActionName" : "my-scale-out-scheduled-action",
```

```
          "Schedule" : "cron(0 18 * * ? *)",
          "EndTime" : "2022-12-31T12:00:00.000Z"
        }
      ]
    }
  }
}
```

YAML

```
ScalableTarget:
  Type: AWS::ApplicationAutoScaling::ScalableTarget
  Properties:
    MaxCapacity: 250
    MinCapacity: 0
    RoleARN: !Join
      - ':'
      - - 'arn:aws:iam:'
        - !Ref 'AWS::AccountId'
        - role/aws-service-role/lambda.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency
    ServiceNamespace: lambda
    ScalableDimension: lambda:function:ProvisionedConcurrency
    ResourceId: !Sub function:${logicalName}:BLUE
    ScheduledActions:
      - ScalableTargetAction:
          MinCapacity: 250
        ScheduledActionName: my-scale-out-scheduled-action
        Schedule: 'cron(0 18 * * ? *)'
        EndTime: '2022-12-31T12:00:00.000Z'
```

使用竞价型实例集的 **at** 表达式创建计划操作

此代码段展示如何使用 AWS::ApplicationAutoScaling::ScalableTarget 资源创建两个仅对 AWS::EC2::SpotFleet 资源发生一次的计划操作。每个一次性计划操作的时区均为 UTC。

此示例利用 Fn::Join 和 Ref 内置函数，使用在同一模板中指定的 AWS::EC2::SpotFleet 资源的逻辑名称来构造 ResourceId 属性。有关更多信息，请参阅内置函数参考。

> **Note**
>
> 竞价型实例集请求必须使用 `maintain` 作为请求类型。一次性请求或 Spot 型限制不支持自动扩展。

JSON

```
{
  "Resources" : {
    "SpotFleetScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 0,
        "MinCapacity" : 0,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ec2.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest" },
        "ServiceNamespace" : "ec2",
        "ScalableDimension" : "ec2:spot-fleet-request:TargetCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "spot-fleet-request",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        },
        "ScheduledActions" : [
          {
            "ScalableTargetAction" : {
              "MaxCapacity" : "10",
              "MinCapacity" : "10"
            },
            "ScheduledActionName" : "my-scale-out-scheduled-action",
            "Schedule" : "at(2022-05-20T13:00:00)"
          },
          {
            "ScalableTargetAction" : {
              "MaxCapacity" : "0",
```

```
              "MinCapacity" : "0"
            },
            "ScheduledActionName" : "my-scale-in-scheduled-action",
            "Schedule" : "at(2022-05-20T21:00:00)"
          }
        ]
      }
    }
  }
}
```

YAML

```
---
Resources:
  SpotFleetScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 0
      MinCapacity: 0
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ec2.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest'
      ServiceNamespace: ec2
      ScalableDimension: 'ec2:spot-fleet-request:TargetCapacity'
      ResourceId: !Join
        - /
        - - spot-fleet-request
          - !Ref logicalName
      ScheduledActions:
        - ScalableTargetAction:
            MaxCapacity: 10
            MinCapacity: 10
          ScheduledActionName: my-scale-out-scheduled-action
          Schedule: 'at(2022-05-20T13:00:00)'
        - ScalableTargetAction:
            MaxCapacity: 0
            MinCapacity: 0
          ScheduledActionName: my-scale-in-scheduled-action
          Schedule: 'at(2022-05-20T21:00:00)'
```

Amazon 账单控制台模板代码段

此示例创建一个包含 10% 全球加价定价规则的定价计划。此定价计划已附加到账单组。账单组还有两个自定义行项目，在账单组总成本的基础上收取 10 美元的费用和 10% 的费用。

JSON

```
{
   "Parameters": {
      "LinkedAccountIds": {
         "Type": "ListNumber"
      },
      "PrimaryAccountId": {
         "Type": "Number"
      }
   },
   "Resources": {
      "TestPricingRule": {
         "Type": "AWS::BillingConductor::PricingRule",
         "Properties": {
            "Name": "TestPricingRule",
            "Description": "Test pricing rule created through Cloudformation. Mark
 everything by 10%.",
            "Type": "MARKUP",
            "Scope": "GLOBAL",
            "ModifierPercentage": 10
         }
      },
      "TestPricingPlan": {
         "Type": "AWS::BillingConductor::PricingPlan",
         "Properties": {
            "Name": "TestPricingPlan",
            "Description": "Test pricing plan created through Cloudformation.",
            "PricingRuleArns": [
               {"Fn::GetAtt": ["TestPricingRule", "Arn"]}
            ]
         }
      },
      "TestBillingGroup": {
         "Type": "AWS::BillingConductor::BillingGroup",
         "Properties": {
            "Name": "TestBillingGroup",
```

```
            "Description": "Test billing group created through Cloudformation with 1
linked account. The linked account is also the primary account.",
            "PrimaryAccountId": {
               "Ref": "PrimaryAccountId"
            },
            "AccountGrouping": {
               "LinkedAccountIds": null
            },
            "ComputationPreference": {
               "PricingPlanArn": {
                 "Fn::GetAtt": ["TestPricingPlan", "Arn"]
               }
            }
         }
      },
      "TestFlatCustomLineItem": {
         "Type": "AWS::BillingConductor::CustomLineItem",
         "Properties": {
            "Name": "TestFlatCustomLineItem",
            "Description": "Test flat custom line item created through Cloudformation
for a $10 charge.",
            "BillingGroupArn": {
              "Fn::GetAtt": ["TestBillingGroup", "Arn"]
            },
            "CustomLineItemChargeDetails": {
               "Flat": {
                  "ChargeValue": 10
               },
               "Type": "FEE"
            }
         }
      },
      "TestPercentageCustomLineItem": {
         "Type": "AWS::BillingConductor::CustomLineItem",
         "Properties": {
            "Name": "TestPercentageCustomLineItem",
            "Description": "Test percentage custom line item created through
Cloudformation for a %10 additional charge on the overall total bill of the billing
group.",
            "BillingGroupArn": {
              "Fn::GetAtt": ["TestBillingGroup", "Arn"]
            },
            "CustomLineItemChargeDetails": {
               "Percentage": {
```

```
                "PercentageValue": 10,
                "ChildAssociatedResources": [
                   {"Fn::GetAtt": ["TestBillingGroup", "Arn"]}
                ]
             },
             "Type": "FEE"
          }
       }
    }
 }
}
```

YAML

```
Parameters:
  LinkedAccountIds:
    Type: ListNumber
  PrimaryAccountId:
    Type: Number
Resources:
  TestPricingRule:
    Type: AWS::BillingConductor::PricingRule
    Properties:
      Name: 'TestPricingRule'
      Description: 'Test pricing rule created through Cloudformation. Mark everything
 by 10%.'
      Type: 'MARKUP'
      Scope: 'GLOBAL'
      ModifierPercentage: 10
  TestPricingPlan:
    Type: AWS::BillingConductor::PricingPlan
    Properties:
      Name: 'TestPricingPlan'
      Description: 'Test pricing plan created through Cloudformation.'
      PricingRuleArns:
        - !GetAtt TestPricingRule.Arn
  TestBillingGroup:
    Type: AWS::BillingConductor::BillingGroup
    Properties:
      Name: 'TestBillingGroup'
      Description: 'Test billing group created through Cloudformation with 1 linked
 account. The linked account is also the primary account.'
      PrimaryAccountId: !Ref PrimaryAccountId
```

```
      AccountGrouping:
        LinkedAccountIds: !Ref LinkedAccountIds
      ComputationPreference:
        PricingPlanArn: !GetAtt TestPricingPlan.Arn
  TestFlatCustomLineItem:
    Type: AWS::BillingConductor::CustomLineItem
    Properties:
      Name: 'TestFlatCustomLineItem'
      Description: 'Test flat custom line item created through Cloudformation for a $10
charge.'
      BillingGroupArn: !GetAtt TestBillingGroup.Arn
      CustomLineItemChargeDetails:
        Flat:
          ChargeValue: 10
        Type: 'FEE'
  TestPercentageCustomLineItem:
    Type: AWS::BillingConductor::CustomLineItem
    Properties:
      Name: 'TestPercentageCustomLineItem'
      Description: 'Test percentage custom line item created through Cloudformation for
a %10 additional charge on the overall total bill of the billing group.'
      BillingGroupArn: !GetAtt TestBillingGroup.Arn
      CustomLineItemChargeDetails:
        Percentage:
          PercentageValue: 10
          ChildAssociatedResources:
            - !GetAtt TestBillingGroup.Arn
        Type: 'FEE'
```

Amazon CloudFormation 模板代码段

主题

- 嵌套堆栈
- 等待条件

嵌套堆栈

在模板中嵌套堆栈

此示例模板包含一个称为 myStack 的嵌套堆栈资源。在 Amazon CloudFormation 通过模板创建堆栈时，它创建 myStack，其模板是在 TemplateURL 属性中指定的。输出值 StackRef

返回 myStack 的堆栈 ID，值 OutputFromNestedStack 从 myStack 资源中返回输出值 BucketName。Outputs.*nestedstackoutputname* 是预留格式，用于指定嵌套堆栈的输出值，可以在包含模板的任意位置中使用该格式。

有关更多信息，请参阅 AWS::CloudFormation::Stack。

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myStack" : {
        "Type" : "AWS::CloudFormation::Stack",
        "Properties" : {
         "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-1/
S3_Bucket.template",
              "TimeoutInMinutes" : "60"
        }
        }
    },
    "Outputs": {
       "StackRef": {"Value": { "Ref" : "myStack"}},
       "OutputFromNestedStack" : {
             "Value" : { "Fn::GetAtt" : [ "myStack", "Outputs.BucketName" ] }
       }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cloudformation-templates-us-east-1/
S3_Bucket.template
      TimeoutInMinutes: '60'
Outputs:
  StackRef:
    Value: !Ref myStack
  OutputFromNestedStack:
```

```
    Value: !GetAtt myStack.Outputs.BucketName
```

使用输入参数在模板中嵌套堆栈

此示例模板包含指定输入参数的堆栈资源。Amazon CloudFormation 通过该模板创建堆栈时，会将 Parameters 属性中声明的值对作为输入参数，供创建 myStackWithParams 堆栈的模板使用。在此示例中，已指定 InstanceType 和 KeyName 参数。

有关更多信息，请参阅 AWS::CloudFormation::Stack。

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myStackWithParams" : {
          "Type" : "AWS::CloudFormation::Stack",
        "Properties" : {
            "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-
east-1/EC2ChooseAMI.template",
            "Parameters" : {
                "InstanceType" : "t2.micro",
                "KeyName" : "mykey"
            }
           }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myStackWithParams:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cloudformation-templates-us-east-1/
EC2ChooseAMI.template
      Parameters:
        InstanceType: t2.micro
        KeyName: mykey
```

等待条件

将等待条件用于 Amazon EC2 实例

> **Important**
>
> 对于 Amazon EC2 和 Auto Scaling 资源，我们建议您使用 CreationPolicy 属性而非等待条件。当实例创建过程成功完成后，将“CreationPolicy”属性添加到这些资源，并使用 cfn-signal 帮助程序脚本发送信号。

如果您无法使用创建策略，请查看以下示例模板，该模板声明了带等待条件的 Amazon EC2 实例。`myWaitCondition` 等待条件使用 `myWaitConditionHandle` 发出信号，使用 `DependsOn` 属性指定等待条件将在创建 Amazon EC2 实例资源后触发，并使用 `Timeout` 属性将等待条件的持续时间指定为 4500 秒。此外，向等待条件发出信号的预签名 URL 将通过 `Ec2Instance` 资源的 `UserData` 属性传输至 Amazon EC2 实例，以使运行在该 Amazon EC2 实例上的应用程序或脚本可以检索预签名 URL，并使用该 URL 向等待条件发出成功或失败信号。您需要使用 `cfn-signal`，或创建向等待条件发出信号的应用程序或脚本。输出值 `ApplicationData` 包含从等待条件信号中传回的数据。

有关更多信息，请参阅 在 CloudFormation 模板中创建等待条件。

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Mappings" : {
        "RegionMap" : {
            "us-east-1" : {
                "AMI" : "ami-0123456789abcdef0"
            },
            "us-west-1" : {
                "AMI" : "ami-0987654321fedcba0"
            },
            "eu-west-1" : {
                "AMI" : "ami-0abcdef123456789a"
            },
            "ap-northeast-1" : {
                "AMI" : "ami-0fedcba987654321b"
            },
            "ap-southeast-1" : {
```

```
                "AMI" : "ami-0c1d2e3f4a5b6c7d8"
            }
        }
    },
    "Resources" : {
        "Ec2Instance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "UserData" : { "Fn::Base64" : {"Ref" : "myWaitHandle"}},
                "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" :
 "AWS::Region" }, "AMI" ]}
            }
        },
        "myWaitHandle" : {
            "Type" : "AWS::CloudFormation::WaitConditionHandle",
            "Properties" : {
            }
        },
        "myWaitCondition" : {
            "Type" : "AWS::CloudFormation::WaitCondition",
            "DependsOn" : "Ec2Instance",
            "Properties" : {
                "Handle" : { "Ref" : "myWaitHandle" },
                "Timeout" : "4500"
            }
        }
    },
    "Outputs" : {
        "ApplicationData" : {
            "Value" : { "Fn::GetAtt" : [ "myWaitCondition", "Data" ]},
            "Description" : "The data passed back as part of signalling the
 WaitCondition."
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Mappings:
  RegionMap:
    us-east-1:
      AMI: ami-0123456789abcdef0
```

```
    us-west-1:
      AMI: ami-0987654321fedcba0
    eu-west-1:
      AMI: ami-0abcdef123456789a
    ap-northeast-1:
      AMI: ami-0fedcba987654321b
    ap-southeast-1:
      AMI: ami-0c1d2e3f4a5b6c7d8
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      UserData:
        Fn::Base64: !Ref myWaitHandle
      ImageId:
        Fn::FindInMap:
        - RegionMap
        - Ref: AWS::Region
        - AMI
  myWaitHandle:
    Type: AWS::CloudFormation::WaitConditionHandle
    Properties: {}
  myWaitCondition:
    Type: AWS::CloudFormation::WaitCondition
    DependsOn: Ec2Instance
    Properties:
      Handle: !Ref myWaitHandle
      Timeout: '4500'
Outputs:
  ApplicationData:
    Value: !GetAtt myWaitCondition.Data
    Description: The data passed back as part of signalling the WaitCondition.
```

使用 cfn-signal 帮助程序脚本向等待条件发出信号

此示例演示将成功信号发送到等待条件的 `cfn-signal` 命令行。您需要在 EC2 实例的 `UserData` 属性中定义命令行。

JSON

```
"UserData": {
  "Fn::Base64": {
    "Fn::Join": [
```

```
      "",
      [
         "#!/bin/bash -xe\n",
         "/opt/aws/bin/cfn-signal --exit-code 0 '",
         {
           "Ref": "myWaitHandle"
         },
         "'\n"
      ]
    ]
  }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
    #!/bin/bash -xe
    /opt/aws/bin/cfn-signal --exit-code 0 '${myWaitHandle}'
```

使用 Curl 向等待条件发出信号

此示例显示将成功信号发送到等待条件的 Curl 命令行。

```
curl -T /tmp/a "https://cloudformation-waitcondition-test.s3.amazonaws.com/
arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack
%2Fstack-gosar-20110427004224-test-stack-with-WaitCondition--VEYW
%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVoo
%3D"
```

其中文件 /tmp/a 包含以下 JSON 结构：

```
{
  "Status" : "SUCCESS",
  "Reason" : "Configuration Complete",
  "UniqueId" : "ID1234",
  "Data" : "Application has completed configuration."
}
```

此示例显示发送同一成功信号的 Curl 命令行，将 JSON 发送为命令行参数的情况除外。

```
curl -X PUT -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS","Reason" :
 "Configuration Complete","UniqueId" : "ID1234","Data" : "Application
 has completed configuration."}' "https://cloudformation-waitcondition-
test.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack
%2Fstack-gosar-20110427004224-test-stack-with-WaitCondition--VEYW
%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVoo
%3D"
```

Amazon CloudFront 模板代码段

在 Amazon CloudFormation 中，可以将这些示例模板代码段用于 Amazon CloudFront 分发资源。有关更多信息，请参阅 Amazon CloudFront resource type reference。

主题

- 具有 Amazon S3 源的 Amazon CloudFront 分布资源
- 带自定义源的 Amazon CloudFront 分配资源
- 具有多源支持的 Amazon CloudFront 分配
- 以 Lambda 函数为源的 Amazon CloudFront 分发
- 另请参阅

具有 Amazon S3 源的 Amazon CloudFront 分布资源

以下示例模板显示使用 S3Origin 的 Amazon CloudFront 分发和旧式来源访问身份（OAI）。有关改用来源访问控制（OAC）的信息，请参阅《Amazon CloudFront 开发人员指南》中的限制对 Amazon Simple Storage Service 源的访问。

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myDistribution" : {
            "Type" : "AWS::CloudFront::Distribution",
            "Properties" : {
                "DistributionConfig" : {
                    "Origins" : [ {
                        "DomainName" : "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "Id" : "myS3Origin",
```

```
                        "S3OriginConfig" : {
                            "OriginAccessIdentity" : "origin-access-identity/
cloudfront/E127EXAMPLE51Z"
                        }
                    }],
                    "Enabled" : "true",
                    "Comment" : "Some comment",
                    "DefaultRootObject" : "index.html",
                    "Logging" : {
                        "IncludeCookies" : "false",
                        "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
                        "Prefix" : "myprefix"
                    },
                    "Aliases" : [ "mysite.example.com", "yoursite.example.com" ],
                    "DefaultCacheBehavior" : {
                        "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS",
 "PATCH", "POST", "PUT" ],
                        "TargetOriginId" : "myS3Origin",
                        "ForwardedValues" : {
                            "QueryString" : "false",
                            "Cookies" : { "Forward" : "none" }
                        },
                        "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
                        "ViewerProtocolPolicy" : "allow-all"
                    },
                   "PriceClass" : "PriceClass_200",
                   "Restrictions" : {
                       "GeoRestriction" : {
                           "RestrictionType" : "whitelist",
                           "Locations" : [ "AQ", "CV" ]
                       }
                   },
                   "ViewerCertificate" : { "CloudFrontDefaultCertificate" : "true" }
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
```

```
myDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Origins:
      - DomainName: amzn-s3-demo-bucket.s3.amazonaws.com
        Id: myS3Origin
        S3OriginConfig:
          OriginAccessIdentity: origin-access-identity/cloudfront/E127EXAMPLE51Z
      Enabled: 'true'
      Comment: Some comment
      DefaultRootObject: index.html
      Logging:
        IncludeCookies: 'false'
        Bucket: amzn-s3-demo-logging-bucket.s3.amazonaws.com
        Prefix: myprefix
      Aliases:
      - mysite.example.com
      - yoursite.example.com
      DefaultCacheBehavior:
        AllowedMethods:
        - DELETE
        - GET
        - HEAD
        - OPTIONS
        - PATCH
        - POST
        - PUT
        TargetOriginId: myS3Origin
        ForwardedValues:
          QueryString: 'false'
          Cookies:
            Forward: none
        TrustedSigners:
        - 1234567890EX
        - 1234567891EX
        ViewerProtocolPolicy: allow-all
      PriceClass: PriceClass_200
      Restrictions:
        GeoRestriction:
          RestrictionType: whitelist
          Locations:
          - AQ
          - CV
```

```
        ViewerCertificate:
          CloudFrontDefaultCertificate: 'true'
```

带自定义源的 Amazon CloudFront 分配资源

以下示例模板显示使用 CustomOrigin 的 Amazon CloudFront 分发。

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myDistribution" : {
            "Type" : "AWS::CloudFront::Distribution",
            "Properties" : {
                "DistributionConfig" : {
                    "Origins" : [ {
                            "DomainName" : "www.example.com",
                            "Id" : "myCustomOrigin",
                            "CustomOriginConfig" : {
                                "HTTPPort" : "80",
                                "HTTPSPort" : "443",
                                "OriginProtocolPolicy" : "http-only"
                            }
                    } ],
                    "Enabled" : "true",
                    "Comment" : "Somecomment",
                    "DefaultRootObject" : "index.html",
                    "Logging" : {
                        "IncludeCookies" : "true",
                        "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
                        "Prefix": "myprefix"
                    },
                    "Aliases" : [
                        "mysite.example.com",
                        "*.yoursite.example.com"
                    ],
                    "DefaultCacheBehavior" : {
                        "TargetOriginId" : "myCustomOrigin",
                        "SmoothStreaming" : "false",
                        "ForwardedValues" : {
                            "QueryString" : "false",
                            "Cookies" : { "Forward" : "all" }
```

```
                        },
                        "TrustedSigners" : [
                            "1234567890EX",
                            "1234567891EX"
                        ],
                        "ViewerProtocolPolicy" : "allow-all"
                    },
                    "CustomErrorResponses" : [ {
                        "ErrorCode" : "404",
                        "ResponsePagePath" : "/error-pages/404.html",
                        "ResponseCode" : "200",
                        "ErrorCachingMinTTL" : "30"
                    } ],
                    "PriceClass" : "PriceClass_200",
                    "Restrictions" : {
                        "GeoRestriction" : {
                            "RestrictionType" : "whitelist",
                            "Locations" : [ "AQ", "CV" ]
                        }
                    },
                    "ViewerCertificate": { "CloudFrontDefaultCertificate" : "true" }
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
        - DomainName: www.example.com
          Id: myCustomOrigin
          CustomOriginConfig:
            HTTPPort: '80'
            HTTPSPort: '443'
            OriginProtocolPolicy: http-only
        Enabled: 'true'
```

```
        Comment: Somecomment
        DefaultRootObject: index.html
        Logging:
          IncludeCookies: 'true'
          Bucket: amzn-s3-demo-logging-bucket.s3.amazonaws.com
          Prefix: myprefix
        Aliases:
        - mysite.example.com
        - "*.yoursite.example.com"
        DefaultCacheBehavior:
          TargetOriginId: myCustomOrigin
          SmoothStreaming: 'false'
          ForwardedValues:
            QueryString: 'false'
            Cookies:
              Forward: all
          TrustedSigners:
          - 1234567890EX
          - 1234567891EX
          ViewerProtocolPolicy: allow-all
        CustomErrorResponses:
        - ErrorCode: '404'
          ResponsePagePath: "/error-pages/404.html"
          ResponseCode: '200'
          ErrorCachingMinTTL: '30'
        PriceClass: PriceClass_200
        Restrictions:
          GeoRestriction:
            RestrictionType: whitelist
            Locations:
            - AQ
            - CV
        ViewerCertificate:
          CloudFrontDefaultCertificate: 'true'
```

具有多源支持的 Amazon CloudFront 分配

以下示例模板演示如何声明带多源支持的 CloudFront 分配。在 DistributionConfig 中，提供了源列表，并且设置了 DefaultCacheBehavior。

JSON

```
{
```

```
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myDistribution" : {
            "Type" : "AWS::CloudFront::Distribution",
            "Properties" : {
                "DistributionConfig" : {
                    "Origins" : [ {
                        "Id" : "myS3Origin",
                        "DomainName" : "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "S3OriginConfig" : {
                            "OriginAccessIdentity" : "origin-access-identity/
cloudfront/E127EXAMPLE51Z"
                        }
                     },
                     {
                         "Id" : "myCustomOrigin",
                         "DomainName" : "www.example.com",
                         "CustomOriginConfig" : {
                             "HTTPPort" : "80",
                             "HTTPSPort" : "443",
                             "OriginProtocolPolicy" : "http-only"
                         }
                     }
                   ],
                   "Enabled" : "true",
                   "Comment" : "Some comment",
                   "DefaultRootObject" : "index.html",
                   "Logging" : {
                       "IncludeCookies" : "true",
                       "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
                       "Prefix" : "myprefix"
                   },
                   "Aliases" : [ "mysite.example.com", "yoursite.example.com" ],
                   "DefaultCacheBehavior" : {
                       "TargetOriginId" : "myS3Origin",
                       "ForwardedValues" : {
                           "QueryString" : "false",
                           "Cookies" : { "Forward" : "all" }
                        },
                       "TrustedSigners" : [ "1234567890EX", "1234567891EX"  ],
                       "ViewerProtocolPolicy" : "allow-all",
                       "MinTTL" : "100",
                       "SmoothStreaming" : "true"
                   },
```

```
                    "CacheBehaviors" : [ {
                            "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS",
 "PATCH", "POST", "PUT" ],
                            "TargetOriginId" : "myS3Origin",
                            "ForwardedValues" : {
                                "QueryString" : "true",
                                "Cookies" : { "Forward" : "none" }
                            },
                            "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
                            "ViewerProtocolPolicy" : "allow-all",
                            "MinTTL" : "50",
                            "PathPattern" : "images1/*.jpg"
                        },
                        {
                            "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS",
 "PATCH", "POST", "PUT" ],
                            "TargetOriginId" : "myCustomOrigin",
                            "ForwardedValues" : {
                                "QueryString" : "true",
                                "Cookies" : { "Forward" : "none" }
                            },
                            "TrustedSigners" : [ "1234567890EX", "1234567891EX"  ],
                            "ViewerProtocolPolicy" : "allow-all",
                            "MinTTL" : "50",
                            "PathPattern" : "images2/*.jpg"
                        }
                    ],
                    "CustomErrorResponses" : [ {
                        "ErrorCode" : "404",
                        "ResponsePagePath" : "/error-pages/404.html",
                        "ResponseCode" : "200",
                        "ErrorCachingMinTTL" : "30"
                    } ],
                    "PriceClass" : "PriceClass_All",
                    "ViewerCertificate" : { "CloudFrontDefaultCertificate" : "true" }
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
        - Id: myS3Origin
          DomainName: amzn-s3-demo-bucket.s3.amazonaws.com
          S3OriginConfig:
            OriginAccessIdentity: origin-access-identity/cloudfront/E127EXAMPLE51Z
        - Id: myCustomOrigin
          DomainName: www.example.com
          CustomOriginConfig:
            HTTPPort: '80'
            HTTPSPort: '443'
            OriginProtocolPolicy: http-only
        Enabled: 'true'
        Comment: Some comment
        DefaultRootObject: index.html
        Logging:
          IncludeCookies: 'true'
          Bucket: amzn-s3-demo-logging-bucket.s3.amazonaws.com
          Prefix: myprefix
        Aliases:
        - mysite.example.com
        - yoursite.example.com
        DefaultCacheBehavior:
          TargetOriginId: myS3Origin
          ForwardedValues:
            QueryString: 'false'
            Cookies:
              Forward: all
          TrustedSigners:
          - 1234567890EX
          - 1234567891EX
          ViewerProtocolPolicy: allow-all
          MinTTL: '100'
          SmoothStreaming: 'true'
        CacheBehaviors:
        - AllowedMethods:
          - DELETE
```

```
          - GET
          - HEAD
          - OPTIONS
          - PATCH
          - POST
          - PUT
          TargetOriginId: myS3Origin
          ForwardedValues:
            QueryString: 'true'
            Cookies:
              Forward: none
          TrustedSigners:
          - 1234567890EX
          - 1234567891EX
          ViewerProtocolPolicy: allow-all
          MinTTL: '50'
          PathPattern: images1/*.jpg
        - AllowedMethods:
          - DELETE
          - GET
          - HEAD
          - OPTIONS
          - PATCH
          - POST
          - PUT
          TargetOriginId: myCustomOrigin
          ForwardedValues:
            QueryString: 'true'
            Cookies:
              Forward: none
          TrustedSigners:
          - 1234567890EX
          - 1234567891EX
          ViewerProtocolPolicy: allow-all
          MinTTL: '50'
          PathPattern: images2/*.jpg
        CustomErrorResponses:
        - ErrorCode: '404'
          ResponsePagePath: "/error-pages/404.html"
          ResponseCode: '200'
          ErrorCachingMinTTL: '30'
        PriceClass: PriceClass_All
        ViewerCertificate:
```

```
          CloudFrontDefaultCertificate: 'true'
```

以 Lambda 函数为源的 Amazon CloudFront 分发

以下示例创建了一个前端为指定的 Lambda 函数 URL（作为参数提供）的 CloudFront 分发，启用了仅限 HTTPS 访问、缓存、压缩和全球交付。此示例将 Lambda URL 配置为自定义 HTTPS 来源，并且应用标准 Amazon 缓存策略。该分发针对性能进行了优化，支持 HTTP/2 和 IPv6，输出 CloudFront 域名，允许用户通过 CDN 支持的安全端点访问 Lambda 函数。有关更多信息，请参阅 Amazon 博客上的 Using Amazon CloudFront with Amazon Lambda as origin to accelerate your web applications。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "LambdaEndpoint": {
            "Type": "String",
            "Description": "The Lambda function URL endpoint without the 'https://'"
        }
    },
    "Resources": {
        "MyDistribution": {
            "Type": "AWS::CloudFront::Distribution",
            "Properties": {
                "DistributionConfig": {
                    "PriceClass": "PriceClass_All",
                    "HttpVersion": "http2",
                    "IPV6Enabled": true,
                    "Origins": [
                        {
                            "DomainName": {
                                "Ref": "LambdaEndpoint"
                            },
                            "Id": "LambdaOrigin",
                            "CustomOriginConfig": {
                                "HTTPSPort": 443,
                                "OriginProtocolPolicy": "https-only"
                            }
                        }
                    ],
                    "Enabled": "true",
                    "DefaultCacheBehavior": {
                        "TargetOriginId": "LambdaOrigin",
```

```
                        "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e58f6",
                        "ViewerProtocolPolicy": "redirect-to-https",
                        "SmoothStreaming": "false",
                        "Compress": "true"
                    }
                }
            }
        }
    },
    "Outputs": {
        "CloudFrontDomain": {
            "Description": "CloudFront default domain name configured",
            "Value": {
                "Fn::Sub": "https://${MyDistribution.DomainName}/"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  LambdaEndpoint:
    Type: String
    Description: The Lambda function URL endpoint without the 'https://'
Resources:
  MyDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        PriceClass: PriceClass_All
        HttpVersion: http2
        IPV6Enabled: true
        Origins:
        - DomainName: !Ref LambdaEndpoint
          Id: LambdaOrigin
          CustomOriginConfig:
            HTTPSPort: 443
            OriginProtocolPolicy: https-only
        Enabled: 'true'
        DefaultCacheBehavior:
          TargetOriginId: LambdaOrigin
```

```
          CachePolicyId: '658327ea-f89d-4fab-a63d-7e88639e58f6'
          ViewerProtocolPolicy: redirect-to-https
          SmoothStreaming: 'false'
          Compress: 'true'
Outputs:
  CloudFrontDomain:
    Description: CloudFront default domain name configured
    Value: !Sub https://${MyDistribution.DomainName}/
```

另请参阅

有关向 Route 53 记录添加自定义别名以便更好地为 CloudFront 分发命名的示例，请参阅 CloudFront 分配的别名资源记录集。

Amazon CloudWatch 模板代码段

您可以在 Amazon CloudFormation 中使用这些示例模板代码段来帮助描述 Amazon CloudWatch 资源。有关更多信息，请参阅 Amazon CloudWatch resource type reference。

主题

- 账单警报
- CPU 使用率警报
- 恢复 Amazon Elastic Compute Cloud 实例
- 创建基本控制面板
- 使用并排显示小部件创建控制面板

账单警报

在下面的示例中，Amazon CloudWatch 在您的 Amazon 账户费用超过警报阈值时发送电子邮件通知。要接收使用情况通知，请启用账单提醒。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的创建账单警报以监控预估 Amazon 费用。

JSON

```
"SpendingAlarm": {
  "Type": "AWS::CloudWatch::Alarm",
  "Properties": {
    "AlarmDescription": { "Fn::Join": ["", [
      "Alarm if AWS spending is over $",
```

```
      { "Ref": "AlarmThreshold" }
    ]]},
    "Namespace": "AWS/Billing",
    "MetricName": "EstimatedCharges",
    "Dimensions": [{
      "Name": "Currency",
      "Value" : "USD"
    }],
    "Statistic": "Maximum",
    "Period": "21600",
    "EvaluationPeriods": "1",
    "Threshold": { "Ref": "AlarmThreshold" },
    "ComparisonOperator": "GreaterThanThreshold",
    "AlarmActions": [{
      "Ref": "BillingAlarmNotification"
    }],
    "InsufficientDataActions": [{
      "Ref": "BillingAlarmNotification"
    }]
  }
}
```

YAML

```
SpendingAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription:
      'Fn::Join':
        - ''
        - - Alarm if AWS spending is over $
          - !Ref: AlarmThreshold
    Namespace: AWS/Billing
    MetricName: EstimatedCharges
    Dimensions:
    - Name: Currency
      Value: USD
    Statistic: Maximum
    Period: '21600'
    EvaluationPeriods: '1'
    Threshold:
      !Ref: "AlarmThreshold"
    ComparisonOperator: GreaterThanThreshold
```

```
    AlarmActions:
    - !Ref: "BillingAlarmNotification"
    InsufficientDataActions:
    - !Ref: "BillingAlarmNotification"
```

CPU 使用率警报

以下示例代码段创建一个警报，该警报在三个评估周期内 Amazon EC2 实例的平均 CPU 利用率超出 90% 的时间大于 60 秒时发送通知。

JSON

```
"CPUAlarm" : {
  "Type" : "AWS::CloudWatch::Alarm",
  "Properties" : {
    "AlarmDescription" : "CPU alarm for my instance",
    "AlarmActions" : [ { "Ref" : "logical name of an AWS::SNS::Topic resource" } ],
    "MetricName" : "CPUUtilization",
    "Namespace" : "AWS/EC2",
    "Statistic" : "Average",
    "Period" : "60",
    "EvaluationPeriods" : "3",
    "Threshold" : "90",
    "ComparisonOperator" : "GreaterThanThreshold",
    "Dimensions" : [ {
      "Name" : "InstanceId",
      "Value" : { "Ref" : "logical name of an AWS::EC2::Instance resource" }
    } ]
  }
}
```

YAML

```
CPUAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: CPU alarm for my instance
    AlarmActions:
    - !Ref: "logical name of an AWS::SNS::Topic resource"
    MetricName: CPUUtilization
    Namespace: AWS/EC2
    Statistic: Average
    Period: '60'
```

```
    EvaluationPeriods: '3'
    Threshold: '90'
    ComparisonOperator: GreaterThanThreshold
    Dimensions:
    - Name: InstanceId
      Value: !Ref: "logical name of an AWS::EC2::Instance resource"
```

恢复 Amazon Elastic Compute Cloud 实例

当以下 CloudWatch 警报在连续的 15 分钟时间内存在状态检查故障时，它将恢复 EC2 实例。有关警报操作的更多信息，请参阅《Amazon CloudWatch 用户指南》中的创建警报以停止、终止、重启或恢复 EC2 实例。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters" : {
    "RecoveryInstance" : {
      "Description" : "The EC2 instance ID to associate this alarm with.",
      "Type" : "AWS::EC2::Instance::Id"
    }
  },
  "Resources": {
    "RecoveryTestAlarm": {
      "Type": "AWS::CloudWatch::Alarm",
      "Properties": {
        "AlarmDescription": "Trigger a recovery when instance status check fails for 15
 consecutive minutes.",
        "Namespace": "AWS/EC2" ,
        "MetricName": "StatusCheckFailed_System",
        "Statistic": "Minimum",
        "Period": "60",
        "EvaluationPeriods": "15",
        "ComparisonOperator": "GreaterThanThreshold",
        "Threshold": "0",
        "AlarmActions": [ {"Fn::Join" : ["", ["arn:aws:automate:", { "Ref" :
 "AWS::Region" }, ":ec2:recover" ]]} ],
        "Dimensions": [{"Name": "InstanceId","Value": {"Ref": "RecoveryInstance"}}]
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  RecoveryInstance:
    Description: The EC2 instance ID to associate this alarm with.
    Type: AWS::EC2::Instance::Id
Resources:
  RecoveryTestAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: Trigger a recovery when instance status check fails for 15
        consecutive minutes.
      Namespace: AWS/EC2
      MetricName: StatusCheckFailed_System
      Statistic: Minimum
      Period: '60'
      EvaluationPeriods: '15'
      ComparisonOperator: GreaterThanThreshold
      Threshold: '0'
      AlarmActions: [ !Sub "arn:aws:automate:${AWS::Region}:ec2:recover" ]
      Dimensions:
      - Name: InstanceId
        Value: !Ref: RecoveryInstance
```

创建基本控制面板

以下示例创建一个简单的 CloudWatch 控制面板，其中包含一个显示 CPU 使用率的指标小部件和一个显示消息的文本小部件。

JSON

```
{
    "BasicDashboard": {
        "Type": "AWS::CloudWatch::Dashboard",
        "Properties": {
            "DashboardName": "Dashboard1",
            "DashboardBody": "{\"widgets\":[{\"type\":\"metric\",\"x\":0,\"y\":0,
\"width\":12,\"height\":6,\"properties\":{\"metrics\":[[\"AWS/EC2\",\"CPUUtilization
\",\"InstanceId\",\"i-012345\"]],\"period\":300,\"stat\":\"Average\",\"region\":\"us-
east-1\",\"title\":\"EC2 Instance CPU\"}},{\"type\":\"text\",\"x\":0,\"y\":7,\"width
\":3,\"height\":3,\"properties\":{\"markdown\":\"Hello world\"}}]}"
        }
```

```
    }
}
```

YAML

```
BasicDashboard:
  Type: AWS::CloudWatch::Dashboard
  Properties:
    DashboardName: Dashboard1
    DashboardBody: '{"widgets":
[{"type":"metric","x":0,"y":0,"width":12,"height":6,"properties":{"metrics":[["AWS/
EC2","CPUUtilization","InstanceId","i-012345"]],"period":300,"stat":"Average","region":"us-
east-1","title":"EC2 Instance CPU"}},
{"type":"text","x":0,"y":7,"width":3,"height":3,"properties":{"markdown":"Hello
 world"}}]}'
```

使用并排显示小部件创建控制面板

以下示例创建一个包含两个指标小部件 (并排显示) 的控制面板。

JSON

```
{
    "DashboardSideBySide": {
        "Type": "AWS::CloudWatch::Dashboard",
        "Properties": {
            "DashboardName": "Dashboard1",
            "DashboardBody": "{\"widgets\":[{\"type\":\"metric\",\"x\":0,\"y\":0,
\"width\":12,\"height\":6,\"properties\":{\"metrics\":[[\"AWS/EC2\",\"CPUUtilization
\",\"InstanceId\",\"i-012345\"]],\"period\":300,\"stat\":\"Average\",\"region\":
\"us-east-1\",\"title\":\"EC2 Instance CPU\"}},{\"type\":\"metric\",\"x\":12,\"y\":0,
\"width\":12,\"height\":6,\"properties\":{\"metrics\":[[\"AWS/S3\",\"BucketSizeBytes\",
\"BucketName\",\"amzn-s3-demo-bucket\"]],\"period\":86400,\"stat\":\"Maximum\",\"region
\":\"us-east-1\",\"title\":\"amzn-s3-demo-bucket bytes\"}}]}"
        }
    }
}
```

YAML

```
DashboardSideBySide:
  Type: AWS::CloudWatch::Dashboard
```

```
  Properties:
    DashboardName: Dashboard1
    DashboardBody: '{"widgets":
[{"type":"metric","x":0,"y":0,"width":12,"height":6,"properties":{"metrics":[["AWS/
EC2","CPUUtilization","InstanceId","i-012345"]],"period":300,"stat":"Average","region":"us-
east-1","title":"EC2 Instance CPU"}},
{"type":"metric","x":12,"y":0,"width":12,"height":6,"properties":
{"metrics":[["AWS/S3","BucketSizeBytes","BucketName","amzn-s3-demo-
bucket"]],"period":86400,"stat":"Maximum","region":"us-east-1","title":"amzn-s3-demo-
bucket bytes"}}]}'
```

Amazon CloudWatch Logs 模板代码段

Amazon CloudWatch Logs 能够监控系统、应用程序和来自 Amazon EC2 实例或其他源的自定义日志文件。您可以使用 Amazon CloudFormation 配置和管理日志组和指标筛选条件。有关 Amazon CloudWatch Logs 的更多信息，请参阅 Amazon CloudWatch Logs 用户指南。

主题

- 从 Linux 实例将日志发送到 CloudWatch Logs
- 从 Windows 实例将日志发送到 CloudWatch Logs
- 另请参阅

从 Linux 实例将日志发送到 CloudWatch Logs

以下模板演示了如何在集成 CloudWatch Logs 的 Amazon Linux 2023 上设置 Web 服务器。该模板将执行以下任务：

- 安装 Apache 和 PHP。
- 配置 CloudWatch Logs 代理，将 Apache 访问日志转发到 CloudWatch Logs。
- 设置一个 IAM 角色让 CloudWatch 代理将日志数据发送到 CloudWatch Logs。
- 创建自定义警报和通知，监控 404 错误或高带宽使用情况。

来自 Web 服务器的日志事件为 CloudWatch 警报提供了指标数据。两个指标筛选条件描述了将日志信息转换成 CloudWatch 指标的方法。404 指标用于统计出现 404 错误的次数。大小指标用于跟踪请求的大小。如果两分钟内出现两次以上的 404 错误，或平均请求大小在高于 3500 KB 的情况下持续 10 分钟，这两个 CloudWatch 警报将发送通知。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Sample template that sets up and configures CloudWatch Logs on
 Amazon Linux 2023 instance.",
    "Parameters": {
        "KeyName": {
            "Description": "Name of an existing EC2 KeyPair to enable SSH access to the
 instances",
            "Type": "AWS::EC2::KeyPair::KeyName",
            "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
        },
        "SSHLocation": {
            "Description": "The IP address range that can be used to SSH to the EC2
 instances",
            "Type": "String",
            "MinLength": "9",
            "MaxLength": "18",
            "Default": "0.0.0.0/0",
            "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\
\d{1,2})",
            "ConstraintDescription": "must be a valid IP CIDR range of the form
 x.x.x.x/x."
        },
        "OperatorEmail": {
            "Description": "Email address to notify when CloudWatch alarms are
 triggered (404 errors or high bandwidth usage)",
            "Type": "String"
        }
    },
    "Resources": {
        "LogRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ec2.amazonaws.com"
                                ]
```

```
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "LogRolePolicy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "logs:PutLogEvents",
                                    "logs:DescribeLogStreams",
                                    "logs:DescribeLogGroups",
                                    "logs:CreateLogGroup",
                                    "logs:CreateLogStream"
                                ],
                                "Resource": "*"
                            }
                        ]
                    }
                }
            ]
        }
    },
    "LogRoleInstanceProfile": {
        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [{"Ref": "LogRole"}]
        }
    },
    "WebServerSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Enable HTTP access via port 80 and SSH access via
port 22",
            "SecurityGroupIngress": [
```

```
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "CidrIp": "0.0.0.0/0"
                    },
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 22,
                        "ToPort": 22,
                        "CidrIp": {"Ref": "SSHLocation"}
                    }
                ]
            }
        },
        "WebServerHost": {
            "Type": "AWS::EC2::Instance",
            "Metadata": {
                "Comment": "Install a simple PHP application on Amazon Linux 2023",
                "AWS::CloudFormation::Init": {
                    "config": {
                        "packages": {
                            "dnf": {
                                "httpd": [],
                                "php": [],
                                "php-fpm": []
                            }
                        },
                        "files": {
                            "/etc/amazon-cloudwatch-agent/amazon-cloudwatch-
agent.json": {
                                "content": {
                                    "logs": {
                                        "logs_collected": {
                                            "files": {
                                                "collect_list": [{
                                                    "file_path": "/var/log/httpd/
access_log",
                                                    "log_group_name": {"Ref":
 "WebServerLogGroup"},
                                                    "log_stream_name": "{instance_id}/
apache.log",
                                                    "timestamp_format": "%d/%b/%Y:%H:
%M:%S %z"
```

```
                                            }]
                                        }
                                    }
                                }
                            },
                            "mode": "000644",
                            "owner": "root",
                            "group": "root"
                        },
                        "/var/www/html/index.php": {
                            "content": "<?php\necho '<h1>AWS CloudFormation sample
PHP application on Amazon Linux 2023</h1>';\n?>\n",
                            "mode": "000644",
                            "owner": "apache",
                            "group": "apache"
                        },
                        "/etc/cfn/cfn-hup.conf": {
                            "content":  {
                                "Fn::Join": [
                                    "",
                                    [
                                        "[main]\n",
                                        "stack=",
                                        {"Ref": "AWS::StackId"},
                                        "\n",
                                        "region=",
                                        {"Ref": "AWS::Region"},
                                        "\n"
                                    ]
                                ]
                            },
                            "mode": "000400",
                            "owner": "root",
                            "group": "root"
                        },
                        "/etc/cfn/hooks.d/cfn-auto-reloader.conf": {
                            "content": {
                                "Fn::Join": [
                                    "",
                                    [
                                        "[cfn-auto-reloader-hook]\n",
                                        "triggers=post.update\n",

"path=Resources.WebServerHost.Metadata.AWS::CloudFormation::Init\n",
```

```
                                        "action=/opt/aws/bin/cfn-init -s ",
                                        {"Ref": "AWS::StackId"},
                                        " -r WebServerHost ",
                                        " --region     ",
                                        {"Ref": "AWS::Region"},
                                        "\n",
                                        "runas=root\n"
                                    ]
                                ]
                            }
                        }
                    },
                    "services": {
                        "systemd": {
                            "httpd": {
                                "enabled": "true",
                                "ensureRunning": "true"
                            },
                            "php-fpm": {
                                "enabled": "true",
                                "ensureRunning": "true"
                            }
                        }
                    }
                }
            },
            "CreationPolicy": {
                "ResourceSignal": {
                    "Timeout": "PT5M"
                }
            },
            "Properties": {
                "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-
ami-kernel-6.1-x86_64}}",
                "KeyName": {"Ref": "KeyName"},
                "InstanceType": "t3.micro",
                "SecurityGroupIds": [{"Ref": "WebServerSecurityGroup"}],
                "IamInstanceProfile": {"Ref": "LogRoleInstanceProfile"},
                "UserData": {"Fn::Base64": {"Fn::Join": [ "", [
                    "#!/bin/bash\n",
                    "dnf update -y aws-cfn-bootstrap\n",
                    "dnf install -y amazon-cloudwatch-agent\n",
```

```
                    "/opt/aws/bin/cfn-init -v --stack ", {"Ref": "AWS::StackName"}, "
 --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "\n",
                    "\n",
                    "# Verify Apache log directory exists and create if needed\n",
                    "mkdir -p /var/log/httpd\n",
                    "\n",
                    "# Start CloudWatch agent\n",
                    "/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -
a fetch-config -m ec2 -c file:/etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json
 -s\n",
                    "\n",
                    "# Signal success\n",
                    "/opt/aws/bin/cfn-signal -e $? --stack ", {"Ref":
 "AWS::StackName"}, " --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "\n"
                ]]}}
            }
        },
        "WebServerLogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "DeletionPolicy": "Retain",
            "UpdateReplacePolicy": "Retain",
            "Properties": {
                "RetentionInDays": 7
            }
        },
        "404MetricFilter": {
            "Type": "AWS::Logs::MetricFilter",
            "Properties": {
                "LogGroupName": {"Ref": "WebServerLogGroup"},
                "FilterPattern": "[ip, identity, user_id, timestamp, request,
 status_code = 404, size, ...]",
                "MetricTransformations": [
                    {
                        "MetricValue": "1",
                        "MetricNamespace": "test/404s",
                        "MetricName": "test404Count"
                    }
                ]
            }
        },
        "BytesTransferredMetricFilter": {
            "Type": "AWS::Logs::MetricFilter",
            "Properties": {
                "LogGroupName": {"Ref": "WebServerLogGroup"},
```

```
                "FilterPattern": "[ip, identity, user_id, timestamp, request,
status_code, size, ...]",
                "MetricTransformations": [
                    {
                        "MetricValue": "$size",
                        "MetricNamespace": "test/BytesTransferred",
                        "MetricName": "testBytesTransferred"
                    }
                ]
            }
        },
        "404Alarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": "The number of 404s is greater than 2 over 2
minutes",
                "MetricName": "test404Count",
                "Namespace": "test/404s",
                "Statistic": "Sum",
                "Period": "60",
                "EvaluationPeriods": "2",
                "Threshold": "2",
                "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
                "ComparisonOperator": "GreaterThanThreshold"
            }
        },
        "BandwidthAlarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": "The average volume of traffic is greater 3500 KB
over 10 minutes",
                "MetricName": "testBytesTransferred",
                "Namespace": "test/BytesTransferred",
                "Statistic": "Average",
                "Period": "300",
                "EvaluationPeriods": "2",
                "Threshold": "3500",
                "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
                "ComparisonOperator": "GreaterThanThreshold"
            }
        },
        "AlarmNotificationTopic": {
            "Type": "AWS::SNS::Topic",
            "Properties": {
```

```
                "Subscription": [{"Endpoint": {"Ref": "OperatorEmail"}, "Protocol":
 "email"}]
            }
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The instance ID of the web server",
            "Value": {"Ref": "WebServerHost"}
        },
        "WebsiteURL": {
            "Value": {"Fn::Sub": "http://${WebServerHost.PublicDnsName}"},
            "Description": "URL for the web server"
        },
        "PublicIP": {
            "Description": "Public IP address of the web server",
            "Value": {"Fn::GetAtt": ["WebServerHost","PublicIp"]
            }
        },
        "CloudWatchLogGroupName": {
            "Description": "The name of the CloudWatch log group",
            "Value": {"Ref": "WebServerLogGroup"}
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Sample template that sets up and configures CloudWatch Logs on Amazon
 Linux 2023 instance.
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
```

```
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  OperatorEmail:
    Description: Email address to notify when CloudWatch alarms are triggered (404
 errors or high bandwidth usage)
    Type: String
Resources:
  LogRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      Policies:
        - PolicyName: LogRolePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'logs:PutLogEvents'
                  - 'logs:DescribeLogStreams'
                  - 'logs:DescribeLogGroups'
                  - 'logs:CreateLogGroup'
                  - 'logs:CreateLogStream'
                Resource: '*'
  LogRoleInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles:
        - !Ref LogRole
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
```

```
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref SSHLocation
  WebServerHost:
    Type: AWS::EC2::Instance
    Metadata:
      Comment: Install a simple PHP application on Amazon Linux 2023
      'AWS::CloudFormation::Init':
        config:
          packages:
            dnf:
              httpd: []
              php: []
              php-fpm: []
          files:
            /etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json:
              content: !Sub |
                {
                  "logs": {
                    "logs_collected": {
                      "files": {
                        "collect_list": [
                          {
                            "file_path": "/var/log/httpd/access_log",
                            "log_group_name": "${WebServerLogGroup}",
                            "log_stream_name": "{instance_id}/apache.log",
                            "timestamp_format": "%d/%b/%Y:%H:%M:%S %z"
                          }
                        ]
                      }
                    }
                  }
                }
              mode: '000644'
              owner: root
              group: root
            /var/www/html/index.php:
              content: |
                <?php echo '<h1>AWS CloudFormation sample PHP application on Amazon
  Linux 2023</h1>';
```

```
                ?>
              mode: '000644'
              owner: apache
              group: apache
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.WebServerHost.Metadata.AWS::CloudFormation::Init
                action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerHost --
region ${AWS::Region}
                runas=root
          services:
            systemd:
              httpd:
                enabled: 'true'
                ensureRunning: 'true'
              php-fpm:
                enabled: 'true'
                ensureRunning: 'true'
    CreationPolicy:
      ResourceSignal:
        Timeout: PT5M
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-
kernel-6.1-x86_64}}'
      KeyName: !Ref KeyName
      InstanceType: t3.micro
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      IamInstanceProfile: !Ref LogRoleInstanceProfile
      UserData: !Base64
        Fn::Sub: |
          #!/bin/bash
          dnf update -y aws-cfn-bootstrap
          dnf install -y amazon-cloudwatch-agent
```

```
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource WebServerHost
 --region ${AWS::Region}

          # Verify Apache log directory exists and create if needed
          mkdir -p /var/log/httpd

          # Start CloudWatch agent
          /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -c file:/etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json -s

          # Signal success
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerHost --region ${AWS::Region}
          echo "Done"
  WebServerLogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      RetentionInDays: 7
  404MetricFilter:
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref WebServerLogGroup
      FilterPattern: >-
        [ip, identity, user_id, timestamp, request, status_code = 404, size, ...]
      MetricTransformations:
        - MetricValue: '1'
          MetricNamespace: test/404s
          MetricName: test404Count
  BytesTransferredMetricFilter:
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref WebServerLogGroup
      FilterPattern: '[ip, identity, user_id, timestamp, request, status_code,
 size, ...]'
      MetricTransformations:
        - MetricValue: $size
          MetricNamespace: test/BytesTransferred
          MetricName: testBytesTransferred
  404Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: The number of 404s is greater than 2 over 2 minutes
```

```
      MetricName: test404Count
      Namespace: test/404s
      Statistic: Sum
      Period: '60'
      EvaluationPeriods: '2'
      Threshold: '2'
      AlarmActions:
        - !Ref AlarmNotificationTopic
      ComparisonOperator: GreaterThanThreshold
  BandwidthAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: The average volume of traffic is greater 3500 KB over 10
 minutes
      MetricName: testBytesTransferred
      Namespace: test/BytesTransferred
      Statistic: Average
      Period: '300'
      EvaluationPeriods: '2'
      Threshold: '3500'
      AlarmActions:
        - !Ref AlarmNotificationTopic
      ComparisonOperator: GreaterThanThreshold
  AlarmNotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !Ref OperatorEmail
          Protocol: email
Outputs:
  InstanceId:
    Description: The instance ID of the web server
    Value: !Ref WebServerHost
  WebsiteURL:
    Value: !Sub 'http://${WebServerHost.PublicDnsName}'
    Description: URL for the web server
  PublicIP:
    Description: Public IP address of the web server
    Value: !GetAtt WebServerHost.PublicIp
  CloudWatchLogGroupName:
    Description: The name of the CloudWatch log group
    Value: !Ref WebServerLogGroup
```

从 Windows 实例将日志发送到 CloudWatch Logs

以下模板为 Windows 2012R2 实例配置 CloudWatch Logs。

Windows 上的 CloudWatch Logs 代理（Windows 2012R2 和 Windows 2016 AMI 上的 SSM 代理）仅在启动之后才会发送日志，因此不会发送启动之前生成的任何日志。针对此问题，该模板帮助确保在任何日志写入之前启动代理，方法为：

- 将代理安装程序配置为 cfn-init `config` 中的第一个 `configSets` 项。
- 使用 `waitAfterCompletion` 在启动代理的命令之后插入一个暂停。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Sample template that sets up and configures CloudWatch Logs on
 Windows 2012R2 instance.",
    "Parameters": {
        "KeyPair": {
            "Description": "Name of an existing EC2 KeyPair to enable RDP access to the
 instances",
            "Type": "AWS::EC2::KeyPair::KeyName",
            "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
        },
        "RDPLocation": {
            "Description": "The IP address range that can be used to RDP to the EC2
 instances",
            "Type": "String",
            "MinLength": "9",
            "MaxLength": "18",
            "Default": "0.0.0.0/0",
            "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\
\d{1,2})",
            "ConstraintDescription": "must be a valid IP CIDR range of the form
 x.x.x.x/x."
        },
        "OperatorEmail": {
            "Description": "Email address to notify when CloudWatch alarms are
 triggered (404 errors)",
            "Type": "String"
        }
    },
```

```
    "Resources": {
        "WebServerSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Enable HTTP access via port 80 and RDP access via
port 3389",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": "80",
                        "ToPort": "80",
                        "CidrIp": "0.0.0.0/0"
                    },
                    {
                        "IpProtocol": "tcp",
                        "FromPort": "3389",
                        "ToPort": "3389",
                        "CidrIp": {"Ref": "RDPLocation"}
                    }
                ]
            }
        },
        "LogRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ec2.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
                ],
```

```
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "LogRolePolicy",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "logs:Create*",
                                        "logs:PutLogEvents",
                                        "s3:GetObject"
                                    ],
                                    "Resource": [
                                        "arn:aws:logs:*:*:*",
                                        "arn:aws:s3:::*"
                                    ]
                                }
                            ]
                        }
                    }
                ]
            }
        },
        "LogRoleInstanceProfile": {
            "Type": "AWS::IAM::InstanceProfile",
            "Properties": {
                "Path": "/",
                "Roles": [{"Ref": "LogRole"}]
            }
        },
        "WebServerHost": {
            "Type": "AWS::EC2::Instance",
            "CreationPolicy": {
                "ResourceSignal": {
                    "Timeout": "PT15M"
                }
            },
            "Metadata": {
                "AWS::CloudFormation::Init": {
                    "configSets": {
                        "config": [
                            "00-ConfigureCWLogs",
```

```
                            "01-InstallWebServer",
                            "02-ConfigureApplication",
                            "03-Finalize"
                        ]
                    },
                    "00-ConfigureCWLogs": {
                        "files": {
                            "C:\\Program Files\\Amazon\\SSM\\Plugins\\awsCloudWatch\
\AWS.EC2.Windows.CloudWatch.json": {
                                "content": {
                                    "EngineConfiguration": {
                                        "Components": [
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "ApplicationEventLog",
                                                "Parameters": {
                                                    "Levels": "7",
                                                    "LogName": "Application"
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "SystemEventLog",
                                                "Parameters": {
                                                    "Levels": "7",
                                                    "LogName": "System"
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "SecurityEventLog",
                                                "Parameters": {
                                                    "Levels": "7",
                                                    "LogName": "Security"
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "EC2ConfigLog",
                                                "Parameters": {
```

```
                                                "CultureName": "en-US",
                                                "Encoding": "ASCII",
                                                "Filter": "EC2ConfigLog.txt",
                                                "LogDirectoryPath": "C:\\Program
 Files\\Amazon\\Ec2ConfigService\\Logs",
                                                "TimeZoneKind": "UTC",
                                                "TimestampFormat": "yyyy-MM-
ddTHH:mm:ss.fffZ:"
                                            }
                                        },
                                        {
                                            "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                            "Id": "CfnInitLog",
                                            "Parameters": {
                                                "CultureName": "en-US",
                                                "Encoding": "ASCII",
                                                "Filter": "cfn-init.log",
                                                "LogDirectoryPath": "C:\\cfn\\log",
                                                "TimeZoneKind": "Local",
                                                "TimestampFormat": "yyyy-MM-dd
 HH:mm:ss,fff"
                                            }
                                        },
                                        {
                                            "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                            "Id": "IISLogs",
                                            "Parameters": {
                                                "CultureName": "en-US",
                                                "Encoding": "UTF-8",
                                                "Filter": "",
                                                "LineCount": "3",
                                                "LogDirectoryPath": "C:\\inetpub\
\logs\\LogFiles\\W3SVC1",
                                                "TimeZoneKind": "UTC",
                                                "TimestampFormat": "yyyy-MM-dd
 HH:mm:ss"
                                            }
                                        },
                                        {
                                            "FullName":
 "AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent,AWS.E
                                            "Id": "MemoryPerformanceCounter",
```

```
                                                "Parameters": {
                                                    "CategoryName": "Memory",
                                                    "CounterName": "Available MBytes",
                                                    "DimensionName": "",
                                                    "DimensionValue": "",
                                                    "InstanceName": "",
                                                    "MetricName": "Memory",
                                                    "Unit": "Megabytes"
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchApplicationEventLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
ApplicationEventLog",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchSystemEventLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
SystemEventLog",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchSecurityEventLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
```

```
                                                "LogStream": "{instance_id}/
SecurityEventLog",
                                                "Region": {"Ref": "AWS::Region"},
                                                "SecretKey": ""
                                            }
                                        },
                                        {
                                            "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                            "Id": "CloudWatchEC2ConfigLog",
                                            "Parameters": {
                                                "AccessKey": "",
                                                "LogGroup": {"Ref": "LogGroup"},
                                                "LogStream": "{instance_id}/
EC2ConfigLog",
                                                "Region": {"Ref": "AWS::Region"},
                                                "SecretKey": ""
                                            }
                                        },
                                        {
                                            "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                            "Id": "CloudWatchCfnInitLog",
                                            "Parameters": {
                                                "AccessKey": "",
                                                "LogGroup": {"Ref": "LogGroup"},
                                                "LogStream": "{instance_id}/
CfnInitLog",
                                                "Region": {"Ref": "AWS::Region"},
                                                "SecretKey": ""
                                            }
                                        },
                                        {
                                            "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                            "Id": "CloudWatchIISLogs",
                                            "Parameters": {
                                                "AccessKey": "",
                                                "LogGroup": {"Ref": "LogGroup"},
                                                "LogStream": "{instance_id}/
IISLogs",
                                                "Region": {"Ref": "AWS::Region"},
                                                "SecretKey": ""
                                            }
```

```
                                        },
                                        {
                                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch",
                                            "Id": "CloudWatch",
                                            "Parameters": {
                                                "AccessKey": "",
                                                "NameSpace": "Windows/Default",
                                                "Region": {"Ref": "AWS::Region"},
                                                "SecretKey": ""
                                            }
                                        }
                                    ],
                                    "Flows": {
                                        "Flows": [

"ApplicationEventLog,CloudWatchApplicationEventLog",

"SystemEventLog,CloudWatchSystemEventLog",

"SecurityEventLog,CloudWatchSecurityEventLog",
                                            "EC2ConfigLog,CloudWatchEC2ConfigLog",
                                            "CfnInitLog,CloudWatchCfnInitLog",
                                            "IISLogs,CloudWatchIISLogs",
                                            "MemoryPerformanceCounter,CloudWatch"
                                        ]
                                    },
                                    "PollInterval": "00:00:05"
                                },
                                "IsEnabled": true
                            }
                        }
                    },
                    "commands": {
                        "0-enableSSM": {
                            "command": "powershell.exe -Command \"Set-Service -Name
AmazonSSMAgent -StartupType Automatic\" ",
                            "waitAfterCompletion": "0"
                        },
                        "1-restartSSM": {
                            "command": "powershell.exe -Command \"Restart-Service
AmazonSSMAgent \"",
                            "waitAfterCompletion": "30"
                        }
```

```
                        }
                    },
                    "01-InstallWebServer": {
                        "commands": {
                            "01_install_webserver": {
                                "command": "powershell.exe -Command \"Install-
WindowsFeature Web-Server  -IncludeAllSubFeature\"",
                                "waitAfterCompletion": "0"
                            }
                        }
                    },
                    "02-ConfigureApplication": {
                        "files": {
                            "c:\\Inetpub\\wwwroot\\index.htm": {
                                "content": "<html> <head> <title>Test Application
 Page</title> </head> <body> <h1>Congratulations!! Your IIS server is configured.</h1>
 </body> </html>"
                            }
                        }
                    },
                    "03-Finalize": {
                        "commands": {
                            "00_signal_success": {
                                "command": {
                                    "Fn::Sub": "cfn-signal.exe -e 0 --resource
 WebServerHost --stack ${AWS::StackName} --region ${AWS::Region}"
                                },
                                "waitAfterCompletion": "0"
                            }
                        }
                    }
                }
            },
            "Properties": {
                "KeyName": {
                    "Ref": "KeyPair"
                },
                "ImageId": "{{resolve:ssm:/aws/service/ami-windows-latest/
Windows_Server-2012-R2_RTM-English-64Bit-Base}}",
                "InstanceType": "t2.xlarge",
                "SecurityGroupIds": [{"Ref": "WebServerSecurityGroup"}],
                "IamInstanceProfile": {"Ref": "LogRoleInstanceProfile"},
                "UserData": {
                    "Fn::Base64": {
```

```
                        "Fn::Join": [
                            "",
                            [
                                "<script>\n",
                                "wmic product where \"description='Amazon SSM Agent' \"
 uninstall\n",
                                "wmic product where \"description='aws-cfn-bootstrap'
 \" uninstall \n",
                                "start /wait c:\\Windows\\system32\\msiexec /passive /
qn /i https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-win64-
latest.msi\n",
                                "powershell.exe -Command \"iwr https://
s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/
AmazonSSMAgentSetup.exe  -UseBasicParsing -OutFile C:\\AmazonSSMAgentSetup.exe\"\n",
                                "start /wait C:\\AmazonSSMAgentSetup.exe /install /
quiet\n",
                                "cfn-init.exe -v -c config -s ", {"Ref":
 "AWS::StackName"}, " --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "
 \n",
                                "</script>\n"
                            ]
                        ]
                    }
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "Properties": {
                "RetentionInDays": 7
            }
        },
        "404MetricFilter": {
            "Type": "AWS::Logs::MetricFilter",
            "Properties": {
                "LogGroupName": {"Ref": "LogGroup"},
                "FilterPattern": "[timestamps, serverip, method, uri, query, port,
 dash, clientip, useragent, status_code = 404, ...]",
                "MetricTransformations": [
                    {
                        "MetricValue": "1",
                        "MetricNamespace": "test/404s",
                        "MetricName": "test404Count"
                    }
```

```
                ]
            }
        },
        "404Alarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": "The number of 404s is greater than 2 over 2
minutes",
                "MetricName": "test404Count",
                "Namespace": "test/404s",
                "Statistic": "Sum",
                "Period": "60",
                "EvaluationPeriods": "2",
                "Threshold": "2",
                "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
                "ComparisonOperator": "GreaterThanThreshold"
            }
        },
        "AlarmNotificationTopic": {
            "Type": "AWS::SNS::Topic",
            "Properties": {
                "Subscription": [{"Endpoint": {"Ref": "OperatorEmail"}, "Protocol":
"email"}]
            }
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The instance ID of the web server",
            "Value": {"Ref": "WebServerHost"}
        },
        "WebsiteURL": {
            "Value": {"Fn::Sub": "http://${WebServerHost.PublicDnsName}"},
            "Description": "URL for the web server"
        },
        "PublicIP": {
            "Description": "Public IP address of the web server",
            "Value": {"Fn::GetAtt": ["WebServerHost","PublicIp"]}
        },
        "CloudWatchLogGroupName": {
            "Description": "The name of the CloudWatch log group",
            "Value": {"Ref": "LogGroup"}
        }
    }
```

```
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: >-
  Sample template that sets up and configures CloudWatch Logs on Windows 2012R2
 instance.
Parameters:
  KeyPair:
    Description: Name of an existing EC2 KeyPair to enable RDP access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  RDPLocation:
    Description: The IP address range that can be used to RDP to the EC2 instances
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  OperatorEmail:
    Description: Email address to notify when CloudWatch alarms are triggered (404
 errors)
    Type: String
Resources:
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80 and RDP access via port 3389
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '80'
          ToPort: '80'
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: '3389'
          ToPort: '3389'
          CidrIp: !Ref RDPLocation
  LogRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
      Path: /
      Policies:
        - PolicyName: LogRolePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'logs:Create*'
                  - 'logs:PutLogEvents'
                  - 's3:GetObject'
                Resource:
                  - 'arn:aws:logs:*:*:*'
                  - 'arn:aws:s3:::*'
  LogRoleInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles:
        - !Ref LogRole
  WebServerHost:
    Type: AWS::EC2::Instance
    CreationPolicy:
      ResourceSignal:
        Timeout: PT15M
    Metadata:
      'AWS::CloudFormation::Init':
        configSets:
          config:
            - 00-ConfigureCWLogs
            - 01-InstallWebServer
            - 02-ConfigureApplication
            - 03-Finalize
        00-ConfigureCWLogs:
```

```
          files:
            'C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch
\AWS.EC2.Windows.CloudWatch.json':
              content: !Sub |
                {
                  "EngineConfiguration": {
                      "Components": [
                          {
                              "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                              "Id": "ApplicationEventLog",
                              "Parameters": {
                                  "Levels": "7",
                                  "LogName": "Application"
                              }
                          },
                          {
                              "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                              "Id": "SystemEventLog",
                              "Parameters": {
                                  "Levels": "7",
                                  "LogName": "System"
                              }
                          },
                          {
                              "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                              "Id": "SecurityEventLog",
                              "Parameters": {
                                  "Levels": "7",
                                  "LogName": "Security"
                              }
                          },
                          {
                              "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                              "Id": "EC2ConfigLog",
                              "Parameters": {
                                  "CultureName": "en-US",
                                  "Encoding": "ASCII",
                                  "Filter": "EC2ConfigLog.txt",
                                  "LogDirectoryPath": "C:\\Program Files\\Amazon\
\Ec2ConfigService\\Logs",
```

```
                                "TimeZoneKind": "UTC",
                                "TimestampFormat": "yyyy-MM-ddTHH:mm:ss.fffZ:"
                            }
                        },
                        {
                            "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                            "Id": "CfnInitLog",
                            "Parameters": {
                                "CultureName": "en-US",
                                "Encoding": "ASCII",
                                "Filter": "cfn-init.log",
                                "LogDirectoryPath": "C:\\cfn\\log",
                                "TimeZoneKind": "Local",
                                "TimestampFormat": "yyyy-MM-dd HH:mm:ss,fff"
                            }
                        },
                        {
                            "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                            "Id": "IISLogs",
                            "Parameters": {
                                "CultureName": "en-US",
                                "Encoding": "UTF-8",
                                "Filter": "",
                                "LineCount": "3",
                                "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\
\W3SVC1",
                                "TimeZoneKind": "UTC",
                                "TimestampFormat": "yyyy-MM-dd HH:mm:ss"
                            }
                        },
                        {
                            "FullName":
 "AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent,AWS.E
                            "Id": "MemoryPerformanceCounter",
                            "Parameters": {
                                "CategoryName": "Memory",
                                "CounterName": "Available MBytes",
                                "DimensionName": "",
                                "DimensionValue": "",
                                "InstanceName": "",
                                "MetricName": "Memory",
                                "Unit": "Megabytes"
```

```
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchApplicationEventLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/ApplicationEventLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchSystemEventLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/SystemEventLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchSecurityEventLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/SecurityEventLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchEC2ConfigLog",
                            "Parameters": {
                                "AccessKey": "",
```

```
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/EC2ConfigLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchCfnInitLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/CfnInitLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchIISLogs",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/IISLogs",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatch",
                            "Parameters": {
                                "AccessKey": "",
                                "NameSpace": "Windows/Default",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        }
                    ],
                    "Flows": {
                        "Flows": [
```

```
                            "ApplicationEventLog,CloudWatchApplicationEventLog",
                            "SystemEventLog,CloudWatchSystemEventLog",
                            "SecurityEventLog,CloudWatchSecurityEventLog",
                            "EC2ConfigLog,CloudWatchEC2ConfigLog",
                            "CfnInitLog,CloudWatchCfnInitLog",
                            "IISLogs,CloudWatchIISLogs",
                            "MemoryPerformanceCounter,CloudWatch"
                        ]
                    },
                    "PollInterval": "00:00:05"
                },
                "IsEnabled": true
            }
        commands:
          0-enableSSM:
            command: >-
              powershell.exe -Command "Set-Service -Name AmazonSSMAgent
              -StartupType Automatic"
            waitAfterCompletion: '0'
          1-restartSSM:
            command: powershell.exe -Command "Restart-Service AmazonSSMAgent "
            waitAfterCompletion: '30'
      01-InstallWebServer:
        commands:
          01_install_webserver:
            command: >-
              powershell.exe -Command "Install-WindowsFeature Web-Server
              -IncludeAllSubFeature"
            waitAfterCompletion: '0'
      02-ConfigureApplication:
        files:
          'c:\Inetpub\wwwroot\index.htm':
            content: >-
              <html> <head> <title>Test Application Page</title> </head>
              <body> <h1>Congratulations !! Your IIS server is
              configured.</h1> </body> </html>
      03-Finalize:
        commands:
          00_signal_success:
            command: !Sub >-
              cfn-signal.exe -e 0 --resource WebServerHost --stack
              ${AWS::StackName} --region ${AWS::Region}
            waitAfterCompletion: '0'
    Properties:
```

```
      KeyName: !Ref KeyPair
      ImageId: "{{resolve:ssm:/aws/service/ami-windows-latest/Windows_Server-2012-
R2_RTM-English-64Bit-Base}}"
      InstanceType: t2.xlarge
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      IamInstanceProfile: !Ref LogRoleInstanceProfile
      UserData: !Base64
        'Fn::Sub': >
          <script>

          wmic product where "description='Amazon SSM Agent' " uninstall

          wmic product where "description='aws-cfn-bootstrap' " uninstall

          start /wait c:\\Windows\\system32\\msiexec /passive /qn /i
          https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-win64-
latest.msi

          powershell.exe -Command "iwr
          https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/
AmazonSSMAgentSetup.exe
          -UseBasicParsing -OutFile C:\\AmazonSSMAgentSetup.exe"

          start /wait C:\\AmazonSSMAgentSetup.exe /install /quiet

          cfn-init.exe -v -c config -s ${AWS::StackName} --resource
          WebServerHost --region ${AWS::Region}

          </script>
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      RetentionInDays: 7
  404MetricFilter:
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref LogGroup
      FilterPattern: >-
        [timestamps, serverip, method, uri, query, port, dash, clientip,
        useragent, status_code = 404, ...]
      MetricTransformations:
        - MetricValue: '1'
          MetricNamespace: test/404s
```

```
          MetricName: test404Count
  404Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: The number of 404s is greater than 2 over 2 minutes
      MetricName: test404Count
      Namespace: test/404s
      Statistic: Sum
      Period: '60'
      EvaluationPeriods: '2'
      Threshold: '2'
      AlarmActions:
        - !Ref AlarmNotificationTopic
      ComparisonOperator: GreaterThanThreshold
  AlarmNotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !Ref OperatorEmail
          Protocol: email
Outputs:
  InstanceId:
    Description: The instance ID of the web server
    Value: !Ref WebServerHost
  WebsiteURL:
    Value: !Sub 'http://${WebServerHost.PublicDnsName}'
    Description: URL for the web server
  PublicIP:
    Description: Public IP address of the web server
    Value: !GetAtt
      - WebServerHost
      - PublicIp
  CloudWatchLogGroupName:
    Description: The name of the CloudWatch log group
    Value: !Ref LogGroup
```

另请参阅

有关 CloudWatch Logs 资源的更多信息，请参阅 AWS::Logs::LogGroup 或 AWs::Logs::MetricFilter。

Amazon DynamoDB 模板代码段

主题

- 具有 Amazon DynamoDB 表的 Application Auto Scaling
- 另请参阅

具有 Amazon DynamoDB 表的 Application Auto Scaling

此示例为 AWS::DynamoDB::Table 资源设置 Application Auto Scaling。此模板定义扩展表的 TargetTrackingScaling 吞吐量的 WriteCapacityUnits 扩展策略。

JSON

```
{
    "Resources": {
        "DDBTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "AttributeDefinitions": [
                    {
                        "AttributeName": "ArtistId",
                        "AttributeType": "S"
                    },
                    {
                        "AttributeName": "Concert",
                        "AttributeType": "S"
                    },
                    {
                        "AttributeName": "TicketSales",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "ArtistId",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "Concert",
                        "KeyType": "RANGE"
                    }
                ],
                "GlobalSecondaryIndexes": [
                    {
                        "IndexName": "GSI",
```

```
                        "KeySchema": [
                            {
                                "AttributeName": "TicketSales",
                                "KeyType": "HASH"
                            }
                        ],
                        "Projection": {
                            "ProjectionType": "KEYS_ONLY"
                        },
                        "ProvisionedThroughput": {
                            "ReadCapacityUnits": 5,
                            "WriteCapacityUnits": 5
                        }
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 5
                }
            }
        },
        "WriteCapacityScalableTarget": {
            "Type": "AWS::ApplicationAutoScaling::ScalableTarget",
            "Properties": {
                "MaxCapacity": 15,
                "MinCapacity": 5,
                "ResourceId": {
                    "Fn::Join": [
                        "/",
                        [
                            "table",
                            {
                                "Ref": "DDBTable"
                            }
                        ]
                    ]
                },
                "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable" },
                "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
                "ServiceNamespace": "dynamodb"
            }
        },
```

```
        "WriteScalingPolicy": {
            "Type": "AWS::ApplicationAutoScaling::ScalingPolicy",
            "Properties": {
                "PolicyName": "WriteAutoScalingPolicy",
                "PolicyType": "TargetTrackingScaling",
                "ScalingTargetId": {
                    "Ref": "WriteCapacityScalableTarget"
                },
                "TargetTrackingScalingPolicyConfiguration": {
                    "TargetValue": 50,
                    "ScaleInCooldown": 60,
                    "ScaleOutCooldown": 60,
                    "PredefinedMetricSpecification": {
                        "PredefinedMetricType": "DynamoDBWriteCapacityUtilization"
                    }
                }
            }
        }
    }
}
```

YAML

```
Resources:
  DDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: "ArtistId"
          AttributeType: "S"
        - AttributeName: "Concert"
          AttributeType: "S"
        - AttributeName: "TicketSales"
          AttributeType: "S"
      KeySchema:
        - AttributeName: "ArtistId"
          KeyType: "HASH"
        - AttributeName: "Concert"
          KeyType: "RANGE"
      GlobalSecondaryIndexes:
        - IndexName: "GSI"
          KeySchema:
            - AttributeName: "TicketSales"
```

```
              KeyType: "HASH"
          Projection:
            ProjectionType: "KEYS_ONLY"
          ProvisionedThroughput:
            ReadCapacityUnits: 5
            WriteCapacityUnits: 5
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
  WriteCapacityScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 15
      MinCapacity: 5
      ResourceId: !Join
        - /
        - - table
          - !Ref DDBTable
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable'
      ScalableDimension: dynamodb:table:WriteCapacityUnits
      ServiceNamespace: dynamodb
  WriteScalingPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: WriteAutoScalingPolicy
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref WriteCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 50.0
        ScaleInCooldown: 60
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: DynamoDBWriteCapacityUtilization
```

另请参阅

有关更多信息，请参阅 Amazon 数据库博客文章 How to use Amazon CloudFormation to configure auto scaling for DynamoDB tables and indexes。

有关 DynamoDB 资源的更多信息，请参阅 AWS::DynamoDB::Table。

Amazon EC2 CloudFormation 模板代码段

Amazon EC2 在 Amazon Web Services 云 中提供了可扩展的计算容量。您可以使用 Amazon EC2 启动所需数量的虚拟服务器，配置安全性和联网以及管理存储。这些虚拟服务器（称为实例）可以运行各种操作系统和应用程序，并且可以根据您的具体要求进行自定义。您可以利用 Amazon EC2 进行扩展或缩减，以便应对需求变化或使用高峰。

您可以使用 CloudFormation 模板定义和预置 Amazon EC2 实例作为基础设施的一部分。模板可以让您轻松地以可重复且一致的方式管理和自动部署 Amazon EC2 资源。

以下示例模板代码段描述了 Amazon EC2 的 CloudFormation 资源或组件。这些代码段旨在集成到模板中，不打算独立运行。

代码段类别

- 使用 CloudFormation 配置 Amazon EC2 实例
- 使用 CloudFormation 创建启动模板
- 使用 CloudFormation 管理安全组
- 分配弹性 IP 地址并将其与 CloudFormation 关联
- 使用 CloudFormation 配置 Amazon VPC 资源

使用 CloudFormation 配置 Amazon EC2 实例

以下代码段演示了如何使用 CloudFormation 配置 Amazon EC2 实例。

代码段类别

- Amazon EC2 的常规配置
- 指定实例的块设备映射

Amazon EC2 的常规配置

以下代码段演示了如何使用 CloudFormation 对 Amazon EC2 实例进行常规配置。

示例代码段

- 在指定的可用区中创建 Amazon EC2 实例
- 使用 EBS 卷和用户数据配置标记的 Amazon EC2 实例

- 在用户数据中定义 DynamoDB 表名用于启动 Amazon EC2 实例
- 使用 DeletionPolicy 创建 Amazon EBS 卷

在指定的可用区中创建 Amazon EC2 实例

以下代码段使用 AWS::EC2::Instance 资源在指定的可用区中创建 Amazon EC2 实例。可用区的代码由其区域代码后跟一个字母标识符组成。您可以在单个可用区中启动实例。

JSON

```
"Ec2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "AvailabilityZone": "aa-example-1a",
        "ImageId": "ami-1234567890abcdef0"
    }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
```

使用 EBS 卷和用户数据配置标记的 Amazon EC2 实例

以下代码段使用标记、EBS 卷和用户数据创建了一个 Amazon EC2 实例。该实例使用 AWS::EC2::Instance 资源。必须在同一个模板中定义 AWS::EC2::SecurityGroup 资源、AWS::SNS::Topic 资源和 AWS::EC2::Volume 资源。必须在模板的 `Parameters` 部分中定义 `KeyName`。

标签可帮助您按用途、拥有者或环境等首选项对 Amazon 资源进行分类。用户数据允许在启动期间向实例预置自定义脚本或数据。这些数据有助于实现任务自动化、软件配置、软件包安装以及初始化期间对实例执行的其他操作。

有关标记资源的更多信息，请参阅《Amazon EC2 用户指南》中的标记 Amazon EC2 资源。

有关用户数据的信息，请参阅《Amazon EC2 用户指南》中的使用实例元数据管理 EC2 实例。

JSON

```
"Ec2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "KeyName": { "Ref": "KeyName" },
    "SecurityGroups": [ { "Ref": "Ec2SecurityGroup" } ],
    "UserData": {
      "Fn::Base64": {
        "Fn::Join": [ ":", [
            "PORT=80",
            "TOPIC=",
            { "Ref": "MySNSTopic" }
          ]
        ]
      }
    },
    "InstanceType": "aa.size",
    "AvailabilityZone": "aa-example-1a",
    "ImageId": "ami-1234567890abcdef0",
    "Volumes": [
      {
        "VolumeId": { "Ref": "MyVolumeResource" },
        "Device": "/dev/sdk"
      }
    ],
    "Tags": [ { "Key": "Name", "Value": "MyTag" } ]
  }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    KeyName: !Ref KeyName
    SecurityGroups:
      - !Ref Ec2SecurityGroup
    UserData:
      Fn::Base64:
        Fn::Join:
          - ":"
          - - "PORT=80"
```

```
            - "TOPIC="
            - !Ref MySNSTopic
    InstanceType: aa.size
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
    Volumes:
      - VolumeId: !Ref MyVolumeResource
        Device: "/dev/sdk"
    Tags:
      - Key: Name
        Value: MyTag
```

在用户数据中定义 DynamoDB 表名用于启动 Amazon EC2 实例

以下代码段创建了一个 Amazon EC2 实例，并在用户数据中定义了一个 DynamoDB 表名，以便在启动时传递给实例。该实例使用 AWS::EC2::Instance 资源。您可以在用户数据中定义参数或动态值，以便在启动时传递 EC2 实例。

有关用户数据的更多信息，请参阅《Amazon EC2 用户指南》中的使用实例元数据管理 EC2 实例。

JSON

```
"Ec2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "UserData": {
            "Fn::Base64": {
                "Fn::Join": [
                    "",
                    [
                        "TableName=",
                        {
                            "Ref": "DynamoDBTableName"
                        }
                    ]
                ]
            }
        },
        "AvailabilityZone": "aa-example-1a",
        "ImageId": "ami-1234567890abcdef0"
    }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    UserData:
      Fn::Base64:
        Fn::Join:
          - ''
          - - 'TableName='
            - Ref: DynamoDBTableName
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
```

使用 **DeletionPolicy** 创建 Amazon EBS 卷

以下代码段使用 Amazon EC2 AWS::EC2::Volume 资源创建 Amazon EBS 卷。您可以使用 Size 或 SnapshotID 属性来定义卷，但不能同时使用两者。DeletionPolicy 属性设置为在删除堆栈时创建卷的快照。

有关 DeletionPolicy 属性的更多信息，请参阅 DeletionPolicy 属性。

有关创建 Amazon EBS 卷的更多信息，请参阅创建 Amazon EBS 卷。

JSON

此代码段会创建具有指定大小的 Amazon EBS 卷。大小虽设置为 10，但可按需加以调整。AWS::EC2::Volume 资源允许指定大小或快照 ID，但不允许同时指定这两者。

```
"MyEBSVolume": {
    "Type": "AWS::EC2::Volume",
    "Properties": {
        "Size": "10",
        "AvailabilityZone": {
            "Ref": "AvailabilityZone"
        }
    },
    "DeletionPolicy": "Snapshot"
}
```

此代码段使用提供的快照 ID 创建一个 Amazon EBS 卷。AWS::EC2::Volume 资源允许指定大小或快照 ID，但不允许同时指定这两者。

```
"MyEBSVolume": {
    "Type": "AWS::EC2::Volume",
    "Properties": {
        "SnapshotId" : "snap-1234567890abcdef0",
        "AvailabilityZone": {
            "Ref": "AvailabilityZone"
        }
    },
    "DeletionPolicy": "Snapshot"
}
```

YAML

此代码段会创建具有指定大小的 Amazon EBS 卷。大小虽设置为 10，但可按需加以调整。AWS::EC2::Volume 资源允许指定大小或快照 ID，但不允许同时指定这两者。

```
MyEBSVolume:
  Type: AWS::EC2::Volume
  Properties:
    Size: 10
    AvailabilityZone:
      Ref: AvailabilityZone
  DeletionPolicy: Snapshot
```

此代码段使用提供的快照 ID 创建一个 Amazon EBS 卷。AWS::EC2::Volume 资源允许指定大小或快照 ID，但不允许同时指定这两者。

```
MyEBSVolume:
  Type: AWS::EC2::Volume
  Properties:
    SnapshotId: snap-1234567890abcdef0
    AvailabilityZone:
      Ref: AvailabilityZone
  DeletionPolicy: Snapshot
```

指定实例的块设备映射

块设备映射定义了挂载到某个实例的块设备（包含实例存储卷和 EBS 卷）。您可以在创建 AMI 时指定块设备映射，以便让从该 AMI 启动的所有实例都可使用该映射。另外，您可以在启动实例时指定块设备映射，这样该映射会覆盖您从启动实例的 AMI 中指定的块设备映射。

您可以使用以下模板代码段中 AWS::EC2::Instance 资源的 BlockDeviceMappings 属性，为 EBS 或实例存储卷指定块设备映射。

有关块设备映射的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 实例上卷的块设备映射。

场景

- 为两个 EBS 卷指定块设备映射
- 为实例存储卷指定块设备映射

为两个 EBS 卷指定块设备映射

JSON

```
"Ec2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
      "KeyName": { "Ref": "KeyName" },
      "InstanceType": { "Ref": "InstanceType" },
      "SecurityGroups": [{ "Ref": "Ec2SecurityGroup" }],
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": { "VolumeSize": "50" }
        },
        {
          "DeviceName": "/dev/sdm",
          "Ebs": { "VolumeSize": "100" }
        }
      ]
    }
  }
}
```

YAML

```
EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
```

```
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      KeyName: !Ref KeyName
      InstanceType: !Ref InstanceType
      SecurityGroups:
        - !Ref Ec2SecurityGroup
      BlockDeviceMappings:
        -
          DeviceName: /dev/sda1
          Ebs:
            VolumeSize: 50
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeSize: 100
```

为实例存储卷指定块设备映射

JSON

```
"Ec2Instance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
    "KeyName" : { "Ref" : "KeyName" },
    "InstanceType": { "Ref": "InstanceType" },
    "SecurityGroups" : [{ "Ref" : "Ec2SecurityGroup" }],
    "BlockDeviceMappings" : [
      {
        "DeviceName"  : "/dev/sdc",
        "VirtualName" : "ephemeral0"
      }
    ]
  }
}
```

YAML

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
```

```
    ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-
gp2}}'
    KeyName: !Ref KeyName
    InstanceType: !Ref InstanceType
    SecurityGroups:
      - !Ref Ec2SecurityGroup
    BlockDeviceMappings:
      - DeviceName: /dev/sdc
        VirtualName: ephemeral0
```

使用 CloudFormation 创建启动模板

本节提供了使用 CloudFormation 创建 Amazon EC2 启动模板的示例。启动模板允许您创建用于在 Amazon 中配置和预置 Amazon EC2 实例的模板。通过启动模板，您可以存储启动参数，而无需在每次启动实例时都指定这些参数。有关更多示例，请参阅 AWS::EC2::LaunchTemplate 资源中的示例部分。

有关启动模板的更多信息，请参阅《Amazon EC2 用户指南》中的在 Amazon EC2 启动模板中存储实例启动参数。

有关创建启动模板用于自动扩缩组的信息，请参阅《Amazon EC2 Auto Scaling User Guide》中的 Auto Scaling launch templates。

代码段类别

- 创建指定安全组、标签、用户数据和 IAM 角色的启动模板

创建指定安全组、标签、用户数据和 IAM 角色的启动模板

此代码段显示 AWS::EC2::LaunchTemplate 资源，其中包含启动实例的配置信息。您可以为 ImageId、InstanceType、SecurityGroups、UserData 和 TagSpecifications 属性指定值。SecurityGroups 属性会指定一个现有 EC2 安全组和一个新安全组。Ref 函数会获取在堆栈模板中其他位置声明的 AWS::EC2::SecurityGroup 资源 myNewEC2SecurityGroup 的 ID。

启动模板包括自定义用户数据的部分。在本节中，您可以传入实例启动时运行的配置任务和脚本。在此示例中，用户数据安装 Amazon Systems Manager 代理并启动该代理。

启动模板还包含一个 IAM 角色，该角色允许在实例上运行的应用程序代表您执行操作。此示例显示启动模板的 AWS::IAM::Role 资源，其使用 IamInstanceProfile 属性来指定 IAM 角色。Ref 函数会获取 AWS::IAM::InstanceProfile 资源 myInstanceProfile 的名称。要配置 IAM 角色的权限，请指定 ManagedPolicyArns 属性的值。

JSON

```
{
  "Resources":{
    "myLaunchTemplate":{
      "Type":"AWS::EC2::LaunchTemplate",
      "Properties":{
        "LaunchTemplateName":{ "Fn::Sub": "${AWS::StackName}-launch-template" },
        "LaunchTemplateData":{
          "ImageId":"ami-02354e95b3example",
          "InstanceType":"t3.micro",
          "IamInstanceProfile":{
            "Name":{
              "Ref":"myInstanceProfile"
            }
          },
          "SecurityGroupIds":[
            {
              "Ref":"myNewEC2SecurityGroup"
            },
            "sg-083cd3bfb8example"
          ],
          "UserData":{
            "Fn::Base64":{
              "Fn::Join": [
                "", [
                  "#!/bin/bash\n",
                  "cd /tmp\n",
                  "yum install -y https://s3.amazonaws.com/ec2-downloads-windows/
SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm\n",
                  "systemctl enable amazon-ssm-agent\n",
                  "systemctl start amazon-ssm-agent\n"
                ]
              ]
            }
          },
          "TagSpecifications":[
            {
              "ResourceType":"instance",
              "Tags":[
                {
                  "Key":"environment",
                  "Value":"development"
                }
```

```
            ]
          },
          {
            "ResourceType":"volume",
            "Tags":[
              {
                "Key":"environment",
                "Value":"development"
              }
            ]
          }
        ]
      }
    }
  },
  "myInstanceRole":{
    "Type":"AWS::IAM::Role",
    "Properties":{
      "RoleName":"InstanceRole",
      "AssumeRolePolicyDocument":{
        "Version": "2012-10-17",
        "Statement":[
          {
            "Effect":"Allow",
            "Principal":{
              "Service":[
                "ec2.amazonaws.com"
              ]
            },
            "Action":[
              "sts:AssumeRole"
            ]
          }
        ]
      },
      "ManagedPolicyArns":[
        "arn:aws:iam::aws:policy/myCustomerManagedPolicy"
      ]
    }
  },
  "myInstanceProfile":{
    "Type":"AWS::IAM::InstanceProfile",
    "Properties":{
      "Path":"/",
```

```
        "Roles":[
          {
            "Ref":"myInstanceRole"
          }
        ]
      }
    }
  }
}
```

YAML

```
---
Resources:
  myLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: ami-02354e95b3example
        InstanceType: t3.micro
        IamInstanceProfile:
          Name: !Ref myInstanceProfile
        SecurityGroupIds:
        - !Ref myNewEC2SecurityGroup
        - sg-083cd3bfb8example
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            cd /tmp
            yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/
latest/linux_amd64/amazon-ssm-agent.rpm
            systemctl enable amazon-ssm-agent
            systemctl start amazon-ssm-agent
        TagSpecifications:
        - ResourceType: instance
          Tags:
          - Key: environment
            Value: development
        - ResourceType: volume
          Tags:
          - Key: environment
            Value: development
```

```
  myInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: InstanceRole
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: 'Allow'
          Principal:
            Service:
            - 'ec2.amazonaws.com'
          Action:
          - 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/myCustomerManagedPolicy'
  myInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: '/'
      Roles:
      - !Ref myInstanceRole
```

使用 CloudFormation 管理安全组

以下代码段演示了如何使用 CloudFormation 管理安全组和 Amazon EC2 实例，从而控制对 Amazon 资源的访问权限。

代码段类别

- 将 Amazon EC2 实例与安全组关联
- 使用入口规则创建安全组
- 使用安全组入口规则创建弹性负载均衡器

将 Amazon EC2 实例与安全组关联

以下示例代码段演示了如何使用 CloudFormation 将 Amazon EC2 实例与默认的 Amazon VPC 安全组关联起来。

示例代码段

- 将 Amazon EC2 实例与默认 VPC 安全组关联
- 创建带有附加卷和安全组的 Amazon EC2 实例

将 Amazon EC2 实例与默认 VPC 安全组关联

以下代码段创建了一个 Amazon VPC、一个 VPC 内的子网和一个 Amazon EC2 实例。VPC 是使用 AWS::EC2::VPC 资源创建的。VPC 的 IP 地址范围在较大的模板中定义并由 MyVPCCIDRRange 参数引用。

子网是使用 AWS::EC2:: Subnet 资源在 VPC 中创建的。子网与 VPC 关联，后者被引用为 MyVPC。

使用 AWS::EC2::Instance 资源在 VPC 和子网内启动 EC2 实例。此资源指定用于启动实例的亚马逊机器映像（AMI）、实例将在其中运行的子网以及要与该实例关联的安全组。ImageId 使用 Systems Manager 参数动态检索最新的 Amazon Linux 2 AMI。

安全组 ID 是使用 Fn::GetAtt 函数获取的，该函数从 MyVPC 资源中检索默认安全组。

该实例位于代码段中定义的 MySubnet 资源中。

使用 CloudFormation 创建 VPC 时，Amazon 会在 VPC 中自动创建默认资源，包括默认安全组。但是，当您在 CloudFormation 模板中定义 VPC 时，可能无法在创建模板时访问这些默认资源的 ID。要访问和使用模板中指定的默认资源，您可以使用内置函数，例如 Fn::GetAtt。此函数允许您使用由 CloudFormation 自动创建的默认资源。

JSON

```
"MyVPC": {
    "Type": "AWS::EC2::VPC",
    "Properties": {
        "CidrBlock": {
            "Ref": "MyVPCCIDRRange"
        },
        "EnableDnsSupport": false,
        "EnableDnsHostnames": false,
        "InstanceTenancy": "default"
    }
},
"MySubnet": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
        "CidrBlock": {
            "Ref": "MyVPCCIDRRange"
        },
        "VpcId": {
            "Ref": "MyVPC"
        }
```

```
    }
},
"MyInstance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
        "SecurityGroupIds": [
            {
                "Fn::GetAtt": [
                    "MyVPC",
                    "DefaultSecurityGroup"
                ]
            }
        ],
        "SubnetId": {
            "Ref": "MySubnet"
        }
    }
}
```

YAML

```
MyVPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock:
      Ref: MyVPCCIDRRange
    EnableDnsSupport: false
    EnableDnsHostnames: false
    InstanceTenancy: default
MySubnet:
  Type: AWS::EC2::Subnet
  Properties:
    CidrBlock:
      Ref: MyVPCCIDRRange
    VpcId:
      Ref: MyVPC
MyInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-
gp2}}'
```

```
    SecurityGroupIds:
      - Fn::GetAtt:
          - MyVPC
          - DefaultSecurityGroup
    SubnetId:
      Ref: MySubnet
```

创建带有附加卷和安全组的 Amazon EC2 实例

以下代码段使用从指定 AMI 启动的 AWS::EC2::Instance 资源创建 Amazon EC2 实例。该实例与一个安全组关联，该安全组使用 AWS::EC2::SecurityGroup 资源允许来自指定 IP 地址端口 22 上的传入 SSH 流量。它使用 AWS::EC2::Volume 资源创建一个 100GB 的 Amazon EBS 卷。根据 GetAtt 函数的指定，这个卷在与实例相同的可用区中创建，并安装到 /dev/sdh 设备上的实例。

有关创建 Amazon EBS 卷的更多信息，请参阅创建 Amazon EBS 卷。

JSON

```
"Ec2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "SecurityGroups": [
            {
                "Ref": "InstanceSecurityGroup"
            }
        ],
        "ImageId": "ami-1234567890abcdef0"
    }
},
"InstanceSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Enable SSH access via port 22",
        "SecurityGroupIngress": [
            {
                "IpProtocol": "tcp",
                "FromPort": "22",
                "ToPort": "22",
                "CidrIp": "192.0.2.0/24"
            }
        ]
    }
},
```

```
"NewVolume": {
    "Type": "AWS::EC2::Volume",
    "Properties": {
        "Size": "100",
        "AvailabilityZone": {
            "Fn::GetAtt": [
                "Ec2Instance",
                "AvailabilityZone"
            ]
        }
    }
},
"MountPoint": {
    "Type": "AWS::EC2::VolumeAttachment",
    "Properties": {
        "InstanceId": {
            "Ref": "Ec2Instance"
        },
        "VolumeId": {
            "Ref": "NewVolume"
        },
        "Device": "/dev/sdh"
    }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    SecurityGroups:
      - !Ref InstanceSecurityGroup
    ImageId: ami-1234567890abcdef0
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable SSH access via port 22
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 192.0.2.0/24
NewVolume:
```

```
  Type: AWS::EC2::Volume
  Properties:
    Size: 100
    AvailabilityZone: !GetAtt [Ec2Instance, AvailabilityZone]
MountPoint:
  Type: AWS::EC2::VolumeAttachment
  Properties:
    InstanceId: !Ref Ec2Instance
    VolumeId: !Ref NewVolume
    Device: /dev/sdh
```

使用入口规则创建安全组

以下示例代码段演示了如何使用 CloudFormation 配置具有特定入口规则的安全组。

代码段

- 使用入口规则创建安全组以允许 SSH 和 HTTP 访问
- 使用入口规则创建安全组以允许来自指定 CIDR 范围的 HTTP 和 SSH 访问
- 使用入口规则创建交叉引用安全组

使用入口规则创建安全组以允许 SSH 和 HTTP 访问

以下代码段描述了两个使用 AWS::EC2::SecurityGroup 资源的安全组入口规则。第一个入口规则允许从名为 MyAdminSecurityGroup 的现有安全组访问 SSH（端口 22），该安全组由账号为 1111-2222-3333 的 Amazon 账户所有。第二个入口规则允许从名为 MySecurityGroupCreatedInCFN 的另一个安全组访问 HTTP（端口 80），该安全组是在同一个模板中创建的。Ref 函数用于引用在同一模板中创建的安全组的逻辑名称。

在第一个入口规则中，必须为 SourceSecurityGroupName 和 SourceSecurityGroupOwnerId 属性添加一个值。在第二个入口规则中，MySecurityGroupCreatedInCFNTemplate 引用了在同一个模板中创建的另一个安全组。验证逻辑名称 MySecurityGroupCreatedInCFNTemplate 是否与您在较大模板中指定的安全组资源的实际逻辑名称相匹配。

有关安全组的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 实例的 Amazon EC2 安全组。

JSON

```
"SecurityGroup": {
```

```
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Allow connections from specified source security group",
        "SecurityGroupIngress": [
            {
                "IpProtocol": "tcp",
                "FromPort": "22",
                "ToPort": "22",
                "SourceSecurityGroupName": "MyAdminSecurityGroup",
                "SourceSecurityGroupOwnerId": "1111-2222-3333"
            },
            {
                "IpProtocol": "tcp",
                "FromPort": "80",
                "ToPort": "80",
                "SourceSecurityGroupName": {
                    "Ref": "MySecurityGroupCreatedInCFNTemplate"
                }
            }
        ]
    }
}
```

YAML

```
SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow connections from specified source security group
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: '22'
        ToPort: '22'
        SourceSecurityGroupName: MyAdminSecurityGroup
        SourceSecurityGroupOwnerId: '1111-2222-3333'
      - IpProtocol: tcp
        FromPort: '80'
        ToPort: '80'
        SourceSecurityGroupName:
          Ref: MySecurityGroupCreatedInCFNTemplate
```

使用入口规则创建安全组以允许来自指定 CIDR 范围的 HTTP 和 SSH 访问

以下代码段为具有两个入站规则的 Amazon EC2 实例创建了一个安全组。入站规则允许来自指定 CIDR 范围指定端口上的传入 TCP 流量。AWS::EC2::SecurityGroup 资源用于指定规则。您必须为每个规则指定一个协议。对于 TCP，您必须指定端口或端口范围。如果未指定源安全组或 CIDR 范围，堆栈虽会成功启动，但规则不会应用于安全组。

有关安全组的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 实例的 Amazon EC2 安全组。

JSON

```
"ServerSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "Allow connections from specified CIDR ranges",
    "SecurityGroupIngress": [
      {
        "IpProtocol": "tcp",
        "FromPort": "80",
        "ToPort": "80",
        "CidrIp": "192.0.2.0/24"
      },
      {
        "IpProtocol": "tcp",
        "FromPort": "22",
        "ToPort": "22",
        "CidrIp": "192.0.2.0/24"
      }
    ]
  }
}
```

YAML

```
ServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow connections from specified CIDR ranges
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
```

```
        ToPort: 80
        CidrIp: 192.0.2.0/24
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 192.0.2.0/24
```

使用入口规则创建交叉引用安全组

以下代码段使用 AWS::EC2::SecurityGroup 资源创建两个 Amazon EC2 安全组：SGroup1 和 SGroup2。允许在两个安全组之间通信的入口规则是使用 AWS::EC2::SecurityGroupIngress 资源创建的。SGroup1Ingress 为 SGroup1 建立了一个入口规则，允许来自源安全组 SGroup2 端口 80 上的传入 TCP 流量。SGroup2Ingress 为 SGroup2 建立了一个入口规则，允许来自源安全组 SGroup1 端口 80 上的传入 TCP 流量。

JSON

```
"SGroup1": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "EC2 instance access"
    }
},
"SGroup2": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "EC2 instance access"
    }
},
"SGroup1Ingress": {
    "Type": "AWS::EC2::SecurityGroupIngress",
    "Properties": {
        "GroupName": {
            "Ref": "SGroup1"
        },
        "IpProtocol": "tcp",
        "ToPort": "80",
        "FromPort": "80",
        "SourceSecurityGroupName": {
            "Ref": "SGroup2"
        }
    }
},
```

```
"SGroup2Ingress": {
    "Type": "AWS::EC2::SecurityGroupIngress",
    "Properties": {
        "GroupName": {
            "Ref": "SGroup2"
        },
        "IpProtocol": "tcp",
        "ToPort": "80",
        "FromPort": "80",
        "SourceSecurityGroupName": {
            "Ref": "SGroup1"
        }
    }
}
```

YAML

```
SGroup1:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: EC2 Instance access
SGroup2:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: EC2 Instance access
SGroup1Ingress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupName: !Ref SGroup1
    IpProtocol: tcp
    ToPort: 80
    FromPort: 80
    SourceSecurityGroupName: !Ref SGroup2
SGroup2Ingress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupName: !Ref SGroup2
    IpProtocol: tcp
    ToPort: 80
    FromPort: 80
    SourceSecurityGroupName: !Ref SGroup1
```

使用安全组入口规则创建弹性负载均衡器

以下模板在指定的可用区中创建了 AWS::ElasticLoadBalancing::LoadBalancer 资源。AWS::ElasticLoadBalancing::LoadBalancer 资源配置为在端口 80 上侦听 HTTP 流量，并将请求定向到端口 80 上的实例。弹性负载均衡器负责对实例间传入的 HTTP 流量进行负载均衡。

此外，此模板将生成一个与负载均衡器关联的 AWS::EC2::SecurityGroup 资源。此安全组是使用单个入口规则（描述为 `ELB ingress group`）创建的，该规则允许端口 80 上的传入 TCP 流量。此入口规则的来源是使用从负载均衡器资源检索属性的 `Fn::GetAtt` 函数定义的。`SourceSecurityGroupOwnerId` 使用 `Fn::GetAtt` 来获取负载均衡器源安全组的 `OwnerAlias`。`SourceSecurityGroupName` 使用 `Fn::Getatt` 来获取 ELB 源安全组的 `GroupName`。

此设置可确保 ELB 和实例之间的安全通信。

有关负载均衡的更多信息，请参阅 Elastic Load Balancing 用户指南。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MyELB": {
            "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
            "Properties": {
                "AvailabilityZones": [
                    "aa-example-1a"
                ],
                "Listeners": [
                    {
                        "LoadBalancerPort": "80",
                        "InstancePort": "80",
                        "Protocol": "HTTP"
                    }
                ]
            }
        },
        "MyELBIngressGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "ELB ingress group",
                "SecurityGroupIngress": [
```

```
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "SourceSecurityGroupOwnerId": {
                            "Fn::GetAtt": [
                                "MyELB",
                                "SourceSecurityGroup.OwnerAlias"
                            ]
                        },
                        "SourceSecurityGroupName": {
                            "Fn::GetAtt": [
                                "MyELB",
                                "SourceSecurityGroup.GroupName"
                            ]
                        }
                    }
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyELB:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      AvailabilityZones:
        - aa-example-1a
      Listeners:
        - LoadBalancerPort: '80'
          InstancePort: '80'
          Protocol: HTTP
  MyELBIngressGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: ELB ingress group
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '80'
```

```
          ToPort: '80'
          SourceSecurityGroupOwnerId:
            Fn::GetAtt:
              - MyELB
              - SourceSecurityGroup.OwnerAlias
          SourceSecurityGroupName:
            Fn::GetAtt:
              - MyELB
              - SourceSecurityGroup.GroupName
```

分配弹性 IP 地址并将其与 CloudFormation 关联

以下模板代码段展示了与 Amazon EC2 中的弹性 IP 地址（EIP）相关的示例。这些示例涵盖您实例的 EIP 分配、关联和管理。

示例代码段

- 分配弹性 IP 地址并将其与 Amazon EC2 实例关联
- 通过指定 IP 地址将弹性 IP 地址与 Amazon EC2 实例关联
- 通过指定 IP 地址的分配 ID 将弹性 IP 地址与 Amazon EC2 实例关联

分配弹性 IP 地址并将其与 Amazon EC2 实例关联

以下代码段分配一个 Amazon EC2 弹性 IP（EIP）地址并将其与使用 AWS::EC2::EIP 资源的 Amazon EC2 实例关联。使用自带 IP 地址（BYOIP），您可以从 Amazon 拥有的地址池或从公共 IPv4 地址范围（您引入到 Amazon 中以与 Amazon 资源一起使用的地址范围）创建的地址池中分配 EIP 地址。在本示例中，EIP 是从 Amazon 拥有的地址池中分配的。

有关弹性 IP 地址的更多信息，请参阅《Amazon EC2 用户指南》中的弹性 IP 地址。

JSON

```
"ElasticIP": {
    "Type": "AWS::EC2::EIP",
    "Properties": {
        "InstanceId": {
            "Ref": "Ec2Instance"
        }
    }
}
```

YAML

```
ElasticIP:
  Type: AWS::EC2::EIP
  Properties:
    InstanceId: !Ref EC2Instance
```

通过指定 IP 地址将弹性 IP 地址与 Amazon EC2 实例关联

以下代码段使用 AWS::EC2::EIPAssociation 资源将现有的 Amazon EC2 弹性 IP 地址与 EC2 实例关联。您必须先分配一个弹性 IP 地址，才能在自己的账户中使用。弹性 IP 地址可以与单个实例关联。

JSON

```
"IPAssoc": {
  "Type": "AWS::EC2::EIPAssociation",
  "Properties": {
    "InstanceId": {
      "Ref": "Ec2Instance"
    },
    "EIP": "192.0.2.0"
  }
}
```

YAML

```
IPAssoc:
  Type: AWS::EC2::EIPAssociation
  Properties:
    InstanceId: !Ref EC2Instance
    EIP: 192.0.2.0
```

通过指定 IP 地址的分配 ID 将弹性 IP 地址与 Amazon EC2 实例关联

以下代码段使用 AWS::EC2::EIPAssociation 资源指定分配 ID，将现有的弹性 IP 地址与 Amazon EC2 实例关联。分配弹性 IP 地址时，即会为弹性 IP 地址分配一个分配 ID。

JSON

```
"IPAssoc": {
    "Type": "AWS::EC2::EIPAssociation",
```

```
    "Properties": {
        "InstanceId": {
            "Ref": "Ec2Instance"
        },
        "AllocationId": "eipalloc-1234567890abcdef0"
    }
}
```

YAML

```
IPAssoc:
  Type: AWS::EC2::EIPAssociation
  Properties:
    InstanceId: !Ref EC2Instance
    AllocationId: eipalloc-1234567890abcdef0
```

使用 CloudFormation 配置 Amazon VPC 资源

本节提供使用 CloudFormation 配置 Amazon VPC 资源的示例。VPC 允许您在 Amazon 中创建一个虚拟网络，这些代码段展示了如何配置 VPC 各个方面来满足自己的网络要求。

示例代码段

- 在 VPC 中启用 IPv6 仅出口互联网访问
- 弹性网络接口 (ENI) 模板代码段

在 VPC 中启用 IPv6 仅出口互联网访问

仅出口互联网网关允许 VPC 中的实例访问互联网并阻止互联网上的资源与实例通信。以下代码段支持在 VPC 中启用 IPv6 仅出口互联网访问。它使用 AWS::EC2::VPC 资源创建 IPv4 地址范围为 10.0.0/16 的 VPC。将路由表与使用 AWS::EC2::RouteTable 资源的 VPC 资源关联起来。路由表会管理 VPC 内实例的路由。AWS::EC2::EgressOnlyInternetGateway 用于创建仅出口互联网网关，为来自 VPC 中实例的出站流量启用 IPv6 通信，同时阻止入站流量。AWS::EC2::Route 资源用于在路由表中创建 IPv6 路由，该路由会将所有出站 IPv6 流量 (::/0) 定向到仅出口互联网网关。

有关仅出口互联网网关的更多信息，请参阅《Amazon VPC 用户指南》中的使用仅出口互联网网关允许出站 IPv6 流量。

JSON

```
"DefaultIpv6Route": {
```

```
    "Type": "AWS::EC2::Route",
    "Properties": {
        "DestinationIpv6CidrBlock": "::/0",
        "EgressOnlyInternetGatewayId": {
            "Ref": "EgressOnlyInternetGateway"
        },
        "RouteTableId": {
            "Ref": "RouteTable"
        }
    }
},
"EgressOnlyInternetGateway": {
    "Type": "AWS::EC2::EgressOnlyInternetGateway",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        }
    }
},
"RouteTable": {
    "Type": "AWS::EC2::RouteTable",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        }
    }
},
"VPC": {
    "Type": "AWS::EC2::VPC",
    "Properties": {
        "CidrBlock": "10.0.0.0/16"
    }
}
```

YAML

```
DefaultIpv6Route:
  Type: AWS::EC2::Route
  Properties:
    DestinationIpv6CidrBlock: "::/0"
    EgressOnlyInternetGatewayId:
      Ref: "EgressOnlyInternetGateway"
    RouteTableId:
```

```
      Ref: "RouteTable"
EgressOnlyInternetGateway:
  Type: AWS::EC2::EgressOnlyInternetGateway
  Properties:
    VpcId:
      Ref: "VPC"
RouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId:
      Ref: "VPC"
VPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: "10.0.0.0/16"
```

弹性网络接口 (ENI) 模板代码段

创建带有附加弹性网络接口（ENI）的 Amazon EC2 实例

以下示例代码段使用 AWS::EC2::Instance 资源在指定的 Amazon VPC 和子网中创建 Amazon EC2 实例。它将两个网络接口（ENI）连接到实例，通过附加的 ENI 将弹性 IP 地址关联到实例，并针对 SSH 和 HTTP 访问配置安全组。创建实例时，用户数据作为启动配置的一部分提供给实例。用户数据包括以 base64 格式编码的脚本，可确保将其传递到实例。当实例启动时，脚本会作为引导过程的一部分自动运行。它会安装 ec2-net-utils、配置网络接口并启动 HTTP 服务。

为了根据所选区域确定合适的亚马逊机器映像（AMI），该代码段使用了一个可在 RegionMap 映射中查找值的 Fn::FindInMap 函数。此映射必须在较大的模板中定义。这两个网络接口是使用 AWS::EC2::NetworkInterface 资源创建的。使用分配给 vpc 域的 AWS::EC2::EIP 资源指定弹性 IP 地址。这些弹性 IP 地址与使用 AWS::EC2::EIPAssociation 资源的网络接口关联。

Outputs 部分定义了您想在堆栈创建后访问的值或资源。在此代码段中，定义的输出是 InstancePublicIp，表示堆栈创建的 EC2 实例的公有 IP 地址。您可以使用 Amazon CloudFormation 控制台的输出选项卡或使用 describe-stacks 命令来检索此输出。

有关弹性网络接口的更多信息，请参阅弹性网络接口。

JSON

```
"Resources": {
    "ControlPortAddress": {
        "Type": "AWS::EC2::EIP",
```

```
        "Properties": {
            "Domain": "vpc"
        }
    },
    "AssociateControlPort": {
        "Type": "AWS::EC2::EIPAssociation",
        "Properties": {
            "AllocationId": {
                "Fn::GetAtt": [
                    "ControlPortAddress",
                    "AllocationId"
                ]
            },
            "NetworkInterfaceId": {
                "Ref": "controlXface"
            }
        }
    },
    "WebPortAddress": {
        "Type": "AWS::EC2::EIP",
        "Properties": {
            "Domain": "vpc"
        }
    },
    "AssociateWebPort": {
        "Type": "AWS::EC2::EIPAssociation",
        "Properties": {
            "AllocationId": {
                "Fn::GetAtt": [
                    "WebPortAddress",
                    "AllocationId"
                ]
            },
            "NetworkInterfaceId": {
                "Ref": "webXface"
            }
        }
    },
    "SSHSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "VpcId": {
                "Ref": "VpcId"
            },
```

```
            "GroupDescription": "Enable SSH access via port 22",
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 22,
                    "IpProtocol": "tcp",
                    "ToPort": 22
                }
            ]
        }
    },
    "WebSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "VpcId": {
                "Ref": "VpcId"
            },
            "GroupDescription": "Enable HTTP access via user-defined port",
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 80,
                    "IpProtocol": "tcp",
                    "ToPort": 80
                }
            ]
        }
    },
    "controlXface": {
        "Type": "AWS::EC2::NetworkInterface",
        "Properties": {
            "SubnetId": {
                "Ref": "SubnetId"
            },
            "Description": "Interface for controlling traffic such as SSH",
            "GroupSet": [
                {
                    "Fn::GetAtt": [
                        "SSHSecurityGroup",
                        "GroupId"
                    ]
                }
            ],
            "SourceDestCheck": true,
```

```
            "Tags": [
                {
                    "Key": "Network",
                    "Value": "Control"
                }
            ]
        }
    },
    "webXface": {
        "Type": "AWS::EC2::NetworkInterface",
        "Properties": {
            "SubnetId": {
                "Ref": "SubnetId"
            },
            "Description": "Interface for web traffic",
            "GroupSet": [
                {
                    "Fn::GetAtt": [
                        "WebSecurityGroup",
                        "GroupId"
                    ]
                }
            ],
            "SourceDestCheck": true,
            "Tags": [
                {
                    "Key": "Network",
                    "Value": "Web"
                }
            ]
        }
    },
    "Ec2Instance": {
        "Type": "AWS::EC2::Instance",
        "Properties": {
            "ImageId": {
                "Fn::FindInMap": [
                    "RegionMap",
                    {
                        "Ref": "AWS::Region"
                    },
                    "AMI"
                ]
            },
```

```
            "KeyName": {
                "Ref": "KeyName"
            },
            "NetworkInterfaces": [
                {
                    "NetworkInterfaceId": {
                        "Ref": "controlXface"
                    },
                    "DeviceIndex": "0"
                },
                {
                    "NetworkInterfaceId": {
                        "Ref": "webXface"
                    },
                    "DeviceIndex": "1"
                }
            ],
            "Tags": [
                {
                    "Key": "Role",
                    "Value": "Test Instance"
                }
            ],
            "UserData": {
                "Fn::Base64": {
                    "Fn::Sub": "#!/bin/bash -xe\nyum install ec2-net-utils -y\nec2ifup
 eth1\nservice httpd start\n"
                }
            }
        }
    }
},
"Outputs": {
    "InstancePublicIp": {
        "Description": "Public IP Address of the EC2 Instance",
        "Value": {
            "Fn::GetAtt": [
                "Ec2Instance",
                "PublicIp"
            ]
        }
    }
}
```

YAML

```
Resources:
  ControlPortAddress:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
  AssociateControlPort:
    Type: AWS::EC2::EIPAssociation
    Properties:
      AllocationId:
        Fn::GetAtt:
          - ControlPortAddress
          - AllocationId
      NetworkInterfaceId:
        Ref: controlXface
  WebPortAddress:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
  AssociateWebPort:
    Type: AWS::EC2::EIPAssociation
    Properties:
      AllocationId:
        Fn::GetAtt:
          - WebPortAddress
          - AllocationId
      NetworkInterfaceId:
        Ref: webXface
  SSHSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
        Ref: VpcId
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          FromPort: 22
          IpProtocol: tcp
          ToPort: 22
  WebSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
```

```
      Ref: VpcId
    GroupDescription: Enable HTTP access via user-defined port
    SecurityGroupIngress:
      - CidrIp: 0.0.0.0/0
        FromPort: 80
        IpProtocol: tcp
        ToPort: 80
controlXface:
  Type: AWS::EC2::NetworkInterface
  Properties:
    SubnetId:
      Ref: SubnetId
    Description: Interface for controlling traffic such as SSH
    GroupSet:
      - Fn::GetAtt:
          - SSHSecurityGroup
          - GroupId
    SourceDestCheck: true
    Tags:
      - Key: Network
        Value: Control
webXface:
  Type: AWS::EC2::NetworkInterface
  Properties:
    SubnetId:
      Ref: SubnetId
    Description: Interface for web traffic
    GroupSet:
      - Fn::GetAtt:
          - WebSecurityGroup
          - GroupId
    SourceDestCheck: true
    Tags:
      - Key: Network
        Value: Web
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId:
      Fn::FindInMap:
        - RegionMap
        - Ref: AWS::Region
        - AMI
    KeyName:
```

```
        Ref: KeyName
      NetworkInterfaces:
        - NetworkInterfaceId:
            Ref: controlXface
          DeviceIndex: "0"
        - NetworkInterfaceId:
            Ref: webXface
          DeviceIndex: "1"
      Tags:
        - Key: Role
          Value: Test Instance
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum install ec2-net-utils -y
          ec2ifup eth1
          service httpd start
Outputs:
  InstancePublicIp:
    Description: Public IP Address of the EC2 Instance
    Value:
      Fn::GetAtt:
        - Ec2Instance
        - PublicIp
```

Amazon Elastic Container Service 示例模板

Amazon Elastic Container Service (Amazon ECS) 是一项容器管理服务，可让您轻松地在 Amazon Elastic Compute Cloud (Amazon EC2) 实例集群上运行、停止和管理 Docker 容器。

使用 AL2023 Amazon ECS-Optimized-AMI 创建集群

定义一个集群，该集群使用在 Amazon EC2 上启动 AL2023 实例的容量提供程序。

> **Important**
>
> 有关最新的 AMI ID，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 优化的 AMI。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "EC2 ECS cluster that starts out empty, with no EC2 instances yet. An
 ECS capacity provider automatically launches more EC2 instances as required on the fly
 when you request ECS to launch services or standalone tasks.",
  "Parameters": {
      "InstanceType": {
          "Type": "String",
          "Description": "EC2 instance type",
          "Default": "t2.medium",
          "AllowedValues": [
              "t1.micro",
              "t2.2xlarge",
              "t2.large",
              "t2.medium",
              "t2.micro",
              "t2.nano",
              "t2.small",
              "t2.xlarge",
              "t3.2xlarge",
              "t3.large",
              "t3.medium",
              "t3.micro",
              "t3.nano",
              "t3.small",
              "t3.xlarge"
          ]
      },
      "DesiredCapacity": {
          "Type": "Number",
          "Default": "0",
          "Description": "Number of EC2 instances to launch in your ECS cluster."
      },
      "MaxSize": {
          "Type": "Number",
          "Default": "100",
          "Description": "Maximum number of EC2 instances that can be launched in your
 ECS cluster."
      },
      "ECSAMI": {
          "Description": "The Amazon Machine Image ID used for the cluster",
          "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
```

```
          "Default": "/aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/
image_id"
      },
      "VpcId": {
          "Type": "AWS::EC2::VPC::Id",
          "Description": "VPC ID where the ECS cluster is launched",
          "Default": "vpc-1234567890abcdef0"
      },
      "SubnetIds": {
          "Type": "List<AWS::EC2::Subnet::Id>",
          "Description": "List of subnet IDs where the EC2 instances will be launched",
          "Default": "subnet-021345abcdef67890"
      }
  },
  "Resources": {
      "ECSCluster": {
          "Type": "AWS::ECS::Cluster",
          "Properties": {
              "ClusterSettings": [
                  {
                      "Name": "containerInsights",
                      "Value": "enabled"
                  }
              ]
          }
      },
      "ECSAutoScalingGroup": {
          "Type": "AWS::AutoScaling::AutoScalingGroup",
          "DependsOn": [
              "ECSCluster",
              "EC2Role"
          ],
          "Properties": {
              "VPCZoneIdentifier": {
                  "Ref": "SubnetIds"
              },
              "LaunchTemplate": {
                  "LaunchTemplateId": {
                      "Ref": "ContainerInstances"
                  },
                  "Version": {
                      "Fn::GetAtt": [
                          "ContainerInstances",
                          "LatestVersionNumber"
```

```
                    ]
                }
            },
            "MinSize": 0,
            "MaxSize": {
                "Ref": "MaxSize"
            },
            "DesiredCapacity": {
                "Ref": "DesiredCapacity"
            },
            "NewInstancesProtectedFromScaleIn": true
        },
        "UpdatePolicy": {
            "AutoScalingReplacingUpdate": {
                "WillReplace": "true"
            }
        }
    },
    "ContainerInstances": {
        "Type": "AWS::EC2::LaunchTemplate",
        "Properties": {
            "LaunchTemplateName": "asg-launch-template",
            "LaunchTemplateData": {
                "ImageId": {
                    "Ref": "ECSAMI"
                },
                "InstanceType": {
                    "Ref": "InstanceType"
                },
                "IamInstanceProfile": {
                    "Name": {
                        "Ref": "EC2InstanceProfile"
                    }
                },
                "SecurityGroupIds": [
                    {
                        "Ref": "ContainerHostSecurityGroup"
                    }
                ],
                "UserData": {
                    "Fn::Base64": {
                        "Fn::Sub": "#!/bin/bash -xe\n echo ECS_CLUSTER=${ECSCluster}
>> /etc/ecs/ecs.config\n yum install -y aws-cfn-bootstrap\n /opt/aws/bin/cfn-init -
```

```
v --stack ${AWS::StackId} --resource ContainerInstances --configsets full_install --
region ${AWS::Region} &\n"
                    }
                },
                "MetadataOptions": {
                    "HttpEndpoint": "enabled",
                    "HttpTokens": "required"
                }
            }
        }
    },
    "EC2InstanceProfile": {
        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [
                {
                    "Ref": "EC2Role"
                }
            ]
        }
    },
    "CapacityProvider": {
        "Type": "AWS::ECS::CapacityProvider",
        "Properties": {
            "AutoScalingGroupProvider": {
                "AutoScalingGroupArn": {
                    "Ref": "ECSAutoScalingGroup"
                },
                "ManagedScaling": {
                    "InstanceWarmupPeriod": 60,
                    "MinimumScalingStepSize": 1,
                    "MaximumScalingStepSize": 100,
                    "Status": "ENABLED",
                    "TargetCapacity": 100
                },
                "ManagedTerminationProtection": "ENABLED"
            }
        }
    },
    "CapacityProviderAssociation": {
        "Type": "AWS::ECS::ClusterCapacityProviderAssociations",
        "Properties": {
            "CapacityProviders": [
```

```
                {
                    "Ref": "CapacityProvider"
                }
            ],
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DefaultCapacityProviderStrategy": [
                {
                    "Base": 0,
                    "CapacityProvider": {
                        "Ref": "CapacityProvider"
                    },
                    "Weight": 1
                }
            ]
        }
    },
    "ContainerHostSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Access to the EC2 hosts that run containers",
            "VpcId": {
                "Ref": "VpcId"
            }
        }
    },
    "EC2Role": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ec2.amazonaws.com"
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
```

```
            },
            "Path": "/",
            "ManagedPolicyArns": [
                "arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role",
                "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
            ]
        }
    },
    "ECSTaskExecutionRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ecs-tasks.amazonaws.com"
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ],
                        "Condition": {
                            "ArnLike": {
                                "aws:SourceArn": {
                                    "Fn::Sub": "arn:${AWS::Partition}:ecs:
${AWS::Region}:${AWS::AccountId}:*"
                                }
                            },
                            "StringEquals": {
                                "aws:SourceAccount": {
                                      "Fn::Sub": "${AWS::AccountId}"
                                  }
                            }
                        }
                    }
                ]
            },
            "Path": "/",
            "ManagedPolicyArns": [
                "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
```

```
                ]
            }
        }
    },
    "Outputs": {
        "ClusterName": {
            "Description": "The ECS cluster into which to launch resources",
            "Value": "ECSCluster"
        },
        "ECSTaskExecutionRole": {
            "Description": "The role used to start up a task",
            "Value": "ECSTaskExecutionRole"
        },
        "CapacityProvider": {
            "Description": "The cluster capacity provider that the service should use to
 request capacity when it wants to start up a task",
            "Value": "CapacityProvider"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: EC2 ECS cluster that starts out empty, with no EC2 instances yet.
  An ECS capacity provider automatically launches more EC2 instances as required
  on the fly when you request ECS to launch services or standalone tasks.
Parameters:
  InstanceType:
    Type: String
    Description: EC2 instance type
    Default: "t2.medium"
    AllowedValues:
      - t1.micro
      - t2.2xlarge
      - t2.large
      - t2.medium
      - t2.micro
      - t2.nano
      - t2.small
      - t2.xlarge
      - t3.2xlarge
```

```
      - t3.large
      - t3.medium
      - t3.micro
      - t3.nano
      - t3.small
      - t3.xlarge
  DesiredCapacity:
    Type: Number
    Default: "0"
    Description: Number of EC2 instances to launch in your ECS cluster.
  MaxSize:
    Type: Number
    Default: "100"
    Description: Maximum number of EC2 instances that can be launched in your ECS
 cluster.
  ECSAMI:
    Description: The Amazon Machine Image ID used for the cluster
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID where the ECS cluster is launched
    Default: vpc-1234567890abcdef0
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
    Description: List of subnet IDs where the EC2 instances will be launched
    Default: "subnet-021345abcdef67890"
Resources:
# This is authorizes ECS to manage resources on your
  # account on your behalf. This role is likely already created on your account
  # ECSRole:
  #  Type: AWS::IAM::ServiceLinkedRole
  #  Properties:
  #    AWSServiceName: 'ecs.amazonaws.com'

   # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterSettings:
        - Name: containerInsights
          Value: enabled

  # Autoscaling group. This launches the actual EC2 instances that will register
```

```
  # themselves as members of the cluster, and run the docker containers.
  ECSAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    DependsOn:
      # This is to ensure that the ASG gets deleted first before these
    # resources, when it comes to stack teardown.
      - ECSCluster
      - EC2Role
    Properties:
      VPCZoneIdentifier:
        Ref: SubnetIds
      LaunchTemplate:
        LaunchTemplateId: !Ref ContainerInstances
        Version: !GetAtt ContainerInstances.LatestVersionNumber
      MinSize: 0
      MaxSize:
        Ref: MaxSize
      DesiredCapacity:
        Ref: DesiredCapacity
      NewInstancesProtectedFromScaleIn: true
    UpdatePolicy:
      AutoScalingReplacingUpdate:
        WillReplace: "true"
  # The config for each instance that is added to the cluster
  ContainerInstances:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: "asg-launch-template"
      LaunchTemplateData:
        ImageId:
          Ref: ECSAMI
        InstanceType:
          Ref: InstanceType
        IamInstanceProfile:
          Name: !Ref EC2InstanceProfile
        SecurityGroupIds:
          - !Ref ContainerHostSecurityGroup
        # This injected configuration file is how the EC2 instance
      # knows which ECS cluster on your AWS account it should be joining
        UserData:
          Fn::Base64: !Sub |
           #!/bin/bash -xe
            echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
            yum install -y aws-cfn-bootstrap
```

```
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource
ContainerInstances --configsets full_install --region ${AWS::Region} &
        # Disable IMDSv1, and require IMDSv2
       MetadataOptions:
         HttpEndpoint: enabled
         HttpTokens: required
 EC2InstanceProfile:
   Type: AWS::IAM::InstanceProfile
   Properties:
     Path: /
     Roles:
     - !Ref EC2Role
 # Create an ECS capacity provider to attach the ASG to the ECS cluster
 # so that it autoscales as we launch more containers
 CapacityProvider:
   Type: AWS::ECS::CapacityProvider
   Properties:
     AutoScalingGroupProvider:
       AutoScalingGroupArn: !Ref ECSAutoScalingGroup
       ManagedScaling:
         InstanceWarmupPeriod: 60
         MinimumScalingStepSize: 1
         MaximumScalingStepSize: 100
         Status: ENABLED
         # Percentage of cluster reservation to try to maintain
         TargetCapacity: 100
       ManagedTerminationProtection: ENABLED
  # Create a cluster capacity provider assocation so that the cluster
 # will use the capacity provider
 CapacityProviderAssociation:
   Type: AWS::ECS::ClusterCapacityProviderAssociations
   Properties:
     CapacityProviders:
       - !Ref CapacityProvider
     Cluster: !Ref ECSCluster
     DefaultCapacityProviderStrategy:
       - Base: 0
         CapacityProvider: !Ref CapacityProvider
         Weight: 1
 # A security group for the EC2 hosts that will run the containers.
 # This can be used to limit incoming traffic to or outgoing traffic
 # from the container's host EC2 instance.
 ContainerHostSecurityGroup:
   Type: AWS::EC2::SecurityGroup
```

```
    Properties:
      GroupDescription: Access to the EC2 hosts that run containers
      VpcId:
        Ref: VpcId
  # Role for the EC2 hosts. This allows the ECS agent on the EC2 hosts
  # to communciate with the ECS control plane, as well as download the docker
  # images from ECR to run on your host.
  EC2Role:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
      # See reference: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/
security-iam-awsmanpol.html#security-iam-awsmanpol-AmazonEC2ContainerServiceforEC2Role
        - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
      # This managed policy allows us to connect to the instance using SSM
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
  # This is a role which is used within Fargate to allow the Fargate agent
  # to download images, and upload logs.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ecs-tasks.amazonaws.com
            Action:
              - sts:AssumeRole
            Condition:
              ArnLike:
                aws:SourceArn: !Sub arn:${AWS::Partition}:ecs:${AWS::Region}:
${AWS::AccountId}:*
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
```

```
      Path: /
      # This role enables all features of ECS. See reference:
    # https://docs.aws.amazon.com/AmazonECS/latest/developerguide/security-iam-
awsmanpol.html#security-iam-awsmanpol-AmazonECSTaskExecutionRolePolicy
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ClusterName:
    Description: The ECS cluster into which to launch resources
    Value: ECSCluster
  ECSTaskExecutionRole:
    Description: The role used to start up a task
    Value: ECSTaskExecutionRole
  CapacityProvider:
    Description: The cluster capacity provider that the service should use to
      request capacity when it wants to start up a task
    Value: CapacityProvider
```

部署服务

以下模板定义了一项服务，该服务使用容量提供程序请求运行 AL2023 容量。容器将在 AL2023 实例上线时随之启动：

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "An example service that deploys in AWS VPC networking mode on EC2
 capacity. Service uses a capacity provider to request EC2 instances to run on. Service
 runs with networking in private subnets, but still accessible to the internet via a
 load balancer hosted in public subnets.",
  "Parameters": {
      "VpcId": {
          "Type": "String",
          "Description": "The VPC that the service is running inside of"
      },
      "PublicSubnetIds": {
          "Type": "List<AWS::EC2::Subnet::Id>",
          "Description": "List of public subnet ID's to put the load balancer in"
      },
      "PrivateSubnetIds": {
          "Type": "List<AWS::EC2::Subnet::Id>",
          "Description": "List of private subnet ID's that the AWS VPC tasks are in"
```

```
      },
      "ClusterName": {
          "Type": "String",
          "Description": "The name of the ECS cluster into which to launch capacity."
      },
      "ECSTaskExecutionRole": {
          "Type": "String",
          "Description": "The role used to start up an ECS task"
      },
      "CapacityProvider": {
          "Type": "String",
          "Description": "The cluster capacity provider that the service should use to
 request capacity when it wants to start up a task"
      },
      "ServiceName": {
          "Type": "String",
          "Default": "web",
          "Description": "A name for the service"
      },
      "ImageUrl": {
          "Type": "String",
          "Default": "public.ecr.aws/docker/library/nginx:latest",
          "Description": "The url of a docker image that contains the application
 process that will handle the traffic for this service"
      },
      "ContainerCpu": {
          "Type": "Number",
          "Default": 256,
          "Description": "How much CPU to give the container. 1024 is 1 CPU"
      },
      "ContainerMemory": {
          "Type": "Number",
          "Default": 512,
          "Description": "How much memory in megabytes to give the container"
      },
      "ContainerPort": {
          "Type": "Number",
          "Default": 80,
          "Description": "What port that the application expects traffic on"
      },
      "DesiredCount": {
          "Type": "Number",
          "Default": 2,
          "Description": "How many copies of the service task to run"
```

```
        }
    },
    "Resources": {
        "TaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "Family": {
                    "Ref": "ServiceName"
                },
                "Cpu": {
                    "Ref": "ContainerCpu"
                },
                "Memory": {
                    "Ref": "ContainerMemory"
                },
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "EC2"
                ],
                "ExecutionRoleArn": {
                    "Ref": "ECSTaskExecutionRole"
                },
                "ContainerDefinitions": [
                    {
                        "Name": {
                            "Ref": "ServiceName"
                        },
                        "Cpu": {
                            "Ref": "ContainerCpu"
                        },
                        "Memory": {
                            "Ref": "ContainerMemory"
                        },
                        "Image": {
                            "Ref": "ImageUrl"
                        },
                        "PortMappings": [
                            {
                                "ContainerPort": {
                                    "Ref": "ContainerPort"
                                },
                                "HostPort": {
                                    "Ref": "ContainerPort"
                                }
```

```
                    }
                ],
                "LogConfiguration": {
                    "LogDriver": "awslogs",
                    "Options": {
                        "mode": "non-blocking",
                        "max-buffer-size": "25m",
                        "awslogs-group": {
                            "Ref": "LogGroup"
                        },
                        "awslogs-region": {
                            "Ref": "AWS::Region"
                        },
                        "awslogs-stream-prefix": {
                            "Ref": "ServiceName"
                        }
                    }
                }
            }
        ]
    }
},
"Service": {
    "Type": "AWS::ECS::Service",
    "DependsOn": "PublicLoadBalancerListener",
    "Properties": {
        "ServiceName": {
            "Ref": "ServiceName"
        },
        "Cluster": {
            "Ref": "ClusterName"
        },
        "PlacementStrategies": [
            {
                "Field": "attribute:ecs.availability-zone",
                "Type": "spread"
            },
            {
                "Field": "cpu",
                "Type": "binpack"
            }
        ],
        "CapacityProviderStrategy": [
            {
```

```
                    "Base": 0,
                    "CapacityProvider": {
                        "Ref": "CapacityProvider"
                    },
                    "Weight": 1
                }
            ],
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "SecurityGroups": [
                        {
                            "Ref": "ServiceSecurityGroup"
                        }
                    ],
                    "Subnets": {
                        "Ref": "PrivateSubnetIds"
                    }
                }
            },
            "DeploymentConfiguration": {
                "MaximumPercent": 200,
                "MinimumHealthyPercent": 75
            },
            "DesiredCount": {
                "Ref": "DesiredCount"
            },
            "TaskDefinition": {
                "Ref": "TaskDefinition"
            },
            "LoadBalancers": [
                {
                    "ContainerName": {
                        "Ref": "ServiceName"
                    },
                    "ContainerPort": {
                        "Ref": "ContainerPort"
                    },
                    "TargetGroupArn": {
                        "Ref": "ServiceTargetGroup"
                    }
                }
            ]
        }
    },
```

```
        "ServiceSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Security group for service",
                "VpcId": {
                    "Ref": "VpcId"
                }
            }
        },
        "ServiceTargetGroup": {
            "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
            "Properties": {
                "HealthCheckIntervalSeconds": 6,
                "HealthCheckPath": "/",
                "HealthCheckProtocol": "HTTP",
                "HealthCheckTimeoutSeconds": 5,
                "HealthyThresholdCount": 2,
                "TargetType": "ip",
                "Port": {
                    "Ref": "ContainerPort"
                },
                "Protocol": "HTTP",
                "UnhealthyThresholdCount": 10,
                "VpcId": {
                    "Ref": "VpcId"
                },
                "TargetGroupAttributes": [
                    {
                        "Key": "deregistration_delay.timeout_seconds",
                        "Value": 0
                    }
                ]
            }
        },
        "PublicLoadBalancerSG": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Access to the public facing load balancer",
                "VpcId": {
                    "Ref": "VpcId"
                },
                "SecurityGroupIngress": [
                    {
                        "CidrIp": "0.0.0.0/0",
```

```
                    "IpProtocol": -1
                }
            ]
        }
    },
    "PublicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internet-facing",
            "LoadBalancerAttributes": [
                {
                    "Key": "idle_timeout.timeout_seconds",
                    "Value": "30"
                }
            ],
            "Subnets": {
                "Ref": "PublicSubnetIds"
            },
            "SecurityGroups": [
                {
                    "Ref": "PublicLoadBalancerSG"
                }
            ]
        }
    },
    "PublicLoadBalancerListener": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
        "Properties": {
            "DefaultActions": [
                {
                    "Type": "forward",
                    "ForwardConfig": {
                        "TargetGroups": [
                            {
                                "TargetGroupArn": {
                                    "Ref": "ServiceTargetGroup"
                                },
                                "Weight": 100
                            }
                        ]
                    }
                }
            ],
            "LoadBalancerArn": {
```

```
                    "Ref": "PublicLoadBalancer"
                },
                "Port": 80,
                "Protocol": "HTTP"
            }
        },
        "ServiceIngressfromLoadBalancer": {
            "Type": "AWS::EC2::SecurityGroupIngress",
            "Properties": {
                "Description": "Ingress from the public ALB",
                "GroupId": {
                    "Ref": "ServiceSecurityGroup"
                },
                "IpProtocol": -1,
                "SourceSecurityGroupId": {
                    "Ref": "PublicLoadBalancerSG"
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >-
  An example service that deploys in AWS VPC networking mode on EC2 capacity.
  Service uses a capacity provider to request EC2 instances to run on. Service
  runs with networking in private subnets, but still accessible to the internet
  via a load balancer hosted in public subnets.
Parameters:
  VpcId:
    Type: String
    Description: The VPC that the service is running inside of
  PublicSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of public subnet ID's to put the load balancer in
  PrivateSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of private subnet ID's that the AWS VPC tasks are in
```

```
  ClusterName:
    Type: String
    Description: The name of the ECS cluster into which to launch capacity.
  ECSTaskExecutionRole:
    Type: String
    Description: The role used to start up an ECS task
  CapacityProvider:
    Type: String
    Description: >-
      The cluster capacity provider that the service should use to request
      capacity when it wants to start up a task
  ServiceName:
    Type: String
    Default: web
    Description: A name for the service
  ImageUrl:
    Type: String
    Default: 'public.ecr.aws/docker/library/nginx:latest'
    Description: >-
      The url of a docker image that contains the application process that will
      handle the traffic for this service
  ContainerCpu:
    Type: Number
    Default: 256
    Description: How much CPU to give the container. 1024 is 1 CPU
  ContainerMemory:
    Type: Number
    Default: 512
    Description: How much memory in megabytes to give the container
  ContainerPort:
    Type: Number
    Default: 80
    Description: What port that the application expects traffic on
  DesiredCount:
    Type: Number
    Default: 2
    Description: How many copies of the service task to run
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: !Ref ServiceName
      Cpu: !Ref ContainerCpu
      Memory: !Ref ContainerMemory
```

```
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - EC2
      ExecutionRoleArn: !Ref ECSTaskExecutionRole
      ContainerDefinitions:
        - Name: !Ref ServiceName
          Cpu: !Ref ContainerCpu
          Memory: !Ref ContainerMemory
          Image: !Ref ImageUrl
          PortMappings:
            - ContainerPort: !Ref ContainerPort
              HostPort: !Ref ContainerPort
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-group: !Ref LogGroup
              awslogs-region: !Ref AWS::Region
              awslogs-stream-prefix: !Ref ServiceName
  Service:
    Type: AWS::ECS::Service
    DependsOn: PublicLoadBalancerListener
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ClusterName
      PlacementStrategies:
        - Field: 'attribute:ecs.availability-zone'
          Type: spread
        - Field: cpu
          Type: binpack
      CapacityProviderStrategy:
        - Base: 0
          CapacityProvider: !Ref CapacityProvider
          Weight: 1
      NetworkConfiguration:
        AwsvpcConfiguration:
          SecurityGroups:
            - !Ref ServiceSecurityGroup
          Subnets: !Ref PrivateSubnetIds
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: !Ref DesiredCount
```

```
      TaskDefinition: !Ref TaskDefinition
      LoadBalancers:
        - ContainerName: !Ref ServiceName
          ContainerPort: !Ref ContainerPort
          TargetGroupArn: !Ref ServiceTargetGroup
  ServiceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for service
      VpcId: !Ref VpcId
  ServiceTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
      TargetType: ip
      Port: !Ref ContainerPort
      Protocol: HTTP
      UnhealthyThresholdCount: 10
      VpcId: !Ref VpcId
      TargetGroupAttributes:
        - Key: deregistration_delay.timeout_seconds
          Value: 0
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: -1
  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
        - Key: idle_timeout.timeout_seconds
          Value: '30'
      Subnets: !Ref PublicSubnetIds
      SecurityGroups:
        - !Ref PublicLoadBalancerSG
```

```
  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          ForwardConfig:
            TargetGroups:
              - TargetGroupArn: !Ref ServiceTargetGroup
                Weight: 100
      LoadBalancerArn: !Ref PublicLoadBalancer
      Port: 80
      Protocol: HTTP
  ServiceIngressfromLoadBalancer:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId: !Ref ServiceSecurityGroup
      IpProtocol: -1
      SourceSecurityGroupId: !Ref PublicLoadBalancerSG
  LogGroup:
    Type: AWS::Logs::LogGroup
```

Amazon Elastic File System 示例模板

Amazon Elastic File System (Amazon EFS) 是适用于 Amazon Elastic Compute Cloud (Amazon EC2) 实例的文件存储服务。借助 Amazon EFS，您的应用程序在需要存储时可获得存储，因为存储容量会在您添加和删除文件时自动增加和缩小。

以下示例模板部署 EC2 实例（位于自动扩缩组中），这些实例与 Amazon EFS 文件系统相关联。为了将实例与文件系统关联，这些实例运行 cfn-init 帮助程序脚本，以便下载和安装 `nfs-utils` yum 软件包，创建新目录，然后使用文件系统的 DNS 名称来在该目录下挂载文件系统。文件系统的 DNS 名称将解析为 Amazon EC2 实例的可用区中的挂载目标的 IP 地址。有关 DNS 名称结构的更多信息，请参阅《Amazon Elastic File System 用户指南》中的挂载文件系统。

要测量网络文件系统活动，模板应包含自定义 Amazon CloudWatch 指标。模板还创建 VPC、子网和安全组。要允许实例与文件系统通信，VPC 必须已启用 DNS，而且挂载目标和 EC2 实例必须位于子网指定的同一可用区 (AZ) 中。

挂载目标的安全组支持至 TCP 端口 2049 的网络连接，NFSv4 客户端挂载文件系统需要此连接。有关 EC2 实例和挂载目标的安全组的更多信息，请参阅《Amazon Elastic File System 用户指南》中的安全性。

> **Note**
>
> 如果更新挂载目标导致了挂载目标被替换，则使用关联文件系统的实例或应用程序可能会中断。这可能会导致未提交的写入丢失。为了避免中断，请在您更新挂载目标时通过将所需容量设置为零来停止实例。这使得实例可以在删除挂载目标之前卸载文件系统。挂载更新完成之后，通过设置所需容量在后续更新中启动您的实例。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "This template creates an Amazon EFS file system and mount target and
 associates it with Amazon EC2 instances in an Auto Scaling group. **WARNING** This
 template creates Amazon EC2 instances and related resources. You will be billed for
 the AWS resources used if you create a stack from this template.",
  "Parameters": {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [
        "t1.micro",
        "t2.nano",
        "t2.micro",
        "t2.small",
        "t2.medium",
        "t2.large",
        "m1.small",
        "m1.medium",
        "m1.large",
        "m1.xlarge",
        "m2.xlarge",
        "m2.2xlarge",
        "m2.4xlarge",
        "m3.medium",
        "m3.large",
        "m3.xlarge",
        "m3.2xlarge",
        "m4.large",
        "m4.xlarge",
        "m4.2xlarge",
```

```
        "m4.4xlarge",
        "m4.10xlarge",
        "c1.medium",
        "c1.xlarge",
        "c3.large",
        "c3.xlarge",
        "c3.2xlarge",
        "c3.4xlarge",
        "c3.8xlarge",
        "c4.large",
        "c4.xlarge",
        "c4.2xlarge",
        "c4.4xlarge",
        "c4.8xlarge",
        "g2.2xlarge",
        "g2.8xlarge",
        "r3.large",
        "r3.xlarge",
        "r3.2xlarge",
        "r3.4xlarge",
        "r3.8xlarge",
        "i2.xlarge",
        "i2.2xlarge",
        "i2.4xlarge",
        "i2.8xlarge",
        "d2.xlarge",
        "d2.2xlarge",
        "d2.4xlarge",
        "d2.8xlarge",
        "hi1.4xlarge",
        "hs1.8xlarge",
        "cr1.8xlarge",
        "cc2.8xlarge",
        "cg1.4xlarge"
      ],
      "ConstraintDescription" : "must be a valid EC2 instance type."
    },
    "KeyName": {
      "Type": "AWS::EC2::KeyPair::KeyName",
      "Description": "Name of an existing EC2 key pair to enable SSH access to the EC2
instances"
    },
    "AsgMaxSize": {
      "Type": "Number",
```

```
      "Description": "Maximum size and initial desired capacity of Auto Scaling Group",
      "Default": "2"
    },
    "SSHLocation" : {
      "Description" : "The IP address range that can be used to connect to the EC2
instances by using SSH",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "VolumeName" : {
      "Description" : "The name to be used for the EFS volume",
      "Type": "String",
      "MinLength": "1",
      "Default": "myEFSvolume"
    },
    "MountPoint" : {
      "Description" : "The Linux mount point for the EFS volume",
      "Type": "String",
      "MinLength": "1",
      "Default": "myEFSvolume"
    }
  },
  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "HVM64"  },
      "t2.nano"     : { "Arch" : "HVM64"  },
      "t2.micro"    : { "Arch" : "HVM64"  },
      "t2.small"    : { "Arch" : "HVM64"  },
      "t2.medium"   : { "Arch" : "HVM64"  },
      "t2.large"    : { "Arch" : "HVM64"  },
      "m1.small"    : { "Arch" : "HVM64"  },
      "m1.medium"   : { "Arch" : "HVM64"  },
      "m1.large"    : { "Arch" : "HVM64"  },
      "m1.xlarge"   : { "Arch" : "HVM64"  },
      "m2.xlarge"   : { "Arch" : "HVM64"  },
      "m2.2xlarge"  : { "Arch" : "HVM64"  },
      "m2.4xlarge"  : { "Arch" : "HVM64"  },
      "m3.medium"   : { "Arch" : "HVM64"  },
      "m3.large"    : { "Arch" : "HVM64"  },
      "m3.xlarge"   : { "Arch" : "HVM64"  },
```

```
      "m3.2xlarge"  : { "Arch" : "HVM64"  },
      "m4.large"    : { "Arch" : "HVM64"  },
      "m4.xlarge"   : { "Arch" : "HVM64"  },
      "m4.2xlarge"  : { "Arch" : "HVM64"  },
      "m4.4xlarge"  : { "Arch" : "HVM64"  },
      "m4.10xlarge" : { "Arch" : "HVM64"  },
      "c1.medium"   : { "Arch" : "HVM64"  },
      "c1.xlarge"   : { "Arch" : "HVM64"  },
      "c3.large"    : { "Arch" : "HVM64"  },
      "c3.xlarge"   : { "Arch" : "HVM64"  },
      "c3.2xlarge"  : { "Arch" : "HVM64"  },
      "c3.4xlarge"  : { "Arch" : "HVM64"  },
      "c3.8xlarge"  : { "Arch" : "HVM64"  },
      "c4.large"    : { "Arch" : "HVM64"  },
      "c4.xlarge"   : { "Arch" : "HVM64"  },
      "c4.2xlarge"  : { "Arch" : "HVM64"  },
      "c4.4xlarge"  : { "Arch" : "HVM64"  },
      "c4.8xlarge"  : { "Arch" : "HVM64"  },
      "g2.2xlarge"  : { "Arch" : "HVMG2"  },
      "g2.8xlarge"  : { "Arch" : "HVMG2"  },
      "r3.large"    : { "Arch" : "HVM64"  },
      "r3.xlarge"   : { "Arch" : "HVM64"  },
      "r3.2xlarge"  : { "Arch" : "HVM64"  },
      "r3.4xlarge"  : { "Arch" : "HVM64"  },
      "r3.8xlarge"  : { "Arch" : "HVM64"  },
      "i2.xlarge"   : { "Arch" : "HVM64"  },
      "i2.2xlarge"  : { "Arch" : "HVM64"  },
      "i2.4xlarge"  : { "Arch" : "HVM64"  },
      "i2.8xlarge"  : { "Arch" : "HVM64"  },
      "d2.xlarge"   : { "Arch" : "HVM64"  },
      "d2.2xlarge"  : { "Arch" : "HVM64"  },
      "d2.4xlarge"  : { "Arch" : "HVM64"  },
      "d2.8xlarge"  : { "Arch" : "HVM64"  },
      "hi1.4xlarge" : { "Arch" : "HVM64"  },
      "hs1.8xlarge" : { "Arch" : "HVM64"  },
      "cr1.8xlarge" : { "Arch" : "HVM64"  },
      "cc2.8xlarge" : { "Arch" : "HVM64"  }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"        : {"HVM64" : "ami-0ff8a91507f77f867", "HVMG2" :
 "ami-0a584ac55a7631c0c"},
      "us-west-2"        : {"HVM64" : "ami-a0cfeed8", "HVMG2" :
 "ami-0e09505bc235aa82d"},
```

```
      "us-west-1"        : {"HVM64" : "ami-0bdb828fd58c52235", "HVMG2" :
"ami-066ee5fd4a9ef77f1"},
      "eu-west-1"        : {"HVM64" : "ami-047bb4163c506cd98", "HVMG2" :
"ami-0a7c483d527806435"},
      "eu-west-2"        : {"HVM64" : "ami-f976839e", "HVMG2" : "NOT_SUPPORTED"},
      "eu-west-3"        : {"HVM64" : "ami-0ebc281c20e89ba4b", "HVMG2" :
"NOT_SUPPORTED"},
      "eu-central-1"     : {"HVM64" : "ami-0233214e13e500f77", "HVMG2" :
"ami-06223d46a6d0661c7"},
      "ap-northeast-1"   : {"HVM64" : "ami-06cd52961ce9f0d85", "HVMG2" :
"ami-053cdd503598e4a9d"},
      "ap-northeast-2"   : {"HVM64" : "ami-0a10b2721688ce9d2", "HVMG2" :
"NOT_SUPPORTED"},
      "ap-northeast-3"   : {"HVM64" : "ami-0d98120a9fb693f07", "HVMG2" :
"NOT_SUPPORTED"},
      "ap-southeast-1"   : {"HVM64" : "ami-08569b978cc4dfa10", "HVMG2" :
"ami-0be9df32ae9f92309"},
      "ap-southeast-2"   : {"HVM64" : "ami-09b42976632b27e9b", "HVMG2" :
"ami-0a9ce9fecc3d1daf8"},
      "ap-south-1"       : {"HVM64" : "ami-0912f71e06545ad88", "HVMG2" :
"ami-097b15e89dbdcfcf4"},
      "us-east-2"        : {"HVM64" : "ami-0b59bfac6be064b78", "HVMG2" :
"NOT_SUPPORTED"},
      "ca-central-1"     : {"HVM64" : "ami-0b18956f", "HVMG2" : "NOT_SUPPORTED"},
      "sa-east-1"        : {"HVM64" : "ami-07b14488da8ea02a0", "HVMG2" :
"NOT_SUPPORTED"},
      "cn-north-1"       : {"HVM64" : "ami-0a4eaf6c4454eda75", "HVMG2" :
"NOT_SUPPORTED"},
      "cn-northwest-1"   : {"HVM64" : "ami-6b6a7d09", "HVMG2" : "NOT_SUPPORTED"}
    }
  },
  "Resources": {
    "CloudWatchPutMetricsRole" : {
      "Type"  : "AWS::IAM::Role",
      "Properties" : {
          "AssumeRolePolicyDocument" : {
              "Statement" : [ {
                  "Effect" : "Allow",
                  "Principal" : {
                      "Service" : [ "ec2.amazonaws.com" ]
                  },
                  "Action" : [ "sts:AssumeRole" ]
              } ]
          },
```

```
          "Path" : "/"
      }
    },
    "CloudWatchPutMetricsRolePolicy" : {
        "Type" : "AWS::IAM::Policy",
        "Properties" : {
            "PolicyName" : "CloudWatch_PutMetricData",
            "PolicyDocument" : {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Sid": "CloudWatchPutMetricData",
                  "Effect": "Allow",
                  "Action": ["cloudwatch:PutMetricData"],
                  "Resource": ["*"]
                }
              ]
            },
            "Roles" : [ { "Ref" : "CloudWatchPutMetricsRole" } ]
        }
    },
    "CloudWatchPutMetricsInstanceProfile" : {
      "Type" : "AWS::IAM::InstanceProfile",
      "Properties" : {
        "Path" : "/",
        "Roles" : [ { "Ref" : "CloudWatchPutMetricsRole" } ]
      }
    },
    "VPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "EnableDnsSupport" : "true",
        "EnableDnsHostnames" : "true",
        "CidrBlock": "10.0.0.0/16",
        "Tags": [ {"Key": "Application", "Value": { "Ref": "AWS::StackId"} } ]
      }
    },
    "InternetGateway" : {
      "Type" : "AWS::EC2::InternetGateway",
      "Properties" : {
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },
          { "Key" : "Network", "Value" : "Public" }
        ]
```

```
      }
    },
    "GatewayToInternet" : {
      "Type" : "AWS::EC2::VPCGatewayAttachment",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "InternetGatewayId" : { "Ref" : "InternetGateway" }
      }
    },
    "RouteTable":{
      "Type":"AWS::EC2::RouteTable",
      "Properties":{
        "VpcId": {"Ref":"VPC"}
      }
    },
    "SubnetRouteTableAssoc": {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "RouteTableId" : {"Ref":"RouteTable"},
        "SubnetId" : {"Ref":"Subnet"}
      }
    },
    "InternetGatewayRoute": {
        "Type":"AWS::EC2::Route",
        "Properties":{
            "DestinationCidrBlock":"0.0.0.0/0",
            "RouteTableId":{"Ref":"RouteTable"},
            "GatewayId":{"Ref":"InternetGateway"}
        }
    },
    "Subnet": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": { "Ref": "VPC" },
        "CidrBlock": "10.0.0.0/24",
        "Tags": [ { "Key": "Application", "Value": { "Ref": "AWS::StackId" } } ]
      }
    },
    "InstanceSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "VpcId": { "Ref": "VPC" },
        "GroupDescription": "Enable SSH access via port 22",
        "SecurityGroupIngress": [
```

```
          { "IpProtocol": "tcp", "FromPort": 22, "ToPort": 22, "CidrIp": { "Ref":
 "SSHLocation" } },
          { "IpProtocol": "tcp", "FromPort": 80, "ToPort": 80, "CidrIp": "0.0.0.0/0" }
         ]
      }
    },
    "MountTargetSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "VpcId": { "Ref": "VPC" },
        "GroupDescription": "Security group for mount target",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 2049,
            "ToPort": 2049,
            "CidrIp": "0.0.0.0/0"
          }
        ]
      }
    },
    "FileSystem": {
      "Type": "AWS::EFS::FileSystem",
      "Properties": {
        "PerformanceMode": "generalPurpose",
        "FileSystemTags": [
          {
            "Key": "Name",
            "Value": { "Ref" : "VolumeName" }
          }
        ]
      }
    },
    "MountTarget": {
      "Type": "AWS::EFS::MountTarget",
      "Properties": {
        "FileSystemId": { "Ref": "FileSystem" },
        "SubnetId": { "Ref": "Subnet" },
        "SecurityGroups": [ { "Ref": "MountTargetSecurityGroup" } ]
      }
    },
    "LaunchConfiguration": {
      "Type": "AWS::AutoScaling::LaunchConfiguration",
      "Metadata" : {
```

```
        "AWS::CloudFormation::Init" : {
          "configSets" : {
            "MountConfig" : [ "setup", "mount" ]
          },
          "setup" : {
            "packages" : {
              "yum" : {
                "nfs-utils" : []
              }
            },
            "files" : {
              "/home/ec2-user/post_nfsstat" : {
                "content" : { "Fn::Join" : [ "", [
                      "#!/bin/bash\n",
                      "\n",
                      "INPUT=\"$(cat)\"\n",
                      "CW_JSON_OPEN='{ \"Namespace\": \"EFS\", \"MetricData\": [ '\n",
                      "CW_JSON_CLOSE=' ] }'\n",
                      "CW_JSON_METRIC=''\n",
                      "METRIC_COUNTER=0\n",
                      "\n",
                      "for COL in 1 2 3 4 5 6; do\n",
                      "\n",
                      " COUNTER=0\n",
                      " METRIC_FIELD=$COL\n",
                      " DATA_FIELD=$(($COL+($COL-1)))\n",
                      "\n",
                      " while read line; do\n",
                      "   if [[ COUNTER -gt 0 ]]; then\n",
                      "\n",
                      "     LINE=`echo $line | tr -s ' ' `\n",
                      "     AWS_COMMAND=\"aws cloudwatch put-metric-data --region ",
 { "Ref": "AWS::Region" }, "\"\n",
                      "     MOD=$(( $COUNTER % 2))\n",
                      "\n",
                      "     if [ $MOD -eq 1 ]; then\n",
                      "       METRIC_NAME=`echo $LINE | cut -d ' ' -f $METRIC_FIELD`
\n",
                      "     else\n",
                      "       METRIC_VALUE=`echo $LINE | cut -d ' ' -f $DATA_FIELD`\n",
                      "     fi\n",
                      "\n",
                      "     if [[ -n \"$METRIC_NAME\" && -n \"$METRIC_VALUE\" ]]; then
\n",
```

```
                      "       INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-
data/instance-id)\n",
                      "       CW_JSON_METRIC=\"$CW_JSON_METRIC { \\\"MetricName\\\": \\
\"$METRIC_NAME\\\", \\\"Dimensions\\\": [{\\\"Name\\\": \\\"InstanceId\\\", \\\"Value\\
\": \\\"$INSTANCE_ID\\\"} ], \\\"Value\\\": $METRIC_VALUE },\"\n",
                      "       unset METRIC_NAME\n",
                      "       unset METRIC_VALUE\n",
                      "\n",
                      "       METRIC_COUNTER=$((METRIC_COUNTER+1))\n",
                      "       if [ $METRIC_COUNTER -eq 20 ]; then\n",
                      "         # 20 is max metric collection size, so we have to
 submit here\n",
                      "         aws cloudwatch put-metric-data --region ", { "Ref":
 "AWS::Region" }, " --cli-input-json \"`echo $CW_JSON_OPEN ${CW_JSON_METRIC%?}
 $CW_JSON_CLOSE`\"\n",
                      "\n",
                      "         # reset\n",
                      "         METRIC_COUNTER=0\n",
                      "         CW_JSON_METRIC=''\n",
                      "       fi\n",
                      "     fi  \n",
                      "\n",
                      "\n",
                      "\n",
                      "     COUNTER=$((COUNTER+1))\n",
                      "   fi\n",
                      "\n",
                      "   if [[ \"$line\" == \"Client nfs v4:\" ]]; then\n",
                      "     # the next line is the good stuff \n",
                      "     COUNTER=$((COUNTER+1))\n",
                      "   fi\n",
                      " done <<< \"$INPUT\"\n",
                      "done\n",
                      "\n",
                      "# submit whatever is left\n",
                      "aws cloudwatch put-metric-data --region ", { "Ref":
 "AWS::Region" }, " --cli-input-json \"`echo $CW_JSON_OPEN ${CW_JSON_METRIC%?}
 $CW_JSON_CLOSE`\""
                  ] ] },
               "mode": "000755",
               "owner": "ec2-user",
               "group": "ec2-user"
            },
            "/home/ec2-user/crontab" : {
```

```
                "content" : { "Fn::Join" : [ "", [
                  "* * * * * /usr/sbin/nfsstat | /home/ec2-user/post_nfsstat\n"
                ] ] },
                "owner": "ec2-user",
                "group": "ec2-user"
              }
            },
            "commands" : {
              "01_createdir" : {
                "command" : {"Fn::Join" : [ "", [ "mkdir /", { "Ref" :
 "MountPoint" }]]}
              }
            }
          },
          "mount" : {
            "commands" : {
              "01_mount" : {
                "command" : { "Fn::Sub": "sudo mount -t nfs4 -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 ${FileSystem}.efs.
${AWS::Region}.amazonaws.com:/ /${MountPoint}"}
              },
              "02_permissions" : {
                "command" : {"Fn::Join" : [ "", [ "chown ec2-user:ec2-user /",
 { "Ref" : "MountPoint" }]]}
              }
            }
          }
        }
      },
      "Properties": {
        "AssociatePublicIpAddress" : true,
        "ImageId": {
          "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" }, {
            "Fn::FindInMap": [ "AWSInstanceType2Arch", { "Ref": "InstanceType" },
 "Arch" ]
          } ]
        },
        "InstanceType": { "Ref": "InstanceType" },
        "KeyName": { "Ref": "KeyName" },
        "SecurityGroups": [ { "Ref": "InstanceSecurityGroup" } ],
        "IamInstanceProfile" : { "Ref" : "CloudWatchPutMetricsInstanceProfile" },
        "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
             "#!/bin/bash -xe\n",
             "yum install -y aws-cfn-bootstrap\n",
```

```
            "/opt/aws/bin/cfn-init -v ",
            "         --stack ", { "Ref" : "AWS::StackName" },
            "         --resource LaunchConfiguration ",
            "         --configsets MountConfig ",
            "         --region ", { "Ref" : "AWS::Region" }, "\n",

            "crontab /home/ec2-user/crontab\n",

            "/opt/aws/bin/cfn-signal -e $? ",
            "         --stack ", { "Ref" : "AWS::StackName" },
            "         --resource AutoScalingGroup ",
            "         --region ", { "Ref" : "AWS::Region" }, "\n"
        ]]}}
      }
    },
    "AutoScalingGroup": {
      "Type": "AWS::AutoScaling::AutoScalingGroup",
      "DependsOn": ["MountTarget", "GatewayToInternet"],
      "CreationPolicy" : {
        "ResourceSignal" : {
          "Timeout" : "PT15M",
          "Count"   : { "Ref": "AsgMaxSize" }
        }
      },
      "Properties": {
        "VPCZoneIdentifier": [ { "Ref": "Subnet" } ],
        "LaunchConfigurationName": { "Ref": "LaunchConfiguration" },
        "MinSize": "1",
        "MaxSize": { "Ref": "AsgMaxSize" },
        "DesiredCapacity": { "Ref": "AsgMaxSize" },
        "Tags": [ {
          "Key": "Name",
          "Value": "EFS FileSystem Mounted Instance",
          "PropagateAtLaunch": "true"
        } ]
      }
    }
  },
  "Outputs" : {
    "MountTargetID" : {
      "Description" : "Mount target ID",
      "Value" :  { "Ref" : "MountTarget" }
    },
```

```
    "FileSystemID" : {
      "Description" : "File system ID",
      "Value" :  { "Ref" : "FileSystem" }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: This template creates an Amazon EFS file system and mount target and
  associates it with Amazon EC2 instances in an Auto Scaling group. **WARNING** This
  template creates Amazon EC2 instances and related resources. You will be billed
  for the AWS resources used if you create a stack from this template.
Parameters:
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.small
    AllowedValues:
      - t1.micro
      - t2.nano
      - t2.micro
      - t2.small
      - t2.medium
      - t2.large
      - m1.small
      - m1.medium
      - m1.large
      - m1.xlarge
      - m2.xlarge
      - m2.2xlarge
      - m2.4xlarge
      - m3.medium
      - m3.large
      - m3.xlarge
      - m3.2xlarge
      - m4.large
      - m4.xlarge
      - m4.2xlarge
      - m4.4xlarge
      - m4.10xlarge
      - c1.medium
```

```
      - c1.xlarge
      - c3.large
      - c3.xlarge
      - c3.2xlarge
      - c3.4xlarge
      - c3.8xlarge
      - c4.large
      - c4.xlarge
      - c4.2xlarge
      - c4.4xlarge
      - c4.8xlarge
      - g2.2xlarge
      - g2.8xlarge
      - r3.large
      - r3.xlarge
      - r3.2xlarge
      - r3.4xlarge
      - r3.8xlarge
      - i2.xlarge
      - i2.2xlarge
      - i2.4xlarge
      - i2.8xlarge
      - d2.xlarge
      - d2.2xlarge
      - d2.4xlarge
      - d2.8xlarge
      - hi1.4xlarge
      - hs1.8xlarge
      - cr1.8xlarge
      - cc2.8xlarge
      - cg1.4xlarge
    ConstraintDescription: must be a valid EC2 instance type.
  KeyName:
    Type: AWS::EC2::KeyPair::KeyName
    Description: Name of an existing EC2 key pair to enable SSH access to the ECS
      instances
  AsgMaxSize:
    Type: Number
    Description: Maximum size and initial desired capacity of Auto Scaling Group
    Default: '2'
  SSHLocation:
    Description: The IP address range that can be used to connect to the EC2 instances
      by using SSH
    Type: String
```

```
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})"
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  VolumeName:
    Description: The name to be used for the EFS volume
    Type: String
    MinLength: '1'
    Default: myEFSvolume
  MountPoint:
    Description: The Linux mount point for the EFS volume
    Type: String
    MinLength: '1'
    Default: myEFSvolume
Mappings:
  AWSInstanceType2Arch:
    t1.micro:
      Arch: HVM64
    t2.nano:
      Arch: HVM64
    t2.micro:
      Arch: HVM64
    t2.small:
      Arch: HVM64
    t2.medium:
      Arch: HVM64
    t2.large:
      Arch: HVM64
    m1.small:
      Arch: HVM64
    m1.medium:
      Arch: HVM64
    m1.large:
      Arch: HVM64
    m1.xlarge:
      Arch: HVM64
    m2.xlarge:
      Arch: HVM64
    m2.2xlarge:
      Arch: HVM64
    m2.4xlarge:
      Arch: HVM64
    m3.medium:
```

```
      Arch: HVM64
    m3.large:
      Arch: HVM64
    m3.xlarge:
      Arch: HVM64
    m3.2xlarge:
      Arch: HVM64
    m4.large:
      Arch: HVM64
    m4.xlarge:
      Arch: HVM64
    m4.2xlarge:
      Arch: HVM64
    m4.4xlarge:
      Arch: HVM64
    m4.10xlarge:
      Arch: HVM64
    c1.medium:
      Arch: HVM64
    c1.xlarge:
      Arch: HVM64
    c3.large:
      Arch: HVM64
    c3.xlarge:
      Arch: HVM64
    c3.2xlarge:
      Arch: HVM64
    c3.4xlarge:
      Arch: HVM64
    c3.8xlarge:
      Arch: HVM64
    c4.large:
      Arch: HVM64
    c4.xlarge:
      Arch: HVM64
    c4.2xlarge:
      Arch: HVM64
    c4.4xlarge:
      Arch: HVM64
    c4.8xlarge:
      Arch: HVM64
    g2.2xlarge:
      Arch: HVMG2
    g2.8xlarge:
```

```
      Arch: HVMG2
    r3.large:
      Arch: HVM64
    r3.xlarge:
      Arch: HVM64
    r3.2xlarge:
      Arch: HVM64
    r3.4xlarge:
      Arch: HVM64
    r3.8xlarge:
      Arch: HVM64
    i2.xlarge:
      Arch: HVM64
    i2.2xlarge:
      Arch: HVM64
    i2.4xlarge:
      Arch: HVM64
    i2.8xlarge:
      Arch: HVM64
    d2.xlarge:
      Arch: HVM64
    d2.2xlarge:
      Arch: HVM64
    d2.4xlarge:
      Arch: HVM64
    d2.8xlarge:
      Arch: HVM64
    hi1.4xlarge:
      Arch: HVM64
    hs1.8xlarge:
      Arch: HVM64
    cr1.8xlarge:
      Arch: HVM64
    cc2.8xlarge:
      Arch: HVM64
  AWSRegionArch2AMI:
    us-east-1:
      HVM64: ami-0ff8a91507f77f867
      HVMG2: ami-0a584ac55a7631c0c
    us-west-2:
      HVM64: ami-a0cfeed8
      HVMG2: ami-0e09505bc235aa82d
    us-west-1:
      HVM64: ami-0bdb828fd58c52235
```

```
      HVMG2: ami-066ee5fd4a9ef77f1
    eu-west-1:
      HVM64: ami-047bb4163c506cd98
      HVMG2: ami-0a7c483d527806435
    eu-west-2:
      HVM64: ami-f976839e
      HVMG2: NOT_SUPPORTED
    eu-west-3:
      HVM64: ami-0ebc281c20e89ba4b
      HVMG2: NOT_SUPPORTED
    eu-central-1:
      HVM64: ami-0233214e13e500f77
      HVMG2: ami-06223d46a6d0661c7
    ap-northeast-1:
      HVM64: ami-06cd52961ce9f0d85
      HVMG2: ami-053cdd503598e4a9d
    ap-northeast-2:
      HVM64: ami-0a10b2721688ce9d2
      HVMG2: NOT_SUPPORTED
    ap-northeast-3:
      HVM64: ami-0d98120a9fb693f07
      HVMG2: NOT_SUPPORTED
    ap-southeast-1:
      HVM64: ami-08569b978cc4dfa10
      HVMG2: ami-0be9df32ae9f92309
    ap-southeast-2:
      HVM64: ami-09b42976632b27e9b
      HVMG2: ami-0a9ce9fecc3d1daf8
    ap-south-1:
      HVM64: ami-0912f71e06545ad88
      HVMG2: ami-097b15e89dbdcfcf4
    us-east-2:
      HVM64: ami-0b59bfac6be064b78
      HVMG2: NOT_SUPPORTED
    ca-central-1:
      HVM64: ami-0b18956f
      HVMG2: NOT_SUPPORTED
    sa-east-1:
      HVM64: ami-07b14488da8ea02a0
      HVMG2: NOT_SUPPORTED
    cn-north-1:
      HVM64: ami-0a4eaf6c4454eda75
      HVMG2: NOT_SUPPORTED
    cn-northwest-1:
```

```
      HVM64: ami-6b6a7d09
      HVMG2: NOT_SUPPORTED
Resources:
  CloudWatchPutMetricsRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: "/"
  CloudWatchPutMetricsRolePolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: CloudWatch_PutMetricData
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Sid: CloudWatchPutMetricData
          Effect: Allow
          Action:
          - cloudwatch:PutMetricData
          Resource:
          - "*"
      Roles:
      - Ref: CloudWatchPutMetricsRole
  CloudWatchPutMetricsInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: "/"
      Roles:
      - Ref: CloudWatchPutMetricsRole
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
      CidrBlock: 10.0.0.0/16
      Tags:
      - Key: Application
```

```
      Value:
        Ref: AWS::StackId
  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
      - Key: Application
        Value:
          Ref: AWS::StackName
      - Key: Network
        Value: Public
  GatewayToInternet:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId:
        Ref: VPC
      InternetGatewayId:
        Ref: InternetGateway
  RouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId:
        Ref: VPC
  SubnetRouteTableAssoc:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId:
        Ref: RouteTable
      SubnetId:
        Ref: Subnet
  InternetGatewayRoute:
    Type: AWS::EC2::Route
    Properties:
      DestinationCidrBlock: 0.0.0.0/0
      RouteTableId:
        Ref: RouteTable
      GatewayId:
        Ref: InternetGateway
  Subnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId:
        Ref: VPC
      CidrBlock: 10.0.0.0/24
```

```
      Tags:
      - Key: Application
        Value:
          Ref: AWS::StackId
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
        Ref: VPC
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp:
          Ref: SSHLocation
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
  MountTargetSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
        Ref: VPC
      GroupDescription: Security group for mount target
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 2049
        ToPort: 2049
        CidrIp: 0.0.0.0/0
  FileSystem:
    Type: AWS::EFS::FileSystem
    Properties:
      PerformanceMode: generalPurpose
      FileSystemTags:
      - Key: Name
        Value:
          Ref: VolumeName
  MountTarget:
    Type: AWS::EFS::MountTarget
    Properties:
      FileSystemId:
        Ref: FileSystem
```

```
      SubnetId:
        Ref: Subnet
      SecurityGroups:
      - Ref: MountTargetSecurityGroup
  LaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Metadata:
      AWS::CloudFormation::Init:
        configSets:
          MountConfig:
          - setup
          - mount
        setup:
          packages:
            yum:
              nfs-utils: []
          files:
            "/home/ec2-user/post_nfsstat":
              content: !Sub |
                #!/bin/bash

                INPUT="$(cat)"
                CW_JSON_OPEN='{ "Namespace": "EFS", "MetricData": [ '
                CW_JSON_CLOSE=' ] }'
                CW_JSON_METRIC=''
                METRIC_COUNTER=0

                for COL in 1 2 3 4 5 6; do

                 COUNTER=0
                 METRIC_FIELD=$COL
                 DATA_FIELD=$(($COL+($COL-1)))

                 while read line; do
                   if [[ COUNTER -gt 0 ]]; then

                     LINE=`echo $line | tr -s ' ' `
                     AWS_COMMAND="aws cloudwatch put-metric-data --region
${AWS::Region}"
                     MOD=$(( $COUNTER % 2))

                     if [ $MOD -eq 1 ]; then
                       METRIC_NAME=`echo $LINE | cut -d ' ' -f $METRIC_FIELD`
                     else
```

```
                      METRIC_VALUE=`echo $LINE | cut -d ' ' -f $DATA_FIELD`
                    fi

                    if [[ -n "$METRIC_NAME" && -n "$METRIC_VALUE" ]]; then
                      INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/
instance-id)
                      CW_JSON_METRIC="$CW_JSON_METRIC { \"MetricName\": \"$METRIC_NAME
\", \"Dimensions\": [{\"Name\": \"InstanceId\", \"Value\": \"$INSTANCE_ID\"} ], \"Value
\": $METRIC_VALUE },"
                      unset METRIC_NAME
                      unset METRIC_VALUE

                      METRIC_COUNTER=$((METRIC_COUNTER+1))
                      if [ $METRIC_COUNTER -eq 20 ]; then
                        # 20 is max metric collection size, so we have to submit here
                        aws cloudwatch put-metric-data --region ${AWS::Region} --cli-
input-json "`echo $CW_JSON_OPEN ${!CW_JSON_METRIC%?} $CW_JSON_CLOSE`"

                        # reset
                        METRIC_COUNTER=0
                        CW_JSON_METRIC=''
                      fi
                    fi



                    COUNTER=$((COUNTER+1))
                  fi

                  if [[ "$line" == "Client nfs v4:" ]]; then
                    # the next line is the good stuff
                    COUNTER=$((COUNTER+1))
                  fi
                 done <<< "$INPUT"
                done

                # submit whatever is left
                aws cloudwatch put-metric-data --region ${AWS::Region} --cli-input-json
 "`echo $CW_JSON_OPEN ${!CW_JSON_METRIC%?} $CW_JSON_CLOSE`"
              mode: '000755'
              owner: ec2-user
              group: ec2-user
            "/home/ec2-user/crontab":
              content: "* * * * * /usr/sbin/nfsstat | /home/ec2-user/post_nfsstat\n"
```

```
              owner: ec2-user
              group: ec2-user
          commands:
            01_createdir:
              command: !Sub "mkdir /${MountPoint}"
        mount:
          commands:
            01_mount:
              command: !Sub >
                mount -t nfs4 -o nfsvers=4.1 ${FileSystem}.efs.
${AWS::Region}.amazonaws.com:/ /${MountPoint}
            02_permissions:
              command: !Sub "chown ec2-user:ec2-user /${MountPoint}"
    Properties:
      AssociatePublicIpAddress: true
      ImageId:
        Fn::FindInMap:
        - AWSRegionArch2AMI
        - Ref: AWS::Region
        - Fn::FindInMap:
          - AWSInstanceType2Arch
          - Ref: InstanceType
          - Arch
      InstanceType:
        Ref: InstanceType
      KeyName:
        Ref: KeyName
      SecurityGroups:
      - Ref: InstanceSecurityGroup
      IamInstanceProfile:
        Ref: CloudWatchPutMetricsInstanceProfile
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 LaunchConfiguration --configsets MountConfig --region ${AWS::Region}
          crontab /home/ec2-user/crontab
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 AutoScalingGroup --region ${AWS::Region}
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    DependsOn:
    - MountTarget
```

```
    - GatewayToInternet
    CreationPolicy:
      ResourceSignal:
        Timeout: PT15M
        Count:
          Ref: AsgMaxSize
    Properties:
      VPCZoneIdentifier:
      - Ref: Subnet
      LaunchConfigurationName:
        Ref: LaunchConfiguration
      MinSize: '1'
      MaxSize:
        Ref: AsgMaxSize
      DesiredCapacity:
        Ref: AsgMaxSize
      Tags:
      - Key: Name
        Value: EFS FileSystem Mounted Instance
        PropagateAtLaunch: 'true'
Outputs:
  MountTargetID:
    Description: Mount target ID
    Value:
      Ref: MountTarget
  FileSystemID:
    Description: File system ID
    Value:
      Ref: FileSystem
```

Elastic Beanstalk 模板代码段

通过 Elastic Beanstalk，您可以迅速在 Amazon 中部署和管理应用程序，而无需为运行这些应用程序的基础设施操心。以下示例模板可以帮助您在 Amazon CloudFormation 模板中描述 Elastic Beanstalk 资源。

Elastic Beanstalk 示例 PHP

下面的示例模板部署一个存储在 Amazon S3 存储桶中的示例 PHP Web 应用程序。该环境也是一种自动扩缩的负载均衡环境，至少包含两个、最多包含六个 Amazon EC2 实例。它显示了一个使用传统启动配置的 Elastic Beanstalk 环境。有关改用启动模板的信息，请参阅《Amazon Elastic Beanstalk开发者指南》中的启动模板。

使用解决方案堆栈名称替换 *solution-stack*（平台版本）。如需可用解决方案堆栈的列表，请使用 Amazon CLI 命令 aws elasticbeanstalk list-available-solution-stacks。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "sampleApplication": {
            "Type": "AWS::ElasticBeanstalk::Application",
            "Properties": {
                "Description": "AWS Elastic Beanstalk Sample Application"
            }
        },
        "sampleApplicationVersion": {
            "Type": "AWS::ElasticBeanstalk::ApplicationVersion",
            "Properties": {
                "ApplicationName": {
                    "Ref": "sampleApplication"
                },
                "Description": "AWS ElasticBeanstalk Sample Application Version",
                "SourceBundle": {
                    "S3Bucket": {
                        "Fn::Sub": "elasticbeanstalk-samples-${AWS::Region}"
                    },
                    "S3Key": "php-newsample-app.zip"
                }
            }
        },
        "sampleConfigurationTemplate": {
            "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
            "Properties": {
                "ApplicationName": {
                    "Ref": "sampleApplication"
                },
                "Description": "AWS ElasticBeanstalk Sample Configuration Template",
                "OptionSettings": [
                    {
                        "Namespace": "aws:autoscaling:asg",
                        "OptionName": "MinSize",
                        "Value": "2"
                    },
                    {
                        "Namespace": "aws:autoscaling:asg",
```

```
                    "OptionName": "MaxSize",
                    "Value": "6"
                },
                {
                    "Namespace": "aws:elasticbeanstalk:environment",
                    "OptionName": "EnvironmentType",
                    "Value": "LoadBalanced"
                },
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "MyInstanceProfile"
                    }
                }
            ],
            "SolutionStackName": "solution-stack"
        }
    },
    "sampleEnvironment": {
        "Type": "AWS::ElasticBeanstalk::Environment",
        "Properties": {
            "ApplicationName": {
                "Ref": "sampleApplication"
            },
            "Description": "AWS ElasticBeanstalk Sample Environment",
            "TemplateName": {
                "Ref": "sampleConfigurationTemplate"
            },
            "VersionLabel": {
                "Ref": "sampleApplicationVersion"
            }
        }
    },
    "MyInstanceRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
```

```
                                    "ec2.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "Description": "Beanstalk EC2 role",
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/AWSElasticBeanstalkWebTier",
                    "arn:aws:iam::aws:policy/AWSElasticBeanstalkMulticontainerDocker",
                    "arn:aws:iam::aws:policy/AWSElasticBeanstalkWorkerTier"
                ]
            }
        },
        "MyInstanceProfile": {
            "Type": "AWS::IAM::InstanceProfile",
            "Properties": {
                "Roles": [
                    {
                        "Ref": "MyInstanceRole"
                    }
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  sampleApplication:
    Type: AWS::ElasticBeanstalk::Application
    Properties:
      Description: AWS Elastic Beanstalk Sample Application
  sampleApplicationVersion:
    Type: AWS::ElasticBeanstalk::ApplicationVersion
    Properties:
      ApplicationName:
        Ref: sampleApplication
```

```
      Description: AWS ElasticBeanstalk Sample Application Version
      SourceBundle:
        S3Bucket: !Sub "elasticbeanstalk-samples-${AWS::Region}"
        S3Key: php-newsample-app.zip
  sampleConfigurationTemplate:
    Type: AWS::ElasticBeanstalk::ConfigurationTemplate
    Properties:
      ApplicationName:
        Ref: sampleApplication
      Description: AWS ElasticBeanstalk Sample Configuration Template
      OptionSettings:
      - Namespace: aws:autoscaling:asg
        OptionName: MinSize
        Value: '2'
      - Namespace: aws:autoscaling:asg
        OptionName: MaxSize
        Value: '6'
      - Namespace: aws:elasticbeanstalk:environment
        OptionName: EnvironmentType
        Value: LoadBalanced
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref MyInstanceProfile
      SolutionStackName: solution-stack
  sampleEnvironment:
    Type: AWS::ElasticBeanstalk::Environment
    Properties:
      ApplicationName:
        Ref: sampleApplication
      Description: AWS ElasticBeanstalk Sample Environment
      TemplateName:
        Ref: sampleConfigurationTemplate
      VersionLabel:
        Ref: sampleApplicationVersion
  MyInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
```

```
            Action:
              - sts:AssumeRole
      Description: Beanstalk EC2 role
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSElasticBeanstalkWebTier
        - arn:aws:iam::aws:policy/AWSElasticBeanstalkMulticontainerDocker
        - arn:aws:iam::aws:policy/AWSElasticBeanstalkWorkerTier
  MyInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref MyInstanceRole
```

Elastic Load Balancing 模板代码段

要创建应用程序负载均衡器、网络负载均衡器或网关负载均衡器，请使用以 AWS::ElasticLoadBalancingV2 开头的 V2 资源类型。要创建经典负载均衡器，请使用以 AWS::ElasticLoadBalancing 开头的资源类型。

内容

- ELBv2 资源
- 经典负载均衡器资源

ELBv2 资源

此示例定义了一个应用程序负载均衡器，该负载均衡器具有 HTTP 侦听器和用于将流量转发到目标组的默认操作。负载均衡器使用默认运行状况检查设置。目标组有两个注册的 EC2 实例。

YAML

```
Resources:
  myLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
```

```
        - !Ref mySecurityGroup

  myHTTPlistener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: HTTP
      Port: 80
      DefaultActions:
        - Type: "forward"
          TargetGroupArn: !Ref myTargetGroup

  myTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Name: "my-target-group"
      Protocol: HTTP
      Port: 80
      TargetType: instance
      VpcId: !Ref myVPC
      Targets:
        - Id: !GetAtt Instance1.InstanceId
          Port: 80
        - Id: !GetAtt Instance2.InstanceId
          Port: 80
```

JSON

```
{
    "Resources": {
        "myLoadBalancer": {
            "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
            "Properties": {
                "Name": "my-alb",
                "Type": "application",
                "Scheme": "internal",
                "Subnets": [
                    {
                        "Ref": "subnet-AZ1"
                    },
                    {
                        "Ref": "subnet-AZ2"
                    }
```

```
                ],
                "SecurityGroups": [
                    {
                        "Ref": "mySecurityGroup"
                    }
                ]
            }
        },
        "myHTTPlistener": {
            "Type": "AWS::ElasticLoadBalancingV2::Listener",
            "Properties": {
                "LoadBalancerArn": {
                    "Ref": "myLoadBalancer"
                },
                "Protocol": "HTTP",
                "Port": 80,
                "DefaultActions": [
                    {
                        "Type": "forward",
                        "TargetGroupArn": {
                            "Ref": "myTargetGroup"
                        }
                    }
                ]
            }
        },
        "myTargetGroup": {
            "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
            "Properties": {
                "Name": "my-target-group",
                "Protocol": "HTTP",
                "Port": 80,
                "TargetType": "instance",
                "VpcId": {
                    "Ref": "myVPC"
                },
                "Targets": [
                    {
                        "Id": {
                            "Fn::GetAtt": [
                                "Instance1",
                                "InstanceId"
                            ]
                        },
```

```
                    "Port": 80
                },
                {
                    "Id": {
                        "Fn::GetAtt": [
                            "Instance2",
                            "InstanceId"
                        ]
                    },
                    "Port": 80
                }
            ]
        }
    }
}
```

经典负载均衡器资源

此示例定义了一个经典负载均衡器，该负载均衡器具有 HTTP 侦听器且没有注册的 EC2 实例。负载均衡器使用默认运行状况检查设置。

YAML

```
myLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
    - "cn-north-1a"
    Listeners:
    - LoadBalancerPort: '80'
      InstancePort: '80'
      Protocol: HTTP
```

JSON

```
"myLoadBalancer" : {
    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties" : {
        "AvailabilityZones" : [ "cn-north-1a" ],
        "Listeners" : [ {
            "LoadBalancerPort" : "80",
```

```
            "InstancePort" : "80",
            "Protocol" : "HTTP"
        } ]
    }
}
```

此示例定义了一个经典负载均衡器，该负载均衡器具有 HTTP 侦听器、两个注册的 EC2 实例和自定义运行状况检查设置。

YAML

```
myClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
    - "cn-north-1a"
    Instances:
    - Ref: Instance1
    - Ref: Instance2
    Listeners:
    - LoadBalancerPort: '80'
      InstancePort: '80'
      Protocol: HTTP
    HealthCheck:
      Target: HTTP:80/
      HealthyThreshold: '3'
      UnhealthyThreshold: '5'
      Interval: '30'
      Timeout: '5'
```

JSON

```
"myClassicLoadBalancer" : {
    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties" : {
        "AvailabilityZones" : [ "cn-north-1a" ],
        "Instances" : [
            { "Ref" : "Instance1" },
            { "Ref" : "Instance2" }
        ],
        "Listeners" : [ {
            "LoadBalancerPort" : "80",
```

```
            "InstancePort" : "80",
            "Protocol" : "HTTP"
        } ],

        "HealthCheck" : {
            "Target" : "HTTP:80/",
            "HealthyThreshold" : "3",
            "UnhealthyThreshold" : "5",
            "Interval" : "30",
            "Timeout" : "5"
        }
    }
}
```

Amazon Identity and Access Management 模板代码段

本部分包含 Amazon Identity and Access Management 模板代码段。

主题

- 声明 IAM 用户资源
- 声明 IAM 访问密钥资源
- 声明 IAM 组资源
- 将用户添加到组中
- 声明 IAM policy
- 声明 Amazon S3 存储桶策略
- 声明 Amazon SNS 主题策略
- 声明 Amazon SQS 策略
- IAM 角色模板示例

> **Important**
>
> 使用包含 IAM 资源的模板创建或更新堆栈时，您必须确认 IAM 功能的使用。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。

声明 IAM 用户资源

此代码段显示如何声明 AWS::IAM::User 资源以创建 IAM 用户。此用户使用路径 "/" 和密码为 myP@ssW0rd 的登录配置文件进行声明。

名为 giveaccesstoqueueonly 的策略文档为用户授予权限以对 Amazon SQS 队列资源 myqueue 执行所有 Amazon SQS 操作，并拒绝对所有其他 Amazon SQS 队列资源进行访问。Fn::GetAtt 函数将获取 AWS::SQS::Queue 资源 myqueue 的 Arn 属性。

可以在用户中添加名为 giveaccesstotopiconly 的策略文档，以便为用户授予权限以对 Amazon SNS 主题资源 mytopic 执行所有 Amazon SNS 操作，并拒绝对所有其他 Amazon SNS 资源进行访问。Ref 函数将获取 AWS::SNS::Topic 资源 mytopic 的 ARN。

JSON

```
"myuser" : {
   "Type" : "AWS::IAM::User",
   "Properties" : {
      "Path" : "/",
      "LoginProfile" : {
         "Password" : "myP@ssW0rd"
      },
      "Policies" : [ {
         "PolicyName" : "giveaccesstoqueueonly",
         "PolicyDocument" : {
            "Version": "2012-10-17",
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sqs:*" ],
               "Resource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
               } ]
            }, {
               "Effect" : "Deny",
               "Action" : [ "sqs:*" ],
               "NotResource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
               } ]
            }
         ] }
      }, {
         "PolicyName" : "giveaccesstotopiconly",
         "PolicyDocument" : {
```

```
            "Version": "2012-10-17",
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sns:*" ],
               "Resource" : [ { "Ref" : "mytopic" } ]
            }, {
               "Effect" : "Deny",
               "Action" : [ "sns:*" ],
               "NotResource" : [ { "Ref" : "mytopic" } ]
            } ]
         }
      } ]
   }
}
```

YAML

```
myuser:
  Type: AWS::IAM::User
  Properties:
    Path: "/"
    LoginProfile:
      Password: myP@ssW0rd
    Policies:
    - PolicyName: giveaccesstoqueueonly
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - sqs:*
          Resource:
          - !GetAtt myqueue.Arn
        - Effect: Deny
          Action:
          - sqs:*
          NotResource:
          - !GetAtt myqueue.Arn
    - PolicyName: giveaccesstotopiconly
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
```

```
          Action:
          - sns:*
          Resource:
          - !Ref mytopic
        - Effect: Deny
          Action:
          - sns:*
          NotResource:
          - !Ref mytopic
```

声明 IAM 访问密钥资源

此代码段显示的是 AWS::IAM::AccessKey 资源。myaccesskey 资源创建访问密钥并将其分配给在模板中声明为 AWS::IAM::User 资源的 IAM 用户。

JSON

```
"myaccesskey" : {
   "Type" : "AWS::IAM::AccessKey",
   "Properties" : {
      "UserName" : { "Ref" : "myuser" }
   }
}
```

YAML

```
myaccesskey:
  Type: AWS::IAM::AccessKey
  Properties:
    UserName:
      !Ref myuser
```

您可使用 AWS::IAM::AccessKey 函数获取 Fn::GetAtt 资源的私有密钥。检索密钥的一种方式是将其放入 Output 值中。您可使用 Ref 函数获取访问密钥。以下 Output 值声明获取 myaccesskey 的访问密钥和私有密钥。

JSON

```
"AccessKeyformyaccesskey" : {
```

```
   "Value" : { "Ref" : "myaccesskey" }
},
"SecretKeyformyaccesskey" : {
   "Value" : {
      "Fn::GetAtt" : [ "myaccesskey", "SecretAccessKey" ]
   }
}
```

YAML

```
AccessKeyformyaccesskey:
  Value:
    !Ref myaccesskey
SecretKeyformyaccesskey:
  Value: !GetAtt myaccesskey.SecretAccessKey
```

您还可以将 Amazon 访问密钥和私有密钥传输给在模板中定义的 Amazon EC2 实例或自动扩缩组。以下 AWS::EC2::Instance 声明使用 `UserData` 属性传递 `myaccesskey` 资源的访问密钥和私有密钥。

JSON

```
"myinstance" : {
   "Type" : "AWS::EC2::Instance",
   "Properties" : {
      "AvailabilityZone" : "cn-north-1a",
      "ImageId" : "ami-0ff8a91507f77f867",
      "UserData" : {
         "Fn::Base64" : {
            "Fn::Join" : [
               "", [
                  "ACCESS_KEY=", {
                     "Ref" : "myaccesskey"
                  },
                  "&",
                  "SECRET_KEY=",
                  {
                     "Fn::GetAtt" : [
                        "myaccesskey",
                        "SecretAccessKey"
                     ]
                  }
```

```
               ]
            ]
         }
      }
   }
}
```

YAML

```
myinstance:
  Type: AWS::EC2::Instance
  Properties:
    AvailabilityZone: "cn-north-1a"
    ImageId: ami-0ff8a91507f77f867
    UserData:
      Fn::Base64: !Sub "ACCESS_KEY=${myaccesskey}&SECRET_KEY=
${myaccesskey.SecretAccessKey}"
```

声明 IAM 组资源

此代码段显示的是 AWS::IAM::Group 资源。该组有一个路径 ("/myapplication/")。可以在组中添加名为 myapppolicy 的策略文档，以允许组的用户对 Amazon SQS 队列资源 myqueue 执行所有 Amazon SQS 操作，并拒绝对 myqueue 以外的所有其他 Amazon SQS 资源进行访问。

要分配一个策略给资源，IAM 需要该资源的 Amazon 资源名称（ARN）。在此代码段中，Fn::GetAtt 函数将获取 AWS::SQS::Queue 资源队列的 ARN。

JSON

```
"mygroup" : {
   "Type" : "AWS::IAM::Group",
   "Properties" : {
      "Path" : "/myapplication/",
      "Policies" : [ {
         "PolicyName" : "myapppolicy",
         "PolicyDocument" : {
            "Version": "2012-10-17",
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sqs:*" ],
               "Resource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
```

```
               } ]
            },
            {
               "Effect" : "Deny",
               "Action" : [ "sqs:*" ],
               "NotResource" : [ { "Fn::GetAtt" : [ "myqueue", "Arn" ] } ]
            }
         ] }
      } ]
   }
}
```

YAML

```
mygroup:
  Type: AWS::IAM::Group
  Properties:
    Path: "/myapplication/"
    Policies:
    - PolicyName: myapppolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - sqs:*
          Resource: !GetAtt myqueue.Arn
        - Effect: Deny
          Action:
          - sqs:*
          NotResource: !GetAtt myqueue.Arn
```

将用户添加到组中

AWS::IAM::UserToGroupAddition 资源会将用户添加到组。在以下代码段中，addUserToGroup 资源将以下用户添加到名为 myexistinggroup2 的现有组中：现有用户 existinguser1 和在模板中声明为 AWS::IAM::User 资源的用户 myuser。

JSON

```
"addUserToGroup" : {
   "Type" : "AWS::IAM::UserToGroupAddition",
```

```
   "Properties" : {
      "GroupName" : "myexistinggroup2",
      "Users" : [ "existinguser1", { "Ref" : "myuser" } ]
   }
}
```

YAML

```
addUserToGroup:
  Type: AWS::IAM::UserToGroupAddition
  Properties:
    GroupName: myexistinggroup2
    Users:
    - existinguser1
    - !Ref myuser
```

声明 IAM policy

此代码段显示如何创建策略并使用名为 mypolicy 的 AWS::IAM::Policy 资源将该策略应用于多个组。mypolicy 资源包含一个 PolicyDocument 属性，该属性允许对 S3 存储桶 (由 ARN GetObject 表示) 中的对象执行 PutObject、PutObjectAcl 和 arn:aws:s3:::myAWSBucket 操作。mypolicy 资源将策略应用于名为 myexistinggroup1 的现有组以及在模板中声明为 AWS::IAM::Group 资源的组 mygroup。此示例显示如何使用 Groups 属性将策略应用于组；但您也可以使用 Users 属性将策略文档添加到用户列表。

JSON

```
"mypolicy" : {
   "Type" : "AWS::IAM::Policy",
   "Properties" : {
      "PolicyName" : "mygrouppolicy",
      "PolicyDocument" : {
         "Version": "2012-10-17",
         "Statement" : [ {
            "Effect" : "Allow",
            "Action" : [
               "s3:GetObject" , "s3:PutObject" , "s3:PutObjectAcl" ],
            "Resource" : "arn:aws:s3:::myAWSBucket/*"
         } ]
      },
      "Groups" : [ "myexistinggroup1", { "Ref" : "mygroup" } ]
```

```
   }
}
```

YAML

```
mypolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: mygrouppolicy
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Action:
        - s3:GetObject
        - s3:PutObject
        - s3:PutObjectAcl
        Resource: arn:aws:s3:::myAWSBucket/*
    Groups:
    - myexistinggroup1
    - !Ref mygroup
```

声明 Amazon S3 存储桶策略

此代码段说明如何创建策略并将其应用于使用 AWS::S3::BucketPolicy 资源的 Amazon S3 存储桶。mybucketpolicy 资源声明一个策略文档，以允许 user1 IAM 用户对应用了该策略的 S3 存储桶中的所有对象执行 GetObject 操作。在此代码段中，Fn::GetAtt 函数将获取 user1 资源的 ARN。mybucketpolicy 资源将此策略应用于 AWS::S3::BucketPolicy 资源 mybucket。Refmybucket function 获取 资源的存储桶名称。

JSON

```
"mybucketpolicy" : {
   "Type" : "AWS::S3::BucketPolicy",
   "Properties" : {
      "PolicyDocument" : {
         "Id" : "MyPolicy",
         "Version": "2012-10-17",
         "Statement" : [ {
            "Sid" : "ReadAccess",
            "Action" : [ "s3:GetObject" ],
```

```
            "Effect" : "Allow",
            "Resource" : { "Fn::Join" : [
                  "", [ "arn:aws:s3:::", { "Ref" : "mybucket" } , "/*" ]
               ] },
            "Principal" : {
               "AWS" : { "Fn::GetAtt" : [ "user1", "Arn" ] }
            }
         } ]
      },
      "Bucket" : { "Ref" : "mybucket" }
   }
}
```

YAML

```
mybucketpolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    PolicyDocument:
      Id: MyPolicy
      Version: '2012-10-17'
      Statement:
      - Sid: ReadAccess
        Action:
        - s3:GetObject
        Effect: Allow
        Resource: !Sub "arn:aws:s3:::${mybucket}/*"
        Principal:
          AWS: !GetAtt user1.Arn
    Bucket: !Ref mybucket
```

声明 Amazon SNS 主题策略

此代码段说明如何创建策略并将其应用于使用 AWS::SNS::TopicPolicy 资源的 Amazon SNS 主题。mysnspolicy 资源包含一个 PolicyDocument 属性，该属性允许 AWS::IAM::User 资源 myuser 对 AWS::SNS::Topic 资源 mytopic 执行 Publish 操作。在此代码段中，Fn::GetAtt 函数将获取 myuser 资源的 ARN，而 Ref 函数将获取 mytopic 资源的 ARN。

JSON

```
"mysnspolicy" : {
   "Type" : "AWS::SNS::TopicPolicy",
```

```
   "Properties" : {
      "PolicyDocument" :  {
         "Id" : "MyTopicPolicy",
         "Version": "2012-10-17",
         "Statement" : [ {
            "Sid" : "My-statement-id",
            "Effect" : "Allow",
            "Principal" : {
               "AWS" : { "Fn::GetAtt" : [ "myuser", "Arn" ] }
            },
            "Action" : "sns:Publish",
            "Resource" : "*"
         } ]
      },
      "Topics" : [ { "Ref" : "mytopic" } ]
   }
}
```

YAML

```
mysnspolicy:
  Type: AWS::SNS::TopicPolicy
  Properties:
    PolicyDocument:
      Id: MyTopicPolicy
      Version: '2012-10-17'
      Statement:
      - Sid: My-statement-id
        Effect: Allow
        Principal:
          AWS: !GetAtt myuser.Arn
        Action: sns:Publish
        Resource: "*"
    Topics:
    - !Ref mytopic
```

声明 Amazon SQS 策略

该代码段说明如何使用 AWS::SQS::QueuePolicy 资源创建策略并将其应用于 Amazon SQS 队列。PolicyDocument 属性可使现有用户 myapp（由其 ARN 指定）对现有队列（按其 URL 指定）和 AWS::SQS::Queue 资源 myqueue 执行 SendMessage 操作。Ref 函数获取 资源的 URL。myqueue

JSON

```
"mysqspolicy" : {
   "Type" : "AWS::SQS::QueuePolicy",
   "Properties" : {
      "PolicyDocument" : {
         "Id" : "MyQueuePolicy",
         "Version": "2012-10-17",
         "Statement" : [ {
            "Sid" : "Allow-User-SendMessage",
            "Effect" : "Allow",
            "Principal" : {
               "AWS" : "arn:aws:iam::123456789012:user/myapp"
            },
            "Action" : [ "sqs:SendMessage" ],
            "Resource" : "*"
         } ]
      },
      "Queues" : [
         "https://sqs.us-east-2aws-region.amazonaws.com/123456789012/myexistingqueue",
         { "Ref" : "myqueue" }
      ]
   }
}
```

YAML

```
mysqspolicy:
  Type: AWS::SQS::QueuePolicy
  Properties:
    PolicyDocument:
      Id: MyQueuePolicy
      Version: '2012-10-17'
      Statement:
      - Sid: Allow-User-SendMessage
        Effect: Allow
        Principal:
          AWS: arn:aws:iam::123456789012:user/myapp
        Action:
        - sqs:SendMessage
        Resource: "*"
    Queues:
    - https://sqs.aws-region.amazonaws.com/123456789012/myexistingqueue
```

```
    - !Ref myqueue
```

IAM 角色模板示例

本部分提供 EC2 实例 IAM 角色的 CloudFormation 模板示例。

有关更多信息，请参阅《Amazon EC2 用户指南》中的适用于 Amazon EC2 的 IAM 角色。

带 EC2 的 IAM 角色

在此示例中，实例配置文件由 EC2 实例的 `IamInstanceProfile` 属性引用。实例策略和角色策略都引用 AWS::IAM::Role。

JSON

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "myEC2Instance": {
         "Type": "AWS::EC2::Instance",
         "Version": "2009-05-15",
         "Properties": {
            "ImageId": "ami-0ff8a91507f77f867",
            "InstanceType": "m1.small",
            "Monitoring": "true",
            "DisableApiTermination": "false",
            "IamInstanceProfile": {
               "Ref": "RootInstanceProfile"
            }
         }
      },
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
```

```
            },
            "Path": "/"
         }
      },
      "RolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "root",
            "PolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
               } ]
            },
            "Roles": [ { "Ref": "RootRole" } ]
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "RootRole" } ]
         }
      }
   }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myEC2Instance:
    Type: AWS::EC2::Instance
    Version: '2009-05-15'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: m1.small
      Monitoring: 'true'
      DisableApiTermination: 'false'
      IamInstanceProfile:
        !Ref RootInstanceProfile
```

```
  RootRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: "/"
  RolePolicies:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: root
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action: "*"
          Resource: "*"
      Roles:
      - !Ref RootRole
  RootInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: "/"
      Roles:
      - !Ref RootRole
```

带自动扩缩组的 IAM 角色

在此示例中，实例配置文件由 Amazon EC2 Auto Scaling 启动配置的 `IamInstanceProfile` 属性引用。

JSON

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "myLCOne": {
```

```
         "Type": "AWS::AutoScaling::LaunchConfiguration",
         "Version": "2009-05-15",
         "Properties": {
            "ImageId": "ami-0ff8a91507f77f867",
            "InstanceType": "m1.small",
            "InstanceMonitoring": "true",
            "IamInstanceProfile": { "Ref": "RootInstanceProfile" }
         }
      },
      "myASGrpOne": {
         "Type": "AWS::AutoScaling::AutoScalingGroup",
         "Version": "2009-05-15",
         "Properties": {
            "AvailabilityZones": [ "cn-north-1a" ],
            "LaunchConfigurationName": { "Ref": "myLCOne" },
            "MinSize": "0",
            "MaxSize": "0",
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": "120"
         }
      },
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "root",
            "PolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
```

```
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
               } ]
            },
            "Roles": [ { "Ref": "RootRole" } ]
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "RootRole" } ]
         }
      }
   }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myLCOne:
    Type: AWS::AutoScaling::LaunchConfiguration
    Version: '2009-05-15'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: m1.small
      InstanceMonitoring: 'true'
      IamInstanceProfile:
        !Ref RootInstanceProfile
  myASGrpOne:
    Type: AWS::AutoScaling::AutoScalingGroup
    Version: '2009-05-15'
    Properties:
      AvailabilityZones:
      - "cn-north-1a"
      LaunchConfigurationName:
        !Ref myLCOne
      MinSize: '0'
      MaxSize: '0'
      HealthCheckType: EC2
      HealthCheckGracePeriod: '120'
```

```
  RootRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: "/"
  RolePolicies:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: root
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action: "*"
          Resource: "*"
      Roles:
      - !Ref RootRole
  RootInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: "/"
      Roles:
      - !Ref RootRole
```

Amazon Lambda 模板

以下模板使用 Amazon Lambda（Lambda）函数和自定义资源向现有安全组列表附加新的安全组。当您需要动态构建安全组列表以使列表同时包含新安全组及现有安全组时，此函数非常有用。例如，您可以将现有安全组列表作为参数值传递，向此列表追加一个新值，然后将您所有的值关联到一个 EC2 实例。有关 Lambda 函数资源类型的更多信息，请参阅 AWS::Lambda::Function。

在此示例中，当 CloudFormation 创建 `AllSecurityGroups` 自定义资源时，CloudFormation 会调用 `AppendItemToListFunction` Lambda 函数。CloudFormation 将现有的安全组列表以及一个新的安

全组（NewSecurityGroup）传递给该函数，而该函数会将新的安全组附加到列表中，然后返回修改后的列表。CloudFormation 使用修改后的列表将所有安全组与 MyEC2Instance 资源关联起来。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "ExistingSecurityGroups": {
            "Type": "List<AWS::EC2::SecurityGroup::Id>"
        },
        "ExistingVPC": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "The VPC ID that includes the security groups in the
 ExistingSecurityGroups parameter."
        },
        "InstanceType": {
            "Type": "String",
            "Default": "t2.micro",
            "AllowedValues": [
                "t2.micro",
                "t3.micro"
            ]
        }
    },

    "Resources": {
        "SecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Allow HTTP traffic to the host",
                "VpcId": {
                    "Ref": "ExistingVPC"
                },
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "CidrIp": "0.0.0.0/0"
                    }
                ],
                "SecurityGroupEgress": [
```

```
                {
                    "IpProtocol": "tcp",
                    "FromPort": 80,
                    "ToPort": 80,
                    "CidrIp": "0.0.0.0/0"
                }
            ]
        }
    },
    "AllSecurityGroups": {
        "Type": "Custom::Split",
        "Properties": {
            "ServiceToken": {
                "Fn::GetAtt": [
                    "AppendItemToListFunction",
                    "Arn"
                ]
            },
            "List": {
                "Ref": "ExistingSecurityGroups"
            },
            "AppendedItem": {
                "Ref": "SecurityGroup"
            }
        }
    },
    "AppendItemToListFunction": {
        "Type": "AWS::Lambda::Function",
        "Properties": {
            "Handler": "index.handler",
            "Role": {
                "Fn::GetAtt": [
                    "LambdaExecutionRole",
                    "Arn"
                ]
            },
            "Code": {
                "ZipFile": {
                    "Fn::Join": [
                        "",
                        [
                            "var response = require('cfn-response');",
                            "exports.handler = function(event, context) {",
```

```
                                "   var responseData = {Value:
 event.ResourceProperties.List};",
                                "
 responseData.Value.push(event.ResourceProperties.AppendedItem);",
                                "   response.send(event, context, response.SUCCESS,
 responseData);",
                                "};"
                            ]
                        ]
                    }
                },
                "Runtime": "nodejs20.x"
            }
        },
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-
ami-hvm-x86_64-gp2}}",
                "SecurityGroupIds": {
                    "Fn::GetAtt": [
                        "AllSecurityGroups",
                        "Value"
                    ]
                },
                "InstanceType": {
                    "Ref": "InstanceType"
                }
            }
        },
        "LambdaExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "lambda.amazonaws.com"
                                ]
                            },
                            "Action": [
```

```
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "root",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "logs:*"
                                    ],
                                    "Resource": "arn:aws:logs:*:*:*"
                                }
                            ]
                        }
                    }
                ]
            }
        }
    },
    "Outputs": {
        "AllSecurityGroups": {
            "Description": "Security Groups that are associated with the EC2 instance",
            "Value": {
                "Fn::Join": [
                    ", ",
                    {
                        "Fn::GetAtt": [
                            "AllSecurityGroups",
                            "Value"
                        ]
                    }
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  ExistingSecurityGroups:
    Type: List<AWS::EC2::SecurityGroup::Id>
  ExistingVPC:
    Type: AWS::EC2::VPC::Id
    Description: The VPC ID that includes the security groups in the
 ExistingSecurityGroups parameter.
  InstanceType:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t3.micro
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP traffic to the host
      VpcId: !Ref ExistingVPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
  AllSecurityGroups:
    Type: Custom::Split
    Properties:
      ServiceToken: !GetAtt AppendItemToListFunction.Arn
      List: !Ref ExistingSecurityGroups
      AppendedItem: !Ref SecurityGroup
  AppendItemToListFunction:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
```

```
      Code:
        ZipFile: !Join
          - ''
          - - var response = require('cfn-response');
            - exports.handler = function(event, context) {
            - '   var responseData = {Value: event.ResourceProperties.List};'
            - '   responseData.Value.push(event.ResourceProperties.AppendedItem);'
            - '   response.send(event, context, response.SUCCESS, responseData);'
            - '};'
      Runtime: nodejs20.x
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      SecurityGroupIds: !GetAtt AllSecurityGroups.Value
      InstanceType: !Ref InstanceType
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - logs:*
                Resource: arn:aws:logs:*:*:*
Outputs:
  AllSecurityGroups:
    Description: Security Groups that are associated with the EC2 instance
    Value: !Join
      - ', '
```

```
      - !GetAtt AllSecurityGroups.Value
```

Amazon Redshift 模板代码段

Amazon Redshift 是云中一种完全托管的 PB 级数据仓库服务。您可以使用 Amazon CloudFormation 预置和管理 Amazon Redshift 集群。

Amazon Redshift 集群

下述示例模板根据在创建堆栈时指定的参数值创建 Amazon Redshift 集群。与 Amazon Redshift 集群关联的集群参数组可实现用户活动日志记录。该模板还在模板中定义的 Amazon VPC 中启动 Amazon Redshift 集群。VPC 包含一个 Internet 网关，以便您从 Internet 访问 Amazon Redshift 集群。然而，还必须启用集群与 Internet 网关之间的通信，这是通过路由表条目实现的。

> **Note**
>
> 该模板包含 `IsMultiNodeCluster` 条件，以便仅当 `NumberOfNodes` 参数值设置为 `ClusterType` 时才声明 `multi-node` 参数。

该示例定义了 `MysqlRootPassword` 参数，并将其 `NoEcho` 属性设置为 `true`。如果您将 `NoEcho` 属性设置为 `true`，则对于描述堆栈或堆栈事件的任何调用，CloudFormation 返回使用星号 (*****) 遮蔽的参数值，但存储在下面指定位置的信息除外。

> **Important**
>
> 使用 `NoEcho` 属性不会遮蔽在以下各区段中存储的任何信息：
>
> - `Metadata` 模板区段。CloudFormation 不会转换、修改或编辑您在 `Metadata` 部分中包含的任何信息。有关更多信息，请参阅 Metadata。
> - `Outputs` 模板区段。有关更多信息，请参阅 Outputs。
> - 资源定义的 `Metadata` 属性。有关更多信息，请参阅 `Metadata` 属性。
>
> 强烈建议您不要使用这些机制来包含敏感信息，例如密码。

> **Important**
>
> 我们建议不要将敏感信息直接嵌入 CloudFormation 模板中，而应使用堆栈模板中的动态参数来引用在 CloudFormation 外部存储和管理的敏感信息，例如 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager 中的敏感信息。
> 有关更多信息，请参阅请勿将凭证嵌入您的模板最佳实践。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters" : {
    "DatabaseName" : {
      "Description" : "The name of the first database to be created when the cluster is
 created",
      "Type" : "String",
      "Default" : "dev",
      "AllowedPattern" : "([a-z]|[0-9])+"
    },
    "ClusterType" : {
      "Description" : "The type of cluster",
      "Type" : "String",
      "Default" : "single-node",
      "AllowedValues" : [ "single-node", "multi-node" ]
    },
    "NumberOfNodes" : {
      "Description" : "The number of compute nodes in the cluster. For multi-node
 clusters, the NumberOfNodes parameter must be greater than 1",
      "Type" : "Number",
      "Default" : "1"
    },
    "NodeType" : {
      "Description" : "The type of node to be provisioned",
      "Type" : "String",
      "Default" : "ds2.xlarge",
      "AllowedValues" : [ "ds2.xlarge", "ds2.8xlarge", "dc1.large", "dc1.8xlarge" ]
    },
    "MasterUsername" : {
      "Description" : "The user name that is associated with the master user account
 for the cluster that is being created",
      "Type" : "String",
```

```
      "Default" : "defaultuser",
      "AllowedPattern" : "([a-z])([a-z]|[0-9])*"
    },
    "MasterUserPassword" :  {
      "Description" : "The password that is associated with the master user account for
 the cluster that is being created.",
      "Type" : "String",
      "NoEcho" : "true"
    },
    "InboundTraffic" : {
      "Description" : "Allow inbound traffic to the cluster from this CIDR range.",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" : "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\
\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },
    "PortNumber" : {
      "Description" : "The port number on which the cluster accepts incoming
 connections.",
      "Type" : "Number",
      "Default" : "5439"
    }
  },
  "Conditions" : {
    "IsMultiNodeCluster" : {
      "Fn::Equals" : [{ "Ref" : "ClusterType" }, "multi-node" ]
    }
  },
  "Resources" : {
    "RedshiftCluster" : {
      "Type" : "AWS::Redshift::Cluster",
      "DependsOn" : "AttachGateway",
      "Properties" : {
        "ClusterType" : { "Ref" : "ClusterType" },
        "NumberOfNodes" : { "Fn::If" : [ "IsMultiNodeCluster",  { "Ref" :
 "NumberOfNodes" }, { "Ref" : "AWS::NoValue" }]},
        "NodeType" : { "Ref" : "NodeType" },
        "DBName" : { "Ref" : "DatabaseName" },
        "MasterUsername" : { "Ref" : "MasterUsername" },
        "MasterUserPassword" : { "Ref" : "MasterUserPassword" },
        "ClusterParameterGroupName" : { "Ref" : "RedshiftClusterParameterGroup" },
```

```
        "VpcSecurityGroupIds" : [ { "Ref" : "SecurityGroup" } ],
        "ClusterSubnetGroupName" : { "Ref" : "RedshiftClusterSubnetGroup" },
        "PubliclyAccessible" : "true",
        "Port" : { "Ref" : "PortNumber" }
      }
    },
    "RedshiftClusterParameterGroup" : {
      "Type" : "AWS::Redshift::ClusterParameterGroup",
      "Properties" : {
        "Description" : "Cluster parameter group",
        "ParameterGroupFamily" : "redshift-1.0",
        "Parameters" : [{
          "ParameterName" : "enable_user_activity_logging",
          "ParameterValue" : "true"
        }]
      }
    },
    "RedshiftClusterSubnetGroup" : {
      "Type" : "AWS::Redshift::ClusterSubnetGroup",
      "Properties" : {
        "Description" : "Cluster subnet group",
        "SubnetIds" : [ { "Ref" : "PublicSubnet" } ]
      }
    },
    "VPC" : {
      "Type" : "AWS::EC2::VPC",
      "Properties" : {
        "CidrBlock" : "10.0.0.0/16"
      }
    },
    "PublicSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "CidrBlock" : "10.0.0.0/24",
        "VpcId" : { "Ref" : "VPC" }
      }
    },
    "SecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Security group",
        "SecurityGroupIngress" : [ {
          "CidrIp" : { "Ref": "InboundTraffic" },
          "FromPort" : { "Ref" : "PortNumber" },
```

```
        "ToPort" : { "Ref" : "PortNumber" },
        "IpProtocol" : "tcp"
      } ],
      "VpcId" : { "Ref" : "VPC" }
    }
  },
  "myInternetGateway" : {
    "Type" : "AWS::EC2::InternetGateway"
  },
  "AttachGateway" : {
    "Type" : "AWS::EC2::VPCGatewayAttachment",
    "Properties" : {
      "VpcId" : { "Ref" : "VPC" },
      "InternetGatewayId" : { "Ref" : "myInternetGateway" }
    }
  },
  "PublicRouteTable" : {
    "Type" : "AWS::EC2::RouteTable",
    "Properties" : {
      "VpcId" : {
        "Ref" : "VPC"
      }
    }
  },
  "PublicRoute" : {
    "Type" : "AWS::EC2::Route",
    "DependsOn" : "AttachGateway",
    "Properties"  : {
      "RouteTableId" : {
        "Ref" : "PublicRouteTable"
      },
      "DestinationCidrBlock" : "0.0.0.0/0",
      "GatewayId" : {
        "Ref" : "myInternetGateway"
      }
    }
  },
  "PublicSubnetRouteTableAssociation" : {
    "Type" : "AWS::EC2::SubnetRouteTableAssociation",
    "Properties" : {
      "SubnetId" : {
        "Ref" : "PublicSubnet"
      },
      "RouteTableId" : {
```

```
          "Ref" : "PublicRouteTable"
        }
      }
    }
  },
  "Outputs" : {
    "ClusterEndpoint" : {
      "Description" : "Cluster endpoint",
      "Value" : { "Fn::Join" : [ ":", [ { "Fn::GetAtt" : [ "RedshiftCluster",
 "Endpoint.Address" ] }, { "Fn::GetAtt" : [ "RedshiftCluster",
 "Endpoint.Port" ] } ] ] }
    },
    "ClusterName" : {
      "Description" : "Name of cluster",
      "Value" : { "Ref" : "RedshiftCluster" }
    },
    "ParameterGroupName" : {
      "Description" : "Name of parameter group",
      "Value" : { "Ref" : "RedshiftClusterParameterGroup" }
    },
    "RedshiftClusterSubnetGroupName" : {
      "Description" : "Name of cluster subnet group",
      "Value" : { "Ref" : "RedshiftClusterSubnetGroup" }
    },
    "RedshiftClusterSecurityGroupName" : {
      "Description" : "Name of cluster security group",
      "Value" : { "Ref" : "SecurityGroup" }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  DatabaseName:
    Description: The name of the first database to be created when the cluster is
      created
    Type: String
    Default: dev
    AllowedPattern: "([a-z]|[0-9])+"
  ClusterType:
    Description: The type of cluster
```

```
    Type: String
    Default: single-node
    AllowedValues:
    - single-node
    - multi-node
  NumberOfNodes:
    Description: The number of compute nodes in the cluster. For multi-node clusters,
      the NumberOfNodes parameter must be greater than 1
    Type: Number
    Default: '1'
  NodeType:
    Description: The type of node to be provisioned
    Type: String
    Default: ds2.xlarge
    AllowedValues:
    - ds2.xlarge
    - ds2.8xlarge
    - dc1.large
    - dc1.8xlarge
  MasterUsername:
    Description: The user name that is associated with the master user account for
      the cluster that is being created
    Type: String
    Default: defaultuser
    AllowedPattern: "([a-z])([a-z]|[0-9])*"
  MasterUserPassword:
    Description: The password that is associated with the master user account for
      the cluster that is being created.
    Type: String
    NoEcho: 'true'
  InboundTraffic:
    Description: Allow inbound traffic to the cluster from this CIDR range.
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})"
    ConstraintDescription: must be a valid CIDR range of the form x.x.x.x/x.
  PortNumber:
    Description: The port number on which the cluster accepts incoming connections.
    Type: Number
    Default: '5439'
Conditions:
  IsMultiNodeCluster:
```

```
    Fn::Equals:
    - Ref: ClusterType
    - multi-node
Resources:
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    DependsOn: AttachGateway
    Properties:
      ClusterType:
        Ref: ClusterType
      NumberOfNodes:
        Fn::If:
        - IsMultiNodeCluster
        - Ref: NumberOfNodes
        - Ref: AWS::NoValue
      NodeType:
        Ref: NodeType
      DBName:
        Ref: DatabaseName
      MasterUsername:
        Ref: MasterUsername
      MasterUserPassword:
        Ref: MasterUserPassword
      ClusterParameterGroupName:
        Ref: RedshiftClusterParameterGroup
      VpcSecurityGroupIds:
      - Ref: SecurityGroup
      ClusterSubnetGroupName:
        Ref: RedshiftClusterSubnetGroup
      PubliclyAccessible: 'true'
      Port:
        Ref: PortNumber
  RedshiftClusterParameterGroup:
    Type: AWS::Redshift::ClusterParameterGroup
    Properties:
      Description: Cluster parameter group
      ParameterGroupFamily: redshift-1.0
      Parameters:
      - ParameterName: enable_user_activity_logging
        ParameterValue: 'true'
  RedshiftClusterSubnetGroup:
    Type: AWS::Redshift::ClusterSubnetGroup
    Properties:
      Description: Cluster subnet group
```

```
      SubnetIds:
      - Ref: PublicSubnet
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  PublicSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.0.0/24
      VpcId:
        Ref: VPC
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group
      SecurityGroupIngress:
      - CidrIp:
          Ref: InboundTraffic
        FromPort:
          Ref: PortNumber
        ToPort:
          Ref: PortNumber
        IpProtocol: tcp
      VpcId:
        Ref: VPC
  myInternetGateway:
    Type: AWS::EC2::InternetGateway
  AttachGateway:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId:
        Ref: VPC
      InternetGatewayId:
        Ref: myInternetGateway
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId:
        Ref: VPC
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: AttachGateway
    Properties:
```

```
      RouteTableId:
        Ref: PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId:
        Ref: myInternetGateway
  PublicSubnetRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId:
        Ref: PublicSubnet
      RouteTableId:
        Ref: PublicRouteTable
Outputs:
  ClusterEndpoint:
    Description: Cluster endpoint
    Value: !Sub "${RedshiftCluster.Endpoint.Address}:${RedshiftCluster.Endpoint.Port}"
  ClusterName:
    Description: Name of cluster
    Value:
      Ref: RedshiftCluster
  ParameterGroupName:
    Description: Name of parameter group
    Value:
      Ref: RedshiftClusterParameterGroup
  RedshiftClusterSubnetGroupName:
    Description: Name of cluster subnet group
    Value:
      Ref: RedshiftClusterSubnetGroup
  RedshiftClusterSecurityGroupName:
    Description: Name of cluster security group
    Value:
      Ref: SecurityGroup
```

另请参阅

AWS::Redshift::Cluster

Amazon RDS 模板代码段

主题

- Amazon RDS 数据库实例资源
- Amazon RDS Oracle Database 数据库实例资源

- 适用于 CIDR 范围的 Amazon RDS DBSecurityGroup 资源
- 带 Amazon EC2 安全组的 Amazon RDS DBSecurityGroup
- 多 VPC 安全组
- VPC 安全组中的 Amazon RDS 数据库实例

Amazon RDS 数据库实例资源

此示例显示了使用托管主用户密码的 Amazon RDS 数据库实例资源。有关更多信息，请参阅《Amazon RDS 用户指南》中的使用 Amazon Secrets Manager 管理密码和《Aurora 用户指南》中的使用 Amazon Secrets Manager 管理密码。由于没有指定可选的 `EngineVersion` 属性，因此会将默认引擎版本用于此数据库实例。有关默认引擎版本和其他默认设置的详细信息，请参阅 CreateDBInstance。`DBSecurityGroups` 属性向名为 `MyDbSecurityByEC2SecurityGroup` 和 MyDbSecurityByCIDRIPGroup 的 `AWS::RDS::DBSecurityGroup` 资源授予网络入口权限。有关更多信息，请参阅 AWS::RDS::DBInstance。数据库实例资源还有一个设置为 `Snapshot` 的 `DeletionPolicy` 属性。如果设置了 `Snapshot` `DeletionPolicy`，则在堆栈删除期间，Amazon CloudFormation 会首先拍摄该数据库实例的快照，然后再将其删除。

JSON

```
"MyDB" : {
 "Type" : "AWS::RDS::DBInstance",
 "Properties" : {
     "DBSecurityGroups" : [
        {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" :
 "MyDbSecurityByCIDRIPGroup"} ],
     "AllocatedStorage" : "5",
     "DBInstanceClass" : "db.t2.small",
     "Engine" : "MySQL",
     "MasterUsername" : "MyName",
     "ManageMasterUserPassword" : true,
     "MasterUserSecret" : {
        "KmsKeyId" : {"Ref" : "KMSKey"}
     }
 },
 "DeletionPolicy" : "Snapshot"
}
```

YAML

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    DBSecurityGroups:
    - Ref: MyDbSecurityByEC2SecurityGroup
    - Ref: MyDbSecurityByCIDRIPGroup
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: MySQL
    MasterUsername: MyName
    ManageMasterUserPassword: true
    MasterUserSecret:
      KmsKeyId: !Ref KMSKey
  DeletionPolicy: Snapshot
```

Amazon RDS Oracle Database 数据库实例资源

此示例创建了使用托管主用户密码的 Oracle Database 数据库实例资源。有关更多信息，请参阅《Amazon RDS 用户指南》中的使用 Amazon Secrets Manager 管理密码。此示例将 `Engine` 指定为 `oracle-ee` 并使用“自带许可”许可模式。有关 Oracle Database 数据库实例设置的详细信息，请参阅 CreateDBInstance。DBSecurityGroups 属性授权对名为 MyDbSecurityByEC2SecurityGroup 和 MyDbSecurityByCIDRIPGroup 的 `AWS::RDS::DBSecurityGroup` 资源的网络访问。有关更多信息，请参阅 AWS::RDS::DBInstance。数据库实例资源还有一个设置为 `Snapshot` 的 `DeletionPolicy` 属性。如果设置了 `Snapshot` `DeletionPolicy`，则在堆栈删除期间，Amazon CloudFormation 会首先拍摄该数据库实例的快照，然后再将其删除。

JSON

```
"MyDB" : {
 "Type" : "AWS::RDS::DBInstance",
 "Properties" : {
     "DBSecurityGroups" : [
        {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" :
 "MyDbSecurityByCIDRIPGroup"} ],
     "AllocatedStorage" : "5",
     "DBInstanceClass" : "db.t2.small",
     "Engine" : "oracle-ee",
     "LicenseModel" : "bring-your-own-license",
     "MasterUsername" : "master",
```

```
     "ManageMasterUserPassword" : true,
     "MasterUserSecret" : {
        "KmsKeyId" : {"Ref" : "KMSKey"}
     }
 },
 "DeletionPolicy" : "Snapshot"
}
```

YAML

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    DBSecurityGroups:
    - Ref: MyDbSecurityByEC2SecurityGroup
    - Ref: MyDbSecurityByCIDRIPGroup
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: oracle-ee
    LicenseModel: bring-your-own-license
    MasterUsername: master
    ManageMasterUserPassword: true
    MasterUserSecret:
      KmsKeyId: !Ref KMSKey
  DeletionPolicy: Snapshot
```

适用于 CIDR 范围的 Amazon RDS DBSecurityGroup 资源

此示例说明一个 Amazon RDS DBSecurityGroup 资源，该资源具有以 ddd.ddd.ddd.ddd/dd 格式指定的 CIDR 范围的入口授权。有关详细信息，请参阅 AWS::RDS::DBSecurityGroup 和 Ingress。

JSON

```
"MyDbSecurityByCIDRIPGroup" : {
 "Type" : "AWS::RDS::DBSecurityGroup",
 "Properties" : {
     "GroupDescription" : "Ingress for CIDRIP",
     "DBSecurityGroupIngress" : {
         "CIDRIP" : "192.168.0.0/32"
     }
 }
}
```

YAML

```
MyDbSecurityByCIDRIPGroup:
  Type: AWS::RDS::DBSecurityGroup
  Properties:
    GroupDescription: Ingress for CIDRIP
    DBSecurityGroupIngress:
      CIDRIP: "192.168.0.0/32"
```

带 Amazon EC2 安全组的 Amazon RDS DBSecurityGroup

此示例展示的 AWS::RDS::DBSecurityGroup 资源具有 MyEc2SecurityGroup 所引用 Amazon EC2 安全组中的入口授权。

要执行此操作，您需要定义一个 EC2 安全组，然后在 DBSecurityGroup 中使用 Ref 内置函数引用该 EC2 安全组。

JSON

```
"DBInstance" : {
   "Type": "AWS::RDS::DBInstance",
   "Properties": {
      "DBName"            : { "Ref" : "DBName" },
      "Engine"            : "MySQL",
      "MasterUsername"    : { "Ref" : "DBUsername" },
      "DBInstanceClass"   : { "Ref" : "DBClass" },
      "DBSecurityGroups"  : [ { "Ref" : "DBSecurityGroup" } ],
      "AllocatedStorage"  : { "Ref" : "DBAllocatedStorage" },
      "MasterUserPassword": { "Ref" : "DBPassword" }
   }
},

"DBSecurityGroup": {
   "Type": "AWS::RDS::DBSecurityGroup",
   "Properties": {
      "DBSecurityGroupIngress": {
         "EC2SecurityGroupName": {
            "Fn::GetAtt": ["WebServerSecurityGroup", "GroupName"]
         }
      },
      "GroupDescription" : "Frontend Access"
   }
```

```
},

"WebServerSecurityGroup" : {
   "Type" : "AWS::EC2::SecurityGroup",
   "Properties" : {
      "GroupDescription" : "Enable HTTP access via port 80 and SSH access",
      "SecurityGroupIngress" : [
         {"IpProtocol" : "tcp", "FromPort" : 80, "ToPort" : 80, "CidrIp" :
 "0.0.0.0/0"},
         {"IpProtocol" : "tcp", "FromPort" : 22, "ToPort" : 22, "CidrIp" : "0.0.0.0/0"}
      ]
   }
}
```

YAML

该示例提取自下面的完整示例：Drupal_Single_Instance_With_RDS.template

```
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName:
      Ref: DBName
    Engine: MySQL
    MasterUsername:
      Ref: DBUsername
    DBInstanceClass:
      Ref: DBClass
    DBSecurityGroups:
    - Ref: DBSecurityGroup
    AllocatedStorage:
      Ref: DBAllocatedStorage
    MasterUserPassword:
      Ref: DBPassword
DBSecurityGroup:
  Type: AWS::RDS::DBSecurityGroup
  Properties:
    DBSecurityGroupIngress:
      EC2SecurityGroupName:
        Ref: WebServerSecurityGroup
    GroupDescription: Frontend Access
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
```

```
    GroupDescription: Enable HTTP access via port 80 and SSH access
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 80
      ToPort: 80
      CidrIp: 0.0.0.0/0
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: 0.0.0.0/0
```

多 VPC 安全组

此示例显示的是具有 AWS::RDS::DBSecurityGroupIngress 中的多个 Amazon EC2 VPC 安全组的入口授权的 AWS::RDS::DBSecurityGroup 资源。

JSON

```
{
   "Resources" : {
      "DBinstance" : {
         "Type" : "AWS::RDS::DBInstance",
         "Properties" : {
            "AllocatedStorage" : "5",
            "DBInstanceClass" : "db.t2.small",
           "DBName" : {"Ref": "MyDBName" },
            "DBSecurityGroups" : [ { "Ref" : "DbSecurityByEC2SecurityGroup" } ],
            "DBSubnetGroupName" : { "Ref" : "MyDBSubnetGroup" },
            "Engine" : "MySQL",
           "MasterUserPassword": { "Ref" : "MyDBPassword" },
           "MasterUsername"    : { "Ref" : "MyDBUsername" }
        },
         "DeletionPolicy" : "Snapshot"
      },
      "DbSecurityByEC2SecurityGroup" : {
         "Type" : "AWS::RDS::DBSecurityGroup",
         "Properties" : {
            "GroupDescription" : "Ingress for Amazon EC2 security group",
           "EC2VpcId" : { "Ref" : "MyVPC" },
            "DBSecurityGroupIngress" : [ {
               "EC2SecurityGroupId" : "sg-b0ff1111",
               "EC2SecurityGroupOwnerId" : "111122223333"
            }, {
```

```
                "EC2SecurityGroupId" : "sg-ffd722222",
                "EC2SecurityGroupOwnerId" : "111122223333"
             } ]
          }
       }
    }
}
```

YAML

```
Resources:
  DBinstance:
    Type: AWS::RDS::DBInstance
    Properties:
      AllocatedStorage: '5'
      DBInstanceClass: db.t2.small
      DBName:
        Ref: MyDBName
      DBSecurityGroups:
      - Ref: DbSecurityByEC2SecurityGroup
      DBSubnetGroupName:
        Ref: MyDBSubnetGroup
      Engine: MySQL
      MasterUserPassword:
        Ref: MyDBPassword
      MasterUsername:
        Ref: MyDBUsername
    DeletionPolicy: Snapshot
  DbSecurityByEC2SecurityGroup:
    Type: AWS::RDS::DBSecurityGroup
    Properties:
      GroupDescription: Ingress for Amazon EC2 security group
      EC2VpcId:
        Ref: MyVPC
      DBSecurityGroupIngress:
      - EC2SecurityGroupId: sg-b0ff1111
        EC2SecurityGroupOwnerId: '111122223333'
      - EC2SecurityGroupId: sg-ffd722222
        EC2SecurityGroupOwnerId: '111122223333'
```

VPC 安全组中的 Amazon RDS 数据库实例

该示例显示一个与 Amazon EC2 VPC 安全组关联的 Amazon RDS 数据库实例。

JSON

```
{
  "DBEC2SecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties" : {
      "GroupDescription": "Open database for access",
      "SecurityGroupIngress" : [{
        "IpProtocol" : "tcp",
        "FromPort" : 3306,
        "ToPort" : 3306,
        "SourceSecurityGroupName" : { "Ref" : "WebServerSecurityGroup" }
      }]
    }
  },
  "DBInstance" : {
    "Type": "AWS::RDS::DBInstance",
    "Properties": {
      "DBName"            : { "Ref" : "DBName" },
      "Engine"            : "MySQL",
      "MultiAZ"           : { "Ref": "MultiAZDatabase" },
      "MasterUsername"    : { "Ref" : "DBUser" },
      "DBInstanceClass"   : { "Ref" : "DBClass" },
      "AllocatedStorage"  : { "Ref" : "DBAllocatedStorage" },
      "MasterUserPassword": { "Ref" : "DBPassword" },
      "VPCSecurityGroups" : [ { "Fn::GetAtt": [ "DBEC2SecurityGroup", "GroupId" ] } ]
    }
  }
}
```

YAML

```
DBEC2SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Open database for access
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 3306
      ToPort: 3306
      SourceSecurityGroupName:
        Ref: WebServerSecurityGroup
DBInstance:
```

```
  Type: AWS::RDS::DBInstance
  Properties:
    DBName:
      Ref: DBName
    Engine: MySQL
    MultiAZ:
      Ref: MultiAZDatabase
    MasterUsername:
      Ref: DBUser
    DBInstanceClass:
      Ref: DBClass
    AllocatedStorage:
      Ref: DBAllocatedStorage
    MasterUserPassword:
      Ref: DBPassword
    VPCSecurityGroups:
    - !GetAtt DBEC2SecurityGroup.GroupId
```

Route 53 模板代码段

主题

- 使用托管区名称或 ID 的 Amazon Route 53 资源记录集
- 使用 RecordSetGroup 设置加权资源记录集
- 使用 RecordSetGroup 设置别名资源记录集
- CloudFront 分配的别名资源记录集

使用托管区名称或 ID 的 Amazon Route 53 资源记录集

创建 Amazon Route 53 资源记录集时，必须指定要添加该记录集的托管区。CloudFormation 提供了两种指定托管区的方法：

- 您可以使用 `HostedZoneId` 属性明确指定托管区域。
- 您可以使用 `HostedZoneName` 属性让 CloudFormation 查找托管区。如果使用 `HostedZoneName` 属性且存在多个名称相同的托管区，则 CloudFormation 不会创建堆栈。

使用 HostedZoneId 添加 RecordSet

此示例添加了一个 Amazon Route 53 资源记录集，其中包含域名 `mysite.example.com` 的 SPF 记录（使用 `HostedZoneId` 属性指定托管区）。

JSON

```
"myDNSRecord" : {
  "Type" : "AWS::Route53::RecordSet",
  "Properties" :
  {
    "HostedZoneId" : "Z3DG6IL3SJCGPX",
    "Name" : "mysite.example.com.",
    "Type" : "SPF",
    "TTL" : "900",
    "ResourceRecords" : [ "\"v=spf1 ip4:192.168.0.1/16 -all\"" ]
  }
}
```

YAML

```
myDNSRecord:
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: Z3DG6IL3SJCGPX
    Name: mysite.example.com.
    Type: SPF
    TTL: '900'
    ResourceRecords:
    - '"v=spf1 ip4:192.168.0.1/16 -all"'
```

使用 HostedZoneName 添加 RecordSet

此示例使用 HostedZoneName 属性为域名“mysite.example.com”添加 Amazon Route 53 资源记录集，以指定托管区。

JSON

```
"myDNSRecord2" : {
            "Type" : "AWS::Route53::RecordSet",
            "Properties" : {
                "HostedZoneName" : "example.com.",
                "Name" : "mysite.example.com.",
                "Type" : "A",
                "TTL" : "900",
                "ResourceRecords" : [
                    "192.168.0.1",
                    "192.168.0.2"
```

```
                ]
            }
        }
```

YAML

```
myDNSRecord2:
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneName: example.com.
    Name: mysite.example.com.
    Type: A
    TTL: '900'
    ResourceRecords:
    - 192.168.0.1
    - 192.168.0.2
```

使用 RecordSetGroup 设置加权资源记录集

此示例使用 AWS::Route53::RecordSetGroup 为“example.com.”托管区域设置两条 CNAME 记录。RecordSets 属性中含有 "mysite.example.com" DNS 名称的 CNAME 记录集。每个记录集都含有一个标识符 (SetIdentifier) 和权重 (Weight)。路由到资源的 Internet 流量比例基于以下计算：

- Frontend One: 140/(140+60) = 140/200 = 70%
- Frontend Two: 60/(140+60) = 60/200 = 30%

有关加权资源记录集的更多信息，请参阅《Amazon Route 53 开发人员指南》中的加权路由。

JSON

```
        "myDNSOne" : {
            "Type" : "AWS::Route53::RecordSetGroup",
            "Properties" : {
                "HostedZoneName" : "example.com.",
                "Comment" : "Weighted RR for my frontends.",
                "RecordSets" : [
                  {
                    "Name" : "mysite.example.com.",
                    "Type" : "CNAME",
                    "TTL" : "900",
                    "SetIdentifier" : "Frontend One",
```

```
                  "Weight" : "140",
                  "ResourceRecords" : ["example-ec2.amazonaws.com"]
                },
                {
                  "Name" : "mysite.example.com.",
                  "Type" : "CNAME",
                  "TTL" : "900",
                  "SetIdentifier" : "Frontend Two",
                  "Weight" : "60",
                  "ResourceRecords" : ["example-ec2-larger.amazonaws.com"]
                }
                ]
            }
        }
```

YAML

```
myDNSOne:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Weighted RR for my frontends.
    RecordSets:
    - Name: mysite.example.com.
      Type: CNAME
      TTL: '900'
      SetIdentifier: Frontend One
      Weight: '140'
      ResourceRecords:
      - example-ec2.amazonaws.com
    - Name: mysite.example.com.
      Type: CNAME
      TTL: '900'
      SetIdentifier: Frontend Two
      Weight: '60'
      ResourceRecords:
      - example-ec2-larger.amazonaws.com
```

使用 RecordSetGroup 设置别名资源记录集

以下示例使用 AWS::Route53::RecordSetGroup 设置名为 example.com 的别名资源记录集，将流量路由到 ELB 版本 1（经典）负载均衡器和版本 2（应用程序或网络）负载均衡器。AliasTarget 属性使

用 GetAtt 内置函数指定 myELB LoadBalancer 的托管区 ID 和 DNS 名称。GetAtt 将根据是要将流量路由到版本 1 还是版本 2 的负载均衡器来检索 myELB 资源的不同属性：

- 版本 1 负载均衡器：CanonicalHostedZoneNameID 和 DNSName
- 版本 2 负载均衡器：CanonicalHostedZoneID 和 DNSName

有关别名资源记录集的更多信息，请参阅《Route 53 开发人员指南》中的在别名记录和非别名记录之间进行选择。

适用于版本 1 负载均衡器的 JSON

```
      "myELB" : {
        "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties" : {
            "AvailabilityZones" : [ "cn-north-1a" ],
            "Listeners" : [ {
                "LoadBalancerPort" : "80",
                "InstancePort" : "80",
                "Protocol" : "HTTP"
            } ]
        }
      },
      "myDNS" : {
        "Type" : "AWS::Route53::RecordSetGroup",
        "Properties" : {
          "HostedZoneName" : "example.com.",
          "Comment" : "Zone apex alias targeted to myELB LoadBalancer.",
          "RecordSets" : [
            {
              "Name" : "example.com.",
              "Type" : "A",
              "AliasTarget" : {
                  "HostedZoneId" : { "Fn::GetAtt" : ["myELB",
"CanonicalHostedZoneNameID"] },
                  "DNSName" : { "Fn::GetAtt" : ["myELB","DNSName"] }
              }
            }
          ]
        }
    }
```

适用于版本 1 负载均衡器的 YAML

```
myELB:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
    - "cn-north-1a"
    Listeners:
    - LoadBalancerPort: '80'
      InstancePort: '80'
      Protocol: HTTP
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Zone apex alias targeted to myELB LoadBalancer.
    RecordSets:
    - Name: example.com.
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt 'myELB.CanonicalHostedZoneNameID'
        DNSName: !GetAtt 'myELB.DNSName'
```

适用于版本 2 负载均衡器的 JSON

```
      "myELB" : {
        "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties" : {
            "Subnets" : [
                {"Ref": "SubnetAZ1"},
                {"Ref" : "SubnetAZ2"}
            ]
        }
      },
      "myDNS" : {
        "Type" : "AWS::Route53::RecordSetGroup",
        "Properties" : {
          "HostedZoneName" : "example.com.",
          "Comment" : "Zone apex alias targeted to myELB LoadBalancer.",
          "RecordSets" : [
            {
              "Name" : "example.com.",
              "Type" : "A",
```

```
              "AliasTarget" : {
                  "HostedZoneId" : { "Fn::GetAtt" : ["myELB",
 "CanonicalHostedZoneID"] },
                  "DNSName" : { "Fn::GetAtt" : ["myELB","DNSName"] }
              }
            }
          ]
        }
    }
```

适用于版本 2 负载均衡器的 YAML

```
myELB:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Subnets:
    - Ref: SubnetAZ1
    - Ref: SubnetAZ2
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Zone apex alias targeted to myELB LoadBalancer.
    RecordSets:
    - Name: example.com.
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt 'myELB.CanonicalHostedZoneID'
        DNSName: !GetAtt 'myELB.DNSName'
```

CloudFront 分配的别名资源记录集

下面的示例创建了将自定义域名指向 CloudFront 分发的别名 A 记录。myHostedZoneID 假定要么是引用同一模板中的实际 AWS::Route53::HostedZone 资源，要么是参数。myCloudFrontDistribution 指同一模板中的 AWS::CloudFront::Distribution 资源。别名记录会使用标准的 CloudFront 托管区 ID (Z2FDTNDATAQYW2)，并使用 Fn::GetAtt 自动解析分发的域名。此设置允许在不需要 IP 地址的情况下将网络流量从自定义域路由到 CloudFront 分发。

> **Note**
>
> 在创建别名资源记录集时，必须为 Z2FDTNDATAQYW2 属性指定 HostedZoneId。无法在私有区域中创建 CloudFront 的别名资源记录集。

JSON

```
{
    "myDNS": {
        "Type": "AWS::Route53::RecordSetGroup",
        "Properties": {
            "HostedZoneId": {
                "Ref": "myHostedZoneID"
            },
            "RecordSets": [
                {
                    "Name": {
                        "Ref": "myRecordSetDomainName"
                    },
                    "Type": "A",
                    "AliasTarget": {
                        "HostedZoneId": "Z2FDTNDATAQYW2",
                        "DNSName": {
                            "Fn::GetAtt": [
                                "myCloudFrontDistribution",
                                "DomainName"
                            ]
                        },
                        "EvaluateTargetHealth": false
                    }
                }
            ]
        }
    }
}
```

YAML

```
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
```

```
    HostedZoneId: !Ref myHostedZoneID
    RecordSets:
      - Name: !Ref myRecordSetDomainName
        Type: A
        AliasTarget:
          HostedZoneId: Z2FDTNDATAQYW2
          DNSName: !GetAtt
            - myCloudFrontDistribution
            - DomainName
          EvaluateTargetHealth: false
```

Amazon S3 模板代码段

可以使用这些 Amazon S3 示例模板帮助在 CloudFormation 中描述 Amazon S3 存储桶。有关更多示例，请参阅 `AWS::S3::Bucket` 资源中的示例部分。

主题

- 使用默认值创建 Amazon S3 存储桶
- 为虚拟主机创建一个带 DeletionPolicy 的 Amazon S3 存储桶
- 使用自定义域创建静态网站

使用默认值创建 Amazon S3 存储桶

此示例使用 AWS::S3::Bucket 创建带默认设置的桶。

JSON

```
"myS3Bucket" : {
      "Type" : "AWS::S3::Bucket"
      }
```

YAML

```
MyS3Bucket:
    Type: AWS::S3::Bucket
```

为虚拟主机创建一个带 **`DeletionPolicy`** 的 Amazon S3 存储桶

此示例将创建一个存储桶作为网站，并禁用阻止公有访问（为网站托管设置的存储桶需要公有读取权限）。然后，将公有存储桶策略添加到存储桶。由于此存储桶资源有一个设置为 `Retain` 的

DeletionPolicy 属性，因此 CloudFormation 不会在其删除堆栈时删除此存储桶。Output 部分使用 Fn::GetAtt 检索 S3Bucket 资源的 WebsiteURL 属性和 DomainName 属性。

> **Note**
>
> 以下示例假设 BlockPublicPolicy 和 RestrictPublicBuckets 阻止公有访问设置已在账户级别禁用。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "PublicAccessBlockConfiguration": {
                    "BlockPublicAcls": false,
                    "BlockPublicPolicy": false,
                    "IgnorePublicAcls": false,
                    "RestrictPublicBuckets": false
                },
                "WebsiteConfiguration": {
                    "IndexDocument": "index.html",
                    "ErrorDocument": "error.html"
                }
            },
            "DeletionPolicy": "Retain",
            "UpdateReplacePolicy": "Retain"
        },
        "BucketPolicy": {
            "Type": "AWS::S3::BucketPolicy",
            "Properties": {
                "PolicyDocument": {
                    "Id": "MyPolicy",
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Sid": "PublicReadForGetBucketObjects",
                            "Effect": "Allow",
                            "Principal": "*",
```

```
                        "Action": "s3:GetObject",
                        "Resource": {
                            "Fn::Join": [
                                "",
                                [
                                    "arn:aws:s3:::",
                                    {
                                        "Ref": "S3Bucket"
                                    },
                                    "/*"
                                ]
                            ]
                        }
                    }
                ]
            },
            "Bucket": {
                "Ref": "S3Bucket"
            }
        }
    }
},
"Outputs": {
    "WebsiteURL": {
        "Value": {
            "Fn::GetAtt": [
                "S3Bucket",
                "WebsiteURL"
            ]
        },
        "Description": "URL for website hosted on S3"
    },
    "S3BucketSecureURL": {
        "Value": {
            "Fn::Join": [
                "",
                [
                    "https://",
                    {
                        "Fn::GetAtt": [
                            "S3Bucket",
                            "DomainName"
                        ]
                    }
```

```
                    ]
                ]
            },
            "Description": "Name of S3 bucket to hold website content"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      PublicAccessBlockConfiguration:
        BlockPublicAcls: false
        BlockPublicPolicy: false
        IgnorePublicAcls: false
        RestrictPublicBuckets: false
      WebsiteConfiguration:
        IndexDocument: index.html
        ErrorDocument: error.html
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
  BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      PolicyDocument:
        Id: MyPolicy
        Version: 2012-10-17
        Statement:
          - Sid: PublicReadForGetBucketObjects
            Effect: Allow
            Principal: '*'
            Action: 's3:GetObject'
            Resource: !Join
              - ''
              - - 'arn:aws:s3:::'
                - !Ref S3Bucket
                - /*
      Bucket: !Ref S3Bucket
Outputs:
```

```
  WebsiteURL:
    Value: !GetAtt
      - S3Bucket
      - WebsiteURL
    Description: URL for website hosted on S3
  S3BucketSecureURL:
    Value: !Join
      - ''
      - - 'https://'
        - !GetAtt
          - S3Bucket
          - DomainName
    Description: Name of S3 bucket to hold website content
```

使用自定义域创建静态网站

可以将 Route 53 和已注册的域一起使用。以下示例假设您已在 Route 53 中为您的域创建了托管区域。该示例创建两个用于网站托管的存储桶。根存储桶托管内容，另一个存储桶将 www.*domainname*.com 请求重定向到根存储桶。该记录集将您的域名映射到 Amazon S3 端点。

您还需要添加存储桶策略，如上面的示例所示。

有关使用自定义域的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的教程：使用注册到 Route 53 的自定义域配置静态网站。

> **Note**
>
> 以下示例假设 `BlockPublicPolicy` 和 `RestrictPublicBuckets` 阻止公有访问设置已在账户级别禁用。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Mappings" : {
        "RegionMap" : {
            "us-east-1" : { "S3hostedzoneID" : "Z3AQBSTGFYJSTF", "websiteendpoint" :
 "s3-website-us-east-1.amazonaws.com" },
            "us-west-1" : { "S3hostedzoneID" : "Z2F56UZL2M1ACD", "websiteendpoint" :
 "s3-website-us-west-1.amazonaws.com" },
```

```
            "us-west-2" : { "S3hostedzoneID" : "Z3BJ6K6RIION7M", "websiteendpoint" :
"s3-website-us-west-2.amazonaws.com" },
            "eu-west-1" : { "S3hostedzoneID" : "Z1BKCTXD74EZPE", "websiteendpoint" :
"s3-website-eu-west-1.amazonaws.com" },
            "ap-southeast-1" : { "S3hostedzoneID" : "Z3O0J2DXBE1FTB",
"websiteendpoint" : "s3-website-ap-southeast-1.amazonaws.com" },
            "ap-southeast-2" : { "S3hostedzoneID" : "Z1WCIGYICN2BYD",
"websiteendpoint" : "s3-website-ap-southeast-2.amazonaws.com" },
            "ap-northeast-1" : { "S3hostedzoneID" : "Z2M4EHUR26P7ZW",
"websiteendpoint" : "s3-website-ap-northeast-1.amazonaws.com" },
            "sa-east-1" : { "S3hostedzoneID" : "Z31GFT0UA1I2HV", "websiteendpoint" :
"s3-website-sa-east-1.amazonaws.com" }
        }
    },
    "Parameters": {
        "RootDomainName": {
            "Description": "Domain name for your website (example.com)",
            "Type": "String"
        }
    },
    "Resources": {
        "RootBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName" : {"Ref":"RootDomainName"},
                "PublicAccessBlockConfiguration": {
                    "BlockPublicAcls": false,
                    "BlockPublicPolicy": false,
                    "IgnorePublicAcls": false,
                    "RestrictPublicBuckets": false
                },
                "WebsiteConfiguration": {
                    "IndexDocument":"index.html",
                    "ErrorDocument":"404.html"
                }
            }
        },
        "WWWBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": {
                    "Fn::Join": ["", ["www.", {"Ref":"RootDomainName"}]]
                },
                "AccessControl": "BucketOwnerFullControl",
```

```
                "WebsiteConfiguration": {
                    "RedirectAllRequestsTo": {
                        "HostName": {"Ref": "RootBucket"}
                    }
                }
            }
        },
        "myDNS": {
            "Type": "AWS::Route53::RecordSetGroup",
            "Properties": {
                "HostedZoneName": {
                    "Fn::Join": ["", [{"Ref": "RootDomainName"}, "."]]
                },
                "Comment": "Zone apex alias.",
                "RecordSets": [
                    {
                        "Name": {"Ref": "RootDomainName"},
                        "Type": "A",
                        "AliasTarget": {
                            "HostedZoneId": {"Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "S3hostedzoneID"]},
                            "DNSName": {"Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "websiteendpoint"]}
                        }
                    },
                    {
                        "Name": {
                            "Fn::Join": ["", ["www.", {"Ref":"RootDomainName"}]]
                        },
                        "Type": "CNAME",
                        "TTL" : "900",
                        "ResourceRecords" : [
                            {"Fn::GetAtt":["WWWBucket", "DomainName"]}
                        ]
                    }
                ]
            }
        }
    },
    "Outputs": {
        "WebsiteURL": {
            "Value": {"Fn::GetAtt": ["RootBucket", "WebsiteURL"]},
            "Description": "URL for website hosted on S3"
        }
```

```
    }
}
```

YAML

```
Parameters:
  RootDomainName:
    Description: Domain name for your website (example.com)
    Type: String
Mappings:
  RegionMap:
    us-east-1:
      S3hostedzoneID: Z3AQBSTGFYJSTF
      websiteendpoint: s3-website-us-east-1.amazonaws.com
    us-west-1:
      S3hostedzoneID: Z2F56UZL2M1ACD
      websiteendpoint: s3-website-us-west-1.amazonaws.com
    us-west-2:
      S3hostedzoneID: Z3BJ6K6RIION7M
      websiteendpoint: s3-website-us-west-2.amazonaws.com
    eu-west-1:
      S3hostedzoneID: Z1BKCTXD74EZPE
      websiteendpoint: s3-website-eu-west-1.amazonaws.com
    ap-southeast-1:
      S3hostedzoneID: Z3O0J2DXBE1FTB
      websiteendpoint: s3-website-ap-southeast-1.amazonaws.com
    ap-southeast-2:
      S3hostedzoneID: Z1WCIGYICN2BYD
      websiteendpoint: s3-website-ap-southeast-2.amazonaws.com
    ap-northeast-1:
      S3hostedzoneID: Z2M4EHUR26P7ZW
      websiteendpoint: s3-website-ap-northeast-1.amazonaws.com
    sa-east-1:
      S3hostedzoneID: Z31GFT0UA1I2HV
      websiteendpoint: s3-website-sa-east-1.amazonaws.com
Resources:
  RootBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref RootDomainName
      PublicAccessBlockConfiguration:
        BlockPublicAcls: false
        BlockPublicPolicy: false
```

```
        IgnorePublicAcls: false
        RestrictPublicBuckets: false
      WebsiteConfiguration:
        IndexDocument: index.html
        ErrorDocument: 404.html
  WWWBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub
        - www.${Domain}
        - Domain: !Ref RootDomainName
      AccessControl: BucketOwnerFullControl
      WebsiteConfiguration:
        RedirectAllRequestsTo:
          HostName: !Ref RootBucket
  myDNS:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      HostedZoneName: !Sub
        - ${Domain}.
        - Domain: !Ref RootDomainName
      Comment: Zone apex alias.
      RecordSets:
        - Name: !Ref RootDomainName
          Type: A
          AliasTarget:
            HostedZoneId: !FindInMap [ RegionMap, !Ref 'AWS::Region', S3hostedzoneID]
            DNSName: !FindInMap [ RegionMap, !Ref 'AWS::Region', websiteendpoint]
        - Name: !Sub
            - www.${Domain}
            - Domain: !Ref RootDomainName
          Type: CNAME
          TTL: 900
          ResourceRecords:
            - !GetAtt WWWBucket.DomainName
Outputs:
  WebsiteURL:
    Value: !GetAtt RootBucket.WebsiteURL
    Description: URL for website hosted on S3
```

Amazon SNS 模板代码段

此示例显示 Amazon SNS 主题资源。它需要有效的电子邮件地址。

JSON

```
"MySNSTopic" : {
    "Type" : "AWS::SNS::Topic",
    "Properties" : {
        "Subscription" : [ {
            "Endpoint" : "add valid email address",
            "Protocol" : "email"
        } ]
    }
}
```

YAML

```
MySNSTopic:
  Type: AWS::SNS::Topic
  Properties:
    Subscription:
    - Endpoint: "add valid email address"
      Protocol: email
```

Amazon SQS 模板代码段

此示例显示的是 Amazon SQS 队列。

JSON

```
"MyQueue" : {
    "Type" : "AWS::SQS::Queue",
    "Properties" : {
        "VisibilityTimeout" : "value"
    }
}
```

YAML

```
MyQueue:
  Type: AWS::SQS::Queue
  Properties:
    VisibilityTimeout: value
```

Amazon Timestream 模板代码片段

借助适用于 InfluxDB 的 Amazon Timestream，应用程序开发人员和 DevOps 团队可以在 Amazon 上轻松使用开源 API 为实时时间序列应用程序运行完全托管的 InfluxDB 数据库。您可以快速创建 InfluxDB 数据库，用来处理要求严格的时间序列工作负载。只需几个简单的 API 调用，就可以在 Amazon 上安装、迁移、操作和扩展具有自动软件修补、备份和恢复功能的 InfluxDB 数据库。此外还可以在 GitHub 中的 awslabs/amazon-timestream-tools/tree/mainline/integrations/cloudformation/timestream-influxdb 页面上找到这些示例。

主题

- 使用默认值的精简样本
- 带参数的更完整示例

这些 Amazon CloudFormation 模板将创建下列所需的资源，以成功创建、连接到和监控适用于 InfluxDB 的 Amazon Timestream 实例：

Amazon VPC

- VPC
- 一个或多个 Subnet
- InternetGateway
- RouteTable
- SecurityGroup

Amazon S3

- Bucket

Amazon Timestream

- InfluxDBInstance

使用默认值的精简样本

此示例会尽可能使用默认值部署多可用区和可公开访问的实例。

JSON

```
{
  "Metadata": {
    "AWS::CloudFormation::Interface": {
      "ParameterGroups": [
        {
          "Label": {"default": "Amazon Timestream for InfluxDB Configuration"},
          "Parameters": [
            "DbInstanceName",
            "InfluxDBPassword"
          ]
        }
      ],
      "ParameterLabels": {
        "VPCCIDR": {"default": "VPC CIDR"}
      }
    }
  },
  "Parameters": {
    "DbInstanceName": {
      "Description": "The name that uniquely identifies the DB instance when
interacting with the Amazon Timestream for InfluxDB API and CLI commands. This name
will also be a prefix included in the endpoint. DB instance names must be unique per
customer and per Region.",
      "Type": "String",
      "Default": "mydbinstance",
      "MinLength": 3,
      "MaxLength": 40,
      "AllowedPattern": "^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$"
    },
    "InfluxDBPassword": {
      "Description": "The password of the initial admin user created in InfluxDB. This
password will allow you to access the InfluxDB UI to perform various administrative
tasks and also use the InfluxDB CLI to create an operator token. These attributes will
be stored in a Secret created in Amazon Secrets Manager in your account.",
      "Type": "String",
      "NoEcho": true,
      "MinLength": 8,
      "MaxLength": 64,
      "AllowedPattern": "^[a-zA-Z0-9]+$"
    }
  },
  "Resources": {
```

```
    "VPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {"CidrBlock": "10.0.0.0/16"}
    },
    "InternetGateway": {"Type": "AWS::EC2::InternetGateway"},
    "InternetGatewayAttachment": {
      "Type": "AWS::EC2::VPCGatewayAttachment",
      "Properties": {
        "InternetGatewayId": {"Ref": "InternetGateway"},
        "VpcId": {"Ref": "VPC"}
      }
    },
    "Subnet1": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": {"Ref": "VPC"},
        "AvailabilityZone": {
          "Fn::Select": [
            0,
            {"Fn::GetAZs": ""}
          ]
        },
        "CidrBlock": {
          "Fn::Select": [
            0,
            {
              "Fn::Cidr": [
                {
                  "Fn::GetAtt": [
                    "VPC",
                    "CidrBlock"
                  ]
                },
                2,
                12
              ]
            }
          ]
        },
        "MapPublicIpOnLaunch": true
      }
    },
    "Subnet2": {
      "Type": "AWS::EC2::Subnet",
```

```
      "Properties": {
        "VpcId": {"Ref": "VPC"},
        "AvailabilityZone": {
          "Fn::Select": [
            1,
            {"Fn::GetAZs": ""}
          ]
        },
        "CidrBlock": {
          "Fn::Select": [
            1,
            {
              "Fn::Cidr": [
                {
                  "Fn::GetAtt": [
                    "VPC",
                    "CidrBlock"
                  ]
                },
                2,
                12
              ]
            }
          ]
        },
        "MapPublicIpOnLaunch": true
      }
    },
    "RouteTable": {
      "Type": "AWS::EC2::RouteTable",
      "Properties": {
        "VpcId": {"Ref": "VPC"}
      }
    },
    "DefaultRoute": {
      "Type": "AWS::EC2::Route",
      "DependsOn": "InternetGatewayAttachment",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": {"Ref": "InternetGateway"}
      }
    },
    "Subnet1RouteTableAssociation": {
```

```
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet1"}
      }
    },
    "Subnet2RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet2"}
      }
    },
    "InfluxDBSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupName": "influxdb-sg",
        "GroupDescription": "Security group allowing port 8086 ingress for InfluxDB",
        "VpcId": {"Ref": "VPC"}
      }
    },
    "InfluxDBSecurityGroupIngress": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "GroupId": {"Ref": "InfluxDBSecurityGroup"},
        "IpProtocol": "tcp",
        "CidrIp": "0.0.0.0/0",
        "FromPort": 8086,
        "ToPort": 8086
      }
    },
    "InfluxDBLogsS3Bucket": {
      "Type": "AWS::S3::Bucket",
      "DeletionPolicy": "Retain"
    },
    "InfluxDBLogsS3BucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {"Ref": "InfluxDBLogsS3Bucket"},
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "s3:PutObject",
```

```
              "Effect": "Allow",
              "Resource": {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/
*"},
              "Principal": {"Service": "timestream-influxdb.amazonaws.com"}
            },
            {
              "Action": "s3:*",
              "Effect": "Deny",
              "Resource": [
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/*"},
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}"}
              ],
              "Principal": "*",
              "Condition": {
                "Bool": {"aws:SecureTransport": false}
              }
            }
          ]
        }
      }
    },
    "DbInstance": {
      "Type": "AWS::Timestream::InfluxDBInstance",
      "DependsOn": "InfluxDBLogsS3BucketPolicy",
      "Properties": {
        "AllocatedStorage": 20,
        "DbInstanceType": "db.influx.medium",
        "Name": {"Ref": "DbInstanceName"},
        "Password": {"Ref": "InfluxDBPassword"},
        "PubliclyAccessible": true,
        "DeploymentType": "WITH_MULTIAZ_STANDBY",
        "VpcSecurityGroupIds": [
          {"Ref": "InfluxDBSecurityGroup"}
        ],
        "VpcSubnetIds": [
          {"Ref": "Subnet1"},
          {"Ref": "Subnet2"}
        ],
        "LogDeliveryConfiguration": {
          "S3Configuration": {
            "BucketName": {"Ref": "InfluxDBLogsS3Bucket"},
            "Enabled": true
          }
        }
```

```
      }
    }
  },
  "Outputs": {
    "VPC": {
      "Description": "A reference to the VPC used to create network resources",
      "Value": {"Ref": "VPC"}
    },
    "Subnets": {
      "Description": "A list of the subnets created",
      "Value": {
        "Fn::Join": [
          ",",
          [
            {"Ref": "Subnet1"},
            {"Ref": "Subnet2"}
          ]
        ]
      }
    },
    "Subnet1": {
      "Description": "A reference to the subnet in the 1st Availability Zone",
      "Value": {"Ref": "Subnet1"}
    },
    "Subnet2": {
      "Description": "A reference to the subnet in the 2nd Availability Zone",
      "Value": {"Ref": "Subnet2"}
    },
    "InfluxDBSecurityGroup": {
      "Description": "Security group with port 8086 ingress rule",
      "Value": {"Ref": "InfluxDBSecurityGroup"}
    },
    "InfluxDBLogsS3Bucket": {
      "Description": "S3 Bucket containing InfluxDB logs from the DB instance",
      "Value": {"Ref": "InfluxDBLogsS3Bucket"}
    },
    "DbInstance": {
      "Description": "A reference to the Timestream for InfluxDB DB instance",
      "Value": {"Ref": "DbInstance"}
    },
    "InfluxAuthParametersSecretArn": {
      "Description": "The Amazon Resource Name (ARN) of the Amazon Secrets Manager
secret containing the initial InfluxDB authorization parameters. The secret value is
```

```
 a JSON formatted key-value pair holding InfluxDB authorization values: organization,
 bucket, username, and password.",
      "Value": {
        "Fn::GetAtt": [
          "DbInstance",
          "InfluxAuthParametersSecretArn"
        ]
      }
    },
    "Endpoint": {
      "Description": "The endpoint URL to connect to InfluxDB",
      "Value": {
        "Fn::Join": [
          "",
          [
            "https://",
            {
              "Fn::GetAtt": [
                "DbInstance",
                "Endpoint"
              ]
            },
            ":8086"
          ]
        ]
      }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Amazon Timestream for InfluxDB Configuration"
        Parameters:
          - DbInstanceName
          - InfluxDBPassword
    ParameterLabels:
      VPCCIDR:
```

```
        default: VPC CIDR

Parameters:
  DbInstanceName:
    Description: The name that uniquely identifies the DB instance when interacting
 with the Amazon Timestream for InfluxDB API and CLI commands. This name will also be a
 prefix included in the endpoint. DB instance names must be unique per customer and per
 Region.
    Type: String
    Default: mydbinstance
    MinLength: 3
    MaxLength: 40
    AllowedPattern: ^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$
  InfluxDBPassword:
    Description: The password of the initial admin user created in InfluxDB. This
 password will allow you to access the InfluxDB UI to perform various administrative
 tasks and also use the InfluxDB CLI to create an operator token. These attributes will
 be stored in a Secret created in Amazon Secrets Manager in your account.
    Type: String
    NoEcho: true
    MinLength: 8
    MaxLength: 64
    AllowedPattern: ^[a-zA-Z0-9]+$

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  Subnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Select [0, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
      MapPublicIpOnLaunch: true
  Subnet2:
```

```
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs  '']
      CidrBlock: !Select [1, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
      MapPublicIpOnLaunch: true
  RouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
  DefaultRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref RouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  Subnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet1
  Subnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet2
  InfluxDBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "influxdb-sg"
      GroupDescription: "Security group allowing port 8086 ingress for InfluxDB"
      VpcId: !Ref VPC
  InfluxDBSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref InfluxDBSecurityGroup
      IpProtocol: tcp
      CidrIp: 0.0.0.0/0
      FromPort: 8086
      ToPort: 8086
  InfluxDBLogsS3Bucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
```

```
  InfluxDBLogsS3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref InfluxDBLogsS3Bucket
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Action: "s3:PutObject"
            Effect: Allow
            Resource: !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/*
            Principal:
              Service: timestream-influxdb.amazonaws.com
          - Action: "s3:*"
            Effect: Deny
            Resource:
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/*
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}
            Principal: "*"
            Condition:
              Bool:
                aws:SecureTransport: false
  DbInstance:
    Type: AWS::Timestream::InfluxDBInstance
    DependsOn: InfluxDBLogsS3BucketPolicy
    Properties:
      AllocatedStorage: 20
      DbInstanceType: db.influx.medium
      Name: !Ref DbInstanceName
      Password: !Ref InfluxDBPassword
      PubliclyAccessible: true
      DeploymentType: WITH_MULTIAZ_STANDBY
      VpcSecurityGroupIds:
        - !Ref InfluxDBSecurityGroup
      VpcSubnetIds:
        - !Ref Subnet1
        - !Ref Subnet2
      LogDeliveryConfiguration:
        S3Configuration:
          BucketName: !Ref InfluxDBLogsS3Bucket
          Enabled: true

Outputs:
  # Network Resources
  VPC:
```

```
    Description: A reference to the VPC used to create network resources
    Value: !Ref VPC
  Subnets:
    Description: A list of the subnets created
    Value: !Join [",", [!Ref Subnet1, !Ref Subnet2]]
  Subnet1:
    Description: A reference to the subnet in the 1st Availability Zone
    Value: !Ref Subnet1
  Subnet2:
    Description: A reference to the subnet in the 2nd Availability Zone
    Value: !Ref Subnet2
  InfluxDBSecurityGroup:
    Description: Security group with port 8086 ingress rule
    Value: !Ref InfluxDBSecurityGroup

  # Timestream for InfluxDB Resources
  InfluxDBLogsS3Bucket:
    Description: S3 Bucket containing InfluxDB logs from the DB instance
    Value: !Ref InfluxDBLogsS3Bucket
  DbInstance:
    Description: A reference to the Timestream for InfluxDB DB instance
    Value: !Ref DbInstance
  InfluxAuthParametersSecretArn:
    Description: "The Amazon Resource Name (ARN) of the Amazon Secrets Manager secret
containing the initial InfluxDB authorization parameters. The secret value is a JSON
formatted key-value pair holding InfluxDB authorization values: organization, bucket,
username, and password."
    Value: !GetAtt DbInstance.InfluxAuthParametersSecretArn
  Endpoint:
    Description: The endpoint URL to connect to InfluxDB
    Value: !Join ["", ["https://", !GetAtt DbInstance.Endpoint, ":8086"]]
```

带参数的更完整示例

此示例模板会根据提供的参数动态更改网络资源。参数包括 PubliclyAccessible 和 DeploymentType。

JSON

```
{
  "Metadata": {
    "AWS::CloudFormation::Interface": {
```

```
      "ParameterGroups": [
        {
          "Label": {"default": "Network Configuration"},
          "Parameters": ["VPCCIDR"]
        },
        {
          "Label": {"default": "Amazon Timestream for InfluxDB Configuration"},
          "Parameters": [
            "DbInstanceName",
            "InfluxDBUsername",
            "InfluxDBPassword",
            "InfluxDBOrganization",
            "InfluxDBBucket",
            "DbInstanceType",
            "DbStorageType",
            "AllocatedStorage",
            "PubliclyAccessible",
            "DeploymentType"
          ]
        }
      ],
      "ParameterLabels": {
        "VPCCIDR": {"default": "VPC CIDR"}
      }
    }
  },
  "Parameters": {
    "VPCCIDR": {
      "Description": "Please enter the IP range (CIDR notation) for the new VPC",
      "Type": "String",
      "Default": "10.0.0.0/16"
    },
    "DbInstanceName": {
      "Description": "The name that uniquely identifies the DB instance when
interacting with the Amazon Timestream for InfluxDB API and CLI commands. This name
will also be a prefix included in the endpoint. DB instance names must be unique per
customer and per Region.",
      "Type": "String",
      "Default": "mydbinstance",
      "MinLength": 3,
      "MaxLength": 40,
      "AllowedPattern": "^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$"
    },
    "InfluxDBUsername": {
```

```
      "Description": "The username of the initial admin user created in InfluxDB. Must
start with a letter and can't end with a hyphen or contain two consecutive hyphens.
For example, my-user1. This username will allow you to access the InfluxDB UI to
perform various administrative tasks and also use the InfluxDB CLI to create an
operator token. These attributes will be stored in a Secret created in Amazon Secrets
Manager in your account.",
      "Type": "String",
      "Default": "admin",
      "MinLength": 1,
      "MaxLength": 64
    },
    "InfluxDBPassword": {
      "Description": "The password of the initial admin user created in InfluxDB. This
password will allow you to access the InfluxDB UI to perform various administrative
tasks and also use the InfluxDB CLI to create an operator token. These attributes will
be stored in a Secret created in Amazon Secrets Manager in your account.",
      "Type": "String",
      "NoEcho": true,
      "MinLength": 8,
      "MaxLength": 64,
      "AllowedPattern": "^[a-zA-Z0-9]+$"
    },
    "InfluxDBOrganization": {
      "Description": "The name of the initial organization for the initial admin user
in InfluxDB. An InfluxDB organization is a workspace for a group of users.",
      "Type": "String",
      "Default": "org",
      "MinLength": 1,
      "MaxLength": 64
    },
    "InfluxDBBucket": {
      "Description": "The name of the initial InfluxDB bucket. All InfluxDB data
is stored in a bucket. A bucket combines the concept of a database and a retention
period (the duration of time that each data point persists). A bucket belongs to an
organization.",
      "Type": "String",
      "Default": "bucket",
      "MinLength": 2,
      "MaxLength": 64,
      "AllowedPattern": "^[^_\\\"][^\\\"]*$"
    },
    "DeploymentType": {
      "Description": "Specifies whether the Timestream for InfluxDB is deployed as
Single-AZ or with a MultiAZ Standby for High availability",
```

```
      "Type": "String",
      "Default": "WITH_MULTIAZ_STANDBY",
      "AllowedValues": [
        "SINGLE_AZ",
        "WITH_MULTIAZ_STANDBY"
      ]
    },
    "AllocatedStorage": {
      "Description": "The amount of storage to allocate for your DB storage type in GiB
(gibibytes).",
      "Type": "Number",
      "Default": 400,
      "MinValue": 20,
      "MaxValue": 16384
    },
    "DbInstanceType": {
      "Description": "The Timestream for InfluxDB DB instance type to run InfluxDB
on.",
      "Type": "String",
      "Default": "db.influx.medium",
      "AllowedValues": [
        "db.influx.medium",
        "db.influx.large",
        "db.influx.xlarge",
        "db.influx.2xlarge",
        "db.influx.4xlarge",
        "db.influx.8xlarge",
        "db.influx.12xlarge",
        "db.influx.16xlarge"
      ]
    },
    "DbStorageType": {
      "Description": "The Timestream for InfluxDB DB storage type to read and write
InfluxDB data.",
      "Type": "String",
      "Default": "InfluxIOIncludedT1",
      "AllowedValues": [
        "InfluxIOIncludedT1",
        "InfluxIOIncludedT2",
        "InfluxIOIncludedT3"
      ]
    },
    "PubliclyAccessible": {
```

```
      "Description": "Configures the DB instance with a public IP to facilitate
access.",
      "Type": "String",
      "Default": true,
      "AllowedValues": [
        true,
        false
      ]
    }
  },
  "Conditions": {
    "IsMultiAZ": {
      "Fn::Equals": [
        {"Ref": "DeploymentType"},
        "WITH_MULTIAZ_STANDBY"
      ]
    },
    "IsPublic": {
      "Fn::Equals": [
        {"Ref": "PubliclyAccessible"},
        true
      ]
    }
  },
  "Resources": {
    "VPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": {"Ref": "VPCCIDR"}
      }
    },
    "InternetGateway": {
      "Type": "AWS::EC2::InternetGateway",
      "Condition": "IsPublic"
    },
    "InternetGatewayAttachment": {
      "Type": "AWS::EC2::VPCGatewayAttachment",
      "Condition": "IsPublic",
      "Properties": {
        "InternetGatewayId": {"Ref": "InternetGateway"},
        "VpcId": {"Ref": "VPC"}
      }
    },
    "Subnet1": {
```

```
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": {"Ref": "VPC"},
        "AvailabilityZone": {
          "Fn::Select": [
            0,
            {"Fn::GetAZs": ""}
          ]
        },
        "CidrBlock": {
          "Fn::Select": [
            0,
            {
              "Fn::Cidr": [
                {
                  "Fn::GetAtt": [
                    "VPC",
                    "CidrBlock"
                  ]
                },
                2,
                12
              ]
            }
          ]
        },
        "MapPublicIpOnLaunch": {
          "Fn::If": [
            "IsPublic",
            true,
            false
          ]
        }
      }
    },
    "Subnet2": {
      "Type": "AWS::EC2::Subnet",
      "Condition": "IsMultiAZ",
      "Properties": {
        "VpcId": {"Ref": "VPC"},
        "AvailabilityZone": {
          "Fn::Select": [
            1,
            {"Fn::GetAZs": ""}
```

```
          ]
        },
        "CidrBlock": {
          "Fn::Select": [
            1,
            {
              "Fn::Cidr": [
                {
                  "Fn::GetAtt": [
                    "VPC",
                    "CidrBlock"
                  ]
                },
                2,
                12
              ]
            }
          ]
        },
        "MapPublicIpOnLaunch": {
          "Fn::If": [
            "IsPublic",
            true,
            false
          ]
        }
      }
    },
    "RouteTable": {
      "Type": "AWS::EC2::RouteTable",
      "Properties": {
        "VpcId": {"Ref": "VPC"}
      }
    },
    "DefaultRoute": {
      "Type": "AWS::EC2::Route",
      "Condition": "IsPublic",
      "DependsOn": "InternetGatewayAttachment",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": {"Ref": "InternetGateway"}
      }
    },
```

```
    "Subnet1RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet1"}
      }
    },
    "Subnet2RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Condition": "IsMultiAZ",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet2"}
      }
    },
    "InfluxDBSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupName": "influxdb-sg",
        "GroupDescription": "Security group allowing port 8086 ingress for InfluxDB",
        "VpcId": {"Ref": "VPC"}
      }
    },
    "InfluxDBSecurityGroupIngress": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "GroupId": {"Ref": "InfluxDBSecurityGroup"},
        "IpProtocol": "tcp",
        "CidrIp": "0.0.0.0/0",
        "FromPort": 8086,
        "ToPort": 8086
      }
    },
    "InfluxDBLogsS3Bucket": {
      "Type": "AWS::S3::Bucket",
      "DeletionPolicy": "Retain"
    },
    "InfluxDBLogsS3BucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {"Ref": "InfluxDBLogsS3Bucket"},
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
```

```
            {
              "Action": "s3:PutObject",
              "Effect": "Allow",
              "Resource": {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/
*"},
              "Principal": {"Service": "timestream-influxdb.amazonaws.com"}
            },
            {
              "Action": "s3:*",
              "Effect": "Deny",
              "Resource": [
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/*"},
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}"}
              ],
              "Principal": "*",
              "Condition": {
                "Bool": {"aws:SecureTransport": false}
              }
            }
          ]
        }
      }
    },
    "DbInstance": {
      "Type": "AWS::Timestream::InfluxDBInstance",
      "DependsOn": "InfluxDBLogsS3BucketPolicy",
      "Properties": {
        "DbStorageType": {"Ref": "DbStorageType"},
        "AllocatedStorage": {"Ref": "AllocatedStorage"},
        "DbInstanceType": {"Ref": "DbInstanceType"},
        "Name": {"Ref": "DbInstanceName"},
        "Username": {"Ref": "InfluxDBUsername"},
        "Password": {"Ref": "InfluxDBPassword"},
        "Organization": {"Ref": "InfluxDBOrganization"},
        "Bucket": {"Ref": "InfluxDBBucket"},
        "PubliclyAccessible": {
          "Fn::If": [
            "IsPublic",
            true,
            false
          ]
        },
        "DeploymentType": {"Ref": "DeploymentType"},
        "VpcSecurityGroupIds": [
```

```
          {"Ref": "InfluxDBSecurityGroup"}
        ],
        "VpcSubnetIds": {
          "Fn::If": [
            "IsMultiAZ",
            [
              {"Ref": "Subnet1"},
              {"Ref": "Subnet2"}
            ],
            [
              {"Ref": "Subnet1"}
            ]
          ]
        },
        "LogDeliveryConfiguration": {
          "S3Configuration": {
            "BucketName": {"Ref": "InfluxDBLogsS3Bucket"},
            "Enabled": true
          }
        }
      }
    }
  },
  "Outputs": {
    "VPC": {
      "Description": "A reference to the VPC used to create network resources",
      "Value": {"Ref": "VPC"}
    },
    "Subnets": {
      "Description": "A list of the subnets created",
      "Value": {
        "Fn::If": [
          "IsMultiAZ",
          {
            "Fn::Join": [
              ",",
              [
                {"Ref": "Subnet1"},
                {"Ref": "Subnet2"}
              ]
            ]
          },
          {"Ref": "Subnet1"}
        ]
```

```
      }
    },
    "Subnet1": {
      "Description": "A reference to the subnet in the 1st Availability Zone",
      "Value": {"Ref": "Subnet1"}
    },
    "Subnet2": {
      "Condition": "IsMultiAZ",
      "Description": "A reference to the subnet in the 2nd Availability Zone",
      "Value": {"Ref": "Subnet2"}
    },
    "InfluxDBSecurityGroup": {
      "Description": "Security group with port 8086 ingress rule",
      "Value": {"Ref": "InfluxDBSecurityGroup"}
    },
    "InfluxDBLogsS3Bucket": {
      "Description": "S3 Bucket containing InfluxDB logs from the DB instance",
      "Value": {"Ref": "InfluxDBLogsS3Bucket"}
    },
    "DbInstance": {
      "Description": "A reference to the Timestream for InfluxDB DB instance",
      "Value": {"Ref": "DbInstance"}
    },
    "InfluxAuthParametersSecretArn": {
      "Description": "The Amazon Resource Name (ARN) of the Amazon Secrets Manager
secret containing the initial InfluxDB authorization parameters. The secret value is
a JSON formatted key-value pair holding InfluxDB authorization values: organization,
bucket, username, and password.",
      "Value": {
        "Fn::GetAtt": [
          "DbInstance",
          "InfluxAuthParametersSecretArn"
        ]
      }
    },
    "Endpoint": {
      "Description": "The endpoint URL to connect to InfluxDB",
      "Value": {
        "Fn::Join": [
          "",
          [
            "https://",
            {
              "Fn::GetAtt": [
```

```
                "DbInstance",
                "Endpoint"
              ]
            },
            ":8086"
          ]
        ]
      }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Network Configuration"
        Parameters:
          - VPCCIDR
      -
        Label:
          default: "Amazon Timestream for InfluxDB Configuration"
        Parameters:
          - DbInstanceName
          - InfluxDBUsername
          - InfluxDBPassword
          - InfluxDBOrganization
          - InfluxDBBucket
          - DbInstanceType
          - DbStorageType
          - AllocatedStorage
          - PubliclyAccessible
          - DeploymentType
    ParameterLabels:
      VPCCIDR:
        default: VPC CIDR

Parameters:
  # Network Configuration
  VPCCIDR:
```

```
    Description: Please enter the IP range (CIDR notation) for the new VPC
    Type: String
    Default: 10.0.0.0/16
  # Timestream for InfluxDB Configuration
  DbInstanceName:
    Description: The name that uniquely identifies the DB instance when interacting
with the Amazon Timestream for InfluxDB API and CLI commands. This name will also be a
prefix included in the endpoint. DB instance names must be unique per customer and per
Region.
    Type: String
    Default: mydbinstance
    MinLength: 3
    MaxLength: 40
    AllowedPattern: ^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$
  # InfluxDB initial user configurations
  InfluxDBUsername:
    Description: The username of the initial admin user created in InfluxDB. Must start
with a letter and can't end with a hyphen or contain two consecutive hyphens. For
example, my-user1. This username will allow you to access the InfluxDB UI to perform
various administrative tasks and also use the InfluxDB CLI to create an operator
token. These attributes will be stored in a Secret created in Amazon Secrets Manager
in your account.
    Type: String
    Default: admin
    MinLength: 1
    MaxLength: 64
  InfluxDBPassword:
    Description: The password of the initial admin user created in InfluxDB. This
password will allow you to access the InfluxDB UI to perform various administrative
tasks and also use the InfluxDB CLI to create an operator token. These attributes will
be stored in a Secret created in Amazon in your account.
    Type: String
    NoEcho: true
    MinLength: 8
    MaxLength: 64
    AllowedPattern: ^[a-zA-Z0-9]+$
  InfluxDBOrganization:
    Description: The name of the initial organization for the initial admin user in
InfluxDB. An InfluxDB organization is a workspace for a group of users.
    Type: String
    Default: org
    MinLength: 1
    MaxLength: 64
  InfluxDBBucket:
```

```
    Description: The name of the initial InfluxDB bucket. All InfluxDB data is stored
in a bucket. A bucket combines the concept of a database and a retention period (the
duration of time that each data point persists). A bucket belongs to an organization.
    Type: String
    Default: bucket
    MinLength: 2
    MaxLength: 64
    AllowedPattern: ^[^_\"][^\"]*$
  DeploymentType:
    Description: Specifies whether the Timestream for InfluxDB is deployed as Single-AZ
or with a MultiAZ Standby for High availability
    Type: String
    Default: WITH_MULTIAZ_STANDBY
    AllowedValues:
      - SINGLE_AZ
      - WITH_MULTIAZ_STANDBY
  AllocatedStorage:
    Description: The amount of storage to allocate for your DB storage type in GiB
(gibibytes).
    Type: Number
    Default: 400
    MinValue: 20
    MaxValue: 16384
  DbInstanceType:
    Description: The Timestream for InfluxDB DB instance type to run InfluxDB on.
    Type: String
    Default: db.influx.medium
    AllowedValues:
      - db.influx.medium
      - db.influx.large
      - db.influx.xlarge
      - db.influx.2xlarge
      - db.influx.4xlarge
      - db.influx.8xlarge
      - db.influx.12xlarge
      - db.influx.16xlarge
  DbStorageType:
    Description: The Timestream for InfluxDB DB storage type to read and write InfluxDB
data.
    Type: String
    Default: InfluxIOIncludedT1
    AllowedValues:
      - InfluxIOIncludedT1
      - InfluxIOIncludedT2
```

```
      - InfluxIOIncludedT3
  PubliclyAccessible:
    Description: Configures the DB instance with a public IP to facilitate access.
    Type: String
    Default: true
    AllowedValues:
      - true
      - false

Conditions:
  IsMultiAZ: !Equals [!Ref DeploymentType, WITH_MULTIAZ_STANDBY]
  IsPublic: !Equals [!Ref PubliclyAccessible, true]

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VPCCIDR
  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Condition: IsPublic
  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Condition: IsPublic
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  Subnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Select [0, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
      MapPublicIpOnLaunch: !If [IsPublic, true, false]
  Subnet2:
    Type: AWS::EC2::Subnet
    Condition: IsMultiAZ
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs  '']
      CidrBlock: !Select [1, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
      MapPublicIpOnLaunch: !If [IsPublic, true, false]
  RouteTable:
    Type: AWS::EC2::RouteTable
```

```
    Properties:
      VpcId: !Ref VPC
  DefaultRoute:
    Type: AWS::EC2::Route
    Condition: IsPublic
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref RouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  Subnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet1
  Subnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Condition: IsMultiAZ
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet2
  InfluxDBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "influxdb-sg"
      GroupDescription: "Security group allowing port 8086 ingress for InfluxDB"
      VpcId: !Ref VPC
  InfluxDBSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref InfluxDBSecurityGroup
      IpProtocol: tcp
      CidrIp: 0.0.0.0/0
      FromPort: 8086
      ToPort: 8086
  InfluxDBLogsS3Bucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
  InfluxDBLogsS3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref InfluxDBLogsS3Bucket
      PolicyDocument:
        Version: '2012-10-17'
```

```
        Statement:
          - Action: "s3:PutObject"
            Effect: Allow
            Resource: !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/*
            Principal:
              Service: timestream-influxdb.amazonaws.com
          - Action: "s3:*"
            Effect: Deny
            Resource:
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/*
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}
            Principal: "*"
            Condition:
              Bool:
                aws:SecureTransport: false
  DbInstance:
    Type: AWS::Timestream::InfluxDBInstance
    DependsOn: InfluxDBLogsS3BucketPolicy
    Properties:
      DbStorageType: !Ref DbStorageType
      AllocatedStorage: !Ref AllocatedStorage
      DbInstanceType: !Ref DbInstanceType
      Name: !Ref DbInstanceName
      Username: !Ref InfluxDBUsername
      Password: !Ref InfluxDBPassword
      Organization: !Ref InfluxDBOrganization
      Bucket: !Ref InfluxDBBucket
      PubliclyAccessible: !If [IsPublic, true, false]
      DeploymentType: !Ref DeploymentType
      VpcSecurityGroupIds:
        - !Ref InfluxDBSecurityGroup
      VpcSubnetIds: !If
        - IsMultiAZ
        -
          - !Ref Subnet1
          - !Ref Subnet2
        -
          - !Ref Subnet1
      LogDeliveryConfiguration:
        S3Configuration:
          BucketName: !Ref InfluxDBLogsS3Bucket
          Enabled: true

Outputs:
```

```
  # Network Resources
  VPC:
    Description: A reference to the VPC used to create network resources
    Value: !Ref VPC
  Subnets:
    Description: A list of the subnets created
    Value: !If
      - IsMultiAZ
      - !Join [",", [!Ref Subnet1, !Ref Subnet2]]
      - !Ref Subnet1
  Subnet1:
    Description: A reference to the subnet in the 1st Availability Zone
    Value: !Ref Subnet1
  Subnet2:
    Condition: IsMultiAZ
    Description: A reference to the subnet in the 2nd Availability Zone
    Value: !Ref Subnet2
  InfluxDBSecurityGroup:
    Description: Security group with port 8086 ingress rule
    Value: !Ref InfluxDBSecurityGroup

  # Timestream for InfluxDB Resources
  InfluxDBLogsS3Bucket:
    Description: S3 Bucket containing InfluxDB logs from the DB instance
    Value: !Ref InfluxDBLogsS3Bucket
  DbInstance:
    Description: A reference to the Timestream for InfluxDB DB instance
    Value: !Ref DbInstance
  InfluxAuthParametersSecretArn:
    Description: "The Amazon Resource Name (ARN) of the Amazon Secrets Manager secret
containing the initial InfluxDB authorization parameters. The secret value is a JSON
formatted key-value pair holding InfluxDB authorization values: organization, bucket,
username, and password."
    Value: !GetAtt DbInstance.InfluxAuthParametersSecretArn
  Endpoint:
    Description: The endpoint URL to connect to InfluxDB
    Value: !Join ["", ["https://", !GetAtt DbInstance.Endpoint, ":8086"]]
```

使用 CloudFormation 部署基于 Windows 的堆栈

本页面提供指向基于 Windows 的部署中常用 CloudFormation 资源技术参考文档的链接。

CloudFormation 支持通过 Infrastructure as Code (IaC) 部署和管理 MicrosoftWindows 堆栈。您可以使用 CloudFormation 自动预置基于 Windows 的 EC2 实例、Amazon RDS 上的 SQL Server，以及通过 Amazon Directory Service 部署的 Microsoft Active Directory。

Amazon 提供专为 Windows 平台设计的预配置亚马逊机器映像（AMI），可帮助您在 Amazon EC2 上快速部署应用程序。这些 AMI 包括默认的 Microsoft 设置和 Amazon 特定的自定义设置。使用 CloudFormation，您可以选择适当的 AMI，启动实例，然后使用远程桌面连接访问该实例，就如同访问任何其他 Windows Server 一样。AMI 包含必要的软件组件，包括 EC2Launch（版本因 Windows Server 版本而异）、Amazon Systems Manager、CloudFormation、Amazon Tools for PowerShell 以及各种网络、存储和图形驱动程序，以确保最佳性能并与 Amazon 服务兼容。有关更多信息，请参阅 Amazon Windows AMI Reference。

CloudFormation 还支持软件配置工具，例如 `UserData` 脚本，这些工具可以在 EC2 实例首次启动时运行 PowerShell 或批处理命令。它还提供帮助程序脚本（`cfn-init`、`cfn-signal`、`cfn-get-metadata` 和 `cfn-hup`），并支持用于管理 Windows 实例上的包、文件和服务的 `AWS::CloudFormation::Init` 元数据。

对于企业环境，CloudFormation 支持域加入、通过 EC2 许可模式管理 Windows 许可证以及使用 Amazon Secrets Manager 进行安全凭证处理。结合版本控制的模板和可重复的部署，CloudFormation 可帮助组织跨多个 Amazon Web Services 区域和账户维护一致、安全和可扩展的 Windows 环境。

有关基于 Windows 的部署中常用 CloudFormation 资源的详细信息，请参阅以下技术参考主题。

资源类型	说明
AWS::EC2::Instance	用于启动 Windows EC2 实例。
AWS::EC2::SecurityGroup	为 Windows 工作负载定义防火墙规则。
AWS::AutoScaling::AutoScalingGroup AWS::EC2::LaunchTemplate	用于扩展 Windows EC2 实例。
AWS::DirectoryService::MicrosoftAD	用于部署 Microsoft Active Directory。
AWS::FSx::FileSystem	用于部署 FSx for Windows File Server。
AWS::RDS::DBInstance	用于在 Amazon RDS 上预置 SQL Server。

资源类型	说明
AWS::CloudFormation::Init	在 EC2 元数据中用于配置实例。 有关更多信息，请参阅 引导基于 Windows 的 CloudFormation 堆栈。
AWS::SecretsManager::Secret	用于安全地管理凭证和 Windows 密码。
AWS::SSM::Parameter	用于安全地存储配置值。
AWS::IAM::InstanceProfile AWS::IAM::Role	用于向在 EC2 实例上运行的应用程序授予权限。

引导基于 Windows 的 CloudFormation 堆栈

本主题将介绍如何引导 Windows 堆栈并排除堆栈创建问题。

主题

- EC2 实例中的用户数据功能
- CloudFormation 帮助程序脚本
- 启动 Windows 堆栈的示例
- 转义文件路径中的 Windows 反斜杠
- 管理 Windows 服务
- 解决堆栈创建故障问题

EC2 实例中的用户数据功能

使用 Amazon EC2 的用户数据功能，可以在启动 EC2 实例时将脚本或配置信息传递至实例。

对于 Windows EC2 实例：

- 您可以使用批处理脚本（通过 `<script>` 标签）或 PowerShell 脚本（通过 `<powershell>` 标签）。
- 脚本执行由 EC2Launch 处理。

> **Important**
>
> 如果您要创建自己的 Windows AMI 以用于 CloudFormation，请确保正确配置 EC2Launch v2。CloudFormation 引导工具需要使用 EC2Launch v2 才能在创建堆栈期间正确初始化和配置 Windows 实例。有关更多信息，请参阅《Amazon EC2 用户指南》中的使用 EC2Launch v2 代理在 EC2 Windows 实例启动期间执行任务。
> 有关 Amazon Windows AMI 的信息，请参阅《Amazon Windows AMI Reference》。

CloudFormation 帮助程序脚本

帮助程序脚本是一种实用工具，用于在引导过程中配置实例。它们可与 Amazon EC2 用户数据功能配合使用，提供强大的配置选项。

CloudFormation 提供以下 Python 帮助程序脚本，用于在您在堆栈中创建的 Amazon EC2 实例上安装软件和启动服务：

- `cfn-init` – 用于检索和解释资源元数据、安装软件包、创建文件和启动服务。
- `cfn-signal` – 用于通过 `CreationPolicy` 发送信号，因此您能够在先决条件资源或应用程序准备就绪时同步堆栈中的其他资源。
- `cfn-get-metadata` – 用于检索资源的元数据或特定密钥的路径。
- `cfn-hup` – 用于检查元数据更新，并在检测到更改时，执行自定义挂钩。

您可以从您的模板中直接调用脚本。该脚本可与在同一模板中定义的资源元数据共同运行。脚本可在堆栈创建过程中的 Amazon EC2 实例上运行。

有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 CloudFormation 帮助程序脚本参考。

启动 Windows 堆栈的示例

让我们检查一下执行以下操作的 Windows Server 模板中的示例代码段：

- 从 Windows Server 2022 AMI 启动名为 `TestInstance` 的 EC2 实例。
- 创建一个简单的测试文件来验证 `cfn-init` 是否正常工作。
- 配置 `cfn-hup` 以进行持续的配置管理。
- 使用 `CreationPolicy` 确保实例发出成功完成的信号。

cfn-init 帮助程序脚本可用于根据模板中 AWS::CloudFormation::Init 资源的信息来执行上述各种操作。

AWS::CloudFormation::Init 部分名为 TestInstance，以以下声明开头。

```
TestInstance:
  Type: AWS::EC2::Instance
  Metadata:
    AWS::CloudFormation::Init:
      configSets:
        default:
          - create_files
          - start_services
```

在此之后，声明 AWS::CloudFormation::Init 的 files 部分。

```
      create_files:
        files:
          c:\cfn\test.txt:
            content: !Sub |
              Hello from ${AWS::StackName}
          c:\cfn\cfn-hup.conf:
            content: !Sub |
              [main]
              stack=${AWS::StackName}
              region=${AWS::Region}
              interval=2
          c:\cfn\hooks.d\cfn-auto-reloader.conf:
            content: !Sub |
              [cfn-auto-reloader-hook]
              triggers=post.update
              path=Resources.TestInstance.Metadata.AWS::CloudFormation::Init
              action=cfn-init.exe -v -s ${AWS::StackName} -r TestInstance -c default --
region ${AWS::Region}
```

将在此处创建这三个文件，并将其置于服务器实例上的 C:\cfn 目录中：

- test.txt，一个简单的测试文件，用于验证 cfn-init 是否正常工作以及是否可以创建包含动态内容的文件。
- cfn-hup.conf，cfn-hup 的配置文件，检查间隔为 2 分钟。

- cfn-auto-reloader.conf 为挂钩的配置文件，当 AWS::CloudFormation::Init 中的元数据发生变化时，cfn-hup 将用其初始化更新（调用 cfn-init）。

接下来是 start_services 部分，它配置 Windows 服务。

```
      start_services:
        services:
          windows:
            cfn-hup:
              enabled: true
              ensureRunning: true
              files:
                - c:\cfn\cfn-hup.conf
                - c:\cfn\hooks.d\cfn-auto-reloader.conf
```

本节确保 cfn-hup 服务已启动，并在配置文件经修改时自动重启。该服务会监控 CloudFormation 元数据的更改，并在检测到更新时重新运行 cfn-init。

下面是 Properties 部分。

```
TestInstance:
  Type: AWS::EC2::Instance
  CreationPolicy:
    ResourceSignal:
      Timeout: PT20M
  Metadata:
    AWS::CloudFormation::Init:
      # ... metadata configuration ...
  Properties:
    InstanceType: t2.large
    ImageId: '{{resolve:ssm:/aws/service/ami-windows-latest/Windows_Server-2022-
English-Full-Base}}'
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
    KeyName: !Ref KeyPairName
    UserData:
      Fn::Base64: !Sub |
        <powershell>
        cfn-init.exe -v -s ${AWS::StackName} -r TestInstance -c default --region
 ${AWS::Region}
        cfn-signal.exe -e $lastexitcode --stack ${AWS::StackName} --resource
 TestInstance --region ${AWS::Region}
```

```
        </powershell>
```

在本部分中，UserData 属性包含一个 PowerShell 脚本，该脚本通过将 EC2Launch 用 <powershell> 标记括起得以执行。该脚本使用 default configSet 运行 cfn-init，然后使用 cfn-signal 将退出代码报告给 CloudFormation。CreationPolicy 用于确保在堆栈创建完成之前实例已正确配置。

ImageId 属性使用 Systems Manager Parameter Store 公共参数自动检索最新的 Windows Server 2022 AMI ID。此方法无需特定于区域的 AMI 映射，并确保您始终获取最新的 AMI。使用 Systems Manager 参数作为 AMI ID 是维护最新 AMI 引用的最佳实践。如果您计划连接到实例，请确保 SecurityGroupIds 属性引用允许 RDP 访问的安全组。

CreationPolicy 声明为资源属性的一部分，并指定超时期限。用户数据中的 cfn-signal 命令会在实例配置完成时发出信号：

```
TestInstance:
  Type: AWS::EC2::Instance
  CreationPolicy:
    ResourceSignal:
      Timeout: PT20M
  Properties:
    # ... other properties ...
```

由于引导过程很少，仅创建文件并启动服务，因此 CreationPolicy 会等待 20 分钟 (PT20M) 后才会超时。超时时间采用 ISO 8601 持续时间格式指定。请注意，Windows 实例的启动时间通常比 Linux 实例更长，因此请进行全面测试以确定最符合您需求的超时值。

如果一切顺利，CreationPolicy 将成功完成，您可以使用其公有 IP 地址访问 Windows Server 实例。堆栈创建完成后，实例 ID 和公有 IP 地址将显示在 CloudFormation 控制台的输出选项卡中。

```
Outputs:
  InstanceId:
    Value: !Ref TestInstance
    Description: Instance ID of the Windows Server
  PublicIP:
    Value: !GetAtt TestInstance.PublicIp
    Description: Public IP address of the Windows Server
```

您还可以手动验证引导是否正确运行，方法是通过 RDP 连接到实例并检查文件 C:\cfn\test.txt 是否存在且包含预期内容。有关连接 Windows 实例的更多信息，请参阅《Amazon EC2 用户指南》中的使用 RDP 连接到 Windows 实例。

转义文件路径中的 Windows 反斜杠

在 CloudFormation 模板中引用 Windows 路径时，请务必根据所使用的模板格式正确转义反斜杠 (\)。

- 对于 JSON 模板，您必须在 Windows 文件路径中使用双反斜杠，因为 JSON 将反斜杠视为转义字符。第一个反斜杠会转义第二个反斜杠，因此会被解释为单个字面反斜杠。

```
"commands" : {
  "1-extract" : {
    "command" : "C:\\SharePoint\\SharePointFoundation2010.exe /extract:C:\\SharePoint
\\SPF2010 /quiet /log:C:\\SharePoint\\SharePointFoundation2010-extract.log"
  }
}
```

- 对于 YAML 模板，单反斜杠通常足够了。

```
commands:
  1-extract:
    command: C:\SharePoint\SharePointFoundation2010.exe /extract:C:\SharePoint
\SPF2010 /quiet /log:C:\SharePoint\SharePointFoundation2010-extract.log
```

管理 Windows 服务

除了使用 `windows` 密钥而不使用 `sysvinit` 之外，管理 Windows 服务的方式与管理 Linux 服务的方式相同。以下示例将启动 `cfn-hup` 服务，将其设置为“自动”，并在 `cfn-init` 修改 `c:\cfn\cfn-hup.conf` 或 `c:\cfn\hooks.d\cfn-auto-reloader.conf` 配置文件时重启服务。

```
      services:
        windows:
          cfn-hup:
            enabled: true
            ensureRunning: true
            files:
              - c:\cfn\cfn-hup.conf
              - c:\cfn\hooks.d\cfn-auto-reloader.conf
```

您可以使用名称（而不是显示名称）来引用服务，以相同方式管理其他 Windows 服务。

解决堆栈创建故障问题

如果您的堆栈在创建过程中出现问题，默认行为将为失败时回滚。正常情况下，这属于一项良好默认，因为它免除了不必要的收费，但是它会让调试堆栈创建失败的原因变得非常困难。

要在使用 CloudFormation 控制台创建或更新堆栈时关闭此行为，请在堆栈故障选项下选择保留成功预置的资源选项。有关更多信息，请参阅 选择预置资源时如何处理失败。通过此操作，您可以登录您的实例，并查看日志文件，以精确找出运行启动脚本时出现的问题。

须查看的重要日志文件包括：

- EC2 配置日志位于：`%ProgramData%\Amazon\EC2Launch\log\agent.log`
- `C:\cfn\log\cfn-init.log` 处的 cfn-init 日志（检查退出代码和错误消息以了解具体故障点）

有关更多日志，请参阅《Amazon EC2 用户指南》中的以下主题：

- EC2Launch 目录结构
- EC2Launch v2 目录结构

有关排查引导问题的更多信息，请参阅如何排查无法在具有 Windows 实例的 CloudFormation 堆栈中引导的帮助程序脚本的问题？。

使用 CloudFormation 提供的资源类型扩展模板的功能

CloudFormation 提供了多种资源类型，您可以在堆栈模板中使用这些资源类型来扩展其功能，超越简单的堆栈模板。

这些资源类型包括：

资源类型	说明	文档
自定义资源	您可以利用此 `AWS::CloudFormation::CustomResource` 资源类型来创建自定义资源，从而执行特定的预置任务或包含不能通过 CloudFormation 资源类型提供的资源。	自定义资源
宏	此 `AWS::CloudFormation::Macro` 资源类型定义了可以在 CloudFormation 模板上执行	模板宏

资源类型	说明	文档
	自定义处理的可复用代码片段。宏可以在堆栈创建或更新期间修改模板、生成其他资源或执行其他自定义操作。	
嵌套堆栈	借助此 AWS::CloudFormation::Stack 资源类型，您可以在 CloudFormation 模板中创建嵌套堆栈，从而提高堆栈架构的模块化和可复用性水平。	嵌套堆栈
堆栈集	此 AWS::CloudFormation::StackSet 资源类型会创建或更新一种 CloudFormation 堆栈集，也就是一个可以跨多个 Amazon Web Services 账户 和区域部署的堆栈容器。	使用 StackSets 管理堆栈
等待条件	在满足特定条件之前（例如，长时间运行进程的成功完成或外部资源的可用性），该 AWS::CloudFormation::WaitCondition 资源类型会暂停堆栈的创建或更新。	等待条件
等待条件句柄	该 AWS::CloudFormation::WaitConditionHandle 资源类型可与 AWS::CloudFormation::WaitCondition 资源类型配合使用，它提供了预签名 URL，用于发送表示特定条件已得到满足的信号。这些信号允许堆栈创建或更新过程继续进行下去。	等待条件

使用自定义资源创建自定义预置逻辑

自定义资源为您提供了一种将自定义预置逻辑写入 CloudFormation 模板，并让 CloudFormation 在创建、更新（如果更改了自定义资源）或删除堆栈期间运行该逻辑的方式。预置要求涉及无法使用 CloudFormation 的内置资源类型表达的复杂逻辑或工作流程时，这会很有帮助。

例如，您可能需要包含不可作为 CloudFormation 资源类型的资源。您可以使用自定义资源包含这些资源。这样，您仍然可以在一个堆栈中管理所有相关资源。

要在 CloudFormation 模板中定义自定义资源，可以使用 `AWS::CloudFormation::CustomResource` 或 `Custom::MyCustomResourceTypeName` 资源类型。自定义资源需要一个属性，即服务令牌，它指定 CloudFormation 发送请求的目标，如 Amazon SNS 主题或 Lambda 函数。

以下主题提供有关如何使用自定义资源的信息。

主题

- 自定义资源的工作原理
- 响应超时
- CloudFormation 自定义资源请求和响应参考
- Amazon SNS 支持的自定义资源
- Lambda 支持的自定义资源

> **Note**
>
> CloudFormation 注册表和自定义资源各有其优点。自定义资源具有以下优点：
>
> - 您不需要注册资源。
> - 您无需注册即可将整个资源作为模板的一部分。
> - 支持 `Create`、`Update` 和 `Delete` 操作
>
> 基于注册表的资源具有以下优点：
>
> - 支持对第三方应用程序资源进行建模、预置和管理
> - 支持 `Create`、`Read`、`Update`、`Delete` 和 `List` (CRUDL) 操作
> - 支持对私有和第三方资源类型的偏差检测
>
> 与自定义资源不同，基于注册表的资源不需要关联 Amazon SNS 主题或 Lambda 函数即可执行 CRUDL 操作。有关更多信息，请参阅 通过 CloudFormation 注册表管理扩展。

自定义资源的工作原理

设置新自定义资源的一般过程包括以下步骤。这些步骤涉及两个角色：拥有该自定义资源的自定义资源提供者和创建包含某个自定义资源类型的模板的模板开发者。这两个角色可以是同一个人，但如果不是同一个人，自定义资源提供者应该与模板开发者合作。

1. 自定义资源提供者编写的逻辑决定了如何处理来自 CloudFormation 的请求以及如何对自定义资源执行操作。
2. 自定义资源提供者可创建 Amazon SNS 主题或 Lambda 函数，然后 CloudFormation 可向该主题或函数发送请求。该 Amazon SNS 主题或 Lambda 函数必须位于堆栈所在区域。
3. 自定义资源提供者将 Amazon SNS 主题 ARN 或 Lambda 函数 ARN 提供给模板开发者。
4. 模板开发者定义其 CloudFormation 模板中的自定义资源。这包括服务令牌和任何输入数据参数。服务令牌和输入数据的结构由自定义资源提供者定义。服务令牌指定了 Amazon SNS 主题 ARN 或 Lambda 函数 ARN，并且始终为必填项，而输入数据是可选的，具体取决于自定义资源。

现在，每当有人使用模板创建、更新或删除该自定义资源时，CloudFormation 都会向指定的服务令牌发送请求并等待响应，然后才会执行堆栈操作。

以下总结了使用该模板创建堆栈的流程：

1. CloudFormation 向指定的服务令牌发送请求。该请求包含请求类型以及自定义资源用于向其发送请求的预签名 Amazon S3 存储桶 URL 等信息。有关请求中包含的内容的更多信息，请参阅CloudFormation 自定义资源请求和响应参考。

   以下示例数据演示了 CloudFormation 在 `Create` 请求中包含的内容：在本示例中，`ResourceProperties` 允许 CloudFormation 创建发送到 Lambda 函数的自定义负载。

```
{
   "RequestType" : "Create",
   "RequestId" : "unique id for this create request",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "ResponseURL" : "http://pre-signed-S3-url-for-response",
   "ResourceType" : "Custom::TestResource",
   "LogicalResourceId" : "MyTestResource",
   "ResourceProperties" : {
      "Name" : "Value",
      "List" : [ "1", "2", "3" ]
   }
```

```
}
```

2. 自定义资源提供商处理 CloudFormation 请求并向预签名 URL 返回 SUCCESS 或 FAILED 响应。custom resource provider 提供采用 JSON 格式文件的响应并将响应上传到预签名的 S3 URL。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用预签名 URL 上传对象。

   在响应中，custom resource provider还可以包含template developer可以访问的名称-值对。例如，如果请求成功，响应可以包含输出数据，如果请求失败，可以包含错误消息。有关响应的更多信息，请参阅CloudFormation 自定义资源请求和响应参考。

   > **Important**
   >
   > 如果名称值对包含敏感信息，应使用 NoEcho 字段遮蔽自定义资源的输出。否则，这些值通过显示属性值的 API（例如 DescribeStackEvents）可见。
   > 有关使用 NoEcho 遮蔽敏感信息的更多信息，请参阅 请勿将凭证嵌入您的模板 最佳实践。

   custom resource provider负责侦听和响应请求。例如，对于 Amazon SNS 通知，自定义资源提供商必须侦听并响应发送到特定主题 ARN 的通知。CloudFormation 在预签名 URL 位置等待并侦听响应。

   以下示例数据说明自定义资源在响应中可以包含的内容：

```
{
   "Status" : "SUCCESS",
   "RequestId" : "unique id for this create request",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MyTestResource",
   "PhysicalResourceId" : "TestResource1",
   "Data" : {
      "OutputName1" : "Value1",
      "OutputName2" : "Value2",
   }
}
```

3. 获得 SUCCESS 响应后，CloudFormation 继续堆栈操作。如果收到 FAILED 响应或未返回任何响应，则操作失败。来自自定义资源的所有输出数据都存储在预签名 URL 位置。模板开发人员可使用 Fn::GetAtt 函数检索该数据。

> **Note**
>
> 如果您使用 Amazon PrivateLink，则 VPC 中的自定义资源必须能够访问特定于 CloudFormation 的 S3 存储桶。自定义资源必须将响应发送到预签名的 Amazon S3 URL。如果这些资源不能向 Amazon S3 发送响应，则 CloudFormation 将不会收到响应，并且堆栈操作会失败。有关更多信息，请参阅 使用接口端点 (Amazon PrivateLink) 访问 Amazon CloudFormation。

响应超时

自定义资源的默认超时为 3600 秒（1 小时）。如果在此期间内没有收到任何响应，则该堆栈操作将会失败。

您可以根据预期从自定义资源收到响应将需要的时间来调整超时值。例如，在预置将调用某个 Lambda 函数的自定义资源并且预期会在五分钟内收到响应时，您可以在堆栈模板中通过指定 `ServiceTimeout` 属性将超时设置为五分钟。有关更多信息，请参阅 CloudFormation 自定义资源请求和响应参考。这样，如果 Lambda 函数出现错误导致其停滞，CloudFormation 将在五分钟后使堆栈操作失败，而不是等待整整一个小时。

但是，请注意不要将超时值设置得过低。为避免意外超时，请确保您的自定义资源有足够的时间来执行必要的操作并返回响应。

CloudFormation 自定义资源请求和响应参考

Amazon CloudFormation 通过与您的自定义资源提供商通信的请求-响应协议管理自定义资源。每个请求都包含一个请求类型（`Create`、`Update` 或 `Delete`），并遵循这一高级工作流：

1. 模板开发者在模板中使用 `ServiceToken` 和 `ServiceTimeout` 定义自定义资源并启动堆栈操作。
2. CloudFormation 通过 SNS 或 Lambda 向自定义资源提供商发送 JSON 请求。
3. 该自定义资源提供商会处理请求，并在超时时间到期之前向预签名的 Amazon S3 存储桶 URL 返回一个 JSON 响应。
4. CloudFormation 会读取响应并继续堆栈操作。如果在超时期限结束之前没有收到任何响应，则该请求被视为不成功，并且堆栈操作将会失败。

有关更多信息，请参阅 自定义资源的工作原理。

本节介绍每种请求类型的结构、参数和预期响应。

> **Note**
>
> 响应正文的总大小不能超过 4096 字节。

模板设置

在模板中定义自定义资源时，模板开发者会使用具有以下属性的 AWS::CloudFormation::CustomResource：

`ServiceToken`

来自与堆栈相同区域的 Amazon SNS 主题 ARN 或 Lambda 函数 ARN。

是否必需：是

类型：字符串

`ServiceTimeout`

自定义资源操作超时前的最长时间（以秒为单位）。此值必须介于 1 和 3600 之间。默认值：3600 秒（1 小时）。

必需：否

类型：字符串

支持其他资源属性。资源属性将以 `ResourceProperties` 形式包含在请求中。自定义资源提供商必须确定哪些属性有效以及它们的可接受值。

请求对象

Create

当模板开发者创建包含自定义资源的堆栈时，CloudFormation 会发送一条将 `RequestType` 设置为 `Create` 的请求。

创建请求包含以下字段：

`RequestType`

`Create`.

是否必需：是

类型：字符串

`RequestId`

请求的唯一 ID。

将 `StackId` 与 `RequestId` 组合形成一个值，您可以使用该值唯一标识对特定自定义资源的请求。

是否必需：是

类型：字符串

`StackId`

标识包含自定义资源的堆栈的 Amazon 资源名称（ARN）。

将 `StackId` 与 `RequestId` 组合形成一个值，您可以使用该值唯一标识对特定自定义资源的请求。

是否必需：是

类型：字符串

`ResponseURL`

响应 URL 标识一个预签名 S3 存储桶，该存储桶接收从自定义资源提供商到 CloudFormation 的响应。

是否必需：是

类型：字符串

`ResourceType`

CloudFormation 模板中模板开发人员选择的自定义资源的资源类型。自定义资源类型名称的长度最多为 60 个字符，并且可包含字母数字字符和以下字符：`_@-`。

是否必需：是

类型：字符串

`LogicalResourceId`

CloudFormation 模板中模板开发人员选择的自定义资源的名称（逻辑 ID）。

是否必需：是

类型：字符串

`ResourceProperties`

此字段包含模板开发人员发送的 `Properties` 对象的内容。其内容由自定义资源提供商定义。

必需：否

类型：JSON 对象

示例

```
{
   "RequestType" : "Create",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
   "ResponseURL" : "pre-signed-url-for-create-response",
   "ResourceType" : "Custom::MyCustomResourceType",
   "LogicalResourceId" : "resource-logical-id",
   "ResourceProperties" : {
      "key1" : "string",
      "key2" : [ "list" ],
      "key3" : { "key4" : "map" }
   }
}
```

Update

当模板开发者更改模板中某个自定义资源的属性并更新堆栈时，CloudFormation 会向自定义资源提供者发送一条将 `RequestType` 设置为 `Update` 的请求。这意味着自定义资源代码无需检测资源中的更改，因为在请求类型为 `Update` 时，代码会知道其属性已经更改。

更新请求包含以下字段：

`RequestType`

`Update`.

是否必需：是

类型：字符串

`RequestId`

请求的唯一 ID。

将 `StackId` 与 `RequestId` 组合形成一个值，您可以使用该值唯一标识对特定自定义资源的请求。

是否必需：是

类型：字符串

`StackId`

标识包含自定义资源的堆栈的 Amazon 资源名称（ARN）。

将 `StackId` 与 `RequestId` 组合形成一个值，您可以使用该值唯一标识对特定自定义资源的请求。

是否必需：是

类型：字符串

`ResponseURL`

响应 URL 标识一个预签名 S3 存储桶，该存储桶接收从自定义资源提供商到 CloudFormation 的响应。

是否必需：是

类型：字符串

`ResourceType`

CloudFormation 模板中模板开发人员选择的自定义资源的资源类型。自定义资源类型名称的长度最多为 60 个字符，并且可包含字母数字字符和以下字符：`_@-`。更新期间不能更改类型。

是否必需：是

类型：字符串

`LogicalResourceId`

CloudFormation 模板中模板开发人员选择的自定义资源的名称（逻辑 ID）。

是否必需：是

类型：字符串

`PhysicalResourceId`

自定义资源提供商定义的物理 ID，该 ID 对于该提供程序是唯一的。

是否必需：是

类型：字符串

`ResourceProperties`

此字段包含模板开发人员发送的 `Properties` 对象的内容。其内容由自定义资源提供商定义。

必需：否

类型：JSON 对象

`OldResourceProperties`

仅用于 `Update` 请求。模板开发人员之前在 CloudFormation 模板中声明的资源属性值。

是否必需：是

类型：JSON 对象

示例

```
{
   "RequestType" : "Update",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
   "ResponseURL" : "pre-signed-url-for-update-response",
   "ResourceType" : "Custom::MyCustomResourceType",
   "LogicalResourceId" : "resource-logical-id",
   "PhysicalResourceId" : "provider-defined-physical-id",
   "ResourceProperties" : {
      "key1" : "new-string",
      "key2" : [ "new-list" ],
      "key3" : { "key4" : "new-map" }
   },
   "OldResourceProperties" : {
      "key1" : "string",
      "key2" : [ "list" ],
```

```
      "key3" : { "key4" : "map" }
   }
}
```

Delete

当模板开发者删除该堆栈或从模板中移除该自定义资源然后更新堆栈时，CloudFormation 会发送一条将 `RequestType` 设置为 `Delete` 的请求。

删除请求包含以下字段：

`RequestType`

`Delete`.

是否必需：是

类型：字符串

`RequestId`

请求的唯一 ID。

是否必需：是

类型：字符串

`StackId`

标识包含自定义资源的堆栈的 Amazon 资源名称（ARN）。

是否必需：是

类型：字符串

`ResponseURL`

响应 URL 标识一个预签名 S3 存储桶，该存储桶接收从自定义资源提供商到 CloudFormation 的响应。

是否必需：是

类型：字符串

`ResourceType`

CloudFormation 模板中模板开发人员选择的自定义资源的资源类型。自定义资源类型名称的长度最多为 60 个字符，并且可包含字母数字字符和以下字符：`_@-`。

是否必需：是

类型：字符串

`LogicalResourceId`

CloudFormation 模板中模板开发人员选择的自定义资源的名称（逻辑 ID）。

是否必需：是

类型：字符串

`PhysicalResourceId`

自定义资源提供商定义的物理 ID，该 ID 对于该提供程序是唯一的。

是否必需：是

类型：字符串

`ResourceProperties`

此字段包含模板开发人员发送的 `Properties` 对象的内容。其内容由自定义资源提供商定义。

必需：否

类型：JSON 对象

示例

```
{
   "RequestType" : "Delete",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
   "ResponseURL" : "pre-signed-url-for-delete-response",
   "ResourceType" : "Custom::MyCustomResourceType",
   "LogicalResourceId" : "resource-logical-id",
   "PhysicalResourceId" : "provider-defined-physical-id",
   "ResourceProperties" : {
```

```
      "key1" : "string",
      "key2" : [ "list" ],
      "key3" : { "key4" : "map" }
   }
}
```

响应对象

对于所有的请求类型，自定义资源提供商会向预签名 URL 发送一条响应。如果自定义资源提供商没有发送响应，CloudFormation 会等待直到操作超时。

该响应必须为具有以下字段的 JSON 对象：

`Status`

必须是 SUCCESS 或 FAILED。

是否必需：是

类型：字符串

`RequestId`

请求的唯一 ID。完全按照请求中显示的值复制该值。

是否必需：是

类型：字符串

`StackId`

标识包含自定义资源的堆栈的 Amazon 资源名称（ARN）。完全按照请求中显示的值复制该值。

是否必需：是

类型：字符串

`LogicalResourceId`

CloudFormation 模板中模板开发人员选择的自定义资源的名称（逻辑 ID）。完全按照请求中显示的值复制该值。

是否必需：是

类型：字符串

`PhysicalResourceId`

该值应该为自定义资源供应商的唯一标识符，并且最大为 1KB。该值必须为非空字符串，并且对于同一资源的所有响应都必须相同。

更新自定义资源时，`PhysicalResourceId` 返回的值决定了更新行为。如果值保持相同，则 CloudFormation 将其视为正常更新。如果值发生更改，CloudFormation 会将该更新解读为替换，并向旧资源发送删除请求。有关更多信息，请参阅 AWS::CloudFormation::CustomResource。

是否必需：是

类型：字符串

`Reason`

描述失败响应的原因。

如果 `Status` 为 `FAILED`，则必需。否则，它是可选的。

必需：条件

类型：字符串

`NoEcho`

指示在使用 `Fn::GetAtt` 函数检索时是否遮蔽自定义资源的输出。如果设置为 `true`，则除模板 *`Metadata`* 区段中存储的值外，所有返回值都将用星号 (*****) 遮蔽。CloudFormation 不会转换、修改或编辑您在 `Metadata` 部分中包含的任何信息。默认值为 `false`。

有关使用 `NoEcho` 遮蔽敏感信息的更多信息，请参阅 请勿将凭证嵌入您的模板 最佳实践。

仅适用于 `Create` 和 `Update` 响应。不支持 `Delete` 响应。

必需：否

类型：布尔值

`Data`

自定义资源提供商定义的名称值对，随响应一起发送。您可以使用 `Fn::GetAtt` 在模板中按名称访问此处提供的值。

仅适用于 `Create` 和 `Update` 响应。不支持 `Delete` 响应。

> **Important**
>
> 如果名称值对包含敏感信息，应使用 NoEcho 字段遮蔽自定义资源的输出。否则，这些值通过显示属性值的 API（例如 DescribeStackEvents）可见。

必需：否

类型：JSON 对象

成功响应示例

Create 和 Update 响应

```
{
   "Status": "SUCCESS",
   "RequestId": "unique-request-id",
   "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
   "LogicalResourceId": "resource-logical-id",
   "PhysicalResourceId": "provider-defined-physical-id",
   "NoEcho": true,
   "Data": {
      "key1": "value1",
      "key2": "value2"
   }
}
```

Delete 响应

```
{
   "Status": "SUCCESS",
   "RequestId": "unique-request-id",
   "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
   "LogicalResourceId": "resource-logical-id",
   "PhysicalResourceId": "provider-defined-physical-id"
}
```

失败的响应示例

```
{
```

```
    "Status": "FAILED",
    "RequestId": "unique-request-id",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
    "LogicalResourceId": "resource-logical-id",
    "PhysicalResourceId": "provider-defined-physical-id",
    "Reason": "Required failure reason string"
}
```

Amazon SNS 支持的自定义资源

以下主题展示了如何使用服务令牌配置自定义资源，该令牌指定了 CloudFormation 向其发送请求的 Amazon SNS 主题。您还将了解自定义资源堆栈创建、更新和删除过程中发送和接收的事件和消息的顺序。

通过自定义资源和 Amazon SNS，您可以启用方案，例如向堆栈添加新资源和向堆栈注入动态数据。例如，创建堆栈时，CloudFormation 可以向被 Amazon EC2 实例上运行的应用程序监控的主题发送 `Create` 请求。Amazon SNS 通知触发应用程序以执行其他预置任务，如检索列在白名单中的弹性 IP 池。完成后，应用程序发送响应（和任何输出数据），通知 CloudFormation 继续执行堆栈操作。

将 Amazon SNS 主题指定为自定义资源的目标时，CloudFormation 会在涉及自定义资源的堆栈操作期间，向指定的 SNS 主题发送消息。要处理这些消息并执行必要的操作，必须有一个订阅 SNS 主题的受支持端点。

有关自定义资源及其工作原理的介绍，请参阅自定义资源的工作原理。有关 Amazon SNS 及其工作原理的信息，请参阅 Amazon Simple Notification Service Developer Guide。

使用 Amazon SNS 创建自定义资源

主题

- 步骤 1：堆栈创建
- 步骤 2：堆栈更新
- 步骤 3：堆栈删除

步骤 1：堆栈创建

1. 模板开发人员创建包含自定义资源的 CloudFormation 堆栈。

   在以下模板示例中，对具有逻辑 ID `Custom::SeleniumTester` 的自定义资源使用自定义资源类型名称 *MySeleniumTest*。自定义资源类型名称必须是字母数字字符，最大长度为 60 个字符。

自定义资源类型是使用服务令牌、可选提供商特定属性以及由自定义资源提供商定义的可选 Fn::GetAtt 属性声明的。使用这些属性和特性可以将信息从template developer传递给custom resource provider，反之亦然。服务令牌指定资源提供商已配置的 Amazon SNS 主题。

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "MySeleniumTest" : {
         "Type": "Custom::SeleniumTester",
         "Version" : "1.0",
         "Properties" : {
            "ServiceToken": "arn:aws:sns:us-west-2:123456789012:CRTest",
            "seleniumTester" : "SeleniumTest()",
            "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/",
 "http://search.mysite.com" ],
            "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
         }
      }
   },
   "Outputs" : {
      "topItem" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
      },
      "numRespondents" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
      }
   }
}
```

> **Note**
>
> 在提供商响应 CloudFormation 期间，自定义资源提供商会返回使用 `Fn::GetAtt` 访问的数据的名称和值。如果custom resource provider是第三方，则template developer必须从custom resource provider获取这些返回值的名称。

2. CloudFormation 使用 `"RequestType" : "Create"` 向资源提供者发送一条 Amazon SNS 通知，其中包含有关堆栈的信息、堆栈模板中的自定义资源属性和用于响应的 S3 URL。

   用于发送通知的 SNS 主题嵌入在模板的 `ServiceToken` 属性中。要避免使用硬编码值，模板开发人员可以使用模板参数，以便在启动堆栈时输入值。

以下示例显示一个自定义资源 Create 请求，其中包含一个用 Custom::SeleniumTester 的 LogicalResourceId 创建的自定义资源类型名称 MySeleniumTester：

```
{
   "RequestType" : "Create",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "ResponseURL" : "http://pre-signed-S3-url-for-response",
   "ResourceType" : "Custom::SeleniumTester",
   "LogicalResourceId" : "MySeleniumTester",
   "ResourceProperties" : {
      "seleniumTester" : "SeleniumTest()",
      "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
search.mysite.com" ],
      "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
   }
}
```

有关 Create 请求的请求对象的详细信息，请参阅 请求和响应参考 主题。

3. custom resource provider处理template developer发送的数据，并确定 Create 请求是否已成功。然后，资源提供者使用 CloudFormation 发送的 S3 URL 来发送 SUCCESS 或 FAILED 响应。

   根据响应类型，CloudFormation 将需要不同的响应字段。有关特定请求类型的响应字段的信息，请参阅 请求和响应参考 部分中该请求类型的文档。

   在响应创建或更新请求时，custom resource provider 可以在响应的 Data 字段中返回数据元素。这些是名称/值对，名称对应于用于堆栈模板中的自定义资源的 Fn::GetAtt 属性。该值是模板开发人员对具有此属性名称的资源调用 Fn::GetAtt 时返回的数据。

   以下是自定义资源响应的示例：

```
{
   "Status" : "SUCCESS",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1",
   "Data" : {
```

```
      "resultsPage" : "http://www.myexampledomain/test-results/guid",
      "lastUpdate" : "2012-11-14T03:30Z"
   }
}
```

有关 Create 请求的响应对象的详细信息，请参阅 请求和响应参考 主题。

StackId、RequestId 和 LogicalResourceId 字段必须从请求中逐字复制。

4. CloudFormation 将堆栈状态声明为 CREATE_COMPLETE 或 CREATE_FAILED。如果堆栈已成功创建，模板开发人员通过 Fn::GetAtt 访问已创建的自定义资源的输出值，可以使用这些值。

   例如，用于举例说明的自定义资源模板使用 Fn::GetAtt 将资源输出复制到堆栈输出：

```
"Outputs" : {
   "topItem" : {
      "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
   },
   "numRespondents" : {
      "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
   }
}
```

步骤 2：堆栈更新

要更新现有堆栈，您必须提交一个指定了堆栈资源属性更新的模板，如下面的示例所示。CloudFormation 只更新模板中指定了更改的资源。有关更多信息，请参阅 理解堆栈资源的更新行为。

您可以更新需要替换基础物理资源的自定义资源。在 CloudFormation 模板中更新某个自定义资源时，CloudFormation 会向该自定义资源发送更新请求。如果需要替换自定义资源，新的自定义资源必须使用新的物理 ID 发送响应。CloudFormation 收到响应时，会比较新旧自定义资源的 PhysicalResourceId。如果不同，CloudFormation 会将更新视为替换，并向旧资源发送删除请求，如 步骤 3：堆栈删除 中所示。

> **Note**
>
> 如果您没有对自定义资源进行更改，CloudFormation 在堆栈更新过程中不会向资源发送请求。

1. template developer启动对包含自定义资源的堆栈的更新。在更新期间，template developer可以在堆栈模板中指定新属性。

   下面是一个使用自定义资源类型的堆栈模板 Update 示例：

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "MySeleniumTest" : {
         "Type": "Custom::SeleniumTester",
         "Version" : "1.0",
         "Properties" : {
            "ServiceToken": "arn:aws:sns:us-west-2:123456789012:CRTest",
            "seleniumTester" : "SeleniumTest()",
            "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/",
 "http://search.mysite.com",
               "http://mynewsite.com" ],
            "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
         }
      }
   },
   "Outputs" : {
      "topItem" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
      },
      "numRespondents" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
      }
   }
}
```

2. CloudFormation 会使用 "RequestType" : "Update" 向资源提供者发送一条 Amazon SNS 通知，其中包含与 Create 调用类似的信息，不同的是，OldResourceProperties 字段包含旧的资源属性，而 ResourceProperties 包含已更新的（如果有）资源属性。

   以下是一个 Update 请求的示例：

```
{
   "RequestType" : "Update",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "ResponseURL" : "http://pre-signed-S3-url-for-response",
```

```
   "ResourceType" : "Custom::SeleniumTester",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1",
   "ResourceProperties" : {
      "seleniumTester" : "SeleniumTest()",
      "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
search.mysite.com",
         "http://mynewsite.com" ],
      "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
   },
   "OldResourceProperties" : {
      "seleniumTester" : "SeleniumTest()",
      "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
search.mysite.com" ],
      "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
   }
}
```

有关 Update 请求的请求对象的详细信息，请参阅 请求和响应参考 主题。

3. 自定义资源提供商处理由 CloudFormation 发送的数据。自定义资源执行更新并向 S3 URL 发送 SUCCESS 或 FAILED 响应。然后，CloudFormation 比较新旧自定义资源的 PhysicalResourceIDs。如果不同，CloudFormation 将认为更新需要替换，并向旧资源发送删除请求。下面的示例说明custom resource provider对 Update 请求的响应。

```
{
   "Status" : "SUCCESS",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester2"
}
```

有关 Update 请求的响应对象的详细信息，请参阅 请求和响应参考 主题。

StackId、RequestId 和 LogicalResourceId 字段必须从请求中逐字复制。

4. CloudFormation 将堆栈状态声明为 UPDATE_COMPLETE 或 UPDATE_FAILED。如果更新失败，堆栈将回滚。如果堆栈更新成功，template developer可以使用 Fn::GetAtt 访问已创建自定义资源的任何新输出值。

步骤 3：堆栈删除

1. 模板开发人员删除包含自定义资源的堆栈。CloudFormation 将获取堆栈模板中指定的当前属性及 SNS 主题，并准备向自定义资源提供商发出请求。
2. CloudFormation 使用 "RequestType" : "Delete" 向资源提供者发送一条 Amazon SNS 通知，其中包含有关堆栈的当前信息、堆栈模板中的自定义资源属性和用于响应的 S3 URL。

   只要您删除堆栈或进行自定义资源的删除或替换更新，CloudFormation 都会比较新旧自定义资源的 PhysicalResourceId。如果不同，CloudFormation 会将更新视为替换，并向旧资源 (OldPhysicalResource) 发送删除请求，如下面的 Delete 请求示例所示。

```
{
   "RequestType" : "Delete",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "ResponseURL" : "http://pre-signed-S3-url-for-response",
   "ResourceType" : "Custom::SeleniumTester",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1",
   "ResourceProperties" : {
      "seleniumTester" : "SeleniumTest()",
      "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
search.mysite.com",
         "http://mynewsite.com" ],
      "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
   }
}
```

   有关 Delete 请求的请求对象的详细信息，请参阅 请求和响应参考 主题。

   DescribeStackResource、DescribeStackResources 和 ListStackResources 显示用户定义的名称 (如果已指定)。
3. 自定义资源提供商处理 CloudFormation 发送的数据，并确定 Delete 请求是否已成功。然后，资源提供者使用 CloudFormation 发送的 S3 URL 来发送 SUCCESS 或 FAILED 响应。要成功删除带自定义资源的堆栈，custom resource provider 必须成功响应删除请求。

   以下是custom resource provider响应 Delete 请求的示例：

```
{
```

```
   "Status" : "SUCCESS",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1"
}
```

有关 Delete 请求的响应对象的详细信息，请参阅 请求和响应参考 主题。

StackId、RequestId 和 LogicalResourceId 字段必须从请求中逐字复制。

4. CloudFormation 将堆栈状态声明为 DELETE_COMPLETE 或 DELETE_FAILED。

Lambda 支持的自定义资源

如果将 Lambda 函数与自定义资源关联，则在创建、更新或删除自定义资源时就会调用该函数。CloudFormation 将调用 Lambda API 来调用此函数，并将所有请求数据（如请求类型和资源属性）传递给此函数。通过将 Lambda 函数的强大功能和可自定义性与 CloudFormation 结合，您可以支持各种方案，例如在堆栈创建期间动态查找 AMI ID，或者实现和使用实用工具函数（例如，字符串反转函数）。

有关自定义资源及其工作原理的介绍，请参阅自定义资源的工作原理。

主题

- 演练：使用 Lambda 支持的自定义资源创建延迟机制
- cfn-response 模块

演练：使用 Lambda 支持的自定义资源创建延迟机制

本演练向您展示如何使用示例 CloudFormation 模板配置和启动 Lambda 支持的自定义资源。此模板创建了一种延迟机制，可在指定时间暂停堆栈部署。当您需要在资源预置期间引入故意的延迟时（例如在创建依赖资源之前等待资源稳定时），这可能非常有用。

> **Note**
>
> 虽然之前建议使用 Lambda 支持的自定义资源来检索 AMI ID，但现在建议使用 Amazon Systems Manager 参数。此方法可以使您的模板提高可重用性和更易于维护。有关更多信息，请参阅 获取 Systems Manager Parameter Store 中的纯文本值。

主题

- 概述
- 示例模板
- 示例模板演练
- 先决条件
- 启动堆栈
- 清理资源
- 相关信息

概述

此演练中使用的示例堆栈模板创建了 Lambda 支持的自定义资源。此自定义资源在堆栈创建过程中引入了可配置的延迟（默认为 60 秒）。只有在修改自定义资源的属性时，才会在堆栈更新期间出现延迟。

此模板预置以下资源：

- 自定义资源，
- Lambda 函数，和
- 使 Lambda 能够将日志写入到 CloudWatch 的 IAM 角色。

它还定义了两个输出：

- 函数等待的实际时间。
- 每次执行 Lambda 函数期间生成的唯一标识符。

> **Note**
>
> CloudFormation 是一项免费服务，但 Lambda 会根据您的函数请求数量和代码执行时间收费。有关 Lambda 定价的更多信息，请参阅 Amazon Lambda 定价。

示例模板

您可以查看具有以下延迟机制的 Lambda 支持的自定义资源示例模板：

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "LambdaExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": { "Service": ["lambda.amazonaws.com"] },
            "Action": ["sts:AssumeRole"]
          }]
        },
        "Path": "/",
        "Policies": [{
          "PolicyName": "AllowLogs",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": ["logs:*"],
              "Resource": "*"
            }]
          }
        }]
      }
    },
    "CFNWaiter": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "python3.9",
        "Timeout": 900,
        "Role": { "Fn::GetAtt": ["LambdaExecutionRole", "Arn"] },
        "Code": {
          "ZipFile": { "Fn::Join": ["\n", [
            "from time import sleep",
            "import json",
            "import cfnresponse",
            "import uuid",
            "",
            "def handler(event, context):",
            "  wait_seconds = 0",
```

```
            "  id = str(uuid.uuid1())",
            "  if event[\"RequestType\"] in [\"Create\", \"Update\"]:",
            "    wait_seconds = int(event[\"ResourceProperties\"].get(\"ServiceTimeout
\", 0))",
            "    sleep(wait_seconds)",
            "  response = {",
            "    \"TimeWaited\": wait_seconds,",
            "    \"Id\": id ",
            "  }",
            "  cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
 \"Waiter-\"+id)"
          ]]}
        }
      }
    },
    "CFNWaiterCustomResource": {
      "Type": "AWS::CloudFormation::CustomResource",
      "Properties": {
        "ServiceToken": { "Fn::GetAtt": ["CFNWaiter", "Arn"] },
        "ServiceTimeout": 60
      }
    }
  },
  "Outputs": {
    "TimeWaited": {
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "TimeWaited"] },
      "Export": { "Name": "TimeWaited" }
    },
    "WaiterId": {
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "Id"] },
      "Export": { "Name": "WaiterId" }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - "lambda.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        - PolicyName: "AllowLogs"
          PolicyDocument:
            Statement:
              - Effect: "Allow"
                Action:
                  - "logs:*"
                Resource: "*"
  CFNWaiter:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Runtime: python3.9
      Timeout: 900
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile:
          !Sub |
          from time import sleep
          import json
          import cfnresponse
          import uuid

          def handler(event, context):
            wait_seconds = 0
            id = str(uuid.uuid1())
            if event["RequestType"] in ["Create", "Update"]:
              wait_seconds = int(event["ResourceProperties"].get("ServiceTimeout", 0))
              sleep(wait_seconds)
            response = {
              "TimeWaited": wait_seconds,
              "Id": id
            }
            cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
"Waiter-"+id)
  CFNWaiterCustomResource:
```

```
    Type: AWS::CloudFormation::CustomResource
    Properties:
      ServiceToken: !GetAtt CFNWaiter.Arn
      ServiceTimeout: 60
Outputs:
  TimeWaited:
    Value: !GetAtt CFNWaiterCustomResource.TimeWaited
    Export:
      Name: TimeWaited
  WaiterId:
    Value: !GetAtt CFNWaiterCustomResource.Id
    Export:
      Name: WaiterId
```

示例模板演练

以下代码段解释了示例模板的相关部分，以帮助您了解 Lambda 函数如何与自定义资源关联并理解输出。

AWS::Lambda::Function 资源 CFNWaiter

AWS::Lambda::Function 资源指定了函数的源代码、处理程序名称、运行时环境和执行角色的 Amazon 资源名称（ARN）。

由于 Handler 属性使用 Python 源代码，因此将其设置为 index.handler。有关使用内联函数源代码时可接受的处理程序标识符的更多信息，请参阅 AWS::Lambda::Function 代码。

由于源文件是 Python 代码，因此 Runtime 指定为 python3.9。

Timeout 设置为 900 秒。

Role 属性使用 Fn::GetAtt 函数来获取在模板中的 AWS::IAM::Role 资源中声明的 LambdaExecutionRole 执行角色的 ARN。

Code 属性使用 Python 函数以内联方式定义函数代码。示例模板中的 Python 函数执行下面的操作：

- 使用 UUID 创建唯一 ID
- 检查请求是创建请求还是更新请求
- 在 Create 或 Update 请求期间休眠 ServiceTimeout 指定的持续时间
- 返回等待时间和唯一 ID

JSON

```
...
    "CFNWaiter": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "python3.9",
        "Timeout": 900,
        "Role": { "Fn::GetAtt": ["LambdaExecutionRole", "Arn"] },
        "Code": {
          "ZipFile": { "Fn::Join": ["\n", [
            "from time import sleep",
            "import json",
            "import cfnresponse",
            "import uuid",
            "",
            "def handler(event, context):",
            "  wait_seconds = 0",
            "  id = str(uuid.uuid1())",
            "  if event[\"RequestType\"] in [\"Create\", \"Update\"]:",
            "    wait_seconds = int(event[\"ResourceProperties\"].get(\"ServiceTimeout
\", 0))",
            "    sleep(wait_seconds)",
            "  response = {",
            "    \"TimeWaited\": wait_seconds,",
            "    \"Id\": id ",
            "  }",
            "  cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
 \"Waiter-\"+id)"
          ]]}
        }
      }
    },
...
```

YAML

```
...
  CFNWaiter:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
```

```
      Runtime: python3.9
      Timeout: 900
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile:
          !Sub |
          from time import sleep
          import json
          import cfnresponse
          import uuid

          def handler(event, context):
            wait_seconds = 0
            id = str(uuid.uuid1())
            if event["RequestType"] in ["Create", "Update"]:
              wait_seconds = int(event["ResourceProperties"].get("ServiceTimeout", 0))
              sleep(wait_seconds)
            response = {
              "TimeWaited": wait_seconds,
              "Id": id
            }
            cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
 "Waiter-"+id)
...
```

AWS::IAM::Role 资源 LambdaExecutionRole

AWS::IAM:Role 资源为 Lambda 函数创建一个执行角色，其中包括允许 Lambda 使用该角色的代入角色策略。它还包含允许 CloudWatch Logs 访问的策略。

JSON

```
...
    "LambdaExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": { "Service": ["lambda.amazonaws.com"] },
            "Action": ["sts:AssumeRole"]
          }]
```

```
        },
        "Path": "/",
        "Policies": [{
          "PolicyName": "AllowLogs",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": ["logs:*"],
              "Resource": "*"
            }]
          }
        }]
      }
    },
...
```

YAML

```
...
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - "lambda.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        - PolicyName: "AllowLogs"
          PolicyDocument:
            Statement:
              - Effect: "Allow"
                Action:
                  - "logs:*"
                Resource: "*"
...
```

AWS::CloudFormation::CustomResource 资源 CFNWaiterCustomResource

自定义资源使用 !GetAtt CFNWaiter.Arn 将其 ARN 链接到 Lambda 函数。它将为创建和更新操作实现 60 秒的等待时间，如 ServiceTimeout 中设置的那样。只有当属性被修改时，才会调用资源进行更新操作。

JSON

```
...
    "CFNWaiterCustomResource": {
      "Type": "AWS::CloudFormation::CustomResource",
      "Properties": {
        "ServiceToken": { "Fn::GetAtt": ["CFNWaiter", "Arn"] },
        "ServiceTimeout": 60
      }
    }
  },
...
```

YAML

```
...
  CFNWaiterCustomResource:
    Type: AWS::CloudFormation::CustomResource
    Properties:
      ServiceToken: !GetAtt CFNWaiter.Arn
      ServiceTimeout: 60
...
```

Outputs

此模板的 Outputs 是 TimeWaited 和 WaiterId。TimeWaited 值使用 Fn::GetAtt 函数提供等待者资源实际等待的时间。WaiterId 使用 Fn::GetAtt 函数来提供生成并与执行关联的唯一 ID。

JSON

```
...
  "Outputs": {
    "TimeWaited": {
```

```
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "TimeWaited"] },
      "Export": { "Name": "TimeWaited" }
    },
    "WaiterId": {
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "Id"] },
      "Export": { "Name": "WaiterId" }
    }
  }
}
...
```

YAML

```
...
Outputs:
  TimeWaited:
    Value: !GetAtt CFNWaiterCustomResource.TimeWaited
    Export:
      Name: TimeWaited
  WaiterId:
    Value: !GetAtt CFNWaiterCustomResource.Id
    Export:
      Name: WaiterId
...
```

先决条件

您必须拥有 IAM 权限才能使用所有相应的服务，例如 Lambda 和 CloudFormation。

启动堆栈

要创建 堆栈

1. 从示例模板部分找到您喜欢的模板（YAML 或 JSON），并将其保存到您的计算机，名称为 `samplelambdabackedcustomresource.template`。
2. 通过以下网址打开 CloudFormation 控制台：https://console.aws.amazon.com/cloudformation/。
3. 在堆栈页面，选择右上角的创建堆栈，然后选择使用新资源（标准）。
4. 在先决条件 – 准备模板中，选择选择现有模板。
5. 对于指定模板，选择上传模板文件，然后选择选择文件。
6. 选择您之前保存的 `samplelambdabackedcustomresource.template` 模板文件。

7. 选择下一步。
8. 对于堆栈名称，键入 **SampleCustomResourceStack**，然后选择下一步。
9. 在本演练中，您无需添加标记或指定高级设置，因此请选择 Next。
10. 确保堆栈名称看起来正确，然后选择创建。

CloudFormation 可能需要几分钟的时间创建堆栈。要监控进度，可查看堆栈事件。有关更多信息，请参阅 从 CloudFormation 控制台查看堆栈信息。

如果堆栈创建成功，则堆栈中的所有资源（例如 Lambda 函数和自定义资源）都已创建。您已成功使用 Lambda 函数和自定义资源。

如果 Lambda 函数返回错误，则在 CloudWatch Logs 控制台中查看此函数的日志。日志流的名称与自定义资源的物理 ID 相同，您可通过查看堆栈的资源来查找该名称。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的查看日志数据。

清理资源

删除堆栈以清理您创建的所有堆栈资源，从而避免为不必要的资源付费。

删除堆栈

1. 从 CloudFormation 控制台中，选择 SampleCustomResourceStack 堆栈。
2. 选择 Actions，然后选择 Delete Stack。
3. 在确认消息中，选择 Yes, Delete。

您创建的所有资源都会被删除。

既然您已了解如何创建和使用 Lambda 支持的自定义资源，可以使用本演练中的示例模板和代码来生成和试验其他堆栈和函数。

相关信息

- CloudFormation 自定义资源参考
- AWS::CloudFormation::CustomResource

`cfn-response` 模块

在 CloudFormation 模板中，您可以指定 Lambda 函数作为自定义资源的对象。如果使用 `ZipFile` 属性指定函数的源代码，可以加载 `cfn-response` 模块，以将 Lambda 函数的响应发送到自定义资

源。cfn-response 模块是一个库，可简化向调用 Lambda 函数的自定义资源发送响应的过程。该模块包含一个 send 方法，可以通过 Amazon S3 预签名 URL (ResponseURL) 将响应对象发送到自定义资源。

只有当您使用 cfn-response 属性编写您的源代码时，ZipFile 模块可用。它对于存储在 Amazon S3 存储桶中的源代码不可用。对于 存储桶中的代码，您必须编写自己的函数来发送响应。

> **Note**
>
> 在执行 send 方法后，Lambda 函数将终止，因此，将忽略在该方法之后写入的任何内容。

加载 cfn-response 模块

对于 Node.js 函数，使用 require() 函数来加载 cfn-response 模块。例如，以下代码示例创建一个名为 cfn-response 的 response 对象：

```
var response = require('cfn-response');
```

对于 Python，使用 import 语句来加载 cfnresponse 模块，如下例所示：

> **Note**
>
> 使用这个确切的导入语句。如果使用导入语句的其他变体，CloudFormation 将不会包含响应模块。

```
import cfnresponse
```

send 方法参数

您可以将以下参数用于 send 方法。

event

自定义资源请求中的字段。

`context`

Lambda 函数特定的对象，可用于指定何时函数和任何回调完成执行，或从 Lambda 执行环境中访问信息。有关更多信息，请参阅《Amazon Lambda 开发人员指南》中的使用 Node.js 构建 Lambda 函数。

`responseStatus`

函数是否成功完成。使用 `cfnresponse` 模块常量指定状态：`SUCCESS` 表示执行成功；`FAILED` 表示执行失败。

`responseData`

自定义资源响应对象的 `Data` 字段。数据为名称-值对列表。

`physicalResourceId`

可选。调用此函数的自定义资源的唯一标识符。默认情况下，该模块使用与 Lambda 函数关联的 Amazon CloudWatch Logs 日志流的名称。

为 `PhysicalResourceId` 返回的值可以更改自定义资源更新操作。如果返回的值相同，则将其视为正常更新。如果返回的值不同，CloudFormation 会将该更新视为替换，并向旧资源发送删除请求。有关更多信息，请参阅 AWS::CloudFormation::CustomResource。

`noEcho`

可选。指示在使用 `Fn::GetAtt` 函数检索自定义资源的输出时是否遮蔽它。如果设置为 `true`，则所有返回的值都将用星号 (*****) 遮蔽，但存储在下面指定位置的信息除外。默认情况下，此值为 `false`。

> **Important**
>
> 使用 `NoEcho` 属性不会遮蔽在以下各区段中存储的任何信息：
>
> - `Metadata` 模板区段。CloudFormation 不会转换、修改或编辑您在 `Metadata` 部分中包含的任何信息。有关更多信息，请参阅 Metadata。
> - `Outputs` 模板区段。有关更多信息，请参阅 Outputs。
> - 资源定义的 `Metadata` 属性。有关更多信息，请参阅 `Metadata` 属性。
>
> 强烈建议您不要使用这些机制来包含敏感信息，例如密码。

有关使用 `NoEcho` 遮蔽敏感信息的更多信息，请参阅 请勿将凭证嵌入您的模板 最佳实践。

示例

Node.js

在下面的 Node.js 示例中，内联 Lambda 函数获取输入值并将其乘以 5。内联函数对于较小的函数尤为有用，因为它们让您能够在模板中直接指定源代码，而不必先创建包再将包上传到 Amazon S3 存储桶中。函数使用 `cfn-response` send 方法将结果发回给调用此函数的自定义资源。

JSON

```
"ZipFile": { "Fn::Join": ["", [
  "var response = require('cfn-response');",
  "exports.handler = function(event, context) {",
  "  var input = parseInt(event.ResourceProperties.Input);",
  "  var responseData = {Value: input * 5};",
  "  response.send(event, context, response.SUCCESS, responseData);",
  "};"
]]}
```

YAML

```
ZipFile: >
  var response = require('cfn-response');
  exports.handler = function(event, context) {
    var input = parseInt(event.ResourceProperties.Input);
    var responseData = {Value: input * 5};
    response.send(event, context, response.SUCCESS, responseData);
  };
```

Python

在下面的 Python 示例中，内联 Lambda 函数采用一个整数值并将它乘以 5。

JSON

```
"ZipFile" : { "Fn::Join" : ["\n", [
  "import json",
  "import cfnresponse",
  "def handler(event, context):",
  "   responseValue = int(event['ResourceProperties']['Input']) * 5",
  "   responseData = {}",
  "   responseData['Data'] = responseValue",
```

```
  "   cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
 \"CustomResourcePhysicalID\")"
]]}
```

YAML

```
ZipFile: |
  import json
  import cfnresponse
  def handler(event, context):
    responseValue = int(event['ResourceProperties']['Input']) * 5
    responseData = {}
    responseData['Data'] = responseValue
    cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
 "CustomResourcePhysicalID")
```

模块源代码

主题

- 异步 Node.js 源代码
- Node.js 源代码
- Python 源代码

异步 Node.js 源代码

以下是在处理程序异步进行处理时，Node.js 函数的响应模块源代码。请查看这些内容，以理解此模块的功能并帮助您实现自己的响应函数。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

exports.SUCCESS = "SUCCESS";
exports.FAILED = "FAILED";

exports.send = function(event, context, responseStatus, responseData,
 physicalResourceId, noEcho) {

    return new Promise((resolve, reject) => {
        var responseBody = JSON.stringify({
            Status: responseStatus,
```

```
            Reason: "See the details in CloudWatch Log Stream: " +
 context.logStreamName,
            PhysicalResourceId: physicalResourceId || context.logStreamName,
            StackId: event.StackId,
            RequestId: event.RequestId,
            LogicalResourceId: event.LogicalResourceId,
            NoEcho: noEcho || false,
            Data: responseData
        });

        console.log("Response body:\n", responseBody);

        var https = require("https");
        var url = require("url");

        var parsedUrl = url.parse(event.ResponseURL);
        var options = {
            hostname: parsedUrl.hostname,
            port: 443,
            path: parsedUrl.path,
            method: "PUT",
            headers: {
                "content-type": "",
                "content-length": responseBody.length
            }
        };

        var request = https.request(options, function(response) {
            console.log("Status code: " + parseInt(response.statusCode));
            resolve(context.done());
        });

        request.on("error", function(error) {
            console.log("send(..) failed executing https.request(..): " +
 maskCredentialsAndSignature(error));
            reject(context.done(error));
        });

        request.write(responseBody);
        request.end();
    })
}

function maskCredentialsAndSignature(message) {
```

```
    return message.replace(/X-Amz-Credential=[^&\s]+/i, 'X-Amz-Credential=*****')
        .replace(/X-Amz-Signature=[^&\s]+/i, 'X-Amz-Signature=*****');
}
```

Node.js 源代码

以下是在处理程序不是异步进行处理时，Node.js 函数的响应模块源代码。请查看这些内容，以理解此模块的功能并帮助您实现自己的响应函数。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

exports.SUCCESS = "SUCCESS";
exports.FAILED = "FAILED";

exports.send = function(event, context, responseStatus, responseData,
 physicalResourceId, noEcho) {

    var responseBody = JSON.stringify({
        Status: responseStatus,
        Reason: "See the details in CloudWatch Log Stream: " + context.logStreamName,
        PhysicalResourceId: physicalResourceId || context.logStreamName,
        StackId: event.StackId,
        RequestId: event.RequestId,
        LogicalResourceId: event.LogicalResourceId,
        NoEcho: noEcho || false,
        Data: responseData
    });

    console.log("Response body:\n", responseBody);

    var https = require("https");
    var url = require("url");

    var parsedUrl = url.parse(event.ResponseURL);
    var options = {
        hostname: parsedUrl.hostname,
        port: 443,
        path: parsedUrl.path,
        method: "PUT",
        headers: {
            "content-type": "",
            "content-length": responseBody.length
```

```
        }
    };

    var request = https.request(options, function(response) {
        console.log("Status code: " + parseInt(response.statusCode));
        context.done();
    });

    request.on("error", function(error) {
        console.log("send(..) failed executing https.request(..): " +
 maskCredentialsAndSignature(error));
        context.done();
    });

    request.write(responseBody);
    request.end();
}
```

Python 源代码

以下是 Python 函数的响应模块源代码。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0

from __future__ import print_function
import urllib3
import json
import re

SUCCESS = "SUCCESS"
FAILED = "FAILED"

http = urllib3.PoolManager()


def send(event, context, responseStatus, responseData, physicalResourceId=None,
 noEcho=False, reason=None):
    responseUrl = event['ResponseURL']

    responseBody = {
        'Status' : responseStatus,
        'Reason' : reason or "See the details in CloudWatch Log Stream:
 {}".format(context.log_stream_name),
```

```
        'PhysicalResourceId' : physicalResourceId or context.log_stream_name,
        'StackId' : event['StackId'],
        'RequestId' : event['RequestId'],
        'LogicalResourceId' : event['LogicalResourceId'],
        'NoEcho' : noEcho,
        'Data' : responseData
    }

    json_responseBody = json.dumps(responseBody)

    print("Response body:")
    print(json_responseBody)

    headers = {
        'content-type' : '',
        'content-length' : str(len(json_responseBody))
    }

    try:
        response = http.request('PUT', responseUrl, headers=headers,
 body=json_responseBody)
        print("Status code:", response.status)


    except Exception as e:

        print("send(..) failed executing http.request(..):",
 mask_credentials_and_signature(e))


def mask_credentials_and_signature(message):
    message = re.sub(r'X-Amz-Credential=[^&\s]+', 'X-Amz-Credential=*****', message,
 flags=re.IGNORECASE)
    return re.sub(r'X-Amz-Signature=[^&\s]+', 'X-Amz-Signature=*****', message,
 flags=re.IGNORECASE)
```

使用模板宏对 CloudFormation 模板执行自定义处理

您可以使用宏对模板执行自定义处理，包括查找并替换操作等简单操作以及整个模板的大型转换。

要了解可能性的广度，请考虑 `AWS::Include` 和 `AWS::Serverless` 转换，这些转换是由 CloudFormation 托管的宏：

- AWS::Include 转换可让您将样板文件模板代码段插入您的模板中。
- AWS::Serverless 转换采用以 Amazon Serverless Application Model（Amazon SAM）语法编写的完整模板，将其转换并扩展为符合规范的 CloudFormation 模板。有关无服务器应用程序和 Amazon SAM 的更多信息，请参阅 Amazon Serverless Application Model 开发人员指南。

主题

- 计费
- 宏示例
- 相关资源
- CloudFormation 宏概述
- 创建 CloudFormation 宏定义
- 简单字符串替换宏的示例
- 对所处理的模板进行故障排除

计费

当宏运行时，将对 Lambda 函数的所有者收取与执行该函数相关的任何费用。

`AWS::Include` 和 `AWS::Serverless` 转换是由 CloudFormation 托管的宏。使用它们不产生任何费用。

宏示例

除了本部分的示例之外，您还可以在 GitHub 存储库中找到示例宏，包括源代码和模板。这些示例按“原样”提供，用于教学目的。

相关资源

- AWS::CloudFormation::Macro
- CloudFormation 模板 Transform 部分
- Fn::Transform
- AWS::Serverless 转换
- AWS::Include 转换

CloudFormation 宏概述

使用宏处理模板有两个主要步骤：创建宏本身，然后使用宏来对模板执行处理。

要创建宏定义，您必须创建以下内容：

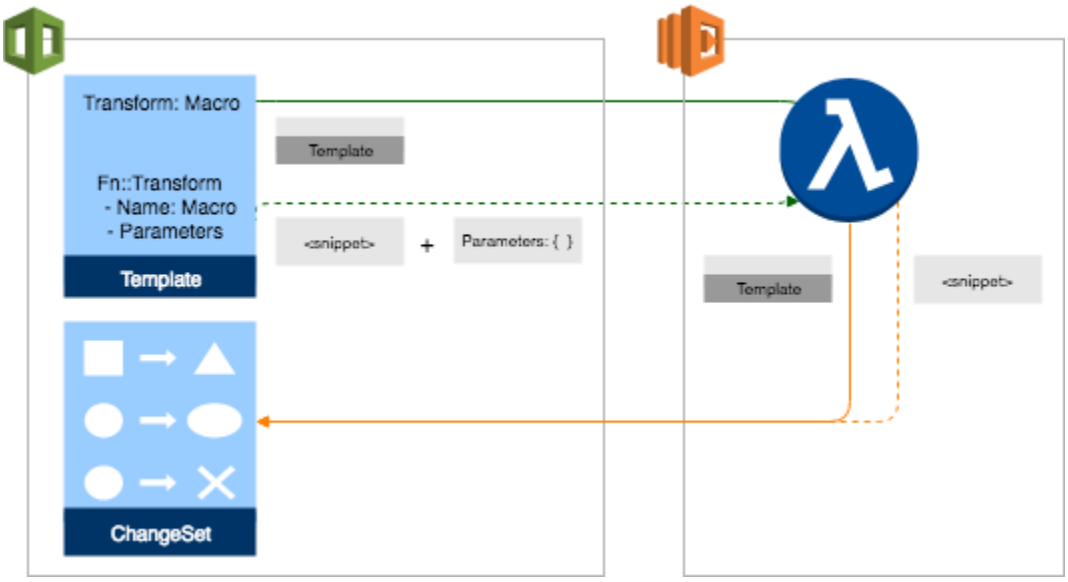
- 一个执行模板处理的 Lambda 函数。此 Lambda 函数接受一个代码段或整个模板，以及您定义的任何其他参数。它返回已处理的模板代码段或整个模板作为响应。
- AWS::CloudFormation::Macro 类型的资源，使用户能够从 CloudFormation 模板内调用 Lambda 函数。此资源指定要为此宏调用的 Lambda函数的 ARN，以及辅助调试的其他可选属性。要在账户中创建此资源，请创建一个包含 `AWS::CloudFormation::Macro` 资源的模板，然后利用该模板创建具有自行管理权限的堆栈或堆栈集。Amazon CloudFormationStackSets 目前不支持通过引用 宏的模板创建或更新具有服务托管权限的堆栈集。

要使用宏，请在您的模板中引用宏：

- 要处理模板的一段或部分，请在 `Fn::Transform` 函数中引用宏（位置是相对于要转换的模板内容而言的）。当使用 `Fn::Transform` 时，您还可以传递它需要的任何指定参数。
- 要处理整个模板，请在模板的 Transform 部分中引用宏。

接下来，您通常会创建一个更改集，然后执行它。（处理宏可能会添加您可能不知道的多个资源。为确保您了解宏所引入的所有更改，我们强烈建议您使用更改集。）CloudFormation 将指定的模板内容以及任何其他指定参数一并传递给在宏资源中指定的 Lambda 函数。Lambda 函数返回已处理的模板内容，无论是代码段还是整个模板。

在调用模板中的所有宏之后，CloudFormation 会生成包含已处理模板内容的更改集。检查更改集后，执行它来应用更改。



如何直接创建堆栈

要使用引用宏的模板创建或更新堆栈，通常需要创建一个更改集，然后执行它。更改集描述 CloudFormation 将基于处理后的模板执行的操作。处理宏可添加您可能不知道的多个资源。为确保您已了解宏所引入的所有更改，我们强烈建议您使用更改集。在查看更改集后，您可以执行它来实际应用更改。

宏可以将 IAM 资源添加到您的模板中。对于这些资源，CloudFormation 要求您确认其功能。由于 CloudFormation 无法在处理模板前了解所添加的资源，因此您可能需要在创建更改集时确认 IAM 功能，具体取决于引用的宏是否包含 IAM 资源。这样，当您运行该更改集时，CloudFormation 会具有创建 IAM 资源所需的功能。

要直接从处理的模板来创建或更新堆栈，而不事先查看更改集中提议的更改，可以在 `CreateStack` 或 `UpdateStack` 请求期间指定 `CAPABILITY_AUTO_EXPAND` 功能。仅当您知道宏在处理什么时，才应直接从包含宏的堆栈模板创建堆栈。您不能将更改集与堆栈集宏一起使用，必须直接更新堆栈集。

有关更多信息，请参阅《Amazon CloudFormation API 参考》 中的 CreateStack 或 UpdateStack。

> **Important**
>
> 如果堆栈集模板引用了一个或多个宏，则必须直接从处理过的模板创建堆栈集，而无需先查看更改集中生成的更改。处理宏可添加您可能不知道的多个资源。在通过直接引用宏的模板创建或更新堆栈集之前，请确保您知道宏执行的处理操作。

为减少从引用宏的模板启动堆栈的步骤数，可使用 `package` 和 `deploy` Amazon CLI 命令。有关更多信息，请参阅使用 Amazon CLI 将本地构件上传到 S3 存储桶和创建包含转换的堆栈。

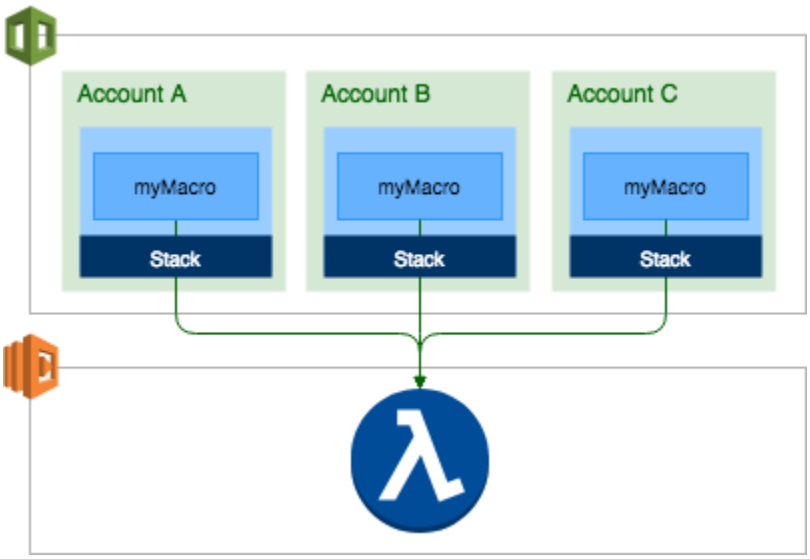
注意事项

使用宏时，请记住以下注意事项和限制：

- 只有在提供 Lambda 的 Amazon Web Services 区域才支持宏。有关提供 Lambda 的区域的列表，请参阅 Amazon Lambda 端点和配额。
- 任何已处理模板代码段都必须是有效的 JSON。
- 任何已处理的模板代码段都必须通过创建堆栈、更新堆栈、创建堆栈集或更新堆栈集等操作的验证检查。
- CloudFormation 会先解析宏，然后再处理模板。生成的模板必须是有效的 JSON，并且不得超出模板大小限制。
- 由于 CloudFormation 处理模板中元素的顺序，宏无法在其返回给 CloudFormation 的已处理模板内容中包含模块。有关更多信息，请参阅 宏评估顺序。
- 使用更新回滚功能时，CloudFormation 会使用原始模板的副本。即使包含的代码段发生更改，它也会回滚到原始模板。
- 宏中不能嵌套宏，因为我们不会迭代处理宏。
- 目前在宏中不支持 `Fn::ImportValue` 内部函数。
- 模板中包含的内部函数会在任何宏之后被评估。因此，宏返回的已处理模板内容可以包括对内部函数的调用，并且像往常一样对其进行评估。
- StackSets 目前不支持通过引用 CloudFormation 宏的模板，使用服务托管权限创建或更新堆栈集。

宏账户范围和权限

您只能在创建宏的账户中将宏用作资源。宏的名称在给定账户中必须是唯一的。但是，您可以通过在底层 Lambda 函数上启用跨账户访问，然后创建宏定义以在多个账户中引用该函数，从而在多个账户中提供相同的功能。在下面的示例中，三个账户包含宏定义，每个宏定义都指向相同的 Lambda 函数。



要创建宏定义，用户必须具有在指定账户中创建堆栈或堆栈集的权限。

要使 CloudFormation 成功运行模板中包含的宏，用户必须对底层 Lambda 函数具有 Invoke 权限。为防止可能的权限升级，CloudFormation 会在运行宏时模拟用户。

有关更多信息，请参阅《Amazon Lambda 开发人员指南》中的在 Amazon Lambda 中管理权限和《服务授权参考》中的 Amazon Lambda 操作、资源和条件键。

创建 CloudFormation 宏定义

创建宏定义时，该宏定义会使底层 Lambda 函数在指定的账户中可用，以便 CloudFormation 可以调用该函数来处理模板。

事件映射

当 CloudFormation 调用宏的 Lambda 函数时，它会以 JSON 格式发送请求，其结构如下所示：

```
{
    "region" : "us-east-1",
    "accountId" : "$ACCOUNT_ID",
    "fragment" : { ... },
    "transformId" : "$TRANSFORM_ID",
    "params" : { ... },
    "requestId" : "$REQUEST_ID",
    "templateParameterValues" : { ... }
}
```

- `region`

宏所在的区域。

- `accountId`

  宏从中调用 Lambda 函数的账户的账户 ID。

- `fragment`

  可用于自定义处理的模板内容，采用 JSON 格式。

  - 对于 `Transform` 模板部分中包含的宏，这是整个模板，除 `Transform` 部分外。
  - 对于在 `Fn::Transform` 内部函数调用中包含的宏，这包括基于内部函数在模板中的位置的所有同级节点（及其子节点），除 `Fn::Transform` 函数外。有关更多信息，请参阅 宏模板范围。

- `transformId`

  调用此函数的宏的名称。

- `params`

  对于 `Fn::Transform` 函数调用，是函数的任何指定参数。CloudFormation 在将这些参数传递给函数之前不会进行评估。

  对于在 `Transform` 模板部分中包含的宏，此部分为空。

- `requestId`

  调用此函数的请求的 ID。

- `templateParameterValues`

  在模板的 Parameters 部分中指定的任何参数。CloudFormation 在将这些参数传递给函数之前会进行评估。

响应格式

CloudFormation 要求 Lambda 函数以如下 JSON 格式返回响应：

```
{
    "requestId" : "$REQUEST_ID",
    "status" : "$STATUS",
    "fragment" : { ... },
    "errorMessage": "optional error message for failures"
}
```

- `requestId`

  调用此函数的请求的 ID。这必须在调用函数时与 CloudFormation 提供的请求 ID 匹配。

- `status`

  请求的状态（不区分大小写）。应设置为 `success`。CloudFormation 会将任何其他响应都视为失败。

- `fragment`

  CloudFormation 要包含在已处理模板中的已处理模板内容，包括同级内容。CloudFormation 会将传递给 Lambda 函数的模板内容替换为在 Lambda 响应中收到的模板片段。

  已处理模板内容必须是有效的 JSON，并且其在已处理模板中的包含必须生成有效的模板。

  如果函数实际上没有更改 CloudFormation 传递的模板内容，但您仍需要将该内容包含在已处理模板中，则函数需要将该模板内容在其响应中返回到 CloudFormation。

- `errorMessage`

  解释转换失败原因的错误消息。CloudFormation 将在堆栈的 Stack details（堆栈详细信息）页面的 Events（事件）窗格中显示此错误消息。

  例如：

```
Error creating change set: Transform
                           Amazon Web Services ## account
                           number::macro name failed with:
                           error message string.
```

创建宏定义

创建 CloudFormation 宏定义

1. 构建 Lambda 函数来处理模板内容。该函数可以处理模板的任何部分，甚至是整个模板。
2. 创建包含 `AWS::CloudFormation::Macro` 资源类型的 CloudFormation 模板，并指定 `Name` 和 `FunctionName` 属性。`FunctionName` 属性必须包含在 CloudFormation 运行宏时调用的 Lambda 函数的 ARN。
3. （可选）为了帮助调试，您还可以在为宏创建 `AWS::CloudFormation::Macro` 资源类型时指定 `LogGroupName` 和 `LogRoleArn` 属性。借助这些属性，您可以指定 CloudFormation 在调用宏

的底层 Lambda 函数时向其发送错误日志信息的 CloudWatch Logs 日志组，以及 CloudFormation 在将日志条目发送到这些日志时应代入的角色。

4. 在要使用宏的账户中使用带宏的模板创建堆栈。或在管理员账户中使用带宏的模板创建具有自行管理权限的堆栈集，然后在目标账户中创建堆栈实例。
5. 在 CloudFormation 成功创建包含宏定义的堆栈之后，该宏便可在这些账户中使用。您可以通过在模板中，在与要处理的模板内容相关的适当位置引用宏来使用宏。

宏模板范围

在模板的 `Transform` 部分中引用的宏可以处理该模板的全部内容。

在 `Fn::Transform` 函数中引用的宏可以处理模板中该 `Fn::Transform` 函数的任何同级元素（包括子元素）的内容。

例如，在下面的模板示例中，`AWS::Include` 可以根据包含它的 `Fn::Transform` 函数的位置处理 `MyBucket` 属性。`MyMacro` 可以处理整个模板的内容，因为它包含在 `Transform` 部分中。

```
# Start of processable content for MyMacro
AWSTemplateFormatVersion: 2010-09-09
 Transform: [MyMacro]
 Resources:
    WaitCondition:
      Type: AWS::CloudFormation::WaitCondition
    MyBucket:
      Type: AWS::S3::Bucket
      # Start of processable content for AWS::Include
      Properties:
        BucketName: amzn-s3-demo-bucket1
        Tags: [{"key":"value"}]
        'Fn::Transform':
          - Name: 'AWS::Include'
              Parameters:
                Location: s3://amzn-s3-demo-bucket2/MyFileName.yaml
        CorsConfiguration: []
        # End of processable content for AWS::Include
    MyEc2Instance:
      Type: AWS::EC2::Instance
      Properties:
        ImageID: ami-1234567890abcdef0
# End of processable content for MyMacro
```

宏评估顺序

您可以在给定模板中引用多个宏，包括由 CloudFormation 托管的转换，例如 AWS::Include 和 AWS::Serverless。

将会根据宏在模板中的位置，从嵌套最深的宏向外直到最一般的宏，按顺序评估宏。对于模板中相同位置的宏，将会根据列出的顺序按顺序进行评估。

诸如 `AWS::Include` 和 `AWS::Transform` 之类的转换在操作顺序和范围方面被视为与任何其他宏相同。

例如，在下面的模板示例中，CloudFormation 会首先评估 `PolicyAdder` 宏，因为这是模板中最深的嵌套宏。然后，CloudFormation 会首先评估 `MyMacro`，然后再评估 `AWS::Serverless`，因为在 `Transform` 部分中它列于 `AWS::Serverless` 之前。

```
AWSTemplateFormatVersion: 2010-09-09
 Transform: [MyMacro, AWS::Serverless]
 Resources:
    WaitCondition:
      Type: AWS::CloudFormation::WaitCondition
    MyBucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: amzn-s3-demo-bucket
        Tags: [{"key":"value"}]
        'Fn::Transform':
          - Name: PolicyAdder
        CorsConfiguration: []
    MyEc2Instance:
      Type: AWS::EC2::Instance
      Properties:
        ImageID: ami-1234567890abcdef0
```

简单字符串替换宏的示例

下面的示例演示了使用宏的过程：从在模板中定义宏，到为宏创建 Lambda 函数，再到在模板中使用宏。

在此示例中，我们创建一个简单的宏，它插入指定的字符串来替换已处理模板中的指定目标内容。然后我们将使用它在已处理模板的指定位置插入空白 `WaitHandleCondition`。

创建宏

在使用宏之前，我们首先要完成两件事：创建执行所需模板处理的 Lambda 函数，然后通过创建宏定义使该 Lambda 函数可用于 CloudFormation。

以下示例模板包含示例宏的定义。为了使宏在特定 Amazon Web Services 账户中可用，应从模板创建一个堆栈。宏定义指定宏名称、简要描述，并引用在模板中使用此宏时 CloudFormation 调用的 Lambda 函数的 ARN。（我们没有包含 `LogGroupName` 或 `LogRoleARN` 属性来用于记录错误日志。）

在本示例中，假定从此模板创建的堆栈名为 `JavaMacroFunc`。由于宏 `Name` 属性设置为堆栈名称，因此生成的宏也名为 `JavaMacroFunc`。

```
AWSTemplateFormatVersion: 2010-09-09
  Resources:
    Macro:
      Type: AWS::CloudFormation::Macro
      Properties:
        Name: !Sub '${AWS::StackName}'
        Description: Adds a blank WaitConditionHandle named WaitHandle
        FunctionName: 'arn:aws:lambda:us-east-1:012345678910:function:JavaMacroFunc'
```

使用宏

为了使用宏，我们将使用 `Fn::Transform` 内置函数将其包含在模板中。

当我们使用下面的模板创建堆栈时，CloudFormation 会调用我们的示例宏。底层 Lambda 函数将一个指定的字符串替换为另一个指定的字符串。在这种情况下，结果是空的 `AWS::CloudFormation::WaitConditionHandle` 插入到已处理模板中。

```
Parameters:
  ExampleParameter:
    Type: String
    Default: 'SampleMacro'

Resources:
  2a:
    Fn::Transform:
      Name: "JavaMacroFunc"
      Parameters:
        replacement: 'AWS::CloudFormation::WaitConditionHandle'
```

```
        target: '$$REPLACEMENT$$'
    Type: '$$REPLACEMENT$$'
```

- 要调用的宏被指定为 JavaMacroFunc，它来自上一个宏定义示例。
- 将会向该宏传递两个参数，target 和 replacement，它们代表目标字符串及其所需的替换值。
- 该宏可以对 Type 节点的内容进行操作，因为 Type 是引用该宏的 Fn::Transform 函数的同级节点。
- 生成的 AWS::CloudFormation::WaitConditionHandle 名为 2a。
- 该模板还包含一个该宏也可以访问的模板参数 ExampleParameter（但在这种情况下不使用）。

Lambda 输入数据

当 CloudFormation 在堆栈创建期间处理我们的示例模板时，它将以下事件映射传递给在 JavaMacroFunc 宏定义中引用的 Lambda 函数。

- region : us-east-1
- accountId : 012345678910
- fragment :

```
{
  "Type": "$$REPLACEMENT$$"
}
```

- transformId : 012345678910::JavaMacroFunc
- params :

```
{
    "replacement": "AWS::CloudFormation::WaitConditionHandle",
    "target": "$$REPLACEMENT$$"
}
```

- requestId : 5dba79b5-f117-4de0-9ce4-d40363bfb6ab
- templateParameterValues :

```
{
    "ExampleParameter": "SampleMacro"
}
```

`fragment` 包含 JSON，表示该宏可以处理的模板片段。此片段由 `Fn::Transform` 函数调用的同级函数调用组成，但不包括函数调用本身。此外，`params` 包含 JSON，表示宏参数。在这种情况下，是替换和目标。同样，`templateParameterValues` 包含 JSON，表示为整个模板指定的参数。

Lambda 函数代码

以下是 Lambda 函数的实际代码，它是 `JavaMacroFunc` 示例宏的底层。它迭代响应中包含的模板片段（无论是字符串、列表还是映射格式），从而查找指定的目标字符串。如果它找到指定的目标字符串，则 Lambda 函数会使用指定的替换字符串替换目标字符串。否则，该函数会保持模板片段不变。然后，该函数将下面详细讨论的预期属性的映射返回到 CloudFormation。

```
package com.macroexample.lambda.demo;

import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class LambdaFunctionHandler implements RequestHandler<Map<String, Object>,
 Map<String, Object>> {

 private static final String REPLACEMENT = "replacement";
 private static final String TARGET = "target";
 private static final String PARAMS = "params";
 private static final String FRAGMENT = "fragment";
 private static final String REQUESTID = "requestId";
 private static final String STATUS = "status";
 private static final String SUCCESS = "SUCCESS";
 private static final String FAILURE = "FAILURE";
    @Override
    public Map<String, Object> handleRequest(Map<String, Object> event, Context
 context) {
        // TODO: implement your handler
     final Map<String, Object> responseMap = new HashMap<String, Object>();
        responseMap.put(REQUESTID, event.get(REQUESTID));
        responseMap.put(STATUS, FAILURE);
     try {
         if (!event.containsKey(PARAMS)) {
          throw new RuntimeException("Params are required");
         }
```

```
        final Map<String, Object> params = (Map<String, Object>) event.get(PARAMS);
        if (!params.containsKey(REPLACEMENT) || !params.containsKey(TARGET)) {
         throw new RuntimeException("replacement or target under Params are
required");
        }

     final String replacement = (String) params.get(REPLACEMENT);
     final String target = (String) params.get(TARGET);
     final Object fragment = event.getOrDefault(FRAGMENT, new HashMap<String,
Object>());
     final Object retFragment;
     if (fragment instanceof String) {
      retFragment = iterateAndReplace(replacement, target, (String) fragment);
     } else if (fragment instanceof List) {
      retFragment = iterateAndReplace(replacement, target, (List<Object>) fragment);
     } else if (fragment instanceof Map) {
      retFragment = iterateAndReplace(replacement, target, (Map<String, Object>)
fragment);
     } else {
      retFragment = fragment;
     }
        responseMap.put(STATUS, SUCCESS);
        responseMap.put(FRAGMENT, retFragment);
        return responseMap;
    } catch (Exception e) {
     e.printStackTrace();
     context.getLogger().log(e.getMessage());
     return responseMap;
    }
   }

   private Map<String, Object> iterateAndReplace(final String replacement, final
String target, final Map<String, Object> fragment) {
    final Map<String, Object> retFragment = new HashMap<String, Object>();
    final List<String> replacementKeys = new ArrayList<>();
    fragment.forEach((k, v) -> {
     if (v instanceof String) {
      retFragment.put(k, iterateAndReplace(replacement, target, (String)v));
     } else if (v instanceof List) {
      retFragment.put(k, iterateAndReplace(replacement, target, (List<Object>)v));
     } else if (v instanceof Map ) {
      retFragment.put(k, iterateAndReplace(replacement, target, (Map<String, Object>)
v));
```

```
      } else {
       retFragment.put(k, v);
      }
     });
     return retFragment;
    }

    private List<Object> iterateAndReplace(final String replacement, final String
 target, final List<Object> fragment) {
     final List<Object> retFragment = new ArrayList<>();
     fragment.forEach(o -> {
      if (o instanceof String) {
       retFragment.add(iterateAndReplace(replacement, target, (String) o));
      } else if (o instanceof List) {
       retFragment.add(iterateAndReplace(replacement, target, (List<Object>) o));
      } else if (o instanceof Map) {
       retFragment.add(iterateAndReplace(replacement, target, (Map<String, Object>)
 o));
      } else {
       retFragment.add(o);
      }
     });
     return retFragment;
    }

    private String iterateAndReplace(final String replacement, final String target,
 final String fragment) {
     System.out.println(replacement + " == " + target + " == " + fragment );
     if (fragment != null AND_AND fragment.equals(target))
      return replacement;
     return fragment;
    }
}
```

Lambda 函数响应

以下是 Lambda 函数返回到 CloudFormation 进行处理的映射。

- `requestId` : `5dba79b5-f117-4de0-9ce4-d40363bfb6ab`
- `status` : `SUCCESS`
- `fragment` :

```
{
```

```
  "Type": "AWS::CloudFormation::WaitConditionHandle"
}
```

从 CloudFormation 发送的 `requestId` 匹配项，以及 SUCCESS 的 `status` 值表示 Lambda 函数成功处理了请求中包含的模板片段。在此响应中，`fragment` 包含 JSON，表示要插入到已处理模板中的内容，而不是原始模板代码段。

生成的已处理模板

在 CloudFormation 从 Lambda 函数收到成功响应后，它将返回的模板片段插入到已处理模板中。

下面是我们示例的生成的已处理模板。不再包括引用 `JavaMacroFunc` 宏的 `Fn::Transform` 内置函数调用。Lambda 函数返回的模板片段包含在适当的位置，结果是内容 `"Type": "$$REPLACEMENT$$"` 已替换为 `"Type": "AWS::CloudFormation::WaitConditionHandle"`。

```
{
    "Parameters": {
        "ExampleParameter": {
            "Default": "SampleMacro",
            "Type": "String"
        }
    },
    "Resources": {
        "2a": {
            "Type": "AWS::CloudFormation::WaitConditionHandle"
        }
    }
}
```

对所处理的模板进行故障排除

使用宏时，可以在 CloudFormation 控制台中找到所处理的模板。

模板的阶段指示其处理状态：

- `Original`：用户最初提交的用于创建或更新堆栈或堆栈集的模板。
- `Processed`：CloudFormation 在处理任何引用的宏之后，用于创建或更新堆栈或堆栈集的模板。即使原始模板格式化为 YAML，已处理模板也会格式化为 JSON。

要排除故障，可使用所处理的模板。如果模板不引用宏，则原始模板和已处理模板是相同的。

有关更多信息，请参阅 从 CloudFormation 控制台查看堆栈信息。

要使用 Amazon CLI 获取所处理的模板，应使用 get-template 命令。

大小限制

当处理的堆栈模板直接传递到 `CreateStack`、`UpdateStack` 或 `ValidateTemplate` 请求时，其最大大小为 51200 字节；或者当使用 Amazon S3 模板 URL 作为 S3 对象传递时，为 1MB。但是，在处理期间，CloudFormation 会更新模板的临时状态，因为它连续处理模板中包含的宏。因此，在处理期间，模板的大小可能会临时超出完全处理的模板所允许的大小。CloudFormation 允许为这些处理中的模板留出一些缓冲。但是，在设计模板和宏时，要始终考虑处理的堆栈模板所允许的最大大小。

如果在处理模板期间 CloudFormation 返回一个 `Transformation data limit exceeded` 错误，则说明您的模板超出了 CloudFormation 所允许的处理期间模板的最大大小。

要解决这一问题，可执行下列操作：

- 将您的模板分成多个模板，以避免超出为处理中的模板规定的最大大小。例如：
  - 使用嵌套堆栈模板来封装模板的不同部分。有关更多信息，请参阅 使用嵌套堆栈将模板拆分为可重复使用的部分。
  - 创建多个堆栈，并使用跨堆栈引用来相互交换信息。有关更多信息，请参阅 引用其他 CloudFormation 堆栈中的资源输出。
- 减小给定宏返回的模板片段大小。CloudFormation 不会篡改宏返回的片段内容。

使用嵌套堆栈将模板拆分为可重复使用的部分

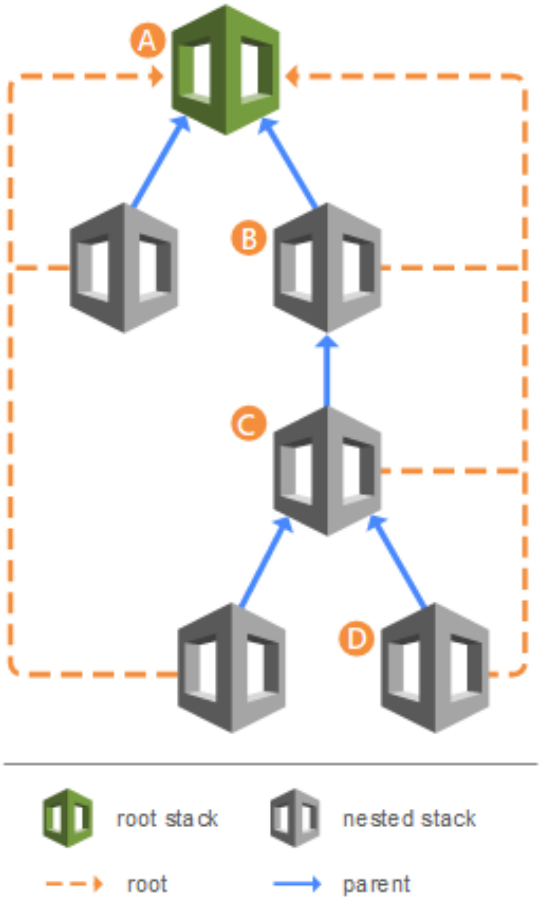
随着基础设施的发展，您可能会发现自己在多个模板中反复创建相同的资源配置。为避免这种冗余，您可以将这些常用配置分成专用模板。然后，您可以使用其他模板中的 AWS::CloudFormation::Stack 资源来引用这些专用模板，创建嵌套堆栈。

例如，假如您拥有用于大多数堆栈的负载均衡器配置。您可以为负载均衡器创建专用模板，而不是将相同的配置复制并粘贴到您的模板中。然后，您可以从需要相同负载均衡器配置的其他模板中引用此模板。

嵌套堆栈本身可以包含其他嵌套堆栈，构成一个堆栈层次结构，如下图所示。根堆栈是所有嵌套堆栈最终归属的顶级堆栈。每个嵌套堆栈都有一个直属父堆栈。对于第一级的嵌套堆栈而言，根堆栈也是父堆栈。

- 堆栈 A 是该层次结构中所有其他嵌套堆栈的根堆栈。

- 对于堆栈 B 来说，堆栈 A 既是父堆栈，也是根堆栈。
- 对于堆栈 D，堆栈 C 是父堆栈；而对于堆栈 C 来说，堆栈 B 是父堆栈。



主题

- 拆分模板之前和之后的示例
- 嵌套堆栈架构的示例
- 在嵌套堆栈上执行堆栈操作
- 相关信息
- Express Mode 和嵌套堆栈

拆分模板之前和之后的示例

此示例演示如何提取单个大型 CloudFormation 模板，然后使用嵌套模板将其重组为结构化程度更高、可重复使用的设计。最初，“嵌套堆栈前”模板显示在一个文件中定义了所有资源。随着资源数量不断增

长，该文件可能会变得混乱且难以管理。“嵌套堆栈后”模板将资源拆分为若干较小的独立模板。每个嵌套堆栈都处理一组特定的相关资源，从而使整体结构更有条理，更易于维护。

嵌套堆栈前

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  InstanceType:
    Type: String
    Default: t2.micro
    Description: The EC2 instance type

  Environment:
    Type: String
    Default: Production
    Description: The deployment
 environment

Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
      InstanceType: !Ref InstanceType

  MyS3Bucket:
    Type: AWS::S3::Bucket
```

嵌套堆栈后

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyFirstNestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazona
ws.com/ amzn-s3-demo-bucket  /first-ne
sted-stack.yaml
      Parameters:
        # Pass parameters to the
 nested stack if needed
        InstanceType: t3.micro

  MySecondNestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazona
ws.com/ amzn-s3-demo-bucket  /second-n
ested-stack.yaml
      Parameters:
        # Pass parameters to the
 nested stack if needed
        Environment: Testing
    DependsOn: MyFirstNestedStack
```

嵌套堆栈架构的示例

本部分演示了一种嵌套堆栈架构，该架构由引用嵌套堆栈的顶级堆栈组成。嵌套堆栈部署 Node.js Lambda 函数，从顶级堆栈接收参数值，并返回通过顶级堆栈公开的输出。

主题

- 步骤 1：在本地系统上为嵌套堆栈创建模板
- 步骤 2：在本地系统上为顶级堆栈创建模板
- 步骤 3：打包并部署模板

步骤 1：在本地系统上为嵌套堆栈创建模板

以下示例显示嵌套堆栈模板的格式。

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Nested stack template for Lambda function deployment
Parameters:
  MemorySize:
    Type: Number
    Default: 128
    MinValue: 128
    MaxValue: 10240
    Description: Lambda function memory allocation (128-10240 MB)
Resources:
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: !Sub "${AWS::StackName}-Function"
      Runtime: nodejs18.x
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile: |
          exports.handler = async (event) => {
            return {
              statusCode: 200,
              body: JSON.stringify('Hello from Lambda!')
            };
          };
      MemorySize: !Ref MemorySize
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
```

```
Outputs:
  LambdaArn:
    Description: ARN of the created Lambda function
    Value: !GetAtt LambdaFunction.Arn
```

步骤 2：在本地系统上为顶级堆栈创建模板

以下示例显示了顶层堆栈模板的格式以及引用您在上一步中所创建堆栈的 AWS::CloudFormation::Stack 资源。

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Top-level stack template that deploys a nested stack
Resources:
  NestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: /path_to_template/nested-template.yaml
      Parameters:
        MemorySize: 256
Outputs:
  NestedStackLambdaArn:
    Description: ARN of the Lambda function from nested stack
    Value: !GetAtt NestedStack.Outputs.LambdaArn
```

步骤 3：打包并部署模板

> **Note**
>
> 在本地使用模板时，Amazon CLI package 命令可以帮助您准备部署模板。它会自动处理将本地构件上传到 Amazon S3（包括 TemplateURL），并生成一个新的模板文件，其中包含对这些 S3 位置的更新引用。有关更多信息，请参阅 使用 Amazon CLI 将本地构件上传到 S3 存储桶。

接下来，您可以使用 package 命令将嵌套模板上传到 Amazon S3 存储桶。

```
aws cloudformation package \
  --s3-bucket amzn-s3-demo-bucket \
  --template /path_to_template/top-level-template.yaml \
  --output-template-file packaged-template.yaml \
```

```
  --output json
```

该命令在 `--output-template-file` 指定的路径上生成一个新模板。它将 `TemplateURL` 引用替换为 Amazon S3 位置，如下图所示。

结果模板

```
AWSTemplateFormatVersion: 2010-09-09
Description: Top-level stack template that deploys a nested stack
Resources:
  NestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/8b3bb7aa7abfc6e37e2d06b869484bed.template
      Parameters:
        MemorySize: 256
Outputs:
  NestedStackLambdaArn:
    Description: ARN of the Lambda function from nested stack
    Value:
      Fn::GetAtt:
      - NestedStack
      - Outputs.LambdaArn
```

运行 package 命令后，您可以使用 deploy 命令部署已处理的模板。对于包含 IAM 资源的嵌套堆栈，您必须通过包含 `--capabilities` 选项来确认 IAM 功能。

```
aws cloudformation deploy \
  --template-file packaged-template.yaml \
  --stack-name stack-name \
  --capabilities CAPABILITY_NAMED_IAM
```

在嵌套堆栈上执行堆栈操作

使用嵌套堆栈时，在操作过程中必须谨慎处理。某些堆栈操作（如堆栈更新等）应从根堆栈启动，而不是直接在嵌套堆栈上执行。更新根堆栈时，只有包含模板更改的嵌套堆栈才会更新。

此外，嵌套堆栈的存在可能会影响根堆栈上的操作。例如，如果一个嵌套堆栈卡在 UPDATE_ROLLBACK_IN_PROGRESS 状态，则根堆栈将等待该嵌套堆栈完成其回滚后再继续。在继续更新操作之前，请确保您拥有 IAM 权限，以在其回滚时取消堆栈更新。有关更多信息，请参阅 使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限。

按照以下过程查找根堆栈和嵌套堆栈。

查看嵌套堆栈的根堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面上，选择您要查看其根堆栈的嵌套堆栈的名称。

   嵌套堆栈的堆栈名称上方会显示 NESTED。
3. 在堆栈信息选项卡的概述部分中，选择列为根堆栈的堆栈名称。

查看属于根堆栈的嵌套堆栈

1. 从要查看其嵌套堆栈的根堆栈中，选择资源选项卡。
2. 在类型列中，查找类型为 AWS::CloudFormation::Stack 的资源。

相关信息

- 嵌套现有的堆栈
- 理解堆栈资源的更新行为
- 继续回滚失败的嵌套堆栈更新
- 嵌套堆栈回滚失败

Express Mode 和嵌套堆栈

当您在父堆栈上使用“Express Mode”时，该设置会传播到层次结构中的所有嵌套堆栈。您不需要为每个嵌套堆栈分别指定 Express Mode。一旦 CloudFormation 应用配置，整个层次结构中的所有资源都会立即完成。

有关 Express Mode 的更多信息，请参阅 Express Mode。

在 CloudFormation 模板中创建等待条件

本主题介绍了如何在模板中创建等待条件以协调堆栈资源的创建或跟踪配置过程的进度。例如，您可以在应用程序配置部分完成后开始另一个资源的创建，或者也可以在安装和配置过程中发送信号以跟踪其进度。

当 CloudFormation 创建含有等待条件的堆栈时：

- 它会跟其他任何资源一样创建等待条件，并将等待条件的状态设置为 CREATE_IN_PROGRESS。
- CloudFormation 等待至收到必要数量的成功信号或等待条件的超时时段到期。
- 如果在超时时段到期之前收到了必要数量的成功信号：
  - 等待条件的状态更改为 CREATE_COMPLETE
  - 堆栈创建继续
- 如果超时到期或收到了故障信号：
  - 等待条件的状态更改为 CREATE_FAILED
  - 堆栈回滚

Important

对于 Amazon EC2 和 Auto Scaling 资源，我们建议您使用 CreationPolicy 属性而非等待条件。当实例创建过程成功完成后，将“CreationPolicy”属性添加到这些资源，并使用 cfn-signal 帮助程序脚本发送信号。
有关更多信息，请参阅 CreationPolicy 属性。

Note

如果使用 Amazon PrivateLink，则响应等待条件的 VPC 中的资源必须能够访问 CloudFormation 特定的 Amazon Simple Storage Service (Amazon S3) 存储桶。资源必须将等待条件响应发送到预签名 Amazon S3 URL。如果这些资源不能向 Amazon S3 发送响应，则 CloudFormation 将不会收到响应，并且堆栈操作会失败。有关更多信息，请参阅使用接口端点 (Amazon PrivateLink) 访问 Amazon CloudFormation以及使用存储桶策略控制从 VPC 端点的访问。

主题

- 在模板中创建等待条件
- 等待条件信号语法
- 访问信号数据

在模板中创建等待条件

1. 等待条件句柄

首先在堆栈模板中定义 AWS::CloudFormation::WaitConditionHandle 资源。该资源会生成发送信号所需的预签名 URL。这样您就可以发送信号，而无需提供 Amazon 凭证。例如：

```
Resources:
  MyWaitHandle:
    Type: AWS::CloudFormation::WaitConditionHandle
```

2. 等待条件

接下来，在堆栈模板中定义 AWS::CloudFormation::WaitCondition 资源。AWS::CloudFormation::WaitCondition 的基本结构如下所示：

```
  MyWaitCondition:
    Type: AWS::CloudFormation::WaitCondition
    Properties:
      Handle: String
      Timeout: String
      Count: Integer
```

AWS::CloudFormation::WaitCondition 资源有两个必要属性和一个可选属性。

- Handle（必要）：对模板中声明的 WaitConditionHandle 的引用。
- Timeout（必要）：CloudFormation 在收到必要数量的信号之前等待的秒数。Timeout 是一种最低时限属性，表示超时不会早于您指定的时间发生，但可能在指定时间后不久发生。您可指定的最长时间为 43200 秒 (12 小时)。
- Count（可选）：CloudFormation 在将等待条件状态设置为 CREATE_COMPLETE 并继续创建堆栈前之前必须收到的成功信号数量。如果未指定，则默认值为 1。

通常，您想要等待条件在特定资源创建后立即开始。可以通过为等待条件添加 DependsOn 属性来完成此操作。向等待条件添加 DependsOn 属性时，CloudFormation 先在 DependsOn 属性中创建资源，然后创建等待条件。有关更多信息，请参阅 DependsOn 属性。

以下示例展示等待条件：

- 在成功创建 MyEC2Instance 资源后开始

- 将 MyWaitHandle 资源用作 WaitConditionHandle
- 超时为 4500 秒
- 默认 Count 为 1（因为未指定任何 Count 属性）

```
MyWaitCondition:
  Type: AWS::CloudFormation::WaitCondition
  DependsOn: MyEC2Instance
  Properties:
    Handle: !Ref MyWaitHandle
    Timeout: '4500'
```

3. 发送信号

为向 CloudFormation 发出成功或失败的信号，通常需要运行一些代码或脚本。例如，在 EC2 实例上运行的应用程序可能会执行一些额外的配置工作，然后向 CloudFormation 发出信号指示已完成。

信号必须发送至由等待条件句柄生成的预签名 URL。使用预签名 URL 发送成功或失败信号。

发送信号

1. 要检索模板中的预签名 URL，需使用带有等待条件句柄逻辑名称的 Ref 内置函数。

   如下例所示，您的模板可声明一个 Amazon EC2 实例，并使用 Amazon EC2 UserData 属性将预签名 URL 传递到该 EC2 实例。这样在这些实例上运行的脚本或应用程序就可以向 CloudFormation 发出成功或失败的信号。

   ```
   MyEC2Instance:
     Type: AWS::EC2::Instance
     Properties:
     InstanceType: t2.micro  # Example instance type
     ImageId: ami-055e3d4f0bbeb5878  # Change this as needed (Amazon Linux 2023 in
   us-west-2)
     UserData:
       Fn::Base64:
         Fn::Join:
           - ""
           - - "SignalURL="
             - { "Ref": "MyWaitHandle" }
   ```

   这会产生类似于下述信息的 UserData 输出：

```
SignalURL=https://amzn-s3-demo-bucket.s3.amazonaws.com/....
```

注意：在 Amazon Web Services 管理控制台和命令行工具中，预签名 URL 显示为等待条件句柄资源的物理 ID。

2. （可选）要检测堆栈进入等待条件的时间，可以使用以下方法之一：

   - 如果您通过启用的通知创建堆栈，那么 CloudFormation 将向指定主题发布针对每项堆栈事件的通知。如果您或您的应用程序订阅了该主题，您可以监测针对等待条件句柄创建事件的通知，并通过通知消息来检索预签名 URL。
   - 您还可以使用 Amazon Web Services 管理控制台、Amazon CLI 或 SDK 来监控堆栈事件。

3. 您需要通过预签名 URL 发送 HTTP 请求消息之后，才能发送信号。请求方法必须为 PUT，并且 `Content-Type` 标题必须为空字符串或省略。请求消息必须为 [等待条件信号语法] 指定表单的 JSON 结构。

   您必须发送由 `Count` 属性指定的成功信号数，以便 CloudFormation 继续创建堆栈。如果您的 `Count` 超过 1，则在发送到特定等待条件的所有信号中，每个信号的 `UniqueId` 值必须是唯一的。`UniqueId` 是随机的字母数字字符串。

   `curl` 命令是发送信号的一种方式。以下示例显示了向等待条件发送成功信号的 `curl` 命令行。

```
$ curl -T /tmp/a \
  "https://amzn-s3-demo-bucket.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-
west-2%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-
WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVo
%3D"
```

   其中文件 */tmp/a* 包含以下 JSON 结构：

```
{
   "Status" : "SUCCESS",
   "Reason" : "Configuration Complete",
   "UniqueId" : "ID1234",
   "Data" : "Application has completed configuration."
}
```

   此示例显示发送同一成功信号的 `curl` 命令行，不同的是，它将 JSON 结构作为命令行参数发送。

```
$ curl -X PUT \
  -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS","Reason" : "Configuration
 Complete","UniqueId" : "ID1234","Data" : "Application has completed
 configuration."}' \
  "https://amzn-s3-demo-bucket.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-
west-2%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-
WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVo
%3D"
```

等待条件信号语法

在向等待条件句柄生成的 URL 发送信号时，您必须使用以下 JSON 格式：

```
{
  "Status" : "StatusValue",
  "UniqueId" : "Some UniqueId",
  "Data" : "Some Data",
  "Reason" : "Some Reason"
}
```

属性

`Status` 字段必须是以下任一值：

- `SUCCESS`
- `FAILURE`

`UniqueId` 字段标识发送至 CloudFormation 的信号。如果等待条件的 `Count` 属性大于 1，那么在针对特定等待条件发送的所有信号中，`UniqueId` 值必须唯一；否则，CloudFormation 会将该信号视为对先前已发送的具有相同 `UniqueId` 的信号的重传，并将其忽略。

`Data` 字段可包含要通过信号发回的任何信息。可以通过使用模板中的 Fn::GetAtt 函数来访问 `Data` 值。

`Reason` 字段为字符串，除与 JSON 格式相一致外，其内容中无任何其他限制。

访问信号数据

要访问由有效信号发送的数据，可以在 CloudFormation 模板中为等待条件创建输出值。例如：

```
Outputs:
  WaitConditionData:
    Description: The data passed back as part of signalling the WaitCondition
    Value: !GetAtt MyWaitCondition.Data
```

然后可以使用 describe-stacks 命令或 CloudFormation 控制台的输出选项卡来查看该数据。

`Fn::GetAtt` 函数将返回 `UniqueId` 和 `Data` 作为 JSON 结构中的名称/值对。例如：

```
{"Signal1":"Application has completed configuration."}
```

使用 CloudFormation 模块创建可跨模板包含的可重复使用的资源配置

模块是一种供您以透明、易管理和可重复的方式打包资源配置以便跨堆栈模板实现包含的方法。模块可以将常见服务配置和最佳实践封装为模块化、可自定义的构建基块，供您包含在堆栈模板中。模块使您能够将包含最佳实践、专家领域知识和可接受的指南（例如安全性、合规性、治理和行业法规）在内的资源配置纳入模板中，而无需深入了解资源实施的复杂性。

例如，网络领域专家可以创建一个包含遵守安全准则的内置安全组和入口/出口规则的模块。然后，您可以将该模块纳入到模板中，以便在堆栈中预置安全的网络基础架构，而无需花费时间弄清楚 VPC、子网、安全组和网关的工作方式。此外，由于模块具有版本控制，因此如果安全指南随时间而发生变化，模块作者可以创建包含这些更改的新版本模块。

在模板中使用模块的特点包括：

- 可预测性 – 模块必须遵守其在 CloudFormation 注册表中注册的架构，这样在您将模块纳入到模板中后，您就知道该模块可以解析到哪些资源。
- 可重用性 – 您可以在多个模板和账户中使用同一模块。
- 可追溯性 – CloudFormation 可以保留堆栈中的哪些资源是从模块预配置的知识，使您能够轻松了解资源变化的来源。
- 可管理性：注册模块后，您可以通过 CloudFormation 注册表对其进行管理，包括版本控制以及账户和区域可用性。

模块可以包含：

- 要从模块配置的一个或多个资源以及任何关联的数据，例如输出数据或条件。
- 使您能够在使用模块时指定自定义值的任何模块参数。

有关开发模块的信息，请参阅《CloudFormation CLI User Guide》中的 Developing modules。

主题

- 使用模块时的注意事项
- 了解模块版本控制
- 使用 CloudFormation 私有注册表中的模块
- 使用参数指定模块值
- CloudFormation 模板中的引用模块资源

使用模块时的注意事项

- 使用模块不收取任何额外费用。您只需为模块解析到堆栈中的资源付费。
- CloudFormation 配额（例如堆栈中允许的最大资源数或模板正文的最大大小）适用于处理过的模板，无论该模板中包含的资源是否来自模块。有关更多信息，请参阅 了解 CloudFormation 配额。
- 您在堆栈级别指定的标签将分配给派生自模块的各个资源。
- CloudFormation 处理模板时，在模块级别指定的帮助程序脚本不会传播到模块中包含的各个资源。
- 模块中指定的输出将在模板级别传播到输出。

  将为每个输出分配一个逻辑 ID，该 ID 是模块逻辑名称和模块中定义的输出名称的拼接。有关更多信息，请参阅 获取从已部署的 CloudFormation 堆栈导出的输出。
- 模块中指定的参数不会传播到模板级别的参数。

  但是，您可以创建引用模块级参数的模板级参数。有关更多信息，请参阅 使用参数指定模块值。

了解模块版本控制

CloudFormation 注册表充当存储库，您可以在其中注册和管理模块，以供在您的 Amazon Web Services 账户和区域内使用。您可以在您的账户和区域内注册来自各种来源的模块，包括 Amazon、第三方发布者和您自己的自定义扩展。有关更多信息，请参阅 通过 CloudFormation 注册表管理扩展。

模块可以有不同的版本，因此您可以指定要使用的模块版本。当您需要更新或修改模块而不破坏依赖该模块的现有堆栈时，此版本控制功能特别有用。

使用模块的多个版本时，请注意以下事项：

- 在堆栈操作期间，CloudFormation 会使用当前在执行堆栈操作的 Amazon Web Services 账户和区域中注册为默认版本的模块版本。包括嵌套在其他模块中的模块。

  因此，请注意，如果您在不同账户或区域中将同一模块的不同版本注册为默认版本，则使用同一个模板可能会导致不同的结果。
- 在堆栈操作期间，CloudFormation 会使用当前在执行堆栈操作的 Amazon Web Services 账户和区域中注册为默认版本的资源版本。这包括通过纳入模块生成的资源。
- 更改模块的默认版本不会启动任何堆栈更新操作。但是，在您下次使用包含该模块的任何模板执行堆栈操作（例如堆栈更新）时，CloudFormation 将在操作中使用新的默认版本。

  但在指定了使用以前的模板选项的情况下执行堆栈更新时例外，如下所述。
- 对于堆栈更新操作，如果您指定使用以前的模板选项，则 CloudFormation 会使用先前处理过的模板进行堆栈更新，不会针对您可能对模块作出的任何更改重新处理模块。
- 为了保证结果统一，如果要在堆栈模板中纳入模块以用于堆栈集，则应确保在计划部署堆栈实例的所有账户和区域中将模块的同一版本设置为默认版本。这包括嵌套在其他模块中的模块。有关更多信息，请参阅 使用 StackSets 跨账户和区域管理堆栈。

激活第三方公共模块的要求

要在您的账户和区域中成功激活第三方公有模块，模块中包含的每个第三方公有扩展（资源或模块）必须满足以下条件：

- 扩展激活：必须在您要使用扩展的账户和区域中激活该扩展。有关更多信息，请参阅 使用 CloudFormation 注册表中的第三方公有扩展。
- 别名注册：如果模块中的扩展使用类型名称别名，则该扩展必须使用相同的类型名称别名在您的账户和区域中注册。有关更多信息，请参阅 使用别名以引用扩展。
- 版本兼容性：当前激活的扩展版本必须是模块中指定的该扩展支持的主要版本之一。

如果您没有激活正确的第三方公有扩展和扩展版本，CloudFormation 将使操作失败，并显示一条错误消息，其中列出在成功激活该模块之前需要激活的扩展和版本。

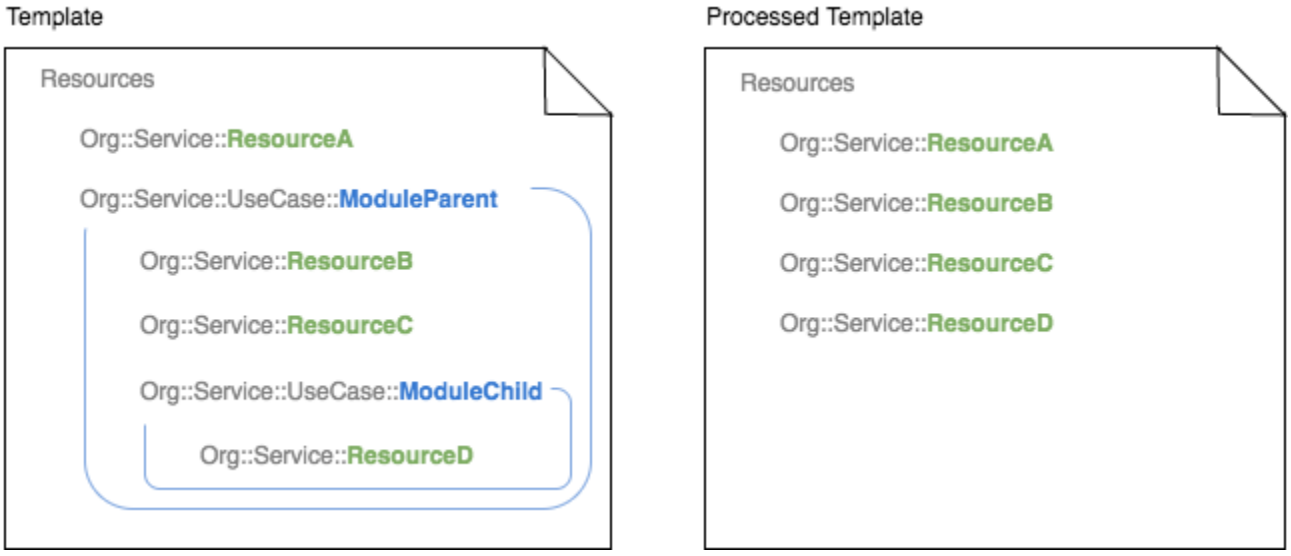
使用 CloudFormation 私有注册表中的模块

本主题介绍如何在 CloudFormation 模板中使用模块。将模块视为可以添加到模板中的预制资源包。

要使用模块，步骤如下：

- 注册模块：您在 CloudFormation 注册表中将模块注册为私有扩展。请确保模块已在您工作的 Amazon Web Services 账户和区域注册。有关更多信息，请参阅 CoudFormation 注册表概念。
- 将其包含在模板中：将模块添加到 CloudFormation 模板的 Resources 部分，就像处理其他资源一样。您还需要提供模块所需的所有属性。
- 创建或更新堆栈：当您启动堆栈操作时，CloudFormation 会生成一个已处理的模板，该模板将所有包含的模块解析为适当的资源。
- 预览更改：在进行更改之前，您可以使用更改集来查看将添加或更改哪些资源。有关更多信息，请参阅 使用更改集更新 CloudFormation 堆栈。

参阅以下示例：您有一个包含资源和模块的模板。模板包含一个单独的资源 `ResourceA` 和一个模块 `ModuleParent`。该模块包含两个资源（`ResourceB` 和 `ResourceC`）以及一个嵌套模块 `ModuleChild`。`ModuleChild` 包含单个资源 `ResourceD`。如果您使用此模板创建堆栈，CloudFormation 会处理模板并将模块解析为适当的资源。生成的堆栈有四个资源：`ResourceA`、`ResourceB`、`ResourceC` 和 `ResourceD`。



CloudFormation 会跟踪堆栈中的哪些资源是从模块创建而成的。您可以在给定堆栈的 Events（事件）、Resources（资源）和 Drifts（偏差）选项卡上查看此信息，更改集预览中也包含此信息。

模块与模板中的资源可以区别开来，因为它们遵循以下四部分命名约定，而不是资源所用的典型的三部分命名约定：

```
organization::service::use-case::MODULE
```

使用参数指定模块值

在 CloudFormation 中，您可以通过在堆栈创建或更新期间提供输入值，从而使用模板参数来自定义堆栈。这些参数允许您根据需要更改堆栈的某些方面。有关定义模板参数的更多信息，请参阅 CloudFormation 模板 Parameters 语法。

同样，模块也可以有参数。这些模块参数使您能够从使用自定义值的模板（或其他模块）向模块输入自定义值。然后，模块可以使用这些自定义值来设置其所包含资源的属性值。

您还可以定义设置模块属性的模板参数，以便您可以输入在堆栈操作时传递给模块的值。

如果模块包含具有自己的模块参数的嵌套模块，则您可以：

- 直接在父模块中为嵌套模块的参数指定值。
- 在父模块中定义相应的模块参数，使嵌套模块的参数能够由包含父模块的模板（或模块）来设置。

使用模板参数指定模块参数值

以下示例展示了如何定义将值传递给模块的模板参数。

包含 `My::S3::SampleBucket::MODULE` 的此模板定义了模板参数 `BucketName`，该参数使用户能够在堆栈操作期间指定 S3 存储桶名称。

```
# Template containing My::S3::SampleBucket::MODULE
Parameters:
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
  MyBucket:
    Type: 'My::S3::SampleBucket::MODULE'
    Properties:
      BucketName: !Ref BucketName
```

在父模块中指定子模块中资源的属性

以下示例展示了如何在嵌套在另一个模块中的模块中指定参数值。

第一个模块 My::S3::SampleBucketPrivate::MODULE 将是子模块。该模块定义了两个参数：BucketName 和 AccessControl。为这些参数指定的值用于指定模块所包含的 AWS::S3::Bucket 资源的 BucketName 和 AccessControl 属性。以下为 My::S3::SampleBucketPrivate::MODULE 的模板片段。

```
# My::S3::SampleBucketPrivate::MODULE
AWSTemplateFormatVersion: 2010-09-09
Description: A sample S3 Bucket with Versioning and DeletionPolicy.
Parameters:
  BucketName:
    Description: Name for the bucket
    Type: String
  AccessControl:
    Description: AccessControl for the bucket
    Type: String
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref BucketName
      AccessControl: !Ref AccessControl
      DeletionPolicy: Retain
      VersioningConfiguration:
        Status: Enabled
```

接下来，上一个模块嵌套在父模块 My::S3::SampleBucket::MODULE 中。父模块 My::S3::SampleBucket::MODULE 通过以下方式设置子模块参数：

- 其将 My::S3::SampleBucketPrivate::MODULE 的 AccessControl 参数设置为 Private。
- 对于 BucketName，其定义了一个模块参数，该参数将允许存储桶名称在包含 My::S3::SampleBucket::MODULE 的模板（或模块）中进行指定。

```
# My::S3::SampleBucket::MODULE
AWSTemplateFormatVersion: 2010-09-09
Description: A sample S3 Bucket. With Private AccessControl.
Parameters:
```

```
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
  MyBucket:
    Type: 'My::S3::SampleBucketPrivate::MODULE'
    Properties:
      BucketName: !Ref BucketName
      AccessControl: Private
```

为模块参数指定约束

模块参数不支持约束强制执行。要对模块参数执行约束检查，请创建具有所需约束的模板参数，然后在模块参数中引用该模板参数。有关定义模板参数的更多信息，请参阅 CloudFormation 模板 Parameters 语法。

CloudFormation 模板中的引用模块资源

在 CloudFormation 模板中，您经常需要根据一个资源的名称或属性来设置另一个资源的属性。有关更多信息，请参阅 引用资源。

要引用 CloudFormation 模板中某个模块内包含的资源，必须将以下两个逻辑名称组合在一起：

- 您在模板中包含模块时为模块本身指定的逻辑名称。
- 该模块内特定资源的逻辑名称。

您可以将这两个逻辑名称组合在一起，两者之间可以使用也可以不使用句点 (.)。例如，如果模块的逻辑名称为 `MyModule`，而资源的逻辑名称为 `MyBucket`，则可以将该资源称为 `MyModule.MyBucket` 或 `MyModuleMyBucket`。

要查找模块内资源的逻辑名称，您可以查阅该模块的架构（该架构可在 CloudFormation 注册表中找到），也可以使用 DescribeType 操作。架构列出了属于该模块的所有资源及其逻辑名称。

获得完整的逻辑名称后，您可以使用 `GetAtt` 和 `Ref` 之类的 CloudFormation 函数来访问模块资源上的属性值。

例如，您有一个 `My::S3::SampleBucket::MODULE` 模块，其中包含逻辑名称为 `S3Bucket` 的 `AWS::S3::Bucket` 资源。要使用 `Ref` 函数引用此存储桶的名称，请将模板中模块的名称（`MyBucket`）与模块中资源的逻辑名称（`S3Bucket`）组合使用。完整的逻辑名称为 `MyBucket.S3Bucket` 或 `MyBucketS3Bucket`。

示例 模板

以下示例模板使用 My::S3::SampleBucket::MODULE 模块创建 S3 存储桶。它还会创建一个 Amazon SQS 队列，并将其名称设置为与模块中的存储桶名称相同。此外，该模板还输出所创建的 S3 存储桶的 Amazon 资源名称（ARN）。

```
# Template that uses My::S3::SampleBucket::MODULE
Parameters:
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
  MyBucket:
    Type: My::S3::SampleBucket::MODULE
    Properties:
      BucketName: !Ref BucketName
  exampleQueue:
    Type: AWS::SQS::Queue
    Properties:
      QueueName: !Ref MyBucket.S3Bucket
Outputs:
  BucketArn:
    Value: !GetAtt MyBucket.S3Bucket.Arn
```

使用 Amazon CloudFormation 堆栈将 Amazon 资源作为单个单元进行管理

堆栈是可作为单个单元管理的一系列 Amazon 资源。换句话说，您可通过创建、更新和删除堆栈来创建、更新和删除一组资源。

创建堆栈涉及部署一个指定了资源及其配置的 CloudFormation 模板，然后 CloudFormation 会预置和配置资源。

更新堆栈涉及对模板或参数进行更改。CloudFormation 将您提交的更改与堆栈的当前状态进行比较，然后仅更新有更改的资源。CloudFormation 可能会中断资源或者替换更新的资源，具体取决于您所更新的属性。有关资源更新行为的更多信息，请参阅理解堆栈资源的更新行为。

CloudFormation 提供了两种堆栈更新方法：

- 堆栈集 – 使用更改集，您可以预览 CloudFormation 将对堆栈进行的更改，然后决定是否应用这些更改。更改集是 JSON 格式的文档，其中汇总了 CloudFormation 将对堆栈进行的更改。如果您希望确保 CloudFormation 不会进行意外的更改或者希望考虑多个选项，可使用更改集。例如，在更新期间可以使用更改集来验证 CloudFormation 不会替换堆栈中的数据库实例。
- 直接更新 – 直接更新堆栈时，您负责提交更改，而 CloudFormation 会立即进行部署。在您需要快速部署更新时，请使用直接更新。

Express Mode 让您可以在资源配置被应用后立即完成堆栈操作，无需等待资源达到完全稳定状态。在迭代开发工作流中，当速度比等待资源完全稳定更为重要时，可使用 Express Mode。有关更多信息，请参阅 Express Mode。

删除堆栈会删除与其关联的资源。例如，某个堆栈可能包含运行 Web 应用程序所需的所有资源，如 Web 服务器、数据库和联网规则。如果您不再需要该 Web 应用程序，则只需删除该堆栈及其所有相关资源。

> **Note**
>
> 将按照堆栈资源运行的时间 (即使您已立即删除堆栈) 向您收取费用。

CloudFormation 会确保所有堆栈资源都已创建或删除（视情况）。由于 CloudFormation 将堆栈资源视为单个单元，因此必须成功创建或删除所有这些资源后才能创建或删除该堆栈。如果有任何资源无法

创建，CloudFormation 将回滚该堆栈，并自动删除已创建的任何资源。如果资源无法删除，则任何剩余的资源都将保留，直到能够成功删除堆栈。

主题

- 用于管理堆栈的界面
- 代理设置指南
- 通过 CloudFormation 控制台创建堆栈
- 从 CloudFormation 控制台查看堆栈信息
- 更新堆栈模板
- 理解堆栈资源的更新行为
- 使用更改集更新 CloudFormation 堆栈
- 验证堆栈部署
- 直接更新堆栈
- 取消堆栈更新
- 通过 CloudFormation 控制台删除堆栈
- 监控堆栈进度
- 使用回滚触发器在突破警报阈值时回滚 CloudFormation 堆栈
- 使用偏移检测功能检测堆栈和资源的非托管配置更改
- 将 Amazon 资源导入 CloudFormation 堆栈
- 堆栈重构
- 资源类型支持
- 使用快速创建链接创建 CloudFormation 堆栈
- 使用 Express Mode 更快地部署 Amazon CloudFormation 堆栈
- 了解堆栈部署时间
- Amazon CLI 和 PowerShell 的 CloudFormation 堆栈操作命令示例

用于管理堆栈的界面

您可以通过以下界面来管理 CloudFormation 堆栈：

- CloudFormation 控制台 – 提供您可用来访问堆栈的 Web 界面。要使用 CloudFormation 控制台，您可以登录 Amazon Web Services 管理控制台，使用导航栏顶部的搜索框来搜索 CloudFormation，然后从搜索结果中选择 CloudFormation。

- Amazon Command Line Interface – 提供了适用于大量 Amazon 服务的命令，包括 CloudFormation，并且 Windows、Mac 和 Linux 上均支持这些命令。有关 CloudFormation 命令的信息，请参阅《Amazon CLI 命令参考》中的 cloudformation。
- Amazon Tools for PowerShell – 一组根据 适用于 .NET 的 Amazon SDK 公开的功能构建的 PowerShell 模块。借助 Tools for PowerShell，您可以从 PowerShell 命令行，为针对 Amazon 资源的操作编写脚本。您可以在 Amazon Tools for PowerShell Cmdlet 参考中找到适用于 CloudFormation 的 cmdlets。
- 查询 API – 提供了您使用 HTTPS 请求调用的低级别 API 操作。如果您在应用程序中发出 API 调用，则必须编写代码来处理低级别的详细信息，例如生成哈希值以签署请求。有关适用于 CloudFormation 的 API 操作的更多信息，请参阅《Amazon CloudFormation API 参考》中的操作。
- Amazon SDK – 提供特定语言的 API，并处理多种连接详细信息，例如计算签名、处理请求重试和错误处理。有关更多信息，请参阅用于 Amazon Web Services 的工具。
-

代理设置指南

GitHub 上的 Agent Toolkit for Amazon 提供了 `aws-core` 插件及其捆绑的 `aws-cloudformation` 技能。该技能可帮助 AI 编程代理创作和验证 CloudFormation 模板、排查部署故障以及生成 Amazon CDK 代码。该插件还配置 Amazon MCP 服务器以进行文档搜索和 Amazon API 访问。

与您的编码代理协作

如需快速入门参考，您可以将此链接直接拖放到代理的上下文中：

```
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/agent-setup-guide.md
```

选择与您的开发环境相匹配的安装方法。

先决条件

确保满足以下先决条件：

- uv 已安装在您的系统上。
- 系统上已安装 Node.js (v18+)（这是技能安装的必要条件）。
- （可选）在本地机器上配置了具有 IAM 凭证的 Amazon 账户。执行 Amazon API 调用（模板验证、部署故障排除）的工具需要凭证，但文档搜索则不需要。如果您没有配置凭证，请参阅配置 Amazon CLI，以获取详细说明。

安装内容

组件	提供的功能
`aws-cloudformation` 技能	采用安全默认设置的模板编写、三层验证管道（cfn-lint、cfn-guard、更改集），以及利用 CloudFormation 事件和 CloudTrail 关联分析对失败堆栈进行根本原因诊断。
Amazon IaC MCP 服务器	文档检索（CloudFormation 和 CDK）、模板验证、合规性检查、部署故障排除以及 CDK 最佳实操——所有这些操作均可在您的本地机器上运行。

Claude Code

安装 **`aws-core`** 插件

运行以下命令以从官方 Claude Code 市场安装该插件：

```
/plugin install aws-core@claude-plugins-official
```

这将一步安装 `aws-cloudformation` 技能、其他核心 Amazon 技能、代理挂钩和 Amazon MCP 服务器配置。

安装 Amazon IaC MCP 服务器

将 `AWS_PROFILE` 替换为您的本地 Amazon 配置文件名称。

```
claude mcp add awslabs-aws-iac-mcp \
  -e AWS_PROFILE=default \
  -e AWS_REGION=us-east-1 \
  -e FASTMCP_LOG_LEVEL=ERROR \
  --scope user \
  -- uvx awslabs.aws-iac-mcp-server@latest
```

Codex

安装 CloudFormation 技能

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

安装 Amazon IaC MCP 服务器

将 AWS_PROFILE 替换为您的本地 Amazon 配置文件名称。

```
codex mcp add awslabs-aws-iac-mcp \
  --env AWS_PROFILE=default \
  --env AWS_REGION=us-east-1 \
  --env FASTMCP_LOG_LEVEL=ERROR \
  -- uvx awslabs.aws-iac-mcp-server@latest
```

Cursor

安装 CloudFormation 技能

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

安装 Amazon IaC MCP 服务器

将以下内容添加到 "mcpServers" 内的 .cursor/mcp.json 中。将 AWS_PROFILE 替换为您的本地 Amazon 配置文件名称。

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

Kiro

在 Kiro CLI 中安装 CloudFormation 技能

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

安装 Amazon IaC MCP 服务器

将以下内容添加到 "mcpServers" 内的 ~/.kiro/settings/mcp.json 中。将 AWS_PROFILE 替换为您的本地 Amazon 配置文件名称。

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  },
  "disabled": false
}
```

安装 Kiro powers（来自 Kiro IDE）

安装以下 Kiro 能力包，它可按需为 Kiro 代理提供专用的 CloudFormation 上下文和工具：

- CloudFormation 和 CDK - 通过 Kiro 能力包链接安装。

GitHub Copilot

安装 CloudFormation 技能

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

安装 Amazon IaC MCP 服务器

将以下内容添加到 "mcpServers" 内的 .vscode/mcp.json 中。将 AWS_PROFILE 替换为您的本地 Amazon 配置文件名称。

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

Windsurf

安装 CloudFormation 技能

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

安装 Amazon IaC MCP 服务器

将以下内容添加到 "mcpServers" 内的 /.codeium/windsurf/mcp_config.json 中。将 AWS_PROFILE 替换为您的本地 Amazon 配置文件名称。

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

OpenCode

安装 CloudFormation 技能

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

安装 Amazon IaC MCP 服务器

将以下内容添加到 "mcpServers" 内的 /.config/opencode/opencode.jsonc 中。将 AWS_PROFILE 替换为您的本地 Amazon 配置文件名称。

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

适用于与代理技能和 MCP 服务器配置兼容的所有其他代理

对于任何其他支持开源代理技能格式和 MCP 服务器配置的代理，请按照以下步骤操作：

安装 CloudFormation 技能

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

安装 Amazon IaC MCP 服务器

在 "mcpServers" 下的代理 MCP 客户端配置文件中添加 Amazon IaC MCP 服务器。将 AWS_PROFILE 替换为您的本地 Amazon 配置文件名称。

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

代理可以为您做些什么

配置完成后，您的代理可以帮助您：

任务	提示示例
创建模板	“为具有公有子网和私有子网的 VPC 创建 CloudFormation 模板”
部署前验证	“验证位于 ./template.yaml 的模板并检查是否存在安全问题”
排查故障原因	“我在 us-east-1 中的‘my-app’堆栈部署失败。发生了什么？”
Search documentation (搜索文档)	“AWS::ECS::Service 支持哪些属性？”
CDK 指南	“给我介绍一下 Lambda 函数的 CDK 最佳实操”
合规性检查	“检查我的模板是否符合安全最佳实操”

IAM 权限

MCP 服务器需要以下 Amazon 权限才能实现全部功能：

对于模板验证和合规性：无需 Amazon 权限（仅限本地验证）。

对于部署故障排查：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResources",
        "cloudtrail:LookupEvents"
      ],
      "Resource": "*"
    }
```

```
  ]
}
```

相关资源

- GitHub 上的 Amazon IaC MCP 服务器
- GitHub 上的 Agent Toolkit for Amazon
- CloudFormation 用户指南
- Amazon CDK 开发人员指南
- Amazon IaC MCP 服务器简介（博客）

通过 CloudFormation 控制台创建堆栈

您可以创建堆栈模板，然后通过 CloudFormation 控制台或命令行工具使用该模板来创建堆栈。控制台提供了向导式界面，其中包含多个预定义选项，可以简化堆栈创建过程。

主题

- 创建堆栈
- 配置堆栈选项
- 预览堆栈的配置

创建堆栈

按照本节中的步骤部署模板并创建堆栈。

先决条件

- 您必须已创建堆栈模板。有关更多信息，请参阅 使用 CloudFormation 模板。

更新堆栈（控制台）

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择 Amazon Web Services 区域以在其中创建堆栈。
3. 在堆栈页面，选择右上角的堆栈，然后选择使用新资源（标准）。

您也可以选择使用现有资源（导入资源）选项，以便导入您的模板中描述的现有 Amazon 资源。有关此选项的更多信息，请参阅将 Amazon 资源导入 CloudFormation 堆栈。

4. 在创建堆栈页面上执行以下任一操作：

- 要使用现有模板，请在先决条件 – 准备模板中选择选择现有模板。然后在指定模板下，根据模板的位置选择 Amazon S3 URL 或上传模板文件。
  - 如果选择 Amazon S3 URL，请提供指向 S3 存储桶中模板文件的 URL。

    如果模板包含嵌套堆栈（例如，子目录下其他模板文档中描述的堆栈），请确保您的 S3 存储桶包含所需的文件和目录。

    如果模板来自启用了版本控制的存储桶，则可以通过在 URL 后附加 ?versionId=*version-id* 来指定模板的具体版本。有关启用版本控制的存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用启用版本控制的存储桶中的对象。

    该 URL 必须指向一个最大大小为 1MB 的模板，该模板存储在您具有读取权限的 S3 存储桶中。URL 的最大长度为 1024 个字符。某些资源可能需要存储桶与堆栈位于同一区域。
  - 如果选择上传模板文件，请选择选择文件来选择本地计算机中的模板文件。模板文件大小必须为 1 MB 或更小。

    如果您选择了您的模板，CloudFormation 会上传此文件并显示 S3 URL。CloudFormation 会将其上传到您的 Amazon Web Services 账户中的 Amazon S3 存储桶。如果您的 Amazon Web Services 账户中已有使用 CloudFormation 创建的 S3 存储桶，则 CloudFormation 会将该模板添加到该存储桶。如果还没有由 CloudFormation 创建的现有存储桶，则会为您上传模板文件的每个区域创建一个唯一的存储桶。

    使用由 CloudFormation 创建的 S3 存储桶时应注意以下事项：

    - 具有您的 Amazon Web Services 账户中的 Amazon S3 权限的任何人均可访问这些存储桶。
    - CloudFormation 创建默认情况下已启用服务器端加密的存储桶，从而对存储桶中存储的所有对象进行加密。

      例如，对于 CloudFormation 创建的存储桶，您可以使用 Amazon S3 控制台（https://console.aws.amazon.com/s3/）或 Amazon CLI 直接管理加密选项。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的为 Amazon S3 存储桶设置默认服务器端加密行为。

- 您可通过手动将模板上传到 Amazon S3 来使用您自己的存储桶并管理其权限。每当您创建或更新堆栈时，请指定模板文件的 Amazon S3 URL。
- 如果尚未准备好模板，请选择使用模板编辑室构建。有关更多信息，请参阅 模板工作室。

5. 选择下一步继续操作并验证模板。

   CloudFormation 会首先验证模板以捕获语法错误和某些语义错误（例如循环依赖项），然后才会继续操作。在验证期间，CloudFormation 首先检查模板是否是有效的 JSON。如果不是，CloudFormation 会检查模板是否是有效的 YAML。如果两种检查都失败，CloudFormation 会返回模板验证错误。

6. 在 Specify stack details (指定堆栈详细信息) 页面上，在堆栈名称框中键入堆栈名称。

   堆栈名称是帮助您从堆栈列表中查找特定堆栈的标识符。堆栈名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 128 个字符。

7. 在参数部分中，指定模板中所定义参数的值：
8. 选择下一步以继续创建堆栈。
9. （可选）在配置堆栈选项页面上，更改默认堆栈选项。有关更多信息，请参阅 配置堆栈选项。
10. 如果模板包含 IAM 资源，则对于功能，请选择我确认该模板可能会创建 IAM 资源以指定您要在模板中使用 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。
11. 选择下一步以继续。
12. 在检查并创建页面上，检查堆栈的详细信息。

    要在启动堆栈之前更改任何值，请在包含要更改的设置的部分中选择编辑。

13. （可选）在创建堆栈前，您可以创建更改集来预览堆栈的配置。在检查并创建页面上，选择创建更改集，然后按照说明进行操作。有关更多信息，请参阅 预览堆栈的配置。
14. 选择提交以启动堆栈。

> **Note**
>
> 在配置资源之前，CloudFormation 会自动验证您的模板。如果验证检测到错误（例如，属性语法无效或资源名称冲突），在任何资源被配置之前，堆栈创建过程就会停止。查看验证结果，请打开堆栈的事件选项卡并选择操作 ID。在“操作”视图页面中，选择“部署验证”选项卡。

然后，CloudFormation 将会创建模板中定义的所有资源。

您可以在新堆栈的事件选项卡上监控堆栈创建的进度和状态。有关更多信息，请参阅 监控堆栈进度。

使用命令行创建堆栈

您可以使用以下任一命令：

- create-stack (Amazon CLI)
- New-CFNStack (Amazon Tools for Windows PowerShell)

有关使用命令行创建堆栈的示例，请参阅 Amazon CLI 和 PowerShell 的 CloudFormation 堆栈操作命令示例。

配置堆栈选项

在配置堆栈选项页面上，您可以配置 CloudFormation 堆栈的选项，例如标签、堆栈事件通知或堆栈策略等。

您可以设置以下堆栈选项：

标签

您最多可以为堆栈以及 CloudFormation 支持标记的任何资源添加 50 个标签键对。标签是由客户定义的键和值，可以分配给 Amazon 资源以用于成本跟踪等目的。

Key 包含任意字母数字字符或空格。标签键最长可达 127 个字符。

Value (值) 可由任何字母数字字符或空格组成。标签值最长可达 255 个字符。

创建堆栈后，添加、更新或移除堆栈级别标签将会启动堆栈更新。将相应更新支持堆栈级别标签传播的所有资源。

权限：

CloudFormation 可以代入的现有 IAM 服务角色。无需使用您的账户凭证，CloudFormation 会使用角色的凭证来创建您的堆栈。有关更多信息，请参阅 Amazon CloudFormation 服务角色。

> **Note**
>
> 如果您的账户中启用了角色管理器，CloudFormation 会为您附加一个服务角色，而此处所述的服务角色选择将替换为自定义选项。要使用其它角色，请选择自定义。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色创建。

堆栈故障选项

指定所有堆栈部署和更改集操作的预置故障选项。有关更多信息，请参阅 选择预置资源时如何处理失败。

当堆栈状态为 CREATE_FAILED 或 UPDATE_FAILED 时，Roll back all stack resources（回滚所有堆栈资源）选项将回滚模板中指定的所有资源。

对于创建操作，Preserve successfully provisioned resources（保留成功预置的资源）选项将保留成功资源的状态，而失败的资源将保持失败状态，直到执行下一次更新操作。

对于更新和更改集操作，Preserve successfully provisioned resources（保留成功预置的资源）选项将保留成功资源的状态，同时将失败的资源回滚到上一个已知的稳定状态。失败的资源将处于 UPDATE_FAILED 状态。没有上一个已知稳定状态的资源将在下一次堆栈操作时删除。

您也可以设置以下高级选项来创建堆栈：

Express Mode

在“部署选项”下，您可以将部署模式选为“Express”。Express Mode 会在资源配置被应用后立即完成堆栈操作，无需等待资源达到完全稳定状态。有关更多信息，请参阅 Express Mode。

Stack policy (堆栈策略

定义在堆栈更新期间要防止意外更新的资源。默认情况下，堆栈更新期间所有资源都可更新。

您可以直接以 JSON 形式输入堆栈策略，也可以上传包含堆栈策略的 JSON 文件。有关更多信息，请参阅 防止更新堆栈资源。

回滚配置

您可在堆栈创建和更新期间让 CloudFormation 监控堆栈的状态，并可在堆栈超出您指定的任何警报阈值时回滚操作。指定 CloudFormation 应监控的 CloudWatch 警报。在堆栈操作或监控期间，如果有任何警报进入 ALARM 状态，CloudFormation 就会回滚整个堆栈操作。有关更多信息，请参阅 使用回滚触发器在突破警报阈值时回滚 CloudFormation 堆栈。

通知选项

您可以指定将发送有关堆栈事件的通知的新或现有 Amazon Simple Notification Service 主题。

如果您创建 Amazon SNS 主题，则必须指定名称和电子邮件地址（将向该位置发送堆栈事件通知）。

堆栈创建选项

包含以下用于创建堆栈的选项，但这些选项不能作为堆栈更新的一部分。

超时

指定 CloudFormation 应分配的在堆栈创建操作超时前的时间量（以分钟为单位）。如果 CloudFormation 无法在分配的时间内创建整个堆栈，它将因超时而导致堆栈创建失败并回滚该堆栈。

默认情况下，堆栈创建从不超时。但是，单个资源可能会根据它们所实施服务的性质而具有自己的超时。例如，如果堆栈中的单个资源发生超时，则堆栈创建也会超时，即使尚未达到您为堆栈创建指定的超时也不例外。

终止保护

防止意外删除堆栈。如果用户尝试删除已启用终止保护的堆栈，则删除操作会失败，并且堆栈及其状态将保持不变。有关更多信息，请参阅 CloudFormation 堆栈的删除保护。

默认情况下，终止保护处于已禁用状态。

预览堆栈的配置

要在创建堆栈前预览 CloudFormation 堆栈的配置，请创建更改集。利用此功能，您在执行更改集前可检查各种配置并对堆栈进行更正和更改。有关更改集的更多信息，请参阅 使用更改集更新 CloudFormation 堆栈。

为新堆栈创建更改集

要为新堆栈创建更改集，请选择您的堆栈模板，并像创建新堆栈一样指定堆栈配置，然后选择创建新的更改集，而不是新堆栈。

为新堆栈创建更改集

1. 在检查和创建页面上，选择创建更改集。
2. 在创建更改集对话框中，输入更改集的名称，并根据需要输入描述。选择创建更改集。

   当您为新堆栈创建更改集时，CloudFormation 执行以下操作：

   - 启动状态为 `REVIEW_IN_PROGRESS` 的新堆栈。
   - 为反映您在之前步骤中指定的堆栈配置的新堆栈创建更改集。

CloudFormation 会显示建议的堆栈的更改集页面。CloudFormation 创建更改集时，该更改集的状态为 CREATE_IN_PROGRESS，执行状态为 UNAVAILABLE。CloudFormation 成功创建完更改集后，会将更改集状态设置为 CREATE_COMPLETE，其执行状态为 AVAILABLE。堆栈状态已更新为 REVIEW_IN_PROGRESS。此时，您可以执行更改集以完成新堆栈的创建。

在更改窗格中，CloudFormation 会显示您的堆栈的建议配置。

如果 CloudFormation 无法创建更改集，它会将更改集状态设置为 CREATE_FAILED。修复状态原因字段中显示的错误，然后创建新的更改集。在此阶段，您在执行下一个更改集之前可对堆栈尝试各种配置和进行更正和更改。

3. 要基于更改集完成创建新堆栈，请选择 Execute（执行），指定回滚配置，然后选择 Execute change set（执行更改集）。

从 CloudFormation 控制台查看堆栈信息

创建 CloudFormation 堆栈后，您可以使用 Amazon Web Services 管理控制台查看其数据和资源。

查看有关CloudFormation堆栈的信息

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择堆栈所在的 Amazon Web Services 区域。
3. 在堆栈页面上，选择堆栈名称。CloudFormation 将显示选定堆栈的堆栈详细信息。

> **Note**
>
> 如果堆栈已删除，则可以使用筛选状态选项找到该堆栈。有关更多信息，请参阅 通过 CloudFormation 控制台查看已删除的堆栈。

您可查看以下堆栈信息：

堆栈信息

显示有关堆栈及其配置的一般信息，包括：

概述

显示堆栈名称、堆栈 ID、根堆栈和 IAM 角色，以及状态信息，例如堆栈状态、偏差状态和终止保护等。

标签

显示与堆栈关联的任何标签。

Stack policy (堆栈策略)

描述可防止堆栈更新的堆栈资源。要能够更新这些资源，必须在堆栈更新过程中显式允许这些资源。有关更多信息，请参阅 防止更新堆栈资源。

回滚配置

显示您指定 CloudFormation 应在堆栈操作期间或指定的监控期间监控的任何 CloudWatch 警报。在堆栈操作或监控期间，如果有任何警报进入 ALARM 状态，CloudFormation 就会回滚整个堆栈操作。有关更多信息，请参阅 使用回滚触发器在突破警报阈值时回滚 CloudFormation 堆栈。

通知选项

显示发送有关堆栈事件的通知的 Amazon Simple Notification Service 主题（如果已指定）。

Events

显示在创建、更新或删除堆栈时跟踪的操作。有关更多信息，请参阅 监控堆栈进度。

为给定堆栈操作所触发的所有事件分配的客户端请求令牌都是相同的，可使用此令牌跟踪操作。从控制台发起的堆栈操作使用令牌格式 Console-StackOperation-ID，这有助于您轻松识别堆栈操作。例如，如果使用控制台创建堆栈，则将向每个生成的堆栈事件分配以下格式的相同令牌：`Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002`。

资源

显示作为堆栈一部分的资源。

输出

显示在堆栈模板中声明的输出。有关更多信息，请参阅 获取从已部署的 CloudFormation 堆栈导出的输出。

参数

显示堆栈的参数及其值。

对于包含 Systems Manager 参数的堆栈，已解析的值列会显示在 Systems Manager 参数在堆栈定义中使用的值。有关更多信息，请参阅 使用 CloudFormation 提供的参数类型在运行时指定现有资源。

模板

显示堆栈的模板。

对于包含宏的堆栈，可以选择 View original template（查看原始模板）以查看用户提交的模板，或选择 View processed template（查看处理后的模板）以查看 CloudFormation 处理引用的宏后的模板。CloudFormation 使用处理的模板创建或更新堆栈。

更改集

显示堆栈的更改集。

有关更多信息，请参阅 查看 CloudFormation 堆栈的更改集。

Git 同步

显示堆栈的 Git 同步控制面板。

有关更多信息，请参阅 Git 同步状态控制面板。

更新堆栈模板

要修改 CloudFormation 堆栈中的资源或属性，您必须更新堆栈的模板。以该堆栈的现有模板为基础，然后对其进行更改。如果模板存储在源控制系统中，请以该模板的副本为起点。否则，您可以从 CloudFormation 获取该模板的副本。

如果只需要修改堆栈的参数或设置（如堆栈的 Amazon SNS 主题），则可以重复使用现有的模板，而不获取副本。

您可以使用文本编辑器或 模板工作室来修改 CloudFormation 堆栈模板。

使用模板工作室更新现有堆栈模板

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面上，选择要更新的堆栈的名称。
3. 开始更新，然后选择在模板工作室中打开。

4. 编辑模板，然后选择保存并继续。模板工作室会在保存之前尝试验证草稿。如果验证失败，您可以确认仍要保存。保存后，您将返回至堆栈更新。

使用 Amazon CLI 更新现有的堆栈模板

1. 要获取要更新的堆栈的模板，请使用 get-template CLI 命令。
2. 将模板复制并粘贴到文本文件中，进行修改，然后保存它。只 复制模板。该命令将模板括在引号中，但是不得复制模板两端的引号。模板本身以左括号开始，以右括号结束。指定对本文件中堆栈资源的更改。

更改模板时应记住以下几点：

- 您无法添加、修改或删除不支持更新的资源使用的参数。
- 对于大多数资源，更改资源的逻辑名称相当于删除该资源并将其替换为新资源。与重命名的资源关联的任何其他资源也需要更新，并且可能会导致它们被替换。其他资源需要您更新属性（不仅仅是逻辑名称）才能启动更新。
- 某些资源可能对您可以为某些属性设置的值有限制。例如，对 RDS 数据库实例的 `AllocatedStorage` 属性的更改必须大于当前值。如果您的更新违反了这些规则，该部分将会失败。
- 更新一个资源也会影响引用该资源的其他资源。如果您使用 `Ref` 函数 或 `Fn::GetAtt` 函数 之类的函数来设置基于其他资源的属性，则当被引用的资源发生更改时，CloudFormation 也会更新引用资源。
- 要了解更新特定资源属性的影响，请参阅 Amazon 资源和属性类型参考。对于每种属性，更新将产生以下一种影响：
  - 更新要求：没有中断
  - 更新要求：时而中断
  - 更新要求：替换
- 您可以使用 validate-template CLI 命令或通过在控制台上指定模板来验证模板的 JSON 或 YAML 语法。控制台会自动执行验证。但这些方法仅会验证模板的语法，而不会验证您为资源指定的属性值对于该资源是否有效。有关更复杂的验证和最佳实践检查，请参阅验证模板。

> **Note**
>
> 有时，CloudFormation 不允许您尝试进行某些更改，并且会告诉您不允许进行更改。然而，该消息可能是异步出现的，因为在默认情况下，CloudFormation 创建和更新资源的顺序是不确定的。

理解堆栈资源的更新行为

提交更新时，Amazon CloudFormation 会基于提交的内容与堆栈当前模板之间的差异来更新资源。尚未更改的资源在更新过程中会不中断地运行。对于更新后的资源，Amazon CloudFormation 使用下列更新行为之一：

无中断更新

Amazon CloudFormation 更新资源时不会中断该资源的运行，也不会更改该资源的物理 ID。例如，如果您更新 AWS::CloudTrail::Trail 资源的某些属性，则 Amazon CloudFormation 会更新跟踪而不会出现中断。

有部分中断的更新

Amazon CloudFormation 更新资源，并且出现一些中断。例如，如果您对 AWS::EC2::Instance 资源更新特定属性，则在 Amazon CloudFormation 和 Amazon EC2 重新配置实例期间，该实例可能有时会中断。

替换

Amazon CloudFormation 会在更新过程中重新创建资源，这还会生成新的物理 ID。Amazon CloudFormation 通常会先创建替换资源，将对其他相关资源的引用更改为指向替换资源，然后删除旧资源。例如，如果您更新 AWS::EC2::Instance 资源类型的 `AvailabilityZone` 属性，则 Amazon CloudFormation 会创建新资源并将当前 EC2 实例资源替换为新资源。

如果您添加或移除需要替换的属性，则它也会触发更新。即使属性的实际值没有改变，也会进行更新。

Amazon CloudFormation 使用的方法取决于您为给定资源类型更新的属性。Amazon 资源和属性类型参考中介绍了每个属性的更新行为。

根据更新行为，您可以决定何时修改资源以减小这些更改对应用程序产生的影响。具体来说，您可以计划好更新过程中必须替换 资源的时间。例如，如果您更新 AWS::RDS::DBInstance 资源类型的 `Port`

属性，则 Amazon CloudFormation 将通过使用更新后的端口设置创建新数据库实例来替换数据库实例并删除旧的数据库实例。在更新之前，您可以计划执行以下操作来准备好进行数据库替换：

- 拍摄当前数据库的快照。
- 准备一个策略，指定使用该数据库实例的应用程序在数据库实例替换期间将如何处理中断。
- 确保使用该数据库实例的应用程序考虑更新的端口设置以及您进行的任何其他更新。
- 使用数据库快照在新数据库实例上还原数据库。

该示例并不详尽，但它旨在让您了解针对在更新过程中替换资源的情况，需要计划的事项。

> **Note**
>
> 如果模板包括一个或多个嵌套堆栈，则 Amazon CloudFormation 也会为每个嵌套堆栈启动更新。这对于确定嵌套堆栈是否已修改是必要的。Amazon CloudFormation 只更新嵌套堆栈中那些在相应模板中指定了更改的资源。

使用更改集更新 CloudFormation 堆栈

当您需要更新堆栈时，在实施之前了解您的更改会如何影响正在运行的资源可以帮助您充满自信地更新堆栈。借助更改集功能，您可以预览对堆栈提议的更改可能会对正在运行的资源产生的影响，包括对资源属性和特性的影响。无论更改是否会删除或替换任何关键资源，只有在您决定执行更改集时，CloudFormation 才会对堆栈进行更改，这让您可以决定是要继续执行提议的更改，还是通过创建其他更改集来探索其他更改。您可以使用 CloudFormation 控制台、Amazon CLI 或 CloudFormation API 来创建和管理更改集。

主题

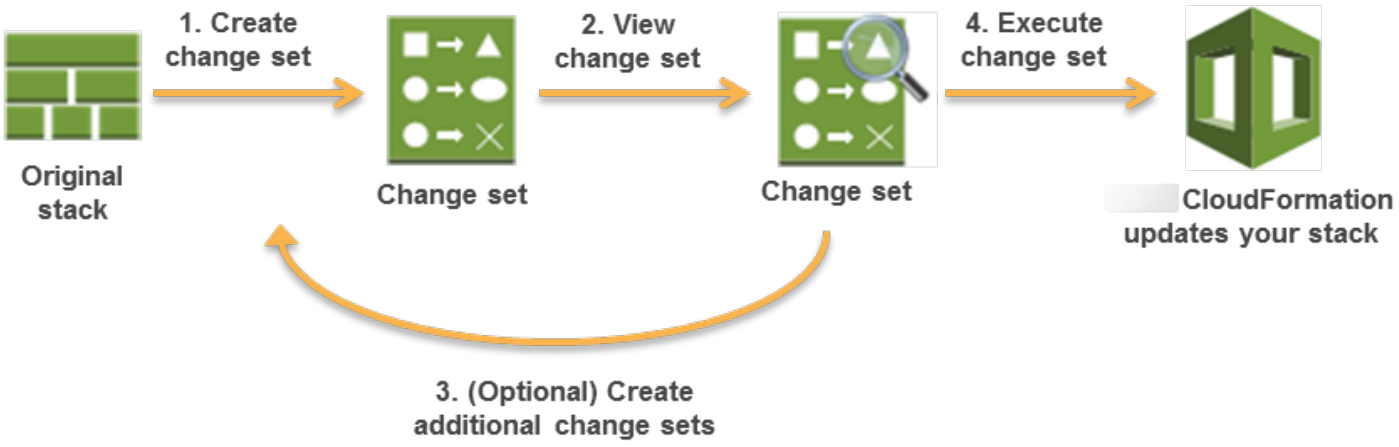
- 创建 CloudFormation 堆栈的更改集
- 查看 CloudFormation 堆栈的更改集
- 使用偏差感知更改集
- 执行 CloudFormation 堆栈的更改集
- 删除 CloudFormation 堆栈的更改集
- CloudFormation 堆栈的更改集示例
- 嵌套堆栈的更改集

> **Important**
>
> 更改集并不能保证 CloudFormation 能够成功更新堆栈。不过，部署前验证会在创建更改集时检查常见的失败原因，包括属性语法错误、资源名称冲突、服务配额限制以及其他约束条件。由于运行时条件（例如，自定义资源逻辑或特定于服务的约束）导致的失败，在执行过程中仍可能发生。

> **Note**
>
> 您可以使用带有更改集的 Express Mode 来更快地完成堆栈操作。在创建更改集时指定 `--deployment-config '{"mode": "EXPRESS"}'` 参数。执行更改集时，CloudFormation 会应用 Express Mode，在应用配置后立即完成资源操作。有关更多信息，请参阅 Express Mode。

更改集概述

下表总结了您可以如何使用更改集来更新堆栈：



1. 通过为您要更新的堆栈提交更改来创建更改集。您可以提交修改后的堆栈模板或者修改后的输入参数值。CloudFormation 将您的堆栈与所提交更改进行对比，生成更改集；此时不会更改您的堆栈。
2. 查看更改集以了解将更改哪些堆栈设置和资源。例如，您可以查看 CloudFormation 将添加、修改或删除哪些资源。此外，您还可以查看 CloudFormation 将要修改的资源属性和特性（例如标签）的更改前后比较。

3. 可选：如果您希望在决定所要进行的更改之前考虑其他更改，请创建其他的更改集。创建多个更改集有助您了解和评估不同的更改将对资源和属性造成的影响。您可以根据需要创建任意数量的更改集。
4. 执行包含您希望应用到堆栈的更改的更改集。CloudFormation 使用这些更改更新您的堆栈。

> **Note**
>
> 在您执行更改之后，CloudFormation 将删除与堆栈关联的所有更改集，因为它们不再适用于更新后的堆栈。

您还可以删除更改集以防止执行不应该应用的更改集。

创建 CloudFormation 堆栈的更改集

要为正在运行的堆栈创建更改集，请通过提供修改后的模板、新的输入参数值或两者来提交您想要进行的更改。CloudFormation 通过将堆栈与您提交的更改进行比较来生成更改集。

您可以在创建更改集之前或在创建更改集期间修改模板。

Create a change set (console)

创建更改集

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面上，选择您要为其创建更改集的正在运行的堆栈。
4. 在堆栈详细信息窗格中，选择堆栈操作，然后选择创建更改集。
5. 在为 ***stack-name*** 创建更改集页面上，执行下列操作之一以修改输入参数值，指定更新模板的位置或修改模板：

任务	处理建议
修改输入参数值	选择使用现有模板，然后选择下一步，以继续输入或修改输入参数值。

任务	处理建议
指定更新模板的位置	如果您已修改模板，请选择替换现有模板，然后执行以下任一操作： • 对于存储在 Amazon S3 存储桶中的模板，选择Amazon S3 URL。输入或粘贴模板的 URL，然后选择 Next (下一步)。 如果模板位于启用了版本控制的存储桶中，则可以通过在 URL 后附加 `?versionId=` *`version-id`* 来指定模板的具体版本。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用启用版本控制的存储桶中的对象。 • 对于在计算机本地存储的模板，选择 Upload a template file (上传模板文件)。选择 Choose File (选择文件) 以导航到此文件并选中它，然后选择 Next (下一步)。
修改模板	如果尚未修改模板，请选择在模板工作室中打开。编辑模板，然后选择保存并继续以返回到为 ***`stack-name`*** 创建更改集页面。选择下一步。

6. 在指定堆栈详细信息页面上，指定更改集的名称，您还可以选择在概述部分指定更改集的描述以标识其用途。如果您的模板包含参数，则在 Specify stack details (指定堆栈详细信息) 页面上，输入或修改适用的输入参数值，然后选择 Next (下一步)。

   如果您重用堆栈的模板，CloudFormation 会使用堆栈中的当前值填充各个参数，但在 `NoEcho` 属性中声明的参数除外。要为这些参数使用现有值，请选择使用现有值。

   有关使用 `NoEcho` 来遮蔽敏感信息以及使用动态参数管理密钥的更多信息，请参阅 请勿将凭证嵌入您的模板 最佳实践。

7. 在配置堆栈选项页面上，更新堆栈的标签、IAM 服务角色、堆栈策略、回滚配置、Amazon SNS 通知主题（如果适用）或更改集。

> **Note**
>
> 默认情况下，嵌套堆栈的更改集处于已启用状态，这将为模板中指定的所有嵌套堆栈创建更改集。要仅为当前堆栈创建更改集，请选择已禁用。有关嵌套堆栈更改集的详细信息，请参阅 嵌套堆栈的更改集。

8. 如果模板包含 IAM 资源，则对于功能，请选择我确认 CloudFormation 可能会创建 IAM 资源。IAM 资源可以修改您 Amazon 账户中的权限；查看这些资源以确保只允许您打算执行的操作。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。
9. 选择下一步以继续。
10. 在审核 ***stack-name*** 页面上，检查此更改集的更改。
11. 选择提交。

    您将重定向到更改集的详细信息页面的更改选项卡。在 CloudFormation 生成更改集时，更改集的状态为 CREATE_PENDING。在创建了更改集之后，CloudFormation 将状态设置为 CREATE_COMPLETE。在 Changes (更改) 部分中，CloudFormation 列出将对您的堆栈进行的所有更改。有关更多信息，请参阅 查看 CloudFormation 堆栈的更改集。

    选择属性级别更改列中的查看详细信息，以查看在属性级别所做的更改。

    如果 CloudFormation 无法创建更改集（报告 FAILED 状态），请修复 Status (状态) 字段中显示的错误，然后重新创建更改集。
12. 确认更改无误后，选择执行更改集

Create a change set for nested stacks (console)

创建嵌套堆栈的更改集

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面上，选择您要为其创建更改集的正在运行的堆栈。
4. 在堆栈详细信息窗格中，选择堆栈操作，然后选择创建更改集。
5. 在为 ***stack-name*** 创建更改集页面上，执行下列操作之一以修改输入参数值，指定更新模板的位置或修改模板：

任务	处理建议
修改输入参数值	选择使用现有模板，然后选择下一步，以继续输入或修改输入参数值。

任务	处理建议
指定更新模板的位置	如果您已修改模板，请选择替换现有模板，然后执行以下任一操作： • 对于存储在 Amazon S3 存储桶中的模板，选择Amazon S3 URL。输入或粘贴模板的 URL，然后选择 Next (下一步)。 如果模板位于启用了版本控制的存储桶中，则可以通过在 URL 后附加 `?versionId=` *`version-id`* 来指定模板的具体版本。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用启用版本控制的存储桶中的对象。 • 对于在计算机本地存储的模板，选择 Upload a template file (上传模板文件)。选择 Choose File (选择文件) 以导航到此文件并选中它，然后选择 Next (下一步)。
修改模板	如果尚未修改模板，请选择在模板工作室中打开。编辑模板，然后选择保存并继续以返回到为 ***`stack-name`*** 创建更改集页面。选择下一步。

6. 在指定堆栈详细信息页面上，指定更改集的名称，您还可以选择在概述部分指定更改集的描述以标识其用途。如果您的模板包含参数，则在 Specify stack details (指定堆栈详细信息) 页面上，输入或修改适用的输入参数值，然后选择 Next (下一步)。

   如果您重用堆栈的模板，CloudFormation 会使用堆栈中的当前值填充各个参数，但在 `NoEcho` 属性中声明的参数除外。要为这些参数使用现有值，请选择使用现有值。

   有关使用 `NoEcho` 来遮蔽敏感信息以及使用动态参数管理密钥的更多信息，请参阅 请勿将凭证嵌入您的模板 最佳实践。

7. 在配置堆栈选项页面上，更新堆栈的标签、IAM 服务角色、堆栈策略、回滚配置、Amazon SNS 通知主题（如果适用）或更改集。有关更多信息，请参阅 配置堆栈选项。

> **Note**
>
> 默认情况下，嵌套堆栈的更改集处于已启用状态，这将为模板中指定的所有嵌套堆栈创建更改集。有关嵌套堆栈更改集的详细信息，请参阅 嵌套堆栈的更改集。

8. 如果模板包含 IAM 资源，则对于功能，请选择我确认 CloudFormation 可能会创建 IAM 资源。IAM 资源可以修改您 Amazon 账户中的权限；查看这些资源以确保只允许您打算执行的操作。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。
9. 选择下一步以继续。
10. 在审核 ***stack-name*** 页面上，检查此更改集的更改。
11. 选择提交。

> **Note**
>
> 为嵌套堆栈创建更改集时，CloudFormation 属性级的更改集无法解析跨堆栈引用。如果子堆栈中的资源引用了父堆栈的输出，而父堆栈已被修改，则更改集可以标记这些资源以便进行有条件替换

您将重定向到更改集的详细信息页面的更改选项卡。在 CloudFormation 生成更改集时，更改集的状态为 CREATE_PENDING。在创建了更改集之后，CloudFormation 将状态设置为 CREATE_COMPLETE。在 Changes (更改) 部分中，CloudFormation 列出将对您的堆栈进行的所有更改。有关更多信息，请参阅 查看 CloudFormation 堆栈的更改集。

如果 CloudFormation 无法创建更改集（报告 FAILED 状态），请修复 Status (状态) 字段中显示的错误，然后重新创建更改集。

12. 确认更改无误后，选择执行更改集

创建更改集 (Amazon CLI)

- 使用 create-change-set 命令。

  您以命令选项的形式提交更改。您可以指定新参数值和/或修改后的模板。例如，以下命令为 SampleChangeSet 堆栈创建名为 MyStack 的更改集。更改集使用当前堆栈的模板，但 Purpose 参数具有不同值：

```
aws cloudformation create-change-set --stack-name MyStack \
    --change-set-name SampleChangeSet --use-previous-template \
    --parameters \
      ParameterKey="InstanceType",UsePreviousValue=true
 ParameterKey="KeyPairName",UsePreviousValue=true
 ParameterKey="Purpose",ParameterValue="production"
```

> **Note**
>
> 除了对属性语法进行验证和检测资源名称冲突外，更改集的创建过程还会检查服务配额限制、记录器冲突以及 ECR 存储库的删除就绪状态。这些检查以 WARN 模式运行——CloudFormation 会成功创建更改集，但系统会发出警告以提示潜在问题。我们建议您在执行更改集之前先解决这些问题。

Express Mode

要创建使用 Express Mode 的更改集，请添加 `--deployment-config` 参数：

```
aws cloudformation create-change-set --stack-name MyStack \
    --change-set-name MyExpressChangeSet \
    --use-previous-template \
    --deployment-config '{"mode": "EXPRESS"}'
```

当您执行此更改集时，CloudFormation 会使用“Express Mode”，在应用配置后立即完成资源操作。有关更多信息，请参阅 Express Mode。

查看 CloudFormation 堆栈的更改集

创建更改集之后，您可在执行之前查看提议的更改。您可以使用 CloudFormation 控制台、Amazon CLI 或 CloudFormation API 查看更改集。CloudFormation 控制台提供了更改摘要以及 JSON 格式的更改详细列表。Amazon CLI 和 Amazon CloudFormation API 以 JSON 格式返回详细的更改列表。

View a change set (console)

查看更改集

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面上，选择包含您要查看的更改集的堆栈名称。
4. 在导航窗格中，选择 Change Sets (更改集) 以查看堆栈的更改集的列表。
5. 选择要查看的更改集的名称。

   CloudFormation 控制台将您引导到更改集的详细信息页面，在其中您可以查看创建更改集的时间、其状态、用于生成更改集的输入，以及更改摘要。

在更改部分中，每一行表示 CloudFormation 将添加、修改或删除的资源。

- 添加 – CloudFormation 在您将资源添加到堆栈的模板上时创建资源。
- 修改 – CloudFormation 在您更改堆栈模板中资源的属性时修改资源。
- 删除 – CloudFormation 在您删除堆栈模板中的资源时删除资源。

> **Note**
>
> 修改可能会导致中断或替换（重新创建）资源。有关资源更新行为的更多信息，请参阅理解堆栈资源的更新行为。

要重点查看特定更改，请使用筛选视图。例如，筛选特定资源类型，如 `AWS::EC2::Instance`。要筛选特定资源，请指定其逻辑或物理 ID，例如 `myWebServer` 或 `i-123abcd4`。

6. 在更改部分中，选择属性级别更改列中的查看详细信息，以查看资源的属性值更改。
7. CloudFormation 控制台将引导您打开资源的属性级别更改页面，您可以在其中查看执行更改集之前的资源模板配置，以及执行更改集之后将会具有的模板配置。

   属性级别更改部分的表格显示了受影响属性的路径、更改类型、更改前值和更改后值。在该表格中，选中要在模板的更改前和更改后视图中突出显示的每项更改的复选框，以查看将在属性级别进行的具体更改。

   - 添加 – 添加的属性以绿色突出显示。
   - 修改 – 修改后的属性以蓝色突出显示。
   - 移除 – 已删除的属性以红色突出显示。

View a change set for nested stack (console)

查看嵌套堆栈的更改集（控制台）

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面上，选择包含您要查看的更改集的堆栈名称。

4. 在导航窗格中，选择更改集以查看堆栈的更改集列表。
5. 选择要查看的更改集的名称。

   CloudFormation 控制台将您引导到更改集的详细信息页面，在其中您可以查看创建更改集的时间、其状态、用于生成更改集的输入，以及更改摘要。

   在更改部分中，每一行表示 CloudFormation 将添加、修改、删除或显示动态状态的资源。

   - 添加 – CloudFormation 在您将资源添加到堆栈的模板上时创建资源。
   - 修改 – CloudFormation 在您更改堆栈模板中资源的属性时修改资源。
   - 删除 – CloudFormation 在您删除堆栈模板中的资源时删除资源。
   - 动态 – CloudFormation 无法从嵌套堆栈的模板中确定确切的资源更改操作。

   > **Note**
   >
   > 修改可能会导致中断或替换（重新创建）资源。有关资源更新行为的更多信息，请参阅理解堆栈资源的更新行为。

   要重点查看特定更改，请使用筛选视图。例如，筛选特定资源类型，如 **`AWS::CloudFormation::Stack`**。要筛选特定资源，请指定其逻辑或物理 ID，例如 **`DeadLetterQueue`** 或 **`NestedStack`**。
6. 在更改部分中，选择要查看的嵌套更改集的查看嵌套更改集。

   CloudFormation 控制台会将您引导到嵌套更改集的详细信息页面。您可以选择转到根更改集以查看根更改集，也可以选择查看父更改集以查看父更改集。有关更多信息，请参阅嵌套堆栈的更改集。

   > **Note**
   >
   > 为嵌套堆栈创建更改集时，CloudFormation 属性级的更改集无法解析跨堆栈引用。如果子堆栈中的资源引用了父堆栈的输出，而父堆栈已被修改，则更改集可以标记这些资源以便进行有条件替换

查看更改集 (Amazon CLI)

1. 要获取更改集的 ID，请运行 change-sets 命令。

   指定具有您要查看的更改集的堆栈名称，如下例中所示：

```
aws cloudformation list-change-sets --stack-name MyStack
```

   CloudFormation 返回更改集列表，类似于以下内容：

```
{
    "Summaries": [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
            "Status": "CREATE_COMPLETE",
            "ChangeSetName": "SampleChangeSet",
            "CreationTime": "2020-11-18T20:44:05.889Z",
            "StackName": "MyStack",
            "ChangeSetId": "arn:aws:cloudformation:us-
east-1:123456789012:changeSet/SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000"
        },
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
            "Status": "CREATE_COMPLETE",
            "ChangeSetName": "SampleChangeSet-conditional",
            "CreationTime": "2020-11-18T21:15:56.398Z",
            "StackName": "MyStack",
            "ChangeSetId": "arn:aws:cloudformation:us-
east-1:123456789012:changeSet/SampleChangeSet-conditional/1a2345b6-0000-00a0-
a123-00abc0abc000"
        },
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
            "Status": "CREATE_COMPLETE",
            "ChangeSetName": "SampleChangeSet-replacement",
            "CreationTime": "2020-11-18T21:03:37.706Z",
            "StackName": "MyStack",
            "ChangeSetId": "arn:aws:cloudformation:us-
east-1:123456789012:changeSet/SampleChangeSet-replacement/1a2345b6-0000-00a0-
a123-00abc0abc000"
```

```
        }
    ]
}
```

2. 运行 describe-change-set 命令，指定您要查看的更改集的 ID。例如：

```
aws cloudformation describe-change-set \
  --change-set-name arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

CloudFormation 返回有关指定更改集的信息。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet-direct",
    "Parameters": [
        {
            "ParameterValue": "testing",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "ellioty-useast1",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
```

```
                        }
                    }
                ],
                "Action": "Modify",
                "Scope": [
                    "Tags"
                ],
                "LogicalResourceId": "MyEC2Instance",
                "Replacement": "False"
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T23:35:25.813Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-direct/9edde307-960d-4e6e-ad66-b09ea2f20255"
}
```

将 `--include-property-values` 与 describe-change-set 结合使用可列出属性级别的更改。

`Changes` 键列出对资源的更改。如果您要执行此更改集，CloudFormation 将更新 `i-1abc23d4` EC2 实例的标签。有关各字段的说明，请参阅《Amazon CloudFormation API 参考》中的 Change 数据类型。

有关更改集的其他示例，请参阅CloudFormation 堆栈的更改集示例。

查看更改集中的属性级别更改（Amazon CLI）

- 以下命令列出了与 `AWS::EC2::NetworkInterface` 资源的一个更改集相关的属性级别更改，包括移除 `Ipv4Prefixes` 属性、修改该资源的 `Description` 并添加一个 `Tag`：

```
aws cloudformation describe-change-set --include-property-values \
  --change-set-name arn:aws:cloudformation:us-east-1:123456789012:changeSet/
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841
```

下面是示例输出。

```
"ChangeSetName": "ExampleChangeSet",
```

```
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:803642222207:changeSet/
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841",
    "StackId": "arn:aws:cloudformation:us-east-1:803642222207:stack/ExampleStack/
ab664180-f686-11ee-9e29-12cd92393671",
    "StackName": "ExampleStack",
    "Description": null,
    "Parameters": null,
    "CreationTime": "2024-04-09T18:04:59.935000+00:00",
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": null,
    "NotificationARNs": [],
    "RollbackConfiguration": {
        "RollbackTriggers": []
    },
    "Capabilities": [],
    "Tags": null,
    "ParentChangeSetId": null,
    "IncludeNestedStacks": true,
    "RootChangeSetId": null,
    "OnStackFailure": null,
 {
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId":
 "EC2NetworkInterface00eni067fd35b649a05b7100Tpyls",
                "PhysicalResourceId": "eni-067fd35b649a05b71",
                "ResourceType": "AWS::EC2::NetworkInterface",
                "Replacement": "False",
                "Scope": [
                    "Properties",
                    "Tags"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "Ipv4Prefixes",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Ipv4Prefixes",
                            "BeforeValue": "[]",
```

```
                            "AttributeChangeType": "Remove"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    },
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "Description",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Description",
                            "BeforeValue": "",
                            "AfterValue": "Description",
                            "AttributeChangeType": "Modify"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    },
                    {
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Tags/0",
                            "AfterValue": "{\"Key\":\"Test\",\"Value\":\"Test\"}",
                            "AttributeChangeType": "Add"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    }
                ],
                "BeforeContext": "{\"Properties\":{\"Description\":\"\",
\"PrivateIpAddress\":\"172.31.76.2\",\"PrivateIpAddresses\":[{\"PrivateIpAddress
\":\"172.31.76.2\",\"Primary\":\"true\"}],\"SecondaryPrivateIpAddressCount\":\"0\",
\"Ipv6PrefixCount\":\"0\",\"Ipv4Prefixes\":[],\"Ipv4PrefixCount\":\"0\",\"GroupSet
\":[\"sg-05a45689b1059e82d\"],\"Ipv6Prefixes\":[],\"SubnetId\":\"subnet-455e8969\",
\"SourceDestCheck\":\"true\",\"InterfaceType\":\"interface\",\"Tags\":[]},
\"UpdateReplacePolicy\":\"Retain\",\"DeletionPolicy\":\"Retain\"}",
                "AfterContext": "{\"Properties\":{\"Description\":\"Description\",
\"PrivateIpAddress\":\"172.31.76.2\",\"PrivateIpAddresses\":[{\"PrivateIpAddress
\":\"172.31.76.2\",\"Primary\":\"true\"}],\"SecondaryPrivateIpAddressCount
\":\"0\",\"Ipv6PrefixCount\":\"0\",\"Ipv4PrefixCount\":\"0\",\"GroupSet\":
[\"sg-05a45689b1059e82d\"],\"Ipv6Prefixes\":[],\"SubnetId\":\"subnet-455e8969\",
\"SourceDestCheck\":\"true\",\"InterfaceType\":\"interface\",\"Tags\":[{\"Value\":
```

```
\"Test\",\"Key\":\"Test\"}]},\"UpdateReplacePolicy\":\"Retain\",\"DeletionPolicy\":
\"Retain\"}"
            }
        }
    ],
    "ChangeSetName": "ExampleChangeSet",
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841",
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/ExampleStack/
ab664180-f686-11ee-9e29-12cd92393671",
```

使用偏差感知更改集

偏差感知更改集是增强的 CloudFormation 更改集，使您能够以安全的方式识别和管理堆栈偏差。在 CloudFormation 之外使用底层服务的 Amazon Web Services 管理控制台、Amazon CLI 或 SDK 更改堆栈资源时，堆栈会偏离预期的模板配置。偏差感知更改集将模板与堆栈资源的实际状态进行比较，并使偏差的资源与其模板定义保持一致。当资源的模板定义更新为与其实际状态相匹配时，偏差感知更改集会在不修改资源的情况下重置资源的偏差状态。

使用偏差感知更改集的好处包括：

- 预览偏差覆盖：检测部署是否会恢复在事件响应期间所做的关键带外更改。
- 系统的偏差协调：通过使用模板值覆盖偏差的属性或更新模板以匹配实际资源状态，将偏差的资源与合规的模板定义安全地对齐。
- 增强的回滚功能：如果部署失败，请将资源恢复到部署前的实际状态，而不是以前的模板状态。
- 完全可见性：在进行更改之前，请确切了解您的部署将如何影响实际的资源配置。

传统的更改集可以将您当前的模板与堆栈之前的模板进行对比，但并未考虑到堆栈的偏差情况。偏差感知更改集通过提供以下三种比较来解决此问题：

- 实际状态：资源的实时配置。在创建更改集时，CloudFormation 将通过底层服务 API 读取您账户中的资源配置。
- 之前的部署状态：在您上次进行 CloudFormation 部署所用的模板中定义的配置。
- 所需状态：在新模板中定义的配置。

偏差感知更改集能够将所有堆栈资源的实际状态更新至与期望状态一致的状态，即便该资源在模板中并未被明确更改。

注意事项

- 资源类型支持：偏差感知更改集支持将数百种资源类型的所需状态与实际状态进行对比。对于不支持的资源类型，偏差感知更改集会回退到将之前的部署状态与所需状态进行对比。请参阅资源类型支持限制部分了解详细信息。
- 只写属性：对于包含敏感数据（密码、机密）的属性，将偏差感知更改集与之前的部署值而不是实际值进行比较。
- Amazon 托管属性：偏差感知更改集会保留 Amazon 服务对托管属性所做的带外更改，例如自动扩缩组的所需容量。请参阅 Amazon 托管的属性部分了解详细信息。
- 外部标签键：偏差感知更改集不会移除或修改模板中未指定的标签键，从而防止与基于属性的访问权限控制（ABAC）系统发生冲突。
- 替换偏差的资源：偏差感知更改集不支持对不可变属性的偏差协调处理。
- 跨堆栈连接：某些资源类型（例如 `AWS::IAM::Policy`）可能会影响其他资源类型的实际状态，例如 `AWS::IAM::Role`。偏差感知更改集可以处理堆栈中资源的连接。如果通过连接来自其他堆栈的辅助资源来修改资源，则偏差感知更改集会将修改检测为偏差，并可以恢复连接。常用的可连接资源包括 `AWS::IAM::Policy`、`AWS::IAM::ManagedPolicy`、`AWS::EC2::SecurityGroupIngress` 和 `AWS::EC2::SecurityGroupEgress`。

Amazon 托管属性

您可以为活动的 Amazon 管理配置特定的资源属性。例如，您可以允许 Amazon Relational Database Service（Amazon RDS）自动升级 Amazon RDS 表的次要引擎版本。这些更改可能会在 CloudFormation 中显示为堆栈偏差。偏差感知更改集会识别出 Amazon 托管属性预计会出现偏差，如果您未在其模板中修改该属性，则保留其实际值不变。Amazon 托管属性的热门示例包括：

- 启用 Amazon RDS 表的 `AutoMinorVersionUpgrade` 属性以允许自动更新引擎版本。
- 使用 `AWS::ApplicationAutoScaling::ScalableTarget` 资源为 Amazon DynamoDB 表的读/写容量单位和 Amazon Elastic Container Service 集群的所需数量之类的属性启用自动扩缩功能。
- 对自动扩缩组使用 `AWS::AutoScaling::ScalingPolicy`。

偏差感知更改集阐明了标识为 Amazon 托管的属性。有关详细信息，请参阅 Amazon CLI 一节。

使用偏差感知更改集（控制台）

您可以通过 CloudFormation 控制台使用与传统更改集相同的工作流来创建和管理偏差感知更改集，并且还可以选择不同的部署模式。

创建偏差感知更改集

要创建偏差感知更改集（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面上，选择要更新的堆栈。
3. 选择堆栈操作，然后选择为当前堆栈创建更改集。
4. 在创建更改集页面上，对于更改集类型，选择偏差感知更改集。
5. 选择创建更改集。
6. 在更改集详细信息页面上，审核显示每个受影响资源的实际状态、先前部署状态和所需状态的三向比较。
7. 如果您对更改感到满意，请选择执行更改集。

查看偏差感知更改集

当您在控制台中查看偏差感知更改集时，您会看到与传统更改集相比的增强信息：

- 堆栈偏差状态：指示堆栈是否已与其上次部署产生偏差。
- 属性比较：显示受影响资源的实际状态和所需状态之间的 JSON 差异。
- 偏差指标：清晰地标出 JSON 差异中出现偏差的属性。单击查看偏差以查看某个属性的先前部署值。
- 值源指示：显示属性的值是来自实际状态还是先前的部署状态。

使用偏差感知更改集（Amazon CLI）

通过在 create-change-set 命令中添加 `--deployment-mode REVERT_DRIFT` 参数，您可以使用 Amazon CLI 创建和管理偏差感知更改集。

创建偏差感知更改集

要创建偏差感知更改集

使用带有 --deployment-mode REVERT_DRIFT 参数的 create-change-set 命令来创建偏差感知更改集。

```
aws cloudformation create-change-set \
  --stack-name my-stack \
  --change-set-name my-drift-aware-changeset \
  --template-body file://updated-template.yaml \
  --deployment-mode REVERT_DRIFT \
  --capabilities CAPABILITY_IAM
```

查看偏差感知更改集

要查看偏差感知更改集的详细信息

使用 describe-change-set 命令查看偏差感知更改集的详细信息：

```
aws cloudformation describe-change-set \
  --change-set-name my-drift-aware-changeset \
  --stack-name my-stack
```

响应包括偏差感知更改集的增强信息：

- StackDriftStatus：显示堆栈是否已出现偏差（DRIFTED、IN_SYNC、NOT_CHECKED 或 UNKNOWN）。
- ResourceDriftStatus：显示每个资源（DELETED、MODIFIED、IN_SYNC 或 NOT_CHECKED）的偏差状态。
- BeforeValueFrom：表示某个属性的“之前”值是来自 ACTUAL_STATE 还是 PREVIOUS_DEPLOYMENT_STATE。
- Drift：包含某个属性的偏差详细信息，包括 PreviousValue、ActualValue 和 DriftDetectionTimestamp。
- ResourceDriftIgnoredProperties：包含更改集不会恢复偏差的资源的属性以及忽略偏差的原因。

有关更多信息，请参阅《Amazon CloudFormation API 参考》中的 DescribeChangeSet。

执行偏差感知更改集

要执行偏差感知更改集

查看更改集后，使用 execute-change-set 命令应用更改：

```
aws cloudformation execute-change-set \
  --change-set-name my-drift-aware-changeset \
  --stack-name my-stack
```

如果部署失败，CloudFormation 会将资源回滚到实际的部署前状态，并保留自上次部署以来所做的带外更改。

资源类型支持限制

偏差感知更改集支持对所有支持偏差检测的资源类型的实际状态、先前的部署状态和所需状态进行三向比较，但以下资源类型除外：

- AWS::SageMaker::MlflowTrackingServer
- AWS::Route53Resolver::FirewallRuleGroup
- AWS::MediaLive::Multiplexprogram
- AWS::VpcLattice::ResourceConfiguration
- AWS::S3::StorageLensGroup
- AWS::Bedrock::AgentAlias
- AWS::MSK::Cluster
- AWS::RDS::DBProxy
- AWS::Redshift::ClusterParameterGroup
- AWS::QBusiness::Index
- AWS::NetworkManager::CoreNetwork
- AWS::IAM::OIDCProvider
- AWS::Organizations::ResourcePolicy
- AWS::SNS::TopicInlinePolicy
- AWS::Route53::KeySigningKey
- AWS::DataZone::PolicyGrant
- AWS::Transfer::Certificate
- AWS::SageMaker::ImageVersion
- AWS::Neptune::DBParameterGroup
- AWS::ODB::CloudVmCluster
- AWS::RolesAnywhere::TrustAnchor

- AWS::Detective::Graph
- AWS::Maester::DocumentType
- AWS::SageMaker::ModelPackageGroup
- AWS::S3Express::BucketPolicy
- AWS::Panorama::PackageVersion
- AWS::S3Tables::TableBucketPolicy

对于不支持三向比较的资源，偏差感知更改集会回退到资源的先前部署状态和所需状态的比较。

执行 CloudFormation 堆栈的更改集

要对您的堆栈进行更改集中所述的更改，请执行更改集。

> **Important**
>
> 在您执行更改集之后，CloudFormation 将删除与堆栈关联的所有其他更改集，因为它们对于更新后的堆栈不再有效。如果更新失败，您需要创建新的更改集。

> **Note**
>
> 如果更改集是使用 Express Mode (`--deployment-config '{"mode": "EXPRESS"}'`) 创建的，则当您执行更改集时，CloudFormation 会应用 Express Mode。Express Mode 在应用配置后立即完成资源操作，无需等待完全稳定。有关更多信息，请参阅 Express Mode。

堆栈策略和执行更改集

如果您在有堆栈策略关联的堆栈上执行更改集，CloudFormation 将在更新堆栈时强制执行策略。执行更改集时，您不能指定覆盖现有策略的临时堆栈策略。要更新受保护的资源，您必须更新堆栈策略或者使用直接更新方法。有关更多信息，请参阅 直接更新堆栈。

Execute a change set (console)

执行更改集

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。

2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面上，选择要更新的堆栈的名称。
4. 在导航窗格中，选择更改集以查看堆栈的更改集列表。
5. 选择要执行的更改集的名称。
6. 在更改集的详细信息页面上，选择执行更改集。

   CloudFormation 立即开始更新堆栈。CloudFormation 控制台会将您定向到 Events (事件) 选项卡，您可以在其中监控堆栈更新的进度。有关更多信息，请参阅 监控堆栈进度。

Execute a change set for nested stacks (console)

执行嵌套堆栈的更改集

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面上，选择要更新的堆栈的名称。您必须选择与根更改集关联的堆栈名称。
4. 在导航窗格中，选择更改集以查看堆栈的更改集列表。
5. 选择要执行的根更改集的名称。
6. 在更改集的详细信息页面上，选择执行更改集。

> **Note**
>
> 如果在 创建 CloudFormation 堆栈的更改集 过程中为嵌套堆栈的更改集选择了已启用，则 CloudFormation 会执行根更改集和嵌套更改集中描述的更改。

CloudFormation 立即开始更新堆栈。CloudFormation 控制台会将您定向到 Events (事件) 选项卡，您可以在其中监控堆栈更新的进度。有关更多信息，请参阅 监控堆栈进度。

执行更改集 (Amazon CLI)

- 运行 execute-change-set 命令。

  指定您要执行的更改集的更改集 ID，如下例中所示：

```
aws cloudformation execute-change-set \
    --change-set-name \
      arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

示例中的命令会执行 ID 为 `arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000` 的更改集。

运行命令之后，CloudFormation 开始更新堆栈。要查看堆栈的进度，请使用 describe-stacks 命令。

删除 CloudFormation 堆栈的更改集

删除更改集会将其从堆栈更改集列表中删除。删除更改集会阻止您或其他用户意外执行不应该应用的更改集。除非您删除更改集，否则 CloudFormation 将保留所有更改集，直至您更新了堆栈。

Delete a change set

删除更改集（控制台）

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面中，选择包含您要删除的更改集的堆栈名称。
4. 在导航窗格中，选择更改集以查看堆栈的更改集列表。
5. 选择要删除的更改集的名称。
6. 在更改集的详细信息页面上，选择删除更改集。

   CloudFormation 立即开始从堆栈的更改集列表中删除更改集，然后重定向到堆栈页面。

Delete a change set for nested stacks (console)

删除嵌套堆栈的更改集

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。

2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面中，选择与根更改集关联的堆栈名称。
4. 在导航窗格中，选择更改集以查看堆栈的更改集列表。
5. 选择要删除的更改集的名称。
6. 在更改集的详细信息页面上，选择 Delete (删除)。如果选择删除更改集，则将删除嵌套更改集的整个层次结构。

> **Note**
>
> 嵌套堆栈的更改集的删除操作是异步的，将会显示 DELETE_PENDING 状态，然后显示 DELETE_IN_PROGRESS 状态。删除更改集操作完成后，更改集会从列表中移除。如果处于 REVIEW_IN_PROGRESS 状态的嵌套堆栈是在创建更改集期间创建的，则也将删除这些堆栈。

CloudFormation 会立即开始从堆栈的更改集列表中删除更改集。

> **Note**
>
> 如果您的嵌套堆栈处于正在操作状态，请参阅 嵌套堆栈回滚失败 中的“纠正错误”。

删除更改集 (Amazon CLI)

- 运行 delete-change-set 命令，指定您要删除的更改集的 ID，如下例中所示：

```
aws cloudformation delete-change-set \
    --change-set-name \
      arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

CloudFormation 堆栈的更改集示例

此部分提供 CloudFormation 将为通用堆栈更改创建的更改集示例。它们显示如何直接编辑模板；修改单个输入参数；计划资源重新创建 (替换)，这可防止您丢失未备份的数据或者中断正在堆栈中运行的应用程序；以及添加和删除资源。为了说明更改集的工作原理，我们将演练提交的更改，并说明生成的更

改集。由于每个示例在之前的示例基础上构建并且假定您已了解之前的示例，我们建议您按顺序阅读。有关更改集中各个字段的说明，请参阅《Amazon CloudFormation API 参考》中的更改数据类型。

您可以使用控制台、Amazon CLI 或 CloudFormation DescribeChangeSet API 操作来查看更改集详细信息。

我们使用以下示例模板从堆栈生成以下各个更改集：

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample EC2 instance template for testing change sets.",
  "Parameters" : {
    "Purpose" : {
      "Type" : "String",
      "Default" : "testing",
      "AllowedValues" : ["testing", "production"],
      "Description" : "The purpose of this instance."
    },
    "KeyPairName" : {
      "Type": "AWS::EC2::KeyPair::KeyName",
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the
 instance"
    },
    "InstanceType" : {
      "Type" : "String",
      "Default" : "t2.micro",
      "AllowedValues" : ["t2.micro", "t2.small", "t2.medium"],
      "Description" : "The EC2 instance type."
    }
  },
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyPairName" },
        "InstanceType" : { "Ref" : "InstanceType" },
        "ImageId" : "ami-8fcee4e5",
        "Tags" : [
          {
            "Key" : "Purpose",
            "Value" : { "Ref" : "Purpose" }
          }
        ]
```

```
      }
    }
  }
}
```

直接编辑模板

当您在堆栈模板中直接修改资源以生成更改集时，CloudFormation 将更改分为直接修改类，这与由更新后的参数值启动的更改相对。以下更改集添加新标签到 i-1abc23d4 实例，是直接修改的示例。所有其他输入值，例如参数值和功能均不变，因此我们将侧重于 Changes 结构。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet-direct",
    "Parameters": [
        {
            "ParameterValue": "testing",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "MyKeyName",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
```

```
                    }
                ],
                "Action": "Modify",
                "Scope": [
                    "Tags"
                ],
                "LogicalResourceId": "MyEC2Instance",
                "Replacement": "False"
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T23:35:25.813Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-direct/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

在 Changes 结构中，只有一个 ResourceChange 结构。此结构描述一些信息，诸如 CloudFormation 将更改的资源类型、CloudFormation 将采取的操作、资源的 ID、更改的范围，以及更改是否需要替换（此时 CloudFormation 将创建新资源，然后删除旧资源）。在示例中，更改集指示 CloudFormation 将修改 i-1abc23d4 EC2 实例的 Tags 属性，无需替换实例。

在 Details 结构中，CloudFormation 将此更改标记为直接修改，永远无需重新创建实例（替换）。您可以充满信心地执行此更改，知道 CloudFormation 不会替换实例。

CloudFormation 将此更改显示为 Static 评估。静态评估意味着 CloudFormation 可以在执行更改集之前确定标签的值。在一些情况下，CloudFormation 只有在执行更改集之后才可以确定值。CloudFormation 将这些更改标记为 Dynamic 评估。例如，如果您引用有条件替换的更新后资源，CloudFormation 无法确定是否会更改对更新后资源的引用。

修改输入参数值

在您修改输入参数值时，CloudFormation 为使用更新后参数值的每个资源生成两个更改。在本示例中，我们希望突出显示这些更改的样子以及您应该关注哪些信息。以下示例通过仅更改 Purpose 输入参数的值生成。

Purpose 参数指定 EC2 实例的标签键值。在示例中，参数值从 testing 更改为 production。新值显示在 Parameters 结构中。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet",
    "Parameters": [
        {
            "ParameterValue": "production",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "MyKeyName",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Dynamic",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
                    },
                    {
                        "CausingEntity": "Purpose",
                        "ChangeSource": "ParameterReference",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
                    }
                ],
```

```
                "Action": "Modify",
                "Scope": [
                    "Tags"
                ],
                "LogicalResourceId": "MyEC2Instance",
                "Replacement": "False"
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T23:59:18.447Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

Changes 结构的工作原理与 直接编辑模板 示例中类似。只有一个 ResourceChange 结构；它描述了对 Tags EC2 实例的 i-1abc23d4 属性的更改。

但是，在 Details 结构中，更改集显示了 Tags 属性的两个更改，即使只更改了一个参数值。引用更改后参数值的资源 (使用 Ref 内部函数) 始终会导致两个更改：一个为 Dynamic 评估，另一个为 Static 评估。您可以通过查看以下字段查看这些更改类型：

- 对于 Static 评估更改，请查看 ChangeSource 字段。在此示例中，ChangeSource 字段等于 ParameterReference，意味着此更改是更新参数引用值的结果。更改集必须包含类似的 Dynamic 评估更改。
- 您可以通过比较所有更改的 Dynamic 结构来查找匹配的 Target 评估更改，这会包含同样的信息。在此示例中，所有更改的 Target 结构包含 Attribute 和 RequireRecreation 字段的相同值。

对于这些类型的更改，重点在于静态评估，这向您提供了有关更改最详细的信息。在此示例中，静态评估显示更改是对参数引用值 (ParameterReference) 进行更改的结果。进行更改的确切参数由 CauseEntity 字段 (Purpose 参数) 指定。

确定“Replacement”字段中的值

ResourceChange 结构中的 Replacement 字段指示 CloudFormation 是否重新创建资源。计划重新创建资源 (替换) 可防止您丢失未备份的数据或者中断正在堆栈中运行的应用程序。

Replacement 字段中的值取决于更改是否需要替换，这由更改的 RequiresRecreation 结构中的 Target 字段指示。例如，如果 RequiresRecreation 字段为 Never，则 Replacement 字段为 False。但是，如果单个资源上有多个更改，并且每个更改有不同的 RequiresRecreation 字段值，CloudFormation 将使用最具侵入性的行为更新资源。换而言之，即使多个更改中只有一个需要替换，CloudFormation 必须替换资源，因此，将 Replacement 字段设置为 True。

以下更改集通过更改每个参数 (Purpose、InstanceType 和 KeyPairName) 的值生成，这些参数均由 EC2 实例使用。通过这些更改，CloudFormation 需要替换实例，因为 Replacement 字段等于 True。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet-multiple",
    "Parameters": [
        {
            "ParameterValue": "production",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "MyNewKeyName",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.small",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-7bef86f8",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Dynamic",
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "KeyName",
                            "RequiresRecreation": "Always"
```

```
                    }
                },
                {
                    "ChangeSource": "DirectModification",
                    "Evaluation": "Dynamic",
                    "Target": {
                        "Attribute": "Properties",
                        "Name": "InstanceType",
                        "RequiresRecreation": "Conditionally"
                    }
                },
                {
                    "ChangeSource": "DirectModification",
                    "Evaluation": "Dynamic",
                    "Target": {
                        "Attribute": "Tags",
                        "RequiresRecreation": "Never"
                    }
                },
                {
                    "CausingEntity": "KeyPairName",
                    "ChangeSource": "ParameterReference",
                    "Evaluation": "Static",
                    "Target": {
                        "Attribute": "Properties",
                        "Name": "KeyName",
                        "RequiresRecreation": "Always"
                    }
                },
                {
                    "CausingEntity": "InstanceType",
                    "ChangeSource": "ParameterReference",
                    "Evaluation": "Static",
                    "Target": {
                        "Attribute": "Properties",
                        "Name": "InstanceType",
                        "RequiresRecreation": "Conditionally"
                    }
                },
                {
                    "CausingEntity": "Purpose",
                    "ChangeSource": "ParameterReference",
                    "Evaluation": "Static",
                    "Target": {
```

```
                        "Attribute": "Tags",
                        "RequiresRecreation": "Never"
                    }
                }
            ],
            "Action": "Modify",
            "Scope": [
                "Tags",
                "Properties"
            ],
            "LogicalResourceId": "MyEC2Instance",
            "Replacement": "True"
        },
        "Type": "Resource"
    }
],
"CreationTime": "2020-11-18T00:39:35.974Z",
"Capabilities": [],
"StackName": "MyStack",
"NotificationARNs": [],
"ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-multiple/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

通过查看各个更改 (Details 结构中的静态评估) 确定需要替换资源的更改。在此示例中，各个更改具有不同的 RequireRecreation 字段值，但对 KeyName 属性的更改是最具侵入性的更新行为，始终需要重新创建。CloudFormation 将替换实例，因为其键名称已更改。

如果键名称未更改，则对 InstanceType 属性的更改将为最具侵入性的更新行为 (Conditionally)，因此 Replacement 字段将为 Conditionally。要查找 CloudFormation 替换实例的条件，请查看 AWS::EC2::Instance 资源类型的 InstanceType 属性的更新行为。

添加和删除资源

以下示例通过提交修改后的模板生成，此修改删除了 EC2 实例，并添加了自动扩缩组和启动配置。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet-addremove",
    "Parameters": [
        {
```

```
            "ParameterValue": "testing",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "MyKeyName",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "Action": "Add",
                "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
                "Scope": [],
                "Details": [],
                "LogicalResourceId": "AutoScalingGroup"
            },
            "Type": "Resource"
        },
        {
            "ResourceChange": {
                "Action": "Add",
                "ResourceType": "AWS::AutoScaling::LaunchConfiguration",
                "Scope": [],
                "Details": [],
                "LogicalResourceId": "LaunchConfig"
            },
            "Type": "Resource"
        },
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [],
                "Action": "Remove",
                "Scope": [],
                "LogicalResourceId": "MyEC2Instance"
            },
            "Type": "Resource"
        }
```

```
    ],
    "CreationTime": "2020-11-18T01:44:08.444Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-addremove/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

在 Changes 结构中，有三个 ResourceChange 结构，每个资源一个。对于每个资源，Action 字段指示 CloudFormation 添加还是删除资源。Scope 和 Details 字段为空，因为它们仅适用于修改后的资源。

对于新资源，CloudFormation 无法确定一些字段的值，直至您执行更改集。例如，CloudFormation 不提供自动扩缩组的物理 ID 和启动配置，因为它们尚不存在。在您执行更改集时，CloudFormation 创建新资源。

查看属性级别更改

以下示例说明了对 Amazon EC2 实例的 Tag 属性的属性级别更改。标签 Value 和 Key 将变为 Test。

```
"ChangeSetName": "SampleChangeSet",
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/38d91d27-798d-4736-9bf1-fb7c46207807",
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
SampleEc2Template/68edcdc0-f6b6-11ee-966c-126d572cdd11",
    "StackName": "SampleEc2Template",
    "Description": "A sample EC2 instance template for testing change sets.",
    "Parameters": [
        {
            "ParameterKey": "KeyPairName",
            "ParameterValue": "BatchTest"
        },
        {
            "ParameterKey": "Purpose",
            "ParameterValue": "testing"
        },
        {
            "ParameterKey": "InstanceType",
            "ParameterValue": "t2.micro"
        }
    ],
```

```
    "CreationTime": "2024-04-09T21:29:10.759000+00:00",
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": null,
    "NotificationARNs": [],
    "RollbackConfiguration": {
:...skipping...
{
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId": "MyEC2Instance",
                "PhysicalResourceId": "i-0cc7856a36315e62b",
                "ResourceType": "AWS::EC2::Instance",
                "Replacement": "False",
                "Scope": [
                    "Tags"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Tags/0/Value",
                            "BeforeValue": "testing",
                            "AfterValue": "Test",
                            "AttributeChangeType": "Modify"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    },
                    {
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Tags/0/Key",
                            "BeforeValue": "Purpose",
                            "AfterValue": "Test",
                            "AttributeChangeType": "Modify"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
```

```
                    }
                ],
                "BeforeContext": "{\"Properties\":{\"KeyName\":\"BatchTest\",\"ImageId
\":\"ami-8fcee4e5\",\"InstanceType\":\"t2.micro\",\"Tags\":[{\"Value\":\"testing\",
\"Key\":\"Purpose\"}]}}",
                "AfterContext": "{\"Properties\":{\"KeyName\":\"BatchTest\",\"ImageId
\":\"ami-8fcee4e5\",\"InstanceType\":\"t2.micro\",\"Tags\":[{\"Value\":\"Test\",\"Key
\":\"Test\"}]}}"
            }
        }
    ]
```

Details 结构说明了 Key 和 Value 在更改集执行之前的值，以及更改集执行之后将会具有的值。

嵌套堆栈的更改集

使用嵌套堆栈的更改集，您可以预览整个嵌套堆栈层次结构中应用程序和基础架构资源的更改，并在确认所有更改均是预期更改后继续进行更新。

有关嵌套堆栈的更改集的更多详细信息，请参阅以下各节：

主题

- 更改集和嵌套堆栈概述
- 使用嵌套堆栈的更改集（控制台）
- 使用嵌套堆栈的更改集 (Amazon CLI)

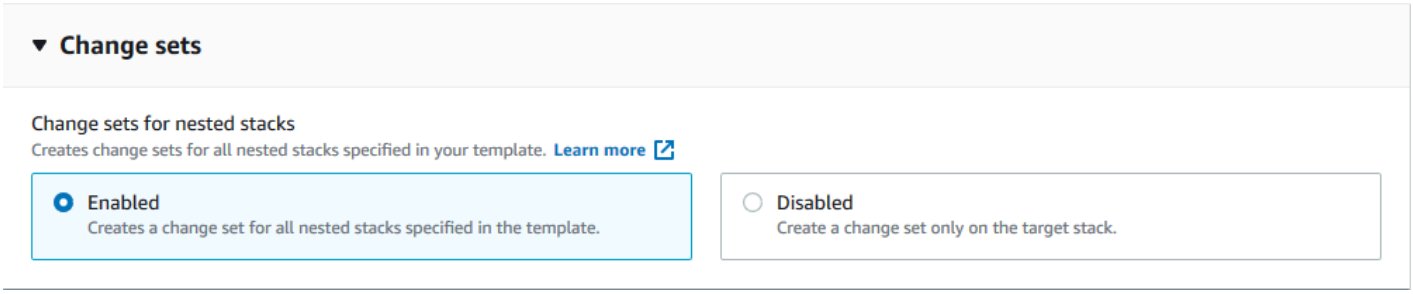
更改集和嵌套堆栈概述

嵌套堆栈的更改集将以下功能组合在一起，以扩大对整个堆栈层次结构作出的更改的预览范围：

- 更改集是 CloudFormation 的一种功能，可用来预览对堆栈的建议更改将会对现有或新创建资源带来的影响。创建更改集时，CloudFormation 通过将堆栈与您提交的资源更改进行比较，提供建议更改列表。有关更改集的更多信息，请参阅使用更改集更新 CloudFormation 堆栈。
- 嵌套堆栈是作为其他堆栈的一部分创建的堆栈。例如，在一个嵌套堆栈中可能具有与联网和安全相关的资源，而在另一个嵌套堆栈中具有应用程序资源。以这种方式对应用程序模型进行分区有助于提高代码的可维护性和重复使用。有关嵌套堆栈的更多信息，请参阅使用嵌套堆栈将模板拆分为可重复使用的部分。

使用嵌套堆栈的更改集（控制台）

- 创建更改集 – 通过从堆栈层次结构任何级别提交更改来创建更改集。您可以提交修改后的堆栈模板或修改后的输入参数值，CloudFormation 会比较嵌套堆栈与您为生成更改集而提交的更改。默认情况下，CloudFormation 控制台中的嵌套堆栈的更改集为已启用状态。有关更多信息，请参阅 创建 CloudFormation 堆栈的更改集。



> **Note**
>
> 根更改集是指与从中创建更改集的整个层次结构的堆栈关联的更改集。必须从根更改集中执行或删除嵌套堆栈的更改集。有关更多信息，请参阅 在嵌套堆栈上执行堆栈操作。

- 查看更改集 – 在执行嵌套堆栈内资源的更改之前，可视化查看这些更改。您可以导航到当前堆栈及其嵌套更改集，在更改集的更改部分查看建议更改。有关更多信息，请参阅 查看 CloudFormation 堆栈的更改集。
- 执行更改集 – 执行与当前堆栈及其子代相关的更改集中所述的更改。执行操作必须从根更改集进行。有关更多信息，请参阅 执行 CloudFormation 堆栈的更改集。
- 删除更改集 – 从当前堆栈中删除更改集。删除更改集有助于防止您或其他用户意外启动不应该应用的更改集。删除操作必须从根更改集中执行。有关更多信息，请参阅 删除 CloudFormation 堆栈的更改集。

使用嵌套堆栈的更改集 (Amazon CLI)

create-change-set

- create-change-set - 默认情况下，未对 Amazon CLI 启用嵌套堆栈的更改集。要为整个堆栈层次结构创建更改集，请指定 `--include-nested-stacks` 选项。有关更多信息，请参阅 创建 CloudFormation 堆栈的更改集。

以下 Amazon CLI 示例为指定的根堆栈创建了一个更改集。

```
aws cloudformation create-change-set \
    --stack-name my-root-stack \
    --change-set-name my-root-stack-change-set \
    --template-body file://template.yaml \
    --capabilities CAPABILITY_IAM \
    --include-nested-stacks
```

下面是示例输出。

```
{
    "Id":"arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:Stack/my-root-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204"
}
```

describe-change-set

- describe-change-set –返回执行更改集时 CloudFormation 将作出的更改的列表。如果指定的更改集包含属于嵌套堆栈的子更改集，则 ChangeSetId 将返回有关该更改集的信息。有关更多信息，请参阅 查看 CloudFormation 堆栈的更改集。

以下 Amazon CLI 示例描述了指定根堆栈的更改集。

```
aws cloudformation describe-change-set \
    --change-set-name my-root-stack-change-set \
    --stack-name my-root-stack
```

下面是示例输出。

```
{
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId": "ChildStack",
                "PhysicalResourceId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-nested-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99205",
                "ResourceType": "AWS::CloudFormation::Stack",
                "Replacement": "False",
```

```
                "ChangeSetId": "arn:aws:cloudformation:us-
west-2:123456789012:changeSet/my-nested-stack-change-set/4eca1a01-e285-
xmpl-8026-9a1967bfb4b0",
                "Scope": [
                    "Properties"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "RequiresRecreation": "Never"
                        },
                        "Evaluation": "Dynamic",
                        "ChangeSource": "Automatic"
                    }
                ]
            }
        }
    ],
    "ChangeSetName": "my-root-stack-change-set",
    "ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-
stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-root-stack/
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
    "StackName": "my-root-stack",
    "IncludeNestedStacks": true,
    "ParentChangeSetId": null,
    "RootChangeSetId": null,
    "Description": null,
    "Parameters": null,
    "CreationTime": "2020-11-18T05:20:56.651Z",
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": null,
    "NotificationARNs": [

    ],
    "RollbackConfiguration": {

    },
    "Capabilities": [
        "CAPABILITY_IAM"
    ],
    "Tags": null
```

```
}
```

以下 Amazon CLI 示例描述了指定嵌套堆栈的更改集。

```
aws cloudformation describe-change-set \
    --change-set-name my-nested-stack-change-set \
    --stack-name my-nested-stack
```

下面是示例输出。

```
{
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId": "function",
                "PhysicalResourceId": "my-function",
                "ResourceType": "AWS::Lambda::Function",
                "Replacement": "False",
                "ChangeSetId": null,
                "Scope": [
                    "Properties"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "Timeout",
                            "RequiresRecreation": "Never"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    }
                ]
            }
        }
    ],
    "ChangeSetName": "my-nested-stack-change-set",
    "ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-nested-
stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-nested-stack/
d0a825a0-e4cd-xmpl-b9fb-061c69e99205",
```

```
    "ParentChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "RootChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "IncludeNestedStacks": true,
    "StackName": "my-nested-stack",
    "Description": null,
    "Parameters": null,
    "CreationTime": "2020-11-18T05:20:56.651Z",
    "ExecutionStatus": "UNAVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": "Executable from root change set",
    "NotificationARNs": [

    ],
    "RollbackConfiguration": {

    },
    "Capabilities": [
        "CAPABILITY_IAM"
    ],
    "Tags": null
}
```

execute-change-set

- execute-change-set – 使用在创建指定的更改集时提供的输入信息创建或更新堆栈。要为整个堆栈层次结构创建更改集，必须在 create-change-set 操作期间指定 `--include-nested-stacks` 选项。有关更多信息，请参阅 执行 CloudFormation 堆栈的更改集。

> **Note**
>
> execute-change-set 必须从根更改集中执行，并将该更改集应用于整个堆栈层次结构。

以下 Amazon CLI 示例执行了指定根堆栈的更改集。

```
aws cloudformation execute-change-set \
    --stack-name my-root-stack \
    --change-set-name my-root-stack-change-set
```

delete-change-set

- delete-change-set – 删除指定的更改集。删除更改集可确保不会使用错误的更改集。删除更改集与使用 `--include-nested-stacks` 选项创建的更改集是不同步的。有关更多信息，请参阅 删除 CloudFormation 堆栈的更改集。

> **Note**
>
> delete-change-set 必须从根更改集中执行，并会删除更改集的整个层次结构。如果处于 `REVIEW_IN_PROGRESS` 状态的嵌套堆栈是在 create-change-set 操作过程期间创建的，则它们也会被删除。

以下 Amazon CLI 示例删除了指定根堆栈的更改集。

```
aws cloudformation delete-change-set \
    --stack-name my-root-stack \
    --change-set-name my-root-stack-change-set
```

验证堆栈部署

借助部署前验证，您可以在 CloudFormation 配置资源之前识别并解决潜在的部署问题。部署前验证会在“创建堆栈”、“更新堆栈”和“创建更改集”操作中自动运行，并在几秒钟内检测到常见错误。

主题

- 部署前验证的工作原理
- 注意事项
- 先决条件
- 禁用部署前验证
- 验证堆栈部署（控制台）
- 验证堆栈部署（Amazon CLI）
- 验证类型
- 资源限制

部署前验证的工作原理

部署前验证涉及以下阶段：

1. 启动堆栈操作 – 按常规方式创建堆栈、更新堆栈或创建变更集。默认情况下，部署前验证在所有三项操作上启用。
2. 验证执行：CloudFormation 会针对您的模板和目标环境运行多项验证检查。CloudFormation 支持六种类型的部署前验证。属性语法验证和资源名称冲突检测在所有支持的操作上运行。仅在创建更改集期间运行另外四项验证：S3 存储桶空性验证、服务配额检查、记录器冲突检测和 ECR 存储库删除就绪检查。
3. 审核验证结果：CloudFormation 会对发现的任何问题提供详细的反馈，包括在模板中定位问题位置的精确路径，无需手动调试模板。
4. 解决问题：在继续部署之前，通过更新模板或解决冲突来解决已识别的问题。
5. 放心执行 - 在确信常见故障场景已提前经过验证的情况下，继续进行部署。

> **Note**
>
> 对于“创建堆栈”和“更新堆栈”操作，验证作为操作本身的一部分自动执行。如果验证以“FAIL”模式失败，操作将在任何资源被配置之前停止，且堆栈状态将反映验证失败的情况。对于更改集，更改集状态会显示 `FAILED`。

注意事项

使用部署前验证时，请牢记以下内容：

- 部署前验证侧重于常见的部署故障场景。这并不能保证您的部署会成功，但能降低常见故障发生的可能性。
- 验证模式的运作方式各不相同：
  - 当验证过程中检测到错误时，FAIL 模式会在任何资源被配置之前终止操作。对于 CreateStack 和 UpdateStack 操作，操作将失败，且堆栈状态将反映验证失败的情况。对于 CreateChangeSet，更改集状态将显示为“FAILED”。这适用于属性语法错误以及资源命名冲突的情况。
  - WARN 模式允许在验证失败的情况下仍能成功创建更改集，同时会给出警告信息，供开发人员在执行操作前进行查看和解决。这适用于约束违规问题，例如 S3 存储桶为空等，而这些问题通常可以通过人工干预来解决。

- 对于更改集操作，验证结果与特定的更改集相关联。如果您修改模板，则需要创建新的更改集以获取更新的验证结果。对于 CreateStack 和 UpdateStack 操作，请使用 `describe-events` 命令并指定操作 ID 来获取验证结果。
- S3 存储桶验证仅检查对象是否存在，而不检查存储桶策略或其他可能阻止删除的约束。
- 部署前验证会在验证运行期间为 CreateStack 和 UpdateStack 操作增加少量延迟。通常仅需几秒钟。如果您需要跳过特定操作的验证，请使用 `DisableValidation` 参数。

先决条件

要使用部署前验证，您必须具备：

- 创建堆栈、更新堆栈或创建更改集，以及读取您账户中资源所需的 IAM 权限。在创建变更集期间运行的验证检查需要以下额外权限：
  - 服务配额检查 - `cloudwatch:GetMetricData`、`lambda:GetAccountSettings`、`servicequotas:GetServiceQ` 和 `iam:GetAccountSummary`
  - 记录器冲突检查 - `config:ListConfigurationRecorders`
  - S3 存储桶空性和 ECR 存储库检查 - `s3:ListBucketV2` 和 `ecr:ListImages`
- 访问您部署堆栈所在的 Amazon Web Services 区域。
- 您要在部署之前验证的 CloudFormation 模板。

禁用部署前验证

部署前验证默认自动运行。要跳过特定操作的验证，请使用 `DisableValidation` 参数。

在以下情况下禁用验证：

- 当您已经通过其他方式（例如，`cfn-lint`、`cdk validate` 或 CI 检查）验证模板时
- 当您需要最大限度地降低对时间敏感型部署的操作延迟时
- 当已知的误报阻碍了您的部署，而您需要继续进行时

Amazon CLI 示例：

```
aws cloudformation create-stack --stack-name MyStack --template-body file://
template.yaml --disable-validation
```

```
aws cloudformation update-stack --stack-name MyStack --template-body file://
template.yaml --disable-validation
```

API：在 CreateStack 或 UpdateStack API 调用中将 DisableValidation 参数设置为 true。

> **Note**
>
> 禁用验证意味着 CloudFormation 不会在尝试配置资源之前捕获常见错误。

验证堆栈部署（控制台）

部署前验证会在 CreateStack 和 UpdateStack 操作中自动运行。无需执行其他步骤。

要在堆栈创建或更新期间进行验证

1. 按常规流程创建或更新您的堆栈。
2. 如果验证过程中检测到错误，操作将在配置开始前停止。堆栈状态将反映验证失败的情况。
3. 要查看验证结果，请打开堆栈的事件选项卡并选择操作 ID，或者点击横幅中的链接或“状态原因”列中的链接。“操作”视图页面将直接打开至“部署验证”选项卡，该选项卡显示验证详细信息，包括逻辑 ID、资源类型、验证类型、模式和状态原因。
4. 在模板中修复已识别的问题，然后重试该操作。

使用更改集进行验证（控制台）

您还可以使用更改集在部署前对模板进行验证。更改集支持全部六种验证类型，包括仅在创建更改集时可用的 WARN 模式验证（S3 存储桶空性检查、服务配额验证、配置记录器冲突检测以及 ECR 存储库删除就绪检查）。您可以使用更改集在创建新堆栈前对其进行验证，或对现有堆栈的更新进行验证。

要使用更改集进行验证

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的堆栈所在的 Amazon Web Services 区域。
3. 在堆栈页面上，选择您要为其创建更改集的正在运行的堆栈。
4. 在堆栈详细信息窗格中，选择更新堆栈，然后选择创建更改集。
5. 在为 ***stack-name*** 创建更改集页面上，上传更新后的模板或指定模板来源。

6. 选择下一步以继续完成其余的更改集配置步骤。
7. 如果模板包含 IAM 资源，则对于功能，请选择我确认 CloudFormation 可能会创建 IAM 资源。IAM 资源可以修改您 Amazon 账户中的权限；查看这些资源以确保只允许您打算执行的操作。有关更多信息，请参阅确认 CloudFormation 模板中的 IAM 资源。
8. 在审核页面上，选择创建更改集。
9. CloudFormation 会创建更改集并执行验证检查。在部署验证选项卡中审核验证结果。
10. 如果验证通过或者您对警告感到满意，请选择执行更改集来部署您的更改。
11. 如果验证失败，请修复问题并创建新的更改集以重新验证您的部署。

验证堆栈部署（Amazon CLI）

在执行 `create-stack` 和 `update-stack` 命令时，部署前验证会自动运行。

创建堆栈并进行验证

使用以下命令来创建 堆栈。部署前验证会自动运行。

```
aws cloudformation create-stack \
  --stack-name MyStack \
  --template-body file://template.yaml
```

更新堆栈并进行验证

使用以下命令来更新堆栈。部署前验证会自动运行。

```
aws cloudformation update-stack \
  --stack-name MyStack \
  --template-body file://updated-template.yaml
```

查看验证结果

使用以下命令来查看堆栈操作的验证结果。

```
aws cloudformation describe-events \
  --stack-name MyStack
```

验证错误的示例输出：

```
{
   "OperationEvents":[
      {
         "EventId":"9b5c9a29-4704-4ad0-8082-afb49418d55b",
         "StackId":"arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/
c3908380-b357-11f0-a97f-0ad08f35df65",
         "OperationId":"f558b823-e1e3-4de3-a222-e6b930ddcad4",
         "OperationType":"CREATE_STACK",
         "OperationStatus":"FAILED",
         "EventType":"STACK_EVENT",
         "Timestamp":"2025-10-27T17:10:02.923Z",
         "StartTime":"2025-10-27T17:09:57.537Z",
         "EndTime":"2025-10-27T17:10:02.923Z"
      },
      {
         "EventId":"2d8c3262-3468-4283-82fb-6e780e9e4f1d",
         "StackId":"arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/
c3908380-b357-11f0-a97f-0ad08f35df65",
         "OperationId":"f558b823-e1e3-4de3-a222-e6b930ddcad4",
         "OperationType":"CREATE_STACK",
         "EventType":"VALIDATION_ERROR",
         "LogicalResourceId":"NotificationBucket",
         "PhysicalResourceId":"",
         "ResourceType":"AWS::S3::Bucket",
         "Timestamp":"2025-10-27T17:10:02.461Z",
         "ValidationFailureMode":"FAIL",
         "ValidationName":"PROPERTY_VALIDATION",
         "ValidationStatus":"FAILED",
         "ValidationStatusReason":"#/NotificationConfiguration/QueueConfigurations/0:
 required key [Event] not found",
         "ValidationPath":"/Resources/NotificationBucket/Properties/
NotificationConfiguration/QueueConfigurations/0"
      }
   ]
}
```


使用更改集进行验证 (Amazon CLI)

更改集还会揭示在直接堆栈操作中不可用的 WARN 模式验证。

要使用更改集进行验证

1. 使用 create-change-set 命令：

```
aws cloudformation create-change-set \
  --stack-name MyStack \
  --change-set-name MyChangeSet \
  --change-set-type "CREATE" \
  --template-body file://updated-template.yaml
```

该命令将同时返回更改集 ARN 和堆栈 ARN。

2. 使用 describe-events 命令并指定更改集 ARN，以查看验证状态和结果。

```
aws cloudformation describe-events \
  --change-set-name "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
MyChangeSet/94498df5-1afb-43b1-9869-9f82b2d877ac"
```

3. 通过更新模板来解决任何验证错误，然后创建新的更改集。
4. 验证通过后，执行该更改集：

```
aws cloudformation execute-change-set \
  --change-set-name MyChangeSet \
  --stack-name MyStack
```

验证类型

部署前验证包括以下类型的检查：

- 属性语法验证：根据 Amazon 资源架构验证资源属性。它会检查必需的属性和有效的属性值，并识别已弃用或不受支持的属性组合。
- 资源名称冲突检测：检查与现有 Amazon 资源的命名是否存在冲突。它会验证资源名称是否符合 Amazon 命名要求，并在尝试部署之前识别出潜在的冲突。
- S3 存储桶空置性验证：尝试删除包含对象的 S3 存储桶时发出警告。它提供对象计数以帮助评测删除影响，并帮助防止常见的 S3 删除失败。
- 服务配额验证 – 检查创建资源是否会超过您的 Amazon 服务配额。该验证目前仅在创建更改集时运行，且在“警告”模式下进行。操作将继续进行，您将收到有关潜在配额问题的警报。
- 记录器冲突检测 - 当您的模板向未启用记录功能的账户添加规则，或者在已有记录器处于活动状态的账户中定义记录器时，会发出警告。该验证目前仅在创建更改集时运行，且在“警告”模式下进行。
- ECR 存储库删除就绪检查 - 验证待删除的 ECR 存储库是否为空，或者是否已配置适当的强制删除设置。该验证目前仅在创建更改集时运行，且在“警告”模式下进行。

下表总结了每种验证类型、其模式以及其执行的操作。

验证类型	Mode	可用于
属性语法验证	FAIL	CreateStack、UpdateStack、CreateChangeSet
资源名称冲突 (RAE)	FAIL	CreateStack、UpdateStack、CreateChangeSet
S3 存储桶空性	警告	CreateChangeSet
服务配额	警告	CreateChangeSet
记录器冲突	警告	CreateChangeSet
ECR 存储库删除就绪	警告	CreateChangeSet

每种验证类型都提供了特定的错误信息以及错误在模板中的具体位置，以帮助您迅速解决问题。

资源限制

部署前验证不支持以下资源类型：

- `AWS::ApiGatewayV2::ApiGatewayManagedOverrides`
- `AWS::ApiGatewayV2::Stage`
- `AWS::AppMesh::GatewayRoute`
- `AWS::AppMesh::Mesh`
- `AWS::AppMesh::Route`
- `AWS::AppMesh::VirtualGateway`
- `AWS::AppMesh::VirtualNode`
- `AWS::AppMesh::VirtualRouter`
- `AWS::AppMesh::VirtualService`
- `AWS::AppStream::Fleet`
- `AWS::AppStream::Stack`
- `AWS::AppStream::StackFleetAssociation`

- AWS::AppStream::StackUserAssociation
- AWS::AppStream::User
- AWS::AppSync::ApiCache
- AWS::AppSync::ApiKey
- AWS::AppSync::GraphQLSchema
- AWS::AutoScalingPlans::ScalingPlan
- AWS::Budgets::Budget
- AWS::CertificateManager::Certificate
- AWS::Cloud9::EnvironmentEC2
- AWS::CloudFormation::CustomResource
- AWS::CloudFormation::Macro
- AWS::CloudFormation::WaitCondition
- AWS::CloudFormation::WaitConditionHandle
- AWS::CloudFront::StreamingDistribution
- AWS::CloudWatch::AnomalyDetector
- AWS::CloudWatch::InsightRule
- AWS::CodeBuild::Project
- AWS::CodeBuild::ReportGroup
- AWS::CodeBuild::SourceCredential
- AWS::CodeCommit::Repository
- AWS::CodeDeploy::DeploymentGroup
- AWS::CodeStar::GitHubRepository
- AWS::Config::ConfigurationRecorder
- AWS::Config::DeliveryChannel
- AWS::Config::OrganizationConfigRule
- AWS::Config::RemediationConfiguration
- AWS::DAX::Cluster
- AWS::DAX::ParameterGroup
- AWS::DAX::SubnetGroup

- AWS::DirectoryService::MicrosoftAD
- AWS::DLM::LifecyclePolicy
- AWS::DMS::Certificate
- AWS::DMS::Endpoint
- AWS::DMS::EventSubscription
- AWS::DMS::ReplicationInstance
- AWS::DMS::ReplicationSubnetGroup
- AWS::DMS::ReplicationTask
- AWS::DocDB::DBCluster
- AWS::DocDB::DBClusterParameterGroup
- AWS::DocDB::DBInstance
- AWS::DocDB::DBSubnetGroup
- AWS::DocDB::EventSubscription
- AWS::EC2::ClientVpnAuthorizationRule
- AWS::EC2::ClientVpnEndpoint
- AWS::EC2::ClientVpnRoute
- AWS::EC2::ClientVpnTargetNetworkAssociation
- AWS::EC2::NetworkInterfacePermission
- AWS::ElastiCache::CacheCluster
- AWS::ElastiCache::ReplicationGroup
- AWS::ElastiCache::SecurityGroup
- AWS::ElastiCache::SecurityGroupIngress
- AWS::ElasticLoadBalancing::LoadBalancer
- AWS::ElasticLoadBalancingV2::ListenerCertificate
- AWS::Elasticsearch::Domain
- AWS::EMR::Cluster
- AWS::EMR::InstanceFleetConfig
- AWS::EMR::InstanceGroupConfig
- AWS::FSx::FileSystem
- AWS::FSx::Snapshot

- AWS::FSx::StorageVirtualMachine
- AWS::FSx::Volume
- AWS::Glue::Classifier
- AWS::Glue::Connection
- AWS::Glue::CustomEntityType
- AWS::Glue::DataCatalogEncryptionSettings
- AWS::Glue::DataQualityRuleset
- AWS::Glue::DevEndpoint
- AWS::Glue::MLTransform
- AWS::Glue::Partition
- AWS::Glue::SecurityConfiguration
- AWS::Glue::Table
- AWS::Glue::TableOptimizer
- AWS::Glue::Workflow
- AWS::Greengrass::ConnectorDefinition
- AWS::Greengrass::ConnectorDefinitionVersion
- AWS::Greengrass::CoreDefinition
- AWS::Greengrass::CoreDefinitionVersion
- AWS::Greengrass::DeviceDefinition
- AWS::Greengrass::DeviceDefinitionVersion
- AWS::Greengrass::FunctionDefinition
- AWS::Greengrass::FunctionDefinitionVersion
- AWS::Greengrass::Group
- AWS::Greengrass::GroupVersion
- AWS::Greengrass::LoggerDefinition
- AWS::Greengrass::LoggerDefinitionVersion
- AWS::Greengrass::ResourceDefinition
- AWS::Greengrass::ResourceDefinitionVersion
- AWS::Greengrass::SubscriptionDefinition
- AWS::Greengrass::SubscriptionDefinitionVersion

- AWS::IAM::AccessKey
- AWS::IAM::UserToGroupAddition
- AWS::IoT::PolicyPrincipalAttachment
- AWS::IoT::ThingPrincipalAttachment
- AWS::IoTThingsGraph::FlowTemplate
- AWS::KinesisAnalytics::Application
- AWS::KinesisAnalytics::ApplicationOutput
- AWS::KinesisAnalytics::ApplicationReferenceDataSource
- AWS::KinesisAnalyticsV2::ApplicationCloudWatchLoggingOption
- AWS::KinesisAnalyticsV2::ApplicationOutput
- AWS::KinesisAnalyticsV2::ApplicationReferenceDataSource
- AWS::LakeFormation::DataLakeSettings
- AWS::LakeFormation::Permissions
- AWS::LakeFormation::Resource
- AWS::ManagedBlockchain::Member
- AWS::ManagedBlockchain::Node
- AWS::MediaConvert::JobTemplate
- AWS::MediaConvert::Preset
- AWS::MediaConvert::Queue
- AWS::MediaLive::Channel
- AWS::MediaLive::Input
- AWS::MediaLive::InputSecurityGroup
- AWS::MediaStore::Container
- AWS::OpsWorks::App
- AWS::OpsWorks::ElasticLoadBalancerAttachment
- AWS::OpsWorks::Instance
- AWS::OpsWorks::Layer
- AWS::OpsWorks::Stack
- AWS::OpsWorks::UserProfile
- AWS::OpsWorks::Volume

- AWS::Pinpoint::ADMChannel
- AWS::Pinpoint::APNSChannel
- AWS::Pinpoint::APNSSandboxChannel
- AWS::Pinpoint::APNSVoipChannel
- AWS::Pinpoint::APNSVoipSandboxChannel
- AWS::Pinpoint::App
- AWS::Pinpoint::ApplicationSettings
- AWS::Pinpoint::BaiduChannel
- AWS::Pinpoint::Campaign
- AWS::Pinpoint::EmailChannel
- AWS::Pinpoint::EmailTemplate
- AWS::Pinpoint::EventStream
- AWS::Pinpoint::GCMChannel
- AWS::Pinpoint::PushTemplate
- AWS::Pinpoint::Segment
- AWS::Pinpoint::SMSChannel
- AWS::Pinpoint::SmsTemplate
- AWS::Pinpoint::VoiceChannel
- AWS::PinpointEmail::ConfigurationSet
- AWS::PinpointEmail::ConfigurationSetEventDestination
- AWS::PinpointEmail::DedicatedIpPool
- AWS::PinpointEmail::Identity
- AWS::QLDB::Ledger
- AWS::RDS::DBSecurityGroup
- AWS::RDS::DBSecurityGroupIngress
- AWS::Redshift::ClusterSecurityGroup
- AWS::Redshift::ClusterSecurityGroupIngress
- AWS::Route53::RecordSet
- AWS::Route53::RecordSetGroup
- AWS::SageMaker::CodeRepository

- AWS::SageMaker::EndpointConfig
- AWS::SageMaker::Model
- AWS::SageMaker::NotebookInstance
- AWS::SageMaker::NotebookInstanceLifecycleConfig
- AWS::SageMaker::Workteam
- AWS::SDB::Domain
- AWS::ServiceCatalog::AcceptedPortfolioShare
- AWS::ServiceCatalog::LaunchRoleConstraint
- AWS::ServiceCatalog::Portfolio
- AWS::ServiceCatalog::StackSetConstraint
- AWS::ServiceDiscovery::HttpNamespace
- AWS::ServiceDiscovery::Instance
- AWS::ServiceDiscovery::PrivateDnsNamespace
- AWS::ServiceDiscovery::PublicDnsNamespace
- AWS::ServiceDiscovery::Service
- AWS::SES::ReceiptFilter
- AWS::SES::ReceiptRule
- AWS::SES::ReceiptRuleSet
- AWS::SSM::MaintenanceWindow
- AWS::SSM::MaintenanceWindowTarget
- AWS::SSM::MaintenanceWindowTask
- AWS::WAF::ByteMatchSet
- AWS::WAF::IPSet
- AWS::WAF::Rule
- AWS::WAF::SizeConstraintSet
- AWS::WAF::SqlInjectionMatchSet
- AWS::WAF::WebACL
- AWS::WAF::XssMatchSet
- AWS::WAFRegional::ByteMatchSet
- AWS::WAFRegional::GeoMatchSet

- AWS::WAFRegional::IPSet
- AWS::WAFRegional::RateBasedRule
- AWS::WAFRegional::RegexPatternSet
- AWS::WAFRegional::Rule
- AWS::WAFRegional::SizeConstraintSet
- AWS::WAFRegional::SqlInjectionMatchSet
- AWS::WAFRegional::WebACL
- AWS::WAFRegional::WebACLAssociation
- AWS::WAFRegional::XssMatchSet
- AWS::WorkSpaces::Workspace
- AWS::AmazonMQ::ConfigurationAssociation
- AWS::ApiGateway::DomainNameAccessAssociation
- AWS::AppConfig::ExtensionAssociation
- AWS::AppStream::ApplicationEntitlementAssociation
- AWS::AppStream::ApplicationFleetAssociation
- AWS::AppSync::DomainNameApiAssociation
- AWS::AppSync::SourceApiAssociation
- AWS::CleanRooms::ConfiguredTableAssociation
- AWS::CleanRooms::IdNamespaceAssociation
- AWS::CodeGuruReviewer::RepositoryAssociation
- AWS::Cognito::IdentityPoolRoleAttachment
- AWS::Cognito::UserPoolRiskConfigurationAttachment
- AWS::Cognito::UserPoolUICustomizationAttachment
- AWS::Cognito::UserPoolUserToGroupAttachment
- AWS::Connect::IntegrationAssociation
- AWS::Deadline::QueueFleetAssociation
- AWS::Deadline::QueueLimitAssociation
- AWS::EC2::EIPAssociation
- AWS::EC2::EnclaveCertificateIamRoleAssociation
- AWS::EC2::GatewayRouteTableAssociation

- AWS::EC2::IPAMResourceDiscoveryAssociation
- AWS::EC2::IpPoolRouteTableAssociation
- AWS::EC2::LocalGatewayRouteTableVPCAssociation
- AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation
- AWS::EC2::NetworkInterfaceAttachment
- AWS::EC2::RouteServerAssociation
- AWS::EC2::SecurityGroupVpcAssociation
- AWS::EC2::SubnetNetworkAclAssociation
- AWS::EC2::SubnetRouteTableAssociation
- AWS::EC2::TransitGatewayAttachment
- AWS::EC2::TransitGatewayMulticastDomainAssociation
- AWS::EC2::TransitGatewayPeeringAttachment
- AWS::EC2::TransitGatewayRouteTableAssociation
- AWS::EC2::TransitGatewayVpcAttachment
- AWS::EC2::VPCDHCPOptionsAssociation
- AWS::EC2::VPCGatewayAttachment
- AWS::EC2::VolumeAttachment
- AWS::ECS::ClusterCapacityProviderAssociations
- AWS::EKS::PodIdentityAssociation
- AWS::FSx::DataRepositoryAssociation
- AWS::FSx::S3AccessPointAttachment
- AWS::GlobalAccelerator::CrossAccountAttachment
- AWS::LakeFormation::TagAssociation
- AWS::NetworkFirewall::VpcEndpointAssociation
- AWS::NetworkManager::ConnectAttachment
- AWS::NetworkManager::CustomerGatewayAssociation
- AWS::NetworkManager::DirectConnectGatewayAttachment
- AWS::NetworkManager::LinkAssociation
- AWS::NetworkManager::SiteToSiteVpnAttachment
- AWS::NetworkManager::TransitGatewayRouteTableAttachment

- AWS::NetworkManager::VpcAttachment
- AWS::Notifications::ChannelAssociation
- AWS::Notifications::ManagedNotificationAccountContactAssociation
- AWS::Notifications::ManagedNotificationAdditionalChannelAssociation
- AWS::Notifications::OrganizationalUnitAssociation
- AWS::ResourceExplorer2::DefaultViewAssociation
- AWS::Route53Profiles::ProfileAssociation
- AWS::Route53Profiles::ProfileResourceAssociation
- AWS::Route53Resolver::FirewallRuleGroupAssociation
- AWS::Route53Resolver::ResolverQueryLoggingConfigAssociation
- AWS::Route53Resolver::ResolverRuleAssociation
- AWS::SSM::Association
- AWS::SecretsManager::SecretTargetAttachment
- AWS::SecurityHub::PolicyAssociation
- AWS::ServiceCatalog::PortfolioPrincipalAssociation
- AWS::ServiceCatalog::PortfolioProductAssociation
- AWS::ServiceCatalog::ServiceActionAssociation
- AWS::ServiceCatalog::TagOptionAssociation
- AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation
- AWS::ServiceCatalogAppRegistry::ResourceAssociation
- AWS::VpcLattice::ServiceNetworkResourceAssociation
- AWS::VpcLattice::ServiceNetworkServiceAssociation
- AWS::VpcLattice::ServiceNetworkVpcAssociation
- AWS::WAFv2::WebACLAssociation
- AWS::Wisdom::AssistantAssociation
- AWS::WorkspacesInstances::VolumeAssociation
- AWS::IAM::Policy
- AWS::SNS::TopicPolicy
- AWS::SQS::QueuePolicy
- AWS::EC2::NetworkAclEntry

- AWS::EC2::VPNGatewayRoutePropagation
- AWS::CloudFormation::Stack
- AWS::CloudWatch::MetricStream
- AWS::WorkSpaces::ConnectionAlias
- AWS::IoT::ProvisioningTemplate
- AWS::MediaPackage::Channel
- AWS::CloudFront::OriginRequestPolicy
- AWS::Route53Resolver::ResolverQueryLoggingConfig
- AWS::NetworkManager::TransitGatewayRegistration
- AWS::ImageBuilder::Image
- AWS::Config::ConformancePack
- AWS::S3::AccessPoint
- AWS::CodeStarConnections::Connection
- AWS::CloudFront::CachePolicy
- AWS::FMS::NotificationChannel
- AWS::ImageBuilder::InfrastructureConfiguration
- AWS::Detective::Graph
- AWS::EC2::CarrierGateway
- AWS::CloudWatch::CompositeAlarm
- AWS::CodeArtifact::Repository
- AWS::GroundStation::DataflowEndpointGroup
- AWS::ElasticLoadBalancingV2::Listener
- AWS::ImageBuilder::ImageRecipe
- AWS::NetworkManager::Device
- AWS::Kendra::DataSource
- AWS::Timestream::Database
- AWS::CodeGuruProfiler::ProfilingGroup
- AWS::Lambda::EventSourceMapping
- AWS::ECR::Repository
- AWS::WAFv2::IPSet

- AWS::GameLift::Alias
- AWS::IoTSiteWise::Asset
- AWS::OpsWorksCM::Server
- AWS::IoT::Authorizer
- AWS::WAFv2::RuleGroup
- AWS::NetworkManager::Site
- AWS::ResourceGroups::Group
- AWS::MediaPackage::PackagingConfiguration
- AWS::ImageBuilder::ImagePipeline
- AWS::ECS::TaskDefinition
- AWS::Macie::CustomDataIdentifier
- AWS::MediaPackage::OriginEndpoint
- AWS::Logs::LogGroup
- AWS::CodeArtifact::Domain
- AWS::Kendra::Faq
- AWS::ECS::TaskSet
- AWS::WAFv2::RegexPatternSet
- AWS::ECS::Cluster
- AWS::SSO::Assignment
- AWS::GlobalAccelerator::Listener
- AWS::ServiceCatalog::CloudFormationProvisionedProduct
- AWS::RDS::DBProxy
- AWS::EC2::FlowLog
- AWS::ImageBuilder::Component
- AWS::CloudFront::RealtimeLogConfig
- AWS::NetworkManager::GlobalNetwork
- AWS::RDS::DBProxyTargetGroup
- AWS::WAFv2::WebACL
- AWS::IVS::StreamKey
- AWS::IVS::PlaybackKeyPair

- AWS::Macie::Session
- AWS::Route53::HealthCheck
- AWS::Synthetics::Canary
- AWS::Lambda::CodeSigningConfig
- AWS::EFS::AccessPoint
- AWS::Timestream::Table
- AWS::MediaPackage::PackagingGroup
- AWS::ECS::PrimaryTaskSet
- AWS::Config::ConfigurationAggregator
- AWS::GroundStation::Config
- AWS::IoTSiteWise::AssetModel
- AWS::SES::ConfigurationSet
- AWS::ImageBuilder::DistributionConfiguration
- AWS::Config::OrganizationConformancePack
- AWS::EC2::LocalGatewayRoute
- AWS::KMS::Key
- AWS::Detective::MemberInvitation
- AWS::EKS::FargateProfile
- AWS::MediaPackage::Asset
- AWS::GlobalAccelerator::EndpointGroup
- AWS::Macie::FindingsFilter
- AWS::IoT::Certificate
- AWS::SageMaker::MonitoringSchedule
- AWS::IVS::Channel
- AWS::Kendra::Index
- AWS::EventSchemas::RegistryPolicy
- AWS::KinesisFirehose::DeliveryStream
- AWS::GlobalAccelerator::Accelerator
- AWS::EC2::PrefixList

- AWS::GameLift::GameServerGroup
- AWS::NetworkManager::Link
- AWS::EFS::FileSystem
- AWS::Route53::HostedZone
- AWS::GroundStation::MissionProfile
- AWS::KMS::Alias
- AWS::FMS::Policy
- AWS::SSO::PermissionSet
- AWS::StepFunctions::StateMachine
- AWS::QLDB::Stream
- AWS::IoTSiteWise::Gateway
- AWS::ECS::Service
- AWS::ECS::CapacityProvider
- AWS::EC2::SecurityGroup
- AWS::EC2::SecurityGroupIngress
- AWS::EC2::SecurityGroupEgress
- AWS::EC2::EC2Fleet
- AWS::IAM::Group
- AWS::IAM::Role
- AWS::IAM::User
- AWS::ApiGateway::GatewayResponse
- AWS::S3::BucketPolicy
- AWS::SNS::Topic
- AWS::SNS::Subscription
- AWS::RDS::DBInstance
- AWS::RDS::DBParameterGroup
- AWS::RDS::DBCluster
- AWS::RDS::DBClusterParameterGroup
- AWS::RDS::DBSubnetGroup
- AWS::RDS::EventSubscription

- AWS::RDS::GlobalCluster
- AWS::RDS::OptionGroup
- AWS::Neptune::DBInstance
- AWS::Neptune::DBParameterGroup
- AWS::Neptune::DBCluster
- AWS::Neptune::DBClusterParameterGroup
- AWS::Neptune::DBSubnetGroup
- AWS::Redshift::Cluster
- AWS::Redshift::ClusterParameterGroup
- AWS::Redshift::ClusterSubnetGroup
- AWS::Redshift::EndpointAccess
- AWS::Redshift::EndpointAuthorization
- AWS::Redshift::EventSubscription
- AWS::Redshift::ScheduledAction
- AWS::ElastiCache::SubnetGroup
- AWS::ApiGateway::BasePathMapping
- AWS::ApiGatewayV2::DomainName
- AWS::ApplicationAutoScaling::ScalableTarget
- AWS::Connect::ApprovedOrigin
- AWS::EKS::Cluster
- AWS::IAM::InstanceProfile
- AWS::Lambda::EventInvokeConfig

直接更新堆栈

在您需要将更新快速部署到堆栈时，请执行直接更新。利用直接更新功能，您可以提交指定要对堆栈资源执行的更新的模板或输入参数，然后 CloudFormation 会立即部署更新。如果您希望使用模板来进行更新，则可修改当前模板并将其存储在本地或存储在 Amazon S3 存储桶中。

对于不支持更新的资源属性，您必须保留当前值。要在更新之前预览 CloudFormation 将对堆栈进行的更改，请使用更改集。有关更多信息，请参阅 使用更改集更新 CloudFormation 堆栈。

更新堆栈时，CloudFormation 可能会中断资源或者替换更新的资源，具体取决于您所更新的属性。有关资源更新行为的更多信息，请参阅理解堆栈资源的更新行为。

部署前验证

当您执行直接堆栈更新时，CloudFormation 会在修改任何资源之前，根据常见故障场景自动对您的模板进行验证。如果验证过程中检测到错误，更新将停止，且不会对任何资源进行更改。

要在控制台中查看验证结果，请打开堆栈的事件选项卡并选择操作 ID。然后选择部署验证选项卡。若要通过 Amazon CLI 查看结果，请使用 describe-events 命令。要跳过验证，请使用 Amazon CLI 中的 `--disable-validation` 标志，或在 API 调用中将 `DisableValidation` 设置为 `true`。

更新 堆栈 (控制台)

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 在堆栈页面上，选择要更新的正在运行的堆栈。
4. 在堆栈详细信息窗格中，选择 Update (更新)。
5. 如果您尚未修改堆栈模板，请选择使用当前模板，然后选择下一步。

   如果您已修改模板，请选择替换当前模板，然后在指定模板部分指定更新模板的位置：

   - 对于在计算机本地存储的模板，选择 Upload a template file (上传模板文件)。选择 Choose File (选择文件) 以导航到此文件并选中它，然后选择 Next (下一步)。

     > **Note**
     >
     > 如果您上传了一个本地模板文件，则 CloudFormation 会将其上传至您的 Amazon Web Services 账户中的 Amazon Simple Storage Service（Amazon S3）存储桶。如果您还没有使用 CloudFormation 创建的 S3 存储桶，它会为每个区域创建一个唯一的存储桶，您可以向其上传模板文件。如果您的 Amazon Web Services 账户中已有使用 CloudFormation 创建的 S3 存储桶，则 CloudFormation 会将该模板添加到该存储桶。要记住的有关 CloudFormation 所创建的 S3 存储桶的注意事项
     >
     > - 具有您的 Amazon Web Services 账户中的 Amazon S3 权限的任何人均可访问这些存储桶。
     > - CloudFormation 创建默认情况下已启用服务器端加密的存储桶，从而对存储桶中存储的所有对象进行加密。

例如，您可以使用 Amazon S3 控制台（ https://console.aws.amazon.com/s3/ ）或 Amazon CLI 直接管理 CloudFormation 创建的桶的加密选项。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的为 Amazon S3 存储桶设置默认服务器端加密行为。

- 您可通过手动将模板上传到 Amazon S3 来使用您自己的存储桶并管理其权限。每当您创建或更新堆栈时，请指定模板文件的 Amazon S3 URL。

- 对于存储在 Amazon S3 存储桶中的模板，选择Amazon S3 URL。输入或粘贴模板的 URL，然后选择 Next (下一步)。

  如果模板位于启用了版本控制的存储桶中，则可以通过在 URL 后附加 ?versionId=*version-id* 来指定模板的具体版本。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用启用版本控制的存储桶中的对象。

如果检测到任何语法问题，控制台会提供错误消息，这将有助更正模板。

6. 如果您的模板包含参数，则可在 Specify stack details (指定堆栈详细信息) 页上输入或修改参数值，然后选择 Next (下一步)。

   CloudFormation 会使用当前在堆栈中设置的值填充每个参数（使用 NoEcho 属性声明的参数除外）；不过您仍可通过选中使用现有值来使用当前值。

   有关使用 NoEcho 来遮蔽敏感信息的更多信息（除了使用动态参数管理密钥以外），请参阅 请勿将凭证嵌入您的模板 最佳实践。

7. 在 Configure stack options（配置堆栈选项）页面上，您可以更新应用于堆栈的标签和权限，并修改高级选项（例如堆栈策略、回滚配置），或更新 Amazon SNS 通知主题。有关这些选项的详细信息，请参阅配置堆栈选项。

8. 如果模板包含 IAM 资源，则对于功能，请选择我确认该模板可能会创建 IAM 资源以指定您要在模板中使用 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。

9. 选择下一步以继续。

10. 审查提交的堆栈信息和更改。

    检查您是否已提交正确的信息，例如正确的参数值或模板 URL。

    在更改集预览部分中，检查 CloudFormation 是否将执行所有预期的更改。例如，您可以检查 CloudFormation 是否添加、移除和修改了您要添加、移除或修改的资源。CloudFormation 通过为堆栈创建更改集来生成此预览。有关更多信息，请参阅 使用更改集更新 CloudFormation 堆栈。

11. 如果您对所做更改表示满意，请选择 Updata stack（更新堆栈）。

> **Note**
>
> 此时，您还可以选择查看更改集以更全面地查看建议的更新。要执行此操作，请选择 View change set（查看更改集）而不是 Update stack（更新堆栈）。CloudFormation 会显示根据您的更新生成的更改集。准备好执行堆栈更新后，请选择 Execute (执行)。

CloudFormation 会显示堆栈的堆栈详细信息页面，其中已选中事件窗格。堆栈现在的状态为 `UPDATE_IN_PROGRESS`。CloudFormation 成功完成堆栈更新后，会将堆栈状态设置为 `UPDATE_COMPLETE`。

如果堆栈更新失败，CloudFormation 会自动回滚更改，并将状态设置为 `UPDATE_ROLLBACK_COMPLETE`。

> **Note**
>
> 当更新处于 `UPDATE_IN_PROGRESS` 状态时，则可以取消更新。有关更多信息，请参阅取消堆栈更新。

使用命令行更新堆栈

您可以使用以下任一命令：

- update-stack (Amazon CLI)
- Update-CFNStack (Amazon Tools for Windows PowerShell)

有关使用命令行更新堆栈的示例，请参阅 Amazon CLI 和 PowerShell 的 CloudFormation 堆栈操作命令示例。

Express Mode

要使用 Express Mode 来更新堆栈，请添加 `--deployment-config` 参数。Express Mode 会在资源配置应用后立即完成堆栈操作。有关更多信息，请参阅 Express Mode。

```
aws cloudformation update-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
```

```
    --deployment-config '{"mode": "EXPRESS"}'
```

取消堆栈更新

堆栈更新开始后，如果堆栈仍处于 UPDATE_IN_PROGRESS 状态，则您可以取消堆栈更新。更新完成之后，将无法取消它。但是，您可以通过任何先前的设置再次更新堆栈。

如果您取消堆栈更新，那么堆栈将回滚到启动堆栈更新之前已存在的堆栈配置。

主题

- 取消堆栈更新（控制台）
- 取消堆栈更新（Amazon CLI）

取消堆栈更新（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择堆栈所在的 Amazon Web Services 区域。
3. 在堆栈页面上，选择当前正在更新的堆栈。其状态必须为 UPDATE_IN_PROGRESS。
4. 选择 Stack actions (堆栈操作)，然后选择 Cancel update stack (取消更新堆栈)。
5. 要继续取消更新，请选择 Cancel update (取消更新)。否则，选择 Cancel (取消) 以恢复更新。

堆栈进入 UPDATE_ROLLBACK_IN_PROGRESS 状态。更新取消完成后，堆栈状态设置为 UPDATE_ROLLBACK_COMPLETE。

取消堆栈更新（Amazon CLI）

使用命令 cancel-update-stack 可取消更新。有关更多信息，请参阅 取消堆栈更新。

通过 CloudFormation 控制台删除堆栈

如果您不再需要某个堆栈中的资源，则可以删除整个堆栈。

删除堆栈时，除非您使用删除策略来保留特定的资源，否则 CloudFormation 会删除该堆栈中的所有资源。有关更多信息，请参阅 DeletionPolicy 属性。

删除堆栈（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择堆栈所在的 Amazon Web Services 区域。
3. 在堆栈页面上，选择您要删除的堆栈。该堆栈当前必须处于运行状态。
4. 选择 Delete。
5. 当系统提示进行确认时，选择 Delete（删除）。

> **Note**
>
> 堆栈删除开始后，就无法停止堆栈删除操作。堆栈进入 DELETE_IN_PROGRESS 状态。堆栈删除过程完成之后，堆栈将处于 DELETE_COMPLETE 状态。在默认情况下，DELETE_COMPLETE 状态的堆栈不会显示在 CloudFormation 控制台上。要显示已删除的堆栈，您必须按照 通过 CloudFormation 控制台查看已删除的堆栈 中的描述更改堆栈查看筛选条件。

强制删除堆栈（控制台）

堆栈删除可能会失败，因为无法删除堆栈中的某个资源。例如，CloudFormation 将无法删除其他堆栈也依赖的资源。尚未删除的任何资源将保留，直到您能成功删除堆栈。如果删除失败并返回 DELETE_FAILED 状态，则可以选择使用以下两种方法之一重试。

1. 在 CloudFormation 控制台的堆栈页面中，选择要强制删除的堆栈。
2. 在堆栈详细信息窗格中，选择重试删除。
3. 选择以下选项：
   - 删除此堆栈但保留资源：此选项让您可以在强制删除堆栈过程中选择最初未能删除但想要保留的特定资源。
   - 强制删除整个堆栈：此选项会保留所有未能删除的资源以及这些资源的依赖项。
4. 选择删除，即可根据所选择的内容开始强制删除过程。

查看保留的资源（控制台）

删除堆栈后，可以在控制台中查看保留的资源。

1. 在堆栈列表中，选择筛选状态并选择已删除。
2. 选择此已删除的堆栈。
3. 选择资源选项卡。
4. 所有保留的资源都显示 DELETE_SKIPPED 状态。
5. 选择要查看的已保留资源。

使用命令行删除堆栈

您可以使用以下任一命令：

- delete-stack (Amazon CLI)
- Remove-CFNStack (Amazon Tools for Windows PowerShell)

有关使用命令行删除堆栈的示例，请参阅 Amazon CLI 和 PowerShell 的 CloudFormation 堆栈操作命令示例。

Express Mode

要使用 Express Mode 来删除堆栈，请在 `delete-stack` 命令中添加 `--deployment-config` 参数。Express Mode 会在 CloudFormation 针对每个资源发出删除 API 调用后立即完成删除操作，无需等待资源被完全被移除。有关更多信息，请参阅 Express Mode。

```
aws cloudformation delete-stack --stack-name myteststack \
    --deployment-config '{"mode": "EXPRESS"}'
```

> **Note**
>
> 在 CloudFormation 控制台中，删除堆栈时，您可以将部署模式选为“Express”，以便使用 Express Mode。

相关资源

有关堆栈删除错误排查的帮助，请参阅删除堆栈失败问题排查主题。

有关防止意外删除堆栈的信息，请参阅CloudFormation 堆栈的删除保护。

通过 CloudFormation 控制台查看已删除的堆栈

默认情况下，CloudFormation 控制台不会显示处于 DELETE_COMPLETE 状态的堆栈。要显示有关删除的堆栈的信息，您必须更改堆栈视图。

查看已删除的堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择删除的堆栈所在的 Amazon Web Services 区域。
3. 在堆栈页面上，从筛选状态下拉列表中选择已删除。

CloudFormation 会列出状态为 DELETE_COMPLETE 的所有堆栈。

另请参阅

- 通过 CloudFormation 控制台删除堆栈
- 从 CloudFormation 控制台查看堆栈信息

监控堆栈进度

本节介绍如何监控当前正在进行的堆栈部署。CloudFormation 提供了按时间顺序排列的详细部署事件列表，显示部署进度和部署过程中遇到的任何问题。

主题

- 查看 CloudFormation 堆栈事件
- 按操作查看堆栈事件
- 查看 CloudFormation 堆栈部署的时间线
- 了解 CloudFormation 堆栈创建事件
- 监控堆栈更新的进度
- 继续回滚更新
- 确定堆栈失败的原因
- 选择预置资源时如何处理失败

查看 CloudFormation 堆栈事件

可以查看堆栈事件来监控堆栈及堆栈中资源的进度和状态。堆栈事件有助于您了解何时创建、更新或删除资源，以及堆栈部署是否按预期进行。

主题

- 查看堆栈事件（控制台）
- 查看堆栈事件（Amazon CLI）
- 堆栈状态代码

查看堆栈事件（控制台）

查看堆栈事件

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了堆栈的 Amazon Web Services 区域。
3. 在 CloudFormation 控制台的 Stacks（堆栈）页面中，选择堆栈名称。CloudFormation 将显示选定堆栈的堆栈详细信息。
4. 选择事件选项卡以查看 CloudFormation 为堆栈生成的堆栈事件。

CloudFormation 每分钟自动刷新堆栈事件。此外，CloudFormation 会在新堆栈事件发生时显示新事件可用徽章。选择“刷新”图标可将这些事件加载到列表中。通过查看堆栈创建事件，您可以了解导致堆栈创建（或失败，如果您正在调试堆栈）的事件的顺序。

创建堆栈时，该堆栈会在堆栈页面列出，其状态为 `CREATE_IN_PROGRESS`。堆栈创建成功后，其状态将变为 `CREATE_COMPLETE`。

有关更多信息，请参阅了解 CloudFormation 堆栈创建事件和监控堆栈更新的进度。

查看堆栈事件（Amazon CLI）

您也可以在创建堆栈时使用 describe-stack-events 命令查看所报告的事件。

以下 describe-stack-events 命令会描述 *my-stack* 堆栈事件。

```
aws cloudformation describe-stack-events --stack-name my-stack
```

以下为响应示例。


```
{
    "StackEvents": [
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "7b755820-7edf-11f0-ab15-0673b09f3847",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:aws-
region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2025-08-21T22:37:56.243000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServer-CREATE_COMPLETE-2025-08-21T22:37:54.356Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServer",
            "PhysicalResourceId": "i-099df76cb31b866a9",
            "ResourceType": "AWS::EC2::Instance",
            "Timestamp": "2025-08-21T22:37:54.356000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"UserData\":
\"IyEvYmluL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYmluL2Nmbi1pbml0IC12I
\",\"ImageId\":\"ami-0bbc328167dee8f3c\",\"InstanceType\":\"t2.micro\",
\"SecurityGroupIds\":[\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\"],\"Tags\":
[{\"Value\":\"Bootstrap Tutorial Web Server\",\"Key\":\"Name\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServer-CREATE_IN_PROGRESS-2025-08-21T22:37:31.226Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServer",
            "PhysicalResourceId": "i-099df76cb31b866a9",
            "ResourceType": "AWS::EC2::Instance",
            "Timestamp": "2025-08-21T22:37:31.226000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
```

```
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"UserData\":
\"IyEvYmluL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYmluL2Nmbi1pbml0IC12I
\",\"ImageId\":\"ami-0bbc328167dee8f3c\",\"InstanceType\":\"t2.micro\",
\"SecurityGroupIds\":[\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\"],\"Tags\":
[{\"Value\":\"Bootstrap Tutorial Web Server\",\"Key\":\"Name\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServer-CREATE_IN_PROGRESS-2025-08-21T22:37:29.210Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServer",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::EC2::Instance",
            "Timestamp": "2025-08-21T22:37:29.210000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceProperties": "{\"UserData\":
\"IyEvYmluL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYmluL2Nmbi1pbml0IC12I
\",\"ImageId\":\"ami-0bbc328167dee8f3c\",\"InstanceType\":\"t2.micro\",
\"SecurityGroupIds\":[\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\"],\"Tags\":
[{\"Value\":\"Bootstrap Tutorial Web Server\",\"Key\":\"Name\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServerSecurityGroup-
CREATE_COMPLETE-2025-08-21T22:37:28.803Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServerSecurityGroup",
            "PhysicalResourceId": "my-stack-WebServerSecurityGroup-n8A43bQT1ty2",
            "ResourceType": "AWS::EC2::SecurityGroup",
            "Timestamp": "2025-08-21T22:37:28.803000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"GroupDescription\":\"Allow HTTP access from my IP
 address\",\"SecurityGroupIngress\":[{\"CidrIp\":\"0.0.0.0/0\",\"Description\":\"HTTP
\",\"FromPort\":\"80\",\"ToPort\":\"80\",\"IpProtocol\":\"tcp\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
```

```
            "EventId": "WebServerSecurityGroup-
CREATE_IN_PROGRESS-2025-08-21T22:37:22.626Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServerSecurityGroup",
            "PhysicalResourceId": "my-stack-WebServerSecurityGroup-n8A43bQT1ty2",
            "ResourceType": "AWS::EC2::SecurityGroup",
            "Timestamp": "2025-08-21T22:37:22.626000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"GroupDescription\":\"Allow HTTP access from my IP
 address\",\"SecurityGroupIngress\":[{\"CidrIp\":\"0.0.0.0/0\",\"Description\":\"HTTP
\",\"FromPort\":\"80\",\"ToPort\":\"80\",\"IpProtocol\":\"tcp\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServerSecurityGroup-
CREATE_IN_PROGRESS-2025-08-21T22:37:20.186Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServerSecurityGroup",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::EC2::SecurityGroup",
            "Timestamp": "2025-08-21T22:37:20.186000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceProperties": "{\"GroupDescription\":\"Allow HTTP access from my IP
 address\",\"SecurityGroupIngress\":[{\"CidrIp\":\"0.0.0.0/0\",\"Description\":\"HTTP
\",\"FromPort\":\"80\",\"ToPort\":\"80\",\"IpProtocol\":\"tcp\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "64740fe0-7edf-11f0-8a36-06453a64f325",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:aws-
region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2025-08-21T22:37:17.819000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "User Initiated",
            "ClientRequestToken": "token"
        }
```

```
    ]
}
```

最新发生的事件先报告。下表列出了 describe-stack-events 命令返回的字段：

字段	说明
EventId	事件标识符。
StackName	事件对应的堆栈名称。
StackId	事件对应的堆栈标识符。
LogicalResourceId	资源的逻辑标识符。
PhysicalResourceId	资源的物理标识符。
ResourceProperties	资源的属性。
ResourceType	资源类型。
Timestamp	事件发生的时间。
ResourceStatus	资源的状态，可以是以下状态代码之一：CREATE_COMPLETE | CREATE_FAILED | CREATE_IN_PROGRESS | DELETE_COMPLETE | DELETE_FAILED | DELETE_IN_PROGRESS | DELETE_SKIPPED | IMPORT_COMPLETE | IMPORT_IN_PROGRESS | IMPORT_ROLLBACK_COMPLETE | IMPORT_ROLLBACK_FAILED | IMPORT_ROLLBACK_IN_PROGRESS | REVIEW_IN_PROGRESS | ROLLBACK_COMPLETE | ROLLBACK_FAILED | ROLLBACK_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_FAILED | UPDATE_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_IN_PROGRESS DELETE_SKIPPED 状态适用于带保留的删除策略属性的资源。

字段	说明
DetailedStatus	堆栈的具体状态。如果 CONFIGURATION_COMPLETE 存在，则堆栈资源配置阶段已完成，资源的稳定化正在进行中。
ResourceStatusReason	有关状态的更多信息。

堆栈状态代码

下表说明了堆栈状态代码：

堆栈状态和可选的详细状态	说明
CREATE_COMPLETE	成功创建一个或多个堆栈。
CREATE_IN_PROGRESS	正在创建一个或多个堆栈。
CREATE_FAILED	一个或多个堆栈创建失败。查看堆栈事件可了解所有相关错误消息。创建失败的可能原因包括：没有足够的权限使用堆栈中的所有资源，参数值被 Amazon 服务拒绝，或者在资源创建期间超时。
DELETE_COMPLETE	成功删除一个或多个堆栈。已删除的堆栈在 90 天内保留可供查看。
DELETE_FAILED	一个或多个堆栈删除失败。由于删除失败，您可能有一些资源仍在运行，但是您无法使用或更新堆栈。再次删除堆栈或查看堆栈事件可了解所有相关错误消息。
DELETE_IN_PROGRESS	正在删除一个或多个堆栈。
REVIEW_IN_PROGRESS	正在创建一个或多个具有预期 StackId 但没有任何模板或资源的堆栈。 **Important** 具有此状态代码的堆栈将针对可能的最大堆栈数量进行计数。

堆栈状态和可选的详细状态	说明
ROLLBACK_COMPLETE	在堆栈创建失败或明确取消堆栈创建后成功删除一个或多个堆栈。堆栈返回到之前的工作状态。在创建堆栈操作期间创建的所有资源都被删除。 此状态仅在堆栈创建失败后存在。它表示已适当清除未完全创建的堆栈中的所有操作。在此状态下，只能执行删除操作。
ROLLBACK_FAILED	在堆栈创建失败或明确取消堆栈创建后删除一个或多个堆栈失败。删除堆栈或查看堆栈事件了解所有相关错误消息。
ROLLBACK_IN_PROGRESS	在堆栈创建失败或明确取消堆栈创建后正在删除一个或多个堆栈。
UPDATE_COMPLETE	成功更新一个或多个堆栈。
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	成功更新一个或多个堆栈后正在删除堆栈的旧资源。对于需要替换资源的堆栈更新，CloudFormation 首先创建新资源，然后删除旧资源，以帮助减少堆栈中断。在这种状态下，堆栈已更新并且可用，但 CloudFormation 仍然会删除旧资源。
UPDATE_FAILED	未成功更新一个或多个堆栈。查看堆栈事件可了解所有相关错误消息。
UPDATE_IN_PROGRESS	正在更新一个或多个堆栈。
UPDATE_ROLLBACK_COMPLETE	在堆栈更新失败后将一个或多个堆栈成功返回之前的工作状态。
UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS	堆栈更新失败后正在删除一个或多个堆栈的新资源。在这种状态下，堆栈已回滚到之前的工作状态并且可用，但 CloudFormation 仍会删除它在堆栈更新期间创建的所有新资源。
UPDATE_ROLLBACK_FAILED	在堆栈更新失败后将一个或多个堆栈返回之前的工作状态失败。在这种状态下，您可以删除堆栈或继续回滚。您可能需要修复错误，然后堆栈才能返回工作状态。您也可以联系 Amazon Web Services 支持 将堆栈恢复到可用状态。

堆栈状态和可选的详细状态	说明
UPDATE_ROLLBACK_IN_PROGRESS	堆栈更新失败后正在将一个或多个堆栈返回之前的工作状态。
IMPORT_IN_PROGRESS	当前正在执行导入操作。
IMPORT_COMPLETE	对于堆栈中支持 resource import 的所有资源，已成功完成导入操作。
IMPORT_ROLLBACK_IN_PROGRESS	导入将回滚到以前的模板配置。
IMPORT_ROLLBACK_FAILED	对于堆栈中的至少一个资源，导入回滚操作失败。将提供 CloudFormation 成功导入的资源的结果。
IMPORT_ROLLBACK_COMPLETE	导入成功回滚到以前的模板配置。

按操作查看堆栈事件

您可以查看按操作分组的堆栈事件，以更好地了解对您的堆栈所做更改的顺序和范围。基于操作的分组可帮助您一起跟踪相关事件，从而更轻松地监控进度并在问题发生时对其进行故障排查。

每个堆栈操作（创建、更新、删除、回滚）都会被分配一个唯一的操作 ID，用于对所有相关事件进行分组。这使您能够专注于特定的操作，并迅速找出故障的根本原因。

主题

- 先决条件
- 按操作查看堆栈事件（控制台）
- 按操作查看堆栈事件（Amazon CLI）
- 堆栈状态代码

先决条件

要使用 DescribeEvents API，您必须具有必要 IAM 权限：DescribeEvents。

按操作查看堆栈事件（控制台）

要查看按操作分组的堆栈事件

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了堆栈的 Amazon Web Services 区域。
3. 在堆栈页面上，选择堆栈名称。CloudFormation 将显示选定堆栈的堆栈详细信息。
4. 选择事件选项卡以查看 CloudFormation 为堆栈生成的堆栈事件。
5. 事件会按照操作 ID 自动进行分组。每个操作都会以可展开的部分呈现，该部分会显示操作类型、状态以及时间戳。
6. 单击操作 ID 可打开详细视图，该视图仅显示与该特定操作相关的事件。
7. 在操作详细信息视图中，选中仅显示故障复选框，以仅显示用于根本原因分析的失败事件。

CloudFormation 每分钟自动刷新堆栈事件。新堆栈事件发生时显示新堆栈事件可用徽章。选择“刷新”图标可将这些事件加载到列表中。

通过查看按操作分组的堆栈事件，您可以了解每个操作的事件顺序，并快速确定哪个特定操作导致了问题（如果您正在调试堆栈）。

当您的堆栈操作正在运行时，其会被列出 `CREATE_IN_PROGRESS`、`UPDATE_IN_PROGRESS` 或 `DELETE_IN_PROGRESS` 状态。成功完成操作后，其状态将更改为 `CREATE_COMPLETE`、`UPDATE_COMPLETE` 或 `DELETE_COMPLETE`。

有关更多信息，请参阅了解 CloudFormation 堆栈创建事件和监控堆栈更新的进度。

按操作查看堆栈事件（Amazon CLI）

您可以使用带有操作 ID 筛选功能的 `describe-events` 命令，以查看特定操作的事件。

获取上次操作 ID

通过 describe-stacks API 提供的堆栈描述现在包括显示最近的操作 ID 及其类型的 LastOperations 信息。这使得您能够迅速确定哪些操作已经完成以及它们的当前状态，而无需通过事件日志进行解析。

```
aws cloudformation describe-stacks --stack-name MyStack
```

以下是响应示例，显示最后一个操作是更新操作失败后的回滚。

```
{
    "Stacks": [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/07580010-bb79-11f0-8f6c-0289bb5c804f",
            "StackName": "MyStack",
            "Description": "A simple CloudFormation template to create an S3 bucket.",

            "CreationTime": "2025-11-07T01:28:13.778000+00:00",
            "LastUpdatedTime": "2025-11-07T01:43:39.838000+00:00",
            "RollbackConfiguration": {},
            "StackStatus": "UPDATE_ROLLBACK_COMPLETE",
            "DisableRollback": false,
            "NotificationARNs": [],
            "Tags": [],
            "EnableTerminationProtection": false,
            "DriftInformation": {
                "StackDriftStatus": "NOT_CHECKED"
            },
            "LastOperations": [
                {
                    "OperationType": "ROLLBACK",
                    "OperationId": "d0f12313-7bdb-414d-a879-828a99b36f29"
                },
                {
                    "OperationType": "UPDATE_STACK",
                    "OperationId": "1c211b5a-4538-4dc9-bfed-e07734371e57"
                }
            ]
        }
    ]
}
```

按操作 ID 筛选事件

下面的 `describe-events` 命令描述了特定操作 ID 的事件：

```
aws cloudformation describe-events \
  --operation-id 1c211b5a-4538-4dc9-bfed-e07734371e57
```

要仅查看失败事件以进行故障排除，请使用 `--filter FailedEvents=true` 参数：

```
aws cloudformation describe-events \
```

```
  --operation-id 1c211b5a-4538-4dc9-bfed-e07734371e57 \
  --filter FailedEvents=true
```

新的操作 ID 筛选功能使您能够专注于特定操作及其相关事件。此功能特别适用于：

- 对特定故障进行故障排除：将事件与失败的操作隔离开来，以了解出了什么问题。
- 监控长期运行的操作：跟踪复杂的更新或大型堆栈部署的进度。
- 审计更改：审核与特定更新操作相关的所有事件。
- 根本原因分析：使用故障筛选器快速确定部署问题的根源。

堆栈状态代码

下表列出了 `describe-events` 命令在使用操作 ID 筛选时返回的字段：

字段	说明
EventId	事件标识符。
OperationId	生成此事件的操作的唯一标识符。
StackName	事件对应的堆栈名称。
StackId	事件对应的堆栈标识符。
LogicalResourceId	资源的逻辑标识符。
PhysicalResourceId	资源的物理标识符。
ResourceProperties	资源的属性。
ResourceType	资源类型。
Timestamp	事件发生的时间。
ResourceStatus	资源的状态（CREATE_COMPLETE 、UPDATE_FAILED 等）。

字段	说明
DetailedStatus	堆栈的具体状态。如果 CONFIGURATION_COMPLETE 存在，则堆栈资源配置阶段已完成，资源的稳定化正在进行中。
ResourceStatusReason	有关状态的更多信息。

查看 CloudFormation 堆栈部署的时间线

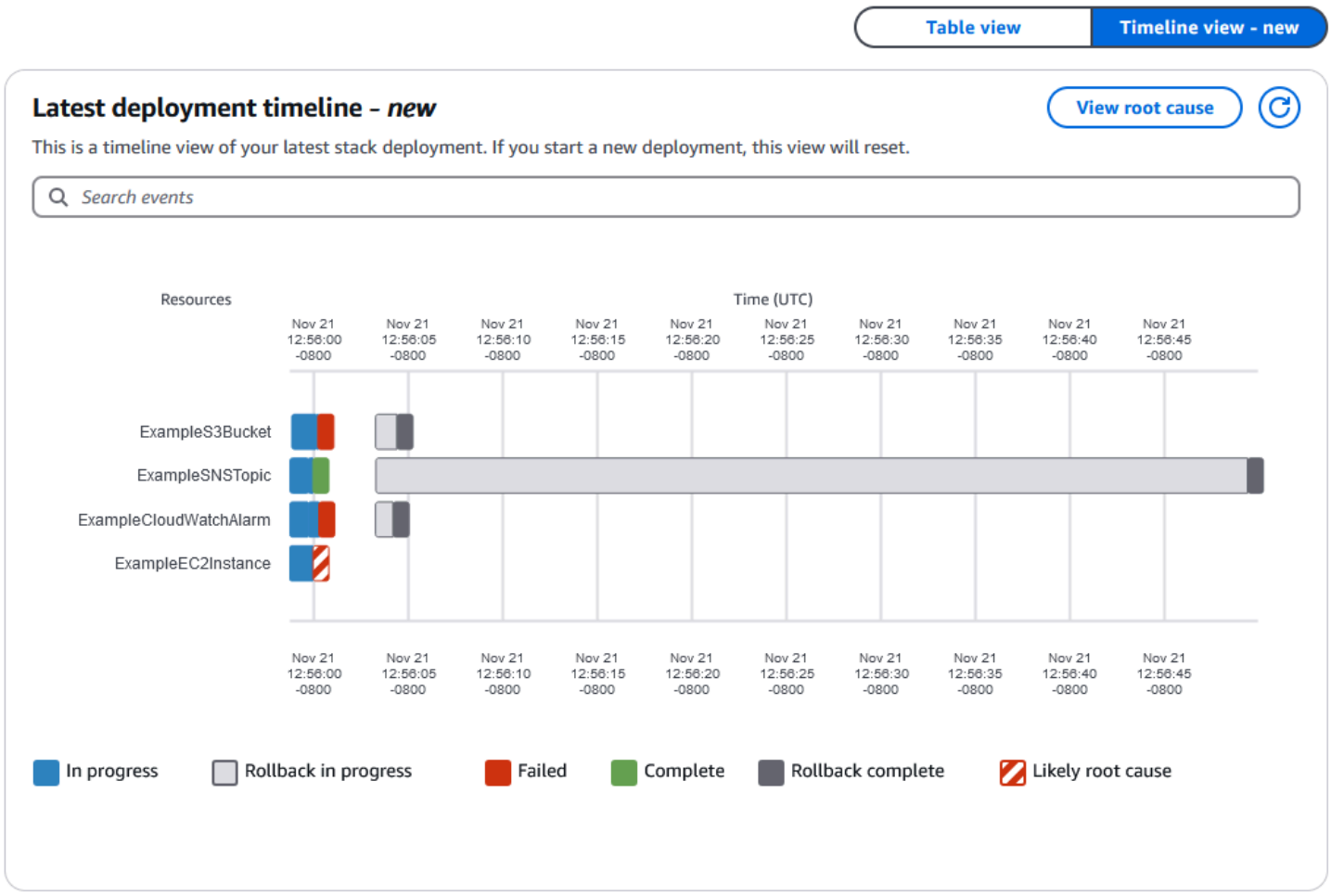
堆栈部署时间线图以可视化形式表示堆栈部署时间线。该图显示堆栈及其各个资源的部署状态，以及每种状态的更改时间。堆栈部署状态用相应颜色表示。

主题

- 了解堆栈部署时间线图
- 查看堆栈部署时间线图（控制台）

了解堆栈部署时间线图

下图显示由于 Amazon EC2 实例资源启动失败而未能进行堆栈部署的时间线图。

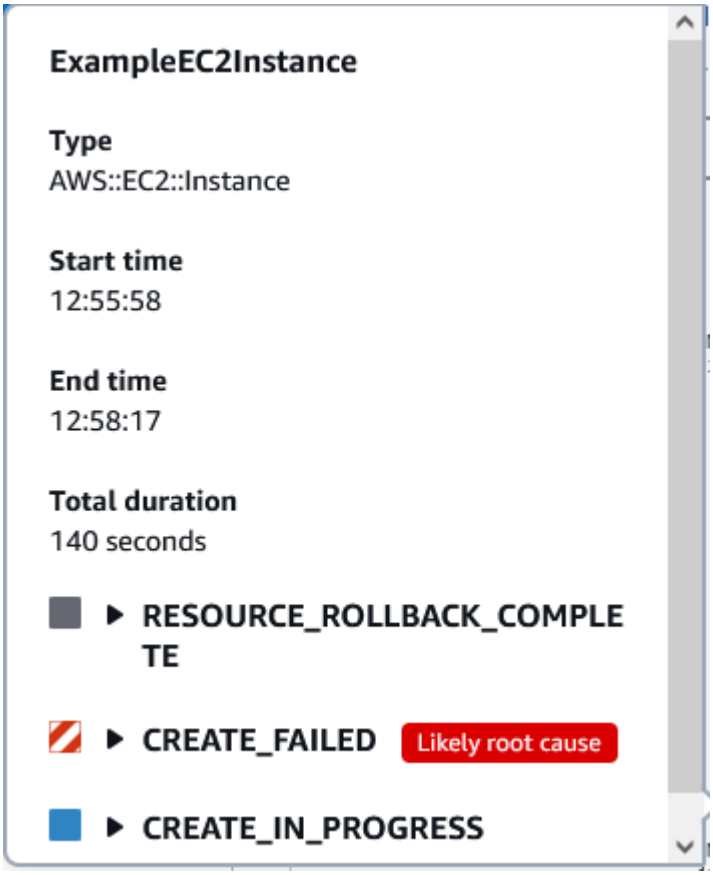


堆栈资源的名称位于图中左侧，相对于部署时间的日期和时间位于图中顶部。

各个资源均以“进行中”的状态开头。每次成功部署后，状态栏都会更改为完成。资源部署失败时，状态栏更改为失败。当资源部署失败且堆栈部署也失败时，导致堆栈部署失败的资源将收到可能为根失败的状态。

堆栈部署操作失败后，成功部署的资源开始回滚并更改为回滚中的状态。资源完成回滚后，状态更改为回滚完成。

选中各个资源可以更精细地显示有关部署时间线的详细信息：



选中资源会显示部署的类型、部署开始时间、部署结束时间和总持续时间。此外还可以在下面的下拉菜单中，找到每种部署状态的开始时间、结束时间和持续时间。如果资源部署失败，将提供失败原因。

有关堆栈状态的更多信息，请参阅堆栈状态代码。

查看堆栈部署时间线图（控制台）

查看堆栈部署时间线图：

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了堆栈的 Amazon Web Services 区域。
3. 在 CloudFormation 控制台的堆栈页面中，选择堆栈名称。CloudFormation 将显示选定堆栈的堆栈详细信息。
4. 选择事件选项卡以查看 CloudFormation 为堆栈生成的堆栈事件。
5. 选择时间线图按钮查看堆栈的时间线图。

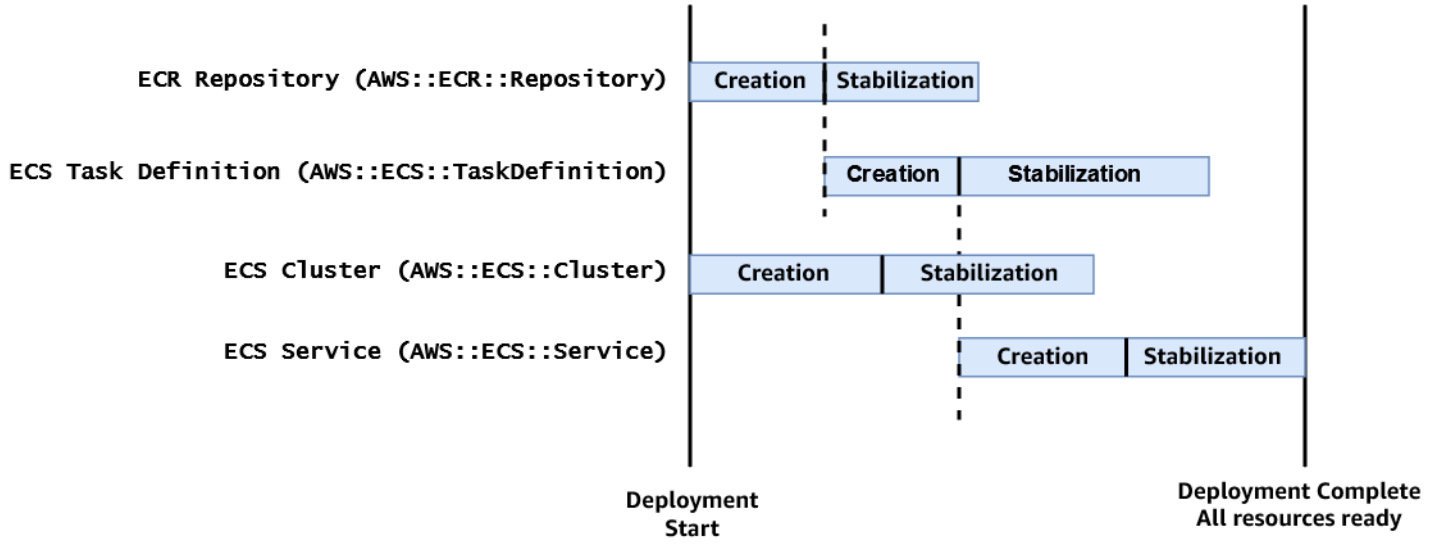
了解 CloudFormation 堆栈创建事件

在堆栈部署期间，将会发生多个事件来创建、配置和验证堆栈模板中定义的资源。了解这些事件有助您优化堆栈创建过程并简化部署。

- 资源创建事件 – 当每个资源的创建过程启动时，都会设置一个 CREATE_IN_PROGRESS 事件的状态。该事件指示正在预置该资源。
- 最终一致性检查 – 在堆栈创建过程中，很大一部分时间都用于对堆栈所创建资源执行最终一致性检查。在此阶段，服务会执行内部一致性检查，以确保资源完全正常运行并符合各项 Amazon Web Services 服务所定义的服务稳定性标准。
- 配置完成事件 – 当每个资源完成预置的最终一致性检查阶段后，将会设置 CONFIGURATION_COMPLETE 事件的详细状态。
- 资源创建完成事件 – 按规定创建和配置好资源，并且配置与模板中指定的内容一致后，将会设置 CREATE_COMPLETE 事件的状态。

在不需要资源最终一致性检查的情况下，例如验证预生产堆栈配置或跨堆栈预调配，您可以利用 CONFIGURATION_COMPLETE 事件来简化堆栈创建过程。您可以通过多种方式来使用此事件。例如，您可以将其用作视觉信号，以跳过等待资源的过程或堆栈一致性检查完成的过程。或者，您可以使用持续集成和持续交付（CI/CD）来创建自动机制，以触发其他操作。

> **Important**
>
> 虽然利用 CONFIGURATION_COMPLETE 事件可以加快堆栈创建时间，但您也应该注意其利弊。首先，只有支持偏差检测的少部分资源类型才支持该功能。有关支持偏差检测的资源类型的列表，请参阅 资源类型支持。这种方法可能不适用于所有场景，尤其是资源需要彻底的最终一致性检查以确保整个云环境（例如，在生产环境）完全准备就绪的情况。建议您仔细评测您的部署要求，以及每种资源的一致性检查的重要程度。利用 CONFIGURATION_COMPLETE 事件来优化部署速度，同时不影响基础设施的完整性和可靠性。有关如何充分利用此事件并了解其对特定使用案例的影响的指导，请参阅详细的博客 How we sped up CloudFormation deployments with optimistic stabilization。
>
> 由于不能保证会设置 CONFIGURATION_COMPLETE 事件，因此任何使用该事件的场景都应准备好在未设置任何 CONFIGURATION_COMPLETE 事件时处理 CREATE_COMPLETE 事件。



堆栈部署开始时，AWS::ECR::Repository 和 AWS::ECS::Cluster 资源都会启动创建过程（ResourceStatus = CREATE_IN_PROGRESS）。当 AWS::ECR::Repository 资源类型开始最终一致性检查（DetailedStatus = CONFIGURATION_COMPLETE）后，AWS::ECS::TaskDefinition 资源就可以开始创建过程了。同样，一旦 AWS::ECS::TaskDefinition 资源开始最终一致性检查，AWS::ECS::Service 资源就会开始创建过程。

CREATE_IN_PROGRESS 和 CREATE_COMPLETE 事件

- [堆栈]：CREATE_IN_PROGRESS
- [资源]：ECR 存储库CREATE_IN_PROGRESS
- [资源]：ECS 集群CREATE_IN_PROGRESS
- [资源]：ECR 存储库 CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [资源]：ECS 任务定义 CREATE_IN_PROGRESS
- [资源]：ECS 集群 CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [资源]：ECS 任务定义 CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [资源]：ECS 服务 CREATE_IN_PROGRESS
- [资源]：ECR 存储库CREATE_COMPLETE
- [资源]：ECS 集群CREATE_COMPLETE
- [资源]：ECS 服务 CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [堆栈]：CREATE_IN_PROGRESS、CONFIGURATION_COMPLETE
- [资源]：ECS 任务定义 CREATE_COMPLETE

- [资源]：ECS 服务 CREATE_COMPLETE
- [堆栈]：CREATE_COMPLETE

监控堆栈更新的进度

您可以通过查看堆栈事件来监控堆栈更新进度。堆栈的事件选项卡会显示堆栈创建和更新过程中的每个重要步骤（按照每个事件的时间排序排列，最新的事件显示在最上方）。有关更多信息，请参阅 监控堆栈进度。

主题

- 成功更新堆栈期间生成的事件
- 资源更新失败时生成的事件

成功更新堆栈期间生成的事件

堆栈更新过程开始时会为堆栈标记一个 UPDATE_IN_PROGRESS 事件：

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_IN_PROGRESS
```

接下来显示的事件标明更新模板中已发生更改的各个资源的更新开始和完成。例如，更新名为 MyDB 的 AWS::RDS::DBInstance 资源将产生以下条目：

```
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

当 CloudFormation 报告已开始更新资源时，将会记录一个 UPDATE_IN_PROGRESS 事件。在资源创建成功时，将记录 UPDATE_COMPLETE 事件。

CloudFormation 成功更新堆栈时，您会看到以下事件：

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_COMPLETE
```

> **Important**
>
> 在堆栈更新操作期间，如果 CloudFormation 需要替换现有资源，它会首先创建一个新资源，然后删除旧资源。但是，在某些情况下，CloudFormation 可能无法删除旧资源（例如，当用户不具有删除给定类型的资源的权限时）。
> CloudFormation 会尝试删除旧资源三次。如果 CloudFormation 无法删除旧资源，其会将旧资源从堆栈中移除并继续更新堆栈。堆栈更新完成后，CloudFormation 会发出 UPDATE_COMPLETE 堆栈事件，但同时会包含一个 StatusReason，以说明无法删除一个或多个资源。CloudFormation 还会针对特定资源发布一个 DELETE_FAILED 事件，以及相应的 StatusReason，以详细说明 CloudFormation 未能删除资源的原因。
> 旧资源仍然存在并会继续产生费用，但无法再通过 CloudFormation 进行访问。要删除旧资源，请使用基础服务的控制台或 API 直接访问旧资源。
> 对于已从堆栈模板中移除的资源也是如此，因此将在堆栈更新期间从堆栈中将它们删除。

资源更新失败时生成的事件

如果资源更新失败，CloudFormation 将报告一个包含失败原因的 UPDATE_FAILED 事件。例如，假设您的更新模板指定了资源不支持的属性更改，如缩减 AWS::RDS::DBInstance 资源的 AllocatedStorage 大小，则将看到与以下类似的事件：

```
2011-09-30 09:36 PDT AWS::RDS::DBInstance MyDB UPDATE_FAILED Size cannot be less than
 current size; requested: 5; current: 10
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

如果资源更新失败，CloudFormation 会将升级期间更新的所有资源回滚到更新前的配置。下面列出了您将在更新回滚期间看到的事件示例：

```
2011-09-30 09:38 PDT AWS::CloudFormation::Stack MyStack UPDATE_ROLLBACK_COMPLETE
2011-09-30 09:38 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
2011-09-30 09:37 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
2011-09-30 09:37 PDT AWS::CloudFormation::Stack MyStack UPDATE_ROLLBACK_IN_PROGRESS The
 following resource(s) failed to update: [MyDB]
```

继续回滚更新

有时，当 CloudFormation 尝试回滚堆栈更新时，可能无法回滚在更新过程中所做的全部更改。这被称为 UPDATE_ROLLBACK_FAILED 状态。例如，您可能有一个堆栈开始回滚到已在 CloudFormation 之外删除的旧数据库实例。CloudFormation 并不知道此数据库已删除，因此，它假定此数据库实例仍存在并尝试回滚到此实例，从而导致更新回滚失败。

处于 UPDATE_ROLLBACK_FAILED 状态的堆栈无法更新，但可以将其回滚到工作状态（UPDATE_ROLLBACK_COMPLETE）。将堆栈恢复到其初始设置后，您可以尝试再次更新。

在大多数情况下，您必须修复导致更新回滚失败的错误，然后才能继续回滚堆栈。但在某些情况 (如堆栈操作超时) 下，您可在不做任何更改的情况下继续执行回滚操作。

如果无法修复该错误，则可以强制完成回滚。CloudFormation 会跳过所有未能回滚的资源，并将堆栈返回到 UPDATE_ROLLBACK_COMPLETE。有关更多信息，请参阅 强制完成回滚。

> **Note**
>
> 如果您使用嵌套堆栈，则在回滚父堆栈时还会尝试回滚所有子堆栈。

继续回滚更新（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择堆栈所在的 Amazon Web Services 区域。
3. 在堆栈页面上，选择要更新的堆栈，然后依次选择堆栈操作、继续更新回滚。

   如果 错误故障排除 中的解决方案都不起作用，则您可使用高级选项跳过 CloudFormation 无法成功回滚的资源。您必须查找并键入要跳过的资源的逻辑 ID。应只指定在 UpdateRollback 期间而不是转发更新期间进入 UPDATE_FAILED 状态的资源。

   > **Warning**
   >
   > CloudFormation 将指定资源的状态设置为 UPDATE_COMPLETE 并继续回滚堆栈。回滚完成后，已跳过资源的状态将与堆栈模板中资源的状态不一致。在执行另一个堆栈更新之前，您必须更新堆栈或资源以使其互相保持一致。如果您没有这样做，后续堆栈更新可能失败，并且堆栈将变得无法恢复。

指定成功回滚堆栈所需的资源的最小数量。例如，失败的资源更新可能导致从属资源失败。在此情况下，可能不必跳过从属资源。

要跳过属于嵌套堆栈的资源，请使用以下格式：*NestedStackName*.*ResourceLogicalID*。如果要在 Type: AWS::CloudFormation::Stack 列表中指定堆栈资源 (ResourcesToSkip) 的逻辑 ID，则其相应的嵌入式堆栈必须处于以下状态之一：DELETE_IN_PROGRESS、DELETE_COMPLETE 或 DELETE_FAILED。

继续回滚更新 (Amazon CLI)

- 使用 continue-update-rollback 命令和 --stack-name 选项指定要继续回滚的堆栈的 ID。

  要强制完成回滚，请添加 --force-rollback 选项。有关更多信息，请参阅 强制完成回滚。

强制完成回滚

当无法修复导致更新回滚失败的错误时，请使用强制回滚。与其在 --resources-to-skip 中查找和列出每个资源，不如让 CloudFormation 跳过未能回滚的资源。

当您强制回滚时，CloudFormation 会继续回滚堆栈。由于回滚失败，CloudFormation 会跳过所有处于 UPDATE_FAILED 状态的资源（包括嵌套堆栈中的资源）。CloudFormation 将每个跳过的资源的状态设置为 UPDATE_COMPLETE，并附上状态原因 Resource skipped during UpdateRollback。回滚完成后，堆栈状态原因会报告跳过了一个或多个资源。

CloudFormation 仅跳过回滚失败的资源。在向前更新期间失败的资源不会被跳过。

以下 continue-update-rollback 示例强制完成 *MyStack* 堆栈的回滚。

```
aws cloudformation continue-update-rollback --stack-name MyStack \
    --force-rollback
```

不能在同一个请求中使用 --force-rollback 和 --resources-to-skip。要仅跳过一组特定资源，请使用 --resources-to-skip，而不使用 --force-rollback。

> **Warning**
>
> 回滚完成后，被跳过的资源将不再与堆栈模板中的资源相匹配。在执行另一个堆栈更新之前，您必须更新堆栈或资源以使其互相保持一致。如果您没有这样做，后续堆栈更新可能失败，并且堆栈将变得无法恢复。

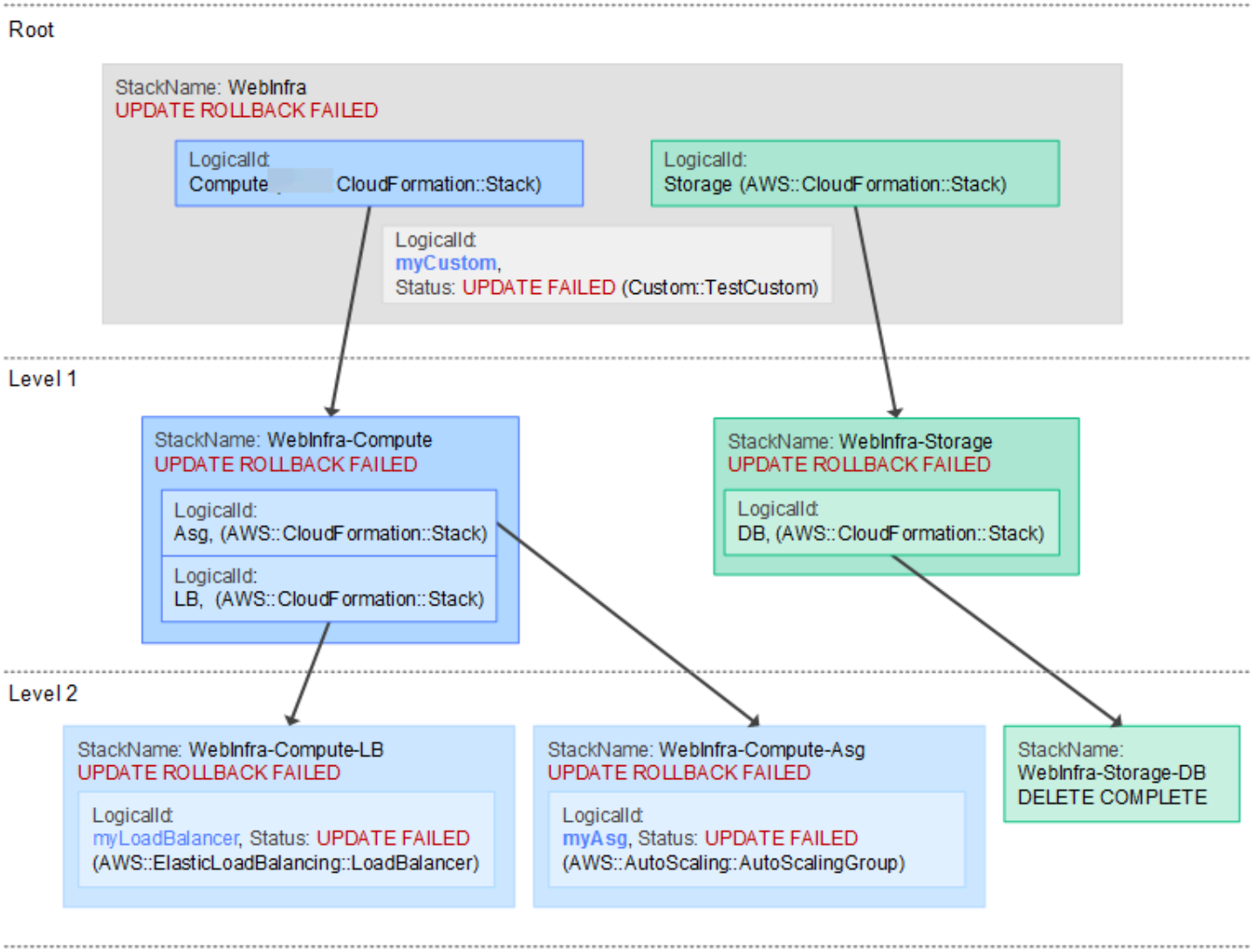
偏差检测将跳过的资源报告为 NOT_CHECKED。要了解跳过的资源是否与模板不同，请对该资源运行 DetectStackResourceDrift。

继续回滚失败的嵌套堆栈更新

如果相互之间有多个嵌套堆栈，则可能需要跳过多个嵌套级别的资源，才能使整个堆栈层次结构恢复到正常工作状态。

例如，您有一个名为 WebInfra 的根堆栈，其中包含两个较小的堆栈：WebInfra-Compute 和 WebInfra-Storage。这两个堆栈中也有自己的嵌套堆栈。

如果更新期间出现问题并且更新过程失败，则整个堆栈层次结构最终可能会处于 UPDATE_ROLLBACK_FAILED 状态，如下图所示。



> **Note**
>
> 为简单起见，本示例中的堆栈名称被截断了。子堆栈名称通常由 CloudFormation 生成，并且包含唯一的随机字符串，因此实际名称可能不够“用户友好”。

要使用 `continue-update-rollback` 命令将根堆栈置于正常工作状态，必须跳过未能回滚的资源。您可以使用 `--force-rollback` 选项在所有嵌套级别跳过它们，也可以使用 `--resources-to-skip` 选项列出它们。

以下 continue-update-rollback 示例强制完成整个层次结构的回滚。您无需查找嵌套堆栈名称或逻辑 ID。

```
aws cloudformation continue-update-rollback --stack-name WebInfra \
```

```
    --force-rollback
```

以下 continue-update-rollback 示例从之前失败的堆栈更新中恢复回滚操作。在此示例中，`--resources-to-skip` 选项包含以下项目：

- *myCustom*
- *WebInfra-Compute-Asg.myAsg*
- *WebInfra-Compute-LB.myLoadBalancer*
- *WebInfra-Storage.DB*

对于根堆栈的资源，您只需要提供逻辑 ID 即可，例如 *myCustom*。但是，对于嵌套堆栈中包含的资源，您必须同时提供嵌套堆栈名称及其逻辑 ID，并以句点分隔。例如 *WebInfra-Compute-Asg.myAsg*。

```
aws cloudformation continue-update-rollback --stack-name WebInfra \
    --resources-to-skip myCustom WebInfra-Compute-Asg.myAsg WebInfra-Compute-
LB.myLoadBalancer WebInfra-Storage.DB
```

查找嵌套堆栈的堆栈名称

您可以在子堆栈的堆栈 ID 或 Amazon 资源名称（ARN）中找到此信息。

以下 ARN 示例引用一个名为 `WebInfra-Storage-Z2VKC706XKXT` 的堆栈。

```
arn:aws:cloudformation:us-east-1:123456789012:stack/WebInfra-Storage-Z2VKC706XKXT/
ea9e7f90-54f7-11e6-a032-028f3d2330bd
```

查找嵌套堆栈的逻辑 ID

您可以在子堆栈父级的模板定义中找到子堆栈的逻辑 ID。图中 `WebInfra-Storage-DB` 子堆栈的 `LogicalId` 是其父级 `WebInfra-Storage` 中的 `DB`。

在 CloudFormation 控制台中，您还可以在 Resources（资源）选项卡或 Events（事件）选项卡上堆栈资源的 Logical ID（逻辑 ID）列中找到该逻辑 ID。有关更多信息，请参阅 从 CloudFormation 控制台查看堆栈信息。

> **Note**
>
> 该 `RollbackStack` 操作还支持 Express Mode。您可以结合使用 `--deployment-config '{"mode": "EXPRESS"}'` 和 rollback-stack 命令，以便更快地回滚失败的堆栈。有关更多信息，请参阅 Express Mode。

确定堆栈失败的原因

如果堆栈创建失败，CloudFormation 可帮助您确定可能是堆栈失败根本原因的事件。根据场景和您的权限，如果事件中提供的状态原因不明确，Amazon CloudTrail 事件可以提供有关根本原因的更多详细信息。

确定堆栈失败的根本原因

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面上，选择失败的堆栈。
3. 选择事件选项卡。
4. 选择检测根本原因。CloudFormation 会分析失败，并通过向特定事件状态添加可能的根本原因标签，指出可能是失败原因的事件。有关 CloudFormation 控制台中状态的进一步说明，请参阅状态原因。
5. 选择标签为可能的根本原因的失败状态，了解有关失败原因的更多信息。视场景和您的权限而定，您可以查看详细的 CloudTrail 事件。选择状态可能会产生以下结果
   - 提供与此问题相关的、并且可能有助于解决问题的 CloudTrail 事件。查看 CloudTrail 事件。
   - 找不到与此问题相关的任何有助于解决问题的 CloudTrail 事件。
   - 当前权限不允许您查看 CloudTrail 事件。了解更多。
   - 正在检查可用的 CloudTrail 事件，请在几分钟后再次检查。
   - 获取 CloudTrail 事件时发生错误。如需执行手动检查，请访问 CloudTrail 控制台。
6. 如果状态原因中提供的原因不明确，并且根本原因显示指向 CloudTrail 控制台的链接，请打开该链接查看事件，以查找关于根本原因的详细信息。

有关 CloudTrail 事件的更多信息，请参阅 Understanding CloudTrail events 和 CloudTrail record contents。

有关 CloudTrail 事件历史记录的更多信息，请参阅 Working with CloudTrail Event history。

> **Note**
>
> 嵌套堆栈不支持检测根本原因。

选择预置资源时如何处理失败

如果堆栈操作失败，您无需每次都回滚已成功预置的资源并从头开始。相反，您可以对处于 CREATE_FAILED 或 UPDATE_FAILED 状态的资源进行问题排查，然后出现问题的位置恢复预置。

要执行操作，必须启用“保留已成功预置的资源”选项。此选项适用于所有堆栈部署和更改集操作。

- 对于堆栈创建，如果您选择保留已成功预置的资源选项，CloudFormation 将保留已成功创建的资源的状态，并将失败的资源保持在失败状态，直到执行下一次更新操作为止。
- 在更新和更改集操作过程中，选择保留已成功预置的资源将保留已成功资源的状态，同时将失败的资源回滚到上一个已知的稳定状态。失败的资源将处于 UPDATE_FAILED 状态。没有上一个已知稳定状态的资源将在下一次堆栈操作时删除。

主题

- 堆栈故障选项概述
- 暂停堆栈回滚的必需条件
- 保留成功预置的资源（控制台）
- 保留成功预置的资源（Amazon CLI）

堆栈故障选项概述

在从 CloudFormation 控制台、API 或 Amazon CLI 发出操作之前，请指定预置资源失败时的行为。然后，继续进行资源的部署过程，无需进行任何其他修改。如果发生操作故障，CloudFormation 会在每条独立预置路径中的第一个故障处停止。CloudFormation 识别资源之间的依赖关系以并行处理独立的预置操作。然后继续在每条独立的预置路径上预置资源，直到遇到故障。一条路径中的故障不会影响其他预置路径。CloudFormation 将继续预置资源，直到完成或因其他故障而停止。

修复任何问题以继续部署过程。CloudFormation 在对之前无法成功预置的资源重试预置操作之前执行必要的更新。您可以通过提交 Retry（重试）、Update（更新）或 Roll back（回滚）操作来修复

问题。例如，如果您预置 Amazon EC2 实例并且 EC2 实例在创建操作期间失败，您可能想要调查错误，而不是立即回滚失败的资源。您可以查看系统状态检查和实例状态检查，然后在问题解决后选择 Retry（重试）操作。

如果堆栈操作失败，并且您已从 Stack failure options（堆栈故障选项）菜单中指定 Preserve successfully provisioned resources（保留成功预置的资源），您可以选择以下选项。

- Retry（重试）- 重试失败资源的预置操作并继续预置模板，直到堆栈操作成功完成或下一次失败。如果资源由于不需要修改模板的问题（例如 Amazon Identity and Access Management（IAM）权限）而无法预置，请选择此选项。
- Update（更新）- 已预置的资源在模板更新时进行更新。将重试未能创建或更新的资源。如果由于模板错误而无法预置资源，并且您已修改模板，请选择此选项。更新处于 `FAILED` 状态的堆栈时，必须为 Stack failure options（堆栈故障选项）选择 Preserve successfully provisioned resources（保留成功预置的资源）才能继续更新您的堆栈。
- Roll back（回滚）- CloudFormation 将堆栈回滚到上一个已知的稳定状态。

暂停堆栈回滚的必需条件

要阻止 CloudFormation 自动回滚和删除已成功创建的资源，必须满足以下条件。

1. 创建或更新堆栈时，必须选择保留已成功预置的资源选项。该选项会告诉 CloudFormation 不要删除已成功创建的资源，即使整个堆栈操作失败。
2. 堆栈操作必须已经失败，这意味着堆栈状态为 `CREATE_FAILED` 或 `UPDATE_FAILED`。

> **Note**
>
> 不支持不可变的更新类型。

保留成功预置的资源（控制台）

Create stack

在创建堆栈操作期间保留成功预置的资源

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。

2. 在堆栈页面，选择右上角的创建堆栈，然后选择使用新资源（标准）。
3. 在先决条件 – 准备模板中，选择选择现有模板。
4. 在 Specify template (指定模板) 下，选择指定包含堆栈模板的 S3 存储桶的 URL，或者上传堆栈模板文件。然后选择下一步。
5. 在 Specify stack details（指定堆栈详细信息）页面上，在 Stack name（堆栈名称）框中输入堆栈名称。
6. 在 Parameters（参数）部分中，指定在堆栈模板中定义的参数。

   您可使用或更改带默认值的任何参数。
7. 如果对参数值满意，选择 Next（下一步）。
8. 在 Configure stack options（配置堆栈选项）页面上，可以设置堆栈的其他选项。
9. 对于 Stack failure options（堆栈故障选项），选择 Preserve successfully provisioned resources（保留成功预置的资源）。
10. 如果对堆栈选项满意，选择 Next（下一步）。
11. 在 Review（审核）页面上查看堆栈，然后选择 Create stack（创建堆栈）。

结果：创建失败的资源将堆栈状态转换为 `CREATE_FAILED`，以防止堆栈操作遇到故障时堆栈回滚。成功预置的资源处于 `CREATE_COMPLETE` 状态。您可在 Stack events（堆栈事件）选项卡中监控堆栈。

Update stack

在更新堆栈操作期间保留成功预置的资源

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 选择要更新的堆栈，然后选择 Update（更新）。
3. 在 Update stack（更新堆栈）页面上，使用以下选项之一选择堆栈模板：

   - 使用现有模板
   - 替换当前模板
   - 在模板工作室中打开

   接受您的设置并选择 Next（下一步）。
4. 在 Specify stack details（指定堆栈详细信息）页面上，指定在堆栈模板中定义的参数。

您可使用或更改带默认值的任何参数。

5. 如果对参数值满意，选择 Next（下一步）。
6. 在 Configure stack options（配置堆栈选项）页面上，可以设置堆栈的其他选项。
7. 对于 Behavior on provisioning failure（预置失败时的行为），选择 Preserve successfully provisioned resources（保留成功预置的资源）。
8. 如果对堆栈选项满意，选择 Next（下一步）。
9. 在 Review（查看）页面上查看堆栈，然后选择 Update stack（更新堆栈）。

结果：更新失败的资源将堆栈状态转换为 UPDATE_FAILED，并回滚到上一个已知的稳定状态。没有上一个已知稳定状态的资源将在下一次堆栈操作时被 CloudFormation 删除。成功预置的资源处于 CREATE_COMPLETE 或 UPDATE_COMPLETE 状态。您可在 Stack events（堆栈事件）选项卡中监控堆栈。

Change set

> **Note**
>
> 您可以为状态为 CREATE_FAILED 或 UPDATE_FAILED 的堆栈启动更改集，但不能为状态为 UPDATE_ROLLBACK_FAILED 的堆栈启动更改集。

在更改集操作期间保留成功预置的资源

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 选择包含要启动的更改集的堆栈，然后选择 Change sets（更改集）选项卡。
3. 选择更改集，然后选择 Execute（执行）。
4. 对于 Execute change set（执行更改集），选择 Preserve successfully provisioned resources（保留成功预置的资源）选项。
5. 选择 Execute change set（执行更改集）。

结果：更新失败的资源将堆栈状态转换为 UPDATE_FAILED，并回滚到上一个已知的稳定状态。没有上一个已知稳定状态的资源将在下一次堆栈操作时被 CloudFormation 删除。成功预置的资源处于 CREATE_COMPLETE 或 UPDATE_COMPLETE 状态。您可在 Stack events（堆栈事件）选项卡中监控堆栈。

保留成功预置的资源（Amazon CLI）

Create stack

在堆栈创建操作期间保留成功预置的资源

在 create-stack 操作期间指定 `--disable-rollback` 选项或 `on-failure DO_NOTHING` 枚举。

1. 为 create-stack 命令提供堆栈名称和模板并采用 `--disable-rollback` 选项。

```
aws cloudformation create-stack --stack-name myteststack \
    --template-body file://template.yaml \
    --disable-rollback
```

该命令将返回以下输出。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

2. 使用 describe-stacks 命令描述堆栈状态。

```
aws cloudformation describe-stacks --stack-name myteststack
```

该命令将返回以下输出。

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template",
            "Tags": [],
            "Outputs": [],
            "StackStatusReason": “The following resource(s) failed to create:
 [MyBucket]”,
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
```

```
            "StackStatus": "CREATE_FAILED",
            "DisableRollback": true
        }
    ]
}
```

Update stack

在堆栈更新操作期间保留成功预置的资源

1. 为 update-stack 命令提供现有堆栈名称和模板并采用 `--disable-rollback` 选项。

```
aws cloudformation update-stack --stack-name myteststack \
    --template-url https://s3.amazonaws.com/amzn-s3-demo-bucket/updated.template
 --disable-rollback
```

该命令将返回以下输出。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

2. 使用 describe-stacks 或 describe-stack-events 命令描述堆栈状态。

```
aws cloudformation describe-stacks --stack-name myteststack
```

该命令将返回以下输出。

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template",
            "Tags": [],
            "Outputs": [],
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "UPDATE_COMPLETE",
```

```
            "DisableRollback": true
        }
    ]
}
```

Change set

> **Note**
>
> 您可以为状态为 CREATE_FAILED 或 UPDATE_FAILED 的堆栈启动更改集，但不能为状态为 UPDATE_ROLLBACK_FAILED 的堆栈启动更改集。

在更改集操作期间保留成功预置的资源

在 execute-change-set 操作期间指定 --disable-rollback 选项。

1. 为 execute-change-set 命令提供堆栈名称和模板并采用 --disable-rollback 选项。

```
aws cloudformation execute-change-set --stack-name myteststack \
    --change-set-name my-change-set --template-body file://template.yaml
```

该命令将返回以下输出。

```
{
 "Id": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/my-change-set/
bc9555ba-a949-xmpl-bfb8-f41d04ec5784",
 "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

2. 启动更改集并采用 --disable-rollback 选项。

```
aws cloudformation execute-change-set --stack-name myteststack \
    --change-set-name my-change-set --disable-rollback
```

3. 使用 describe-stacks 或 describe-stack-events 命令确定堆栈状态。

```
aws cloudformation describe-stack-events --stack-name myteststack
```

该命令将返回以下输出。

```
{
   "StackEvents": [
     {
        "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
        "EventId": "49c966a0-7b74-11ea-8071-024244bb0672",
        "StackName": "myteststack",
        "LogicalResourceId": " MyBucket",
        "PhysicalResourceId": "myteststack-MyBucket-abcdefghijk1",
        "ResourceType": "AWS::S3::Bucket",
        "Timestamp": "2020-04-10T21:43:17.015Z",
        "ResourceStatus": "UPDATE_FAILED"
        "ResourceStatusReason": "User XYZ is not allowed to perform
 S3::UpdateBucket on MyBucket"
     }
}
```

4. 修复权限错误并重试该操作。

```
aws cloudformation update-stack --stack-name mytеststack \
    --use-previous-template --disable-rollback
```

该命令将返回以下输出。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

5. 使用 describe-stacks 或 describe-stack-events 命令描述堆栈状态。

```
aws cloudformation describe-stacks --stack-name mytеststack
```

该命令将返回以下输出。

```
{
    "Stacks":  [
        {
```

```
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template",
            "Tags": [],
            "Outputs": [],
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "UPDATE_COMPLETE",
            "DisableRollback": true
        }
    ]
}
```

回滚堆栈

您可以使用 rollback-stack 命令将堆栈状态为 CREATE_FAILED 或 UPDATE_FAILED 的堆栈回滚到上一个稳定状态。

以下 rollback-stack 命令将会回滚指定的堆栈。

```
aws cloudformation rollback-stack --stack-name myteststack
```

该命令将返回以下输出。

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

> **Note**
>
> 如果堆栈不包含上一个已知的稳定状态，则 rollback-stack 操作将删除该堆栈。

Express Mode 和回滚

当您使用 Express Mode 时，CloudFormation 会默认禁用回滚功能。如果资源操作失败，CloudFormation 不会自动回滚之前所完成的更改。相反，堆栈将进入失败状态，您可以先排查问题，然后再重试。

要在 Express Mode 下重新启用回滚功能，请在执行堆栈操作时，将 A--deployment-config 参数中的 disableRollback 设置为 false：

```
aws cloudformation create-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --deployment-config '{"mode": "EXPRESS", "disableRollback": false}'
```

> **Note**
>
> 不可变更新操作不支持“禁用回滚”。如果更新操作需要替换资源，且操作失败，则无法保留失败状态以供重试。

有关 Express Mode 的更多信息，请参阅 Express Mode。

使用回滚触发器在突破警报阈值时回滚 CloudFormation 堆栈

利用回滚触发器，您可让 Amazon CloudFormation 在堆栈创建和更新期间监控应用程序的状态，并可在应用程序超出您指定的警报阈值时回滚操作。对于您创建的每个回滚触发器，您需要指定 CloudFormation 应监控的 CloudWatch 警报。CloudFormation 会在堆栈创建或更新操作期间以及所有资源部署完毕之后的指定时间段内，一直监控指定的警报。在堆栈操作或监控期间，如果有任何警报进入 ALARM 状态，CloudFormation 就会回滚整个堆栈操作。

您可以将监控时间设置为 0 (默认值) 到 180 分钟之间。在堆栈创建或更新操作部署了所有必需的资源之后，CloudFormation 仍会在这段时间内监控所有的回滚触发器。在堆栈操作或监控期间，如果有任何警报进入 ALARM 状态，CloudFormation 就会回滚整个堆栈操作。对于更新操作，如果监控期间结束而没有任何警报进入 ALARM 状态，CloudFormation 将继续照常处置旧资源。如果您设置了监控时间，但没有指定任何回滚触发器，CloudFormation 仍然会等待指定的时长，然后清理掉更新操作的旧资源。您可以使用此监控期间来执行任何所需的手动堆栈验证，然后根据需要手动取消堆栈创建或更新操作。如果您将监控时间设置为 0 分钟，CloudFormation 仍会在堆栈创建和更新操作期间监控回滚触发器，并在有警报进入 ALARM 状态时回滚操作。对于没有发生警报的更新操作，系统会在该操作完成时立即开始处置旧资源。

默认情况下，CloudFormation 仅在警报进入 ALARM 状态（而不是 INSUFFICIENT_DATA 状态）时才回滚堆栈操作。如果还要让 CloudFormation 在有警报进 INSUFFICIENT_DATA 状态时回滚堆栈操作，请编辑 CloudWatch 警报以便将缺少数据视为 breaching。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的配置 CloudWatch 警报处理缺少数据的方式。

在更新操作期间回滚堆栈时，CloudFormation 不会监控回滚触发器。

您可以添加最多 5 个回滚触发器。要添加回滚触发器，请指定 CloudWatch 警报的 Amazon 资源名称（ARN）。目前，`AWS::CloudWatch::Alarm` 和 `AWS::CloudWatch::CompositeAlarm` 类型可以用作回滚触发器。有关 CloudWatch 警报的更多信息，请参阅《Amazon CloudWatch 用户指南》中的使用 Amazon CloudWatch 警报。

如果给定的 CloudWatch 警报缺失，则整个堆栈操作都会失败并且回滚。

请注意，访问 CloudWatch 需要凭证。这些凭证必须有权访问 Amazon 资源，例如检索有关您的资源的 CloudWatch 指标数据。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的 Amazon CloudWatch 的身份验证和访问控制。

在堆栈创建或更新期间添加回滚触发器

在堆栈创建或更新期间添加回滚触发器（控制台）

1. 创建或更新堆栈时，在配置堆栈选项页面的高级选项下，展开回滚配置部分。
2. 输入一个介于 `0` 到 `180` 分钟之间的监控时间。默认值为 `0`。
3. 指定要用作回滚触发器的 CloudWatch 警报或复合警报的 ARN，然后选择添加 CloudWatch 警报 ARN。

   例如，以下是 CloudWatch 警报或复合警报的 ARN，`arn:aws:cloudwatch:`*`us-east-1`*`:`*`123456789012`*`:alarm:`*`MyAlarmName`*。
4. 选择下一步并查看堆栈的详细信息。
5. 做好准备后，选择提交以创建或更新堆栈。

在堆栈创建或更新期间添加回滚触发器（Amazon CLI）

使用带 `--rollback-configuration` 选项的 create-stack 命令或 update-stack 命令。

例如，以下 update-stack 命令会将 *`MyCompositeAlarm`* 设置为监视周期为 5 分钟的回滚触发器：

```
aws cloudformation update-stack --stack-name MyStack \
  --use-previous-template \
  --rollback-configuration \
  "RollbackTriggers=[{Arn=arn:aws:cloudwatch:us-
east-1:123456789012:alarm:MyCompositeAlarm,Type=AWS::CloudWatch::CompositeAlarm}],MonitoringTim
```

将回滚触发器添加到更改集

将回滚触发器添加到更改集（控制台）

1. 创建更改集期间，在配置堆栈选项页面的高级选项下，展开回滚配置部分。
2. 输入一个介于 0 到 180 分钟之间的监控时间。默认值为 5。
3. 指定要用作回滚触发器的 CloudWatch 警报或复合警报的 ARN，然后选择添加 CloudWatch 警报 ARN。

   例如，以下是 CloudWatch 警报或复合警报的 ARN，`arn:aws:cloudwatch:`*`us-east-1`*`:123456789012:alarm:`*`MyAlarmName`*。
4. 选择下一步并查看更改集的详细信息。
5. 做好准备后，选择创建更改集以创建更改集。

将回滚触发器添加到更改集（Amazon CLI）

使用带 `--rollback-configuration` 选项的 create-change-set 命令。

查看堆栈的回滚触发器

要查看堆栈的回滚触发器，请查看 Rollback configuration（回滚配置）部分。

1. 在堆栈页面上，从左侧列表中选择要查看的堆栈。
2. 在堆栈信息选项卡上，展开回滚配置部分，查看回滚触发器。

查看更改集的回滚触发器

要查看更改集的回滚触发器，请查看 Rollback configuration（回滚配置）部分。

1. 在堆栈页面上，从左侧列表中选择要查看的堆栈。
2. 选择更改集选项卡，然后选择要查看的更改集。
3. 选择输入选项卡，然后查看回滚配置部分。

使用偏移检测功能检测堆栈和资源的非托管配置更改

即使您通过 CloudFormation 管理您的资源，用户也可以在 CloudFormation之外更改这些资源。用户可以使用创建资源的底层服务直接编辑资源。例如，您可以使用 Amazon EC2 控制台来更新作为

CloudFormation 堆栈的一部分创建的服务器实例。有些更改可能是偶然的，而有些更改可能会故意响应对时间敏感的操作事件。无论如何，在 CloudFormation 之外进行的更改会导致堆栈更新或删除操作变得复杂。您可以使用偏差检测来识别在 CloudFormation 管理之外进行配置更改的堆栈资源。然后，您可以采取纠正措施，使堆栈资源再次与其在堆栈模板中的定义同步，例如直接更新偏离的资源，以便它们与其模板定义一致。解决偏差有助于确保配置一致性和成功的堆栈操作。

主题

- 什么是偏差？
- 偏差检测状态代码
- 检测偏差时的注意事项
- 在整个 CloudFormation 堆栈上检测偏差
- 在单个堆栈资源上检测偏差
- 通过导入操作消除偏差

什么是偏差？

通过偏离检测，您可以检测堆栈的实际配置是否与其预期配置不同或已经偏离。使用 CloudFormation 可在整个堆栈上或在堆栈内的单个资源上检测偏差。如果某个资源的任何实际属性值与预期的属性值不同，则认为该资源已偏离。这包括是否已删除属性或资源。如果堆栈的一个或多个资源已偏离，则认为堆栈已偏离。

为了确定资源是否已偏离，CloudFormation 确定预期的资源属性值（如堆栈模板中所定义）以及任何指定为模板参数的值。然后 CloudFormation 将这些预期值与堆栈中当前存在的资源属性的实际值进行比较。如果资源的一个或多个属性已被删除或其值已更改，则认为该资源已偏离。

CloudFormation 生成有关已偏离的堆栈中每个资源的详细信息。

CloudFormation 在那些支持偏差检测的 Amazon 资源上检测偏差。不支持偏差检测的资源会被分配 NOT_CHECKED 的偏差状态。有关支持偏差检测的 Amazon 资源的列表，请参阅 资源类型支持。

此外，CloudFormation 还支持对可预置的私有资源类型（即其预置类型为 `FULLY_MUTABLE` 或 `IMMUTABLE`）进行偏差检测。要对私有资源类型的资源执行偏差检测，您在账户中注册的资源类型的默认版本必须可预置。有关资源预置类型的更多信息，请参阅《Amazon CloudFormation API Reference》中 DescribeType 操作的 `ProvisioningType` 参数和《Amazon CLI Command Reference》中 DescribeType 命令的参数。有关私有资源的更多信息，请参阅 通过 CloudFormation 注册表管理扩展。

您可以在具有以下状态的堆栈上执行偏差检测：CREATE_COMPLETE、UPDATE_COMPLETE、UPDATE_ROLLBACK_COMPLETE 和 UPDATE_ROLLBACK_FAILED。

当在堆栈上检测偏差时，CloudFormation 不在任何属于该堆栈的嵌套堆栈上检测偏差。有关更多信息，请参阅 使用嵌套堆栈将模板拆分为可重复使用的部分。相反，您可以直接在嵌套堆栈上启动偏差检测操作。

> **Note**
>
> CloudFormation 仅通过堆栈模板或通过指定模板参数来确定显式设置的属性值的偏差。这不包括资源属性的默认值。要使 CloudFormation 跟踪资源属性以确定偏差，请显式设置属性值，即使您将其设置为默认值也是如此。CloudFormation 还会检测堆栈级别标签上的偏差。

偏差检测状态代码

本节中的表描述了偏差检测使用的各种状态类型：

- 偏差检测操作状态描述偏差操作的当前状态。
- 偏差状态

  对于堆栈集，它根据属于堆栈集的堆栈实例的偏差状态以描述堆栈集的总体偏差状态。

  对于堆栈实例，它根据堆栈实例的关联堆栈的偏差状态以描述堆栈实例的偏差状态。

  对于堆栈，它根据堆栈资源的偏差状态以描述堆栈的总体偏差状态。
- 资源偏差状态描述单个资源的偏差状态。

下表列出了 CloudFormation 分配给堆栈偏差检测操作的状态代码。

偏差检测操作状态	说明
DETECTION_COMPLETE	已经对支持偏差检测的堆栈中的所有资源成功完成了堆栈偏差检测操作。
DETECTION_FAILED	堆栈偏差检测操作对于堆栈中的至少一个资源失败。结果将可用于 CloudFormation 在其上成功完成偏差检测的资源。

偏差检测操作状态	说明
DETECTION_IN_PROGRESS	堆栈偏差检测操作目前正在进行中。

下表列出了 CloudFormation 分配给堆栈的偏差状态代码。

偏差状态	说明
DRIFTED	对于堆栈：堆栈与其预期模板配置不同或已偏离。如果堆栈的一个或多个资源已偏离，则认为堆栈已偏离。 对于堆栈实例：如果与堆栈实例关联的堆栈已偏离，则认为该实例已偏离。 对于堆栈集：如果一个或多个堆栈实例已偏离，则认为堆栈集已偏离。
NOT_CHECKED	CloudFormation 尚未检查堆栈、堆栈集或堆栈实例是否与其预期模板配置不同。
IN_SYNC	每个受支持资源的当前配置与其预期的模板配置相匹配。没有支持偏差检测的资源的堆栈、堆栈集或堆栈实例也具有 IN_SYNC 状态。

下表列出了 CloudFormation 分配给堆栈资源的偏差代码。

资源偏差状态	说明
DELETED	资源与预期的模板配置不同，因为资源已被删除。
MODIFIED	资源与预期的模板配置不同。
NOT_CHECKED	CloudFormation 没有检查资源是否与预期的模板配置不同。
IN_SYNC	资源的当前配置与其预期的模板配置相匹配。

下表列出了 CloudFormation 分配给与预期模板配置不同的资源属性的差异类型状态代码。

属性差异类型	说明
ADD	已将值添加到数据类型为数组或列表的资源属性。
REMOVE	属性已从当前资源配置中删除。
NOT_EQUAL	当前属性值与堆栈模板中定义的预期值不同。

检测偏差时的注意事项

为了在堆栈上成功执行偏差检测，用户必须具有以下权限：

- 支持在堆栈中包含的偏差检测的每个资源的读取权限。例如，如果堆栈包含 AWS::EC2::Instance 资源，则您必须具有 ec2:DescribeInstances 权限才能在堆栈上执行偏差检测。
- cloudformation:DetectStackDrift
- cloudformation:DetectStackResourceDrift
- cloudformation:BatchDescribeTypeConfigurations

有关在 CloudFormation 中设置权限的更多信息，请参阅 使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限。

在某些边缘情况下，CloudFormation 可能无法始终返回准确的偏差结果。您应该了解这些边缘情况，以便正确解释您的偏差检测结果。

- 在某些情况下，属性数组中包含的对象将报告为偏差，而实际上它们是从负责该资源的底层服务提供给属性的默认值。
- 某些资源与相关资源具有附件关系，使得资源可以实际附加或删除在相同或另一模板中定义的另一资源的属性值。例如，如果 AWS::EC2::SecurityGroupIngress 和 AWS::EC2::SecurityGroupEgress 资源也可用于连接和删除 AWS::EC2::SecurityGroup 资源中的值。在这些情况下，CloudFormation 在执行偏差比较之前分析附件的堆栈模板。但是，CloudFormation 无法跨堆栈执行此分析，因此可能不会返回准确的偏差结果，其中附加的资源驻留在不同的堆栈中。

  支持偏差检测并允许或要求其他资源中的附件的资源包括：

资源类型	附件资源类型
AWS::SNS::Topic	AWS::SNS::Subscription
AWS::IAM::User	AWS::IAM::UserToGroupAddition
AWS::IAM::Group	AWS::IAM::ManagedPolicy
AWS::IAM::Role	AWS::IAM::Policy
AWS::IAM::User	AWS::IAM::RolePolicy
AWS::ElasticLoadBalancingV2 ::Listener	AWS::ElasticLoadBalancingV2 ::ListenerCertificate
AWS::EC2::SecurityGroup	AWS::EC2::SecurityGroupEgress
	AWS::EC2::SecurityGroupIngress

- CloudFormation 不对任何资源的 KMSKeyId 属性执行偏差检测。由于 Amazon KMS 密钥可以由多个别名引用，因此 CloudFormation 无法保证此属性有一致准确的偏差结果。
- 您可以在堆栈模板中指定某些资源属性，就其本质而言，CloudFormation 无法将其与生成的堆栈资源中的属性进行比较。因此，这些属性不能包括在偏差检测结果中。这些属性可分为两大类：
  - CloudFormation 无法在堆栈模板中映射回初始资源属性值的属性值。

    例如，CloudFormation 无法将 Lambda 函数的源代码映射回 AWS::Lambda::Function 资源的 Code 属性类型，因此 CloudFormation 无法将其包括在偏差检测结果中。
  - 负责资源的服务不返回的属性值。

    某些属性值被有意设计成永远不会由资源所属的服务返回。这些属性值往往包含机密信息，例如密码或其他不应泄露的敏感数据。例如，IAM 服务永远不会返回 AWS::IAM::User LoginProfile 属性类型的 Password 属性值，因此 CloudFormation 不能将其包含在偏差检测结果中。
  - 数组中的对象实际上可能是服务默认值，而不是在手动添加的偏差。
- 如果您遇到任何误报，请使用 CloudFormation 控制台中的反馈链接将您的评论发送给我们，或通过 Amazon Web Services re:Post 与我们联系。
- 某些属性可能有相等但不完全相同的输入值。为避免误报，应确保预期配置与实际配置相匹配。

- 例如，资源属性的预期配置可能为 1024MB，而同一资源属性的实际配置可能为 1GB。1024MB 和 1GB 相等但不完全相同。

  当在此资源属性上运行偏差检测时，它将发出偏差结果信号。

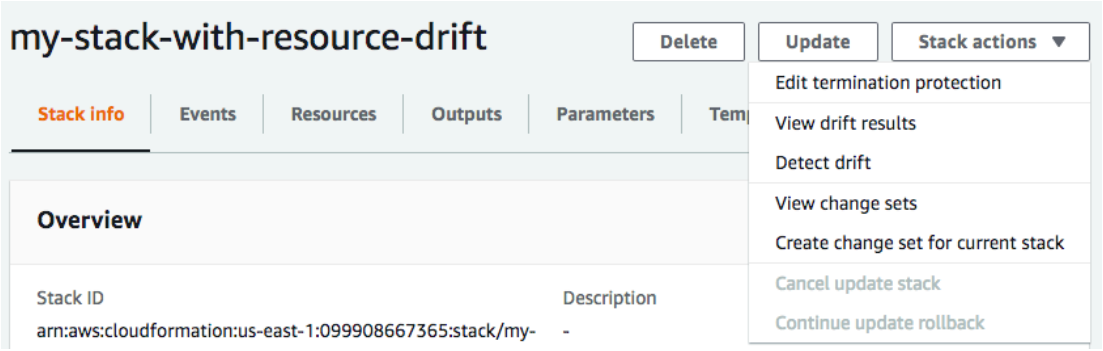
  为避免误报，请将资源属性的预期配置更改为 1024MB，然后运行偏差检测。

在整个 CloudFormation 堆栈上检测偏差

在堆栈上执行偏差检测操作会确定堆栈是否已偏离其预期的模板配置，并返回有关支持偏差检测的堆栈中每个资源的偏差状态的详细信息。

使用 Amazon Web Services 管理控制台 在整个堆栈上检测偏差

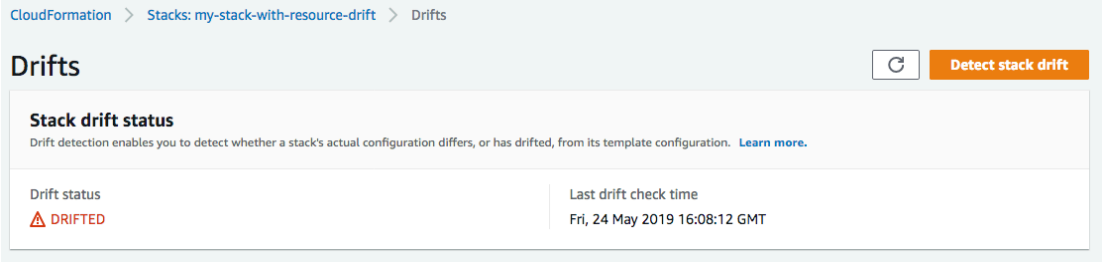
1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 从堆栈列表中，选择要执行偏差检测的堆栈。在堆栈详细信息窗格中，选择 Stack actions (堆栈操作)，然后选择 Detect drift (检测偏差)。



   CloudFormation 会显示一个信息栏，指示已为选定堆栈启动偏差检测。
3. 请耐心等待，直到 CloudFormation 完成偏差检测操作。偏差检测操作完成后，CloudFormation 会更新堆栈的偏差状态和上一次偏差检查时间。这些字段在堆栈详细信息页面上 Stack info（堆栈信息）窗格的 Overview（概述）部分中列出。

   偏差检测操作可能需要几分钟，具体取决于堆栈中包含的资源数量。您一次只能在给定堆栈上运行一个偏差检测操作。即使您关闭信息栏，CloudFormation 也会继续执行偏差检测操作。
4. 查看堆栈及其资源的偏差检测结果。在选定堆栈的情况下，从 Stack actions (堆栈操作) 菜单中选择 View drift results (查看偏差结果)。

CloudFormation 会列出堆栈的整体偏差状态，以及最后一次在堆栈或其任何单个资源上启动偏差检测的时间。如果堆栈的一个或多个资源已偏离，则认为堆栈已偏离。



在 Resource drift details (资源偏差状态) 部分中，CloudFormation 会列出每个堆栈资源、其偏差状态，以及最后一次在资源上启动偏差检测的时间。将会显示每个资源的逻辑 ID 和物理 ID，以帮助您识别它们。此外，对于状态为 MODIFIED (已修改) 的资源，CloudFormation 会显示资源偏差详细信息。

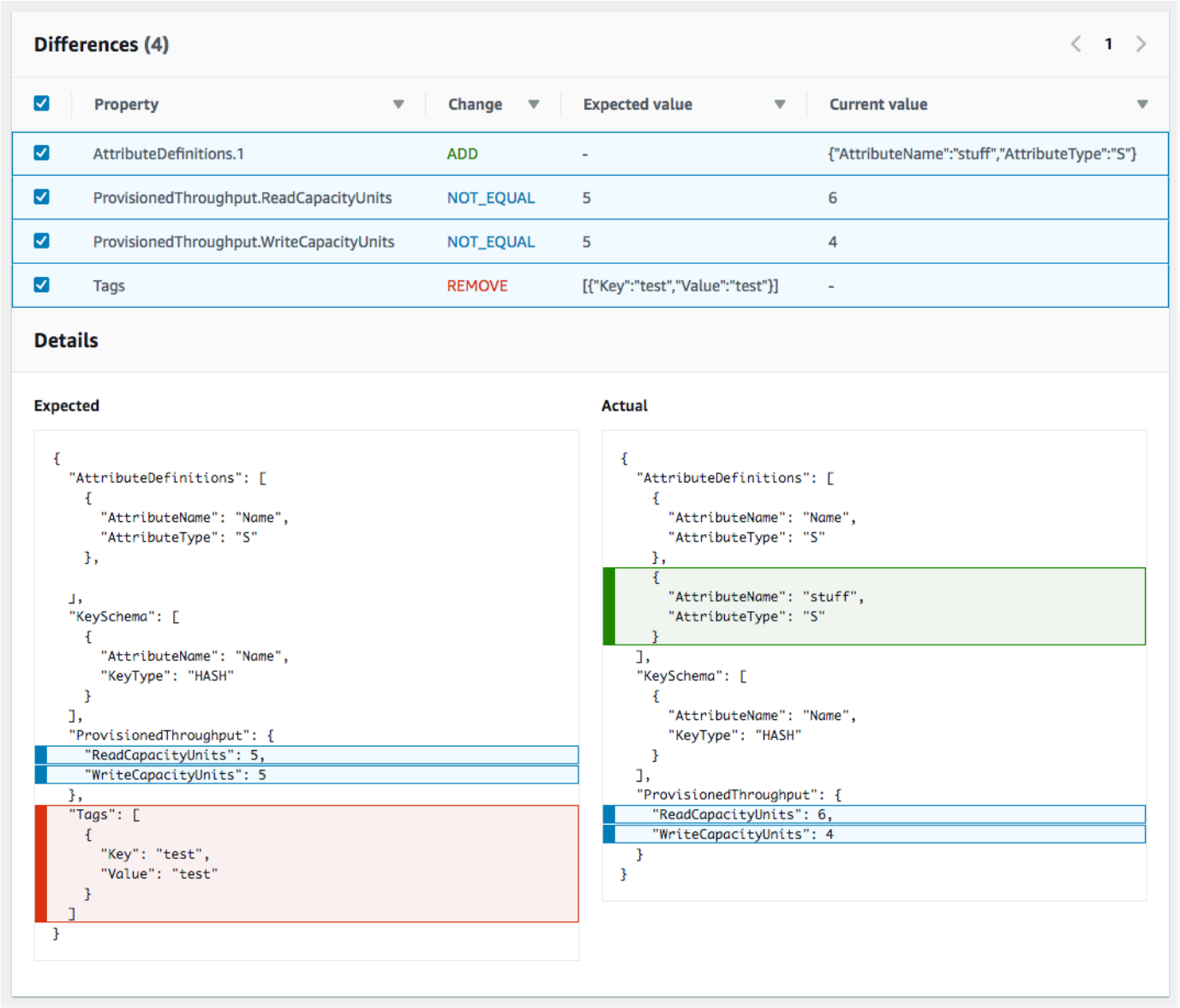
您可以使用偏差状态列根据资源偏差状态对资源排序。

- 查看已修改资源的详细信息。
  - 选定已修改的资源，选择 View drift details（查看偏差详细信息）。

    CloudFormation 会显示该资源的偏差详细信息页面。此页面会列出资源的预期和当前属性值，以及两者之间的任何区别。

    要突出显示区别，请在区别部分中选择属性名称。

    - 添加的属性在 Details（详细信息）部分的 Current（当前）列中以绿色突出显示。
    - 删除的属性在 Details（详细信息）部分的 Expected（预期）列中以红色突出显示。
    - 其值已更改的属性在 Expected (预期) 和 Current (当前) 列中以黄色突出显示。



使用 Amazon CLI 在整个堆栈上检测偏差

> **Important**
>
> 查看堆栈的上次偏差检查时间，并确认它早于资源偏差结果中显示的时间戳，以防止使用过时的数据。

要使用 Amazon CLI 在整个堆栈上检测偏差，请使用以下 Amazon CLI 命令：

- detect-stack-drift，用于对堆栈启动偏差检测操作。
- describe-stack-drift-detection-status，用于监控堆栈偏差检测操作的状态。
- describe-stack-resource-drifts，可查看堆栈偏差检测操作的详细信息。

1. 使用 detect-stack-drift 可在整个堆栈上检测偏差。指定堆栈名称或 ARN。您还可以指定要用作此偏差检测操作的筛选条件的任何特定资源的逻辑 ID。

   ```
   aws cloudformation detect-stack-drift --stack-name my-stack-with-resource-drift
   ```

   输出：

   ```
   {
       "StackDriftDetectionId": "624af370-311a-11e8-b6b7-500cexample"
   }
   ```

2. 由于堆栈偏差检测操作可能会长时间运行，因此请使用 describe-stack-drift-detection-status 来监控偏差操作的状态。此命令获取 detect-stack-drift 命令返回的堆栈偏移检测 ID。

   在下面的示例中，我们采用了上面的 detect-stack-drift 示例返回的堆栈偏差检测 ID，并将其作为参数传递给 describe-stack-drift-detection-status。该参数返回操作详细信息，以显示偏差检测操作已完成，单个堆栈资源已偏离，并且整个堆栈被视为已因此偏离。

   ```
   aws cloudformation describe-stack-drift-detection-status --stack-drift-detection-
   id 624af370-311a-11e8-b6b7-500cexample
   ```

   输出：

   ```
   {
       "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-with-
   resource-drift/489e5570-df85-11e7-a7d9-50example",
       "StackDriftDetectionId": "624af370-311a-11e8-b6b7-500cexample",
       "StackDriftStatus": "DRIFTED",
       "Timestamp": "2018-03-26T17:23:22.279Z",
       "DetectionStatus": "DETECTION_COMPLETE",
       "DriftedStackResourceCount": 1
   }
   ```

3. 堆栈偏差检测操作完成后，使用 describe-stack-resource-drifts 命令可查看结果，包括已偏离的资源的实际和预期属性值。

以下示例使用 `--stack-resource-drift-status-filters` 选项来请求已修改或删除的资源的堆栈偏差信息。该请求返回有关一个已修改资源的信息，包括其值已更改的两个属性的详细信息。不会删除任何资源。

```
aws cloudformation describe-stack-resource-drifts --stack-name my-stack-with-resource-drift --stack-resource-drift-status-filters MODIFIED DELETED
```

输出：

```
{
    "StackResourceDrifts": [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
            "ActualProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,\"DelaySeconds\":120,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",\"maxReceiveCount\":12},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-Queue-494PBHCO76H4\"}",
            "ResourceType": "AWS::SQS::Queue",
            "Timestamp": "2018-03-26T17:23:34.489Z",
            "PhysicalResourceId": "https://sqs.us-east-1.amazonaws.com/099908667365/my-stack-with-resource-drift-Queue-494PBHCO76H4",
            "StackResourceDriftStatus": "MODIFIED",
            "ExpectedProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,\"DelaySeconds\":20,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",\"maxReceiveCount\":10},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-Queue-494PBHCO76H4\"}",
            "PropertyDifferences": [
                {
                    "PropertyPath": "/DelaySeconds",
                    "ActualValue": "120",
                    "ExpectedValue": "20",
                    "DifferenceType": "NOT_EQUAL"
                },
                {
                    "PropertyPath": "/RedrivePolicy/maxReceiveCount",
```

```
                    "ActualValue": "12",
                    "ExpectedValue": "10",
                    "DifferenceType": "NOT_EQUAL"
                }
            ],
            "LogicalResourceId": "Queue"
        }
    ]
}
```

在单个堆栈资源上检测偏差

您可以在堆栈中的特定资源上检测偏差，而不是在整个堆栈上。当您只需要再次确定特定资源现在是否与其预期的模板配置匹配时，这尤其有用。

当在资源上执行偏差检测时，CloudFormation 还会更新整体堆栈偏差状态和上一次偏差检查时间（如果适用）。例如，假设堆栈的偏差状态为 IN_SYNC。您让 CloudFormation 对该堆栈中包含的一个或多个资源执行偏差检测，CloudFormation 检测到这些资源中的一个或多个已偏离。CloudFormation 将堆栈偏差状态更新为 DRIFTED。相反，假设您有一个偏差状态为 DRIFTED 的堆栈，因为只有一个偏离的资源。如果您将该资源设置回其预期的属性值，然后再次在该资源上检测偏差，则 CloudFormation 会将资源偏差状态和堆栈偏差状态更新为 IN_SYNC，而无需您再次在整个堆栈上检测偏差。

使用 Amazon Web Services 管理控制台 在单个资源上检测偏差

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 从堆栈列表中，选择包含所需资源的堆栈。CloudFormation 会显示该堆栈的堆栈详细信息。
3. 在左侧导航窗格中的堆栈下，选择堆栈操作，然后选择检测偏差。
4. 在 Resource drift status (资源偏差状态) 下面，选择所需资源，然后选择 Detect drift for resource (检测资源偏差)。

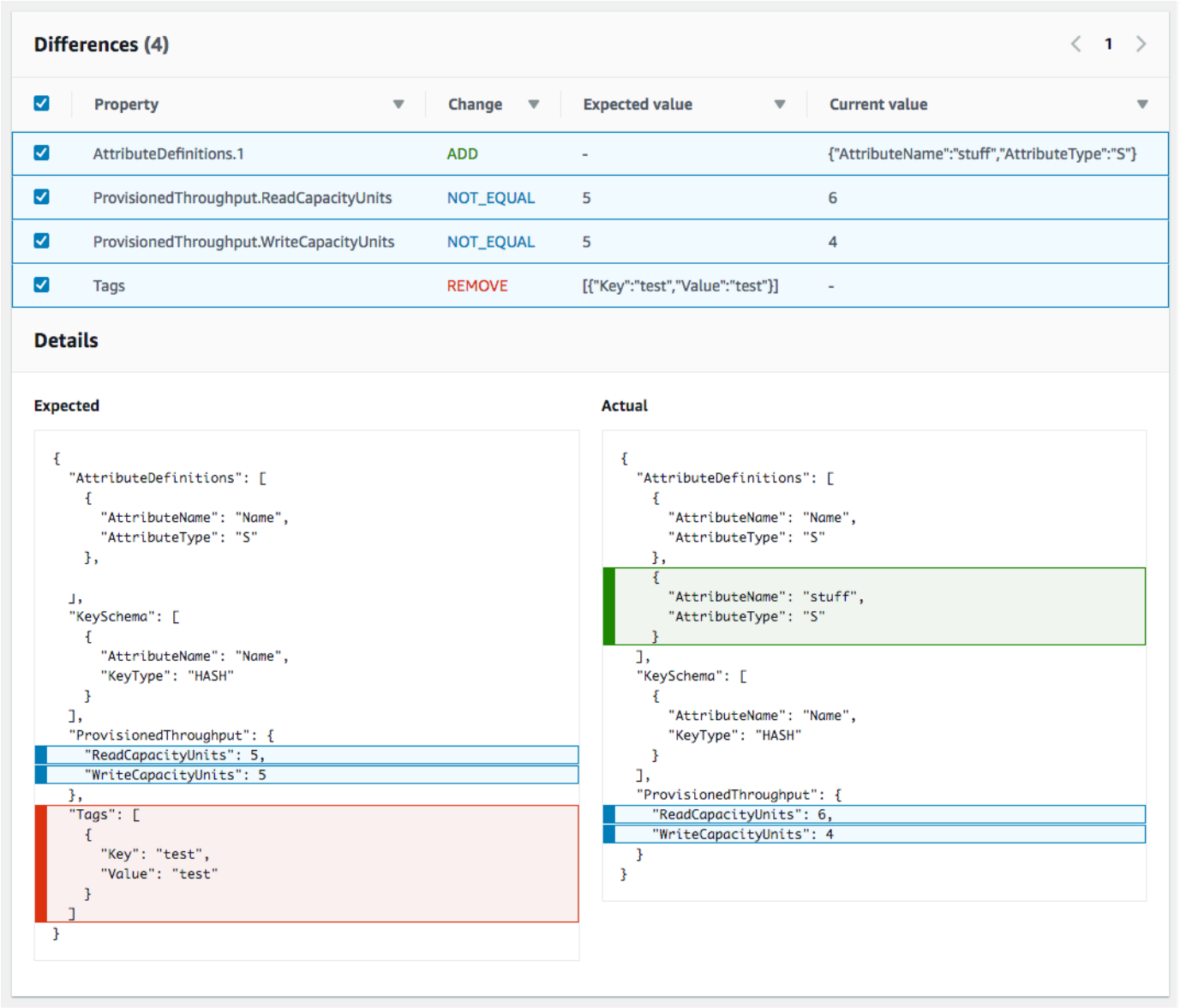
   CloudFormation 在所选资源上执行偏差检测。如果成功，CloudFormation 会更新资源的偏差状态，并在必要时更新整体堆栈偏差状态。CloudFormation 还会更新上次对资源以及整个堆栈进行偏差检测时的时间戳。如果资源已被修改，则 CloudFormation 会显示有关资源的预期和当前属性值的详细偏差信息。
5. 查看资源的偏差检测结果。

- 查看已修改资源的详细信息。
  - 选定已修改的资源，选择 View drift details（查看偏差详细信息）。

    CloudFormation 显示该资源的偏差详细信息，包括资源的预期和当前属性值，以及两者之间的任何区别。

    要突出显示区别，请在区别部分中选择属性名称。

    - 添加的属性在 Details（详细信息）部分的 Current（当前）列中以绿色突出显示。
    - 删除的属性在 Details（详细信息）部分的 Expected（预期）列中以红色突出显示。
    - 其值已更改的属性在 Expected (预期) 和 Current (当前) 列中以黄色突出显示。



使用 Amazon CLI 在单个资源上检测偏差

-

> **Important**
>
> 查看堆栈资源的上次偏差检查时间，并确认它早于资源偏差结果中显示的时间戳，以防止使用过时的数据。

要在使用 Amazon CLI 的单个资源上检测偏差，使用 detect-stack-resource-drift 命令。指定资源的逻辑 ID，以及包含它的堆栈。

以下示例在特定堆栈资源 my-drifted-resource 上运行偏差检测操作。该响应返回确认资源已被修改的信息，包括其值已更改的两个属性的详细信息。

```
aws cloudformation detect-stack-resource-drift \
    --stack-name my-stack-with-resource-drift \
    --logical-resource-id my-drifted-resource
```

输出：

```
{
    "StackResourceDrift": {
        "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-
with-resource-drift/489e5570-df85-11e7-a7d9-50example",
        "ActualProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,\"DelaySeconds
\":120,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",
\"maxReceiveCount\":12},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize
\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-
Queue-494PBHCO76H4\"}",
        "ResourceType": "AWS::SQS::Queue",
        "Timestamp": "2018-03-26T18:54:28.462Z",
        "PhysicalResourceId": "https://sqs.us-east-1.amazonaws.com/099908667365/my-
stack-with-resource-drift-Queue-494PBHCO76H4",
        "StackResourceDriftStatus": "MODIFIED",
        "ExpectedProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,
\"DelaySeconds\":20,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",
\"maxReceiveCount\":10},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize
\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-
Queue-494PBHCO76H4\"}",
        "PropertyDifferences": [
            {
                "PropertyPath": "/DelaySeconds",
                "ActualValue": "120",
                "ExpectedValue": "20",
                "DifferenceType": "NOT_EQUAL"
            },
            {
```

```
                "PropertyPath": "/RedrivePolicy/maxReceiveCount",
                "ActualValue": "12",
                "ExpectedValue": "10",
                "DifferenceType": "NOT_EQUAL"
            }
        ],
        "LogicalResourceId": "my-drifted-resource"
    }
}
```

通过导入操作消除偏差

在某些情况下，资源的配置可能已偏离其预期配置，而您希望接受新配置作为预期配置。在大多数情况下，您可以通过使用新配置更新堆栈模板中的资源定义来消除偏差结果，然后执行堆栈更新。但是，如果新配置更新了需要替换的资源属性，则在堆栈更新期间将重新创建该资源。如果要保留现有资源，则可以使用资源导入功能来更新资源并消除偏差结果，而不会导致替换资源。

通过导入操作消除资源偏差的过程包括以下基本步骤：

- 将设置为 Retain 的 DeletionPolicy 属性添加到资源中。这可确保在从堆栈中移除现有资源时，保留而不是删除该资源。
- 从模板中移除资源并运行堆栈更新操作。这将从堆栈中移除资源，但不会删除它。
- 描述资源在堆栈模板中的实际状态，然后将现有资源导入回堆栈。这会将资源添加回堆栈中，并消除导致了偏差结果的属性差异。

有关资源导入的更多信息，请参阅手动将 Amazon 资源导入 CloudFormation 堆栈。有关支持导入的资源列表，请参阅资源类型支持。

在此示例中，我们使用以下名为 `templateToImport.json` 的模板。

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
         "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
```

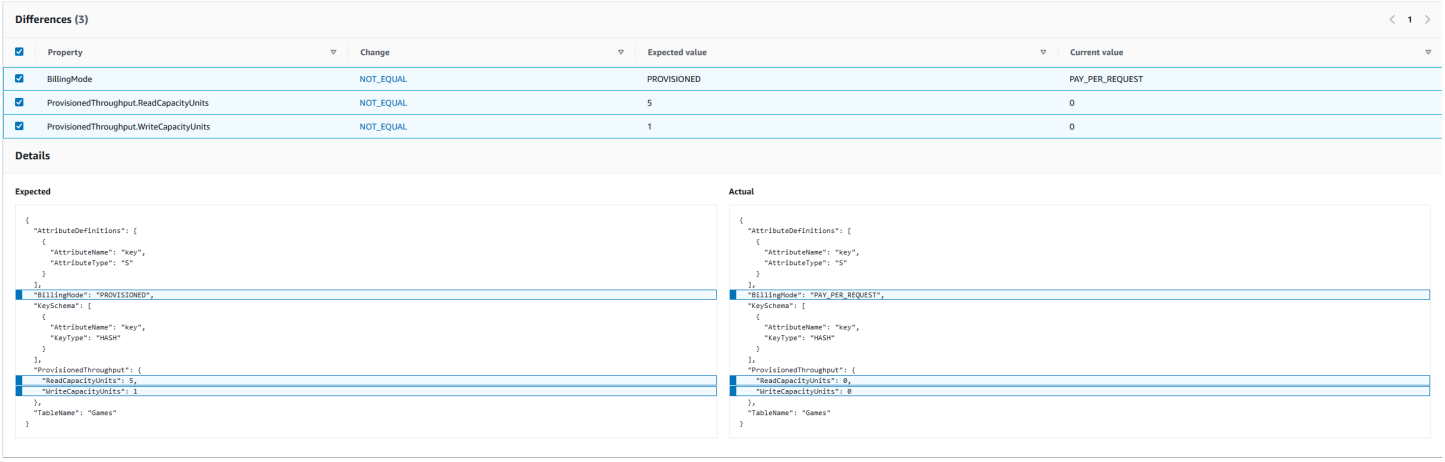
```
            "AttributeDefinitions":[
               {
                  "AttributeName":"key",
                  "AttributeType":"S"
               }
            ],
            "KeySchema":[
               {
                  "AttributeName":"key",
                  "KeyType":"HASH"
               }
            ],
            "BillingMode": "PROVISIONED",
            "ProvisionedThroughput":{
               "ReadCapacityUnits":5,
               "WriteCapacityUnits":1
            }
         }
      },
      "GamesTable": {
          "Type": "AWS::DynamoDB::Table",
          "Properties": {
              "TableName": "Games",
              "AttributeDefinitions": [
                  {
                      "AttributeName": "key",
                      "AttributeType": "S"
                  }
              ],
              "KeySchema": [
                  {
                      "AttributeName": "key",
                      "KeyType": "HASH"
                  }
              ],
              "BillingMode": "PROVISIONED",
              "ProvisionedThroughput": {
                  "ReadCapacityUnits": 5,
                  "WriteCapacityUnits": 1
              }
          }
      }
  }
```

```
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  GamesTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Games
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
```

在此示例中，我们假设用户更改了 CloudFormation 之外的资源。运行偏差检测后，我们发现 `GamesTable` 的 `BillingMode` 已修改为 `PAY_PER_REQUEST`。有关偏差检测的更多信息，请参阅[使用偏移检测功能检测堆栈和资源的非托管配置更改]。



我们的堆栈现在已经过时，我们的资源处于活动状态，但我们希望保留预期的资源配置。为此，我们可以通过导入操作来消除偏差，而不会中断服务。

使用 CloudFormation 控制台通过导入操作消除偏差

步骤 1：使用 Retain 删除策略更新堆栈

使用带有 **`Retain`** 选项的 **`DeletionPolicy`** 属性更新堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面上，选择已偏差的堆栈。
3. 选择更新，然后从堆栈详细信息窗格中选择替换当前模板。
4. 在指定模板页面上，使用下列方法之一提供更新的模板，其中包含带有 `Retain` 选项的 `DeletionPolicy` 属性：

   - 选择 Amazon S3 URL，然后在文本框中指定您的模板的 URL。
   - 选择 Upload a template file (上传模板文件)，然后浏览您的模板。

   然后选择下一步。
5. 查看指定堆栈详细信息页面，然后选择下一步。
6. 查看配置堆栈选项页面，然后选择下一步。
7. 在查看 ***`stack-name`*** 页面上，选择更新堆栈。

结果：在堆栈的事件页面上，状态为 `UPDATE_COMPLETE`。

要在不中断服务的情况下通过导入操作消除偏差，请为要从堆栈中移除的资源指定 Retain DeletionPolicy。在以下示例中，我们向 GamesTable 资源添加了一个设置为 Retain 的 DeletionPolicy 属性。

Example JSON

```
  "GamesTable": {
      "Type": "AWS::DynamoDB::Table",
      "DeletionPolicy": "Retain",
      "Properties": {
          "TableName": "Games",
```

Example YAML

```
GamesTable:
  Type: AWS::DynamoDB::Table
  DeletionPolicy: Retain
  Properties:
    TableName: Games
```

步骤 2：移除偏差的资源、相关参数和输出

移除偏差的资源、相关参数和输出

1. 选择更新，然后从堆栈详细信息窗格中选择替换当前模板。
2. 在指定模板页面上，使用下列方法之一提供更新的模板，该模板的资源、相关参数和输出已从堆栈模板中移除：

   - 选择 Amazon S3 URL，然后在文本框中指定您的模板的 URL。
   - 选择 Upload a template file (上传模板文件)，然后浏览您的模板。

   然后选择下一步。
3. 查看指定堆栈详细信息页面，然后选择下一步。
4. 查看配置堆栈选项页面，然后选择下一步。
5. 在查看 ***stack-name*** 页面上，选择更新堆栈。

结果：逻辑 ID GamesTable 在堆栈的事件页面上具有状态 DELETE_SKIPPED。

等待 CloudFormation 完成堆栈更新操作。堆栈更新操作完成后，从堆栈模板中移除资源、相关参数和输出。然后，导入更新后的模板。完成这些操作后，示例模板现在如下所示。

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
         "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "BillingMode": "PROVISIONED",
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

Example YAML

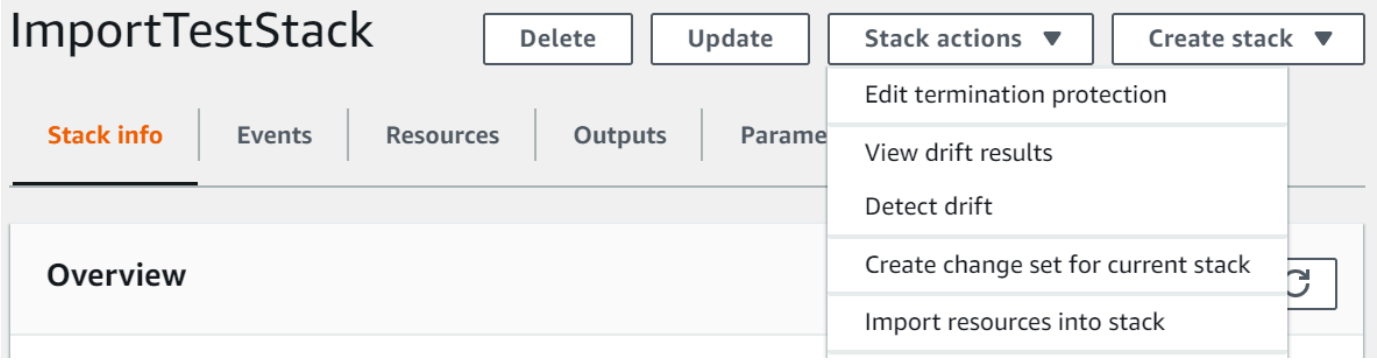
```
AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
```

```
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
```

步骤 3：更新模板以匹配资源的实时状态

更新模板以匹配资源的实时状态

1. 要导入更新后的模板，请选择堆栈操作，然后选择将资源导入到堆栈。



2. 在导入概述页面上，查看在该操作期间需要提供的内容列表，然后选择下一步。
3. 在指定模板页面上，使用下列方法之一提供更新的模板：

   - 选择 Amazon S3 URL，然后在文本框中指定您的模板的 URL。
   - 选择 Upload a template file (上传模板文件)，然后浏览您的模板。

   然后选择下一步。
4. 在 Identify resources（标识资源）页面上，标识每个目标资源。有关更多信息，请参阅 资源标识符。

   a. 在 Identifier property (标识符属性) 下面，选择资源标识符的类型。例如，`TableName` 属性标识 `AWS::DynamoDB::Table` 资源。

b. 在标识符值下，输入实际属性值。在示例模板中，GamesTable 资源的 TableName 为 Games。

c. 选择下一步。

5. 查看指定堆栈详细信息页面，然后选择下一步。

6. 在导入概述页面上，查看正在导入的资源，然后选择导入资源。这会将 AWS::DynamoDB::Table 资源类型导入回您的堆栈中。

结果：在此示例中，我们通过导入操作消除了资源偏差，而不会中断服务。您可以在 CloudFormation 控制台的“Events (事件)”选项卡中检查导入操作的进度。导入的资源将具有 IMPORT_COMPLETE 状态，后跟 CREATE_COMPLETE 状态，并且资源导入完成作为状态原因。

等待 CloudFormation 完成堆栈更新操作。堆栈更新操作完成后，更新模板以匹配资源的实际偏差状态。例如，BillingMode 将设置为 PAY_PER_REQUEST，ReadCapacityUnits 和 WriteCapacityUnits 将设置为 0。

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
         "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "BillingMode": "PROVISIONED",
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
```

```
                "WriteCapacityUnits":1
              }
           }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "BillingMode": "PAY_PER_REQUEST",
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 0,
                    "WriteCapacityUnits": 0
                }
            }
        }
    }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
```

```
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  GamesTable:
    Type: AWS::DynamoDB::Table
    DeletionPolicy: Retain
    Properties:
      TableName: Games
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PAY_PER_REQUEST
      ProvisionedThroughput:
        ReadCapacityUnits: 0
        WriteCapacityUnits: 0
```

将 Amazon 资源导入 CloudFormation 堆栈

您可以将现有资源导入 CloudFormation 堆栈。如果您想在无需删除和重新创建相关资源的情况下，开始使用 CloudFormation 来管理在 CloudFormation 之外创建的资源，这将非常有用。

CloudFormation 提供了以下用于将现有资源导入堆栈的选项：

- IaC 生成器是一种导入工具，它可自动扫描现有资源并根据其当前状态生成 CloudFormation 模板，然后可以使用此模板将这些资源导入堆栈。
- 资源导入是一种手动过程，您可以在其中描述 CloudFormation 模板中的现有资源，然后将其导入堆栈。这种方法要求您在模板中手动指定资源属性和配置。
- 自动导入是一个自动过程，您可以在 CloudFormation 模板中描述现有资源，然后 CloudFormation 将具有匹配自定义名称的资源导入到堆栈中。
- 堆栈重构功能可简化 CloudFormation 堆栈中资源的重新整理，同时仍保留现有资源属性和数据。借助堆栈重构功能，您可以在堆栈之间移动资源，将整体堆栈拆分为较小的组件，或者将多个堆栈整合为一个。

除将现有资源纳入 CloudFormation 管理范围之外，资源导入功能对于下列场景也非常实用：

- 在堆栈之间移动资源 – 您可以将资源从一个堆栈导入另一个堆栈，从而可以根据需要重新组织基础设施。
- 嵌套现有堆栈 – 您可以将现有堆栈作为嵌套堆栈导入另一个堆栈，从而实现模块化和可重复使用的基础设施设计。

CloudFormation 支持导入广泛的资源。有关更多信息，请参阅 资源类型支持。

主题

- 手动将 Amazon 资源导入 CloudFormation 堆栈
- 自动将 Amazon 资源导入 CloudFormation 堆栈
- 恢复导入操作

手动将 Amazon 资源导入 CloudFormation 堆栈

借助资源导入功能，您可以将现有 Amazon 资源导入到新的或现有的 CloudFormation 堆栈中。在导入操作期间，您创建一个更改集以将现有资源导入到堆栈中，或者从现有资源中创建新的堆栈。您在导入期间提供以下内容。

- 描述整个堆栈的模板，包括原始堆栈资源和您要导入的资源。每个要导入的资源都必须具有 DeletionPolicy 属性。
- 要导入的资源的标识符，CloudFormation 可以使用这些标识符将模板中的逻辑 ID 与现有资源进行映射。

> **Note**
>
> 使用资源导入功能时，CloudFormation 仅支持一层嵌套。这意味着，您无法将堆栈导入到子堆栈中，也无法导入具有子堆栈的堆栈。

主题

- 资源标识符
- 资源导入验证
- 资源导入状态代码

- 导入操作期间的注意事项
- 其他 资源
- 从现有的资源中创建堆栈
- 将现有的资源导入到堆栈
- 在堆栈之间移动资源
- 嵌套现有的堆栈

资源标识符

提供两个值来标识正在导入的每个资源。

- 标识符属性。这是一个资源属性，可用于标识每种资源类型。例如，可以使用 `BucketName` 标识 `AWS::S3::Bucket` 资源。

  用于标识要导入的资源的资源属性因资源类型而异。可在 CloudFormation 控制台中找到资源属性。创建包含要导入的资源的模板后，可以启动导入过程，您可以在其中找到要导入的资源的标识符属性。某些资源类型可能存在多种标识方式，您可以在下拉列表中选择要使用的属性。

  或者，您也可以通过调用 get-template-summary CLI 命令并将堆栈模板的 S3 URL 指定为 `--template-url` 选项的值，获取要导入的资源的标识符属性。
- 标识符值。这是资源的实际属性值。例如，`BucketName` 属性的实际值可能是 `MyS3Bucket`。

  可从资源的服务控制台获取标识符属性的值。

资源导入验证

在导入操作期间，CloudFormation 执行以下验证。

- 要导入的资源存在。
- 每个要导入的资源的属性和配置值符合资源类型架构要求，该架构定义了它接受的属性、必需的属性和支持的属性值。
- 在模板中指定了必需的属性。每种资源类型的必需属性如 Amazon 资源和属性类型参考中所述。
- 要导入的资源不属于同一区域中的其他堆栈。此检查按区域进行，但某些资源类型不是特定于区域的，例如 IAM 角色、Route 53 托管区域或 CloudFront 分发。对于这些类型，在导入资源之前，请确保该资源尚未由其他区域中的堆栈托管。

CloudFormation 不检查模板配置与资源属性的实际配置是否匹配。

> **Important**
>
> 验证模板中定义的资源及其属性是否与资源导入的预期配置相匹配，以避免意外更改。

资源导入状态代码

下表描述了可与资源导入功能一起使用的各种状态类型。

导入操作状态	说明
IMPORT_IN_PROGRESS	正在执行导入操作。
IMPORT_COMPLETE	堆栈中所有资源的导入操作均已完成。
IMPORT_ROLLBACK_IN_PROGRESS	回滚导入操作正在回滚以前的模板配置。
IMPORT_ROLLBACK_FAILED	导入回滚操作失败。
IMPORT_ROLLBACK_COMPLETE	导入已回滚到以前的模板配置。

导入操作期间的注意事项

- 在导入完成后以及执行后续堆栈操作之前，我们建议对导入的资源运行偏差检测。偏差检测确保模板配置与实际配置匹配。有关更多信息，请参阅 在整个 CloudFormation 堆栈上检测偏差。
- 导入操作不允许创建新资源，删除资源或更改属性配置。
- 要导入的每个资源必须具有 DeletionPolicy 属性，才能成功完成导入操作。DeletionPolicy 可以设置为任何可能的值。只有要导入的资源才需要 DeletionPolicy。已属于堆栈的资源不需要使用 DeletionPolicy。
- 您无法将同一资源导入同一区域内的多个堆栈。
- 您可以使用 cloudformation:ImportResourceTypes IAM policy 条件控制用户可以在导入操作期间使用的资源类型。有关更多信息，请参阅 CloudFormation 的策略条件键。

- CloudFormation 堆栈限制在导入资源时适用。有关限制的更多信息，请参阅 了解 CloudFormation 配额。

其他 资源

要通过资源导入解决堆栈偏差问题，请参阅通过导入操作消除偏差。

从现有的资源中创建堆栈

本主题演示了如何通过在模板中描述现有的 Amazon 资源，从而利用这些资源创建堆栈。要改为扫描现有的资源并自动生成模板，以用于将现有资源导入 CloudFormation 或在新账户中复制资源，请参阅使用 IaC 生成器为现有资源生成模板。

先决条件

在开始之前，您必须具有以下内容：

- 描述新堆栈中要包含的所有资源的模板。您可将模板保存在本地或 Amazon S3 存储桶中。
- 对于要导入的每个资源，请包括以下内容：
  - 定义资源当前配置的属性和属性值。
  - 资源的唯一标识符，例如资源名称。有关更多信息，请参阅 资源标识符。
  - DeletionPolicy 属性。

主题

- 示例 模板
- 使用 Amazon Web Services 管理控制台 从现有的资源中创建堆栈
- 使用 Amazon CLI 从现有的资源中创建堆栈

示例 模板

在本演练中，我们假设您使用以下名为 `TemplateToImport.json` 的示例模板，该模板指定了在 CloudFormation 之外创建的两个 DynamoDB 表。`ServiceTable` 和 `GamesTable` 都是要导入的目标。

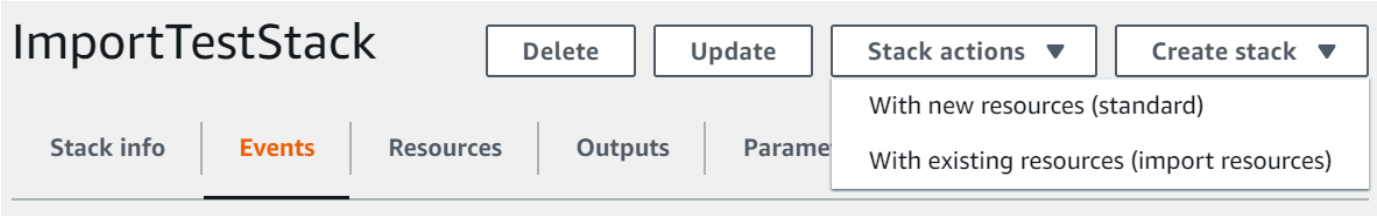
> **Note**
>
> 此模板仅作为示例。要将其用于自己的测试目的，请使用您账户中的资源替换示例资源。

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Service",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
```

```
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

使用 Amazon Web Services 管理控制台 从现有的资源中创建堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在 Stacks (堆栈) 页面上，选择 Create stack (创建堆栈)，然后选择 With existing resources (import resources) (使用现有的资源 (导入资源))。

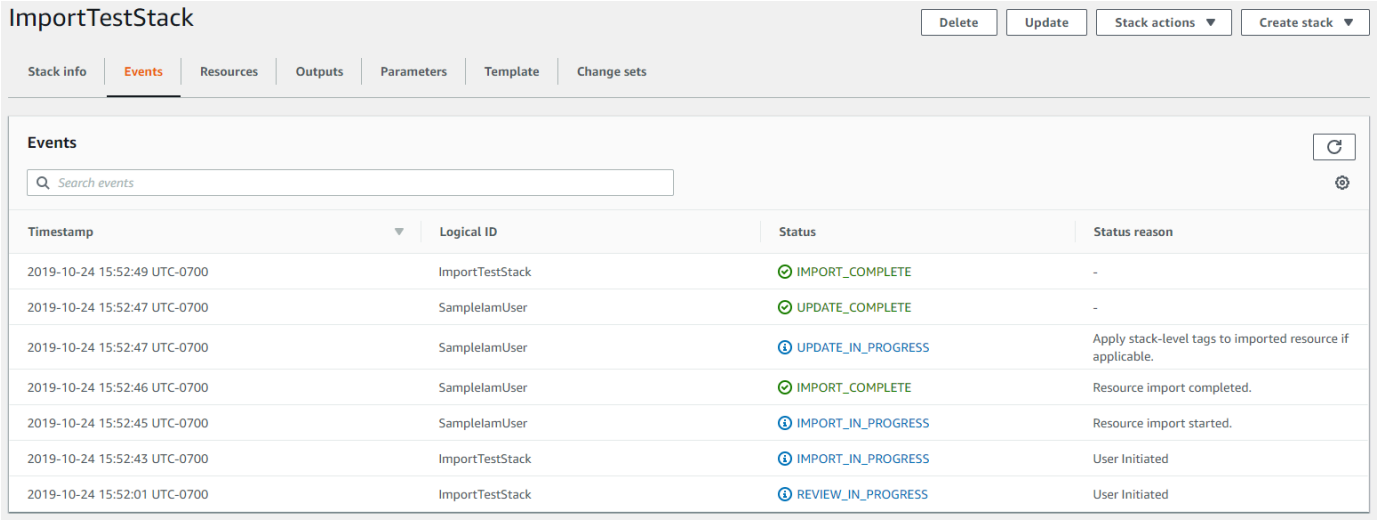


3. 访问 Import overview (导入概述) 页面，以查看在该操作期间需要提供的内容列表。然后选择下一步。
4. 在 Specify template (指定模板) 页面上，使用以下方法之一提供模板，然后选择 Next (下一步)。
   - 选择 Amazon S3 URL，然后在文本框中指定您的模板的 URL。
   - 选择 Upload a template file (上传模板文件)，然后浏览您的模板。
5. 在 Identify resources（标识资源）页面上，标识每个目标资源。有关更多信息，请参阅 资源标识符。
   a. 在 Identifier property (标识符属性) 下面，选择资源标识符的类型。例如，可以使用 `TableName` 属性标识 `AWS::DynamoDB::Table` 资源。
   b. 在 Identifier value (标识符值) 下面，键入实际属性值。例如，示例模板中的 `GamesTable` 资源的 `TableName` 为 *`Games`*。
   c. 选择下一步。
6. 在 Specify stack details (指定堆栈详细信息) 页面上，修改任何参数，然后选择 Next (下一步)。这会自动创建一个更改集。

> **Important**
>
> 如果您修改的现有参数启动创建、更新或删除操作，导入操作将失败。

7. 在 Review ***stack-name*** (查看 <堆栈名称>) 页面上，确认正在导入正确的资源，然后选择 Import resources (导入资源)。这会自动执行在上一步中创建的更改集。

   将显示新堆栈的 Stack details（堆栈详细信息）页面的 Events（事件）窗格。



8. （可选）对堆栈运行偏差检测，以确保导入的资源的模板和实际配置匹配。有关检测偏差的更多信息，请参阅 在整个 CloudFormation 堆栈上检测偏差。

9. （可选）如果导入的资源与预期的模板配置不匹配，请更正模板配置或直接更新资源。在本演练中，我们更正模板配置以与其实际配置匹配。

   a. 为受影响的资源恢复导入操作。

   b. 再次将导入目标添加到模板中，从而确保模板配置与实际配置匹配。

   c. 使用修改的模板重复步骤 2 - 8，以再次导入资源。

使用 Amazon CLI 从现有的资源中创建堆栈

1. 要了解哪些属性标识模板中的每种资源类型，请运行 get-template-summary 命令并指定模板的 S3 URL。例如，可以使用 `TableName` 属性标识 `AWS::DynamoDB::Table` 资源。对于示例模板中的 `GamesTable` 资源，`TableName` 的值为 `Games`。您在下一步中需要此信息。

```
aws cloudformation get-template-summary \
```

```
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json
```

有关更多信息，请参阅 资源标识符。

2. 按照以下 JSON 字符串格式，编写模板中的实际资源及其唯一标识符的列表。

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifie
{"TableName":"Games"}},
{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"ServiceTable","ResourceIdentifi
{"TableName":"Service"}}]
```

或者，您可以在配置文件中指定 JSON 格式参数。

例如，要导入 ServiceTable 和 GamesTable，可以创建一个包含以下配置的 *ResourcesToImport.txt* 文件。

```
[
   {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"GamesTable",
      "ResourceIdentifier":{
         "TableName":"Games"
      }
   },
   {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"ServiceTable",
      "ResourceIdentifier":{
         "TableName":"Service"
      }
   }
]
```

3. 要创建更改集，请使用以下 create-change-set 命令并替换占位符文本。对于 --change-set-type 选项，请指定 **IMPORT** 的值。对于 --resources-to-import 选项，将示例 JSON 字符串替换为您刚刚创建的实际 JSON 字符串。

```
aws cloudformation create-change-set \
    --stack-name TargetStack --change-set-name ImportChangeSet \
    --change-set-type IMPORT \
```

```
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json \
    --resources-to-
import '[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceI
{"TableName":"Games"}},
{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"ServiceTable","ResourceIdentifi
{"TableName":"Service"}}]'
```

> **Note**
>
> `--resources-to-import` 不支持内联 YAML。JSON 字符串中对转义引号的要求因终端而异。有关更多信息，请参阅《Amazon Command Line Interface User Guide》中的 Using quotation marks inside strings。

或者，您可以使用文件 URL 作为 `--resources-to-import` 选项的输入内容，如以下示例所示。

```
--resources-to-import file://ResourcesToImport.txt
```

4. 查看更改集，以确保将导入正确的资源。

```
aws cloudformation describe-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

5. 要启动更改集并导入资源，请使用以下 execute-change-set 命令并替换占位符文本。在成功完成该操作后（`IMPORT_COMPLETE`），将成功导入资源。

```
aws cloudformation execute-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

6. （可选）对 `IMPORT_COMPLETE` 堆栈运行偏差检测，以确保导入的资源的模板和实际配置匹配。有关检测偏差的更多信息，请参阅 在单个堆栈资源上检测偏差。

   a. 对指定堆栈运行偏差检测。

   ```
   aws cloudformation detect-stack-drift --stack-name TargetStack
   ```

   如果成功，此命令将返回以下示例输出：

```
{ "Stack-Drift-Detection-Id" : "624af370-311a-11e8-b6b7-500cexample" }
```

b. 查看指定堆栈偏差检测 ID 的偏差检测操作进度。

```
aws cloudformation describe-stack-drift-detection-status \
    --stack-drift-detection-id 624af370-311a-11e8-b6b7-500cexample
```

c. 查看已在指定堆栈中检查偏差的资源的偏差信息。

```
aws cloudformation describe-stack-resource-drifts --stack-name TargetStack
```

7. （可选）如果导入的资源与预期的模板配置不匹配，请更正模板配置或直接更新资源。在本演练中，我们更正模板配置以与其实际配置匹配。

   a. 为受影响的资源恢复导入操作。

   b. 再次将导入目标添加到模板中，从而确保模板配置与实际配置匹配。

   c. 使用修改的模板重复步骤 3 - 6，以再次导入资源。

将现有的资源导入到堆栈

本主题演示了如何通过在模板中描述现有的 Amazon 资源，从而将其导入现有的堆栈。要改为扫描现有的资源并自动生成模板，以用于将现有资源导入 CloudFormation 或在新账户中复制资源，请参阅 使用 IaC 生成器为现有资源生成模板。

先决条件

在开始之前，您必须具有以下内容：

- 描述整个堆栈的模板，包括堆栈已包含的资源和要导入的资源。您可将模板保存在本地或 Amazon S3 存储桶中。

  获取正在运行的堆栈的模板副本

  1. 通过以下网址打开 CloudFormation 控制台：https://console.aws.amazon.com/cloudformation/。
  2. 从堆栈列表中，选择要从中检索模板的堆栈。
  3. 在堆栈详细信息窗格中，选择模板选项卡，然后选择复制到剪贴板。
  4. 将代码粘贴到文本编辑器中，从而开始向模板添加其他资源。

- 对于要导入的每个资源，请包括以下内容：
  - 定义资源当前配置的属性和属性值。
  - 资源的唯一标识符，例如资源名称。有关更多信息，请参阅 资源标识符。
  - DeletionPolicy 属性。

主题

- 示例 模板
- 使用 Amazon Web Services 管理控制台 将现有资源导入到堆栈
- 使用 Amazon CLI 将现有资源导入到堆栈

示例 模板

在本演练中，我们假设您使用以下名为 TemplateToImport.json 的示例模板，该模板指定了两个 DynamoDB 表。ServiceTable 当前是该堆栈的一部分，而 GamesTable 是您要导入的表。

> **Note**
>
> 此模板仅作为示例。要将其用于自己的测试目的，请使用您账户中的资源替换示例资源。

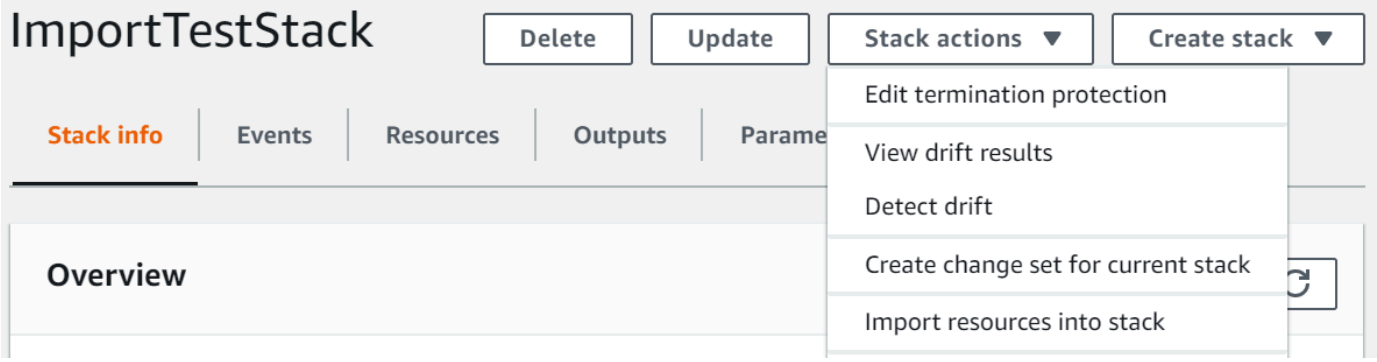
```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "TableName": "Service",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
```

```
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

使用 Amazon Web Services 管理控制台 将现有资源导入到堆栈

> **Note**
>
> 在导入资源时，CloudFormation 控制台不支持使用内置函数 `Fn::Transform`。您可以使用 Amazon CLI 导入使用 `Fn::Transform` 函数的资源。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在 Stacks (堆栈) 页面上，选择要将资源导入到的堆栈。
3. 选择 Stack actions (堆栈操作)，然后选择 Import resources into stack (将资源导入到堆栈)。



4. 查看 Import overview (导入概述) 页面，然后选择 Next (下一步)。
5. 在 Specify template (指定模板) 页面上，使用以下方法之一提供更新的模板，然后选择 Next (下一步)。

   - 选择 Amazon S3 URL，然后在文本框中指定您的模板的 URL。
   - 选择 Upload a template file (上传模板文件)，然后浏览您的模板。

6. 在 Identify resources (标识资源) 页面上，标识每个目标资源。有关更多信息，请参阅 资源标识符。

   a. 在 Identifier property (标识符属性) 下面，选择资源标识符的类型。例如，可以使用 `TableName` 属性标识 `AWS::DynamoDB::Table` 资源。

   b. 在 Identifier value (标识符值) 下面，键入实际属性值。例如，示例模板中的 `GamesTable` 资源的 `TableName` 为 *`Games`*。

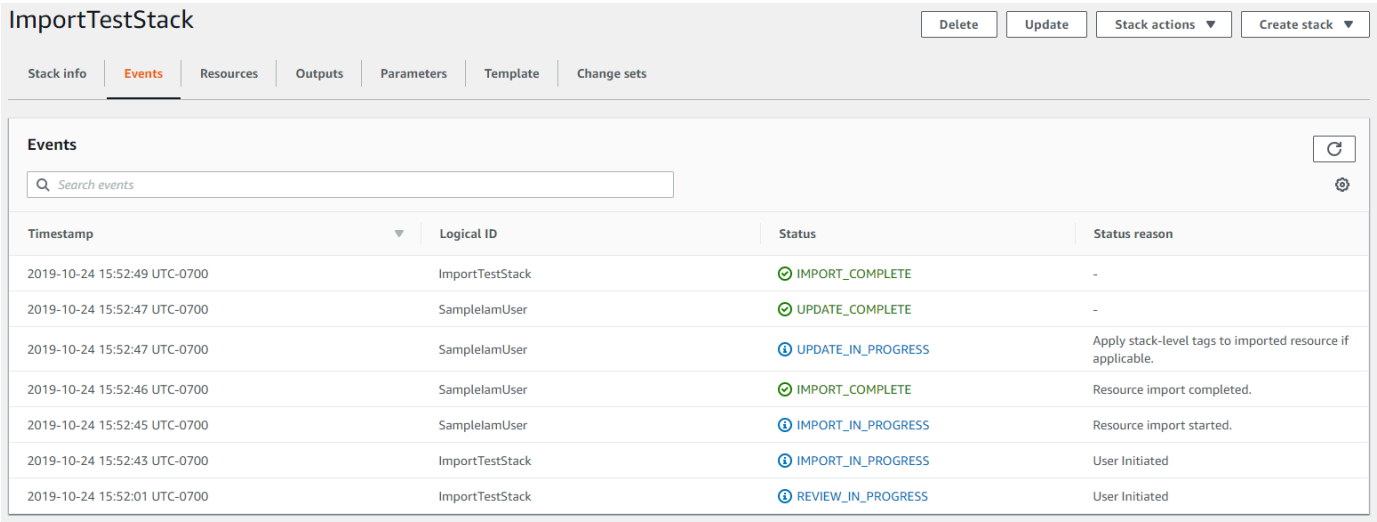
   c. 选择下一步。

7. 在 Specify stack details (指定堆栈详细信息) 页面上，更新任何参数，然后选择 Next (下一步)。这会自动创建一个更改集。

   > **Note**
   >
   > 如果您修改的现有参数启动创建、更新或删除操作，导入操作将失败。

8. 在 Review ***`stack-name`*** (查看 <堆栈名称>) 页面上，查看要导入的资源，然后选择 Import resources (导入资源)。这会自动执行在上一步中创建的更改集。此时，任何堆栈级标签将应用于导入的资源。有关更多信息，请参阅 配置堆栈选项。

将显示堆栈的 Events (事件) 页面。



9. （可选）对堆栈运行偏差检测，以确保导入的资源的模板和实际配置匹配。有关检测偏差的更多信息，请参阅 [在整个 CloudFormation 堆栈上检测偏差]。

10. （可选）如果导入的资源与预期的模板配置不匹配，请更正模板配置或直接更新资源。有关导入偏差的资源的更多信息，请参阅 [通过导入操作消除偏差]。

使用 Amazon CLI 将现有资源导入到堆栈

1. 要了解哪些属性标识模板中的每种资源类型，请运行 get-template-summary 命令并指定模板的 S3 URL。例如，可以使用 `TableName` 属性标识 `AWS::DynamoDB::Table` 资源。对于示例模板中的 `GamesTable` 资源，`TableName` 的值为 `Games`。您在下一步中需要此信息。

```
aws cloudformation get-template-summary \
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json
```

有关更多信息，请参阅 [资源标识符]。

2. 按照以下 JSON 字符串格式，编写要导入的实际资源及其唯一标识符的列表。

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifie
{"TableName":"Games"}}]
```

或者，您可以在配置文件中指定 JSON 格式参数。

例如，要导入 `GamesTable`，可以创建一个包含以下配置的 *`ResourcesToImport.txt`* 文件。

```
[
  {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"GamesTable",
      "ResourceIdentifier": {
        "TableName":"Games"
      }
  }
]
```

3. 要创建更改集，请使用以下 create-change-set 命令并替换占位符文本。对于 `--change-set-type` 选项，请指定 **`IMPORT`** 的值。对于 `--resources-to-import` 选项，将示例 JSON 字符串替换为您刚刚创建的实际 JSON 字符串。

```
aws cloudformation create-change-set \
    --stack-name TargetStack --change-set-name ImportChangeSet \
    --change-set-type IMPORT \
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json \
    --resources-to-
import '[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceI
{"TableName":"Games"}}]'
```

> **Note**
>
> `--resources-to-import` 不支持内联 YAML。JSON 字符串中对转义引号的要求因终端而异。有关更多信息，请参阅《Amazon Command Line Interface User Guide》中的 Using quotation marks inside strings。

或者，您可以使用文件 URL 作为 `--resources-to-import` 选项的输入内容，如以下示例所示。

```
--resources-to-import file://ResourcesToImport.txt
```

4. 查看更改集，以确保将导入正确的资源。

```
aws cloudformation describe-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

5. 要启动更改集并导入资源，请使用以下 execute-change-set 命令并替换占位符文本。此时，任何堆栈级标签将应用于导入的资源。有关更多信息，请参阅 配置堆栈选项。在成功完成该操作后（IMPORT_COMPLETE），将成功导入资源。

```
aws cloudformation execute-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

6. （可选）对 IMPORT_COMPLETE 堆栈运行偏差检测，以确保导入的资源的模板和实际配置匹配。有关检测偏差的更多信息，请参阅 在整个 CloudFormation 堆栈上检测偏差。

   a. 对指定堆栈运行偏差检测。

   ```
   aws cloudformation detect-stack-drift --stack-name TargetStack
   ```

   如果成功，此命令将返回以下示例输出：

   ```
   { "Stack-Drift-Detection-Id" : "624af370-311a-11e8-b6b7-500cexample" }
   ```

   b. 查看指定堆栈偏差检测 ID 的偏差检测操作进度。

   ```
   aws cloudformation describe-stack-drift-detection-status \
       --stack-drift-detection-id 624af370-311a-11e8-b6b7-500cexample
   ```

   c. 查看已在指定堆栈中检查偏差的资源的偏差信息。

   ```
   aws cloudformation describe-stack-resource-drifts --stack-name TargetStack
   ```

7. （可选）如果导入的资源与预期的模板配置不匹配，请更正模板配置或直接更新资源。有关导入偏差的资源的更多信息，请参阅 通过导入操作消除偏差。

在堆栈之间移动资源

通过使用 resource import 功能，您可以在堆栈之间移动资源或重构 堆栈。您需要先在要移动的资源中添加 Retain 删除策略，以确保在从源堆栈中删除资源并将其导入到目标堆栈时保留该资源。

如果您不熟悉导入，建议您先查看 将 Amazon 资源导入 CloudFormation 堆栈 主题中的介绍信息。

> **Important**
>
> 并非所有资源都支持导入操作。在从堆栈中删除资源之前，请参阅支持导入操作的资源。如果您从堆栈中删除不支持导入操作的资源，则无法将该资源导入其他堆栈中或将其重新放回源堆栈中。

使用 Amazon Web Services 管理控制台重构堆栈

1. 在源模板中，为要移动的资源指定 `Retain` DeletionPolicy。

   在以下示例源模板中，`Games` 是该重构的目标。

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
        "GamesTable": {
```

```
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

2. 打开 CloudFormation 控制台以执行堆栈更新，从而应用删除策略。

   a. 在堆栈页面上，在选择了堆栈的情况下，选择更新。

   b. 在 Prepare template (准备模板) 下面，选择 Replace current template (替换当前模板)。

   c. 在 Specify template (指定模板) 下面，在 `GamesTable` 上为更新的源模板提供 `DeletionPolicy` 属性，然后选择 Next (下一步)。

      - 选择 Amazon S3 URL，然后在文本框中指定更新的源模板的 URL。
      - 选择 Upload a template file (上传模板文件)，然后浏览更新的源模板文件。

   d. 在 Specify stack details (指定堆栈详细信息) 页面上，不需要进行任何更改。选择下一步。

   e. 在 Configure stack options (配置堆栈选项) 页面上，不需要进行任何更改。选择下一步。

   f. 在查看 ***SourceStackName*** 页面上，检查所做的更改。如果您的模板包含 IAM 资源，请选择 I acknowledge that this template may create IAM resources (我确认此模板可创建 IAM 资源) 以指定您要使用模板中的 IAM 资源。有关在模板中使用 IAM 资源的更多信息，请参阅 使用

Amazon Identity and Access Management 控制 CloudFormation 访问权限。然后，创建更改集以更新源堆栈，或者直接更新源堆栈。

3. 从源模板中删除资源、相关参数和输出，然后将它们添加到目标模板中。

现在，源模板如下所示。

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

以下示例目标模板当前具有 PlayersTable 资源，现在还包含 GamesTable。

Example JSON

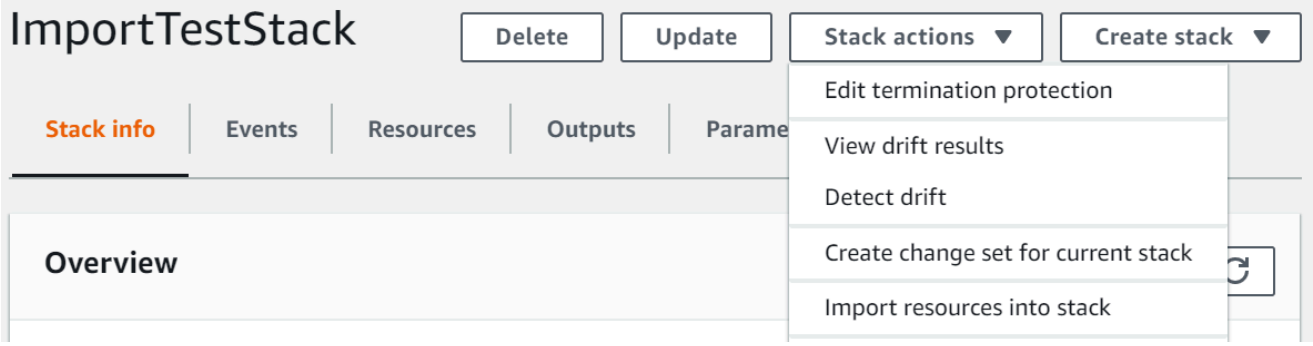
```
{
```

```
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "PlayersTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "TableName": "Players",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
```

```
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

4. 重复步骤 2 - 3 以再次更新源堆栈，此次是从堆栈中删除目标资源。
5. 执行导入操作以将 GamesTable 添加到目标堆栈中。

   a. 在 Stacks (堆栈) 页面上，在选择了父堆栈的情况下，选择 Stack actions (堆栈操作)，然后选择 Import resources into stack (将资源导入到堆栈)。



   b. 访问 Import overview (导入概述) 页面，以查看在该操作期间需要提供的内容列表。然后选择下一步。

   c. 在 Specify template（指定模板）页面上，完成以下操作之一，然后选择 Next（下一步）。

      - 选择 Amazon S3 URL，然后在文本框中指定一个 URL。
      - 选择 Upload a template file (上传模板文件)，然后浏览要上传的文件。

   d. 在 Identify resources (标识资源) 页面上，标识您要移动的资源（在本示例中为 GamesTable）。有关更多信息，请参阅 资源标识符。

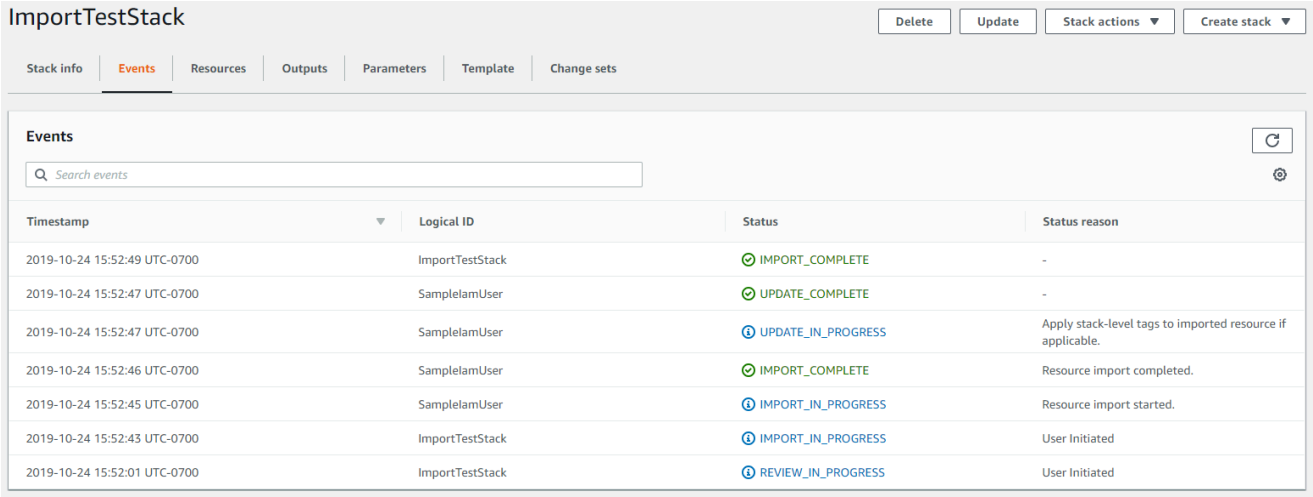
      1. 在 Identifier property (标识符属性) 下面，选择资源标识符的类型。例如，可以使用 TableName 属性标识 AWS::DynamoDB::Table 资源。
      2. 在 Identifier value (标识符值) 下面，键入实际属性值。例如 *GamesTables*。
      3. 选择下一步。

   e. 在 Specify stack details (指定堆栈详细信息) 页面上，修改任何参数，然后选择 Next (下一步)。这会自动创建一个更改集。

> **Important**
>
> 如果您修改的现有参数启动创建、更新或删除操作，导入操作将失败。

f. 在查看 ***TargetStackName*** 页面上，确认正在导入正确的资源，然后选择导入资源。这会自动启动在上一步中创建的更改集。此时，任何堆栈级标签将应用于导入的资源。

g. 将显示父堆栈的 Stack details（堆栈详细信息）页面的 Events（事件）窗格。



> **Note**
>
> 在该导入操作完成后，无需对父堆栈运行偏差检测，因为 `AWS::CloudFormation::Stack` 资源已由 CloudFormation 进行管理。

使用 Amazon CLI重构堆栈

1. 在源模板中，为要移动的资源指定 `Retain` DeletionPolicy。

   在以下示例源模板中，`GamesTable` 是该重构的目标。

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
```

```
            "Type":"AWS::DynamoDB::Table",
            "Properties":{
               "TableName":"Service",
               "AttributeDefinitions":[
                  {
                     "AttributeName":"key",
                     "AttributeType":"S"
                  }
               ],
               "KeySchema":[
                  {
                     "AttributeName":"key",
                     "KeyType":"HASH"
                  }
               ],
               "ProvisionedThroughput":{
                  "ReadCapacityUnits":5,
                  "WriteCapacityUnits":1
               }
            }
         },
         "GamesTable": {
             "Type": "AWS::DynamoDB::Table",
             "DeletionPolicy": "Retain",
             "Properties": {
                 "TableName": "Games",
                 "AttributeDefinitions": [
                     {
                         "AttributeName": "key",
                         "AttributeType": "S"
                     }
                 ],
                 "KeySchema": [
                     {
                         "AttributeName": "key",
                         "KeyType": "HASH"
                     }
                 ],
                 "ProvisionedThroughput": {
                     "ReadCapacityUnits": 5,
                     "WriteCapacityUnits": 1
                 }
             }
         }
```

```
    }
}
```

2. 更新源堆栈以将删除策略应用于资源。

```
aws cloudformation update-stack --stack-name SourceStackName
```

3. 从源模板中删除资源、相关参数和输出，然后将它们添加到目标模板中。

   现在，源模板如下所示。

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

   以下示例目标模板当前具有 PlayersTable 资源，现在还包含 GamesTable。

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "PlayersTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "TableName": "Players",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
```

```
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

4. 更新源堆栈以从堆栈中删除 GamesTable 资源及其相关参数和输出。

```
aws cloudformation update-stack --stack-name SourceStackName
```

5. 按照以下 JSON 字符串格式，编写要导入的实际资源及其唯一标识符的列表。有关更多信息，请参阅 资源标识符。

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifie
{"TableName":"Games"}}]
```

或者，您可以在配置文件中指定 JSON 格式参数。

例如，要导入 GamesTable，可以创建一个包含以下配置的 *ResourcesToImport.txt* 文件。

```
[
  {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"GamesTable",
      "ResourceIdentifier": {
        "TableName":"Games"
      }
  }
]
```

6. 要创建更改集，请使用以下 create-change-set 命令并替换占位符文本。对于 `--change-set-type` 选项，请指定 **IMPORT** 的值。对于 `--resources-to-import` 选项，将示例 JSON 字符串替换为您刚刚创建的实际 JSON 字符串。

```
aws cloudformation create-change-set \
    --stack-name TargetStackName --change-set-name ImportChangeSet \
    --change-set-type IMPORT \
```

```
    --template-body file://TemplateToImport.json \
    --resources-to-import
 "'[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdenti
{"TableName":"Games"}}]'"
```

> **Note**
>
> `--resources-to-import` 不支持内联 YAML。JSON 字符串中对转义引号的要求因终端而异。有关更多信息，请参阅《Amazon Command Line Interface User Guide》中的 Using quotation marks inside strings。

或者，您可以使用文件 URL 作为 `--resources-to-import` 选项的输入内容，如以下示例所示。

```
--resources-to-import file://ResourcesToImport.txt
```

7. 查看更改集，以确保将正确的资源导入到目标堆栈中。

```
aws cloudformation describe-change-set \
    --change-set-name ImportChangeSet
```

8. 要启动更改集并导入资源，请使用以下 execute-change-set 命令并替换占位符文本。此时，任何堆栈级标签将应用于导入的资源。在成功完成该操作后（`IMPORT_COMPLETE`），将成功导入资源。

```
aws cloudformation execute-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStackName
```

> **Note**
>
> 在该导入操作完成后，无需对目标堆栈运行偏差检测，因为资源已由 CloudFormation 进行管理。

嵌套现有的堆栈

可以使用 `resource import` 功能将现有堆栈嵌套在另一个现有堆栈中。嵌套堆栈是您从其他模板中声明和引用的通用组件。这样，您就可以避免将相同的配置复制并粘贴到您的模板中，并简化堆栈更

新。如果您具有通用组件的模板，您可以使用 AWS::CloudFormation::Stack 资源从另一个模板中引用该模板。有关嵌套堆栈的更多信息，请参阅使用嵌套堆栈将模板拆分为可重复使用的部分。

在使用 resource import 时，Amazon CloudFormation 仅支持一层嵌套。这意味着，您无法将堆栈导入到子堆栈中，也无法导入具有子堆栈的堆栈。

如果您不熟悉导入，建议您先查看 手动将 Amazon 资源导入 CloudFormation 堆栈 主题中的介绍信息。

嵌套堆栈导入验证

在嵌套堆栈导入操作期间，Amazon CloudFormation 执行以下验证。

- 父堆栈模板中的嵌套 AWS::CloudFormation::Stack 定义与实际嵌套堆栈的模板匹配。
- 父堆栈模板中的嵌套 AWS::CloudFormation::Stack 定义的标签与实际嵌套堆栈资源的标签匹配。

使用 Amazon Web Services 管理控制台嵌套现有的堆栈

1. 使用 Retain DeletionPolicy 将 AWS::CloudFormation::Stack 资源添加到父堆栈模板中。在以下示例父堆栈模板中，MyNestedStack 是导入目标。

   JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
```

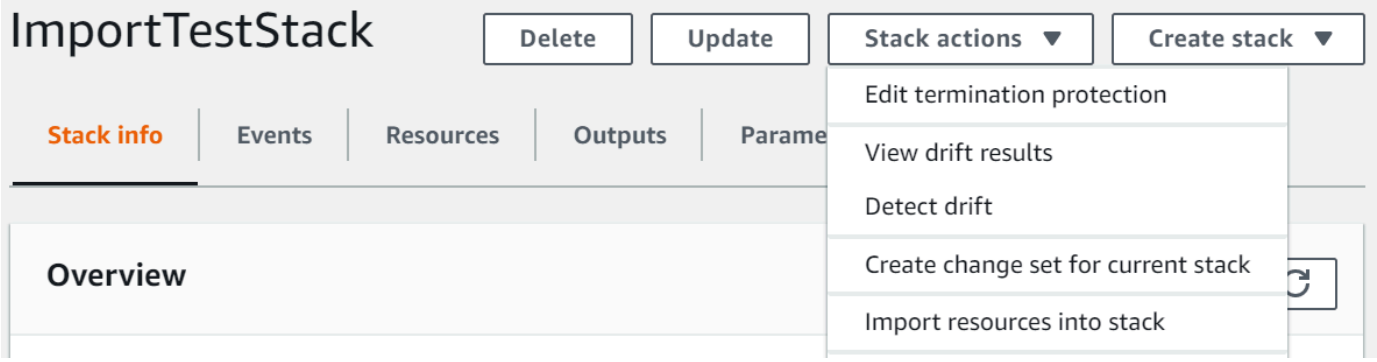
```
            ],
            "ProvisionedThroughput":{
               "ReadCapacityUnits":5,
               "WriteCapacityUnits":1
            }
         }
      },
    "MyNestedStack" : {
      "Type" : "AWS::CloudFormation::Stack",
      "DeletionPolicy": "Retain",
      "Properties" : {
      "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template",
        "Parameters" : {
          "InstanceType" : "t1.micro",
          "KeyName" : "mykey"
        }
      }
    }
  }
}
```

YAML

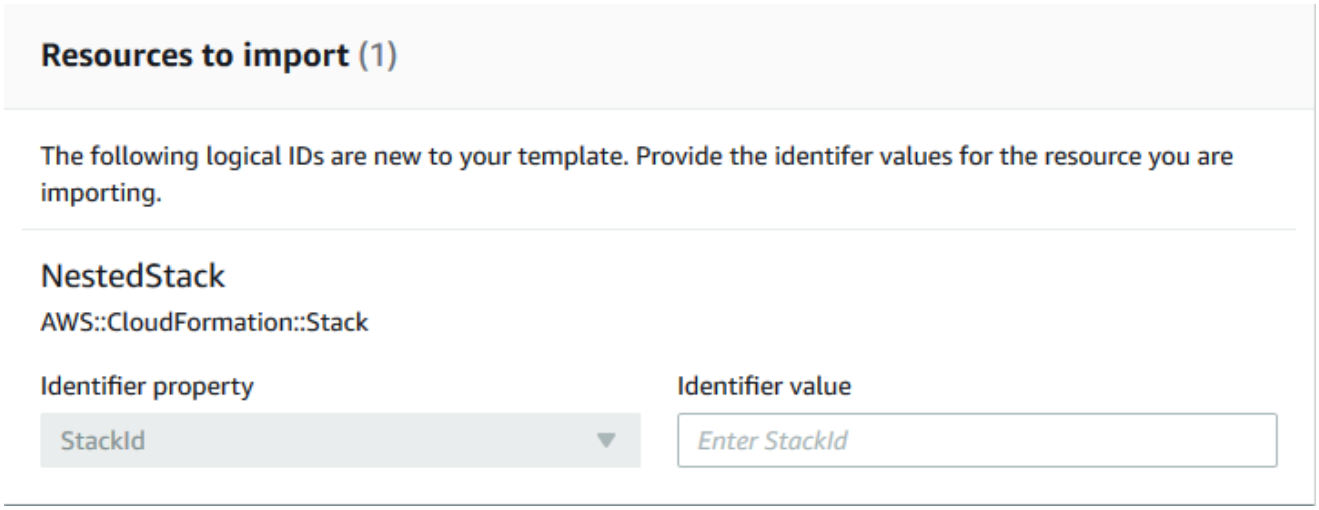
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  MyNestedStack:
    Type: AWS::CloudFormation::Stack
    DeletionPolicy: Retain
    Properties:
```

```
      TemplateURL: >-
        https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template
      Parameters:
        InstanceType: t1.micro
        KeyName: mykey
```

2. 打开 Amazon CloudFormation 控制台。
3. 在 Stacks (堆栈) 页面上，在选择了父堆栈的情况下，选择 Stack actions (堆栈操作)，然后选择 Import resources into stack (将资源导入到堆栈)。



4. 访问 Import overview (导入概述) 页面，以查看在该操作期间需要提供的内容列表。然后选择下一步。
5. 在 Specify template (指定模板) 页面上，使用以下方法之一提供更新的父模板，然后选择 Next (下一步)。
   - 选择 Amazon S3 URL，然后在文本框中指定您的模板的 URL。
   - 选择 Upload a template file (上传模板文件)，然后浏览您的模板。
6. 在 Identify resources (标识资源) 页面上，标识 `AWS::CloudFormation::Stack` 资源。
   a. 在 Identifier property (标识符属性) 下面，选择资源标识符的类型。例如，可以使用 `StackId` 属性标识 `AWS::CloudFormation::Stack` 资源。
   b. 在标识符值下，键入要导入的堆栈的 ARN。例如 *arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10*。



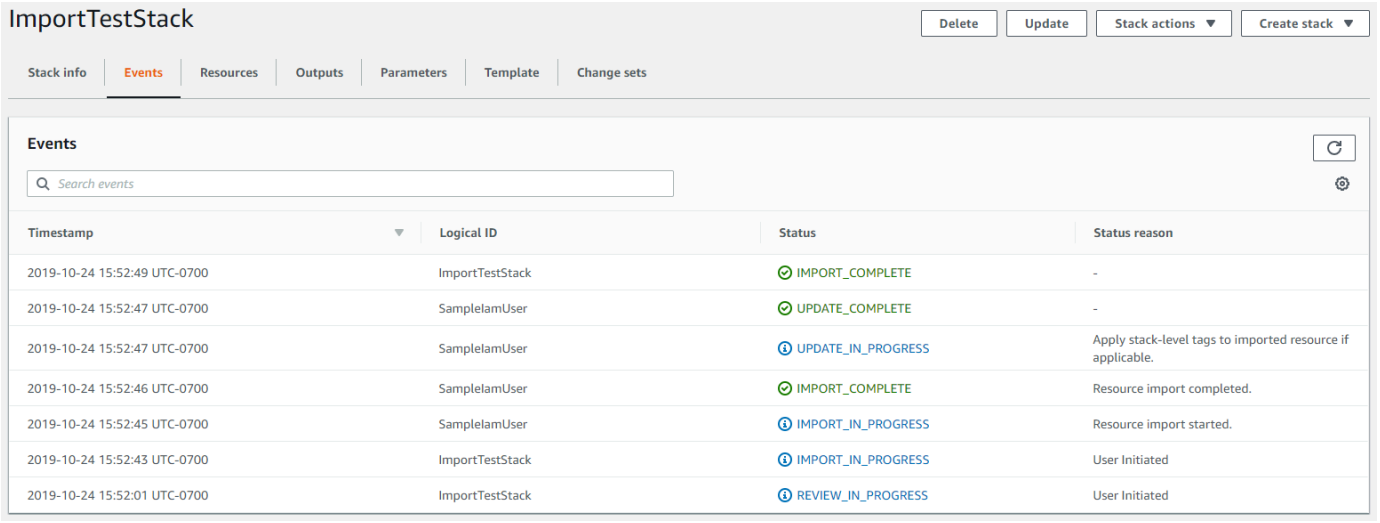
c. 选择下一步。

7. 在 Specify stack details (指定堆栈详细信息) 页面上，修改任何参数，然后选择 Next (下一步)。这会自动创建一个更改集。

> **Important**
>
> 如果您修改的现有参数启动创建、更新或删除操作，导入操作将失败。

8. 在查看 ***MyParentStack*** 页面上，确认正在导入正确的资源，然后选择导入资源。这会自动执行在上一步中创建的更改集。此时，任何堆栈级标签将应用于导入的资源。

9. 将显示父堆栈的 Stack details（堆栈详细信息）页面的 Events（事件）窗格。



> **Note**
>
> 在该导入操作完成后，无需对父堆栈运行偏差检测，因为 `AWS::CloudFormation::Stack` 资源已由 Amazon CloudFormation 进行管理。

使用 Amazon CLI嵌套现有的堆栈

1. 使用 `Retain` DeletionPolicy 将 `AWS::CloudFormation::Stack` 资源添加到父堆栈模板中。在以下示例父模板中，`MyNestedStack` 是导入目标。

   JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
    "MyNestedStack" : {
      "Type" : "AWS::CloudFormation::Stack",
      "DeletionPolicy": "Retain",
```

```
      "Properties" : {
      "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template",
        "Parameters" : {
          "InstanceType" : "t1.micro",
          "KeyName" : "mykey"
        }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  MyNestedStack:
    Type: AWS::CloudFormation::Stack
    DeletionPolicy: Retain
    Properties:
      TemplateURL: >-
        https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template
      Parameters:
        InstanceType: t1.micro
        KeyName: mykey
```

2. 编写 JSON 字符串，如以下示例所示，并进行以下修改：

   - 将 *MyNestedStack* 替换为模板中指定的目标资源的逻辑 ID。

- 将 *arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10* 替换为要导入的堆栈的 ARN。

```
[{"ResourceType":"AWS::CloudFormation::Stack","LogicalResourceId":"MyNestedStack","Resource
{"StackId":"arn:aws:cloudformation:us-east-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10"}}]
```

或者，您也可以在配置文件中指定参数。

例如，要导入 MyNestedStack，可以创建一个包含以下配置的 *ResourcesToImport.txt* 文件。

JSON

```
[
  {
      "ResourceType":"AWS::CloudFormation::Stack",
      "LogicalResourceId":"MyNestedStack",
      "ResourceIdentifier": {
        "StackId":"arn:aws:cloudformation:us-west-2:12345678910:stack/
mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10"
      }
  }
]
```

YAML

```
ResourceType: AWS::CloudFormation::Stack
  LogicalResourceId: MyNestedStack
  ResourceIdentifier:
    StackId: >-
      arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10
```

3. 要创建更改集，请使用以下 create-change-set 命令并替换占位符文本。对于 `--change-set-type` 选项，请指定 **`IMPORT`** 的值。对于 `--resources-to-import` 选项，将示例 JSON 字符串替换为您刚刚创建的实际 JSON 字符串。

```
aws cloudformation create-change-set \
    --stack-name MyParentStack --change-set-name ImportChangeSet \
```

```
    --change-set-type IMPORT \
    --template-body file://TemplateToImport.json \
    --resources-to-
import '[{"ResourceType":"AWS::CloudFormation::Stack","LogicalResourceId":"MyNestedStack","
{"StackId":"arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10"}}]'
```

> **Note**
>
> `--resources-to-import` 不支持内联 YAML。JSON 字符串中对转义引号的要求因终端而异。有关更多信息，请参阅《Amazon Command Line Interface User Guide》中的 Using quotation marks inside strings。

或者，您可以使用文件 URL 作为 `--resources-to-import` 选项的输入内容，如以下示例所示。

```
--resources-to-import file://ResourcesToImport.txt
```

如果成功，此命令将返回以下示例输出：

```
{
    "Id": "arn:aws:cloudformation:us-west-2:12345678910:changeSet/
ImportChangeSet/8ad75b3f-665f-46f6-a200-0b4727a9442e",
    "StackId": "arn:aws:cloudformation:us-west-2:12345678910:stack/
MyParentStack/4e345b70-1281-11ef-b027-027366d8e82b"
}
```

4. 查看更改集，以确保导入正确的堆栈。

```
aws cloudformation describe-change-set --change-set-name ImportChangeSet
```

5. 要启动更改集并将堆栈导入父源堆栈，请使用以下 execute-change-set 命令并替换占位符文本。此时，任何堆栈级标签将应用于导入的资源。在成功完成导入操作后（IMPORT_COMPLETE），将成功嵌套堆栈。

```
aws cloudformation execute-change-set --change-set-name ImportChangeSet
```

> **Note**
>
> 在该导入操作完成后，无需对父堆栈运行偏差检测，因为 `AWS::CloudFormation::Stack` 资源已由 Amazon CloudFormation 进行管理。

自动将 Amazon 资源导入 CloudFormation 堆栈

现在，您可以在创建或更新 CloudFormation 堆栈时自动导入命名资源。命名资源是具有自定义名称的资源。有关更多信息，请参阅 CloudFormation 模板参考中的名称类型。

当您启动自动导入时，CloudFormation 会检查是否存在与您的模板匹配的现有资源，并在部署期间导入它们。对于嵌套堆栈，请从根堆栈创建更改集。

在导入完成后以及执行后续堆栈操作之前，我们建议对导入的资源运行偏差检测。偏差检测确保模板配置与实际配置匹配。有关更多信息，请参阅 在整个 CloudFormation 堆栈上检测偏差。

要导入资源，它们需要满足下面的要求：

- 资源必须具有您的模板中定义的静态自定义名称。目前不支持动态名称（使用 !Ref 或其他函数）。
- 资源必须具有 `Retain` 或 `RetainExceptOnCreate` 的 `DeletionPolicy`。
- 资源不能已属于另一个 CloudFormation 堆栈。
- 资源类型必须支持 CloudFormation 导入操作。有关更多信息，请参阅 资源类型支持。
- 资源类型的主 ID 或附加标识符必须位于模板中。不支持具有只读属性的主 ID 或附加标识符。要找出某一类型的主 ID 或附加标识符，请在资源架构中查找 `primaryIdentifier` 和 `additionalIdentifiers` 属性。有关这些内容的更多信息，请参阅资源类型定义架构。

Example自动导入示例

以下示例使用更改集 `CreateChangeSet` 根据模板文件 `template.yaml` 创建名为 `my-stack` 的堆栈，并自动导入匹配的资源。

```
aws cloudformation create-change-set \
  --stack-name my-stack \
  --change-set-name CreateChangeSet \
  --change-set-type CREATE \
  --template-body file://template.yaml \
  --import-existing-resources
```

问题排查

如果自动导入失败，请执行下列操作进行故障排除：

- 验证您的模板中的资源名称是否与资源名称完全匹配
- 验证资源是否尚未由其他堆栈管理
- 确保资源类型支持自动导入
- 验证您的模板是否包括资源类型的所有必需属性

支持的资源类型

下表列出了当前支持自动导入的 Amazon 资源类型。

资源
AWS::ACMPCA::CertificateAuthorityActivation
AWS::ACMPCA::Permission
AWS::APS::ResourcePolicy
AWS::ARCZonalShift::ZonalAutoshiftConfiguration
AWS::ApiGateway::BasePathMapping
AWS::ApiGateway::DocumentationVersion
AWS::ApiGateway::DomainName
AWS::ApiGateway::Method
AWS::ApiGateway::Model
AWS::ApiGateway::Stage
AWS::ApiGatewayV2::DomainName
AWS::ApiGatewayV2::Stage
AWS::AppFlow::Connector

资源
AWS::AppFlow::ConnectorProfile
AWS::AppFlow::Flow
AWS::AppIntegrations::EventIntegration
AWS::AppStream::AppBlockBuilder
AWS::AppStream::ApplicationEntitlementAssociation
AWS::AppStream::ApplicationFleetAssociation
AWS::AppStream::DirectoryConfig
AWS::AppStream::Entitlement
AWS::AppStream::ImageBuilder
AWS::AppStream::Stack
AWS::AppStream::StackFleetAssociation
AWS::AppStream::StackUserAssociation
AWS::AppStream::User
AWS::AppSync::DomainName
AWS::AppSync::SourceApiAssociation
AWS::ApplicationAutoScaling::ScalableTarget
AWS::Athena::CapacityReservation
AWS::Athena::DataCatalog
AWS::Athena::PreparedStatement
AWS::Athena::WorkGroup

资源
AWS::AutoScaling::AutoScalingGroup
AWS::AutoScaling::LaunchConfiguration
AWS::AutoScaling::LifecycleHook
AWS::AutoScaling::WarmPool
AWS::Backup::BackupVault
AWS::Backup::LogicallyAirGappedBackupVault
AWS::Backup::RestoreTestingPlan
AWS::Backup::RestoreTestingSelection
AWS::Backup::TieringConfiguration
AWS::Batch::ComputeEnvironment
AWS::Batch::JobDefinition
AWS::Batch::QuotaShare
AWS::Batch::ServiceEnvironment
AWS::Bedrock::ResourcePolicy
AWS::BedrockAgentCore::ResourcePolicy
AWS::BedrockAgentCore::WorkloadIdentity
AWS::CUR::ReportDefinition
AWS::Cases::Field
AWS::Cassandra::Keyspace
AWS::Cassandra::Table

资源
AWS::Cassandra::Type
AWS::CloudFormation::ModuleDefaultVersion
AWS::CloudFormation::Stack
AWS::CloudFormation::StackSet
AWS::CloudFront::ConnectionFunction
AWS::CloudFront::KeyValueStore
AWS::CloudFront::MonitoringSubscription
AWS::CloudFront::TrustStore
AWS::CloudTrail::ResourcePolicy
AWS::CloudTrail::Trail
AWS::CloudWatch::Alarm
AWS::CloudWatch::CompositeAlarm
AWS::CloudWatch::Dashboard
AWS::CloudWatch::LogAlarm
AWS::CloudWatch::MetricStream
AWS::CodeDeploy::Application
AWS::CodeDeploy::DeploymentConfig
AWS::CodeDeploy::DeploymentGroup
AWS::CodeGuruProfiler::ProfilingGroup
AWS::CodePipeline::CustomActionType

资源
AWS::CodePipeline::Pipeline
AWS::CodeStarConnections::SyncConfiguration
AWS::Cognito::IdentityPoolPrincipalTag
AWS::Cognito::IdentityPoolRoleAttachment
AWS::Cognito::UserPoolDomain
AWS::Cognito::UserPoolGroup
AWS::Cognito::UserPoolIdentityProvider
AWS::Cognito::UserPoolResourceServer
AWS::Cognito::UserPoolRiskConfigurationAttachment
AWS::Cognito::UserPoolUICustomizationAttachment
AWS::Cognito::UserPoolUser
AWS::Cognito::UserPoolUserToGroupAttachment
AWS::Config::AggregationAuthorization
AWS::Config::ConfigRule
AWS::Config::ConfigurationAggregator
AWS::Config::ConformancePack
AWS::Config::OrganizationConformancePack
AWS::Config::StoredQuery
AWS::Connect::ApprovedOrigin
AWS::Connect::DataTableAttribute

资源
AWS::Connect::IntegrationAssociation
AWS::Connect::PredefinedAttribute
AWS::ControlTower::EnabledControl
AWS::CustomerProfiles::CalculatedAttributeDefinition
AWS::CustomerProfiles::Domain
AWS::CustomerProfiles::EventStream
AWS::CustomerProfiles::EventTrigger
AWS::CustomerProfiles::Integration
AWS::CustomerProfiles::ObjectType
AWS::CustomerProfiles::Recommender
AWS::CustomerProfiles::SegmentDefinition
AWS::DMS::DataMigration
AWS::DMS::DataProvider
AWS::DMS::InstanceProfile
AWS::DMS::MigrationProject
AWS::DMS::ReplicationConfig
AWS::DataBrew::Dataset
AWS::DataBrew::Job
AWS::DataBrew::Project
AWS::DataBrew::Recipe

资源
AWS::DataBrew::Ruleset
AWS::DataBrew::Schedule
AWS::DataZone::Domain
AWS::DataZone::Environment
AWS::DataZone::EnvironmentActions
AWS::DataZone::EnvironmentBlueprintConfiguration
AWS::DataZone::EnvironmentProfile
AWS::DataZone::FormType
AWS::DataZone::Project
AWS::Deadline::QueueFleetAssociation
AWS::Deadline::QueueLimitAssociation
AWS::Detective::MemberInvitation
AWS::Detective::OrganizationAdmin
AWS::DevOpsAgent::PrivateConnection
AWS::DocDB::GlobalCluster
AWS::DynamoDB::GlobalTable
AWS::DynamoDB::Table
AWS::EC2::EnclaveCertificateIamRoleAssociation
AWS::EC2::GatewayRouteTableAssociation
AWS::EC2::KeyPair

资源
AWS::EC2::LocalGatewayRoute
AWS::EC2::NetworkInterfaceAttachment
AWS::EC2::NetworkPerformanceMetricSubscription
AWS::EC2::RouteServerAssociation
AWS::EC2::RouteServerPropagation
AWS::EC2::SecurityGroupVpcAssociation
AWS::EC2::SqlHaStandbyDetectedInstance
AWS::EC2::TransitGatewayMeteringPolicyEntry
AWS::EC2::TransitGatewayMulticastDomainAssociation
AWS::EC2::TransitGatewayMulticastGroupMember
AWS::EC2::TransitGatewayMulticastGroupSource
AWS::EC2::TransitGatewayRoute
AWS::EC2::TransitGatewayRouteTableAssociation
AWS::EC2::TransitGatewayRouteTablePropagation
AWS::EC2::VPCDHCPOptionsAssociation
AWS::EC2::VPCEndpointServicePermissions
AWS::EC2::VPNConnectionRoute
AWS::EC2::VolumeAttachment
AWS::ECR::PublicRepository
AWS::ECR::PullThroughCacheRule

资源
AWS::ECR::PullTimeUpdateExclusion
AWS::ECR::Repository
AWS::ECR::RepositoryCreationTemplate
AWS::ECS::CapacityProvider
AWS::ECS::Cluster
AWS::ECS::ClusterCapacityProviderAssociations
AWS::ECS::PrimaryTaskSet
AWS::EKS::AccessEntry
AWS::EKS::Addon
AWS::EKS::Cluster
AWS::EKS::FargateProfile
AWS::EKS::IdentityProviderConfig
AWS::EMR::SecurityConfiguration
AWS::EMR::StudioSessionMapping
AWS::EMR::WALWorkspace
AWS::ElastiCache::CacheCluster
AWS::ElastiCache::ReplicationGroup
AWS::ElastiCache::ServerlessCache
AWS::ElastiCache::SubnetGroup
AWS::ElastiCache::User

资源
AWS::ElastiCache::UserGroup
AWS::ElasticBeanstalk::Application
AWS::ElasticBeanstalk::Environment
AWS::EntityResolution::IdMappingWorkflow
AWS::EntityResolution::IdNamespace
AWS::EntityResolution::MatchingWorkflow
AWS::EntityResolution::PolicyStatement
AWS::EntityResolution::SchemaMapping
AWS::Events::ApiDestination
AWS::Events::Archive
AWS::Events::Connection
AWS::Events::Endpoint
AWS::Events::EventBus
AWS::Events::EventBusPolicy
AWS::FIS::TargetAccountConfiguration
AWS::FMS::NotificationChannel
AWS::FSx::S3AccessPointAttachment
AWS::GameLift::ContainerGroupDefinition
AWS::GameLift::GameSessionQueue
AWS::GameLift::Location

资源
AWS::GameLift::MatchmakingConfiguration
AWS::GameLift::MatchmakingRuleSet
AWS::Glue::Crawler
AWS::Glue::Database
AWS::Glue::Job
AWS::Glue::SchemaVersionMetadata
AWS::Glue::Trigger
AWS::Glue::UsageProfile
AWS::GuardDuty::Filter
AWS::GuardDuty::Master
AWS::GuardDuty::Member
AWS::IAM::Group
AWS::IAM::GroupPolicy
AWS::IAM::InstanceProfile
AWS::IAM::Role
AWS::IAM::RolePolicy
AWS::IAM::ServerCertificate
AWS::IAM::User
AWS::IAM::UserPolicy
AWS::InternetMonitor::Monitor

资源
AWS::IoT::AccountAuditConfiguration
AWS::IoT::Authorizer
AWS::IoT::BillingGroup
AWS::IoT::CertificateProvider
AWS::IoT::Command
AWS::IoT::CustomMetric
AWS::IoT::Dimension
AWS::IoT::DomainConfiguration
AWS::IoT::FleetMetric
AWS::IoT::JobTemplate
AWS::IoT::Logging
AWS::IoT::MitigationAction
AWS::IoT::ProvisioningTemplate
AWS::IoT::RoleAlias
AWS::IoT::ScheduledAudit
AWS::IoT::SecurityProfile
AWS::IoT::SoftwarePackage
AWS::IoT::SoftwarePackageVersion
AWS::IoT::Thing
AWS::IoT::ThingGroup

资源
AWS::IoT::ThingType
AWS::IoT::TopicRule
AWS::IoTAnalytics::Channel
AWS::IoTAnalytics::Dataset
AWS::IoTAnalytics::Datastore
AWS::IoTAnalytics::Pipeline
AWS::IoTEvents::AlarmModel
AWS::IoTEvents::DetectorModel
AWS::IoTEvents::Input
AWS::IoTFleetWise::Campaign
AWS::IoTFleetWise::DecoderManifest
AWS::IoTFleetWise::Fleet
AWS::IoTFleetWise::ModelManifest
AWS::IoTFleetWise::SignalCatalog
AWS::IoTFleetWise::StateTemplate
AWS::IoTFleetWise::Vehicle
AWS::IoTTwinMaker::ComponentType
AWS::IoTTwinMaker::Entity
AWS::IoTTwinMaker::Scene
AWS::IoTTwinMaker::SyncJob

资源
AWS::IoTTwinMaker::Workspace
AWS::IoTWireless::Destination
AWS::IoTWireless::NetworkAnalyzerConfiguration
AWS::IoTWireless::PartnerAccount
AWS::KMS::Alias
AWS::KafkaConnect::Connector
AWS::KafkaConnect::CustomPlugin
AWS::KafkaConnect::WorkerConfiguration
AWS::Kinesis::ResourcePolicy
AWS::Kinesis::Stream
AWS::KinesisAnalyticsV2::Application
AWS::KinesisFirehose::DeliveryStream
AWS::KinesisVideo::SignalingChannel
AWS::KinesisVideo::Stream
AWS::LakeFormation::DataCellsFilter
AWS::LakeFormation::Tag
AWS::Lambda::CapacityProvider
AWS::Lambda::Function
AWS::Lex::ResourcePolicy
AWS::Lightsail::Alarm

资源
AWS::Lightsail::Bucket
AWS::Lightsail::Certificate
AWS::Lightsail::Container
AWS::Lightsail::Database
AWS::Lightsail::DatabaseSnapshot
AWS::Lightsail::Disk
AWS::Lightsail::DiskSnapshot
AWS::Lightsail::Distribution
AWS::Lightsail::Domain
AWS::Lightsail::Instance
AWS::Lightsail::InstanceSnapshot
AWS::Lightsail::LoadBalancer
AWS::Lightsail::LoadBalancerTlsCertificate
AWS::Lightsail::StaticIp
AWS::Location::APIKey
AWS::Location::GeofenceCollection
AWS::Location::Map
AWS::Location::PlaceIndex
AWS::Location::RouteCalculator
AWS::Location::Tracker

资源
AWS::Location::TrackerConsumer
AWS::Logs::DeliveryDestination
AWS::Logs::DeliverySource
AWS::Logs::Destination
AWS::Logs::Integration
AWS::Logs::LogGroup
AWS::Logs::LogStream
AWS::Logs::MetricFilter
AWS::Logs::ResourcePolicy
AWS::Logs::ScheduledQuery
AWS::Logs::SubscriptionFilter
AWS::Logs::Transformer
AWS::LookoutEquipment::InferenceScheduler
AWS::LookoutVision::Project
AWS::M2::Deployment
AWS::MSK::BatchScramSecret
AWS::MSK::ClusterPolicy
AWS::MSK::Replicator
AWS::MSK::Topic
AWS::MWAA::Environment

资源
AWS::MWAAServerless::Workflow
AWS::Macie::FindingsFilter
AWS::MediaConnect::BridgeOutput
AWS::MediaConnect::BridgeSource
AWS::MediaConnect::FlowVpcInterface
AWS::MediaLive::Multiplexprogram
AWS::MediaPackage::Asset
AWS::MediaPackage::Channel
AWS::MediaPackage::OriginEndpoint
AWS::MediaPackage::PackagingConfiguration
AWS::MediaPackage::PackagingGroup
AWS::MediaPackageV2::Channel
AWS::MediaPackageV2::ChannelGroup
AWS::MediaPackageV2::ChannelPolicy
AWS::MediaPackageV2::OriginEndpoint
AWS::MediaPackageV2::OriginEndpointPolicy
AWS::MediaTailor::Channel
AWS::MediaTailor::ChannelPolicy
AWS::MediaTailor::LiveSource
AWS::MediaTailor::PlaybackConfiguration

资源
AWS::MediaTailor::SourceLocation
AWS::MediaTailor::VodSource
AWS::MemoryDB::ACL
AWS::MemoryDB::Cluster
AWS::MemoryDB::ParameterGroup
AWS::MemoryDB::SubnetGroup
AWS::MemoryDB::User
AWS::Neptune::DBCluster
AWS::Neptune::DBClusterParameterGroup
AWS::Neptune::DBInstance
AWS::Neptune::DBParameterGroup
AWS::Neptune::DBSubnetGroup
AWS::Neptune::EventSubscription
AWS::NeptuneGraph::Graph
AWS::NeptuneGraph::PrivateGraphEndpoint
AWS::NetworkFirewall::LoggingConfiguration
AWS::NetworkManager::ConnectPeer
AWS::NetworkManager::CoreNetwork
AWS::NetworkManager::CoreNetworkPrefixListAssociation
AWS::NetworkManager::CustomerGatewayAssociation

资源
AWS::NetworkManager::DirectConnectGatewayAttachment
AWS::NetworkManager::LinkAssociation
AWS::NetworkManager::SiteToSiteVpnAttachment
AWS::NetworkManager::TransitGatewayRegistration
AWS::NetworkManager::VpcAttachment
AWS::Notifications::ChannelAssociation
AWS::Notifications::ManagedNotificationAccountContactAssociation
AWS::Notifications::ManagedNotificationAdditionalChannelAssociation
AWS::Notifications::NotificationHub
AWS::Notifications::OrganizationalUnitAssociation
AWS::NovaAct::WorkflowDefinition
AWS::ObservabilityAdmin::OrganizationCentralizationRule
AWS::ObservabilityAdmin::TelemetryEnrichment
AWS::ObservabilityAdmin::TelemetryPipelines
AWS::Omics::AnnotationStore
AWS::Omics::Configuration
AWS::Omics::VariantStore
AWS::OpenSearchServerless::AccessPolicy
AWS::OpenSearchServerless::Collection
AWS::OpenSearchServerless::CollectionGroup

资源
AWS::OpenSearchServerless::CollectionIndex
AWS::OpenSearchServerless::Index
AWS::OpenSearchServerless::LifecyclePolicy
AWS::OpenSearchServerless::SecurityConfig
AWS::OpenSearchServerless::SecurityPolicy
AWS::OpenSearchServerless::VpcEndpoint
AWS::OpenSearchService::Application
AWS::OpenSearchService::Domain
AWS::PCAConnectorAD::ServicePrincipalName
AWS::PCAConnectorAD::TemplateGroupAccessControlEntry
AWS::Panorama::PackageVersion
AWS::PaymentCryptography::Alias
AWS::Pinpoint::InAppTemplate
AWS::Pipes::Pipe
AWS::Proton::EnvironmentTemplate
AWS::Proton::ServiceTemplate
AWS::QBusiness::Permission
AWS::QuickSight::ActionConnector
AWS::QuickSight::Analysis
AWS::QuickSight::CustomPermissions

资源
AWS::QuickSight::Dashboard
AWS::QuickSight::DataSet
AWS::QuickSight::DataSource
AWS::QuickSight::Folder
AWS::QuickSight::RefreshSchedule
AWS::QuickSight::Template
AWS::QuickSight::Theme
AWS::QuickSight::Topic
AWS::QuickSight::VPCConnection
AWS::RDS::CustomDBEngineVersion
AWS::RDS::DBCluster
AWS::RDS::DBClusterParameterGroup
AWS::RDS::DBInstance
AWS::RDS::DBParameterGroup
AWS::RDS::DBProxy
AWS::RDS::DBProxyEndpoint
AWS::RDS::DBProxyTargetGroup
AWS::RDS::DBShardGroup
AWS::RDS::DBSubnetGroup
AWS::RDS::EventSubscription

资源
AWS::RDS::GlobalCluster
AWS::RDS::OptionGroup
AWS::RUM::AppMonitor
AWS::Redshift::Cluster
AWS::Redshift::ClusterParameterGroup
AWS::Redshift::EndpointAccess
AWS::Redshift::EndpointAuthorization
AWS::Redshift::EventSubscription
AWS::Redshift::ScheduledAction
AWS::RedshiftServerless::Namespace
AWS::RedshiftServerless::Snapshot
AWS::RedshiftServerless::Workgroup
AWS::Rekognition::Collection
AWS::Rekognition::Project
AWS::Rekognition::StreamProcessor
AWS::ResourceGroups::Group
AWS::Route53::DNSSEC
AWS::Route53::KeySigningKey
AWS::Route53RecoveryReadiness::Cell
AWS::Route53RecoveryReadiness::ReadinessCheck

资源
AWS::Route53RecoveryReadiness::RecoveryGroup
AWS::Route53RecoveryReadiness::ResourceSet
AWS::Route53Resolver::ResolverConfig
AWS::S3::AccessPoint
AWS::S3::Bucket
AWS::S3::BucketPolicy
AWS::S3::MultiRegionAccessPoint
AWS::S3::MultiRegionAccessPointPolicy
AWS::S3::StorageLens
AWS::S3::StorageLensGroup
AWS::S3Express::AccessPoint
AWS::S3Express::BucketPolicy
AWS::S3Express::DirectoryBucket
AWS::S3Files::FileSystemPolicy
AWS::S3ObjectLambda::AccessPoint
AWS::S3ObjectLambda::AccessPointPolicy
AWS::S3Outposts::BucketPolicy
AWS::S3Tables::Namespace
AWS::S3Tables::TableBucketPolicy
AWS::S3Tables::TablePolicy

资源
AWS::S3Vectors::Index
AWS::S3Vectors::VectorBucket
AWS::S3Vectors::VectorBucketPolicy
AWS::SES::ConfigurationSet
AWS::SES::ContactList
AWS::SES::CustomVerificationEmailTemplate
AWS::SES::DedicatedIpPool
AWS::SES::EmailIdentity
AWS::SES::MultiRegionEndpoint
AWS::SES::Tenant
AWS::SMSVOICE::ConfigurationSet
AWS::SMSVOICE::OptOutList
AWS::SMSVOICE::ResourcePolicy
AWS::SMSVOICE::SenderId
AWS::SNS::TopicInlinePolicy
AWS::SQS::QueueInlinePolicy
AWS::SSM::Document
AWS::SSM::Parameter
AWS::SSM::ResourceDataSync
AWS::SSO::ApplicationAssignment

资源
AWS::SSO::Assignment
AWS::SSO::InstanceAccessControlAttributeConfiguration
AWS::SageMaker::App
AWS::SageMaker::AppImageConfig
AWS::SageMaker::Cluster
AWS::SageMaker::Device
AWS::SageMaker::DeviceFleet
AWS::SageMaker::FeatureGroup
AWS::SageMaker::Image
AWS::SageMaker::InferenceExperiment
AWS::SageMaker::MlflowTrackingServer
AWS::SageMaker::ModelCard
AWS::SageMaker::Pipeline
AWS::SageMaker::ProcessingJob
AWS::SageMaker::Space
AWS::SageMaker::StudioLifecycleConfig
AWS::SageMaker::UserProfile
AWS::Scheduler::Schedule
AWS::Scheduler::ScheduleGroup
AWS::SecurityHub::SecurityControl

资源
AWS::SecurityHub::Standard
AWS::SecurityLake::AwsLogSource
AWS::SecurityLake::SubscriberNotification
AWS::ServiceCatalog::PortfolioPrincipalAssociation
AWS::ServiceCatalog::PortfolioProductAssociation
AWS::ServiceCatalog::PortfolioShare
AWS::ServiceCatalog::ServiceActionAssociation
AWS::ServiceCatalog::TagOptionAssociation
AWS::ServiceCatalogAppRegistry::Application
AWS::ServiceCatalogAppRegistry::AttributeGroup
AWS::Signer::ProfilePermission
AWS::SimSpaceWeaver::Simulation
AWS::StepFunctions::StateMachine
AWS::SupportApp::SlackChannelConfiguration
AWS::SupportApp::SlackWorkspaceConfiguration
AWS::Synthetics::Canary
AWS::Synthetics::Group
AWS::Timestream::Database
AWS::Timestream::Table
AWS::Transfer::User

资源
AWS::VerifiedPermissions::PolicyStoreAlias
AWS::VpcLattice::AccessLogSubscription
AWS::VpcLattice::AuthPolicy
AWS::VpcLattice::DomainVerification
AWS::VpcLattice::Listener
AWS::VpcLattice::ResourcePolicy
AWS::VpcLattice::Rule
AWS::VpcLattice::Service
AWS::VpcLattice::ServiceNetwork
AWS::VpcLattice::ServiceNetworkServiceAssociation
AWS::VpcLattice::ServiceNetworkVpcAssociation
AWS::VpcLattice::TargetGroup
AWS::WAFv2::LoggingConfiguration
AWS::WAFv2::WebACLAssociation
AWS::WorkspacesInstances::VolumeAssociation
AWS::XRay::ResourcePolicy

恢复导入操作

要恢复导入操作，请为要从模板中删除的资源指定 Retain 删除策略，以确保从堆栈中删除时保留该资源。

使用 Amazon Web Services 管理控制台恢复导入操作

1. 为要从堆栈中删除的资源指定 `Retain` DeletionPolicy。在以下示例模板中，`GamesTable` 是该恢复操作的目标。

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
```

```
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

2. 打开 CloudFormation 控制台以执行堆栈更新，从而应用删除策略。

   a. 在 Stacks (堆栈) 页面上，在选择了堆栈的情况下，选择 Update (更新)，然后选择 Update stack (standard) (更新堆栈 (标准))。

   b. 在 Prepare template (准备模板) 下面，选择 Replace current template (替换当前模板)。

   c. 在 Specify template (指定模板) 下面，在 `GamesTable` 上为更新的源模板提供 `DeletionPolicy` 属性，然后选择 Next (下一步)。

      - 选择 Amazon S3 URL，然后在文本框中指定更新的源模板的 URL。
      - 选择 Upload a template file (上传模板文件)，然后浏览更新的源模板文件。

   d. 在 Specify stack details (指定堆栈详细信息) 页面上，不需要进行任何更改。选择下一步。

   e. 在 Configure stack options (配置堆栈选项) 页面上，不需要进行任何更改。选择下一步。

   f. 在查看 ***MyStack*** 页面上，检查所做的更改。如果您的模板包含 IAM 资源，请选择 I acknowledge that this template may create IAM resources (我确认此模板可创建 IAM 资源) 以指定您要使用模板中的 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。然后，创建更改集以更新源堆栈，或者直接更新源堆栈。

3. 从堆栈模板中删除资源、相关参数和输出。在该示例中，模板现在如下所示。

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
```

```
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

4. 重复步骤 2，以从堆栈中删除资源 (GamesTable) 及其相关参数和输出。

使用 Amazon CLI恢复导入操作

1. 为要从堆栈中删除的资源指定 `Retain` DeletionPolicy。在以下示例模板中，`GamesTable` 是该恢复操作的目标。

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
```

```
            "AttributeDefinitions":[
               {
                  "AttributeName":"key",
                  "AttributeType":"S"
               }
            ],
            "KeySchema":[
               {
                  "AttributeName":"key",
                  "KeyType":"HASH"
               }
            ],
            "ProvisionedThroughput":{
               "ReadCapacityUnits":5,
               "WriteCapacityUnits":1
            }
         }
      },
      "GamesTable": {
          "Type": "AWS::DynamoDB::Table",
          "DeletionPolicy": "Retain",
          "Properties": {
              "TableName": "Games",
              "AttributeDefinitions": [
                  {
                      "AttributeName": "key",
                      "AttributeType": "S"
                  }
              ],
              "KeySchema": [
                  {
                      "AttributeName": "key",
                      "KeyType": "HASH"
                  }
              ],
              "ProvisionedThroughput": {
                  "ReadCapacityUnits": 5,
                  "WriteCapacityUnits": 1
              }
          }
      }
  }
}
```

2. 更新堆栈以将删除策略应用于资源。

```
aws cloudformation update-stack --stack-name MyStack
```

3. 从堆栈模板中删除资源、相关参数和输出。在该示例中，模板现在如下所示。

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

4. 更新堆栈以从堆栈中删除资源 (GamesTable) 及其相关参数和输出。

```
aws cloudformation update-stack --stack-name MyStack
```

堆栈重构

通过堆栈重构，可以重新组织 CloudFormation 堆栈中的资源，同时保留现有资源属性和数据。您可以在堆栈之间移动资源，将大堆栈拆分为小堆栈，或将多个堆栈合并为一个堆栈。

主题

- 堆栈重构的工作原理
- 堆栈重构注意事项
- 先决条件
- 重构堆栈（控制台）
- 重构堆栈（Amazon CLI）
- 资源限制

堆栈重构的工作原理

重构堆栈涉及下列阶段：

1. 评估当前基础设施 – 检查现有 CloudFormation 堆栈和资源，以确定堆栈重构的机会。
2. 规划重构 – 定义应如何组织资源。应注意相关依赖项、命名惯例和操作限制。这些因素可能会影响 CloudFormation 稍后进行的验证。
3. 确定目标堆栈 – 决定要将资源重构到哪些堆栈。您可以在至少 2 个堆栈（使用控制台）和最多 5 个堆栈（使用 Amazon CLI）之间移动资源。资源可以在嵌套堆栈之间移动。
4. 更新模板 – 更改 CloudFormation 模板以反映计划的更改，例如在模板之间移动资源定义。在此过程中，您可以重命名逻辑 ID。
5. 创建堆栈重构 – 提供要重构的堆栈名称和模板列表。
6. 检查重构影响并解决任何冲突 – CloudFormation 验证您提供的模板并检查跨堆栈依赖项、存在标签更新问题的资源类型以及资源逻辑 ID 冲突。

   如果验证成功，CloudFormation 将生成执行期间发生的重构操作的预览。

   如果验证失败，请解决已发现的问题并重试。对于冲突，请提供资源逻辑 ID 映射，显示冲突资源的源和目标。
7. 执行重构 – 确认更改符合重构目标后，完成堆栈重构。
8. 监控 – 跟踪执行状态以确保操作成功完成。

堆栈重构注意事项

在重构堆栈时，请记住以下几点：

- 堆栈重构仅限于重新整理现有资源。在重构过程中，您无法创建或删除资源、修改资源配置，也无法更改或添加新的参数、条件或映射关系。要进行这些更改，请先更新您的堆栈，然后再执行堆栈重构。
- 您不能将同一资源重构到多个堆栈。
- 您无法重构引用源堆栈和目标堆栈之间的值不同的伪参数的资源，例如 `AWS::StackName`。
- CloudFormation 不支持空堆栈。如果重构会导致堆栈没有资源，则必须先向堆栈添加至少一个资源，然后再运行 create-stack-refactor。这可以是像 `AWS::SNS::Topic` 或 `AWS::CloudFormation::WaitCondition` 这样的简单资源。例如：

```
Resources:
  MySimpleSNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: MySimpleTopic
```

- 无论策略允许或拒绝什么，堆栈重构都不支持附加了堆栈策略的堆栈。

先决条件

要重构堆栈，您必须已经创建了修订后的模板。

使用 get-template 命令检索要重构的堆栈的 CloudFormation 模板。

```
aws cloudformation get-template --stack-name Stack1
```

拥有模板后，请使用您选择的集成式开发环境 (IDE) 对其进行更新，以便使用所需的结构和资源组织。

重构堆栈（控制台）

使用以下过程通过控制台重构堆栈。

要重构堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。

2. 在屏幕顶部的导航栏中，选择您的堆栈所在的 Amazon Web Services 区域。
3. 在左侧导航窗格中，选择堆栈重构。
4. 在堆栈重构页面上，选择启动堆栈重构。
5. 对于描述，提供一段描述，以帮助您确定您的堆栈重构。然后选择下一步。
6. 对于堆栈 1，请执行以下操作：

   a. 选择更新现有堆栈的模板或创建新堆栈。

      如果您选择了更新现有堆栈的模板，则从列表中选择一个现有堆栈。或者选择输入堆栈 ARN，以输入现有堆栈的 ARN。

      如果您选择了创建新堆栈，请在堆栈名称中提供新堆栈的名称。

   b. 在用重构后的模板替换现有模板下，选择 Amazon S3 URL 或上传模板文件，为堆栈 1 上传所需的模板。

   c. 选择下一步。
7. 对于堆栈 2，请执行以下操作：

   a. 选择更新现有堆栈的模板或创建新堆栈。

      如果您选择了更新现有堆栈的模板，则从列表中选择一个现有堆栈。或者选择输入堆栈 ARN，以输入现有堆栈的 ARN。

      如果您选择了创建新堆栈，请在堆栈名称中提供新堆栈的名称。

   b. 在用重构后的模板替换现有模板下，选择 Amazon S3 URL 或上传模板文件，为堆栈 2 上传所需的模板。

   c. 选择下一步。
8. 在指定逻辑资源 ID 重命名页面上，确保 CloudFormation 知道如何通过将所显示的任何资源映射到正确的逻辑 ID 来实现对栈的重构操作。在堆栈重构的过程中，如果更改了任何资源的逻辑 ID，则需要通过提供源堆栈名称、原始逻辑 ID、目标堆栈名称和重命名的逻辑 ID 来指定其重命名方式。在某些情况下，CloudFormation 控制台可能会自动识别资源映射，您只需在继续操作前确认预填好的资源映射正确即可。
9. 选择下一步。
10. 在审核并执行页面上，审核前面的步骤中所做的所有选择，并确认所有设置都已正确完成。
11. 准备好重构堆栈时，请选择执行堆栈重构。

重构堆栈（Amazon CLI）

用于堆栈重构的 Amazon CLI 命令包括：

- create-stack-refactory，验证和生成计划更改的预览。
- describe-stack-refactor，检索堆栈重构操作的状态和详细信息。
- execute-stack-refactor，完成经过验证的堆栈重构操作。
- list-stack-refactors，列出您账户中的所有堆栈重构操作及其当前状态和基本信息。
- list-stack-refactor-actions，显示重构执行期间 CloudFormation 将对每个堆栈和资源执行的特定操作的预览。

使用以下过程通过 Amazon CLI 重构堆栈。

要重构堆栈

1. 使用 create-stack-refactor 命令，为要重构的堆栈提供堆栈名称和更新的模板。包括 `--enable-stack-creation` 选项以允许 CloudFormation 创建新堆栈（如果它们尚不存在）。

```
aws cloudformation create-stack-refactor \
  --stack-definitions \
    StackName=Stack1,TemplateBody@=file://template1-updated.yaml \
    StackName=Stack2,TemplateBody@=file://template2-updated.yaml \
  --enable-stack-creation
```

该命令会返回 `StackRefactorId`，您将在后续步骤中使用它。

```
{
    "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841"
}
```

如果在模板验证期间检测到冲突（您可以在下一步中确认），请使用带有 `--resource-mappings` 选项的 create-stack-refactor 命令。

```
aws cloudformation create-stack-refactor \
  --stack-definitions \
    StackName=Stack1,TemplateBody@=file://template1-updated.yaml \
    StackName=Stack2,TemplateBody@=file://template2-updated.yaml \
  --enable-stack-creation \
```

```
  --resource-mappings file://resource-mapping.json
```

下面是一个 resource-mapping.json 示例文件。

```
[
    {
        "Source": {
            "StackName": "Stack1",
            "LogicalResourceId": "MySNSTopic"
        },
        "Destination": {
            "StackName": "Stack2",
            "LogicalResourceId": "MyLambdaSNSTopic"
        }
    }
]
```

2. 使用 describe-stack-refactor 命令确保 Status 为 CREATE_COMPLETE。这确认验证是否已完成。

```
aws cloudformation describe-stack-refactor \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

输出示例：

```
{
    "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841",
    "StackIds": [
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
    ],
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE"
}
```

3. 使用 list-stack-refactor-actions 命令预览将执行的特定操作。

```
aws cloudformation list-stack-refactor-actions \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

输出示例：

```
{
    "StackRefactorActions": [
        {
            "Action": "MOVE",
            "Entity": "RESOURCE",
            "PhysicalResourceId": "MyTestLambdaRole",
            "Description": "No configuration changes detected.",
            "Detection": "AUTO",
            "TagResources": [],
            "UntagResources": [],
            "ResourceMapping": {
                "Source": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
                    "LogicalResourceId": "MyLambdaRole"
                },
                "Destination": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack2/5da91650-94b1-11f0-81cf-0a23500e151b",
                    "LogicalResourceId": "MyLambdaRole"
                }
            }
        },
        {
            "Action": "MOVE",
            "Entity": "RESOURCE",
            "PhysicalResourceId": "MyTestFunction",
            "Description": "Resource configuration changes will be validated during
 refactor execution.",
            "Detection": "AUTO",
            "TagResources": [
                {
                    "Key": "aws:cloudformation:stack-name",
                    "Value": "Stack2"
                },
                {
                    "Key": "aws:cloudformation:logical-id",
                    "Value": "MyFunction"
                },
                {
                    "Key": "aws:cloudformation:stack-id",
```

```
                    "Value": "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
                }
            ],
            "UntagResources": [
                "aws:cloudformation:stack-name",
                "aws:cloudformation:logical-id",
                "aws:cloudformation:stack-id"
            ],
            "ResourceMapping": {
                "Source": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
                    "LogicalResourceId": "MyFunction"
                },
                "Destination": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack2/5da91650-94b1-11f0-81cf-0a23500e151b",
                    "LogicalResourceId": "MyFunction"
                }
            }
        }
    ]
}
```

4. 检查并确认更改后，使用 execute-stack-refactor 命令完成堆栈重构操作。

```
aws cloudformation execute-stack-refactor \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

5. 使用 describe-stack-refactor 命令监视执行状态。

```
aws cloudformation describe-stack-refactor \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

输出示例：

```
{
    "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841",
    "StackIds": [
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
```

```
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
    ],
    "ExecutionStatus": "SUCCEEDED",
    "Status": "COMPLETE"
}
```

资源限制

- 堆栈重构仅支持 `provisioningType` 为 `FULLY_MUTABLE` 的资源类型，您可以使用 describe-type 命令进行检查。
- CloudFormation 将在重构创建期间验证资源资格，并在 describe-stack-refactor 命令的输出中报告任何不受支持的资源。
- 以下资源不支持堆栈重构：
  - `AWS::ACMPCA::Certificate`
  - `AWS::ACMPCA::CertificateAuthority`
  - `AWS::ACMPCA::CertificateAuthorityActivation`
  - `AWS::ApiGateway::BasePathMapping`
  - `AWS::ApiGateway::Method`
  - `AWS::AppConfig::ConfigurationProfile`
  - `AWS::AppConfig::Deployment`
  - `AWS::AppConfig::Environment`
  - `AWS::AppConfig::Extension`
  - `AWS::AppConfig::ExtensionAssociation`
  - `AWS::AppStream::DirectoryConfig`
  - `AWS::AppStream::StackFleetAssociation`
  - `AWS::AppStream::StackUserAssociation`
  - `AWS::AppStream::User`
  - `AWS::AutoScaling::AutoScalingGroup`
  - `AWS::BackupGateway::Hypervisor`
  - `AWS::CertificateManager::Certificate`
  - `AWS::CloudFormation::CustomResource`

- AWS::CloudFormation::Macro
- AWS::CloudFormation::WaitCondition
- AWS::CloudFormation::WaitConditionHandle
- AWS::CodeDeploy::DeploymentGroup
- AWS::CodePipeline::CustomActionType
- AWS::Cognito::UserPoolRiskConfigurationAttachment
- AWS::Cognito::UserPoolUICustomizationAttachment
- AWS::Cognito::UserPoolUserToGroupAttachment
- AWS::Config::ConfigRule
- AWS::Config::ConfigurationRecorder
- AWS::Config::DeliveryChannel
- AWS::DataBrew::Dataset
- AWS::DataBrew::Job
- AWS::DataBrew::Project
- AWS::DataBrew::Recipe
- AWS::DataBrew::Ruleset
- AWS::DataBrew::Schedule
- AWS::DataZone::DataSource
- AWS::DataZone::Environment
- AWS::DataZone::EnvironmentBlueprintConfiguration
- AWS::DataZone::EnvironmentProfile
- AWS::DataZone::Project
- AWS::DataZone::SubscriptionTarget
- AWS::DirectoryService::MicrosoftAD
- AWS::DynamoDB::GlobalTable
- AWS::EC2::CustomerGateway
- AWS::EC2::EIP
- AWS::EC2::LaunchTemplate
- AWS::EC2::NetworkInterfacePermission
- AWS::EC2::PlacementGroup

- AWS::EC2::SpotFleet
- AWS::EC2::VPCDHCPOptionsAssociation
- AWS::EC2::VolumeAttachment
- AWS::EMR::Cluster
- AWS::EMR::InstanceFleetConfig
- AWS::EMR::InstanceGroupConfig
- AWS::ElastiCache::CacheCluster
- AWS::ElastiCache::ReplicationGroup
- AWS::ElastiCache::SecurityGroup
- AWS::ElastiCache::SecurityGroupIngress
- AWS::ElasticBeanstalk::ConfigurationTemplate
- AWS::ElasticLoadBalancing::LoadBalancer
- AWS::ElasticLoadBalancingV2::ListenerCertificate
- AWS::Elasticsearch::Domain
- AWS::FIS::ExperimentTemplate
- AWS::Glue::Schema
- AWS::GuardDuty::IPSet
- AWS::GuardDuty::PublishingDestination
- AWS::GuardDuty::ThreatIntelSet
- AWS::IAM::AccessKey
- AWS::IAM::UserToGroupAddition
- AWS::ImageBuilder::Component
- AWS::IoT::PolicyPrincipalAttachment
- AWS::IoT::ThingPrincipalAttachment
- AWS::IoTFleetWise::Campaign
- AWS::IoTWireless::WirelessDeviceImportTask
- AWS::Lambda::EventInvokeConfig
- AWS::Lex::BotVersion
- AWS::M2::Application
- AWS::MSK::Configuration

- AWS::MSK::ServerlessCluster
- AWS::Maester::DocumentType
- AWS::MediaTailor::Channel
- AWS::NeptuneGraph::PrivateGraphEndpoint
- AWS::Omics::AnnotationStore
- AWS::Omics::ReferenceStore
- AWS::Omics::SequenceStore
- AWS::OpenSearchServerless::Collection
- AWS::OpsWorks::App
- AWS::OpsWorks::ElasticLoadBalancerAttachment
- AWS::OpsWorks::Instance
- AWS::OpsWorks::Layer
- AWS::OpsWorks::Stack
- AWS::OpsWorks::UserProfile
- AWS::OpsWorks::Volume
- AWS::PCAConnectorAD::Connector
- AWS::PCAConnectorAD::DirectoryRegistration
- AWS::PCAConnectorAD::Template
- AWS::PCAConnectorAD::TemplateGroupAccessControlEntry
- AWS::Panorama::PackageVersion
- AWS::QuickSight::Theme
- AWS::RDS::DBSecurityGroup
- AWS::RDS::DBSecurityGroupIngress
- AWS::Redshift::ClusterSecurityGroup
- AWS::Redshift::ClusterSecurityGroupIngress
- AWS::RefactorSpaces::Environment
- AWS::RefactorSpaces::Route
- AWS::RefactorSpaces::Service
- AWS::RoboMaker::RobotApplication
- AWS::RoboMaker::SimulationApplication

- AWS::Route53::RecordSet
- AWS::Route53::RecordSetGroup
- AWS::SDB::Domain
- AWS::SageMaker::InferenceComponen
- AWS::ServiceCatalog::PortfolioPrincipalAssociation
- AWS::ServiceCatalog::PortfolioProductAssociation
- AWS::ServiceCatalog::PortfolioShare
- AWS::ServiceCatalog::TagOptionAssociation
- AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation
- AWS::ServiceCatalogAppRegistry::ResourceAssociation
- AWS::StepFunctions::StateMachineVersion
- AWS::Synthetics::Canary
- AWS::VoiceID::Domain
- AWS::WAF::ByteMatchSet
- AWS::WAF::IPSet
- AWS::WAF::Rule
- AWS::WAF::SizeConstraintSet
- AWS::WAF::SqlInjectionMatchSet
- AWS::WAF::WebACL
- AWS::WAF::XssMatchSet
- AWS::WAFv2::IPSet
- AWS::WAFv2::RegexPatternSet
- AWS::WAFv2::RuleGroup
- AWS::WAFv2::WebACL
- AWS::WorkSpaces::Workspace

资源类型支持

下表列出了当前支持导入、偏差检测和基础设施即代码（IaC）生成器操作的 Amazon 资源类型。每个资源类型名称链接到《Amazon CloudFormation 模板参考指南》中其对应的参考主题。

支持资源导入的资源也可以支持自动导入。有关更多信息，请参阅 将 Amazon 资源导入 CloudFormation 堆栈。

这并不是 Amazon 资源的详尽列表。如果未列出特定资源类型，则可能意味着无法通过 Amazon 云端控制 API 访问该资源。有关更多信息，请参阅《Cloud Control API 用户指南》中的 Resource types that support Cloud Control API。每个 Amazon 服务独立确定可通过 Cloud Control API 访问哪些资源类型。

CloudFormation 还支持对可预置私有资源类型（预置类型为 FULLY_MUTABLE 或 IMMUTABLE 的资源类型）的导入和偏差检测操作。要导入或对私有资源类型执行偏差检测，您必须首先在您的账户中注册该资源类型的默认版本，并确保它是可预置的。有关更多信息，请参阅 使用已与您共享的第三方私有扩展。

请注意，IaC 生成器仅支持与您所在区域的 Cloud Control API 兼容的 Amazon 资源。

要以编程方式访问有关公共和私有可预置资源类型的信息，您可以使用 Amazon 云端控制 API。有关更多信息，请参阅《Cloud Control API 用户指南》中的确定资源类型是否支持 Cloud Control API。

要开始使用导入、偏差检测或 IaC 生成器，请先查看以下一些有用的主题：

- 将 Amazon 资源导入 CloudFormation 堆栈
- 使用偏移检测功能检测堆栈和资源的非托管配置更改
- 使用 IaC 生成器为现有资源生成模板

资源	导入	偏差检测	IaC 生成器
AWS::ACMPCA::Certificate	✓ 是	✗ 否	✗ 否
AWS::ACMPCA::CertificateAuthority	✓ 是	✓ 是	✓ 是
AWS::ACMPCA::CertificateAuthorityActivation	✓ 是	✗ 否	✗ 否
AWS::ACMPCA::Permission	✓ 是	✓ 是	✗ 否
AWS::AIOps::InvestigationGroup	✓ 是	✓ 是	✓ 是
AWS::APS::AnomalyDetector	✓ 是	✓ 是	✓ 是
AWS::APS::ResourcePolicy	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::APS::RuleGroupsNamespace	✓ 是	✓ 是	✓ 是
AWS::APS::Scraper	✓ 是	✓ 是	✓ 是
AWS::APS::Workspace	✓ 是	✓ 是	✓ 是
AWS::ARCRegionSwitch::Plan	✓ 是	✓ 是	✓ 是
AWS::ARCZonalShift::AutoshiftObserverNotificationStatus	✓ 是	✓ 是	✓ 是
AWS::ARCZonalShift::ZonalAutoshiftConfiguration	✓ 是	✓ 是	✓ 是
AWS::AWSExternalAnthropic::Workspace	✓ 是	✓ 是	✗ 否
AWS::AccessAnalyzer::Analyzer	✓ 是	✓ 是	✓ 是
AWS::AmazonMQ::Broker	✓ 是	✓ 是	✓ 是
AWS::AmazonMQ::Configuration	✓ 是	✓ 是	✓ 是
AWS::Amplify::App	✓ 是	✓ 是	✓ 是
AWS::Amplify::Branch	✓ 是	✓ 是	✓ 是
AWS::Amplify::Domain	✓ 是	✓ 是	✓ 是
AWS::AmplifyUIBuilder::Component	✓ 是	✓ 是	✓ 是
AWS::AmplifyUIBuilder::Form	✓ 是	✓ 是	✓ 是
AWS::AmplifyUIBuilder::Theme	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::Account	✓ 是	✓ 是	✗ 否
AWS::ApiGateway::ApiKey	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::Authorizer	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::ApiGateway::BasePathMapping	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::BasePathMappingV2	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::ClientCertificate	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::Deployment	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::DocumentationPart	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::DocumentationVersion	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::DomainName	✓ 是	✗ 否	✓ 是
AWS::ApiGateway::DomainNameAccessAssociation	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::DomainNameV2	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::GatewayResponse	✓ 是	✗ 否	✓ 是
AWS::ApiGateway::Method	✓ 是	✓ 是	✗ 否
AWS::ApiGateway::Model	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::RequestValidator	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::Resource	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::RestApi	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::Stage	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::UsagePlan	✓ 是	✓ 是	✓ 是
AWS::ApiGateway::UsagePlanKey	✓ 是	✗ 否	✓ 是
AWS::ApiGateway::VpcLink	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::ApiGatewayV2::Api	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::ApiMapping	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::Authorizer	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::Deployment	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::DomainName	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::Integration	✓ 是	✗ 否	✓ 是
AWS::ApiGatewayV2::IntegrationResponse	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::Model	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::Route	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::RouteResponse	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::RoutingRule	✓ 是	✓ 是	✓ 是
AWS::ApiGatewayV2::Stage	✓ 是	✓ 是	✗ 否
AWS::ApiGatewayV2::VpcLink	✓ 是	✓ 是	✓ 是
AWS::AppConfig::Application	✓ 是	✓ 是	✓ 是
AWS::AppConfig::ConfigurationProfile	✓ 是	✓ 是	✓ 是
AWS::AppConfig::Deployment	✓ 是	✓ 是	✓ 是
AWS::AppConfig::DeploymentStrategy	✓ 是	✓ 是	✓ 是
AWS::AppConfig::Environment	✓ 是	✓ 是	✓ 是
AWS::AppConfig::Extension	✓ 是	✓ 是	✓ 是
AWS::AppConfig::ExtensionAssociation	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::AppConfig::HostedConfigurationVersion	✓ 是	✓ 是	✓ 是
AWS::AppFlow::Connector	✓ 是	✓ 是	✓ 是
AWS::AppFlow::ConnectorProfile	✓ 是	✓ 是	✓ 是
AWS::AppFlow::Flow	✓ 是	✓ 是	✓ 是
AWS::AppIntegrations::Application	✓ 是	✓ 是	✓ 是
AWS::AppIntegrations::DataIntegration	✓ 是	✓ 是	✓ 是
AWS::AppIntegrations::EventIntegration	✓ 是	✓ 是	✓ 是
AWS::AppRunner::AutoScalingConfiguration	✓ 是	✓ 是	✓ 是
AWS::AppRunner::ObservabilityConfiguration	✓ 是	✓ 是	✓ 是
AWS::AppRunner::Service	✓ 是	✓ 是	✓ 是
AWS::AppRunner::VpcConnector	✓ 是	✓ 是	✓ 是
AWS::AppRunner::VpcIngressConnection	✓ 是	✓ 是	✓ 是
AWS::AppStream::AppBlock	✓ 是	✓ 是	✗ 否
AWS::AppStream::AppBlockBuilder	✓ 是	✓ 是	✓ 是
AWS::AppStream::Application	✓ 是	✓ 是	✗ 否
AWS::AppStream::ApplicationEntitlementAssociation	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::AppStream::ApplicationFleetAssociation	✓ 是	✓ 是	✗ 否
AWS::AppStream::DirectoryConfig	✓ 是	✓ 是	✓ 是
AWS::AppStream::Entitlement	✓ 是	✓ 是	✗ 否
AWS::AppStream::ImageBuilder	✓ 是	✓ 是	✓ 是
AWS::AppStream::Stack	✓ 是	✓ 是	✗ 否
AWS::AppStream::StackFleetAssociation	✓ 是	✓ 是	✗ 否
AWS::AppStream::StackUserAssociation	✓ 是	✓ 是	✗ 否
AWS::AppStream::User	✓ 是	✓ 是	✗ 否
AWS::AppSync::Api	✓ 是	✓ 是	✓ 是
AWS::AppSync::ChannelNamespace	✓ 是	✓ 是	✓ 是
AWS::AppSync::DataSource	✓ 是	✓ 是	✓ 是
AWS::AppSync::DomainName	✓ 是	✓ 是	✓ 是
AWS::AppSync::DomainNameApiAssociation	✓ 是	✓ 是	✗ 否
AWS::AppSync::FunctionConfiguration	✓ 是	✓ 是	✓ 是
AWS::AppSync::GraphQLApi	✗ 否	✓ 是	✗ 否
AWS::AppSync::Resolver	✓ 是	✓ 是	✓ 是
AWS::AppSync::SourceApiAssociation	✓ 是	✓ 是	✓ 是
AWS::AppTest::TestCase	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::ApplicationAutoScaling::ScalableTarget	✓ 是	✓ 是	✓ 是
AWS::ApplicationAutoScaling::ScalingPolicy	✓ 是	✗ 否	✓ 是
AWS::ApplicationInsights::Application	✓ 是	✓ 是	✓ 是
AWS::ApplicationSignals::Discovery	✓ 是	✓ 是	✓ 是
AWS::ApplicationSignals::GroupingConfiguration	✓ 是	✓ 是	✓ 是
AWS::ApplicationSignals::ServiceLevelObjective	✓ 是	✓ 是	✓ 是
AWS::Athena::CapacityReservation	✓ 是	✓ 是	✓ 是
AWS::Athena::DataCatalog	✓ 是	✓ 是	✓ 是
AWS::Athena::NamedQuery	✓ 是	✓ 是	✓ 是
AWS::Athena::PreparedStatement	✓ 是	✓ 是	✓ 是
AWS::Athena::WorkGroup	✓ 是	✗ 否	✓ 是
AWS::AuditManager::Assessment	✓ 是	✗ 否	✓ 是
AWS::AutoScaling::AutoScalingGroup	✓ 是	✓ 是	✓ 是
AWS::AutoScaling::LaunchConfiguration	✓ 是	✓ 是	✓ 是
AWS::AutoScaling::LifecycleHook	✓ 是	✓ 是	✓ 是
AWS::AutoScaling::ScalingPolicy	✓ 是	✓ 是	✓ 是
AWS::AutoScaling::ScheduledAction	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::AutoScaling::WarmPool	✓ 是	✓ 是	✓ 是
AWS::B2BI::Capability	✓ 是	✓ 是	✓ 是
AWS::B2BI::Partnership	✓ 是	✓ 是	✓ 是
AWS::B2BI::Profile	✓ 是	✓ 是	✓ 是
AWS::B2BI::Transformer	✓ 是	✓ 是	✓ 是
AWS::BCM::Dashboard	✓ 是	✓ 是	✗ 否
AWS::BCMDataExports::Export	✓ 是	✓ 是	✓ 是
AWS::Backup::BackupPlan	✓ 是	✓ 是	✓ 是
AWS::Backup::BackupSelection	✓ 是	✗ 否	✓ 是
AWS::Backup::BackupVault	✓ 是	✓ 是	✓ 是
AWS::Backup::Framework	✓ 是	✓ 是	✓ 是
AWS::Backup::LogicallyAirGappedBackupVault	✓ 是	✓ 是	✓ 是
AWS::Backup::ReportPlan	✓ 是	✓ 是	✓ 是
AWS::Backup::RestoreTestingPlan	✓ 是	✓ 是	✓ 是
AWS::Backup::RestoreTestingSelection	✓ 是	✓ 是	✓ 是
AWS::Backup::TieringConfiguration	✓ 是	✓ 是	✗ 否
AWS::BackupGateway::Hypervisor	✓ 是	✓ 是	✓ 是
AWS::Batch::ComputeEnvironment	✓ 是	✓ 是	✓ 是
AWS::Batch::ConsumableResource	✓ 是	✓ 是	✓ 是
AWS::Batch::JobDefinition	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Batch::JobQueue	✓ 是	✓ 是	✓ 是
AWS::Batch::QuotaShare	✓ 是	✓ 是	✗ 否
AWS::Batch::SchedulingPolicy	✓ 是	✓ 是	✓ 是
AWS::Batch::ServiceEnvironment	✓ 是	✓ 是	✓ 是
AWS::BcmPricingCalculator::BillScenario	✓ 是	✓ 是	✗ 否
AWS::Bedrock::Agent	✓ 是	✓ 是	✓ 是
AWS::Bedrock::AgentAlias	✓ 是	✓ 是	✓ 是
AWS::Bedrock::ApplicationInferenceProfile	✓ 是	✓ 是	✓ 是
AWS::Bedrock::AutomatedReasoningPolicy	✓ 是	✓ 是	✓ 是
AWS::Bedrock::AutomatedReasoningPolicyVersion	✓ 是	✓ 是	✓ 是
AWS::Bedrock::Blueprint	✓ 是	✓ 是	✓ 是
AWS::Bedrock::DataAutomationLibrary	✓ 是	✓ 是	✗ 否
AWS::Bedrock::DataAutomationProject	✓ 是	✓ 是	✓ 是
AWS::Bedrock::DataSource	✓ 是	✓ 是	✓ 是
AWS::Bedrock::EnforcedGuardrailConfiguration	✓ 是	✓ 是	✗ 否
AWS::Bedrock::Flow	✓ 是	✓ 是	✓ 是
AWS::Bedrock::FlowAlias	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Bedrock::FlowVersion	✓ 是	✓ 是	✓ 是
AWS::Bedrock::Guardrail	✓ 是	✓ 是	✓ 是
AWS::Bedrock::GuardrailVersion	✓ 是	✓ 是	✗ 否
AWS::Bedrock::IntelligentPromptRouter	✓ 是	✓ 是	✓ 是
AWS::Bedrock::KnowledgeBase	✓ 是	✗ 否	✓ 是
AWS::Bedrock::Prompt	✓ 是	✓ 是	✓ 是
AWS::Bedrock::PromptVersion	✓ 是	✓ 是	✓ 是
AWS::Bedrock::ResourcePolicy	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::ApiKeyCredentialProvider	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::BrowserCustom	✓ 是	✓ 是	✓ 是
AWS::BedrockAgentCore::BrowserProfile	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::CodeInterpreterCustom	✓ 是	✓ 是	✓ 是
AWS::BedrockAgentCore::ConfigurationBundle	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::Dataset	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::Evaluator	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::Gateway	✓ 是	✓ 是	✓ 是
AWS::BedrockAgentCore::GatewayTarget	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::BedrockAgentCore::Harness	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::Memory	✓ 是	✓ 是	✓ 是
AWS::BedrockAgentCore::OAuth2CredentialProvider	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::OnlineEvaluationConfig	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::PaymentConnector	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::PaymentCredentialProvider	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::PaymentManager	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::Policy	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::PolicyEngine	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::ResourcePolicy	✓ 是	✓ 是	✗ 否
AWS::BedrockAgentCore::Runtime	✓ 是	✓ 是	✓ 是
AWS::BedrockAgentCore::RuntimeEndpoint	✓ 是	✓ 是	✓ 是
AWS::BedrockAgentCore::WorkloadIdentity	✓ 是	✓ 是	✓ 是
AWS::BedrockMantle::Project	✓ 是	✓ 是	✗ 否
AWS::Billing::BillingView	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::BillingConductor::BillingGroup	✓ 是	✓ 是	✓ 是
AWS::BillingConductor::CustomLineItem	✓ 是	✓ 是	✓ 是
AWS::BillingConductor::PricingPlan	✓ 是	✓ 是	✓ 是
AWS::BillingConductor::PricingRule	✓ 是	✓ 是	✓ 是
AWS::Braket::SpendingLimit	✓ 是	✓ 是	✗ 否
AWS::Budgets::BudgetsAction	✓ 是	✓ 是	✓ 是
AWS::CE::AnomalyMonitor	✓ 是	✓ 是	✓ 是
AWS::CE::AnomalySubscription	✓ 是	✓ 是	✓ 是
AWS::CE::CostCategory	✓ 是	✓ 是	✓ 是
AWS::CUR::ReportDefinition	✓ 是	✓ 是	✓ 是
AWS::Cases::CaseRule	✓ 是	✓ 是	✗ 否
AWS::Cases::Domain	✓ 是	✓ 是	✗ 否
AWS::Cases::Field	✓ 是	✓ 是	✗ 否
AWS::Cases::Layout	✓ 是	✓ 是	✗ 否
AWS::Cases::Template	✓ 是	✓ 是	✗ 否
AWS::Cassandra::Keyspace	✓ 是	✓ 是	✓ 是
AWS::Cassandra::Table	✓ 是	✓ 是	✓ 是
AWS::Cassandra::Type	✓ 是	✓ 是	✓ 是
AWS::CertificateManager::Account	✓ 是	✓ 是	✗ 否
AWS::Chatbot::CustomAction	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Chatbot::MicrosoftTeamsChannelConfiguration	✓ 是	✓ 是	✓ 是
AWS::Chatbot::SlackChannelConfiguration	✓ 是	✓ 是	✓ 是
AWS::Chime::AppInstance	✓ 是	✓ 是	✗ 否
AWS::Chime::AppInstanceBot	✓ 是	✓ 是	✗ 否
AWS::CleanRooms::AnalysisTemplate	✓ 是	✓ 是	✓ 是
AWS::CleanRooms::Collaboration	✓ 是	✓ 是	✓ 是
AWS::CleanRooms::ConfiguredTable	✓ 是	✓ 是	✓ 是
AWS::CleanRooms::ConfiguredTableAssociation	✓ 是	✓ 是	✓ 是
AWS::CleanRooms::IdMappingTable	✓ 是	✓ 是	✓ 是
AWS::CleanRooms::IdNamespaceAssociation	✓ 是	✓ 是	✓ 是
AWS::CleanRooms::Membership	✓ 是	✓ 是	✓ 是
AWS::CleanRooms::PrivacyBudgetTemplate	✓ 是	✓ 是	✓ 是
AWS::CleanRoomsML::ConfiguredModelAlgorithm	✓ 是	✓ 是	✗ 否
AWS::CleanRoomsML::ConfiguredModelAlgorithmAssociation	✓ 是	✓ 是	✗ 否
AWS::CleanRoomsML::TrainingDataset	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::GuardHook	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::CloudFormation::HookDefaultVersion	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::HookTypeConfig	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::HookVersion	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::LambdaHook	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::ModuleDefaultVersion	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::ModuleVersion	✓ 是	✓ 是	✗ 否
AWS::CloudFormation::PublicTypeVersion	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::Publisher	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::ResourceDefaultVersion	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::ResourceVersion	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::Stack	✓ 是	✗ 否	✓ 是
AWS::CloudFormation::StackSet	✓ 是	✓ 是	✓ 是
AWS::CloudFormation::TypeActivation	✓ 是	✓ 是	✓ 是
AWS::CloudFront::AnycastIpList	✓ 是	✓ 是	✓ 是
AWS::CloudFront::CachePolicy	✓ 是	✓ 是	✓ 是
AWS::CloudFront::CloudFrontOriginAccessIdentity	✓ 是	✓ 是	✓ 是
AWS::CloudFront::ConnectionFunction	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::CloudFront::ConnectionGroup	✓ 是	✓ 是	✓ 是
AWS::CloudFront::ContinuousDeploymentPolicy	✓ 是	✓ 是	✓ 是
AWS::CloudFront::Distribution	✓ 是	✓ 是	✓ 是
AWS::CloudFront::DistributionTenant	✓ 是	✓ 是	✓ 是
AWS::CloudFront::Function	✓ 是	✓ 是	✓ 是
AWS::CloudFront::KeyGroup	✓ 是	✓ 是	✓ 是
AWS::CloudFront::KeyValueStore	✓ 是	✓ 是	✓ 是
AWS::CloudFront::MonitoringSubscription	✓ 是	✓ 是	✗ 否
AWS::CloudFront::OriginAccessControl	✓ 是	✓ 是	✓ 是
AWS::CloudFront::OriginRequestPolicy	✓ 是	✓ 是	✓ 是
AWS::CloudFront::PublicKey	✓ 是	✓ 是	✓ 是
AWS::CloudFront::RealtimeLogConfig	✓ 是	✓ 是	✓ 是
AWS::CloudFront::ResponseHeadersPolicy	✓ 是	✓ 是	✓ 是
AWS::CloudFront::TrustStore	✓ 是	✓ 是	✓ 是
AWS::CloudFront::VpcOrigin	✓ 是	✓ 是	✓ 是
AWS::CloudTrail::Channel	✓ 是	✓ 是	✓ 是
AWS::CloudTrail::Dashboard	✓ 是	✓ 是	✓ 是
AWS::CloudTrail::EventDataStore	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::CloudTrail::ResourcePolicy	✓ 是	✓ 是	✗ 否
AWS::CloudTrail::Trail	✓ 是	✓ 是	✓ 是
AWS::CloudWatch::Alarm	✓ 是	✓ 是	✓ 是
AWS::CloudWatch::AlarmMuteRule	✓ 是	✓ 是	✗ 否
AWS::CloudWatch::CompositeAlarm	✓ 是	✓ 是	✓ 是
AWS::CloudWatch::Dashboard	✓ 是	✓ 是	✓ 是
AWS::CloudWatch::LogAlarm	✓ 是	✓ 是	✗ 否
AWS::CloudWatch::MetricStream	✓ 是	✓ 是	✓ 是
AWS::CloudWatch::OTelEnrichment	✓ 是	✓ 是	✗ 否
AWS::CodeArtifact::Domain	✓ 是	✓ 是	✓ 是
AWS::CodeArtifact::PackageGroup	✓ 是	✓ 是	✓ 是
AWS::CodeArtifact::Repository	✓ 是	✓ 是	✓ 是
AWS::CodeBuild::Fleet	✓ 是	✓ 是	✓ 是
AWS::CodeConnections::Connection	✓ 是	✓ 是	✓ 是
AWS::CodeDeploy::Application	✓ 是	✓ 是	✓ 是
AWS::CodeDeploy::DeploymentConfig	✓ 是	✓ 是	✓ 是
AWS::CodeDeploy::DeploymentGroup	✓ 是	✓ 是	✗ 否
AWS::CodeGuruProfiler::ProfilingGroup	✓ 是	✓ 是	✓ 是
AWS::CodeGuruReviewer::RepositoryAssociation	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::CodePipeline::CustomActionType	✓ 是	✓ 是	✓ 是
AWS::CodePipeline::Pipeline	✓ 是	✓ 是	✓ 是
AWS::CodePipeline::Webhook	✓ 是	✓ 是	✓ 是
AWS::CodeStarConnections::Connection	✓ 是	✓ 是	✓ 是
AWS::CodeStarConnections::RepositoryLink	✓ 是	✓ 是	✓ 是
AWS::CodeStarConnections::SyncConfiguration	✓ 是	✓ 是	✓ 是
AWS::CodeStarNotifications::NotificationRule	✓ 是	✓ 是	✓ 是
AWS::Cognito::IdentityPool	✓ 是	✓ 是	✓ 是
AWS::Cognito::IdentityPoolPrincipalTag	✓ 是	✓ 是	✓ 是
AWS::Cognito::IdentityPoolRoleAttachment	✓ 是	✗ 否	✓ 是
AWS::Cognito::LogDeliveryConfiguration	✓ 是	✓ 是	✗ 否
AWS::Cognito::ManagedLoginBranding	✓ 是	✓ 是	✗ 否
AWS::Cognito::Terms	✓ 是	✓ 是	✓ 是
AWS::Cognito::UserPool	✓ 是	✗ 否	✓ 是
AWS::Cognito::UserPoolClient	✓ 是	✓ 是	✓ 是
AWS::Cognito::UserPoolDomain	✓ 是	✓ 是	✗ 否
AWS::Cognito::UserPoolGroup	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Cognito::UserPoolIdentityProvider	✓ 是	✓ 是	✓ 是
AWS::Cognito::UserPoolResourceServer	✓ 是	✓ 是	✓ 是
AWS::Cognito::UserPoolRiskConfigurationAttachment	✓ 是	✓ 是	✗ 否
AWS::Cognito::UserPoolUICustomizationAttachment	✓ 是	✓ 是	✗ 否
AWS::Cognito::UserPoolUser	✓ 是	✗ 否	✓ 是
AWS::Cognito::UserPoolUserToGroupAttachment	✓ 是	✓ 是	✗ 否
AWS::Comprehend::DocumentClassifier	✓ 是	✓ 是	✓ 是
AWS::Comprehend::Flywheel	✓ 是	✓ 是	✓ 是
AWS::ComputeOptimizer::AutomationRule	✓ 是	✓ 是	✗ 否
AWS::Config::AggregationAuthorization	✓ 是	✓ 是	✓ 是
AWS::Config::ConfigRule	✓ 是	✓ 是	✓ 是
AWS::Config::ConfigurationAggregator	✓ 是	✓ 是	✓ 是
AWS::Config::ConformancePack	✓ 是	✓ 是	✓ 是
AWS::Config::OrganizationConformancePack	✓ 是	✓ 是	✓ 是
AWS::Config::RemediationConfiguration	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::Config::StoredQuery	✓ 是	✓ 是	✓ 是
AWS::Connect::AgentStatus	✓ 是	✓ 是	✓ 是
AWS::Connect::ApprovedOrigin	✓ 是	✓ 是	✓ 是
AWS::Connect::ContactFlow	✓ 是	✓ 是	✓ 是
AWS::Connect::ContactFlowModule	✓ 是	✓ 是	✓ 是
AWS::Connect::ContactFlowModuleAlias	✓ 是	✓ 是	✗ 否
AWS::Connect::ContactFlowModuleVersion	✓ 是	✓ 是	✗ 否
AWS::Connect::ContactFlowVersion	✓ 是	✓ 是	✓ 是
AWS::Connect::DataTable	✓ 是	✓ 是	✓ 是
AWS::Connect::DataTableAttribute	✓ 是	✓ 是	✗ 否
AWS::Connect::DataTableRecord	✓ 是	✓ 是	✗ 否
AWS::Connect::EmailAddress	✓ 是	✓ 是	✓ 是
AWS::Connect::EvaluationForm	✓ 是	✓ 是	✓ 是
AWS::Connect::HoursOfOperation	✓ 是	✓ 是	✓ 是
AWS::Connect::Instance	✓ 是	✓ 是	✓ 是
AWS::Connect::InstanceStorageConfig	✓ 是	✓ 是	✓ 是
AWS::Connect::IntegrationAssociation	✓ 是	✓ 是	✓ 是
AWS::Connect::Notification	✓ 是	✓ 是	✗ 否
AWS::Connect::PhoneNumber	✓ 是	✓ 是	✓ 是
AWS::Connect::PredefinedAttribute	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Connect::Prompt	✓ 是	✓ 是	✓ 是
AWS::Connect::Queue	✓ 是	✓ 是	✓ 是
AWS::Connect::QuickConnect	✓ 是	✓ 是	✓ 是
AWS::Connect::RoutingProfile	✓ 是	✓ 是	✓ 是
AWS::Connect::Rule	✓ 是	✓ 是	✗ 否
AWS::Connect::SecurityKey	✓ 是	✓ 是	✓ 是
AWS::Connect::SecurityProfile	✓ 是	✓ 是	✓ 是
AWS::Connect::TaskTemplate	✓ 是	✓ 是	✓ 是
AWS::Connect::TrafficDistributionGroup	✓ 是	✓ 是	✓ 是
AWS::Connect::User	✓ 是	✓ 是	✓ 是
AWS::Connect::UserHierarchyGroup	✓ 是	✓ 是	✓ 是
AWS::Connect::UserHierarchyStructure	✓ 是	✓ 是	✗ 否
AWS::Connect::View	✓ 是	✓ 是	✓ 是
AWS::Connect::ViewVersion	✓ 是	✓ 是	✓ 是
AWS::Connect::Workspace	✓ 是	✓ 是	✗ 否
AWS::ConnectCampaigns::Campaign	✓ 是	✗ 否	✓ 是
AWS::ConnectCampaignsV2::Campaign	✓ 是	✓ 是	✓ 是
AWS::ControlTower::EnabledBaseline	✓ 是	✓ 是	✓ 是
AWS::ControlTower::EnabledControl	✓ 是	✓ 是	✓ 是
AWS::ControlTower::LandingZone	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::CustomerProfiles::CalculatedAttributeDefinition	✓ 是	✓ 是	✓ 是
AWS::CustomerProfiles::Domain	✓ 是	✓ 是	✓ 是
AWS::CustomerProfiles::DomainObjectType	✓ 是	✓ 是	✗ 否
AWS::CustomerProfiles::EventStream	✓ 是	✓ 是	✓ 是
AWS::CustomerProfiles::EventTrigger	✓ 是	✓ 是	✓ 是
AWS::CustomerProfiles::Integration	✓ 是	✓ 是	✓ 是
AWS::CustomerProfiles::ObjectType	✓ 是	✓ 是	✓ 是
AWS::CustomerProfiles::Recommender	✓ 是	✓ 是	✗ 否
AWS::CustomerProfiles::SegmentDefinition	✓ 是	✓ 是	✓ 是
AWS::DMS::DataMigration	✓ 是	✓ 是	✓ 是
AWS::DMS::DataProvider	✓ 是	✓ 是	✓ 是
AWS::DMS::InstanceProfile	✓ 是	✓ 是	✓ 是
AWS::DMS::MigrationProject	✓ 是	✓ 是	✓ 是
AWS::DMS::ReplicationConfig	✓ 是	✓ 是	✓ 是
AWS::DSQL::Cluster	✓ 是	✓ 是	✓ 是
AWS::DataBrew::Dataset	✓ 是	✓ 是	✓ 是
AWS::DataBrew::Job	✓ 是	✓ 是	✓ 是
AWS::DataBrew::Project	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::DataBrew::Recipe	✓ 是	✓ 是	✓ 是
AWS::DataBrew::Ruleset	✓ 是	✓ 是	✓ 是
AWS::DataBrew::Schedule	✓ 是	✓ 是	✓ 是
AWS::DataPipeline::Pipeline	✓ 是	✓ 是	✓ 是
AWS::DataSync::Agent	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationAzureBlob	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationEFS	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationFSxLustre	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationFSxONTAP	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationFSxOpenZFS	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationFSxWindows	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationHDFS	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationNFS	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationObjectStorage	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationS3	✓ 是	✓ 是	✓ 是
AWS::DataSync::LocationSMB	✓ 是	✓ 是	✓ 是
AWS::DataSync::Task	✓ 是	✓ 是	✓ 是
AWS::DataZone::Connection	✓ 是	✓ 是	✓ 是
AWS::DataZone::DataSource	✓ 是	✓ 是	✓ 是
AWS::DataZone::Domain	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::DataZone::DomainUnit	✓ 是	✓ 是	✓ 是
AWS::DataZone::Environment	✓ 是	✓ 是	✓ 是
AWS::DataZone::EnvironmentActions	✓ 是	✓ 是	✓ 是
AWS::DataZone::EnvironmentBlueprintConfiguration	✓ 是	✓ 是	✓ 是
AWS::DataZone::EnvironmentProfile	✓ 是	✓ 是	✓ 是
AWS::DataZone::FormType	✓ 是	✓ 是	✗ 否
AWS::DataZone::GroupProfile	✓ 是	✓ 是	✓ 是
AWS::DataZone::Owner	✓ 是	✓ 是	✓ 是
AWS::DataZone::PolicyGrant	✓ 是	✓ 是	✓ 是
AWS::DataZone::Project	✓ 是	✓ 是	✓ 是
AWS::DataZone::ProjectMembership	✓ 是	✓ 是	✓ 是
AWS::DataZone::ProjectProfile	✓ 是	✓ 是	✓ 是
AWS::DataZone::SubscriptionTarget	✓ 是	✓ 是	✓ 是
AWS::DataZone::UserProfile	✓ 是	✓ 是	✓ 是
AWS::Deadline::Farm	✓ 是	✓ 是	✓ 是
AWS::Deadline::Fleet	✓ 是	✓ 是	✓ 是
AWS::Deadline::LicenseEndpoint	✓ 是	✓ 是	✓ 是
AWS::Deadline::Limit	✓ 是	✓ 是	✓ 是
AWS::Deadline::MeteredProduct	✓ 是	✓ 是	✓ 是
AWS::Deadline::Monitor	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Deadline::Queue	✓ 是	✓ 是	✓ 是
AWS::Deadline::QueueEnvironment	✓ 是	✓ 是	✓ 是
AWS::Deadline::QueueFleetAssociation	✓ 是	✓ 是	✓ 是
AWS::Deadline::QueueLimitAssociation	✓ 是	✓ 是	✓ 是
AWS::Deadline::StorageProfile	✓ 是	✓ 是	✓ 是
AWS::Detective::Graph	✓ 是	✓ 是	✓ 是
AWS::Detective::MemberInvitation	✓ 是	✓ 是	✓ 是
AWS::Detective::OrganizationAdmin	✓ 是	✓ 是	✓ 是
AWS::DevOpsAgent::AgentSpace	✓ 是	✓ 是	✓ 是
AWS::DevOpsAgent::Association	✓ 是	✓ 是	✓ 是
AWS::DevOpsAgent::PrivateConnection	✓ 是	✓ 是	✗ 否
AWS::DevOpsAgent::Service	✓ 是	✓ 是	✗ 否
AWS::DevOpsGuru::LogAnomalyDetectionIntegration	✓ 是	✓ 是	✓ 是
AWS::DevOpsGuru::NotificationChannel	✓ 是	✓ 是	✓ 是
AWS::DevOpsGuru::ResourceCollection	✓ 是	✓ 是	✓ 是
AWS::DeviceFarm::DevicePool	✓ 是	✓ 是	✓ 是
AWS::DeviceFarm::InstanceProfile	✓ 是	✓ 是	✓ 是
AWS::DeviceFarm::NetworkProfile	✓ 是	✓ 是	✓ 是
AWS::DeviceFarm::Project	✓ 是	✓ 是	✓ 是
AWS::DeviceFarm::TestGridProject	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::DeviceFarm::VPCEConfiguration	✓ 是	✓ 是	✓ 是
AWS::DirectConnect::Connection	✓ 是	✓ 是	✗ 否
AWS::DirectConnect::DirectConnectGateway	✓ 是	✓ 是	✗ 否
AWS::DirectConnect::DirectConnectGatewayAssociation	✓ 是	✓ 是	✗ 否
AWS::DirectConnect::Lag	✓ 是	✓ 是	✗ 否
AWS::DirectConnect::PrivateVirtualInterface	✓ 是	✓ 是	✗ 否
AWS::DirectConnect::PublicVirtualInterface	✓ 是	✓ 是	✗ 否
AWS::DirectConnect::TransitVirtualInterface	✓ 是	✓ 是	✗ 否
AWS::DirectoryService::SimpleAD	✓ 是	✓ 是	✓ 是
AWS::DocDB::GlobalCluster	✓ 是	✓ 是	✗ 否
AWS::DocDBElastic::Cluster	✓ 是	✓ 是	✓ 是
AWS::DynamoDB::GlobalTable	✓ 是	✓ 是	✓ 是
AWS::DynamoDB::Table	✓ 是	✓ 是	✓ 是
AWS::EC2::CapacityManagerDataExport	✓ 是	✓ 是	✓ 是
AWS::EC2::CapacityReservation	✓ 是	✓ 是	✓ 是
AWS::EC2::CapacityReservationFleet	✓ 是	✓ 是	✓ 是
AWS::EC2::CarrierGateway	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::EC2::CustomerGateway	✓ 是	✓ 是	✓ 是
AWS::EC2::DHCPOptions	✓ 是	✓ 是	✓ 是
AWS::EC2::EC2Fleet	✓ 是	✓ 是	✓ 是
AWS::EC2::EIP	✓ 是	✓ 是	✓ 是
AWS::EC2::EIPAssociation	✓ 是	✗ 否	✓ 是
AWS::EC2::EgressOnlyInternetGateway	✓ 是	✓ 是	✓ 是
AWS::EC2::EnclaveCertificateIamRoleAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::FlowLog	✓ 是	✓ 是	✓ 是
AWS::EC2::GatewayRouteTableAssociation	✓ 是	✓ 是	✗ 否
AWS::EC2::Host	✓ 是	✓ 是	✓ 是
AWS::EC2::IPAM	✓ 是	✓ 是	✓ 是
AWS::EC2::IPAMAllocation	✓ 是	✓ 是	✓ 是
AWS::EC2::IPAMPool	✓ 是	✓ 是	✓ 是
AWS::EC2::IPAMPoolCidr	✓ 是	✓ 是	✓ 是
AWS::EC2::IPAMPrefixListResolver	✓ 是	✓ 是	✗ 否
AWS::EC2::IPAMPrefixListResolverTarget	✓ 是	✓ 是	✗ 否
AWS::EC2::IPAMResourceDiscovery	✓ 是	✓ 是	✓ 是
AWS::EC2::IPAMResourceDiscoveryAssociation	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::EC2::IPAMScope	✓ 是	✓ 是	✓ 是
AWS::EC2::Instance	✓ 是	✓ 是	✓ 是
AWS::EC2::InstanceConnectEndpoint	✓ 是	✓ 是	✓ 是
AWS::EC2::InternetGateway	✓ 是	✓ 是	✓ 是
AWS::EC2::IpPoolRouteTableAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::KeyPair	✓ 是	✓ 是	✓ 是
AWS::EC2::LaunchTemplate	✓ 是	✓ 是	✓ 是
AWS::EC2::LocalGatewayRoute	✓ 是	✓ 是	✓ 是
AWS::EC2::LocalGatewayRouteTable	✓ 是	✓ 是	✓ 是
AWS::EC2::LocalGatewayRouteTableVPCAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::LocalGatewayVirtualInterface	✓ 是	✓ 是	✓ 是
AWS::EC2::LocalGatewayVirtualInterfaceGroup	✓ 是	✓ 是	✓ 是
AWS::EC2::NatGateway	✓ 是	✓ 是	✓ 是
AWS::EC2::NetworkAcl	✓ 是	✓ 是	✓ 是
AWS::EC2::NetworkInsightsAccessScope	✓ 是	✓ 是	✓ 是
AWS::EC2::NetworkInsightsAccessScopeAnalysis	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::EC2::NetworkInsightsAnalysis	✓ 是	✓ 是	✓ 是
AWS::EC2::NetworkInsightsPath	✓ 是	✓ 是	✓ 是
AWS::EC2::NetworkInterface	✓ 是	✓ 是	✓ 是
AWS::EC2::NetworkInterfaceAttachment	✓ 是	✓ 是	✓ 是
AWS::EC2::NetworkPerformanceMetricSubscription	✓ 是	✓ 是	✓ 是
AWS::EC2::PlacementGroup	✓ 是	✓ 是	✓ 是
AWS::EC2::PrefixList	✓ 是	✓ 是	✓ 是
AWS::EC2::Route	✓ 是	✓ 是	✓ 是
AWS::EC2::RouteServer	✓ 是	✓ 是	✓ 是
AWS::EC2::RouteServerAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::RouteServerEndpoint	✓ 是	✓ 是	✓ 是
AWS::EC2::RouteServerPeer	✓ 是	✓ 是	✓ 是
AWS::EC2::RouteServerPropagation	✓ 是	✓ 是	✓ 是
AWS::EC2::RouteTable	✓ 是	✓ 是	✓ 是
AWS::EC2::SecurityGroup	✓ 是	✓ 是	✓ 是
AWS::EC2::SecurityGroupEgress	✓ 是	✗ 否	✓ 是
AWS::EC2::SecurityGroupIngress	✓ 是	✗ 否	✓ 是
AWS::EC2::SecurityGroupVpcAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::SnapshotBlockPublicAccess	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::EC2::SpotFleet	✓ 是	✓ 是	✓ 是
AWS::EC2::SqlHaStandbyDetectedInstance	✓ 是	✓ 是	✗ 否
AWS::EC2::Subnet	✓ 是	✓ 是	✓ 是
AWS::EC2::SubnetCidrBlock	✓ 是	✗ 否	✓ 是
AWS::EC2::SubnetNetworkAclAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::SubnetRouteTableAssociation	✓ 是	✗ 否	✓ 是
AWS::EC2::TrafficMirrorFilter	✓ 是	✓ 是	✓ 是
AWS::EC2::TrafficMirrorFilterRule	✓ 是	✓ 是	✓ 是
AWS::EC2::TrafficMirrorSession	✓ 是	✓ 是	✓ 是
AWS::EC2::TrafficMirrorTarget	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGateway	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayAttachment	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayConnect	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayConnectPeer	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayMeteringPolicy	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayMeteringPolicyEntry	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayMulticastDomain	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::EC2::TransitGatewayMulticastDomainAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayMulticastGroupMember	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayMulticastGroupSource	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayPeeringAttachment	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayRoute	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayRouteTable	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayRouteTableAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayRouteTablePropagation	✓ 是	✓ 是	✓ 是
AWS::EC2::TransitGatewayVpcAttachment	✓ 是	✓ 是	✓ 是
AWS::EC2::VPC	✓ 是	✓ 是	✓ 是
AWS::EC2::VPCBlockPublicAccessExclusion	✓ 是	✓ 是	✓ 是
AWS::EC2::VPCBlockPublicAccessOptions	✓ 是	✓ 是	✗ 否
AWS::EC2::VPCCidrBlock	✓ 是	✓ 是	✓ 是
AWS::EC2::VPCDHCPOptionsAssociation	✓ 是	✓ 是	✓ 是
AWS::EC2::VPCEncryptionControl	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::EC2::VPCEndpoint	是	否	是
AWS::EC2::VPCEndpointConnectionNotification	是	是	是
AWS::EC2::VPCEndpointService	是	是	是
AWS::EC2::VPCEndpointServicePermissions	是	是	是
AWS::EC2::VPCGatewayAttachment	是	否	是
AWS::EC2::VPCPeeringConnection	是	是	是
AWS::EC2::VPNConcentrator	是	是	是
AWS::EC2::VPNConnection	是	是	是
AWS::EC2::VPNConnectionRoute	是	是	是
AWS::EC2::VPNGateway	是	是	是
AWS::EC2::VerifiedAccessEndpoint	是	是	是
AWS::EC2::VerifiedAccessGroup	是	是	是
AWS::EC2::VerifiedAccessInstance	是	是	是
AWS::EC2::VerifiedAccessTrustProvider	是	是	是
AWS::EC2::Volume	是	是	是
AWS::EC2::VolumeAttachment	是	是	是
AWS::ECR::PublicRepository	是	是	是
AWS::ECR::PullThroughCacheRule	是	是	是

资源	导入	偏差检测	IaC 生成器
AWS::ECR::PullTimeUpdateExclusion	✓ 是	✓ 是	✓ 是
AWS::ECR::RegistryPolicy	✓ 是	✓ 是	✓ 是
AWS::ECR::RegistryScanningConfiguration	✓ 是	✓ 是	✓ 是
AWS::ECR::ReplicationConfiguration	✓ 是	✓ 是	✓ 是
AWS::ECR::Repository	✓ 是	✓ 是	✓ 是
AWS::ECR::RepositoryCreationTemplate	✓ 是	✓ 是	✓ 是
AWS::ECR::SigningConfiguration	✓ 是	✓ 是	✓ 是
AWS::ECS::CapacityProvider	✓ 是	✓ 是	✓ 是
AWS::ECS::Cluster	✓ 是	✓ 是	✓ 是
AWS::ECS::ClusterCapacityProviderAssociations	✓ 是	✗ 否	✓ 是
AWS::ECS::Daemon	✓ 是	✓ 是	✗ 否
AWS::ECS::DaemonTaskDefinition	✓ 是	✓ 是	✗ 否
AWS::ECS::ExpressGatewayService	✓ 是	✓ 是	✗ 否
AWS::ECS::PrimaryTaskSet	✓ 是	✓ 是	✗ 否
AWS::ECS::Service	✓ 是	✓ 是	✓ 是
AWS::ECS::TaskDefinition	✓ 是	✓ 是	✓ 是
AWS::ECS::TaskSet	✓ 是	✓ 是	✗ 否
AWS::EFS::AccessPoint	✓ 是	✓ 是	✓ 是
AWS::EFS::FileSystem	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::EFS::MountTarget	✓ 是	✓ 是	✓ 是
AWS::EKS::AccessEntry	✓ 是	✓ 是	✓ 是
AWS::EKS::Addon	✓ 是	✓ 是	✓ 是
AWS::EKS::Capability	✓ 是	✓ 是	✓ 是
AWS::EKS::Cluster	✓ 是	✗ 否	✓ 是
AWS::EKS::FargateProfile	✓ 是	✓ 是	✓ 是
AWS::EKS::IdentityProviderConfig	✓ 是	✓ 是	✓ 是
AWS::EKS::Nodegroup	✓ 是	✗ 否	✓ 是
AWS::EKS::PodIdentityAssociation	✓ 是	✓ 是	✓ 是
AWS::EMR::SecurityConfiguration	✓ 是	✓ 是	✓ 是
AWS::EMR::Step	✓ 是	✓ 是	✗ 否
AWS::EMR::Studio	✓ 是	✓ 是	✓ 是
AWS::EMR::StudioSessionMapping	✓ 是	✓ 是	✓ 是
AWS::EMR::WALWorkspace	✓ 是	✓ 是	✓ 是
AWS::EMRContainers::Endpoint	✓ 是	✓ 是	✗ 否
AWS::EMRContainers::SecurityConfiguration	✓ 是	✓ 是	✗ 否
AWS::EMRContainers::VirtualCluster	✓ 是	✓ 是	✓ 是
AWS::EMRServerless::Application	✓ 是	✓ 是	✓ 是
AWS::EVS::Environment	✓ 是	✓ 是	✓ 是
AWS::ElastiCache::CacheCluster	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::ElastiCache::GlobalReplicationGroup	✓ 是	✓ 是	✓ 是
AWS::ElastiCache::ParameterGroup	✓ 是	✓ 是	✓ 是
AWS::ElastiCache::ReplicationGroup	✓ 是	✓ 是	✗ 否
AWS::ElastiCache::ServerlessCache	✓ 是	✓ 是	✓ 是
AWS::ElastiCache::SubnetGroup	✓ 是	✓ 是	✓ 是
AWS::ElastiCache::User	✓ 是	✓ 是	✓ 是
AWS::ElastiCache::UserGroup	✓ 是	✓ 是	✓ 是
AWS::ElasticBeanstalk::Application	✓ 是	✓ 是	✓ 是
AWS::ElasticBeanstalk::ApplicationVersion	✓ 是	✓ 是	✓ 是
AWS::ElasticBeanstalk::ConfigurationTemplate	✓ 是	✓ 是	✓ 是
AWS::ElasticBeanstalk::Environment	✓ 是	✓ 是	✓ 是
AWS::ElasticLoadBalancing::LoadBalancer	✓ 是	✓ 是	✗ 否
AWS::ElasticLoadBalancingV2::Listener	✓ 是	✓ 是	✓ 是
AWS::ElasticLoadBalancingV2::ListenerRule	✓ 是	✓ 是	✓ 是
AWS::ElasticLoadBalancingV2::LoadBalancer	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::ElasticLoadBalancingV2::TargetGroup	✓ 是	✓ 是	✓ 是
AWS::ElasticLoadBalancingV2::TrustStore	✓ 是	✓ 是	✓ 是
AWS::ElasticLoadBalancingV2::TrustStoreRevocation	✓ 是	✓ 是	✓ 是
AWS::ElementalInference::Dictionary	✓ 是	✓ 是	✗ 否
AWS::ElementalInference::Feed	✓ 是	✓ 是	✗ 否
AWS::EntityResolution::IdMappingWorkflow	✓ 是	✓ 是	✓ 是
AWS::EntityResolution::IdNamespace	✓ 是	✓ 是	✓ 是
AWS::EntityResolution::MatchingWorkflow	✓ 是	✓ 是	✓ 是
AWS::EntityResolution::PolicyStatement	✓ 是	✓ 是	✓ 是
AWS::EntityResolution::SchemaMapping	✓ 是	✓ 是	✓ 是
AWS::EventSchemas::Discoverer	✓ 是	✓ 是	✓ 是
AWS::EventSchemas::Registry	✓ 是	✓ 是	✓ 是
AWS::EventSchemas::RegistryPolicy	✓ 是	✓ 是	✗ 否
AWS::EventSchemas::Schema	✓ 是	✓ 是	✓ 是
AWS::Events::ApiDestination	✓ 是	✓ 是	✓ 是
AWS::Events::Archive	✓ 是	✓ 是	✓ 是
AWS::Events::Connection	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Events::Endpoint	✓ 是	✓ 是	✓ 是
AWS::Events::EventBus	✓ 是	✓ 是	✓ 是
AWS::Events::EventBusPolicy	✓ 是	✗ 否	✓ 是
AWS::Events::Rule	✓ 是	✓ 是	✓ 是
AWS::Evidently::Experiment	✓ 是	✓ 是	✗ 否
AWS::Evidently::Feature	✓ 是	✗ 否	✗ 否
AWS::Evidently::Launch	✓ 是	✓ 是	✗ 否
AWS::Evidently::Project	✓ 是	✓ 是	✗ 否
AWS::Evidently::Segment	✓ 是	✓ 是	✓ 是
AWS::FIS::ExperimentTemplate	✓ 是	✓ 是	✓ 是
AWS::FIS::TargetAccountConfiguration	✓ 是	✓ 是	✓ 是
AWS::FMS::NotificationChannel	✓ 是	✓ 是	✓ 是
AWS::FMS::Policy	✓ 是	✗ 否	✓ 是
AWS::FMS::ResourceSet	✓ 是	✓ 是	✓ 是
AWS::FSx::DataRepositoryAssociation	✓ 是	✓ 是	✓ 是
AWS::FSx::S3AccessPointAttachment	✓ 是	✓ 是	✓ 是
AWS::FinSpace::Environment	✓ 是	✓ 是	✓ 是
AWS::Forecast::Dataset	✓ 是	✓ 是	✓ 是
AWS::Forecast::DatasetGroup	✓ 是	✓ 是	✓ 是
AWS::FraudDetector::Detector	✓ 是	✗ 否	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::FraudDetector::EntityType	✓ 是	✓ 是	✓ 是
AWS::FraudDetector::EventType	✓ 是	✓ 是	✓ 是
AWS::FraudDetector::Label	✓ 是	✓ 是	✓ 是
AWS::FraudDetector::List	✓ 是	✓ 是	✓ 是
AWS::FraudDetector::Outcome	✓ 是	✓ 是	✓ 是
AWS::FraudDetector::Variable	✓ 是	✓ 是	✓ 是
AWS::GameLift::Alias	✓ 是	✓ 是	✓ 是
AWS::GameLift::Build	✓ 是	✓ 是	✓ 是
AWS::GameLift::ContainerFleet	✓ 是	✓ 是	✓ 是
AWS::GameLift::ContainerGroupDefinition	✓ 是	✓ 是	✓ 是
AWS::GameLift::Fleet	✓ 是	✓ 是	✓ 是
AWS::GameLift::GameServerGroup	✓ 是	✓ 是	✓ 是
AWS::GameLift::GameSessionQueue	✓ 是	✓ 是	✓ 是
AWS::GameLift::Location	✓ 是	✓ 是	✓ 是
AWS::GameLift::MatchmakingConfiguration	✓ 是	✓ 是	✓ 是
AWS::GameLift::MatchmakingRuleSet	✓ 是	✓ 是	✓ 是
AWS::GameLift::Script	✓ 是	✓ 是	✓ 是
AWS::GameLiftStreams::Application	✓ 是	✓ 是	✗ 否
AWS::GameLiftStreams::StreamGroup	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::GlobalAccelerator::Accelerator	✓ 是	✓ 是	✓ 是
AWS::GlobalAccelerator::CrossAccountAttachment	✓ 是	✓ 是	✓ 是
AWS::GlobalAccelerator::EndpointGroup	✓ 是	✓ 是	✓ 是
AWS::GlobalAccelerator::Listener	✓ 是	✓ 是	✓ 是
AWS::Glue::Catalog	✓ 是	✓ 是	✗ 否
AWS::Glue::Crawler	✓ 是	✗ 否	✓ 是
AWS::Glue::Database	✓ 是	✓ 是	✓ 是
AWS::Glue::IdentityCenterConfiguration	✓ 是	✓ 是	✗ 否
AWS::Glue::Integration	✓ 是	✓ 是	✓ 是
AWS::Glue::IntegrationResourceProperty	✓ 是	✓ 是	✓ 是
AWS::Glue::Job	✓ 是	✓ 是	✓ 是
AWS::Glue::Registry	✓ 是	✓ 是	✓ 是
AWS::Glue::Schema	✓ 是	✓ 是	✓ 是
AWS::Glue::SchemaVersion	✓ 是	✓ 是	✓ 是
AWS::Glue::SchemaVersionMetadata	✓ 是	✓ 是	✓ 是
AWS::Glue::Trigger	✓ 是	✓ 是	✓ 是
AWS::Glue::UsageProfile	✓ 是	✓ 是	✓ 是
AWS::Grafana::Workspace	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::GreengrassV2::ComponentVersion	✓ 是	✓ 是	✓ 是
AWS::GreengrassV2::Deployment	✓ 是	✓ 是	✓ 是
AWS::GroundStation::Config	✓ 是	✓ 是	✓ 是
AWS::GroundStation::DataflowEndpointGroup	✓ 是	✓ 是	✓ 是
AWS::GroundStation::DataflowEndpointGroupV2	✓ 是	✓ 是	✗ 否
AWS::GroundStation::MissionProfile	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::Detector	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::Filter	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::IPSet	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::MalwareProtectionPlan	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::Master	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::Member	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::PublishingDestination	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::ThreatEntitySet	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::ThreatIntelSet	✓ 是	✓ 是	✓ 是
AWS::GuardDuty::TrustedEntitySet	✓ 是	✓ 是	✓ 是
AWS::HealthImaging::Datastore	✓ 是	✓ 是	✓ 是
AWS::HealthLake::FHIRDatastore	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::IAM::Group	✓ 是	✓ 是	✓ 是
AWS::IAM::GroupPolicy	✓ 是	✓ 是	✗ 否
AWS::IAM::InstanceProfile	✓ 是	✓ 是	✓ 是
AWS::IAM::ManagedPolicy	✓ 是	✓ 是	✓ 是
AWS::IAM::OIDCProvider	✓ 是	✓ 是	✓ 是
AWS::IAM::Role	✓ 是	✓ 是	✓ 是
AWS::IAM::RolePolicy	✓ 是	✓ 是	✗ 否
AWS::IAM::SAMLProvider	✓ 是	✓ 是	✓ 是
AWS::IAM::ServerCertificate	✓ 是	✓ 是	✓ 是
AWS::IAM::ServiceLinkedRole	✓ 是	✓ 是	✗ 否
AWS::IAM::User	✓ 是	✓ 是	✓ 是
AWS::IAM::UserPolicy	✓ 是	✓ 是	✗ 否
AWS::IAM::VirtualMFADevice	✓ 是	✓ 是	✓ 是
AWS::IVS::Channel	✓ 是	✓ 是	✓ 是
AWS::IVS::EncoderConfiguration	✓ 是	✓ 是	✓ 是
AWS::IVS::IngestConfiguration	✓ 是	✓ 是	✓ 是
AWS::IVS::PlaybackKeyPair	✓ 是	✓ 是	✓ 是
AWS::IVS::PlaybackRestrictionPolicy	✓ 是	✓ 是	✓ 是
AWS::IVS::PublicKey	✓ 是	✓ 是	✓ 是
AWS::IVS::RecordingConfiguration	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::IVS::Stage	✓ 是	✓ 是	✓ 是
AWS::IVS::StorageConfiguration	✓ 是	✓ 是	✓ 是
AWS::IVS::StreamKey	✓ 是	✓ 是	✓ 是
AWS::IVSChat::LoggingConfiguration	✓ 是	✓ 是	✓ 是
AWS::IVSChat::Room	✓ 是	✓ 是	✓ 是
AWS::IdentityStore::Group	✓ 是	✓ 是	✓ 是
AWS::IdentityStore::GroupMembership	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::Component	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::ContainerRecipe	✓ 是	✗ 否	✓ 是
AWS::ImageBuilder::DistributionConfiguration	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::Image	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::ImagePipeline	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::ImageRecipe	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::InfrastructureConfiguration	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::LifecyclePolicy	✓ 是	✓ 是	✓ 是
AWS::ImageBuilder::Workflow	✓ 是	✓ 是	✓ 是
AWS::Inspector::AssessmentTarget	✓ 是	✓ 是	✓ 是
AWS::Inspector::AssessmentTemplate	✓ 是	✓ 是	✓ 是
AWS::Inspector::ResourceGroup	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::InspectorV2::CisScanConfiguration	✓ 是	✓ 是	✓ 是
AWS::InspectorV2::CodeSecurityIntegration	✓ 是	✓ 是	✓ 是
AWS::InspectorV2::CodeSecurityScanConfiguration	✓ 是	✓ 是	✓ 是
AWS::InspectorV2::Filter	✓ 是	✓ 是	✓ 是
AWS::Interconnect::Connection	✓ 是	✓ 是	✗ 否
AWS::InternetMonitor::Monitor	✓ 是	✓ 是	✓ 是
AWS::Invoicing::InvoiceUnit	✓ 是	✓ 是	✓ 是
AWS::IoT::AccountAuditConfiguration	✓ 是	✓ 是	✓ 是
AWS::IoT::Authorizer	✓ 是	✓ 是	✓ 是
AWS::IoT::BillingGroup	✓ 是	✓ 是	✓ 是
AWS::IoT::CACertificate	✓ 是	✓ 是	✓ 是
AWS::IoT::Certificate	✓ 是	✓ 是	✓ 是
AWS::IoT::CertificateProvider	✓ 是	✓ 是	✓ 是
AWS::IoT::Command	✓ 是	✓ 是	✓ 是
AWS::IoT::CustomMetric	✓ 是	✓ 是	✓ 是
AWS::IoT::Dimension	✓ 是	✓ 是	✓ 是
AWS::IoT::DomainConfiguration	✓ 是	✓ 是	✓ 是
AWS::IoT::EncryptionConfiguration	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::IoT::FleetMetric	✓ 是	✓ 是	✓ 是
AWS::IoT::JobTemplate	✓ 是	✓ 是	✓ 是
AWS::IoT::Logging	✓ 是	✓ 是	✓ 是
AWS::IoT::MitigationAction	✓ 是	✓ 是	✓ 是
AWS::IoT::Policy	✓ 是	✗ 否	✓ 是
AWS::IoT::ProvisioningTemplate	✓ 是	✓ 是	✓ 是
AWS::IoT::ResourceSpecificLogging	✓ 是	✓ 是	✓ 是
AWS::IoT::RoleAlias	✓ 是	✓ 是	✓ 是
AWS::IoT::ScheduledAudit	✓ 是	✓ 是	✓ 是
AWS::IoT::SecurityProfile	✓ 是	✗ 否	✓ 是
AWS::IoT::SoftwarePackage	✓ 是	✓ 是	✓ 是
AWS::IoT::SoftwarePackageVersion	✓ 是	✓ 是	✓ 是
AWS::IoT::Thing	✓ 是	✓ 是	✓ 是
AWS::IoT::ThingGroup	✓ 是	✓ 是	✓ 是
AWS::IoT::ThingType	✓ 是	✓ 是	✓ 是
AWS::IoT::TopicRule	✓ 是	✓ 是	✓ 是
AWS::IoT::TopicRuleDestination	✓ 是	✓ 是	✓ 是
AWS::IoTAnalytics::Channel	✓ 是	✓ 是	✓ 是
AWS::IoTAnalytics::Dataset	✓ 是	✓ 是	✓ 是
AWS::IoTAnalytics::Datastore	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::IoTAnalytics::Pipeline	✓ 是	✓ 是	✓ 是
AWS::IoTCoreDeviceAdvisor::SuiteDefinition	✓ 是	✓ 是	✓ 是
AWS::IoTEvents::AlarmModel	✓ 是	✓ 是	✓ 是
AWS::IoTEvents::DetectorModel	✓ 是	✓ 是	✓ 是
AWS::IoTEvents::Input	✓ 是	✓ 是	✓ 是
AWS::IoTFleetWise::Campaign	✓ 是	✓ 是	✓ 是
AWS::IoTFleetWise::DecoderManifest	✓ 是	✓ 是	✓ 是
AWS::IoTFleetWise::Fleet	✓ 是	✓ 是	✓ 是
AWS::IoTFleetWise::ModelManifest	✓ 是	✗ 否	✓ 是
AWS::IoTFleetWise::SignalCatalog	✓ 是	✗ 否	✓ 是
AWS::IoTFleetWise::StateTemplate	✓ 是	✓ 是	✓ 是
AWS::IoTFleetWise::Vehicle	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::AccessPolicy	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::Asset	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::AssetModel	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::ComputationModel	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::Dashboard	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::Dataset	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::Gateway	✓ 是	✓ 是	✓ 是
AWS::IoTSiteWise::Portal	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::IoTSiteWise::Project	✓ 是	✓ 是	✓ 是
AWS::IoTTwinMaker::ComponentType	✓ 是	✓ 是	✓ 是
AWS::IoTTwinMaker::Entity	✓ 是	✓ 是	✓ 是
AWS::IoTTwinMaker::Scene	✓ 是	✓ 是	✓ 是
AWS::IoTTwinMaker::SyncJob	✓ 是	✓ 是	✓ 是
AWS::IoTTwinMaker::Workspace	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::Destination	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::DeviceProfile	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::FuotaTask	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::MulticastGroup	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::NetworkAnalyzerConfiguration	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::PartnerAccount	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::ServiceProfile	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::TaskDefinition	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::WirelessDevice	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::WirelessDeviceImportTask	✓ 是	✓ 是	✓ 是
AWS::IoTWireless::WirelessGateway	✓ 是	✓ 是	✓ 是
AWS::KMS::Alias	✓ 是	✗ 否	✓ 是
AWS::KMS::Key	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::KMS::ReplicaKey	✓ 是	✗ 否	✓ 是
AWS::KafkaConnect::Connector	✓ 是	✗ 否	✓ 是
AWS::KafkaConnect::CustomPlugin	✓ 是	✓ 是	✓ 是
AWS::KafkaConnect::WorkerConfiguration	✓ 是	✓ 是	✓ 是
AWS::Kendra::DataSource	✓ 是	✓ 是	✓ 是
AWS::Kendra::Faq	✓ 是	✓ 是	✓ 是
AWS::Kendra::Index	✓ 是	✓ 是	✓ 是
AWS::KendraRanking::ExecutionPlan	✓ 是	✓ 是	✓ 是
AWS::Kinesis::ResourcePolicy	✓ 是	✓ 是	✗ 否
AWS::Kinesis::Stream	✓ 是	✓ 是	✓ 是
AWS::Kinesis::StreamConsumer	✓ 是	✓ 是	✓ 是
AWS::KinesisAnalyticsV2::Application	✓ 是	✓ 是	✓ 是
AWS::KinesisFirehose::DeliveryStream	✓ 是	✗ 否	✓ 是
AWS::KinesisVideo::SignalingChannel	✓ 是	✓ 是	✓ 是
AWS::KinesisVideo::Stream	✓ 是	✓ 是	✓ 是
AWS::LakeFormation::DataCellsFilter	✓ 是	✓ 是	✓ 是
AWS::LakeFormation::PrincipalPermissions	✓ 是	✓ 是	✗ 否
AWS::LakeFormation::Tag	✓ 是	✓ 是	✓ 是
AWS::LakeFormation::TagAssociation	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::Lambda::Alias	✓ 是	✓ 是	✓ 是
AWS::Lambda::CapacityProvider	✓ 是	✓ 是	✓ 是
AWS::Lambda::CodeSigningConfig	✓ 是	✓ 是	✓ 是
AWS::Lambda::EventInvokeConfig	✓ 是	✓ 是	✓ 是
AWS::Lambda::EventSourceMapping	✓ 是	✗ 否	✓ 是
AWS::Lambda::Function	✓ 是	✓ 是	✓ 是
AWS::Lambda::LayerVersion	✓ 是	✓ 是	✓ 是
AWS::Lambda::LayerVersionPermission	✓ 是	✓ 是	✓ 是
AWS::Lambda::Permission	✓ 是	✓ 是	✓ 是
AWS::Lambda::Url	✓ 是	✓ 是	✓ 是
AWS::Lambda::Version	✓ 是	✓ 是	✓ 是
AWS::LaunchWizard::Deployment	✓ 是	✓ 是	✓ 是
AWS::Lex::Bot	✓ 是	✓ 是	✓ 是
AWS::Lex::BotAlias	✓ 是	✓ 是	✓ 是
AWS::Lex::BotVersion	✓ 是	✓ 是	✓ 是
AWS::Lex::ResourcePolicy	✓ 是	✓ 是	✓ 是
AWS::LicenseManager::Grant	✓ 是	✓ 是	✓ 是
AWS::LicenseManager::License	✓ 是	✓ 是	✓ 是
AWS::Lightsail::Alarm	✓ 是	✓ 是	✓ 是
AWS::Lightsail::Bucket	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Lightsail::Certificate	✓ 是	✓ 是	✓ 是
AWS::Lightsail::Container	✓ 是	✓ 是	✓ 是
AWS::Lightsail::Database	✓ 是	✓ 是	✓ 是
AWS::Lightsail::DatabaseSnapshot	✓ 是	✓ 是	✗ 否
AWS::Lightsail::Disk	✓ 是	✓ 是	✓ 是
AWS::Lightsail::DiskSnapshot	✓ 是	✓ 是	✓ 是
AWS::Lightsail::Distribution	✓ 是	✓ 是	✓ 是
AWS::Lightsail::Domain	✓ 是	✓ 是	✓ 是
AWS::Lightsail::Instance	✓ 是	✓ 是	✓ 是
AWS::Lightsail::InstanceSnapshot	✓ 是	✓ 是	✓ 是
AWS::Lightsail::LoadBalancer	✓ 是	✓ 是	✓ 是
AWS::Lightsail::LoadBalancerTlsCertificate	✓ 是	✓ 是	✓ 是
AWS::Lightsail::StaticIp	✓ 是	✓ 是	✓ 是
AWS::Location::APIKey	✓ 是	✓ 是	✓ 是
AWS::Location::GeofenceCollection	✓ 是	✓ 是	✓ 是
AWS::Location::Map	✓ 是	✓ 是	✓ 是
AWS::Location::PlaceIndex	✓ 是	✓ 是	✓ 是
AWS::Location::RouteCalculator	✓ 是	✓ 是	✓ 是
AWS::Location::Tracker	✓ 是	✓ 是	✓ 是
AWS::Location::TrackerConsumer	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Logs::AccountPolicy	✓ 是	✓ 是	✓ 是
AWS::Logs::Delivery	✓ 是	✓ 是	✓ 是
AWS::Logs::DeliveryDestination	✓ 是	✓ 是	✓ 是
AWS::Logs::DeliverySource	✓ 是	✓ 是	✓ 是
AWS::Logs::Destination	✓ 是	✓ 是	✓ 是
AWS::Logs::Integration	✓ 是	✓ 是	✓ 是
AWS::Logs::LogAnomalyDetector	✓ 是	✓ 是	✓ 是
AWS::Logs::LogGroup	✓ 是	✓ 是	✓ 是
AWS::Logs::LogStream	✓ 是	✓ 是	✓ 是
AWS::Logs::MetricFilter	✓ 是	✓ 是	✓ 是
AWS::Logs::QueryDefinition	✓ 是	✓ 是	✓ 是
AWS::Logs::ResourcePolicy	✓ 是	✓ 是	✓ 是
AWS::Logs::ScheduledQuery	✓ 是	✓ 是	✗ 否
AWS::Logs::SubscriptionFilter	✓ 是	✓ 是	✓ 是
AWS::Logs::Transformer	✓ 是	✓ 是	✓ 是
AWS::LookoutEquipment::InferenceScheduler	✓ 是	✓ 是	✓ 是
AWS::LookoutVision::Project	✓ 是	✓ 是	✓ 是
AWS::M2::Application	✓ 是	✓ 是	✓ 是
AWS::M2::Deployment	✓ 是	✓ 是	✓ 是
AWS::M2::Environment	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::MPA::ApprovalTeam	✓ 是	✓ 是	✓ 是
AWS::MPA::IdentitySource	✓ 是	✓ 是	✓ 是
AWS::MSK::BatchScramSecret	✓ 是	✓ 是	✓ 是
AWS::MSK::Cluster	✓ 是	✓ 是	✓ 是
AWS::MSK::ClusterPolicy	✓ 是	✓ 是	✓ 是
AWS::MSK::Configuration	✓ 是	✓ 是	✓ 是
AWS::MSK::Replicator	✓ 是	✓ 是	✓ 是
AWS::MSK::ServerlessCluster	✓ 是	✓ 是	✓ 是
AWS::MSK::Topic	✓ 是	✓ 是	✗ 否
AWS::MSK::VpcConnection	✓ 是	✓ 是	✓ 是
AWS::MWAA::Environment	✓ 是	✓ 是	✓ 是
AWS::MWAAServerless::Workflow	✓ 是	✓ 是	✗ 否
AWS::Macie::AllowList	✓ 是	✓ 是	✓ 是
AWS::Macie::CustomDataIdentifier	✓ 是	✓ 是	✓ 是
AWS::Macie::FindingsFilter	✓ 是	✓ 是	✓ 是
AWS::Macie::Session	✓ 是	✓ 是	✓ 是
AWS::ManagedBlockchain::Accessor	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::Bridge	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::BridgeOutput	✓ 是	✓ 是	✗ 否
AWS::MediaConnect::BridgeSource	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::MediaConnect::Flow	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::FlowEntitlement	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::FlowOutput	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::FlowSource	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::FlowVpcInterface	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::Gateway	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::RouterInput	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::RouterNetworkInterface	✓ 是	✓ 是	✓ 是
AWS::MediaConnect::RouterOutput	✓ 是	✓ 是	✓ 是
AWS::MediaLive::ChannelPlacementGroup	✓ 是	✓ 是	✓ 是
AWS::MediaLive::CloudWatchAlarmTemplate	✓ 是	✓ 是	✓ 是
AWS::MediaLive::CloudWatchAlarmTemplateGroup	✓ 是	✓ 是	✓ 是
AWS::MediaLive::Cluster	✓ 是	✓ 是	✓ 是
AWS::MediaLive::EventBridgeRuleTemplate	✓ 是	✓ 是	✓ 是
AWS::MediaLive::EventBridgeRuleTemplateGroup	✓ 是	✓ 是	✓ 是
AWS::MediaLive::Multiplex	✓ 是	✓ 是	✓ 是
AWS::MediaLive::Multiplexprogram	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::MediaLive::Network	✓ 是	✓ 是	✓ 是
AWS::MediaLive::SdiSource	✓ 是	✓ 是	✓ 是
AWS::MediaLive::SignalMap	✓ 是	✓ 是	✓ 是
AWS::MediaPackage::Asset	✓ 是	✓ 是	✓ 是
AWS::MediaPackage::Channel	✓ 是	✓ 是	✓ 是
AWS::MediaPackage::OriginEndpoint	✓ 是	✓ 是	✓ 是
AWS::MediaPackage::PackagingConfiguration	✓ 是	✓ 是	✓ 是
AWS::MediaPackage::PackagingGroup	✓ 是	✓ 是	✓ 是
AWS::MediaPackageV2::Channel	✓ 是	✓ 是	✓ 是
AWS::MediaPackageV2::ChannelGroup	✓ 是	✓ 是	✓ 是
AWS::MediaPackageV2::ChannelPolicy	✓ 是	✓ 是	✗ 否
AWS::MediaPackageV2::OriginEndpoint	✓ 是	✓ 是	✓ 是
AWS::MediaPackageV2::OriginEndpointPolicy	✓ 是	✓ 是	✗ 否
AWS::MediaTailor::Channel	✓ 是	✓ 是	✓ 是
AWS::MediaTailor::ChannelPolicy	✓ 是	✓ 是	✗ 否
AWS::MediaTailor::LiveSource	✓ 是	✓ 是	✓ 是
AWS::MediaTailor::PlaybackConfiguration	✓ 是	✓ 是	✓ 是
AWS::MediaTailor::SourceLocation	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::MediaTailor::VodSource	✓ 是	✓ 是	✓ 是
AWS::MemoryDB::ACL	✓ 是	✓ 是	✓ 是
AWS::MemoryDB::Cluster	✓ 是	✓ 是	✓ 是
AWS::MemoryDB::MultiRegionCluster	✓ 是	✓ 是	✓ 是
AWS::MemoryDB::ParameterGroup	✓ 是	✓ 是	✓ 是
AWS::MemoryDB::SubnetGroup	✓ 是	✓ 是	✓ 是
AWS::MemoryDB::User	✓ 是	✓ 是	✓ 是
AWS::Neptune::DBCluster	✓ 是	✓ 是	✓ 是
AWS::Neptune::DBClusterParameterGroup	✓ 是	✓ 是	✓ 是
AWS::Neptune::DBInstance	✓ 是	✓ 是	✓ 是
AWS::Neptune::DBParameterGroup	✓ 是	✓ 是	✓ 是
AWS::Neptune::DBSubnetGroup	✓ 是	✓ 是	✓ 是
AWS::Neptune::EventSubscription	✓ 是	✓ 是	✓ 是
AWS::NeptuneGraph::Graph	✓ 是	✓ 是	✓ 是
AWS::NeptuneGraph::GraphSnapshot	✓ 是	✓ 是	✗ 否
AWS::NeptuneGraph::PrivateGraphEndpoint	✓ 是	✓ 是	✓ 是
AWS::NetworkFirewall::Firewall	✓ 是	✓ 是	✓ 是
AWS::NetworkFirewall::FirewallPolicy	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::NetworkFirewall::LoggingConfiguration	✓ 是	✓ 是	✗ 否
AWS::NetworkFirewall::RuleGroup	✓ 是	✓ 是	✓ 是
AWS::NetworkFirewall::TLSInspectionConfiguration	✓ 是	✓ 是	✓ 是
AWS::NetworkFirewall::VpcEndpointAssociation	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::ConnectAttachment	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::ConnectPeer	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::CoreNetwork	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::CoreNetworkPrefixListAssociation	✓ 是	✓ 是	✗ 否
AWS::NetworkManager::CustomerGatewayAssociation	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::Device	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::DirectConnectGatewayAttachment	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::GlobalNetwork	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::Link	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::LinkAssociation	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::Site	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::NetworkManager::SiteToSiteVpnAttachment	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::TransitGatewayPeering	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::TransitGatewayRegistration	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::TransitGatewayRouteTableAttachment	✓ 是	✓ 是	✓ 是
AWS::NetworkManager::VpcAttachment	✓ 是	✓ 是	✓ 是
AWS::Notifications::ChannelAssociation	✓ 是	✓ 是	✓ 是
AWS::Notifications::EventRule	✓ 是	✓ 是	✓ 是
AWS::Notifications::ManagedNotificationAccountContactAssociation	✓ 是	✓ 是	✓ 是
AWS::Notifications::ManagedNotificationAdditionalChannelAssociation	✓ 是	✓ 是	✓ 是
AWS::Notifications::NotificationConfiguration	✓ 是	✓ 是	✓ 是
AWS::Notifications::NotificationHub	✓ 是	✓ 是	✓ 是
AWS::Notifications::OrganizationalUnitAssociation	✓ 是	✓ 是	✓ 是
AWS::NotificationsContacts::EmailContact	✓ 是	✓ 是	✓ 是
AWS::NovaAct::WorkflowDefinition	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::ODB::CloudAutonomousVmCluster	✓ 是	✓ 是	✓ 是
AWS::ODB::CloudExadataInfrastructure	✓ 是	✓ 是	✓ 是
AWS::ODB::CloudVmCluster	✓ 是	✓ 是	✓ 是
AWS::ODB::OdbNetwork	✓ 是	✓ 是	✓ 是
AWS::ODB::OdbPeeringConnection	✓ 是	✓ 是	✓ 是
AWS::OSIS::Pipeline	✓ 是	✓ 是	✓ 是
AWS::Oam::Link	✓ 是	✓ 是	✓ 是
AWS::Oam::Sink	✓ 是	✓ 是	✓ 是
AWS::ObservabilityAdmin::OrganizationCentralizationRule	✓ 是	✓ 是	✓ 是
AWS::ObservabilityAdmin::OrganizationTelemetryRule	✓ 是	✓ 是	✓ 是
AWS::ObservabilityAdmin::S3TableIntegration	✓ 是	✓ 是	✓ 是
AWS::ObservabilityAdmin::TelemetryEnrichment	✓ 是	✓ 是	✗ 否
AWS::ObservabilityAdmin::TelemetryPipelines	✓ 是	✓ 是	✓ 是
AWS::ObservabilityAdmin::TelemetryRule	✓ 是	✓ 是	✓ 是
AWS::Omics::AnnotationStore	✓ 是	✓ 是	✓ 是
AWS::Omics::Configuration	✓ 是	✓ 是	✗ 否
AWS::Omics::ReferenceStore	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Omics::RunGroup	✓ 是	✓ 是	✓ 是
AWS::Omics::SequenceStore	✓ 是	✓ 是	✓ 是
AWS::Omics::VariantStore	✓ 是	✓ 是	✓ 是
AWS::Omics::Workflow	✓ 是	✓ 是	✓ 是
AWS::Omics::WorkflowVersion	✓ 是	✓ 是	✓ 是
AWS::OpenSearchServerless::AccessPolicy	✓ 是	✓ 是	✓ 是
AWS::OpenSearchServerless::Collection	✓ 是	✓ 是	✓ 是
AWS::OpenSearchServerless::CollectionGroup	✓ 是	✓ 是	✗ 否
AWS::OpenSearchServerless::CollectionIndex	✓ 是	✓ 是	✗ 否
AWS::OpenSearchServerless::Index	✓ 是	✓ 是	✓ 是
AWS::OpenSearchServerless::LifecyclePolicy	✓ 是	✓ 是	✓ 是
AWS::OpenSearchServerless::SecurityConfig	✓ 是	✓ 是	✓ 是
AWS::OpenSearchServerless::SecurityPolicy	✓ 是	✓ 是	✓ 是
AWS::OpenSearchServerless::VpcEndpoint	✓ 是	✓ 是	✓ 是
AWS::OpenSearchService::Application	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::OpenSearchService::Domain	✓ 是	✓ 是	✗ 否
AWS::Organizations::Account	✓ 是	✓ 是	✓ 是
AWS::Organizations::Organization	✓ 是	✓ 是	✓ 是
AWS::Organizations::OrganizationalUnit	✓ 是	✓ 是	✓ 是
AWS::Organizations::Policy	✓ 是	✓ 是	✓ 是
AWS::Organizations::ResourcePolicy	✓ 是	✓ 是	✓ 是
AWS::PCAConnectorAD::Connector	✓ 是	✓ 是	✓ 是
AWS::PCAConnectorAD::DirectoryRegistration	✓ 是	✓ 是	✓ 是
AWS::PCAConnectorAD::ServicePrincipalName	✓ 是	✓ 是	✓ 是
AWS::PCAConnectorAD::Template	✓ 是	✓ 是	✓ 是
AWS::PCAConnectorAD::TemplateGroupAccessControlEntry	✓ 是	✓ 是	✓ 是
AWS::PCAConnectorSCEP::Challenge	✓ 是	✓ 是	✓ 是
AWS::PCAConnectorSCEP::Connector	✓ 是	✓ 是	✓ 是
AWS::PCS::Cluster	✓ 是	✓ 是	✓ 是
AWS::PCS::ComputeNodeGroup	✓ 是	✓ 是	✓ 是
AWS::PCS::Queue	✓ 是	✓ 是	✓ 是
AWS::Panorama::ApplicationInstance	✓ 是	✓ 是	✓ 是
AWS::Panorama::Package	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Panorama::PackageVersion	✓ 是	✓ 是	✗ 否
AWS::PaymentCryptography::Alias	✓ 是	✓ 是	✓ 是
AWS::PaymentCryptography::Key	✓ 是	✓ 是	✓ 是
AWS::Personalize::Dataset	✓ 是	✓ 是	✓ 是
AWS::Personalize::DatasetGroup	✓ 是	✓ 是	✓ 是
AWS::Personalize::Schema	✓ 是	✓ 是	✓ 是
AWS::Personalize::Solution	✓ 是	✓ 是	✓ 是
AWS::Pinpoint::InAppTemplate	✓ 是	✓ 是	✓ 是
AWS::Pipes::Pipe	✓ 是	✓ 是	✓ 是
AWS::Proton::EnvironmentAccountConnection	✓ 是	✓ 是	✓ 是
AWS::Proton::EnvironmentTemplate	✓ 是	✓ 是	✓ 是
AWS::Proton::ServiceTemplate	✓ 是	✓ 是	✓ 是
AWS::QBusiness::Application	✓ 是	✓ 是	✓ 是
AWS::QBusiness::DataAccessor	✓ 是	✓ 是	✓ 是
AWS::QBusiness::DataSource	✓ 是	✓ 是	✓ 是
AWS::QBusiness::Index	✓ 是	✓ 是	✓ 是
AWS::QBusiness::Permission	✓ 是	✓ 是	✓ 是
AWS::QBusiness::Plugin	✓ 是	✓ 是	✓ 是
AWS::QBusiness::Retriever	✓ 是	✓ 是	✓ 是
AWS::QBusiness::WebExperience	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::QLDB::Stream	✓ 是	✓ 是	✓ 是
AWS::QuickSight::ActionConnector	✓ 是	✓ 是	✗ 否
AWS::QuickSight::Analysis	✓ 是	✓ 是	✓ 是
AWS::QuickSight::CustomPermissions	✓ 是	✓ 是	✓ 是
AWS::QuickSight::Dashboard	✓ 是	✓ 是	✓ 是
AWS::QuickSight::DataSet	✓ 是	✓ 是	✓ 是
AWS::QuickSight::DataSource	✓ 是	✓ 是	✓ 是
AWS::QuickSight::Folder	✓ 是	✓ 是	✓ 是
AWS::QuickSight::RefreshSchedule	✓ 是	✓ 是	✓ 是
AWS::QuickSight::Template	✓ 是	✓ 是	✓ 是
AWS::QuickSight::Theme	✓ 是	✓ 是	✓ 是
AWS::QuickSight::Topic	✓ 是	✓ 是	✓ 是
AWS::QuickSight::VPCConnection	✓ 是	✓ 是	✓ 是
AWS::RAM::Permission	✓ 是	✓ 是	✓ 是
AWS::RAM::ResourceShare	✓ 是	✓ 是	✓ 是
AWS::RDS::CustomDBEngineVersion	✓ 是	✓ 是	✓ 是
AWS::RDS::DBCluster	✓ 是	✓ 是	✓ 是
AWS::RDS::DBClusterParameterGroup	✓ 是	✓ 是	✓ 是
AWS::RDS::DBInstance	✓ 是	✓ 是	✓ 是
AWS::RDS::DBParameterGroup	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::RDS::DBProxy	✓ 是	✓ 是	✓ 是
AWS::RDS::DBProxyEndpoint	✓ 是	✓ 是	✓ 是
AWS::RDS::DBProxyTargetGroup	✓ 是	✓ 是	✓ 是
AWS::RDS::DBShardGroup	✓ 是	✓ 是	✓ 是
AWS::RDS::DBSubnetGroup	✓ 是	✓ 是	✓ 是
AWS::RDS::EventSubscription	✓ 是	✓ 是	✓ 是
AWS::RDS::GlobalCluster	✓ 是	✓ 是	✓ 是
AWS::RDS::Integration	✓ 是	✓ 是	✓ 是
AWS::RDS::OptionGroup	✓ 是	✓ 是	✓ 是
AWS::RTBFabric::InboundExternalLink	✓ 是	✓ 是	✓ 是
AWS::RTBFabric::Link	✓ 是	✓ 是	✓ 是
AWS::RTBFabric::LinkRoutingRule	✓ 是	✓ 是	✗ 否
AWS::RTBFabric::OutboundExternalLink	✓ 是	✓ 是	✓ 是
AWS::RTBFabric::RequesterGateway	✓ 是	✓ 是	✓ 是
AWS::RTBFabric::ResponderGateway	✓ 是	✓ 是	✓ 是
AWS::RUM::AppMonitor	✓ 是	✓ 是	✓ 是
AWS::Rbin::Rule	✓ 是	✓ 是	✓ 是
AWS::Redshift::Cluster	✓ 是	✓ 是	✓ 是
AWS::Redshift::ClusterParameterGroup	✓ 是	✓ 是	✓ 是
AWS::Redshift::ClusterSubnetGroup	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Redshift::EndpointAccess	✓ 是	✓ 是	✓ 是
AWS::Redshift::EndpointAuthorization	✓ 是	✓ 是	✓ 是
AWS::Redshift::EventSubscription	✓ 是	✓ 是	✓ 是
AWS::Redshift::Integration	✓ 是	✓ 是	✓ 是
AWS::Redshift::ScheduledAction	✓ 是	✓ 是	✓ 是
AWS::RedshiftServerless::Namespace	✓ 是	✓ 是	✓ 是
AWS::RedshiftServerless::Snapshot	✓ 是	✓ 是	✓ 是
AWS::RedshiftServerless::Workgroup	✓ 是	✓ 是	✓ 是
AWS::RefactorSpaces::Application	✓ 是	✓ 是	✓ 是
AWS::RefactorSpaces::Environment	✓ 是	✓ 是	✓ 是
AWS::RefactorSpaces::Route	✓ 是	✓ 是	✓ 是
AWS::RefactorSpaces::Service	✓ 是	✓ 是	✓ 是
AWS::Rekognition::Collection	✓ 是	✓ 是	✓ 是
AWS::Rekognition::Project	✓ 是	✓ 是	✓ 是
AWS::Rekognition::StreamProcessor	✓ 是	✓ 是	✓ 是
AWS::ResilienceHub::App	✓ 是	✓ 是	✓ 是
AWS::ResilienceHub::ResiliencyPolicy	✓ 是	✓ 是	✓ 是
AWS::ResilienceHubV2::Policy	✓ 是	✓ 是	✗ 否
AWS::ResilienceHubV2::Service	✓ 是	✓ 是	✗ 否
AWS::ResilienceHubV2::ServiceFunction	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::ResilienceHubV2::System	✓ 是	✓ 是	✗ 否
AWS::ResilienceHubV2::UserJourney	✓ 是	✓ 是	✗ 否
AWS::ResourceExplorer2::DefaultViewAssociation	✓ 是	✓ 是	✗ 否
AWS::ResourceExplorer2::Index	✓ 是	✓ 是	✓ 是
AWS::ResourceExplorer2::View	✓ 是	✓ 是	✓ 是
AWS::ResourceGroups::Group	✓ 是	✓ 是	✓ 是
AWS::ResourceGroups::TagSyncTask	✓ 是	✓ 是	✓ 是
AWS::RolesAnywhere::CRL	✓ 是	✓ 是	✓ 是
AWS::RolesAnywhere::Profile	✓ 是	✓ 是	✓ 是
AWS::RolesAnywhere::TrustAnchor	✓ 是	✓ 是	✓ 是
AWS::Route53::CidrCollection	✓ 是	✓ 是	✓ 是
AWS::Route53::DNSSEC	✓ 是	✓ 是	✓ 是
AWS::Route53::HealthCheck	✓ 是	✓ 是	✓ 是
AWS::Route53::HostedZone	✓ 是	✓ 是	✓ 是
AWS::Route53::KeySigningKey	✓ 是	✓ 是	✓ 是
AWS::Route53GlobalResolver::AccessSource	✓ 是	✓ 是	✗ 否
AWS::Route53GlobalResolver::AccessToken	✓ 是	✓ 是	✗ 否
AWS::Route53GlobalResolver::DnsView	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::Route53GlobalResolver::FirewallDomainList	✓ 是	✓ 是	✗ 否
AWS::Route53GlobalResolver::FirewallRule	✓ 是	✓ 是	✗ 否
AWS::Route53GlobalResolver::GlobalResolver	✓ 是	✓ 是	✗ 否
AWS::Route53GlobalResolver::HostedZoneAssociation	✓ 是	✓ 是	✗ 否
AWS::Route53Profiles::Profile	✓ 是	✓ 是	✓ 是
AWS::Route53Profiles::ProfileAssociation	✓ 是	✓ 是	✓ 是
AWS::Route53Profiles::ProfileResourceAssociation	✓ 是	✓ 是	✓ 是
AWS::Route53RecoveryControl::Cluster	✓ 是	✓ 是	✓ 是
AWS::Route53RecoveryControl::ControlPanel	✓ 是	✓ 是	✓ 是
AWS::Route53RecoveryControl::RoutingControl	✓ 是	✓ 是	✓ 是
AWS::Route53RecoveryControl::SafetyRule	✓ 是	✓ 是	✓ 是
AWS::Route53RecoveryReadiness::Cell	✓ 是	✓ 是	✓ 是
AWS::Route53RecoveryReadiness::ReadinessCheck	✓ 是	✓ 是	✓ 是
AWS::Route53RecoveryReadiness::RecoveryGroup	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Route53RecoveryReadiness::ResourceSet	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::FirewallDomainList	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::FirewallRuleGroup	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::FirewallRuleGroupAssociation	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::OutpostResolver	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::ResolverConfig	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::ResolverDNSSECConfig	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::ResolverEndpoint	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::ResolverQueryLoggingConfig	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::ResolverQueryLoggingConfigAssociation	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::ResolverRule	✓ 是	✓ 是	✓ 是
AWS::Route53Resolver::ResolverRuleAssociation	✓ 是	✓ 是	✓ 是
AWS::S3::AccessGrant	✓ 是	✓ 是	✓ 是
AWS::S3::AccessGrantsInstance	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::S3::AccessGrantsLocation	✓ 是	✓ 是	✓ 是
AWS::S3::AccessPoint	✓ 是	✓ 是	✓ 是
AWS::S3::Bucket	✓ 是	✓ 是	✓ 是
AWS::S3::BucketPolicy	✓ 是	✗ 否	✓ 是
AWS::S3::MultiRegionAccessPoint	✓ 是	✓ 是	✓ 是
AWS::S3::MultiRegionAccessPointPolicy	✓ 是	✗ 否	✓ 是
AWS::S3::StorageLens	✓ 是	✓ 是	✓ 是
AWS::S3::StorageLensGroup	✓ 是	✓ 是	✓ 是
AWS::S3Express::AccessPoint	✓ 是	✓ 是	✓ 是
AWS::S3Express::BucketPolicy	✓ 是	✓ 是	✓ 是
AWS::S3Express::DirectoryBucket	✓ 是	✓ 是	✓ 是
AWS::S3Files::AccessPoint	✓ 是	✓ 是	✗ 否
AWS::S3Files::FileSystem	✓ 是	✓ 是	✗ 否
AWS::S3Files::FileSystemPolicy	✓ 是	✓ 是	✗ 否
AWS::S3Files::MountTarget	✓ 是	✓ 是	✗ 否
AWS::S3ObjectLambda::AccessPoint	✓ 是	✓ 是	✓ 是
AWS::S3ObjectLambda::AccessPointPolicy	✓ 是	✓ 是	✗ 否
AWS::S3Outposts::AccessPoint	✓ 是	✓ 是	✓ 是
AWS::S3Outposts::Bucket	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::S3Outposts::BucketPolicy	✓ 是	✓ 是	✗ 否
AWS::S3Outposts::Endpoint	✓ 是	✓ 是	✓ 是
AWS::S3Tables::Namespace	✓ 是	✓ 是	✓ 是
AWS::S3Tables::Table	✓ 是	✓ 是	✓ 是
AWS::S3Tables::TableBucket	✓ 是	✓ 是	✓ 是
AWS::S3Tables::TableBucketPolicy	✓ 是	✓ 是	✓ 是
AWS::S3Tables::TablePolicy	✓ 是	✓ 是	✓ 是
AWS::S3Vectors::Index	✓ 是	✓ 是	✓ 是
AWS::S3Vectors::VectorBucket	✓ 是	✓ 是	✓ 是
AWS::S3Vectors::VectorBucketPolicy	✓ 是	✓ 是	✓ 是
AWS::SES::ConfigurationSet	✓ 是	✓ 是	✓ 是
AWS::SES::ConfigurationSetEventDestination	✗ 否	✓ 是	✓ 是
AWS::SES::ContactList	✓ 是	✓ 是	✓ 是
AWS::SES::CustomVerificationEmailTemplate	✓ 是	✓ 是	✗ 否
AWS::SES::DedicatedIpPool	✓ 是	✓ 是	✓ 是
AWS::SES::EmailIdentity	✓ 是	✓ 是	✓ 是
AWS::SES::MailManagerAddonInstance	✓ 是	✓ 是	✓ 是
AWS::SES::MailManagerAddonSubscription	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::SES::MailManagerAddressList	✓ 是	✓ 是	✓ 是
AWS::SES::MailManagerArchive	✓ 是	✓ 是	✓ 是
AWS::SES::MailManagerIngressPoint	✓ 是	✓ 是	✓ 是
AWS::SES::MailManagerRelay	✓ 是	✓ 是	✓ 是
AWS::SES::MailManagerRuleSet	✓ 是	✓ 是	✓ 是
AWS::SES::MailManagerTrafficPolicy	✓ 是	✓ 是	✓ 是
AWS::SES::MultiRegionEndpoint	✓ 是	✓ 是	✓ 是
AWS::SES::Template	✓ 是	✓ 是	✓ 是
AWS::SES::Tenant	✓ 是	✓ 是	✗ 否
AWS::SES::VdmAttributes	✓ 是	✓ 是	✗ 否
AWS::SMSVOICE::ConfigurationSet	✓ 是	✓ 是	✓ 是
AWS::SMSVOICE::OptOutList	✓ 是	✓ 是	✓ 是
AWS::SMSVOICE::PhoneNumber	✓ 是	✓ 是	✓ 是
AWS::SMSVOICE::Pool	✓ 是	✓ 是	✓ 是
AWS::SMSVOICE::ProtectConfiguration	✓ 是	✓ 是	✓ 是
AWS::SMSVOICE::ResourcePolicy	✓ 是	✓ 是	✓ 是
AWS::SMSVOICE::SenderId	✓ 是	✓ 是	✓ 是
AWS::SNS::Subscription	✓ 是	✗ 否	✓ 是
AWS::SNS::Topic	✓ 是	✓ 是	✓ 是
AWS::SNS::TopicInlinePolicy	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::SQS::Queue	✓ 是	✓ 是	✓ 是
AWS::SQS::QueueInlinePolicy	✓ 是	✓ 是	✗ 否
AWS::SSM::Association	✓ 是	✓ 是	✓ 是
AWS::SSM::Document	✓ 是	✗ 否	✓ 是
AWS::SSM::MaintenanceWindow	✓ 是	✓ 是	✗ 否
AWS::SSM::MaintenanceWindowTarget	✓ 是	✓ 是	✗ 否
AWS::SSM::MaintenanceWindowTask	✓ 是	✓ 是	✗ 否
AWS::SSM::Parameter	✓ 是	✓ 是	✓ 是
AWS::SSM::PatchBaseline	✓ 是	✓ 是	✓ 是
AWS::SSM::ResourceDataSync	✓ 是	✓ 是	✓ 是
AWS::SSM::ResourcePolicy	✓ 是	✓ 是	✓ 是
AWS::SSMContacts::Contact	✓ 是	✓ 是	✓ 是
AWS::SSMContacts::ContactChannel	✓ 是	✓ 是	✓ 是
AWS::SSMContacts::Plan	✓ 是	✓ 是	✓ 是
AWS::SSMContacts::Rotation	✓ 是	✓ 是	✓ 是
AWS::SSMGuiConnect::Preferences	✓ 是	✓ 是	✓ 是
AWS::SSMIncidents::ReplicationSet	✓ 是	✓ 是	✓ 是
AWS::SSMIncidents::ResponsePlan	✓ 是	✓ 是	✓ 是
AWS::SSMQuickSetup::ConfigurationManager	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::SSMQuickSetup::LifecycleAutomation	✓ 是	✓ 是	✓ 是
AWS::SSO::Application	✓ 是	✓ 是	✓ 是
AWS::SSO::ApplicationAssignment	✓ 是	✓ 是	✓ 是
AWS::SSO::Assignment	✓ 是	✓ 是	✓ 是
AWS::SSO::Instance	✓ 是	✓ 是	✓ 是
AWS::SSO::InstanceAccessControlAttributeConfiguration	✓ 是	✓ 是	✓ 是
AWS::SageMaker::App	✓ 是	✓ 是	✓ 是
AWS::SageMaker::AppImageConfig	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Cluster	✓ 是	✓ 是	✓ 是
AWS::SageMaker::DataQualityJobDefinition	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Device	✓ 是	✓ 是	✗ 否
AWS::SageMaker::DeviceFleet	✓ 是	✓ 是	✗ 否
AWS::SageMaker::Domain	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Endpoint	✗ 否	✓ 是	✗ 否
AWS::SageMaker::FeatureGroup	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Image	✓ 是	✓ 是	✓ 是
AWS::SageMaker::ImageVersion	✓ 是	✓ 是	✓ 是
AWS::SageMaker::InferenceComponent	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::SageMaker::InferenceExperiment	✓ 是	✓ 是	✓ 是
AWS::SageMaker::MlflowApp	✓ 是	✓ 是	✗ 否
AWS::SageMaker::MlflowTrackingServer	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Model	✓ 是	✓ 是	✗ 否
AWS::SageMaker::ModelBiasJobDefinition	✓ 是	✓ 是	✓ 是
AWS::SageMaker::ModelCard	✓ 是	✓ 是	✓ 是
AWS::SageMaker::ModelExplainabilityJobDefinition	✓ 是	✓ 是	✓ 是
AWS::SageMaker::ModelPackage	✓ 是	✓ 是	✓ 是
AWS::SageMaker::ModelPackageGroup	✓ 是	✓ 是	✓ 是
AWS::SageMaker::ModelQualityJobDefinition	✓ 是	✓ 是	✓ 是
AWS::SageMaker::MonitoringSchedule	✓ 是	✗ 否	✓ 是
AWS::SageMaker::PartnerApp	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Pipeline	✓ 是	✓ 是	✓ 是
AWS::SageMaker::ProcessingJob	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Project	✓ 是	✓ 是	✓ 是
AWS::SageMaker::Space	✓ 是	✓ 是	✓ 是
AWS::SageMaker::StudioLifecycleConfig	✓ 是	✓ 是	✓ 是
AWS::SageMaker::UserProfile	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Scheduler::Schedule	✓ 是	✓ 是	✓ 是
AWS::Scheduler::ScheduleGroup	✓ 是	✓ 是	✓ 是
AWS::SecretsManager::ResourcePolicy	✓ 是	✗ 否	✓ 是
AWS::SecretsManager::RotationSchedule	✓ 是	✓ 是	✓ 是
AWS::SecretsManager::Secret	✓ 是	✓ 是	✓ 是
AWS::SecretsManager::SecretTargetAttachment	✓ 是	✗ 否	✓ 是
AWS::SecurityAgent::AgentSpace	✓ 是	✓ 是	✗ 否
AWS::SecurityAgent::Application	✓ 是	✓ 是	✗ 否
AWS::SecurityAgent::Pentest	✓ 是	✓ 是	✗ 否
AWS::SecurityAgent::TargetDomain	✓ 是	✓ 是	✗ 否
AWS::SecurityHub::AggregatorV2	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::AutomationRule	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::AutomationRuleV2	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::ConfigurationPolicy	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::ConnectorV2	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::DelegatedAdmin	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::FindingAggregator	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::Hub	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::SecurityHub::HubV2	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::Insight	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::OrganizationConfiguration	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::PolicyAssociation	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::ProductSubscription	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::SecurityControl	✓ 是	✓ 是	✓ 是
AWS::SecurityHub::Standard	✓ 是	✓ 是	✓ 是
AWS::SecurityLake::AwsLogSource	✓ 是	✓ 是	✓ 是
AWS::SecurityLake::DataLake	✓ 是	✓ 是	✓ 是
AWS::SecurityLake::Subscriber	✓ 是	✓ 是	✓ 是
AWS::SecurityLake::SubscriberNotification	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::CloudFormationProvisionedProduct	✓ 是	✗ 否	✗ 否
AWS::ServiceCatalog::LaunchNotificationConstraint	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::LaunchRoleConstraint	✓ 是	✓ 是	✗ 否
AWS::ServiceCatalog::LaunchTemplateConstraint	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::Portfolio	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::ServiceCatalog::PortfolioPrincipalAssociation	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::PortfolioProductAssociation	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::PortfolioShare	✓ 是	✗ 否	✓ 是
AWS::ServiceCatalog::ResourceUpdateConstraint	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::ServiceAction	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::ServiceActionAssociation	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::StackSetConstraint	✓ 是	✓ 是	✗ 否
AWS::ServiceCatalog::TagOption	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalog::TagOptionAssociation	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalogAppRegistry::Application	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalogAppRegistry::AttributeGroup	✓ 是	✓ 是	✓ 是
AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation	✓ 是	✗ 否	✓ 是
AWS::ServiceCatalogAppRegistry::ResourceAssociation	✓ 是	✗ 否	✓ 是
AWS::Shield::DRTAccess	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::Shield::ProactiveEngagement	✓ 是	✓ 是	✓ 是
AWS::Shield::Protection	✓ 是	✓ 是	✓ 是
AWS::Shield::ProtectionGroup	✓ 是	✓ 是	✓ 是
AWS::Signer::ProfilePermission	✓ 是	✓ 是	✓ 是
AWS::Signer::SigningProfile	✓ 是	✓ 是	✓ 是
AWS::SimSpaceWeaver::Simulation	✓ 是	✓ 是	✓ 是
AWS::StepFunctions::Activity	✓ 是	✓ 是	✓ 是
AWS::StepFunctions::StateMachine	✓ 是	✓ 是	✓ 是
AWS::StepFunctions::StateMachineAlias	✓ 是	✓ 是	✓ 是
AWS::StepFunctions::StateMachineVersion	✓ 是	✓ 是	✓ 是
AWS::SupportApp::AccountAlias	✓ 是	✓ 是	✓ 是
AWS::SupportApp::SlackChannelConfiguration	✓ 是	✓ 是	✓ 是
AWS::SupportApp::SlackWorkspaceConfiguration	✓ 是	✓ 是	✓ 是
AWS::Synthetics::Canary	✓ 是	✓ 是	✓ 是
AWS::Synthetics::Group	✓ 是	✓ 是	✓ 是
AWS::SystemsManagerSAP::Application	✓ 是	✓ 是	✓ 是
AWS::Timestream::Database	✓ 是	✓ 是	✓ 是
AWS::Timestream::InfluxDBCluster	✓ 是	✓ 是	✗ 否

资源	导入	偏差检测	IaC 生成器
AWS::Timestream::InfluxDBInstance	✓ 是	✓ 是	✓ 是
AWS::Timestream::ScheduledQuery	✓ 是	✓ 是	✓ 是
AWS::Timestream::Table	✓ 是	✓ 是	✓ 是
AWS::Transfer::Agreement	✓ 是	✓ 是	✓ 是
AWS::Transfer::Certificate	✓ 是	✓ 是	✓ 是
AWS::Transfer::Connector	✓ 是	✓ 是	✓ 是
AWS::Transfer::Profile	✓ 是	✓ 是	✓ 是
AWS::Transfer::Server	✓ 是	✓ 是	✓ 是
AWS::Transfer::User	✓ 是	✓ 是	✓ 是
AWS::Transfer::WebApp	✓ 是	✓ 是	✓ 是
AWS::Transfer::Workflow	✓ 是	✓ 是	✓ 是
AWS::UXC::AccountCustomization	✓ 是	✓ 是	✗ 否
AWS::VerifiedPermissions::IdentitySource	✓ 是	✓ 是	✓ 是
AWS::VerifiedPermissions::Policy	✓ 是	✓ 是	✓ 是
AWS::VerifiedPermissions::PolicyStore	✓ 是	✓ 是	✓ 是
AWS::VerifiedPermissions::PolicyStoreAlias	✓ 是	✓ 是	✗ 否
AWS::VerifiedPermissions::PolicyTemplate	✓ 是	✓ 是	✓ 是
AWS::VoiceID::Domain	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::VpcLattice::AccessLogSubscription	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::AuthPolicy	✓ 是	✓ 是	✗ 否
AWS::VpcLattice::DomainVerification	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::Listener	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::ResourceConfiguration	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::ResourceGateway	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::ResourcePolicy	✓ 是	✓ 是	✗ 否
AWS::VpcLattice::Rule	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::Service	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::ServiceNetwork	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::ServiceNetworkResourceAssociation	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::ServiceNetworkServiceAssociation	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::ServiceNetworkVpcAssociation	✓ 是	✓ 是	✓ 是
AWS::VpcLattice::TargetGroup	✓ 是	✓ 是	✓ 是
AWS::WAFv2::IPSet	✓ 是	✓ 是	✓ 是
AWS::WAFv2::LoggingConfiguration	✓ 是	✓ 是	✓ 是
AWS::WAFv2::RegexPatternSet	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::WAFv2::RuleGroup	✓ 是	✓ 是	✓ 是
AWS::WAFv2::WebACL	✓ 是	✓ 是	✓ 是
AWS::WAFv2::WebACLAssociation	✓ 是	✓ 是	✗ 否
AWS::Wisdom::AIAgent	✓ 是	✓ 是	✓ 是
AWS::Wisdom::AIAgentVersion	✓ 是	✓ 是	✓ 是
AWS::Wisdom::AIGuardrail	✓ 是	✓ 是	✓ 是
AWS::Wisdom::AIGuardrailVersion	✓ 是	✓ 是	✓ 是
AWS::Wisdom::AIPrompt	✓ 是	✓ 是	✓ 是
AWS::Wisdom::AIPromptVersion	✓ 是	✓ 是	✓ 是
AWS::Wisdom::Assistant	✓ 是	✓ 是	✓ 是
AWS::Wisdom::AssistantAssociation	✓ 是	✓ 是	✓ 是
AWS::Wisdom::KnowledgeBase	✓ 是	✓ 是	✓ 是
AWS::Wisdom::MessageTemplate	✓ 是	✓ 是	✓ 是
AWS::Wisdom::MessageTemplateVersion	✓ 是	✓ 是	✓ 是
AWS::Wisdom::QuickResponse	✓ 是	✓ 是	✓ 是
AWS::WorkSpaces::ConnectionAlias	✓ 是	✓ 是	✗ 否
AWS::WorkSpaces::Workspace	✓ 是	✓ 是	✗ 否
AWS::WorkSpaces::WorkspacesPool	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesThinClient::Environment	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::BrowserSettings	✓ 是	✓ 是	✓ 是

资源	导入	偏差检测	IaC 生成器
AWS::WorkSpacesWeb::DataProtectionSettings	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::IdentityProvider	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::IpAccessSettings	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::NetworkSettings	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::Portal	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::SessionLogger	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::TrustStore	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::UserAccessLoggingSettings	✓ 是	✓ 是	✓ 是
AWS::WorkSpacesWeb::UserSettings	✓ 是	✓ 是	✓ 是
AWS::WorkspacesInstances::Volume	✓ 是	✓ 是	✓ 是
AWS::WorkspacesInstances::VolumeAssociation	✓ 是	✓ 是	✓ 是
AWS::WorkspacesInstances::WorkspaceInstance	✓ 是	✓ 是	✓ 是
AWS::XRay::Group	✓ 是	✓ 是	✓ 是
AWS::XRay::ResourcePolicy	✓ 是	✓ 是	✓ 是
AWS::XRay::SamplingRule	✓ 是	✓ 是	✓ 是
AWS::XRay::TransactionSearchConfig	✓ 是	✓ 是	✓ 是

使用快速创建链接创建 CloudFormation 堆栈

快速创建链接提供了一种简化的方法，可以直接从 CloudFormation 控制台中的 URL 启动 CloudFormation 堆栈。通过将模板 URL、堆栈名称和模板参数指定为 URL 查询参数，您可以预先填充单个创建堆栈页面并加快堆栈创建速度。这将减少向导页面数和所需的用户输入量，从而简化堆栈创建流程。它还将优化模板重用，因为您可以创建多个 URL 来为同一模板指定不同的值。

URL 格式

快速创建链接遵循以下 URL 格式：

```
https://region-code.console.aws.amazon.com/cloudformation/home?region=region-code#/
stacks/create/review
   ?templateURL=TemplateURL
   &stackName=StackName
   &param_parameterName=parameterValue
```

CloudFormation 支持以下 URL 查询参数：

模板 URL

必需。templateURL 参数指定位于 Amazon S3 存储桶中的堆栈模板的 URL。为避免预签名的 S3 URL 出现访问问题，请确保对该 URL 进行 URL 编码。

支持的 S3 URL 格式：

- https://s3.*region-code*.amazonaws.com/*bucket-name*/*template-name*
- https://*bucket-name*.s3.*region-code*.amazonaws.com/*template-name*
- https://s3-*region-code*.amazonaws.com/*bucket-name*/*template-name*（旧版格式）

堆栈名称

可选。使用 stackName 参数可指定要创建的 CloudFormation 堆栈的名称。堆栈名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 128 个字符。

模板参数

可选。对于堆栈模板中并非 NoEcho 参数类型的参数，请使用 URL 查询字符串中的格式 param_*parameterName*。URL 参数必须包括 param_ 前缀，并且参数名称段必须与模板中的参数名称完全匹配。例如：param_DBName。

CloudFormation 忽略模板不存在的参数，以及在定义时将其 NoEcho 属性设置为 `true` 类型的任意参数（通常为用户名和密码）。URL 参数将覆盖模板中指定的默认值。您可以包含所需数量的参数。

> **Important**
>
> 我们建议不要将敏感信息直接嵌入 CloudFormation 模板中，而应使用堆栈模板中的动态参数来引用在 CloudFormation 外部存储和管理的敏感信息，例如 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager 中的敏感信息。
> 有关更多信息，请参阅请勿将凭证嵌入您的模板最佳实践。

所有查询参数名称都区分大小写。在创建堆栈之前，用户可以在控制台中覆盖这些值。

示例

以下示例基于 WordPress 基本单一实例示例模板。查询字符串包含必需的 `templateURL` 参数以及 `stackName`、`DBName`、`InstanceType` 和 `KeyName` 参数。

为清晰起见，以下 URL 包含换行符。

```
https://us-east-2.console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/
create/review
   ?templateURL=https://s3.us-east-2.amazonaws.com/cloudformation-templates-us-east-2/
WordPress_Single_Instance.template
   &stackName=MyWPBlog
   &param_DBName=mywpblog
   &param_InstanceType=t2.medium
```

以下 URL 包括与上一个示例相同的参数，但删除了换行符。这是实际的 URL 格式。

```
https://us-east-2.console.aws.amazon.com/cloudformation/home?
region=us-east-2#/stacks/create/review?templateURL=https://
s3.us-east-2.amazonaws.com/cloudformation-templates-us-east-2/
WordPress_Single_Instance.template&stackName=MyWPBlog&param_DBName=mywpblog&param_InstanceType=
```

使用快速创建链接创建堆栈

打开快速创建链接时，您会被定向到 CloudFormation 控制台。控制台将直接打开快速创建堆栈页面，并且提供的值将自动用于参数。

使用快速创建链接创建堆栈（控制台）

1. 在快速创建堆栈页面上，对于模板、模板 URL，确认模板 URL 是否正确。
2. 展开查看模板部分以验证模板。
3. 对于堆栈名称，请验证预先填充的堆栈名称。
4. 查看参数部分。验证预填充的参数值是否正确。填写 URL 中未指定的任何必填参数。如果需要，修改任何预填充的值。
5. 接下来，您可以配置以下设置：

   - 标签：使用键值对组织资源。
   - 权限：选择堆栈操作的 IAM 服务角色。
   - 堆栈故障选项：选择回滚（默认）或保留资源。
   - 堆栈策略：控制资源更新权限。
   - 回滚配置：配置基于 CloudWatch 警报的回滚。
   - 通知选项：为堆栈事件设置 Amazon SNS 通知。
   - 堆栈创建选项：定义最长堆栈创建时间并启用终止保护以防止意外删除。

   有关更多信息，请参阅 配置堆栈选项。
6. 对于功能，填写所有必需的确认信息。例如，如果您的模板包含 IAM 资源，请选择我确认此模板可创建 IAM 资源以指定您要使用模板中的 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。
7. （可选）在创建堆栈前，您可以创建更改集来预览堆栈的配置。选择创建更改集，然后按照说明进行操作。有关更多信息，请参阅 预览堆栈的配置。
8. 准备就绪后，选择创建堆栈以启动堆栈，然后在事件选项卡中监控堆栈创建进度。有关更多信息，请参阅 监控堆栈进度。

使用 Express Mode 更快地部署 Amazon CloudFormation 堆栈

借助 CloudFormation Express Mode，您可以在 CloudFormation 应用资源配置后立即完成堆栈操作，无需等待资源完全稳定。在迭代开发工作流程中，使用 Express Mode 可以更快地创建、更新和删除堆栈。

Express Mode 适用于所有现有的 CloudFormation 模板。您无需进行任何模板更改即可使用 Express Mode。

主题

- 何时使用 Express Mode
- Express Mode 的工作原理
- CloudFormation 在 Express Mode下如何处理依赖关系
- Express Mode 入门
- Express Mode 下的回滚行为
- Express Mode 和嵌套堆栈
- 资源就绪性注意事项
- 不支持的功能和限制
- 与 Amazon CDK 热交换部署的对比
- 相关资源

何时使用 Express Mode

在开发和迭代工作流程中，当速度比等待资源完全稳定更为重要时，可使用 Express Mode。在以下情况下，可使用 Express Mode：

- 在开发过程中，您需要对模板更改进行迭代。
- 您希望更快地获得有关资源配置是否有效的反馈。
- 在继续操作之前，您无需等待资源完全投入运行。

对于生产环境部署，若您需要在确认资源已完全稳定并正常运行后才将部署视为完成，可使用默认部署模式。

Express Mode 的工作原理

在默认部署模式下，CloudFormation 会等待每个资源达到完全稳定的状态，然后再将该资源报告为“已完成”。例如，在创建 Amazon EC2 实例时，CloudFormation 会等待该实例进入 `running` 状态。

在 Express Mode 下，CloudFormation 通常会在用于创建、更新或删除资源的 API 调用成功后，立即将该资源操作视为已完成。在大多数情况下，CloudFormation 不会等待资源达到最终运行状态。

这意味着在 Express Mode 下的堆栈操作完成后：

- 资源可能仍在初始化或传播更改。
- 资源可能尚未准备好处理流量或接受连接。

- 删除操作可能仍在进行中。

CloudFormation 在 Express Mode下如何处理依赖关系

Express Mode 会遵循模板中定义的资源依赖关系。如果某个资源引用了另一个资源的 ID 或属性（通过 `Ref` 或 `Fn::GetAtt`），CloudFormation 会在启动被引用资源之前，确认该资源的配置已应用完毕。如果某个依赖资源因被引用的资源尚未就绪而遇到临时故障，CloudFormation 会重试该操作。

彼此之间没有依赖关系的资源可以并行执行。

Express Mode 不会改变资源的依赖顺序。该顺序会在堆栈操作报告“完成”时发生变化。

Express Mode 入门

您可以将 Express Mode 与 Amazon Command Line Interface (Amazon CLI)、Amazon Cloud Development Kit (Amazon CDK) (Amazon CDK)、CloudFormation 控制台和更改集一起使用。以下各节介绍了如何通过每种方法启用 Express Mode。

将 Express Mode 与 Amazon CLI 结合使用

要在将 Express Mode 与 Amazon CLI 结合使用，请将 `--deployment-config` 参数添加到您的堆栈操作命令中。

使用 Express Mode 来创建堆栈

以下命令使用 Express Mode 来创建堆栈：

```
aws cloudformation create-stack --stack-name my-stack \
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS"}'
```

使用 Express Mode 来更新堆栈

以下命令使用 Express Mode 来更新堆栈：

```
aws cloudformation update-stack --stack-name my-stack \
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS"}'
```

使用 Express Mode 来删除堆栈

以下命令使用 Express Mode 来删除堆栈：

```
aws cloudformation delete-stack --stack-name my-stack \
    --deployment-config '{"mode": "EXPRESS"}'
```

将 Express Mode 与 Amazon CDK 结合使用

要将 Express Mode 与 Amazon Cloud Development Kit (Amazon CDK) (Amazon CDK) 结合使用，请在 `cdk deploy` 命令中使用 `--express` 标志：

```
cdk deploy --express
```

将 Express Mode 与 Amazon SAM 结合使用

要将 Express Mode 与 Amazon SAM CLI 结合使用，请在您的部署或同步命令中添加 `--express` 标志：

```
sam deploy --express
```

```
sam sync --express
```

要在项目配置中保留该设置，请使用 `--save-params`：

```
sam deploy --express --save-params
```

这会将 `express = true` 保存到您的 `samconfig.toml` 文件中，以便后续部署时自动使用 Express Mode。

启用 Express Mode 后，Amazon SAM CLI 会将部署配置传递给 CloudFormation。`--disable-rollback` 标志控制部署配置中的回滚行为。

将 Express Mode 与控制台结合使用

在 CloudFormation 控制台中创建或更新堆栈时，您可以在配置堆栈选项步骤中启用 Express Mode。在“部署选项”下，将部署模式选为“Express”。

将 Express Mode 与更改集结合使用

您可以在创建更改集时指定部署配置，从而将 Express Mode 与更改集结合使用：

```
aws cloudformation create-change-set --stack-name my-stack \
    --change-set-name my-change-set \
```

```
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS"}'
```

当您执行更改集时，CloudFormation 会应用在创建更改集时指定的 Express Mode 部署配置。

Express Mode 下的回滚行为

当您使用 Express Mode 时，CloudFormation 会默认禁用回滚功能。如果资源操作失败，CloudFormation 不会自动回滚所做的更改。

要在使用 Express Mode 时重新启用回滚，请在 `--deployment-config` 参数中将 `disableRollback` 设置为 `false`：

```
aws cloudformation create-stack --stack-name my-stack \
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS", "disableRollback": false}'
```

Express Mode 和嵌套堆栈

当您在父堆栈上使用“Express Mode”时，该设置会传播到层次结构中的所有嵌套堆栈。您不需要为每个嵌套堆栈分别指定 Express Mode。根堆栈及其嵌套堆栈中的所有资源，一旦其配置被应用，便立即完成。

资源就绪性注意事项

Express Mode 会在资源达到完全稳定状态之前就完成资源操作，因此任何资源都可能需要额外的时间才能完全准备好处理流量或完成跨区域传播。下表通过不同资源类型的示例，说明了您可以预期的资源行为类型。这些示例仅供说明，并不代表所支持资源的完整列表。

资源类型	操作	就绪行为
Amazon CloudFront 分配	创建/更新	配置被应用后，分发可能需要几分钟才能在全球范围内部署完成。
Amazon EC2 实例	创建	堆栈操作完成后，实例可能仍在初始化中（运行用户数据脚本、状态检查）。
Amazon Lambda 函数	删除	删除操作报告“完成”后，函数资源可能仍在进行清理。

资源类型	操作	就绪行为
Amazon Elastic Container Service (Amazon ECS) 服务	创建/更新	在堆栈操作完成后，服务任务可能仍在启动中，或正进入稳定状态。

不支持的功能和限制

以下功能和服务在 Express Mode 下不受支持：

CloudFormation StackSets

CloudFormation 不支持将 Express Mode 与 StackSets 操作结合使用。

以下资源在 Express Mode 下具有特定的行为：

自定义资源

自定义资源（包括 `AWS::CloudFormation::CustomResource` 和 `Custom::*` 资源）遵循其默认行为。即使在 Express Mode 下，CloudFormation 也会等待自定义资源发送响应信号，才会将该操作视为“完成”。要防止运行缓慢或卡住的自定义资源延迟快速迭代的工作流，请使用 `ServiceTimeout` 属性来设置最大等待时间。有关更多信息，请参阅 响应超时。

与 Amazon CDK 热交换部署的对比

下表将 Express Mode 与 Amazon CDK 热交换部署进行对比。

功能	Express Mode	Amazon CDK 热交换
部署机制	所有资源操作均使用 CloudFormation	绕过 CloudFormation 并直接调用服务 API
模板兼容性	适用于所有现有的 CloudFormation 模板	支持有限的资源类型
堆栈状态	堆栈状态与模板保持一致	堆栈状态可能会偏离模板

功能	Express Mode	Amazon CDK 热交换
回滚支持	支持回滚（默认禁用）	不支持回滚
资源覆盖范围	所有 CloudFormation 资源类型	有限的资源类型（Amazon Lambda, Amazon ECS、Amazon Step Functions 等）
推荐用途	开发和迭代工作流	仅使用受支持资源的开发工作流

相关资源

- 了解堆栈部署时间
- 选择预置资源时如何处理失败
- 嵌套堆栈
- 使用更改集更新堆栈
- 《Amazon CloudFormation API 参考》中的 CreateStack。

了解堆栈部署时间

当您创建、更新或删除堆栈时，CloudFormation 会预置模板中的各项资源，待资源准备就绪、能够处理流量时，它会报告堆栈操作已完成。本主题介绍了 CloudFormation 如何预置堆栈、哪些因素影响堆栈操作的耗时以及如何缩短部署时间。

主题

- CloudFormation 如何预置堆栈
- 哪些因素影响部署时间
- 缩短部署时间的方法
- 相关资源

CloudFormation 如何预置堆栈

CloudFormation 会读取模板中资源之间的依赖项。通过 `Ref`、`Fn::GetAtt` 或 `DependsOn` 引用其他资源的资源将在其引用的资源之后预置。彼此之间没有依赖项的资源会并行预置。

对于每个资源，CloudFormation 都会应用资源配置，然后确认该资源已准备就绪、能够处理流量。每个资源准备就绪后，CloudFormation 就会向前推进，因此依赖资源会尽早启动。因此，依赖链较长的堆栈的改进幅度最大。当堆栈中的所有资源均准备就绪时，CloudFormation 会报告堆栈操作已完成。

此行为会自动应用于堆栈操作。您无需更改模板，也无需设置任何参数。

哪些因素影响部署时间

依赖链

相互依赖的最长资源链设置了堆栈操作的最短时间。不属于该链的资源是并行预置的，通常不会增加总时间。

需要更长时间才能准备就绪的资源

在包含 Amazon RDS 数据库实例、Elastic Load Balancing 负载均衡器、NAT 网关或 Amazon CloudFront 分配等资源的工作负载中，大部分部署时间都花在等待这些资源准备就绪并能够处理流量上。

引用资源属性的输出

如果堆栈输出引用了某个资源属性（例如使用 `Fn::GetAtt`），则 CloudFormation 会等到该属性值可用后再完成堆栈操作。这段等待可能会增加操作所需的时间。

缩短部署时间的方法

- 使用快速模式进行开发迭代。一旦 CloudFormation 应用了资源配置，快速模式就会完成堆栈操作，并且资源在后台继续准备就绪。有关更多信息，请参阅 Express Mode。
- 根据资源的更改频率对其进行分组。将准备就绪耗时较长的资源（例如数据库）与经常更新的资源分置于不同的堆栈中。随后对频繁更改的堆栈所做的更新将不包含那些运行时间较长的资源。
- 移除不需要的依赖项。仅在资源必须等待其他资源时声明 `DependsOn`。更少的依赖项使 CloudFormation 能够并行预置更多资源。
- 仅在需要时在输出中引用资源属性。引用资源属性的输出可以扩展堆栈操作，直到这些值可用。

相关资源

- CloudFormation 的工作原理
- Express Mode
- 使用更改集更新 CloudFormation 堆栈

Amazon CLI 和 PowerShell 的 CloudFormation 堆栈操作命令示例

以下命令行示例演示了如何使用 Amazon CLI 和 PowerShell 执行各个 CloudFormation 操作。这些示例仅包括最常用的操作。有关完整列表，请参阅《Amazon CLI 命令参考》中的 cloudformation。

本指南中的示例使用反斜杠 (\) 的惯例来表示长命令行在下一行继续。

主题

- 取消堆栈更新
- 继续回滚更新
- 创建堆栈
- 创建包含转换的堆栈
- 删除堆栈
- 描述堆栈事件
- 描述堆栈资源
- 描述堆栈资源
- 描述堆栈
- 获取模板
- 列出堆栈资源
- 列出堆栈
- 更新堆栈
- 验证模板
- 使用 Amazon CLI 将本地构件上传到 S3 存储桶

取消堆栈更新

cancel-update-stack 命令用于取消堆栈更新。有关更多信息，请参阅 取消堆栈更新。

CLI

Amazon CLI

取消正在进行的堆栈更新

以下 `cancel-update-stack` 命令会取消 `myteststack` 堆栈上的堆栈更新：

```
aws cloudformation cancel-update-stack --stack-name myteststack
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CancelUpdateStack。

PowerShell

适用于 PowerShell V4 的工具

示例 1：取消指定堆栈的更新。

```
Stop-CFNUpdateStack -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 CancelUpdateStack。

Tools for PowerShell V5

示例 1：取消指定堆栈的更新。

```
Stop-CFNUpdateStack -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 CancelUpdateStack。

继续回滚更新

使用 continue-update-rollback 命令继续回滚更新。有关更多信息，请参阅 继续回滚更新。

CLI

Amazon CLI

重试更新回滚

以下 `continue-update-rollback` 示例从之前失败的堆栈更新中恢复回滚操作。

```
aws cloudformation continue-update-rollback \
    --stack-name my-stack
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ContinueUpdateRollback。

PowerShell

适用于 PowerShell V4 的工具

示例 1：继续指定堆栈的回滚，该堆栈的状态应为“UPDATE_ROLLBACK_FAILED”。如果继续回滚成功，堆栈将进入“UPDATE_ROLLBACK_COMPLETE”状态。

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 ContinueUpdateRollback。

Tools for PowerShell V5

示例 1：继续指定堆栈的回滚，该堆栈的状态应为“UPDATE_ROLLBACK_FAILED”。如果继续回滚成功，堆栈将进入“UPDATE_ROLLBACK_COMPLETE”状态。

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 ContinueUpdateRollback。

创建堆栈

使用 create-stack 命令创建堆栈。您必须提供堆栈名称、有效模板的位置和所有输入参数。参数键名称区分大小写。如果您输入错误的参数键名称，则 CloudFormation 不会创建堆栈，并且会报告模板不包含该参数。

以下示例演示了如何使用指定的名称、模板和输入参数创建新堆栈。

CLI

Amazon CLI

创建 Amazon CloudFormation 堆栈

以下 `create-stacks` 命令将使用 `sampletemplate.json` 模板创建名为 `myteststack` 的堆栈：

```
aws cloudformation create-stack --stack-name myteststack --template-body file://
sampletemplate.json --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
 ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2
```

输出：

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的“堆栈”。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateStack。

PowerShell

适用于 PowerShell V4 的工具

示例 1：使用指定名称创建新堆栈。模板是从提供的内容中解析出来的，使用了自定义参数（“PK1”和“PK2”代表模板内容中声明的参数名称，“PV1”和“PV2”代表这些参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则不会将其回滚。

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

示例 2：使用指定名称创建新堆栈。模板是从提供的内容中解析出来的，使用了自定义参数（“PK1”和“PK2”代表模板内容中声明的参数名称，“PV1”和“PV2”代表这些参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚。

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
```

```
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

示例 3：使用指定名称创建新堆栈。模板是从 Amazon S3 URL 获得的，使用了自定义参数（“PK1”代表模板内容中声明的参数名称，“PV1”代表该参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚（与指定 -DisableRollback $false 时相同）。

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 4：使用指定名称创建新堆栈。模板是从 Amazon S3 URL 获得的，使用了自定义参数（“PK1”代表模板内容中声明的参数名称，“PV1”代表该参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚（与指定 -DisableRollback $false 时相同）。指定的通知 AEN 将收到发布的堆栈相关事件。

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 CreateStack。

Tools for PowerShell V5

示例 1：使用指定名称创建新堆栈。模板是从提供的内容中解析出来的，使用了自定义参数（“PK1”和“PK2”代表模板内容中声明的参数名称，“PV1”和“PV2”代表这些参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则不会将其回滚。

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
```

```
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

示例 2：使用指定名称创建新堆栈。模板是从提供的内容中解析出来的，使用了自定义参数（“PK1”和“PK2”代表模板内容中声明的参数名称，“PV1”和“PV2”代表这些参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚。

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

示例 3：使用指定名称创建新堆栈。模板是从 Amazon S3 URL 获得的，使用了自定义参数（“PK1”代表模板内容中声明的参数名称，“PV1”代表该参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚（与指定 -DisableRollback $false 时相同）。

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 4：使用指定名称创建新堆栈。模板是从 Amazon S3 URL 获得的，使用了自定义参数（“PK1”代表模板内容中声明的参数名称，“PV1”代表该参数的值）。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚（与指定 -DisableRollback $false 时相同）。指定的通知 AEN 将收到发布的堆栈相关事件。

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
```

```
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
                -NotificationARN @( "arn1", "arn2" )
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 CreateStack。

> **Note**
>
> 您可以使用 Amazon CLI `--template-url` 选项在 Amazon S3 或 Amazon Systems Manager 中指定模板文件位置。
> 对于 Amazon S3，URL 必须以 `https://` 开头。不支持 S3 静态网站 URL。
>
> ```
> --template-url https://s3.region-code.amazonaws.com/bucket-name/template-name
> ```
>
> 对于 Amazon Systems Manager，请使用以下格式：
>
> ```
> --template-url "ssm-doc://arn:aws:ssm:region-code:account-id:document/document-name"
> ```

Express Mode

要使用 Express Mode 来创建堆栈，请添加 `--deployment-config` 参数。Express Mode 会在 CloudFormation 应用资源配置后立即完成栈操作。有关更多信息，请参阅 Express Mode。

```
aws cloudformation create-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --deployment-config '{"mode": "EXPRESS"}'
```

创建包含转换的堆栈

使用 deploy 命令创建包含转换的堆栈。使用模板创建包含转换的堆栈时，您必须使用更改集。`deploy` 命令将两个步骤（创建更改集和执行变更集）合并为一个命令。

Amazon CLI

以下 `deploy` 命令将使用指定的名称、模板和输入参数创建一个堆栈。

```
aws cloudformation deploy --stack-name myteststack \
```

```
--template /path_to_template/my-template.json \
--parameter-overrides Key1=Value1 Key2=Value2
```

删除堆栈

使用 delete-stack 命令删除堆栈。有关更多信息，请参阅 删除堆栈。

CLI

Amazon CLI

删除堆栈

以下 delete-stack 示例删除指定的堆栈。

```
aws cloudformation delete-stack \
    --stack-name my-stack
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteStack。

PowerShell

适用于 PowerShell V4 的工具

示例 1：删除指定的堆栈。

```
Remove-CFNStack -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 DeleteStack。

Tools for PowerShell V5

示例 1：删除指定的堆栈。

```
Remove-CFNStack -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 DeleteStack。

如果删除失败并返回 DELETE_FAILED 状态，可以选择使用 --deletion-mode 选项强制删除堆栈。以下是可以与 deletion-mode 一起使用的值：

- STANDARD：正常删除堆栈。这是默认删除模式。
- FORCE_DELETE_STACK：删除堆栈并跳过所有无法删除的资源。

Amazon CLI

以下 delete-stack 命令将 FORCE_DELETE_STACK 值与 deletion-mode 参数结合使用，强制删除 *myteststack* 堆栈：

```
aws cloudformation delete-stack --stack-name myteststack \
    --deletion-mode FORCE_DELETE_STACK
```

此命令不生成任何输出。

使用 FORCE_DELETE_STACK 后，您可以使用 list-stack-resources 命令列出在堆栈删除过程中跳过的资源。保留的资源将显示为 DELETE_SKIPPED 状态。有关更多信息，请参阅 列出堆栈资源。

Express Mode

要使用 Express Mode 来删除堆栈，请添加 --deployment-config 参数。

```
aws cloudformation delete-stack --stack-name myteststack \
    --deployment-config '{"mode": "EXPRESS"}'
```

有关更多信息，请参阅 Express Mode。

描述堆栈事件

使用 describe-stack-events 命令描述堆栈事件。有关更多信息，请参阅 监控堆栈进度。

CLI

Amazon CLI

描述堆栈事件

以下 describe-stack-events 示例显示指定堆栈的 2 个最新事件。

```
aws cloudformation describe-stack-events \
    --stack-name my-stack \
    --max-items 2

{
    "StackEvents": [
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "EventId": "4e1516d0-e4d6-xmpl-b94f-0a51958a168c",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.556Z",
            "ResourceStatus": "UPDATE_COMPLETE"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "EventId": "4dd3c810-e4d6-xmpl-bade-0aaf8b31ab7a",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.127Z",
            "ResourceStatus": "UPDATE_COMPLETE_CLEANUP_IN_PROGRESS"
        }
    ],

 "NextToken": "eyJOZXh0VG9XMPLiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DescribeStackEvents。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回指定堆栈的所有堆栈相关事件。

```
Get-CFNStackEvent -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 DescribeStackEvents。

Tools for PowerShell V5

示例 1：返回指定堆栈的所有堆栈相关事件。

```
Get-CFNStackEvent -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 DescribeStackEvents。

描述堆栈资源

CLI

Amazon CLI

获取有关堆栈资源的信息

以下 `describe-stack-resource` 示例显示指定堆栈中名为 `MyFunction` 的资源的详细信息。

```
aws cloudformation describe-stack-resource \
    --stack-name MyStack \
    --logical-resource-id MyFunction
```

输出：

```
{
    "StackResourceDetail": {
        "StackName": "MyStack",
        "StackId": "arn:aws:cloudformation:us-east-2:123456789012:stack/MyStack/
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
        "LogicalResourceId": "MyFunction",
        "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
        "ResourceType": "AWS::Lambda::Function",
        "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
        "ResourceStatus": "UPDATE_COMPLETE",
```

```
        "Metadata": "{}",
        "DriftInformation": {
            "StackResourceDriftStatus": "IN_SYNC"
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DescribeStackResource。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回与指定堆栈关联的模板中用逻辑 ID “MyDBInstance”标识的资源的描述。

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 DescribeStackResource。

Tools for PowerShell V5

示例 1：返回与指定堆栈关联的模板中用逻辑 ID “MyDBInstance”标识的资源的描述。

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 DescribeStackResource。

描述堆栈资源

CLI

Amazon CLI

获取有关堆栈资源的信息

以下 `describe-stack-resources` 示例显示指定堆栈中的资源详细信息。

```
aws cloudformation describe-stack-resources \
    --stack-name my-stack
```

输出：


```
{
    "StackResources": [
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "bucket",
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
            "Timestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
            "Timestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        }
    ]
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DescribeStackResources。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回与指定堆栈关联的最多 100 个资源的 Amazon 资源描述。要获取与堆栈关联的所有资源的详细信息，请使用 Get-CFNStackResourceSummary，该命令还支持对结果进行手动分页。

```
Get-CFNStackResourceList -StackName "myStack"
```

示例 2：返回与指定堆栈关联的模板中用逻辑 ID “Ec2Instance”标识的 Amazon EC2 实例的描述。

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

示例 3：对于包含用实例 ID “i-123456”标识的 Amazon EC2 实例的堆栈，返回与该堆栈关联的最多 100 个资源的描述。要获取与堆栈关联的所有资源的详细信息，请使用 Get-CFNStackResourceSummary，该命令还支持对结果进行手动分页。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

示例 4：返回在堆栈模板中用逻辑 ID “Ec2Instance”标识的 Amazon EC2 实例的描述。该堆栈用其包含的资源的物理资源 ID 进行标识，在此例中即为实例 ID 为“i-123456”的 Amazon EC2 实例。也可以根据模板内容使用其他物理资源来识别堆栈，例如 Amazon S3 存储桶。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 DescribeStackResources。

Tools for PowerShell V5

示例 1：返回与指定堆栈关联的最多 100 个资源的 Amazon 资源描述。要获取与堆栈关联的所有资源的详细信息，请使用 Get-CFNStackResourceSummary，该命令还支持对结果进行手动分页。

```
Get-CFNStackResourceList -StackName "myStack"
```

示例 2：返回与指定堆栈关联的模板中用逻辑 ID “Ec2Instance”标识的 Amazon EC2 实例的描述。

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

示例 3：对于包含用实例 ID “i-123456”标识的 Amazon EC2 实例的堆栈，返回与该堆栈关联的最多 100 个资源的描述。要获取与堆栈关联的所有资源的详细信息，请使用 Get-CFNStackResourceSummary，该命令还支持对结果进行手动分页。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

示例 4：返回在堆栈模板中用逻辑 ID “Ec2Instance”标识的 Amazon EC2 实例的描述。该堆栈用其包含的资源的物理资源 ID 进行标识，在此例中即为实例 ID 为“i-123456”的 Amazon EC2 实例。也可以根据模板内容使用其他物理资源来识别堆栈，例如 Amazon S3 存储桶。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 DescribeStackResources。

描述堆栈

CLI

Amazon CLI

描述 Amazon CloudFormation 堆栈

以下 `describe-stacks` 命令显示 `myteststack` 堆栈的摘要信息：

```
aws cloudformation describe-stacks --stack-name myteststack
```

输出：

```
{
```

```
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
            "Tags": [],
            "Outputs": [
                {
                    "Description": "Name of S3 bucket to hold website content",
                    "OutputKey": "BucketName",
                    "OutputValue": "myteststack-s3bucket-jssofi1zie2w"
                }
            ],
            "StackStatusReason": null,
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "CREATE_COMPLETE",
            "DisableRollback": false
        }
    ]
}
```

有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的“堆栈”。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DescribeStacks。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
```

```
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DescribeStacks。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回描述用户的所有堆栈的堆栈实例集合。

```
Get-CFNStack
```

示例 2：返回描述指定堆栈的堆栈实例

```
Get-CFNStack -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 DescribeStacks。

Tools for PowerShell V5

示例 1：返回描述用户的所有堆栈的堆栈实例集合。

```
Get-CFNStack
```

示例 2：返回描述指定堆栈的堆栈实例

```
Get-CFNStack -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 DescribeStacks。

默认情况下，`describe-stacks` 命令返回参数值。为了防止返回密码等敏感参数值，请在 CloudFormation 模板中包含一个设置为 `TRUE` 的 `NoEcho` 属性。

Important

使用 `NoEcho` 属性不会遮蔽在以下各区段中存储的任何信息：

- `Metadata` 模板区段。CloudFormation 不会转换、修改或编辑您在 `Metadata` 部分中包含的任何信息。有关更多信息，请参阅 Metadata。
- `Outputs` 模板区段。有关更多信息，请参阅 Outputs。
- 资源定义的 `Metadata` 属性。有关更多信息，请参阅 `Metadata` 属性。

强烈建议您不要使用这些机制来包含敏感信息，例如密码。

> **Important**
>
> 我们建议不要将敏感信息直接嵌入 CloudFormation 模板中，而应使用堆栈模板中的动态参数来引用在 CloudFormation 外部存储和管理的敏感信息，例如 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager 中的敏感信息。
> 有关更多信息，请参阅请勿将凭证嵌入您的模板最佳实践。

获取模板

CLI

Amazon CLI

查看 Amazon CloudFormation 堆栈的模板正文

以下 `get-template` 命令显示 `myteststack` 堆栈的模板：

```
aws cloudformation get-template --stack-name myteststack
```

输出：

```
{
    "TemplateBody": {
        "AWSTemplateFormatVersion": "2010-09-09",
        "Outputs": {
            "BucketName": {
                "Description": "Name of S3 bucket to hold website content",
                "Value": {
                    "Ref": "S3Bucket"
                }
            }
        },
        "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
        "Resources": {
            "S3Bucket": {
                "Type": "AWS::S3::Bucket",
                "Properties": {
```

```
                    "AccessControl": "PublicRead"
                }
            }
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetTemplate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回与指定堆栈关联的模板。

```
Get-CFNTemplate -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 GetTemplate。

Tools for PowerShell V5

示例 1：返回与指定堆栈关联的模板。

```
Get-CFNTemplate -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 GetTemplate。

列出堆栈资源

CLI

Amazon CLI

列出堆栈中的资源

以下命令显示指定堆栈中的资源列表。

```
aws cloudformation list-stack-resources \
```

```
    --stack-name my-stack
```

输出：

```
{
    "StackResourceSummaries": [
        {
            "LogicalResourceId": "bucket",
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
            "LastUpdatedTimestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
            "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
            "LastUpdatedTimestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        }
    ]
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListStackResources。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回与指定堆栈关联的所有资源的描述。

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 ListStackResources。

Tools for PowerShell V5

示例 1：返回与指定堆栈关联的所有资源的描述。

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 ListStackResources。

列出堆栈

使用 list-stacks 命令列出堆栈。要只列出具有指定状态代码的堆栈，请添加 `--stack-status-filter` 选项。可以为 `--stack-status-filter` 选项指定一个或多个堆栈状态代码。有关更多信息，请参阅 堆栈状态代码。

CLI

Amazon CLI

列出 Amazon CloudFormation 堆栈

以下 `list-stacks` 命令显示具有 `CREATE_COMPLETE` 状态的所有堆栈的摘要信息：

```
aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE
```

输出：

```
[
    {
```

```
        "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
        "TemplateDescription": "AWS CloudFormation Sample Template S3_Bucket:
 Sample template showing how to create a publicly accessible S3 bucket.
 **WARNING** This template creates an S3 bucket. You will be billed for the AWS
 resources used if you create a stack from this template.",
        "StackStatusReason": null,
        "CreationTime": "2013-08-26T03:27:10.190Z",
        "StackName": "myteststack",
        "StackStatus": "CREATE_COMPLETE"
    }
]
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListStacks。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回所有堆栈的摘要信息。

```
Get-CFNStackSummary
```

示例 2：返回当前正在创建的所有堆栈的摘要信息。

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

示例 3：返回当前正在创建或更新的所有堆栈的摘要信息。

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 ListStacks。

Tools for PowerShell V5

示例 1：返回所有堆栈的摘要信息。

```
Get-CFNStackSummary
```

示例 2：返回当前正在创建的所有堆栈的摘要信息。

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

示例 3：返回当前正在创建或更新的所有堆栈的摘要信息。

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 ListStacks。

更新堆栈

使用 update-stack 命令可直接更新堆栈。您指定要更新的堆栈、参数值和功能，如果您希望使用更新后的模板，还需指定模板的名称。有关更多信息，请参阅 直接更新堆栈。

CLI

Amazon CLI

更新 Amazon CloudFormation 堆栈

以下 update-stack 命令更新 mystack 堆栈的模板和输入参数：

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template --
parameters ParameterKey=KeyPairName,ParameterValue=SampleKeyPair
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,SampleSubnetID2
```

以下 update-stack 命令更新 mystack 堆栈的 SubnetIDs 参数值：如果您没有指定参数值，则将使用模板中指定的默认值：

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template
 --parameters ParameterKey=KeyPairName,UsePreviousValue=true
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,UpdatedSampleSubnetID2
```

以下 update-stack 命令向 mystack 堆栈添加两个堆栈通知主题：

```
aws cloudformation update-stack --stack-name mystack --use-previous-template --
notification-arns "arn:aws:sns:use-east-1:123456789012:mytopic1" "arn:aws:sns:us-
east-1:123456789012:mytopic2"
```

有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的 Amazon 堆栈更新。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateStack。

PowerShell

适用于 PowerShell V4 的工具

示例 1：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”代表模板中声明的参数名称，“PV1”代表参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 2：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”和“PK2”代表模板中声明的参数名称，“PV1”和“PV2”代表请求的参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

示例 3：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”代表模板中声明的参数名称，“PV2”代表参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 4：使用从 Amazon S3 获得的指定模板和自定义参数更新堆栈“MyStack”。“PK1”和“PK2”代表模板中声明的参数名称，“PV1”和“PV2”代表请求的参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

示例 5：使用从 Amazon S3 获得的指定模板和自定义参数更新堆栈“MyStack”（在此例中假设该堆栈包含 IAM 资源）。“PK1”和“PK2”代表模板中声明的参数名称，“PV1”和“PV2”代表请求的参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。由于堆栈包含 IAM 资源，您需要指定 -Capabilities "CAPABILITY_IAM" 参数，否则更新将失败并出现“InsufficientCapabilities”错误。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 UpdateStack。

Tools for PowerShell V5

示例 1：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”代表模板中声明的参数名称，“PV1”代表参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 2：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”和“PK2”代表模板中声明的参数名称，“PV1”和“PV2”代表请求的参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

示例 3：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”代表模板中声明的参数名称，“PV2”代表参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 4：使用从 Amazon S3 获得的指定模板和自定义参数更新堆栈“MyStack”。“PK1”和“PK2”代表模板中声明的参数名称，“PV1”和“PV2”代表请求的参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

示例 5：使用从 Amazon S3 获得的指定模板和自定义参数更新堆栈“MyStack”（在此例中假设该堆栈包含 IAM 资源）。“PK1”和“PK2”代表模板中声明的参数名称，“PV1”和“PV2”代表请求的参数值。也可以使用“键”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。由于堆栈包含 IAM 资源，您需要指定 -Capabilities "CAPABILITY_IAM" 参数，否则更新将失败并出现“InsufficientCapabilities”错误。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 UpdateStack。

> **Note**
>
> 要移除所有通知，请为 `--notification-arns` 选项指定 `[]`。

Express Mode

要使用 Express Mode 来更新堆栈，请添加 `--deployment-config` 参数：

```
aws cloudformation update-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --deployment-config '{"mode": "EXPRESS"}'
```

有关更多信息，请参阅 Express Mode。

验证模板

使用 validate-template 命令检查模板文件的语法错误。

在验证期间，CloudFormation 首先检查模板是否是有效的 JSON。如果不是，CloudFormation 会检查模板是否是有效的 YAML。如果两种检查都失败，CloudFormation 会返回模板验证错误。

CLI

Amazon CLI

验证 Amazon CloudFormation 模板

以下 `validate-template` 命令验证 `sampletemplate.json` 模板：

```
aws cloudformation validate-template --template-body file://sampletemplate.json
```

输出：

```
{
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template
 showing how to create a publicly accessible S3 bucket. **WARNING** This template
 creates an S3 bucket. You will be billed for the AWS resources used if you
 create a stack from this template.",
    "Parameters": [],
    "Capabilities": []
}
```

有关更多信息，请参阅《Amazon CloudTrail 用户指南》中的“使用 Amazon CloudFormation 模板”。

- 有关 API 的详细信息，请参阅《Amazon CLI 命令参考》中的 ValidateTemplate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：验证指定模板的内容。输出详细说明了模板的功能、描述和参数。

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

示例 2：验证通过 Amazon S3 URL 访问的指定模板。输出详细说明了模板的功能、描述和参数。

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- 有关 API 的详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V4）》中的 ValidateTemplate。

Tools for PowerShell V5

示例 1：验证指定模板的内容。输出详细说明了模板的功能、描述和参数。

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

示例 2：验证通过 Amazon S3 URL 访问的指定模板。输出详细说明了模板的功能、描述和参数。

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- 有关 API 的详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference（V5）》中的 ValidateTemplate。

以下是产生验证错误的响应示例。

```
{
    "ResponseMetadata": {
        "RequestId": "4ae33ec0-1988-11e3-818b-e15a6df955cd"
    },
    "Errors": [
        {
```

```
            "Message": "Template format error: JSON not well-formed. (line 11, column
 8)",
            "Code": "ValidationError",
            "Type": "Sender"
        }
    ],
    "Capabilities": [],
    "Parameters": []
}
A client error (ValidationError) occurred: Template format error: JSON not well-formed.
 (line 11, column 8)
```

> **Note**
>
> `validate-template` 命令仅用于检查模板的语法。它不能确保您为资源指定的属性值对该资源有效，也无法确定创建堆栈时将存在的资源数量。
> 要检查操作有效性，需要尝试创建堆栈。Amazon CloudFormation 堆栈没有沙盒或测试区域，因此您需要为测试期间创建的资源付费。

使用 Amazon CLI 将本地构件上传到 S3 存储桶

现在，您可以使用 Amazon CLI 命令将 CloudFormation 模板引用的本地构件上传到 Amazon S3 存储桶。本地构件是您在模板中引用的文件。您可以在模板中指定本地构件，然后使用 package 命令快速上传它们，而不是将文件手动上传到 S3 存储桶，然后向模板添加其位置。

本地项目是 package 命令将向 Amazon S3 上传的文件或文件夹的路径。例如，构件可以是 Amazon Lambda 函数的源代码或 Amazon API Gateway REST API 的 OpenAPI 文件的本地路径。

使用 package 命令时：

- 如果您指定一个文件，此命令会将该文件直接上传到 S3 存储桶。
- 如果您指定一个文件夹，则命令将为该文件夹创建一个 `.zip` 文件，然后上传该 `.zip` 文件。
- 如果您未指定路径，则命令将为工作目录创建一个 `.zip` 文件，然后上传该文件。

您可指定绝对路径或相对路径，其中相对路径相对于模板的位置。

在上传项目之后，此命令将返回模板的副本，并将对本地项目的引用替换为命令已将项目上传到的 S3 位置。然后，您可使用返回的模板创建或更新堆栈。

> **Note**
>
> 您只能对 package 命令支持的资源属性使用本地项目。有关此命令的更多信息和支持的资源属性的列表，请参阅 Amazon CLI 命令参考中的 package 文档。

先决条件

在开始之前，您必须有一个现有的 Amazon S3 存储桶。

打包并部署包含本地构件的模板

以下模板指定 Lambda 函数的源代码的本地项目。此源代码存储在 `/home/user/code/lambdafunction` 文件夹中。

原始模板

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Transform": "AWS::Serverless-2016-10-31",
  "Resources": {
    "MyFunction": {
      "Type": "AWS::Serverless::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "nodejs18.x",
        "CodeUri": "/home/user/code/lambdafunction"
      }
    }
  }
}
```

以下 package 命令创建函数源代码文件夹的 `.zip` 文件，然后将该文件上传到指定存储桶的根目录。

```
aws cloudformation package \
  --s3-bucket amzn-s3-demo-bucket \
  --template /path_to_template/template.json \
  --output-template-file packaged-template.json \
  --output json
```

该命令在 `--output-template-file` 指定的路径上生成一个新模板。它将构件引用替换为 Amazon S3 位置，如下图所示。`.zip` 文件是使用文件夹内容的 MD5 校验和命名的，而不是使用文件夹名称本身。

结果模板

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Transform": "AWS::Serverless-2016-10-31",
  "Resources": {
    "MyFunction": {
      "Type": "AWS::Serverless::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "nodejs18.x",
        "CodeUri": "s3://amzn-s3-demo-bucket/md5 checksum"
      }
    }
  }
}
```

打包模板的构件后，使用 deploy 命令部署已处理的模板。

```
aws cloudformation deploy \
  --template-file packaged-template.json \
  --stack-name stack-name
```

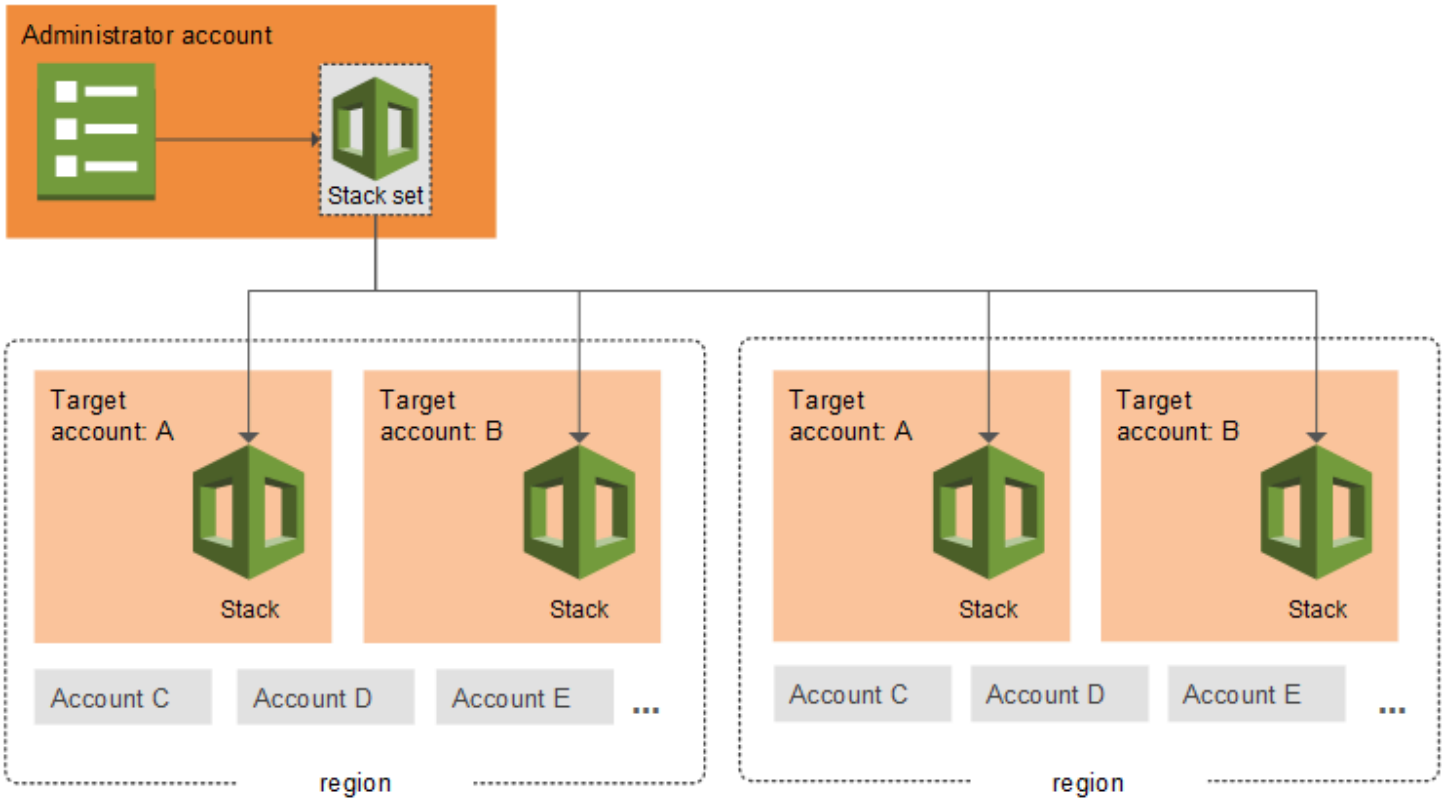
部署大于 51200 字节的模板时，使用带有 `--s3-bucket` 选项的 deploy 命令将模板上传到 S3，如下例所示。

```
aws cloudformation deploy \
  --template-file packaged-template.json \
  --stack-name stack-name \
  --s3-bucket amzn-s3-demo-bucket
```

> **Note**
>
> 有关使用 package 命令上传本地构件的另一个示例，请参阅使用嵌套堆栈将模板拆分为可重复使用的部分。

使用 StackSets 跨账户和区域管理堆栈

Amazon CloudFormation StackSets 让您能够在单个操作中跨多个账户和 Amazon Web Services 区域区域创建、更新或删除堆栈，从而扩展了堆栈的功能。通过使用管理员账户，您可定义和管理 CloudFormation 模板，并以该模板为基础，跨指定的 Amazon Web Services 区域在选定目标账户中预置堆栈。



本节介绍如何开始使用 StackSet，解答有关 StackSet 创建、更新和删除以及故障排除的常见问题。

主题

- StackSets 概念
- 使用 Amazon CloudFormation StackSet 的先决条件
- 使用示例模板开始使用 StackSet。
- 创建具有自行管理权限的 Amazon CloudFormation StackSet
- 创建具有服务托管权限的 Amazon CloudFormation StackSet
- 为 Amazon Organizations 中的堆栈集启用或禁用自动部署
- 更新 Amazon CloudFormation StackSet
- 将堆栈添加到 CloudFormation 堆栈集

- 覆盖 CloudFormation StackSet 中的堆栈参数值
- 从 Amazon CloudFormation StackSet 中删除堆栈
- 删除 Amazon CloudFormation StackSet
- 使用目标账户门防止 StackSet 部署失败
- 为 Amazon CloudFormation StackSet 选择并发模式
- 对 CloudFormation StackSets 进行偏差检测
- 将堆栈导入 Amazon CloudFormation StackSet
- 使用 Amazon CloudFormation StackSet 的最佳实践
- Amazon CloudFormation StackSet 示例模板
- Amazon CloudFormation StackSets 故障排除

StackSets 概念

以下术语和概念对您了解和使用 StackSets 非常有用。

主题

- 管理员账户和目标账户
- Amazon CloudFormation 堆栈集
- StackSet 的权限模型
- 堆栈实例
- StackSet 操作
- StackSet 操作选项
- 标签
- StackSets 状态代码
- 堆栈实例状态代码

管理员账户和目标账户

管理员账户是您用来创建 StackSet 的 Amazon Web Services 账户。对于具有服务托管权限的 StackSet，管理员账户是组织的管理账户或委派管理员账户。您可以登录到创建 StackSet 的 Amazon 管理员账户来管理 StackSet。

目标账户是在其中创建、更新或删除 StackSet 中一个或多个堆栈的账户。先在管理员账户与目标账户之间设置信任关系，然后才能使用 StackSet 在目标账户中创建堆栈。

Amazon CloudFormation 堆栈集

StackSet 充当跨指定 Amazon Web Services 账户 和区域部署的多个堆栈的容器。每个堆栈都基于同一 CloudFormation 模板，但您可以使用参数自定义单个堆栈。

在定义 StackSet 后，您可以在指定的目标账户和 Amazon Web Services 区域中创建、更新或删除堆栈。创建、更新或删除堆栈时，您还可以指定操作首选项。例如，包括要执行操作的区域顺序、堆栈操作停止前的容错阈值以及并发执行堆栈操作的账户数。

StackSet 是一种区域性资源。如果在一个 Amazon Web Services 区域中创建 StackSet，则只能在查看该区域时查看或更改它。

StackSet 的权限模型

您可以使用自行管理权限或服务托管权限创建 StackSet。

使用自行管理权限时，您将创建 StackSets 需要的 IAM 角色来跨账户和区域进行部署。对于在您管理 StackSet 所用的账户与您将堆栈实例部署到的账户之间建立信任关系，这些角色必不可少。使用此权限模型，StackSets 可以部署到您有权创建 IAM 角色的任何 Amazon Web Services 账户 中。

使用服务托管 权限时，您可以将堆栈实例部署到 Amazon Organizations 管理的账户。使用此权限模型，您不必创建必要的 IAM 角色；StackSets 会代表您创建 IAM 角色。使用此模型，您还可以为将来添加到组织的账户开启自动部署。

Amazon Organizations 与 CloudFormation 集成，可帮助您在扩展和增长 Amazon 资源时集中管理和治理您的环境。

- 管理账户 - 您用于创建组织的账户。有关更多信息，请参阅 Amazon Organizations 术语和概念。
- 委派管理员 - 兼容的 Amazon 服务可以将组织中的 Amazon 成员账户注册为该服务中组织账户的管理员。有关更多信息，请参阅可与 Amazon Organizations 一起使用的 Amazon 服务。

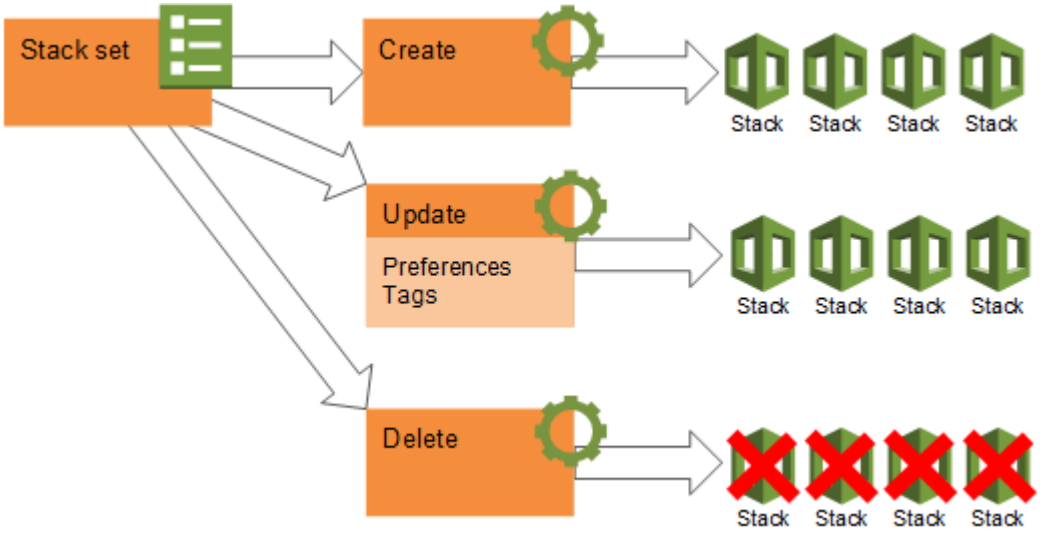
有关创建和管理具有服务托管权限的 StackSet 的更多信息，请参阅以下主题：

- 使用 Amazon Organizations 为 StackSet 激活可信访问权限
- 注册委派管理员成员账户
- 创建具有服务托管权限的 Amazon CloudFormation StackSet

堆栈实例

堆栈实例 是对区域内的目标账户中的堆栈的引用。堆栈实例可在没有堆栈的情况下存在。例如，如果出于某个原因无法创建堆栈，则堆栈实例将显示堆栈创建失败的原因。一个堆栈实例仅与一个 StackSet 关联。

下图显示 StackSet、堆栈操作和堆栈之间的逻辑关系。更新 StackSet 时，所有账户和区域中所有关联的堆栈实例都会更新。



StackSet 操作

您可以在 StackSet 上执行以下操作。

创建 StackSet

创建新 StackSet 涉及指定用于创建堆栈的 CloudFormation 模板、指定要在其中创建堆栈的目标账户以及指定要在其中将堆栈部署到目标账户的 Amazon Web Services 区域。StackSet 可确保将具有相同设置的相同堆栈资源，一致地部署到所选区域内的所有指定目标账户。

更新 StackSet

更新 StackSet 时，将更改推送到 StackSet 中的堆栈。您可以通过下列方法之一更新 StackSet。您的模板更新始终会影响所有堆栈；您不能选择只更新 StackSet 中部分堆栈的模板，但不更新其他堆栈的模板。

- 更改模板中的现有设置或添加新资源，例如，更新特定服务的参数设置或添加新的 Amazon EC2 实例。
- 将该模板替换为其他模板。

- 跨现有或其他区域在现有或其他目标账户中添加堆栈。

删除 堆栈

在删除堆栈时，您将从指定的区域的指定目标账户中，删除堆栈及其所有关联资源。您可以通过以下方法删除堆栈。

- 从某些目标账户中删除堆栈，同时使其他目标账户中的其他堆栈保持运行。
- 从某些区域中删除堆栈，同时使其他区域中的堆栈保持运行。
- 从 StackSet 中删除堆栈，但保存它们，以便通过选择保留堆栈选项来使其独立于 StackSet 继续运行。之后，您便可以管理 CloudFormation 中 StackSet 之外的保留堆栈。
- 在准备删除整个 StackSet 时，先删除 StackSet 中的所有堆栈。

删除 StackSet

StackSet 中没有堆栈实例时，才能删除它。

StackSet 操作选项

本节描述的选项可以控制时间和允许的失败次数，以便成功执行 StackSet 操作并防止丢失堆栈资源。

最大并发账户数

利用此设置（适用于创建、更新和删除工作流），您可以指定在其中一次执行一项操作的目标账户的最大数量或百分比。较小的数量或百分比意味着一次在较少的目标账户中执行一项操作。按照 Deployment order（部署顺序）框中指定的顺序一次在一个区域中执行操作。例如，如果您将堆栈部署到两个区域内的 10 个目标账户，将 Maximum concurrent accounts（最大并发账户数）设置为 50 和 By percentage（按百分比），则会在移到下一个区域并开始部署到前 5 个目标账户之前，将堆栈部署到第一个区域中的 5 个账户，然后部署到第一个区域内的另外 5 个账户。

当选择按百分比时，如果指定的百分比不表示您指定的账户的整数数量，则 CloudFormation 会向下舍入。例如，如果您将堆栈部署到 10 个目标账户，并将最大并发账户数设置为 25 和按百分比，则 CloudFormation 会从同时部署 2.5 个堆栈（这是不可能的）向下舍入到同时部署 2 个堆栈。

请注意，此设置可让您指定操作数的最大值。对于大型部署，在某些情况下，并行运行的实际账户的数量可能由于服务限制而减少。

最大并发账户数可能取决于容错值，而后者又取决于您的并发模式。如果您将并发模式设置为严格容错，则最大并发账户数最多可以比容错设置大 1。

并发模式

此设置可在创建、更新和删除工作流时使用，让您能够选择并发级别在 StackSet 操作期间的行为。有关更多信息，请参阅 为 Amazon CloudFormation StackSet 选择并发模式。

容错

利用此设置（适用于创建、更新和删除工作流时），您可以指定每个区域中可发生的堆栈操作失败的最大数量或百分比，一旦超出此值，CloudFormation 就会自动停止操作。较小的数量或百分比意味着在较少的堆栈上执行该操作，但您能够更快地开始对失败的操作进行故障排除。例如，如果您将在 3 个区域内的 10 个目标账户中更新 10 个堆栈，则将 Failure tolerance (容错) 设置为 20 和 By percentage (按百分比) 意味着一个区域中最多 2 个堆栈更新可以失败，此操作继续。如果同一区域中的第三个堆栈失败，则 CloudFormation 将停止此操作。如果堆栈在第一个区域中无法进行更新，则更新操作将在该区域中继续进行，然后移到下一个区域。如果两个堆栈无法在第二个区域中进行更新，则容错能力达到 20%；如果该区域中的第三个堆栈失败，则 CloudFormation 停止更新操作，并且不会转入后续区域。

当选择按百分比时，如果指定的百分比不表示每个区域内的堆栈的整数数量，则 CloudFormation 会向下舍入。例如，如果您将堆栈部署到 3 个区域中的 10 个目标账户，并且将容错设置为 25 和按百分比，则 CloudFormation 会从每个区域的 2.5 个堆栈的容错能力（这是不可能的）向下舍入到每个区域的 2 个堆栈的容错能力。

保留堆栈

利用此设置（适用于删除堆栈工作流），您可以使堆栈及其资源继续运行，甚至在从 StackSet 中删除堆栈后也是如此。保留堆栈时，Amazon CloudFormation 会将堆栈按原样保留在各自的账户和区域中。这会取消堆栈与 StackSet 的关联，但将保存堆栈及其资源。删除堆栈操作完成后，您可在创建堆栈的目标账户（而不是管理员账户）中管理 CloudFormation 中的保留堆栈。

区域并发

此设置可在创建、更新和删除工作流时使用，用于选择如何将 StackSets 部署到区域中。

顺序 - 只要区域的部署失败次数不超过指定的容错能力，就按照区域 Deployment order（部署顺序）框中指定的顺序，一次将 StackSets 操作部署到一个区域。顺序部署是默认选项。

并行 – 只要区域的部署失败次数不超过指定的容错值，就将 StackSets 操作同时部署到所有指定区域。

标签

您可以通过指定键/值对在 StackSet 创建和更新操作期间添加标签。标签对于针对账单和成本分配来分类和筛选 StackSet 资源非常有用。有关如何在 Amazon 中使用标签的更多信息，请参阅《Amazon 账单与成本管理 User Guide》中的 Organizing and tracking costs using Amazon cost allocation tags。在指定键值对后，选择 + 以保存标签。您可以通过选择标签右侧的红色 X 来删除不再使用的标签。

您应用于 StackSet 的标签将应用于所有堆栈以及由您的堆栈创建的资源。您还可以在 CloudFormation 中的仅堆栈级别添加标签；但这些标签可能不会显示在 StackSets 中。

尽管 StackSets 目前未添加任何系统定义的标签，但您不应使用字符串 `aws:` 作为任何标签的键名称的开头。

StackSets 状态代码

Amazon CloudFormation StackSets 会为 StackSet 操作生成状态代码。

下表描述了 StackSet 操作的状态代码。

`RUNNING`

操作目前正在进行中。

`SUCCEEDED`

操作已完成，未超出操作的容错能力。

`FAILED`

无法完成操作的堆栈的数量超出了用户定义的容错能力。您为操作设置的容错能力值将在堆栈创建和更新操作期间应用于每个区域。如果区域内的失败堆栈数超出了容错能力，则该区域中的操作的状态将更改为 `FAILED`。整个操作的状态也将设置为 `FAILED`，并且 CloudFormation 会在任何剩余区域中取消该操作。

`QUEUED`

[`Service-managed permissions`] 对于需要一系列操作的自动部署，操作将排队等待执行。例如：

- 将账户从一个组织部门 (OU) `OU1` 移动到另一个组织部门 `OU2` 会触发自动部署。StackSets 运行删除操作，从目标区域中的目标 `OU1` 账户删除堆栈实例，并排队一个创建操作，用于将堆栈实例添加到目标区域中的目标 `OU2` 账户。
- 将账户 `AccountA` 添加到 OU 会触发自动部署。StackSets 运行创建操作，将堆栈实例添加到目标区域中的 `AccountA`。如果在此创建操作运行时，您将另一个账户 `AccountB` 添加到 OU

中，StackSets 将排队第二个创建操作。第一个创建操作完成后，StackSets 运行第二个创建操作，将堆栈实例添加到目标区域中的 AccountB。

`STOPPING`

根据用户请求，正在停止该操作。

`STOPPED`

根据用户请求，已停止该操作。

堆栈实例状态代码

Amazon CloudFormation StackSets 为堆栈实例生成状态代码。

下表描述了 StackSets 内堆栈实例的状态代码。

`CURRENT`

堆栈与 StackSet 的最新状态保持一致。

`OUTDATED`

如果存在下列原因之一，堆栈无法与 StackSet 的最新状态保持一致。

- 关联堆栈上的 CreateStackSet 或 UpdateStackSet 操作失败。
- 堆栈是失败的 CreateStackSet 或 UpdateStackSet 操作的一部分，或在创建或更新堆栈前已停止。

`INOPERABLE`

DeleteStackInstances 操作已失败，并且堆栈处于不稳定状态。将从进一步的 UpdateStackSet 操作中排除处于此状态的堆栈。您可能需要执行 DeleteStackInstances 操作，同时将 `RetainStacks` 设置为 `true`，以删除堆栈实例，然后手动删除堆栈。

`CANCELLED`

指定账户和区域中的操作已被取消。发生此情况可能是因为用户已停止 StackSet 操作，或者因为已超出 StackSet 操作的容错能力。

`FAILED`

指定账户和区域中的操作失败。如果 StackSet 操作在一个区域内足够数量的账户中失败，则可能会超出整个 StackSet 操作的容错能力。

FAILED_IMPORT

在指定账户和区域中导入堆栈实例失败，并且堆栈处于不稳定状态。修复导致失败的问题后，可以重试导入操作。如果在一个区域中，足够数量的 StackSet 操作在足够数量的账户中失败，则可能是超出了整个 StackSet 操作的容错能力。

PENDING

指定账户和区域中的操作尚未开始。

RUNNING

指定账户和区域中的操作当前正在进行中。

SKIPPED_SUSPENDED_ACCOUNT

指定账户和区域中的操作已被跳过，因为操作时该账户已被暂停。

SUCCEEDED

指定账户和区域中的操作已成功完成。

使用 Amazon CloudFormation StackSet 的先决条件

StackSets 扩展了堆栈的功能，因此您可以在一次操作中跨多个账户和区域创建、更新或删除堆栈。

由于 StackSet 会跨多个账户执行堆栈操作，您需要具有在 Amazon Web Services 账户中定义的必要权限，然后才能创建第一个 StackSet。

您可以使用自行管理权限或服务托管权限管理 StackSets。

- 对于自行管理的 StackSets，必须在每个目标账户和 Amazon Web Services 区域中创建和管理 IAM 角色。有关更多信息，请参阅 授予自行管理的权限。
- 对于服务托管的 StackSets，则无需在每个账户中手动创建和管理 IAM 角色，Amazon 为您处理角色的创建和权限。有关更多信息，请参阅 激活可信访问权限。

主题

- 准备在默认禁用的 Amazon Web Services 区域执行 StackSet 操作
- 授予自行管理的权限
- 使用 Amazon Organizations 为 StackSet 激活可信访问权限

准备在默认禁用的 Amazon Web Services 区域执行 StackSet 操作

2019 年 3 月 20 日之后推出的 Amazon Web Services 区域 默认处于禁用状态，如亚太地区（香港）。您必须先为您的账户启用这些区域，然后才能使用它们。有关更多信息，请参阅《Amazon 账户管理 参考指南》中的启用或禁用账户中的 Amazon Web Services 区域。

要在默认为已被禁用的区域中在 StackSet 的管理员账户（如果使用自行管理的权限）或组织的管理账户（如果使用服务托管权限）创建 StackSet，必须首先为管理员账户或管理账户启用该区域。

要使 Amazon CloudFormation 成功创建或更新堆栈实例：

- 目标账户必须位于当前已为该目标账户启用的区域中。
- StackSet 的管理员账户或组织的管理账户必须已经启用了与目标账户相同的区域。

> **Important**
>
> 请注意，在 StackSet 操作期间，除了所涉及的 StackSet 和 StackSet 实例外，管理员账户和目标账户还会交换有关账户本身的元数据。
> 此外，如果您禁用的区域包含 StackSet 实例所在的账户，您需要自行删除任何此类实例或资源（根据需要）。此外，请注意，关于禁用区域中目标账户的元数据将保留在管理员账户中。

授予自行管理的权限

本主题提供有关如何创建 StackSets 所需的 IAM 服务角色，以通过自行管理权限跨账户和 Amazon Web Services 区域 进行部署的说明。对于在您管理 StackSet 所用的账户与您将堆栈实例部署到的账户之间建立信任关系，这些角色必不可少。使用此权限模型，StackSets 可以部署到您有权创建 IAM 角色的任何 Amazon Web Services 账户 中。

要使用服务托管权限，请参阅 激活可信访问权限。

主题

- 自行管理权限概述
- 授予管理员账户的所有用户管理所有目标账户中堆栈的权限
- 为 StackSet 操作设置高级权限选项
- 设置全局键以缓解混淆代理问题

自行管理权限概述

必须先在各个账户中创建 IAM 服务角色，然后才能创建具有自行管理权限的 StackSet。

基本步骤如下：

1. 确定哪个 Amazon Web Services 账户是管理员账户。

   StackSet 是在该管理员账户中创建的。目标账户是在其中创建属于 StackSet 的各个堆栈的账户。
2. 确定您要如何为 StackSet 构建权限。

   利用最简单的（也是最宽松的）权限配置，您可以允许管理员账户中的所有用户和组创建和更新通过该账户管理的所有 StackSet。如果您需要更精细的控制，则可设置权限以指定：
   - 哪些用户和组可在哪些目标账户中执行 StackSet 操作。
   - 用户和组可将哪些资源包含在 StackSet 中。
   - 特定用户和组可执行哪些 StackSet 操作。
3. 在管理员账户和目标账户中创建所需的 IAM 服务角色以定义所需的权限。

   具体而言，所需的两个角色是：
   - AWSCloudFormationStackSetAdministrationRole – 该角色将部署到管理员账户。
   - AWSCloudFormationStackSetExecutionRole – 该角色将部署到创建堆栈实例的所有账户。

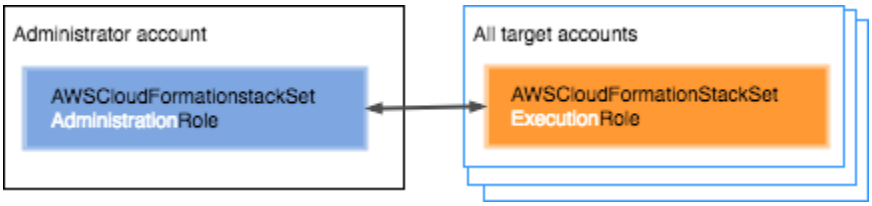
授予管理员账户的所有用户管理所有目标账户中堆栈的权限

本节将介绍如何设置权限，以便管理员账户的所有用户和组在所有目标账户中执行 StackSet 操作。本节会指导您在管理员和目标账户中创建所需的 IAM 服务角色。而后，管理员账户中的任何人都可以创建、更新或删除任何目标账户中的任何堆栈。

如果以这种方式构建权限，用户在创建或更新 StackSet 时便不会传递管理员角色。

> **Important**
>
> 即使您未指定 `AdministrationRoleARN` 参数，调用 `CreateStackSet` 的 IAM 主体或 `UpdateStackSet` 也必须拥有 AWSCloudFormationStackSetAdministrationRole 角色的 `iam:PassRole` 权限。CloudFormation 需要此权限，才能代表您使用默认管理角色。
> 以下是授予所需权限的示例策略：
>
> ```
> {
> ```

```
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::account-id:role/
AWSCloudFormationStackSetAdministrationRole"
}
```



Administrator account

在管理员账户中，创建一个名为 AWSCloudFormationStackSetAdministrationRole 的 IAM 角色。

为此，您可以通过 CloudFormation 模板创建堆栈，网址为 https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AWSCloudFormationStackSetAdministrationRole.yml。

Example权限策略示例

前述模板创建的管理员角色包括以下权限策略：

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWSCloudFormationStackSetExecutionRole"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Example示例信任策略 1

前述模板还包括以下信任策略，该策略授予使用管理员角色的服务权限以及附加到该角色的权限。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Example示例信任策略 2

要将堆栈实例部署到位于默认情况下禁用的区域中的目标账户，还需要包含该区域的区域服务主体。默认情况下禁用的每个区域都有自己的区域服务主体。

以下示例信任策略授予在亚太地区（香港）区域 (ap-east-1) 使用管理员角色的服务权限，该区域在默认情况下禁用。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
            "cloudformation.amazonaws.com",
            "cloudformation.ap-east-1.amazonaws.com"
         ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

有关更多信息，请参阅 准备在默认禁用的 Amazon Web Services 区域执行 StackSet 操作。有关区域代码的列表，请参阅《Amazon Web Services 一般参考 Guide》中的 Regional endpoints。

Target accounts

在每个目标账户中，创建一个信任管理员账户的名为 AWSCloudFormationStackSetExecutionRole 的服务角色。该角色必须具有该确切名称。为此，您可以通过 CloudFormation 模板创建堆栈，网址为 https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AWSCloudFormationStackSetExecutionRole.yml。当您使用此模板时，系统将提示您提供目标账户必须与其具有信任关系的管理员账户的账户 ID。

> **Important**
>
> 请注意，此模板将向管理员授予访问权限。在您使用此模板创建目标账户执行角色之后，您必须将策略声明中的权限的范围限定为您正使用堆栈集创建的资源的类型。

目标账户服务角色需要权限才能执行 CloudFormation 模板中指定的任何操作。例如，如果您的模板正在创建 S3 存储桶，则您需要权限才能为 S3 创建新对象。目标账户始终需要完整的 CloudFormation 权限，其中包含创建、更新、删除和描述堆栈的权限。

Example权限策略示例 1

此模板创建的角色将在目标账户上启用以下策略。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

Example权限策略示例 2

以下示例显示了一个包含让堆栈集运行所需的最低 权限的策略声明。要在使用 CloudFormation 以外的服务资源的目标账户中创建堆栈，必须将这些服务操作和资源添加到每个目标账户的 AWSCloudFormationStackSetExecutionRole 策略声明中。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Example示例信任策略

以下信任关系是由此模板创建的。管理员账户的 ID 显示为 *admin_account_id*。

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

您可以配置现有目标账户执行角色的信任关系以信任管理员账户中的特定角色。如果您在管理员账户中删除此角色并创建一个新角色来替代它，则您必须使用新的管理员账户角色 (由上一示例中的 *admin_account_id* 表示) 配置您的目标账户信任关系。

为 StackSet 操作设置高级权限选项

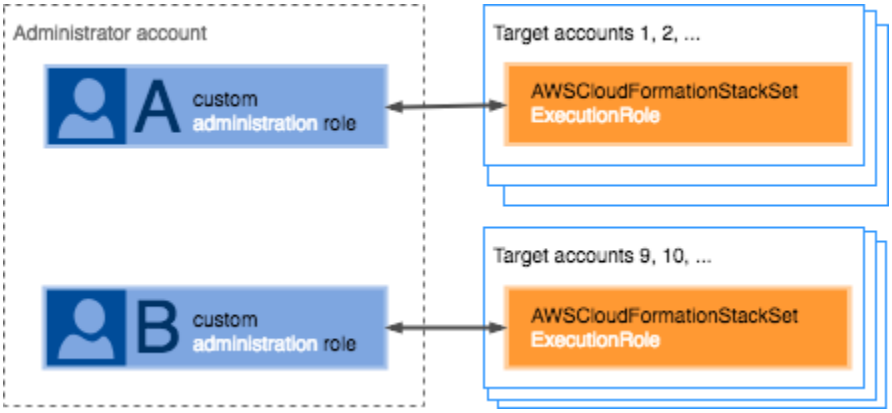
如果您需要更精细地控制用户和组通过单个管理员账户创建的 StackSet，您可以使用 IAM 角色指定：

- 哪些用户和组可在哪些目标账户中执行 StackSet 操作。
- 用户和组可将哪些资源包含在 StackSet 中。
- 特定用户和组可执行哪些 StackSet 操作。

控制哪些用户可以在特定目标账户中执行 StackSet 操作

使用自定义管理员角色，控制哪些用户和组可以在哪些目标账户中执行 StackSet 操作。您可能希望控制管理员账户的哪些用户可以在哪些目标账户中执行 StackSet 操作。为此，您可以在每个目标账户和特定的自定义管理角色之间创建信任关系，而不是在管理员账户中创建 AWSCloudFormationStackSetAdministrationRole 服务角色。然后，激活特定用户和组以在特定目标账户中执行 StackSet 操作时使用自定义管理角色。

例如，您可以在管理员账户中创建角色 A 和角色 B。您可以授予角色 A 访问目标账户 1 至账户 8 的权限。您可以授予角色 B 访问目标账户 9 至账户 16 的权限。



设置必要的权限包括定义自定义管理员角色、为目标账户创建服务角色并授予用户在执行 StackSet 操作时传递自定义管理员角色的权限。

一般而言，下面是具有必要权限后的工作原理：在创建 StackSet 时，用户必须指定自定义管理员角色。用户必须具有将角色传递到 CloudFormation 的权限。此外，自定义管理员角色还必须与为 StackSet 指定的目标账户具有信任关系。CloudFormation 会创建 StackSet 并将自定义管理员角色与其关联。在更新 StackSet 时，用户必须明确指定自定义管理员角色，即使它与之前用于此 StackSet 的自定义管理员角色相同，也是如此。CloudFormation 根据上述要求使用该角色来更新堆栈。

Administrator account

Example权限策略示例

对于每个 StackSet，创建一个自定义管理员角色，该角色具有担任目标账户执行角色的权限。

每个目标账户中的目标账户执行角色名称必须相同。如果角色名称为 AWSCloudFormationStackSetExecutionRole，StackSets 会在创建 StackSet 时会自动使用该名称。如果指定自定义角色名称，则用户在创建 StackSet 时必须提供执行角色名称。

使用以下权限策略，创建具有自定义名称的 IAM 服务角色：在以下示例中，*custom_execution_role* 是指目标账户中的执行角色。

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/custom_execution_role"
            ],
            "Effect": "Allow"
        }
    ]
}
```

要在单个语句中指定多个账户，请使用逗号将其隔开。

```
"Resource": [
  "arn:aws:iam::target_account_id_1:role/custom_execution_role",
  "arn:aws:iam::target_account_id_2:role/custom_execution_role"
]
```

您可以通过使用通配符 (*) 而不是账户 ID，指定所有目标账户。

```
"Resource": [
  "arn:aws:iam::*:role/custom_execution_role"
]
```

Example示例信任策略 1

您必须为服务角色提供信任策略，从而定义哪些 IAM 主体可以担任该角色。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Example示例信任策略 2

要将堆栈实例部署到位于默认情况下禁用的区域中的目标账户，还需要包含该区域的区域服务主体。默认情况下禁用的每个区域都有自己的区域服务主体。

以下示例信任策略授予在亚太地区（香港）区域 (ap-east-1) 使用管理员角色的服务权限，该区域在默认情况下禁用。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
            "cloudformation.amazonaws.com",
            "cloudformation.ap-east-1.amazonaws.com"
         ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

有关更多信息，请参阅 准备在默认禁用的 Amazon Web Services 区域执行 StackSet 操作。有关区域代码列表，请参阅《Amazon General Reference Guide》中的 Regional endpoints。

Example示例传递角色策略

您还需要为 IAM 用户提供 IAM 权限策略，允许该用户在执行 StackSet 操作时传递自定义管理员角色。有关更多信息，请参阅向用户授予将角色传递给 Amazon 服务的权限。

在以下示例中，*customized_admin_role* 是指用户需要传递的管理员角色。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/customized_admin_role"
    }
  ]
}
```

Target accounts

在每个目标账户中，创建一个服务角色，该角色信任要用于此账户的自定义管理员角色。

目标账户角色需要权限才能执行 CloudFormation 模板中指定的任何操作。例如，如果您的模板正在创建 S3 存储桶，则您需要在 S3 中创建新对象的权限。目标账户始终需要完整的 CloudFormation 权限，其中包含创建、更新、删除和描述堆栈的权限。

每个目标账户中的目标账户角色名称必须相同。如果角色名称为 AWSCloudFormationStackSetExecutionRole，StackSets 会在创建 StackSet 时会自动使用该名称。如果指定自定义角色名称，则用户在创建 StackSet 时必须提供执行角色名称。

Example权限策略示例

以下示例显示了一个包含让堆栈集运行所需的最低 权限的策略声明。要在使用非 CloudFormation 服务资源的目标账户中创建堆栈，必须将这些服务操作和资源添加到权限策略。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Example示例信任策略

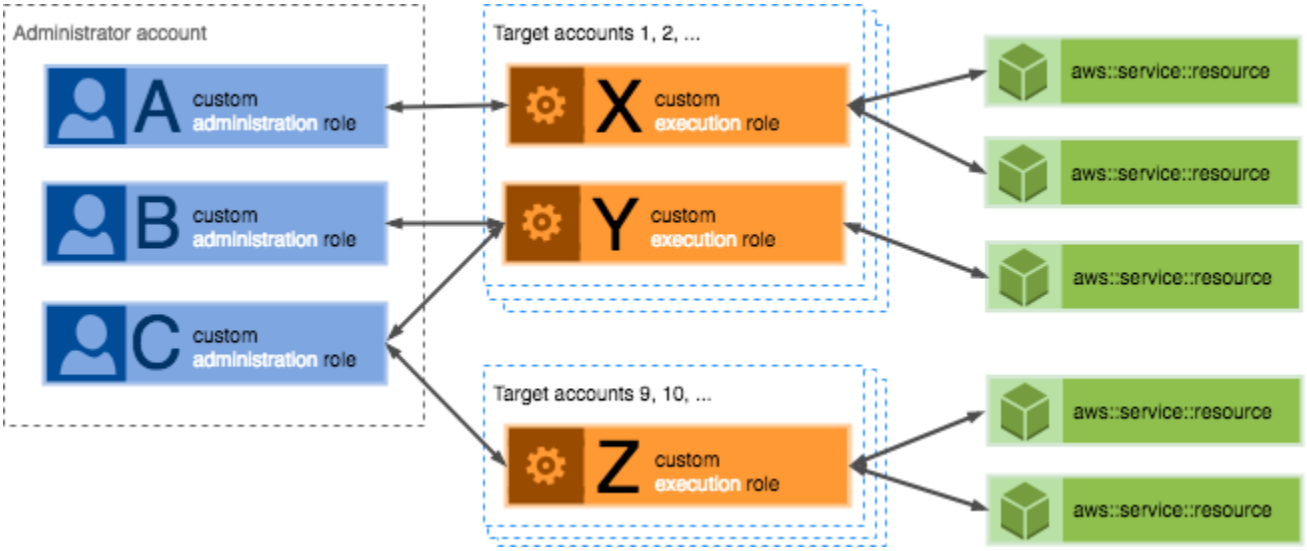
在创建角色以定义信任关系时，必须提供以下信任策略：

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/customized_admin_role"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

控制用户可以在特定 StackSet 中包含哪些资源

使用自定义执行角色可以控制用户和组可将哪些堆栈资源包含在 StackSet 中。例如，您可能希望设置一个组，该组只能将与 Amazon S3 相关的资源包含在他们创建的 StackSet 中，而另一个团队只能包含 DynamoDB 资源。为此，您可以在每个组的自定义管理员角色与每组资源的自定义执行角色之间建立信任关系。自定义执行角色定义可将哪些堆栈资源包含在 StackSet 中。自定义管理员角色位于管理员账户中，而自定义执行角色位于每个目标账户（要在其中使用定义的资源创建 StackSet 的账户）中。然后，激活特定用户和组以在执行 StackSet 操作时使用自定义管理角色。

例如，您可以在管理员账户中创建自定义管理员角色 A、B 和 C。有权使用角色 A 的用户和组可以创建相应的 StackSet，这些 StackSet 包含自定义执行角色 X 中专门列出的堆栈资源，而不包含角色 Y 或 Z 中的堆栈资源或任何执行角色中未包含的资源。



在更新 StackSet 时，用户必须明确指定自定义管理员角色，即使它与之前用于此 StackSet 的自定义管理员角色相同，也是如此。只要用户具有对该 StackSet 执行操作的权限，CloudFormation 就会使用指定的自定义管理员角色执行更新。

同样，用户还可以指定自定义的执行角色。如果用户指定了自定义执行角色，CloudFormation 将根据上述要求使用该角色来更新堆栈。如果用户未指定自定义执行角色，CloudFormation 将使用以前与 StackSet 关联的自定义执行角色来执行更新，只要用户具有对该 StackSet 执行操作的权限。

Administrator account

在管理员账户中创建自定义管理员角色，如 控制哪些用户可以在特定目标账户中执行 StackSet 操作 中所述。包含自定义管理员角色和希望该角色使用的自定义执行角色之间的信任关系。

Example权限策略示例

以下示例是针对为目标账户定义的 AWSCloudFormationStackSetExecutionRole 以及自定义执行角色的权限策略。

```
{
  "Version":"2012-10-17",
   "Statement": [
    {
      "Sid": "Stmt1487980684000",
```

```
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWSCloudFormationStackSetExecutionRole",
        "arn:aws:iam::*:role/custom_execution_role"
      ]
    }
  ]
}
```

Target accounts

在要在其中创建 StackSet 的目标账户中，创建一个自定义的执行角色，该角色授予对您要用户和组能够将其包含在 StackSet 中的服务和资源的权限。

Example权限策略示例

以下示例提供了 StackSet 的最小权限以及创建 Amazon DynamoDB 表的权限。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamoDb:createTable"
      ],
      "Resource": "*"
    }
  ]
}
```

Example示例信任策略

在创建角色以定义信任关系时，必须提供以下信任策略：

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/customized_admin_role"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

为特定 StackSet 操作设置权限

此外，您可以为能够执行特定的 StackSet 操作（例如，创建、更新或删除 StackSet 或堆栈实例）的用户和组设置权限。有关更多信息，请参阅《服务授权参考》中的 CloudFormation 的操作、资源和条件键。

设置全局键以缓解混淆代理问题

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务) 调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，Amazon 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

建议在资源策略中使用 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键，以限制 Amazon CloudFormation StackSets 为其他服务提供的资源访问权限。如果使用两个全局条件上下文键，在同一策略语句中使用时，`aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户必须使用相同的账户 ID。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符（`*`）的 `aws:SourceArn` 全局上下文条件键。例如 `arn:aws:cloudformation::123456789012:*`。

请尽可能使用 aws:SourceArn，因为它更具体。仅当无法确定正确的 ARN 或 ARN 模式时使用 aws:SourceAccount。

当 StackSets 在管理员账户中担任管理角色时，StackSets 会填充管理员账户 ID 和 StackSets Amazon 资源名称（ARN）。因此，您可以在信任关系中为全局键 aws:SourceAccount 和 aws:SourceArn 定义条件，以防止混淆代理问题。以下示例演示如何使用 StackSets 中的 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键来防范混淆代理问题。

Administrator account

Example **aws:SourceAccount** 和 **aws:SourceArn** 的全局键示例

使用 StackSets 时，请在 AWSCloudFormationStackSetAdministrationRole 信任策略中定义全局键 aws:SourceAccount 和 aws:SourceArn 以防出现混淆代理问题。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:cloudformation:*:111122223333:stackset/*"
        }
      }
    }
  ]
}
```

Example StackSets ARN

指定关联的 StackSets ARN 以进行更精细的控制。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": [
                        "arn:aws:cloudformation:STACKSETS-
REGION:111122223333:stackset/STACK-SET-ID-1",
                        "arn:aws:cloudformation:STACKSETS-
REGION:111122223333:stackset/STACK-SET-ID-2"
                    ]
                }
            }
        }
    ]
}
```

使用 Amazon Organizations 为 StackSet 激活可信访问权限

本主题提供有关如何使用 Amazon Organizations 激活 StackSets 所需的可信访问，以通过服务托管权限跨账户和 Amazon Web Services 区域 进行部署的说明。要使用自行管理权限，请参阅 授予自行管理的权限。

在创建具有服务托管权限的 StackSet 之前，您必须先完成以下任务：

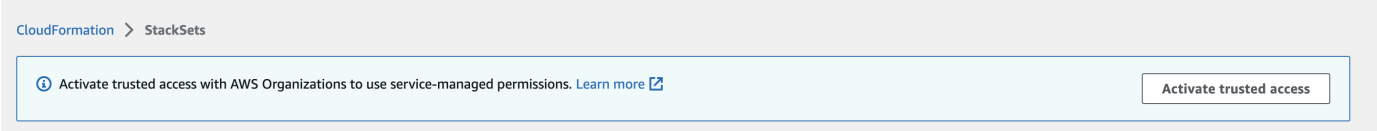
- 在 Amazon Organizations 中启用所有功能。仅启用了整合账单功能时，您无法创建具有服务托管权限的 StackSet。
- 激活 Amazon Organizations 的可信访问权限。此操作允许 CloudFormation 在管理帐户中创建服务关联角色。激活可信访问权限后，当您创建具有服务管理权限的 StackSet 时，CloudFormation 会在目标（成员）账户中创建必要的服务关联角色和名为 `stacksets-exec-*` 的服务角色。

激活可信访问权限后，管理账户和委派管理员账户可以为组织创建和管理服务托管 StackSet。

要激活可信访问权限，您必须是管理账户的管理员用户。管理员用户 是对您的 Amazon Web Services 账户拥有完全权限的用户。有关更多信息，请参阅《Amazon 账户管理 参考指南》中的创建管理员用户。有关保护管理账户安全的建议，请参阅《Amazon Organizations 用户指南》中的管理账户最佳实践。

激活可信访问权限

1. 以管理账户的管理员身份登录 Amazon，并通过 https://console.amazonaws.cn/cloudformation 打开 CloudFormation 控制台。
2. 从导航窗格中，选择 StackSets。如果停用可信访问权限，则会显示一个横幅，提示您激活可信访问权限。



3. 选择激活可信访问权限。

显示以下横幅即表明可信访问权限已成功激活。



> **Note**
>
> “激活组织访问权限”与“启用组织访问权限”相同，“停用组织访问权限”与“禁用组织访问权限”相同。这些术语已根据营销指南进行更新。

停用可信访问权限

请参阅《Amazon Organizations 用户指南》 中的 Amazon CloudFormation StackSets 和 Amazon Organizations。

必须先注销所有委派管理员，然后才能停用 Amazon Organizations 的可信访问权限。有关更多信息，请参阅 注册委派管理员。

> **Note**
>
> 有关如何使用 API 操作而不是控制台来激活或停用可信访问权限的信息，请参阅：
>
> - ActivateOrganizationsAccess
> - DeactivateOrganizationsAccess
> - DescribeOrganizationsAccess

服务关联角色

管理账户使用 AWSServiceRoleForCloudFormationStackSetsOrgAdmin 服务相关角色。只有在停用了 Amazon Organizations 的可信访问权限时，您才能修改或删除此角色。

各目标账户使用 awsServiceRoleformationsStacksetsorgMember 服务相关角色。只有在满足以下两个条件时，您才能修改或删除此角色：停用了 Amazon Organizations 的可信访问权限时，或者从目标组织或组织部门（OU）中删除了账户。

有关更多信息，请参阅《Amazon Organizations 用户指南》中的 Amazon CloudFormation StackSets 和 Amazon Organizations。

注册委派管理员成员账户

除了组织的管理账户，具有委派管理员权限的成员账户也可以为组织创建和管理具有服务托管权限的 StackSet。具有服务托管权限的 StackSet 是在管理账户中创建的，包括由委派管理员创建的 StackSet。要注册为组织的委派管理员，您的成员账户必须在组织中。有关加入组织的更多信息，请参阅邀请 Amazon Web Services 账户加入组织。

您的组织一次最多可以有五个注册的委派管理员。委派管理员可以选择部署到组织中的所有账户或部署到特定 OU。必须先激活 Amazon Organizations 的可信访问权限，然后才能将委派管理员部署到 Organizations 托管的账户。有关更多信息，请参阅 使用 Amazon Organizations 为 StackSet 激活可信访问权限。

> **Important**
>
> 请注意以下事项：
>
> - 委派管理员具有部署到组织中的账户的完全权限。管理账户不能限制委派管理员部署到特定 OU 或执行特定 StackSet 操作的权限。

- 确保您的委派管理员拥有 `organizations:ListDelegatedAdministrators` 权限，以避免任何潜在错误。

您可以在以下区域为组织注册委派管理员：美国东部（俄亥俄州）、美国东部（弗吉尼亚州北部）、美国西部（北加利福尼亚）、美国西部（俄勒冈州）、亚太地区（孟买）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）、亚太地区（东京）、加拿大（中部）、欧洲地区（法兰克福）、欧洲地区（爱尔兰）、欧洲地区（伦敦）、欧洲地区（巴黎）、欧洲地区（斯德哥尔摩）、以色列（特拉维夫）、南美洲（圣保罗）、Amazon GovCloud（美国东部）和 Amazon GovCloud（美国西部）。

Amazon CloudFormation 控制台、Amazon CLI 或 Amazon 开发工具包可用于注册和取消注册委派管理员。

要注册委派管理员（控制台），请执行以下操作：

1. 以管理账户的管理员身份登录 Amazon，并通过 https://console.amazonaws.cn/cloudformation/ 打开 Amazon CloudFormation 控制台。
2. 从导航窗格中，选择 StackSets。
3. 在 Delegated administrators（委派管理员）下，选择 Register delegated administrator（注册委派管理员）。
4. 在 Register delegated administrator（注册委派管理员）对话框中，选择 Register delegated administrator（注册委派管理员）。

   成功消息表示成员账户已成功注册为委派管理员。

要取消注册委派管理员（控制台），请执行以下操作：

1. 以管理账户的管理员身份登录 Amazon，并通过 https://console.amazonaws.cn/ 打开 Amazon CloudFormation 控制台。
2. 从导航窗格中，选择 StackSets。
3. 在 Delegated administrators（委派管理员）下，选择要取消注册的账户，然后选择 Deregister（取消注册）。

   成功消息表示成员账户的委派管理员身份已成功取消注册。

   您可以随时再次注册此账户。

注册委派管理员（Amazon CLI）

1. 打开 Amazon CLI。
2. 运行 `register-delegated-administrator` 命令。

```
$ aws organizations register-delegated-administrator \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com \
  --account-id="memberAccountId"
```

3. 运行 `list-delegated-administrators` 命令验证指定的成员账户是否已成功注册为委派管理员。

```
$ aws organizations list-delegated-administrators \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com
```

取消注册委派管理员（Amazon CLI）

1. 打开 Amazon CLI。
2. 运行 `deregister-delegated-administrator` 命令。

```
$ aws organizations deregister-delegated-administrator \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com \
  --account-id="memberAccountId"
```

3. 运行 `list-delegated-administrators` 命令验证指定的成员账户的委派管理员身份是否已成功取消注册。

```
$ aws organizations list-delegated-administrators \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com
```

   您可以随时再次注册此账户。

使用示例模板开始使用 StackSet。

本教程将介绍如何通过 Amazon Web Services 管理控制台开始使用 StackSet。它会指导您如何使用示例模板创建 StackSet。您将学会如何跨多个区域部署堆栈、监控 StackSet 操作以及查看结果。

在本教程中，您将在美国西部（俄勒冈州）(us-west-2) 区域和美国东部（弗吉尼亚州北部）区域 (us-east-1) 中的 Amazon Web Services 账户中创建一个启用 Amazon Config 的 StackSet。借助 StackSet，您可以在一次操作中跨多个账户和区域创建、更新或删除堆栈，因此它是大规模管理基础设施的理想解决方案。虽然为了方便起见，本教程使用单一账户，但它有效地演示了 StackSet 的多区域功能。

位于以下 S3 存储桶中的示例模板可供使用：https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSConfig.yml。

> **Note**
>
> StackSet 使用免费，但您需要为创建 StackSet 使用的 Amazon 资源付费，例如本教程中的 Amazon Config。有关更多信息，请参阅 Amazon Config 定价。

主题

- 先决条件
- 使用控制台中的示例模板创建 StackSet
- 监控 StackSet 创建
- 查看 StackSet 结果
- 更新 StackSet
- 将堆栈添加到 StackSet
- 清理
- 后续步骤

先决条件

开始本教程之前，请确保您已完成以下先决条件：

- 您必须已设置自行管理权限所需的 IAM 角色。要创建 StackSet 并在单个账户中部署堆栈，您的账户中需要有以下角色：
  - `AWSCloudFormationStackSetAdministrationRole`
  - `AWSCloudFormationStackSetExecutionRole`

  有关这些角色设置的详细说明，请参阅授予自行管理的权限。

使用控制台中的示例模板创建 StackSet

创建启用 Amazon Config 的 StackSet

1. 打开 CloudFormation 控制台。
2. 在屏幕顶部的导航栏中，选择要从中管理 StackSet 的 Amazon Web Services 区域。

   您可以选择支持 StackSet 的任何区域。您选择的区域不会影响您可以使用 StackSet 部署的区域范围。
3. 从导航窗格中，选择 StackSets。
4. 从 StackSets 页面顶部，选择创建 StackSet。
5. 在权限下，选择自助服务权限，然后选择您在完成先决条件期间创建的 IAM 角色。
   - 在 IAM 管理员角色中，选择 AWSCloudFormationStackSetAdministrationRole。
   - 在 IAM 执行角色名称中，选择 AWSCloudFormationStackSetExecutionRole。
6. 在 Prerequisite - Prepare template (先决条件 – 准备模板) 下，选择 Use a sample template (使用示例模板)。
7. 在选择示例模板下，选择启用 Amazon Config 模板。然后选择下一步。

   此模板会在账户中创建启用 Amazon Config 所必需的资源，包括配置记录器和交付渠道。
8. 在指定 StackSet 详细信息页面的 StackSet 名称中，输入 **my-awsconfig-stackset**。
9. 在 StackSet 描述中，输入 **A StackSet that enables Config across multiple Regions**。
10. 在参数下，按以下方式配置 Amazon Config 设置：
    a. 对于支持所有资源类型，请保留默认值 true，从而记录所有受支持的资源类型。
    b. 对于包含全局资源类型，保留默认值 false，从而排除 IAM 角色等全局资源。
    c. 请将如非全部支持时的资源类型列表设置为 <All>。
    d. 对于包含 Config 服务相关角色资源的区域，请将 <DeployToAnyRegion> 替换为 **us-west-2**。

       这表示仅在将堆栈部署到美国西部（俄勒冈州）区域时，才会创建名为 `AWSServiceRoleForConfig` 的服务相关角色。您将在此过程的后期阶段选择部署区域。
    e. 对于配置记录器记录频率，请选择每日记录。
11. 选择下一步以继续。

12. 在配置 StackSet 选项页面上，选择添加新标签，然后指定键/值对来添加标签。

    a. 对于键，输入 **`Stage`**。

    b. 对于值，输入 **`Test`**。

    您应用到 StackSet 的标签也将应用到堆栈创建的所有资源。

13. 在执行配置中，请选择活动，启用 CloudFormation 的优化操作处理：

    - 为了缩短部署时间，无冲突的操作会同时运行。
    - 存在冲突的操作会自动排入队列，按照请求的顺序进行处理。

    如果有操作正在运行或排队，CloudFormation 会将所有传入操作排入队列，即使这些操作没有冲突。在此期间，您不能更改执行设置。

14. 选择下一步。

15. 在设置部署选项页面的向 StackSet 添加堆栈中，选择部署新堆栈。

16. 对于 Accounts (账户)，选择 Deploy stacks in accounts (在账户中部署堆栈)。

17. 在文本框中输入 Amazon Web Services 账户 ID。

18. 在指定区域中，按此顺序选择以下区域：

    1. 美国西部（俄勒冈州）区域 (`us-west-2`)
    2. 美国东部（弗吉尼亚州北部）区域 (`us-east-1`)

    如需要，选择美国西部（俄勒冈州）区域旁边的向上箭头，将其移至列表中的第一个条目。区域顺序决定了其部署顺序。

19. 在部署选项中，配置以下设置：

    a. 对于 Maximum concurrent accounts（最大并发账户数），请保留 Number（数量）和 1 的默认值。

       如果是多账户部署，此设置表示，CloudFormation 一次仅在一个账户中部署您的堆栈。

    b. 对于 Failure tolerance (容错)，请保留 Number (数量) 和 0 的默认值。

       这意味着，只要您指定区域之一中只要有一个堆栈部署失败，CloudFormation 就会停止当前区域中的部署并取消剩余区域中的部署。

    c. 在区域并发中，请选择顺序（默认设置）。

此设置可确保 CloudFormation 完成一个区域的部署后，然后再移至下一个区域。

d. 在并发模式中，保留默认值严格容错。

对于多账户部署，这会在操作失败时降低账户并发级别，保持在容错能力 +1 的范围内。

20. 选择下一步。
21. 在审核页面上，审核您的选择。要进行更改，请在相关部分选择编辑。
22. 准备好创建 StackSet 后，选择提交。

监控 StackSet 创建

选择提交后，CloudFormation 会开始创建您的 StackSet 并将堆栈部署到您账户中的指定区域。StackSet 详细信息页面会自动打开，您可以在其中监控操作进度。

监控 StackSet 创建

1. 在 StackSet 详细信息页面上会默认显示操作选项卡，显示当前正在进行的操作。
2. 操作状态最开始应为 RUNNING。CloudFormation 根据您配置的部署选项在指定的区域中创建堆栈。
3. 要查看有关操作的更多详细信息，请在列表中选择操作 ID。
4. 在操作详细信息页面，可以查看每个区域中创建的堆栈实例的状态。
5. 操作状态变为 SUCCEEDED 后表示 StackSet 及其所有堆栈实例已成功创建。

查看 StackSet 结果

StackSet 创建完成后，您可以查看已部署的堆栈实例，验证您的账户是否已在各个指定区域中启用 Amazon Config。

查看 StackSet 结果

1. 在 StackSet 详细信息页面上，选择堆栈实例选项卡。
2. 您应该会看到您的账户在各个指定区域中创建的堆栈实例列表。每个堆栈实例的状态都应为 SUCCEEDED，此状态表示已成功部署。
3. 要验证您的账户是否已启用 Amazon Config，可以查看每个部署区域的 Amazon Config 控制台。

更新 StackSet

创建 StackSet 后，您可能会需要对其进行更新以修改参数值或添加更多区域。本节介绍如何更新 Amazon Config 记录频率参数。

更新 StackSet

1. 在 StackSet 页面上，选择 **my-awsconfig-stackset**。
2. 选定 StackSet 之后，从操作菜单中选择编辑 StackSet 详细信息。
3. 在选择模板页面的先决条件 – 准备模板中，选择使用当前模板。
4. 选择下一步。
5. 在指定 StackSet 详细信息页面的参数下，找到配置记录器的记录频率，将其从每日更改为连续。
6. 选择下一步。
7. 在配置 StackSet 选项页面上，保持所有设置不变，然后选择下一步。
8. 在设置部署选项页面上，指定您的账户 ID 以及您在创建 StackSet 时使用的区域。
9. 在部署选项中，保持与之前相同的设置。
10. 选择下一步。
11. 在审核页面上，检查您的更改内容，然后选择提交。
12. CloudFormation 会开始更新 StackSet。您可以在 StackSet 详细信息页面的操作选项卡上监控进度。

将堆栈添加到 StackSet

您可以通过部署到其他区域来向 StackSet 添加更多堆栈。本节介绍如何向新区域添加堆栈。

将堆栈添加到 StackSet

1. 在 StackSet 页面上，选择 **my-awsconfig-stackset**。
2. 选择 StackSet 后，从操作菜单中选择将堆栈添加到 StackSet。
3. 在设置部署选项页面的向 StackSet 添加堆栈中，选择部署新堆栈。
4. 在账户中，选择在账户中部署堆栈，然后输入您的账户 ID。
5. 在指定区域中，选择一个新的区域，例如欧洲地区（爱尔兰）(eu-west-1)。
6. 在部署选项中，保持与之前相同的设置。
7. 选择下一步。

8. 在指定覆盖页面上，将属性值保留为指定值，然后选择下一步。
9. 在审核页面上，检查您的选择，然后选择提交。
10. CloudFormation 会开始在指定区域中创建新堆栈。您可以在 StackSet 详细信息页面的操作选项卡上监控进度。

清理

为避免因不需要的 Amazon Config 资源产生费用，您应该从 StackSet 中删除堆栈、删除 StackSet 本身以及删除在本教程学习期间创建的 IAM 角色，以便进行清理。所有资源都部署在您的账户中，因此清理非常简单。

从 StackSet 中删除堆栈

1. 在 StackSet 页面上，选择 **my-awsconfig-stackset**。
2. 选定 StackSet 后，从操作菜单中选择从 StackSet 中删除堆栈。
3. 在设置部署选项页面的账户中，选择在账户中部署堆栈，然后输入您的账户 ID。
4. 在指定区域中，选择部署了堆栈的所有区域。
5. 在部署选项中，保留默认设置。
6. 确保未打开保留堆栈，以便删除堆栈及其资源。
7. 选择下一步。
8. 在审核页面上，检查您的选择，然后选择提交。
9. CloudFormation 会开始从 StackSet 中删除堆栈。您可以在 StackSet 详细信息页面的操作选项卡上监控进度。

删除 StackSet

1. 删除所有堆栈后，在 StackSet 页面上，选择 **my-awsconfig-stackset**。
2. 选定 StackSet 后，从操作菜单中选择删除 StackSet。
3. 当系统提示进行确认时，选择删除。

删除 IAM 服务角色

由于您只部署到自己的账户，因此您只需要从该单一账户中删除 IAM 角色即可，清理工作比多账户部署简单得多。

1. 打开 IAM 控制台。
2. 从导航窗格中，选择 Roles。
3. 在搜索框中，输入 **AWSCloudFormationStackSet** 查找您在本教程学习期间创建的角色。
4. 勾选 AWSCloudFormationStackSetAdministrationRole 旁边的复选框。
5. 在该页面的顶部，选择删除。
6. 在确认对话框中，输入 **delete**，然后选择删除。
7. 重复同样的过程删除 AWSCloudFormationStackSetExecutionRole。

删除 StackSet 后，由于 `AWS::S3::Bucket` 资源的 `DeletionPolicy` 属性，每个 Amazon Web Services 区域都将保留一个 Amazon S3 存储桶。这样可以保留您的 Amazon Config 历史数据。如果您不再需要这些数据，可以安全地手动删除该存储桶。您必须先清空存储桶，然后才能删除存储桶。清空存储桶将删除存储桶中的所有对象。

清空和删除 Amazon S3 存储桶

1. 打开 Amazon S3 控制台。
2. 在控制台左侧的导航窗格中，选择 Buckets（存储桶）。
3. 在存储桶列表中，您将看到在部署的每个区域中为此 StackSet 创建的存储桶。选择为此 StackSet 创建的存储桶名称旁边的选项，然后选择清空。
4. 在清空存储桶页面上，通过在文本字段中键入 **permanently delete** 来确认要清空存储桶，然后选择清空。
5. 在清空存储桶：状态页面上监控存储桶清空过程的进度。
6. 要返回到存储桶列表，请选择退出。
7. 选择存储桶名称旁的选项，然后选择删除。
8. 在系统提示进行确认时，键入存储桶的名称，然后选择删除存储桶。
9. 在存储桶清单上监控存储桶删除过程的进度。Amazon S3 完成存储桶删除后，会将存储桶从列表中移除。
10. 重复此过程，删除不同区域中为此 StackSet 创建的存储桶。

后续步骤

恭喜您！您已经成功使用示例模板创建了 StackSet，将堆栈部署到账户内的多个区域，更新了 StackSet，添加了更多堆栈，还清理了资源。通过学习单一账户部署，可以简化您的清理过程，同时还能让您学到 StackSet 的核心多区域功能。

要了解有关 StackSet 的更多信息，请参阅以下主题：

- 覆盖 CloudFormation StackSet 中的堆栈参数值：了解如何覆盖特定账户和区域的参数值。
- 创建具有服务托管权限的 Amazon CloudFormation StackSet：探索如何使用 Amazon Organizations 为多账户部署创建 StackSet。

创建具有自行管理权限的 Amazon CloudFormation StackSet

本主题介绍如何创建具有自行管理权限的 StackSet，以便跨 Amazon Web Services 账户和区域部署堆栈。

> **Note**
>
> 在此之前，您必须首先创建 StackSet 所需的 IAM 角色，以便在 StackSet 的管理账户与需要部署堆栈的账户之间建立可信关系。有关更多信息，请参阅 授予自行管理的权限。

主题

- 创建具有自行管理权限的 StackSet（控制台）
- 创建具有自行管理权限的 StackSet（Amazon CLI）

创建具有自行管理权限的 StackSet（控制台）

创建 StackSet

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择要从中管理 StackSet 的 Amazon Web Services 区域。
3. 从导航窗格中，选择 StackSets。
4. 从 StackSets 页面顶部，选择创建 StackSet。
5. 在权限下，选择自助服务权限，然后选择您创建的 IAM 角色。
6. 在先决条件 - 准备模板下，选择模板已就绪。
7. 在 Specify template (指定模板) 下，选择指定包含堆栈模板的 S3 存储桶的 URL，或者上传堆栈模板文件。然后选择下一步。
8. 在指定 StackSet 详细信息页面上，提供 StackSet 名称，指定所有参数，然后选择下一步。

9. 选择下一步以继续。
10. 在 Configure StackSet options (配置堆栈集选项) 页面上，在 Tags (标签) 下指定要应用于堆栈中资源的所有标签。有关如何在 Amazon 中使用标签的更多信息，请参阅《Amazon 账单与成本管理 User Guide》中的 Organizing and tracking costs using Amazon cost allocation tags。
11. 在执行配置中，请选择活动，启用 CloudFormation 的优化操作处理：

    - 为了缩短部署时间，无冲突的操作会同时运行。
    - 存在冲突的操作会自动排入队列，按照请求的顺序进行处理。

    如果有操作正在运行或排队，CloudFormation 会将所有传入操作排入队列，即使这些操作没有冲突。在此期间，您不能更改执行设置。
12. 如果模板包含 IAM 资源，则对于功能，请选择我确认该模板可能会创建 IAM 资源以指定您要在模板中使用 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。
13. 选择下一步。
14. 在设置部署选项页面的向 StackSet 添加堆栈中，选择部署新堆栈。
15. 对于 Accounts (账户)，选择 Deploy stacks in accounts (在账户中部署堆栈)。在文本框中粘贴您的目标 Amazon Web Services 账户编号，并使用逗号分隔多个编号。

    > **Note**
    >
    > 如果您还想在该账户中部署堆栈，则可以将您的管理员账户 ID 也包含在其中。

16. 对于指定区域，选择要在其中部署堆栈的区域。
17. 对于部署选项，请执行以下操作：

    - 对于最大并发账户数，请指定并发处理的账户数量。
    - 在容错中，请指定每个区域允许的账户失败次数上限。达到此次数限制后，该操作将停止，不会继续到其他区域。
    - 对于区域并发，请选择处理区域的方式：顺序（一次处理一个区域）或并行（并发处理多个区域）。
    - 对于并发模式，请选择在操作执行期间的并发行为方式。
      - 严格容错：操作失败时降低账户并发级别，保持在容错 +1 范围内。
      - 软容错 – 即使失败，仍保持您指定的并发级别（最大并发账户数的值）。
18. 选择下一步。

19. 在审核页面上，审核您的选择。要进行更改，请在相关部分选择编辑。
20. 准备好创建 StackSet 后，选择提交。

    CloudFormation 会开始创建 StackSet。选择提交后会打开 StackSet 详细信息页面，可以在其中查看 StackSet 中的堆栈创建进度和状态。

创建具有自行管理权限的 StackSet（Amazon CLI）

请按照本部分中的步骤，使用 Amazon CLI 来执行以下操作：

- 创建 StackSet 容器。
- 部署堆栈实例。

创建 StackSet

1. 使用 create-stack-set 命令创建一个新 StackSet（名为 *my-stackset*）。以下示例中，使用存储在 S3 存储桶中的模板，包含将 *KeyPairName* 设置为 *TestKey* 的参数。

```
aws cloudformation create-stack-set \
  --stack-set-name my-stackset \
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/
MyApp.template \
  --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
```

2. 在 create-stack-set 命令完成后，运行 list-stack-sets 命令可查看您的 StackSet 是否已创建。您应在结果中看到新的 StackSet。

```
aws cloudformation list-stack-sets
```

3. 使用 create-stack-instances 命令将堆栈部署到 StackSet 中。以下示例中，在两个区域（*us-west-2* 和 *us-east-1*）的两个 Amazon Web Services 账户（*account_ID_1* 和 *account_ID_2*）中部署堆栈。

   使用 `--operation-preferences` 选项设置并发账户处理和其他部署首选项。此示例使用基于计数的设置。请注意，`MaxConcurrentCount` 不得大于 `FailureToleranceCount` + 1。对于基于百分比的设置，请改用 `FailureTolerancePercentage` 或 `MaxConcurrentPercentage`。

```
aws cloudformation create-stack-instances \
```

```
  --stack-set-name my-stackset \
  --accounts account_ID_1 account_ID_2 \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

有关更多信息，请参阅《Amazon CloudFormation API 参考》中的 CreateStackInstances。

4. 使用 describe-stack-set-operation 命令验证堆栈是否已成功创建。在 `--operation-id` 选项中，指定作为 create-stack-instances 输出部分内容返回的操作 ID。

```
aws cloudformation describe-stack-set-operation \
  --stack-set-name my-stackset \
  --operation-id operation_ID
```

创建具有服务托管权限的 Amazon CloudFormation StackSet

使用服务托管权限时，您可以将堆栈部署到特定区域中由 Amazon Organizations 托管的账户。使用此模型，您无需在每个目标账户和Amazon Web Services 区域中创建 IAM 角色。CloudFormation 代表您创建 IAM 角色。有关更多信息，请参阅 激活可信访问权限。

主题

- 注意事项
- 创建具有服务管理权限的 StackSet（控制台）
- 创建具有服务管理权限的 StackSet（Amazon CLI）

注意事项

在创建具有服务托管权限的 StackSet 之前，请考虑以下事项：

- 具有服务托管权限的 StackSets 可以由组织的管理账户或委派管理员账户启动，但所有操作均由管理账户执行。
- CloudFormation 不会将堆栈部署到管理账户，即使该账户是您组织的一部分或属于某组织单元 (OU) 也是如此。
- StackSet 可以整个组织（包括所有账户）或指定 OU 为目标。当 StackSet 以父 OU 为目标时，它会自动包含任何子 OU。默认情况下，当 StackSet 以特定 OU 为目标时，它会包含这些 OU 内的所有账户。但是，您可以使用账户筛选条件，来仅以特定账户为目标。
- 多个堆栈集可以将同一组织或 OU 作为目标。

- 不能以组织外的账户为目标。
- 您部署 StackSets 的授权取决于分配给您用于登录管理账户的 IAM 主体（用户、角色或组）的权限。有关授予向组织进行部署的权限的示例 IAM policy，请参阅 根据区域和资源类型限制堆栈集操作。
- 委派管理员具有部署到组织中的任何账户的完全权限。管理账户不能限制委派管理员部署到特定 OU 或 StackSet 操作的权限。
- 自动部署设置在 StackSet 级别应用。您无法为 OU、账户或区域选择性地调整自动部署。
- 使用服务托管权限的 StackSet 不支持包含宏或转换的嵌套堆栈或模板。

创建具有服务管理权限的 StackSet（控制台）

创建 StackSet

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择要从中管理 StackSet 的 Amazon Web Services 区域。
3. 从导航窗格中，选择 StackSets。
4. 从 StackSets 页面顶部，选择创建 StackSet。
5. 在权限下方，选择服务托管权限。

   > **Note**
   >
   > 如果为可信访问禁用 Amazon Organizations，则会显示横幅。创建或更新具有服务托管权限的 StackSet 需要可信访问。只有组织管理账户中的管理员具有对 使用 Amazon Organizations 为 StackSet 激活可信访问权限 的权限。

6. 在先决条件 - 准备模板下，选择模板已就绪。
7. 在 Specify template (指定模板) 下，选择指定包含堆栈模板的 S3 存储桶的 URL，或者上传堆栈模板文件。然后选择下一步。
8. 在指定 StackSet 详细信息页面上，提供 StackSet 名称，指定所有参数，然后选择下一步。
9. 在 Configure StackSet options (配置堆栈集选项) 页面上，在 Tags (标签) 下指定要应用于堆栈中资源的所有标签。有关如何在 Amazon 中使用标签的更多信息，请参阅《Amazon 账单与成本管理 User Guide》中的 Organizing and tracking costs using Amazon cost allocation tags。
10. 在执行配置中，请选择活动，启用 CloudFormation 的优化操作处理：

- 为了缩短部署时间，无冲突的操作会同时运行。
- 存在冲突的操作会自动排入队列，按照请求的顺序进行处理。

如果有操作正在运行或排队，CloudFormation 会将所有传入操作排入队列，即使这些操作没有冲突。在此期间，您不能更改执行设置。

11. 如果模板包含 IAM 资源，则对于功能，请选择我确认该模板可能会创建 IAM 资源以指定您要在模板中使用 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。
12. 选择下一步以继续，并激活可信访问权限（如果尚未激活）。
13. 在设置部署选项页面的部署目标下，执行以下操作之一：

    - 要部署到您组织中的所有账户，请选择部署到组织。
    - 要部署到特定 OU 中的所有账户，请选择部署到组织单位（OU）。选择 Add an OU (添加 OU)，然后将目标 OU ID 粘贴到文本框中。对每个新目标 OU 重复执行此操作。

    如果您选择部署到组织单元（OU），则对于账户筛选条件类型，可以通过选择以下选项之一并提供账号将部署目标设置为特定的个人账户。

    - 无（默认）：将堆栈部署到指定 OU 中的所有账户。
    - 交集：将堆栈部署到选定 OU 中特定的单独账户。
    - 差集：除特定账户外，将堆栈部署到选定 OU 中的所有账户。
    - 并集：将堆栈部署到指定 OU 以及额外单独账户。
14. 在自动部署下，选择是否自动部署到将来添加到目标组织或 OU 的账户。有关更多信息，请参阅 为 Amazon Organizations 中的堆栈集启用或禁用自动部署。
15. 如果启用自动部署，请在 Account removal behavior (账户删除行为) 下，选择从目标组织或 OU 中删除账户时是保留还是删除堆栈资源。

    > **Note**
    >
    > 选择保留堆栈后，堆栈将从 StackSet 中移除，但堆栈及其相关资源将会保留。资源会保持当前状态，但不再是 StackSet 的一部分。

16. 对于指定区域，选择要在其中部署堆栈的区域。
17. 对于部署选项，请执行以下操作：

- 对于最大并发账户数，请指定并发处理的账户数量。
- 在容错中，请指定每个区域允许的账户失败次数上限。达到此次数限制后，该操作将停止，不会继续到其他区域。
- 对于区域并发，请选择处理区域的方式：顺序（一次处理一个区域）或并行（并发处理多个区域）。
- 对于并发模式，请选择在操作执行期间的并发行为方式。
  - 严格容错：操作失败时降低账户并发级别，保持在容错 +1 范围内。
  - 软容错 – 即使失败，仍保持您指定的并发级别（最大并发账户数的值）。
- 对于 StackSet 依赖项，添加依赖的 StackSet ARN，最多保持在 10 个依赖项内。有关更多信息，请参阅 为 Amazon Organizations 中的堆栈集启用或禁用自动部署。

18. 选择下一步以继续。
19. 在审核页面上，验证 StackSet 将部署到正确区域中的正确账户，然后选择创建 StackSet。

    将打开 StackSet details (堆栈集详细信息) 页面。您可以在 StackSet 中查看堆栈创建的进度和状态。

创建具有服务管理权限的 StackSet（Amazon CLI）

请按照本部分中的步骤，使用 Amazon CLI 来执行以下操作：

- 创建 StackSet 容器。
- 部署堆栈实例。

> **Note**
>
> 担任委派管理员时，您必须在命令中包含 `--call-as DELEGATED_ADMIN`。

Deploy to your organization

创建 StackSet

1. 使用 create-stack-set 命令创建一个新 StackSet（名为 *`my-stackset`*）。以下示例使用存储在 S3 存储桶中的模板，启用自动部署，会在删除账户时保留堆栈。有关更多信息，请参阅 为 Amazon Organizations 中的堆栈集启用或禁用自动部署。

```
aws cloudformation create-stack-set \
  --stack-set-name my-stackset \
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/
MyApp.template \
  --permission-model SERVICE_MANAGED \
  --auto-deployment
 Enabled=true,RetainStacksOnAccountRemoval=true,DependsOn=ARN1,ARN2
```

2. 使用 list-stack-sets 命令确认 StackSet 已创建。结果中会列出您的新 StackSet。

```
aws cloudformation list-stack-sets
```

   - 如果您在登录成员账户时将 `--call-as` 选项设置为 `DELEGATED_ADMIN`，则 list-stack-sets 会返回组织管理账户中具有服务托管权限的所有堆栈集。
   - 如果您在登录您的 Amazon Web Services 账户时将 `--call-as` 选项设置为 `SELF`，则 list-stack-sets 会返回 Amazon Web Services 账户中所有自行管理的堆栈集。
   - 如果您在登录组织管理账户时将 `--call-as` 选项设置为 `SELF`，则 list-stack-sets 会返回组织管理账户中的所有堆栈集。

3. 使用 create-stack-instances 命令将堆栈添加到 StackSet 中。在 `--deployment-targets` 选项中，指定要部署到组织中所有账户的组织根 ID。

   使用 `--operation-preferences` 选项设置并发账户处理和其他部署首选项。此示例使用基于计数的设置。请注意，`MaxConcurrentCount` 不得大于 `FailureToleranceCount` + 1。对于基于百分比的设置，请改用 `FailureTolerancePercentage` 或 `MaxConcurrentPercentage`。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=r-a1b2c3d4e5 \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

   有关更多信息，请参阅《Amazon CloudFormation API 参考》中的 CreateStackInstances。

4. 使用在 create-stack-instances 输出中返回的 `operation-id` 以及以下 describe-stack-set-operation 命令，验证堆栈是否已成功创建。

```
aws cloudformation describe-stack-set-operation \
  --stack-set-name my-stackset \
```

```
  --operation-id operation_ID
```

Deploy to organizational units (OUs)

创建 StackSet

1. 使用 create-stack-set 命令创建一个新 StackSet（名为 *my-stackset*）。以下示例中，使用存储在 S3 存储桶中的模板，包含将 *KeyPairName* 设置为 *TestKey* 的参数

```
aws cloudformation create-stack-set \
  --stack-set-name my-stackset \
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/
MyApp.template \
  --permission-model SERVICE_MANAGED \
  --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
```

2. 使用 list-stack-sets 命令确认 StackSet 已创建。结果中会列出您的新 StackSet。

```
aws cloudformation list-stack-sets
```

   - 如果您在登录成员账户时将 `--call-as` 选项设置为 `DELEGATED_ADMIN`，则 list-stack-sets 会返回组织管理账户中具有服务托管权限的所有堆栈集。
   - 如果您在登录您的 Amazon Web Services 账户时将 `--call-as` 选项设置为 `SELF`，则 list-stack-sets 会返回 Amazon Web Services 账户中所有自行管理的堆栈集。
   - 如果您在登录组织管理账户时将 `--call-as` 选项设置为 `SELF`，则 list-stack-sets 会返回组织管理账户中的所有堆栈集。

3. 使用 create-stack-instances 命令将堆栈添加到 StackSet 中。在 `--deployment-targets` 选项中，指定部署目的地 OU ID。

   使用 `--operation-preferences` 选项设置并发账户处理和其他部署首选项。此示例使用基于计数的设置。请注意，`MaxConcurrentCount` 不得大于 `FailureToleranceCount` + 1。对于基于百分比的设置，请改用 `FailureTolerancePercentage` 或 `MaxConcurrentPercentage`。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo,ou-rcuk-slr5lh0a \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

有关更多信息，请参阅《Amazon CloudFormation API 参考》中的 CreateStackInstances。

4. 使用在 create-stack-instances 输出中返回的 `operation-id` 以及以下 describe-stack-set-operation 命令，验证堆栈是否已成功创建。

```
aws cloudformation describe-stack-set-operation \
  --stack-set-name my-stackset \
  --operation-id operation_ID
```

Deploy to specific accounts in OUs

您可以特定的组织单元（OU）为目标，并使用账户筛选条件来精确控制哪些账户接收堆栈部署。默认情况下，如果未指定账户筛选条件，则堆栈会部署到指定 OU 内的所有账户。

在 Amazon CLI 中，您可以使用 `--deployment-targets` 选项指定账户筛选条件。有关更多信息，请参阅 DeploymentTargets。

使用 create-stack-set 命令创建 StackSet 容器后，使用以下示例之一将堆栈部署到特定账户。

以 OU 中的特定账户为目标

以下示例中，仅向 OU1 中的账户 A1 和 A2 部署堆栈。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets
 OrganizationalUnitIds=OU1,Accounts=A1,A2,AccountFilterType=INTERSECTION \
  --regions us-west-2 us-east-1
```

从 OU 中排除账户

以下示例中，仅向 OU1 中账户 A1 和 A2 之外的所有账户部署堆栈。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets
 OrganizationalUnitIds=OU1,Accounts=A1,A2,AccountFilterType=DIFFERENCE \
  --regions us-west-2 us-east-1
```

为 Amazon Organizations 中的堆栈集启用或禁用自动部署

将新的 Amazon Organizations 账户添加到目标组织或组织单元（OU）后，CloudFormation 会自动将额外的堆栈部署到这些账户。您可以启用自动部署，并选择当账户从目标 OU 移出时，是删除还是保留堆栈及其相关资源。对于使用服务托管权限的堆栈集，这些设置可以随时修改。

自动部署的工作原理

启用自动部署后，当账户添加到目标组织或 OU、从目标组织或 OU 移除或在目标 OU 之间移动时，都会触发自动部署。

例如，考虑以 us-east-1 区域中的 OU1 为目标的 StackSet1 和以 us-east-1 区域中的 OU2 为目标的 StackSet2。OU1 包含 AccountA。

如果在启用自动部署的情况下将 AccountA 从 OU1 移至 OU2，CloudFormation 会自动运行删除操作，从 AccountA 中移除 StackSet1 堆栈，并在队列中排入一个将 StackSet2 堆栈添加到 AccountA 的创建操作。

注意事项

以下是使用自动部署时的注意事项：

- 自动部署功能在 StackSet 级别启用。您无法为 OU、账户或区域选择性地调整自动部署。
- 覆盖的参数值仅适用于当前已在目标 OU 及其子 OU 中的账户。将来添加到目标 OU 及其子 OU 的账户将使用 StackSet 默认值，而不使用覆盖的值。
- 自动部署不考虑账户级别的目标筛选器。如果您的目标是特定账户，而且启用了自动部署，StackSet 将继续部署到已部署组织中任何新添加的账户。要防止部署到新添加的账户，请禁用自动部署。
- 依赖项管理：每个 StackSet 最多定义 10 个依赖项，每个账户最多定义 100 个依赖项。例如，如果您有五个 StackSet，且每个都有五个依赖项，则您有 25 个依赖项计入 100 个依赖项上限。您可以通过服务配额控制台请求增加限额。删除 StackSet 或停用 Organizations 时，依赖项将被移除。
- 建议在使用自动部署时启用托管执行。托管执行将允许在多个目标账户中进行的自动部署操作在 StackSet 中同时执行，从而提高处理速度，对于规模较大的 Organizations 尤其如此。

启用或禁用自动部署（控制台）

启用或禁用自动部署

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了 StackSet 的 Amazon Web Services 区域。
3. 从导航窗格中，选择 StackSets。
4. 在 StackSet 页面上，选择要更新的 StackSet 名称旁边的选项。
5. 从右上角的操作菜单中选择编辑自动部署。
6. 在打开的对话框中，执行以下操作：
   a. 对于自动部署，选择激活或停用。
   b. 对于账户移除行为，请选择删除堆栈或保留堆栈。保留的资源会保持当前状态，但不再是 StackSet 的一部分。
   c. 对于 StackSet 依赖项，添加依赖的 StackSet ARN，最多保持在 10 个依赖项内。
7. 选择保存。

启用或禁用自动部署（Amazon CLI）

启用或禁用自动部署

1. 使用带 `--auto-deployment` 选项的 update-stack-set 命令。

   以下命令可启用自动部署。

   ```
   aws cloudformation update-stack-set --stack-set-name my-stackset \
     --use-previous-template --auto-deployment
    Enabled=true,RetainStacksOnAccountRemoval=true,DependsOn=ARN1,ARN2
   ```

   或者，要禁用自动部署，可指定 `Enabled=false` 作为 `--auto-deployment` 选项的值，如下例所示。

   ```
   aws cloudformation update-stack-set --stack-set-name my-stackset \
     --use-previous-template --auto-deployment Enabled=false
   ```

2. 使用 update-stack-set 输出中返回的操作 ID，运行 describe-stack-set-operation 来验证 StackSet 是否已成功更新。

```
aws cloudformation describe-stack-set-operation --operation-id operation_ID
```

更新 Amazon CloudFormation StackSet

您可以使用 CloudFormation 控制台或 Amazon CLI 更新您的 StackSet。

要在 StackSet 中添加和移除账户和区域，请参阅将堆栈添加到 StackSets和从堆栈集中删除堆栈。要覆盖堆栈的参数值，请参阅覆盖堆栈参数。

主题

- 更新 StackSet（控制台）
- 更新 StackSet（Amazon CLI）

更新 StackSet（控制台）

更新 StackSet

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了 StackSet 的 Amazon Web Services 区域。
3. 从导航窗格中，选择 StackSets。
4. 在 StackSet 页面上，选择您要更新的 StackSet。
5. 选定 StackSet 之后，从操作菜单中选择编辑 StackSet 详细信息。
6. 在选择模板页面上，根据需要更新权限部分，或跳到下一步。
7. 在先决条件 - 准备模板中，选择使用当前模板以使用当前模板，或选择替换当前模板以指定另一个模板的 S3 URL，或上传新模板。
8. 选择下一步。
9. 在指定 StackSet 详细信息页面的 StackSet 描述中，根据需要更新 StackSet 的描述。
10. 对于参数，请根据需要更新参数值。
11. 选择下一步。

12. 在配置 StackSet 选项页面上，对于标签，根据需要修改标签。您可以添加、更新或删除标签。有关如何在 Amazon 中使用标签的更多信息，请参阅《Amazon 账单与成本管理 User Guide》中的 Organizing and tracking costs using Amazon cost allocation tags。
13. 对于执行配置，您可以根据需要更新执行配置。

    > **Note**
    >
    > 请记住，当操作正在运行或已排队时，您无法更改执行设置。

14. 如果模板包含 IAM 资源，则对于功能，请选择我确认该模板可能会创建 IAM 资源以指定您要在模板中使用 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。
15. 选择下一步。
16. 在设置部署选项页面上，提供用于更新的账户和区域。

    CloudFormation 将在第一个区域内的指定账户中部署堆栈更新，然后移到下一个区域，依此类推，前提是区域的部署失败不超过指定的容错能力。

    a. [自行管理权限] 对于账户、部署位置，选择在账户中部署堆栈。在文本框中粘贴用于创建 StackSet 的目标账户 ID，用逗号分隔多个数字。

       [服务管理权限] 请执行下列操作之一：

       - 选择 Deploy to organizational units (OUs) (部署到组织单位 (OU))。输入用于创建 StackSet 的目标 OU。
       - 选择 Deploy to accounts (部署到账户)。粘贴用于创建 StackSet 的目标 OU ID 或账户 ID。

    b. 对于在指定区域，指定您希望 CloudFormation 部署更新的顺序。

    c. 对于部署选项，请执行以下操作：

       - 对于最大并发账户数，请指定并发处理的账户数量。
       - 在容错中，请指定每个区域允许的账户失败次数上限。达到此次数限制后，该操作将停止，不会继续到其他区域。
       - 对于区域并发，请选择处理区域的方式：顺序（一次处理一个区域）或并行（并发处理多个区域）。
       - 对于并发模式，请选择在操作执行期间的并发行为方式。
         - 严格容错：操作失败时降低账户并发级别，保持在容错 +1 范围内。
         - 软容错 – 即使失败，仍保持您指定的并发级别（最大并发账户数的值）。

- [服务管理权限] 对于 StackSet 依赖项，添加依赖的 StackSet ARN，最多保持在 10 个依赖项内。有关更多信息，请参阅 为 Amazon Organizations 中的堆栈集启用或禁用自动部署。

d. 选择下一步以继续。

17. 在审核页面上，审核您的选择。要进行更改，请在相关部分选择编辑。
18. 如果准备继续，则请选择提交。

CloudFormation 会开始对您的 StackSet 应用更新，然后显示 StackSet 详细信息的操作选项卡。您可在操作选项卡上查看更新操作的进度和状态。

更新 StackSet（Amazon CLI）

> **Note**
>
> 担任委派管理员时，您必须在命令中包含 `--call-as DELEGATED_ADMIN`。

1. 更新 StackSet

使用 update-stack-set 命令对 StackSet 进行更改。

在以下示例命令中，我们使用 `--parameters` 选项来更新 StackSet。具体来说，我们将传递通道配置的默认快照传输频率从 `TwentyFour_Hours` 更改为 `Twelve_Hours`。由于我们仍在使用当前模板，因此添加了 `--use-previous-template` 选项。

使用 `--operation-preferences` 选项设置并发账户处理和其他部署首选项。这些示例使用基于计数的设置。请注意，`MaxConcurrentCount` 不得大于 `FailureToleranceCount` + 1。对于基于百分比的设置，请改用 `FailureTolerancePercentage` 或 `MaxConcurrentPercentage`。

[自行管理权限] 对于 `--accounts` 选项，提供您希望更新到目标的账户 ID。

```
aws cloudformation update-stack-set --stack-set-name my-stackset \
  --use-previous-template \
  --parameters ParameterKey=MaximumExecutionFrequency,ParameterValue=Twelve_Hours \
  --accounts account_ID_1 account_ID_2 \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

[服务托管权限] 对于 `--deployment-targets` 选项，提供您希望更新到目标的组织（根）ID 或组织单元（OU）ID。

```
aws cloudformation update-stack-set --stack-set-name my-stackset \
  --use-previous-template \
  --parameters ParameterKey=MaximumExecutionFrequency,ParameterValue=Twelve_Hours \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo,ou-rcuk-slr5lh0a \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

有关更多信息，请参阅 Amazon CloudFormation API 参考中的 UpdateStackSet。

2. 通过运行 describe-stack-set-operation 命令显示更新操作的状态和结果，来验证是否已成功更新 StackSet。对于 `--operation-id`，使用由您的 update-stack-set 命令返回的操作 ID。

```
aws cloudformation describe-stack-set-operation \
  --operation-id operation_ID
```

将堆栈添加到 CloudFormation 堆栈集

创建 StackSet 时，可以为 StackSet 创建堆栈。利用 CloudFormation，您还能在创建 StackSet 后的任何时间为其他账户和区域添加更多堆栈。您可以使用 CloudFormation 控制台或 Amazon CLI 添加堆栈。

主题

- 将堆栈添加到 StackSet（控制台）
- 将堆栈添加到 StackSet（Amazon CLI）

将堆栈添加到 StackSet（控制台）

将堆栈添加到 StackSet

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了 StackSet 的 Amazon Web Services 区域。
3. 从导航窗格中，选择 StackSets。在 StackSet 页面上，选择您创建的 StackSet。

4. 选择 StackSet 后，从操作菜单中选择将堆栈添加到 StackSet。
5. 在设置部署选项页面上，执行以下操作：

   a. 对于将堆栈添加到 StackSet，选择部署新堆栈。

   b. 接下来，根据 StackSet 的权限配置执行以下操作：

      - [自行管理权限] 对于账户、部署位置，选择在账户中部署堆栈。在文本框中粘贴您的目标账号，并使用逗号分隔多个号码。
      - [服务管理权限] 对于部署目标，请执行以下操作之一：
        - 选择 Deploy to organization (部署到组织) 可部署到您组织中的所有账户。
        - 选择 Deploy to organizational units (OUs) (部署到组织单位 (OU)) 可部署到特定 OU 中的所有账户。选择 Add another OU (添加其他 OU)，然后将目标 OU ID 粘贴到文本框中。对每个新目标 OU 重复执行此操作。CloudFormation 还可以所选目标的任何子 OU 为目标。

      > **Note**
      >
      > 如果您添加的 OU 已是 StackSet 的目标，同时该 OU 中有任何账户尚未具有 StackSet 中的堆栈（例如，在创建 StackSet 后添加到 OU 并且已禁用自动部署的账户），则 CloudFormation 会在该账户中创建新的堆栈。

   c. 对于指定区域，指定要部署到在上一步中所指定目标账户中的哪个 Amazon Web Services 区域。默认情况下，CloudFormation 将在第一个区域内的指定账户中部署堆栈，然后移到下一个区域，依此类推，前提是区域的部署失败不超过指定的容错能力。

   d. 对于部署选项，请执行以下操作：

      - 对于最大并发账户数，请指定并发处理的账户数量。
      - 在容错中，请指定每个区域允许的账户失败次数上限。达到此次数限制后，该操作将停止，不会继续到其他区域。
      - 对于区域并发，请选择处理区域的方式：顺序（一次处理一个区域）或并行（并发处理多个区域）。
      - 对于并发模式，请选择在操作执行期间的并发行为方式。
        - 严格容错：操作失败时降低账户并发级别，保持在容错 +1 范围内。
        - 软容错 – 即使失败，仍保持您指定的并发级别（最大并发账户数的值）。

   e. 选择下一步。

6. 在 Specify Overrides (指定覆盖) 页面上，将属性值保留为指定值。您不会覆盖您要创建的堆栈的任何属性值。选择下一步。
7. 在审核页面上，审核您的选择。要进行更改，请在相关部分选择编辑。
8. 如果准备继续，则请选择提交。

   CloudFormation 会开始创建堆栈。选择提交后会打开 StackSet 详细信息页面，可以在其中查看 StackSet 中的堆栈创建进度和状态。完成后，新堆栈应该会在堆栈实例选项卡上列出。

将堆栈添加到 StackSet（Amazon CLI）

> **Note**
>
> 担任委派管理员时，您必须在命令中包含 `--call-as DELEGATED_ADMIN`。

将堆栈添加到具有自行管理权限的 StackSet

使用 create-stack-instances CLI 命令。在 `--accounts` 选项中，请提供您要为其创建堆栈的账户 ID。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --accounts account_id --regions eu-west-1 us-west-2
```

将堆栈添加到具有服务托管权限的 StackSet

使用 create-stack-instances CLI 命令。在 `--deployment-targets` 选项中，请提供要为其创建堆栈的组织（根）ID 或 OU ID。有关针对特定账户的命令示例，请参阅创建具有服务管理权限的 StackSet（Amazon CLI）。

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-r1qi0wl7 --regions eu-west-1 us-
west-2
```

> **Note**
>
> 如果您添加的 OU 已是 StackSet 的目标，同时该 OU 中有任何账户尚未具有 StackSet 中的堆栈（例如，在创建 StackSet 后添加到 OU 并且已禁用自动部署的账户），则 CloudFormation 会在该账户中创建新的堆栈。

覆盖 CloudFormation StackSet 中的堆栈参数值

在某些情况下，您可能希望特定区域或账户中堆栈的属性值与 StackSet 中指定的属性值不同。例如，您可能希望根据账户是用于开发还是生产来为给定参数指定不同的值。对于这些情况，CloudFormation 允许您按账户和区域覆盖堆栈中的参数值。您可在首次创建堆栈时覆盖模板参数值，也可以覆盖现有堆栈的参数值。您只能将之前在堆栈中覆盖的参数还原为 StackSet 中指定的值。

参数值覆盖适用于选定账户和区域中的堆栈。StackSet 更新期间，不会更新任何堆栈覆盖参数值，而是会保留覆盖值。

您只能覆盖在 StackSet 中指定的参数值；要添加或删除参数本身，您需要更新 StackSet 模板。如果将参数添加到 StackSet 模板，则必须先使用 StackSet 中指定的新参数和值更新所有堆栈，然后才能覆盖堆栈中的参数值。使用新参数更新所有堆栈后，您可按需覆盖各个堆栈中的参数值。

要了解在创建堆栈时如何覆盖 StackSet 参数值，请参阅将堆栈添加到 StackSets。

主题

- 覆盖堆栈参数（控制台）
- 覆盖堆栈参数（Amazon CLI）

覆盖堆栈参数（控制台）

覆盖特定堆栈的参数

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了 StackSet 的 Amazon Web Services 区域。
3. 从导航窗格中，选择 StackSets。在 StackSet 页面上，选择您的 StackSet。
4. 选定 StackSet 后，从操作菜单中选择覆盖 StackSet 参数。
5. 在设置部署选项页面上，提供要为其创建覆盖的堆栈的账户和区域。

   默认情况下，CloudFormation 将在第一个区域内的指定账户中部署堆栈，然后移到下一个区域，依此类推，前提是区域的部署失败不超过指定的容错能力。

   a. [自行管理权限] 对于部署位置，选在账户中部署堆栈。粘贴创建 StackSet 时用的部分或全部目标账户 ID。

      [服务管理权限] 请执行下列操作之一：

- 选择 Deploy to organizational units (OUs) (部署到组织单位 (OU))。输入一个或多个创建 StackSet 时使用的目标 OU。覆盖的参数值仅适用于当前已在目标 OU 及其子 OU 中的账户。将来添加到目标 OU 及其子 OU 的账户将使用 StackSet 默认值，而不使用覆盖的值。
- 选择 Deploy to accounts (部署到账户)。粘贴创建 StackSet 时使用的部分或全部目标 OU ID 或账户 ID。

b. 在指定区域中，添加已为此 StackSet 部署堆栈的一个或多个区域。

如果您添加多个区域，Specify regions (指定区域) 下的区域顺序决定其部署顺序。

c. 对于部署选项，请执行以下操作：

- 对于最大并发账户数，请指定并发处理的账户数量。
- 在容错中，请指定每个区域允许的账户失败次数上限。达到此次数限制后，该操作将停止，不会继续到其他区域。
- 对于区域并发，请选择处理区域的方式：顺序（一次处理一个区域）或并行（并发处理多个区域）。
- 对于并发模式，请选择在操作执行期间的并发行为方式。
  - 严格容错：操作失败时降低账户并发级别，保持在容错 +1 范围内。
  - 软容错 – 即使失败，仍保持您指定的并发级别（最大并发账户数的值）。

d. 选择下一步。

6. 在指定覆盖页面上，选中要覆盖的参数的复选框，然后从编辑覆盖值菜单中选择覆盖 StackSet 值。
7. 在覆盖 StackSet 参数值页面上，进行更改，然后选择保存更改。

> **Note**
>
> 要将任何覆盖参数设置回使用在堆栈中指定的值，请检查所有参数，然后从编辑覆盖值菜单中选择设置为 StackSet值。这样将会在更新堆栈后移除所有覆盖值。

8. 在审核页面上，审核您的选择。要进行更改，请在相关部分选择编辑。
9. 如果准备继续，则请选择提交。

CloudFormation 会开始更新堆栈。在选择提交后打开的 StackSet 详细信息页面中查看堆栈的进度和状态。

覆盖堆栈参数（Amazon CLI）

> **Note**
>
> 担任委派管理员时，您必须在命令中包含 `--call-as DELEGATED_ADMIN`。

覆盖特定堆栈的参数

1. 使用 update-stack-instances Amazon CLI 命令并指定 `--parameter-overrides` 选项。

   [自行管理权限] 对于 `--accounts` 选项，提供要为其覆盖堆栈参数值的账户 ID。

   ```
   aws cloudformation update-stack-instances --stack-set-name my-stackset \
     --parameter-overrides ParameterKey=Subnets,ParameterValue=subnet-1baa3351\
   \,subnet-27b86940 \
     --accounts account_id --regions us-east-1
   ```

   [服务托管权限] 对于 `--deployment-targets` 选项，提供要为其覆盖堆栈参数的组织根 ID、OU ID 或 Amazon Organizations 账户 ID。在此示例中，我们使用 *ou-rcuk-1x5j1lwo* ID 覆盖 OU 中所有账户内的堆栈参数值。

   覆盖的参数值仅适用于当前位于目标 OU 及其子 OU 中的账户。将来添加到此目标 OU 及其子 OU 的账户将使用 StackSet 默认值，而不是覆盖值。

   ```
   aws cloudformation update-stack-instances --stack-set-name my-stackset \
     --parameter-overrides ParameterKey=Subnets,ParameterValue=subnet-1baa3351\
   \,subnet-27b86940 \
     --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo \
     --regions us-east-1
   ```

2. 通过运行 describe-stack-set-operation 命令来显示更新操作的状态和结果，验证堆栈上的参数值是否已被成功覆盖。对于 `--operation-id`，使用由您的 update-stack-instances 命令返回的操作 ID。

   ```
   aws cloudformation describe-stack-set-operation --operation-id operation_ID
   ```

从 Amazon CloudFormation StackSet 中删除堆栈

您可以使用 CloudFormation 控制台或 Amazon CLI 从堆栈集中删除堆栈。

主题

- 从 StackSet 中删除堆栈（控制台）
- 从 StackSet 中删除堆栈（Amazon CLI）

> **Note**
>
> 删除顶层组织单元（OU）的堆栈后，该 OU 将不再是 StackSet 的目标。

从 StackSet 中删除堆栈（控制台）

删除堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了 StackSet 的 Amazon Web Services 区域。
3. 从导航窗格中，选择 StackSets。在 StackSet 页面上，选择 StackSet。
4. 选择 StackSet 后，从操作菜单中选择从 StackSet 中删除堆栈。
5. 在设置部署选项页面上，首先选择要删除堆栈的账户和区域。

   a. [自行管理权限] 对于账户，选择在账户中部署堆栈或在组织单元中部署堆栈。

      如果选择在账户中部署堆栈，请将目标账号粘贴到目标账号文本框中，并使用逗号分隔多个账号。

      如果选择在组织单元中部署堆栈，请将目标 OU ID 粘贴到组织编号文本框中，以将指定组织所属的所有账户作为目标。

   b. [服务管理权限] 对于组织单位（OU），请指定目标 OU ID。

      > **Important**
      >
      > CloudFormation 将从指定目标 OU 及其子组织单元中删除的堆栈。

对于账户筛选条件类型，您可以选择以下选项之一并提供账号，从而细化要删除堆栈的账户。

- 无（默认）– 从指定 OU 中的所有账户内删除堆栈。
- 交集 – 仅从选定 OU 中特定单独账户内删除堆栈。
- 差集 – 除特定账户外，从选定 OU 中的所有账户内删除堆栈。
- 并集 – 从指定 OU 以及额外单独账户中删除堆栈。

c. 对于指定区域，请选择要从其中的目标账户删除堆栈的区域。

6. 对于部署选项，请执行以下操作：

- 对于最大并发账户数，请指定并发处理的账户数量。
- 在容错中，请指定每个区域允许的账户失败次数上限。达到此次数限制后，该操作将停止，不会继续到其他区域。
- 对于保留堆栈，启用此选项可在从 StackSet 中移除堆栈时保存堆栈及其关联的资源。资源会保持当前状态，但不再是 StackSet 的一部分。
- 对于区域并发，请选择处理区域的方式：顺序（一次处理一个区域）或并行（并发处理多个区域）。
- 对于并发模式，请选择在操作执行期间的并发行为方式。
  - 严格容错：操作失败时降低账户并发级别，保持在容错 +1 范围内。
  - 软容错 – 即使失败，仍保持您指定的并发级别（最大并发账户数的值）。

7. 选择下一步。
8. 在审核页面上，审核您的选择。要进行更改，请在相关部分选择编辑。
9. 准备好从 StackSet 中删除堆栈后，请选择提交。

完成堆栈删除后，您可在 StackSet 详细信息页面的堆栈实例选项卡上验证相关堆栈是否已从 StackSet 中删除。

从 StackSet 中删除堆栈（Amazon CLI）

> **Note**
>
> 担任委派管理员时，您必须在命令中包含 `--call-as DELEGATED_ADMIN`。

使用 delete-stack-instances 命令和 StackSet 的名称。

在这些示例中，我们使用 `--no-retain-stacks`选项，因为我们将不会保留任何堆栈。如果要保留堆栈及其资源，请使用 `--retain-stacks`，而不是 `--no-retain-stacks`。

对于 `--regions`，请指定要从中删除堆栈的 Amazon Web Services 区域，例如 `us-west-2` 和 `us-east-1`。

使用 `--operation-preferences` 选项设置并发账户处理和其他首选项。这些示例使用基于计数的设置。请注意，`MaxConcurrentCount` 不得大于 `FailureToleranceCount` + 1。对于基于百分比的设置，请改用 `FailureTolerancePercentage` 或 `MaxConcurrentPercentage`。

删除堆栈（自行管理权限）

在 `--accounts` 选项中，指定要从中删除堆栈的账户 ID。

```
aws cloudformation delete-stack-instances --stack-set-name my-stackset \
  --accounts account_ID_1 account_ID_2 \
  --regions us-west-2 us-east-1 \
  --no-retain-stacks \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

删除堆栈（服务管理权限）

在 `--deployment-targets` 中，指定要从中删除堆栈的组织根 ID 或组织单元（OU）ID。

> **Important**
>
> CloudFormation 将从指定目标 OU 及其子组织单元中删除的堆栈。

```
aws cloudformation delete-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5jlwo,ou-rcuk-slr5lh0a \
  --regions us-west-2 us-east-1 \
  --no-retain-stacks \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

有关更多信息，请参阅《Amazon CloudFormation API 参考》中的 DeleteStackInstances。

在完成堆栈删除后，您也可以选择运行 describe-stack-set-operation 命令来显示堆栈删除操作的状态和结果，验证是否已从 StackSet 中删除堆栈。对于 `--operation-id`，使用由您的 delete-stack-instances 命令返回的操作 ID。

```
aws cloudformation describe-stack-set-operation --stack-set-name my-stackset \
  --operation-id ddf16f54-ad62-4d9b-b0ab-3ed8e9example
```

删除 Amazon CloudFormation StackSet

要删除 StackSet，您必须首先删除 StackSet 中的所有堆栈。有关如何删除所有堆栈的信息，请参阅从堆栈集中删除堆栈。

主题

- 删除 StackSet（控制台）
- 删除 StackSet（Amazon CLI）
- 删除服务角色 (可选)

删除 StackSet（控制台）

删除 StackSet

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了 StackSet 的 Amazon Web Services 区域。
3. 在 StackSet 页面上，选择 StackSet。
4. 选定 StackSet 后，从操作菜单中选择删除 StackSet。
5. 当系统提示您确认是否要删除该 StackSet 时，选择删除 StackSet。

删除 StackSet（Amazon CLI）

> **Note**
>
> 担任委派管理员时，您必须在命令中包含 `--call-as DELEGATED_ADMIN`。

删除 StackSet

1. 使用以下 delete-stack-set 命令。在提示您确认时，键入 **y**，然后按 Enter。

```
aws cloudformation delete-stack-set --stack-set-name my-stackset
```

2. 通过运行 list-stack-sets 命令验证 StackSet 是否已删除。list-stack-sets 命令的结果应显示状态为 DELETED 的堆栈。

```
aws cloudformation list-stack-sets
```

删除服务角色 (可选)

如果您不再需要使用 CloudFormation 执行 StackSet 操作所需的 IAM 服务角色，则建议您删除这些角色。

对于自行管理的堆栈集，指您创建的角色。有关这些角色的更多信息，请参阅 授予自行管理的权限。

对于服务托管的 StackSet，为 StackSet 自动创建的角色，在组织管理账户中的后缀为 CloudFormationStackSetsOrgAdmin，在每个目标账户中的后缀为 CloudFormationStackSetsOrgMember。有关更多信息，请参阅 服务关联角色。

删除服务角色（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 从导航窗格中，选择角色，然后选中要删除的角色旁的复选框。
3. 在页面顶部的 Role actions 菜单中，选择 Delete role。
4. 在确认对话框中，选择 Yes, Delete（是的，删除）。如果确定，您就可以继续执行删除操作，即使仍在加载上次访问服务数据。

删除服务角色（Amazon CLI）

- 使用以下 delete-role 命令。在提示您确认时，键入 **y**，然后按 Enter。

```
aws iam delete-role --role-name role name
```

有关删除角色的更多信息，请参阅《IAM 用户指南》中的删除角色或实例配置文件。

使用目标账户门防止 StackSet 部署失败

账户门是一项可选功能，用于 CloudFormation 在目标账户中开始执行 StackSet 操作之前验证该账户是否满足特定要求。此验证是通过充当先决条件检查的 Amazon Lambda 函数执行。

账户门的一个常见示例是确认目标账户上无有效或未解决的 CloudWatch 警报。CloudFormation 将在您每次在目标账户中启动堆栈操作时调用此 Lambda 函数，并且仅在此函数返回 `SUCCEEDED` 代码时继续。如果 Lambda 函数返回 `FAILED` 状态，则 CloudFormation 不会继续执行请求的操作。如果您未配置账户门 Lambda 函数，CloudFormation 会跳过检查并继续执行操作。

如果您的目标账户的账户门检查失败，则失败的操作将计入您指定的堆栈容错数量或百分比。有关容错能力的更多信息，请参阅StackSet 操作选项。

账户门仅适用于 StackSets 操作。此功能不适用于 StackSet 外部的其他 CloudFormation 操作。

要求

必须满足以下要求才能使用账户门：

- Lambda 函数必须命名为 `AWSCloudFormationStackSetAccountGate` 才能使用此功能。
- AWSCloudFormationStackSetExecutionRole 要有权调用 Lambda 函数。如果没有这些权限，CloudFormation 会跳过账户门检查，并继续堆栈操作。
- 必须在目标账户中添加 Lambda `InvokeFunction` 权限，账户门才能正常工作。目标账户信任策略必须具有与管理员账户的信任关系。下面是一个示例策略语句，可授予 Lambda `InvokeFunction` 权限。

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "*"
        }
    ]
}
```

用于创建 Lambda 函数的 CloudFormation 模板

使用以下示例模板创建 Lambda `AWSCloudFormationStackSetAccountGate` 函数。要使用这些模板中的任意一种创建新堆栈，请参阅通过 CloudFormation 控制台创建堆栈。

模板位置	说明
https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AccountGateSucceeded.yml	创建实现 Lambda 账户门函数（将返回 `SUCCEEDED` 状态）的堆栈。
https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AccountGateFailed.yml	创建实现 Lambda 账户门函数（将返回 `FAILED` 状态）的堆栈。

为 Amazon CloudFormation StackSet 选择并发模式

并发模式是 StackSetOperationPreferences 的参数，允许您选择并发级别在 StackSet 操作期间的行为方式。您可以在以下模式之间进行选择：

- 严格容错：此选项可动态降低并发级别，确保失败账户的数量永远不会超过容错值 +1。初始实际并发数设置为最大并发账户数值或容错值 +1 中的较小值。然后，实际并发数按失败次数成比例减少。这是默认行为。
- 软容错：此选项将容错与实际并发数分离。这允许 StackSet 操作在最大并发账户数值设置的并发级别上运行，而不管失败次数是多少。

严格容错会降低 StackSet 操作失败时的部署速度，因为每次失败都会降低并发数。软容错会优先考虑部署速度，同时仍能利用 CloudFormation 安全功能。这可以让您查看和解决因常见问题而导致的 StackSet 操作失败，例如与现有资源、服务配额和权限有关的问题。

有关 StackSets 堆栈操作失败的更多信息，请参阅 堆栈操作失败的常见原因。

有关最大并发账户数和容错的更多信息，请参阅 StackSet 操作选项。

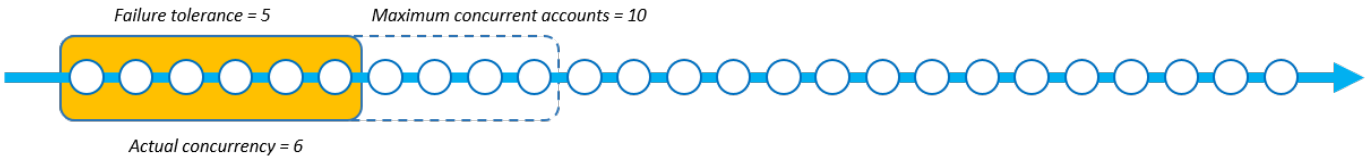
每种并发模式的工作原理

下图直观地显示了每种并发模式在 StackSet 操作期间的工作原理。节点字符串表示对单个 Amazon Web Services 区域 的部署，且每个节点都是一个目标 Amazon Web Services 账户。

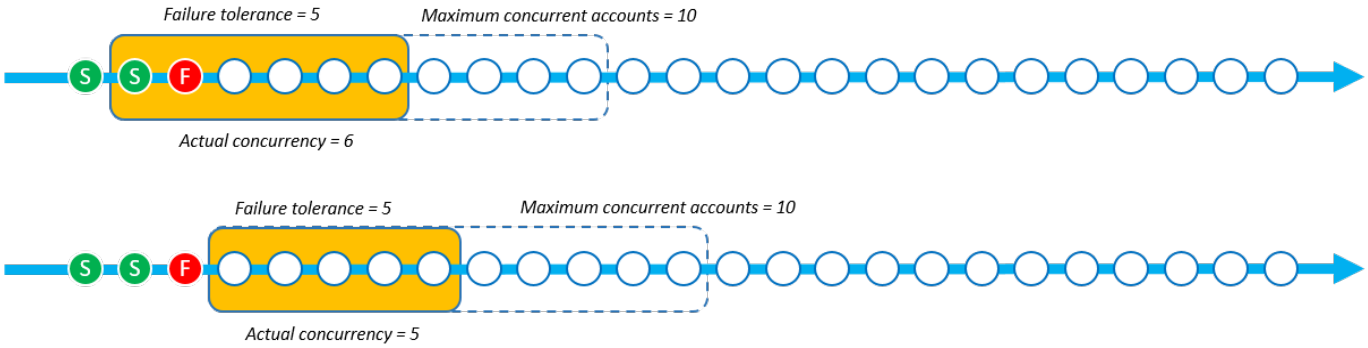
严格容错

如果使用严格容错的 StackSet 操作将容错值设置为 5，最大并发账户数的值设置为 10 时，实际并发数为 6。实际并发数为 6，因为 5+1 的容错值低于最大并发账户数的值。

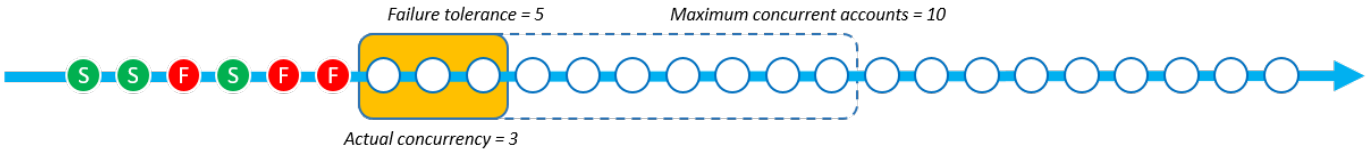
下图显示了容错值对最大并发账户数值的影响，以及它们对 StackSet 操作实际并发数的影响：



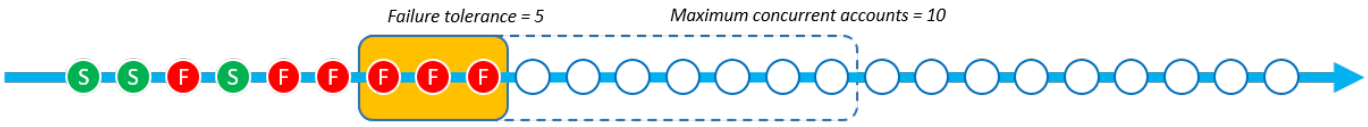
当部署开始且出现失败的堆栈实例时，为了提供安全的部署体验，实际并发数会降低。当 StackSets 部署 1 个堆栈实例失败时，实际并发数从 6 个降低到 5 个。



严格容错模式按失败堆栈实例的数量成比例降低实际并发数。在以下示例中，当 StackSets 再部署的 2 个堆栈实例失败时，实际并发数从 5 个减少到 3 个，从而使失败的堆栈实例总数降至 3 个。



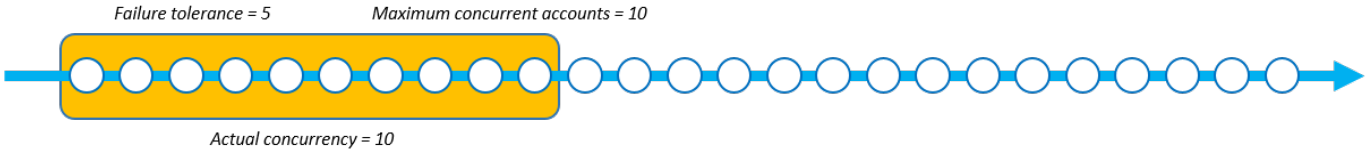
如果失败的堆栈实例数等于定义的容错值 +1 时，StackSets 的 StackSet 操作失败。在以下示例中，当有 6 个失败的堆栈实例且容错值为 5 时，StackSets 操作失败。



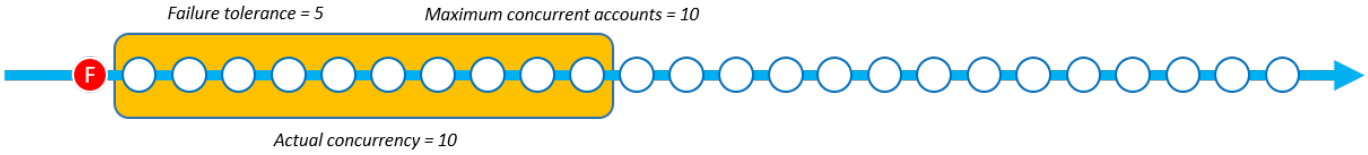
在此示例中，CloudFormation 在停止 StackSet 操作之前部署了 9 个堆栈实例（3 个成功，6 个失败）。

软容错

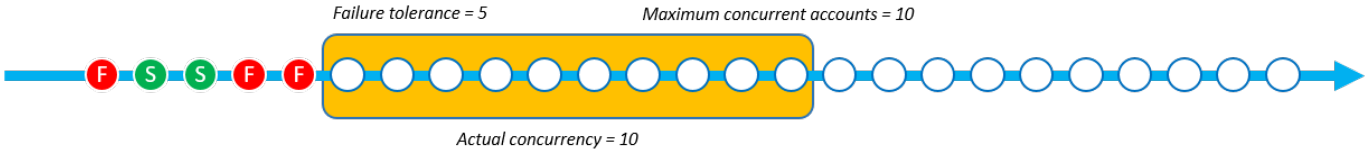
当使用软容错的 StackSet 操作将容错值设置为 5，最大并发账户数的值设置为 10 时，实际并发数为 10。



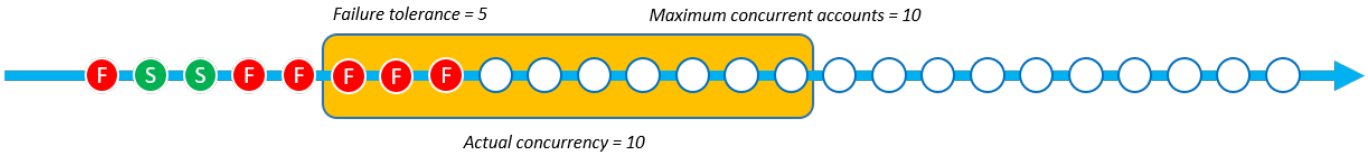
当部署开始且出现失败的堆栈实例时，实际并发数不会改变。在以下示例中，1 个堆栈操作失败，但实际并发数仍为 10。



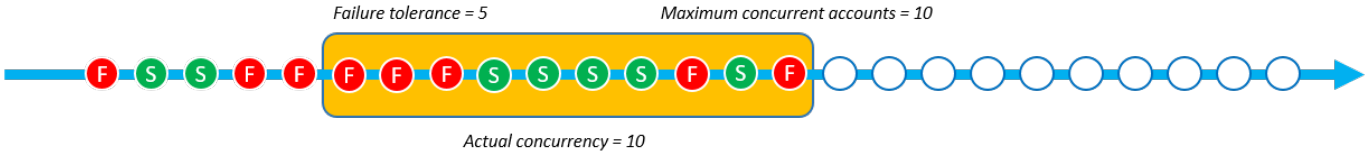
即使再有 2 个堆栈实例失败，实际并发数仍保持在 10。



当失败的堆栈实例超过容错值时，StackSets 的 StackSet 操作失败。在以下示例中，当有 6 个失败的堆栈实例且容错计数为 5 时，StackSets 操作失败。不过，在并发队列中的其余操作完成之前，操作不会结束。



StackSets 继续部署已在并发队列中的堆栈实例。这意味着失败的堆栈实例的数量可能高于容错。在以下示例中，有 8 个失败的堆栈实例，因为尽管 StackSet 操作的容错已达到 5，但并发队列仍有 7 个操作要执行。



在此示例中，StackSets 在停止堆栈操作之前部署了 15 个堆栈实例（7 个成功，8 个失败）。

根据部署速度在“严格容错”和“软容错”之间进行选择

在严格容错模式和软容错模式之间进行选择，具体取决于 StackSet 部署的首选速度以及允许的部署失败次数。

下表显示了每种并发模式如何处理在尝试部署总共 1000 个堆栈实例时失败的 StackSet 操作。在每种情况下，容错的值设置为 100 个堆栈实例，最大并发账户数的值设置为 250 个堆栈实例。

虽然 StackSets 实际上以滑动窗口的形式对账户进行排队（参阅 每种并发模式的工作原理），但此示例通过分批显示操作的方式来演示每种模式的速度。

严格容错

这个示例使用严格容错模式，相对于前面每个批次中发生的失败数量，降低了实际并发数。每个批次有 20 个失败的实例，这会将下一个批次的实际并发数降低 20，直到 StackSet 操作达到容错值 100。

在下表中，第一批的初始实际并发数为 101 个堆栈实例。实际并发数为 101，因为这是最大并发账户数（250）和容错（100）+1 中的较低值。每个批次包含 20 个失败的堆栈实例部署，这会使后续每个批次的实际并发数降低 20 个堆栈实例。

严格容错	第 1 批	第 2 批	第 3 批	第 4 批	第 5 批	第 6 批
实际并发数	101	81	61	41	21	-
失败的实例数	20	20	20	20	20	-
成功的堆栈实例数	81	61	41	21	1	-

当 StackSet 操作达到 100 个堆栈实例的容错时，使用严格容错的操作分五批完成了 305 个堆栈实例部署。StackSet 操作在失败之前成功部署了 205 个堆栈实例。

软容错

此示例使用软容错模式，无论失败实例的数量如何，都将保持相同的实际并发数，该值由 250 个堆栈实例的最大并发账户值定义。在达到 100 个实例的容错值之前，StackSet 操作会保持相同的实际并发数。

在下表中，第一批的初始实际并发数为 250 个堆栈实例。实际并发数为 250，因为最大并发账户数的值设置为 250，且软容错模式允许 StackSets 使用此值作为实际并发数，不管失败数量是多少。虽然此示例的每个批次中都有 50 个失败的操作，但实际并发数仍不受影响。

软容错	第 1 批	第 2 批	第 3 批	第 4 批	第 5 批	第 6 批
实际并发数	250	250	-	-	-	-
失败的实例数	50	50	-	-	-	-
成功的堆栈实例数	200	200	-	-	-	-

通过相同的最大并发账户值和容错值，使用软容错模式的操作分两批完成了 500 个堆栈实例部署。StackSet 操作在失败之前成功部署了 400 个堆栈实例。

选择并发模式（控制台）

创建或更新 StackSet 时，在设置部署选项页面上，为并发模式选择严格容错或软容错。

选择并发模式（Amazon CLI）

您可以通过以下 StackSets 命令使用 `ConcurrencyMode` 参数：

- create-stack-instances
- delete-stack-instances
- detect-stack-set-drift
- import-stacks-to-stack-set
- update-stack-instances
- update-stack-set

这些命令有一个名为 `--operation-preferences` 的现有参数，可以使用 `ConcurrencyMode` 设置。`ConcurrencyMode` 可以设置为以下任一值：

- `STRICT_FAILURE_TOLERANCE`

- SOFT_FAILURE_TOLERANCE

以下示例使用 STRICT_FAILURE_TOLERANCE ConcurrencyMode 创建堆栈实例，FailureToleranceCount 设置为 10，MaxConcurrentCount 设置为 5。

```
aws cloudformation create-stack-instances \
  --stack-set-name example-stackset \
  --accounts 123456789012 \
  --regions eu-west-1  \
  --operation-preferences
 ConcurrencyMode=STRICT_FAILURE_TOLERANCE,FailureToleranceCount=10,MaxConcurrentCount=5
```

> **Note**
>
> 有关创建和更新 StackSet 的详细步骤，请参阅以下主题：
>
> - 创建 StackSets（自行管理权限）
> - 创建 StackSets（服务托管权限）
> - 更新 StackSets

对 CloudFormation StackSets 进行偏差检测

即使在您通过 CloudFormation 管理堆栈及其包含的资源时，用户也可以在 CloudFormation 外部更改这些资源。用户可以使用创建资源的底层服务直接编辑资源。通过对 StackSet 执行偏移检测，您可以确定属于该 StackSet 的任何堆栈实例是否与它们的预期配置不同或已偏移。

主题

- CloudFormation 如何对 StackSet 执行偏移检测
- 检测 StackSet 上的偏移（控制台）
- 检测 StackSet 上的偏移（Amazon CLI）
- 停止对 StackSet 的偏移检测

CloudFormation 如何对 StackSet 执行偏移检测

在 CloudFormation 对 StackSet 执行偏移检测时，它对与 StackSet 中的每个堆栈实例关联的堆栈执行偏移检测。为此，CloudFormation 将堆栈中的每个资源的当前状态与该资源的预期状态进行比较，该

状态是堆栈的模板以及任何指定的输入参数中定义的。如果资源的当前状态不同于预期状态，则认为该资源已偏离。如果堆栈中的一个或多个资源已偏离，则将堆栈本身视为已偏离，并将与堆栈关联的堆栈实例也视为已偏离。如果 StackSet 中的一个或多个堆栈实例已偏移，则将 StackSet 本身视为已偏移。

偏差检测确定非托管更改；即，在 CloudFormation 外部对堆栈进行的更改。直接通过 CloudFormation 对堆栈进行的更改（而不是在 StackSet 级别）不会被视为偏移。例如，假设您具有一个与 StackSet 的堆栈实例关联的堆栈。如果您使用 CloudFormation 更新该堆栈以使用不同的模板，则不会将其视为偏移，即使该堆栈现在具有与属于 StackSet 的任何其他堆栈不同的模板。这是因为，该堆栈仍与它在 CloudFormation 中的预期模板和参数配置匹配。

有关 CloudFormation 如何对堆栈执行偏差检测的详细信息，请参阅使用偏移检测功能检测堆栈和资源的非托管配置更改。

由于分别 CloudFormation 对每个堆栈执行偏差检测，因此，它在确定堆栈是否偏离时考虑任何覆盖的参数值。有关在堆栈实例中覆盖模板参数的更多信息，请参阅覆盖 CloudFormation StackSet 中的堆栈参数值。

如果直接对与堆栈实例关联的堆栈执行偏差检测，则无法在 StackSets 控制台页面中查看这些偏差结果。

检测 StackSet 上的偏移（控制台）

检测 StackSet 上的偏移

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在 StackSet 页面上，选择要执行偏移检测的 StackSet。
3. 从 Actions (操作) 菜单中，选择 Detect drifts (检测偏差)。

   CloudFormation 会显示一个信息栏，指示已为选定堆栈启动偏移检测。
4. 可选：要监控偏差检测操作进度，请执行以下操作：

   a. 选择 StackSet 名称以显示 Stackset 详细信息页面。

   b. 选择 Operations (操作) 选项卡，选择偏差检测操作，然后选择 View drift details (查看偏差详细信息)。

   CloudFormation 显示 Operation details (操作详细信息) 对话框。

5. 请耐心等待，直到 CloudFormation 完成偏差检测操作。偏移检测操作完成后，CloudFormation 会更新堆栈的偏移状态和上次偏移检查时间。这些字段在所选 StackSet 的 StackSet 详细信息页面的概述选项卡上列出。

   偏移检测操作可能需要一些时间，具体取决于 StackSet 中包含的堆栈实例数量以及 StackSet 中包含的资源数量。您一次只能在一个 StackSet 上运行一个偏移检测操作。即使您关闭信息栏，CloudFormation 也会继续执行偏差检测操作。

6. 要查看 StackSet 中的堆栈实例的偏移检测结果，请选择堆栈实例选项卡。

   Stack name (堆栈名称) 列列出与每个堆栈实例关联的堆栈的名称，Drift status (偏差状态) 列列出该堆栈的偏差状态。如果堆栈的一个或多个资源已偏离，则认为堆栈已偏离。

7. 要查看与特定堆栈实例关联的堆栈的偏差检测结果，请执行以下操作：

   a. 选择操作选项卡。

   b. 选择要查看其偏差检测结果的偏差操作。拆分面板将显示堆栈实例状态和所选操作的原因。对于偏差操作，状态原因列显示堆栈实例的偏差状态。

   c. 选择要查看其偏差详细信息的堆栈实例，然后选择查看资源偏差。在资源偏差页面上的资源偏差状态中，会列出每个堆栈资源及其偏差状态，以及最后一次在资源上启动偏差检测的时间。将会显示每个资源的逻辑 ID 和物理 ID，以帮助您识别它们。

8. 您可以使用偏差状态列根据资源偏差状态对资源排序。

   查看已修改资源的详细信息：

   - 选定资源后，选择查看偏差详细信息。

     CloudFormation 会显示该资源的偏差详细信息页面。此页面列出资源的差异。其中还列出资源的预期和当前属性值。

     > **Note**
     >
     > 如果堆栈所属的区域和账户与您当前登录的地区和账户不同，则检测偏差按钮将被禁用，您将无法查看详细信息。

检测 StackSet 上的偏移（Amazon CLI）

要使用 Amazon CLI 在整个堆栈上检测偏差，请使用以下步骤：

检测 StackSet 上的偏移

1. 使用 detect-stack-set-drift 命令检测整个 StackSet 及其关联堆栈实例上的偏移。

   以下示例中，对 `stack-set-drift-example` StackSet 启动偏移检测。

   ```
   aws cloudformation detect-stack-set-drift \
       --stack-set-name stack-set-drift-example
   ```

   输出：

   ```
   {
       "OperationId": "c36e44aa-3a83-411a-b503-cb611example"
   }
   ```

2. 由于 StackSet 偏移检测操作可能是长时间运行的操作，因此，请使用 describe-stack-set-operation 命令监控偏移操作的状态。该命令使用 detect-stack-set-drift 命令返回的 StackSet 操作 ID。

   以下示例使用上一示例中的操作 ID 返回有关 StackSet 偏移检测操作的信息。在此示例中，该操作仍在运行。在与该 StackSet 关联的七个堆栈实例中，发现一个堆栈实例已偏移，两个实例正在进行同步，而其余四个堆栈实例仍在进行偏移检测。由于一个实例已偏移，StackSet 本身的偏移状态现在为 DRIFTED。

   ```
   aws cloudformation describe-stack-set-operation \
       --stack-set-name stack-set-drift-example \
       --operation-id c36e44aa-3a83-411a-b503-cb611example
   ```

   输出：

   ```
   {
       "StackSetOperation": {
           "Status": "RUNNING",
           "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
   AWSCloudFormationStackSetAdministrationRole",
           "OperationPreferences": {
               "RegionOrder": []
           },
           "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
           "StackSetDriftDetectionDetails": {
               "DriftedStackInstancesCount": 1,
   ```

```
            "TotalStackInstancesCount": 7,
            "LastDriftCheckTimestamp": "2019-12-04T20:34:28.543Z",
            "InSyncStackInstancesCount": 2,
            "InProgressStackInstancesCount": 4,
            "DriftStatus": "DRIFTED",
            "FailedStackInstancesCount": 0
        },
        "Action": "DETECT_DRIFT",
        "CreationTimestamp": "2019-12-04T20:33:13.673Z",
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22example",
        "OperationId": "c36e44aa-3a83-411a-b503-cb611example"
    }
}
```

该示例稍后执行相同的命令，以显示在偏差检测操作完成后返回的信息。在与该 StackSet 关联的总共 7 个堆栈实例中，有两个实例已偏移，从而使 StackSet 本身的偏移状态显示为 DRIFTED。

```
aws cloudformation describe-stack-set-operation \
    --stack-set-name stack-set-drift-example \
    --operation-id c36e44aa-3a83-411a-b503-cb611example
```

输出：

```
{
    "StackSetOperation": {
        "Status": "SUCCEEDED",
        "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
AWSCloudFormationStackSetAdministrationRole",
        "OperationPreferences": {
            "RegionOrder": []
        }
        "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
        "EndTimestamp": "2019-12-04T20:37:32.829Z",
        "StackSetDriftDetectionDetails": {
            "DriftedStackInstancesCount": 2,
            "TotalStackInstancesCount": 7,
            "LastDriftCheckTimestamp": "2019-12-04T20:36:55.612Z",
            "InSyncStackInstancesCount": 5,
            "InProgressStackInstancesCount": 0,
            "DriftStatus": "DRIFTED",
            "FailedStackInstancesCount": 0
```

```
        },
        "Action": "DETECT_DRIFT",
        "CreationTimestamp": "2019-12-04T20:33:13.673Z",
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22example",
        "OperationId": "c36e44aa-3a83-411a-b503-cb611example"
    }
}
```

3. 在 StackSet 偏移检测操作完成后，请使用 describe-stack-set、list-stack-instances、describe-stack-instance 和 list-stack-instance-resource-drifts 命令查看结果。

   describe-stack-set 命令包括 describe-stack-set-operation 命令返回的相同详细偏差信息。

   ```
   aws cloudformation describe-stack-set \
       --stack-set-name stack-set-drift-example
   ```

   输出：

   ```
   {
       "StackSet": {
           "Status": "ACTIVE",
           "Description": "Demonstration of drift detection on stack sets.",
           "Parameters": [],
           "Tags": [
               {
                   "Value": "Drift detection",
                   "Key": "Feature"
               }
           ],
           "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
           "Capabilities": [],
           "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
   AWSCloudFormationStackSetAdministrationRole",
           "StackSetDriftDetectionDetails": {
               "DriftedStackInstancesCount": 2,
               "TotalStackInstancesCount": 7,
               "LastDriftCheckTimestamp": "2019-12-04T20:36:55.612Z",
               "InProgressStackInstancesCount": 0,
               "DriftStatus": "DRIFTED",
               "DriftDetectionStatus": "COMPLETED",
               "InSyncStackInstancesCount": 5,
               "FailedStackInstancesCount": 0
   ```

```
        },
        "StackSetARN": "arn:aws:cloudformation:us-east-1:123456789012:stackset/
stack-set-drift-example:bd1f4017-d4f9-432e-a73f-8c22example",
        "TemplateBody": [details omitted],
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22ebexample",
        "StackSetName": "stack-set-drift-example"
    }
}
```

您可以使用 list-stack-instances 命令返回有关与 StackSet 关联的堆栈实例的摘要信息，包括每个堆栈实例的偏移状态。

在本示例中，通过对示例 StackSet 执行 list-stack-instances 并将偏移状态筛选设置为 DRIFTED，我们可以确定哪两个堆栈实例具有 DRIFTED 偏移状态。

```
aws cloudformation list-stack-instances \
    --stack-set-name stack-set-drift-example \
    --filters Name=DRIFT_STATUS,Values=DRIFTED
```

输出：

```
{
"Summaries": [
        {
"StackId": "arn:aws:cloudformation:eu-west-1:123456789012:stack/
StackSet-stack-set-drift-example-b0fb6083-60c0-4e39-
af15-2f071e0db90c/0e4f0940-16d4-11ea-93d8-0641cexample",
            "Status": "CURRENT",
            "Account": "012345678910",
            "Region": "eu-west-1",
            "LastDriftCheckTimestamp": "2019-12-04T20:37:32.687Z",
            "DriftStatus": "DRIFTED",
            "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
            "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
        },
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
StackSet-stack-set-drift-example-b7fde68e-e541-44c2-b33d-
ef2e2988071a/008e6030-16d4-11ea-8090-12f89example",
            "Status": "CURRENT",
```

```
            "Account": "123456789012",
            "Region": "us-east-1",
            "LastDriftCheckTimestamp": "2019-12-04T20:34:28.275Z",
            "DriftStatus": "DRIFTED",
            "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
            "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
        },

        [additional stack instances omitted]

    ]
}
```

describe-stack-instance 命令也返回该信息，但仅为单个堆栈实例返回该信息，如以下示例所示。

```
aws cloudformation describe-stack-instance \
    --stack-set-name stack-set-drift-example \
    --stack-instance-account 012345678910 --stack-instance-region us-east-1
```

输出：

```
{
    "StackInstance": {
        "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
StackSet-stack-set-drift-example-b7fde68e-e541-44c2-b33d-
ef2e2988071a/008e6030-16d4-11ea-8090-12f89example",
        "Status": "CURRENT",
        "Account": "123456789012",
        "Region": "us-east-1",
        "ParameterOverrides": [],
        "DriftStatus": "DRIFTED",
        "LastDriftCheckTimestamp": "2019-12-04T20:34:28.275Z",
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
        "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
    }
}
```

4. 在确定哪些堆栈实例已偏离后，您可以使用有关 list-stack-instances 或 describe-stack-instance 命令返回的堆栈实例的信息以运行 list-stack-instance-resource-drifts 命令。该命令会返回详细信息，说明对于特定偏差操作而言，堆栈中的哪些资源已产生偏差。

以下示例使用 --stack-instance-resource-drift-statuses 参数来请求在上一个偏差操作示例中已修改或删除的资源的堆栈偏差信息。该请求会返回有关已修改的一个资源的信息，包括有关其中两个属性及其已更改值的详细信息。不会删除任何资源。

```
aws cloudformation list-stack-instance-resource-drifts \
    --stack-set-name my-stack-set-with-resource-drift \
    --stack-instance-account 123456789012 \
    --stack-instance-region us-east-1 \
    --operation-id c36e44aa-3a83-411a-b503-cb611example \
    --stack-instance-resource-drift-statuses MODIFIED DELETED
```

输出：

```
{
    "Summaries": [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/my-
stack-set-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
            "ResourceType": "AWS::SQS::Queue",
            "Timestamp": "2018-03-26T17:23:34.489Z",
            "PhysicalResourceId": "https://sqs.us-
east-1.amazonaws.com/123456789012/my-stack-with-resource-drift-Queue-494PBHCO76H4",
            "StackResourceDriftStatus": "MODIFIED",
            "PropertyDifferences": [
                {
                    "PropertyPath": "/DelaySeconds",
                    "ActualValue": "120",
                    "ExpectedValue": "20",
                    "DifferenceType": "NOT_EQUAL"
                },
                {
                    "PropertyPath": "/RedrivePolicy/maxReceiveCount",
                    "ActualValue": "12",
                    "ExpectedValue": "10",
                    "DifferenceType": "NOT_EQUAL"
                }
            ],
            "LogicalResourceId": "Queue"
        }
    ]
}
```

停止对 StackSet 的偏移检测

由于对 StackSet 的偏移检测可能是长时间运行的操作，因此，在某些情况下，您可能希望停止当前对 StackSet 运行的偏移检测操作。

停止对 StackSet 的偏移检测（控制台）

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在 StackSet 页面上，选择 StackSet 的名称。

   CloudFormation 将显示所选 StackSet 的 StackSet 详细信息页面。
3. 在 StackSets details (堆栈集详细信息) 页面上，选择 Operations (操作) 选项卡，然后选择偏差检测操作。
4. 选择 Stop operation (停止操作)。

停止对 StackSet 的偏移检测（Amazon CLI）

- 使用 stop-stack-set-operation 命令。您必须同时提供 StackSet 名称和偏移检测 StackSet 操作的操作 ID。

```
aws cloudformation stop-stack-set-operation \
    --stack-set-name stack-set-drift-example \
    --operation-id 624af370-311a-11e8-b6b7-500cexample
```

将堆栈导入 Amazon CloudFormation StackSet

堆栈导入操作可以将现有堆栈导入新的或现有的 StackSet，因此您可以通过一次操作将现有堆栈迁移到 StackSet。

对于自行管理的 StackSets，导入操作可以导入管理员账户或不同目标账户和 Amazon Web Services 区域 中的堆栈。对于服务托管的 StackSets，导入操作可以导入与管理账户相同的 Amazon Organizations 中的任何堆栈。

以下是将堆栈导入 StackSets 时的注意事项和限制：

- 导入操作使用内联堆栈 ID 时最多可以导入 10 个堆栈，使用 Amazon S3 对象时最多可以导入 200 个堆栈。

- 不支持 NoEcho 属性。包含 NoEcho 的堆栈不会通过 StackSet 导入功能导入到新的 StackSet 中。
- 堆栈只能属于一个 StackSet。
- 在堆栈导入操作中将标签明确指定为参数，就可以将堆栈标签实现到 StackSet。
- 堆栈的自定义参数覆盖在导入操作期间不受影响。
- 不能使用编辑 StackSet 详细信息选项更新已导入堆栈实例的参数。必须使用覆盖 StackSet 参数。有关覆盖参数的更多信息，请参阅覆盖堆栈参数。
- StackSets 配额和堆栈实例在导入堆栈时适用。有关限额的更多信息，请参阅了解 CloudFormation 配额。

主题

- Amazon CloudFormation StackSets 的自行管理堆栈导入
- Amazon CloudFormation StackSets 的服务托管堆栈导入
- 恢复堆栈导入至 Amazon CloudFormation StackSet

Amazon CloudFormation StackSets 的自行管理堆栈导入

CloudFormation 堆栈导入操作可以将现有堆栈导入新的或现有的 StackSet，因此您通过一次操作就可以将现有堆栈迁移到 StackSet。通过使用堆栈导入，您可以避免停机和中断，而无需删除和重新创建这些资源。将堆栈导入 StackSet后，原来的堆栈将成为指定的 StackSet 的堆栈实例。

自行管理式堆栈导入的注意事项

- 堆栈导入操作需要在其中创建 StackSet 的管理员账户和包含堆栈的目标账户。
- 目标账户必须有权使用 GetTemplate 操作，输入堆栈 ID 或 ARN。因此，您的管理员账户必须被授予 AWSCloudFormationStackSetAdministrationRole 或 AWSCloudFormationStackSetsExectionRole 权限。

主题

- 将现有堆栈导入新 StackSet（控制台）
- 将现有堆栈导入现有 StackSet（控制台）
- 将堆栈导入 StackSet（Amazon CLI）

将现有堆栈导入新 StackSet（控制台）

开始之前，请标识要导入的堆栈。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 从导航窗格中，选择 StackSets。
3. 在 StackSets (堆栈集) 页面顶部，选择 Create StackSet (创建堆栈集)。
4. 在选择模板页面上，通过以下选项之一指定模板，然后选择下一步。
   - 选择 Amazon S3 URL，然后在文本框中指定您的模板的 URL。
   - 选择 Upload a template file（上传模板文件），然后浏览您的模板。
   - 选择 From stack ID（从堆栈 ID），然后输入堆栈 ID。
5. 在指定 StackSet 详细信息页面中，输入要创建的 StackSet 的名称，然后选择下一步。

   （可选）输入 StackSet 的描述。
6. 在配置堆栈集选项页面上，查看您的选项，然后选择下一步。
7. 在设置部署选项页面上，选择将堆栈导入堆栈集。
8. 在 Stacks to import（要导入的堆栈）字段中，输入要导入的堆栈的堆栈 ID。例如 *arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/f449b250-b969-11e0-a185-5081d0136786*。

   （可选）选择添加另一个堆栈 ID，然后输入您要导入的另一个堆栈的堆栈 ID。每个堆栈导入操作最多可添加 10 个堆栈。
9. 查看您的部署选项，然后选择下一步。
10. 在审核页面上，检查您的选择和 StackSet 的属性。准备好将堆栈导入 StackSet 后，选择提交。

结果：导入的堆栈现在是指定 StackSet 的堆栈实例。要了解有关堆栈导入状态的更多信息，请参阅 StackSets 状态代码。

将现有堆栈导入现有 StackSet（控制台）

开始之前，请标识要导入的堆栈。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。

2. 从导航窗格中，选择 StackSets。
3. 在 StackSet 页面中，选择要导入堆栈的 StackSet。
4. 选择 StackSet 后，从操作菜单中选择将堆栈添加到 StackSet。
5. 在设置部署选项页面中，选择将堆栈导入堆栈集，然后在要导入的堆栈字段中输入要导入的堆栈的堆栈 ID。例如 *arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/f449b250-b969-11e0-a185-5081d0136786*。

   （可选）选择添加另一个堆栈 ID，然后输入您要导入的另一个堆栈的堆栈 ID。每个堆栈导入操作最多可添加 10 个堆栈。
6. 选择下一步。
7. 在指定覆盖页面上，查看您的选择，然后选择下一步。
8. 在审核页面上，检查您的选择和 StackSet 的属性。准备好创建 StackSet 后，选择提交。

结果：导入的堆栈现在是指定 StackSet 的堆栈实例。要了解有关堆栈导入状态的更多信息，请参阅 StackSets 状态代码。

将堆栈导入 StackSet（Amazon CLI）

将现有堆栈导入新 StackSet

以下 `create-stack-set` 命令将创建一个 StackSet 并导入指定堆栈。要导入的堆栈由其 ARN 标识。将占位符文本替换为您自己的信息。

```
aws cloudformation create-stack-set \
  --stack-set-name MyStackSet \
  --stack-id arn:aws:cloudformation:us-east-1:123456789012:stack/
StackToImport/466df9e0-0dff-08e3-8e2f-5088487c4896 \
  --administration-role-arn arn:aws:iam::123456789012:role/
AWSCloudFormationStackSetAdministrationRole \
  --execution-role-name AWSCloudFormationStackSetExecutionRole
```

将现有堆栈导入现有 StackSet

以下 `import-stacks-to-stack-sets` 命令会将指定堆栈导入到 *MyStackSet* StackSet。要导入的堆栈由其 ARN 标识。将占位符文本替换为您自己的信息。

```
aws cloudformation import-stacks-to-stack-set \
```

```
  --stack-set MyStackSet \
  --stack-ids arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/
f449b250-b969-11e0-a185-5081d0136786
```

要指定多个堆栈，请使用以下格式作为 `--stack-ids` 选项的值。

```
--stack-ids "arn_1" "arn_2"
```

将导入的堆栈克隆到其他区域和账户

运行 `create-stack-instances` 命令会将堆栈实例添加到 StackSet。将占位符文本替换为您自己的信息。

```
aws cloudformation create-stack-instances \
  --stack-set-name MyStackSet \
  --accounts '["account_ID_1","account_ID_2"]' \
  --regions '["region_1","region_2"]'
```

Amazon CloudFormation StackSets 的服务托管堆栈导入

CloudFormation 堆栈导入操作可以将现有堆栈导入新的或现有的 StackSet，因此您通过一次操作就可以将现有堆栈迁移到 StackSet。StackSets 扩展了堆栈的功能，因此您可以在一次操作中跨多个账户和区域创建、更新或删除堆栈。

服务托管式堆栈导入的注意事项

- 堆栈导入操作需要一个管理账户或委派管理员账户，您可以在其中管理关联的 Amazon Organizations 组织，例如使用 StackSets 启用信任访问。
- 目标账户必须是由管理账户或委派管理员账户管理的 Amazon Organizations 成员。
- 目标堆栈位于其中一个目标 OU 中。
- 目标账应为 Amazon Organizations 的成员。
- Amazon Organizations 访问对于 Organizations 应处于 `ACTIVATED` 状态。
- 所导入的堆栈应位于任意成员账户中，而不是管理账户中。

主题

- 将服务托管堆栈导入新 StackSet（控制台）

- 创建服务托管堆栈并将其导入现有 StackSet（控制台）
- 将服务托管堆栈导入现有 StackSet（控制台）
- 将服务托管堆栈导入 StackSet（Amazon CLI）

将服务托管堆栈导入新 StackSet（控制台）

使用 Amazon Web Services 管理控制台 将堆栈导入新 StackSet

要将新堆栈导入 StackSet，请标识包含要导入资源的堆栈。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 从导航窗格中，选择 StackSets。
3. 在 StackSets (堆栈集) 页面顶部，选择 Create StackSet (创建堆栈集)。
4. 在选择模板页面上，执行以下操作：

   a. 在堆栈集权限模型中，选择服务托管权限。
   b. 在先决条件 – 准备模板中，选择模板已准备就绪，然后使用以下选项之一选择您的模板：

      - 对于 Amazon S3 URL，请在“Amazon S3 URL”字段中输入您的 Amazon S3 URL。
      - 在上传模板文件中，选择本地计算机中的 CloudFormation 模板。

   接受您的设置并选择 Next（下一步）。
5. 在指定堆栈集详细信息页面上，执行以下操作：

   a. 在 StackSet 名称框中输入 StackSet 名称。
   b. （可选）在 StackSet description（堆栈集描述）部分中输入描述。

   在配置堆栈集选项页面上，查看您的选项，然后选择下一步。
6. 在设置部署选项页面上，执行以下操作：

   a. 在将堆栈添加到堆栈集中，选择将堆栈导入堆栈集。
   b. 对于 Stacks to import（要导入的堆栈），选择堆栈导入方法。

      i. 对于 Stack ID（堆栈 ID），输入您的堆栈 ID。
      ii. 对于堆栈 URL，请输入该 Amazon S3 URL。

7. 在关联组织单位下，执行以下操作：

   a. 选择与组织关联以使用根 OU。

   b. 选择与组织单位（OU）关联以输入要导入堆栈的父 OU ID。例如，如果 `Stack 1` 和 `Stack 2` 位于 `OU1` 下，而 `Stack 3` 位于 `OU2` 下，则输入 `OU1` 和 `OU2`。

   接受您的设置并选择 Next（下一步）。

8. 在 Review（审核）页面上，检查您的设置，然后选择 Submit（提交）。

创建服务托管堆栈并将其导入现有 StackSet（控制台）

要将现有堆栈导入新 StackSet，请标识包含要导入资源的堆栈。

创建 StackSet 并导入堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 从导航窗格中，选择 StackSets。
3. 在 StackSets (堆栈集) 页面顶部，选择 Create StackSet (创建堆栈集)。
4. 在选择模板页面上，执行以下操作：

   a. 在堆栈集权限模型中，选择服务托管权限。

   b. 在先决条件 – 准备模板中，选择模板已准备就绪，然后使用以下选项之一选择您的模板：

      - 对于 Amazon S3 URL，请在“Amazon S3 URL”字段中输入您的 Amazon S3 URL。
      - 在上传模板文件中，选择本地计算机中的 CloudFormation 模板。

   接受您的设置并选择 Next（下一步）。

5. 在指定堆栈集详细信息页面上，执行以下操作：

   a. 在 StackSet 名称框中输入 StackSet 名称。

   b. （可选）在 StackSet description（堆栈集描述）部分中输入描述。

   在配置堆栈集选项页面上，查看您的选项，然后选择下一步。

6. 在设置部署选项页面上，执行以下操作：

- 对于 Add stacks to stack set（将堆栈添加到堆栈集），选择 Deploy new stacks（部署新堆栈）。

7. 在关联组织单位部分，执行以下操作：

   a. 选择与组织关联以使用根 OU。

   b. 选择与组织单位（OU）关联以输入要导入堆栈的父 OU ID。例如，如果 `Stack 1` 和 `Stack 2` 位于 0U1 下，而 `Stack 3` 位于 0U2 下，则输入 0U1 和 0U2。

8. 对于 Specify regions（指定区域）和 Deployment options（部署选项），查看您的选择。

   接受您的设置并选择 Next（下一步）。

9. 在 Review（审核）页面上，检查您的设置，然后选择 Submit（提交）。

将服务托管堆栈导入现有 StackSet（控制台）

选择 StackSet 并确定您要导入的堆栈。

将堆栈导入现有 StackSet

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 从导航窗格中，选择 StackSets。
3. 选择要将堆栈导入到的 StackSet，然后从操作下拉列表中选择将堆栈添加到 StackSet。
4. 在设置部署选项页面上，执行以下操作：

   a. 在将堆栈添加到堆栈集中，选择将堆栈导入堆栈集。

   b. 在要导入的堆栈下，执行以下操作：

      i. 对于 Stack ID（堆栈 ID），输入您的堆栈 ID。

      ii. 对于堆栈 URL，请输入该 Amazon S3 URL。

   c. 在关联组织单位下，执行以下操作：

      i. 选择与组织关联以使用根 OU。

      ii. 选择与组织单位（OU）关联以输入要导入堆栈的父 OU ID。例如，如果 `Stack 1` 和 `Stack 2` 位于 0U1 下，而 `Stack 3` 位于 0U2 下，则输入 0U1 和 0U2。

      接受您的设置并选择 Next（下一步）。

5. 查看 Specify overrides（指定覆盖）页面，然后选择 Next（下一步）。
6. 确认并检查 Review（审核）页面，然后选择 Submit（提交）。

将服务托管堆栈导入 StackSet（Amazon CLI）

创建 StackSet 后，可以通过传递所导入堆栈的堆栈 ID 来导入堆栈。您还可以传递要将其映射到的 OU ID 列表。

CloudFormation 将在这些 OU 中导入用户提供的堆栈，并将这些 OU 用作 StackSet 的部署目标。输入中显示的堆栈 ID 将在内部映射到 OU ID 列表输入中最近的 OU。如果堆栈不属于输入列表中的现有 OU ID，则 Amazon CLI 将返回 `StackNotFoundException` 错误。

`import-stacks-to-stack-set` 操作为 OU ID 输入中的堆栈创建堆栈实例。以下 Amazon CLI 示例使用 `import-stacks-to-stack-set` 操作将堆栈导入 StackSet。

- 要使用 `import-stacks-to-stack-sets` 操作，请指定要导入到堆栈集的 `stack-ids` 或 `stack-ids-url`。

```
aws cloudformation import-stacks-to-stack-set \
  --stack-set-name ServiceMangedStackSet \
  --stack-ids "arn:123456789012:us-east-1:Stack1" \
  --organizational-unit-ids ou-examplerootid111-exampleouid111
```

```
aws cloudformation import-stacks-to-stack-set \
  --stack-set-name ServiceMangedStackSet \
  --stack-ids-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/file-
name.json \
  --organizational-unit-ids ou-examplerootid111-exampleouid111
```

> **Note**
>
> `import-stacks-to-stack-sets` 操作要求您指定至少一个组织单位 ID（OU ID），以便它可以将要导入的堆栈关联到该特定 OU。此操作不会为关联 OU 中的其他成员账户创建堆栈实例。要更新关联 OU 的成员账户，请使用 `create-stack-instances` 或 `update-stack-instances`。

`create-stack-set` 使用用户提供的模板（直接上传或来自 Amazon S3）为 OU 下的所有账户创建堆栈实例。以下 Amazon CLI 示例使用 `create-stack-set` 操作将堆栈导入新 StackSet。

- 要使用 `create-stack-set` 操作，请指定 StackSet 名称并将堆栈导入新创建的 StackSet。

```
aws cloudformation create-stack-set \
  --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/file-name.json \
  --permission-model SERVICE_MANAGED \
  --auto-deployment Enabled=true
```

恢复堆栈导入至 Amazon CloudFormation StackSet

如果堆栈实例中有不必要的更改，可以恢复堆栈实例的导入。

恢复堆栈实例导入时，CloudFormation 会从 StackSet 中删除堆栈实例，但保留堆栈的资源。

要恢复堆栈导入操作，请完成以下步骤：

1. 将堆栈实例删除后要保留的每个资源的 `DeletionPolicy` 属性指定为 `Retain`。有关更多信息，请参阅 恢复导入操作。
2. 从 StackSet 中删除堆栈实例。有关更多信息，请参阅 从 Amazon CloudFormation StackSet 中删除堆栈。
3. 删除 StackSet。有关更多信息，请参阅 删除 Amazon CloudFormation StackSet。

使用 Amazon CloudFormation StackSet 的最佳实践

本节介绍了定义 StackSet 模板、创建堆栈或将堆栈添加到 StackSet 或更新 StackSet 的最佳实践。

如果不熟悉 CloudFormation，请查看 CloudFormation 最佳实践 主题获取更多有助于您更高效、安全地使用 CloudFormation 的建议。

主题

- 定义模板
- 创建堆栈或将堆栈添加到 StackSet
- 更新 StackSet 中的堆栈

定义模板

- 定义要在多个区域内的多个账户中标准化的模板。
- 在创建模板时，如果在同一账户的多个区域中创建全局资源（如 IAM 角色和 Amazon S3 桶），请确保全局资源没有发生命名冲突。
- 每个 StackSet 都有一个模板和参数集。与 StackSet 关联的所有账户中创建的堆栈都相同。在创作模板时，使其足够精细以便您能够实现控制与标准化的良好平衡。
- 建议您将模板存储在 Amazon S3 桶中。

创建堆栈或将堆栈添加到 StackSet

- 在将大量堆栈实例添加到 StackSet 之前，验证是否能将堆栈实例添加到初始 StackSet。
- 选择适用于您的使用案例的部署 (推出) 选项。
  - 对于更传统的部署，请将 Maximum Concurrent Accounts 设置为 1，并将 Failure Tolerance 设置为 0。在 Region Order 列表中将影响最小的区域设置为第一位。从一个区域开始。
  - 要更快地进行部署，请根据需要增大 Maximum Concurrent Accounts 和 Failure Tolerance 的值。
- 针对 StackSet 的操作可能需要花费大量时间，具体取决于包含的堆栈实例数量。

更新 StackSet 中的堆栈

- 默认情况下，更新 StackSet 时会更新所有堆栈实例。如果您在两个区域中各有 20 个账户，则您将有 40 个堆栈实例，并且所有实例都将在您更新 StackSet 时进行更新。

  对于具有大量堆栈实例的 StackSet，建议您先选择更新几个测试账户中的堆栈实例以测试更新的模板版本，然后再更新所有堆栈实例。
- 要对 StackSet 中的各个堆栈更新进行更精细的控制，请计划创建多个 StackSet。
- 更新包含大量堆栈的 StackSet 可能需要花费大量时间。在此版本中，一次仅允许对 StackSet 执行一项操作。提前计划更新，以免妨碍您在 StackSet 上执行其他操作。

Amazon CloudFormation StackSet 示例模板

本节包含一些链接，它们指向可以帮助您在企业中使用 Amazon CloudFormation StackSet 的 CloudFormation 示例模板。该部分中列出的模板启用 Amazon CloudTrail 或 Amazon Config 及其包含的规则。

> **Important**
>
> 作为允许 Amazon Config 访问 Amazon S3 存储桶时的最佳安全实践，强烈建议您在存储桶策略中使用 `AWS:SourceAccount` 条件来限制访问权限。新模板已更新，含有 `AWS:SourceAccount`。如果现有的存储桶策略未遵循此最佳安全实践，强烈建议您编辑该存储桶策略纳入此保护。这样可以确保 Amazon Config 仅代表预期用户授予访问权限。

描述	S3 链接
启用了 Amazon CloudTrail	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml
启用了 Amazon Config	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSConfig.yml
使用集中式日志记录启用 Amazon Config	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSConfigForOrganizations.yml
跨 Amazon 组织或特定 Amazon 账户启用 Amazon Data Lifecycle Manager 默认策略	• EBS 快照默认策略 – https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/DataLifecycleManagerEBSSnapshotDefaultPolicy.yaml • EBS 支持的 AMI 默认策略 – https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/DataLifecycleManagerAMIDefaultPolicy.yaml https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEncryptedVolumes.yml

描述	S3 链接
配置 Amazon Config 规则以确定是否启用 CloudTrail	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleCloudtrailEnabled.yml
配置 Amazon Config 规则以确定是否启用根 MFA	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleRootAccountMFAEnabled.yml
配置 Amazon Config 规则以确定是否附加 EIP	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEipAttached.yml
配置 Amazon Config 规则以确定是否加密 EBS 卷	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEncryptedVolumes.yml

Amazon CloudFormation StackSets 故障排除

本主题包含一些常见的 StackSets 问题以及这些问题的建议的解决方案。

主题

- 堆栈操作失败的常见原因
- 重试失败的堆栈创建或更新操作
- 堆栈实例删除操作失败
- 堆栈导入操作失败
- StackSets 操作的堆栈实例失败计数

堆栈操作失败的常见原因

问题：堆栈操作失败，并且堆栈实例状态为 OUTDATED。

原因：有几个导致堆栈操作失败的常见原因。

- 目标账户中的权限不足，无法创建模板中指定的资源。

- CloudFormation 模板可能存在错误。验证 CloudFormation 中的模板并修复错误，然后再尝试创建 StackSet。
- 模板可能尝试创建必须唯一但不唯一的全球资源 (如 S3 存储桶)。
- 指定的目标账号不存在。检查您在向导的 Set deployment options 页面上指定的目标账号。
- 管理员账户与目标账户没有信任关系。
- 目标账户中已存在模板中指定的资源的最大数量。例如，您可能已达到目标账户中允许的 IAM 角色数限制，但模板创建更多 IAM 角色。
- 您已达到 StackSet 中允许的最大堆栈数量。有关每个 StackSet 的最大堆栈数，请参阅 了解 CloudFormation 配额。

解决方案：有关创建 StackSet 之前目标和管理员账户所需权限的更多信息，请参阅授予管理员账户的所有用户管理所有目标账户中堆栈的权限。

重试失败的堆栈创建或更新操作

问题：堆栈创建或更新失败，并且堆栈实例状态为 OUTDATED。要查找堆栈创建或更新失败的原因，请打开 CloudFormation 控制台，查看堆栈的事件，这些事件的状态为 DELETED（创建操作失败时）或 FAILED（更新操作失败时）。浏览堆栈事件，然后查找 Status reason 列。Status reason 的值解释了堆栈操作失败的原因。

在确定堆栈创建失败的根本原因并准备好重试堆栈创建后，请执行下列步骤。

解决方案：执行下列步骤以重试堆栈操作。

1. 在控制台中，选择包含操作失败的堆栈的 StackSet。
2. 在操作菜单上，选择 Edit StackSet details (编辑堆栈集详细信息) 以重试创建或更新堆栈。
3. 在指定模板页面上，要使用相同的 CloudFormation 模板，请保留默认选项使用当前模板。如果您的堆栈操作因模板需要更改而失败，并且您想上传修订后的模板，请改为选择 Upload a template to Amazon S3，然后选择 Browse 以选择更新的模板。上传完修订后的模板后，选择 Next。
4. 在指定堆栈详细信息页面上，如果您不更改特定于模板的任何参数，请选择下一步。
5. 在 Set deployment options 页上，更改 Maximum concurrent accounts 和 Failure tolerance 的默认值 (如果需要)。有关这些设置的更多信息，请参阅 StackSet 操作选项。
6. 在审核页面上，检查所选的内容，然后选中相应的复选框以确认所需的 IAM 功能。选择提交。
7. 如果您的堆栈未成功更新，请在解决任何阻止堆栈创建的基本问题后重复此过程。

堆栈实例删除操作失败

问题：堆栈删除操作失败。

原因：对于任何已启用终止保护的堆栈，堆栈删除操作将失败。

解决方案：确定该堆栈是否已启用终止保护。如果已启用，则禁用终止保护，然后再重新执行堆栈实例删除操作。

堆栈导入操作失败

问题：堆栈导入操作无法将现有堆栈导入新的或现有的 StackSet。堆栈实例处于 `INOPERABLE` 状态。

解决方案：通过完成以下任务恢复堆栈导入操作。

1. 使用从 StackSet 删除堆栈选项，并在配置过程中启用 RetainStacks，然后继续从 StackSet 中删除堆栈实例。有关更多信息，请参阅 从 Amazon CloudFormation StackSet 中删除堆栈。
2. 您将看到 StackSet 的堆栈实例会更新，删除 `INOPERABLE` 堆栈实例。
3. 根据导入失败错误修复堆栈实例，然后重试堆栈导入操作。

StackSets 操作的堆栈实例失败计数

如果堆栈实例无法预置或更新，堆栈实例失败计数会发出警报。由于以下一个或多个原因，这些堆栈实例未部署：

- 现有资源具有类似配置
- 缺少依赖项，例如 Amazon Identity and Access Management（IAM）角色
- 其他冲突因素

如果您想以最大并发进行部署，则最大并发计数最多比容错计数多一。例如，如果容错计数为 9，则最大并发计数不能超过 10。即使某些堆栈实例无法更新，这也会导致操作返回 `SUCCEEDED`。新的堆栈实例失败计数让您能够确定，操作是否仅在符合条件的情况下成功，因为容错计数设置为允许所有失败。

您可以使用 Amazon Web Services 管理控制台、Amazon SDK 或 Amazon CLI 获取失败计数并筛选堆栈实例，以确定需要重新部署哪些实例。

使用控制台

若要查看失败堆栈实例的数量，请执行以下操作：

1. 打开 CloudFormation 控制台，然后选择 StackSet。
2. 选择您的 StackSet，然后选择 Operations（操作）选项卡。
3. 在 Status（状态）列中选择状态以查看状态详细信息。您将在状态详细信息中找到特定操作的失败堆栈实例的数量。

若要查看操作的堆栈实例的账户、区域和状态，请执行以下操作：

1. 在状态详细信息中，选择失败的堆栈实例计数。示例：Stack instances: **<number of failed stack instances>**（堆栈实例：）。
2. 通过选择面板标头来展开侧面板。侧面板中的结果是所选操作完成后堆栈实例的状态。

若要查看操作的当前堆栈实例详细信息，请执行以下操作：

1. 选择 Stack Instances（堆栈实例）选项卡。
2. 按 Last operation ID（上次操作 ID）筛选。结果为上次修改实例的操作的当前状态和状态原因。您可以将此筛选条件与 Amazon 账户、Amazon 区域、详细状态和 偏差状态结合使用，以进一步优化搜索结果。

使用 Amazon CLI

若要获取失败的堆栈实例数量，请调用 `describe-stack-set-operation` 或 `list-stack-set-operations` 并参阅 `StatusDetails`。

```
aws cloudformation describe-stack-set-operation --stack-set-name ss1 \
    --operation-id 5550e62f-c822-4331-88fa-21c1d7bafc60
```

```
{
    "StackSetOperation": {
        "OperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60",
        "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
        "Action": "CREATE",
        "Status": "SUCCEEDED",
        "OperationPreferences": {
```

```
            "RegionOrder": [],
            "FailureToleranceCount": 10,
            "MaxConcurrentCount": 10
        },
        "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
AWSCloudFormationStackSetAdministrationRole",
        "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
        "CreationTimestamp": "2022-10-26T17:18:53.947000+00:00",
        "EndTimestamp": "2022-10-26T17:19:35.304000+00:00",
        "StatusDetails": {
            "FailedStackInstancesCount": 3
        }
    }
}
```

```
aws cloudformation list-stack-set-operations --stack-set-name ss1
```

```
{
    "Summaries": [
        {
            "OperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60",
            "Action": "CREATE",
            "Status": "SUCCEEDED",
            "CreationTimestamp": "2022-10-26T17:18:53.947000+00:00",
            "EndTimestamp": "2022-10-26T17:19:35.304000+00:00",
            "StatusDetails": {
                "FailedStackInstancesCount": 3
            },
            "OperationPreferences": {
                "RegionOrder": [],
                "FailureToleranceCount": 10,
                "MaxConcurrentCount": 10
            }
        }
    ]
}
```

若要获取特定操作的历史概述，请使用 `list-stack-set-operation-results` 查看操作完成后每个堆栈实例的状态和状态原因。参阅以下实例以找到 `Status` 和 `StatusReason`：

```
aws cloudformation list-stack-set-operation-results --stack-set-name ss1 \
```

```
  --operation-id 5550e62f-c822-4331-88fa-21c1d7bafc60 --
filters Name=OPERATION_RESULT_STATUS,Values=FAILED
```


```
{
    "Summaries": [
        {
            "Account": "123456789012",
            "Region": "us-west-2",
            "Status": "FAILED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "AccountGateResult": {
                "Status": "SKIPPED",
                "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'."
            },
            "OrganizationalUnitId": ""
        },
        {
            "Account": "123456789012",
            "Region": "us-west-1",
            "Status": "FAILED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "AccountGateResult": {
                "Status": "SKIPPED",
                "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'."
            },
            "OrganizationalUnitId": ""
        },
        {
            "Account": "123456789012",
            "Region": "us-east-1",
            "Status": "FAILED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "AccountGateResult": {
```

```
                "Status": "SKIPPED",
                "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'."
            },
            "OrganizationalUnitId": ""
        }
    ]
}
```

将 `list-stack-instances` 和 DETAILED_STATUS 以及 LAST_OPERATION_ID 筛选条件一起使用，以获取在上次尝试部署堆栈实例的操作中失败的堆栈实例列表。请参阅示例中带有 DETAILED_STATUS 和 LAST_OPERATION_ID 的 `--filters` 标志：

```
aws cloudformation list-stack-instances --stack-set-name ss1 \
  --filters Name=DETAILED_STATUS,Values=FAILED Name=LAST_OPERATION_ID,Values=5550e62f-
c822-4331-88fa-21c1d7bafc60
```

```
{
    "Summaries": [
        {
            "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
            "Region": "us-east-1",
            "Account": "123456789012",
            "Status": "OUTDATED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "StackInstanceStatus": {
                "DetailedStatus": "FAILED"
            },
            "OrganizationalUnitId": "",
            "DriftStatus": "NOT_CHECKED",
            "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
        },
        {
            "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
            "Region": "us-west-1",
            "Account": "123456789012",
            "Status": "OUTDATED",
```

```
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "StackInstanceStatus": {
                "DetailedStatus": "FAILED"
            },
            "OrganizationalUnitId": "",
            "DriftStatus": "NOT_CHECKED",
            "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
        },
        {
            "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
            "Region": "us-west-2",
            "Account": "123456789012",
            "Status": "OUTDATED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "StackInstanceStatus": {
                "DetailedStatus": "FAILED"
            },
            "OrganizationalUnitId": "",
            "DriftStatus": "NOT_CHECKED",
            "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
        }
    ]
}
```

要查找上一个操作 ID 来修改堆栈实例，请使用 `list-stack-instances` 或 `describe-stack-instance` 来获取 `LastOperationId`：

```
aws cloudformation describe-stack-instance --stack-set-name ss1 \
  --stack-instance-account 123456789012 --stack-instance-region us-east-2
```

```
{
    "StackInstance": {
        "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
        "Region": "us-west-2",
        "Account": "123456789012",
        "ParameterOverrides": [],
        "Status": "OUTDATED",
        "StackInstanceStatus": {
            "DetailedStatus": "FAILED"
```

```
        },
        "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
        "OrganizationalUnitId": "",
        "DriftStatus": "NOT_CHECKED",
        "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
    }
}
```

使用 Git 同步将堆栈与存储在 Git 存储库中的源代码同步。

借助 Git 同步，您可以使用源代码控制来管理 CloudFormation 堆栈。为此，您可以配置 CloudFormation 来监控 Git 存储库。系统会监控存储库中两个文件的更改：

- 定义堆栈的 CloudFormation 模板文件
- 堆栈部署文件，其中包含配置堆栈的参数

凭借 Git 同步，您可以使用拉取请求和版本跟踪，从一个集中位置配置、部署和更新 CloudFormation 堆栈。提交对模板或部署文件的更改后，CloudFormation 将自动更新堆栈。如果您使用拉取请求，则 CloudFormation 可以在拉取请求上留下注释，解释在实际更新堆栈之前将对堆栈进行哪些更改。但是，您需要先启用此功能。

Git 同步提供了一个控制台界面，您可以使用该界面链接到存储库、生成堆栈部署文件、更新 CloudFormation 模板并向仓库提交拉取请求。Git 同步还提供了一个状态控制面板，可用于监控、编辑活动的 Git 同步堆栈部署并对其进行故障排查。创建堆栈时，可通过 CloudFormation 控制台访问 Git 同步。您也可以使用 Amazon CodeConnections 访问 Git 同步。有关更多信息，请参阅《开发人员工具控制台用户指南》中的针对已链接存储库使用同步配置。

Git 同步支持 GitHub、GitHub Enterprise、GitLab、Bitbucket 和 GitLab 自行管理存储库。

> **Note**
>
> Git 同步在以下区域可用：美国东部（弗吉尼亚州北部）、美国东部（俄亥俄州）、美国西部（北加利福尼亚）、美国西部（俄勒冈州）、加拿大（中部）、亚太地区（孟买）、亚太地区（东京)、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）、欧洲地区（爱尔兰）、欧洲地区（伦敦）、欧洲地区（巴黎）、欧洲地区（斯德哥尔摩）、欧洲地区（法兰克福）、欧洲地区（米兰）和南美洲（圣保罗）。

主题

- Git 同步如何与 CloudFormation 的协同工作
- 使用 Git 同步使堆栈与 Git 存储库同步的先决条件
- 使用 Git 同步从存储库源代码创建堆栈
- 启用 CloudFormation 在拉取请求中发布堆栈更改摘要

- Git 同步状态控制面板

Git 同步如何与 CloudFormation 的协同工作

本主题描述了 Git 同步的工作原理，并介绍使用它所需掌握的重要概念。

主题

- Git 同步的工作原理
- 关于拉取请求的注释
- 堆栈部署文件
- CloudFormation 模板文件
- 模板定义存储库

Git 同步的工作原理

要使用 Git 同步，您必须首先使用 Amazon CodeConnections 服务将 Git 提供程序连接到 CloudFormation。在本指南的过程中，通过 Amazon CodeConnections 控制台创建了连接。或者，您也可以使用 Amazon CLI 创建连接。您可以使用以下任意 Git 提供程序：

- （GitHub）
- GitHub Enterprise
- GitLab
- Bitbucket
- GitLab 自托管

接下来，您创建用一个来定义堆栈的 CloudFormation 模板，并将其添加到存储库中。此模板文件受到监控。提交更改后，CloudFormation 会自动更新堆栈。

在 CloudFormation 控制台中，您可以创建一个新的堆栈，然后选择从 Git 同步，告知 CloudFormation 使用 Git 同步。您要指定希望 CloudFormation 监控的存储库和分支，并在存储库中指定用于定义堆栈的 CloudFormation 模板。

在配置过程中，您可以从存储库中提供自己的堆栈部署文件，也可以让 Git 同步为您生成一个。堆栈部署文件包含用于配置堆栈中资源的参数和值。此堆栈部署文件受到监控。提交更改后，CloudFormation 会自动更新堆栈。

Git 同步会在存储库中创建一个拉取请求，以便将您的堆栈与 CloudFormation 模板文件和堆栈部署文件同步。如果 Git 同步为您生成了堆栈部署文件，该文件将通过 Git 同步提交到您的存储库。

然后，您可以将拉取请求合并到存储库中，以便 CloudFormation 预置堆栈，使用您的部署参数对其进行配置，并开始监控存储库的更改。

从此时开始，只要您对模板文件或堆栈部署文件进行更改并将其提交到存储库，CloudFormation 就会自动检测到更改。如果您的团队使用拉取请求，则您的团队成员可以在部署更改之前对其进行审核和批准。拉取请求被接受后，CloudFormation 将会立即部署更改。

在 CloudFormation 控制台中，您可以监控堆栈的 Git 同步配置的状态，以及查看应用到堆栈的提交历史记录。控制台还提供了用于重新配置 Git 同步和排查问题的工具。

关于拉取请求的注释

您可以通过在控制台中打开启用对拉取请求的评论选项，从而选择让 CloudFormation 通过 Amazon CodeConnections 服务在拉取请求中创建代码更改的摘要。通过在拉取请求中提供更改摘要，让团队成员可以在合并拉取请求之前轻松检查和了解建议修改的影响。有关更多信息，请参阅 启用 CloudFormation 在拉取请求中发布堆栈更改摘要。

堆栈部署文件

堆栈部署文件是 JavaScript Object Notation（JSON）或 YAML 标准格式的文件，其中包含管理 CloudFormation 堆栈的参数和值。其更改将受到监控。将对文件的更改提交到存储库时，关联的堆栈会自动更新。

堆栈部署文件包含一个键值对和两个字典：

- `template-file-path`

  这是 CloudFormation 模板文件的完整存储库路径。模板文件声明了与此部署文件关联的 CloudFormation 堆栈的资源。

- `parameters`

  参数字典包含键值对，用于配置堆栈中的资源。堆栈部署文件最多可包含 50 个参数。

- `tags`

  标签字典包含可选的键值对，可用于识别堆栈中的资源并对其进行分类。堆栈部署文件最多可包含 50 个标签。

您可以提供自己的堆栈部署文件，也可以让 Git 同步为您创建一个，然后自动向存储库提交拉取请求。您可以通过编辑堆栈部署文件，将更改提交到存储库来管理参数和标签。

下面是 Git 同步堆栈部署文件的示例：

```
template-file-path: fargate-srvc/my-stack-template.yaml

parameters:
    image: public.ecr.aws/lts/nginx:latest
    task_size: x-small
    max_capacity: 5
    port: 8080
    env: production
tags:
    cost-center: '123456'
    org: 'AWS'
```

CloudFormation 模板文件

模板文件包含构成 CloudFormation 堆栈的 Amazon 资源的声明。通过 Git 同步，模板文件将存储在 Git 存储库中，并由堆栈部署文件引用。您可以通过编辑堆栈部署文件，将更改提交到存储库来管理堆栈。

有关更多信息，请参阅 使用 CloudFormation 模板。

模板定义存储库

模板定义存储库是通过 Git 同步链接到 CloudFormation 的 Git 存储库。监控存储库中 CloudFormation 模板和堆栈部署文件的更改。提交文件更改后，关联的堆栈会自动更新。

> **Important**
>
> 在 Git 同步控制台中配置模板定义存储库时，请从 Git 连接中选择正确的存储库和分支。Git 同步仅监控配置的存储库和分支，了解 CloudFormation 模板和堆栈部署文件的更改。

使用 Git 同步使堆栈与 Git 存储库同步的先决条件

在使 CloudFormation 堆栈与 Git 存储库同步之前，确认满足以下要求。

主题

- Git 存储库
- CloudFormation 模板
- Git 同步服务角色
- 控制台用户的 IAM 权限

Git 存储库

您必须将 Git 存储库托管在以下平台之一上。

- （GitHub）
- GitHub Enterprise
- GitLab
- Bitbucket
- GitLab 自托管

存储库可以是公有的，也可以是私有的。您需要通过 Connections 控制台将 Git 存储库连接到 CloudFormation。

CloudFormation 模板

您的 Git 存储库必须包含 CloudFormation 模板文件，且该文件已签入您打算连接到 Git 同步的分支。堆栈部署文件将引用此模板。

Git 同步服务角色

Git 同步需要 IAM 角色。您可以选择在配置 Git 同步时为堆栈创建 IAM 角色，也可以使用现有角色。

> **Note**
>
> 自动生成的 IAM 角色仅将权限应用于为其生成角色的堆栈。要重复使用自动生成的 IAM 角色，必须编辑新堆栈的角色。

Git 同步服务角色的必要权限

为 Git 同步提供的 IAM 角色需要以下权限。

- cloudformation:CreateChangeSet
- cloudformation:DeleteChangeSet
- cloudformation:DescribeChangeSet
- cloudformation:DescribeStackEvents
- cloudformation:DescribeStacks
- cloudformation:ExecuteChangeSet
- cloudformation:ListChangeSets
- cloudformation:ValidateTemplate
- events:PutRule
- events:PutTargets

> **Note**
>
> 上述所需权限会自动添加到 Git 同步生成的 IAM 角色中。

以下示例 IAM 角色包括 Git 同步的必备权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SyncToCloudFormation",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:GetTemplate",
                "cloudformation:ListChangeSets",
                "cloudformation:ListStacks",
```

```
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PolicyForManagedRules",
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:PutTargets"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                "events:ManagedBy":
 ["cloudformation.sync.codeconnections.amazonaws.com"]
                }
            }
        },
        {
            "Sid": "PolicyForDescribingRule",
            "Effect": "Allow",
            "Action": "events:DescribeRule",
            "Resource": "*"
        }
    ]
}
```

信任策略

在创建角色以定义信任关系时，必须提供以下信任策略：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CfnGitSyncTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.sync.codeconnections.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

建议您使用 `aws:SourceArn` 和 `aws:SourceAccount` 条件键来防止出现混淆代理人问题。源账户是您的账户 ID，源 ARN 是 Amazon CodeConnections 服务中允许 CloudFormation 连接到您的 Git 存储库的连接的 ARN。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CfnGitSyncTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.sync.codeconnections.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:codeconnections:us-
east-1:123456789012:connection/EXAMPLE64-8aad-4d5d-8878-dfcab0bc441f"
        }
      }
    }
  ]
}
```

有关混淆代理人问题的更多信息，请参阅 防止跨服务混淆代理。

控制台用户的 IAM 权限

要通过 CloudFormation 控制台成功设置 Git 同步，还必须通过 IAM 为最终用户授予权限。

创建和管理与 Git 存储库的连接需要以下 codeconnections 权限。

- codeconnections:CreateRepositoryLink
- codeconnections:CreateSyncConfiguration
- codeconnections:DeleteRepositoryLink
- codeconnections:DeleteSyncConfiguration
- codeconnections:GetRepositoryLink
- codeconnections:GetSyncConfiguration
- codeconnections:ListRepositoryLinks
- codeconnections:ListSyncConfigurations
- codeconnections:ListTagsForResource
- codeconnections:TagResource
- codeconnections:UntagResource
- codeconnections:UpdateRepositoryLink
- codeconnections:UpdateSyncBlocker
- codeconnections:UpdateSyncConfiguration
- codeconnections:UseConnection

在 Git 同步设置过程中，控制台用户还必须拥有以下 cloudformation 权限才能查看和管理堆栈。

- cloudformation:CreateChangeSet
- cloudformation:DeleteChangeSet
- cloudformation:DescribeChangeSet
- cloudformation:DescribeStackEvents
- cloudformation:DescribeStacks
- cloudformation:ExecuteChangeSet
- cloudformation:GetTemplate
- cloudformation:ListChangeSets
- cloudformation:ListStacks
- cloudformation:ValidateTemplate

> **Note**
>
> 尽管对于仅控制台使用场景，变更集权限（cloudformation:CreateChangeSet、cloudformation:DeleteChangeSet、cloudformat
> 可能不是硬性要求，但仍建议授予这些权限，以启用全栈检查和管理功能。

以下示例 IAM 策略包含通过控制台设置 Git 同步所需的用户权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeConnectionsPermissions",
            "Effect": "Allow",
            "Action": [
                "codeconnections:CreateRepositoryLink",
                "codeconnections:CreateSyncConfiguration",
                "codeconnections:DeleteRepositoryLink",
                "codeconnections:DeleteSyncConfiguration",
                "codeconnections:GetRepositoryLink",
                "codeconnections:GetSyncConfiguration",
                "codeconnections:ListRepositoryLinks",
                "codeconnections:ListSyncConfigurations",
                "codeconnections:ListTagsForResource",
                "codeconnections:TagResource",
                "codeconnections:UntagResource",
                "codeconnections:UpdateRepositoryLink",
                "codeconnections:UpdateSyncBlocker",
                "codeconnections:UpdateSyncConfiguration",
                "codeconnections:UseConnection",
                "codeconnections:CreateForcedTargetSync",
                "codeconnections:CreatePullRequestForResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudFormationConsolePermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:GetTemplate",
                "cloudformation:ListChangeSets",
                "cloudformation:ListStacks",
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 创建包含 `codeconnections:CreateForcedTargetSync` 和 `codeconnections:CreatePullRequestForResource` 权限的 IAM 策略时，您可能会在 IAM 控制台中看到一条警告，指出这些操作不存在。可以忽略此警告，策略仍将成功创建。某些 Git 同步操作需要这些权限，尽管 IAM 控制台可能无法识别它们。

使用 Git 同步从存储库源代码创建堆栈

本主题介绍如何创建使用 Git 同步与 Git 存储库同步的 CloudFormation 堆栈。

> **Important**
>
> 在继续之前，请完成上一节中的所有先决条件。

从存储库源代码创建堆栈

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择 Amazon Web Services 区域以在其中创建堆栈。

3. 在堆栈页面上，选择创建堆栈，然后选择使用新资源（标准）。
4. 在创建堆栈页面上，执行以下操作：
   a. 对于先决条件 - 准备模板，保持选中选择现有模板。
   b. 对于指定模板，选择与 Git 同步，然后选择下一步。
5. 在指定堆栈详细信息页面上，对于堆栈名称，请键入堆栈的名称。堆栈名称可包含字母（A-Z 和 a-z）、数字（0-9）和短划线（-）。
6. 对于堆栈部署文件、部署文件创建：
   - 如果您尚未创建堆栈部署文件并将其添加到存储库中，请选择使用以下参数创建文件并将其放入我的存储库。
   - 如果您的存储库中有堆栈部署文件，请选择我提供我存储库中的自有文件。
7. 对于模板定义存储库，请选择选择已链接的 Git 存储库以选择已经链接到 CloudFormation 的 Git 存储库，或者选择链接 Git 存储库以链接新存储库。如果您选择链接 Git 存储库，请执行以下操作：
   a. 在选择存储库提供商中，选择以下选项之一：
      - （GitHub）
      - GitHub Enterprise Server
      - GitLab
      - Bitbucket
      - GitLab 自托管
   b. 对于连接，从列表中选择一个连接。如果连接列表中没有显示任何选项，请选择添加新连接以转到 Connections 控制台，然后创建与您存储库的连接。
8. 在存储库列表中，选择包含堆栈模板文件的 Git 存储库。
9. 在分支列表中，选择希望 Git 同步监控的分支。

> **Note**
>
> Git 同步仅监控所选分支中对 CloudFormation 模板和堆栈部署文件的更改。要应用到堆栈的任何更改都必须提交到此分支。

10. 对于部署文件路径，指定完整路径，包括存储库分支根目录下的堆栈部署文件名。

如果 CloudFormation 正在为您生成文件，那么这就是文件在存储库中提交的位置。如果您提供文件，那么这就是文件在存储库中的位置。

11. 添加 IAM 角色。IAM 角色包含 CloudFormation 从 Git 存储库同步堆栈所需的权限。您可以选择新建 IAM 角色来生成新角色，也可以选择现有 IAM 角色以选择 Amazon Web Services 账户中的现有角色。如果选择生成新角色，该角色中将包含所需的权限。

    > **Note**
    >
    > 如果您的账户中启用了角色管理器，CloudFormation 会为您附加一个服务角色，此处所述的现有 IAM 角色选择将替换为自定义选项。要使用其它角色，请选择自定义。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色创建。

12. 启用或关闭拉取请求注释：

    - 要让 CloudFormation 在堆栈更新的拉取请求中发布更改集信息，请将启用拉取请求注释开关保持开启状态。
    - 如果您关闭此开关，CloudFormation 将不会描述当前堆栈配置与更新存储库文件拉取请求中的建议更改之间的区别。

13. 对于模板文件路径，指定堆栈模板文件在存储库根目录下的完整路径。
14. （可选）要指定堆栈参数，请选择添加参数，为每个参数提供键和值，然后选择下一步。有关更多信息，请参阅 堆栈部署文件。

    例如，要在堆栈部署文件中指定 **`port=8080`** 参数，请执行以下操作：

    a. 选择 Add parameter (添加参数)。

    b. 对于键，输入 **`port`**。

    c. 对于值，输入 **`8080`**。

15. （可选）要指定堆栈标签，请选择添加新标签，为每个标签提供标签键和值，然后选择下一步。有关更多信息，请参阅 堆栈部署文件。
16. 选择下一步，继续配置堆栈选项。有关配置堆栈选项的信息，请参阅 配置堆栈选项。

    完成堆栈配置后，选择下一步以继续。

17. 查看堆栈设置并确认以下内容：

    - 堆栈模板已正确配置并设置为从 Git 同步。
    - 部署文件配置正确。

- 模板定义存储库配置正确，尤其是选择了正确的存储库和分支名称。
- 部署文件的预览正确，包含预期的参数和值。

18. 选择提交以创建堆栈。

    选择提交后，系统会在您的 Git 存储库中自动创建一条拉取请求。必须将此拉取请求合并到您的 Git 存储库中才能创建堆栈。创建堆栈后，CloudFormation 会监控您的 Git 存储库中的更改。

从 Git 存储库更新堆栈

要更新堆栈，请直接更改模板文件或 Git 存储库中的堆栈部署文件。在将更改提交到受监控的分支后，CloudFormation 会自动更新堆栈。如果您使用拉取请求，则系统会在 Stack 更新之前在 Git 存储库中自动创建拉取请求。必须将此拉取请求合并到您的 Git 存储库中才能更新您的堆栈。

在 CloudFormation 控制台中，您可以选择堆栈，然后选择 Git 同步选项卡，以查看有关堆栈状态和同步事件的信息。有关更多信息，请参阅 Git 同步状态控制面板。

启用 CloudFormation 在拉取请求中发布堆栈更改摘要

本主题展示了如何让 CloudFormation 在 Git 存储库的发布拉取请求中发布堆栈更改摘要。

通过启用拉取请求注释功能，您可以允许 CloudFormation 发布注释，描述当前堆栈配置与更新存储库文件时的建议更改之间的差异。此注释提供了将要添加、修改或删除的资源的摘要，让您能够在合并拉取请求之前进行全面的代码审查。

为新堆栈启用拉取请求注释（控制台）

创建堆栈时，在指定堆栈详细信息页面的模板定义存储库下，确保启用拉取请求注释开关处于启用状态。对于新堆栈这是默认设置。

为现有堆栈启用拉取请求注释（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您在其中创建了堆栈的 Amazon Web Services 区域。
3. 在堆栈页面上，选择要更新的正在运行的堆栈。
4. 选择 Git sync 选项卡，然后选择编辑。
5. 在编辑 Git sync 设置页面的模板定义存储库下，打开启用拉取请求注释开关。
6. 选择 Update configuration (更新配置)。

Git 同步状态控制面板

要查看 Amazon CloudFormation Git 同步部署的状态，请在 CloudFormation 控制台中选择堆栈，然后选择 Git 同步选项卡。

Git 同步选项卡分为两个面板：Git 同步状态和最新同步事件。

Git 同步状态

顶部面板提供了有关堆栈的 Git 同步配置的以下信息。

存储库

指向连接到 Git 同步的存储库的链接

存储库提供商

存储库提供商的名称

分支

Git 同步监控的分支的名称

部署文件路径

堆栈的堆栈部署文件的完整路径

存储库同步状态

最近一次同步操作的状态

存储库同步状态消息

最近一次同步操作的消息

Git 同步状态

堆栈的 Git 同步状态

预置状态

置预置作的状态

在面板的右上角，使用以下按钮修改或更新 Git 同步：

- 编辑 - 编辑 Git 同步配置。
- 重试最新提交 - 根据对存储库的最新提交更新堆栈。

- 断开连接 - 断开 Git 同步与堆栈的连接。
- 刷新 - 刷新 Git 同步状态面板。

最新同步事件

最新同步事件面板显示应用于堆栈的提交表。

您可以使用每列标题中的箭头对表进行排序。该表可根据以下内容按升序或降序排序：

- 日期
- 提交 ID
- Event (事件)
- 日期
- 事件类型

支持的堆栈状态

只有当堆栈处于以下受支持状态之一时，才能为堆栈配置 Git 同步：

- CREATE_COMPLETE
- UPDATE_COMPLETE
- UPDATE_ROLLBACK_COMPLETE
- IMPORT_COMPLETE
- IMPORT_ROLLBACK_COMPLETE

下表包含堆栈状态代码的完整列表以及描述：

堆栈状态和可选的详细状态	描述
CREATE_COMPLETE	成功创建一个或多个堆栈。
CREATE_IN_PROGRESS	正在创建一个或多个堆栈。
CREATE_FAILED	一个或多个堆栈创建失败。查看堆栈事件可了解所有相关错误消息。创建失败的可能原因包括：没有足够的权限使用堆栈中

堆栈状态和可选的详细状态	描述
	的所有资源，参数值被 Amazon 服务拒绝，或者在资源创建期间超时。
DELETE_COMPLETE	成功删除一个或多个堆栈。已删除的堆栈在 90 天内保留可供查看。
DELETE_FAILED	一个或多个堆栈删除失败。由于删除失败，您可能有一些资源仍在运行，但是您无法使用或更新堆栈。再次删除堆栈或查看堆栈事件可了解所有相关错误消息。
DELETE_IN_PROGRESS	正在删除一个或多个堆栈。
REVIEW_IN_PROGRESS	正在创建一个或多个具有预期 StackId 但没有任何模板或资源的堆栈。 **Important** 具有此状态代码的堆栈将针对可能的最大堆栈数量进行计数。
ROLLBACK_COMPLETE	在堆栈创建失败或明确取消堆栈创建后成功删除一个或多个堆栈。堆栈返回到之前的工作状态。在创建堆栈操作期间创建的所有资源都被删除。 此状态仅在堆栈创建失败后存在。它表示已适当清除未完全创建的堆栈中的所有操作。在此状态下，只能执行删除操作。
ROLLBACK_FAILED	在堆栈创建失败或明确取消堆栈创建后删除一个或多个堆栈失败。删除堆栈或查看堆栈事件了解所有相关错误消息。
ROLLBACK_IN_PROGRESS	在堆栈创建失败或明确取消堆栈创建后正在删除一个或多个堆栈。
UPDATE_COMPLETE	成功更新一个或多个堆栈。

堆栈状态和可选的详细状态	描述
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	成功更新一个或多个堆栈后正在删除堆栈的旧资源。对于需要替换资源的堆栈更新，CloudFormation 首先创建新资源，然后删除旧资源，以帮助减少堆栈中断。在这种状态下，堆栈已更新并且可用，但 CloudFormation 仍然会删除旧资源。
UPDATE_FAILED	未成功更新一个或多个堆栈。查看堆栈事件可了解所有相关错误消息。
UPDATE_IN_PROGRESS	正在更新一个或多个堆栈。
UPDATE_ROLLBACK_COMPLETE	在堆栈更新失败后将一个或多个堆栈成功返回之前的工作状态。
UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS	堆栈更新失败后正在删除一个或多个堆栈的新资源。在这种状态下，堆栈已回滚到之前的工作状态并且可用，但 CloudFormation 仍会删除它在堆栈更新期间创建的所有新资源。
UPDATE_ROLLBACK_FAILED	在堆栈更新失败后将一个或多个堆栈返回之前的工作状态失败。在这种状态下，您可以删除堆栈或继续回滚。您可能需要修复错误，然后堆栈才能返回工作状态。您也可以联系 Amazon Web Services 支持 将堆栈恢复到可用状态。
UPDATE_ROLLBACK_IN_PROGRESS	堆栈更新失败后正在将一个或多个堆栈返回之前的工作状态。
IMPORT_IN_PROGRESS	当前正在执行导入操作。
IMPORT_COMPLETE	对于堆栈中支持 `resource import` 的所有资源，已成功完成导入操作。
IMPORT_ROLLBACK_IN_PROGRESS	导入将回滚到以前的模板配置。
IMPORT_ROLLBACK_FAILED	对于堆栈中的至少一个资源，导入回滚操作失败。将提供 CloudFormation 成功导入的资源的结果。

堆栈状态和可选的详细状态	描述
IMPORT_ROLLBACK_COMPLETE	导入成功回滚到以前的模板配置。

通过 CloudFormation 注册表管理扩展

Amazon CloudFormation 注册表作为一个集中式中心，用于管理可以集成到您的 Amazon Web Services 账户中的 CloudFormation 模板的扩展。扩展包括来自 Amazon 和第三方发布者的资源类型、模块和挂钩，以及您自己的自定义扩展。利用注册表，您可以像使用 Amazon 提供的资源一样，更轻松地在 CloudFormation 模板中发现和预置扩展。

此部分介绍如何使用 CloudFormation 注册表来激活账户中的第三方扩展，包括：

- 激活公有扩展
- 注册和激活私有扩展

主题

- 相关文档
- CoudFormation 注册表概念
- 查看 CloudFormation 注册表中可用和已激活的扩展
- 使用 CloudFormation 注册表中的第三方公有扩展
- 使用已与您共享的第三方私有扩展
- 编辑账户中扩展的配置数据
- 在 Amazon Config 中记录资源类型

相关文档

如果您是开发人员，有兴趣创建自己的扩展程序，请参阅以下文档：

- 《CloudFormation Command Line Interface User Guide》中的 Developing modules using the CloudFormation CLI
- 《CloudFormation Command Line Interface User Guide》中的 Creating resource types using the CloudFormation CLI
- 《Amazon CloudFormation Hooks User Guide》中的 Developing custom Hooks using the CloudFormation CLI

此外，CloudFormation 注册表中可用的所有可预置 Amazon 资源类型均可与 Amazon 云端控制 API 一起使用，其属性和特性在标准 JSON 架构中定义。有关更多信息，请参阅《Cloud Control API 用户指

南》。使用 Cloud Control API 对 Amazon 资源执行 CRUDL（创建、读取、更新、删除、列表）操作时，您只能对自己 Amazon Web Services 账户 中的 Amazon 源执行此操作。

CoudFormation 注册表概念

本主题介绍关键概念，帮助您了解并开始使用 CloudFormation 注册表。

扩展类型

CloudFormation 提供以下扩展类型：

挂钩

挂钩是一种验证检查，可在 CloudFormation 创建、更新或删除堆栈或特定资源之前对其进行检查。此外，还可以在创建更改集操作期间调用挂钩。它们提供了一种机制，通过根据特定需求验证资源配置来强制执行组织标准和最佳实践。如果挂钩检测到任何配置不符合其挂钩逻辑，则它可能会发出警告，或者使预置过程失败，以防止部署不合规的资源。有关更多信息，请参阅《Amazon CloudFormation 挂钩用户指南》。

有关如何使用 CloudFormation 控制台配置 Hooks 的文档，请参阅《Amazon CloudFormation Hooks 用户指南》的以下部分：

- Amazon Control Tower 主动控制作为 Hooks
- Guard 挂钩
- Lambda 挂钩

模块

模块是可跨多个 CloudFormation 堆栈模板包含的可重复使用的资源配置。它们将复杂或常用的资源配置封装为可重复使用的组件，从而简化 CloudFormation 模板的创建和维护。这可以促进组织基础设施部署的一致性和标准化。

资源类型

资源类型让您可以对 CloudFormation 本身不支持的第三方资源或自定义资源进行建模和自动化。通过开发资源类型，您可以扩展 CloudFormation 的功能，以预置和管理来自各种第三方服务的资源。

公有扩展

公有扩展是在注册表中公开发布以供所有 Amazon Web Services 账户 使用的 CloudFormation 扩展。这些指令包括：

- Amazon 扩展 - Amazon 发布的扩展始终是公开的，默认情况下已激活，因此您无需采取任何措施即可在账户中使用这些扩展。Amazon 控制这些扩展的版本，因此您始终可以使用最新的版本。
- 第三方扩展 - 这些扩展可供 Amazon 以外的发布者通用。要使用第三方公有扩展，您必须首先在账户和区域中激活该扩展。

已激活扩展

您特定 Amazon Web Services 账户 中的 CloudFormation 注册表包括三种类型的已激活扩展：

- Amazon 扩展 - 所有 Amazon 公有扩展程序都会自动激活。
- 已激活第三方扩展 - 这些是您已明确为您的账户和区域激活的第三方公有扩展的本地副本。激活第三方公有扩展时，CloudFormation 会在您账户的注册表中创建一个本地副本。有关更多信息，请参阅使用 CloudFormation 注册表中的第三方公有扩展。
- 私有注册扩展 - 这些是未在公有 CloudFormation 注册表中列出的私有扩展。这些可能是您自己创建的扩展，也可能是贵组织或其他第三方与您共享的扩展。要在您的账户中使用此类私人扩展，您必须先注册该扩展。注册扩展时会将副本上传到您账户中的 CloudFormation 注册表并激活它。有关更多信息，请参阅 使用已与您共享的第三方私有扩展。

在账户中使用私有注册扩展和第三方发布者激活的公有扩展，就像在沙盒环境中使用它们一样。扩展通过版本控制进行管理，因此其配置行为与特定版本相关联。因此，这些扩展的功能与公有扩展相同，都遵守特定版本的规则。

> **Note**
>
> 私有注册扩展和已激活第三方公有扩展可实现在创建、读取、更新、列表和删除操作期间运行的事件处理程序。在 CloudFormation 堆栈中使用这些扩展可能会对您的帐户产生费用，此外还可能对创建的资源产生费用。有关更多信息，请参阅Amazon CloudFormation定价。

查看 CloudFormation 注册表中可用和已激活的扩展

要查看 CloudFormation 注册表中可用和已激活的扩展，您可以使用 Amazon Web Services 管理控制台 或 Amazon CLI。

查看可用和已激活的扩展（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 从导航窗格的注册表下，选择要查看的扩展类别：
   - Public extensions（公有扩展）显示账户中可用的公有扩展。
     1. 在筛选条件、扩展类型中，选择您的扩展类型：资源类型、模块或挂钩。
     2. 在筛选条件、发布者中，选择发布者：Amazon 或第三方。
   - Activated extensions（已激活的扩展）显示账户中已激活的公有和私有扩展。
     1. 选择您的扩展类型：Resource types（资源类型）、Modules（模块）或 Hooks（挂钩）。
     2. 使用筛选条件下拉菜单进一步选择要查看的扩展：
        - Amazon - 列出 Amazon 发布的扩展。Amazon 发布的扩展默认已激活。
        - Third-party（第三方）- 列出此账户中已激活、除 Amazon 以外任何发布者的所有公有扩展。
        - Registered（已注册）- 列出此账户中已激活的所有私有扩展。
   - Publisher（发布者）显示使用此账户已发布的所有公有扩展。
4. 搜索或选择扩展名称以查看扩展详细信息。

查看可用和已激活的扩展（Amazon CLI）

使用 list-types 命令。

使用 CloudFormation 注册表中的第三方公有扩展

要在模板中使用第三方公有扩展，您必须首先为要使用它的账户和区域激活该扩展。激活扩展使其在激活它的账户和区域的堆栈操作中可用。

激活第三方公有扩展时，CloudFormation 会在账户的扩展注册表中为激活的扩展创建一个条目作为私有扩展。这允许您设置该扩展包含的任何配置属性。配置属性定义如何为给定 Amazon Web Services 账户和区域配置扩展。

除了设置配置属性外，您还可以通过以下方式自定义扩展：

- 指定 CloudFormation 用于激活扩展的执行角色，以及为扩展配置日志记录。
- 指定在新的次要版本或补丁版本可用时是否自动更新扩展。
- 指定要使用的别名，而不是第三方公有扩展名称。这有助于避免第三方扩展之间的命名冲突。

主题

- 配置具有 IAM 权限的执行角色和公有扩展访问权限的信任策略
- 自动使用新版本的扩展
- 使用别名以引用扩展
- 处理公有扩展的常用 Amazon CLI 命令
- 激活账户中的第三方公有扩展
- 更新账户中的第三方公有扩展
- 停用账户中的第三方公有扩展

配置具有 IAM 权限的执行角色和公有扩展访问权限的信任策略

从 CloudFormation 注册表中激活公有扩展时，您可以提供一个执行角色，该角色向 CloudFormation 授予在您的 Amazon Web Services 账户和区域中调用该扩展所需的权限。

该执行角色所需的权限在扩展架构的处理程序部分中定义。您必须创建 IAM 策略来授予扩展所需的特定权限，并将其附加到执行角色。

除权限策略外，该执行角色还需要具有允许 CloudFormation 承担角色的信任策略。按照《IAM 用户指南》中使用自定义信任策略创建角色的指导，使用自定义信任策略创建角色。

信任关系

下面所示为您可以使用的示例信任策略。

您可以选择使用带有 `Condition` 字段的一个或多个全局条件上下文键来限制权限范围，以防止跨服务混淆代理。有关更多信息，请参阅 防止跨服务混淆代理。

- 将 aws:SourceAccount 值设置为您的账户 ID。
- 将 aws:SourceArn 值设置为扩展的 ARN。

示例信任策略 1

以下是资源类型扩展的示例 IAM 角色信任策。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "resources.cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
west-2:123456789012:type/resource/Organization-Service-Resource"
                }
            }
        }
    ]
}
```

示例信任策略 2

以下是挂钩扩展的示例 IAM 角色信任策略。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "resources.cloudformation.amazonaws.com",
                    "hooks.cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
west-2:123456789012:type/hook/Organization-Service-Hook"
                }
            }
        }
    ]
}
```

自动使用新版本的扩展

激活扩展时，您还可以指定扩展类型以使用最新的次要版本。每次发布者发布您激活的扩展的新版本时，您的扩展类型都会更新次要版本。

例如，下次使用包含该扩展的模板执行堆栈操作（例如创建或更新堆栈）时，CloudFormation 将使用新的次要版本。

自动或手动更新为新的扩展版本，不会影响堆栈中已预置的任何扩展实例。

CloudFormation 将扩展的主要版本更新视为可能包含重大更改，因此需要您手动更新为扩展的新主要版本。

默认情况下，Amazon 发布的扩展将为其可用的所有账户和区域激活，并且始终使用每个 Amazon Web Services 区域中可用的最新版本。

> **Important**
>
> 由于您可以控制扩展是否以及何时更新为账户中的最新版本，因此最终可能会在不同的账户和区域部署同一扩展的不同版本。

跨这些账户和区域使用包含该扩展的相同模板时，可能会导致意外结果。

使用别名以引用扩展

您不能激活给定 Amazon Web Services 账户和区域中具有给定名称的多个扩展。由于不同的发布者可能提供具有相同扩展名称的公有扩展，CloudFormation 可让您为激活的任何第三方公有扩展指定别名。

如果指定扩展的别名，则 CloudFormation 会将别名视为该账户和区域内的扩展类型名称。必须使用别名来引用模板、API 调用和 CloudFormation 控制台中的扩展。

扩展别名在指定的账户和区域内必须唯一。您可以使用不同的类型名称别名在同一账户和区域中多次激活同一公有资源。

Important

虽然扩展别名只要求在给定账户和区域中唯一，但强烈建议用户不要为跨账户和区域的不同第三方公有扩展分配相同的别名。使用包含跨多个账户或区域的扩展别名的模板时，这样做可能会导致意外结果。

处理公有扩展的常用 Amazon CLI 命令

处理公有扩展的常用命令包括：

- activate-type 可在账户中激活公有第三方模块或资源类型。
- set-type-configuration 可指定您账户中扩展的配置数据，以及禁用和启用挂钩。
- list-types 可列出您账户中的扩展。
- describe-type 可返回有关特定扩展或特定扩展版本的详细信息，包括当前配置数据。
- set-type-default-version 可指定哪个扩展版本是默认版本。
- deactivate-type 可停用先前在您账户中激活的公有第三方模块或资源类型。

激活账户中的第三方公有扩展

以下主题向您展示如何在您的账户中激活第三方公有扩展，从而使其在激活该扩展的账户和区域中可用。

> **Note**
>
> 在继续操作之前，请确认您已创建将与此扩展一起使用的 IAM 角色。

主题

- 激活公有扩展（控制台）
- 激活公有扩展（Amazon CLI）

激活公有扩展（控制台）

请按照本部分中的步骤，使用控制台来执行以下操作：

- 激活第三方公有扩展
- 为您的账户指定其他扩展配置数据

激活公有扩展以在您的账户中使用

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 从导航窗格的注册表下，选择公共扩展程序。
4. 使用筛选条件选择扩展类型，然后选择第三方。（Amazon 发布的扩展默认已激活。）
5. 选择扩展，然后选择激活。

   如果有多个版本的扩展可用，则您可以使用版本菜单选择要激活的扩展版本。默认为最新版本。
6. 对于扩展程序名称，您可以保留使用默认设置选项，或选择覆盖默认设置，然后输入要与此扩展程序一起使用的扩展程序类型别名。别名必须遵循扩展类型的推荐格式。有关更多信息，请参阅 使用别名以引用扩展。
7. 如果要激活的扩展程序是钩子或资源类型，那么对于执行角色 ARN，请指定调用扩展程序时 CloudFormation 要承担的 IAM 角色。有关更多信息，请参阅 配置具有 IAM 权限的执行角色和公有扩展访问权限的信任策略。
8. 对于日志配置，如果需要，则为扩展程序指定日志配置信息。例如：

```
{
    "logRoleArn": "arn:aws:iam::account:role/rolename",
```

```
    "logGroupName": "log-group-name"
}
```

日志记录配置信息并非必需，但建议用于调试目的。要使用带有挂钩的日志记录配置，请添加与指定的执行角色相同的信任策略，以便日志角色可以将日志写入日志组。

`logRoleArn` 和 `logGroupName` 键名称应区分大小写。

9. 对于版本控制、自动更新，请选择接收更新的方式。

   - On（开）- 自动更新为最新次要版本。主要版本手动更新。
   - Off（关）- 永不自动更新为最新版本。所有版本均手动更新。

   有关更多信息，请参阅 自动使用新版本的扩展。

如果扩展需要其他配置，则您可以选择现在指定配置数据，也可以在扩展激活后指定该数据。

> **Important**
>
> 如果您要激活的扩展是挂钩，则必须执行此步骤。您必须为 `HookInvocationStatus` 属性指定 `ENABLED`。此操作启用挂钩架构 `properties` 部分中定义的挂钩属性。有关更多信息，请参阅《Amazon CloudFormation Hooks User Guide》中的 Hooks configuration schema syntax reference。

指定配置数据

1. 对于配置，选择立即配置，然后选择激活扩展。

   CloudFormation 将显示 Configure extension（配置扩展）页面。要查看扩展的当前配置架构，请确保 View configuration schema（查看配置架构）已激活。

2. 在配置 JSON 文本框中，输入一个 JSON 字符串，该字符串表示要为此扩展指定的配置数据。您指定的 JSON 必须根据扩展的配置架构进行验证。
3. 选择配置扩展程序。

如果您希望在激活后配置扩展，则可以跳过此步骤并稍后提供配置数据。

1. 对于配置，选择稍后配置，然后选择激活扩展。

2. 激活扩展后，您可以通过从已激活的扩展页面导航到该扩展并提供配置数据以对其进行配置。

激活公有扩展（Amazon CLI）

请按照本部分中的步骤，使用 Amazon CLI 来执行以下操作：

- 激活第三方公有扩展
- 为您的账户指定其他扩展配置数据

激活公有挂钩

在您的账户中激活挂钩，即授权挂钩使用您的 Amazon Web Services 账户中定义的权限。CloudFormation 会删除不需要的权限，然后再将您的权限传递给挂钩。CloudFormation 建议，客户或挂钩用户在您的账户中激活挂钩之前，先查看挂钩权限并了解允许挂钩使用的权限。

激活公有挂钩以在您的账户中使用（Amazon CLI）

1. 获取并保存挂钩的 ARN。您可以使用 Amazon Web Services 管理控制台 或 Amazon CLI 获取挂钩的 ARN。有关更多信息，请参阅 查看 CloudFormation 注册表中可用和已激活的扩展。

```
export HOOK_TYPE_ARN="arn:aws:cloudformation:us-west-2:123456789012:type/hook/
Organization-Service-Hook/"
```

2. 使用 set-type-configuration 命令指定配置数据。为 `--configuration` 传递的 JSON 必须根据挂钩的配置架构进行验证。要激活所有堆栈操作的挂钩，必须在 `HookConfiguration` 部分中将 `HookInvocationStatus` 属性设置为 `ENABLED`。

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL", "Properties":{}}}}" \
  --type-arn $HOOK_TYPE_ARN --region us-west-2
```

有关 `HookConfiguration` 配置选项的更多信息，请参阅《Amazon CloudFormation Hooks User Guide》中的 Hook configuration schema syntax reference。

激活公有模块和资源类型

激活公有扩展以在您的账户中使用 (Amazon CLI)

- 使用 activate-type 命令来激活扩展，并指定在发布扩展的新次要版本时是否自动更新扩展。

  下面的示例指定要为此账户激活的公有扩展的公共 Amazon 资源名称 (ARN)。此外，它指定 CloudFormation 在每次发布新的次要版本时更新扩展。

```
aws cloudformation activate-type \
  --public-type-arn public_extension_ARN \
  --execution-role-arn arn:aws:iam::123456789012:role/my-execution-role \
  --auto-update true --region us-west-2
```

  此命令返回已激活扩展的 ARN。

```
{
    "Arn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-
Resource-Example"
}
```

更新公有扩展的版本 (Amazon CLI)

使用 activate-type 再次激活该扩展。

使用 `--version-bump` 选项可指定是否将扩展更新为最新 `MAJOR` 版本或最新 `MINOR` 版本。

```
aws cloudformation activate-type --type RESOURCE \
  --type-name Example::Test::1234567890abcdef0 \
  --type-name-alias Example::Test::Alias \
  --version-bump MAJOR --region us-west-2
```

更新账户中的第三方公有扩展

激活第三方公有扩展后，您可以从账户中更新大部分扩展详细信息。

要更新账户中的公有扩展（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。

2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 从导航窗格的注册表下，选择已激活的扩展程序。
4. 找到您要更新的扩展，然后将其选中。有关更多信息，请参阅 查看 CloudFormation 注册表中可用和已激活的扩展。
5. 从操作菜单，选择编辑，然后选择适用的编辑选项：

   - 要更新配置架构，请参阅 编辑账户中扩展的配置数据。
   - 激活或停用自动更新：
     1. 选择编辑自动更新。
     2. 选择开或关，然后选择保存。有关更多信息，请参阅 自动使用新版本的扩展。
   - 更新执行角色：
     1. 选择编辑执行角色。
     2. 指定调用此扩展时希望 CloudFormation 使用的 IAM 角色的 ARN，然后选择保存。有关更多信息，请参阅 配置具有 IAM 权限的执行角色和公有扩展访问权限的信任策略。
   - 更新日志记录配置：
     1. 选择编辑日志记录配置。
     2. 编辑日志记录配置 JSON，然后选择保存。

停用账户中的第三方公有扩展

当您不再需要已激活的第三方公有扩展时，请按照以下步骤在您的账户中将其停用。

主题

- 停用账户中的公有扩展（控制台）
- 停用账户中的公有扩展（Amazon CLI）
- 在您的账户中停用挂钩（Amazon CLI）

停用账户中的公有扩展（控制台）

停用账户中的公有扩展

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。

3. 从导航窗格的注册表下，选择已激活的扩展程序。
4. 找到您想要停用的扩展，然后选中它。有关更多信息，请参阅 查看 CloudFormation 注册表中可用和已激活的扩展。
5. 从操作菜单中，选择停用。
6. 选择停用。

停用账户中的公有扩展（Amazon CLI）

使用以下 deactivate-type 命令。

```
aws cloudformation deactivate-type --type MODULE \
  --type-name Example::Test::Type::MODULE \
  --region us-west-2
```

在您的账户中停用挂钩（Amazon CLI）

停用挂钩可阻止挂钩在 Amazon Web Services 账户中运行，而无需将其移除。

使用 set-type-configuration 命令并将 `HookInvocationStatus` 指定为 `DISABLED` 以停用挂钩。

以下示例指定要禁用的挂钩的 Amazon Web Services 区域和 Amazon 资源名称（ARN）。

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "DISABLED", "FailureMode": "FAIL", "Properties":{}}}}" \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" --
region us-west-2
```

有关更多信息，请参阅《Amazon CloudFormation 挂钩用户指南》中的禁用和启用 Amazon CloudFormation 挂钩。

使用已与您共享的第三方私有扩展

要使用已与您共享的第三方私有扩展，您必须先使用 CloudFormation 在要使用这些扩展的账户和区域中进行注册。注册扩展会将其副本上传到您账户中的 CloudFormation 注册表并激活它。注册私有扩展后，其会在该 Amazon Web Services 账户和区域的 CloudFormation 注册表中显示，您可以在堆栈模板中使用该扩展。

主题

- 用于注册第三方私有扩展的 IAM 权限
- 处理私有扩展的常用 Amazon CLI 命令
- 在您的账户中注册第三方私有扩展。
- 从您的账户中删除第三方私有扩展

用于注册第三方私有扩展的 IAM 权限

作为注册私有扩展的一部分，您可指定一个包含扩展项目包的 Amazon S3 存储桶。此项目包中包含您要注册的扩展必需的所有源文件。注册扩展的用户必须能够访问 Amazon S3 存储桶中的项目包。为此，用户必须具有扩展包的 GetObject 权限。

无论您是使用 Amazon CLI 的 register-type 命令还是 CloudFormation CLI 的 submit 命令，都是如此。

有关更多信息，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

处理私有扩展的常用 Amazon CLI 命令

处理私有扩展的常用命令包括：

- register-type 可在您的账户中注册私人扩展。
- describe-type-registration 可返回注册请求的当前状态。
- list-types 可列出您账户中的扩展。
- describe-type 可返回有关特定扩展或特定扩展版本的详细信息，包括当前配置数据。
- set-type-configuration 可指定您账户中扩展的配置数据，以及禁用和启用挂钩。
- set-type-default-version 可指定哪个扩展版本是默认版本。
- deregister-type 可删除您账户中的私有扩展或扩展版本。

在您的账户中注册第三方私有扩展。

本主题介绍了注册与您共享的第三方私有扩展，以便能够在您的账户中使用的步骤。

> **Note**
>
> 在继续操作之前，请确认您拥有注册私有扩展所需的 IAM 权限。

注册与您共享的私有扩展（Amazon CLI）

1. 找到包含要在您的账户中注册的私有扩展项目包的 Amazon S3 存储桶。
2. 使用 register-type 命令在您的账户中注册私有扩展。

   例如，以下命令在指定的 Amazon Web Services 账户中注册 `My::Resource::Example` 资源类型。

   ```
   aws cloudformation register-type --type RESOURCE \
     --type-name My::Resource::Example \
     --schema-handler-package [s3 object path] --region us-west-2
   ```

   `RegisterType` 是一个异步操作，返回一个注册令牌，您可以使用该令牌跟踪注册请求进度。

   ```
   {
       "RegistrationToken": "f5525280-104e-4d35-bef5-8f1fexample"
   }
   ```

   如果您的扩展调用 Amazon API 作为其功能的一部分，您必须创建一个包含调用这些 Amazon API 所需的权限的 IAM 执行角色，并在您的账户中预置该执行角色。然后，您可以使用 `--execution-role-arn` 选项指定此执行角色。接下来，CloudFormation 担任该执行角色，以便为您的资源类型提供相应的凭证。

   ```
   --execution-role-arn arn:aws:iam::123456789012:role/MyIAMRole
   ```

3. （可选）将注册令牌与 describe-type-registration 命令一起使用以跟踪注册请求进度。

   在 CloudFormation 完成注册请求时，它将请求的进度状态设置为 `COMPLETE`。

   以下示例使用上面的 `describe-type-registration` 命令返回的注册令牌以返回注册状态信息。

   ```
   aws cloudformation describe-type-registration \
     --registration-token f5525280-104e-4d35-bef5-8f1fexample \
     --region us-west-2
   ```

   该命令将返回以下输出。

   ```
   {
   ```

```
    "ProgressStatus": "COMPLETE",
    "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-
Resource-Example",
    "Description": "Deployment is currently in DEPLOY_STAGE of status COMPLETED; ",
    "TypeVersionArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/
My-Resource-Example/00000001"
}
```

> **Important**
>
> 如果您要注册的扩展是挂钩，则必须执行下一步。您必须为 HookInvocationStatus 属性指定 ENABLED。此操作启用挂钩架构 properties 部分中定义的挂钩属性。有关更多信息，请参阅《Amazon CloudFormation Hooks User Guide》中的 Hooks configuration schema syntax reference。

指定挂钩的配置数据（Amazon CLI）

1. 获取并保存挂钩的 ARN。您可以使用 Amazon Web Services 管理控制台 或 Amazon CLI 获取挂钩的 ARN。有关更多信息，请参阅 查看 CloudFormation 注册表中可用和已激活的扩展。

   ```
   export HOOK_TYPE_ARN="arn:aws:cloudformation:us-west-2:123456789012:type/hook/
   Organization-Service-Hook/"
   ```

2. 使用 set-type-configuration 命令指定配置数据。为 --configuration 传递的 JSON 必须根据挂钩的配置架构进行验证。要激活挂钩，必须在 HookConfiguration 部分中将 HookInvocationStatus 属性设置为 ENABLED。

   ```
   aws cloudformation set-type-configuration \
     --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
   {"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL", "Properties":{}}}}" \
     --type-arn $HOOK_TYPE_ARN --region us-west-2
   ```

   有关更多信息，请参阅《Amazon CloudFormation Hooks User Guide》中的 Hooks configuration schema syntax reference。

从您的账户中删除第三方私有扩展

要移除第三方私有扩展或扩展版本，请使用 deregister-type 命令。

您可以取消注册特定的扩展版本或整个扩展。要取消注册扩展，您必须单独取消注册该扩展的所有已注册版本。如果扩展只有一个已注册的版本，则取消注册该版本会导致取消注册扩展本身。您无法取消注册扩展的默认版本，除非其是该扩展唯一的已注册版本，在这种情况下，扩展本身也会取消注册。

> **Warning**
>
> 取消注册私有扩展程序的操作无法撤消。此操作将会：
>
> - 使该扩展程序无法在所有 CloudFormation 操作中使用。
> - 导致使用此扩展程序的未来的堆栈更新失败（适用于模块和资源类型）。尽管您稍后可以自行重新注册该扩展程序，但如果 CloudFormation 依赖的是较早的版本，这样做可能会导致失败。
>
> 在继续操作之前，请使用 list-stacks 和 get-template 命令验证没有活动堆栈在使用此扩展程序。

取消注册扩展命令示例

本节提供的示例展示了取消注册私有扩展的不同方法。

按类型名称取消注册

使用带 `--type` 和 `--type-name` 选项的 deregister-type 命令取消注册您的扩展。

```
aws cloudformation deregister-type \
  --type MODULE \
  --type-name My::S3::SampleBucket::MODULE
```

按类型名称和版本取消注册

要取消注册扩展的特定版本，请在命令中指定 `--version-id` 选项。

```
aws cloudformation deregister-type \
  --type MODULE \
```

```
  --type-name My::S3::SampleBucket::MODULE \
  --version-id 00000001
```

> **Tip**
>
> 要先将扩展的不同版本设置为默认版本，请使用 set-type-default-version 命令。

通过 ARN 取消注册

使用 `--arn` 选项并指定扩展的 ARN 来取消注册。

```
aws cloudformation deregister-type \
  --arn arn:aws:cloudformation:us-west-2:123456789012:type/resource/Organization-Service-Resource
```

编辑账户中扩展的配置数据

本主题提供有关编辑特定区域内账户中扩展配置数据的指导。扩展可包含应用于给定账户和区域所有扩展实例的配置属性。扩展作者会在扩展的配置定义中定义这些属性。如果扩展的配置定义中有任何必需的属性，则必须先指定这些属性，然后才能在您的账户和区域中使用扩展。

有关在开发扩展时如何定义配置定义的更多信息，请参阅以下文档。

- 挂钩配置架构语法参考
- 定义扩展的账户级配置

主题

- 使用动态引用所需的权限
- 编辑扩展的配置数据（控制台）
- 编辑扩展的配置数据（Amazon CLI）

使用动态引用所需的权限

如果配置数据包含对存储在 Amazon Systems Manager 或 Amazon Secrets Manager 中的值的动态引用，则用于预置类型的任何角色（例如，在创建或更新堆栈时）必须具有检索该值的适当权限。具体来说：

- 如果配置数据包含存储在 Amazon Systems Manager Parameter Store 中的参数，则用于预置该类型的用户或角色必须有权调用 GetParameter。
- 如果配置数据包含存储在 Amazon Secrets Manager 中的密钥，则用于预置该类型的用户或角色必须有权调用 GetSecretValue。

有关更多信息，请参阅 使用动态引用获取存储在其他服务中的值。

编辑扩展的配置数据（控制台）

请按照本部分中的步骤，使用控制台来执行以下操作：

- 查看扩展的当前配置数据
- 更新账户的扩展配置数据

查看扩展的当前配置数据

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 从导航窗格的注册表下，选择已激活的扩展程序。
4. 找到您要查看的扩展。有关更多信息，请参阅 查看 CloudFormation 注册表中可用和已激活的扩展。
5. 选择扩展以查看扩展详细信息。
6. 在扩展详细信息页面上，选择 Configuration（配置）选项卡。
7. 展开 Configuration schema（配置架构）选项卡以查看为扩展定义的配置架构。
8. 展开 Configuration（配置）选项卡以查看为此扩展设置的当前配置。

更新扩展的配置数据

1. 在扩展详细信息页面上，从配置选项卡中选择编辑配置。

   或者，从操作中选择编辑，然后选择编辑配置。

   CloudFormation 将显示 Configure extension（配置扩展）页面。确保打开 View configuration schema（查看配置架构）以查看扩展的当前配置定义架构。

2. 在 Configuration JSON（配置 JSON）文本框中，输入一个 JSON 字符串，该字符串表示要为此扩展设置的配置架构。它必须根据 Configuration schema（配置架构）中定义的架构进行验证。
3. 选择配置扩展程序。

编辑扩展的配置数据（Amazon CLI）

请按照本部分中的步骤，使用 Amazon CLI 来执行以下操作：

- 查看扩展的当前配置数据
- 更新账户的扩展配置数据

查看扩展的当前配置数据

- 使用 describe-type 命令返回有关扩展的详细信息。输出的 `ConfigurationSchema` 元素包含给定区域中扩展的当前配置定义。

  或者，使用 batch-describe-type-configurations 命令返回有关多个扩展的配置数据。

更新扩展的配置数据

- 使用 set-type-configuration 命令指定配置数据。为 `--configuration` 传递的 JSON 必须根据扩展的配置架构进行验证。

  在以下示例中，set-type-configuration 命令指定了 `--configuration` 选项的配置数据 *`"{"CredentialKey": "testUserCredential"}"`*。

```
aws cloudformation set-type-configuration --type RESOURCE \
  --type-name My::Resource::Example \
  --configuration-alias default \
  --configuration "{"CredentialKey": "testUserCredential"}" \
  --region us-west-2
```

在 Amazon Config 中记录资源类型

您可以指定 Amazon Config 自动跟踪您的私有资源类型，并将对这些资源的更改作为配置项记录。这使您能够查看这些私有资源类型的配置历史记录，此外还可以编写 Amazon Config 规则 规则以验证配置最佳实践。挂钩扩展需要 Amazon Config。

要让 Amazon Config 自动跟踪您的私有资源类型，请执行以下操作：

- 通过 CloudFormation 管理资源。这包括通过 CloudFormation 执行所有资源创建、更新和删除操作。

  > **Note**
  >
  > 如果您使用 IAM 角色执行堆栈操作，则该 IAM 角色必须有权调用以下 Amazon Config 操作：
  >
  > - PutResourceConfig
  > - DeleteResourceConfig

- 配置 Amazon Config 以记录所有资源类型。有关更多信息，请参阅《Amazon Config Developer Guide》中的 Recording Configurations for Third-Party Resources using the Amazon CLI。

  > **Note**
  >
  > 如果私有资源包含同时定义为必需且只写的属性，Amazon Config 不支持记录这些私有资源。
  > 根据设计，定义为只写的资源属性不会在用于创建 Amazon Config 配置项的架构中返回。因此，包含定义为只写和必需的属性将导致配置项创建失败，因为必需的属性不存在。要查看用于创建配置项的架构，您可以查看 DescribeType 操作的 `schema` 属性。

有关配置项的更多信息，请参阅《Amazon Config 开发人员指南》中的配置项。

防止在配置项目中记录敏感属性

您的资源类型可能包含您视为敏感信息（如密码、密钥或其他敏感数据）的属性，您不希望在配置项目中记录这些属性。要防止在配置项目中记录属性，您可以将该属性包括在资源类型架构的 `writeOnlyproperties` 列表中。作为 `writeOnlyproperties` 列出的资源属性可由用户指定，但不会由 `read` 或 `list` 请求返回。

有关更多信息，请参阅《CloudFormation CLI User Guide》中的 writeOnlyProperties。

利用 CodePipeline 进行持续交付

持续交付是一种发布实践，通过该实践可以自动构建、测试和准备代码更改以便投产。使用 CloudFormation 和 CodePipeline，您可以通过持续交付自动构建和测试对 CloudFormation 模板的更改，然后再将这些更改提升到生产堆栈。此发布过程可让您快速可靠地更改 Amazon 基础设施。

例如，您可以创建一个工作流程，此工作流程在您将更新的模板提交到代码存储库时会自动构建测试堆栈。在 CloudFormation 构建测试堆栈后，您可以对其进行测试，然后决定是否将更改推送至生产堆栈。有关持续交付的好处的更多信息，请参阅什么是持续交付？。

通过为 CloudFormation 堆栈构建管道，使用 CodePipeline 构建持续交付的工作流。CodePipeline 具有与 CloudFormation 的内置集成，因此您可以在管道中指定 CloudFormation 特定的操作，例如创建、更新或删除堆栈。有关 CodePipeline 的更多信息，请参阅《Amazon CodePipeline 用户指南》。

主题

- 演练：为测试和生产堆栈构建管道
- Amazon CloudFormation 配置属性参考
- Amazon CloudFormation 构件
- 将参数覆盖函数用于 CodePipeline 管道

演练：为测试和生产堆栈构建管道

设想一个发布流程：您提交一个 Amazon CloudFormation 模板，然后 CloudFormation 使用该模板自动构建测试堆栈。在您审查测试堆栈后，可预览您的更改将如何修改生产堆栈，然后选择是否实施更改。要完成此工作流程，您可使用 CloudFormation 构建测试堆栈、删除测试堆栈、创建更改集，然后执行更改集。但是，在每个操作中，您都需要手动与 CloudFormation 进行交互。在此演练中，我们将构建一个 CodePipeline 管道来实现上述多个操作的自动化，从而帮助您使用 CloudFormation 堆栈实现持续交付工作流。

先决条件

该演练假定您使用过 CodePipeline 和 CloudFormation，并且知道管道以及 Amazon CloudFormation 模板和堆栈的工作方式。有关 CodePipeline 的更多信息，请参阅《Amazon CodePipeline 用户指南》。您创建管道的同一 Amazon Web Services 区域中还需要有一个 Amazon S3 存储桶。

> **Important**
>
> 示例 WordPress 模板将创建需要 Internet 连接的 EC2 实例。检查您是否有允许 Internet 流量的默认 VPC 和子网。

演练概述

此演练在堆栈中为示例 WordPress 站点构建一个管道。此管道分为三个阶段。每个阶段均必须包含至少一个操作，此操作是管道将在您的构件 (您的输入) 上执行的任务。阶段在管道中组织操作。CodePipeline 必须完成阶段中的所有操作，然后阶段才会处理新构件，例如，您提交新输入以重新运行管道。

在此演练结束时，您将获得执行以下工作流程的管道：

1. 管道的第一个阶段从存储库中检索源构件 (CloudFormation 模板及其配置文件)。

   您将准备一个包含示例 WordPress 模板的构件并将该构件上传到 S3 存储桶。
2. 在第二个阶段，管道将创建测试堆栈，然后等待您审批。

   在您审查测试堆栈后，可选择继续使用原来的管道或创建并提交另一个构件以进行更改。如果您批准，该阶段将删除测试堆栈，然后管道将继续下一阶段。
3. 在第三个阶段，管道将针对生产堆栈创建更改集，然后等待您审批。

   在初始运行中，您将没有生产堆栈。更改集显示 CloudFormation 将创建的所有资源。如果您批准，该阶段将执行更改集并构建您的生产堆栈。

> **Note**
>
> CloudFormation 是一项免费服务。但是，将按各种资源当前的费率向您收取您在堆栈中包含的 Amazon 资源 (如 EC2 实例) 的费用。有关 Amazon 定价的详细信息，请参阅 http://aws.amazon.com 上每种产品的详细信息页。

步骤 1：编辑构件并将构件上传到 S3 存储桶

在构建管道之前，您必须设置源存储库和文件。CodePipeline 将这些源文件复制到管道的构件存储中，然后使用它们在管道中执行操作，例如，创建 CloudFormation 堆栈。

在将 Amazon Simple Storage Service (Amazon S3) 作为源存储库时，CodePipeline 要求您压缩源文件，然后再将它们上传到 S3 存储桶中。压缩的文件是可包含 CloudFormation 模板和/或模板配置文件的 CodePipeline 构件。我们提供了一个包含一个示例 WordPress 模板和两个模板配置文件的构件。两个配置文件指定 WordPress 模板的参数值。CodePipeline 在创建 WordPress 堆栈时使用这些值。一个文件包含测试堆栈的参数值，另一个文件包含生产堆栈的参数值。您将需要编辑配置文件来指定您拥有的现有 EC2 密钥对名称。有关构件的更多信息，请参阅Amazon CloudFormation 构件。

在构建构件后，将构件上传到 S3 存储桶。

编辑和上传项目

1. 下载并打开示例构件：https://s3.amazonaws.com/cloudformation-examples/user-guide/continuous-deployment/wordpress-single-instance.zip。

   此构件包含三个文件：

   - 示例 WordPress 模板：`wordpress-single-instance.yaml`
   - 测试堆栈的模板配置文件：`test-stack-configuration.json`
   - 生产堆栈的模板配置文件：`prod-stack-configuration.json`
2. 提取所有文件，然后使用任意文本编辑器修改模板配置文件。

   打开配置文件以查看这些文件是否包含映射到 WordPress 模板参数的键值对。配置文件指定您的管道在创建测试堆栈和生产堆栈时使用的参数值。

   编辑 `test-stack-configuration.json` 文件以为测试堆栈指定参数值并为生产堆栈指定 `prod-stack-configuration.json` 文件。

   - 将 `DBPassword` 和 `DBRootPassword` 密钥的值更改为您可用于登录 WordPress 数据库的密码。如 WordPress 模板中所定义，参数值必须仅包含字母数字字符。
   - 将 `KeyName` 密钥的值更改为您将在其中创建管道的区域中的现有 EC2 密钥对名称。
3. 将修改的配置文件添加到原始构件 (`.zip`) 文件，并替换重复文件。

   您现在已拥有可上传到 S3 存储桶的自定义构件。
4. 将构件上传到您拥有的 S3 存储桶。

   记下文件的位置。您将在构建管道时指定此文件的位置。

   有关构件和 S3 存储桶的备注：

- 使用位于您将在其中创建管道的同一 Amazon 区域中的桶。
- CodePipeline 要求存储桶已启用版本控制。
- 您还可将不要求您在上传文件之前压缩文件的服务 (如 GitHub 或 CodeCommit) 用于源存储库。
- 构件可包含密码等敏感信息。限制访问以便仅允许的用户可查看文件。当您这样做时，确保 CodePipeline 仍可访问文件。例如，如果您将构件上传到 S3 存储桶，可使用 S3 存储桶策略或用户策略来限制访问。

您现已具有一个构件，CodePipeline 可将此构件拉入您的管道中。在下一步中，您将指定构件的位置并构建 WordPress 管道。

步骤 2：创建管道堆栈

为了创建 WordPress 管道，您将使用示例 CloudFormation 模板。除了构建管道以外，该模板还为 CodePipeline 和 CloudFormation 设置 Amazon Identity and Access Management（IAM）服务角色，为 CodePipeline 构件存储设置 S3 存储桶，以及设置管道将通知（如有关审核的通知）发送到的 Amazon Simple Notification Service（Amazon SNS）主题。示例模板使您能够在单一 CloudFormation 堆栈中轻松预置和配置这些资源。

有关管道配置的更多详细信息，请参阅管道的用途。

> **Important**
>
> 示例 WordPress 模板将创建需要 Internet 连接的 EC2 实例。检查您的默认 VPC 和子网是否允许到 Internet 的流量。

创建管道堆栈

1. 从 https://s3.amazonaws.com/cloudformation-examples/user-guide/continuous-deployment/basic-pipeline.yml 下载示例模板。将该模板保存到您的计算机上。
2. 通过以下网址打开 CloudFormation 控制台：https://console.aws.amazon.com/cloudformation/。
3. 选择一个支持 CodePipeline 和 CloudFormation 的 Amazon 区域。

   有关更多信息，请参阅 Amazon Web Services 一般参考 中的 CodePipeline 端点和限额和 Amazon CloudFormation 端点和限额。
4. 选择创建堆栈。

5. 在 Specify template (指定模板) 下，选择 Upload a template file (上传模板文件)，然后选择您刚下载的模板 `basic-pipeline.yml`。
6. 选择下一步。
7. 对于 Stack name，键入 `sample-WordPress-pipeline`。
8. 在 Parameters 部分中，指定下列参数值，然后选择 Next。设置堆栈参数时，如果您保留 WordPress 模板及其配置文件的相同名称，则可使用默认值。否则，请指定您使用的文件名。

   PipelineName

   管道的名称，如 `WordPress-test-pipeline`。

   S3Bucket

   您将构件 (`.zip` 文件) 保存到的 S3 存储桶的名称。

   SourceS3Key

   构件的文件名。如果您已将构件保存到文件夹中，则可将其作为文件名的一部分包含，如 `folder/subfolder/wordpress-single-instance.zip`。

   电子邮件

   CodePipeline 将管道通知发送到的电子邮件地址，如 `myemail@example.com`。
9. 在本演练中，您无需添加标记或指定高级设置，因此请选择 Next。
10. 确保堆栈名称和模板 URL 正确，然后选择 Create stack (创建堆栈)。
11. 要确认您知道 CloudFormation 可能会创建 IAM 资源，请选中该复选框。

Amazon CloudFormation 可能需要几分钟的时间以创建堆栈。要监控进度，可查看堆栈事件。有关更多信息，请参阅 监控堆栈进度。

在创建您的堆栈后，CodePipeline 将启动您的新管道。要查看其状态，请参阅 CodePipeline 控制台。从管道列表中，选择 WordPress-test-pipeline。

管道的用途

此部分使用示例 WordPress 管道模板中的代码段介绍管道的三个阶段。

第 1 阶段：源

管道的第 1 阶段是您指定源代码位置的源阶段。每当您将修订推送到此位置时，CodePipeline 都将重新运行您的管道。

源代码位于 S3 桶中，并由其文件名标识。您在创建管道堆栈时已将这些值指定为输入参数值。为了允许在后续阶段使用源构件，代码段将指定 OutputArtifacts 属性以及名称为 TemplateSource。要在后面的阶段中使用此构件，可将 TemplateSource 指定为输入构件。

```
- Name: S3Source
  Actions:
    - Name: TemplateSource
      ActionTypeId:
        Category: Source
        Owner: AWS
        Provider: S3
        Version: '1'
      Configuration:
        S3Bucket: !Ref 'S3Bucket'
        S3ObjectKey: !Ref 'SourceS3Key'
      OutputArtifacts:
        - Name: TemplateSource
```

第 2 阶段：TestStage

在 TestStage 阶段，管道将创建测试堆栈，等待审批，然后删除测试堆栈。

对于 CreateStack 操作，管道将使用测试配置文件和 WordPress 模板创建测试堆栈。这两个文件都包含在通过源阶段引入的 TemplateSource 输入构件中。代码段将使用 REPLACE_ON_FAILURE 操作模式。如果堆栈创建失败，管道将替换堆栈，这样一来，您无需清除堆栈或对堆栈进行故障排除，即可重新运行管道。操作模式对于快速迭代测试堆栈很有用。对于 RoleArn 属性，该值是在模板中的其他位置声明的 Amazon CloudFormation 服务角色。

ApproveTestStack 操作将暂停管道并向您在创建管道堆栈时指定的电子邮件地址发送通知。在管道暂停时，您可检查 WordPress 测试堆栈及其资源。可以使用 CodePipeline 批准或拒绝该操作。CustomData 属性包含对您正审批的操作的描述，它是由管道添加到通知电子邮件的。

在批准该操作后，CodePipeline 将移到 DeleteTestStack 操作并删除测试 WordPress 堆栈及其资源。

```
- Name: TestStage
  Actions:
    - Name: CreateStack
      ActionTypeId:
        Category: Deploy
        Owner: AWS
```

```
        Provider: CloudFormation
        Version: '1'
      InputArtifacts:
        - Name: TemplateSource
      Configuration:
        ActionMode: REPLACE_ON_FAILURE
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref TestStackName
        TemplateConfiguration: !Sub "TemplateSource::${TestStackConfig}"
        TemplatePath: !Sub "TemplateSource::${TemplateFileName}"
      RunOrder: '1'
    - Name: ApproveTestStack
      ActionTypeId:
        Category: Approval
        Owner: AWS
        Provider: Manual
        Version: '1'
      Configuration:
        NotificationArn: !Ref CodePipelineSNSTopic
        CustomData: !Sub 'Do you want to create a change set against the production
stack and delete the ${TestStackName} stack?'
      RunOrder: '2'
    - Name: DeleteTestStack
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: DELETE_ONLY
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref TestStackName
      RunOrder: '3'
```

第 3 阶段：ProdStage

管道的 `ProdStage` 阶段将针对现有生产堆栈创建更改集，等待审批，然后执行更改集。

更改集在实施 Amazon CloudFormation 将对生产堆栈进行的所有修改之前提供这些修改的预览。在您首次运行管道时，您将没有正在运行的生产堆栈。更改集显示 Amazon CloudFormation 在创建测试堆栈时执行的操作。要创建更改集，`CreateChangeSet` 操作将使用 WordPress 示例模板和 `TemplateSource` 输入构件中的生产模板配置。

与上一阶段类似，ApproveChangeSet 操作将暂停管道并发送电子邮件通知。在管道暂停后，您可查看更改集以检查针对生产 WordPress 堆栈的所有建议的修改。可以使用 CodePipeline 批准或拒绝该操作以相应地继续运行或停止管道。

在您批准该操作后，ExecuteChangeSet 操作执行更改集，以便 Amazon CloudFormation 执行更改集中描述的所有操作。在初始运行中，Amazon CloudFormation 将创建 WordPress 生产堆栈。在后续运行中，Amazon CloudFormation 将更新堆栈。

```
- Name: ProdStage
  Actions:
    - Name: CreateChangeSet
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      InputArtifacts:
        - Name: TemplateSource
      Configuration:
        ActionMode: CHANGE_SET_REPLACE
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref ProdStackName
        ChangeSetName: !Ref ChangeSetName
        TemplateConfiguration: !Sub "TemplateSource::${ProdStackConfig}"
        TemplatePath: !Sub "TemplateSource::${TemplateFileName}"
      RunOrder: '1'
    - Name: ApproveChangeSet
      ActionTypeId:
        Category: Approval
        Owner: AWS
        Provider: Manual
        Version: '1'
      Configuration:
        NotificationArn: !Ref CodePipelineSNSTopic
        CustomData: !Sub 'A new change set was created for the ${ProdStackName} stack.
 Do you want to implement the changes?'
      RunOrder: '2'
    - Name: ExecuteChangeSet
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
```

```
      Configuration:
        ActionMode: CHANGE_SET_EXECUTE
        ChangeSetName: !Ref ChangeSetName
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref ProdStackName
      RunOrder: '3'
```

步骤 3：查看 WordPress 堆栈

由于 CodePipeline 在整个管道中运行，因此，它使用 CloudFormation 创建测试和生产堆栈。要查看这些堆栈的状态及其输出，可使用 CloudFormation 控制台。

查看堆栈

1. 通过以下网址打开 CloudFormation 控制台：https://console.aws.amazon.com/cloudformation/。
2. 根据您的管道是处于测试阶段还是生产阶段，选择 `Test-MyWordPressSite` 或 `Prod-MyWordPressSite` 堆栈。
3. 要查看堆栈状态，可查看堆栈事件。

如果堆栈处于故障状态，可查看状态描述以找出堆栈错误。修复错误，然后重新运行管道。如果堆栈处于 `CREATE_COMPLETE` 状态，请查看其输出以获取您的 WordPress 站点的 URL。

您已成功使用 CodePipeline 为示例 WordPress 站点构建持续交付工作流程。如果您将更改提交到 S3 存储桶，CodePipeline 将自动检测新版本，然后重新运行您的管道。利用此工作流程，可以在更改生产站点之前更轻松地提交和测试更改。

步骤 4：清理资源

要确保不因任何不必要的服务而产生费用，请删除您的资源。

> **Important**
>
> 在删除管道堆栈前，删除测试和生产 WordPress 堆栈。管道堆栈包含删除 WordPress 堆栈所需的服务角色。如果您已先删除管道堆栈，则可关联另一个服务角色 Amazon 资源名称（ARN）与 WordPress 堆栈，然后删除它们。

删除构件存储中的对象

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 选择 CodePipeline 用作管道构件存储的 S3 存储桶。

   此存储桶的名称遵循以下格式：*stackname-artifactstorebucket-id*。如果您已按照此演练操作，此存储桶的名称可能看上去与以下示例类似：`sample-WordPress-pipeline-artifactstorebucket-12345abcd12345`。

3. 删除构件存储 S3 存储桶中的所有对象。

   当您在下一步中删除管道堆栈时，此存储桶必须为空。否则，CloudFormation 将无法删除此存储桶。

删除堆栈

1. 从 CloudFormation 控制台中，选择要删除的堆栈。

   如果管道创建的 WordPress 堆栈仍在运行，请先选择它们。默认情况下，堆栈名称为 `Test-MyWordPressSite` 和 `Prod-MyWordPressSite`。

   如果您已删除 WordPress 堆栈，请选择 `sample-WordPress-pipeline` 堆栈。

2. 选择 Actions)（操作），然后选择 Delete Stack（删除堆栈）。
3. 在确认消息中，选择 Yes, Delete。

CloudFormation 将删除堆栈的所有堆栈资源，如 EC2 实例、通知主题、服务角色和管道。

现在您已了解如何使用 CodePipeline 构建基本 CloudFormation 工作流程，您可以将示例模板和构件作为起点以构建自己的工作流程。

另请参阅

下列相关资源可帮助您处理这些参数。

- 有关 CodePipeline 中 CloudFormation 操作参数的更多信息， 请参阅《Amazon CodePipeline 用户指南》中的 Amazon CloudFormation 部署操作配置引用。
- 有关操作提供方的示例模板值，例如 `Owner` 字段或 `configuration` 字段的值，请参阅《Amazon CodePipeline 用户指南》中的操作结构参考。
- 要下载 YAML 或 JSON 格式的示例管道堆栈模板，请参阅《Amazon CodePipeline 用户指南》中教程：使用 Amazon CloudFormation 创建管道。

Amazon CloudFormation 配置属性参考

当您构建 CodePipeline 管道时，可向此管道添加 `Deploy` 操作并将 Amazon CloudFormation 作为该操作的提供者。随后，您必须指定管道调用的 Amazon CloudFormation 操作以及该操作的设置。本主题介绍 Amazon CloudFormation 配置属性。要指定属性，您可以使用 CodePipeline 控制台，也可以创建 JSON 对象以用于 Amazon CLI、CodePipeline API 或 Amazon CloudFormation 模板。

主题

- 配置属性 (控制台)
- 配置属性 (JSON 对象)
- 另请参阅

配置属性 (控制台)

CodePipeline 控制台显示配置属性，并根据您选择的操作模式指示所需的属性。

> **Note**
>
> 当您创建管道时，可以仅指定 Create or update a stack (创建或更新堆栈) 或 Create or replace a change set (创建或替换更改集) 操作模式。Advanced (高级) 部分的属性仅在您编辑管道时可用。

操作模式

在处理关联的阶段时 CodePipeline 调用的 Amazon CloudFormation 操作。选择下列操作模式之一：

- Create or replace a change set 根据您提交的堆栈名称和模板创建更改集 (如果更改集不存在)。如果更改集存在，Amazon CloudFormation 将删除它，然后创建一个新的更改集。
- Create or update a stack 创建堆栈 (如果指定的堆栈不存在)。如果堆栈存在，Amazon CloudFormation 将更新堆栈。使用此操作更新现有堆栈。CodePipeline 不会取代堆栈。
- Delete a stack 删除堆栈。如果您指定的堆栈不存在，操作将成功完成，而不会删除堆栈。
- Execute a change set 执行更改集。
- Replace a failed stack 创建堆栈 (如果指定的堆栈不存在)。如果堆栈存在且处于故障状态 (报告为

ROLLBACK_COMPLETE、ROLLBACK_FAILED、CREATE_FAILED、DELETE_FAILED、UPDATE_ROLL
CloudFormation 将删除堆栈，然后创建新的堆栈。如果堆栈未处于故障状态，Amazon CloudFormation 将更新堆栈。使用此操作可替换出现故障的堆栈，而无需对其进行恢复或故障排除。您通常将选择此模式以进行测试。

堆栈名称

与现有堆栈或您要创建的堆栈关联的名称。名称在您创建堆栈的 Amazon 区域中必须是唯一的。

> **Note**
>
> 堆栈名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 128 个字符。

更改集名称

现有更改集的名称或要为指定堆栈创建的新更改集的名称。

模板

Amazon CloudFormation 模板文件的位置，它采用 *ArtifactName*::*TemplateFileName* 格式。

模板配置

模板配置文件的位置，它遵循格式 *ArtifactName*::*TemplateConfigurationFileName*。模板配置文件可以包含模板参数值、堆栈策略和标签。如果您将敏感信息 (例如密码) 包括在此文件中，请限制对此文件的访问。有关更多信息，请参阅 Amazon CloudFormation 构件。

功能

对于包含特定资源的堆栈，明确确认 Amazon CloudFormation 可能创建或更新这些资源。例如，如果堆栈模板包含 CAPABILITY_IAM（IAM）资源，您必须指定 Amazon Identity and Access Management。有关更多信息，请参阅 CreateStack API 操作请求参数。

如果您的堆栈模板包含 IAM 资源，则必须指定此属性。

您可以指定多个功能。

角色名称

Amazon CloudFormation 在处理指定的堆栈中的资源时担任的 IAM 服务角色的名称。

Output file name

在 Advanced (高级) 部分中，您可以指定在执行指定的操作后 CodePipeline 添加到输出构件的输出文件名称，如 `CreateStackOutput.json`。该输出构件包含一个 JSON 文件，该文件具有 Amazon CloudFormation 模板 `Outputs` 部分的内容。

如果您不指定名称，则 CodePipeline 不会生成输出构件。

Parameter overrides

参数在模板中定义，并允许您在创建或更新堆栈时输入自定义值。您可以指定覆盖模板配置文件中的模板参数值的 JSON 对象。所有参数名称必须位于堆栈模板中。有关更多信息，请参阅 CloudFormation 模板 Parameters 语法。

> **Note**
>
> `ParameterOverrides` 属性中可存储的 JSON 对象的大小上限是 1 KB。

建议您使用模板配置文件来指定大多数参数值。仅使用参数覆盖来指定动态参数值。动态参数在您运行管道之前是未知的。

以下示例通过使用参数覆盖函数来定义 `ParameterName` 参数的值。该函数从 CodePipeline 输入构件中检索值。有关参数覆盖函数的更多信息，请参阅将参数覆盖函数用于 CodePipeline 管道。

```
{
  "ParameterName" : { "Fn::GetParam" : ["ArtifactName", "config-file-name.json",
 "ParamName"]}
}
```

配置属性 (JSON 对象)

当您指定 `CloudFormation` 作为阶段操作的提供者时，请在 `Configuration` 属性中定义以下属性。将 JSON 对象用于 Amazon CLI、CodePipeline API 或 Amazon CloudFormation 模板。有关示例，请参阅 演练：为测试和生产堆栈构建管道 和 Amazon CloudFormation 配置属性参考。

`ActionMode`

在处理关联的阶段时 CodePipeline 调用的 Amazon CloudFormation 操作。仅指定下列操作模式之一：

- CHANGE_SET_EXECUTE 执行更改集。
- CHANGE_SET_REPLACE 根据您提交的堆栈名称和模板创建更改集（如果更改集不存在）。如果更改集存在，Amazon CloudFormation 将删除它，然后创建一个新的更改集。
- CREATE_UPDATE 创建堆栈 (如果指定的堆栈不存在)。如果堆栈存在，Amazon CloudFormation 将更新堆栈。使用此操作更新现有堆栈。CodePipeline 不会取代堆栈。
- DELETE_ONLY 删除堆栈。如果您指定的堆栈不存在，操作将成功完成，而不会删除堆栈。
- REPLACE_ON_FAILURE 创建堆栈（如果指定的堆栈不存在）。如果堆栈存在并处于失败状态（报告为 ROLLBACK_COMPLETE、ROLLBACK_FAILED、CREATE_FAILED、DELETE_FAILED 或 UPDATE_ROLLBACK_FAILED），Amazon CloudFormation 将删除堆栈，然后创建新的堆栈。如果堆栈未处于故障状态，Amazon CloudFormation 将更新堆栈。使用自操作可自动替换出现故障的堆栈，而无需对其进行恢复或故障排除。您通常将选择此模式以进行测试。

该属性是必需属性。

`Capabilities`

对于包含特定资源的堆栈，明确确认 Amazon CloudFormation 可能创建或更新这些资源。例如，如果堆栈模板包含 CAPABILITY_IAM（IAM）资源，您必须指定 Amazon Identity and Access Management。有关更多信息，请参阅 CreateStack API 操作请求参数。

该属性是条件属性。如果您的堆栈模板包含 IAM 资源，则必须指定此属性。

您可以指定多个功能。以下示例将 CAPABILITY_IAM 和 CAPABILITY_AUTO_EXPAND 属性添加到模板中：

YAML

```
configuration:
  ActionMode: CHANGE_SET_REPLACE
  Capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
  ChangeSetName: pipeline-changeset
  RoleArn: CloudFormation_Role_ARN
  StackName: my-pipeline-stack
  TemplateConfiguration: 'my-pipeline-stack::template-configuration.json'
  TemplatePath: 'my-pipeline-stack::template-export.yml'
```

JSON

```
 "configuration": {
        "ActionMode": "CHANGE_SET_REPLACE",
```

```
        "Capabilities": "CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND",
        "ChangeSetName": "pipeline-changeset",
        "RoleArn": "CloudFormation_Role_ARN",
        "StackName": "my-pipeline-stack",
        "TemplateConfiguration": "my-pipeline-stack::template-
configuration.json",
        "TemplatePath": "my-pipeline-stack::template-export.yml"
    }
```

`ChangeSetName`

现有更改集的名称或要为指定堆栈创建的新更改集的名称。

以下操作模式需要该属性：CHANGE_SET_REPLACE 和 CHANGE_SET_EXECUTE。对于所有其他操作模式，将忽略此属性。

`OutputFileName`

输出文件的名称，例如 `CreateStackOutput.json`。在执行指定的操作后，CodePipeline 将该文件添加到输出构件中。该输出构件包含一个 JSON 文件，该文件具有 Amazon CloudFormation 模板 `Outputs` 部分的内容。

该属性是可选属性。如果您不指定名称，则 CodePipeline 不会生成输出构件。

`ParameterOverrides`

参数在模板中定义，并允许您在创建或更新堆栈时输入自定义值。您可以指定覆盖模板配置文件中的模板参数值的 JSON 对象。所有参数名称必须位于堆栈模板中。有关更多信息，请参阅 CloudFormation 模板 Parameters 语法。

以下示例显示了如何将 `InstanceType` 和 `KeyName` 参数覆盖添加到模板：

YAML

```
configuration:
  ActionMode: CHANGE_SET_REPLACE
  Capabilities: CAPABILITY_NAMED_IAM
  ChangeSetName: pipeline-changeset
  ParameterOverrides: '{"InstanceType": "t2.small","KeyName": "my-keypair"}'
  RoleArn: CloudFormation_Role_ARN
  StackName: my-pipeline-stack
  TemplateConfiguration: 'my-pipeline-stack::template-configuration.json'
  TemplatePath: 'my-pipeline-stack::template-export.yml'
```

JSON

```
"configuration": {
        "ActionMode": "CHANGE_SET_REPLACE",
        "Capabilities": "CAPABILITY_NAMED_IAM",
        "ChangeSetName": "pipeline-changeset",
        "ParameterOverrides": "{\"InstanceType\": \"t2.small\",\"KeyName\": \"my-
keypair\"}",
        "RoleArn": "CloudFormation_Role_ARN",
        "StackName": "my-pipeline-stack",
        "TemplateConfiguration": "my-pipeline-stack::template-
configuration.json",
        "TemplatePath": "my-pipeline-stack::template-export.yml"
    }
```

> **Note**
>
> 可以存储在 ParameterOverrides 属性中的 JSON 对象的最大大小为 1 KB。

建议您使用模板配置文件来指定大多数参数值。仅使用参数覆盖来指定动态参数值。动态参数值在您运行管道之前是未知的。

以下示例通过使用参数覆盖函数来定义 ParameterName 参数的值。该函数从 CodePipeline 输入构件中检索值。有关参数覆盖函数的更多信息，请参阅将参数覆盖函数用于 CodePipeline 管道。

```
{
  "ParameterName" : { "Fn::GetParam" : ["ArtifactName", "config-file-name.json",
 "ParamName"]}
}
```

该属性是可选属性。

RoleArn

在操作堆栈中的资源时 Amazon CloudFormation 担任的 IAM 服务角色的 Amazon 资源名称（ARN）。

以下操作模式需要此属性：CREATE_UPDATE、REPLACE_ON_FAILURE、DELETE_ONLY 和 CHANGE_SET_REPLACE。执行更改集时不应用 RoleArn。如果您不使用 CodePipeline 来创建更改集，请确保更改集或堆栈具有关联的角色。

StackName

现有堆栈的名称或要创建的堆栈的名称。

所有操作模式都需要该属性。

TemplateConfiguration

TemplateConfiguration 是模板配置文件。您需在针对此操作的输入构件中包含该文件。模板配置文件名遵循以下格式：

Artifactname::*TemplateConfigurationFileName*

Artifactname 是输入构件在 CodePipeline 中显示的名称。例如，源阶段利用构件名称 SourceArtifact 和文件名 test-configuration.json 创建 TemplateConfiguration 名称，如以下示例所示：

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

模板配置文件可包含模板参数值和堆栈策略。如果您将敏感信息 (例如密码) 包括在此文件中，请限制对此文件的访问。有关示例模板配置文件，请参阅 Amazon CloudFormation 构件。

该属性是可选属性。

TemplatePath

TemplatePath 表示 Amazon CloudFormation 模板文件。您需在针对此操作的输入构件中包含该文件。文件名称遵循以下格式：

Artifactname::*TemplateFileName*

Artifactname 是输入构件在 CodePipeline 中显示的名称。例如，源阶段利用构件名称 SourceArtifact 和文件名 template.yaml 创建 TemplatePath 名称，如以下示例所示：

```
"TemplatePath": "SourceArtifact::template.yaml"
```

以下操作模式需要此属性：CREATE_UPDATE、REPLACE_ON_FAILURE 和 CHANGE_SET_REPLACE。对于所有其他操作模式，将忽略此属性。

另请参阅

下列相关资源可帮助您处理这些参数。

- 有关 CodePipeline 中 CloudFormation 操作参数的更多信息， 请参阅《Amazon CodePipeline 用户指南》中的 Amazon CloudFormation 部署操作配置引用。
- 有关操作提供方的示例模板值，例如 `Owner` 字段或 `configuration` 字段的值，请参阅《Amazon CodePipeline 用户指南》中的操作结构参考。
- 要下载 YAML 或 JSON 格式的示例管道堆栈模板，请参阅《Amazon CodePipeline 用户指南》中教程：使用 Amazon CloudFormation 创建管道。

Amazon CloudFormation 构件

在 CodePipeline 运行管道时，CodePipeline 对构件执行一些任务。对于 Amazon CloudFormation，构件可以包含堆栈模板文件和/或模板配置文件。CodePipeline 使用这些构件处理 Amazon CloudFormation 堆栈和更改集。

如果将 Amazon Simple Storage Service（Amazon S3）用作源存储库，您必须将模板和模板配置文件压缩为一个文件，然后再将它们上传到 S3 桶中。对于其他存储库 (如 GitHub 和 Amazon CodeCommit)，可上传构件而无需进行压缩。有关更多信息，请参阅在《Amazon CodePipeline 用户指南》中的创建管道、阶段和操作。

您可以将所需数目的文件添加到您的存储库。例如，您可能希望包含同一模板的两个不同配置：一个用于测试配置，另一个用于生产配置。

本主题描述每个构件类型。

主题

- 堆栈模板文件
- 模板配置文件
- 另请参阅

堆栈模板文件

堆栈模板文件定义 Amazon CloudFormation 预置和配置的资源。这些文件与您在使用 Amazon CloudFormation 创建或更新堆栈时使用的模板文件相同。您可以使用 YAML 或 JSON 格式的模板。有关模板的更多信息，请参阅CloudFormation 模板部分。

模板配置文件

模板配置文件是 JSON 格式的文本文件，可指定模板参数值、堆栈策略和标签。使用这些配置文件可指定参数值或堆栈的堆栈策略。您指定的所有参数值都必须在关联的模板中进行声明。

如果在该文件中包括敏感信息（如密码），请限制对该文件的访问。例如，如果您将构件上传到 S3 存储桶，可使用 S3 存储桶策略或用户策略来限制访问。

要创建配置文件，可使用以下格式：

```
{
  "Parameters" : {
    "NameOfTemplateParameter" : "ValueOfParameter",
    ...
  },
  "Tags" : {
    "TagKey" : "TagValue",
    ...
  },
  "StackPolicy" : {
    "Statement" : [
      StackPolicyStatement
    ]
  }
}
```

以下示例为 TestEC2Key 参数指定 KeyName，并添加一个值为 Department 的 Marketing 标签，以及一个允许所有更新操作（会删除资源的更新操作除外）的堆栈策略。

```
{
  "Parameters" : {
    "KeyName" : "TestEC2Key"
  },
  "Tags" : {
    "Department" : "Marketing"
  },
  "StackPolicy" : {
    "Statement" : [
      {
        "Effect" : "Allow",
        "NotAction" : "Update:Delete",
        "Principal": "*",
        "Resource" : "*"
```

```
      }
    ]
  }
}
```

另请参阅

下列相关资源可帮助您处理这些参数。

- 有关 CodePipeline 中 CloudFormation 操作参数的更多信息， 请参阅《Amazon CodePipeline 用户指南》中的 Amazon CloudFormation 部署操作配置引用。
- 有关操作提供方的示例模板值，例如 `Owner` 字段或 `configuration` 字段的值，请参阅《Amazon CodePipeline 用户指南》中的操作结构参考。
- 要下载 YAML 或 JSON 格式的示例管道堆栈模板，请参阅《Amazon CodePipeline 用户指南》中教程：使用 Amazon CloudFormation 创建管道。

将参数覆盖函数用于 CodePipeline 管道

在 CodePipeline 阶段，您可以为 Amazon CloudFormation 操作指定参数覆盖。通过使用参数覆盖，您可以指定覆盖模板配置文件中的值的模板参数值。Amazon CloudFormation 提供了一些函数以帮助您指定动态值（在管道运行之前未知的值）。

主题

- Fn::GetArtifactAtt
- Fn::GetParam
- 另请参阅

Fn::GetArtifactAtt

`Fn::GetArtifactAtt` 函数将从输入构件 (例如，存储了构件的 S3 存储桶名称) 检索属性的值。使用此函数可指定构件的属性，例如，构件的文件名或 Amazon S3 桶名称。

在运行管道时，CodePipeline 复制文件并将其写入到管道的构件存储（S3 桶）中。CodePipeline 在构件存储中生成文件名。在您运行管道之前，这些文件名是未知的。

例如，在管道中，您可能具有一个源阶段，CodePipeline 在该阶段将 Amazon Lambda 函数源代码复制到构件存储中。在下一个阶段，您使用一个 Amazon CloudFormation 模板以创建 Lambda 函数，但

Amazon CloudFormation 需要使用文件名以创建该函数。您必须使用 Fn::GetArtifactAtt 函数以传递准确的 S3 存储桶和文件名。

语法

使用以下语法可检索构件的属性值。

```
{ "Fn::GetArtifactAtt" : [ "artifactName", "attributeName" ] }
```

`artifactName`

输入构件的名称。您必须声明此构件作为相关操作的输入。

`attributeName`

要检索其值的构件属性的名称。有关每个构件属性的详细信息，请参阅以下“属性”部分。

示例

以下参数覆盖通过检索 LambdaFunctionSource 构件的 S3 桶名称和文件名来指定 BucketName 和 ObjectKey 参数。该示例假定 CodePipeline 复制了 Lambda 函数源代码并将其保存为构件，例如，作为源阶段的一部分。

```
{
  "BucketName" : { "Fn::GetArtifactAtt" : ["LambdaFunctionSource", "BucketName"]},
  "ObjectKey" : { "Fn::GetArtifactAtt" : ["LambdaFunctionSource", "ObjectKey"]}
}
```

属性

您可以检索构件的以下属性。

`BucketName`

存储了构件的 S3 存储桶的名称。

`ObjectKey`

包含由 CodePipeline 生成的构件的 .zip 文件的名称，如 1ABCyZZ.zip。

URL

构件的 Amazon Simple Storage Service (Amazon S3) URL，如 `https://s3.us-west-2.amazonaws.com/artifactstorebucket-yivczw8jma0c/test/TemplateSo/1ABCyZZ.zip`。

Fn::GetParam

`Fn::GetParam` 函数将从 JSON 格式的文件中的键值对返回一个值。JSON 文件必须包含在构件中。

使用此函数可从 Amazon CloudFormation 堆栈中检索输出值并将这些值用作另一个操作的输入。例如，如果为 Amazon CloudFormation 操作指定输出文件名，则 CodePipeline 在 JSON 文件中保存输出，然后将其添加到输出构件的 `.zip` 文件中。使用 `Fn::GetParam` 函数检索输出值，并将其用作另一个操作的输入。

语法

使用以下语法可检索键值对的值。

```
{ "Fn::GetParam" : [ "artifactName", "JSONFileName", "keyName" ] }
```

`artifactName`

构件的名称，必须作为相关操作的输入构件的包含。

`JSONFileName`

构件中包含的 JSON 文件的名称。

`keyName`

要检索其值的键的名称。

示例

以下示例将说明如何在参数覆盖中使用 `Fn::GetParam` 函数。

语法

以下参数覆盖通过从 `WebSiteURL` 构件中的 URL 文件检索 `stack-output.json` 键的值来指定 `WebStackOutput` 参数。

```
{
  "WebSiteURL" : { "Fn::GetParam" : ["WebStackOutput", "stack-output.json", "URL"]}
}
```

Amazon CloudFormation 模板代码段

CodePipeline 管道中的以下 Amazon CloudFormation 模板代码段将演示如何传递堆栈输出。这些代码段说明了管道定义的两个阶段。第一个阶段将创建一个堆栈并将其输出保存在 StackAOutput 构件的 TestOutput.json 文件中。这些值由 OutputFileName 和 OutputArtifacts 属性指定。

阶段的源输入构件的名称为 TemplateSource。堆栈模板的文件名为 teststackA.yaml，配置文件的文件名为 test-configuration.json。在这两个阶段，为 TemplateConfiguration 和 TemplatePath 属性指定的这些值如下所示：

```
TemplateConfiguration: TemplateSource::test-configuration.json
TemplatePath: TemplateSource::teststackA.yaml
```

Example创建堆栈 A 阶段

```
- Name: CreateTestStackA
  Actions:
    - Name: CloudFormationCreate
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: CREATE_UPDATE
        Capabilities: CAPABILITY_IAM
        OutputFileName: TestOutput.json
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: StackA
        TemplateConfiguration: TemplateSource::test-configuration.json
        TemplatePath: TemplateSource::teststackA.yaml
      InputArtifacts:
        - Name: TemplateSource
      OutputArtifacts:
        - Name: StackAOutput
      RunOrder: '1'
```

在后续阶段中，堆栈 B 使用堆栈 A 中的输出。在 ParameterOverrides 属性中，示例使用 Fn::GetParam 函数指定 StackBInputParam 参数。结果值是与 StackAOutputName 键关联的值。

Example创建堆栈 B 阶段

```
- Name: CreateTestStackB
  Actions:
    - Name: CloudFormationCreate
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: CREATE_UPDATE
        Capabilities: CAPABILITY_IAM
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: StackB
        TemplateConfiguration: TemplateSource::test-configuration.json
        TemplatePath: TemplateSource::teststackB.yaml
        ParameterOverrides: |
          {
            "StackBInputParam" : { "Fn::GetParam" : ["StackAOutput", "TestOutput.json",
 "StackAOutputName"]}
          }
      InputArtifacts:
        - Name: TemplateSource
        - Name: StackAOutput
      RunOrder: '1'
```

另请参阅

下列相关资源可帮助您处理这些参数。

- 有关 CodePipeline 中 CloudFormation 操作参数的更多信息， 请参阅《Amazon CodePipeline 用户指南》中的 Amazon CloudFormation 部署操作配置引用。
- 有关操作提供方的示例模板值，例如 Owner 字段或 configuration 字段的值，请参阅《Amazon CodePipeline 用户指南》中的操作结构参考。
- 要下载 YAML 或 JSON 格式的示例管道堆栈模板，请参阅《Amazon CodePipeline 用户指南》中教程：使用 Amazon CloudFormation 创建管道。

Amazon CloudFormation 模板参考指南

有关可用于创建 Amazon CloudFormation 模板的资源和组件的详细信息，现在可在《Amazon CloudFormation 模板参考指南》https://docs.amazonaws.cn/AWSCloudFormation/latest/TemplateReference/introduction.html中找到。本指南可作为模板的技术参考，是对《CloudFormation 用户指南》中入门内容的补充。

《Amazon CloudFormation 模板参考指南》提供了有关以下内容的详细信息：

- 资源类型及其属性
- `DeletionPolicy` 和其他资源级属性
- 内置函数，例如 `Ref` 和 `Sub`
- 转换（CloudFormation 宏）
- EC2 实例配置的帮助程序脚本
- 机器可读资源类型定义

本指南定期更新，以反映以下内容：

- 新资源类型
- 属性更改
- 新返回值
- 文档更新

Amazon CloudFormation 中的安全性

Amazon 的云安全性的优先级最高。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 Amazon 的客户，您也可以从这些数据中心和网络架构受益。

安全性是 Amazon 和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全性 – Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础结构。Amazon 还向您提供可安全使用的服务。作为Amazon 合规性计划的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解适用于 Amazon CloudFormation 的合规性计划，请参阅合规性计划范围内的 Amazon 服务。
- 云中的安全性——您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon CloudFormation 时应用责任共担模型。以下主题说明如何配置 Amazon CloudFormation 以实现您的安全性和合规性目标。您还会了解如何使用其他 Amazon 服务以帮助您监控和保护 Amazon CloudFormation 资源。

主题

- CloudFormation 堆栈的删除保护
- 防止更新堆栈资源
- Amazon CloudFormation 中的数据保护
- 使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限
- 使用 Amazon CloudTrail 记录 Amazon CloudFormation API 调用
- Amazon CloudFormation 中的基础结构安全性
- Amazon CloudFormation 中的故障恢复能力
- 的合规性验证 Amazon CloudFormation
- Amazon CloudFormation 中的配置和漏洞分析
- CloudFormation 的安全最佳实践
- 使用接口端点 (Amazon PrivateLink) 访问 Amazon CloudFormation

CloudFormation 堆栈的删除保护

您可以通过在堆栈上启用终止保护来防止堆栈被意外删除。如果用户尝试删除已启用终止保护的堆栈，则删除操作会失败，并且堆栈及其状态将保持不变。可在创建堆栈时对其启用终止保护。默认情况下，堆栈上的终止保护是禁用的。您可以为状态为 DELETE_IN_PROGRESS 或 DELETE_COMPLETE 之外任何状态的堆栈设置终止保护。

如果在堆栈上启用或禁用终止保护，则也会将相同的选择传递给属于该堆栈的任何嵌套堆栈。不能直接在嵌套堆栈上启用或禁用终止保护。如果用户尝试直接删除一个属于已启用终止保护的堆栈的嵌套堆栈，则操作会失败，并且嵌套堆栈将保持不变。

但是，如果用户执行会删除嵌套堆栈的堆栈更新，则 Amazon CloudFormation 会相应地删除嵌套堆栈。

终止保护不同于禁用回滚。终止保护仅适用于尝试删除堆栈的情况，而禁用回滚适用于创建堆栈失败时的自动回滚。

在创建堆栈时启用终止保护

在 Create stack（创建堆栈）向导中的 Specify stack options（指定堆栈选项）页面上，在 Advanced options（高级选项）下，展开 Termination Protection（终止保护）部分，然后选择 Enable（启用）。有关更多信息，请参阅 配置堆栈选项。

在现有堆栈上启用或禁用终止保护

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 选择想要设置的堆栈。

   > **Note**
   >
   > 如果堆栈名称旁边显示 NESTED，则该堆栈为嵌套堆栈。您只能在该嵌套堆栈所属的根堆栈上更改终止保护。

4. 在堆栈详细信息窗格中，选择 Stack actions (堆栈选项)，然后选择 Edit termination protection (编辑终止保护)。

   CloudFormation 会显示编辑终止保护对话框。
5. 选择 Enable (启用) 或 Disable (禁用)，然后选择 Save (保存)。

在嵌套堆栈上启用或禁用终止保护

如果堆栈名称旁边显示 NESTED，则该堆栈为嵌套堆栈。您只能在该嵌套堆栈所属的根堆栈上更改终止保护。在根堆栈上更改终止保护：

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择您的 Amazon Web Services 区域。
3. 选择想要设置的嵌套堆栈。
4. 在 Stack info (堆栈信息) 窗格中，在概述部分中，选择作为根堆栈列出的堆栈名称。

   CloudFormation 会显示根堆栈的堆栈详细信息。
5. 依次选择 Stack actions (堆栈选项)、Edit termination protection (编辑终止保护)。

   CloudFormation 会显示编辑终止保护对话框。
6. 选择 Enable (启用) 或 Disable (禁用)，然后选择 Save (保存)。

使用命令行启用或禁用终止保护

使用 update-termination-protection 命令。

控制谁可以在堆栈上更改终止保护

要在堆栈上启用或禁用终止保护，用户需要拥有进行 `cloudformation:UpdateTerminationProtection` 操作的权限。例如，以下策略允许用户在堆栈上启用或禁用终止保护。

有关在 Amazon CloudFormation 中指定权限的更多信息，请参阅使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限。

Example授予更改堆栈终止保护权限的示例策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
```

```
            "cloudformation:UpdateTerminationProtection"
        ],
        "Resource":"*"
    }]
}
```

防止更新堆栈资源

在创建堆栈时，允许对所有资源执行所有更新操作。默认情况下，具有堆栈更新权限的任何人均可更新堆栈中的所有资源。在更新期间，一些资源可能需要中断或可能已完全替换，这会生成新的物理 ID 或全新的存储。使用堆栈策略可以防止堆栈资源在堆栈更新过程中被意外更新或删除。堆栈策略是一个 JSON 文档，该文档定义可对指定资源执行的更新操作。

设置堆栈策略后，默认情况下将保护堆栈中的所有资源。要允许对特定资源进行更新，您可在堆栈策略中为这些资源指定明确的 `Allow` 语句。您只能为每个堆栈定义一个堆栈策略，但在一个策略中可以保护多个资源。堆栈策略适用于所有尝试更新堆栈的 CloudFormation 用户。您不能将不同的堆栈策略与不同的用户关联。

堆栈策略仅在堆栈更新过程中适用。与 Amazon Identity and Access Management（IAM）策略不同，它不提供访问控制。仅将堆栈策略用作故障保护功能来防止意外更新特定堆栈资源。要控制对 Amazon 资源或操作的访问，请使用 IAM。

主题

- 示例堆栈策略
- 定义堆栈策略
- 设置堆栈策略
- 更新受保护资源
- 修改堆栈策略
- 更多示例堆栈策略

示例堆栈策略

下面的示例堆栈策略阻止更新 `ProductionDatabase` 资源：

```
{
  "Statement" : [
    {
```

```
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    },
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/ProductionDatabase"
    }
  ]
}
```

当您设置堆栈策略时，将默认保护所有资源。为了允许对所有资源进行更新，我们添加了一个 `Allow` 语句来允许对所有资源执行的所有操作。虽然 `Allow` 语句指定所有资源，但显式 `Deny` 语句将为具有 `ProductionDatabase` 逻辑 ID 的资源覆盖前者。此 `Deny` 语句阻止对 `ProductionDatabase` 资源进行的所有更新操作，例如替换或删除。

需要 `Principal` 元素，但仅支持通配符 (*)，这意味着语句适用于所有委托人。

> **Note**
>
> 在堆栈更新期间，CloudFormation 会自动更新依赖其他更新的资源的资源。例如，CloudFormation 会更新引用所更新的资源的资源。CloudFormation 不会对自动更新的资源进行任何物理更改（例如，资源的 ID），但如果这些资源关联了堆栈策略，则您必须具有更新这些资源的权限。

定义堆栈策略

在创建堆栈时，未设置堆栈策略，因此允许对所有资源执行所有更新操作。要阻止对堆栈资源执行更新操作，可定义一个堆栈策略，然后对堆栈设置该策略。堆栈策略是一个 JSON 文档，用于定义 CloudFormation 用户可以执行的 CloudFormation 堆栈更新操作以及这些操作应用到的资源。在创建堆栈时，可通过指定一个包含堆栈策略的文本文件或键入该策略来设置堆栈策略。在堆栈上设置堆栈策略时，默认情况下会拒绝未显式允许的任何更新。

您可定义一个带 5 个元素的堆栈策略：`Effect`、`Action`、`Principal`、`Resource` 和 `Condition`。下面的伪代码显示了堆栈策略语法。

```
{
```

```
  "Statement" : [
    {
      "Effect" : "Deny_or_Allow",
      "Action" : "update_actions",
      "Principal" : "*",
      "Resource" : "LogicalResourceId/resource_logical_ID",
      "Condition" : {
        "StringEquals_or_StringLike" : {
          "ResourceType" : [resource_type, ...]
        }
      }
    }
  ]
}
```

Effect

确定是拒绝还是允许对指定资源执行指定的操作。您只能指定 Deny 或 Allow，例如：

```
"Effect" : "Deny"
```

> **Important**
>
> 如果堆栈策略包含重叠语句 (同时允许和拒绝对资源进行更新)，则 Deny 语句始终将覆盖 Allow 语句。要确保某一资源受到保护，请对该资源使用 Deny 语句。

处理建议

指定拒绝或允许的更新操作：

更新:修改

指定在对资源应用更改期间不会中断或有某些中断的更新操作。所有资源都保持其物理 ID。

更新:替换

指定会重新创建资源的更新操作。CloudFormation 使用指定的更新创建新资源，然后删除旧资源。由于资源是重新创建的，因此新资源的物理 ID 可能不相同。

更新:删除

指定删除资源的更新操作。从堆栈模板中完全删除资源的更新都需要此操作。

更新:*

指定所有更新操作。星号是通配符，代表所有更新操作。

以下示例说明如何只指定替换和删除操作：

```
"Action" : ["Update:Replace", "Update:Delete"]
```

要允许除某个更新操作之外的所有更新操作，请使用 NotAction。例如，要允许除 Update:Delete 之外的所有更新操作，请使用 NotAction，如本示例中所示：

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "NotAction" : "Update:Delete",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Principal

Principal 元素指定策略应用于的实体。需要此元素，但仅支持通配符 (*)，这意味着策略应用于所有主体。

资源

指定将应用策略的资源的逻辑 ID。要指定资源类型，请使用 Condition 元素。

要指定一个资源，请使用其逻辑 ID。例如：

```
"Resource" : ["LogicalResourceId/myEC2instance"]
```

您可以对逻辑 ID 使用通配符。例如，如果您对所有相关资源使用一个通用逻辑 ID 前缀，则可使用通配符指定所有资源：

```
"Resource" : ["LogicalResourceId/CriticalResource*"]
```

您还可以使用 NotResource 元素，允许对除您指定资源以外的所有资源进行更新。但是，将 NotResource 与 Effect: Allow 结合使用并不能可靠地阻止对被排除资源的更新。若要防止特定资源被更新，请改用显式 Deny 语句，如下例所示：

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/ProductionDatabase"
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Conditions

指定应用策略的资源类型。要指定特定资源的逻辑 ID，请使用 Resource 元素。

您可以指定资源类型（如所有 EC2 和 RDS 数据库实例），如以下示例所示：

```
{
  "Statement" : [
  {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "Update:*",
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "ResourceType" : ["AWS::EC2::Instance", "AWS::RDS::DBInstance"]
      }
    }
  },
  {
    "Effect" : "Allow",
    "Principal" : "*",
```

```
    "Action" : "Update:*",
    "Resource" : "*"
  }
  ]
}
```

Allow 语句授予对所有资源的更新权限，而 Deny 语句拒绝对 EC2 和 RDS 数据库实例的更新。Deny 语句始终覆盖允许操作。

您可以对资源类型使用通配符。例如，您可以使用通配符拒绝所有 Amazon EC2 资源（如实例、安全组和子网）的更新权限，如以下示例所示：

```
"Condition" : {
  "StringLike" : {
    "ResourceType" : ["AWS::EC2::*"]
  }
}
```

使用通配符时，必须使用 StringLike 条件。

设置堆栈策略

您可以使用控制台或 Amazon CLI 在创建堆栈时应用堆栈策略。您也可以使用 Amazon CLI 将堆栈策略应用于现有堆栈。应用堆栈策略后，无法将其从堆栈中删除，但您可以使用 Amazon CLI 修改该策略。

堆栈策略适用于所有尝试更新堆栈的 CloudFormation 用户。您不能将不同的堆栈策略与不同的用户关联。

有关如何编写堆栈策略的信息，请参阅定义堆栈策略。

在创建堆栈时设置堆栈策略（控制台）

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在屏幕顶部的导航栏中，选择 Amazon Web Services 区域以在其中创建堆栈。
3. 在 CloudFormation Stacks (CloudFormation 堆栈) 页上，选择 Create stack (创建堆栈)。
4. 在“创建堆栈”向导的 Configure stack options (配置堆栈选项) 页面上，展开高级部分，然后选择 Stack policy (堆栈策略)。

5. 指定堆栈策略：

   - 要在控制台中直接编写策略，请选择 Enter stack policy (输入堆栈策略)，然后在文本字段中直接键入堆栈策略。
   - 要使用在单独的文件中定义的策略，请依次选择 Upload a file (上传文件)、Choose file (选择文件)，以选择包含堆栈策略的文件。

在创建堆栈时设置堆栈策略（Amazon CLI）

- 将 create-stack 命令与 `--stack-policy-body` 选项结合使用可键入修改的策略，或将此命令与 `--stack-policy-url` 选项结合使用可指定包含策略的文件。

在现有堆栈上设置堆栈策略（仅限 Amazon CLI）

- 将 set-stack-policy 命令与 `--stack-policy-body` 选项结合使用可键入修改的策略，或将此命令与 `--stack-policy-url` 选项结合使用可指定包含策略的文件。

> **Note**
>
> 要将策略添加到现有堆栈中，您必须有权执行 CloudFormation SetStackPolicy 操作。

更新受保护资源

要更新受保护的资源，可创建一个覆盖堆栈策略并允许对这些资源进行更新的临时策略。在更新堆栈时指定覆盖策略。覆盖策略不会永久更改堆栈策略。

要更新受保护的资源，您必须有权使用 CloudFormation SetStackPolicy 操作。有关设置 CloudFormation 权限的信息，请参阅使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限。

> **Note**
>
> 在堆栈更新期间，CloudFormation 会自动更新依赖其他更新的资源的资源。例如，CloudFormation 会更新引用所更新的资源的资源。CloudFormation 不会对自动更新的资源进行任何物理更改（例如，资源的 ID），但如果这些资源关联了堆栈策略，则您必须具有更新这些资源的权限。

更新受保护的资源（控制台）

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 选择要更新的堆栈，然后依次选择 Stack actions (堆栈操作) 和 Update Stack (更新堆栈)。
3. 如果您尚未修改堆栈模板，请选择 Use current template (使用当前模板)，然后单击 Next (下一步)。如果您已修改模板，请选择 Replace current template (替换当前模板)，然后指定在 Specify template (指定模板) 中指定更新模板的位置：

   - 对于在计算机本地存储的模板，选择 Upload a template file (上传模板文件)。选择 Choose File 以导航到文件并选中它，然后单击 Next。
   - 对于存储在 Amazon S3 存储桶中的模板，请选择 Amazon S3 URL。输入或粘贴模板的 URL，然后单击 Next (下一步)。

     如果模板位于启用了版本控制的存储桶中，则可以通过在 URL 后附加 `?versionId=`*`version-id`* 来指定模板的具体版本。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用启用版本控制的存储桶中的对象。
4. 如果您的模板包含参数，则在 Specify stack details (指定堆栈详细信息) 页上，输入或修改参数值，然后选择 Next (下一步)。

   CloudFormation 使用当前在堆栈中设置的值填充每个参数，使用 `NoEcho` 属性声明的参数除外。可以选择使用现有值来对这些参数使用当前值。

   有关使用 `NoEcho` 来遮蔽敏感信息以及使用动态参数管理密钥的更多信息，请参阅 请勿将凭证嵌入您的模板 最佳实践。
5. 指定覆盖堆栈策略。

   a. 在 Configure stack options (配置堆栈选项) 页上的高级选项部分中，选择 Stack policy (堆栈策略)。
   b. 选择 Upload a file (上传文件)。
   c. 单击 Choose file (选择文件)，然后导航到包含覆盖堆栈策略的文件或键入策略。
   d. 选择下一步。

   覆盖策略必须为您要更新的受保护资源指定 `Allow` 语句。例如，要更新所有受保护资源，可以指定允许所有更新的临时覆盖策略：

```
{
```

```
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

> **Note**
>
> CloudFormation 仅在此更新期间应用覆盖策略。覆盖策略不会永久更改堆栈策略。要修改堆栈策略，请参阅修改堆栈策略。

6. 审查提交的堆栈信息和更改。

   检查您是否已提交正确的信息，例如正确的参数值或模板 URL。如果模板包含 IAM 资源，请选择 I acknowledge that this template may create IAM resources (我确认该模板可能会创建 IAM 资源) 以指定您要在模板中使用 IAM 资源。有关更多信息，请参阅 确认 CloudFormation 模板中的 IAM 资源。

   在预览您的更改部分中，检查 CloudFormation 是否会进行所需的所有更改。例如，检查 CloudFormation 是否添加、移除和修改了您要添加、移除或修改的资源。CloudFormation 通过为堆栈创建更改集来生成此预览。有关更多信息，请参阅 使用更改集更新 CloudFormation 堆栈。

7. 如果您对所做更改感到满意，请单击 Update (更新)。

   > **Note**
   >
   > 此时，您还可以选择查看更改集以更全面地查看建议的更新。要执行此操作，请单击 View change set（查看更改集）而不是 Update（更新）。CloudFormation 会显示根据您的更新生成的更改集。准备好执行堆栈更新后，请单击 Execute (执行)。

   CloudFormation 会显示您的堆栈的 Stack details (堆栈详细信息) 页面。堆栈现在的状态为 `UPDATE_IN_PROGRESS`。CloudFormation 成功完成堆栈更新后，会将堆栈状态设置为 `UPDATE_COMPLETE`。

如果堆栈更新失败，CloudFormation 会自动回滚更改，并将状态设置为 UPDATE_ROLLBACK_COMPLETE。

更新受保护资源（Amazon CLI）

- 将 update-stack 命令与 --stack-policy-during-update-body 选项结合使用可键入修改的策略，或将此命令与 --stack-policy-during-update-url 选项结合使用可指定包含策略的文件。

> **Note**
>
> CloudFormation 仅在此更新期间应用覆盖策略。覆盖策略不会永久更改堆栈策略。要修改堆栈策略，请参阅修改堆栈策略。

修改堆栈策略

要保护其他资源或从资源中删除保护，请修改堆栈策略。例如，当您将要保护的数据库添加到堆栈时，会将该数据库的 Deny 语句添加到堆栈策略。要修改策略，您必须有权使用 SetStackPolicy 操作。

使用 Amazon CLI 修改堆栈策略。

修改堆栈策略（Amazon CLI）

- 将 set-stack-policy 命令与 --stack-policy-body 选项结合使用可键入修改的策略，或将此命令与 --stack-policy-url 选项结合使用可指定包含策略的文件。

您无法删除堆栈策略。要从所有资源删除全部保护，您可修改策略以明确允许对所有资源执行全部操作。以下策略允许对所有资源进行全部更新：

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
```

```
}
```

更多示例堆栈策略

以下示例策略说明如何阻止对所有堆栈资源和特定资源进行更新，并阻止特定类型的更新。

阻止对所有堆栈资源的更新

要阻止对所有堆栈资源的更新，以下策略为所有资源的所有更新操作指定 Deny 语句。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

阻止对单个资源的更新

以下策略拒绝对带 MyDatabase 逻辑 ID 的数据库执行的所有更新操作。它使用 Allow 语句允许对所有其他堆栈资源进行全部更新操作。Allow 语句不应用于 MyDatabase 资源，因为 Deny 语句始终覆盖允许操作。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/MyDatabase"
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
```

```
}
```

⚠ 避免使用 **NotResource** 以保护资源

可考虑将 NotResource 元素与 Effect: Allow 结合使用，通过默认拒绝机制来保护资源，如以下示例所示：

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "NotResource" : "LogicalResourceId/ProductionDatabase"
    }
  ]
}
```

此策略无法可靠地阻止对 ProductionDatabase 资源的更新。AmazonCloudFormation 会分别根据逻辑资源 ID 和资源类型来评估堆栈策略。只有当这两项评估均导致“拒绝”状态时，默认拒绝机制才会阻止更新操作。由于该策略仅将逻辑资源 ID 从 Allow 语句中排除，因此资源类型仍被隐式允许。Amazon于是，CloudFormation 允许进行更新。
请始终使用显式 Deny 语句来保护资源，如前面的示例所示。

阻止对资源类型的所有实例进行更新

以下策略拒绝针对 RDS 数据库实例资源类型的所有更新操作。它使用 Allow 语句允许对所有其他堆栈资源进行全部更新操作。Allow 语句不应用于 RDS 数据库实例资源，因为 Deny 语句始终覆盖允许操作。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
```

```
          "ResourceType" : ["AWS::RDS::DBInstance"]
        }
      }
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

阻止针对实例的替换更新

以下策略拒绝会导致替换具有 MyInstance 逻辑 ID 的实例的更新。它使用 Allow 语句允许对所有其他堆栈资源进行全部更新操作。Allow 语句不应用于 MyInstance 资源，因为 Deny 语句始终覆盖允许操作。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:Replace",
      "Principal": "*",
      "Resource" : "LogicalResourceId/MyInstance"
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

阻止对嵌套堆栈进行更新

以下策略将拒绝针对 CloudFormation 堆栈资源类型（嵌套堆栈）的所有更新操作。它使用 Allow 语句允许对所有其他堆栈资源进行全部更新操作。Allow 语句不会应用于 CloudFormation 堆栈资源，因为 Deny 语句始终会覆盖 allow 操作。

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "ResourceType" : ["AWS::CloudFormation::Stack"]
        }
      }
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Amazon CloudFormation 中的数据保护

Amazon 责任共担模式适用于保护 Amazon CloudFormation 中的数据。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础结构。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 Amazon Web Services 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅Amazon 中国区域法律条款。

出于数据保护目的，建议您保护 Amazon Web Services 账户 凭证并使用 Amazon IAM Identity Center 或 Amazon Identity and Access Management（IAM）设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与 Amazon 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 Amazon CloudTrail 设置 API 和用户活动日记账记录。有关使用 CloudTrail 跟踪来捕获 Amazon 活动的信息，请参阅《Amazon CloudTrail 用户指南》中的使用 CloudTrail 跟踪。
- 使用 Amazon 加密解决方案以及 Amazon Web Services 服务中的所有默认安全控制。

- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://www.amazonaws.cn/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括使用控制台、API、Amazon CLI 或 Amazon SDK 处理 Amazon CloudFormation 或其他 Amazon Web Services 服务 时。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

静态加密

按照 Amazon 责任共担模式，Amazon CloudFormation 会存储您的静态加密的数据。客户负责为其账户中存储的数据设置加密和存储策略。例如，我们建议对存储在 S3 存储桶或 SNS 主题中的模板和其他数据启用静态加密。客户同样为 CloudFormation 预置的任何数据存储系统定义加密设置。

传输中加密

在责任共担模式下，Amazon CloudFormation 使用加密通道进行服务通信。

互联网络流量隐私

默认情况下，区域或可用区之间的 Amazon CloudFormation 服务通信是安全加密的。

使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限

通过 Amazon Identity and Access Management (IAM)，您可以创建 IAM 用户并控制他们对 Amazon Web Services 账户 中特定资源的访问。使用 IAM 时，您可以控制用户对 CloudFormation 执行的操作，例如，是否可以查看堆栈模板、创建堆栈或删除堆栈。

除了 CloudFormation 特定的操作之外，您还可以管理每个用户可以使用哪些 Amazon 服务和资源。这样您可以控制用户在使用 CloudFormation 时可访问的具体资源。例如，您可以指定哪些用户可以创建 Amazon EC2 实例、终止数据库实例或更新 VPC。这些权限在他们使用 CloudFormation 执行这些操作时同样适用。

使用以下各节中的信息来控制谁可以访问 CloudFormation。我们还将探讨如何在模板中授权 IAM 资源创建，为 EC2 实例上运行的应用程序提供所需的权限，以及如何使用临时安全凭证来增强您 Amazon 环境的安全性。

为 CloudFormation 定义基于 IAM 身份的策略

要访问 CloudFormation，您需要创建和分配 IAM 策略，赋予 IAM 身份（如用户或角色）调用所需 API 操作的权限。

通过使用 IAM 基于身份的策略，可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。CloudFormation 支持特定的操作、资源和条件秘钥。

如果您刚接触 IAM ，则需要先熟悉 IAM JSON 策略的要素。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。要了解如何创建 IAM 策略，请完成 IAM 文档中的教程创建并附加您的第一个客户管理型策略。

主题

- CloudFormation 的策略操作
- CloudFormation 的控制台特定操作
- CloudFormation 的策略资源
- CloudFormation 的策略条件键

CloudFormation 的策略操作

在 IAM 策略语句的 `Action` 要素中，您可以指定 CloudFormation 所提供的任意 API 操作。您必须在操作名称前加上小写字符串 `cloudformation:`。例如：`cloudformation:CreateStack`、`cloudformation:CreateChangeSet` 和 `cloudformation:UpdateStack`。

要在单个语句中指定多个操作，请使用逗号分隔，如下所示：

```
"Action": [ "cloudformation:action1", "cloudformation:action2" ]
```

您也可以使用通配符指定多项操作。例如，您可以指定名称以单词 `Get` 开头的所有操作，如下所示：

```
"Action": "cloudformation:Get*"
```

要查看与 cloudformation 服务前缀相关的操作的完整列表，请参阅《服务授权参考》中 Amazon CloudFormation 的操作、资源和条件密钥以及 Amazon 云端控制 API 的操作、资源和条件密钥。

示例

以下内容介绍了授予查看 CloudFormation 堆栈权限的权限策略示例。

Example 1：授予查看堆栈权限的示例策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "cloudformation:DescribeStacks",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStackResource",
            "cloudformation:DescribeStackResources"
        ],
        "Resource":"*"
    }]
}
```

创建或删除堆栈的用户需要根据其堆栈模板获得额外权限。例如，如果您的模板描述了 Amazon SQS 队列，则用户必须同时拥有 CloudFormation 和 Amazon SQS 操作的权限，如以下策略示例所示。

Example 2：授予创建和查看堆栈操作以及所有 Amazon SQS 操作的示例策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "sqs:*",
            "cloudformation:CreateStack",
            "cloudformation:DescribeStacks",
```

```
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStackResources",
            "cloudformation:GetTemplate",
            "cloudformation:ValidateTemplate"
        ],
        "Resource":"*"
    }]
}
```

CloudFormation 的控制台特定操作

除了 Amazon Command Line Interface 或 CloudFormation API 所需的权限外，CloudFormation 控制台的用户还需要额外的权限。这些权限支持特定于控制台的功能，例如将模板上传到 Amazon S3 存储桶以及 Amazon 特定参数类型的下拉列表。

对于下面列出的所有操作，请向所有资源授予权限；不要将其限制于特定的堆栈或存储桶。

以下操作仅可通过 CloudFormation 控制台使用，未记录在 API 参考中。该操作允许用户将模板上传到 Amazon S3 存储桶。

- `cloudformation:CreateUploadBucket`

当用户上传模板时，还需要以下 Amazon S3 权限：

- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetObject`
- `s3:CreateBucket`

要查看具有 Amazon 特定参数类型的模板的参数下拉列表中的值，用户需要有进行对应的描述 API 调用的权限。例如，在模板中使用这些参数类型时需要以下权限：

- `ec2:DescribeKeyPairs` – `AWS::EC2::KeyPair::KeyName` 参数类型所必需。
- `ec2:DescribeSecurityGroups` – `AWS::EC2::SecurityGroup::Id` 参数类型所必需。
- `ec2:DescribeSubnets` – `AWS::EC2::Subnet::Id` 参数类型所必需。
- `ec2:DescribeVpcs` – `AWS::EC2::VPC::Id` 参数类型所必需。

有关 Amazon 特定类型的更多信息，请参阅 使用 CloudFormation 提供的参数类型在运行时指定现有资源。

CloudFormation 的策略资源

在 IAM policy 声明中，Resource 元素指定了该声明涵盖的一个或多个对象。对于 CloudFormation，每个 IAM 策略语句适用于您使用 Amazon 资源名称 (ARN) 指定的资源。具体 ARN 格式取决于资源。

有关 CloudFormation 资源类型及其 ARN 的完整列表，请参阅服务授权参考中 Amazon CloudFormation 定义的资源类型。要了解您可以通过每个资源的 ARN 指定哪些操作，请参阅 Amazon CloudFormation 定义的操作。

如以下策略示例所示，您可以为特定堆栈指定操作。当您提供 ARN 时，请将 *placeholder text* 替换为特定于资源的信息。

Example 1：拒绝指定堆栈的删除和更新堆栈操作的策略示例

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "cloudformation:DeleteStack",
                "cloudformation:UpdateStack"
            ],
            "Resource": "arn:aws:cloudformation:us-
east-1:111122223333:stack/MyProductionStack/*"
        }
    ]
}
```

上述策略在堆栈名称末尾使用了一个通配符，因此删除堆栈和更新堆栈都会被拒绝，无论是整个堆栈 ID（如 arn:aws:cloudformation:*region*:*account-id*:stack/*MyProductionStack*/*abc9dbf0-43c2-11e3-a6e8-50fa526be49c*）还是堆栈名称（如 *MyProductionStack*）。

要允许 AWS::Serverless 转换创建变更集，请包含 arn:aws:cloudformation:*region*:aws:transform/Serverless-2016-10-31 资源级权限，如以下策略所示。

Example 2：允许为指定转换执行创建更改集操作的示例策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet"
            ],
            "Resource": "arn:aws:cloudformation:us-east-1:aws:transform/
Serverless-2016-10-31"
        }
    ]
}
```

CloudFormation 的策略条件键

在 IAM 策略语句中，您可以选择性地指定控制策略生效时间的条件。例如，您可以定义一条策略，使用户只能在指定了特定的模板 URL 时才能创建堆栈。您可以定义特定于 CloudFormation 的条件和 Amazon 范围的条件，例如指定策略停止生效的时间的 DateLessThan 条件。有关 Amazon 范围条件列表的更多信息，请参阅《IAM 用户指南》中 IAM 策略元素参考中的“条件”。

> **Note**
>
> 不要使用 aws:SourceIp Amazon 范围的条件。CloudFormation 使用自身的 IP 地址（而不是原始请求的 IP 地址）来预置资源。例如在创建堆栈时，CloudFormation 会从其 IP 地址发送请求来启动 Amazon EC2 实例或创建 Amazon S3 存储桶，而不是从来自 CreateStack 调用或 create-stack 命令的 IP 地址发送请求。

以下列表介绍特定于 CloudFormation 的条件。这些条件仅在用户创建或更新堆栈时应用：

`cloudformation:ChangeSetName`

要与策略关联的 CloudFormation 更改集名称。使用此条件可控制 用户可执行或删除的更改集。

`cloudformation:ImportResourceTypes`

要与策略关联的模板资源类型，如 `AWS::EC2::Instance`。可以使用该条件控制在 用户将资源导入到堆栈时可以使用的资源类型。系统将核对此条件和用户在 `ResourcesToImport` 参数（当前只支持 Amazon CLI 和 API 请求）中声明的资源类型。在使用该参数时，必须指定您希望用户在导入操作期间控制的所有资源类型。有关 `ResourcesToImport` 参数的更多信息，请参阅《Amazon CloudFormation API 参考》中的 CreateChangeSet 操作。

有关可能的 `ResourcesToImport` 列表，请参阅资源类型支持。

使用包含三个部分的资源命名约定来指定用户可以使用的资源类型，上至组织中的所有资源，下至单个资源类型。

organization::*

指定给定组织的所有资源类型。

organization::*service_name*::*

指定给定组织内指定服务的所有资源类型。

organization::*service_name*::*resource_type*

指定一个特定的资源类型。

例如：

AWS::*

指定所有支持的 Amazon 资源类型。

AWS::*service_name*::*

为特定的 Amazon 服务指定所有支持的资源。

AWS::*service_name*::*resource_type*

指定特定的 Amazon 资源类型，如 `AWS::EC2::Instance`（所有 EC2 实例）。

`cloudformation:ResourceTypes`

要与策略关联的模板资源的类型，如 `AWS::EC2::Instance`。使用此条件控制 用户在创建或更新堆栈时可以使用的资源类型。系统将核对此条件和用户在 `ResourceTypes` 参数（当前只支持

Amazon CLI 和 API 请求）中声明的资源类型。使用此参数时，用户必须指定其模板中的所有资源类型。有关 `ResourceTypes` 参数的更多信息，请参阅《Amazon CloudFormation API 参考》中的 CreateStack 操作。

要查看资源类型的列表，请参阅《Amazon CloudFormation 模板参考指南》。

使用包含三个部分的资源命名约定来指定用户可以使用的资源类型，上至组织中的所有资源，下至单个资源类型。

organization::*

指定给定组织的所有资源类型。

organization::*service_name*::*

指定给定组织内指定服务的所有资源类型。

organization::*service_name*::*resource_type*

指定一个特定的资源类型。

例如：

AWS::*

指定所有支持的 Amazon 资源类型。

AWS::*service_name*::*

为特定的 Amazon 服务指定所有支持的资源。

AWS::*service_name*::*resource_type*

指定特定的 Amazon 资源类型，如 `AWS::EC2::Instance`（所有 EC2 实例）。

Alexa::ASK::*

指定 Alexa Skill Kit 中的所有资源类型。

Alexa::ASK::Skill

指定单个 Alexa::ASK::Skill 资源类型。

Custom::*

指定所有自定义资源。

有关更多信息，请参阅 使用自定义资源创建自定义预置逻辑。

`Custom::resource_type`

指定特定的自定义资源类型。

有关更多信息，请参阅 使用自定义资源创建自定义预置逻辑。

`cloudformation:RoleARN`

要与策略关联的 IAM 服务角色的 Amazon 资源名称（ARN）。使用此条件可控制 用户在处理堆栈或更改集时可使用的服务角色。

`cloudformation:StackPolicyUrl`

要与策略关联的 Amazon S3 堆栈策略 URL。使用此条件控制在创建或更新堆栈操作期间 用户可将哪些堆栈策略关联到堆栈。有关堆栈策略的更多信息，请参阅防止更新堆栈资源。

> **Note**
>
> 要确保用户只能使用您上传的堆栈策略创建或更新堆栈，请针对这些用户将 S3 存储桶设为只读。

`cloudformation:TemplateUrl`

要与策略关联的 Amazon S3 模板 URL。使用此条件控制 用户在创建或更新堆栈时可以使用的模板。

> **Note**
>
> 要确保用户只能使用您上传的模板创建或更新堆栈，请针对这些用户将 S3 存储桶设为只读。

> **Note**
>
> 以下 CloudFormation 特定条件适用于同名的 API 参数：
>
> - `cloudformation:ChangeSetName`
> - `cloudformation:RoleARN`

- `cloudformation:StackPolicyUrl`
- `cloudformation:TemplateUrl`

例如，`cloudformation:TemplateUrl` 仅适用于 `CreateStack`、`UpdateStack` 和 `CreateChangeSet` API 的 `TemplateUrl` 参数。

有关使用条件密钥控制访问的 IAM 策略示例，请参阅 CloudFormation 基于 IAM 身份的策略示例。

确认 CloudFormation 模板中的 IAM 资源

在创建堆栈之前，CloudFormation 会验证您的模板。在验证期间，CloudFormation 会检查模板中可能创建的 IAM 资源。IAM 资源（如具有完全访问权限的 IAM 用户）可以访问和修改您的 Amazon Web Services 账户 中的任何资源。因此，建议您在继续之前检查与每个 IAM 资源关联的权限，以便您不会无意中创建具有升级权限的资源。为了确保您已执行此操作，您必须确认模板包含这些资源，从而在 CloudFormation 创建堆栈之前为其提供指定功能。

您可以使用 CloudFormation 控制台、Amazon Command Line Interface（Amazon CLI）或 API 来确认 CloudFormation 模板的功能：

- 在 CloudFormation 控制台中，在“创建堆栈”或“更新堆栈”向导的配置堆栈选项页面上，选择我确认此模板可能创建 IAM 资源。
- 在 Amazon CLI 中，当您使用 create-stack 和 update-stack 命令时，请指定 `--capabilities` 选项的 `CAPABILITY_IAM` 或 `CAPABILITY_NAMED_IAM` 值。如果您的模板包含 IAM 资源，则可以指定任一功能。如果您的模板包含 IAM 资源的自定义名称，则必须指定 `CAPABILITY_NAMED_IAM`。
- 在 API 中，当您使用 CreateStack 和 UpdateStack 操作时，请指定 `Capabilities.member.1=CAPABILITY_IAM` 或 `Capabilities.member.1=CAPABILITY_NAMED_IAM`。如果您的模板包含 IAM 资源，则可以指定任一功能。如果您的模板包含 IAM 资源的自定义名称，则必须指定 `CAPABILITY_NAMED_IAM`。

Important

如果您的模板包含自定义命名的 IAM 资源，请不要重用同一模板来创建多个堆栈。IAM 资源在您的账户内必须是全局唯一的。如果使用同一模板在不同的区域中创建多个堆栈，则您的堆栈可能会共享相同 IAM 资源，而不是每个堆栈均具有唯一资源。在堆栈之间共享的资源可能会造成无法恢复的意外后果。例如，如果您在一个堆栈中删除或更新共享的 IAM 资源，则将无意中修改其他堆栈的资源。

管理 Amazon EC2 实例上运行的应用程序的凭证

如果您有应用程序在 Amazon EC2 实例上运行，并且需要向 Amazon 资源（如 Amazon S3 桶或 DynamoDB 表）发出请求，则应用程序会要求提供 Amazon 安全证书。但是，在启动的每个实例中分配并嵌入长期安全凭证非常困难，而且存在潜在安全风险。建议您创建在启动 Amazon EC2 实例时与该实例关联的 IAM 角色，而不是使用长期凭证，如 IAM 用户凭证。然后，应用程序可以从 Amazon EC2 实例获取临时安全凭证。您不必在实例上嵌入长期凭证。此外，为简化凭证管理工作，您可以为多个 Amazon EC2 实例指定一个角色；无需为每个实例创建唯一凭证。

有关演示如何使用角色启动实例的模板代码段，请参阅 IAM 角色模板示例。

> **Note**
>
> 实例上使用临时安全凭证的应用程序可以调用任何 CloudFormation 操作。但由于 CloudFormation 会与多种其他 Amazon 服务交互，您必须确定要使用的所有服务都支持临时安全凭证。有关接受临时安全凭证的服务列表，请参阅 IAM 用户指南中的使用 IAM 的 Amazon 服务。

授予临时访问权限（联合访问）

在某些情况下，您可能希望授予没有 Amazon 凭证的用户对您的 Amazon Web Services 账户 的临时访问权限。您可以使用 Amazon Security Token Service (Amazon STS)，而不必在每次授予临时访问权限时创建和删除长期证书。例如，您可以使用 IAM 角色。通过一个 IAM 角色，您可以通过编程方式创建并分配很多临时安全凭证 (包括访问密钥、秘密访问密钥和安全令牌)。这些凭证的有效期有限，因此过期后不能用于访问您的 Amazon Web Services 账户。您也可以创建多个 IAM 角色，向每个用户授予不同级别的权限。IAM 角色对于联合身份和单点登录等情况非常有用。

联合身份是可以跨多个系统使用的独特身份。对于已建立本地身份系统 (如 LDAP 或 Active Directory) 的企业用户，可以使用本地身份系统处理所有身份验证。在对用户进行身份验证后，您可从相应的 IAM 用户或角色提供临时安全凭证。例如，您可以创建一个管理员角色和一个开发人员角色，其中管理员对 Amazon 账户拥有完全访问权限，而开发人员只拥有使用 CloudFormation 堆栈的权限。经过身份验证后，管理员有权获取管理员角色的临时安全凭证。而开发人员只能获取开发人员角色的临时安全凭证。

您还可以授予联合用户对 Amazon Web Services 管理控制台的访问权限。使用本地身份系统对用户进行身份验证后，您可以通过编程来构造一个临时 URL 以提供对 Amazon Web Services 管理控制台的直接访问。用户使用临时 URL 时无需登录 Amazon，因为他们已经过身份验证（单一登录）。此外，

因为 URL 是从用户的临时安全凭证构造的，所以通过这些凭证提供的权限可确定用户在 Amazon Web Services 管理控制台中拥有的权限。

您可使用多种不同的 Amazon STS API 来生成临时安全凭证。有关使用哪个 API 的更多信息，请参阅《IAM 用户指南》中的 比较 Amazon STS 凭证。

> **Important**
>
> 在使用从 `GetFederationToken` API 生成的临时安全证书时，您无法使用 IAM。相反，如果您需要使用 IAM，请使用来自角色的临时安全凭证。

CloudFormation 会与多种其他 Amazon 服务交互。将临时安全凭证用于 CloudFormation 时，请确认要使用的所有服务都支持临时安全凭证。有关接受临时安全凭证的服务列表，请参阅 IAM 用户指南中的使用 IAM 的 Amazon 服务。

有关更多信息，请参阅《IAM 用户指南》中的以下相关资源：

- 临时凭证的常见情形
- 使自定义身份凭证代理程序能够访问 Amazon 控制台

CloudFormation 基于 IAM 身份的策略示例

默认情况下，用户和角色没有创建或修改 CloudFormation 资源的权限。他们也无法使用 Amazon Web Services 管理控制台、Amazon Command Line Interface (Amazon CLI) 或 Amazon API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。有关更多信息，请参阅 为 CloudFormation 定义基于 IAM 身份的策略。

以下示例介绍了您可用于允许或拒绝使用一个或多个 CloudFormation 操作的权限的策略语句。

主题

- 需要特定的模板 URL
- 拒绝所有 CloudFormation 导入操作
- 允许对特定资源类型进行导入操作
- 拒绝堆栈模板中的 IAM 资源
- 允许使用特定资源类型创建堆栈
- 根据资源变更型 API 操作控制访问权限

- 根据区域和资源类型限制堆栈集操作
- 允许所有 IaC 生成器操作

需要特定的模板 URL

以下策略只授予使用 *https://s3.amazonaws.com/amzn-s3-demo-bucket/test.template* 模板 URL 创建或更新堆栈的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudformation:TemplateUrl": [
                        "https://s3.amazonaws.com/amzn-s3-demo-bucket/
test.template"
                    ]
                }
            }
        }
    ]
}
```

拒绝所有 CloudFormation 导入操作

以下策略授予完成 CloudFormation 所有操作（导入操作除外）的权限。

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllStackOperations",
      "Effect": "Allow",
      "Action": "cloudformation:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyImport",
      "Effect": "Deny",
      "Action": "cloudformation:*",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "cloudformation:ImportResourceTypes": [
            "*"
          ]
        }
      }
    }
  ]
}
```

允许对特定资源类型进行导入操作

以下策略授予所有堆栈操作权限，以及仅对指定资源（本例中为 `AWS::S3::Bucket`）进行导入操作的权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowImport",
      "Effect": "Allow",
      "Action": "cloudformation:*",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEqualsIgnoreCase": {
```

```
          "cloudformation:ImportResourceTypes": [
            "AWS::S3::Bucket"
          ]
        }
      }
    }
  ]
}
```

拒绝堆栈模板中的 IAM 资源

以下策略授予创建堆栈的权限，但如果堆栈模板包含 IAM 服务中的任何资源，则拒绝请求。此策略还要求用户指定 `ResourceTypes` 参数（仅适用于 Amazon CLI 和 API 请求）。此策略使用显式拒绝语句，以便在任何其他策略授予额外权限时，此策略始终保持有效（显式拒绝语句始终覆盖显式允许语句）。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect" : "Allow",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*"
    },
    {
      "Effect" : "Deny",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*",
      "Condition" : {
        "ForAnyValue:StringLikeIfExists" : {
          "cloudformation:ResourceTypes" : [ "AWS::IAM::*" ]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource": "*",
      "Condition": {
```

```
        "Null": {
          "cloudformation:ResourceTypes": "true"
        }
      }
    }
  ]
}
```

允许使用特定资源类型创建堆栈

以下策略与上一示例类似。除非堆栈模板包含 IAM 服务中的任何资源，否则该策略将授予创建堆栈的权限。它还要求用户指定 ResourceTypes 参数（仅适用于 Amazon CLI 和 API 请求）。此策略更简单，但不使用显式拒绝语句。其他授予额外权限的策略可以覆盖此策略。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect" : "Allow",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*",
      "Condition" : {
        "ForAllValues:StringNotLikeIfExists" : {
          "cloudformation:ResourceTypes" : [ "AWS::IAM::*" ]
        },
        "Null":{
          "cloudformation:ResourceTypes": "false"
        }
      }
    }
  ]
}
```

根据资源变更型 API 操作控制访问权限

以下策略授予根据资源变更型 API 操作名称筛选访问的权限。这可以控制 IAM 用户可以使用哪些 API 在堆栈或堆栈集上添加或删除标签。用于添加或删除标签的操作应作为条件键的值进行添加。以下策略会授予对变更操作 `CreateStack` 的 `TagResource` 权限和 `UntagResource` 权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "CreateActionConditionPolicyForTagUntagResources",
        "Effect": "Allow",
        "Action": [
            "cloudformation:TagResource",
            "cloudformation:UntagResource"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "cloudformation:CreateAction": [
                    "CreateStack"
                ]
            }
        }
    }]
}
```

根据区域和资源类型限制堆栈集操作

以下策略授予服务托管堆栈集权限。具有此策略的用户只能对其模板包含 Amazon S3 资源类型（`AWS::S3::*`）或 `AWS::SES::ConfigurationSet` 资源类型的堆栈集执行操作。如果使用 ID 123456789012 登录到组织管理账户，则用户还可以仅对以 OU（ID 为 *ou-1fsfsrsdsfrewr*）为目标的堆栈集执行操作，并可以仅对以 Amazon Web Services 账户（ID 为 *987654321012*）为目标的堆栈集（ID 为 `stack-set-id`）执行操作。

如果堆栈集模板包含除策略中指定的资源类型以外的其他资源类型，或者如果部署的目标 OU 或账户 ID 不是在对应管理账户和堆栈集的策略中指定的内容，则堆栈集操作将失败。

这些策略限制仅在堆栈集操作以 us-east-1、us-west-2 或 eu-west-2 Amazon Web Services 区域 为目标时适用。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stackset/*",
                "arn:aws:cloudformation:*:*:type/resource/AWS-S3-*",
                "arn:aws:cloudformation:us-west-2:111122223333:type/resource/AWS-SES-ConfigurationSet",
                "arn:aws:cloudformation:*:111122223333:stackset-target/*/ou-1fsfsrsdsfrewr",
                "arn:aws:cloudformation:*:111122223333:stackset-target/stack-set-id/444455556666"
            ],
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudformation:TargetRegion": [
                        "us-east-1",
                        "us-west-2",
                        "eu-west-1"
                    ]
                }
            }
        }
    ]
}
```

允许所有 IaC 生成器操作

以下策略允许访问与 IaC 生成器资源扫描和模板管理相关的 CloudFormation 操作。第一条语句授予描述、列出和启动资源扫描的权限。它还允许访问其他必要的权限（cloudformation:GetResource、cloudformation:ListResources 和

cloudformation:ListTypes）、使 IaC 生成器能够检索有关资源和可用资源类型的信息。第二条语句授予创建、删除、描述、列出和更新所生成模板的全部权限。

您还必须向使用 IaC 生成器扫描资源的任何人授予目标 Amazon 服务的读取权限。有关更多信息，请参阅 扫描资源所需的 IAM 权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ResourceScanningOperations",
            "Effect":"Allow",
            "Action":[
                "cloudformation:DescribeResourceScan",
                "cloudformation:GetResource",
                "cloudformation:ListResources",
                "cloudformation:ListResourceScanRelatedResources",
                "cloudformation:ListResourceScanResources",
                "cloudformation:ListResourceScans",
                "cloudformation:ListTypes",
                "cloudformation:StartResourceScan"
            ],
            "Resource":"*"
        },
        {
            "Sid":"TemplateGeneration",
            "Effect":"Allow",
            "Action":[
                "cloudformation:CreateGeneratedTemplate",
                "cloudformation:DeleteGeneratedTemplate",
                "cloudformation:DescribeGeneratedTemplate",
                "cloudformation:GetResource",
                "cloudformation:GetGeneratedTemplate",
                "cloudformation:ListGeneratedTemplates",
                "cloudformation:UpdateGeneratedTemplate"
            ],
            "Resource":"*"
        }
    ]
}
```

适用于 Amazon CloudFormation 的 Amazon 托管式策略

Amazon 托管式策略是由 Amazon 创建和管理的独立策略。Amazon 托管式策略旨在为许多常见使用案例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住，Amazon 托管式策略可能不会为您的特定使用案例授予最低权限，因为它们可供所有 Amazon 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 Amazon 托管式策略中定义的权限。如果 Amazon 更新在 Amazon 托管策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 Amazon 服务启动或新的 API 操作可用于现有服务时，Amazon 最有可能更新 Amazon 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管式策略。

Amazon 托管式策略：AWSCloudFormationFullAccess

您可以将 `AWSCloudFormationFullAccess` 附加到您的用户、组和角色。

此策略授予的权限允许针对所有 CloudFormation 操作和资源的完全访问。

权限详细信息

该策略包含以下权限。

- `cloudformation` – 允许主体对所有资源执行所有 CloudFormation 操作。

要查看此策略的权限，请参阅《Amazon 托管式策略参考指南》中的 AWSCloudFormationFullAccess。

Amazon 托管式策略：AWSCloudFormationReadOnlyAccess

您可以将 `AWSCloudFormationReadOnlyAccess` 附加到您的用户、组和角色。

此策略授予的权限允许针对 CloudFormation 操作和资源的只读访问。

权限详细信息

该策略包含以下权限。

- `cloudformation` – 允许主体执行只读 CloudFormation 操作，例如描述堆栈、列出资源和查看模板，但不允许创建、更新或删除堆栈。

要查看此策略的权限，请参阅《Amazon 托管式策略参考指南》中的 AWSCloudFormationReadOnlyAccess。

CloudFormation 对 Amazon 托管式策略的更新

查看有关 CloudFormation 的 Amazon 托管式策略更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提示，请订阅 CloudFormation 文档历史记录页面上的 RSS 源。

更改	描述	日期
AWSCloudFormationReadOnlyAccess – 更新现有策略	CloudFormation 添加了新权限，以允许使用 `cloudformation:BatchDescribe*` 操作执行批量描述操作。	2026 年 1 月 30 日
AWSCloudFormationReadOnlyAccess – 更新现有策略	CloudFormation 添加了新权限，以允许使用 `cloudformation:Detect*` 操作来使用堆栈偏差检测功能。	2019 年 11 月 13 日
AWSCloudFormationReadOnlyAccess – 更新现有策略	CloudFormation 添加了新权限，以允许 `cloudformation:EstimateTemplateCost` 、`cloudformation:Get*` 和 `cloudformation:ValidateTemplate` 操作。	2017 年 11 月 2 日
AWSCloudFormationFullAccess – 新策略	CloudFormation 添加了新的 Amazon 托管式策略，此策略提供对 CloudFormation 操作和资源的完全访问权限。	2019 年 7 月 26 日
AWSCloudFormationReadOnlyAccess – 更新现有策略	CloudFormation 添加了新权限，以允许使用 `cloudformation:DetectStackDrift` 和 `cloudformation:DetectStackR`	2019 年 2 月 6 日

更改	描述	日期
	esourceDrift 操作来进行堆栈偏差检测。	
AWSCloudFormationReadOnlyAccess – 新策略	CloudFormation 添加了新的 Amazon 托管式策略，此策略提供对 CloudFormation 操作和资源的只读访问权限。	2015 年 2 月 6 日
CloudFormation 开启了更改跟踪。	CloudFormation 为其 Amazon 托管式策略开启了更改跟踪。	2015 年 2 月 6 日

Amazon CloudFormation 服务角色

服务角色是一个 Amazon Identity and Access Management（IAM）角色，允许 CloudFormation 代表您调用堆栈中的资源。您可以指定一个将允许 CloudFormation 创建、更新或删除堆栈资源的 IAM 角色。默认情况下，CloudFormation 会使用根据您的用户凭证为堆栈操作生成的临时会话。如果指定了服务角色，CloudFormation 将使用该角色的凭证。

使用服务角色可以显式指定 CloudFormation 可执行的操作，这些操作可能不会总是与您或其他用户可执行的操作相同。例如，您可能具有管理员权限，但您可以限制 CloudFormation 的访问权限以仅允许 Amazon EC2 操作。

您可以使用 IAM 服务创建服务角色及其权限策略。有关创建服务角色的更多信息，请参阅《IAM 用户指南》中的创建向 Amazon 服务委派权限的角色。将 CloudFormation (`cloudformation.amazonaws.com`) 指定为可代入该角色的服务。

要将服务角色与堆栈关联，请在创建堆栈时指定角色。有关更多信息，请参阅 配置堆栈选项。您还可以在控制台中更新堆栈时或通过 API DeleteStack 堆栈时更改服务角色。在指定服务角色之前，请确保您具有传递该角色的权限 (`iam:PassRole`)。`iam:PassRole` 权限指定您可以使用哪些角色。有关更多信息，请参阅《IAM 用户指南》中的向用户授予权限以将角色传递给 Amazon 服务。

> **Important**
>
> 在您指定服务角色后，CloudFormation 在该堆栈上执行的所有操作将始终使用该角色。创建堆栈后，无法删除附加到堆栈的服务角色。拥有权限，可对此堆栈执行操作的其他用户可以使用该角色，无论这些用户是否拥有 `iam:PassRole` 权限。如果该角色包含用户不应具有的权

限，则您可能无意中提升了用户的权限。确保该角色授予最小权限。有关更多信息，请参阅《IAM 用户指南》中的应用最低权限许可。

防止跨服务混淆代理

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，Amazon 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

建议在资源策略中使用 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键，以限制 Amazon CloudFormation 为其他服务提供的对特定资源（例如 Amazon CloudFormation 扩展）的访问权限。如果您只希望将一个资源与跨服务访问相关联，请使用。`aws:SourceArn`如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用。`aws:SourceAccount`

确保 `aws:SourceArn` 的值为 Amazon CloudFormation 存储的资源的 ARN。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符（`*`）的 `aws:SourceArn` 全局上下文条件键。例如 `arn:aws:`*`cloudformation`*`:*:`*`123456789012`*`:*`。

如果 `aws:SourceArn` 值不包含账户 ID，则您必须使用两个全局条件上下文键来限制权限。

以下示例演示如何使用 Amazon CloudFormation 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键来防范混淆代理问题。

使用 `aws:SourceArn` 和 `aws:SourceAccount` 条件键的信任策略示例

对于注册表服务，Amazon CloudFormation 调用 Amazon Security Token Service（Amazon STS）以在您的账户中担任服务角色。此角色是为 RegisterType 操作中的 `ExecutionRoleArn` 以及 LoggingConfig 操作中的 `LogRoleArn` 集而配置。有关更多信息，请参阅 配置具有 IAM 权限的执行角色和公有扩展访问权限的信任策略。

此示例角色信任策略使用条件语句，将服务角色的 `AssumeRole` 功能限制为只能对指定账户中的指定 Amazon CloudFormation 扩展执行操作。`aws:SourceArn` 和 `aws:SourceAccount` 条件会得到独立评估。使用服务角色的任何请求都必须满足这两个条件。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "resources.cloudformation.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:cloudformation:us-east-1:123456789012:type/
resource/Organization-Service-Resource"
        }
      }
    }
  ]
}
```

附加信息

有关对 StackSets 使用的服务角色使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键的策略示例，请参阅设置全局键以缓解混淆代理问题。

有关更多信息，请参阅《IAM 用户指南》中更新角色信任策略。

转发访问会话（FAS）请求和权限评估

用户在创建、更新和删除 CloudFormation 堆栈时可以选择指定 IAM 角色 ARN。如果用户未提供任何角色，CloudFormation 会使用默认服务机制与其他 Amazon 服务进行交互。在这种情况下，调用者必须拥有对所管理资源的必要权限。如果用户提供自己的 IAM 角色，CloudFormation 会担任该角色代表用户执行服务交互。

无论用户是否提供 IAM 角色，CloudFormation 都会为每个资源操作生成一个新的范围缩小的 FAS 令牌。因此，在这两种情况下，都会填充 FAS 相关条件键（包括 aws:ViaAWSService）。

FAS 的使用会影响 CloudFormation 操作期间 IAM 策略的评估方式。如果有资源受到 FAS 相关条件键的影响，使用包含该资源的模板创建堆栈时，可能会发生权限被拒绝的情况。

示例 IAM 策略

可考虑使用以下 IAM 策略。Statement2 会始终阻止在 CloudFormation 中创建 AWS::KMS::Key 资源。无论用户在堆栈操作期间是否提供 IAM 角色，系统都会始终强制执行该限制。这是因为由于 FAS 的使用，aws:ViaAWSService 条件键会始终设置为 true。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": [
                "kms:CreateKey"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Statement2",
            "Effect": "Deny",
            "Action": [
                "kms:CreateKey"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        }
```

```
    ]
}
```

示例堆栈模板

例如，如果用户使用以下示例模板创建堆栈，aws:ViaAWSService 设置为 true，角色权限会被 FAS 策略覆盖。拒绝 CreateKey 操作的 IAM 策略的 Statement2 会影响堆栈创建。这会导致出现权限被拒绝的错误。

```
Resources:
  myPrimaryKey:
    Type: AWS::KMS::Key
    Properties:
      Description: An example multi-Region primary key
      KeyPolicy:
        Version: '2012-10-17'
        Id: key-default-1
        Statement:
          - Sid: Enable IAM User Permissions
            Effect: Allow
            Principal:
              AWS: !Join
                - ''
                - - 'arn:aws:iam::'
                  - !Ref AWS::AccountId
                  - ':root'
            Action: kms:*
            Resource: '*'
```

有关 FAS 的更多信息，请参阅《IAM 用户指南》中的转发访问会话。

> **Note**
>
> 大多数资源都遵循此模式。但是，如果您在创建、更新或删除资源时遇到意外成功或失败，并且您的 IAM 策略包含 FAS 相关条件键，则该资源很可能是不遵循此标准模式的一小部分资源。

使用 Amazon CloudTrail 记录 Amazon CloudFormation API 调用

Amazon CloudFormation 与 Amazon CloudTrail 集成，后者是一项提供用户、角色或 Amazon 服务在 CloudFormation 中所执行操作的记录的服务。CloudTrail 将对 CloudFormation 的所有 API 调用均作为事件捕获，包括来自 CloudFormation 控制台的调用和对 CloudFormation API 的代码调用。如果您创建了一个跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 CloudFormation 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的事件历史记录中查看最新事件。通过使用 CloudTrail 收集的信息，您可以确定向 CloudFormation 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《Amazon CloudTrail 用户指南》。

主题

- CloudTrail 中的 CloudFormation 信息
- 了解 CloudFormation 日志文件条目

CloudTrail 中的 CloudFormation 信息

在您创建 Amazon 账户时，将在该账户上启用 CloudTrail。当 CloudFormation 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 Amazon 服务事件一同保存在事件历史记录中。您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅 使用 CloudTrail 事件历史记录查看事件。

要持续记录 Amazon 账户中的事件（包括 CloudFront 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 CloudFormation 操作均由 CloudTrail and 进行记录，并记录在《Amazon CloudFormation API 参考》中。例如，对 `CreateStack`、`DeleteStack` 和 `ListStacks` 部分的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 CloudFormation 日志文件条目

跟踪记录是一种配置，可用于将事件作为日志文件传送到您指定的 Simple Storage Service（Amazon S3）存储桶。CloudTrail 日志文件包含一个或多个记录条目。一个事件表示一个来自任何源的请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公有 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateStack` 操作。该操作由名为 Alice 的 IAM 用户执行。

> **Note**
>
> 仅记录输入参数键名称，不记录参数值。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:02:43Z",
  "eventSource": "cloudformation.amazonaws.com",
```

```
  "eventName": "CreateStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "templateURL": "templateURL",
    "tags": [
      {
        "key": "test",
        "value": "tag"
      }
    ],
    "stackName": "my-test-stack",
    "disableRollback": true,
    "parameters": [
      {
        "parameterKey": "password"
      },
      {
        "parameterKey": "securitygroup"
      }
    ]
  },
  "responseElements": {
    "stackId": "arn:aws:cloudformation:us-east-1:012345678910:stack/my-test-stack/
a38e6a60-b397-11e3-b0fc-08002755629e"
  },
  "requestID": "9f960720-b397-11e3-bb75-a5b75389b02d",
  "eventID": "9bf6cfb8-83e1-4589-9a70-b971e727099b"
}
```

以下示例显示 Alice 对 `my-test-stack` 堆栈调用了 UpdateStack 操作：

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
```

```
  "eventTime": "2014-03-24T21:04:29Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "UpdateStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "templateURL": "templateURL",
    "parameters": [
      {
        "parameterKey": "password"
      },
      {
        "parameterKey": "securitygroup"
      }
    ],
    "stackName": "my-test-stack"
  },
  "responseElements": {
    "stackId": "arn:aws:cloudformation:us-east-1:012345678910:stack/my-test-stack/
a38e6a60-b397-11e3-b0fc-08002755629e"
  },
  "requestID": "def0bf5a-b397-11e3-bb75-a5b75389b02d",
  "eventID": "637707ce-e4a3-4af1-8edc-16e37e851b17"
}
```

以下示例显示 Alice 调用了 ListStacks 操作。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:03:16Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "ListStacks",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
```

```
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "b7d351d7-b397-11e3-bb75-a5b75389b02d",
  "eventID": "918206d0-7281-4629-b778-b91eb0d83ce5"
}
```

以下示例显示 Alice 对 my-test-stack 堆栈调用了 DescribeStacks 操作。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:06:15Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "DescribeStacks",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "stackName": "my-test-stack"
  },
  "responseElements": null,
  "requestID": "224f2586-b398-11e3-bb75-a5b75389b02d",
  "eventID": "9e5b2fc9-1ba8-409b-9c13-587c2ea940e2"
}
```

以下示例显示 Alice 对 my-test-stack 堆栈调用了 DeleteStack 操作。

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
```

```
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:07:15Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "DeleteStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "stackName": "my-test-stack"
  },
  "responseElements": null,
  "requestID": "42dae739-b398-11e3-bb75-a5b75389b02d",
  "eventID": "4965eb38-5705-4942-bb7f-20ebe79aa9aa"
}
```

Amazon CloudFormation 中的基础结构安全性

作为一项托管服务，Amazon CloudFormation 受 Amazon 全球网络安全保护。有关 Amazon 安全服务以及 Amazon 如何保护基础结构的信息，请参阅 Amazon 云安全。要按照基础结构安全最佳实践设计您的 Amazon 环境，请参阅《安全性支柱 Amazon Well-Architected Framework》中的 基础结构保护。

您可以使用 Amazon 发布的 API 调用通过网络访问 Amazon CloudFormation。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

您可以从任何网络位置调用这些 API 操作，但 Amazon CloudFormation 不支持基于资源的访问策略，其中可以包含基于源 IP 地址的限制。您还可以使用 Amazon CloudFormation 策略来控制来自特定 Amazon Virtual Private Cloud (Amazon VPC) 端点或特定 VPC 的访问。事实上，这隔离了在 Amazon CloudFormation 网络中仅从特定 VPC 到给定 Amazon 资源的网络访问。

Amazon CloudFormation 中的故障恢复能力

Amazon全球基础架构围绕Amazon区域和可用区构建。Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和

操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon全球基础设施。

的合规性验证 Amazon CloudFormation

要了解某个 Amazon Web Services 服务是否在特定合规性计划范围内，请参阅按合规性计划划分的范围内的 Amazon Web Services 服务，然后选择您感兴趣的合规性计划。有关常规信息，请参阅 、、Amazon Web Services 合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅在 Amazon Artifact 中下载报告。

您在使用 Amazon Web Services 服务 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。有关您在使用 Amazon Web Services 服务 时的合规责任的更多信息，请参阅 Amazon 安全性文档。

Amazon CloudFormation 中的配置和漏洞分析

配置和 IT 控制是 Amazon 和您（我们的客户）之间的共同责任。有关更多信息，请参阅 Amazon 责任共担模型。

CloudFormation 的安全最佳实践

Amazon CloudFormation 提供了在您开发和实施自己的安全策略时需要考虑的大量安全功能。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合环境或不满足环境要求，请将其视为有用的考虑因素而不是惯例。

主题

- 使用 IAM 控制访问
- 请勿将凭证嵌入您的模板
- 使用 Amazon CloudTrail 记录 CloudFormation 调用

使用 IAM 控制访问

IAM 是一项可用于管理 Amazon 中的用户及其权限的 Amazon 服务。您可以将 IAM 与 CloudFormation 结合使用来指定用户可以执行哪些 CloudFormation 操作，例如查看堆栈模板、创建

堆栈或删除堆栈。此外，管理 CloudFormation 堆栈的任何人都需要具备对这些堆栈中的资源执行操作的权限。例如，如果用户想使用 CloudFormation 启动、更新或终止 Amazon EC2 实例，则他们必须拥有调用相关 Amazon EC2 操作的权限。

在大多数情况下，用户需要完全访问权限来管理模板中的所有资源。CloudFormation 可代表用户发出调用，来创建、修改和删除这些资源。要分隔用户和 CloudFormation 服务之间的权限，请使用服务角色。CloudFormation 使用服务角色的策略发出调用，而不是通过用户的策略。有关更多信息，请参阅 Amazon CloudFormation 服务角色。

请勿将凭证嵌入您的模板

我们建议您在堆栈模板中使用动态引用，而不是在 CloudFormation 模板中嵌入敏感信息。

动态引用提供了一种简洁、功能强大的方法，用于引用在其他服务（例如 Amazon Systems Manager Parameter Store 或 Amazon Secrets Manager）中存储和管理的外部值。当您使用动态引用时，CloudFormation 会在堆栈和更改集合操作期间根据需要检索指定引用的值，并将值传递到相应的资源。但是，CloudFormation 从不存储实际引用值。有关更多信息，请参阅 使用动态引用获取存储在其他服务中的值。

Amazon Secrets Manager 帮助您安全地加密、存储和检索数据库和其他服务的凭证。Amazon Systems Manager Parameter Store 提供安全的分层存储，用于配置数据管理。

有关定义模板参数的信息，请参阅CloudFormation 模板 Parameters 语法。

使用 Amazon CloudTrail 记录 CloudFormation 调用

Amazon CloudTrail 跟踪在您的 Amazon Web Services 账户中进行 CloudFormation API 调用的任何人。当任何人使用 CloudFormation API、CloudFormation 控制台、后端控制台或 CloudFormation Amazon CLI 命令时，系统将会记录 API 调用。启用日志记录并指定用于存储日志的 Amazon S3 存储桶。这样一来，在需要时，您可以审核在您的账户中发起 CloudFormation 调用的人员。有关更多信息，请参阅 使用 Amazon CloudTrail 记录 Amazon CloudFormation API 调用。

使用接口端点 (Amazon PrivateLink) 访问 Amazon CloudFormation

您可以使用 Amazon PrivateLink 在您的 VPC 和 Amazon CloudFormation 之间创建私有连接。您可以像在 VPC 中一样访问 Amazon CloudFormation，而无需使用互联网网关、NAT 设备、VPN 连接或 Amazon Direct Connect 连接。VPC 中的实例不需要公有 IP 地址即可访问 Amazon CloudFormation。

您可以通过创建由 Amazon PrivateLink 提供支持的接口端点来建立此私有连接。我们将在您为接口端点启用的每个子网中创建一个端点网络接口。这些是请求者托管的网络接口，用作发往 Amazon CloudFormation 的流量的入口点。

Amazon CloudFormation 支持通过接口端点调用其所有 API 操作。

Amazon CloudFormation VPC 端点注意事项

请首先确认您已满足《Amazon PrivateLink 指南》中 Access an Amazon service using an interface VPC endpoint 主题描述的先决条件，然后再设置接口端点。

在为 Amazon CloudFormation 设置接口端点时，还需要满足以下其他先决条件和注意事项：

- 如果 VPC 中有资源必须响应自定义资源请求或等待条件，请确保这些资源拥有所需 CloudFormation 特定 Amazon S3 存储桶的访问权限。CloudFormation 在每个区域都有 S3 存储桶，以监视对自定义资源请求或等待条件的响应。如果模板在 VPC 中包含有自定义资源或等待条件，则 VPC 端点策略必须允许用户向以下存储桶发送响应：
  - 对于自定义资源，允许流量前往 `cloudformation-custom-resource-response-region` 存储桶。使用自定义资源时，Amazon Web Services 区域名称不包含连接号。例如 `uswest2`。
  - 对于等待条件，允许流量前往 `cloudformation-waitcondition-region` 存储桶。使用等待条件时，Amazon Web Services 区域名称应包含连接号。例如 `us-west-2`。

  如果端点策略阻止前往这些存储桶的流量，则 CloudFormation 不会收到响应，堆栈操作将失败。例如，如果某个资源位于 `us-west-2` 区域的 VPC 中且必须响应等待条件，则此资源必须能够向 `cloudformation-waitcondition-us-west-2` 存储桶发送响应。

  如需了解当前提供 CloudFormation 服务的 Amazon Web Services 区域列表，请参阅《Amazon Web Services 一般参考》中的 Amazon CloudFormation endpoints and quotas页面。
- VPC 端点当前不支持跨区域请求 - 确保在计划向 CloudFormation 发出 API 调用的同一区域中创建端点。
- VPC 端点仅支持 Amazon 通过 Route 53 提供的 DNS。如果您希望使用自己的 DNS，可以使用条件 DNS 转发。有关更多信息，请参阅《Amazon VPC 用户指南》中的 Amazon VPC 中的 DHCP 选项集。
- 附加到 VPC 端点的安全组必须允许端口 443 上来自 VPC 的私有子网的传入连接。

为 Amazon CloudFormation 创建接口 VPC 端点

您可以使用 Amazon VPC 控制台或 Amazon Command Line Interface（Amazon CLI）为 Amazon CloudFormation 创建 VPC 端点。有关更多信息，请参阅 Amazon PrivateLink 指南中的创建 VPC 端点。

使用以下服务名称为 Amazon CloudFormation创建接口端点：

- com.amazonaws.***region***.cloudformation – 为 CloudFormation API 操作创建端点。

如果为接口端点启用私有 DNS，则可使用其默认区域 DNS 名称向 Amazon CloudFormation 发出 API 请求。例如 `cloudformation.us-east-1.amazonaws.com`。

在支持 FIPS 特定端点的 Amazon Web Services 区域中，您还可以使用以下服务名称为 Amazon CloudFormation 创建接口端点：

- com.amazonaws.***region***.cloudformation-fips – 为符合联邦信息处理标准 (FIPS) 140-2 的 CloudFormation API 创建端点。

有关 CloudFormation 端点的完整列表，请参阅《Amazon Web Services 一般参考》中的 Amazon CloudFormation endpoints and quotas。

为 Amazon CloudFormation创建 VPC 端点策略

端点策略是一种 IAM 资源，您可以将其附加到接口端点。默认端点策略允许通过接口端点完全访问 Amazon CloudFormation API。要控制允许从 VPC 访问 Amazon CloudFormation的权限，请将自定义端点策略附加到接口端点。

端点策略指定以下信息：

- 可执行操作的主体（Amazon Web Services 账户、IAM 用户和 IAM 角色）。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅 Amazon PrivateLink 指南中的使用端点策略控制对 VPC 端点的访问权限。

示例：Amazon CloudFormation操作的 VPC 端点策略

下面是用于 Amazon CloudFormation 的端点策略示例。当附加到端点时，此策略会向所有资源上的所有主体授予对列出的 Amazon CloudFormation 操作的访问权限。以下示例拒绝所有用户通过 VPC 端点创建堆栈的权限，并允许对 Amazon CloudFormation 服务上的所有其他操作进行完全访问。

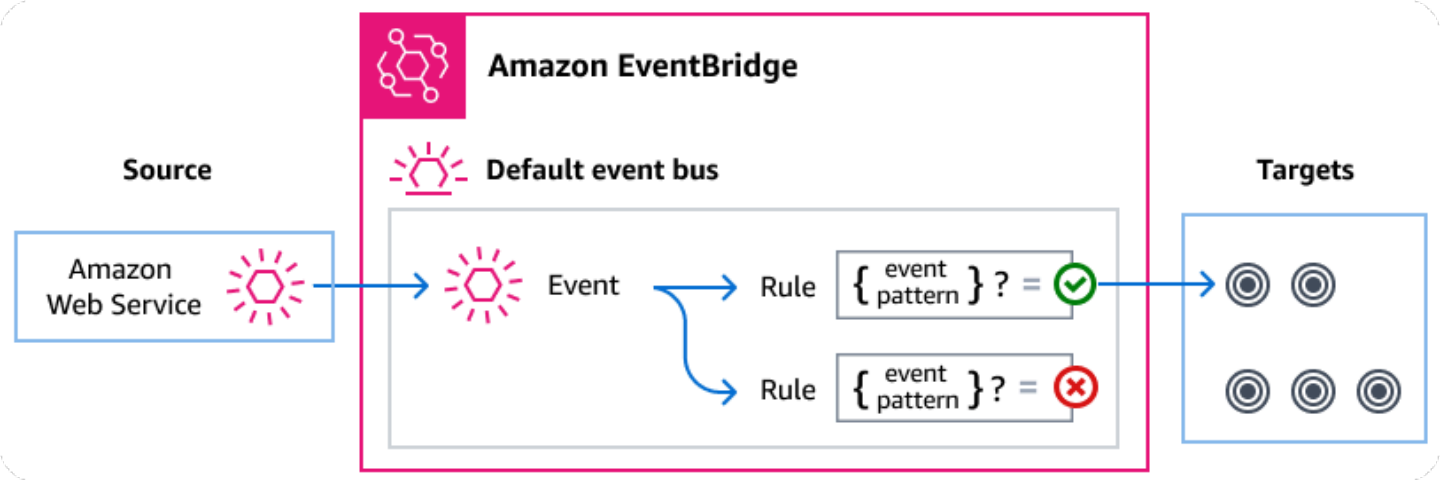
```
{
  "Statement": [
    {
      "Action": "cloudformation:*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    },
    {
      "Action": "cloudformation:CreateStack",
      "Effect": "Deny",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

使用 EventBridge 监控 CloudFormation 和 Git 同步事件

Amazon EventBridge 是一项无服务器服务，使用事件将应用程序组件连接在一起，可让您更轻松地构建可扩展的事件驱动型应用程序。事件驱动型架构是一种构建松耦合软件系统的风格，这些系统通过发出和响应事件来协同工作。事件代表资源或环境中的变化。

与许多 Amazon 服务一样，CloudFormation 会生成事件并将其发送到 EventBridge 的默认事件总线，该总线会在每个 Amazon Web Services 账户中自动配置。事件总线是接收事件并将其传送到零个或多个目的地或目标的路由器。为事件总线指定的规则会在事件到达时进行评估。每条规则都会检查事件是否与规则的事件模式相匹配。如果事件确实匹配，事件总线会将事件发送到指定的目标。

有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Amazon EventBridge 入门。



主题

- CloudFormation 和 Git 同步事件概述
- Amazon EventBridge 权限
- 为 EventBridge 规则创建自定义事件模式
- CloudFormation 事件详细信息参考

CloudFormation 和 Git 同步事件概述

每当对堆栈执行创建、更新、删除或偏差检测操作时，CloudFormation 都可以向 EventBridge 发送事件。CloudFormation 还会向 EventBridge 发送事件，以更改堆栈集和堆栈集实例的状态。您可以使用 EventBridge 规则将事件路由到定义的目标。这些事件会保证传送，但可能会不按顺序传送。

由于 CloudFormation 事件代表对堆栈或堆栈集及其资源的更改，因此您可以使用它们来启动与相应事件关联的工作流。例如：

- 在通过 CloudFormation 预置的所有资源上创建堆栈或堆栈集特定标签。
- 在 CloudFormation 堆栈或堆栈集与 Amazon WorkSpaces Application Manager（Amazon WAM）之间建立关联。
- 为创建的堆栈或堆栈集指定与 AppRegistry 的关联。

以下事件由 CloudFormation 生成并发送到 EventBridge 中的默认事件总线。有关更多信息，请参阅 ???。

Event type	说明	事件源
资源状态更改	在堆栈上执行的任何更新都会更改底层资源属性。 有关支持的 Amazon 资源类型的完整列表，请参阅 Amazon 资源和属性类型参考。	Amazon CloudFormation
堆栈状态更改	表示对给定堆栈的状态更改。 有关代码详细信息，请参阅 堆栈状态代码。	Amazon CloudFormation
偏差检测状态更改	表示用户在给定堆栈上启动的偏差检测更新。 有关支持偏差检测的完全可变和不可变类型的完整列表，请参阅 资源类型支持	Amazon CloudFormation
StackSet 状态更改	表示对给定堆栈集的状态更改。	Amazon CloudFormation
StackSet 堆栈实例状态更改	表示特定 StackSet 堆栈实例的状态更改。 有关代码详细信息，请参阅 堆栈实例状态代码。	Amazon CloudFormation
StackSet 操作状态	表示对给定 StackSet 操作的状态更改。	Amazon CloudFormation

Event type	说明	事件源
	有关代码详细信息，请参阅 StackSets 状态代码。	

此外，Amazon CloudFormation Git 同步还会将存储库同步和资源同步的状态更改事件发送到 EventBridge。

以下 Git 同步事件由 Amazon CodeConnections 生成并发送到 EventBridge 中的默认事件总线。有关更多信息，请参阅 ???。

Event type	说明	事件源
存储库同步状态更改	表示 Git 存储库同步的状态更改。	Amazon CodeConnections
资源同步状态更改	表示 Git 资源同步的状态更改。	Amazon CodeConnections

Amazon EventBridge 权限

CloudFormation 不需要任何其他权限即可向 EventBridge 传输事件。这些事件包含已通过 CloudFormation 的 API 操作获得的信息。

您指定的目标可能需要特定的权限或配置。有关为目标使用特定服务的详细信息，请参阅《Amazon EventBridge User Guide》中的 Amazon EventBridge targets。

为 EventBridge 规则创建自定义事件模式

您可以在 EventBridge 中找到多个用于 CloudFormation 和 Git 同步事件的预定义模式。这简化了事件模式的创建方式。您可以选择表单上的字段值，然后 EventBridge 为您生成模式，而不是编写自己的事件模式。您可以使用这些预定义的事件模式之一创建新规则，也可以创建自己的自定义事件模式。

CloudFormation 等服务将事件传送到默认事件总线时，EventBridge 使用在规则中定义的事件模式来确定是否应将事件传送到规则的目标。事件模式与所需 CloudFormation 事件中的数据相匹配。

每个事件模式是一个 JSON 对象，其中包含：

- 标识发送事件的服务的 source 属性。例如，aws.cloudformation 或 aws.codeconnections。
- （可选）：包含要匹配的事件类型数组的 detail-type 属性。
- （可选）：包含要匹配的其他事件数据的 detail 属性。

  例如，堆栈 ID、涉及的资源、各种资源的状态以及与特定类型事件相关的其他数据。

例如，以下事件模式与所有资源状态更改事件相匹配：

```
{
  "source": ["aws.cloudformation"],
  "detail-type": ["CloudFormation Resource Status Change"]
}
```

以下事件模式使用事件详细信息数据，仅匹配 CloudFormation 创建新的 AWS::S3::Bucket 或 AWS::SNS::Topic 资源的资源状态更改事件：

```
{
  "source": ["aws.cloudformation"],
  "detail-type": ["CloudFormation Resource Status Change"],
  "detail": {
    "status-details": {
      "status": ["CREATE_COMPLETE"]
    },
    "resource-type": ["AWS::S3::Bucket", "AWS::SNS::Topic"]
  }
}
```

有关写入事件模式的更多信息，请参阅《Amazon EventBridge 用户指南》中的事件模式。

CloudFormation 事件详细信息参考

来自 Amazon 服务的所有事件都有一组公共字段，其中包含有关事件的元数据，例如作为事件源的 Amazon 服务、事件的生成时间、事件发生的账户和区域等。有关这些常规字段的定义，请参阅《Amazon Events Reference》中的 Amazon service event metadata。

此外，每个事件都有一个 detail 字段，其中包该特定事件专有的数据。下面的参考定义了各种 CloudFormation 事件的详细信息字段。

在使用 EventBridge 选择和管理 CloudFormation 事件时，请记住以下几点：

- `source` 字段指定事件源。

  例如，`aws.cloudformation` 或 `aws.codeconnections`。
- `detail-type` 字段指定事件类型。

  例如，`CloudFormation Resource Status Change` 或 `CloudFormation Drift Detection Status Change`。
- `detail` 字段包含该特定事件专有的数据。

  例如，堆栈 ID、涉及的资源、各种资源的状态以及与特定类型事件相关的其他数据。

有关如何构造使规则能够匹配 CloudFormation 事件的事件模式的信息，请参阅《Amazon EventBridge 用户指南》中的事件模式。

有关事件以及 EventBridge 如何处理事件的更多信息，请参阅《Amazon EventBridge 用户指南》中的 EventBridge 事件。

主题

- 资源状态更改事件详细信息
- 堆栈状态更改事件详细信息
- 偏差检测状态更改事件详细信息
- StackSet 状态更改事件详细信息
- StackSet 堆栈实例状态更改事件详细信息
- StackSet 操作状态更改事件详细信息
- 存储库同步状态更改事件详细信息
- 资源同步状态更改事件详细信息

资源状态更改事件详细信息

以下是资源状态更改事件的详细信息字段。

之所以包含 `source` 和 `detail-type` 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type": "CloudFormation Resource Status Change",
  "source": "aws.cloudformation",
```

```
  . . .,
  "detail": {
    "stack-id" : "string",
    "logical-resource-id" : "string",
    "physical-resource-id": "string",
    "status-details": {
        "status": "string",
        "status-reason": "string"
    },
     "resource-type": "string",
     "client-request-token": "string"
  }
}
```

`detail-type`

标识事件的类型。

对于资源状态事件，此值为 `CloudFormation Resource Status Change`。

`source`

标识生成事件的服务。对于 CloudFormation 事件，此值为 `aws.cloudformation`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于资源状态事件，这些数据包括：

`stack-id`

与堆栈关联的唯一堆栈 ID。

`logical-resource-id`

资源的逻辑名称，如模板中所规定。

`physical-resource-id`

与 CloudFormation 所支持资源的物理实例 ID 对应的名称或唯一标识符。

`status-details`

`status`

资源的状态。

`status-reason`

资源的状态原因。

`resource-type`

资源的类型。例如 `AWS::S3::Bucket`。

`client-request-token`

用于调用 API 的访问令牌。由给定堆栈操作启动的所有事件分配的客户端请求令牌都是相同的，可使用此令牌跟踪操作。从控制台发起的堆栈操作使用令牌格式 Console-StackOperation-ID，这有助于您轻松识别堆栈操作。例如，如果使用控制台创建堆栈，则将向每个生成的堆栈事件分配以下格式的相同令牌：`Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002`。

Example示例：资源状态更改事件

以下是资源状态事件的示例。此事件详细说明了 CloudFormation 已在指定堆栈中成功创建所请求的资源，即 Amazon S3 存储桶。

```
{
    "version":"0",
    "id":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type":"CloudFormation Resource Status Change",
    "source":"aws.cloudformation",
    "account":"111122223333",
    "time":"2017-12-22T18:43:48Z",
    "region":"us-west-1",
    "resources":[
        "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack"
    ],
    "detail":{
        "stack-id":"arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
        "logical-resource-id":"my-s3-bucket",
        "physical-resource-id":"arn:aws:s3:::my-s3-bucket-us-east-1",
        "status-details":{
            "status":"CREATE_COMPLETE",
            "status-reason":""
        },
        "resource-type":"AWS::S3::Bucket",
        "client-request-token":""
    }
```

```
}
```

堆栈状态更改事件详细信息

以下是堆栈状态更改事件的详细信息字段。

之所以包含 source 和 detail-type 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type":"CloudFormation Stack Status Change",
  "source":"aws.cloudformation",
  . . .,
  "detail":{
    "stack-id":"string",
    "status-details":{
      "status":"string",
      "status-reason":"string"
    },
    "client-request-token":"string"
  }
}
```

`detail-type`

标识事件的类型。

对于堆栈状态事件，此值为 `CloudFormation Stack Status Change`。

`source`

标识生成事件的服务。对于 CloudFormation 事件，此值为 `aws.cloudformation`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于堆栈状态事件，这些数据包括：

`stack-id`

与堆栈关联的唯一堆栈 ID。

`status-details`

`status`

堆栈的状态。

有关堆栈状态代码的完整列表，请参阅 堆栈状态代码。

`status-reason`

资源的状态原因。

`client-request-token`

用于调用 API 的访问令牌。由给定堆栈操作启动的所有事件分配的客户端请求令牌都是相同的，可使用此令牌跟踪操作。从控制台发起的堆栈操作使用令牌格式 Console-StackOperation-ID，这有助于您轻松识别堆栈操作。例如，如果使用控制台创建堆栈，则将向每个生成的堆栈事件分配以下格式的相同令牌：`Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002`。

Example示例：堆栈状态事件

以下是示例堆栈状态事件，其中 CloudFormation 已成功创建所请求的堆栈 `teststack`。

```
{
    "version":"0",
    "id":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type":"CloudFormation Stack Status Change",
    "source":"aws.cloudformation",
    "account":"111122223333",
    "time":"2017-12-22T18:43:48Z",
    "region":"us-west-1",
    "resources":[
        "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack"
    ],
    "detail":{
        "stack-id":"arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
        "status-details":{
            "status":"CREATE_COMPLETE",
            "status-reason":""
        },
        "client-request-token":""
    }
}
```

偏差检测状态更改事件详细信息

以下是堆栈偏差检测事件的详细信息字段。

之所以包含 source 和 detail-type 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type":"CloudFormation Drift Detection Status Change",
  "source":"aws.cloudformation",
  . . .,
  "detail":{
    "stack-id":"string",
    "stack-drift-detection-id":"string",
    "status-details":{
      "stack-drift-status":"string",
      "detection-status":"string"
    },
      "drift-detection-details":{
        "drifted-stack-resource-count":integer
      },
    "client-request-token":"string"
  }
}
```

`detail-type`

标识事件的类型。

对于堆栈偏差检测事件，此值为 `CloudFormation Drift Detection Status Change`。

`source`

标识生成事件的服务。对于 CloudFormation 事件，此值为 `aws.cloudformation`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于堆栈偏差检测事件，这些数据包括：

`stack-id`

与堆栈关联的唯一堆栈 ID。

stack-drift-detection-id

堆栈偏差检测 ID。

status-details

stack-drift-status

堆栈的偏差状态。

detection-status

偏差检测操作的状态。

drift-detection-details

drifted-stack-resource-count

资源数量偏差。值为 -1 时，偏差检测正在进行。所有其他非负整数均表示偏差资源的实际数量。

client-request-token

用于调用 API 的访问令牌。由给定堆栈操作启动的所有事件分配的客户端请求令牌都是相同的，可使用此令牌跟踪操作。从控制台发起的堆栈操作使用令牌格式 Console-StackOperation-ID，这有助于您轻松识别堆栈操作。例如，如果使用控制台创建堆栈，则将向每个生成的堆栈事件分配以下格式的相同令牌：Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002。

Example示例：堆栈偏差检测事件

下面展示的是堆栈偏差检测事件的示例。此事件详细说明了 CloudFormation 已完成对指定堆栈的偏差检测，且因一个资源出现偏差，堆栈当前偏差状态为 DRIFTED。

```
{
    "version":"0",
    "id":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type":"CloudFormation Drift Detection Status Change",
    "source":"aws.cloudformation",
    "account":"111122223333",
    "time":"2017-12-22T18:43:48Z",
    "region":"us-west-1",
    "resources": ["string"],
    "detail":{
        "stack-id":"arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
```

```
        "stack-drift-detection-id":"624af370-311a-11e8-b6b7-500cexample",
        "status-details":{
            "stack-drift-status":"DRIFTED",
            "detection-status":"DETECTION_COMPLETE"
        },
        "drift-detection-details":{
            "drifted-stack-resource-count":1
        },
    "client-request-token":""
    }
}
```

StackSet 状态更改事件详细信息

以下是 StackSet 状态更改事件的详细信息字段。

之所以包含 `source` 和 `detail-type` 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type": "CloudFormation StackSet Status Change",
  "source": "aws.cloudformation",
  . . .,
  "detail": {
    "stack-set-arn" : "string",
    "status-details": {
        "status":"string"
    }
  }
}
```

`detail-type`

标识事件的类型。

对于 StackSet 状态事件，此值为 `CloudFormation StackSet Status Change`。

`source`

标识生成事件的服务。对于 CloudFormation 事件，此值为 `aws.cloudformation`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于 StackSet 状态事件，这些数据包括：

`stack-set-arn`

与堆栈集关联的 Amazon 资源名称（ARN）。

`status-details`

`status`

StackSet 状态。

有效值：`ACTIVE | DELETED`

Example示例：StackSet 状态更改事件

以下是 StackSet 状态更改事件的示例。此事件详细说明了 CloudFormation 已删除指定的堆栈集。

```
{
  "version": "0",
  "id": "42h6hb90-hg0w-11op-b01v-0xhnh0934z09",
  "detail-type": "CloudFormation StackSet Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-23T17:06:18Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test12345:3f3a3fbe-
c937-4eb3-a87d-e36a0af3f663"
  ],
  "detail": {
    "stack-set-arn" : "arn:aws:cloudformation:us-east-1:111122223333:stackset/
test12345:3f3a3fbe-c937-4eb3-a87d-e36a0af3f663",
    "status-details": {
        "status":"DELETED"
    }
  }
}
```

StackSet 堆栈实例状态更改事件详细信息

以下是 StackSet 堆栈实例状态事件的详细信息字段。

之所以包含 `source` 和 `detail-type` 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type": "CloudFormation StackSet StackInstance Status Change",
  "source": "aws.cloudformation",
  . . .,
  "detail": {
    "stack-set-arn" : "string",
    "stack-id" : "string",
    "action" : "string",
    "status-details": {
        "status": "string",
        "status-reason": "string",
        "detailed-status": "string"
      }
    }
  }
}
```

`detail-type`

标识事件的类型。

对于 StackSet 堆栈实例状态事件，此值为 `CloudFormation StackSet StackInstance Status Change`。

`source`

标识生成事件的服务。对于 CloudFormation 事件，此值为 `aws.cloudformation`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于 StackSet 堆栈实例状态事件，这些数据包括：

`stack-set-arn`

与 StackSet 关联的 Amazon 资源名称（ARN）。

`stack-id`

与堆栈实例关联的唯一堆栈 ID。

`action`

堆栈集操作的类型。

有效值：CREATE | UPDATE | DELETE | DETECT_DRIFT

status-details

status

StackSet 实例的状态。

有关更多详细信息，请参阅堆栈实例状态代码。

有效值：CURRENT | OUTDATED| INOPERABLE

status-reason

StackSet 实例的状态原因。

detailed-status

详细 StackSet 实例的详细状态。

有效值：CANCELLED | FAILED | FAILED_IMPORT | INOPERABLE | PENDING | RUNNING | SKIPPED_SUSPENDED_ACCOUNT | SUCCEEDED

Example示例：StackSet 堆栈实例状态更改事件

以下是 StackSet 堆栈实例状态更改事件的示例。

```
{
  "version": "0",
  "id": "42h6hb90-hg0w-11op-b01v-0xhnh0934z09",
  "detail-type": "CloudFormation StackSet StackInstance Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-22T19:19:23Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-
d041-44ad-94f8-b8268aca1e59"
  ],
  "detail": {
     "stack-set-arn": "arn:aws:cloudformation:us-east-1:111122223333:stackset/
test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59",
    "stack-id": "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
    "status-details": {
        "status": "OUTDATED",
```

```
        "status-reason": "User Initiated",
        "detailed-status": "PENDING"
    }
  }
}
```

StackSet 操作状态更改事件详细信息

以下是 StackSet 操作状态更改事件的详细信息字段。

之所以包含 `source` 和 `detail-type` 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type": "CloudFormation StackSet Operation Status Change",
  "source": "aws.cloudformation",
  . . .,
  "detail": {
    "stack-set-arn" : "string",
    "stack-set-operation-id" : "string",
    "status-details": {
        "status": "string"
      }
    }
  }
}
```

`detail-type`

标识事件的类型。

对于 StackSet 操作状态事件，此值为 `CloudFormation StackSet Operation Status Change`。

`source`

标识生成事件的服务。对于 CloudFormation 事件，此值为 `aws.cloudformation`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于 StackSet 操作状态事件，这些数据包括：

`stack-set-arn`

与 StackSet 关联的 Amazon 资源名称（ARN）。

`stack-set-operation-id`

与 StackSet 操作关联的唯一 ID。

`status-details`

`status`

StackSet 操作状态。

有关更多详细信息，请参阅StackSets 状态代码。

有效值：`RUNNING` | `SUCCEEDED` | `FAILED` | `STOPPING` | `STOPPED` | `QUEUED`

Example示例：StackSet 操作状态更改事件

以下是 StackSet 操作状态更改事件的示例。事件详细说明了 CloudFormation 已成功完成对指定堆栈集所请求的操作。

```
{
  "version": "0",
  "id": "4de89905-fd92-6a6b-9509-23c04bcb6a21",
  "detail-type": "CloudFormation StackSet Operation Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-22T05:46:24Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-
d041-44ad-94f8-b8268aca1e59"
  ],
  "detail": {
    "stack-set-arn": "arn:aws:cloudformation:us-east-1:111122223333:stackset/
test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59",
    "stack-set-operation-id": "ce69adce-2221-4483-8c4b-c51f284f25e8",
    "status-details": {
        "status": "SUCCEEDED"
    }
  }
}
```

存储库同步状态更改事件详细信息

以下是存储库同步状态更改事件的详细信息字段。

之所以包含 source 和 detail-type 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type": "Git Sync Repository Sync Status Change",
  "source": "aws.codeconnections",
  . . .,
  "detail": {
    "connectionArn" : "string",
    "providerType" : "string",
    "repositoryName": "string",
    "providerType": "string",
    "repositoryName": "string",
    "repositoryArn": "string",
    "repositoryLinkId": "string",
    "ownerId": "string",
    "commit": "string",
    "branch": "string",
    "syncType": "string",
    "status": "string",
    "previousSync": "string"
    }
  }
}
```

`detail-type`

标识事件的类型。

对于存储库同步状态事件，此值为 `Git Sync Repository Sync Status Change`。

`source`

标识生成事件的服务。对于 Git 同步事件，此值为 `aws.codeconnections`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于存储库同步状态事件，此数据包括：

`connectionArn`

与 Amazon CodeConnections 关联的 Amazon 资源名称（ARN）。

`providerType`

连接到 CloudFormation 的 Git 提供程序。

有效值：`GitHub` |`GitHub Enterprise` |`GitLab` |`BitBucket`

`repositoryName`

Git 存储库名称。

`repositoryArn`

与 Git 存储库关联的 ARN。

`repositoryLinkId`

与存储库链接关联的唯一 ID。

`ownerId`

与存储库所有者关联的唯一 ID。

`commit`

与存储库提交关联的唯一 ID。

`branch`

与存储库分支关联的唯一 ID。

`syncType`

所执行的同步的类型。

`status`

当前存储库同步状态。

有效值：`FAILED` |`INITIATED` |`IN_PROGRESS` |`SUCCEEDED`

`previousSync`

当前状态之前的同步状态。

有效值：`FAILED` |`INITIATED` |`IN_PROGRESS` |`SUCCEEDED`

Example示例：存储库同步状态更改事件

以下是存储库同步状态更改事件的示例。该事件详细说明了 Amazon CodeConnections 已成功同步存储库。

```
{
  "version": "0",
  "id": "1b5d8feb-agbv-4cf7-a9f1-bf3703467718",
  "detail-type": "GitSync Repository Sync Status Change",
  "source": "aws.codeconnections",
  "account": "111122223333",
  "time": "2023-12-22T18:43:48Z",
  "region": "us-east-1",
  "resources": ["arn:aws:aws:codestar-connections:us-east-1:111122223333:repository-
link/550e8400-e29b-41d4-a716-446655440000",],
  "detail": {
    "connectionArn": "arn:aws:codestar-connections:us-east-1:111122223333:connection/
sample-connection-id",
    "providerType": "GitHub",
    "repositoryName": "sample-repository-name",
    "repositoryArn": "arn:aws:aws:codestar-connections:us-
east-1:111122223333:repository-link/550e8400-e29b-41d4-a716-446655440000"
    "repositoryLinkId": "550e8400-e29b-41d4-a716-446655440000"
    "ownerId": "sample-owner-id",
    "commit": "sample-commit-id",
    "branch": "main",
    "syncType": "CFN_STACK_SYNC",
    "status": "SUCCEEDED",
    "previousStatus": "IN_PROGRESS",
  }
}
```

资源同步状态更改事件详细信息

以下是资源同步状态更改事件的详细信息字段。

之所以包含 `source` 和 `detail-type` 字段，是因为其包含事件的特定值。

```
{
  . . .,
  "detail-type": "Git Sync Resource Sync Status Change",
  "source": "aws.codeconnections",
```

```
  . . .,
  "detail": {
    "providerType" : "string",
    "commit" : "string",
    "repositoryName": "string",
    "branch": "string",
    "syncType": "string",
    "syncTarget": "string",
    "status": "string",
    "previousSync": "string"
  }
}
```

`detail-type`

标识事件的类型。

对于存储库同步状态事件，此值为 `Git Sync Repository Sync Status Change`。

`source`

标识生成事件的服务。对于 Git 同步事件，此值为 `aws.codeconnections`。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

对于资源同步状态事件，这些数据包括：

`providerType`

连接到 CloudFormation 的 Git 提供程序。

有效值：`GitHub` |`GitHub Enterprise` |`GitLab` |`BitBucket`

`commit`

与存储库提交关联的唯一 ID。

`repositoryName`

Git 存储库名称。

`branch`

与存储库分支关联的唯一 ID。

syncType

所执行的同步的类型。

syncTarget

资源同步的目标堆栈。

status

当前存储库同步状态。

有效值：FAILED |INITIATED |IN_PROGRESS |SUCCEEDED

previousSync

当前状态之前的同步状态。

有效值：FAILED |INITIATED |IN_PROGRESS |SUCCEEDED

Example示例：资源同步状态更改事件

以下是资源同步状态更改事件的示例。该事件详细说明了 Amazon CodeConnections 已成功同步资源。

```
{
  "version": "0",
  "id": "1b5d8feb-agbv-4cf7-a9f1-bf3703467718",
  "detail-type": "Git Sync Resource Sync Status Change",
  "source": "aws.codeconnections",
  "account": "111122223333",
  "time": "2023-12-22T18:43:48Z",
  "region": "us-east-1",
  "resources": ["arn:aws:aws:cloudformation:us-east-1:111122223333:stack/
targetStack1"],
  "detail": {
    "providerType": "GitHub",
    "commit": "sample-commit-id",
    "repositoryName": "sample-repository-name",
    "branch": "main",
    "syncType": "CFN_STACK_SYNC",
    "syncTarget": "arn:aws:aws:cloudformation:us-east-1:111122223333:stack/
targetStack1",
    "status": "SUCCEEDED",
```

```
    "previousStatus": "IN_PROGRESS"
  }
}
```

了解 CloudFormation 配额

您的 Amazon Web Services 账户具有 CloudFormation 配额，您在创作模板和创建堆栈时可能需要了解这些配额。通过了解这些配额，可以避免需要重新设计模板或堆栈的限制错误。

下表展示了 CloudFormation 的配额。

限额	描述	值	调优策略
cfn-signal 等待条件数据	`cfn-signal` 可以传递的最大数据量。	4,096 个字节	要传递更大的数据量，请将数据发送到某个 Amazon S3 存储桶，然后使用 `cfn-signal` 将 Amazon S3 URL 传递给该存储桶。
自定义资源响应	自定义资源提供程序可以传递的最大数据量。	4,096 个字节	
每模板动态引用数	单个 CloudFormation 堆栈模板中允许的最大动态引用数量。	一个堆栈模板 60 个动态引用	
每个账户的挂钩数	每区域每账户允许的最大钩子数量。	100 个挂钩	
每个资源的挂钩数	可为每个资源配置的最大钩子数量。	100 个挂钩	
钩子配置大小	一个钩子可以存储的最大配置数据量。	204.8KB	
映像	在 CloudFormation 模板中可以声明的最大映射数量。	200 个映像	要指定更多映射，请将您的模板分离为多个模板，例如，通过

限额	描述	值	调优策略
			使用嵌套堆栈来执行此操作。
映射属性	在 CloudFormation 模板中可以为每个映射声明的最大映射属性数量。	200 个属性	要指定更多映射属性，请将属性分离为多个映射。
映射名称和映射属性名称	每个映射名称的最大大小。	255 个字符	
模块	在 CloudFormation 注册表中可为每个区域和每个账户注册的最大模块数。	100 个模块	
模块版本	在 CloudFormation 注册表中可以为给定模块注册的最大版本数。	100 个版本	要注册新版本，请首先使用 DeregisterType 取消注册不再使用的版本。
嵌套堆栈	一个嵌套堆栈每次操作可以创建、更新或删除的最大 CloudFormation 资源数。 例如，您可以拥有一个总资源超过 2500 个的嵌套堆栈层次结构，但单次部署中创建、更新或删除的资源不得超过 2500 个。	2500 个资源	将堆栈层次结构拆分为不同的堆栈。
输出	在 CloudFormation 模板中可以声明的最大输出数量。	200 个输出	

限额	描述	值	调优策略
输出名称	输出名称的最大大小。	255 个字符	
参数	在 CloudFormation 模板中可以声明的最大参数数量。	200 个参数	要指定更多参数，可以使用映射或列表以将多个值分配给一个参数。
参数名称	参数名称的最大大小。	255 个字符	
参数值	参数值的最大大小。	4,096 个字节	要使用更大的参数化值，请创建多个参数，然后使用 `Fn::Join` 函数将多个值连接为单个值。
私有资源	在 CloudFormation 注册表中可以为每个账户和每个区域注册的最大私有资源数量。	50 个私有资源	
私有资源版本	在 CloudFormation 注册表中可以为给定私有资源注册的最大版本数。	50 个私有资源	要注册新版本，请首先使用 DeregisterType 取消注册不再使用的版本。
资源	在 CloudFormation 模板中可以声明的最大资源数量。	500 个资源	要指定更多资源，请将您的模板分离为多个模板，例如，通过使用嵌套堆栈来执行此操作。

限额	描述	值	调优策略
并发堆栈操作中的资源	在您的区域中，给定时间的堆栈操作（创建、更新或删除操作）可以涉及的最大资源数量。	使用 DescribeAccountLimits API 确定特定区域中账户的当前限制。	
资源名称	资源名称的最大大小。	255 个字符	
堆栈	您可以创建的最大 CloudFormation 堆栈数量。	2000 个堆栈	要创建更多堆栈，请删除不需要的堆栈，或请求增加您 Amazon Web Services 账户中的最大堆栈数量。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 服务限额。
堆栈名称	堆栈名称的最大大小。	128 个字符	
StackSets	可以在管理员账户中创建的最大 CloudFormation 堆栈集数量。	1000 个堆栈集	要创建更多堆栈集，请删除不需要的堆栈集，或请求增加您 Amazon Web Services 账户中的最大堆栈集数量。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 服务限额。

限额	描述	值	调优策略
堆栈实例	对于每个堆栈集可以创建的最大堆栈实例数量。	每个堆栈集 100000 个堆栈实例	要创建更多堆栈实例，请删除不需要的堆栈实例，或请求增加您 Amazon Web Services 账户中的最大堆栈实例数量。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 服务限额。
堆栈集实例操作	对于每个区域，每个管理员账户可同时跨所有堆栈集执行操作的最大堆栈实例数。	10000 次操作	此限制适用于区域中包含的所有堆栈集。它包括了堆栈集创建和更新操作以及直接创建、更新或删除堆栈实例所影响的堆栈实例。
StackSets 排队操作	一个堆栈集在给定时间的最大排队操作数。	10000 次操作	
每堆栈集操作使用 S3 对象导入的堆栈数	每个堆栈集操作可以使用 S3 对象导入的最大堆栈数。	200 个堆栈	
每堆栈集操作使用内联堆栈 ID 导入的堆栈数	每个堆栈集操作可以使用内联堆栈 ID 导入的最大堆栈数。	10 个堆栈	

限额	描述	值	调优策略
请求中的模板正文大小	可以在 CreateStack 、UpdateStack 或 ValidateTemplate 请求中传递的最大模板正文大小。	51200 字节	要使用更大的模板正文，请将模板分离为多个模板，例如，使用嵌套堆栈执行此操作。或将模板上传到 Amazon S3 存储桶。
Amazon S3 对象中的模板正文大小	对于具有 Amazon S3 模板 URL 的 CreateStack 、UpdateStack 、ValidateTemplate 请求，可以在 Amazon S3 对象中传递的最大模板正文大小。	1 MB	要使用更大的模板正文，请将模板分离为多个模板，例如，使用嵌套堆栈执行此操作。或者使用缩小功能来减小 CloudFormation 模板大小。
模板描述	模板描述的最大大小。	1024 字节	
每个挂钩的版本数	对于每个钩子可以创建的最大版本数。	100 个版本	

功能可用性

CloudFormation 的所有功能并非已在每个区域推出。有关 Amazon Web Services 区域的更多信息，请参阅全球基础设施区域表。

- 宏目前在以下区域中不可用：
  - 亚太地区（雅加达）
- 以下区域目前不支持模块：
  - 中国（北京）
  - 中国（宁夏）
- 目前，下列区域不支持使用 CloudFormation 通过 CodeDeploy 执行 ECS 蓝/绿部署：

- 非洲（开普敦）
- 亚太地区（大阪）
- 欧洲地区（米兰）

StackSet 和宏

StackSets 现在不支持通过包含宏的模板使用服务托管权限创建或更新堆栈集。其中包括转换，转换是由 CloudFormation 托管的宏。有关宏的更多信息，请参阅模板宏。

CloudFormation 疑难解答

使用 CloudFormation 时，您可能会在创建、更新或删除 CloudFormation 堆栈时遇到问题。以下部分可帮助您解决您可能遇到的一些常见问题。

有关 CloudFormation 的一般性问题，请参阅 Amazon CloudFormation 常见问题。您还可以在 Amazon Web Services re:Post 上的 CloudFormation 社区中搜索答案和发布问题。

主题

- 故障排除指南
- 错误故障排除
- 联系支持人员
- 部署前验证错误
- Express Mode 完成行为

故障排除指南

如果 CloudFormation 创建、更新或删除堆栈失败，则可通过查看错误消息或日志来帮助详细了解问题。以下任务描述了解决 CloudFormation 问题的通用方法。有关特定的错误和解决方案的信息，请参阅错误故障排除部分。

- 使用操作 ID 和故障筛选进行精准的故障排除。CloudFormation 为部署事件分配唯一的操作 ID，从而更易于跟踪和关联相关事件。您可以按操作 ID 筛选堆栈事件，以便专注于特定的部署阶段，并使用故障筛选条件来迅速找出部署失败的根本原因。请参阅 按操作查看堆栈事件。
- 使用 CloudFormation 控制台查看堆栈的状态。在控制台中，您可以在创建、更新或删除堆栈时查看堆栈事件的列表。堆栈事件包括操作 ID，这些 ID 可帮助您关联相关事件并跟踪单个操作。您可以单击操作 ID 来进行重点故障排除。从操作事件列表中，找到失败事件，然后查看该事件的状态原因。状态原因可能包含来自 CloudFormation 或特定服务的，有助于您排查问题的错误消息。有关使用操作 ID 查看堆栈事件的更多信息，请参阅 按操作查看堆栈事件。
- 对于 Amazon EC2 问题，请查看 `cloud-init` 和 cfn 日志。这些日志是在 Amazon EC2 实例上的 `/var/log/` 目录中发布的。这些日志会捕获 CloudFormation 设置实例时的进程和命令输出。对于 Windows，请查看 `%ProgramFiles%\Amazon\EC2ConfigService` 中的 EC2Configure 服务、`%ProgramData%\Amazon\EC2-Windows\Launch\Logs` 中的 EC2 Launch、`%ProgramData%\Amazon\EC2Launch\log` 中的 EC2 Launch v2 和 `C:\cfn\log` 中的 cfn 日志。

您也可以配置 CloudFormation 模板，以将日志发布到 Amazon CloudWatch，从而在 Amazon Web Services 管理控制台中显示日志，无需连接到 Amazon EC2 实例。有关更多信息，请参阅应用程序管理博客中的 View CloudFormation logs in the console。

错误故障排除

遇到以下与 CloudFormation 堆栈有关的错误时，您可以使用以下解决方案来帮助确定问题的根源并解决问题。

主题

- 删除堆栈失败
- 依赖关系错误
- Amazon Config 和 Amazon Systems Manager 冲突
- 传递列表时解析参数出错
- IAM 权限不足
- 无效的值或不受支持的资源属性
- 超出配额
- 嵌套堆栈陷入 UPDATE_COMPLETE_CLEANUP_IN_PROGRESS、UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_ 或 UPDATE_ROLLBACK_IN_PROGRESS 状态
- 无需执行更新
- 资源在创建、更新或删除堆栈操作期间无法稳定工作
- VPC 中不存在安全组
- 更新回滚失败
- 等待条件未收到来自 Amazon EC2 实例的所需数目的信号
- 资源已从堆栈中移除但未删除

删除堆栈失败

要解决此情况，请尝试：

- 一些资源必须为空才能进行删除。例如，必须先删除 Amazon S3 存储桶中的所有对象或删除 Amazon EC2 安全组中的所有实例，然后才能删除该存储桶或安全组。

- 确保您具有删除堆栈中的资源所需的 IAM 权限。除 CloudFormation 权限之外，您必须有权使用相关基础服务，例如 Amazon S3 或 Amazon EC2。
- 如果堆栈因 CloudFormation 无法删除资源而处于 DELETE_FAILED 状态，请使用 RetainResources 参数重新运行删除操作，并指定 CloudFormation 无法删除的资源。CloudFormation 将删除该堆栈，但不会删除保留的资源。如果无法删除某个资源 (如包含您希望保留的对象的 S3 存储桶)，但仍需要删除堆栈，则保留资源很有用。

  在您删除堆栈之后，您可以使用其关联的 Amazon 服务手动删除保留的资源。

  或者，您可以考虑将 FORCE_DELETE_STACK 选项与 DeletionMode 参数一起使用。有关强制删除堆栈的更多信息，请参阅 通过 CloudFormation 控制台删除堆栈。
- 无法删除已启用终止保护的堆栈。如果您尝试删除已启用终止保护的堆栈，则删除操作会失败，并且堆栈及其状态将保持不变。请在堆栈上禁用终止保护，然后再重新执行删除操作。

  这包括根堆栈启用终止保护的嵌套堆栈。请在根堆栈上停用终止保护，然后再重新执行删除操作。强烈建议您不要直接删除嵌套堆栈，而是只在删除根堆栈及其所有资源时作为其一部分进行删除。

  有关更多信息，请参阅 CloudFormation 堆栈的删除保护。
- 对于所有其他问题，如果您拥有 Amazon Web Services 支持，则可创建一个 Amazon Web Services 支持 案例。请参阅联系支持人员。

依赖关系错误

要纠正依赖性错误，请将一个 DependsOn 属性添加到依赖您模板中的其他资源的资源。在某些情况下，您必须显式声明依赖项，以便 CloudFormation 能够按正确的顺序创建或删除资源。例如，如果您使用 Internet 网关在同一个堆栈中创建一个弹性 IP 和一个 VPC，则该弹性 IP 必须依赖 Internet 网关连接。有关更多信息，请参阅 DependsOn 属性。

Amazon Config 和 Amazon Systems Manager 冲突

Amazon Config 和 Amazon Systems Manager 可以自动执行基础设施管理任务，但这些任务可能会导致与 CloudFormation 堆栈的部署发生冲突。请执行以下操作以避免任何潜在的冲突：

- 检查相关 Amazon Web Services 账户和 Amazon Web Services 区域中的 Amazon Config 和 Systems Manager 配置。
- 检查是否存在可能会在 CloudFormation 部署期间触发的活动规则或自动化文档。这些都可能会导致冲突或与部署发生冲突的资源依赖项。

- 检查 CloudFormation 模板中是否存在由 Amazon Config 和 Systems Manager 管理的任何资源。检查是否存在潜在的重叠或相互依赖项，并考虑调整模板或自动化配置以避免冲突。
- 在 CloudFormation 部署期间暂时禁用或暂停任何相关 Amazon Config 规则或 Systems Manager 自动化。不过要记住在成功部署后恢复原始配置，以确保所需的自动化水平和合规性。
- 检查 CloudFormation 日志和错误消息，了解有关提及 Amazon Config 和 Systems Manager 相关问题的项，以帮助查明冲突的根源。

有关 Amazon Config 规则的详细信息，请参阅 Evaluating Resources with Amazon Config 规则。

有关 Systems Manager 自动化的详细信息，请参阅 Amazon Systems Manager 自动化。

传递列表时解析参数出错

在使用 Amazon CLI 或 CloudFormation 传入列表时，请在每个逗号的前面添加一个转义字符 (\)。下面的示例说明如何在使用 Amazon CLI 时指定输入参数。

```
ParameterKey=CIDR,ParameterValue='10.10.0.0/16\,10.10.0.0/24\,10.10.1.0/24'
```

IAM 权限不足

在使用 CloudFormation 堆栈时，您不仅需要有权使用 CloudFormation，还必须有权使用模板中描述的基础服务。例如，如果您正在创建 Amazon S3 存储桶或启动 Amazon EC2 实例，则需要拥有对 Amazon S3 或 Amazon EC2 的权限。检查您的 IAM 策略并确认您具有必要的权限，然后才能使用 CloudFormation 堆栈。有关更多信息，请参阅使用 Amazon Identity and Access Management 控制 CloudFormation 访问权限。

无效的值或不受支持的资源属性

在您创建或更新 CloudFormation 堆栈时，堆栈会因无效的输入参数、不受支持的资源属性名称或不受支持的资源属性值而失败。对于输入参数，请验证资源是否存在。例如，在您指定 Amazon EC2 密钥对或 VPC ID 时，资源必须存在于您从中创建或更新堆栈的账户和区域中。您可以使用 Amazon 特定的参数类型来确保使用有效值。

对于资源属性名称和值，请更新您的模板以使用有效的名称和值。有关更多信息，请参阅 Amazon 资源和属性类型参考。

超出配额

确认您未达到资源配额。例如，您可以启动的默认最大 Amazon EC2 按需型实例数为 5 个。如果尝试创建的 Amazon EC2 按需型实例的数目超出账户配额，则实例创建将失败，并且您会收到错误 `Status=start_failed`。要按服务查看默认 Amazon 限额，请参阅 Amazon Web Services 一般参考 中的 Amazon service quotas。

有关 CloudFormation 配额和调整策略，请参阅 了解 CloudFormation 配额。

此外，如果在更新期间要替换某个资源，则 CloudFormation 会首先创建新资源，然后再删除旧资源。此替换可使您的账户超出资源配额，这将导致更新失败。您可以删除超额的资源或请求提高配额。

嵌套堆栈陷入 UPDATE_COMPLETE_CLEANUP_IN_PROGRESS、UPDATE_ROLLBACK_COMPLETE 或 UPDATE_ROLLBACK_IN_PROGRESS 状态

一个嵌套堆栈无法回滚。由于嵌套堆栈之间可能存在资源依赖关系，因此在所有嵌套堆栈完成更新或回滚之前，CloudFormation 不会开始清理嵌套堆栈资源。当某个嵌套堆栈回滚失败时，无论其他嵌套堆栈处于何种状态，CloudFormation 都会取消所有操作。如果一个嵌套堆栈已完成更新或回滚，但由于另一嵌套堆栈回滚失败而导致该堆栈未能收到 CloudFormation 发出的开始清理的信号，则该堆栈会处于 `UPDATE_COMPLETE_CLEANUP_IN_PROGRESS` 或 `UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS` 状态。如果一个嵌套堆栈无法更新，但未收到开始回滚的信号，则该堆栈会处于 `UPDATE_ROLLBACK_IN_PROGRESS` 状态。

嵌套堆栈可能因为曾在 CloudFormation 外部进行更改而无法回滚，此时堆栈模板无法准确反映堆栈的状态。如果嵌入堆栈中的自动扩缩组在创建或更新时的资源信号超时期间不足，那么该嵌套堆栈也可能无法回滚。

要修复堆栈，请联系 Amazon Web Services 支持 。

无需执行更新

要更新 CloudFormation 堆栈，您必须向 CloudFormation 提交模板或参数值更改。但是，CloudFormation 不会将某些模板更改视为更新，例如对删除策略、更新策略、条件声明或输出声明所做的更改。如果您需要做出此类更改而不进行任何其他更改，则可为任何资源添加或修改元数据属性。元数据属性可以是任意值，因为 CloudFormation 不会解释其内容。有关更多信息，请参阅 Metadata 属性。

资源在创建、更新或删除堆栈操作期间无法稳定工作

资源因操作超出 CloudFormation 超时期限或 Amazon 服务已中断而未响应。对于服务中断，检查相关 Amazon 服务是否正在运行，然后重试堆栈操作。

如果 Amazon 服务已成功运行，请检查您的堆栈是否包含下列资源之一：

- AWS::AutoScaling::AutoScalingGroup，用于创建、更新和删除操作
- AWS::CertificateManager::Certificate，用于创建操作
- AWS::CloudFormation::Stack，用于创建、更新和删除操作
- AWS::ElasticSearch::Domain，用于更新操作
- AWS::RDS::DBCluster，用于创建和更新操作
- AWS::RDS::DBInstance，用于创建、更新和删除操作
- AWS::Redshift::Cluster，用于更新操作

这些资源的操作时间可能会超过默认的超时期间。超时期间取决于您使用的资源和凭证。要延长超时期间，请在执行堆栈操作时指定服务角色。如果您已使用服务角色或您的堆栈包含未列出的资源，请联系 Amazon Web Services 支持。

如果您的堆栈处于 UPDATE_ROLLBACK_FAILED 状态，请参阅更新回滚失败。

VPC 中不存在安全组

确认您指定的 VPC 中存在安全组。如果安全组存在，请确保指定安全组 ID 而非安全组名称。例如，AWS::EC2::SecurityGroupIngress 资源具有 SourceSecurityGroupName 和 SourceSecurityGroupId 属性。对于 VPC 安全组，您必须使用 SourceSecurityGroupId 属性并指定安全组 ID。

更新回滚失败

关联资源无法回到初始状态，这会导致失败回滚（UPDATE_ROLLBACK_FAILED 状态）。例如，您可能有一个堆栈正在回滚到已在 CloudFormation 之外删除的旧数据库实例。由于并不知道该数据库已被删除，因此 CloudFormation 假定该数据库实例仍然存在并尝试回滚到该实例，从而导致更新回滚失败。

您可以手动修复错误并继续执行回滚操作，具体视故障原因而定。通过继续执行回滚操作，您可以使堆栈回到工作状态 (UPDATE_ROLLBACK_COMPLETE 状态)，然后尝试再次更新堆栈。对于导致更新回滚失败的常见错误，下表列出了解决方案：

- 未收到所需数量的信号

  使用 signal-resource 命令向等待信号的资源手动发送所需数量的成功信号，然后继续回滚更新。例如，在更新回滚期间，自动扩缩组中的实例可能未在指定的超时时间内发送成功信号。请向此自动扩缩组手动发送成功信号。当您继续执行更新回滚操作时，CloudFormation 会收到您的信号并继续执行回滚操作。

- 在 CloudFormation 之外更改了资源

  手动同步资源，使其与初始堆栈模板匹配，然后继续回滚更新。例如，假设您手动删除了 CloudFormation 尝试回滚到的资源，则必须手动创建与原堆栈中的名称和属性均相同的资源。

- 权限不足

  检查您是否拥有充足的 IAM 权限来修改资源，然后继续回滚更新。例如，您的 IAM policy 也许允许您创建 S3 存储桶，但不允许您修改存储桶。请向您的策略中添加修改操作。

- 安全令牌无效

  CloudFormation 需要一组新的凭证。无需做出任何更改。继续回滚更新，这会刷新凭证。

- 限制错误

  删除您不需要的资源或请求提高配额，然后继续回滚更新。例如，如果您的账户有 5 个 EC2 按需型实例的配额而更新回滚超出了此配额，则回滚操作会失败。

- 资源不稳定

  资源因操作可能超出 CloudFormation 超时值或 Amazon 服务中断而未响应。无需做出任何更改。在资源操作完成或 Amazon 服务恢复运行后，继续回滚更新。

要继续回滚更新，可以使用 CloudFormation 控制台或 Amazon 命令行界面（Amazon CLI）。有关更多信息，请参阅 继续回滚更新。

如果这些解决方案都不起作用，您可以跳过 CloudFormation 无法成功回滚的资源。您可以通过以下任一方式跳过它们：

- 强制完成回滚。CloudFormation 会跳过未能回滚的各项资源，包括嵌套堆栈中的资源。有关更多信息，请参阅 强制完成回滚。
- 跳过一组特定的资源。有关更多信息，请参阅《Amazon CloudFormation API 参考》中有关 ContinueUpdateRollback 操作的 `ResourcesToSkip` 参数。

CloudFormation 会将跳过的资源的状态设置为 UPDATE_COMPLETE 并继续回滚堆栈。回滚完成后，已跳过资源的状态将与堆栈模板中资源的状态不一致。在执行另一个堆栈更新之前，您必须修改资源或更新堆栈以便让堆栈互相保持一致。如果您没有这样做，后续堆栈更新可能失败并使您的堆栈无法恢复。

等待条件未收到来自 Amazon EC2 实例的所需数目的信号

要解决此情况，请尝试：

- 确保您使用的 AMI 已安装 CloudFormation 帮助程序脚本。如果 AMI 不包含帮助程序脚本，您也可以将这些脚本下载到您的实例。有关更多信息，请参阅《Amazon CloudFormation 模板参考指南》中的 CloudFormation 帮助程序脚本参考。
- 确认已对该实例成功运行 `cfn-signal` 命令。您可以查看日志 (例如，`/var/log/cloud-init.log` 或 `/var/log/cfn-init.log`) 来帮助调试实例启动。您可以通过登录实例来检索日志，但必须禁用失败时回滚选项，否则 CloudFormation 会在创建堆栈失败后删除实例。您还可以向 Amazon CloudWatch 发布日志。对于 Windows，您可以查看 `C:\cfn\log` 中的 cfn 日志和 `%ProgramFiles%\Amazon\EC2ConfigService` 中的 EC2Config 服务日志。
- 确认实例已连接到 Internet。如果实例在 VPC 中，则实例应能通过 NAT 设备 (在私有子网中时) 或通过 Internet 网关 (在公有子网中时) 连接到 Internet。要测试实例的 Internet 连接，可尝试访问公有网页，例如 `http://aws.amazon.com`。例如，您可以对实例运行以下命令。它应返回 HTTP 200 状态代码。

  ```
  curl -I https://aws.amazon.com
  ```

  有关配置 NAT 设备的信息，请参阅《Amazon VPC 用户指南》中的 NAT。

资源已从堆栈中移除但未删除

在堆栈更新期间，CloudFormation 已从堆栈中移除了资源，但没有删除该资源。该资源仍然存在，但无法再通过 CloudFormation 进行访问。在以下情况下，可能会在堆栈更新过程中发生这种情况：

- CloudFormation 需要替换现有资源，所以它首先创建一个新资源，然后尝试删除旧资源。
- 您已从堆栈模板中移除资源，因此 CloudFormation 尝试从堆栈中删除该资源。

但是，在某些情况下，CloudFormation 可能无法删除资源。例如，在用户不具有删除给定类型的资源的权限时。

CloudFormation 尝试删除旧资源三次。如果 CloudFormation 无法删除旧资源，其会将旧资源从堆栈中移除并继续更新堆栈。堆栈更新完成后，CloudFormation 会发出 UPDATE_COMPLETE 堆栈事件，但同时会包含一个 StatusReason，以说明无法删除一个或多个资源。CloudFormation 还会针对特定资源发布一个 DELETE_FAILED 事件，以及相应的 StatusReason，以详细说明 CloudFormation 未能删除资源的原因。

要解决这种情况，请使用基础服务的控制台或 API 直接删除资源。

联系支持人员

如果您拥有 Amazon Web Services 支持，则可在 https://console.aws.amazon.com/support/home#/ 创建一个技术支持案例。在联系支持人员之前，请收集以下信息：

- 堆栈 ID。您可以在 CloudFormation 控制台的概述选项卡上找到堆栈 ID。有关更多信息，请参阅 从 CloudFormation 控制台查看堆栈信息。

  > **Important**
  >
  > 不要在 CloudFormation 外部更改堆栈中的资源。在 CloudFormation 外部更改堆栈中的资源可能使其处于不可恢复的状态。

- 任何堆栈错误消息。有关查看堆栈错误消息的信息，请参阅故障排除指南部分。
- 对于 Amazon EC2 问题，请收集 cloud-init 和 cfn 日志。这些日志是在 Amazon EC2 实例上的 /var/log/ 目录中发布的。这些日志在设置实例时捕获流程和命令输出。对于 Windows，收集 %ProgramFiles%\Amazon\EC2ConfigService 和 C:\cfn\log 中的 EC2Configure 服务和 cfn 日志。

您还可以在 Amazon Web Services re:Post 上的 CloudFormation 社区中搜索答案和发布问题。

部署前验证错误

部署前验证可在 CloudFormation 配置资源之前捕获常见错误。当验证失败时，堆栈操作会立即停止。

属性语法验证失败

原因：您的模板中包含与预期架构不匹配的资源属性值——例如，ARN 模式无效、类型不匹配（字符串而非整数）、枚举值无效，或缺少必填属性。

解决方案：查看事件详细信息中的 `ValidationStatusReason`。其中包含发生错误的确切属性路径。修复模板中的属性值并重试该操作。

资源名称冲突

原因：您的模板指定的资源名称在您的账户中已存在，而目标资源类型不支持使用该名称的现有资源。

解决方案：您可以修改模板中的资源名称，或者删除导致冲突的现有资源。

服务配额警告

原因：创建或更新资源将导致超过您当前的 Amazon 服务配额。

解决方案：通过“服务配额”控制台申请提高配额，或修改模板以确保符合当前限制。

绕过验证

如果您确定某个验证错误属于误报，或者您需要在出现警告的情况下继续操作，请使用 `--disable-validation` Amazon CLI 标志，或在 API 调用中将 `DisableValidation` 设置为 `true`。

Express Mode 完成行为

当您使用“Express Mode”时，一旦 CloudFormation 应用了资源配置，堆栈操作即刻完成。这意味着，当堆栈操作报告成功时，资源可能尚未完全投入运行。

> **Note**
>
> 如果堆栈输出中引用了某个资源属性，即使在 Express Mode 下，CloudFormation 也会等待该资源传播完成，以便读取该属性的值。

常见问题

使用“Express Mode”时可能会出现以下问题。

资源暂不可用

您的应用程序可能需要某个资源在部署后立即完全投入运行——例如，Amazon EC2 实例通过运行状况检查，或者 CloudFront 分发在全球范围内完成部署。如果是这样，请等待该资源达到最终状态后再对其进行依赖。

删除操作仍在进行

在 Express Mode 下完成堆栈删除后，某些资源可能尚未被完全移除。如果您立即尝试使用相同的资源名称创建新的堆栈，可能会遇到名称冲突。

依赖资源故障

由于 Express Mode 不会等待资源稳定，因此那些依赖于前置资源完全正常运行的下游资源可能会发生故障。如果发生这种情况，请考虑使用该堆栈的默认部署模式。

有关 Express Mode 的更多信息，请参阅 Express Mode。

使用 Amazon Q 开发者版解决不成功的 CloudFormation 堆栈部署问题

Amazon Q 开发者版是一款由生成式人工智能（AI）提供支持的对话式助手，可为您理解、构建、扩展和操作 Amazon 应用程序提供帮助。有关 Amazon Q 开发者版的信息，请参阅《Amazon Q 开发者版用户指南》。

本文档介绍如何使用 Amazon Q 开发者版来帮助解决不成功的 CloudFormation 部署问题。此功能设计有助于您快速识别和解决堆栈部署期间遇到的问题。

由 Amazon Bedrock 提供支持： Amazon Q 基于 Amazon Bedrock 构建，包括在 Amazon Bedrock 中实施的自动滥用检测，以确保人工智能的安全性和负责任使用。

主题

- 功能
- 工作原理
- 定价
- 区域可用性
- 限制

功能

使用 Amazon Q 开发者版解决堆栈部署问题的解决方案具有以下功能：

- 错误分析：对 CloudFormation 堆栈部署错误提供易于理解的分析。
- 解决建议：提供分步说明解决所发现的问题。

工作原理

当 CloudFormation 堆栈创建或更新失败时，Amazon Web Services 管理控制台 中会显示一条错误消息。

可以选择该错误消息旁边的使用 Q 进行诊断按钮。

Amazon Q 开发者版会分析错误并提供问题解释。

如需其他帮助，请选择帮我解决以获取详细的解决步骤。

定价

有关定价的信息，请参阅 Amazon Q 开发者版定价。

区域可用性

大多数 Amazon Web Services 区域均提供了 Amazon Q 开发者版。如需目前提供 Amazon Q 开发者版的所有区域列表，请参阅《Amazon Q 开发者版用户指南》中的 Amazon Q 开发者版支持的区域。

限制

- 此功能只能通过 Amazon Web Services 管理控制台进行访问。
- 此功能涵盖了前 20 个最常见的错误。但未涵盖所有可能的错误。
- 复杂的故障排除情况可能仍需要 Amazon Web Services 支持 干预。

Amazon CloudFormation 用户指南的文档历史记录

下表介绍了 2018 年 5 月之后对《Amazon CloudFormation 用户指南》内容的重要更改。要接收有关文档更新的通知，您可以订阅 RSS 源。

> **Important**
>
> 描述自 2018 年 5 月起模板参考内容更新的表格行已移至新的《Amazon CloudFormation 模板参考》中。有关这些更新和任何未来更改，请参阅《Amazon CloudFormation 模板参考》中的 Amazon CloudFormation 模板参考的文档历史记录。

变更	说明	日期
提供 PDF 指南	现在，您可以下载 PDF 格式的《Amazon CloudFormation 用户指南》。	2025 年 5 月 30 日
将模板参考内容移至新指南	CloudFormation 发布了《Amazon CloudFormation 模板参考指南》。有关详细信息，请参阅《Amazon CloudFormation 模板参考指南》https://docs.amazonaws.cn/AWSCloudFormation/latest/UserGuide/link-to-reference-guide.html。	2025 年 5 月 30 日
IaC 生成器支持部分扫描	您现在可以选择要扫描的特定资源类型，从而更轻松地从现有资源生成基础设施即代码（IaC）模板。有关更多信息，请参阅使用 CloudFormation IaC 生成器启动资源扫描。	2025 年 3 月 27 日
堆栈重构	堆栈重构可简化 CloudFormation 堆栈中资源的重新整理，	2025 年 2 月 6 日

	同时仍保留现有资源属性和数据。有关更多信息，请参阅堆栈重构。	
使用 Amazon Q 开发者版对堆栈部署进行故障排除	现在，您可以使用 Amazon Q 开发者版来解决部署 CloudFormation 堆栈时常见的错误。有关更多信息，请参阅使用 Amazon Q 开发者版解决不成功的 CloudFormation 堆栈部署问题。	2024 年 11 月 22 日
堆栈部署时间线图	现在，您可以看到堆栈部署的可视化表示。堆栈部署时间线图显示堆栈部署状态、单个资源部署状态以及部署状态更改时间。有关更多信息，请参阅查看 CloudFormation 堆栈部署的时间线。	2024 年 11 月 11 日
可视化显示扫描的资源和生成的模板	您现在可以在部署基础设施堆栈前，可视化显示扫描摘要详细信息并预览生成的模板，从而简化基础设施即代码（IaC）生成器工作流。有关更多信息，请参阅在 CloudFormation 控制台中查看扫描摘要和根据扫描的资源创建 CloudFormation 堆栈。	2024 年 8 月 22 日
Amazon EventBridge 与 Amazon CloudFormation Git 同步集成	Amazon CloudFormation Git 同步现在将同步状态更改作为事件发布到 Amazon EventBridge。有关更多信息，请参阅存储库同步状态更改事件详细信息和资源同步状态更改事件详细信息。	2024 年 7 月 29 日

强制删除卡住的堆栈	有两个强制删除堆栈的新选项可用于卡住的堆栈删除操作。现在，您可以选择强制删除堆栈但保留资源，也可以选择强制删除整个堆栈。有关更多信息，请参阅通过 CloudFormation 控制台删除堆栈。	2024 年 5 月 22 日
Amazon CloudTrail 事件堆栈操作的根本原因	CloudFormation 通过新的 Amazon CloudTrail 深度链接集成，改善了堆栈操作的问题排查体验。此功能可将 CloudFormation 控制台中的堆栈操作事件直接链接到相关的 CloudTrail 事件。有关更多信息，请参阅确定堆栈失败的原因。	2024 年 5 月 15 日
属性级别更改集	属性级别更改集允许您预览 CloudFormation 部署将对资源属性值所做的更改。有关更多信息，请参阅查看 CloudFormation 堆栈的更改集。	2024 年 4 月 12 日
CloudFormation 引入了 CONFIGURATION_COMPLETE 事件	现在，您可以使用 CONFIGURATION_COMPLETE 事件，以加快涉及资源创建的工作流程。有关更多信息，请参阅了解 CloudFormation 堆栈创建事件。	2024 年 3 月 11 日
从现有 Amazon 资源生成 Amazon CloudFormation 模板和 Amazon CDK 应用程序	现在，您可以使用账户中已预置但尚未由 CloudFormation 管理的资源生成模板。有关更多信息，请参阅使用 IaC 生成器为现有资源生成模板。	2024 年 2 月 2 日

StackSets 并发模式	并发模式是 StackSetOperationPreferences 的参数，允许您选择并发级别在堆栈集操作期间的行为方式。有关更多信息，请参阅为 CloudFormation StackSets 选择并发模式。	2023 年 11 月 9 日
详细的 StackSet 偏差信息	以下 API 可让您查看已偏离 StackSet 模板的堆栈实例以及已偏离的资源。 ListStackInstanceResourceDrifts 返回堆栈实例中资源的偏差信息。 StackInstanceResourceDriftsSummary 该结构包含有关堆栈实例资源偏离的摘要信息。	2023 年 7 月 24 日

用于控制 Amazon Organizations 信任访问权限的 CloudFormation StackSets API	CloudFormation StackSets 为客户提供以下 API 来管理 Amazon Organizations 信任访问权限： ActivateOrganizationsAccess 激活 Amazon Organizations 的可信访问权限。激活 StackSets 和 Organizations 之间的可信访问权限后，管理账户有权为您的组织创建和管理 StackSets。 DeactivateOrganizationsAccess 停用 Amazon Organizations 的可信访问权限。如果停用可信访问权限，管理账户无权为您的组织创建和管理服务托管的 StackSets。 DescribeOrganizationsAccess 检索有关账户 `OrganizationAccess` 状态的信息。此 API 可以由管理账号调用，也可以由委派管理员使用 `CallAs` 参数调用。管理账号也可以不使用 `CallAs` 参数调用此 API。	2023 年 6 月 5 日
`DescribeStackSet` API	`DescribeStackSet` API 在部署了给定堆栈集的区域列表中有一个新参数。有关更多信息，请参阅 DescribeStackSet。	2023 年 2 月 1 日

使用 CloudFormation 和 Amazon EventBridge 管理 StackSets 事件	CloudFormation StackSets 通过 Amazon EventBridge 启动事件通知。创建、更新或删除 CloudFormation 堆栈集后，您可以触发事件驱动的操作。有关更多信息，请参阅使用 EventBridge 监控 CloudFormation 和 Git 同步事件。	2022 年 11 月 16 日
改进了有关堆栈集操作的堆栈实例的见解	CloudFormation StackSets 提供了有关堆栈集操作的堆栈实例的更多详细信息： DescribeStackSetOperation 现在，您可以使用 `DescribeStackSetOperation` 为部署期间的堆栈集操作提供失败堆栈实例的计数。 ListStackInstances 现在，您可以使用筛选选项 `LastOperationID` 列出堆栈集操作的堆栈实例。	2022 年 11 月 4 日
使用 CloudFormation 和 Amazon EventBridge 管理事件	当特定 CloudFormation 事件发生时接收通知。有关更多信息，请参阅使用 EventBridge 监控 CloudFormation 和 Git 同步事件。	2022 年 7 月 20 日

账户级别	CloudFormation 宣布正式发布账户筛选条件类型，该功能允许客户将部署目标限制为单个账户，或包含所提供 OU 中的其他账户。有关更多信息，请参阅服务托管堆栈集的账户级别目标。	2022 年 7 月 7 日
CloudFormation 注册表	CloudFormation 宣布全面推出挂钩功能，该功能允许客户在创建、更新或删除堆栈操作之前调用自定义逻辑来自动执行操作或检查资源配置。有关更多信息，请参阅《Amazon CloudFormation 挂钩用户指南》。	2022 年 2 月 10 日
堆栈故障选项	遇到预置失败时，您可以从故障点开始迭代开发应用程序，而无需回滚成功预置的资源。通过指定堆栈故障选项，您可以对处于 CREATE_FAILED 或 UPDATE_FAILED 状态的资源进行故障排除。您可以为所有堆栈部署和更改集操作预置故障选项。有关更多信息，请参阅选择预置资源时如何处理失败。	2021 年 8 月 30 日
将堆栈导入堆栈集	现在，您可以将现有堆栈导入新的或现有的堆栈集。有关更多信息，请参阅将堆栈导入 CloudFormation StackSets。	2021 年 7 月 28 日

增加配额	现在，您可以在 Amazon Web Services 账户中声明默认最大值为 2000 个堆栈。有关更多信息，请参阅了解 CloudFormation 配额。	2021 年 7 月 15 日
发布公有第三方扩展	现在，您可以使用第三方发布者提供的公有扩展，就像使用 Amazon 的扩展一样。有关更多信息，请参阅使用 CloudFormation 注册表中的第三方公有扩展。	2021 年 6 月 21 日
堆栈集模板中的引用宏	StackSets 现在支持通过引用宏的模板使用自行管理的权限创建或更新堆栈集。有关宏的更多信息，请参阅使用模板宏对 CloudFormation 模板执行自定义处理。	2021 年 4 月 14 日
在动态引用中使用 Systems Manager 参数的最新值	现在，您可以使用 CloudFormation 在创建或更新堆栈时使用最新版本的 Systems Manager 参数。您不再需要指定具体版本。有关更多详细信息，请参阅获取 Systems Manager Parameter Store 中的纯文本值。	2021 年 4 月 13 日
模块支持在资源名称中使用句点分隔符	现在，您可以在为模块中包含的资源指定完全限定的逻辑名称时使用句点作为分隔符。有关更多信息，请参阅 CloudFormation 模板中的引用模块资源。	2021 年 4 月 8 日

CloudFormation StackSets 现在支持并行区域部署	现在，您可以选择将 StackSets 按顺序或并行部署到区域中。有关更多信息，请参阅堆栈集操作选项。	2021 年 4 月 6 日
CloudFormation StackSets 现在支持具有 Amazon Organizations 权限的委派管理员	除了组织的 管理账户 之外，委托管理员账户也可以为其组织创建和管理具有服务托管权限的堆栈集。有关更多信息，请参阅注册委托管理员成员账户和创建具有服务托管权限的 CloudFormation StackSets。	2021 年 2 月 18 日
CloudFormation StackSets 区域可用性	CloudFormation StackSets 现已在亚太地区（大阪）区域开放。有关更多信息，请参阅使用 StackSets 跨账户和区域管理堆栈。	2021 年 2 月 10 日
模块	模块是一种供您以透明、易管理和可重复的方式打包资源配置以便跨堆栈模板实现包含的方法。模块可以将常见服务配置和最佳实践封装为模块化、可自定义的构建基块，供您包含在堆栈模板中。有关更多信息，请参阅使用 CloudFormation 模块创建可跨模板包含的可重复使用的资源配置。	2020 年 11 月 24 日
嵌套堆栈的更改集	使用嵌套堆栈的更改集，您可以预览整个嵌套堆栈层次结构中应用程序和基础架构资源的更改，并在确认所有更改均是预期更改后继续进行更新。有关详细信息，请参阅嵌套堆栈的更改集。	2020 年 11 月 18 日

增加配额	以下 Amazon CloudFormation 配额已更新。 • 现在，您可以声明 Amazon CloudFormation 模板中最多有 200 个映射。 • 现在，您可以声明 Amazon CloudFormation 模板中的每个映射最多有 200 个映射属性。 • 现在，您可以声明 Amazon CloudFormation 模板中最多有 200 个输出。 • 现在，您可以声明 Amazon CloudFormation 模板中最多有 200 个参数。 • 现在，您可以声明 Amazon CloudFormation 模板中最多有 500 资源。 • 现在，您可以在 Amazon S3 对象中传递最大大小为 1 MB 的模板主体。	2020 年 10 月 22 日
私有资源的偏差检测	CloudFormation 现在支持对扩展的 Amazon 资源列表以及在 CloudFormation 注册表中定义为可预置的私有资源执行偏差检测操作。有关更多信息，请参阅资源类型支持。	2020 年 10 月 1 日
注册资源提供程序所需的更新权限	现在，在您的账户中注册资源提供程序要求您有权访问上传到该资源提供程序的 S3 存储桶的模式处理程序包。有关更多信息，请参阅用于注册第三方私有扩展的 IAM 权限。	2020 年 8 月 7 日

资源导入支持可预置的私有资源类型	导入操作现在支持可预置 的私有资源类型，即其预置类型为 FULLY_MUTABLE 或 IMMUTABLE 。有关详细信息，请参阅支持导入操作的资源。	2020 年 6 月 3 日
通过 CodeDeploy 进行 ECS 蓝/绿部署	您现在可以使用 CloudFormation 通过 CodeDeploy 进行 ECS 蓝/绿部署。蓝/绿部署是 Amazon CodeDeploy 提供的一种安全部署策略，旨在最大限度地减少因应用程序版本更改而造成的中断。有关更多信息，请参阅使用 CloudFormation 通过 CodeDeploy 进行 ECS 蓝/绿部署。	2020 年 5 月 19 日
CloudFormation StackSets 区域可用性	CloudFormation StackSets 现已在 Amazon GovCloud（美国西部）区域开放。	2020 年 5 月 18 日
Amazon CloudFormation StackSets 与 Amazon Organizations 集成	现在，您可以使用 StackSets 集中管理到组织中的所有账户或 Amazon Organizations 中特定组织单元（OU）的部署。您可以对添加到您的组织或 OU 的任何新账户启用自动部署功能。跨账户部署所需的权限将由 StackSets 自动处理。有关更多信息，请参阅使用 StackSets 跨账户和区域管理堆栈。	2020 年 2 月 11 日

StackSets 的偏差检测	现在，您可以对堆栈集及其包含的所有堆栈实例运行偏差检测。有关更多信息，请参阅对 CloudFormation StackSets 进行偏差检测。	2019 年 11 月 19 日
现在提供了 CloudFormation 注册表	现在，您可以使用 CloudFormation 控制台查看可以在您的账户中使用的私有和公有资源。有关更多信息，请参阅查看 CloudFormation 注册表中可用和已激活的扩展。	2019 年 11 月 18 日

CloudFormation 注册表 API 操作	现在，可以使用以下 API 操作以在 CloudFormation 注册表中管理类型。 DeregisterType 从 CloudFormation 注册表中删除当前使用的类型或类型版本。 DescribeType 返回有关注册的类型的详细信息。 DescribeTypeRegistration 返回有关类型的注册的信息，包括其当前状态以及类型和版本标识符。 ListTypeRegistrations 返回指定的类型的注册请求标识符列表。 ListTypes 返回有关已在 CloudFormation 中注册的类型的摘要信息。 ListTypeVersions 返回有关类型版本的摘要信息。 RegisterType 使用 CloudFormation 注册表注册类型。通过注册类型，可以在您的 Amazon Web Services 账户的	2019 年 11 月 18 日

	CloudFormation 模板中使用该类型。 SetTypeDefaultVersion 指定类型的默认版本。将在 CloudFormation 操作中使用类型的默认版本。 有关 CloudFormation 注册表的更多信息，请参阅通过 CloudFormation 注册表管理扩展	
添加了资源导入	如果创建了 Amazon 资源而未使用 CloudFormation 进行管理，您可以使用 `resource import` 将此现有资源纳入 CloudFormation 管理。有关更多信息，请参阅使用资源导入将 Amazon 资源导入 CloudFormation 堆栈。	2019 年 11 月 11 日
堆栈集限额提升	现在，您可以在管理员账户中最多创建 100 个堆栈集，每个堆栈集最多创建 2000 个堆栈实例，并且每个管理员账户可以同时在每个区域中最多运行 3500 个堆栈实例操作。有关更多信息，请参阅了解 CloudFormation 配额。	2019 年 8 月 2 日
并发堆栈操作中的资源限制	CloudFormation 现在对并发堆栈操作中的资源数量实施账户限制。此限制按具体区域确定。有关更多信息，请参阅了解 CloudFormation 配额。	2019 年 4 月 30 日

堆栈实例操作限制	对于 StackSets，您现在可以针对每个管理员账户最多同时具有在给定区域中运行的 1500 个堆栈实例操作。有关更多信息，请参阅了解 CloudFormation 配额。	2018 年 12 月 13 日
CAPABILITY_AUTO_EXPAND 功能现已开放	现在，您可以使用 CAPABILITY_AUTO_EXPAND 功能直接从包含宏的堆栈模板创建或更新堆栈，而无需首先查看更改集中生成的更改。有关更多信息，请参阅 Amazon CloudFormation API Reference 中的 CreateStack 或 UpdateStack。	2018 年 12 月 7 日
添加了堆栈偏差检测	现在，您可以检测堆栈的实际配置是否偏离 CloudFormation 中定义的预期模板配置。您可以在整个堆栈或单个堆栈资源上检测偏差。有关更多信息，请参阅使用偏移检测功能检测堆栈和资源的非托管配置更改。	2018 年 11 月 13 日

secretsmanager 动态引用功能现已开放	现在，您可以使用 secretsmanager 动态引用检索在 Amazon Secrets Manager 中存储的全部密钥或密钥值。密钥可以是数据库凭证、密码、第三方 API 密钥，甚至是任意文本。使用 secretsmanager 动态引用可保证 Secrets Manager 和 CloudFormation 都不记录或保留任何已解析的密钥值。有关更多信息，请参阅获取 Secrets Manager 中的密钥或密钥值。	2018 年 11 月 9 日
现在提供宏	现在，您可以使用宏来对模板执行自定义处理，从查找并替换操作等简单操作到整个模板的大量转换。有关更多信息，请参阅使用模板宏对 CloudFormation 模板执行自定义处理。	2018 年 9 月 6 日
CloudFormation 现在支持由 PrivateLink 提供支持的 VPC 端点	您可以使用 VPC 端点在您的 VPC 和 CloudFormation 之间创建私有连接，而无需通过互联网、NAT 实例、VPN 连接或 Amazon Direct Connect 进行访问。有关更多信息，请参阅使用接口端点（Amazon PrivateLink）访问 CloudFormation。	2018 年 8 月 22 日

动态引用支持安全字符串	现在，您可以使用新的动态引用在堆栈模板中指定在其他服务中存储和管理的值，包括 Systems Manager Parameter Store `SecureString` 类型参数。有关更多信息，请参阅获取 Systems Manager Parameter Store 中的安全字符串值。	2018 年 8 月 16 日
堆栈集现在支持自定义的执行角色	现在，您可以使用目标账户中的自定义执行角色控制用户或组可在其堆栈集中包含的堆栈资源。有关更多信息，请参阅为堆栈集操作设置高级权限选项。	2018 年 5 月 30 日
堆栈实例的选择性更新	现在，您可以使用可选的 `Accounts` 和 `Regions` 参数指定在堆栈集更新操作期间将更新其中的堆栈实例的账户和区域。有关更多信息，请参阅《Amazon CloudFormation API Reference》中的 UpdateStackSet。	2018 年 5 月 30 日
CloudFormation 现在创建支持加密的 S3 存储桶	对于 CloudFormation 创建的、用来存储已上传堆栈模板的 Amazon S3 存储桶，默认情况下，现在已启用服务器端加密，因此将对这些存储桶中存储的所有对象进行加密。有关更多信息，请参阅通过 CloudFormation 控制台创建堆栈。	2018 年 5 月 24 日

已添加 FIPS 端点	CloudFormation 现在提供了新端点，它们在以下公共美国区域使用通过 FIPS 140-2 验证的加密模块：US-East-1、US-East-2、US-West-1 和 US-West-2。请参阅 Amazon Web Services 一般参考 中的 Amazon CloudFormation endpoints and quotas 以了解新的符合 FIPS 的端点 URL。	2018 年 5 月 17 日

有关《Amazon CloudFormation Hooks User Guide》的更新，请参阅《Amazon CloudFormation Hooks User Guide》中的 Document history for the Amazon CloudFormation Hooks User Guide。

归档更新

下表描述了在 2018 年 5 月前每次发布《Amazon CloudFormation 用户指南》时进行的重要更改。

更改	发行日期	说明	API 版本
更新的资源	2019 年 7 月 22 日	使用 `encryptionOptions` 属性可为 Amazon MQ 代理指定 Amazon 拥有的密钥 或客户托管密钥。	2010-05-15
堆栈集命名约定	2018 年 4 月 10 日	使用堆栈集创建的 CloudFormation 堆栈现在遵循新的命名约定，其中堆栈名称包含堆栈集名称。	2010-05-15
新资源	2018 年 4 月 10 日	AWS::AppSync::ApiKey 使用 `AWS::AppSync::ApiKey` 资源可创建唯一密钥，您可将该密钥分发给使用 Amazon AppSync 执行 GraphQL 操作的客户端。 AWS::AppSync::DataSource 使用 `AWS::AppSync::DataSource` 资源为 Amazon AppSync 中的解析程序创建数据源。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::AppSync::GraphQLApi 使用 AWS::AppSync::GraphQLApi 资源创建新的 Amazon AppSync GraphQL API。 AWS::AppSync::GraphQLSchema 使用 AWS::AppSync::GraphQLSchema 资源为您的 Amazon AppSync GraphQL API 创建数据模型。 AWS::AppSync::Resolver 使用 AWS::AppSync::Resolver 资源定义您将附加到架构中的字段的逻辑 GraphQL 解析程序。	
更新的资源	2018 年 4 月 10 日	AWS::Config::ConfigurationAggregator 使用 OrganizationAggregationSource 属性类型可指定要聚合到 Amazon Config 配置聚合器中的 Amazon Config 数据的区域以及要用于检索 Amazon Organizations 详细信息的 IAM 角色。	2010-05-15
新资源	2018 年 4 月 4 日	AWS::Config::AggregationAuthorization 使用 AWS::Config::AggregationAuthorization 资源可向聚合器账户授予收集您的 Amazon Config 数据的权限。 AWS::Config::ConfigurationAggregator 使用 AWS::Config::ConfigurationAggregator 资源可为 Amazon Config 创建配置聚合器。	2010-05-15
堆栈集现在支持自定义的管理员角色	2018 年 3 月 29 日	使用自定义的管理员角色来控制哪些用户或组可以在相同的管理员账户中管理特定堆栈集。有关更多信息，请参阅为堆栈集操作设置高级权限选项。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2018 年 3 月 29 日	AWS::EC2::LaunchTemplate 使用 AWS::EC2::LaunchTemplate 资源可为 Amazon EC2 实例创建启动模板。	2010-05-15
更新的资源	2018 年 3 月 29 日	AWS::AutoScaling::AutoScalingGroup 使用 LaunchTemplate 属性指定用于启动实例的启动模板。 AWS::EC2::SpotFleet 在 SpotFleetRequestConfigData 属性类型中，使用 LaunchTemplateConfigs 属性可描述启动模板和覆盖。	2010-05-15
新的内置函数 Fn::Cidr	2018 年 3 月 6 日	返回指定的 Cidr 地址块。有关更多信息，请参阅 Fn::Cidr。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2018 年 3 月 6 日	AWS::ApiGateway::VpcLink 使用 AWS::ApiGateway::VpcLink 资源为 AWS::ApiGateway::RestApi 指定 API Gateway VPC 链接以访问 Amazon Virtual Private Cloud (VPC) 中的资源。 AWS::GuardDuty::Master 使用 AWS::GuardDuty::Master 资源创建 GuardDuty 主账户。 AWS::GuardDuty::Member 使用 AWS::GuardDuty::Member 资源创建 GuardDuty 成员账户。 AWS::SES::ConfigurationSet 使用 AWS::SES::ConfigurationSet 资源创建可应用于您发送的电子邮件的规则组。 AWS::SES::ConfigurationSetEventDestination 使用 AWS::SES::ConfigurationSetEventDestination 资源指定配置集事件目标。 AWS::SES::ReceiptFilter 使用 AWS::SES::ReceiptFilter 资源指定是接受还是拒绝来自某个 IP 地址或 IP 地址范围的邮件。 AWS::SES::ReceiptRule 使用 AWS::SES::ReceiptRule 资源可指定 Amazon SES 在代表您拥有的一个或多个电子邮件地址或域接收邮件时应采取哪些操作。 AWS::SES::ReceiptRuleSet 使用 AWS::SES::ReceiptRuleSet 资源可为 Amazon SES 指定空规则集。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::SES::Template 使用 AWS::SES::Template 资源指定电子邮件内容，包括主题行、HTML 部分和纯文本部分。	

更改	发行日期	说明	API 版本
更新的资源	2018 年 3 月 6 日	AWS::AutoScaling::AutoScalingGroup 使用 AutoScalingGroupName 属性指定自动扩缩组的名称。 AWS::ApiGateway::RestApi 使用 ApiKeySourceType 属性指定根据使用计划对请求进行计量的 API 键的源。 使用 MinimumCompressionSize 属性指定一个可为空的整数，用于对 API 启用压缩或禁用压缩。 AWS::ApplicationAutoScaling::ScalingPolicy 在 TargetTrackingScalingPolicyConfiguration 属性类型中，使用 DisableScaleIn 属性可指定是否禁用按目标跟踪策略缩减。 AWS::EC2::SpotFleet 在 LaunchSpecifications 属性类型中，使用 TagSpecifications 属性可指定要在 SpotFleet 创建过程中应用的标签。 AWS::Elasticsearch::Domain 使用 Arn 属性让 Fn::GetAtt 返回域的 Amazon 资源名称（ARN）。 DomainArn 的 Fn::GetAtt 属性已弃用。 AWS::RDS::DBCluster 使用 DBClusterIdentifier 属性指定数据库集群标识符。 AWS::RDS::DBCluster 使用 DBClusterIdentifier 属性指定数据库集群标识符。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::Redshift::Cluster 使用 ClusterIdentifier 属性指定集群的唯一标识符。 AWS::Route53::HealthCheck 在 HealthCheckConfig 属性类型中，使用 Regions 属性可指定您希望 Route 53 运行状况检查程序从其检查指定端点的区域。 AWS::SSM::Document 使用 Tags 属性可指定要应用于文档的 CloudFormation 资源标签。	
更新的资源	2018 年 2 月 19 日	AWS::CodeBuild::Project 使用 Triggers 属性为项目配置 Webhook 以在每次代码更改推送到存储库时开始自动重新构建源代码。这仅适用于 CloudFormation 中的 GitHub 项目。它不适用于 GitHub Enterprise 项目。	2010-05-15
更新的资源	2018 年 2 月 8 日	AWS::DynamoDB::Table 使用 SSESpecification 属性指定用于启用服务器端加密的设置。	2010-05-15
更新的资源	2018 年 2 月 5 日	AWS::CodeBuild::Project 在 Source CodeBuild Project Source 属性类型中： • 使用 GitCloneDepth 属性指定要下载的历史记录的深度。 • 使用 InsecureSsl 属性指定在连接到您的 GitHub Enterprise 项目存储库时是否忽略 SSL 警告。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2018 年 1 月 23 日	AWS::AutoScaling::LifecycleHook 使用 LifecycleHookName 属性指定生命周期挂钩的名称。 AWS::DynamoDB::Table 现在，AttributeDefinitions 属性在更新时需要替换。 AWS::EC2::Instance 使用 CreditSpecification 属性指定 T2 实例的 CPU 使用率的积分选项。 使用 ElasticGpuSpecifications 属性指定 Elastic GPU，这是 GPU 资源，您可以附加到实例用于提升应用程序的图形性能。 AWS::EC2::VPC 现在，InstanceTenancy 属性在从 "dedicated" 更新到 "default" 时不需要中断。 AWS::ECS::Service 使用 HealthCheckGracePeriodSeconds 属性可指定一个时间段（以秒为单位），在任务首次启动后，Amazon ECS 服务计划程序在此时间段内将忽略不正常的 Elastic Load Balancing 目标运行状况检查。 AWS::IoT::TopicRule 在 DynamoDBAction 属性类型中，不再需要 RangeKeyField 和 RangeKeyValue 属性。 AWS::KinesisAnalytics::ApplicationOutput 在 ApplicationOutput 属性类型中，使用 LambdaOutput 属性可将 Lambda 函数标识为配置应用程序输出时的目标。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::Kinesis::Stream 使用 StreamEncryption 属性为指定流启用或更新使用 Amazon KMS key 的服务器端加密。 AWS::Lambda::Function 使用 ReservedConcurrentExecutions 属性可以指定希望为函数预留的最大并发执行数。 AWS::RDS::DBSubnetGroup 使用 DBSubnetGroupName 属性可以为数据库子网组指定名称。 AWS::S3::Bucket 使用 BucketEncryption 属性可以为桶指定默认加密，即为 Amazon S3 托管密钥 SSE-S3 或 Amazon KMS keys（SSE-KMS）桶使用服务器端加密。 在 ReplicationRule 属性类型中，在标识要复制的源对象过程中使用 SourceSelectionCriteria 属性指定额外的筛选器。 在 ReplicationDestination 属性类型中： • 使用 AccessControlTranslation 属性可以指定拥有目标桶的 Amazon 账户的副本所有权。 • 使用 Account 属性可以指定目标存储桶拥有者的账户 ID。 • 使用 EncryptionConfiguration 属性可以为所复制对象的目标存储桶指定加密相关信息。 AWS::SSM::Association 使用 AssociationName 属性可以指定 SSM 文档与 EC2 实例 (该实例包含用于处理该文档的配置代理) 之间关联的名称。	

更改	发行日期	说明	API 版本
回滚触发器已添加到 CloudFormation 控制台。	2018 年 1 月 15 日	利用回滚触发器，您可让 CloudFormation 在堆栈创建和更新期间监控应用程序的状态，并可在应用程序超出您指定的任何警报的阈值时回滚操作。有关更多信息，请参阅监控和回滚堆栈操作。	2010-05-15
更新的资源	2018 年 1 月 12 日	AWS::SSM::Parameter 使用 `AllowedPattern` 属性指定用于验证参数值有效性的正则表达式。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2017 年 12 月 5 日	AWS::Inspector::AsssmentTarget 使用 AWS::Inspector::AsssmentTarget 资源创建 Amazon Inspector 评估目标。 AWS::Inspector::AssessmentTemplate 使用 AWS::Inspector::AssessmentTemplate 资源创建 Amazon Inspector 评估模板。 AWS::Inspector::ResourceGroup 使用 AWS::Inspector::ResourceGroup 资源创建 Amazon Inspector 资源组，该组定义标识构成 Amazon Inspector 评估目标的 Amazon 资源。 AWS::ServiceDiscovery::Instance 使用 AWS::ServiceDiscovery::Instance 资源可指定有关 Amazon Route 53 创建的实例的信息。 AWS::ServiceDiscovery::PrivateDnsNamespace 使用 AWS::ServiceDiscovery::PrivateDnsNamespace 资源可指定有关 Amazon Route 53 的私有命名空间的信息。 AWS::ServiceDiscovery::PublicDnsNamespace 使用 AWS::ServiceDiscovery::PublicDnsNamespace 资源可指定有关 Amazon Route 53 的公有命名空间的信息。 AWS::ServiceDiscovery::Service 使用 AWS::ServiceDiscovery::Service 资源可为您希望 Amazon Route 53 在您注册实例时创建的最多 5 个记录和一个可选运行状况检查定义一个模板。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 12 月 5 日	AWS::KinesisAnalytics::Application 在 Input 属性类型中，使用 InputProcessingConfiguration 属性可转换记录，就像从流中接收这些记录一样。	2010-05-15
更新的资源	2017 年 12 月 1 日	AWS::CodeBuild::Project 使用 BadgeEnabled 属性为项目的构建徽章生成公开访问 URL。 使用 Cache 属性为构建依赖项配置缓存设置。 使用 VpcConfig 属性使 CodeBuild 能够访问 Amazon VPC 中的资源。 在 EnvironmentVariable 属性类型中，使用 Type 属性可指定环境变量的类型。	2010-05-15
新资源	2017 年 11 月 30 日	AWS::Cloud9::EnvironmentEC2 使用 AWS::Cloud9::EnvironmentEC2 资源可在 Amazon Cloud9 中创建 Amazon EC2 开发环境。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 11 月 29 日	AWS::ECS::TaskDefinition 使用 Cpu 属性可指定任务所需的 CPU 单元的数量。 使用 ExecutionRoleArn 属性可指定执行角色的 ARN。 使用 Memory 属性可指定任务所需的内存量 (MiB)。 使用 RequiresCompatibilities 属性可指定任务需要的启动类型。 AWS::ECS::Service 使用 LaunchType 属性可指定要在其上运行您的服务的启动类型。 使用 NetworkConfiguration 属性可指定服务的网络配置。 使用 PlatformVersion 属性可指定要在其上运行您的服务的平台版本。	2010-05-15
新资源	2017 年 11 月 28 日	AWS::GuardDuty::Detector 使用 AWS::GuardDuty::Detector 资源可创建单个 Amazon GuardDuty 探测器。 AWS::GuardDuty::IPSet 使用 AWS::GuardDuty::IPSet 资源可创建 Amazon GuardDuty IP 集。 AWS::GuardDuty::ThreatIntelSet 使用 AWS::GuardDuty::ThreatIntelSet 资源可创建 ThreatIntelSet。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 11 月 28 日	AWS::CodeDeploy::Application 使用 ComputePlatform 属性可为 CodeDeploy 指定要将应用程序部署到的 Amazon Lambda 计算平台。 AWS::CodeDeploy::DeploymentGroup 在 DeploymentStyle 属性类型中，使用 DeploymentType 属性可指定 Lambda 计算平台上的蓝/绿部署。 AWS::EC2::SpotFleet 在 SpotFleetRequestConfigData 属性类型中，SpotPrice 属性现在是可选的。 AWS::Lambda::Alias 使用 RoutingConfig 属性可指定两个不同版本的 Amazon Lambda 函数，并允许您指示将调用每个版本的流量的百分比。	2010-05-15
新的 CodeDeployLambdaAliasUpdate 更新策略	2017 年 11 月 28 日	使用 CodeDeployLambdaAliasUpdate 更新策略可在 AWS::Lambda::Alias 资源上的版本发生更改时执行 CodeDeploy 部署。有关更多信息，请参阅 UpdatePolicy 属性。	2010-05-15
新的 SSM 参数类型	2017 年 11 月 21 日	使用 SSM 参数类型可使用 Systems Manager Parameter Store 中的现有参数。注意：CloudFormation 当前不支持 SecureString 类型。有关更多信息，请参阅 SSM 参数类型。	2010-05-15

更改	发行日期	说明	API 版本
`ResolvedValue` 数据类型的新 `Parameter` 字段	2017 年 11 月 21 日	`ResolvedValue` 字段将返回在 SSM 参数的堆栈定义中使用的值。有关更多信息，请参阅 Amazon CloudFormation API Reference中的 Parameter 数据类型。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 11 月 20 日	AWS::ApiGateway::ApiKey 使用 CustomerId 属性可指定 Amazon Marketplace 客户标识符。 使用 GenerateDistinctId 属性可指定键标识符是否不同于创建的 API 键值。 AWS::ApiGateway::Authorizer 使用 AuthType 属性可指定一个客户定义的字段，可在 Swagger 导入和导出中使用该字段而不会影响功能。 AWS::ApiGateway::DomainName 使用 EndpointConfiguration 属性可指定 API Gateway 域名的端点类型。 使用 RegionalCertificateArn 属性可引用供域名的区域端点使用的证书。 AWS::ApiGateway::Method 在 Integration 和 IntegrationResponse 属性类型中，使用 ContentHandling 属性可指定如何处理请求负载内容类型转换。 AWS::ApiGateway::RestApi 使用 EndpointConfiguration 属性可指定 API Gateway REST API 的端点类型。 AWS::ApplicationAutoScaling::ScalableTarget 使用 ScheduledActions 属性可指定 Application Auto Scaling 可扩展目标的计划操作。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::ECR::Repository 使用 LifecyclePolicy 属性可指定 Amazon ECR 存储库的生命周期策略。 AWS::ECS::TaskDefinition 在 ContainerDefinition 属性类型中，使用 LinuxParameters 属性可指定 Amazon ECS 容器的特定于 Linux 的选项。 AWS::ElastiCache::ReplicationGroup 使用 AtRestEncryptionEnabled 属性可启用静态加密。 使用 AuthToken 属性可指定用于访问受密码保护的服务器的密码。 使用 TransitEncryptionEnabled 属性可启用传输中加密。 AWS::ElasticLoadBalancingV2::TargetGroup 将 TargetGroupName 属性与 Fn::GetAtt 函数一起使用可获取 Elastic Load Balancing 目标组的名称。 AWS::Elasticsearch::Domain 使用 VPCOptions 属性可为 OpenSearch Service 域指定 VPC 配置。 AWS::EMR::Cluster 使用 EbsRootVolumeSize 属性可为 Amazon EMR 集群指定 EBS 根卷的大小。	

更改	发行日期	说明	API 版本
		AWS::RDS::DBInstance 使用 SourceRegion 和 KmsKeyId 属性可从跨区域源数据库实例创建加密的只读副本。 AWS::Route53::HostedZone 使用 QueryLoggingConfig 属性可为 DNS 查询日志记录指定配置。	
自定义资源 NoEcho 对象的新 Response 字段	2017 年 11 月 20 日	您现在可以使用可选 NoEcho 字段遮蔽自定义资源的输出。有关更多信息，请参阅自定义资源响应对象。 noEcho 方法支持对应的 send 参数。有关更多信息，请参阅 cfn-response 模块。	2010-05-15
添加了针对堆栈集的堆栈实例覆盖。	2017 年 11 月 17 日	CloudFormation StackSets 允许您按账户和区域覆盖堆栈实例中的参数值。在创建堆栈实例时或在更新现有堆栈实例时，可以覆盖参数值。有关更多信息，请参阅覆盖堆栈实例上的参数。	2010-05-15
更新的资源	2017 年 11 月 15 日	AWS::StepFunctions::StateMachine 可在创建状态机时使用 AWS::StepFunctions::StateMachine 指定 StateMachineName，并且可以在不更换状态机的情况下更新 DefinitionString 和 RoleArn。	2010-05-15
StackSets 现在在每个堆栈集中最多支持 500 个堆栈实例。	2017 年 11 月 6 日	您现在可以在每个堆栈集中最多创建 500 个堆栈实例。有关 Amazon CloudFormation 限制的更多信息，请参阅了解 CloudFormation 配额。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2017 年 11 月 2 日	AWS::CloudFront::CloudFrontOriginAccessIdentity 使用 AWS::CloudFront::CloudFront OriginAccessIdentity 资源可指定要与 CloudFront 分配源关联的 Amazon CloudFront 源访问身份。 AWS::CloudFront::StreamingDistribution 可以使用 AWS::CloudFront::StreamingDistribution 资源为 CloudFront 指定 Adobe 实时消息协议 (RTMP) 流分配。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 11 月 2 日	AWS::ApiGateway::Deployment StageDescription 属性类型的 StageName 属性已弃用。 AWS::ApiGateway::Method 使用 OperationName 属性可为 API Gateway 方法分配一个易记名称。 可以使用 RequestValidatorId 属性将请求验证程序与一个方法相关联。 AWS::AutoScaling::AutoScalingGroup 使用 LifecycleHookSpecificationList 属性可指定在 Auto Scaling 启动或终止实例时执行的操作。 AWS::CloudFront::Distribution 使用 Tags 属性可指定要与 CloudFront 分配关联的任意标签组（键/值对）。 在 CacheBehavior 和 DefaultCacheBehavior 属性类型中，使用 LambdaFunctionAssociations 属性可为 CloudFront 分配指定 Lambda 函数关联。 在 CustomOriginConfig 属性类型中，使用 OriginKeepaliveTimeout 属性可指定自定义保持活动超时，使用 OriginReadTimeout 属性可指定自定义源读取超时。 在 DistributionConfig 属性类型中，使用 IPV6Enabled 属性可指定 CloudFront 是否使用您的分配的 IPv6 地址响应 IPv6 DNS 请求。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::CodeDeploy::DeploymentGroup 在 LoadBalancerInfo 属性类型中，使用 TargetGroupInfoList 属性可指定有关在部署中使用的 Elastic Load Balancing 目标组的信息。 AWS::EC2::SecurityGroup、AWS::EC2::SecurityGroupEgress 和 AWS::EC2::SecurityGroupIngress 可以使用 Description 属性指定安全组规则的描述。 AWS::EC2::Subnet Ipv6CidrBlock 属性现已支持 No interruption 更新。 AWS::EC2::VPNGateway 可以使用 AmazonSideAsn 属性为 BGP 会话的 Amazon 端指定私有自治系统编号 (ASN)。 AWS::EC2::VPNConnection 可以使用 VpnTunnelOptionsSpecifications 属性为 VPN 连接配置隧道选项。 AWS::ElasticBeanstalk::ConfigurationTemplate 和 AWS::ElasticBeanstalk::Environment 在 ConfigurationOptionSetting 和 OptionSetting 属性类型中，使用 ResourceName 属性可为基于时间的扩展配置选项指定资源名称。 AWS::EMR::Cluster 可以使用 CustomAmiId 属性为集群指定自定义 Amazon Linux AMI。	

更改	发行日期	说明	API 版本
		AWS::KinesisFirehose::DeliveryStream 可以将 Arn 属性与 Fn::GetAtt 函数一起使用以获取传输流的 Amazon 资源名称（ARN）。 AWS::KMS::Key 使用 Tags 属性可指定要与客户托管密钥关联的任意标签组（键值对）。 AWS::OpsWorks::Layer 和 AWS::OpsWorks::Stack 使用 Tags 属性可指定要与 Amazon OpsWorks 层或堆栈关联的任意标签组（键值对）。 AWS::RDS::OptionGroup 在 OptionConfiguration 属性类型中，使用 OptionVersion 属性可为选项指定版本。 AWS::S3::Bucket 使用 AnalyticsConfigurations 属性可为 Amazon S3 存储桶配置分析筛选器。	

更改	发行日期	说明	API 版本
新资源	2017 年 10 月 24 日	AWS::Glue::Classifier 使用 AWS::Glue::Classifier 资源可创建 Amazon Glue 分类器。 AWS::Glue::Connection 使用 AWS::Glue::Connection 资源可指定到数据来源的 Amazon Glue 连接。 AWS::Glue::Crawler 使用 AWS::Glue::Crawler 资源可指定 Amazon Glue 爬网程序。 AWS::Glue::Database 使用 AWS::Glue::Database 资源可创建 Amazon Glue 数据库。 AWS::Glue::DevEndpoint 可以使用 AWS::Glue::DevEndpoint 资源指定开发端点以远程调试 ETL 脚本。 AWS::Glue::Job 使用 AWS::Glue::Job 资源可指定数据目录中的 Amazon Glue 作业。 AWS::Glue::Partition 使用 AWS::Glue::Partition 资源可创建 Amazon Glue 分区，它表示表数据的一部分。 AWS::Glue::Table 使用 AWS::Glue::Table 资源可创建 Amazon Glue 表。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::Glue::Trigger 使用 AWS::Glue::Trigger 资源可指定运行 Amazon Glue 作业的触发器。	
新资源	2017 年 10 月 11 日	AWS::SSM::MaintenanceWindow 使用 AWS::SSM::MaintenanceWindow 资源可创建 Amazon Systems Manager 维护时段。 AWS::SSM::MaintenanceWindowTarget 使用 AWS::SSM::MaintenanceWindowTarget 资源向维护时段注册目标。 AWS::SSM::MaintenanceWindowTask 使用 AWS::SSM::MaintenanceWindowTask 资源定义维护时段任务。 AWS::SSM::PatchBaseline 使用 AWS::SSM::PatchBaseline 资源可定义 Systems Manager 补丁基准。	2010-05-15
新资源	2017 年 10 月 10 日	AWS::ElasticLoadBalancingV2::ListenerCertificate 使用 AWS::ElasticLoadBalancingV2::ListenerCertificate 资源可为 Elastic Load Balancing 侦听器指定证书。	2010-05-15
新资源	2017 年 9 月 27 日	AWS::Athena::NamedQuery 使用 AWS::Athena::NamedQuery 资源可创建 Amazon Athena 查询。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 9 月 27 日	AWS::EC2::NatGateway 使用 Tags 属性为 NAT 网关指定资源标签。 AWS::ElasticBeanstalk::Application 使用 ResourceLifecycleConfig 属性可定义属于应用程序的资源的生命周期设置，以及 Elastic Beanstalk 为应用生命周期设置而假定的服务角色。 AWS::ElasticBeanstalk::ConfigurationTemplate 和 AWS::ElasticBeanstalk::Environment 使用 PlatformArn 属性可为 Elastic Beanstalk 指定自定义平台。 AWS::ElasticLoadBalancingV2::TargetGroup 在 TargetDescription 属性类型中，使用 AvailabilityZone 属性可指定在其中注册 IP 地址的可用区。 AWS::Events::Rule 在 Target 属性类型中，使用以下属性可进行事件的输入转换并设置 Amazon ECS 任务和 Kinesis 流目标。 • EcsParameters • InputTransformer • KinesisParameters • RunCommandParameters AWS::KinesisFirehose::DeliveryStream 使用 DeliveryStreamType 属性指定流类型，并使用 KinesisStreamSourceConfiguration 属性为用作传输流来源的 Kinesis 流指定流和角色的 ARN。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::RDS::DBInstance 对于 Engine 属性，如果指定了 oracle-se 或 oracle-se1 ，则可以在不替换数据库实例的情况下更新至 oracle-se2 。 AWS::S3::Bucket 使用 AccelerateConfiguration 属性可配置 Amazon S3 存储桶的传输加速状态。	
为堆栈添加终止保护。	2017 年 9 月 26 日	在堆栈上启用终止保护可防止堆栈被意外删除。用户无法删除已启用终止保护的堆栈。有关更多信息，请参阅防止堆栈被删除。	2010-05-15
自版本 1.4-22 起更改了默认的 umask 值	2017 年 9 月 14 日	现在 cfn-hup.conf 配置文件的默认 umask 参数值为 022。有关更多信息，请参阅 cfn-hup。	
更新的资源	2017 年 9 月 7 日	AWS::ElasticLoadBalancingV2::LoadBalancer 使用 SubnetMappings 属性可指定要附加到负载均衡器的子网的 ID。 使用 Type 属性可指定要创建的负载均衡器的类型。 AWS::ElasticLoadBalancingV2::TargetGroup 使用 TargetType 属性可指定此目标组中的目标的注册类型。	2010-05-15
回滚触发器已添加到 CloudFormation API	2017 年 8 月 31 日	利用回滚触发器，您可让 CloudFormation 在堆栈创建和更新期间监控应用程序的状态，并可在应用程序超出您指定的任何警报的阈值时回滚操作。有关更多信息，请参阅《Amazon CloudFormation API Reference》中的 RollbackConfiguration。	2010-05-15

更改	发行日期	说明	API 版本
cfn-hup.conf 文件的新 umask 参数	2017 年 8 月 31 日	使用 cfn-hup.conf 配置文件中的 umask 参数可控制 cfn-hup 守护程序 (版本 1.4-21) 使用的文件权限。有关更多信息，请参阅 cfn-hup。	
更新了用于支持 VPC 规模设定的资源	2017 年 29 月 8 日	AWS::EC2::VPCCidrBlock 使用 CidrBlock 属性将 IPv4 CIDR 块与 VPC 关联。 AWS::EC2::VPC 将 CidrBlockAssociations 属性与 Fn::GetAtt 函数结合使用，以获取与 VPC 关联的 IPv4 CIDR 块关联 ID 的列表。	2010-05-15
更新的资源	2017 年 23 月 8 日	AWS::S3::Bucket 在 Rule 属性类型中，使用 TagFilters 属性可指定用于标识 Amazon S3 存储桶的对象子集的标签。 使用 MetricsConfiguration 属性可为 Amazon S3 存储桶中的 CloudWatch 请求指标指定指标配置。 AWS::IoT::TopicRule 在 Action 属性类型中，使用 DynamoDBv2Action 属性可描述将数据写入 DynamoDB 表的 Amazon IoT 操作。 在 Action 属性类型中，DynamoDBAction 属性现在支持 HashKeyType 和 RangeKeyType 属性。 AWS::Lambda::Permission 使用 EventSourceToken 属性指定调用函数的委托人必须提供的唯一令牌。	2010-05-15

更改	发行日期	说明	API 版本
新伪参数	2017 年 23 月 8 日	使用 `AWS::Partition` 伪参数返回某个资源所在的分区。 使用 `AWS::URLSuffix` 伪参数返回域的后缀。 有关更多信息，请参阅虚拟参数参考。	2010-05-15
用于 DAX 支持的新资源	2017 年 8 月 22 日	AWS::DAX::Cluster 可以使用 `AWS::DAX::Cluster` 资源创建与 Amazon DynamoDB 一起使用的 DAX 集群。 AWS::DAX::ParameterGroup 使用 `AWS::DAX::ParameterGroup` 资源可创建用于 Amazon DynamoDB 的参数组。 AWS::DAX::SubnetGroup 使用 `AWS::DAX::SubnetGroup` 资源创建用于 DAX (DynamoDB Accelerator) 的子网组。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2017 年 8 月 18 日	AWS::ApiGateway::DocumentationPart 和 AWS::ApiGateway::DocumentationPart 使用 AWS::ApiGateway::DocumentationPart 和 AWS::ApiGateway::DocumentationVersion 资源可创建 API Gateway API 文档。 AWS::ApiGateway::GatewayResponse 使用 AWS::ApiGateway::GatewayResponse 资源可为 API Gateway API 创建自定义响应。 AWS::ApiGateway::RequestValidator 使用 AWS::ApiGateway::RequestValidator 资源可为传入 API Gateway API 的请求设置验证规则。 AWS::EC2::NetworkInterfacePermission 使用 AWS::EC2::NetworkInterfacePermission 资源向 Amazon 账户授予对网络接口的权限。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 8 月 18 日	AWS::ApiGateway::Stage 使用 DocumentationVersion 属性指定 API 文档受版本控制的快照。 AWS::AutoScaling::ScalingPolicy 使用 TargetTrackingConfiguration 属性可指定 Auto Scaling 目标跟踪扩展策略配置。 AWS::CloudTrail::Trail 使用 EventSelectors 属性可获取 Amazon S3 数据事件支持。 AWS::CodeDeploy::DeploymentGroup 使用 LoadBalancerInfo 和 DeploymentStyle 属性可为就地部署指定 Elastic Load Balancing 负载均衡器。 使用 AutoRollbackConfiguration 属性为部署配置自动回滚。 AWS::EC2::SpotFleet 在 SpotFleetRequestConfigData 属性类型中，使用 ReplaceUnhealthyInstances 属性可指示 Spot 队列是否应替换运行状况不佳的实例，使用 Type 属性可指定请求类型。 AWS::EC2::Subnet 使用 AssignIpv6AddressOnCreation 和 Ipv6CidrBlock 属性创建具有 IPv6 CIDR 块的子网。 AWS::KinesisFirehose::DeliveryStream 使用 ExtendedS3DestinationConfiguration 属性可在 Amazon S3 中配置目标。	2010-05-15

更改	发行日期	说明	API 版本
		使用每个目标配置中的 ProcessingConfiguration 子属性来调用 Lambda 函数，以便转换传入源数据并将转换后的数据传送到目标。 AWS::RDS::DBCluster 和 AWS::RDS::DBInstance 对于 AWS::RDS::DBCluster 资源以及没有指定 DBClusterIdentifier 属性的 AWS::RDS::DBInstance 资源，默认 DeletionPolicy 现在是 Snapshot。有关更多信息，请参阅 DeletionPolicy 属性。 AWS::S3::Bucket 在 Rule 属性类型中，使用 AbortIncompleteMultipartUpload 属性可指定一个生命周期规则，以中止向 Amazon S3 存储桶进行的未完成分段上传。 AWS::SQS::Queue 使用 KmsMasterKeyId 和 KmsDataKeyReusePeriodSeconds 属性可为 Amazon SQS 配置服务器端加密。 向以下资源的 Fn::GetAtt 内置函数添加了 Arn 属性： • AWS::CloudTrail::Trail。还添加了 SnsTopicArn 。 • AWS::CloudWatch::Alarm • AWS::DynamoDB::Table • AWS::ECS::Cluster • AWS::IoT::Policy • AWS::IoT::TopicRule • AWS::Logs::Destination	

更改	发行日期	说明	API 版本
支持 CodePipeline 构件中的堆栈标签	2017 年 8 月 18 日	现在您可以为模板配置文件中的堆栈指定标签，以用作 CodePipeline 管道的构件。指定的标签将应用于使用模板配置文件创建的堆栈。有关更多信息，请参阅 CloudFormation 构件。	2010-05-15
创建加密文件系统	2017 年 8 月 14 日	AWS::EFS::FileSystem 使用 `Encrypted` 属性可在创建过程中加密 Amazon EFS 文件系统。 使用 `KmsKeyId` 属性指定用于保护加密文件系统的自定义客户托管密钥（可选）。	2010-05-15
用于 Amazon Batch 支持的新资源	2017 年 8 月 8 日	AWS::Batch::ComputeEnvironment 使用 `AWS::Batch::ComputeEnvironment` 资源可定义您的 Amazon Batch 计算环境。 AWS::Batch::JobDefinition 使用 `AWS::Batch::JobDefinition` 资源可指定 Amazon Batch 作业定义的参数。 AWS::Batch::JobQueue 使用 `AWS::Batch::JobQueue` 资源可定义您的 Amazon Batch 作业队列。	2010-05-15

更改	发行日期	说明	API 版本
适用于 Apache Flink 的亚马逊托管服务支持的新资源	2017 年 28 月 7 日	AWS::KinesisAnalytics::Application 使用 AWS::KinesisAnalytics::Application 资源创建适用于 Apache Flink 的亚马逊托管服务应用程序。 AWS::KinesisAnalytics::ApplicationOutput 使用 AWS::KinesisAnalytics::ApplicationOutput 资源向适用于 Apache Flink 的亚马逊托管服务应用程序添加外部目标。 AWS::KinesisAnalytics::ApplicationReferenceDataSource 使用 AWS::KinesisAnalytics::ApplicationReferenceDataSource 资源向现有的适用于 Apache Flink 的亚马逊托管服务应用程序添加引用数据来源。	2010-05-15
使用 StackSets 可跨账户和区域集中管理堆栈	2017 年 25 月 7 日	利用 StackSets，您可在一个操作中跨多个账户和区域创建、更新或删除堆栈。通过使用管理员账户，您可定义和管理 CloudFormation 模板，并以该模板为基础，跨指定的区域在选定目标账户中预置堆栈。有关更多信息，请参阅使用 StackSets 跨账户和区域管理堆栈。	2010-05-15
按客户端请求令牌查看堆栈事件	2017 年 14 月 7 日	在控制台中，堆栈操作在 Events 选项卡上显示客户端请求令牌。为给定堆栈操作所触发的所有事件分配相同的客户端请求令牌，可使用此令牌跟踪操作。有关更多信息，请参阅《Amazon CloudFormation API Reference》中的 Viewing CloudFormation Stack Data and Resources on the Amazon Web Services 管理控制台 和 StackEvent。	2010-05-15
使用堆栈快速创建链接	2017 年 14 月 7 日	使用快速创建链接可快速启动并运行堆栈。您可以指定模板 URL、堆栈名称和模板参数来预填充单个 Create Stack Wizard 页面。有关更多信息，请参阅为堆栈创建快速创建链接。	2010-05-15

更改	发行日期	说明	API 版本
用于 Amazon Database Migration Service 支持的新资源	2017 年 7 月 12 日	AWS::DMS::Certificate 使用 AWS::DMS::Certificate 资源可创建一个 SSL 证书，该证书可对 Amazon DMS 端点与复制实例之间的连接进行加密。 AWS::DMS::Endpoint 使用 AWS::DMS::Endpoint 资源可创建 Amazon DMS 端点。 AWS::DMS::EventSubscription 使用 AWS::DMS::EventSubscription 资源通过 Amazon Simple Notification Service 获取 Amazon DMS 事件的通知。 AWS::DMS::ReplicationInstance 使用 AWS::DMS::ReplicationInstance 资源可创建 Amazon DMS 复制实例。 AWS::DMS::ReplicationSubnetGroup 使用 AWS::DMS::ReplicationSubnetGroup 资源可创建 Amazon DMS 复制子网组。 AWS::DMS::ReplicationTask 使用 AWS::DMS::ReplicationTask 资源可创建 Amazon DMS 复制任务。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2017 年 7 月 5 日	AWS::CloudWatch::Dashboard 使用 AWS::CloudWatch::Dashboard 资源可为 CloudWatch 控制台指定自定义 CloudWatch 控制面板。 AWS::ApiGateway::DomainName 使用 AWS::ApiGateway::DomainName 资源可为部署到 Amazon API Gateway 的 API 指定自定义、友好的 URL。 AWS::EC2::EgressOnlyInternetGateway 使用 AWS::EC2::EgressOnlyInternetGateway 资源可为 VPC 创建仅出口互联网网关。 InstanceFleetConfig 使用 InstanceFleetConfig 资源可为 Amazon EMR 集群配置竞价型实例集。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 7 月 5 日	AWS::ApiGateway::RestApi 使用 BinaryMediaTypes 属性可指定支持的二进制媒体类型。 AWS::ApplicationAutoScaling::ScalingPolicy 使用 TargetTrackingScalingPolicyConfiguration 属性可指定目标跟踪扩展策略配置。 AWS::CloudTrail::Trail 使用 TrailName 属性可为 Amazon CloudTrail 资源指定自定义名称。 使用 Tags 属性可指定资源标签。 AWS::CodeDeploy::DeploymentGroup 使用 AlarmConfiguration 属性可为部署组配置警报。 使用 TriggerConfigurations 属性可为部署组配置通知触发器。 AWS::EMR::Cluster 使用 JobFlowInstancesConfig 属性类型中的 CoreInstanceFleet 属性和 MasterInstanceFleet 属性为 Amazon EMR 集群配置竞价型实例集。 AWS::DynamoDB::Table 使用 TimeToLiveSpecification 属性可为 Amazon DynamoDB 表指定生存时间 (TTL) 设置。 使用 Tags 属性可为 DynamoDB 表指定资源标签。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::EC2::Instance IamInstanceProfile 属性现已支持 No interruption 更新。 AWS::EC2::Route 使用 EgressOnlyInternetGatewayId 属性可为 EC2 路由指定仅出口互联网网关。 AWS::Kinesis::Stream 使用 RetentionPeriodHours 属性可指定存储在分片中的数据记录保持可访问状态的小时数。 AWS::RDS::DBCluster 使用 ReplicationSourceIdentifier 属性可创建数据库集群，以作为其他数据库集群或 Amazon RDS MySQL 数据库实例的只读副本。 AWS::Redshift::Cluster 使用 LoggingProperties 属性可创建审计日志文件并将它们存储在 Amazon S3 中。	
新资源	2017 年 6 月 6 日	AWS::EMR::SecurityConfiguration 使用 AWS::EMR::SecurityConfiguration 资源可创建安全配置，后者存储在该服务中，并且可在创建集群时指定。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 6 月 6 日	AWS::AutoScaling::LifecycleHook 现在，NotificationTargetARN 和 RoleARN 是可选属性。 AWS::CloudWatch::Alarm 现在，您可以在创建 EvaluateLowSampleCountPercentile 资源时使用 ExtendedStatistic 、TreatMissingData 和 AWS::CloudWatch::Alarm 属性。 AWS::EC2::SpotFleet CloudFormation 支持对竞价型实例集属性的可变更改。 SpotFleetRequestConfigData 属性的以下属性支持 Replacement 更新： • AllocationStrategy • IamFleetRole • LaunchSpecifications • SpotPrice • TerminateInstancesWithExpiration • ValidFrom • ValidUntil SpotFleetRequestConfigData 属性的以下属性支持 No interruption 更新： • ExcessCapacityTerminationPolicy • TargetCapacity AWS::EMR::InstanceGroupConfig CloudFormation 现在支持 Amazon EMR 任务实例组的自动扩缩。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::Events::Rule RoleArn 资源的 Rule 属性已弃用。 使用 Target 属性类型上的 RoleArn 属性可指定用于目标的 IAM 角色。 AWS::Kinesis::Stream ShardCount 属性现已支持 No interruption 更新。 AWS::Lambda::Function 使用 TracingConfig 属性可为 Lambda 函数配置跟踪设置。 AWS::Redshift::Cluster、AWS::Redshift::ClusterParameterGroup、AWS::Redshift::ClusterSecurityGroup 和 AWS::Redshift::ClusterSubnetGroup 使用 Tags 属性可指定资源标签。 AWS::RDS::DBCluster 向 ReadEndpoint.Address 内置函数添加了 Fn::GetAtt 属性。 AWS::S3::Bucket 向 Arn 内置函数添加了 Fn::GetAtt 属性。	

更改	发行日期	说明	API 版本
新资源	2017 年 5 月 11 日	以下新资源支持将 Amazon WAF 与弹性负载均衡器 (ELB) 应用程序负载均衡器配合使用。 AWS::WAFRegional::ByteMatchSet 使用 AWS::WAFRegional::ByteMatchSet 资源可标识要检查的 Web 请求部分。 AWS::WAFRegional::IPSet 使用 AWS::WAFRegional::IPSet 资源可根据请求来源的 IP 地址指定允许或阻止哪些 Web 请求。 AWS::WAFRegional::Rule 使用 AWS::WAFRegional::Rule 资源可指定 IPSet、ByteMatchSet 和 SqlInjectionMatchSet 对象 (用于识别要允许、阻止或计数的 Web 请求) 组合。 AWS::WAFRegional::SizeConstraintSet 使用 AWS::WAFRegional::SizeConstraintSet 资源可指定用于检查 Web 请求的大小的大小限制以及要检查的请求的部分。 AWS::WAFRegional::SqlInjectionMatchSet 使用 AWS::WAFRegional::SqlInjectionMatchSet 资源可允许、阻止或对 Web 请求的特定部分中包含恶意 SQL 代码的请求进行计数。 AWS::WAFRegional::WebACL 使用 AWS::WAFRegional::WebACL 资源可识别要允许、阻止或计数的 Web 请求。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::WAFRegional::WebACLAssociation 使用 AWS::WAFRegional::WebACLAssociation 资源可将 Web 访问控制组 (ACL) 关联到某个资源。 AWS::WAFRegional::XssMatchSet 使用 AWS::WAFRegional::XssMatchSet 资源指定您希望 Amazon WAF 检查跨站点脚本攻击的 Web 请求部分以及要检查的标头名称。	

更改	发行日期	说明	API 版本
新资源	2017 年 4 月 28 日	AWS::Cognito::IdentityPool 使用 AWS::Cognito::IdentityPool 资源可创建 Amazon Cognito 身份池。 AWS::Cognito::IdentityPoolRoleAttachment 使用 AWS::Cognito::IdentityPoolRoleAttachment 资源可管理 Amazon Cognito 身份池的角色配置。 AWS::Cognito::UserPool 使用 AWS::Cognito::UserPool 资源可创建 Amazon Cognito 用户池。 AWS::Cognito::UserPoolClient 使用 AWS::Cognito::UserPoolClient 资源可创建用户池客户端。 AWS::Cognito::UserPoolGroup 使用 AWS::Cognito::UserPoolGroup 资源可在 Amazon Cognito 用户池中创建用户组。 AWS::Cognito::UserPoolUser 使用 AWS::Cognito::UserPoolUser 资源可创建 Amazon Cognito 用户池用户。 AWS::Cognito::UserPoolUserToGroupAttachment 使用 AWS::Cognito::UserPoolUserToGroupAttachment 资源可将用户附加到 Amazon Cognito 用户池组。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 4 月 28 日	SourceDetails 使用 MaximumExecutionFrequency 资源的 AWS::Config::ConfigRule 子属性可借助周期触发器为自定义规则运行评估。 AWS::EC2::Volume 现在，我们支持在 CloudFormation 中使用 Amazon Elastic Block Store (Amazon EBS) 弹性卷。现在支持对以下三个属性进行 No interruption 更新：VolumeType 、Size 和 Iops。 AWS::EC2::SecurityGroup 使用 GroupName 属性可为 Amazon EC2 安全组指定名称。 AWS::ECS::Service AWS::ECS::Service 新增三个属性：PlacementConstraints 、PlacementStrategies 和 ServiceName 。 AWS::ECS::TaskDefinition 使用 PlacementConstraints 属性可为服务中的任务定义放置约束。 AWS::ElastiCache::ReplicationGroup 向 ConfigurationEndPoint.Address 内置函数添加了 ConfigurationEndPoint.Port 属性和 Fn::GetAtt 属性。 AWS::ElasticLoadBalancingV2::LoadBalancer 使用 IpAddressType 属性可指定负载均衡器的子网使用的 IP 地址的类型。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::EMR::Cluster CloudFormation 现在支持 Amazon EMR 集群的自动扩缩。 AWS::IAM::ManagedPolicy 使用 ManagedPolicyName 属性可为 IAM 托管策略指定自定义名称。 AWS::Lambda::Function 使用 Tags 属性可向 Lambda 函数添加标签。 AWS::OpsWorks::Instance 向 Fn::GetAtt 内置函数添加了以下属性：AvailabilityZone 、PrivateDnsName 、PrivateIp 和 PublicDnsName 。 AWS::OpsWorks::UserProfile 使用 SshUsername 属性可指定用户的 SSH 名称。 向 SshUsername 内置函数添加了 Fn::GetAtt 属性。 AWS::Redshift::Cluster 使用 IamRoles 属性可提供包含一个或多个 Amazon Identity and Access Management 角色（Amazon Redshift 集群使用它们访问其他 Amazon 服务）的列表。	

更改	发行日期	说明	API 版本
使用 Amazon CloudFormation Designer 编辑 YAML 和 JSON 格式的模板	2017 年 4 月 6 日	现在，使用 Designer 创建 CloudFormation 模板时，您可以在集成编辑器中编辑 YAML 和 JSON 格式的模板。此外，您还可以将 JSON 模板转换为 YAML，或进行反向转换 (具体视您首选的模板编写语言而定)。有关更多信息，请参阅什么是 CloudFormation Designer？。	2010-05-15
新资源	2017 年 4 月 6 日	AWS::SSM::Parameter 使用 `AWS::SSM::Parameter` 资源可在 Parameter Store 中创建 SSM 参数。	2010-05-15
`AWS::Include` 转换	2017 年 3 月 28 日	使用 `AWS::Include` 转换可引用在 Amazon S3 存储桶中存储的可重用代码段。有关更多信息，请参阅 AWS::Include 转换。	2010-05-15
将您的 Amazon VPC 与另一个账户对等	2017 年 3 月 28 日	现在，您可以使用 CloudFormation 将您的 Amazon VPC 与其他 Amazon 账户中的 VPC 对等。有关更多信息，请参阅与另一个 Amazon 账户中的 Amazon VPC 对等。	2010-05-15
新资源	2017 年 3 月 28 日	AWS::ApiGateway::UsagePlanKey 使用 `AWS::ApiGateway::UsagePlanKey` 资源可关联使用计划键，并确定使用计划将应用于哪些用户。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 3 月 28 日	AWS::EC2::VPCPeeringConnection 使用 PeerOwnerId 属性和 PeerRoleArn 属性可与其他 Amazon 账户中的 VPC 对等。 有关更多信息，请参阅与另一个 Amazon 账户中的 Amazon VPC 对等。 AWS::IAM::InstanceProfile 使用 InstanceProfileName 属性可配置实例配置文件。 AWS::Lambda::Function 使用 DeadLetterConfig 属性可配置 Amazon Lambda 如何处理其无法处理的事件。 Runtime 属性不再支持 Node.js v0.10。 AWS::Route53::HealthCheck HealthCheckConfig HealthCheckConfig 属性新增了七个资源子类型：AlarmIdentifier 、ChildHealthChecks 、EnableSNI 、HealthThreshold 、InsufficientDataHealthStatus 、Inverted 和 MeasureLatency 。 AWS::SQS::Queue 使用 ContentBasedDeduplication 和 FifoQueue 属性可创建先进先出 (FIFO) Amazon Simple Queue Service 队列。 AWS::S3::Bucket 现在，您可以为 Amazon S3 存储桶指定 IPv6 域名。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2017 年 2 月 10 日	AWS::StepFunctions::Activity 可以使用 AWS::StepFunctions::Activity 资源创建 Amazon Step Functions 活动。 AWS::StepFunctions::StateMachine 使用 AWS::StepFunctions::StateMachine 资源可创建 Step Functions 状态机。	2010-05-15
新的内置函数	2017 年 1 月 17 日	使用 Fn::Split 函数将字符串拆分为字符串值的列表。有关更多信息，请参阅 Fn::Split。	2010-05-15
对列表导入的控制台支持	2017 年 1 月 17 日	使用 CloudFormation 控制台查看正在导入已导出输出值的所有堆栈。有关更多信息，请参阅列出导入已导出输出值的堆栈。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2017 年 1 月 17 日	AWS::AutoScaling::AutoScalingGroup LoadBalancerNames 属性可在无需替换自动扩缩组的情况下更新。 AWS::ECS::TaskDefinition 添加了 NetworkMode 和 MemoryReservation 属性。 AWS::RDS::DBCluster CloudFormation 支持对 Tags 属性的更新。 AWS::RDS::DBInstance 添加了 Timezone 属性。 FirehoseAction 添加了 Separator 属性。 AWS::OpsWorks::Instance 为 PublicIp 内置函数添加了 Fn::GetAtt 属性。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2016 年 12 月 1 日	AWS::CodeBuild::Project 使用 AWS::CodeBuild::Project 资源创建一个 Amazon CodeBuild 项目，用于定义 CodeBuild 构建源代码的方式。 AWS::SSM::Association 使用 AWS::SSM::Association 资源可将 Amazon EC2 Systems Manager 文档与 EC2 实例关联。 AWS::EC2::SubnetCidrBlock 使用 AWS::EC2::SubnetCidrBlock 资源将单个 IPv6 CIDR 块与 Amazon VPC 子网关联。 AWS::EC2::VPCCidrBlock 使用 AWS::EC2::VPCCidrBlock 资源将单个 Amazon 提供的 IPv6 CIDR 块与 Amazon VPC 关联。	2010-05-15

更改	发行日期	说明	API 版本
更新了面向 IPv6 支持的资源	2016 年 12 月 1 日	AWS::EC2::Instance 添加了 Ipv6AddressCount 和 Ipv6Addresses 属性。 AWS::EC2::NetworkAclEntry 添加了 Ipv6CidrBlock 属性。 AWS::EC2::NetworkInterface 添加了 Ipv6AddressCount 和 Ipv6Addresses 属性。 AWS::EC2::Route 添加了 DestinationIpv6CidrBlock 属性。 AWS::EC2::SecurityGroupEgress 添加了 CidrIpv6 属性。 AWS::EC2::SecurityGroupIngress 添加了 CidrIpv6 属性。 AWS::EC2::SpotFleet 添加了启动规范网络接口的 Ipv6AddressCount 和 Ipv6Addresses 属性。 AWS::EC2::Subnet 添加了 Ipv6CidrBlocks 函数的 Fn::GetAtt 属性。 AWS::EC2::VPC 添加了 Ipv6CidrBlocks 函数的 Fn::GetAtt 属性。 AWS::SSM::Document 添加了 DocumentType 属性。	2010-05-15

更改	发行日期	说明	API 版本
资源规范	2016 年 11 月 22 日	使用 CloudFormation 资源规范构建可帮助您创建 CloudFormation 模板的工具。该规范是机器可读的、JSON 格式的文本文件。有关更多信息，请参阅 CloudFormation 资源规范。	2010-05-15
新资源	2016 年 11 月 22 日	AWS::OpsWorks::UserProfile 使用 `AWS::OpsWorks::UserProfile` 资源可为需要访问 Amazon OpsWorks 堆栈中的实例的用户配置 SSH 访问权限。 AWS::OpsWorks::Volume 使用 `AWS::OpsWorks::Volume` 资源可向 Amazon OpsWorks 堆栈注册 Amazon Elastic Block Store 卷。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2016 年 11 月 22 日	AWS::OpsWorks::App 添加了 DataSources 属性。 AWS::OpsWorks::Instance 添加了 BlockDeviceMappings 、AgentVersion 、ElasticIps 、Hostname、Tenancy 和 Volumes 属性。 AWS::OpsWorks::Layer 添加了 CustomJson 和 VolumeConfigurations 属性。 AWS::OpsWorks::Stack 添加了 ElasticIps 、EcsClusterArn 、RdsDbInstances 、CloneAppIds 、ClonePermissions 和 SourceStackId 属性。 AWS::RDS::DBInstance 添加了 CopyTagsToSnapshot 属性。	2010-05-15
列出导入	2016 年 11 月 22 日	列出已导出输出值的导入以跟踪哪些 CloudFormation 堆栈正在导入该值。有关更多信息，请参阅列出导入已导出输出值的堆栈。	2010-05-15
转换	2016 年 11 月 17 日	指定 CloudFormation 用于处理无服务器应用程序的 Amazon SAM 语法的 Amazon Serverless Application Model（Amazon SAM）。有关更多信息，请参阅 转换。	2010-05-15
新资源	2016 年 11 月 17 日	AWS::SNS::Subscription 使用 AWS::SNS::Subscription 资源可让端点订阅 Amazon Simple Notification Service 主题。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2016 年 11 月 17 日	AWS::Lambda::Function 使用 Environment 属性可指定 Amazon Lambda 函数可访问的键值对（环境变量）。 使用 KmsKeyArn 属性指定 Amazon Lambda 用于加密和解密环境变量的 KMS 密钥。	2010-05-15
新的 CLI 命令	2016 年 11 月 17 日	将本地构件上传到 S3 存储桶 使用 package 命令可将 CloudFormation 模板中引用的本地构件上传到 S3 存储桶。 快速部署带转换的模板 使用 deploy 命令将创建和执行更改集操作合并到单个命令中。此命令对快速创建或更新包含转换的堆栈很有用。	2010-05-15
更新的资源	2016 年 11 月 03 日	AWS::CloudFront::Distribution 对于 DistributionConfig 属性，使用 HttpVersion 属性可指定查看器可用来与 Amazon CloudFront 进行通信的最新 HTTP 版本。 对于 ForwardedValues 属性，使用 QueryStringCacheKeys 属性可指定 CloudFront 用于确定要缓存的内容的查询字符串参数。	2010-05-15
列出堆栈导出	2016 年 11 月 03 日	使用 CloudFormation 控制台、API 或 Amazon CLI 可查看区域的所有已导出输出值的列表。有关更多信息，请参阅导出堆栈输出值。	2010-05-15
使用堆栈进行持续交付	2016 年 11 月 03 日	使用 Amazon CodePipeline 可构建用于 CloudFormation 堆栈的持续交付工作流。有关更多信息，请参阅利用 CodePipeline 进行持续交付。	2010-05-15

更改	发行日期	说明	API 版本
在回滚期间跳过资源	2016 年 11 月 03 日	如果您的堆栈处于 UPDATE_ROLLBACK_FAILED 状态，可使用 ContinueUpdateRollback 操作的 ResourcesToSkip 参数跳过 CloudFormation 无法回滚的资源。有关更多信息，请参阅更新回滚失败中的“故障排除”部分。	2010-05-15
更改集增强功能	2016 年 11 月 03 日	您可以使用更改集创建新堆栈。	2010-05-15
更新的资源	2016 年 10 月 12 日	AWS::ElastiCache::CacheCluster 更新 CacheNodeType 属性，而不替换集群。 AWS::ElastiCache::ReplicationGroup 您可创建一个可包含多个节点组（分区）的 Redis（已启用集群模式）复制组，每个节点组中都有一个主集群和多个只读副本。 AWS::ElastiCache::SubnetGroup 使用 CacheSubnetGroupName 属性可为 Amazon ElastiCache 子网组指定名称。	2010-05-15
新资源	2016 年 10 月 06 日	AWS::ApiGateway::UsagePlan 使用 AWS::ApiGateway::UsagePlan 资源为部署的 Amazon API Gateway API 指定使用计划。 AWS::CodeCommit::Repository 使用 AWS::CodeCommit::Repository 资源创建由 Amazon Web Services 托管的 CodeCommit 存储库。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2016 年 10 月 06 日	AWS::ApiGateway::Authorizer 使用 ProviderARNs 属性可将 Amazon Cognito 用户池用作 Amazon API Gateway API 授权方。 AWS::ApiGateway::Deployment 不再需要 StageName 属性。 AWS::ElasticLoadBalancingV2::TargetGroup 对于 GetAtt 函数，请使用 LoadBalancerArns 属性检索将流量路由至目标组的负载均衡器的 Amazon 资源名称（ARN）。 AWS::RDS::DBInstance 使用 Domain 和 DomainIAMRoleName 属性在用户连接到 RDS 数据库实例时使用 Windows 身份验证。 AWS::EC2::SecurityGroupEgress 使用 DestinationPrefixListId 属性可指定 Amazon VPC 端点的 Amazon 服务前缀。	2010-05-15
跨堆栈引用增强	2016 年 10 月 06 日	使用内部函数来自定义导出的 Name 值或引用 ImportValue 函数中的值。	2010-05-15
CloudFormation 服务角色	2016 年 9 月 26 日	使用 Amazon Identity and Access Management（IAM）服务角色进行 CloudFormation 堆栈操作。CloudFormation 使用角色的凭证代表您调用堆栈资源。有关更多信息，请参阅 Amazon CloudFormation 服务角色。	2010-05-15
新功能	2016 年 9 月 19 日	您可使用 Export 输出字段和 Fn::ImportValue 内部函数让一个堆栈引用另一个堆栈中的资源输出。有关更多信息，请参阅 Outputs、Fn::ImportValue 和演练：引用其他 CloudFormation 堆栈中的资源输出。	2010-05-15

更改	发行日期	说明	API 版本
YAML 支持	2016 年 9 月 19 日	您可以使用 YAML 格式创作 CloudFormation 模板。例如，YAML 还可让您将注释添加到模板或使用内部函数的简短形式。有关更多信息，请参阅 CloudFormation 模板格式。	2010-05-15
新的内置函数	2016 年 9 月 19 日	使用 `Fn::Sub` 函数可将输入字符串中的变量替换为您指定的值。有关更多信息，请参阅 Fn::Sub。	2010-05-15
新资源	2016 年 9 月 19 日	AWS::KMS::Alias 使用 `AWS::KMS::Alias` 资源为 Amazon KMS key 创建别名。	
更新的资源	2016 年 9 月 19 日	AWS::EC2::SpotFleet 对于 `LaunchSpecifications` 属性，使用 `SpotPrice` 属性为特定实例类型指定出价。 AWS::ECS::Cluster 使用 `ClusterName` 属性可为 Amazon Elastic Container Service 集群指定名称。 AWS::ECS::TaskDefinition 使用 `TaskRoleArn` 属性可指定 Amazon Elastic Container Service 容器用来代表您发出 Amazon 调用的 Amazon Identity and Access Management 角色。 使用 `Family` 属性将任务定义注册到特定系列。 AWS::Elasticsearch::Domain 使用 `ElasticsearchVersion` 属性指定要使用哪个版本的 Elasticsearch。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2016 年 8 月 11 日	使用以下 Elastic Load Balancing 应用程序负载均衡器 资源可向多个可用区中的多个目标（如 EC2 实例）分发传入的应用程序流量。 • AWS::ElasticLoadBalancingV2::Listener • AWS::ElasticLoadBalancingV2::ListenerRule • AWS::ElasticLoadBalancingV2::LoadBalancer • AWS::ElasticLoadBalancingV2::TargetGroup	2010-05-15
更新的资源	2016 年 8 月 11 日	AWS::AutoScaling::AutoScalingGroup 使用 TargetGroupARNs 属性可将自动扩缩组与一个或多个 应用程序负载均衡器 目标组关联。 AWS::ECS::Service 对于负载 LoadBalancers 属性，使用 TargetGroupArn 属性可将 Amazon Elastic Container Service 服务与 应用程序负载均衡器 目标组关联。	2010-05-15
新资源	2016 年 8 月 09 日	CloudFormation 添加了以下资源： AWS::ApplicationAutoScaling::ScalableTarget 和 AWS::ApplicationAutoScaling::ScalingPolicy 使用 Application Auto Scaling 扩展策略定义扩展目标资源的时间和方式。 AWS::CertificateManager::Certificate 预置 Amazon Certificate Manager 证书，您可将此证书与其他 Amazon 服务结合使用来启用安全连接。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2016 年 8 月 09 日	CloudFormation 更新了以下资源： AWS::CloudFront::Distribution 对于分发配置 `ViewerCertificate` 属性，您可以指定 Amazon Certificate Manager 证书。对于分发配置 `Origin` 属性，您可以指定自定义源的自定义标头和 SSL 协议。 AWS::EFS::FileSystem 您可以指定 Amazon Elastic File System 文件系统的性能模式。	2010-05-15
新资源	2016 年 7 月 20 日	Amazon IoT 使用 Amazon IoT 可声明 Amazon IoT 策略、X.509 证书、策略与委托人（X.509 证书或其他凭证）之间的关联、Amazon IoT 事务、委托人与事务之间的关联或 Amazon IoT 规则。 • AWS::IoT::Certificate • AWS::IoT::Policy • AWS::IoT::PolicyPrincipalAttachment • AWS::IoT::Thing • AWS::IoT::ThingPrincipalAttachment • AWS::IoT::TopicRule	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2016 年 7 月 20 日	CloudFormation 更新了以下资源： AWS::IAM::Group, AWS::IAM::Role, AWS::IAM::User 使用名称属性可为 Amazon Identity and Access Management（IAM）资源指定自定义名称。 AWS::ApiGateway::Method 对于 `Integration` 属性，您可以使用 `PassthroughBehavior` 属性指定 Amazon API Gateway 将请求传递到目标后端的时间。 AWS::ApiGateway::Model 和 AWS::ApiGateway::RestApi 您可以为 `Schema` 和 `Body` 属性指定 JSON 对象。	2010-05-15
自动扩缩组 UpdatePolicy	2016 年 6 月 9 日	对于 `UpdatePolicy` 属性，使用 `AutoScalingReplacingUpdate` 属性可指定在您更新自动扩缩组时是否替换自动扩缩组和其中包含的实例。在替换期间，CloudFormation 保留旧自动扩缩组，直至成功创建新组；这样，如果更新失败，CloudFormation 可以回滚到旧自动扩缩组。有关更多信息，请参阅 UpdatePolicy 属性。	2010-05-15
新资源	2016 年 6 月 9 日	CloudFormation 添加了以下资源： AWS::EC2::FlowLog 创建捕获指定网络接口、子网或 VPC 的 IP 流量的 Amazon Elastic Compute Cloud 流日志。 AWS::KinesisFirehose::DeliveryStream 创建向目标提供实时流数据的传输流，例如 Amazon Simple Storage Service、Amazon Redshift 或 Amazon OpenSearch Service。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2016 年 6 月 9 日	CloudFormation 更新了以下资源： AWS::Kinesis::Stream 使用 Name 属性可为 Amazon Kinesis 流指定名称。 AWS::Lambda::Function 对于 Code 属性，您可以为 ZipFile 运行时环境使用 nodejs4.3 属性和 cfn 响应模块。 AWS::SNS::Topic CloudFormation 为 Amazon Simple Notification Service 主题资源启用了更新。	2010-05-15
新资源	2016 年 4 月 25 日	使用 AWS::EC2::Host 资源可分配完全专用的物理服务器以启动 EC2 实例。	2010-05-15

更改	发行日期	说明	API 版本
更新的资源	2016 年 4 月 25 日	AWS::EC2::Instance 使用 Affinity 和 HostId 属性可在 Amazon Elastic Compute Cloud 专用主机上启动实例。 AWS::ECS::Service 使用 DeploymentConfiguration 属性配置在部署期间可以运行多少个任务。 AWS::ECS::TaskDefinition CloudFormation 增加了对额外 Amazon Elastic Container Service 容器定义属性的支持。 AWS::GameLift::Fleet 使用 MaxSize 和 MinSize 属性指定您的 Amazon GameLift Servers 实例集中允许的 EC2 实例的最大数量和最小数量。 AWS::Lambda::Function 使用 FunctionName 属性可为您的 Amazon Lambda 函数指定名称。您还可以使用 Python 2.7 指定内联函数。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2016 年 4 月 18 日	Amazon API Gateway 使用 Amazon API Gateway 资源发布、维护和监视任意规模的 API。您可以创建客户端可调用的 API 用于访问您的后端服务，例如运行 EC2 实例的应用程序或运行在 Amazon Lambda 上的代码。 • AWS::ApiGateway::Account • AWS::ApiGateway::ApiKey • AWS::ApiGateway::Authorizer • AWS::ApiGateway::BasePathMapping • AWS::ApiGateway::ClientCertificate • AWS::ApiGateway::Deployment • AWS::ApiGateway::Method • AWS::ApiGateway::Model • AWS::ApiGateway::Resource • AWS::ApiGateway::RestApi • AWS::ApiGateway::Stage AWS::Events::Rule 创建监视对您的账户中 Amazon 资源更改（事件）的 Amazon CloudWatch Events 规则。如果传入事件与您在规则中描述的事件匹配，Amazon CloudWatch Events 会发送消息到并激活您指定的目标，例如 Amazon Lambda 函数或 Amazon Simple Notification Service 主题。 AWS::WAF::SizeConstraintSet 和 AWS::WAF::XssMatchSet 使用两个 Amazon WAF 规则检查 Web 请求的大小或者防止跨站点脚本攻击。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2016 年 3 月 31 日	使用 AWS::Lambda::Alias 资源可为您的 Amazon Lambda 函数创建别名，使用 AWS::Lambda::Version 资源可创建您的函数的版本。	2010-05-15
更新的资源	2016 年 3 月 31 日	CloudFormation 更新了以下资源： AWS::EMR::Cluster 和 AWS::EMR::InstanceGroupConfig 使用 `EbsConfiguration` 属性可以为您的 Amazon EMR 集群或实例组配置 Amazon Elastic Block Store 存储卷。 AWS::Lambda::Function 使用 `VpcConfig` 属性可允许 Amazon Lambda 函数访问 VPC 中的资源。 AWS::S3::Bucket 对于 Amazon Simple Storage Service 生命周期规则，您可指定多个转换规则，这些规则指定何时将对象转换为指定存储类。	2010-05-15
更改集	2016 年 3 月 29 日	更新堆栈之前，使用更改集查看您的更改可能如何影响正在运行的资源。有关更多信息，请参阅使用更改集更新堆栈。	2010-05-15
新资源	2016 年 15 月 3 日	使用 AWS::GameLift::Alias、AWS::GameLift::Build 和 AWS::GameLift::Fleet 资源在 Amazon 中部署多人游戏服务器。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2016 年 2 月 26 日	CloudFormation 添加了以下资源： AWS::ECR::Repository 创建用户可以“推”和“拉”Docker 映像的 Amazon Elastic Container Registry 存储库。 AWS::EC2::NatGateway 使用网络地址转换器 (NAT) 网关在私有子网中启用 EC2 实例，以连接到 Internet。 AWS::Elasticsearch::Domain 创建运行旧版 Elasticsearch OSS 集群的 Amazon OpenSearch Service 域。 AWS::EMR::Cluster, AWS::EMR::InstanceGroupConfig, AWS::EMR::Step 使用 Amazon EMR 资源帮您分析和处理大量数据。您可以创建集群，然后在这些集群上运行任务。	2010-05-15
更新的资源	2016 年 2 月 26 日	CloudFormation 更新了以下资源： AWS::CloudTrail::Trail 使用 `IsMultiRegionTrail` 属性指定是在创建堆栈的区域还是所有区域中创建 Amazon CloudTrail 跟踪。 AWS::Config::ConfigurationRecorder 对于记录组，可以使用 `IncludeGlobalResourceTypes` 属性记录所有全局资源类型。 AWS::RDS::DBCluster 使用 `KmsKeyId` 和 `StorageEncrypted` 属性加密集群中的数据库实例。	2010-05-15

更改	发行日期	说明	API 版本
保留资源	2016 年 2 月 26 日	对于处于 DELETE_FAILED 状态的堆栈，可以使用 RetainResources 参数保留 CloudFormation 无法删除的资源。有关更多信息，请参阅删除堆栈失败。	2010-05-15
更新堆栈标签	2016 年 2 月 26 日	在更新堆栈时，您可以添加、修改或删除堆栈标签。有关更多信息，请参阅 CloudFormation 堆栈更新。	2010-05-15
继续回滚失败的更新回滚操作	2016 年 1 月 25 日	对于处于 UPDATE_ROLLBACK_FAILED 状态的堆栈，您可以继续回滚更新，以使堆栈进入正常工作状态。这样，您可以使堆栈恢复到其初始设置，然后尝试再次更新。有关更多信息，请参阅继续回滚更新。	2010-05-15
可用于亚太 (首尔) 区域的新示例模板。	2016 年 1 月 7 日	以下 CloudFormation 示例模板集合适用于 ap-northeast-2 区域： • 示例解决方案 • 应用程序框架 • Services 有关更多信息，请参阅使用 CloudFormation 模板。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2015 年 12 月 28 日	CloudFormation 添加了以下资源： AWS::DirectoryService::MicrosoftAD 使用 Microsoft Active Directory 资源在 Amazon 中创建 Microsoft Active Directory 目录。 AWS::Logs::Destination 和 AWS::Logs::LogStream 使用 Amazon CloudWatch Logs 资源创建用于实时处理日志数据的目标或分别创建日志流。 AWS::WAF::ByteMatchSet、AWS::WAF::IPSetAWS::WAF::Rule、AWS::WAF::SqlInjectionMatchSet、和 AWS::WAF::WebACL 使用 Amazon WAF 资源控制和监视对您的内容的 Web 请求。	2010-05-15

更改	发行日期	说明	API 版本
资源更新	2015 年 12 月 28 日	CloudFormation 更新了以下资源： AWS::CloudFront::Distribution 对于分发配置，使用 `WebACLId` 属性将 Amazon WAF Web 访问控制列表（ACL）关联到 Amazon CloudFront 分发。对于缓存行为和默认缓存行为，您可以指定生存时间 (TTL) 的默认值和最大值。 AWS::DynamoDB::Table 您可以创建、更新或删除全局二级索引，而不必替换 Amazon DynamoDB 表。 AWS::S3::Bucket 使用 `ReplicationConfiguration` 属性指定复制哪些对象以及将它们存储在何处。 使用 `NotificationConfiguration` 属性中的属性指定筛选条件，以便 Amazon Simple Storage Service 为您指定的对象发送通知。	2010-05-15
参数分组和排序	2015 年 12 月 3 日	当用户使用您的模板创建或更新堆栈时，使用 AWS::CloudFormation::Interface 元数据键可在 CloudFormation 控制台中对参数进行分组和排序。	2010-05-15
更新策略属性	2015 年 12 月 3 日	对于 Auto Scaling 更新策略属性，使用 `MinSuccessfulInstancesPercent` 属性可指定必须有多少实例（百分比）发送成功信号才认为更新成功。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2015 年 12 月 3 日	CloudFormation 添加了以下资源： AWS::CodePipeline::Pipeline 和 AWS::CodePipeline::CustomActionType 使用 CodePipeline 资源创建一个管道，用于描述软件在发布过程中的变更。 AWS::Config::ConfigurationRecorder、AWS::Config::DeliveryChannel 和 AWS::Config::ConfigRule 使用 Amazon Config 资源可监控特定 Amazon 资源的配置更改。 AWS::KMS::Key 使用 Amazon Key Management Service（Amazon KMS）资源在 Amazon KMS 中创建可供用户加密少量数据的客户托管密钥。 AWS::SSM::Document 使用 Amazon EC2 Systems Manager 创建用于指定特定于实例的配置的文档。	2010-05-15

更改	发行日期	说明	API 版本
资源更新	2015 年 12 月 3 日	CloudFormation 更新了以下资源： AWS::AutoScaling::LaunchConfiguration 指定 EBS 卷是否加密。 AWS::AutoScaling::ScalingPolicy 您可以使用两种不同的策略类型（简单扩展和步进扩展）来指定出现 Amazon CloudWatch (CloudWatch) 警报违例时自动扩缩组如何扩展。 AWS::CloudTrail::Trail 使用 CloudWatch 属性向 CloudWatch 日志组发送日志。您可以向跟踪添加标签并指定用于加密日志的 Amazon KMS key。 AWS::CodeDeploy::Application、AWS::CodeDeploy::DeploymentConfig 和 AWS::CodeDeploy::DeploymentGroup 使用 ApplicationName 、DeploymentConfigName 和 DeploymentGroupName 属性为 CodeDeploy 资源指定自定义名称。 AWS::DynamoDB::Table 使用 StreamSpecification 属性指定用于捕获对存储在 Amazon DynamoDB (DynamoDB) 表中的项目所做的更改的设置。 AWS::EC2::Instance 使用 SsmAssociations 属性将 Amazon EC2 Systems Manager 文档关联到实例。 AWS::EC2::SpotFleet 使用 AllocationStrategy 属性指定如何跨 Spot 池分配目标容量。使用 ExcessCapacityTerm	2010-05-15

更改	发行日期	说明	API 版本
		`inationPolicy` 属性指定当目标容量低于 Spot 队列的大小时如何终止实例。 AWS::Redshift::Cluster 使用 `KmsKeyId` 属性指定用于加密 Amazon Redshift 集群中数据的 Amazon KMS key。 AWS::WorkSpaces::Workspace 使用加密属性加密存储在卷上的数据。	
资源更新	2015 年 11 月 4 日	对于 AWS::EC2::Volume 资源，如果某个卷的数据变得不一致，可以使用 `AutoEnableIO` 属性自动恢复 I/O 操作。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2015 年 10 月 1 日	CloudFormation 添加了以下资源： AWS::CodeDeploy::Application、AWS::CodeDeploy::DeploymentGroup 和 AWS::CodeDeploy::DeploymentConfig 使用 CodeDeploy 资源创建部署并将之应用到 EC2 或本地实例。 AWS::DirectoryService::SimpleAD 使用 Simple Active Directory 资源创建 Amazon Directory Service Simple AD（兼容 Microsoft Active Directory 的目录）。 AWS::EC2::PlacementGroup 使用置放群组在低延迟网络中创建实例集群。 AWS::EC2::SpotFleet 使用 Spot 队列启动一个运行可中断任务的 Spot 实例集合。 AWS::Lambda::EventSourceMapping 使用事件源映射资源指定一个流作为 Amazon Lambda（Lambda）函数的事件源。 AWS::Lambda::Permission 使用 Lambda 权限向 Lambda 函数的策略添加语句。 AWS::Logs::SubscriptionFilter 使用订阅筛选条件定义将哪些日志事件交付到 Kinesis 流。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::RDS::DBCluster 和 AWS::RDS::DBClusterParameterGroup 使用集群和集群参数组资源创建 Amazon Aurora 数据库集群。 AWS::WorkSpaces::Workspace 使用 WorkSpaces 创建基于云的桌面体验。	
资源更新	2015 年 10 月 1 日	CloudFormation 更新了以下资源： AWS::ElastiCache::ReplicationGroup 使用 `Fn::GetAtt` 内置函数获取只读副本地址和端口的列表。 AWS::OpsWorks::Stack 使用 `AgentVersion` 属性指定特定的 Amazon OpsWorks 代理。 AWS::OpsWorks::App 使用 `Environment` 属性为 Amazon OpsWorks 应用指定环境变量。 AWS::S3::Bucket 对于 NotificationConfiguration 属性，您可以为 Lambda 函数和 Amazon Simple Queue Service (Amazon SQS) 队列配置通知设置。	2010-05-15
IAM 条件键	2015 年 10 月 1 日	对于 Amazon Identity and Access Management（IAM）策略，可以使用 CloudFormation 特定的条件键指定 IAM 策略何时生效。有关更多信息，请参阅使用 Amazon Identity and Access Management 控制访问权限。	2010-05-15

更改	发行日期	说明	API 版本
Amazon CloudFormation Designer	2015 年 10 月 1 日	使用 Amazon CloudFormation Designer 可通过拖放界面来创建和修改模板。	2010-05-15
新资源	2015 年 8 月 24 日	使用 AWS::EC2::VPCEndpoint 资源在您的 VPC 与其他 Amazon 服务之间建立私有连接。	2010-05-15
资源更新	2015 年 8 月 24 日	CloudFormation 更新了以下资源： AWS::ElasticBeanstalk::Environment 使用 Tags 属性为 Amazon Elastic Beanstalk（Elastic Beanstalk）环境指定标签（键值对）。 AWS::Lambda::Function 对于 Code 属性，使用 ZipFile 属性可将您的 Lambda 函数的源代码直接写入到模板中。目前，只能针对 ZipFile 运行时环境使用 nodejs 属性。但您仍可以为所有运行时环境 (如 java8 和 nodejs) 指向 S3 存储桶中的文件。 AWS::OpsWorks::Instance 使用 EbsOptimized 属性指示实例是否针对 Amazon Elastic Block Store (Amazon EBS) I/O 进行了优化。 AWS::RDS::DBInstance 对于 SourceDBInstanceIdentifier 属性，您可以指定位于其他区域中的数据库实例，以创建跨区域的只读副本。	2010-05-15

更改	发行日期	说明	API 版本
Amazon S3模板 URL	2015 年 8 月 24 日	对于启用了版本控制的存储桶，可以在创建或更新堆栈时在 Amazon S3 模板 URL 中指定版本 ID，例如：`https://s3.amazonaws.com/templates/myTemplate.template?versionId=123ab1cdeKdOW5IH4GAcYbEngcpTJTDW` 。	2010-05-15
新资源	2015 年 8 月 3 日	使用 AWS::EFS::FileSystem 资源创建 Amazon Elastic File System（Amazon EFS）文件系统和 AWS::EFS::MountTarget 资源以创建文件系统的挂载点。	2010-05-15
权限要求更改	2015 年 6 月 11 日	创建或更新 AWS::RDS::DBInstance 资源后，您现在还必须具有调用 `ec2:DescribeAccountAttributes` 操作的权限。	2010-05-15

更改	发行日期	说明	API 版本
新资源	2015 年 6 月 11 日	CloudFormation 添加了以下资源： AWS::DataPipeline::Pipeline 使用数据管道自动执行数据的移动与转换。 Amazon Elastic Container Service 资源 使用 AWS::ECS::Service、AWS::ECS::Cluster 和 AWS::ECS::TaskDefinition 资源在 EC2 实例的集群上创建 Docker 容器。 AWS::ElastiCache::ReplicationGroup 使用复制组创建节点的集合，其中包含一个主读/写集群和最多五个辅助只读集群。 AWS::IAM::ManagedPolicy 使用托管策略在您的 Amazon 账户中创建策略，可使用这些策略来对 IAM 用户、组和角色应用权限。 AWS::Lambda::Function 使用 Lambda 函数运行代码以响应事件。 AWS::RDS::OptionGroup 使用选项组帮助您创建和管理 Amazon Relational Database Service (Amazon RDS) 数据库。	2010-05-15

更改	发行日期	说明	API 版本
资源更新	2015 年 6 月 11 日	CloudFormation 更新了以下资源： AWS::EC2::Subnet 使用 MapPublicIpOnLaunch 属性自动向子网中的实例分配公有 IP 地址。 AWS::ElastiCache::CacheCluster 使用 SnapshotName 属性将快照数据还原到新的 Redis 缓存集群。 AWS::IAM::User 对于 LoginProfile 属性，请使用 PasswordResetRequired 属性，以便在用户登录 Amazon Web Services 管理控制台时要求其设置新密码。 AWS::OpsWorks::Layer 使用 LifecycleEventConfiguration 属性为 Amazon OpsWorks 层配置生命周期事件。 AWS::S3::Bucket 对于 LifecycleConfiguration 属性，请使用 NoncurrentVersionExpirationInDays 和 NoncurrentVersionTransition 属性为非当前对象版本指定生命周期规则。	2010-05-15

更改	发行日期	说明	API 版本
新的参数类型	2015 年 5 月 19 日	在使用 CloudFormation 控制台创建或更新堆栈时，可按 ID、名称或名称标签值搜索 Amazon 特定的参数类型值。 CloudFormation 还添加了对以下 Amazon 特定参数类型的支持。有关更多信息，请参阅参数。 • AWS::EC2::AvailabilityZone::Name • List<AWS::EC2::AvailabilityZone ::Name> • AWS::EC2::Instance::Id • List<AWS::EC2::Instance ::Id> • AWS::EC2::Image::Id • List<AWS::EC2::Image::Id >> • AWS::EC2::SecurityGroup::GroupName • List<AWS::EC2::SecurityGroup::GroupName > • AWS::EC2::Volume::Id • List<AWS::EC2::Volume::Id >> • AWS::Route53::HostedZone::Id • List<AWS::Route53::HostedZone ::Id>	2010-05-15
新资源	2015 年 4 月 16 日	CloudFormation 添加了以下资源： AWS::AutoScaling::LifecycleHook 启动或终止实例后，使用 Auto Scaling 生命周期钩子可控制实例的状态。 AWS::RDS::EventSubscription 使用事件订阅获取有关 Amazon RDS 事件的通知。	2010-05-15

更改	发行日期	说明	API 版本
资源更新	2015 年 4 月 16 日	CloudFormation 更新了以下资源： AWS::AutoScaling::AutoScalingGroup 使用 NotificationConfigurations 属性可指定多个通知。 AWS::AutoScaling::LaunchConfiguration 使用 PlacementTenancy 属性可指定实例的租期。 使用 ClassicLinkVPCId 和 ClassicLinkVPCSecurityGroups 属性可将 EC2-Classic 实例链接到支持 ClassicLink 的 VPC。 AWS::AutoScaling::ScalingPolicy 使用 MinAdjustmentStep 属性可指定在扩展事件期间添加或删除的最小实例数。 AWS::CloudFront::Distribution 对于查看器证书，请使用 MinimumProtocolVersion 属性指定最低协议版本。对于缓存行为，请使用 CachedMethods 属性指定 Amazon CloudFront (CloudFront) 为哪些方法缓存响应。对于源，请使用 OriginPath 指定 CloudFront 用于请求内容的路径。 AWS::ElastiCache::CacheCluster 对于 Memcached 缓存集群，请使用 AZMode 和 PreferredAvailabilityZones 属性指定多个可用区 (AZ) 中的节点。 AWS::EC2::Volume 使用 KmsKeyId 属性可指定加密卷的客户托管密钥。	2010-05-15

更改	发行日期	说明	API 版本
		AWS::OpsWorks::Instance 使用 TimeBasedAutoScaling 属性，可基于指定的计划自动扩展实例。 AWS::OpsWorks::Layer 使用 LoadBasedAutoScaling 属性可指定基于负载的扩展策略。对于卷配置，使用 VolumeType 和 Iops 属性可分别指定卷类型和每秒 I/O 操作数。 AWS::RDS::DBInstance 使用 CharacterSetName 属性可指定所支持的数据库引擎的字符集。 使用 StorageEncrypted 属性可指示是否对数据库实例加密，使用 KmsKeyId 可指定加密数据库实例的客户托管密钥。 AWS::Route53::HealthCheck 使用 HealthCheckTags 属性可将标签与运行状况检查关联。 AWS::Route53::HostedZone 使用 VPCs 属性可创建私有托管区域。 使用 HostedZoneTags 属性可将标签与托管区域关联。	
新的模板部分	2015 年 4 月 16 日	向模板添加 Metadata 部分，以包括用于描述模板的任意 JSON 对象，如设计或实施详细信息。	2010-05-15
资源更新	2015 年 4 月 8 日	对于 AWS::CloudFormation::CustomResource 资源，您可以在 ServiceToken 属性中指定 Lambda 函数的 Amazon 资源名称（ARN）。	2010-05-15

更改	发行日期	说明	API 版本
Amazon RDS 更新	2014 年 12 月 24 日	CloudFormation 为 RDS 数据库实例添加了两个新属性。您可将选项组与数据库实例关联并指定数据库实例存储类型。有关更多信息，请参阅 AWS::RDS::DBInstance。	2010-05-15
Elastic Load Balancing 更新	2014 年 12 月 24 日	您可使用 `ConnectionSettings` 属性指定连接可保持闲置状态的时长。有关更多信息，请参阅 AWS::ElasticLoadBalancing::LoadBalancer。	2010-05-15
Route 53 更新	2014 年 11 月 6 日	现在，您可以配置和管理 Route 53 托管区域、运行状况检查、失效转移记录集和地理位置记录集。	2010-05-15
Auto Scaling 滚动更新增强功能	2014 年 11 月 6 日	在更新期间，您可使用 `WaitOnResourceSignals` 标志来指示 CloudFormation 等待实例发送成功信号。这样一来，CloudFormation 将不会更新下一批实例，直到当前批次准备就绪。有关更多信息，请参阅 UpdatePolicy 属性。	2010-05-15
新的 VPC Fn:GetAtt 属性	2014 年 11 月 6 日	在给定 VPC ID 的情况，您可以检索该 VPC 的默认安全组和网络 ACL。有关更多信息，请参阅 Fn::GetAtt。	2010-05-15
新 Amazon 特定的参数类型	2014 年 11 月 6 日	您可以在 CloudFormation 模板中指定 Amazon 特定的参数类型。在 CloudFormation 控制台中，这些参数类型提供了一个有效值的下拉列表。利用 API 或 Amazon CLI，CloudFormation 可在创建或更新堆栈之前快速验证这些参数类型的值。有关更多信息，请参阅参数。	2010-05-15
CreationPolicy 属性	2014 年 11 月 6 日	利用 CreationPolicy 属性，您可以指示 CloudFormation 等待，直到 EC2 实例上的应用程序准备就绪，然后再继续堆栈创建。您可使用创建策略而不是等待条件或等待条件句柄。有关更多信息，请参阅 CreationPolicy 属性。	2010-05-15
Amazon CloudFront 转发的值	2014 年 9 月 29 日	对于缓存行为，您可将标头转发到源。请参阅 Forwarded Values。	2010-05-15

更改	发行日期	说明	API 版本
Amazon OpsWorks 更新	2014 年 9 月 29 日	对于 Chef 11.10，您可使用 `ChefConfiguration` 属性来启用 Berkshelf。您还可将 Amazon OpsWorks 内置安全组与 Amazon OpsWorks 堆栈结合使用。有关更多信息，请参阅 AWS::OpsWorks::Stack。	2010-05-15
Elastic Load Balancing 加标签支持	2014 年 9 月 29 日	Amazon CloudFormation 为 Elastic Load Balancing 负载均衡器添加堆栈级标签。您还可将自己的标签添加到负载均衡器。请参阅AWS::ElasticLoadBalancing::LoadBalancer。	2010-05-15
Amazon Simple Notification Service 主题策略更新	2014 年 9 月 29 日	您现在可更新 Amazon SNS 主题策略。有关更多信息，请参阅 AWS::SNS::TopicPolicy。	2010-05-15
RDS 数据库实例更新	2014 年 9 月 5 日	您可通过使用 AWS::RDS::DBInstance 资源中的 `PubliclyAccessible` 属性来指定数据库实例是否面向 Internet。	2010-05-15
UpdatePolicy 属性更新	2014 年 9 月 05 日	您可为具有关联的计划操作的自动扩缩组指定更新策略。有关更多信息，请参阅 UpdatePolicy 属性。	2010-05-15
Amazon CloudWatch 支持	2014 年 7 月 10 日	您可以使用 CloudFormation 预置和管理 Amazon CloudWatch Logs（CloudWatch Logs）日志组和指标筛选条件。有关更多信息，请参阅 AWS::Logs::LogGroup 或 AWS::Logs::MetricFilter。	2010-05-15

更改	发行日期	说明	API 版本
Amazon CloudFront 分发配置更新	2014 年 6 月 17 日	您可指定其他 CloudFront 分发配置属性： • 自定义错误响应定义了 4xx 和 5xx HTTP 状态代码的自定义错误消息。 • 价格级别定义了您想为 CloudFront 服务支付的最高价。 • 限制定义了可查看您的内容的人员。 • 查看器证书指定查看器使用 HTTP 时要使用的证书。 • 对于缓存行为，您可以指定允许的 HTTP 方法并指示是否转发 Cookie。 有关更多信息，请参阅 AWS::CloudFront::Distribution。	2010-05-15
EC2 实例更新	2014 年 6 月 17 日	您可指定当您调用实例的操作系统关闭命令时，实例是停止还是终止。有关更多信息，请参阅 AWS::EC2::Instance。	2010-05-15
EBS 卷更新	2014 年 6 月 17 日	您可以将加密的 EBS 卷与受支持的实例类型结合使用。有关更多信息，请参阅 AWS::EC2::Volume。	2010-05-15
新的 Amazon VPC 对等连接	2014 年 6 月 17 日	您可使用 CloudFormation 创建 Amazon Virtual Private Cloud（Amazon VPC）对等连接，这将在两个 VPC 之间建立网络连接。有关更多信息，请参阅 AWS::EC2::VPCPeeringConnection。	2010-05-15
Amazon EC2 Auto Scaling 组更新	2014 年 6 月 17 日	您可指定一个现有的集群置放群组，可在该群组中启动 Amazon EC2 Auto Scaling 组的实例。有关更多信息，请参阅 AWS::AutoScaling::AutoScalingGroup。	2010-05-15
Amazon CloudTrail 支持	2014 年 6 月 17 日	CloudFormation 支持 Amazon CloudTrail，它可捕获从您的 Amazon 账户发起的 API 调用，并在您指定的位置发布日志。有关更多信息，请参阅 AWS::CloudTrail::Trail。	2010-05-15

更改	发行日期	说明	API 版本
更新堆栈增强功能	2014 年 5 月 12 日	CloudFormation 支持更多堆栈更新功能： • 您可以更新 CloudFormation 堆栈参数，无需重新提交堆栈的模板。 • 您可以添加或删除 CloudFormation 堆栈的 Amazon SNS 通知主题。 有关更多信息，请参阅 CloudFormation 堆栈更新。	2010-05-15
Amazon Kinesis 支持	2014 年 5 月 6 日	您可以使用 CloudFormation 创建可从数据来源捕获并传输数据记录的 Amazon Kinesis 流。有关更多信息，请参阅 AWS::Kinesis::Stream。	2010-05-15
新的 S3 存储桶属性	2014 年 5 月 5 日	CloudFormation 支持其他 S3 存储桶属性： • 跨源资源共享 (CORS) 定义存储桶中对象的跨源资源共享。 • 生命周期定义 Amazon S3 如何在对象的生命周期内管理对象。 • 访问日志记录策略捕获有关对存储桶进行的请求的信息。 • 通知定义要报告的事件以及要将消息发送到的 Amazon SNS 主题。 • 版本控制允许存储桶中的所有对象存在多个变体。 • 重定向和路由规则对存储桶网站端点请求的重定向行为进行管理。 有关更多信息，请参阅 AWS::S3::Bucket。	2010-05-15
Amazon EC2 Auto Scaling 支持	2014 年 5 月 5 日	CloudFormation 支持自动扩缩组的指标集合。有关更多信息，请参阅 AWS::AutoScaling::AutoScalingGroup。	2010-05-15

更改	发行日期	说明	API 版本
Fn::If 更新	2014 年 5 月 5 日	您可以使用模板输出部分中的 Fn::If 内置函数。有关更多信息，请参阅条件函数。	2010-05-15
使用 Amazon CloudTrail 进行 API 日志记录	2014 年 4 月 2 日	您可以使用 Amazon CloudTrail（CloudTrail）来记录 CloudFormation 请求。借助 CloudTrail，您可以获取账户的 CloudFormation API 调用历史记录。有关更多信息，请参阅使用 Amazon CloudTrail 记录 CloudFormation API 调用。	2010-05-15
Elastic Load Balancing 更新	2014 年 3 月 20 日	您可以指定访问日志记录策略，以捕获有关对负载均衡器进行的请求的信息。您还可以指定连接耗尽策略，用于描述在实例取消注册或运行状况不佳时如何处理动态请求。有关更多信息，请参阅 AWS::ElasticLoadBalancing::LoadBalancer。	2010-05-15
Amazon OpsWorks 支持	2014 年 3 月 3 日	您可以使用 CloudFormation 配置和管理 Amazon OpsWorks 堆栈。有关更多信息，请参阅 AWS::OpsWorks::Stack 或 Amazon OpsWorks 模板代码段。	2010-05-15
Amazon S3 模板大小限制提高	2014 年 2 月 18 日	您可以在 Amazon S3 中指定最大为 460800 字节的模板大小。	2010-05-15
Amazon Redshift 支持	2014 年 2 月 10 日	您可以使用 CloudFormation 预置和管理 Amazon Redshift 集群。有关更多信息，请参阅 Amazon Redshift 模板代码段或 AWS::Redshift::Cluster。	2010-05-15
S3 存储桶和存储桶策略更新	2014 年 2 月 10 日	您可以更新 S3 存储桶和存储桶策略资源的一些属性。有关更多信息，请参阅 AWS::S3::Bucket 或 AWS::S3::BucketPolicy。	2010-05-15
Elastic Beanstalk 环境和应用程序版本更新	2014 年 2 月 10 日	您可以更新 Elastic Beanstalk 环境配置和应用程序版本。有关更多信息，请参阅 AWS::ElasticBeanstalk::Environment、AWS::ElasticBeanstalk::ConfigurationTemplate或AWS::ElasticBeanstalk::ApplicationVersion。	2010-05-15

更改	发行日期	说明	API 版本
Amazon SQS 更新	2014 年 1 月 29 日	您可以指定用于 Amazon SQS 队列的死信队列。有关更多信息，请参阅 AWS::SQS::Queue。	2010-05-15
Auto Scaling 计划操作	2014 年 1 月 27 日	您可以基于计划扩展自动扩缩组中的 EC2 实例数。通过使用计划，可以扩展应用程序来响应可预测的负载变化。有关更多信息，请参阅 AWS::AutoScaling::ScheduledAction。	2010-05-15
DynamoDB 二级索引	2014 年 1 月 27 日	您可以为 DynamoDB 数据库创建本地和全局二级索引。通过使用二级索引，可以通过主键之外的属性高效地访问数据。有关更多信息，请参阅 AWS::DynamoDB::Table。	2010-05-15
Auto Scaling 更新	2014 年 1 月 2 日	您可以为自动扩缩组或启动配置指定实例 ID。您还可以指定其他的 Auto Scaling 块设备属性。有关更多信息，请参阅 AWS::AutoScaling::AutoScalingGroup 或 AWS::AutoScaling::LaunchConfiguration。	2010-05-15
Amazon SQS 更新	2014 年 1 月 2 日	您可以更新 SQS 队列并指定其他属性。有关更多信息，请参阅 AWS::SQS::Queue。	2010-05-15
限制提高	2014 年 1 月 2 日	您可以在 CloudFormation 模板中指定最多 60 个参数和 60 个输出。	2010-05-15
新控制台	2013 年 12 月 19 日	新 CloudFormation 控制台添加了一些功能，如自动刷新堆栈事件和堆栈参数的字母数字排序。	2010-05-15
跨可用区负载均衡	2013 年 12 月 19 日	凭借跨区域负载均衡，您可以将流量路由到所有可用区 (AZ) 的后端实例。有关更多信息，请参阅 AWS::ElasticLoadBalancing::LoadBalancer。	2010-05-15
Amazon Elastic Beanstalk 环境层	2013 年 12 月 19 日	您可以指定 Amazon Elastic Beanstalk 预置资源是支持 Web 服务器还是处理后台处理任务。有关更多信息，请参阅 AWS::ElasticBeanstalk::Environment。	2010-05-15

更改	发行日期	说明	API 版本
资源名称	2013 年 12 月 19 日	您可以向以下资源分配名称（物理 ID）： • ElastiCache 集群 • Elastic Load Balancing 负载均衡器 • RDS 数据库实例 有关更多信息，请参阅名称类型。	2010-05-15
VPN 支持	2013 年 11 月 22 日	您可以启用虚拟私有网关 (VGW) 将路由传播至 VPC 的路由表。有关更多信息，请参阅 AWS::EC2::VPNGatewayRoutePropagation。	2010-05-15
有条件地创建资源并分配属性	2013 年 11 月 8 日	通过使用输入参数，您可以在 CloudFormation 模板中定义条件，从而控制指定堆栈资源的创建和设置。例如，您可以使用条件为生产环境创建堆栈资源。使用相同的模板，您还可以为测试环境创建具有较低容量的类似堆栈资源。有关更多信息，请参阅条件函数。	2010-05-15
防止意外更新堆栈资源	2013 年 11 月 8 日	您可以防止可能导致无意间更改堆栈资源的堆栈更新。例如，如果堆栈的数据库层很少更新，可以设置一个堆栈策略来防止大多数用户更新该数据库层。有关更多信息，请参阅防止更新堆栈资源。	2010-05-15

更改	发行日期	说明	API 版本
名称资源	2013 年 11 月 8 日	您可以向某些资源分配名称，而不是使用 CloudFormation 生成的物理 ID。以下 CloudFormation 资源支持命名 • Amazon CloudWatch 警报 • DynamoDB 表 • Amazon Elastic Beanstalk 应用程序和环境。 • Amazon S3 存储桶 • Amazon SNS 主题 • Amazon SQS 队列 有关更多信息，请参阅名称类型。	2010-05-15
分配自定义资源类型	2013 年 11 月 8 日	在模板中，您可以为 CloudFormation 自定义资源（`AWS::CloudFormation::CustomResource`）指定自己的资源类型。通过使用您自己的自定义资源类型名称，您可以快速识别堆栈中自定义资源的类型。例如，您可以指定 `"Type": "Custom::MyCustomResource"`。有关更多信息，请参阅 AWS::CloudFormation::CustomResource。	2010-05-15
添加伪参数	2013 年 11 月 8 日	现在，通过引用 `AWS::AccountID` 伪参数，您可以在 CloudFormation 模板内引用 Amazon AccountID。有关更多信息，请参阅虚拟参数参考。	2010-05-15
在 IAM policy 中指定堆栈	2013 年 11 月 8 日	您可以允许或拒绝 IAM 用户、组或角色对特定 CloudFormation 堆栈进行的操作。例如，您可以拒绝对特定堆栈 ID 执行的删除堆栈操作。有关更多信息，请参阅使用 Amazon Identity and Access Management 控制访问权限。	2010-05-15

更改	发行日期	说明	API 版本
联合支持	2013 年 10 月 14 日	CloudFormation 支持 IAM 角色的临时安全凭证，从而实现联合和单一登录 Amazon Web Services 管理控制台 这样的方案。您还可以从 EC2 实例调用 CloudFormation，无需通过使用 IAM 角色来嵌入长期安全凭证。有关 CloudFormation 和 IAM 的更多信息，请参阅使用 Amazon Identity and Access Management 控制访问权限。	2010-05-15
Amazon RDS 只读副本支持	2013 年 9 月 24 日	您现在可以从源数据库实例创建 Amazon RDS 只读副本。有关更多信息，请参阅 AWS::RDS::DBInstance 资源中 `SourceDBInstanceIdentifier` 属性。	2010-05-15
将公有 IP 地址与自动扩缩组中的实例关联	2013 年 9 月 19 日	您现在可以将公有 IP 地址与自动扩缩组中的实例关联。有关更多信息，请参阅 AWS::AutoScaling::LaunchConfiguration。	2010-05-15
其他 VPC 支持	2013 年 9 月 17 日	CloudFormation 添加了多项支持 VPC 和 VPN 功能的增强功能 • 您可以将一个公有 IP 地址和多个私有 IP 地址与 Amazon EC2 网络接口关联。有关更多信息，请参阅 AWS::EC2::NetworkInterface。您还可以将一个主要私有 IP 地址与一个弹性 IP 地址 (EIP) 关联。 • 您可以启用 DNS 支持并指定 DNS 主机名称。有关更多信息，请参阅 AWS::EC2::VPC。 • 您可以在虚拟私有网关与您的 VPN 网关之间指定静态路由。有关更多信息，请参阅 AWS::EC2::VPNConnectionRoute。	2010-05-15

更改	发行日期	说明	API 版本
对 Amazon ElastiCache 的 Redis 和 VPC 安全组支持	2013 年 9 月 3 日	您现在可以将 Redis 指定为 Amazon ElastiCache (ElastiCache) 集群的缓存引擎。您现在还可以对 ElastiCache 集群分配 VPC 安全组。有关更多信息，请参阅 AWS::ElastiCache::CacheCluster。	2010-05-15
并行执行堆栈创建、更新和删除，以及嵌套堆栈更新	2013 年 8 月 12 日	Amazon CloudFormation 现在可并行创建、更新和删除资源，从而提高了这些操作的性能。如果更新一个顶级模板，Amazon CloudFormation 将自动更新已发生更改的嵌套堆栈。有关更多信息，请参阅 CloudFormation 堆栈更新。	2010-05-15
现在可以在 RDS 数据库实例中设置 VPC 安全组	2013 年 2 月 28 日	现在，您可以使用 CloudFormation 将 VPC 安全组分配到 RDS 数据库实例。有关更多信息，请参阅 AWS::RDS::DBInstance 中的 VPCSecurityGroups 属性。	2010-05-15
Amazon EC2 Auto Scaling 组的滚动部署	2013 年 2 月 20 日	CloudFormation 现在支持 Amazon EC2 Auto Scaling 组的更新策略，这些策略描述如何在 Amazon EC2 Auto Scaling 组添加或移除实例时替换或修改 Amazon EC2 Auto Scaling 组中的实例。在创建堆栈更新堆栈过程中，您可以修改这些设置。 有关更多信息和示例，请参阅 UpdatePolicy 属性。	2010-05-15
堆栈更新的取消和回滚操作	2013 年 2 月 20 日	CloudFormation 支持取消堆栈更新的功能。在发出更新请求时，堆栈必须处于 UPDATE_IN_PROGRESS 状态。以下主题中提供了更多信息： • 取消堆栈更新 • cancel-update-stack • CloudFormation API Reference 中的 CancelUpdateStack	2010-05-15

更改	发行日期	说明	API 版本
Amazon EC2 Auto Scaling 组的 EBS 优化实例	2013 年 2 月 20 日	您现在可以在 Amazon EC2 Auto Scaling 组中预置 EBS 优化实例，为自动扩展的实例中的 Amazon Elastic Block Store (Amazon EBS) 提供专用吞吐量。具体的部署实施类似于以前针对优化的 Amazon EBS EC2 实例所发布的支持的实施。 有关更多信息，请参阅 AWS::AutoScaling::LaunchConfiguration 中的新 `EbsOptimized` 属性。	2010-05-15
新文档	2012 年 12 月 21 日	AWS::EC2::Instance 现在提供 `BlockDeviceMappings` 属性，以便您设置 EC2 实例的块设备映射。 通过这一更改，增加了两个新类型： • BlockDeviceMapping • 块设备	2010-05-15
新文档	2012 年 12 月 21 日	增加了新章节，来说明使用最近重新设计过的 Amazon Web Services 管理控制台 来创建和查看堆栈的流程。这些章节如下： • 从 CloudFormation 控制台创建堆栈 • 从 CloudFormation 控制台查看堆栈信息	2010-05-15
新文档	2012 年 11 月 15 日	以下主题提供有关自定义资源的信息： • 自定义资源 • AWS::CloudFormation::CustomResource • 自定义资源参考	2010-05-15

更改	发行日期	说明	API 版本
更新的文档	2012 年 11 月 15 日	CloudFormation 现在支持为 RDS 数据库实例指定预置每秒 I/O 操作数（IOPS）。利用 AWS::RDS::DBInstance 中的新 Iops 属性，您可以以 1000 IOPS 增量将该值设置为 1000-10000。 有关为 RDS 数据库实例指定 IOPS 的更多信息，请参阅《Amazon Relational Database Service 用户指南》中的预调配 IOPS。	2010-05-15
新增和更新的文档	2012 年 8 月 27 日	重新组织了主题，以便更清楚地提供有关使用 Amazon Web Services 管理控制台 和使用 CloudFormation 命令行界面（CLI）的特定信息。 增加了关于标记 CloudFormation 堆栈的信息，包括新的指南和更新过的参考主题： • 新主题：配置堆栈选项。 • 《CloudFormation API reference》中有关标签的新信息：CreateStack、Stack 和 Tag。 有关处理 Windows 堆栈的新信息： • Microsoft Windows 亚马逊机器映像（AMI）和 Amazon CloudFormation 模板 • 启动 Amazon CloudFormation Windows 堆栈 新主题：在 CloudFormation 模板中使用正则表达式。	2010-05-15

更改	发行日期	说明	API 版本
新功能	2012 年 4 月 25 日	CloudFormation 现在通过 Amazon EC2 提供对虚拟私有云（VPC）安全性的完全支持。您现在可以使用单个 CloudFormation 模板并利用每一种类型的 VPC 资源（子网、网关、网络 ACL、路由表等）创建和填充整个 VPC。 有关以下资源类型的文档已更新： • AWS::EC2::SecurityGroup • AWS::EC2::SecurityGroupIngress • AWS::EC2::SecurityGroupEgress • AWS::EC2::Instance • AWS::AutoScaling::AutoScalingGroup • AWS::EC2::EIP • AWS::EC2::EIPAssociation • AWS::ElasticLoadBalancing::LoadBalancer 文档中增加了新资源类型： • AWS::EC2::VPC • AWS::EC2::InternetGateway • AWS::EC2::DHCPOptions • AWS::EC2::DHCPOptions • AWS::EC2::RouteTable • AWS::EC2::NetworkAcl • AWS::EC2::NetworkAclEntry • AWS::EC2::Subnet • AWS::EC2::VPNGateway • AWS::EC2::CustomerGateway	2010-05-15

更改	发行日期	说明	API 版本
新功能	2012 年 4 月 13 日	CloudFormation 现在允许您在更新堆栈时在堆栈中添加或移除元素。CloudFormation 堆栈更新已更新，演练中增加了一个新的部分：更改堆栈的资源，此部分说明如何在更新堆栈时添加和移除资源。	2010-05-15
新功能	2012 年 2 月 2 日	CloudFormation 现在为现有 Amazon Virtual Private Cloud（Amazon VPC）中的资源提供支持。借助此版本，您可以： • 在现有 Amazon VPC 中启动 EC2 专用实例。有关更多信息，请参阅 AWS::EC2::Instance。 • 设置现有 Amazon VPC 中的 EC2 实例的 SourceDestCheck 属性。有关更多信息，请参阅 AWS::EC2::Instance。 • 在现有 Amazon VPC 中创建弹性 IP 地址。有关更多信息，请参阅 AWS::EC2::EIP。 • 使用 CloudFormation 在现有 VPC 中创建 Amazon VPC 安全组和入口/出口规则。有关更多信息，请参阅 AWS::EC2::SecurityGroup。 • 通过设置 AWS::AutoScaling::AutoScalingGroup 资源的 VPCZoneIdentifier 属性，将自动扩缩组与现有 Amazon VPC 关联。有关更多信息，请参阅 AWS::AutoScaling::AutoScalingGroup。 • 向 Amazon VPC 子网挂载一个 Elastic Load Balancing 负载均衡器，并为此负载均衡器创建安全组。有关更多信息，请参阅 AWS::ElasticLoadBalancing::LoadBalancer。 • 在现有 Amazon VPC 中创建 RDS 数据库实例。有关更多信息，请参阅 AWS::RDS::DBInstance。	2010-05-15

更改	发行日期	说明	API 版本
新功能	2012 年 2 月 2 日	您现在可以在现有堆栈中更新以下资源的属性： • AWS::EC2::SecurityGroupIngress • AWS::EC2::SecurityGroupEgress • AWS::EC2::EIPAssociation • AWS::RDS::DBSubnetGroup • AWS::RDS::DBSecurityGroup • AWS::RDS::DBSecurityGroupIngress • AWS::Route53::RecordSetGroup 有关可更新资源的完整列表，以及更新堆栈时要考虑的详细事项，请参阅 CloudFormation 堆栈更新。	2010-05-15
新编指南	2012 年 2 月 2 日	已将现有部分重新组织为新部分：使用 CloudFormation 模板和管理堆栈。已将模板参考移至目录的最高一层。已将估计 CloudFormation 堆栈成本移至“入门”部分。	2010-05-15
新增内容	2012 年 2 月 2 日	新增部分： • 演练：更新堆栈是说明更新 LAMP 堆栈过程的教程。 • 使用 Amazon CloudFormation 在 Amazon EC2 上部署应用程序介绍如何使用 CloudFormation 帮助程序脚本通过存储在模板中的元数据来部署应用程序。 • CloudFormation 帮助程序脚本参考提供 CloudFormation 帮助程序脚本（cfn-init、cfn-get-metadata、cfn-signal 和 cfn-hup）的参考材料。	2010-05-15
新功能	2011 年 5 月 26 日	CloudFormation 现在提供 `list-stacks` 命令，您可使用该命令按堆栈状态列出筛选出的堆栈。删除的堆栈在删除后最多还可在 90 天内列出。有关更多信息，请参阅描述并列出堆栈。	2010-05-15

更改	发行日期	说明	API 版本
新特征	2011 年 5 月 26 日	使用 `describe-stack-resources` 和 `get-template` 命令，现在可从已经删除了 90 天的堆栈中获取信息。有关更多信息，请参阅列出资源和检索模板。	2010-05-15
新链接	2011 年 3 月 1 日	CloudFormation 端点信息现在位于 Amazon Web Services 一般参考 中。有关更多信息，请参阅 Amazon 一般参考中的“区域和终端节点”。	2010-05-15
初始版本	2011 年 2 月 25 日	CloudFormation 的第一个公开发布版。	2010-05-15