



使用者指南

AWS CodeBuild



API 版本 2016-10-06

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

AWS CodeBuild: 使用者指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任從何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 AWS CodeBuild ?	1
註冊 AWS 帳戶	1
如何執行 CodeBuild	1
CodeBuild 的定價	3
如何開始使用 CodeBuild ?	3
概念	3
CodeBuild 的運作方式	3
後續步驟	5
開始使用	6
開始使用主控台	6
步驟 1：建立原始程式碼	7
步驟 2：建立 buildspec 檔案	10
步驟 3：建立兩個 S3 儲存貯體	12
步驟 4：上傳原始程式碼和 Buildspec 檔案	12
步驟 5：建立建置專案	14
步驟 6：執行建置	15
步驟 7：檢視摘要建置資訊	16
步驟 8：檢視詳細建置資訊	16
步驟 9：取得建置輸出成品	17
步驟 10：刪除 S3 儲存貯體	18
包裝	19
AWS CLI使用入門	19
步驟 1：建立原始程式碼	20
步驟 2：建立 buildspec 檔案	23
步驟 3：建立兩個 S3 儲存貯體	25
步驟 4：上傳原始程式碼和 Buildspec 檔案	25
步驟 5：建立建置專案	26
步驟 6：執行建置	30
步驟 7：檢視摘要建置資訊	32
步驟 8：檢視詳細建置資訊	34
步驟 9：取得建置輸出成品	37
步驟 10：刪除 S3 儲存貯體	37
包裝	38
使用案例範例	39

跨服務範例	40
Amazon ECR 範例	40
Amazon EFS 範例	47
AWS CodePipeline 範例	52
AWS Config 範例	63
建置通知範例	64
建置徽章範例	78
使用建置徽章建立建置專案	79
存取 AWS CodeBuild 建置徽章	81
發佈 CodeBuild 組建徽章	82
CodeBuild 徽章狀態	82
測試報告範例	83
執行測試報告範例	83
CodeBuild 的 Docker 範例	89
自訂映像中的 Docker 範例	90
Docker 映像建置伺服器範例	93
Windows Docker 建置範例	95
「將 Docker 映像發佈至 Amazon ECR」範例	98
具有 AWS Secrets Manager 範例的私有登錄檔	106
在 S3 儲存貯體中託管建置輸出	109
多個輸入和輸出範例	112
建立具有多個輸入和輸出的建置專案	112
建立沒有來源的專案	115
Buildspec 檔案範例中的執行時間版本	116
更新 buildspec 檔案中的執行時間版本	116
指定兩個執行時間	120
來源版本範例	124
使用遞交 ID 指定 GitHub 儲存庫版本	126
使用參考和遞交 ID 指定 GitHub 儲存庫版本	127
第三方來源儲存庫範例	128
執行 Bitbucket 範例	129
執行 GitHub Enterprise Server 範例	133
執行 GitHub 提取請求和 Webhook 篩選條件範例	139
教學課程：在 CodeBuild 中使用 Fastlane 簽署 Apple 程式碼，並使用 S3 進行憑證儲存	143
教學課程：使用 GitHub 在 CodeBuild 中使用 Fastlane 簽署 Apple 程式碼以進行憑證儲存 ...	148
在建置時間設定成品名稱	154

執行 Windows 範例	157
執行 Windows 範例	157
目錄結構	158
F# 和 .NET Framework	158
Visual Basic 和 .NET Framework	159
檔案	159
F# 和 .NET Framework	159
Visual Basic 和 .NET Framework	164
規劃組建	178
Buildspec 參考	180
Buildspec 檔案名稱和儲存位置	180
Buildspec 語法	181
version	184
執行方式	184
env	184
proxy	189
階段	189
報告	192
成品	194
快取	200
Buildspec 範例	201
Buildspec 版本	204
Batch buildspec 參考	205
批次	205
batch/build-graph	205
batch/build-list	208
batch/build-matrix	210
batch/build-fanout	212
建置環境參考	214
CodeBuild 提供的 Docker 映像	214
取得目前 Docker 映像的清單	215
EC2 運算映像	215
Lambda 運算映像	217
已棄用 CodeBuild 映像	223
可用的執行時間	224
執行時間版本	249

建置環境運算模式和類型	253
關於運算	253
關於預留容量環境類型	253
關於隨需環境類型	316
建置環境中的 Shell 和命令	328
建置環境中的環境變數	329
建置環境中的背景工作	334
組建專案	336
建立組建專案	336
先決條件	337
建立組建專案 (主控台)	337
建立建置專案 (AWS CLI)	355
建立建置專案AWS SDKs)	373
建立建置專案 (CloudFormation)	374
建立通知規則	374
變更建置專案設定	376
變更建置專案的設定 (主控台)	377
變更建置專案的設定 (AWS CLI)	396
變更建置專案的設定 (AWS SDKs)	397
多個存取字符	398
步驟 1：建立 Secrets Manager 秘密或 AWS CodeConnections 連線	398
步驟 2：授予 CodeBuild 專案 IAM 角色對 Secrets Manager 秘密的存取權	399
步驟 3：設定 Secrets Manager 或 AWS CodeConnections 權杖	401
其他設定選項	404
刪除組建專案	407
刪除建置專案 (主控台)	408
刪除建置專案 (AWS CLI)	408
刪除組建專案AWS SDKs)	409
取得公有建置專案 URLs	409
共用建置專案	410
共用專案	410
相關服務	413
存取共用專案	413
取消共用共用專案	414
識別共用專案	414
共用的專案許可	414

標記建置專案	415
將標籤新增至專案	415
檢視專案的標籤	417
編輯專案的標籤	418
從專案移除標籤	419
使用執行器	420
GitHub Actions	420
GitLab 執行器	442
Buildkite 執行器	455
使用 Webhook	475
使用 Webhook 的最佳實務	475
Bitbucket Webhook 事件	476
GitHub 全域和組織 Webhook	490
GitHub 手動 Webhook	496
GitHub Webhook 事件	497
GitLab 群組 Webhook	513
GitLab 手動 Webhook	519
GitLab Webhook 事件	520
Buildkite 手動 Webhook	534
提取請求評論核准	535
檢視組建專案詳細資訊	540
檢視建置專案的詳細資訊 (主控台)	541
檢視建置專案的詳細資訊 (AWS CLI)	541
檢視建置專案的詳細資訊 (AWS SDKs)	543
檢視組建專案名稱	543
檢視建置專案名稱清單 (主控台)	544
檢視建置專案名稱清單 (AWS CLI)	544
檢視組建專案名稱 (AWS SDKs的清單)	545
Builds	546
手動執行組建	547
在本機執行建置	547
執行建置 (主控台)	551
執行建置 (AWS CLI)	552
執行批次建置 (AWS CLI)	557
自動開始執行建置 (AWS CLI)	558
自動停止執行建置 (AWS CLI)	559

執行組建AWS SDKs)	560
在 Lambda 運算上執行組建	560
在 上執行的精選執行時間環境Docker 映像中，會包含哪些工具和執行時間 AWS Lambda？	561
如果策劃的映像不包含我需要的工具，該怎麼辦？	561
哪些區域支援 CodeBuild 中的 AWS Lambda 運算？	562
AWS Lambda 運算的限制	562
使用 AWS SAM 搭配 CodeBuild Lambda Java 部署 Lambda 函數	562
使用 CodeBuild Lambda Node.js 建立單一頁面 React 應用程式	566
使用 CodeBuild Lambda Python 更新 Lambda 函數組態	568
在預留容量機群上執行組建	572
建立預留容量機群	573
最佳實務	575
我可以跨多個 CodeBuild 專案共用預留容量機群嗎？	575
屬性型運算如何運作？	576
我可以手動指定機群的 Amazon EC2 執行個體嗎？	576
哪些區域支援預留容量機群？	576
如何設定預留容量 macOS 機群？	577
如何為預留容量機群設定自訂 Amazon Machine Image (AMI)？	578
預留容量機群的限制	579
預留容量機群屬性	579
預留容量範例	584
執行批次建置	586
安全角色	586
批次建置類型	586
批次報告模式	590
其他資訊	590
執行平行測試	591
中的支援 AWS CodeBuild	592
在批次建置中啟用平行測試執行	594
使用 codebuild-tests-run CLI 命令	595
使用 codebuild-glob-search CLI 命令	597
關於測試分割	598
自動合併個別建置報告	599
平行測試執行範例	601
快取建置	611

Amazon S3 快取	611
本機快取	618
指定本機快取	619
偵錯組建	621
使用 CodeBuild 沙盒對組建進行偵錯	621
使用 Session Manager 偵錯組建	622
使用 CodeBuild 沙盒對組建進行偵錯	622
使用 Session Manager 偵錯組建	651
刪除建置	656
刪除建置 (AWS CLI)	656
刪除組建 (AWS SDKs)	657
手動重試建置	657
手動重試組建 (主控台)	657
手動重試組建 (AWS CLI)	658
手動重試組建 (AWS SDKs)	658
重試自動建置	659
自動重試建置 (主控台)	659
自動重試建置 (AWS CLI)	659
自動重試組建 (AWS SDKs)	660
停止建置	660
停止建置 (主控台)	660
停止建置 (AWS CLI)	661
停止建置 (AWS SDKs)	661
停止批次建置	661
停止批次建置 (主控台)	662
停止批次建置 (AWS CLI)	662
停止批次建置AWS SDKs)	663
自動觸發建置	663
建立建置觸發	663
編輯建置觸發條件	666
檢視建置的詳細資訊	668
檢視建置的詳細資訊 (主控台)	669
檢視建置的詳細資訊 (AWS CLI)	669
檢視建置詳細資訊AWS SDKs)	670
建置階段轉換	670
檢視組建 IDs	670

檢視建置 ID 清單 (主控台)	671
檢視建置 ID 清單 (AWS CLI)	671
檢視批次建置 IDs的清單 (AWS CLI)	672
檢視組建 IDs(AWS SDKs的清單)	674
檢視組建專案的組建 IDs	674
檢視建置專案的建置 ID 清單 (主控台)	674
檢視建置專案的建置 ID 清單 (AWS CLI)	674
檢視組建專案的批次組建 IDs 清單 (AWS CLI)	676
檢視組建專案的組建 IDs 清單 (AWS SDKs)	677
測試報告	678
建立測試報告	679
建立程式碼涵蓋範圍報告	680
建立程式碼涵蓋範圍報告	681
自動探索報告	682
使用主控台設定報告自動探索	683
使用專案環境變數設定報告自動探索	683
報告群組	683
建立報告群組	684
報告群組命名	689
共用報告群組	690
指定測試檔案	695
指定測試命令	696
標記報表群組	696
更新報告群組	702
測試架構	705
設定 Jasmine	705
設定 Jest	707
設定 pytest	708
設定 RSpec	709
檢視測試報告	710
檢視建置的測試報告	711
檢視報告群組的測試報告	711
檢視您 AWS 帳戶中的測試報告	711
測試報告許可	712
測試報告的 IAM 角色	712
測試報告操作的許可	714

測試報告許可範例	714
測試報告狀態	715
VPC 支援	716
使用案例	716
VPC 最佳實務	717
VPCs的限制	717
允許 CodeBuild 專案中的 Amazon VPC 存取	717
對 VPC 設定進行故障診斷	719
使用 VPC 端點	719
建立 VPC 端點之前	719
為 CodeBuild 建立 VPC 端點	720
為 CodeBuild 建立 VPC 端點政策	720
使用 CodeBuild 受管代理伺服器	721
設定預留容量機群的受管代理組態	722
執行 CodeBuild 預留容量機群	723
使用代理伺服器	723
設定在代理伺服器中執行 CodeBuild 所需的元件	724
在明確代理伺服器中執行 CodeBuild	726
在透明代理伺服器中執行 CodeBuild	730
在代理伺服器中執行套用管理員和其他工具	732
CloudFormation VPC 範本	734
日誌記錄和監控	740
Log CodeBuild API 呼叫	740
關於 CloudTrail 中 AWS CodeBuild 的資訊	740
關於 AWS CodeBuild 日誌檔案項目	741
監控組建	743
CloudWatch 指標	744
CloudWatch 資源使用率指標	746
CloudWatch 維度	748
CloudWatch 警示	748
檢視 CodeBuild 指標	749
檢視 CodeBuild 資源使用率指標	751
在 CloudWatch 中建立 CodeBuild 警示	755
安全	756
資料保護	756
資料加密	758

金鑰管理	758
流量隱私權	759
身分與存取管理	759
管理存取概觀	759
使用身分型政策	763
AWS CodeBuild 許可參考	790
使用標籤控制對 AWS CodeBuild 資源的存取	797
在主控台檢視資源	801
法規遵循驗證	801
恢復能力	802
基礎設施安全性	802
來源提供者存取	802
在 Secrets Manager 秘密中建立和存放權杖	802
GitHub 和 GitHub Enterprise Server 存取	805
Bitbucket 存取	815
GitLab 存取	825
預防跨服務混淆代理人	831
進階主題	833
允許使用者與 CodeBuild 互動	833
允許 CodeBuild 與其他 AWS 服務互動	840
加密建置輸出	847
使用 與 CodeBuild 互動 AWS CLI	849
命令列參考	850
AWS SDKs和工具參考	851
的支援 AWS SDKs和工具 AWS CodeBuild	851
使用 AWS SDKs	852
指定 CodeBuild 端點	853
指定 AWS CodeBuild 端點 (AWS CLI)	854
指定 AWS CodeBuild 端點 (AWS SDK)	854
搭配 CodePipeline 使用 CodeBuild CodePipeline	856
先決條件	857
建立管道 (主控台)	859
建立管道 (AWS CLI)	862
新增建置動作	867
新增測試動作	870
搭配 Codecov 使用 CodeBuild	873

將 Codecov 整合至建置專案	874
搭配 Jenkins 使用 CodeBuild	877
設定 Jenkins	877
安裝外掛程式。	877
使用 外掛程式	877
搭配無伺服器應用程式使用 CodeBuild	880
相關資源	880
第三方通知	880
1) 基本 Docker 影像—windowsservercore	881
2) Windows 基底 Docker 影像—choco	881
3) Windows-base Docker 影像 -- 2.16.2 版	882
4) Windows-base Docker 影像 -microsoft-build-tools -- 15.0.26320.2 版	882
5) Windows-base Docker 影像—nuget.commandline -- 4.5.1 版	885
7) Windows-base Docker 影像—netfx-4.6.2-devpack	885
8) Windows-base Docker 映像—visualfsharptools , 4.0 版	886
9) Windows-base Docker 影像—netfx-pcl-reference-assemblies-4.6	887
10) Windows 基礎 Docker 映像—visualcppbuildtools v 14.0.25420.1	889
11) windows-base Docker 影像—microsoft-windows-netfx3-ondemand-package.cab	892
12) Windows-base Docker 影像—dotnet-sdk	893
使用 CodeBuild 條件索引鍵做為 IAM 服務角色變數	893
AWS CodeBuild 條件索引鍵	894
在專案和機群上強制執行 VPC 連線設定	894
防止對專案 buildspec 進行未經授權的修改	895
限制您建置的運算類型	896
控制環境變數設定	897
在條件索引鍵名稱中使用變數	898
檢查 API 請求中是否存在屬性	898
程式碼範例	900
基本概念	900
動作	900
公有建置 API	917
公有建置動作	917
DescribeBuildBatchesForPublicProject	918
DescribeBuildsForPublicProject	921
GetCloudWatchLogsForPublicBuild	926
GetPresignedUrlsForPublicBuild	929

GetPublicBuild	932
GetPublicBuildBatch	936
GetPublicProject	941
公有建置資料類型	944
BuildBatchForDescribeBuildBatchesPublic	946
PublicBuild	948
PublicBuildArtifacts	952
PublicBuildBatch	953
PublicBuildGroup	957
PublicBuildSummary	959
PublicLogsStatus	961
PublicProject	962
PublicProjectArtifacts	965
PublicProjectBuildBatchConfig	966
PublicProjectEnvironment	967
PublicProjectSource	969
PublicWebhook	971
S3Downloadable	973
疑難排解	975
Apache Maven 建置參考錯誤儲存庫中的成品	976
建置命令預設以 root 身分執行	977
當檔案名稱具有非美國 時，組建可能會失敗。英文字元	978
從 Amazon EC2 參數存放區取得參數時，組建可能會失敗	978
無法存取 CodeBuild 主控台的分支篩選條件	979
無法檢視組建成功或失敗	980
未向來源提供者報告建置狀態	980
找不到並選取 Windows Server Core 2019 平台的基礎映像	980
在 Buildspec 檔案中後來的命令無法辨識先前的命令	981
錯誤：嘗試下載快取時出現「存取遭拒」	981
使用自訂建置映像時發生錯誤："BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE"	981
錯誤：「完成建置之前，發現建置容器失效。因為記憶體不足，或不支援 Docker 映像，建置容器失效。ErrorCode: 500」	982
執行組建時發生錯誤：「無法連接到 Docker 協助程式」	983
錯誤：建立或更新組建專案時，「CodeBuild 無權執行：sts：AssumeRole」	984
錯誤：「呼叫 GetBucketAcl 時發生錯誤：儲存貯體擁有者已變更，或服務角色不再有權呼叫 s3:GetBucketAcl」	985

執行建置時發生錯誤：「無法上傳成品：無效的 arn」	985
錯誤：「Git 複製失敗：無法存取 'your-repository-URL'：SSL 憑證問題：自我簽署憑證」	985
執行建置時發生錯誤：「必須使用指定的端點來定址您嘗試存取的儲存貯體」	986
錯誤：「此建置映像需要選取至少一個執行時間版本。」	986
錯誤："QUEUED: INSUFFICIENT_SUBNET"，當建置佇列中的建置失敗時	987
錯誤：「無法下載快取：RequestError: 傳送請求失敗，原因：x509: 無法載入系統根目錄，未提供根目錄」	988
錯誤：「無法從 S3 下載憑證。AccessDenied」	988
錯誤：「找不到登入資料」	988
在代理伺服器中執行 CodeBuild 時發生 RequestError 逾時錯誤	989
Bourne Shell (sh) 必須存在於建置映像中	991
執行建置時出現警告：「正在略過執行時間的安裝。此建置映像不支援執行時間版本選項」	991
錯誤：「無法驗證 JobWorker 身分」	991
組建無法啟動	991
在本機快取建置中存取 GitHub 中繼資料	992
AccessDenied：報告群組的儲存貯體擁有者不符合 S3 儲存貯體的擁有者... ..	992
錯誤：使用 建立 CodeBuild 專案時，「您的登入資料缺少一或多個必要的權限範圍」 AWS CodeConnections	992
錯誤：使用 Ubuntu 安裝命令建置時，「很抱歉，完全沒有終端請求 - 無法取得輸入」	993
Bitbucket 組建在使用 Secrets Manager OAuth 後失敗並出現身分驗證錯誤	994
配額	995
Service Quotas	995
其他限制	999
組建專案	999
Builds	999
運算機群	1000
報告	1001
Tags (標籤)	1001
文件歷史紀錄	1003
舊版更新	1021
.....	mxxx

什麼是 AWS CodeBuild ？

AWS CodeBuild 是雲端中全受管的建置服務。CodeBuild 可編譯原始碼、執行單元測試，並產生可立即部署的成品。使用 CodeBuild 即可不必佈建、管理、擴展自己的組建伺服器。它提供預先封裝的組建環境，適用於常見的程式設計語言和組建工具，例如 Apache Maven、Gradle 等等。您也可以自訂 CodeBuild 中的組建環境，以使用您自己的組建工具。CodeBuild 會自動擴展以因應高峰建置請求。

CodeBuild 提供以下優點：

- 完全受管 – CodeBuild 不需要設定、修補、更新和管理您自己的建置伺服器。
- 隨需 – CodeBuild 會隨需擴展以符合您的建置需求。您只需針對實際使用的組建分鐘數付費。
- 開箱即用 – CodeBuild 為最熱門的程式設計語言提供預先設定的建置環境。您只需要指向組建指令碼來啟動第一個組建即可。

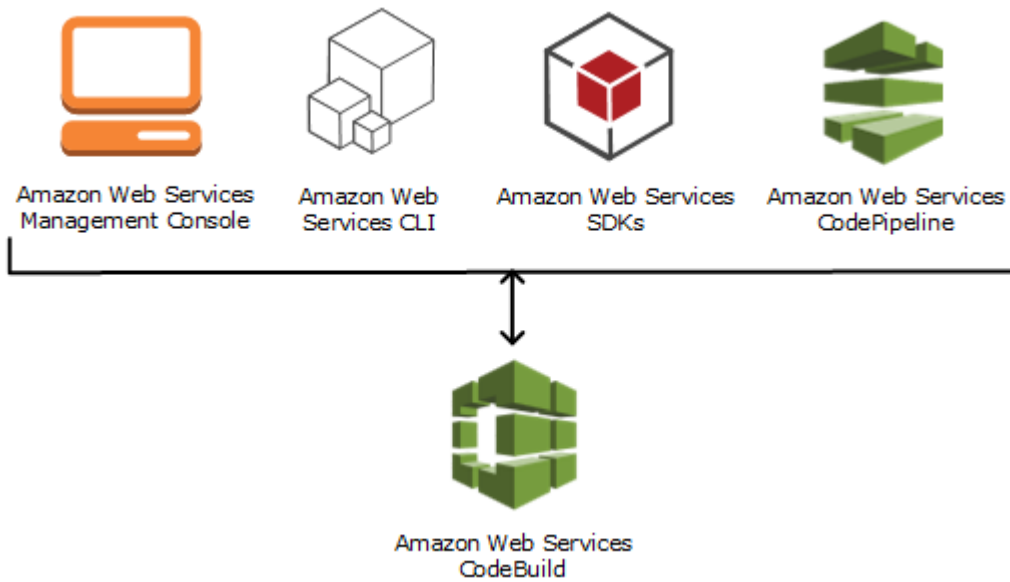
如需詳細資訊，請參閱[AWS CodeBuild](#)。

註冊 AWS 帳戶

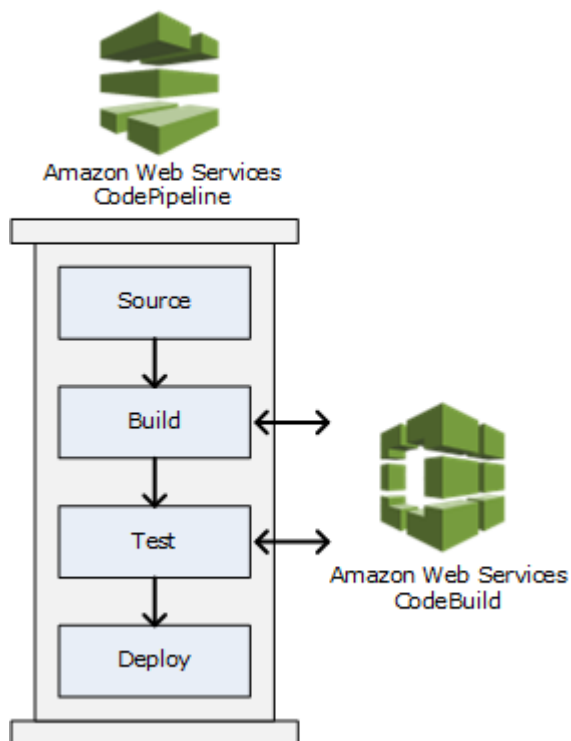
若要開始使用 AWS，您需要 AWS 帳戶。有兩種註冊方式 AWS。如果您想要完全控制帳戶組態和內容的存放位置，或者如果您有 HIPAA 或 FedRAMP 等受管制工作負載，請使用[註冊 AWS（進階）](#)。如果您是新使用者，並想要使用您已擁有的登入來登入，AWS 並在選取 中建立具有預先設定預設值 AWS 的環境[AWS 區域](#)，請使用[註冊 AWS](#)。

如何執行 CodeBuild

您可以使用 AWS CodeBuild 或 AWS CodePipeline 主控台來執行 CodeBuild。您也可以使用 AWS Command Line Interface (AWS CLI) 或 AWS SDKs 自動化 CodeBuild 的執行。



如下圖所示，您可以將 CodeBuild 作為建置或測試動作新增至中管道的建置或測試階段 AWS CodePipeline。AWS CodePipeline 是一種持續交付服務，可用於模型化、視覺化和自動化發行程式碼所需的步驟。其中包括建置您的程式碼。管道是一個工作流程建構，說明程式碼變更如何進行發行程序。



若要使用 CodePipeline 建立管道，然後新增 CodeBuild 組建或測試動作，請參閱 [搭配 CodePipeline 使用 CodeBuild CodePipeline](#)。如需有關 CodePipeline 的詳細資訊，請見《AWS CodePipeline 使用者指南》<https://docs.aws.amazon.com/codepipeline/latest/userguide/>。

CodeBuild 主控台也提供快速搜尋資源的方法，例如儲存庫、建置專案、部署應用程式和管道。選擇 Go to resource (移至資源)，或按 / 鍵，然後輸入資源名稱。任何相符項目都會出現在清單中。搜尋不區分大小寫。您只會看到您有權檢視的資源。如需詳細資訊，請參閱[在主控台檢視資源](#)。

CodeBuild 的定價

如需詳細資訊，請參閱 [CodeBuild 定價](#)。

如何開始使用 CodeBuild？

建議您完成下列步驟：

1. 閱讀 中的資訊，進一步了解 CodeBuild [概念](#)。
2. 遵循 中的指示，在範例案例中實驗 CodeBuild [開始使用主控台](#)。
3. 遵循 中的指示，在您自己的案例中使用 CodeBuild [規劃組建](#)。

AWS CodeBuild 概念

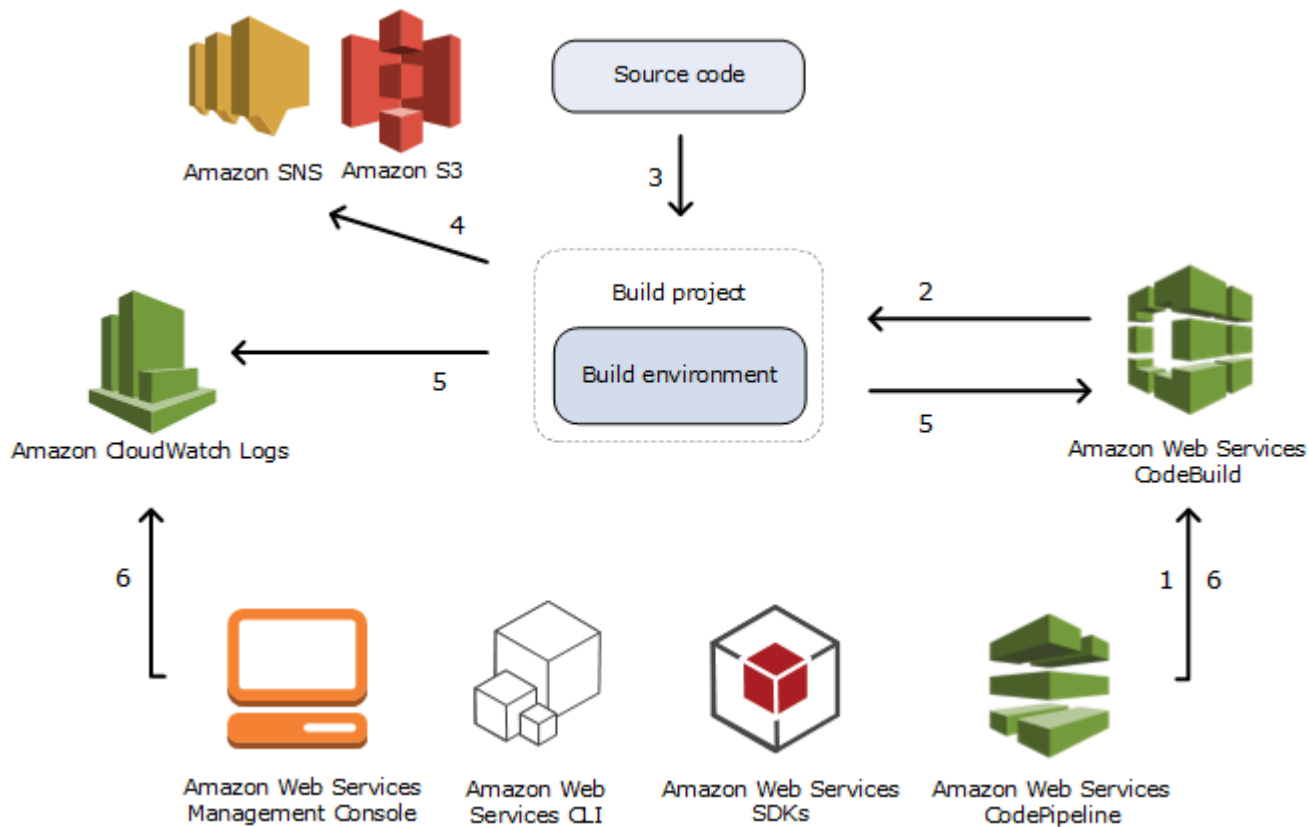
下列概念對於了解 CodeBuild 的運作方式非常重要。

主題

- [CodeBuild 的運作方式](#)
- [後續步驟](#)

CodeBuild 的運作方式

下圖顯示使用 CodeBuild 執行組建時會發生的情況：



1. 作為輸入，您必須向 CodeBuild 提供建置專案。建置專案包含如何執行建置的相關資訊，包括取得原始程式碼的位置、要使用的建置環境、要執行的建置命令，以及儲存建置輸出的位置。建置環境代表作業系統、程式設計語言執行時間，以及 CodeBuild 用來執行建置之工具的組合。如需詳細資訊，請參閱：
 - [建立組建專案](#)
 - [建置環境參考](#)
2. CodeBuild 使用建置專案來建立建置環境。
3. CodeBuild 會將原始程式碼下載到組建環境中，然後使用組建規格 (buildspec)，如組建專案中所定義或直接包含在原始程式碼中。buildspec 是組建命令和相關設定的集合，採用 YAML 格式，可供 CodeBuild 用來執行組建。如需更多資訊，請參閱[Buildspec 參考](#)。
4. 如果有任何建置輸出，建置環境會將其輸出上傳至 S3 儲存貯體。建置環境也可以執行您在 buildspec 中指定的任務（例如，傳送建置通知至 Amazon SNS 主題）。如需範例，請參閱[建置通知範例](#)。
5. 當組建執行時，組建環境會將資訊傳送至 CodeBuild 和 Amazon CloudWatch Logs。

6. 當組建執行時，您可以使用 AWS CodeBuild 主控台、AWS CLI 或 AWS SDKs，從 CodeBuild 取得摘要組建資訊，並從 Amazon CloudWatch Logs 取得詳細組建資訊。如果您使用 AWS CodePipeline 執行組建，您可以從 CodePipeline 取得有限的組建資訊。

後續步驟

現在您已進一步了解 AWS CodeBuild，我們建議您執行下列後續步驟：

1. 遵循 [中的指示](#)，在範例案例中實驗 CodeBuild [開始使用主控台](#)。
2. 遵循 [中的指示](#)，在您自己的案例中使用 CodeBuild [規劃組建](#)。

CodeBuild 入門

在以下教學課程中，您會使用 AWS CodeBuild 將範例原始程式碼輸入檔案的集合建置到可部署版本的原始程式碼。

這兩個教學課程都有相同的輸入和結果，但其中一個使用 AWS CodeBuild 主控台，另一個則使用 AWS CLI。

Important

我們不建議您使用 AWS 根帳戶來完成本教學課程。

主題

- [AWS CodeBuild 開始使用 主控台](#)
- [AWS CodeBuild 開始使用 AWS CLI](#)

AWS CodeBuild 開始使用 主控台

在本教學課程中，您會使用 AWS CodeBuild 將範例原始碼輸入檔案集合 (建置輸入成品或建置輸入) 建置為原始碼的可部署版本 (建置輸出成品或建置輸出)。具體而言，您會指示 CodeBuild 使用常見的建置工具 Apache Maven，將一組 Java 類別檔案建置至 Java Archive (JAR) 檔案。您不需要熟悉 Apache Maven 或 Java，也能完成本教學課程。

您可以透過 CodeBuild 主控台、AWS CodePipeline、AWS CLI 或 AWS SDKs 使用 CodeBuild。本教學課程示範如何使用 CodeBuild 主控台。如需使用 CodePipeline 的資訊，請參閱 [搭配 CodePipeline 使用 CodeBuild CodePipeline](#)。

Important

本教學中的步驟要求您建立資源（例如 S3 儲存貯體），這可能會導致 AWS 您的帳戶產生費用。這包括 CodeBuild 以及與 Amazon S3 相關的 AWS 資源和動作 AWS KMS，以及 CloudWatch Logs 的可能費用。如需詳細資訊，請參閱 [AWS CodeBuild 定價](#)、[Amazon S3 定價](#)、[AWS Key Management Service 定價](#) 和 [Amazon CloudWatch 定價](#)。

主題

- [步驟 1：建立原始程式碼](#)
- [步驟 2：建立 buildspec 檔案](#)
- [步驟 3：建立兩個 S3 儲存貯體](#)
- [步驟 4：上傳原始程式碼和 Buildspec 檔案](#)
- [步驟 5：建立建置專案](#)
- [步驟 6：執行建置](#)
- [步驟 7：檢視摘要建置資訊](#)
- [步驟 8：檢視詳細建置資訊](#)
- [步驟 9：取得建置輸出成品](#)
- [步驟 10：刪除 S3 儲存貯體](#)
- [包裝](#)

步驟 1：建立原始程式碼

(部分：[AWS CodeBuild 開始使用 主控台](#))

在此步驟中，您會建立您希望 CodeBuild 建置到輸出儲存貯體的原始程式碼。此來源碼由兩個 Java 類別檔案及一個 Apache Maven Project Object Model (POM) 檔案組成。

1. 在您本機電腦或執行個體上的空目錄內，建立此目錄結構。

```
(root directory name)
  |-- src
    |-- main
    |   |-- java
    |-- test
    |   |-- java
```

2. 使用您選擇的文字編輯器建立此檔案，將它命名為 MessageUtil.java，然後將它儲存在 src/main/java 目錄中。

```
public class MessageUtil {
    private String message;

    public MessageUtil(String message) {
        this.message = message;
    }
}
```

```
public String printMessage() {
    System.out.println(message);
    return message;
}

public String salutationMessage() {
    message = "Hi!" + message;
    System.out.println(message);
    return message;
}
}
```

此類別檔案會將傳遞給它的字元字串做為輸出建立。MessageUtil 建構函數會設定字元字串。printMessage 方法會建立輸出。salutationMessage 方法會輸出 Hi!，其後跟隨字元字串。

3. 建立此檔案，將它命名為 TestMessageUtil.java，然後將它儲存在 /src/test/java 目錄中。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

    String message = "Robert";
    MessageUtil messageUtil = new MessageUtil(message);

    @Test
    public void testPrintMessage() {
        System.out.println("Inside testPrintMessage()");
        assertEquals(message,messageUtil.printMessage());
    }

    @Test
    public void testSalutationMessage() {
        System.out.println("Inside testSalutationMessage()");
        message = "Hi!" + "Robert";
        assertEquals(message,messageUtil.salutationMessage());
    }
}
```

此類別檔案會將 MessageUtil 類別中的 message 變數設為 Robert。它接著會透過檢查 Robert 和 Hi!Robert 字串是否出現在輸出中，來測試 message 變數是否設定成功。

4. 建立此檔案，將它命名為 pom.xml，然後將它儲存在根 (最上層) 目錄中。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven 會使用此檔案中的說明，將 MessageUtil.java 和 TestMessageUtil.java 轉換成名為 messageUtil-1.0.jar 的檔案，然後執行指定的測試。

此時您的目錄結構看起來應該會如下。

```
(root directory name)
|-- pom.xml
```

```
`-- src
    |-- main
    |     |-- java
    |     |-- MessageUtil.java
    |-- test
    |     |-- java
    |     |-- TestMessageUtil.java
```

步驟 2：建立 buildspec 檔案

(上一個步驟：[步驟 1：建立原始程式碼](#))

在此步驟中，您會建立組建規格 (build spec) 檔案。buildspec 是組建命令和相關設定的集合，採用 YAML 格式，可供 CodeBuild 用來執行組建。如果沒有建置規格，CodeBuild 無法成功將建置輸入轉換為建置輸出，或在建置環境中尋找建置輸出成品，以上傳至輸出儲存貯體。

建立此檔案，將它命名為 buildspec.yml，然後將它儲存在根 (最上層) 目錄中。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

⚠ Important

因為建置規格宣告必須為有效的 YAML，因此建置規格宣告中的間距相當重要。若您建置規格宣告中的空格數與此不符，建置會立即失敗。您可以使用 YAML 驗證程式測試您的建置規格宣告是否為有效的 YAML。

📘 Note

您可以在建立建置專案時分別宣告建置命令，而非在您的來源碼中包含建置規格檔案。這在您希望使用不同建置命令建置來源碼，卻又不想要每次都更新您來源碼的儲存庫時會非常有用。如需詳細資訊，請參閱[Buildspec 語法](#)。

在此建置規格宣告中：

- `version` 代表要使用的建置規格標準版本。此建置規格宣告使用最新版本，`0.2`。
- `phases` 代表建置階段，您可以在其中指示 CodeBuild 執行命令。這些組建階段會在此以 `install`、`pre_build`、`build` 和 `post_build` 的形式列出。您無法變更這些組建階段名稱的拼字，也無法建立更多組建階段名稱。

在此範例中，在 `build` 階段期間，CodeBuild 會執行 `mvn install` 命令。此命令會指示 Apache Maven 編譯、測試，並將編譯過的 Java 類別檔案封裝到建置輸出成品中。為求完整，此範例中的每個建置階段內都置放了一些 `echo` 命令。當您稍後在本教學中檢視詳細的建置資訊時，這些 `echo` 命令的輸出可協助您更加了解 CodeBuild 如何執行命令以及執行順序。(雖然此範例中包含了所有組建階段，但若您不需要在其中一個階段執行任何命令，則可不包含該組建階段。) 對於每個建置階段，CodeBuild 會依列出的順序，從頭到尾執行每個指定的命令。

- `artifacts` 代表 CodeBuild 上傳到輸出儲存貯體的組建輸出成品。`files` 代表要包含在組建輸出中的檔案。CodeBuild 會上傳建置環境中 `target` 相對目錄中找到的單一 `messageUtil-1.0.jar` 檔案。檔案名稱 (`messageUtil-1.0.jar`) 及目錄名稱 (`target`) 是以 Apache Maven 建立及存放組建輸出成品的方式為基礎，僅適用於此範例。在您自己的組建中，檔案名稱及目錄可能會有所不同。

如需更多資訊，請參閱[Buildspec 參考](#)。

此時您的目錄結構看起來應該會如下。

(root directory name)

```
|-- pom.xml
|-- buildspec.yml
`-- src
    |-- main
    |     |-- java
    |     |-- MessageUtil.java
    |-- test
    |     |-- java
    |     |-- TestMessageUtil.java
```

步驟 3：建立兩個 S3 儲存貯體

(上一個步驟：[步驟 2：建立 buildspec 檔案](#))

雖然您可以在本教學課程中使用單一儲存貯體，但是使用兩個儲存貯體可讓查看組建輸入來源和組建輸出目標的過程變得更加簡單。

- 其中一個儲存貯體 (輸入儲存貯體) 會儲存組建輸入。在本教學課程中，此輸入儲存貯體的名稱為 `codebuild-region-ID-account-ID-input-bucket`，其中 *region-ID* 是儲存貯 AWS 體的區域，而 *account-ID* 是 AWS 您的帳戶 ID。
- 另一個儲存貯體 (輸出儲存貯體) 則會儲存組建輸出。在本教學課程中，此輸出儲存貯體的名稱為 `codebuild-region-ID-account-ID-output-bucket`。

如果您為這些儲存貯體選擇了不同的名稱，請務必在本教學課程中使用它們。

這兩個儲存貯體必須與組建位於相同的 AWS 區域。例如，如果您指示 CodeBuild 在美國東部（俄亥俄）區域執行組建，這些儲存貯體也必須位於美國東部（俄亥俄）區域。

如需詳細資訊，請參閱 Amazon Simple Storage Service 主控台使用者指南中的[建立儲存貯體](#)。

Note

雖然 CodeBuild 也支援儲存在 CodeCommit、GitHub 和 Bitbucket 儲存庫中的建置輸入，但本教學課程不會向您展示如何使用它們。如需詳細資訊，請參閱[規劃組建](#)。

步驟 4：上傳原始程式碼和 Buildspec 檔案

(上一個步驟：[步驟 3：建立兩個 S3 儲存貯體](#))

在此步驟中，您會將來源碼和組建規格檔案新增到輸入儲存貯體。

使用您作業系統的 zip 公用程式，建立名為 MessageUtil.zip 的檔案，其中包含 MessageUtil.java、TestMessageUtil.java、pom.xml 和 buildspec.yml。

MessageUtil.zip 檔案的目錄結構看起來必須如下。

```
MessageUtil.zip
|-- pom.xml
|-- buildspec.yml
`-- src
    |-- main
    |   |-- java
    |   |-- MessageUtil.java
    |-- test
    |   |-- java
    |   |-- TestMessageUtil.java
```

Important

請不要包含 (*root directory name*) 目錄，而是只有 (*root directory name*) 目錄中的目錄和檔案。

將 MessageUtil.zip 檔案上傳至名為 codebuild-*region-ID-account-ID*-input-bucket 的輸入儲存貯體。

Important

對於 CodeCommit、GitHub 和 Bitbucket 儲存庫，根據慣例，您必須在每個儲存庫的根（頂層）buildspec.yml 中存放名為 *buildspec.yml* 的組建規格檔案，或將組建規格宣告包含在組建專案定義中。請不要建立包含儲存庫來源碼和建置規格檔案的 ZIP 檔案。

(僅適用於存放在 S3 儲存貯體的建置輸入) 您必須建立 ZIP 檔案，其中包含來源碼，並且根據慣例，於根（最上層）包含名為 buildspec.yml 的建置規格檔案，或是將建置規格宣告其做為建置專案定義的一部分包含在其中。

若您想要針對建置規格檔案使用不同的名稱，或是想要參考位於根以外其他位置的建置規格，您可以指定建置規格覆寫，做為建置專案定義的一部分。如需詳細資訊，請參閱[Buildspec 檔案名稱和儲存位置](#)。

步驟 5：建立建置專案

(上一個步驟：[步驟 4：上傳原始程式碼和 Buildspec 檔案](#))

在此步驟中，您會建立 AWS CodeBuild 使用執行建置的建置專案。建置專案包含如何執行建置的相關資訊，包括取得原始程式碼的位置、要使用的建置環境、要執行的建置命令，以及儲存建置輸出的位置。建置環境代表作業系統、程式設計語言執行時間，以及 CodeBuild 用來執行建置之工具的組合。建置環境會以 Docker 影像表示。如需詳細資訊，請參閱 Docker Docs 網站上的 [Docker 概觀](#) 主題。

在此建置環境中，您會指示 CodeBuild 使用包含 Java 開發套件 (JDK) 和 Apache Maven 版本的 Docker 映像。

建立建置專案

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/codesuite/codebuild/home> 的 AWS CodeBuild 主控台。
2. 使用 AWS 區域選擇器選擇支援 CodeBuild AWS 的區域。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 [AWS CodeBuild 端點和配額](#)。
3. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。
4. 在 Create build project (建立組建專案) 頁面上，於 Project configuration (專案組態) 中，針對 Project name (專案名稱)，輸入此組建專案的名稱 (在此範例中為 codebuild-demo-project)。組建專案名稱在每個 AWS 帳戶中必須是唯一的。如果您使用不同名稱，請在此教學課程中都使用此名稱。

Note

在 Create build project (建立建置專案) 頁面上，您可能會看到與以下訊息相似的錯誤訊息：You are not authorized to perform this operation (您未獲得執行此操作的授權)。這很可能是因為您以沒有建立組建專案許可的使用者 AWS 管理主控台 身分登入。若要修正此問題，請登出 AWS 管理主控台，然後使用屬於下列其中一個 IAM 實體的登入資料重新登入：

- 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱《使用者指南》中的 [建立您的第一個 AWS 帳戶 根使用者和群組](#)。
- AWS 您帳戶中的使用者，其 AWSCodeBuildAdminAccess、AmazonS3ReadOnlyAccess 和 IAMFullAccess 受管政策連接到該使用者或該使用者所屬的 IAM 群組。如果您的 AWS 帳戶中沒有具有這些許可的使用者或群組，而且您無

法將這些許可新增至您的使用者或群組，請聯絡您的 AWS 帳戶管理員尋求協助。如需詳細資訊，請參閱[AWS 的 受管（預先定義）政策 AWS CodeBuild](#)。

這兩個選項都包含管理員許可，可讓您建立建置專案以完成本教學課程。建議您一律使用完成任務所需的最低許可。如需詳細資訊，請參閱[AWS CodeBuild 許可參考](#)。

5. 在來源中，針對來源提供者選擇 Amazon S3。
6. 針對儲存貯體，選擇 codebuild-**region-ID-account-ID**-input-bucket。
7. 針對 S3 object key (S3 物件金鑰)，輸入 **MessageUtil.zip**。
8. 在 Environment (環境) 中，針對 Environment image (環境映像)，請讓 Managed image (受管映像) 維持在選取狀態。
9. 針對作業系統，選擇 Amazon Linux。
10. 針對 Runtime(s) (執行時間)，選擇 Standard (標準)。
11. 針對映像，選擇 aws/codebuild/amazonlinux-x86_64-standard : corretto11。
12. 在 Service role (服務角色) 中，讓 New service role (新服務角色) 維持在選取狀態，然後讓 Role name (角色名稱) 維持不變。
13. 針對 Builds spec，將 Use a builds spec file (使用 builds spec 檔案) 維持在選取狀態。
14. 在成品中，針對類型，選擇 Amazon S3。
15. 針對儲存貯體名稱，選擇 codebuild-**region-ID-account-ID**-output-bucket。
16. 將 Name (名稱) 和 Path (路徑) 欄位保留空白。
17. 選擇 Create build project (建立建置專案)。

步驟 6：執行建置

(上一個步驟：[步驟 5：建立建置專案](#))

在此步驟中，您會 AWS CodeBuild 指示 使用組建專案中的設定來執行組建。

執行建置

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 在建置專案清單中，選擇 codebuild-demo-project，然後選擇開始建置。建置會立即開始。

步驟 7：檢視摘要建置資訊

(上一個步驟：[步驟 6：執行建置](#))

在此步驟中，您會檢視您組建狀態的摘要資訊。

檢視摘要建置資訊

1. 如果未顯示 codebuild-demo-project：**<build-ID>** 頁面，請在導覽列中選擇建置歷史記錄。接著，在組建專案清單中，針對專案，選擇 codebuild-demo-project 的組建執行連結。此處應該只會有一個相符的連結。(如果您之前已完成此教學課程，請在 Completed (已完成) 欄中選擇具有最新值的連結。)
2. 在建置狀態頁面上，在階段詳細資訊中，應該會顯示下列建置階段，並在狀態欄中成功：
 - SUBMITTED
 - QUEUED
 - PROVISIONING
 - DOWNLOAD_SOURCE
 - INSTALL
 - PRE_BUILD
 - BUILD
 - POST_BUILD
 - UPLOAD_ARTIFACTS
 - FINALIZING
 - COMPLETED (已完成)

在 Build Status (建置狀態) 中，應會顯示 Succeeded (成功)。

若您看到 In Progress (進行中)，請選擇重新整理按鈕。

3. 在每個組建階段旁，Duration (期間值) 值表示組建階段的持續時間長度。End time (結束時間) 值表示建置階段的結束時間。

步驟 8：檢視詳細建置資訊

(上一個步驟：[步驟 7：檢視摘要建置資訊](#))

在此步驟中，您會在 CloudWatch Logs 中檢視建置的詳細資訊。

Note

為了保護敏感資訊，CodeBuild 日誌中會隱藏下列項目：

- AWS 存取金鑰 IDs。如需詳細資訊，請參閱AWS Identity and Access Management 《使用者指南》中的[管理 IAM 使用者的存取金鑰](#)。
- 使用參數存放區指定的字串。如需詳細資訊，請參閱《Amazon EC2 Systems Manager Systems Manager 使用者指南》中的 [Systems Manager 參數存放區](#)和 [Systems Manager 參數存放區主控台演練](#)。
- 使用 指定的字串 AWS Secrets Manager。如需詳細資訊，請參閱[金鑰管理](#)。

檢視詳細建置資訊

1. 在先前步驟的建置詳細資訊頁面仍顯示的情況下，建置日誌的最後 10,000 行會顯示在 Build logs (建置日誌) 中。若要在 CloudWatch Logs 中查看整個建置日誌，請選擇檢視整個日誌連結。
2. 在 CloudWatch Logs 日誌串流中，您可以瀏覽日誌事件。根據預設，只會顯示最後一組日誌事件。若要查看更早的事件，請捲動到清單的開頭。
3. 在本教學課程中，大多數日誌事件都包含有關 CodeBuild 下載和安裝建置相依性檔案到其建置環境的詳細資訊，您可能不在乎。您可以使用 Filter events (篩選事件) 方塊，減少顯示的資訊。例如，如果您"[INFO]"在篩選事件中輸入，[INFO]則只會顯示包含的事件。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[篩選和模式語法](#)。

步驟 9：取得建置輸出成品

(上一個步驟：[步驟 8：檢視詳細建置資訊](#))

在此步驟中，您會取得 CodeBuild 建置並上傳至輸出儲存貯體messageUtil-1.0.jar的檔案。

您可以使用 CodeBuild 主控台或 Amazon S3 主控台來完成此步驟。

取得建置輸出成品AWS CodeBuild（主控台）

1. 在 CodeBuild 主控台仍開啟，且建置詳細資訊頁面仍顯示在上一個步驟中時，選擇建置詳細資訊索引標籤，然後向下捲動至成品區段。

Note

如果未顯示建置詳細資訊頁面，請在導覽列中選擇建置歷史記錄，然後選擇建置執行連結。

2. Amazon S3 資料夾的連結位於成品上傳位置下方。此連結會在 Amazon S3 中開啟 資料夾，您可以在其中找到messageUtil-1.0.jar建置輸出成品檔案。

取得建置輸出成品 (Amazon S3 主控台)

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 打開 codebuild-*region-ID-account-ID*-output-bucket。
3. 開啟 codebuild-demo-project 資料夾。
4. 開啟 target 資料夾，您會在其中找到 messageUtil-1.0.jar 建置輸出成品檔案。

步驟 10：刪除 S3 儲存貯體

(上一個步驟：[步驟 9：取得建置輸出成品](#))

若要避免持續向您的 AWS 帳戶收取費用，您可以刪除本教學課程中使用的輸入和輸出儲存貯體。如需說明，請參閱 [《Amazon Simple Storage Service 使用者指南》](#) 中的刪除或清空儲存貯體。

如果您使用 IAM 使用者或管理員 IAM 使用者來刪除這些儲存貯體，則使用者必須擁有更多存取許可。在標記 (### BEGIN ADDING STATEMENT HERE ### 及 ### END ADDING STATEMENTS HERE ###) 之間將下列陳述式新增至使用者的現有存取政策。

這個陳述式中的省略符號 (...) 是為了簡潔起見。請不要移除現有存取政策中的任何陳述式。請勿在政策中輸入這些省略符號。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:DeleteBucket",
        "s3:DeleteObject"
    ],
    "Resource": "*"
}
]
```

包裝

在本教學課程中，您曾使用 AWS CodeBuild 將一組 Java 類別檔案建置至 JAR 檔案。您接著檢視了建置的結果。

您現在可以在自己的案例中嘗試使用 CodeBuild。請遵循中的說明進行[規劃組建](#)。若您覺得尚未準備就緒，建議您嘗試組建我們的一些範例。如需詳細資訊，請參閱[CodeBuild 使用以案例為基礎的範例](#)。

AWS CodeBuild 開始使用 AWS CLI

在本教學課程中，您會使用 AWS CodeBuild 將範例原始程式碼輸入檔案的集合（稱為建置輸入成品或建置輸入）建置為原始程式碼的可部署版本（稱為建置輸出成品或建置輸出）。具體而言，您會指示 CodeBuild 使用常見的建置工具 Apache Maven，將一組 Java 類別檔案建置至 Java Archive (JAR) 檔案。您不需要熟悉 Apache Maven 或 Java，也能完成本教學課程。

您可以透過 CodeBuild 主控台、AWS CodePipeline、AWS CLI 或 AWS SDKs 使用 CodeBuild。本教學課程示範如何搭配使用 CodeBuild AWS CLI。如需使用 CodePipeline 的資訊，請參閱[搭配 CodePipeline 使用 CodeBuild CodePipeline](#)。

Important

本教學中的步驟要求您建立資源（例如 S3 儲存貯體），這可能會導致 AWS 您的帳戶產生費用。這包括 CodeBuild 以及與 Amazon S3 相關的 AWS 資源和動作 AWS KMS，以及 CloudWatch Logs 的可能費用。如需詳細資訊，請參閱[CodeBuild 定價](#)、[Amazon S3 定價](#)、[AWS Key Management Service 定價](#)和 [Amazon CloudWatch 定價](#)。

主題

- [步驟 1：建立原始程式碼](#)
- [步驟 2：建立 buildspec 檔案](#)

- [步驟 3：建立兩個 S3 儲存貯體](#)
- [步驟 4：上傳原始程式碼和 Buildspec 檔案](#)
- [步驟 5：建立建置專案](#)
- [步驟 6：執行建置](#)
- [步驟 7：檢視摘要建置資訊](#)
- [步驟 8：檢視詳細建置資訊](#)
- [步驟 9：取得建置輸出成品](#)
- [步驟 10：刪除 S3 儲存貯體](#)
- [包裝](#)

步驟 1：建立原始程式碼

(部分：[AWS CodeBuild 開始使用 AWS CLI](#))

在此步驟中，您會建立您希望 CodeBuild 建置到輸出儲存貯體的原始程式碼。此來源碼由兩個 Java 類別檔案及一個 Apache Maven Project Object Model (POM) 檔案組成。

1. 在您本機電腦或執行個體上的空目錄內，建立此目錄結構。

```
(root directory name)
|-- src
|   |-- main
|       |-- java
|       |-- test
|           |-- java
```

2. 使用您選擇的文字編輯器建立此檔案，將它命名為 MessageUtil.java，然後將它儲存在 src/main/java 目錄中。

```
public class MessageUtil {
    private String message;

    public MessageUtil(String message) {
        this.message = message;
    }

    public String printMessage() {
        System.out.println(message);
    }
}
```

```
        return message;
    }

    public String salutationMessage() {
        message = "Hi!" + message;
        System.out.println(message);
        return message;
    }
}
```

此類別檔案會將傳遞給它的字元字串做為輸出建立。MessageUtil 建構函數會設定字元字串。printMessage 方法會建立輸出。salutationMessage 方法會輸出 Hi!，其後跟隨字元字串。

3. 建立此檔案，將它命名為 TestMessageUtil.java，然後將它儲存在 /src/test/java 目錄中。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

    String message = "Robert";
    MessageUtil messageUtil = new MessageUtil(message);

    @Test
    public void testPrintMessage() {
        System.out.println("Inside testPrintMessage()");
        assertEquals(message,messageUtil.printMessage());
    }

    @Test
    public void testSalutationMessage() {
        System.out.println("Inside testSalutationMessage()");
        message = "Hi!" + "Robert";
        assertEquals(message,messageUtil.salutationMessage());
    }
}
```

此類別檔案會將 MessageUtil 類別中的 message 變數設為 Robert。它接著會透過檢查 Robert 和 Hi!Robert 字串是否出現在輸出中，來測試 message 變數是否設定成功。

4. 建立此檔案，將它命名為 `pom.xml`，然後將它儲存在根 (最上層) 目錄中。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven 會使用此檔案中的說明，將 `MessageUtil.java` 和 `TestMessageUtil.java` 轉換成名為 `messageUtil-1.0.jar` 的檔案，然後執行指定的測試。

此時您的目錄結構看起來應該會如下。

(root directory name)

```
|-- pom.xml
|-- src
|   |-- main
|       |-- java
|           |-- MessageUtil.java
```

```
    `-- test
        `-- java
            `-- TestMessageUtil.java
```

步驟 2：建立 buildspec 檔案

(上一個步驟：[步驟 1：建立原始程式碼](#))

在此步驟中，您會建立組建規格 (build spec) 檔案。buildspec 是組建命令和相關設定的集合，採用 YAML 格式，可供 CodeBuild 用來執行組建。如果沒有建置規格，CodeBuild 無法成功將建置輸入轉換為建置輸出，或在建置環境中尋找建置輸出成品，以上傳至輸出儲存貯體。

建立此檔案，將它命名為 buildspec.yml，然後將它儲存在根 (最上層) 目錄中。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

Important

因為建置規格宣告必須為有效的 YAML，因此建置規格宣告中的間距相當重要。若您建置規格宣告中的空格數與此不符，建置會立即失敗。您可以使用 YAML 驗證程式測試您的建置規格宣告是否為有效的 YAML。

Note

您可以在建立建置專案時分別宣告建置命令，而非在您的來源碼中包含建置規格檔案。這在您希望使用不同建置命令建置來源碼，卻又不想要每次都更新您來源碼的儲存庫時會非常有用。如需詳細資訊，請參閱[Buildspec 語法](#)。

在此建置規格宣告中：

- `version` 代表要使用的建置規格標準版本。此建置規格宣告使用最新版本，`0.2`。
- `phases` 代表建置階段，您可以在其中指示 CodeBuild 執行命令。這些組建階段會在此以 `install`、`pre_build`、`build` 和 `post_build` 的形式列出。您無法變更這些組建階段名稱的拼字，也無法建立更多組建階段名稱。

在此範例中，在 `build` 階段期間，CodeBuild 會執行 `mvn install` 命令。此命令會指示 Apache Maven 編譯、測試，並將編譯過的 Java 類別檔案封裝到建置輸出成品中。為求完整，此範例中的每個建置階段內都置放了一些 `echo` 命令。當您稍後在本教學中檢視詳細的建置資訊時，這些 `echo` 命令的輸出可協助您更加了解 CodeBuild 如何執行命令以及執行順序。(雖然此範例中包含了所有組建階段，但若您不需要在其中一個階段執行任何命令，則可不必包含該組建階段。) 對於每個建置階段，CodeBuild 會依序從頭到尾執行每個指定的命令，一次一個。

- `artifacts` 代表 CodeBuild 上傳至輸出儲存貯體的組組建輸出成品。 `files` 代表要包含在組建輸出中的檔案。CodeBuild 會上傳建置環境中 `target` 相對目錄中找到的單一 `messageUtil-1.0.jar` 檔案。檔案名稱 (`messageUtil-1.0.jar`) 及目錄名稱 (`target`) 是以 Apache Maven 建立及存放組建輸出成品的方式為基礎，僅適用於此範例。在您自己的組建中，檔案名稱及目錄可能會有所不同。

如需更多資訊，請參閱[Buildspec 參考](#)。

此時您的目錄結構看起來應該會如下。

```
(root directory name)
|-- pom.xml
|-- buildspec.yml
`-- src
    |-- main
    |     |-- java
    |         |-- MessageUtil.java
    |-- test
    |     |-- java
```

```
`-- TestMessageUtil.java
```

步驟 3：建立兩個 S3 儲存貯體

(上一個步驟：[步驟 2：建立 buildspec 檔案](#))

雖然您可以在本教學課程中使用單一儲存貯體，但是使用兩個儲存貯體可讓查看組建輸入來源和組建輸出目標的過程變得更為簡單。

- 其中一個儲存貯體 (輸入儲存貯體) 會儲存組建輸入。在本教學課程中，此輸入儲存貯體的名稱為 `codebuild-region-ID-account-ID-input-bucket`，其中 *region-ID* 是儲存貯 AWS 體的區域，而 *account-ID* 是 AWS 您的帳戶 ID。
- 另一個儲存貯體 (輸出儲存貯體) 則會儲存組建輸出。在本教學課程中，此輸出儲存貯體的名稱為 `codebuild-region-ID-account-ID-output-bucket`。

如果您為這些儲存貯體選擇了不同的名稱，請務必在本教學課程中使用它們。

這兩個儲存貯體必須與建置位於相同的 AWS 區域。例如，如果您指示 CodeBuild 在美國東部（俄亥俄）區域執行組建，這些儲存貯體也必須位於美國東部（俄亥俄）區域。

如需詳細資訊，請參閱 Amazon Simple Storage Service 主控台使用者指南中的[建立儲存貯體](#)。

Note

雖然 CodeBuild 也支援儲存在 CodeCommit、GitHub 和 Bitbucket 儲存庫中的建置輸入，但本教學課程不會向您展示如何使用它們。如需詳細資訊，請參閱[規劃組建](#)。

步驟 4：上傳原始程式碼和 Buildspec 檔案

(上一個步驟：[步驟 3：建立兩個 S3 儲存貯體](#))

在此步驟中，您會將來源碼和組建規格檔案新增到輸入儲存貯體。

使用您作業系統的 zip 公用程式，建立名為 `MessageUtil.zip` 的檔案，其中包含 `MessageUtil.java`、`TestMessageUtil.java`、`pom.xml` 和 `buildspec.yml`。

`MessageUtil.zip` 檔案的目錄結構看起來必須如下。

```
MessageUtil.zip
```

```
|-- pom.xml
|-- buildspec.yml
`-- src
    |-- main
    |     |-- java
    |     |-- MessageUtil.java
    |-- test
    |     |-- java
    |     |-- TestMessageUtil.java
```

⚠ Important

請不要包含 (*root directory name*) 目錄，而是只有 (*root directory name*) 目錄中的目錄和檔案。

將 MessageUtil.zip 檔案上傳至名為 codebuild-*region-ID-account-ID*-input-bucket 的輸入儲存貯體。

⚠ Important

對於 CodeCommit、GitHub 和 Bitbucket 儲存庫，根據慣例，您必須在每個儲存庫的根（頂層）buildspec.yml 中存放名為的組建規格檔案，或將組建規格宣告包含在組建專案定義中。請不要建立包含儲存庫來源碼和建置規格檔案的 ZIP 檔案。

(僅適用於存放在 S3 儲存貯體的建置輸入) 您必須建立 ZIP 檔案，其中包含來源碼，並且根據慣例，於根（最上層）包含名為 buildspec.yml 的建置規格檔案，或是將建置規格宣告其做為建置專案定義的一部分包含在其中。

若您想要針對建置規格檔案使用不同的名稱，或是想要參考位於根以外其他位置的建置規格，您可以指定建置規格覆寫，做為建置專案定義的一部分。如需詳細資訊，請參閱[Buildspec 檔案名稱和儲存位置](#)。

步驟 5：建立建置專案

(上一個步驟：[步驟 4：上傳原始程式碼和 Buildspec 檔案](#))

在此步驟中，您會建立 AWS CodeBuild 使用執行建置的建置專案。建置專案包含如何執行建置的相關資訊，包括取得原始程式碼的位置、要使用的建置環境、要執行的建置命令，以及儲存建置輸出的位

置。建置環境代表作業系統、程式設計語言執行時間，以及 CodeBuild 用來執行建置之工具的組合。建置環境會以 Docker 影像表示。如需詳細資訊，請參閱 Docker Docs 網站上的 [Docker 概觀](#) 主題。

在此建置環境中，您會指示 CodeBuild 使用包含 Java 開發套件 (JDK) 和 Apache Maven 版本的 Docker 映像。

建立建置專案

1. 使用 AWS CLI 執行 create-project 命令：

```
aws codebuild create-project --generate-cli-skeleton
```

即會在輸出中顯示 JSON 格式化資料。將資料複製到 AWS CLI 安裝 create-project.json 的本機電腦或執行個體上名為 `create-project.json` 的檔案。若您選擇使用不同的檔案名稱，請務必在本教學課程範圍中使用它。

修改複製的資料，遵循此格式，並儲存您的結果：

```
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```

將 *serviceIAMRole* 取代為 CodeBuild 服務角色的 Amazon Resource Name (ARN) (例如 `arn:aws:iam::account-ID:role/role-name`)。若要建立服務角色，請參閱 [允許 CodeBuild 與其他 AWS 服務互動](#)。

在此資料中：

- `name` 代表此組建專案的必要識別符 (在此範例中為 `codebuild-demo-project`)。組建專案名稱在您帳戶內的所有組建專案中都必須是唯一的。
- 對於 `source`，`type` 是代表來源碼儲存庫類型的必要值 (在此範例中 `S3` 為 Amazon S3 儲存貯體)。
- 針對 `source`，`location` 代表來源碼的路徑 (在此範例中為輸入儲存貯體名稱，其後跟隨 ZIP 檔案名稱)。
- 對於 `artifacts`，`type` 是代表建置輸出成品儲存庫類型的必要值 (在此範例中 `S3` 為 Amazon S3 儲存貯體)。
- 針對 `artifacts`，`location` 代表您先前建立或找到的輸出儲存貯體名稱 (在此範例中為 `codebuild-region-ID-account-ID-output-bucket`)。
- 對於 `environment`，`type` 是代表建置環境類型的必要值 (在此範例中為 `LINUX_CONTAINER`)。
- 對於 `environment`，`image` 是代表此建置專案使用的 Docker 映像名稱和標籤組合的必要值，如 Docker 映像儲存庫類型所指定 (在此範例中 `aws/codebuild/standard:5.0` 為 CodeBuild Docker 映像儲存庫中的 Docker 映像)。 `aws/codebuild/standard` 是 Docker 映像的名稱。 `5.0` 是 Docker 映像的標籤。

若要尋找更多您可以用於案例中的 Docker 影像，請參閱[建置環境參考](#)。

- 對於 `environment`，`computeType` 是代表 CodeBuild 使用之運算資源的必要值 (在此範例中為 `BUILD_GENERAL1_SMALL`)。

Note

其他原始 JSON 格式資料中的可用值，例如 `description`、`buildspec`、`auth` (包含 `type` 和 `resource`)、`path`、`namespaceType`、`name` (適用於 `artifacts`)、`packaging`、`environmentVariables` (包含 `name` 和 `value`)、`timeoutInMinutes`、`encryptionKey` 和 `tags` (包含 `key` 和 `value`) 為選擇性。本教學課程中不會使用到它們，因此不會在此顯示。如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。

2. 切換到包含您儲存檔案的目錄，然後再次執行 `create-project` 命令。

```
aws codebuild create-project --cli-input-json file://create-project.json
```

若執行成功，則會在輸出中顯示與下列內容相似的資料。

```
{
  "project": {
    "name": "codebuild-demo-project",
    "serviceRole": "serviceIAMRole",
    "tags": [],
    "artifacts": {
      "packaging": "NONE",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "message-util.zip"
    },
    "lastModified": 1472661575.244,
    "timeoutInMinutes": 60,
    "created": 1472661575.244,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:5.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-project"
  }
}
```

- `project` 代表此組建專案的相關資訊。
- `tags` 代表宣告的任何標籤。
- `packaging` 代表組建輸出成品存放在輸出儲存貯體的方式。NONE 表示會在輸出儲存貯體內建立一個資料夾。組建輸出成品會存放在該資料夾中。
- `lastModified` 代表時間 (以 Unix 時間格式表示)，代表組建專案相關資訊最後變更的時間。
- `timeoutInMinutes` 代表如果建置尚未完成，則 CodeBuild 停止建置的分鐘數。(預設為 60 分鐘。)
- `created` 代表時間 (以 Unix 時間格式表示)，代表建置專案的建立時間。

- `environmentVariables` 代表任何已宣告且可供 CodeBuild 在建置期間使用的環境變數。
- `encryptionKey` 代表 CodeBuild 用來加密建置輸出成品之客戶受管金鑰的 ARN。
- `arn` 代表組建專案的 ARN。

Note

執行 `create-project` 命令後，可能會輸出類似下列的錯誤訊息：使用者：***user-ARN*** 未獲授權執行：`Codebuild: CreateProject`。這很可能是因為您使用沒有足夠的許可來使用 CodeBuild 建立建置專案的使用者登入資料 AWS CLI 來設定。若要修正此問題，請使用 AWS CLI 屬於下列其中一個 IAM 實體的登入資料來設定：

- 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱《使用者指南》中的[建立您的第一個 AWS 帳戶 根使用者和群組](#)。
- AWS 您帳戶中的使用者，其 `AWSCodeBuildAdminAccess`、`AmazonS3ReadOnlyAccess` 和 `IAMFullAccess` 受管政策連接到該使用者或該使用者所屬的 IAM 群組。如果您的 AWS 帳戶中沒有具有這些許可的使用者或群組，而且您無法將這些許可新增至您的使用者或群組，請聯絡您的 AWS 帳戶管理員尋求協助。如需詳細資訊，請參閱[AWS 的 受管（預先定義）政策 AWS CodeBuild](#)。

步驟 6：執行建置

(上一個步驟：[步驟 5：建立建置專案](#))

在此步驟中，您會 AWS CodeBuild 指示 使用組建專案中的設定執行組建。

執行建置

1. 使用 AWS CLI 執行 `start-build` 命令：

```
aws codebuild start-build --project-name project-name
```

使用您在先前步驟中的組建專案名稱取代 *project-name* (例如，`codebuild-demo-project`)。

2. 若執行成功，則會在輸出中顯示與下列內容相似的資料：

```
{
```

```
"build": {
  "buildComplete": false,
  "initiator": "user-name",
  "artifacts": {
    "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/
message-util.zip"
  },
  "projectName": "codebuild-demo-project",
  "timeoutInMinutes": 60,
  "buildStatus": "IN_PROGRESS",
  "environment": {
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/standard:5.0",
    "type": "LINUX_CONTAINER",
    "environmentVariables": []
  },
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "currentPhase": "SUBMITTED",
  "startTime": 1472848787.882,
  "id": "codebuild-demo-project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE",
  "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE"
}
```

- build 代表此組建的相關資訊。
 - buildComplete 代表建置完成 (true)。否則為 false。
 - initiator 代表啟動建置的實體。
 - artifacts 代表建置輸出的相關資訊，包括其位置。
 - projectName 代表建置專案的名稱。
 - buildStatus 代表 start-build 命令執行時目前的組建狀態。
 - currentPhase 代表 start-build 命令執行時目前的組建階段。
 - startTime 代表時間 (以 Unix 時間格式表示)，代表建置程序的啟動時間。
 - id 代表組建的 ID。
 - arn 代表組建的 ARN。

記下 `id` 值。下一個步驟需要此值。

步驟 7：檢視摘要建置資訊

(上一個步驟：[步驟 6：執行建置](#))

在此步驟中，您會檢視您組建狀態的摘要資訊。

檢視摘要建置資訊

- 使用 AWS CLI 執行 `batch-get-builds` 命令。

```
aws codebuild batch-get-builds --ids id
```

將 *ID* 取代為上一個步驟的輸出中出現 `id` 的值。

若執行成功，則會在輸出中顯示與下列內容相似的資料。

```
{
  "buildsNotFound": [],
  "builds": [
    {
      "buildComplete": true,
      "phases": [
        {
          "phaseStatus": "SUCCEEDED",
          "endTime": 1472848788.525,
          "phaseType": "SUBMITTED",
          "durationInSeconds": 0,
          "startTime": 1472848787.882
        },
        ... The full list of build phases has been omitted for brevity ...
        {
          "phaseType": "COMPLETED",
          "startTime": 1472848878.079
        }
      ],
      "logs": {
        "groupName": "/aws/codebuild/codebuild-demo-project",
```

```

    "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
    "streamName": "38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
  },
  "artifacts": {
    "md5sum": "MD5-hash",
    "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/message-util.zip",
    "sha256sum": "SHA-256-hash"
  },
  "projectName": "codebuild-demo-project",
  "timeoutInMinutes": 60,
  "initiator": "user-name",
  "buildStatus": "SUCCEEDED",
  "environment": {
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/standard:5.0",
    "type": "LINUX_CONTAINER",
    "environmentVariables": []
  },
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "currentPhase": "COMPLETED",
  "startTime": 1472848787.882,
  "endTime": 1472848878.079,
  "id": "codebuild-demo-project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
  "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
}
]
}

```

- `buildsNotFound` 代表無法取得資訊之任何組建的組建 ID。在此範例中，此處應為空白。
- `builds` 代表可取得資訊的建置相關資訊。在此範例中，輸出內應該會只顯示一個建置相關資訊。
- `phases` 代表組建過程中 CodeBuild 執行的組建置階段。每個組建階段的相關資訊會分別以 `startTime`、`endTime` 和 `durationInSeconds` (組建階段啟動和結束的時間 (以 Unix 時間格式表示)，以及階段持續的時間長度 (以秒數表示)) 列出，以及 `phaseType` (例如 SUBMITTED、PROVISIONING、DOWNLOAD_SOURCE、INSTALL、PRE_BUILD、BUILD、POST_BUILD)

或 COMPLETED) 及 phaseStatus (例如 SUCCEEDED、FAILED、FAULT、TIMED_OUT、IN_PROGRESS 或 STOPPED)。當您第一次執行 batch-get-builds 命令時，可能不會有太多 (甚至完全沒有) 階段。在後續使用相同的組建 ID 執行 batch-get-builds 命令後，輸出內便會出現更多組建階段。

- logs 代表 Amazon CloudWatch Logs 中建置日誌的相關資訊。
- md5sum 和 sha256sum 代表組建輸出成品的 MD5 和 SHA-256 雜湊。只有在組建專案的 packaging 值設為 ZIP 時，這些才會出現在輸出中。(您未在本教學課程中設定此值。) 您可以搭配檢查總和工具使用這些雜湊，確認檔案完整性及真確性。

Note

您也可以使用 Amazon S3 主控台來檢視這些雜湊。選取建置輸出成品旁的方塊，選擇 Actions (動作)，然後選擇 Properties (屬性)。在 Properties (屬性) 窗格中，展開 Metadata (中繼資料)，然後檢視 x-amz-meta-codebuild-content-md5 和 x-amz-meta-codebuild-content-sha256 的值。(在 Amazon S3 主控台中，組建輸出成品的 ETag 值不應解譯為 MD5 或 SHA-256 雜湊。)

如果您使用 AWS SDKs 取得這些雜湊，則值會命名為 codebuild-content-md5 和 codebuild-content-sha256。

- endTime 代表時間 (以 Unix 時間格式表示)，代表建置程序的結束時間。

Note

Amazon S3 中繼資料具有名為的 CodeBuild 標頭 x-amz-meta-codebuild-buildarn，其中包含將成品發佈至 Amazon S3 buildArn 的 CodeBuild 組建的。buildArn 新增以允許來源追蹤通知，並參考產生成品的建置。

步驟 8：檢視詳細建置資訊

(上一個步驟：[步驟 7：檢視摘要建置資訊](#))

在此步驟中，您會在 CloudWatch Logs 中檢視建置的詳細資訊。

Note

為了保護敏感資訊，CodeBuild 日誌中會隱藏下列項目：

- AWS 存取金鑰 IDs。如需詳細資訊，請參閱AWS Identity and Access Management 《使用者指南》中的[管理 IAM 使用者的存取金鑰](#)。
- 使用參數存放區指定的字串。如需詳細資訊，請參閱《Amazon EC2 Systems Manager Systems Manager 使用者指南》中的 [Systems Manager 參數存放區](#)和 [Systems Manager 參數存放區主控台演練](#)。
- 使用 指定的字串 AWS Secrets Manager。如需詳細資訊，請參閱[金鑰管理](#)。

檢視詳細建置資訊

1. 使用您的 Web 瀏覽器前往先前步驟輸出中顯示的 deepLink 位置 (例如, `https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE`)。
2. 在 CloudWatch Logs 日誌串流中，您可以瀏覽日誌事件。根據預設，只會顯示最後一組日誌事件。若要查看更早的事件，請捲動到清單的開頭。
3. 在本教學課程中，大多數日誌事件都包含有關 CodeBuild 下載和安裝建置相依性檔案到其建置環境的詳細資訊，您可能不在乎。您可以使用 Filter events (篩選事件) 方塊，減少顯示的資訊。例如，如果您 "[INFO]" 在篩選事件中輸入，[INFO] 則只會顯示包含的事件。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[篩選和模式語法](#)。

CloudWatch Logs 日誌串流的這些部分與本教學課程相關。

```
...
[Container] 2016/04/15 17:49:42 Entering phase PRE_BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Phase complete: PRE_BUILD Success: true
[Container] 2016/04/15 17:49:42 Entering phase BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering build phase...
[Container] 2016/04/15 17:49:42 Entering build phase...
[Container] 2016/04/15 17:49:42 Running command mvn install
[Container] 2016/04/15 17:49:44 [INFO] Scanning for projects...
[Container] 2016/04/15 17:49:44 [INFO]
[Container] 2016/04/15 17:49:44 [INFO]
-----
[Container] 2016/04/15 17:49:44 [INFO] Building Message Utility Java Sample App 1.0
```

```
[Container] 2016/04/15 17:49:44 [INFO]
-----
...
[Container] 2016/04/15 17:49:55
-----
[Container] 2016/04/15 17:49:55 T E S T S
[Container] 2016/04/15 17:49:55
-----
[Container] 2016/04/15 17:49:55 Running TestMessageUtil
[Container] 2016/04/15 17:49:55 Inside testSalutationMessage()
[Container] 2016/04/15 17:49:55 Hi!Robert
[Container] 2016/04/15 17:49:55 Inside testPrintMessage()
[Container] 2016/04/15 17:49:55 Robert
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time
elapsed: 0.018 sec
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Results :
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
...
[Container] 2016/04/15 17:49:56 [INFO]
-----
[Container] 2016/04/15 17:49:56 [INFO] BUILD SUCCESS
[Container] 2016/04/15 17:49:56 [INFO]
-----
[Container] 2016/04/15 17:49:56 [INFO] Total time: 11.845 s
[Container] 2016/04/15 17:49:56 [INFO] Finished at: 2016-04-15T17:49:56+00:00
[Container] 2016/04/15 17:49:56 [INFO] Final Memory: 18M/216M
[Container] 2016/04/15 17:49:56 [INFO]
-----
[Container] 2016/04/15 17:49:56 Phase complete: BUILD Success: true
[Container] 2016/04/15 17:49:56 Entering phase POST_BUILD
[Container] 2016/04/15 17:49:56 Running command echo Entering post_build phase...
[Container] 2016/04/15 17:49:56 Entering post_build phase...
[Container] 2016/04/15 17:49:56 Phase complete: POST_BUILD Success: true
[Container] 2016/04/15 17:49:57 Preparing to copy artifacts
[Container] 2016/04/15 17:49:57 Assembling file list
[Container] 2016/04/15 17:49:57 Expanding target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Found target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Creating zip artifact
```

在此範例中，CodeBuild 已成功完成建置前、建置後建置階段。它執行了單位測試並成功建置了 messageUtil-1.0.jar 檔案。

步驟 9：取得建置輸出成品

(上一個步驟：[步驟 8：檢視詳細建置資訊](#))

在此步驟中，您會取得 CodeBuild 建置並上傳至輸出儲存貯體messageUtil-1.0.jar的檔案。

您可以使用 CodeBuild 主控台或 Amazon S3 主控台來完成此步驟。

取得建置輸出成品AWS CodeBuild（主控台）

1. 在 CodeBuild 主控台仍開啟，且建置詳細資訊頁面仍顯示在上一個步驟中時，選擇建置詳細資訊索引標籤，然後向下捲動至成品區段。

Note

如果未顯示建置詳細資訊頁面，請在導覽列中選擇建置歷史記錄，然後選擇建置執行連結。

2. Amazon S3 資料夾的連結位於成品上傳位置下方。此連結會在 Amazon S3 中開啟 資料夾，您可以在其中找到messageUtil-1.0.jar建置輸出成品檔案。

取得建置輸出成品 (Amazon S3 主控台)

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 打開 codebuild-*region-ID-account-ID*-output-bucket。
3. 開啟 codebuild-demo-project 資料夾。
4. 開啟 target 資料夾，您會在其中找到 messageUtil-1.0.jar 建置輸出成品檔案。

步驟 10：刪除 S3 儲存貯體

(上一個步驟：[步驟 9：取得建置輸出成品](#))

若要避免持續向您的 AWS 帳戶收取費用，您可以刪除本教學課程中使用的輸入和輸出儲存貯體。如需說明，請參閱 [《Amazon Simple Storage Service 使用者指南》](#) 中的刪除或清空儲存貯體。

如果您使用 IAM 使用者或管理員 IAM 使用者來刪除這些儲存貯體，則使用者必須擁有更多存取許可。在標記 (### BEGIN ADDING STATEMENT HERE ### 及 ### END ADDING STATEMENTS HERE ###) 之間將下列陳述式新增至使用者的現有存取政策。

這個陳述式中的省略符號 (...) 是為了簡潔起見。請不要移除現有存取政策中的任何陳述式。請勿在政策中輸入這些省略符號。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

包裝

在本教學課程中，您曾使用 AWS CodeBuild 將一組 Java 類別檔案建置至 JAR 檔案。您接著檢視了建置的結果。

您現在可以在自己的案例中嘗試使用 CodeBuild。請遵循中的說明進行[規劃組建](#)。若您覺得尚未準備就緒，建議您嘗試組建我們的一些範例。如需詳細資訊，請參閱[CodeBuild 使用以案例為基礎的範例](#)。

CodeBuild 使用以案例為基礎的範例

您可以使用這些以使用案例為基礎的範例來實驗 AWS CodeBuild：

[跨服務範例](#)

要實驗的跨服務範例清單 AWS CodeBuild。

[建置徽章範例](#)

顯示如何使用建置徽章設定 CodeBuild。

[測試報告範例](#)

使用 AWS CLI 來建立、執行和檢視測試報告的結果。

[CodeBuild 的 Docker 範例](#)

示範如何使用自訂 Docker 映像、將 Docker 映像發佈至 Amazon ECR 中的儲存庫，以及在私有登錄檔中使用 Docker 映像。

[在 S3 儲存貯體中託管建置輸出](#)

示範如何使用未加密的建置成品，在 S3 儲存貯體中建立靜態網站。

[多個輸入和輸出範例](#)

示範如何在組建專案中使用多個輸入來源和多個輸出成品。

[平行測試執行範例](#)

示範如何使用 `codebuild-tests-run` CLI 命令在平行執行環境中分割和執行測試。

[Buildspec 檔案範例中的執行時間版本](#)

顯示如何在 buildspec 檔案中指定執行時間及其版本。

[來源版本範例](#)

顯示如何在 CodeBuild 組建專案中使用特定版本的來源。

[CodeBuild 的第三方來源儲存庫範例](#)

示範如何使用 CodeBuild 使用 Webhook 建立 BitBucket、GitHub Enterprise Server 和 GitHub 提取請求。

[使用語意版本控制在建置時間設定成品名稱](#)

示範在建置時如何使用語意版本控制來建立成品名稱。

CodeBuild 的跨服務範例

您可以使用這些跨服務範例來實驗 AWS CodeBuild：

[Amazon ECR 範例](#)

使用 Amazon ECR 儲存庫中的 Docker 映像，使用 Apache Maven 產生單一 JAR 檔案。範例說明將示範如何建立 Docker 映像並將其推送至 Amazon ECR、建立 Go 專案、建置專案、執行專案，以及設定許可，以允許 CodeBuild 連線至 Amazon ECR。

[Amazon EFS 範例](#)

示範如何設定 buildspec 檔案，以便 CodeBuild 專案在 Amazon EFS 檔案系統上掛載和建置。範例說明將示範如何建立 Amazon VPC、在 Amazon VPC 中建立檔案系統、建立和建置使用 Amazon VPC 的專案，然後檢閱產生的專案檔案和變數。

[AWS CodePipeline 範例](#)

示範如何使用 AWS CodePipeline 建立具有批次建置的組建，以及多個輸入來源和多個輸出成品。本節中包含的範例 JSON 檔案會顯示管道結構，這些結構會建立具有個別成品和合併成品的批次建置。提供額外的 JSON 範例，顯示具有多個輸入來源和多個輸出成品的管道結構。

[AWS Config 範例](#)

顯示如何設定 AWS Config。列出要追蹤哪些 CodeBuild 資源，並說明如何查詢 CodeBuild 專案 AWS Config。範例說明將向您顯示與整合的先決條件 AWS Config、設定步驟 AWS Config，以及查詢 CodeBuild 專案和資料的步驟 AWS Config。

[建置通知範例](#)

使用 Apache Maven 來產生單一 JAR 檔案。傳送建置通知給 Amazon SNS 主題的訂閱者。範例指示說明如何設定許可，以便 CodeBuild 可以與 Amazon SNS 和 CloudWatch 通訊、如何在 Amazon SNS 中建立和識別 CodeBuild 主題、如何訂閱主題的收件人，以及如何在 CloudWatch 中設定規則。

CodeBuild 的 Amazon ECR 範例

此範例使用 Amazon Elastic Container Registry (Amazon ECR) 映像儲存庫中的 Docker 映像來建置範例 Go 專案。

⚠ Important

執行此範例可能會導致 AWS 您的帳戶產生費用。這包括 AWS CodeBuild 與 Amazon S3、AWS KMS CloudWatch Logs 和 Amazon ECR 相關的 AWS 資源和動作的可能費用。如需詳細資訊，請參閱 [CodeBuild 定價](#)、[Amazon S3 定價](#)、[AWS Key Management Service 定價](#)、[Amazon CloudWatch 定價](#)和 [Amazon Elastic Container Registry 定價](#)。

主題

- [執行 Amazon ECR 範例](#)

執行 Amazon ECR 範例

使用下列指示來執行 CodeBuild 的 Amazon ECR 範例。

如何執行此範例

1. 若要在 Amazon ECR 中建立 Docker 映像並將其推送至映像儲存庫，請完成 [執行「發佈 Docker 映像到 Amazon ECR」範例](#)區段中的步驟 [「將 Docker 映像發佈至 Amazon ECR」範例](#)。
2. 建立 Go 專案：
 - a. 如本主題的 [Go 專案結構](#)和 [Go 專案檔案](#)章節所述建立檔案，然後將其上傳至 S3 輸入儲存貯體或 AWS CodeCommit、GitHub 或 Bitbucket 儲存庫。

⚠ Important

請勿上傳 *(root directory name)*，僅上傳 *(root directory name)* 內的檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含這些檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 *(root directory name)* 新增至 ZIP 檔案，僅新增 *(root directory name)* 內的檔案即可。

- b. 建立組建專案、執行組建，以及檢視相關的組建資訊。

如果您使用 AWS CLI 來建立建置專案，create-project命令的 JSON 格式輸入可能會與此類似。(以您自己的值取代預留位置。)

```
{
```

```

"name": "sample-go-project",
"source": {
  "type": "S3",
  "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
},
"artifacts": {
  "type": "S3",
  "location": "codebuild-region-ID-account-ID-output-bucket",
  "packaging": "ZIP",
  "name": "GoOutputArtifact.zip"
},
"environment": {
  "type": "LINUX_CONTAINER",
  "image": "aws/codebuild/standard:5.0",
  "computeType": "BUILD_GENERAL1_SMALL"
},
"serviceRole": "arn:aws:iam::account-ID:role/role-name",
"encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}

```

- c. 若要取得建置輸出成品，請開啟您的 S3 輸出儲存貯體。
 - d. 將 *GoOutputArtifact.zip* 檔案下載到您的本機電腦或執行個體，然後解壓縮檔案的內容。在已解壓縮的內容中，取得 hello 檔案。
3. 如果下列其中一項為 true，您必須將許可新增至 Amazon ECR 中的映像儲存庫，讓 AWS CodeBuild 可以將 Docker 映像提取至建置環境。
 - 您的專案使用 CodeBuild 登入資料來提取 Amazon ECR 映像。這個行為可由 ProjectEnvironment 中屬性 imagePullCredentialsType 的值 CODEBUILD 看出。
 - 您的專案使用跨帳戶 Amazon ECR 映像。在這種情況下，您的專案必須使用其服務角色來提取 Amazon ECR 映像。若要啟用這種行為，請將您 ProjectEnvironment 的 imagePullCredentialsType 屬性設定成 SERVICE_ROLE。
1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
 2. 在儲存庫名稱的清單中，選擇您建立或選取的儲存庫名稱。
 3. 從導覽窗格，依序選擇 Permissions (許可)、Edit (編輯) 和 Add statement (新增陳述式)。
 4. 針對 Statement name (陳述式名稱)，輸入識別符 (例如 **CodeBuildAccess**)。
 5. 用於生效時，請保留選取允許。這樣做表示您允許存取另一個 AWS 帳戶。
 6. 用於 Principal (委託人)，執行下列其中一項操作：

- 如果您的專案使用 CodeBuild 登入資料來提取 Amazon ECR 映像，請在服務主體中輸入 **codebuild.amazonaws.com**。
- 如果您的專案使用跨帳戶 Amazon ECR 映像，請在AWS 帳戶 IDs 中輸入IDs。AWS

7. 略過 All IAM entities (所有 IAM 實體) 清單。

8. 針對 Action (動作)，選取所有僅限提取的動

作：ecr:GetDownloadUrlForLayer、ecr:BatchGetImage 和 ecr:BatchCheckLayerAvailability。

9. 針對條件，新增下列項目：

```
{
  "StringEquals":{
    "aws:SourceAccount": "<AWS-account-ID>",
    "aws:SourceArn": "arn:aws:codebuild:<region>:<AWS-account-ID>:project/<project-name>"
  }
}
```

10. 選擇儲存。

此政策會顯示在 Permissions (許可) 中。委託人是您在此程序的步驟 3 中針對 Principal (委託人) 輸入的委託人：

- 如果您的專案使用 CodeBuild 登入資料來提取 Amazon ECR 映像，則 "codebuild.amazonaws.com" 會顯示在服務主體下。
- 如果您的專案使用跨帳戶 Amazon ECR 映像，您要授予存取權 AWS 的帳戶 ID 會顯示在AWS 帳戶 IDs下方。

下列範例政策同時使用 CodeBuild 登入資料和跨帳戶 Amazon ECR 映像。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPrincipal",
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    }
  ],
}
```

```

        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:SourceArn": "arn:aws:codebuild:us-
east-1:111122223333:project/MyProject",
                "aws:SourceAccount": "111122223333"
            }
        },
        {
            "Sid": "CodeBuildAccessCrossAccount",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ],
            "Resource": "*"
        }
    ]
}

```

- 如果您的專案使用 CodeBuild 登入資料，而且您希望 CodeBuild 專案開放存取 Amazon ECR 儲存庫，您可以省略Condition金鑰並新增下列範例政策。

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildAccessPrincipal",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:codecommit:us-
east-2:111122223333:MySharedDemoRepo"
            ],
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ]
        },
    ],
}

```

```

    {
      "Sid": "CodeBuildAccessCrossAccount",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:codecommit:us-
east-2:111122223333:MySharedDemoRepo"
      ],
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    }
  ]
}

```

4. 建立組建專案、執行組建，以及檢視組建資訊。

如果您使用 AWS CLI 來建立建置專案，create-project 命令的 JSON 格式輸入可能會與此類似。(以您自己的值取代預留位置。)

```

{
  "name": "amazon-ecr-sample-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "account-ID.dkr.ecr.region-ID.amazonaws.com/your-Amazon-ECR-repo-name:tag",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}

```


5. 若要取得建置輸出成品，請開啟您的 S3 輸出儲存貯體。
6. 將 *GoOutputArtifact.zip* 檔案下載到您的本機電腦或執行個體，然後解壓縮 *GoOutputArtifact.zip* 檔案的內容。在已解壓縮的內容中，取得 hello 檔案。

Go 專案結構

此範例假設此目錄結構。

```
(root directory name)
### buildspec.yml
### hello.go
```

Go 專案檔案

此範例使用這些檔案。

buildspec.yml (在 *(root directory name)* 中)

```
version: 0.2

phases:
  install:
    runtime-versions:
      golang: 1.13
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Go code
      - go build hello.go
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - hello
```

hello.go (在 *(root directory name)* 中)

```
package main
import "fmt"
```

```
func main() {  
    fmt.Println("hello world")  
    fmt.Println("1+1 =", 1+1)  
    fmt.Println("7.0/3.0 =", 7.0/3.0)  
    fmt.Println(true && false)  
    fmt.Println(true || false)  
    fmt.Println(!true)  
}
```

的 Amazon Elastic File System 範例 AWS CodeBuild

您可能想要在 Amazon Elastic File System 上建立 AWS CodeBuild 組建，Amazon EC2 執行個體是可擴展的共用檔案服務。Amazon EFS 的儲存容量是彈性的，因此它會隨著檔案的新增和移除而成長或縮減。它有簡單的 Web 服務界面，可讓您用來快速輕鬆地建立和設定檔案系統。它還會為您管理所有檔案儲存基礎設施，因此您不需要擔心部署、修補，或維護檔案系統組態。如需詳細資訊，請參閱[什麼是 Amazon Elastic File System ?](#) Amazon Elastic File System 使用者指南中的。

此範例說明如何設定 CodeBuild 專案，使其掛載並建置 Java 應用程式至 Amazon EFS 檔案系統。開始之前，您必須準備好建置上傳至 S3 輸入儲存貯體或 AWS CodeCommit、GitHub、GitHub Enterprise Server 或 Bitbucket 儲存庫的 Java 應用程式。

檔案系統傳輸中的資料會受到加密。若要使用不同的映像加密傳輸中的資料，請參閱[加密傳輸中的資料](#)。

主題

- [AWS CodeBuild 搭配 Amazon Elastic File System 使用](#)
- [Amazon EFS 整合疑難排解](#)

AWS CodeBuild 搭配 Amazon Elastic File System 使用

此範例涵蓋搭配使用 Amazon EFS 所需的四個高階步驟 AWS CodeBuild。這些類別為：

1. 在 AWS 帳戶中建立虛擬私有雲端 (VPC)。
2. 建立使用此 VPC 的檔案系統。
3. 建立並建置使用 VPC 的 CodeBuild 專案。CodeBuild 專案使用下列項目來識別檔案系統：
 - 不重複的檔案系統識別碼。當您在組件專案中指定檔案系統時，您可以選擇識別碼。
 - 檔案系統 ID。當您在 Amazon EFS 主控台中檢視檔案系統時，會顯示 ID。
 - 一個掛載點。這是 Docker 容器中的一個目錄，用於掛載檔案系統。

- 掛載選項。這些包括如何掛載檔案系統的相關詳細資訊。
4. 檢閱建置專案，以確保產生正確的專案檔案和變數。

Note

僅在 Linux 平台上支援在 Amazon EFS 中建立的檔案系統。

主題

- [步驟 1：使用 建立 VPC CloudFormation](#)
- [步驟 2：使用 VPC 建立 Amazon Elastic File System 檔案系統](#)
- [步驟 3：建立要與 Amazon EFS 搭配使用的 CodeBuild 專案](#)
- [步驟 4：檢閱建置專案](#)

步驟 1：使用 建立 VPC CloudFormation

使用 CloudFormation 範本建立 VPC。

1. 請依照 中的指示[CloudFormation VPC 範本](#)，使用 CloudFormation 來建立 VPC。

Note

此 CloudFormation 範本建立的 VPC 有兩個私有子網路和兩個公有子網路。只有在使用 AWS CodeBuild 掛載您在 Amazon EFS 中建立的檔案系統時，才能使用私有子網路。如果您使用其中一個公有子網路，則建置會失敗。

2. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/vpc/> 的 Amazon VPC 主控台。
3. 選擇您建立的 VPC CloudFormation。
4. 在 Description (描述) 標籤上，記下 VPC 的名稱及其 ID。當您稍後在這個範例建立您的 AWS CodeBuild 專案時，會需要這兩個項目。

步驟 2：使用 VPC 建立 Amazon Elastic File System 檔案系統

使用您先前建立的 VPC，為此範例建立簡單的 Amazon EFS 檔案系統。

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/efs/> 開啟 Amazon EFS 主控台。
2. 選擇 Create file system (建立檔案系統)。
3. 從 VPC 中，選擇您在此範例中稍早記下的 VPC 名稱。
4. 將與您的子網路相關聯的可用區域保持選取。
5. 選擇 Next Step (後續步驟)。
6. 在新增標籤中，針對預設名稱索引鍵，在值中輸入 Amazon EFS 檔案系統的名稱。
7. 將 Bursting (爆量) 與 General Purpose (一般用途) 分別選為預設效能及傳輸量模式，然後選擇 Next Step (下一步)。
8. 對於 Configure new client (設定新用戶端)，請選擇 Next Step (下一個步驟)。
9. 選擇 Create File System (建立檔案系統)。
10. (選用) 建議您將政策新增至 Amazon EFS 檔案系統，以強制執行傳輸中資料的加密。在 Amazon EFS 主控台中，選擇檔案系統政策，選擇編輯，選取標示為所有用戶端強制執行傳輸中加密的方塊，然後選擇儲存。

步驟 3：建立要與 Amazon EFS 搭配使用的 CodeBuild 專案

建立使用您先前在此範例中建立之 VPC 的 AWS CodeBuild 專案。當組建執行時，它會掛載先前建立的 Amazon EFS 檔案系統。接下來，組件會儲存由您的 Java 應用程式於檔案系統的掛載點目錄中建立的 .jar 檔案。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 從導覽窗格，選擇 Build projects (建置專案)，然後選擇 Create build project (建立建置專案)。
3. 在專案名稱中，輸入您的專案名稱。
4. 從 Source provider (來源供應商)，選擇包含您希望建置的 Java 應用程式的儲存庫。
5. 輸入 CodeBuild 用來尋找應用程式的資訊，例如儲存庫 URL。每個來源供應商的選項不同。如需詳細資訊，請參閱 [Choose source provider](#)。
6. 從 Environment image (環境映像)，選擇 Managed image (受管映像)。
7. 從 Operating system (作業系統)，選擇 Amazon Linux 2。
8. 從 Runtime(s) (執行時間)，選擇 Standard (標準)。
9. 在影像中，選擇 aws/codebuild/amazonlinux-x86_64-standard : 4.0。

10. 從 Environment type (環境類型) 中選擇 Linux。
11. 在 Service role (服務角色) 下，選擇 New service role (新服務角色)。在角色名稱中，輸入 CodeBuild 為您建立的角色名稱。
12. 展開 Additional configuration (其他組態)。
13. 選取 Enable this flag if you want to build Docker images or want your builds to get elevated privileges (若想建置 Docker 影像或讓您的建置提升權限，請啟用此標記)。

 Note

預設會針對非 VPC 建置啟用 Docker 協助程式。如果您想要將 Docker 容器用於 VPC 建置，請參閱 Docker 文件網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特殊權限模式。

14. 從 VPC，選擇 VPC ID。
15. 從 Subnets (子網路)，選擇與您的 VPC 關聯的一或多個私有子網路。您必須在掛載 Amazon EFS 檔案系統的建置中使用私有子網路。如果您使用公有子網路，則建置會失敗。
16. 從 Security Groups (安全群組) 中選擇預設的安全群組。
17. 在 File systems (檔案系統) 中，輸入下列資訊：
 - 針對 Identifier (識別碼)，輸入唯一的檔案系統識別碼。它必須少於 129 個字元，且僅包含英數字元和底線。CodeBuild 使用此識別符來建立可識別彈性檔案系統的環境變數。環境變數格式是大寫英文字母的 CODEBUILD_<file_system_identifier>。例如，如果您輸入 my_efs，環境變數則為 CODEBUILD_MY_EFS。
 - 針對 ID，請選擇檔案系統 ID。
 - (選用) 在檔案系統中輸入目錄。CodeBuild 掛載此目錄。如果您將目錄路徑保留空白，CodeBuild 會掛載整個檔案系統。此路徑相對於檔案系統的根目錄。
 - 針對掛載點，在掛載檔案系統的建置容器中，輸入目錄的絕對路徑。如果此目錄不存在，CodeBuild 會在建置期間建立目錄。
 - (選用) 輸入掛載選項。如果您將掛載選項保留空白，CodeBuild 會使用其預設掛載選項：

```
nfsvers=4.1
rsize=1048576
wsize=1048576
hard
timeo=600
retrans=2
```

如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的[建議的 NFS 掛載選項](#)。

18. 如為 Build specification (建置規格)，選擇 Insert build commands (插入建置命令)，然後選擇 Switch to editor (切換到編輯器)。
19. 將以下 buildspec 命令輸入到編輯器。使用您在步驟 17 中輸入的識別碼取代 `<file_system_identifier>`。使用大寫英文字母 (例如：CODEBUILD_MY_EFS)。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: corretto11
  build:
    commands:
      - mvn compile -Dpgp.skip=true -Dmaven.repo.local=
        $CODEBUILD_<file_system_identifier>
```

20. 對所有其他設定使用預設值，然後選擇 Create build project (建立建置專案)。當建置完成時，您專案的主控台頁面就會顯示。
21. 選擇 Start build (開始組建)。

步驟 4：檢閱建置專案

建置 AWS CodeBuild 專案之後：

- 您有由 Java 應用程式建立的 .jar 檔案，該檔案建置在掛載點目錄下的 Amazon EFS 檔案系統。
- 可辨識您檔案系統的環境變數，會使用您在建立專案時輸入的檔案系統識別碼建立。

如需詳細資訊，請參閱《Amazon Elastic [File System 使用者指南](#)》中的[掛載檔案系統](#)。Amazon Elastic File System

Amazon EFS 整合疑難排解

以下是使用 CodeBuild 設定 Amazon EFS 時可能遇到的錯誤。

主題

- [CLIENT_ERROR：掛載 '127.0.0.1：/' 失敗。許可遭拒](#)

- [CLIENT_ERROR：掛載 '127.0.0.1：/' 失敗。連線依對等重設](#)
- [VPC_CLIENT_ERROR：未預期的 EC2 錯誤：UnauthorizedOperation](#)

CLIENT_ERROR：掛載 '127.0.0.1：/' 失敗。許可遭拒

使用 CodeBuild 掛載 Amazon EFS 不支援 IAM 授權。如果您使用的是自訂 Amazon EFS 檔案系統政策，您將需要授予所有 IAM 主體的讀取和寫入存取權。例如：

```
"Principal": {  
  "AWS": "*" }  
}
```

CLIENT_ERROR：掛載 '127.0.0.1：/' 失敗。連線依對等重設

此錯誤有兩種可能的原因：

- CodeBuild VPC 子網路與 Amazon EFS 掛載目標位於不同的可用區域。您可以透過在與 Amazon EFS 掛載目標相同的可用區域中新增 VPC 子網路來解決此問題。
- 安全群組沒有與 Amazon EFS 通訊的許可。您可以新增傳入規則，以允許來自 VPC（為您的 VPC 新增主要 CIDR 區塊）或安全群組本身的所有流量來解決此問題。

VPC_CLIENT_ERROR：未預期的 EC2 錯誤：UnauthorizedOperation

當 CodeBuild 專案 VPC 組態中的所有子網路都是公有子網路時，就會發生此錯誤。VPC 中必須至少有一個私有子網路，以確保網路連線。

AWS CodePipeline CodeBuild 的範例

本節說明 CodePipeline 和 CodeBuild 之間的範例整合。

樣本	說明
CodePipeline/CodeBuild 整合和批次建置的範例	這些範例示範如何使用 AWS CodePipeline 來建立使用批次建置的建置專案。
CodePipeline/CodeBuild 與多個輸入來源和輸出成品整合的範例	此範例示範如何使用 AWS CodePipeline 建立使用多個輸入來源建立多個輸出成品的建置專案。

CodePipeline/CodeBuild 整合和批次建置的範例

AWS CodeBuild 支援批次建置。下列範例示範如何使用 AWS CodePipeline 建立使用批次建置的建置專案。

您可以使用定義管道結構的 JSON 格式檔案，然後將其與 搭配使用 AWS CLI 來建立管道。如需詳細資訊，請參閱AWS CodePipeline 《使用者指南》中的[AWS CodePipeline 管道結構參考](#)。

具有個別成品的批次建置

使用下列 JSON 檔案做為管道結構的範例，該結構會建立具有個別成品的批次建置。若要在 CodePipeline 中啟用批次建置，請將 configuration 物件的 BatchEnabled 參數設定為 true。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source1",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "source1"
              }
            ],
            "configuration": {
              "S3Bucket": "<my-input-bucket-name>",
              "S3ObjectKey": "my-source-code-file-name.zip"
            },
            "runOrder": 1
          },
          {
            "inputArtifacts": [],
            "name": "Source2",
```



```
    "actionTypeId": {
      "category": "Source",
      "owner": "AWS",
      "version": "1",
      "provider": "S3"
    },
    "outputArtifacts": [
      {
        "name": "source2"
      }
    ],
    "configuration": {
      "S3Bucket": "<my-other-input-bucket-name>",
      "S3ObjectKey": "my-other-source-code-file-name.zip"
    },
    "runOrder": 1
  }
]
},
{
  "name": "Build",
  "actions": [
    {
      "inputArtifacts": [
        {
          "name": "source1"
        },
        {
          "name": "source2"
        }
      ],
      "name": "Build",
      "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "version": "1",
        "provider": "CodeBuild"
      },
      "outputArtifacts": [
        {
          "name": "build1"
        },
        {
          "name": "build1_artifact1"
        }
      ]
    }
  ]
}
```

```

        },
        {
            "name": "build1_artifact2"
        },
        {
            "name": "build2_artifact1"
        },
        {
            "name": "build2_artifact2"
        }
    ],
    "configuration": {
        "ProjectName": "my-build-project-name",
        "PrimarySource": "source1",
        "BatchEnabled": "true"
    },
    "runOrder": 1
}
]
}
],
"artifactStore": {
    "type": "S3",
    "location": "<AWS-CodePipeline-internal-bucket-name>"
},
"name": "my-pipeline-name",
"version": 1
}
}

```

以下是將使用此管道組態的 CodeBuild builds spec 檔案範例。

```

version: 0.2
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:

```

```
build:
  commands:
    - echo 'file' > output_file

artifacts:
  files:
    - output_file
  secondary-artifacts:
    artifact1:
      files:
        - output_file
    artifact2:
      files:
        - output_file
```

管道 JSON 檔案中指定的輸出成品名稱必須符合 buildspec 檔案中定義的組建和成品的識別符。語法是主要成品的 *buildIdentifier*，次要成品的 *buildIdentifier_artifactIdentifier*。

例如，對於輸出成品名稱 build1，CodeBuild 會將 的主要成品上傳build1到 的位置build1。對於輸出名稱 build1_artifact1，CodeBuild 會將 artifact1 的次要成品上傳build1到 的位置build1_artifact1，以此類推。如果只指定一個輸出位置，則名稱應該是 *buildIdentifier*。

建立 JSON 檔案之後，您可以建立管道。使用 AWS CLI 執行 create-pipeline 命令，並將 檔案傳遞給 --cli-input-json 參數。如需詳細資訊，請參閱AWS CodePipeline 《使用者指南》中的[建立管道 \(CLI\)](#)。

具有合併成品的批次建置

使用下列 JSON 檔案做為管道結構的範例，該結構會建立具有合併成品的批次建置。若要在 CodePipeline 中啟用批次建置，請將 configuration 物件的 BatchEnabled 參數設定為 true。若要將建置成品合併到相同的位置，請將configuration物件的 CombineArtifacts 參數設定為 true。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
```

```
    "name": "Source1",
    "actionTypeId": {
      "category": "Source",
      "owner": "AWS",
      "version": "1",
      "provider": "S3"
    },
    "outputArtifacts": [
      {
        "name": "source1"
      }
    ],
    "configuration": {
      "S3Bucket": "<my-input-bucket-name>",
      "S3ObjectKey": "my-source-code-file-name.zip"
    },
    "runOrder": 1
  },
  {
    "inputArtifacts": [],
    "name": "Source2",
    "actionTypeId": {
      "category": "Source",
      "owner": "AWS",
      "version": "1",
      "provider": "S3"
    },
    "outputArtifacts": [
      {
        "name": "source2"
      }
    ],
    "configuration": {
      "S3Bucket": "<my-other-input-bucket-name>",
      "S3ObjectKey": "my-other-source-code-file-name.zip"
    },
    "runOrder": 1
  }
]
},
{
  "name": "Build",
  "actions": [
    {
```

```
    "inputArtifacts": [
      {
        "name": "source1"
      },
      {
        "name": "source2"
      }
    ],
    "name": "Build",
    "actionTypeId": {
      "category": "Build",
      "owner": "AWS",
      "version": "1",
      "provider": "CodeBuild"
    },
    "outputArtifacts": [
      {
        "name": "output1 "
      }
    ],
    "configuration": {
      "ProjectName": "my-build-project-name",
      "PrimarySource": "source1",
      "BatchEnabled": "true",
      "CombineArtifacts": "true"
    },
    "runOrder": 1
  }
]
}
],
"artifactStore": {
  "type": "S3",
  "location": "<AWS-CodePipeline-internal-bucket-name>"
},
"name": "my-pipeline-name",
"version": 1
}
}
```

以下是將使用此管道組態的 CodeBuild buildspec 檔案範例。

```
version: 0.2
```

```
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:
  build:
    commands:
      - echo 'file' > output_file

artifacts:
  files:
    - output_file
```

如果批次建置已啟用合併成品，則只允許一個輸出。CodeBuild 會將所有組建的主要成品合併為單一 ZIP 檔案。

建立 JSON 檔案之後，您可以建立管道。使用 AWS CLI 執行 `create-pipeline` 命令，並將檔案傳遞給 `--cli-input-json` 參數。如需詳細資訊，請參閱AWS CodePipeline 《使用者指南》中的[建立管道 \(CLI\)](#)。

CodePipeline/CodeBuild 與多個輸入來源和輸出成品整合的範例

AWS CodeBuild 專案可以採用多個輸入來源。也可以建立不只一個輸出成品。此範例示範如何使用 AWS CodePipeline 建立使用多個輸入來源建立多個輸出成品的建置專案。如需詳細資訊，請參閱[多個輸入來源和輸出成品範例](#)。

您可以使用定義管道結構的 JSON 格式檔案，然後將其與 搭配使用 AWS CLI 來建立管道。使用下列 JSON 檔案作為管道結構的範例，以建立具有多個輸入來源和多個輸出成品的組建。稍後在此範例中，您會看到此檔案如何指定多個輸入和輸出。如需詳細資訊，請參閱AWS CodePipeline 《使用者指南》中的 [CodePipeline 管道結構參考](#)。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
```

```
"actions": [  
  {  
    "inputArtifacts": [],  
    "name": "Source1",  
    "actionTypeId": {  
      "category": "Source",  
      "owner": "AWS",  
      "version": "1",  
      "provider": "S3"  
    },  
    "outputArtifacts": [  
      {  
        "name": "source1"  
      }  
    ],  
    "configuration": {  
      "S3Bucket": "my-input-bucket-name",  
      "S3ObjectKey": "my-source-code-file-name.zip"  
    },  
    "runOrder": 1  
  },  
  {  
    "inputArtifacts": [],  
    "name": "Source2",  
    "actionTypeId": {  
      "category": "Source",  
      "owner": "AWS",  
      "version": "1",  
      "provider": "S3"  
    },  
    "outputArtifacts": [  
      {  
        "name": "source2"  
      }  
    ],  
    "configuration": {  
      "S3Bucket": "my-other-input-bucket-name",  
      "S3ObjectKey": "my-other-source-code-file-name.zip"  
    },  
    "runOrder": 1  
  }  
],  
{  
  {  
    "name": "Build",  
    "actionTypeId": {  
      "category": "Build",  
      "owner": "AWS",  
      "version": "1",  
      "provider": "CodeBuild"  
    },  
    "configuration": {  
      "S3Bucket": "my-build-output-bucket-name",  
      "S3ObjectKey": "my-build-output-file-name.zip"  
    },  
    "runOrder": 1  
  }  
}
```

```
"name": "Build",
"actions": [
  {
    "inputArtifacts": [
      {
        "name": "source1"
      },
      {
        "name": "source2"
      }
    ],
    "name": "Build",
    "actionTypeId": {
      "category": "Build",
      "owner": "AWS",
      "version": "1",
      "provider": "AWS CodeBuild"
    },
    "outputArtifacts": [
      {
        "name": "artifact1"
      },
      {
        "name": "artifact2"
      }
    ],
    "configuration": {
      "ProjectName": "my-build-project-name",
      "PrimarySource": "source1"
    },
    "runOrder": 1
  }
],
"artifactStore": {
  "type": "S3",
  "location": "AWS-CodePipeline-internal-bucket-name"
},
"name": "my-pipeline-name",
"version": 1
}
```


在此 JSON 檔案中：

- 您的其中一個輸入來源必須指定為 `PrimarySource`。此來源是 CodeBuild 尋找並執行 `buildspec` 檔案的目錄。在 JSON 檔案的 CodeBuild 階段的 `configuration` 區段中，關鍵字 `PrimarySource` 用來指定主要來源。
- 每個輸入來源安裝在其自己的目錄中。此目錄會存放在內建的环境變數，`$CODEBUILD_SRC_DIR` 表示主要來源和 `$CODEBUILD_SRC_DIR_yourInputArtifactName` 表示所有其他來源。就此範例中的管道而言，兩個輸入來源目錄是 `$CODEBUILD_SRC_DIR` 和 `$CODEBUILD_SRC_DIR_source2`。如需詳細資訊，請參閱[建置環境中的環境變數](#)。
- 在管道的 JSON 檔案中指定的輸出成品名稱，必須符合 `buildspec` 檔案中定義的次要成品名稱。此管道使用下列 `buildspec` 檔案。如需詳細資訊，請參閱[Buildspec 語法](#)。

```
version: 0.2

phases:
  build:
    commands:
      - touch source1_file
      - cd $CODEBUILD_SRC_DIR_source2
      - touch source2_file

artifacts:
  files:
    - '**/*'
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR
      files:
        - source1_file
    artifact2:
      base-directory: $CODEBUILD_SRC_DIR_source2
      files:
        - source2_file
```

建立 JSON 檔案之後，您可以建立管道。使用 AWS CLI 執行 `create-pipeline` 命令，並將檔案傳遞給 `--cli-input-json` 參數。如需詳細資訊，請參閱 AWS CodePipeline 《使用者指南》中的[建立管道 \(CLI\)](#)。

AWS Config 範例搭配 CodeBuild

AWS Config 提供 AWS 資源的清查，以及這些資源的組態變更歷史記錄。AWS Config 現在支援 AWS CodeBuild 做為 AWS 資源，這表示服務可以追蹤 CodeBuild 專案。如需詳細資訊 AWS Config，請參閱[什麼是 AWS Config？](#) 《AWS Config 開發人員指南》中的。

您可以在 AWS Config 主控台的資源庫存頁面上查看下列有關 CodeBuild 資源的資訊：

- CodeBuild 組態的時間軸會變更。
- 每個 CodeBuild 專案的組態詳細資訊。
- 與其他 AWS 資源的關係。
- CodeBuild 專案的變更清單。

主題

- [搭配 使用 CodeBuild AWS Config](#)
- [步驟 3：在 AWS Config 主控台中檢視 AWS CodeBuild 資料](#)

搭配 使用 CodeBuild AWS Config

本主題中的程序說明如何設定 AWS Config 和查詢 CodeBuild 專案。

主題

- [先決條件](#)
- [步驟 1：設定 AWS Config](#)
- [步驟 2：查詢 AWS CodeBuild 專案](#)

先決條件

建立您的 AWS CodeBuild 專案。如需說明，請參閱[建立組建專案](#)。

步驟 1：設定 AWS Config

- [設定 AWS Config \(主控台\)](#)
- [設定 AWS Config \(AWS CLI\)](#)

 Note

完成設定後，最多可能需要 10 分鐘才能在 AWS Config 主控台中看到 AWS CodeBuild 專案。

步驟 2：查詢 AWS CodeBuild 專案

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/config> 開啟 AWS Config 主控台。
2. 在資源庫存頁面上，選取資源類型下的 AWS CodeBuild 專案。向下捲動並選取 CodeBuild project (CodeBuild 專案) 核取方塊。
3. 選擇 Look up (查閱)。
4. 新增 CodeBuild 專案清單後，請在 Config 時間軸欄中選擇 CodeBuild 專案名稱連結。

步驟 3：在 AWS Config 主控台中檢視 AWS CodeBuild 資料

在資源庫存頁面上查詢資源時，您可以選擇 AWS Config 時間軸以檢視 CodeBuild 專案的詳細資訊。資源的詳細資訊頁面提供該資源之組態、關係以及對其所做之變更的相關資訊。

頁面頂端的區塊統稱為時間軸。時間軸顯示記錄的日期和時間。

如需詳細資訊，請參閱 [《開發人員指南》中的在 AWS Config 主控台中檢視組態詳細資訊](#)。AWS Config

CodeBuild 的建置通知範例

Amazon CloudWatch Events 具有的內建支援 AWS CodeBuild。CloudWatch Events 是描述 AWS 資源變更的系統事件串流。使用 CloudWatch Events，您可以撰寫宣告規則，將感興趣的事件與要採取的自動化動作建立關聯。此範例使用 Amazon CloudWatch Events 和 Amazon Simple Notification Service (Amazon SNS)，每當組建成功、失敗、從一個組建階段移至另一個組建階段，或這些事件的任意組合時，就會傳送組建通知給訂閱者。

 Important

執行此範例可能會導致 AWS 您的帳戶產生費用。這包括 CodeBuild 以及與 Amazon CloudWatch 和 Amazon SNS 相關的 AWS 資源和動作的可能費用。如需詳細資訊，請參閱 [CodeBuild 定價](#)、[Amazon CloudWatch 定價](#)和 [Amazon SNS 定價](#)。

主題

- [執行建置通知範例](#)
- [建置通知輸入格式參考](#)

執行建置通知範例

使用下列程序來執行建置通知範例。

如何執行此範例

1. 如果您已在 Amazon SNS 中設定並訂閱要用於此範例的主題，請跳到步驟 4。否則，如果您使用 IAM 使用者而非 AWS 根帳戶或管理員使用者來使用 Amazon SNS，請將下列陳述式（介於 **### BEGIN ADDING STATEMENT HERE ###** 和 **### END ADDING STATEMENT HERE ###** 之間）新增至使用者（或使用者相關聯的 IAM 群組）。不建議使用 AWS 根帳戶。此陳述式可讓您檢視、建立、訂閱和測試 Amazon SNS 中主題的通知傳送。省略符號 (...) 用於簡化和協助您找到新增陳述式的位置。請不要移除任何陳述式，也不要再在現有政策中輸入這些省略符號。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns:GetTopicAttributes",
        "sns:List*",
        "sns:Publish",
        "sns:SetTopicAttributes",
        "sns:Subscribe"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

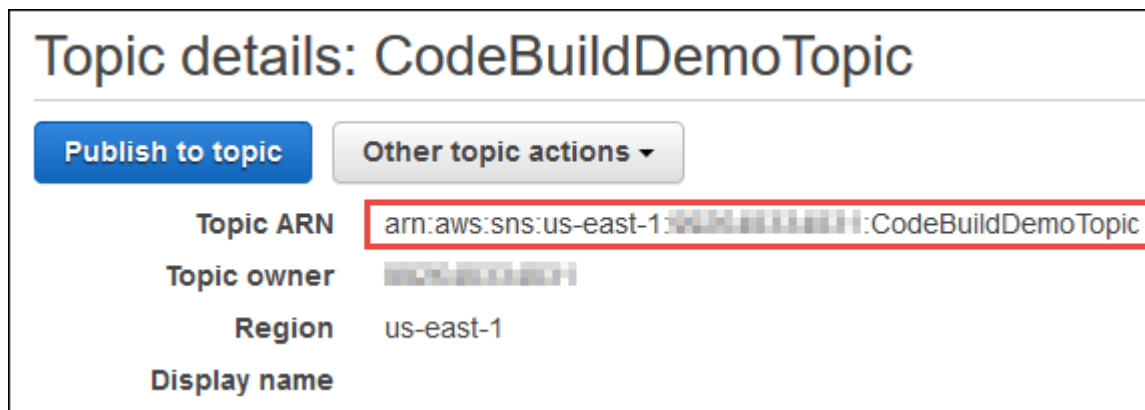
修改此政策的 IAM 實體必須在 IAM 中具有修改政策的許可。

如需詳細資訊，請參閱《IAM 使用者指南》中的[編輯客戶受管政策](#)或「編輯或刪除群組、使用者或角色的內嵌政策」一節中的[使用內嵌政策（主控台）](#)。

2. 在 Amazon SNS 中建立或識別主題。AWS CodeBuild 會使用 CloudWatch Events 透過 Amazon SNS 傳送建置通知至此主題。

建立主題：

1. 開啟位於 <https://console.aws.amazon.com/sns> 的 Amazon SNS 主控台。
2. 請選擇建立主題。
3. 在 Create new topic (建立新主題) 中，針對 Topic name (主題名稱)，輸入主題的名稱 (例如 **CodeBuildDemoTopic**)。(如果您選擇其他名稱，請在此範例中全部換成此名稱。)
4. 請選擇建立主題。
5. 在主題詳細資訊：CodeBuildDemoTopic 頁面上，複製主題 ARN 值。您在下一個步驟中需要使用到此數值。



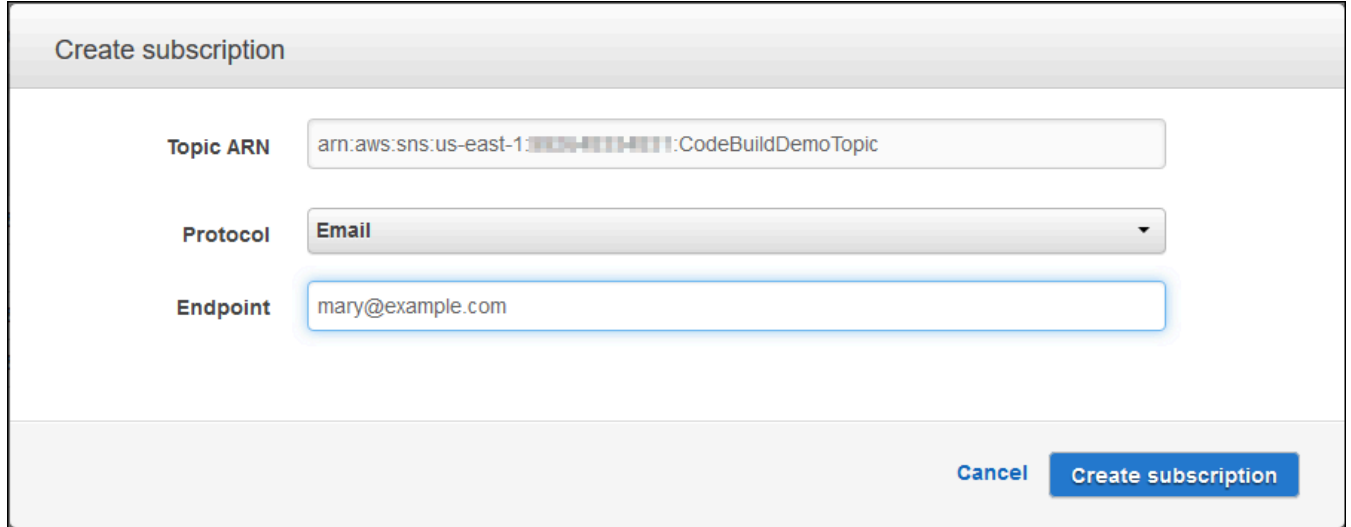
如需詳細資訊，請參閱《Amazon SNS 開發人員指南》中的[建立主題](#)。

3. 讓一或多個收件人訂閱主題來接收電子郵件通知。

讓收件人訂閱主題：

1. 從上一個步驟開啟 Amazon SNS 主控台後，在導覽窗格中，選擇訂閱，然後選擇建立訂閱。

2. 在 Create subscription (建立訂閱) 中，對於 Topic ARN (主題 ARN)，貼上您從上個步驟複製的主題 ARN。
3. 對於通訊協定，選擇電子郵件。
4. 針對 Endpoint (端點)，輸入收件人的完整電子郵件地址。



5. 選擇 Create Subscription (建立訂閱)。
6. Amazon SNS 會傳送訂閱確認電子郵件給收件人。若要開始接收電子郵件通知，收件人必須選擇訂閱確認電子郵件中的 Confirm subscription (確認訂閱) 連結。收件人按一下連結後，如果成功訂閱，Amazon SNS 會在收件人的 Web 瀏覽器中顯示確認訊息。

如需詳細資訊，請參閱《Amazon SNS 開發人員指南》中的[訂閱主題](#)。

4. 如果您使用使用者而非 AWS 根帳戶或管理員使用者來使用 CloudWatch Events，請將下列陳述式（介於 **### BEGIN ADDING STATEMENT HERE ###** 和 **### END ADDING STATEMENT HERE ###** 之間）新增至使用者（或使用者相關聯的 IAM 群組）。不建議使用 AWS 根帳戶。此陳述式用於允許使用者使用 CloudWatch Events。省略符號 (...) 用於簡化和協助您找到新增陳述式的位置。請不要移除任何陳述式，也不要再在現有政策中輸入這些省略符號。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

        "Action": [
            "events:*",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::*:role/Service*"
    }
}

```

Note

修改此政策的 IAM 實體必須在 IAM 中具有修改政策的許可。

如需詳細資訊，請參閱 [《IAM 使用者指南》中的編輯客戶受管政策](#) 或「編輯或刪除群組、使用者或角色的內嵌政策」一節 [中的使用內嵌政策（主控台）](#)。

5. 在 CloudWatch Events 中建立規則。若要這樣做，請開啟 CloudWatch 主控台，網址為 <https://console.aws.amazon.com/cloudwatch>。
6. 在導覽窗格中，在 Events (事件) 下方選擇 Rules (規則)，然後選擇 Create rule (建立規則)。
7. 在 Step 1: Create rule page (步驟 1：建立規則頁面) 上，應該已選取 Event Pattern (事件模式) 和 Build event pattern to match events by service (建構事件的模式，依服務來匹配事件)。
8. 針對服務名稱，選擇 CodeBuild。在 Event Type (事件類型) 中，應該已選取 All Events (所有事件)。
9. 下列程式碼應顯示於 Event Pattern Preview (事件模式預覽) 中：

```

{
  "source": [
    "aws.codebuild"
  ]
}

```

10. 選擇 Edit (編輯)，將 Event Pattern Preview (事件模式預覽) 中的程式碼換成下列兩個規則模式之一。

AWS CodeBuild 中指定的建置專案有建置開始或完成時，這第一個規則模式就會觸發事件。

```

{
  "source": [
    "aws.codebuild"
  ]
}

```

```
],
"detail-type": [
  "CodeBuild Build State Change"
],
"detail": {
  "build-status": [
    "IN_PROGRESS",
    "SUCCEEDED",
    "FAILED",
    "STOPPED"
  ],
  "project-name": [
    "my-demo-project-1",
    "my-demo-project-2"
  ]
}
}
```

在上個規則中，依需要進行下列程式碼變更。

- 若要在有建置開始或完成時觸發事件，請保留 build-status 陣列中所示的所有值，或移除整個 build-status 陣列。
- 若只要在組建完成時才觸發事件，請從 build-status 陣列中移除 IN_PROGRESS。
- 若只要在組建開始時才觸發事件，請從 build-status 陣列中移除所有值，但保留 IN_PROGRESS。
- 若要對所有組建專案觸發事件，請移除整個 project-name 陣列。
- 若只要對個別組建專案觸發事件，請在 project-name 陣列中指定每個組建專案的名稱。

每當 AWS CodeBuild 中指定的建置專案有建置從一個建置階段進入另一個建置階段時，這第二個規則模式就會觸發事件。

```
{
  "source": [
    "aws.codebuild"
  ],
  "detail-type": [
    "CodeBuild Build Phase Change"
  ],
  "detail": {
    "completed-phase": [
```



```
    "SUBMITTED",
    "PROVISIONING",
    "DOWNLOAD_SOURCE",
    "INSTALL",
    "PRE_BUILD",
    "BUILD",
    "POST_BUILD",
    "UPLOAD_ARTIFACTS",
    "FINALIZING"
  ],
  "completed-phase-status": [
    "TIMED_OUT",
    "STOPPED",
    "FAILED",
    "SUCCEEDED",
    "FAULT",
    "CLIENT_ERROR"
  ],
  "project-name": [
    "my-demo-project-1",
    "my-demo-project-2"
  ]
}
```

在上個規則中，依需要進行下列程式碼變更。

- 若要在每次建置階段變更時觸發事件 (對每個建置，最多可傳送九次通知)，請保留 `completed-phase` 陣列中所示的所有值，或移除整個 `completed-phase` 陣列。
- 若只要對個別組建階段變更來觸發事件，請在 `completed-phase` 陣列中移除您不想觸發事件的每個組建階段的名稱。
- 若要在每次組建階段狀態變更時觸發事件，請保留 `completed-phase-status` 陣列中所示的全部值，或移除整個 `completed-phase-status` 陣列。
- 若只要對個別組建階段狀態變更來觸發事件，請在 `completed-phase-status` 陣列中移除您不想觸發事件的每個組建階段狀態的名稱。
- 若要對所有組建專案觸發事件，請移除 `project-name` 陣列。
- 若要對個別組建專案觸發事件，請在 `project-name` 陣列中指定每個組建專案的名稱。

如需事件模式的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[事件模式](#)。

如需使用事件模式進行篩選的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[使用事件模式進行內容型篩選](#)。

 Note

如果希望建置狀態變更和建置階段變更都觸發事件，您必須建立兩個不同的規則：一個用於建置狀態變更，另一個用於建置階段變更。如果您嘗試將兩個規則合併成單一規則，合併的規則可能產生非預期的結果，或完全停止運作。

完成取代程式碼時，選擇 Save (儲存)。

11. 在 Targets (目標) 中，選擇 Add target (新增目標)。
12. 在目標清單中，選擇 SNS topic (SNS 主題)。
13. 對於 Topic (主題)，選擇您稍早識別或建立的主題。
14. 展開 Configure input (設定輸入)，然後選擇 Input Transformer (輸入轉換器)。
15. 在 Input Path (輸入路徑) 方塊中，輸入下列其中一個輸入路徑。

如果規則的 detail-type 值為 CodeBuild Build State Change，請輸入下列程式碼。

```
{"build-id": "${detail.build-id}", "project-name": "${detail.project-name}", "build-status": "${detail.build-status"}
```

如果規則的 detail-type 值為 CodeBuild Build Phase Change，請輸入下列程式碼。

```
{"build-id": "${detail.build-id}", "project-name": "${detail.project-name}", "completed-phase": "${detail.completed-phase}", "completed-phase-status": "${detail.completed-phase-status"}
```

如需其他類型的資訊，請參閱[建置通知輸入格式參考](#)。

16. 在 Input Template (輸入範本) 方塊中，輸入下列其中一個輸入範本。

如果規則的 detail-type 值為 CodeBuild Build State Change，請輸入下列程式碼。

```
"Build '<build-id>' for build project '<project-name>' has reached the build status of '<build-status>'."
```

如果規則的 detail-type 值為 CodeBuild Build Phase Change，請輸入下列程式碼。

```
"Build '<build-id>' for build project '<project-name>' has completed the build phase of '<completed-phase>' with a status of '<completed-phase-status>'."
```

17. 選擇設定詳細資訊。
18. 在 Step 2: Configure rule details (步驟 2：設定規則詳細資訊) 頁面上，輸入名稱和選填的描述。針對 State (狀態)，將 Enabled (啟用) 保留為選取狀態。
19. 選擇建立規則。
20. 建立組建專案、執行組建，以及檢視組建資訊。
21. 確認 CodeBuild 現在已成功傳送建置通知。例如，檢查您的收件匣中是否有組建通知電子郵件。

若要變更規則的行為，請在 CloudWatch 主控台中選擇您要變更的規則，選擇動作，然後選擇編輯。變更規則，選擇 Configure details (設定詳細資訊)，然後選擇 Update rule (更新規則)。

若要停止使用規則來傳送建置通知，請在 CloudWatch 主控台中選擇您要停止使用的規則，選擇動作，然後選擇停用。

若要完全刪除規則，請在 CloudWatch 主控台中選擇您要刪除的規則，選擇動作，然後選擇刪除。

建置通知輸入格式參考

CloudWatch 會以 JSON 格式傳送通知。

組建狀態變更通知採用以下格式：

```
{
  "version": "0",
  "id": "c030038d-8c4d-6141-9545-00ff7b7153EX",
  "detail-type": "CodeBuild Build State Change",
  "source": "aws.codebuild",
  "account": "123456789012",
  "time": "2017-09-01T16:14:28Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-c340-456a-9166-edf953571bEX"
  ],
  "detail": {
    "build-status": "SUCCEEDED",
```

```
"project-name": "my-sample-project",
"build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
"additional-information": {
  "artifact": {
    "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
    "sha256sum":
"6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
    "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
  },
  "environment": {
    "image": "aws/codebuild/standard:5.0",
    "privileged-mode": false,
    "compute-type": "BUILD_GENERAL1_SMALL",
    "type": "LINUX_CONTAINER",
    "environment-variables": []
  },
  "timeout-in-minutes": 60,
  "build-complete": true,
  "initiator": "MyCodeBuildDemoUser",
  "build-start-time": "Sep 1, 2017 4:12:29 PM",
  "source": {
    "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
    "type": "S3"
  },
  "logs": {
    "group-name": "/aws/codebuild/my-sample-project",
    "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
    "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
  },
  "phases": [
    {
      "phase-context": [],
      "start-time": "Sep 1, 2017 4:12:29 PM",
      "end-time": "Sep 1, 2017 4:12:29 PM",
      "duration-in-seconds": 0,
      "phase-type": "SUBMITTED",
      "phase-status": "SUCCEEDED"
    },
    {
      "phase-context": [],
```

```
    "start-time": "Sep 1, 2017 4:12:29 PM",
    "end-time": "Sep 1, 2017 4:13:05 PM",
    "duration-in-seconds": 36,
    "phase-type": "PROVISIONING",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:05 PM",
    "end-time": "Sep 1, 2017 4:13:10 PM",
    "duration-in-seconds": 4,
    "phase-type": "DOWNLOAD_SOURCE",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:10 PM",
    "end-time": "Sep 1, 2017 4:13:10 PM",
    "duration-in-seconds": 0,
    "phase-type": "INSTALL",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:10 PM",
    "end-time": "Sep 1, 2017 4:13:10 PM",
    "duration-in-seconds": 0,
    "phase-type": "PRE_BUILD",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:10 PM",
    "end-time": "Sep 1, 2017 4:14:21 PM",
    "duration-in-seconds": 70,
    "phase-type": "BUILD",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:14:21 PM",
    "end-time": "Sep 1, 2017 4:14:21 PM",
    "duration-in-seconds": 0,
    "phase-type": "POST_BUILD",
```

```

        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:21 PM",
        "duration-in-seconds": 0,
        "phase-type": "UPLOAD_ARTIFACTS",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:26 PM",
        "duration-in-seconds": 4,
        "phase-type": "FINALIZING",
        "phase-status": "SUCCEEDED"
    },
    {
        "start-time": "Sep 1, 2017 4:14:26 PM",
        "phase-type": "COMPLETED"
    }
]
},
"current-phase": "COMPLETED",
"current-phase-context": "[]",
"version": "1"
}
}

```

組建階段變更通知採用以下格式：

```

{
    "version": "0",
    "id": "43ddc2bd-af76-9ca5-2dc7-b695e15adeEX",
    "detail-type": "CodeBuild Build Phase Change",
    "source": "aws.codebuild",
    "account": "123456789012",
    "time": "2017-09-01T16:14:21Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-c340-456a-9166-edf953571bEX"
    ]
}

```

```
],
"detail":{
  "completed-phase": "COMPLETED",
  "project-name": "my-sample-project",
  "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
  "completed-phase-context": "[]",
  "additional-information": {
    "artifact": {
      "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
      "sha256sum":
"6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
      "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
    },
    "environment": {
      "image": "aws/codebuild/standard:5.0",
      "privileged-mode": false,
      "compute-type": "BUILD_GENERAL1_SMALL",
      "type": "LINUX_CONTAINER",
      "environment-variables": []
    },
    "timeout-in-minutes": 60,
    "build-complete": true,
    "initiator": "MyCodeBuildDemoUser",
    "build-start-time": "Sep 1, 2017 4:12:29 PM",
    "source": {
      "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
      "type": "S3"
    },
    "logs": {
      "group-name": "/aws/codebuild/my-sample-project",
      "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
      "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
    },
    "phases": [
      {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:12:29 PM",
        "end-time": "Sep 1, 2017 4:12:29 PM",
        "duration-in-seconds": 0,
        "phase-type": "SUBMITTED",
```

```
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:12:29 PM",
    "end-time": "Sep 1, 2017 4:13:05 PM",
    "duration-in-seconds": 36,
    "phase-type": "PROVISIONING",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:05 PM",
    "end-time": "Sep 1, 2017 4:13:10 PM",
    "duration-in-seconds": 4,
    "phase-type": "DOWNLOAD_SOURCE",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:10 PM",
    "end-time": "Sep 1, 2017 4:13:10 PM",
    "duration-in-seconds": 0,
    "phase-type": "INSTALL",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:10 PM",
    "end-time": "Sep 1, 2017 4:13:10 PM",
    "duration-in-seconds": 0,
    "phase-type": "PRE_BUILD",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
    "start-time": "Sep 1, 2017 4:13:10 PM",
    "end-time": "Sep 1, 2017 4:14:21 PM",
    "duration-in-seconds": 70,
    "phase-type": "BUILD",
    "phase-status": "SUCCEEDED"
  },
  {
    "phase-context": [],
```



```

        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:21 PM",
        "duration-in-seconds": 0,
        "phase-type": "POST_BUILD",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:21 PM",
        "duration-in-seconds": 0,
        "phase-type": "UPLOAD_ARTIFACTS",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:26 PM",
        "duration-in-seconds": 4,
        "phase-type": "FINALIZING",
        "phase-status": "SUCCEEDED"
    },
    {
        "start-time": "Sep 1, 2017 4:14:26 PM",
        "phase-type": "COMPLETED"
    }
]
},
"completed-phase-status": "SUCCEEDED",
"completed-phase-duration-seconds": 4,
"version": "1",
"completed-phase-start": "Sep 1, 2017 4:14:21 PM",
"completed-phase-end": "Sep 1, 2017 4:14:26 PM"
}
}

```

使用 CodeBuild 建置徽章範例

AWS CodeBuild 現在支援使用組建徽章，該徽章提供可嵌入、動態產生的映像 (徽章)，顯示專案的最新組建狀態。此映像可透過為 CodeBuild 專案產生的公開可用 URL 存取。這可讓任何人檢視 CodeBuild 專案的狀態。組建識別證不包含任何安全資訊，因此不需要身分驗證。

主題

- [使用建置徽章建立建置專案](#)
- [存取 AWS CodeBuild 建置徽章](#)
- [發佈 CodeBuild 組建徽章](#)
- [CodeBuild 徽章狀態](#)

使用建置徽章建立建置專案

使用下列其中一個程序，在啟用建置徽章的情況下建立建置專案。您可以使用 AWS CLI 或 AWS 管理主控台。

建立已啟用建置徽章的建置專案 (AWS CLI)

- 如需建立組建專案的詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。若要在 AWS CodeBuild 專案中包含建置徽章，您必須指定 *badgeEnabled*，值為 `true`。

建立已啟用建置徽章的建置專案（主控台）

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。
3. 在 Project name (專案名稱) 中，輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。
4. 在 Source (來源) 中，針對 Source provider (來源供應商)，選擇來源碼提供商類型，然後執行下列其中一項：

Note

CodeBuild 不支援使用 Amazon S3 來源提供者建置徽章。由於 AWS CodePipeline 使用 Amazon S3 進行成品傳輸，因此屬於 CodePipeline 中建立管道一部分的建置專案不支援建置徽章。

- 如果您選擇 CodeCommit，則對於儲存庫，請選擇儲存庫的名稱。選取 Enable build badge (啟用組建徽章)，讓專案的組建狀態變成可見且可嵌入。

- 如果您選擇了 GitHub，請遵循說明來與 GitHub 連線 (或重新連線)。在 GitHub 授權應用程式頁面上，針對組織存取，選擇您要 AWS CodeBuild 能夠存取的每個儲存庫旁的請求存取。選擇 Authorize application (授權應用程式) 後，請回到 AWS CodeBuild 主控台，針對 Repository (儲存庫) 選擇包含來源碼的儲存庫名稱。選取 Enable build badge (啟用組建徽章)，讓專案的組建狀態變成可見且可嵌入。
- 如果您選擇了 Bitbucket，請遵循說明來與 Bitbucket 連線 (或重新連線)。在 Bitbucket Confirm access to your account (確認存取您的帳戶) 頁面上，針對 Organization access (組織存取)，選擇 Grant access (授予存取)。在您選擇授予存取權後，返回 AWS CodeBuild 主控台，針對儲存庫，選擇包含原始碼的儲存庫名稱。選取 Enable build badge (啟用組建徽章)，讓專案的組建狀態變成可見且可嵌入。

 Important

更新您的專案來源可能會影響專案組建徽章的準確度。

5. 在 Environment (環境) 中：

針對 Environment image (環境映像)，執行下列其中一項作業：

- 若要使用由 管理的 Docker 映像 AWS CodeBuild，請選擇受管映像，然後從作業系統、執行時間 (s)、映像和映像版本進行選擇。若可用，請從 Environment type (環境類型) 進行選擇。
- 若要使用另一個 Docker 映像，請選擇 Custom image (自訂映像)。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。如果您選擇其他登錄檔，請針對外部登錄檔 URL，使用 格式在 Docker Hub 中輸入 Docker 映像檔的名稱和標籤 *docker repository/docker image name*。如果您選擇 Amazon ECR，請使用 Amazon ECR 儲存庫和 Amazon ECR 映像來選擇 AWS 帳戶中的 Docker 映像。
- 若要使用私有 Docker 映像，請選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。針對 Image registry (映像登錄) 選擇 Other registry (其他登錄)，然後輸入私人 Docker 映像的憑證的 ARN。登入資料必須由 Secrets Manager 建立。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [What Is AWS Secrets Manager](#)。

6. 在 Service role (服務角色) 中，執行下列其中一項作業：

- 如果您沒有 CodeBuild 服務角色，請選擇新服務角色。在角色名稱中，輸入新角色的名稱。
- 如果您有 CodeBuild 服務角色，請選擇現有服務角色。在角色 ARN 中，選擇服務角色。

 Note

當您使用 主控台 建立或更新組建專案時，您可以同時建立 CodeBuild 服務角色。根據預設，此角色只能與該建置專案搭配運作。如果您使用主控台將此服務角色與另一個建置專案建立關聯，則會更新此角色以與其他建置專案搭配運作。服務角色最多可以與 10 個組建專案搭配運作。

7. 在 Buildspec 中，執行下列其中一項操作：

- 選擇使用 buildspec 檔案以使用來源碼根目錄中的 buildspec.yml 檔案。
- 選擇插入建置命令以使用主控台插入建置命令。

如需更多資訊，請參閱[Buildspec 參考](#)。

8. 在 Artifacts (成品) 中，針對 Type (類型)，執行下列其中一項操作：

- 如果您不要建立建置輸出成品，則請選擇 No artifacts (無成品)。
- 若要將建置輸出存放在 S3 儲存貯體，請選擇 Amazon S3，然後執行下列動作：
 - 如果您想要使用專案名稱做為組建輸出 ZIP 檔案或資料夾名稱，則請將 Name (名稱) 保留空白。否則請輸入名稱。根據預設，成品名稱是專案名稱。如果想使用不同的名稱，請在成品名稱方塊中輸入名稱。如果要輸出 ZIP 檔案，請包含 zip 副檔名。
 - 針對 Bucket name (儲存貯體名稱)，選擇輸出儲存貯體的名稱。
 - 如果您在本程序稍早選擇 Insert build commands (插入組建命令)，然後針對 Output files (輸出檔案)，輸入要放入組建輸出 ZIP 檔案或資料夾之組建中的檔案位置。針對多個位置，以逗號區隔每個位置 (例如，appspec.yml, target/my-app.jar)。如需詳細資訊，請參閱[Buildspec 語法](#)中的 files 描述。

9. 展開 Additional configuration (其他組態)，並適當地選擇選項。

10. 選擇 Create build project (建立建置專案)。在 Review (檢閱) 頁面上，選擇 Start build (開始建置) 來執行建置。

存取 AWS CodeBuild 建置徽章

您可以使用 AWS CodeBuild 主控台或 AWS CLI 來存取建置徽章。

- 在 CodeBuild 主控台的組建專案清單中，於名稱欄中選擇與組建專案對應的連結。在建置專案：***project-name*** 頁面上的組態中，選擇複製徽章 URL。如需詳細資訊，請參閱[檢視建置專案的詳細資訊 \(主控台\)](#)。
- 在中 AWS CLI，執行 `batch-get-projects` 命令。組建識別碼 URL 會包含在輸出的專案環境詳細資訊區段中。如需詳細資訊，請參閱[檢視建置專案的詳細資訊 \(AWS CLI\)](#)。

建置徽章請求 URL 是以常見的預設分支產生，但您可以在來源儲存庫中指定用來執行建置的任何分支。例如：

```
https://codebuild.us-east-1.amazon.com/badges?uuid=...&branch=<branch>
```

您也可以使用徽章 URL 中的 `tag` 參數來取代 `branch` 參數，以從來源儲存庫指定標籤。例如：

```
https://codebuild.us-east-1.amazon.com/badges?uuid=...&tag=<tag>
```

發佈 CodeBuild 組建徽章

您可以使用 Markdown 映像中的建置徽章 URL，在 Markdown 檔案中顯示最新建置的狀態。這有助於顯示來源儲存庫中 `readme.md` 檔案（例如 GitHub 或 CodeCommit）中最新組建的狀態。例如：

```

```

CodeBuild 徽章狀態

CodeBuild 組建徽章可以具有下列其中一種狀態。

- **PASSING** 在特定分支上的最新組建已通過。
- **FAILING** 在特定分支上的最新組建已逾時、故障、錯誤或停止。
- **IN_PROGRESS** 在特定分支上的最新組建正在進行中。
- **UNKNOWN** 專案尚未針對特定的分支執行組建（或沒有針對任何項目執行過）。此外，組建識別證功能可能已停用。

「使用 範例測試報告 AWS CLI」

在建置期間便會執行您在 buildspec 檔案中指定的測試。此範例說明如何使用 AWS CLI 將測試納入 CodeBuild 中的組建。您可以使用 JUnit 建立單位測試，或使用另一個工具建立組態測試。然後您可以評估測試結果，以修正問題或最佳化您的應用程式。

您可以使用 CodeBuild API 或 AWS CodeBuild 主控台來存取測試結果。此範例示範如何設定報告，以將其測試結果匯出至 S3 儲存貯體。

主題

- [執行測試報告範例](#)

執行測試報告範例

使用下列步驟來執行測試報告範例。

主題

- [先決條件](#)
- [步驟 1：建立報告群組](#)
- [步驟 2：使用報告群組設定專案](#)
- [步驟 3：執行和檢視報告的結果](#)

先決條件

- 建立您的測試案例。此範例是假設您已將測試案例納入您的範例測試報告中所撰寫。您可以在 buildspec 檔案中指定測試檔案的位置。

支援下列測試報告檔案格式：

- 小黃瓜 JSON (.json)
- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX (.trx)
- Visual Studio TRX XML (.xml)

使用可以用其中一種格式 (例如, Surefire JUnit 外掛程式、TestNG 或 Cucumber) 建立報告檔案的任何測試框架, 來建立您的測試案例。

- 建立 S3 儲存貯體並記下其名稱。如需詳細資訊, 請參閱[如何建立 S3 儲存貯體?](#) Amazon S3 使用者指南中的。
- 建立 IAM 角色並記下其 ARN。當您建立建置專案時, 需要使用 ARN。
- 如果您的角色沒有下列許可, 請新增這些許可。

```
{
  "Effect": "Allow",
  "Resource": [
    "*"
  ],
  "Action": [
    "codebuild:CreateReportGroup",
    "codebuild:CreateReport",
    "codebuild:UpdateReport",
    "codebuild:BatchPutTestCases"
  ]
}
```

如需詳細資訊, 請參閱[測試報告操作的許可](#)。

步驟 1：建立報告群組

1. 建立名為 CreateReportGroupInput.json 的檔案。
2. 在 S3 儲存貯體中建立要匯出測試結果的資料夾。
3. 將以下內容複製到 CreateReportGroupInput.json。對於 *<bucket-name>*, 請使用 S3 儲存貯體的名稱。對於 *<path-to-folder>*, 請輸入在 S3 儲存貯體中資料夾的路徑。

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<bucket-name>",
      "path": "<path-to-folder>",
      "packaging": "NONE"
    }
  }
}
```

```
    }  
  }  
}
```

4. 在包含 `CreateReportGroupInput.json` 的目錄中執行下列命令：

```
aws codebuild create-report-group --cli-input-json file://  
CreateReportGroupInput.json
```

輸出看起來如下。記下 `reportGroup` 的 ARN。當您建立使用此報告群組的專案時，您會使用此 ARN。

```
{  
  "reportGroup": {  
    "arn": "arn:aws:codebuild:us-west-2:123456789012:report-group/<report-name>",  
    "name": "<report-name>",  
    "type": "TEST",  
    "exportConfig": {  
      "exportConfigType": "S3",  
      "s3Destination": {  
        "bucket": "<s3-bucket-name>",  
        "path": "<folder-path>",  
        "packaging": "NONE",  
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3"  
      }  
    },  
    "created": 1570837165.885,  
    "lastModified": 1570837165.885  
  }  
}
```

步驟 2：使用報告群組設定專案

若要執行報告，請先建立使用報告群組設定的 CodeBuild 建置專案。當您執行建置時，便會執行您的報告群組指定的測試案例。

1. 建立名為 `buildspec.yml` 的 `buildspec` 檔案。
2. 使用下列 YAML 做為您 `buildspec.yml` 檔案的範本。請務必納入執行測試的命令。在 `reports` 區段中，指定包含測試案例結果的檔案。這些檔案會存放您可以使用 CodeBuild 存取的

測試結果。它們會在建立的 30 天後過期。這些檔案不同於您匯出至 S3 儲存貯體的原始測試案例結果檔案。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: openjdk8
  build:
    commands:
      - echo Running tests
      - <enter commands to run your tests>

reports:
  <report-name-or-arn>: #test file information
  files:
    - '<test-result-files>'
  base-directory: '<optional-base-directory>'
  discard-paths: false #do not remove file paths from test result files
```

Note

如果不要使用現有報告群組的 ARN，您也可以指定尚未建立之報告群組的名稱。如果您指定名稱而非 ARN，CodeBuild 會在執行組建時建立報告群組。其名稱包含專案名稱，以及您在 Buildspec 檔案中用此格式指定的名稱：project-name-report-group-name。如需詳細資訊，請參閱[建立測試報告](#)及[報告群組命名](#)。

3. 建立名為 project.json 的檔案。此檔案包含 create-project 命令的輸入。
4. 將以下 JSON 複製到 project.json。對於 source，請輸入包含您的來源檔案之儲存庫的類型和位置。對於 serviceRole，指定您正使用之角色的 ARN。

```
{
  "name": "test-report-project",
  "description": "sample-test-report-project",
  "source": {
    "type": "CODECOMMIT|CODEPIPELINE|GITHUB|S3|BITBUCKET|GITHUB_ENTERPRISE|NO_SOURCE",
    "location": "<your-source-url>"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  }
}
```

```
{,
  "cache": {
    "type": "NO_CACHE"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "small"
  },
  "serviceRole": "arn:aws:iam::<your-aws-account-id>:role/service-role/<your-role-name>"
}
```

5. 在包含 `project.json` 的目錄中執行下列命令：這會建立名為 `test-project` 的專案。

```
aws codebuild create-project --cli-input-json file://project.json
```

步驟 3：執行和檢視報告的結果

在本節中，您會執行先前建立專案的建置。在建置過程中，CodeBuild 會建立包含測試案例結果的報告。報告包含在您所指定的報告群組中。

1. 若要啟動組建，請執行下列命令。`test-report-project` 是上述建立的組建專案名稱。記下在輸出中出現的建置 ID。

```
aws codebuild start-build --project-name test-report-project
```

2. 執行下列命令，以取得建置的相關資訊，包括您報告的 ARN。對於 `<build-id>`，請指定您的建置 ID。在輸出的 `reportArns` 屬性中記下報告 ARN。

```
aws codebuild batch-get-builds --ids <build-id>
```

3. 執行下列命令以取得報告的詳細資訊。對於 `<report-arn>`，請指定您的報告 ARN。

```
aws codebuild batch-get-reports --report-arns <report-arn>
```

輸出看起來如下。此範例輸出指出有多少測試成功、失敗、略過、導致錯誤或傳回不明狀態。

```
{
  "reports": [
```

```

{
  "status": "FAILED",
  "reportGroupArn": "<report-group-arn>",
  "name": "<report-group-name>",
  "created": 1573324770.154,
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<amzn-s3-demo-bucket>",
      "path": "<path-to-your-report-results>",
      "packaging": "NONE",
      "encryptionKey": "<encryption-key>"
    }
  },
  "expired": 1575916770.0,
  "truncated": false,
  "executionId": "arn:aws:codebuild:us-west-2:123456789012:build/<name-of-build-project>:2c254862-ddf6-4831-a53f-6839a73829c1",
  "type": "TEST",
  "arn": "<report-arn>",
  "testSummary": {
    "durationInNanoSeconds": 6657770,
    "total": 11,
    "statusCounts": {
      "FAILED": 3,
      "SKIPPED": 7,
      "ERROR": 0,
      "SUCCEEDED": 1,
      "UNKNOWN": 0
    }
  }
}
],
"reportsNotFound": []
}

```

4. 執行下列命令來列出與報告測試案例相關的資訊。對於 *<report-arn>*，指定您報告的 ARN。對於選用 `--filter` 參數，您可以指定一個狀態結果 (SUCCEEDED、FAILED、SKIPPED、ERROR 或 UNKNOWN)。

```

aws codebuild describe-test-cases \
  --report-arn <report-arn> \
  --filter status=SUCCEEDED|FAILED|SKIPPED|ERROR|UNKNOWN

```

輸出看起來如下。

```
{
  "testCases": [
    {
      "status": "FAILED",
      "name": "Test case 1",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    },
    {
      "status": "SUCCEEDED",
      "name": "Test case 2",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    }
  ]
}
```

CodeBuild 的 Docker 範例

本節說明 Docker 和 之間的範例整合 AWS CodeBuild。

樣本	Description
CodeBuild 自訂映像範例中的 Docker	此範例使用 CodeBuild 和自訂 Docker 組建映像 (docker:dind 在 Docker Hub 中) 來建置和執行 Docker 映像。
CodeBuild 的 Docker 映像建置伺服器範例	此範例會將 Docker 組建卸載至受管映像組建伺服器。

樣本	Description
Windows Docker 建置 CodeBuild 的範例	此範例使用 CodeBuild 建置並執行 Windows Docker 映像。
CodeBuild 的「將 Docker 映像發佈至 Amazon ECR 映像儲存庫」範例	此範例會產生建置輸出 Docker 映像，然後將 Docker 映像推送至 Amazon Elastic Container Registry (Amazon ECR) 映像儲存庫。
具有 CodeBuild AWS Secrets Manager 範例的私有登錄檔	此範例說明如何使用存放在私有登錄檔中的 Docker 映像做為 CodeBuild 執行時間環境。

CodeBuild 自訂映像範例中的 Docker

下列範例會使用 和自訂 Docker 建置映像 (docker:dind 在 Docker Hub 中) 來建置 AWS CodeBuild 和執行 Docker 映像。

若要了解如何使用 CodeBuild 提供的建置映像搭配 Docker 支援來建置 Docker 映像，請參閱我們的 [「將 Docker 映像發佈至 Amazon ECR」範例](#)。

Important

執行此範例可能會導致 AWS 您的帳戶產生費用。這包括 CodeBuild 以及與 Amazon S3 相關的 AWS 資源和動作 AWS KMS，以及 CloudWatch Logs 的可能費用。如需詳細資訊，請參閱 [CodeBuild 定價](#)、[Amazon S3 定價](#)、[AWS Key Management Service 定價](#) 和 [Amazon CloudWatch 定價](#)。

主題

- [在自訂映像範例中執行 Docker](#)

在自訂映像範例中執行 Docker

使用下列程序在自訂映像範例中執行 Docker。如需此範例的詳細資訊，請參閱 [CodeBuild 自訂映像範例中的 Docker](#)。

在自訂映像範例中執行 Docker

1. 如本主題的 [目錄結構](#) 和 [檔案](#) 章節所述建立檔案，然後將其上傳至 S3 輸入儲存貯體或 AWS CodeCommit、GitHub 或 Bitbucket 儲存庫。

Important

請勿上傳 (*root directory name*)，僅上傳 (*root directory name*) 內的檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含這些檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 (*root directory name*) 新增至 ZIP 檔案，僅新增 (*root directory name*) 內的檔案即可。

2. 建立組建專案、執行組建，以及檢視相關的組建資訊。

如果您使用 AWS CLI 來建立建置專案，create-project 命令的 JSON 格式輸入可能會與此類似。(以您自己的值取代預留位置。)

```
{
  "name": "sample-docker-custom-image-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/DockerCustomImageSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "docker:dind",
    "computeType": "BUILD_GENERAL1_SMALL",
    "privilegedMode": false
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

Note

根據預設，非 VPC 建置會啟用 Docker 協助程式。如果您想要使用 Docker 容器進行 VPC 建置，請參閱 Docker 文件網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特殊權限模式。

- 若要查看組建結果，請在組建的日誌中尋找 Hello, World! 字串。如需詳細資訊，請參閱[檢視建置的詳細資訊](#)。

目錄結構

此範例假設此目錄結構。

```
(root directory name)
### buildspec.yml
### Dockerfile
```

檔案

此範例中使用之作業系統的基本映像是 Ubuntu。此範例使用這些檔案。

buildspec.yml (在 *(root directory name)* 中)

```
version: 0.2

phases:
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld echo "Hello, World!"
```

Dockerfile (在 *(root directory name)* 中)

```
FROM maven:3.3.9-jdk-8

RUN echo "Hello World"
```

CodeBuild 的 Docker 映像建置伺服器範例

下列範例會將 Docker 組建卸載至受管映像組建伺服器。您可以調整此範例，在 CodeBuild 專案組態中佈建專用且受管的 Docker 映像建置伺服器。請注意，在為專案主動執行組建時，佈建的執行個體處於作用中狀態，而當組建未執行時，執行個體會停止。佈建的執行個體會在回收前儲存長達一個月。如需詳細資訊，請參閱 [CodeBuild Docker 伺服器功能](#)。

Important

執行此範例可能會導致 AWS 您的帳戶產生費用。這包括 CodeBuild 以及與 Amazon S3 相關的 AWS 資源和動作 AWS KMS，以及 CloudWatch Logs 的可能費用。如需詳細資訊，請參閱 [CodeBuild 定價](#)、[Amazon S3 定價](#)、[AWS Key Management Service 定價](#) 和 [Amazon CloudWatch 定價](#)。

主題

- [設定 Docker 伺服器](#)

設定 Docker 伺服器

使用下列程序為管理 Docker 工作負載和存放 Docker 映像層的 CodeBuild 專案佈建專用運算環境。

設定 Docker 伺服器

1. 如本主題的 [目錄結構](#) 和 [檔案](#) 章節所述建立檔案，然後將其上傳至 S3 輸入儲存貯體或 AWS CodeCommit、GitHub 或 Bitbucket 儲存庫。

Important

請勿上傳 (*root directory name*)，僅上傳 (*root directory name*) 內的檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含這些檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 (*root directory name*) 新增至 ZIP 檔案，僅新增 (*root directory name*) 內的檔案即可。

2. 建立組建專案、執行組建，以及檢視相關的組建資訊：

- a. 在主控台的環境區段中，選擇其他組態，導覽至 Docker 伺服器組態，然後選取為此專案啟用 Docker 伺服器。然後，您可以選擇 Docker 伺服器運算類型並提供登錄檔登入資料。
- b. 如果您使用 AWS CLI 來建立建置專案，create-project 命令的 JSON 格式輸入可能會與此類似。(以您自己的值取代預留位置。)

```
{
  "name": "sample-docker-custom-image-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/DockerServerSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux-x86_64-standard:5.0",
    "computeType": "BUILD_GENERAL1_LARGE",
    "dockerServer": [
      {
        "computeType": "BUILD_GENERAL1_LARGE",
        "securityGroupIds": [ "security-groups-ID" ]
      }
    ]
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name"
}
```

 Note

針對 Docker 伺服器設定的安全群組應允許從專案中設定的 VPC 傳入網路流量。它們應該允許連接埠 9876 上的輸入。

3. 若要查看組建結果，請在組建的日誌中尋找 Hello, World! 字串。如需詳細資訊，請參閱[檢視建置的詳細資訊](#)。

目錄結構

此範例假設此目錄結構。

```
(root directory name)
### buildspec.yml
### Dockerfile
```

檔案

此範例中使用之作業系統的基本映像是 Ubuntu。此範例使用這些檔案。

buildspec.yml (在 *(root directory name)* 中)

```
version: 0.2

phases:
  build:
    commands:
      - docker buildx build -t helloworld --load .
      - docker run helloworld echo "Hello, World!"
```

Dockerfile (在 *(root directory name)* 中)

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

RUN echo "Hello World"
```

Windows Docker 建置 CodeBuild 的範例

下列範例使用 CodeBuild 建置並執行 Windows Docker 映像。

主題

- [執行 Windows Docker 組建範例](#)

執行 Windows Docker 組建範例

使用下列程序來執行 Windows Docker 組建。

執行 Windows Docker 組建範例

1. 如本主題的 [目錄結構](#) 和 [檔案](#) 章節所述建立檔案，然後將其上傳至 S3 輸入儲存貯體或 AWS CodeCommit、GitHub 或 Bitbucket 儲存庫。

⚠ Important

請勿上傳 (*root directory name*)，僅上傳 (*root directory name*) 內的檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含這些檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 (*root directory name*) 新增至 ZIP 檔案，僅新增 (*root directory name*) 內的檔案即可。

2. 建立WINDOWS_EC2機群。

如果您使用 AWS CLI 來建立機群，create-fleet命令的 JSON 格式輸入可能看起來像這樣。(以您自己的值取代預留位置。)

```
{
  "name": "fleet-name",
  "baseCapacity": 1,
  "environmentType": "WINDOWS_EC2",
  "computeType": "BUILD_GENERAL1_MEDIUM"
}
```

3. 建立組建專案、執行組建，以及檢視相關的組建資訊。

如果您使用 AWS CLI 來建立建置專案，create-project命令的 JSON 格式輸入可能會與此類似。(以您自己的值取代預留位置。)

```
{
  "name": "project-name",
  "source": {
    "type": "S3",
    "location": "bucket-name/DockerImageSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "WINDOWS_EC2",
    "image": "Windows",
    "computeType": "BUILD_GENERAL1_MEDIUM",
    "fleet": {
      "fleetArn": "fleet-arn"
    }
  }
}
```

```
  },  
  "serviceRole": "arn:aws:iam::account-ID:role/role-name"  
}
```

4. 若要查看組建結果，請在組建的日誌中尋找 Hello, World! 字串。如需詳細資訊，請參閱[檢視建置的詳細資訊](#)。

目錄結構

此範例假設此目錄結構。

```
(root directory name)  
### buildspec.yml  
### Dockerfile
```

檔案

此範例中使用的作業系統基礎映像為 `mcr.microsoft.com/windows/servercore:ltsc2022`。此範例使用這些檔案。

buildspec.yml (在 (*root directory name*) 中)

```
version: 0.2  
  
phases:  
  pre_build:  
    commands:  
      - docker build -t helloworld .  
  build:  
    commands:  
      - docker images  
      - docker run helloworld powershell -Command "Write-Host 'Hello World!'"
```

Dockerfile (在 (*root directory name*) 中)

```
FROM mcr.microsoft.com/windows/servercore:ltsc2022  
  
RUN powershell -Command "Write-Host 'Hello World'"
```

CodeBuild 的「將 Docker 映像發佈至 Amazon ECR 映像儲存庫」範例

此範例會產生 Docker 映像的建置輸出，然後將 Docker 映像推送至 Amazon Elastic Container Registry (Amazon ECR) 映像儲存庫。您可以調整此範例將 Docker 映像推送至 Docker Hub。如需詳細資訊，請參閱[將 'Publish Docker 映像修改為 Amazon ECR' 範例，以推送至 Docker Hub](#)。

若要了解如何使用自訂 Docker 建置映像 (Docker Hub 中的 `docker:dind`) 來建置 Docker 映像，請參閱我們的[自訂映像中的 Docker 範例](#)。

已參考 `golang:1.12` 來測試過此範例。

此範例使用新的多階段 Docker 組建功能，此功能會產生 Docker 映像作為組建輸出。然後，它會將 Docker 映像推送至 Amazon ECR 映像儲存庫。多階段 Docker 映像組建有助於縮小最終 Docker 映像。如需詳細資訊，請參閱[對 Docker 使用多階段組建](#)。

Important

執行此範例可能會導致 AWS 您的帳戶產生費用。這包括 AWS CodeBuild 與 Amazon S3、AWS KMS CloudWatch Logs 和 Amazon ECR 相關的 AWS 資源和動作的可能費用。如需詳細資訊，請參閱[CodeBuild 定價](#)、[Amazon S3 定價](#)、[AWS Key Management Service 定價](#)、[Amazon CloudWatch 定價](#)和 [Amazon Elastic Container Registry 定價](#)。

主題

- [執行「發佈 Docker 映像到 Amazon ECR」範例](#)
- [將 'Publish Docker 映像修改為 Amazon ECR' 範例，以推送至 Docker Hub](#)

執行「發佈 Docker 映像到 Amazon ECR」範例

使用下列程序來執行將 Docker 映像發佈至 Amazon ECR 的範例。如需此範例的詳細資訊，請參閱[CodeBuild 的「將 Docker 映像發佈至 Amazon ECR 映像儲存庫」範例](#)。

如何執行此範例

1. 如果您在 Amazon ECR 中已有要使用的映像儲存庫，請跳到步驟 3。否則，如果您使用使用者而非 AWS 根帳戶或管理員使用者來使用 Amazon ECR，請將此陳述式（介於 **### BEGIN ADDING STATEMENT HERE ###** 和 **### END ADDING STATEMENT HERE ###** 之間）新增至使用者（或使用者相關聯的 IAM 群組）。不建議使用 AWS 根帳戶。此陳述式允許建立 Amazon ECR 儲存庫來存放 Docker 映像。省略符號 (...) 用於簡化和協助您找到新增陳述式的位置。請

不要移除任何陳述式，也不要再在政策中輸入這些省略符號。如需詳細資訊，請參閱《使用者指南》中的[使用處理內嵌政策 AWS 管理主控台](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:CreateRepository"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

修改此政策的 IAM 實體必須在 IAM 中具有修改政策的許可。

2. 在 Amazon ECR 中建立映像儲存庫。請務必在建立建置環境和執行建置的相同 AWS 區域中建立儲存庫。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的[建立儲存庫](#)。此儲存庫的名稱必須符合您稍後於此程序中指定的儲存庫名稱 (以 IMAGE_REPO_NAME 環境變數表示)。確保 Amazon ECR 儲存庫政策授予 CodeBuild 服務 IAM 角色的映像推送存取權。
3. 將此陳述式 (**### BEGIN ADDING STATEMENT HERE ###** 和 **### END ADDING STATEMENT HERE ###** 之間) 新增至您連接至 AWS CodeBuild 服務角色的政策。此陳述式允許 CodeBuild 將 Docker 映像上傳至 Amazon ECR 儲存庫。省略符號 (...) 用於簡化和協助您找到新增陳述式的位置。請不要移除任何陳述式，也不要再在政策中輸入這些省略符號。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

        "Action": [
            "ecr:BatchCheckLayerAvailability",
            "ecr:CompleteLayerUpload",
            "ecr:GetAuthorizationToken",
            "ecr:InitiateLayerUpload",
            "ecr:PutImage",
            "ecr:UploadLayerPart"
        ],
        "Resource": "*"
    }
}

```

Note

修改此政策的 IAM 實體必須在 IAM 中具有修改政策的許可。

4. 如本主題的 [目錄結構](#) 和 [檔案](#) 章節所述建立檔案，然後將其上傳至 S3 輸入儲存貯體或 AWS CodeCommit、GitHub 或 Bitbucket 儲存庫。如需詳細資訊，請參閱 AWS CodePipeline 《使用者指南》中的 [映像定義檔案參考](#)。

Important

請勿上傳 (*root directory name*)，僅上傳 (*root directory name*) 內的檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含這些檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 (*root directory name*) 新增至 ZIP 檔案，僅新增 (*root directory name*) 內的檔案即可。

5. 建立組建專案、執行組建，以及檢視組建資訊。

如果您使用主控台來建立您的專案：

- a. 針對 Operating system (作業系統)，選擇 Ubuntu。
- b. 針對 Runtime (執行時間)，選擇 Standard (標準)。
- c. 針對映像，選擇 aws/codebuild/standard:5.0。
- d. 新增下列環境變數：

- 值為 *region-ID* 的 AWS_DEFAULT_REGION

- 值為 *account-ID* 的 `AWS_ACCOUNT_ID`
- 值為 `Latest` 的 `IMAGE_TAG`
- 值為 *Amazon-ECR-repo-name* 的 `IMAGE_REPO_NAME`

如果您使用 AWS CLI 建立建置專案，`create-project` 命令的 JSON 格式輸入可能看起來像這樣。(以您自己的值取代預留位置。)

```
{
  "name": "sample-docker-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/DockerSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
        "value": "region-ID"
      },
      {
        "name": "AWS_ACCOUNT_ID",
        "value": "account-ID"
      },
      {
        "name": "IMAGE_REPO_NAME",
        "value": "Amazon-ECR-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ],
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```



```
}
```

6. 確認 CodeBuild 已成功將 Docker 映像推送至儲存庫：

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 選擇儲存庫名稱。此映像應列在 Image tag (映像標籤) 欄中。

目錄結構

此範例假設此目錄結構。

```
(root directory name)
### buildspec.yml
### Dockerfile
```

檔案

此範例使用這些檔案。

buildspec.yml (在 *(root directory name)* 中)

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO_NAME:$IMAGE_TAG
```

Dockerfile (在 *(root directory name)* 中)

```
FROM golang:1.12-alpine AS build
#Install git
RUN apk add --no-cache git
#Get the hello world package from a GitHub repository
RUN go get github.com/golang/example/hello
WORKDIR /go/src/github.com/golang/example/hello
# Build the project and send the output to /bin/HelloWorld
RUN go build -o /bin/HelloWorld

FROM golang:1.12-alpine
#Copy the build's output binary from the previous build container
COPY --from=build /bin/HelloWorld /bin/HelloWorld
ENTRYPOINT ["/bin/HelloWorld"]
```

 Note

CodeBuild 會覆寫自訂 Docker 映像 ENTRYPOINT 的。

將 'Publish Docker 映像修改為 Amazon ECR' 範例，以推送至 Docker Hub

若要將「公開 Docker 影像調整為 Amazon ECR」範例，以便將 Docker 影像推送至 Docker Hub，而不是 Amazon ECR，請編輯範例程式碼。如需範例的詳細資訊，請參閱 [CodeBuild 的「將 Docker 映像發佈至 Amazon ECR 映像儲存庫」範例](#)和 [執行「發佈 Docker 映像到 Amazon ECR」範例](#)。

 Note

如果您使用的是 17.06 之前的 Docker 版本，請移除 `--no-include-email` 選項。

1. 在 `buildspec.yml` 檔案中取代這些 Amazon ECR 特定的程式碼行：

```
...
pre_build:
  commands:
    - echo Logging in to Amazon ECR...
    - aws ecr get-login-password --region $AWS_DEFAULT_REGION |
      docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.
        $AWS_DEFAULT_REGION.amazonaws.com
```

```
build:
  commands:
    - echo Build started on `date`
    - echo Building the Docker image...
    - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
    - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
      $AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
        $IMAGE_REPO_NAME:$IMAGE_TAG
  ...
```

換成 Docker Hub 專用的這幾行程式碼：

```
...
pre_build:
  commands:
    - echo Logging in to Docker Hub...
    # Type the command to log in to your Docker Hub account here.
build:
  commands:
    - echo Build started on `date`
    - echo Building the Docker image...
    - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
    - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $IMAGE_REPO_NAME:$IMAGE_TAG
  ...
```

2. 將編輯的程式碼上傳至 S3 輸入儲存貯體或 AWS CodeCommit、GitHub 或 Bitbucket 儲存庫。

Important

請勿上傳 (*root directory name*)，僅上傳 (*root directory name*) 內的檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含這些檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 (*root directory name*) 新增至 ZIP 檔案，僅新增 (*root directory name*) 內的檔案即可。

3. 將 create-project 命令中 JSON 格式輸入的這幾行程式碼：

```
...
  "environmentVariables": [
    {
      "name": "AWS_DEFAULT_REGION",
      "value": "region-ID"
    },
    {
      "name": "AWS_ACCOUNT_ID",
      "value": "account-ID"
    },
    {
      "name": "IMAGE_REPO_NAME",
      "value": "Amazon-ECR-repo-name"
    },
    {
      "name": "IMAGE_TAG",
      "value": "latest"
    }
  ]
...

```

換成這幾行程式碼：

```
...
  "environmentVariables": [
    {
      "name": "IMAGE_REPO_NAME",
      "value": "your-Docker-Hub-repo-name"
    },
    {
      "name": "IMAGE_TAG",
      "value": "latest"
    }
  ]
...

```

4. 建立建置環境、執行建置，以及檢視相關的建置資訊。
5. 確認 AWS CodeBuild 已成功將 Docker 映像推送至儲存庫。登入 Docker Hub，移至儲存庫，然後選擇 Tags (標籤) 索引標籤。latest 標籤應該包含最近的 Last Updated (上次更新) 值。

具有 CodeBuild AWS Secrets Manager 範例的私有登錄檔

此範例說明如何使用存放在私有登錄檔中的 Docker 映像做為您的 AWS CodeBuild 執行時間環境。私有登錄檔的登入資料會存放在 AWS Secrets Manager。任何私有登錄檔皆可與 CodeBuild 搭配使用。這個範例使用 Docker Hub。

Note

動作可以看到秘密，而且寫入檔案時不會遮罩秘密。

主題

- [私有登錄檔範例需求](#)
- [使用私有登錄檔建立 CodeBuild 專案](#)
- [設定自我託管執行器的私有登錄登入資料](#)

私有登錄檔範例需求

若要搭配使用私有登錄檔 AWS CodeBuild，您必須具有下列項目：

- 存放 Docker Hub 登入資料的 Secrets Manager 秘密。這些登入資料會用來存取您的私有儲存庫。

Note

您需要為建立的秘密付費。

- 私有儲存庫或帳戶。
- 授予 Secrets Manager 秘密存取權的 CodeBuild 服務角色 IAM 政策。

請依照下列步驟建立這些資源，然後使用存放在私有登錄檔中的 Docker 映像建立 CodeBuild 建置專案。

使用私有登錄檔建立 CodeBuild 專案

1. 如需如何建立免費私有儲存庫的詳細資訊，請參閱[儲存庫 Docker Hub](#)。您也可以終端機上執行以下命令來提取映像、取得其 ID，並將其推送到新的儲存庫。

```
docker pull amazonlinux
docker images amazonlinux --format {{.ID}}
docker tag image-id your-username/repository-name:tag
docker login
docker push your-username/repository-name
```

2. 請遵循AWS Secrets Manager 《使用者指南》中的[建立 AWS Secrets Manager 秘密](#)中的步驟。
 - a. 在步驟 3 中，選擇秘密類型，選擇其他類型的秘密。
 - b. 在鍵/值對中，為您的 Docker Hub 使用者名稱建立一個鍵/值對，為您的 Docker Hub 密碼建立一個鍵/值對。
 - c. 繼續遵循[建立 AWS Secrets Manager 秘密](#)中的步驟。
 - d. 在步驟 5 的設定自動輪換頁面上，將其關閉，因為金鑰對應至您的 Docker Hub 登入資料。
 - e. 請依照[建立 AWS Secrets Manager 秘密](#)中的步驟完成。

如需詳細資訊，請參閱[什麼是 AWS Secrets Manager ?](#)

3. 當您在主控台中建立 AWS CodeBuild 專案時，CodeBuild 會為您連接必要的許可。如果您使用 以外的 AWS KMS 金鑰DefaultEncryptionKey，則必須將其新增至服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[修改角色（主控台）](#)。

若要讓您的服務角色使用 Secrets Manager，它至少必須具有 `secretsmanager:GetSecretValue` 許可。

▼ KMS (1 action) Clone Remove

Service	KMS
Actions	Write Decrypt
Resources	☒ Specific ☐ All resources
key	<input type="text" value="arn:aws:kms:<region>:<account>:key/<your-key>"/> EDIT Any
Add ARN to restrict access	
Request conditions	Specify request conditions (optional)

4. 若要使用主控台搭配私有登錄檔中所儲存的环境來建立專案，請在建立專案的同時執行下列動作。如需相關資訊，請參閱[建立組建專案 \(主控台\)](#)。

 Note

如果您的私有登錄檔位於 VPC 中，則必須具有公有網際網路存取。CodeBuild 無法從 VPC 中的私有 IP 地址提取映像。

- a. 在環境映像中，選擇自訂映像。
- b. 針對 Environment type (環境類型)，選擇 Linux 或 Windows。
- c. 針對映像登錄檔，選擇其他登錄檔。
- d. 在外部登錄 URL 中，輸入映像位置，然後在登錄登入資料中 - 選擇性輸入 Secrets Manager 登入資料的 ARN 或名稱。

 Note

如果您目前所在區域不存在您的登入資料，則您必須使用 ARN。如果登入資料存在於不同的區域，則無法使用登入資料名稱。

設定自我託管執行器的私有登錄登入資料

使用下列指示來設定自我託管執行器的登錄登入資料。

 Note

請注意，只有在映像被私有登錄檔的映像覆寫時，才會使用這些登入資料。

AWS Management Console

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案或選取現有的專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [變更建置專案的設定 \(主控台\)](#)。
3. 在環境中，選擇其他組態。

4. 在其他組態中，輸入來自 的秘密名稱或 ARN AWS Secrets Manager for Registry 登入資料 - 選用。

Registry credential - optional

AWS CLI

1. 如果您想要建立新的專案，請執行 create-project 命令。

```
aws codebuild create-project \
  --name project-name \
  --source type=source-type,location=source-location \
  --environment "type=environment-type,image=image,computeType=compute-type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-name-or-arn},imagePullCredentialsType=CODEBUILD|SERVICE_ROLE" \
  --artifacts type=artifacts-type \
  --service-role arn:aws:iam::account-ID:role/service-role/service-role-name
```

2. 如果您想要更新現有的專案，請執行 update-project 命令。

```
aws codebuild update-project \
  --name project-name \
  --environment "type=environment-type,image=image,computeType=compute-type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-name-or-arn}"
```

使用託管於 S3 儲存貯體中的建置輸出建立靜態網站

您可以停用組建中的成品加密。建議您執行此作業，如此一來您就可以將成品發佈至設定為託管網站的位置。(您不能發佈加密成品。) 此範例說明如何使用 Webhooks 觸發建置，並將其成品發佈到設定為網站的 S3 儲存貯體。

1. 按照[設定靜態網站](#)中的說明，將 S3 儲存貯體設定為像網站一樣運作。
2. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
3. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。

4. 在 Project name (專案名稱) 中，輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。
5. 在 Source (來源) 中，針對 Source provider (來源供應商)，選擇 GitHub。請遵循說明來與 GitHub 連線 (或重新連線)，然後選擇 Authorize (授權)。

針對 Webhook，選取 Rebuild every time a code change is pushed to this repository (在每次將程式碼變更推送至此儲存庫時重建)。只有當您選擇 Use a repository in my account (使用帳戶中的儲存庫) 時，才能選取此核取方塊。

Source

Add source

Source 1 - Primary

Source provider

GitHub

Repository

☐ Public repository

☒ Repository in my GitHub account

GitHub repository

↻

Disconnect GitHub account

▼ Additional configuration

Git clone depth

Git clone depth - optional

1

Build Status - optional

☐ Report build statuses to source provider when your builds start and finish

Webhook - optional

☒ Rebuild every time a code change is pushed to this repository

Branch filter - optional

Enter a regular expression

6. 在 Environment (環境) 中：

針對 Environment image (環境映像)，執行下列其中一項作業：

- 若要使用由 管理的 Docker 映像 AWS CodeBuild，請選擇受管映像，然後從作業系統、執行期 (些)、映像和映像版本進行選擇。若可用，請從 Environment type (環境類型) 進行選擇。
- 若要使用另一個 Docker 映像，請選擇 Custom image (自訂映像)。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。如果您選擇其他登錄檔，請針對外部登錄檔 URL，使用 格式，在 Docker Hub 中輸入 Docker 映像檔的名稱和標籤 *docker repository/docker image name*。如果您選擇 Amazon ECR，請使用 Amazon ECR 儲存庫和 Amazon ECR 映像來選擇 AWS 帳戶中的 Docker 映像。
- 若要使用私有 Docker 映像，請選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。針對 Image registry (映像登錄) 選擇 Other registry (其他登錄)，然後輸入私人 Docker 映像的憑證的 ARN。登入資料必須由 Secrets Manager 建立。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [What Is AWS Secrets Manager](#)。

7. 在 Service role (服務角色) 中，執行下列其中一項作業：

- 如果您沒有 CodeBuild 服務角色，請選擇新服務角色。在角色名稱中，輸入新角色的名稱。
- 如果您有 CodeBuild 服務角色，請選擇現有服務角色。在角色 ARN 中，選擇服務角色。

Note

當您使用 主控台 建立或更新組建專案時，您可以同時建立 CodeBuild 服務角色。根據預設，此角色只能與該建置專案搭配運作。如果您使用主控台將此服務角色與另一個建置專案建立關聯，則會更新此角色以與其他建置專案搭配運作。服務角色最多可以與 10 個組建專案搭配運作。

8. 在 Buildspec 中，執行下列其中一項操作：

- 選擇使用 buildspec 檔案以使用來源碼根目錄中的 buildspec.yml 檔案。
- 選擇插入建置命令以使用主控台插入建置命令。

如需更多資訊，請參閱 [Buildspec 參考](#)。

9. 在成品中，針對類型選擇 Amazon S3，將建置輸出存放在 S3 儲存貯體中。

10. 針對 Bucket name (儲存貯體名稱)，選擇您設定做為網站的 S3 儲存貯體的名稱。

11. 如果您選擇在環境中插入組建命令，則針對輸出檔案，輸入您要放入輸出儲存貯體之組建中檔案的位置。如果您有多個位置，請使用逗號分隔每個位置 (例如 `appspec.yml`, `target/my-app.jar`)。如需詳細資訊，請參閱[Artifacts reference-key in the buildspec file](#)。
12. 選取 Disable artifacts encryption (停用成品加密)。
13. 展開 Additional configuration (其他組態)，並適當地選擇選項。
14. 選擇 Create build project (建立建置專案)。在組建專案頁面上，在 Build history (組建歷史記錄) 中選擇 Start build (啟動組建) 來執行組建。
15. (選用) 遵循《[Amazon S3 開發人員指南](#)》中的範例：[使用 Amazon CloudFront 加速您的網站](#)。Amazon S3

多個輸入來源和輸出成品範例

您可以使用多個輸入來源和一組以上的輸出成品來建立 AWS CodeBuild 組建專案。這個範例示範如何設定如下的組建專案：

- 使用不同類型的多個來源和儲存庫。
- 在單一建置中將建置成品發佈至多個 S3 儲存貯體。

在下列範例中，您會建立建置專案，並使用它來執行建置。此範例使用組建專案的 buildspec 檔案，示範如何納入多個來源並建立多組成品。

若要了解如何建立管道，使用 CodeBuild 的多個來源輸入來建立多個輸出成品，請參閱 [CodePipeline/CodeBuild 與多個輸入來源和輸出成品整合的範例](#)。

主題

- [建立具有多個輸入和輸出的建置專案](#)
- [建立不含來源的建置專案](#)

建立具有多個輸入和輸出的建置專案

使用下列程序建立具有多個輸入和輸出的建置專案。

建立具有多個輸入和輸出的建置專案

1. 將您的來源上傳至一或多個 S3 儲存貯體、CodeCommit、GitHub、GitHub Enterprise Server 或 Bitbucket 儲存庫。

2. 選擇哪個來源是主要來源。這是 CodeBuild 在其中尋找並執行 buildspec 檔案的來源。
3. 建立建置專案。如需詳細資訊，請參閱[在 中建立建置專案 AWS CodeBuild](#)。
4. 建立建置專案、執行建置，並取得建置的相關資訊。
5. 如果您使用 AWS CLI 建立建置專案，create-project 命令的 JSON 格式輸入看起來可能會類似以下內容：

```
{
  "name": "sample-project",
  "source": {
    "type": "S3",
    "location": "<bucket/sample.zip>"
  },
  "secondarySources": [
    {
      "type": "CODECOMMIT",
      "location": "https://git-codecommit.us-west-2.amazonaws.com/v1/repos/repo",
      "sourceIdentifier": "source1"
    },
    {
      "type": "GITHUB",
      "location": "https://github.com/awslabs/aws-codebuild-jenkins-plugin",
      "sourceIdentifier": "source2"
    }
  ],
  "secondaryArtifacts": [ss
    {
      "type": "S3",
      "location": "<output-bucket>",
      "artifactIdentifier": "artifact1"
    },
    {
      "type": "S3",
      "location": "<other-output-bucket>",
      "artifactIdentifier": "artifact2"
    }
  ],
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
```

```
"encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

您的主要來源是在 `source` 屬性下方定義。其他所有來源稱為次要來源，出現在 `secondarySources` 下方。所有次要來源安裝在其自己的目錄中。此目錄存放在內建環境變數 `CODEBUILD_SRC_DIR_sourceIdentifier` 中。如需詳細資訊，請參閱[建置環境中的環境變數](#)。

`secondaryArtifacts` 屬性包含成品定義清單。這些成品使用 `buildspec` 檔案的 `secondary-artifacts` 區塊 (巢狀於 `artifacts` 區塊內)。

`buildspec` 檔案中的次要成品具有與成品相同的結構，且依成品識別符區隔。

Note

在 [CodeBuild API](#) 中，次要成品 `artifactIdentifier` 上的是 `CreateProject` 和 `UpdateProject` 中的必要屬性。必須用來參考次要成品。

使用前述的 JSON 格式輸入時，專案的 `buildspec` 檔案可能如下所示：

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: openjdk11
  build:
    commands:
      - cd $CODEBUILD_SRC_DIR_source1
      - touch file1
      - cd $CODEBUILD_SRC_DIR_source2
      - touch file2

artifacts:
  files:
    - '**.*'
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR_source1
      files:
        - file1
```

```
artifact2:
  base-directory: $CODEBUILD_SRC_DIR_source2
  files:
    - file2
```

您可以使用 API 搭配 StartBuild 中的 sourceVersion 屬性，以覆寫主要來源的版本。若要覆寫一個或多個次要來源版本，請使用 secondarySourceVersionOverride 屬性。

中start-build命令的 JSON 格式輸入 AWS CLI 可能如下所示：

```
{
  "projectName": "sample-project",
  "secondarySourcesVersionOverride": [
    {
      "sourceIdentifier": "source1",
      "sourceVersion": "codecommit-branch"
    },
    {
      "sourceIdentifier": "source2",
      "sourceVersion": "github-branch"
    },
  ]
}
```

建立不含來源的建置專案

您可以在設定**NO_SOURCE**來源時選擇來源類型來設定 CodeBuild 專案。當來源類型為 **NO_SOURCE** 時，您將無法指定 buildspec 檔案，因為專案並無任何來源。相反地，您必須在 create-project CLI 命令的 JSON 格式輸入的 buildspec 屬性中指定 YAML 格式 buildspec 字串。這看起來類似下述：

```
{
  "name": "project-name",
  "source": {
    "type": "NO_SOURCE",
    "buildspec": "version: 0.2\n\nphases:\n  build:\n    commands:\n      - command"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL",
```

```
},  
"serviceRole": "arn:aws:iam::account-ID:role/role-name",  
"encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"  
}
```

如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。

CodeBuild 的 buildspec 檔案範例中的執行時間版本

如果您使用 Amazon Linux 2 (AL2) 標準映像版本 1.0 或更新版本，或 Ubuntu 標準映像版本 2.0 或更新版本，您可以在 buildspec 檔案的 runtime-versions 區段中指定一或多個執行時間。下列範例示範如何變更專案執行時間、指定多個執行時間，以及指定相依於另一個執行時間的執行時間。如需支援的執行時間資訊，請參閱[CodeBuild 提供的 Docker 映像](#)。

Note

如果您在建置容器中使用 Docker，您的建置必須以特殊權限模式執行。如需詳細資訊，請參閱[手動執行 AWS CodeBuild 組建及在 中建立建置專案 AWS CodeBuild](#)。

主題

- [更新 buildspec 檔案中的執行時間版本](#)
- [指定兩個執行時間](#)

更新 buildspec 檔案中的執行時間版本

您可以透過更新 buildspec 檔案的 runtime-versions 區段，將專案使用的執行時間修改為新版本。以下範例說明如何指定 Java 版本 8 和 11。

- 指定 Java 版本 8 的 runtime-versions 區段：

```
phases:  
  install:  
    runtime-versions:  
      java: corretto8
```

- 指定 Java 版本 11 的 runtime-versions 區段：

```
phases:
  install:
    runtime-versions:
      java: corretto11
```

下列範例示範如何使用 Ubuntu 標準映像 5.0 或 Amazon Linux 2 標準映像 3.0 指定不同版本的 Python：

- 指定 Python 3.7 版的 runtime-versions 區段：

```
phases:
  install:
    runtime-versions:
      python: 3.7
```

- 指定 Python 3.8 版的 runtime-versions 區段：

```
phases:
  install:
    runtime-versions:
      python: 3.8
```

此範例示範的專案最初使用 Java 8 執行時間，然後更新為 Java 版本 10 執行時間。

- 下載並安裝 Maven。如需詳細資訊，請參閱 Apache Maven 網站上的[下載 Apache Maven](#) 和[安裝 Apache Maven](#)。
- 切換至您本機電腦或執行個體上的空白目錄，然後執行此 Maven 命令。

```
mvn archetype:generate "-DgroupId=com.mycompany.app" "-DartifactId=ROOT" "-DarchetypeArtifactId=maven-archetype-webapp" "-DinteractiveMode=false"
```

如果成功，則會建立此目錄結構和檔案。

```
.
### ROOT
  ### pom.xml
  ### src
    ### main
```



```
### resources
### webapp
### WEB-INF
#   ### web.xml
### index.jsp
```

3. 使用下列內容建立名為 `buildspec.yml` 的檔案。將檔案存放於 *(root directory name)*/`my-web-app` 目錄中。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto8
  build:
    commands:
      - java -version
      - mvn package
artifacts:
  files:
    - '**/*'
  base-directory: 'target/my-web-app'
```

在 `buildspec` 檔案中：

- `runtime-versions` 區段指定專案使用 Java 執行時間版本 8。
- `- java -version` 命令會顯示您的專案在建置時使用的 Java 版本。

您的檔案結構現在看起來應如下。

```
(root directory name)
### my-web-app
### src
#   ### main
#   ### resources
#   ### webapp
#       ### WEB-INF
#           ### web.xml
#               ### index.jsp
### buildspec.yml
```

```
### pom.xml
```

- 將my-web-app目錄的內容上傳至 S3 輸入儲存貯體或 CodeCommit、GitHub 或 Bitbucket 儲存庫。

⚠ Important

請勿上傳 *(root directory name)* 或 *(root directory name)/my-web-app*，僅上傳 *(root directory name)/my-web-app* 中的目錄和檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含目錄結構和檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 *(root directory name)* 或 *(root directory name)/my-web-app* 新增至 ZIP 檔案，僅新增 *(root directory name)/my-web-app* 中的目錄和檔案即可。

- 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
- 建立建置專案。如需詳細資訊，請參閱[建立組建專案 \(主控台\)](#)及[執行建置 \(主控台\)](#)。除了下列設定外，保留所有設定的預設值。
 - 針對 Environment (環境)：
 - 針對 Environment image (環境映像)，選擇 Managed image (受管映像)。
 - 針對 Operating system (作業系統)，請選擇 Amazon Linux 2。
 - 針對 Runtime(s) (執行時間)，選擇 Standard (標準)。
 - 針對映像，選擇 aws/codebuild/amazonlinux-x86_64-standard : 4.0。
- 選擇 Start build (開始組建)。
- 在 Build configuration (組建組態) 上，接受預設值，然後選擇 Start build (開始組建)。
- 建置完成後，在 Build logs (建置日誌) 索引標籤上檢視建置輸出。您應該會看到類似下列的輸出：

```
[Container] Date Time Phase is DOWNLOAD_SOURCE
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src
[Container] Date Time YAML location is /codebuild/output/src460614277/src/buildspec.yml
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'java' runtime version 'corretto8' based on manual selections...
[Container] Date Time Running command echo "Installing Java version 8 ..."
Installing Java version 8 ...
```

```
[Container] Date Time Running command export JAVA_HOME="$JAVA_8_HOME"

[Container] Date Time Running command export JRE_HOME="$JRE_8_HOME"

[Container] Date Time Running command export JDK_HOME="$JDK_8_HOME"

[Container] Date Time Running command for tool_path in "$JAVA_8_HOME"/bin/*
"$JRE_8_HOME"/bin/*;
```

10. 將 runtime-versions 區段更新為 Java 版本 11：

```
install:
  runtime-versions:
    java: corretto11
```

11. 在您儲存變更後，再次執行您的建置並檢視建置輸出。您應該會看到已安裝的 Java 版本是 11。您應該會看到類似下列的輸出：

```
[Container] Date Time Phase is DOWNLOAD_SOURCE
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src
[Container] Date Time YAML location is /codebuild/output/src460614277/src/
buildspec.yml
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'java' runtime version 'corretto11' based on manual
selections...
Installing Java version 11 ...

[Container] Date Time Running command export JAVA_HOME="$JAVA_11_HOME"

[Container] Date Time Running command export JRE_HOME="$JRE_11_HOME"

[Container] Date Time Running command export JDK_HOME="$JDK_11_HOME"

[Container] Date Time Running command for tool_path in "$JAVA_11_HOME"/bin/*
"$JRE_11_HOME"/bin/*;
```

指定兩個執行時間

您可以在相同的 CodeBuild 組建專案中指定多個執行時間。此範例專案使用兩個原始檔案：一個使用 Go 執行時間，另一個使用 Node.js 執行時間。

1. 建立名為 my-source 的目錄。
2. 在 my-source 目錄中，建立名為 golang-app 的目錄。
3. 使用下列內容建立名為 hello.go 的檔案。將檔案存放於 golang-app 目錄中。

```
package main
import "fmt"

func main() {
    fmt.Println("hello world from golang")
    fmt.Println("1+1 =", 1+1)
    fmt.Println("7.0/3.0 =", 7.0/3.0)
    fmt.Println(true && false)
    fmt.Println(true || false)
    fmt.Println(!true)
    fmt.Println("good bye from golang")
}
```

4. 在 my-source 目錄中，建立名為 nodejs-app 的目錄。它所在的層級應該與 golang-app 目錄相同。
5. 使用下列內容建立名為 index.js 的檔案。將檔案存放於 nodejs-app 目錄中。

```
console.log("hello world from nodejs");
console.log("1+1 =" + (1+1));
console.log("7.0/3.0 =" + 7.0/3.0);
console.log(true && false);
console.log(true || false);
console.log(!true);
console.log("good bye from nodejs");
```

6. 使用下列內容建立名為 package.json 的檔案。將檔案存放於 nodejs-app 目錄中。

```
{
  "name": "mycompany-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"run some tests here\""
  },
  "author": "",
  "license": "ISC"
```

```
}
```

7. 使用下列內容建立名為 `buildspec.yml` 的檔案。將檔案存放於 `my-source` 目錄中，層級與 `nodejs-app` 和 `golang-app` 目錄相同。`runtime-versions` 區段指定 Node.js 版本 12 和 Go 版本 1.13 執行時間。

```
version: 0.2

phases:
  install:
    runtime-versions:
      golang: 1.13
      nodejs: 12
  build:
    commands:
      - echo Building the Go code...
      - cd $CODEBUILD_SRC_DIR/golang-app
      - go build hello.go
      - echo Building the Node code...
      - cd $CODEBUILD_SRC_DIR/nodejs-app
      - npm run test
  artifacts:
    secondary-artifacts:
      golang_artifacts:
        base-directory: golang-app
        files:
          - hello
      nodejs_artifacts:
        base-directory: nodejs-app
        files:
          - index.js
          - package.json
```

8. 您的檔案結構現在看起來應如下。

```
my-source
### golang-app
#   ### hello.go
### nodejs.app
#   ### index.js
#   ### package.json
### buildspec.yml
```

9. 將my-source目錄的內容上傳至 S3 輸入儲存貯體或 CodeCommit、GitHub 或 Bitbucket 儲存庫。

 Important

如果您使用的是 S3 輸入儲存貯體，請務必建立包含目錄結構和檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 my-source 新增至 ZIP 檔案，僅新增 my-source 中的目錄和檔案即可。

10. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
11. 建立建置專案。如需詳細資訊，請參閱[建立組建專案 \(主控台\)](#)及[執行建置 \(主控台\)](#)。除了下列設定外，保留所有設定的預設值。
 - 針對 Environment (環境)：
 - 針對 Environment image (環境映像)，選擇 Managed image (受管映像)。
 - 針對 Operating system (作業系統)，請選擇 Amazon Linux 2。
 - 針對 Runtime(s) (執行時間)，選擇 Standard (標準)。
 - 針對映像，選擇 aws/codebuild/amazonlinux-x86_64-standard : 4.0。
12. 選擇 Create build project (建立建置專案)。
13. 選擇 Start build (開始組建)。
14. 在 Build configuration (組建組態) 上，接受預設值，然後選擇 Start build (開始組建)。
15. 建置完成後，在 Build logs (建置日誌) 索引標籤上檢視建置輸出。您應該會看到類似下列的輸出。其中顯示 Go 和 Node.js 執行時間的輸出。還顯示 Go 和 Node.js 應用程式的輸出。

```
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'golang' runtime version '1.13' based on manual
selections...
[Container] Date Time Selecting 'nodejs' runtime version '12' based on manual
selections...
[Container] Date Time Running command echo "Installing Go version 1.13 ..."
Installing Go version 1.13 ...

[Container] Date Time Running command echo "Installing Node.js version 12 ..."
Installing Node.js version 12 ...

[Container] Date Time Running command n $NODE_12_VERSION
installed : v12.20.1 (with npm 6.14.10)
```

```
[Container] Date Time Moving to directory /codebuild/output/src819694850/src
[Container] Date Time Registering with agent
[Container] Date Time Phases found in YAML: 2
[Container] Date Time  INSTALL: 0 commands
[Container] Date Time  BUILD: 1 commands
[Container] Date Time Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase INSTALL
[Container] Date Time Phase complete: INSTALL State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase PRE_BUILD
[Container] Date Time Phase complete: PRE_BUILD State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase BUILD
[Container] Date Time Running command echo Building the Go code...
Building the Go code...

[Container] Date Time Running command cd $CODEBUILD_SRC_DIR/golang-app

[Container] Date Time Running command go build hello.go

[Container] Date Time Running command echo Building the Node code...
Building the Node code...

[Container] Date Time Running command cd $CODEBUILD_SRC_DIR/nodejs-app

[Container] Date Time Running command npm run test

> mycompany-app@1.0.0 test /codebuild/output/src924084119/src/nodejs-app
> echo "run some tests here"

run some tests here
```

使用的來源版本範例 AWS CodeBuild

此範例示範如何使用遞交 ID 以外的格式 (也稱為遞交 SHA) 來指定來源的版本。您可以透過下列方式指定來源的版本：

- 對於 Amazon S3 來源供應商，請使用代表建置輸入 ZIP 檔案之物件的版本 ID。
- 對於 CodeCommit、Bitbucket、GitHub 和 GitHub Enterprise Server，請使用下列其中一項：

- 提取請求做為提取請求參考 (例如, `refs/pull/1/head`)。
- 分支做為分支名稱。
- 遞交 ID。
- 標籤。
- 參考和遞交 ID。參考可為下列其中之一：
 - 標籤 (例如, `refs/tags/mytagv1.0^{full-commit-SHA}`)。
 - 分支 (例如, `refs/heads/mydevbranch^{full-commit-SHA}`)。
 - 提取請求 (例如, `refs/pull/1/head^{full-commit-SHA}`)。
- 對於 GitLab 和 GitLab 自我管理, 請使用下列其中一項：
 - 分支做為分支名稱。
 - 遞交 ID。
 - 標籤。

Note

只有當您的儲存庫是 GitHub 或 GitHub Enterprise Server 時, 才可以指定提取請求來源的版本。

如果您使用參考和遞交 ID 來指定版本, 建置的 `DOWNLOAD_SOURCE` 階段會比您僅提供版本來得更快速。這是因為當您新增參考時, CodeBuild 不需要下載整個儲存庫來尋找遞交。

- 您可以指定僅具有遞交 ID 的來源版本, 例如 `12345678901234567890123467890123456789`。如果您這樣做, CodeBuild 必須下載整個儲存庫才能尋找版本。
- 您可以指定具有參考和遞交 ID 的來源版本, 以此格式: `refs/heads/branchname^{full-commit-SHA}` (例如, `refs/heads/main^{12345678901234567890123467890123456789}`)。如果您這樣做, CodeBuild 只會下載指定的分支來尋找版本。

Note

若要加速建置的 `DOWNLOAD_SOURCE` 階段, 您也可以將 Git 複製深度設定為低數量。CodeBuild 會下載較少版本的儲存庫。

主題

- [使用遞交 ID 指定 GitHub 儲存庫版本](#)
- [使用參考和遞交 ID 指定 GitHub 儲存庫版本](#)

使用遞交 ID 指定 GitHub 儲存庫版本

您可以指定僅具有遞交 ID 的來源版本，例如 12345678901234567890123467890123456789。如果您這樣做，CodeBuild 必須下載整個儲存庫才能尋找版本。

指定具有遞交 ID 的 GitHub 儲存庫版本

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。除了下列設定外，保留所有設定的預設值：
 - 在 Source (來源) 中：
 - 針對 Source provider (來源供應商)，選擇 GitHub。如果您未連接至 GitHub，請遵循指示來連接。
 - 針對 Repository (儲存庫)，選擇 Public repository (公有儲存庫)。
 - 針對 Repository URL (儲存庫 URL)，輸入 **`https://github.com/aws/aws-sdk-ruby.git`**。
 - 在 Environment (環境) 中：
 - 針對 Environment image (環境映像)，選擇 Managed image (受管映像)。
 - 針對 Operating system (作業系統)，請選擇 Amazon Linux 2。
 - 針對 Runtime(s) (執行時間)，選擇 Standard (標準)。
 - 針對映像，選擇 `aws/codebuild/amazonlinux-x86_64-standard : 4.0`。
3. 針對 Build specifications (組建規格)，選擇 Insert build commands (插入組建命令)，然後選擇 Switch to editor (切換到編輯器)。
4. 在 Build commands (組建命令) 中，將預留位置文字更換為以下內容：

```
version: 0.2
```

```
phases:
```

```
  install:
```

```
runtime-versions:
  ruby: 2.6
build:
  commands:
    - echo $CODEBUILD_RESOLVED_SOURCE_VERSION
```

當您使用 Ubuntu 標準映像 2.0，則需要 `runtime-versions` 區段。在此處，指定了 Ruby 版本 2.6 執行階段，但您可以使用任何執行時間。`echo` 命令會顯示存放在 `CODEBUILD_RESOLVED_SOURCE_VERSION` 環境變數中原始程式碼的版本。

5. 在 Build configuration (組建組態) 上，接受預設值，然後選擇 Start build (開始組建)。
 6. 對於 Source version (來源版本)，輸入 **046e8b67481d53bdc86c3f6affdd5d1afae6d369**。這是 <https://github.com/aws/aws-sdk-ruby>.git 儲存庫中遞交的 SHA。
 7. 選擇 Start build (開始組建)。
 8. 當組建完成時，您應該會看到下列：
- 在 Build logs (組建日誌) 標籤上，使用的專案來源的版本。請見此處範例。

```
[Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION
046e8b67481d53bdc86c3f6affdd5d1afae6d369
```

```
[Container] Date Time Phase complete: BUILD State: SUCCEEDED
```

- 在 Environment variables (環境變數) 標籤上，Resolved source version (已解決的來源版本) 會符合用來建立組建的遞交 ID。
- 在 Phase details (階段詳細資訊) 標籤上，DOWNLOAD_SOURCE 階段的持續時間。

使用參考和遞交 ID 指定 GitHub 儲存庫版本

您可以指定具有參考和遞交 ID 的來源版本，以此格式：`refs/heads/branchname^{full-commit-SHA}` (例如，`refs/heads/main^{123456789012345678901234567890123456789}`)。如果您這樣做，CodeBuild 只會下載指定的分支來尋找版本。

使用參考和遞交 ID 指定 GitHub 儲存庫版本。

1. 完成「[使用遞交 ID 指定 GitHub 儲存庫版本](#)」中的步驟。
2. 從左側導覽窗格，選擇 Build projects (組建專案)，然後選擇您稍早建立的專案。

3. 選擇 Start build (開始組建)。
4. 在 Source version (來源版本) 中，輸入 **refs/heads/main^{046e8b67481d53bdc86c3f6affdd5d1afae6d369}**。這是相同的遞交 ID 和分支的參考，格式為 **refs/heads/branchname^{full-commit-SHA}**。
5. 選擇 Start build (開始組建)。
6. 當組建完成時，您應該會看到下列：
 - 在 Build logs (組建日誌) 標籤上，使用的專案來源的版本。請見此處範例。

```
[Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION
046e8b67481d53bdc86c3f6affdd5d1afae6d369
```

```
[Container] Date Time Phase complete: BUILD State: SUCCEEDED
```

- 在 Environment variables (環境變數) 標籤上，Resolved source version (已解決的來源版本) 會符合用來建立組建的遞交 ID。
- 在 Phase details (階段詳細資訊) 標籤上，DOWNLOAD_SOURCE 階段的持續時間應該較您僅使用遞交 ID 來指定來源的版本時更短。

CodeBuild 的第三方來源儲存庫範例

本節說明第三方來源儲存庫與 CodeBuild 之間的範例整合。

樣本	Description
BitBucket pull 請求和 Webhook 篩選條件範例 – 請參閱 執行 CodeBuild 的「Bitbucket pull request and Webhook filter」範例	此範例示範如何使用 Bitbucket 儲存庫建立提取請求。它還說明如何使用 Bitbucket Webhook 來觸發 CodeBuild 來建立專案的建置。
GitHub Enterprise Server 範例 – 請參閱 執行 CodeBuild 的 GitHub Enterprise Server 範例	此範例示範如何在 GitHub Enterprise Server 儲存庫安裝憑證時設定 CodeBuild 專案。它還顯示如何啟用 Webhook，以便 CodeBuild 每次將程式碼變更推送到您的 GitHub Enterprise Server 儲存庫時重建原始程式碼。

樣本	Description
GitHub 提取請求和 Webhook 篩選條件範例 – 請參閱 執行 CodeBuild 的 GitHub 提取請求和 Webhook 篩選條件範例	此範例說明如何使用 GitHub Enterprise Server 儲存庫建立提取請求。它還顯示如何啟用 Webhook，以便 CodeBuild 每次將程式碼變更推送到您的 GitHub Enterprise Server 儲存庫時重建原始程式碼。

執行 CodeBuild 的「Bitbucket pull request and Webhook filter」範例

AWS CodeBuild 當來源儲存庫為 Bitbucket 時，支援 Webhook。這表示對於在 Bitbucket 儲存庫中存放其原始碼的 CodeBuild 組建專案，Webhook 可用來在每次程式碼變更推送至儲存庫時重建原始碼。如需詳細資訊，請參閱[Bitbucket Webhook 事件](#)。

此範例示範如何使用 Bitbucket 儲存庫建立提取請求。它還說明如何使用 Bitbucket Webhook 來觸發 CodeBuild 來建立專案的建置。

Note

使用 Webhook 時，使用者可能會觸發非預期的建置。若要降低此風險，請參閱 [使用 Webhook 的最佳實務](#)。

主題

- [先決條件](#)
- [步驟 1：使用 Bitbucket 建立建置專案並啟用 Webhook](#)
- [步驟 2：使用 Bitbucket Webhook 觸發組建](#)

先決條件

若要執行此範例，您必須將 AWS CodeBuild 專案與您的 Bitbucket 帳戶連線。

Note

CodeBuild 已更新其使用 Bitbucket 的許可。如果您之前已將專案連線至 Bitbucket，現在收到 Bitbucket 連線錯誤，您必須重新連線以授予 CodeBuild 許可來管理您的 Webhook。

步驟 1：使用 Bitbucket 建立建置專案並啟用 Webhook

下列步驟說明如何使用 Bitbucket 做為來源儲存庫建立 AWS CodeBuild 專案，並啟用 Webhook。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。
3. 選擇 Create build project (建立建置專案)。
4. 在 Project configuration (專案組態) 中：

Project name (專案名稱)

輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。

5. 在 Source (來源) 中：

來源提供者

選擇 Bitbucket。依照指示來連接（或重新連線）與 Bitbucket，然後選擇授權。

儲存庫

在我的 Bitbucket 帳戶中選擇儲存庫。

如果您先前尚未連線至 Bitbucket 帳戶，請輸入您的 Bitbucket 使用者名稱和密碼，然後選取儲存 Bitbucket 登入資料。對於 Bitbucket API 字符，請改為輸入您的 Atlassian 帳戶電子郵件地址和 API 字符。

Bitbucket 儲存庫

輸入 Bitbucket 儲存庫的 URL。

6. 在主要來源 Webhook 事件中，選取下列項目。

Note

只有在您在上一個步驟的我的 Bitbucket 帳戶中選擇儲存庫時，才會顯示主要來源 Webhook 事件區段。

1. 當您建立專案時，請選取 Rebuild every time a code change is pushed to this repository (在每次將程式碼變更推送至此儲存庫時重建)。
2. 從 Event type (事件類型)，選擇一或多個事件。
3. 若要篩選事件觸發組建的時間，請在 Start a build under these conditions (在這些情況下開始組建) 下新增一或多個選用的篩選條件。
4. 若要篩選何時不觸發事件，請在 Don't start a build under these conditions (在這些情況下不開始組建) 下新增一或多個選用的篩選條件。
5. 如有需要，請選擇新增篩選條件群組以新增另一個篩選條件群組。

如需 Bitbucket Webhook 事件類型和篩選條件的詳細資訊，請參閱 [Bitbucket Webhook 事件](#)。

7. 在 Environment (環境) 中：

環境映像

選擇下列其中一項：

若要使用由 管理的 Docker 映像 AWS CodeBuild：

選擇受管映像，然後從作業系統、執行時間、映像和映像版本進行選擇。若可用，請從 Environment type (環境類型) 進行選擇。

若要使用另一個 Docker 映像：

選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。如果您選擇其他登錄檔，請針對外部登錄檔 URL，使用 格式，在 Docker Hub 中輸入 Docker 映像檔的名稱和標籤 *docker repository/docker image name*。如果您選擇 Amazon ECR，請使用 Amazon ECR 儲存庫和 Amazon ECR 映像來選擇 AWS 帳戶中的 Docker 映像。

若要使用私有 Docker 映像：

選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。針對 Image registry (映像登錄) 選擇 Other registry (其他登錄)，然後輸入私人 Docker 映像的憑證的 ARN。登入資料必須由 Secrets Manager 建立。如需詳細資訊，請參閱 [什麼是 AWS Secrets Manager ?](#) AWS Secrets Manager 《使用者指南》中的。

服務角色

選擇下列其中一項：

- 如果您沒有 CodeBuild 服務角色，請選擇新服務角色。在角色名稱中，輸入新角色的名稱。
- 如果您有 CodeBuild 服務角色，請選擇現有服務角色。在角色 ARN 中，選擇服務角色。

 Note

當您使用主控台建立或更新組建專案時，您可以同時建立 CodeBuild 服務角色。根據預設，此角色只能與該建置專案搭配運作。如果您使用主控台將此服務角色與另一個建置專案建立關聯，則會更新此角色以與其他建置專案搭配運作。服務角色最多可以與 10 個組建專案搭配運作。

8. 在 Buildspec 中，執行下列其中一項操作：

- 選擇使用 buildspec 檔案以使用來源碼根目錄中的 buildspec.yml 檔案。
- 選擇插入建置命令以使用主控台插入建置命令。

如需更多資訊，請參閱[Buildspec 參考](#)。

9. 在 Artifacts (成品) 中：

類型

選擇下列其中一項：

- 如果您不要建立建置輸出成品，則請選擇 No artifacts (無成品)。
- 若要將建置輸出儲存在 S3 儲存貯體中，請選擇 Amazon S3，然後執行下列動作：
 - 如果您想要使用專案名稱做為組建輸出 ZIP 檔案或資料夾名稱，則請將 Name (名稱) 保留空白。否則請輸入名稱。根據預設，成品名稱是專案名稱。如果想使用不同的名稱，請在成品名稱方塊中輸入名稱。如果要輸出 ZIP 檔案，請包含 zip 副檔名。
 - 針對 Bucket name (儲存貯體名稱)，選擇輸出儲存貯體的名稱。
 - 如果您在本程序稍早選擇 Insert build commands (插入組建命令)，然後針對 Output files (輸出檔案)，輸入要放入組建輸出 ZIP 檔案或資料夾之組建中的檔案位置。針對多個位置，以逗號區隔每個位置 (例如，appspec.yml, target/my-app.jar)。如需詳細資訊，請參閱[Buildspec 語法](#)中的 files 描述。

其他組態

展開 Additional configuration (其他組態)，並適當地設定選項。

10. 選擇 Create build project (建立建置專案)。在 Review (檢閱) 頁面上，選擇 Start build (開始建置) 來執行建置。

步驟 2：使用 Bitbucket Webhook 觸發組建

對於使用 Bitbucket Webhook 的專案，當 Bitbucket 儲存庫偵測到原始程式碼變更時，會 AWS CodeBuild 建立組建。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格上，選擇 Build projects (建置專案)，然後選擇與帶有 Webhook 的 Bitbucket 儲存庫相關聯的專案。如需建立 Bitbucket Webhook 專案的詳細資訊，請參閱 [the section called “步驟 1：使用 Bitbucket 建立建置專案並啟用 Webhook”](#)。
3. 在專案的 Bitbucket 儲存庫中對程式碼做些變更。
4. 在 Bitbucket 儲存庫上建立提取請求。如需詳細資訊，請參閱[提出提取請求](#)。
5. 在 Bitbucket Webhook 頁面中，選擇 View request (檢視請求) 來查看最近事件的清單。
6. 選擇檢視詳細資訊，以查看 CodeBuild 傳回之回應的詳細資訊。這看起來類似下述：

```
"response": "Webhook received and build started: https://us-east-1.console.aws.amazon.com/codebuild/home..."
"statusCode": 200
```

7. 導覽至 Bitbucket 提取請求頁面來查看組建狀態。

執行 CodeBuild 的 GitHub Enterprise Server 範例

AWS CodeBuild 支援 GitHub Enterprise Server 做為來源儲存庫。此範例示範如何在 GitHub Enterprise Server 儲存庫安裝憑證時設定 CodeBuild 專案。它還顯示如何啟用 Webhook，以便 CodeBuild 每次將程式碼變更推送到您的 GitHub Enterprise Server 儲存庫時重建原始程式碼。

主題

- [先決條件](#)
- [步驟 1：使用 GitHub Enterprise Server 建立建置專案並啟用 Webhook](#)

先決條件

1. 為您的 CodeBuild 專案產生個人存取字符。建議您建立 GitHub Enterprise 使用者，並產生此使用者的個人存取字符。將其複製到剪貼簿，以便在建立 CodeBuild 專案時使用。如需詳細資訊，請參閱 GitHub Help 網站上的 [Creating a personal access token for the command line](#)。

當您建立個人存取字符時，請在定義中包含 repo 範圍。

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

☒	repo	Full control of private repositories
☒	repo:status	Access commit status
☒	repo_deployment	Access deployment status
☒	public_repo	Access public repositories

2. 從 GitHub Enterprise Server 下載您的憑證。CodeBuild 使用憑證來建立與儲存庫的信任 SSL 連線。

Linux/macOS 用戶端：

從終端機視窗中，執行下列命令：

```
echo -n | openssl s_client -connect HOST:PORTNUMBER \  
| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /folder/filename.pem
```

將命令中的預留位置取代為下列值：

HOST。GitHub Enterprise Server 儲存庫的 IP 地址。

PORTNUMBER。您用來連線的連接埠號碼 (例如 443)。

folder。從中下載您憑證的資料夾。

filename。您憑證檔案的檔案名稱。

Important

將憑證儲存為 .pem 檔案。

Windows 用戶端：

使用瀏覽器從 GitHub Enterprise 下載您的憑證。若要查看網站的憑證詳細資訊，請選擇鎖定圖示。如需如何匯出憑證的資訊，請參閱您的瀏覽器文件。

 Important

將憑證儲存為 .pem 檔案。

3. 將憑證檔案上傳至 S3 儲存貯體。如需如何建立 S3 儲存貯體的資訊，請參閱[如何建立 S3 儲存貯體？](#) 如需如何將物件上傳至 S3 儲存貯體的資訊，請參閱[如何將檔案和資料夾上傳至儲存貯體？](#)

 Note

此儲存貯體必須與建置位於相同的 AWS 區域。例如，如果您指示 CodeBuild 在美國東部（俄亥俄）區域執行組建，則儲存貯體必須位於美國東部（俄亥俄）區域。

步驟 1：使用 GitHub Enterprise Server 建立建置專案並啟用 Webhook

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。
3. 在 Project name (專案名稱) 中，輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。
4. 在來源中，在來源提供者中選擇 GitHub Enterprise Server。
 - 選擇管理帳戶登入資料，然後選擇個人存取字符。針對服務，選擇 Secrets Manager（建議），然後設定您的秘密。然後在 GitHub Enterprise 個人存取字符中，輸入您的個人存取字符，然後選擇儲存。
 - 在 Repository URL (儲存庫 URL) 中，輸入儲存庫的路徑，包括儲存庫的名稱。
 - 展開 Additional configuration (其他組態)。
 - 選取 Rebuild every time a code change is pushed to this repository (在每次將程式碼變更推送至儲存庫時重建)，以便每次程式碼變更推送至此儲存庫時就重建。
 - 選取 Enable insecure SSL (啟用不安全 SSL)，在連線至 GitHub Enterprise Server 專案儲存庫時忽略 SSL 警告。

Note

建議只將 Enable insecure SSL (啟用不安全 SSL) 用於測試。不應用於生產環境。

SourceAdd source

Source 1 - Primary

Source provider

GitHub Enterprise ▼

Repository URL

https://<host-name>/<user-name>/<repository-name>

Disconnect GitHub Enterprise account

▼ Additional configuration

Git clone depth, Insecure SSL

Git clone depth - optional

1 ▼

Webhook - optional

☒ Rebuild every time a code change is pushed to this repository

Branch filter - optional

Enter a regular expression

Insecure SSL - optional

Enable this flag to ignore SSL warnings while connecting to project source.

☐ Enable insecure SSL

5. 在 Environment (環境) 中：

針對 Environment image (環境映像)，執行下列其中一項作業：

- 若要使用由 管理的 Docker 映像 AWS CodeBuild，請選擇受管映像，然後從作業系統、執行期（些）、映像和映像版本進行選擇。若可用，請從 Environment type (環境類型) 進行選擇。
- 若要使用另一個 Docker 映像，請選擇 Custom image (自訂映像)。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。如果您選擇其他登錄檔，請針對外部登錄檔 URL，使用 格式在 Docker Hub 中輸入 Docker 映像檔的名稱和標籤 *docker repository/docker image name*。如果您選擇 Amazon ECR，請使用 Amazon ECR 儲存庫和 Amazon ECR 映像來選擇 AWS 帳戶中的 Docker 映像。
- 若要使用私有 Docker 映像，請選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。針對 Image registry (映像登錄) 選擇 Other registry (其他登錄)，然後輸入私人 Docker 映像的憑證的 ARN。登入資料必須由 Secrets Manager 建立。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [What Is AWS Secrets Manager](#)。

6. 在 Service role (服務角色) 中，執行下列其中一項作業：

- 如果您沒有 CodeBuild 服務角色，請選擇新服務角色。在角色名稱中，輸入新角色的名稱。
- 如果您有 CodeBuild 服務角色，請選擇現有服務角色。在角色 ARN 中，選擇服務角色。

 Note

當您使用 主控台 建立或更新組建專案時，您可以同時建立 CodeBuild 服務角色。根據預設，此角色只能與該建置專案搭配運作。如果您使用主控台將此服務角色與另一個建置專案建立關聯，則會更新此角色以與其他建置專案搭配運作。服務角色最多可以與 10 個組建專案搭配運作。

7. 展開 Additional configuration (其他組態)。

如果您希望 CodeBuild 使用您的 VPC：

- 針對 VPC，選擇 CodeBuild 使用的 VPC ID。
- 對於 VPC 子網路，選擇包含 CodeBuild 使用之資源的子網路。
- 針對 VPC 安全群組，選擇 CodeBuild 用來允許存取 VPCs 中資源的安全群組。

如需詳細資訊，請參閱 [AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用](#)。

8. 在 Buildspec 中，執行下列其中一項操作：

- 選擇使用 buildspec 檔案以使用來源碼根目錄中的 buildspec.yml 檔案。

- 選擇插入建置命令以使用主控台插入建置命令。

如需更多資訊，請參閱[Buildspec 參考](#)。

9. 在 Artifacts (成品) 中，針對 Type (類型)，執行下列其中一項操作：

- 如果您不要建立建置輸出成品，則請選擇 No artifacts (無成品)。
- 若要將建置輸出存放在 S3 儲存貯體，請選擇 Amazon S3，然後執行下列動作：
 - 如果您想要使用專案名稱做為組建輸出 ZIP 檔案或資料夾名稱，則請將 Name (名稱) 保留空白。否則請輸入名稱。根據預設，成品名稱是專案名稱。如果想使用不同的名稱，請在成品名稱方塊中輸入名稱。如果要輸出 ZIP 檔案，請包含 zip 副檔名。
 - 針對 Bucket name (儲存貯體名稱)，選擇輸出儲存貯體的名稱。
 - 如果您在本程序稍早選擇 Insert build commands (插入組建命令)，然後針對 Output files (輸出檔案)，輸入要放入組建輸出 ZIP 檔案或資料夾之組建中的檔案位置。針對多個位置，以逗號區隔每個位置 (例如，appspec.yml, target/my-app.jar)。如需詳細資訊，請參閱[Buildspec 語法](#)中的 files 描述。

10. 針對 Cache type (快取類型)，選擇以下其中一項：

- 如果您不想要使用快取，請選擇 No cache (無快取)。
- 如果您想要使用 Amazon S3 快取，請選擇 Amazon S3，然後執行下列動作：
 - 針對 Bucket (儲存貯體)，選擇存放快取的 S3 儲存貯體名稱。
 - (選用) 針對快取路徑字首，輸入 Amazon S3 路徑字首。Cache path prefix (快取路徑字首) 值類似目錄名稱。它可讓您將快取存放至儲存貯體的相同目錄下方。

 Important

請不要在路徑字首結尾附加尾端斜線 (/)。

- 如果您想要使用本機快取，請選擇 Local (本機)，然後選擇一或多個本機快取模式。

 Note

「Docker layer cache」(Docker 層快取) 模式僅適用於 Linux。如果您選擇此模式，您的專案必須以特殊權限模式執行。

使用快取可節省大量建置時間，因為建置環境的可重複使用部分存放在快取中，並用於各建置。如需在 buildspec 檔案中指定快取的詳細資訊，請參閱[Buildspec 語法](#)。如需快取的詳細資訊，請參閱「[快取建置以改善效能](#)」。

11. 選擇 Create build project (建立建置專案)。在組建專案頁面上，選擇 Start build (啟動組建)。

執行 CodeBuild 的 GitHub 提取請求和 Webhook 篩選條件範例

AWS CodeBuild 當來源儲存庫為 GitHub 時，支援 Webhook。這表示對於在 GitHub 儲存庫中存放其原始碼的 CodeBuild 組建專案，Webhook 可用來在每次將程式碼變更推送至儲存庫時重建原始碼。如需 CodeBuild 範例，請參閱[AWS CodeBuild 範例](#)。

Note

使用 Webhook 時，使用者可能會觸發非預期的建置。若要降低此風險，請參閱 [使用 Webhook 的最佳實務](#)。

主題

- [步驟 1：使用 GitHub 建立建置專案並啟用 Webhook](#)
- [步驟 2：確認已啟用 Webhook](#)

步驟 1：使用 GitHub 建立建置專案並啟用 Webhook

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。
3. 選擇 Create build project (建立建置專案)。
4. 在 Project configuration (專案組態) 中：

Project name (專案名稱)

輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。

5. 在 Source (來源) 中：

來源提供者

選擇 GitHub。請遵循說明來與 GitHub 連線 (或重新連線)，然後選擇 Authorize (授權)。

儲存庫

選擇 Repository in my GitHub account (GitHub 帳戶中的儲存庫)。

GitHub 儲存庫

輸入 GitHub 儲存庫的 URL。

6. 在主要來源 Webhook 事件中，選取下列項目。

Note

只有在您在上一個步驟的 GitHub 帳戶中選擇儲存庫時，才會顯示主要來源 Webhook 事件區段。

1. 當您建立專案時，請選取 Rebuild every time a code change is pushed to this repository (在每次將程式碼變更推送至此儲存庫時重建)。
2. 從 Event type (事件類型)，選擇一或多個事件。
3. 若要篩選事件觸發組建的時間，請在 Start a build under these conditions (在這些情況下開始組建) 下新增一或多個選用的篩選條件。
4. 若要篩選何時不觸發事件，請在 Don't start a build under these conditions (在這些情況下不開始組建) 下新增一或多個選用的篩選條件。
5. 如有需要，請選擇新增篩選條件群組以新增另一個篩選條件群組。

如需 GitHub Webhook 事件類型和篩選條件的詳細資訊，請參閱 [GitHub Webhook 事件](#)。

7. 在 Environment (環境) 中：

環境映像

選擇下列其中一項：

若要使用由管理的 Docker 映像 AWS CodeBuild：

選擇受管映像，然後從作業系統、執行時間、映像和映像版本進行選擇。若可用，請從 Environment type (環境類型) 進行選擇。

若要使用另一個 Docker 映像：

選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。如果您選擇其他登錄檔，請針對外部登錄檔 URL，使用 格式，在 Docker Hub 中輸入 Docker 映像檔的名稱和標籤 *docker repository/docker image name*。如果您選擇 Amazon ECR，請使用 Amazon ECR 儲存庫和 Amazon ECR 映像來選擇 AWS 帳戶中的 Docker 映像。

若要使用私有 Docker 映像：

選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。針對 Image registry (映像登錄) 選擇 Other registry (其他登錄)，然後輸入私人 Docker 映像的憑證的 ARN。登入資料必須由 Secrets Manager 建立。如需詳細資訊，請參閱 [什麼是 AWS Secrets Manager ?](#) AWS Secrets Manager 《使用者指南》中的。

服務角色

選擇下列其中一項：

- 如果您沒有 CodeBuild 服務角色，請選擇新服務角色。在角色名稱中，輸入新角色的名稱。
- 如果您有 CodeBuild 服務角色，請選擇現有服務角色。在角色 ARN 中，選擇服務角色。

Note

當您使用 主控台 建立或更新組建專案時，您可以同時建立 CodeBuild 服務角色。根據預設，此角色只能與該建置專案搭配運作。如果您使用主控台將此服務角色與另一個建置專案建立關聯，則會更新此角色以與其他建置專案搭配運作。服務角色最多可以與 10 個組建專案搭配運作。

8. 在 Buildspec 中，執行下列其中一項操作：

- 選擇使用 buildspec 檔案以使用來源碼根目錄中的 buildspec.yml 檔案。
- 選擇插入建置命令以使用主控台插入建置命令。

如需更多資訊，請參閱 [Buildspec 參考](#)。

9. 在 Artifacts (成品) 中：

類型

選擇下列其中一項：

- 如果您不要建立建置輸出成品，則請選擇 No artifacts (無成品)。
- 若要將建置輸出存放在 S3 儲存貯體，請選擇 Amazon S3，然後執行下列動作：
 - 如果您想要使用專案名稱做為組建輸出 ZIP 檔案或資料夾名稱，則請將 Name (名稱) 保留空白。否則請輸入名稱。根據預設，成品名稱是專案名稱。如果想使用不同的名稱，請在成品名稱方塊中輸入名稱。如果要輸出 ZIP 檔案，請包含 zip 副檔名。
 - 針對 Bucket name (儲存貯體名稱)，選擇輸出儲存貯體的名稱。
 - 如果您在本程序稍早選擇 Insert build commands (插入組建命令)，然後針對 Output files (輸出檔案)，輸入要放入組建輸出 ZIP 檔案或資料夾之組建中的檔案位置。針對多個位置，以逗號區隔每個位置 (例如，appspec.yml, target/my-app.jar)。如需詳細資訊，請參閱[Buildspec 語法](#)中的 files 描述。

其他組態

展開 Additional configuration (其他組態)，並適當地設定選項。

10. 選擇 Create build project (建立建置專案)。在 Review (檢閱) 頁面上，選擇 Start build (開始建置) 來執行建置。

步驟 2：確認已啟用 Webhook

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 執行以下任意一項：
 - 選擇具有您想要驗證之 Webhook 建置專案的連結，然後選擇 Build details (建置詳細資訊)。
 - 選擇建置專案旁的按鈕，其中包含您要驗證的 Webhook，選擇檢視詳細資訊，然後選擇建置詳細資訊索引標籤。
4. 在主要來源 Webhook 事件中，選擇 Webhook URL 連結。
5. 在您的 GitHub 儲存庫中，在 Settings (設定) 頁面的 Webhooks 下方，確認 Pull Requests (提取請求) 和 Pushes (推送) 已選取。
6. 在您的 GitHub 設定檔設定中，在個人設定、應用程式、授權 OAuth 應用程式下，您應該會看到您的應用程式已獲授權存取您選取的 AWS 區域。

教學課程：在 CodeBuild 中使用 Fastlane 簽署 Apple 程式碼，並使用 S3 進行憑證儲存

[fastlane](#) 是熱門的開放原始碼自動化工具，可自動化 iOS 和 Android 應用程式的 Beta 版部署和版本。它會處理所有繁瑣的任務，例如產生螢幕擷取畫面、處理程式碼簽署，以及釋出您的應用程式。

先決條件

若要完成本教學課程，您必須先設定下列項目：

- 一個 AWS 帳戶
- [Apple 開發人員帳戶](#)
- 儲存憑證的 S3 儲存貯體
- 專案中已安裝的 fastlane - 安裝 fastlane [的指南](#)

步驟 1：在本機電腦上設定與 S3 的 Fastlane Match

[Fastlane Match](#) 是 [Fastlane 工具](#) 之一，允許在您的本機開發環境和 CodeBuild 上進程式碼簽署的無縫組態。Fastlane Match 會將所有程式碼簽署憑證和佈建設定檔存放在 Git 儲存庫/S3 儲存貯體/Google 雲端儲存體中，並在需要時下載和安裝必要的憑證和設定檔。

在此範例組態中，您將設定並使用 Amazon S3 儲存貯體進行儲存。

1. 初始化專案中的相符項目：

```
fastlane match init
```

2. 出現提示時，請選擇 S3 作為儲存模式。
3. 更新您的 `Matchfile` 以使用 S3：

```
storage_mode("s3")
s3_bucket("your-s3-bucket-name")
s3_region("your-aws-region")
type("appstore") # The default type, can be: appstore, adhoc, enterprise or development
```

步驟 2：設定您的 Fastfile

使用下列通道建立或更新 `Fastfile`。

在 CodeBuild 上，每次建置和簽署應用程式時，都需要執行 Fastlane Match。最簡單的方法是將 match 動作新增至建置您應用程式的通道。

```
default_platform(:ios)

platform :ios do
  before_all do
    setup_ci
  end

  desc "Build and sign the app"
  lane :build do
    match(type: "appstore", readonly: true)
    gym(
      scheme: "YourScheme",
      export_method: "app-store"
    )
  end
end
```

Note

請務必 setup_ci 將新增至 中的 before_all 區段，Fastfile 配對動作才能正常運作。這可確保使用具有適當許可的暫時 Fastlane 金鑰鏈。如果不使用此功能，您可能會看到建置失敗或結果不一致。

步驟 3：執行 **fastlane match** 命令以產生個別的憑證和設定檔

指定類型（即開發、應用程式存放區、臨機操作、企業）的 fastlane 比對命令會在遠端存放區中無法使用時產生憑證和設定檔。憑證和設定檔將由 fastlane 存放在 S3 中。

```
bundle exec fastlane match appstore
```

命令執行將為互動式，且 fastlane 會要求設定用於解密憑證的複雜密碼。

步驟 4：為您的專案建立應用程式檔案

為您的專案建立或新增適當的應用程式檔案。

1. 根據您的專案建置需求建立或新增 [Gymfile](#)、[Appfile](#)、[Snapfile](#)、[Deliverfile](#)。
2. 遞交遠端儲存庫的變更

步驟 5：在 Secrets Manager 中建立環境變數

建立兩個秘密來存放 fastlane 工作階段 Cookie 和相符的複雜密碼。如需在 Secrets Manager 中建立秘密的詳細資訊，請參閱[建立 AWS Secrets Manager 秘密](#)。

1. 存取您的 Fastlane 工作階段 Cookie，如下所示。
 - a. 私密金鑰 - FASTLANE_SESSION
 - b. 秘密值 - 在本機電腦上執行下列命令所產生的工作階段 Cookie。

Note

在本機檔案中進行身分驗證後，即可使用此值：`~/.fastlane/spaceship/my_appleid_username/cookie`。

```
fastlane spaceauth -u <apple account>
```

2. Fastlane Match 密碼短語 - 若要讓 Fastlane Match 解密存放在 S3 儲存貯體中的憑證和設定檔，必須將您在比對設定步驟中設定的加密密碼短語新增至 CodeBuild 專案的環境變數。
 - a. 私密金鑰 - MATCH_PASSWORD
 - b. 秘密值 - `<#####>`。在步驟 3 中產生憑證時會設定密碼短語。

Note

在 Secrets Manager 中建立上述秘密時，請記得提供具有下列字首的秘密名稱：`/CodeBuild/`

步驟 6：建立運算機群

為您的專案建立運算機群。

1. 在主控台中，前往 CodeBuild 並建立新的運算機群。
2. 選擇「macOS」做為作業系統，然後選取適當的運算類型和映像。

步驟 7：在 CodeBuild 中建立專案

在 CodeBuild 中建立您的專案。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
3. 設定您的來源提供者（例如 GitHub、CodeCommit）。這是 iOS 專案來源儲存庫，而不是憑證儲存庫。
4. 在 Environment (環境) 中：
 - 選擇預留容量。
 - 針對機群，選取上面建立的機群。
 - 提供 CodeBuild 將為您建立的服務角色名稱。
 - 提供下列環境變數。
 - 名稱：MATCH_PASSWORD、值：*<secrets arn>*、類型：Secrets Manager（在步驟 5 中為 MATCH_PASSWORD 建立的秘密 ARN）
 - 名稱：FASTLANE_SESSION、值：*<secrets arn>*、類型：Secrets Manager（在步驟 5 中為 FASTLANE_SESSION 建立的秘密 ARN）
5. 在 Buildspec 中，新增下列項目：

```
version: 0.2

phases:
  install:
    commands:
      - gem install bundler
      - bundle install
```

```
build:
  commands:
    - echo "Building and signing the app..."
    - bundle exec fastlane build
  post_build:
    commands:
      - echo "Build completed on date"

artifacts:
  files:
    - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

步驟 8：設定 IAM 角色

建立專案後，請確定 CodeBuild 專案的服務角色具有存取包含憑證的 S3 儲存貯體的許可。將下列政策新增至角色：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::your-s3-bucket-name"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::your-s3-bucket-name/*"
    }
  ]
}
```

```
}
```

步驟 9：執行建置

執行組建。您可以在 CodeBuild 中檢閱建置狀態和日誌。

任務完成後，您將能夠檢視任務的日誌。

疑難排解

- 如果您在擷取憑證時遇到問題，請確定您的 IAM 許可已正確設定為 S3 存取。
- 如果您在憑證解密時遇到問題，請確定您在 MATCH_PASSWORD 環境變數中設定正確的密碼短語。
- 對於程式碼簽署問題，請確認您的 Apple 開發人員帳戶具有必要的憑證和設定檔，而且 Xcode 專案中的套件識別符符合佈建設定檔中的項目。

安全考量

以下是本教學課程的安全考量。

- 確保您的 S3 儲存貯體具有適當的安全設定，包括靜態加密。特別是，請確定儲存貯體沒有公有存取權，並限制只能存取 CodeBuild 和需要存取的系統。
- 考慮使用 AWS Secrets Manager 來存放敏感資訊，例如 MATCH_PASSWORD 和 FASTLANE_SESSION。

此範例提供使用 Amazon S3 進行憑證儲存的 CodeBuild 中 Fastlane 的 iOS 程式碼簽署設定。您可能需要根據您的特定專案需求和 CodeBuild 環境來調整一些步驟。這種方法利用 AWS 服務在 AWS 生態系統中增強安全性和整合。

教學課程：使用 GitHub 在 CodeBuild 中使用 Fastlane 簽署 Apple 程式碼以進行憑證儲存

[fastlane](#) 是熱門的開放原始碼自動化工具，可自動化 iOS 和 Android 應用程式的 Beta 版部署和版本。它會處理所有繁瑣的任務，例如產生螢幕擷取畫面、處理程式碼簽署，以及釋出您的應用程式。

此範例示範如何在 Mac 機群上執行的 CodeBuild 專案中使用 Fastlane 設定 Apple 程式碼簽署，並將 GitHub 做為憑證和佈建設定檔的儲存體。

先決條件

若要完成本教學課程，您必須先設定下列項目：

- 一個 AWS 帳戶
- [Apple 開發人員帳戶](#)
- 用於儲存憑證的私有 GitHub 儲存庫
- 專案中安裝的 fastlane - 安裝 fastlane [的指南](#)

步驟 1：在本機電腦上設定與 GitHub 的 Fastlane Match

[Fastlane Match](#) 是 [Fastlane 工具](#)之一，允許在您的本機開發環境和 CodeBuild 上進程式碼簽署的無縫組態。Fastlane Match 會將所有程式碼簽署憑證和佈建設定檔儲存在 Git 儲存庫/S3 儲存貯體/Google 雲端儲存體中，並在需要時下載和安裝必要的憑證和設定檔。

在此範例組態中，我們將設定和使用 Git 儲存庫進行儲存。

1. 初始化專案中的相符項目：

```
fastlane match init
```

2. 出現提示時，請選擇 GitHub 做為儲存模式。
3. 更新您的 `Matchfile` 以使用 GitHub：

```
git_url("https://github.com/your-username/your-certificate-repo.git")
storage_mode("git")
type("development") # The default type, can be: appstore, adhoc, enterprise or
development
```

Note

請務必輸入 Git 儲存庫的 HTTPS URL，以便 Fastlane 成功驗證和複製。否則，當您嘗試使用配對時，可能會看到身分驗證錯誤。

步驟 2：設定您的 Fastfile

使用下列通道建立或更新您的 `Fastfile`。

在 CodeBuild 上，每次建置和簽署應用程式時，都需要執行 Fastlane Match。最簡單的方法是將 match 動作新增至建置您應用程式的通道。

```
default_platform(:ios)

platform :ios do
  before_all do
    setup_ci
  end

  desc "Build and sign the app"
  lane :build do
    match(type: "appstore", readonly: true)
    gym(
      scheme: "YourScheme",
      export_method: "app-store"
    )
  end
end
```

Note

請務必 setup_ci 將新增至 中的 before_all 區段，Fastfile 配對動作才能正常運作。這可確保使用具有適當許可的暫時 Fastlane 金鑰鏈。如果不使用此功能，您可能會看到建置失敗或結果不一致。

步驟 3：執行 **fastlane match** 命令以產生個別的憑證和設定檔

指定類型（即開發、應用程式存放區、隨機操作、企業）的 fastlane 比對命令會在遠端存放區中無法使用時產生憑證和設定檔。憑證和設定檔將由 fastlane 存放在 GitHub 中。

```
bundle exec fastlane match appstore
```

命令執行將為互動式，且 fastlane 會要求設定密碼短語以解密憑證。

步驟 4：為您的專案建立應用程式檔案

為您的專案建立或新增適當的應用程式檔案。

1. 根據您的專案建置需求建立或新增 [Gymfile](#)、[Appfile](#)、[Snapfile](#)、[Deliverfile](#)。
2. 將變更遞交至遠端儲存庫。

步驟 5：在 Secrets Manager 中建立環境變數

建立三個秘密來存放 fastlane 工作階段 Cookie 和相符的複雜密碼。如需在 Secrets Manager 中建立秘密的詳細資訊，請參閱[建立 AWS Secrets Manager 秘密](#)。

1. 存取您的 Fastlane 工作階段 Cookie，如下所示。
 - a. 私密金鑰 - FASTLANE_SESSION
 - b. 秘密值 - 在本機電腦上執行下列命令所產生的工作階段 Cookie。

Note

在本機檔案中進行身分驗證後，即可使用此值：`~/.fastlane/spaceship/my_appleid_username/cookie`。

```
fastlane spaceauth -u <Apple_account>
```

2. Fastlane Match 密碼短語 - 若要讓 Fastlane Match 解密儲存在 Git 儲存庫中的憑證和設定檔，必須將您在比對設定步驟中設定的加密密碼短語新增至 CodeBuild 專案的環境變數。
 - a. 私密金鑰 - MATCH_PASSWORD
 - b. 秘密值 - `<match passphrase to decrypt certificates>`。在步驟 3 中產生憑證時會設定密碼短語。
3. Fastlane MATCH_GIT_BASIC_AUTHORIZATION - 設定比對的基本授權：
 - a. 私密金鑰：

MATCH_GIT_BASIC_AUTHORIZATION

- b. 秘密值 - 值應該是使用者名稱和個人存取字符 (PAT) 的 base64 編碼字串，格式為 `username:password`。您可以使用下列命令產生它：

```
echo -n your_github_username:your_personal_access_token | base64
```

您可以在設定檔 > 設定 > 開發人員設定 > 個人存取字符的 GitHub 主控台上產生 PAT。如需詳細資訊，請參閱下列指南：<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens>。

Note

在 Secrets Manager 中建立上述秘密時，請記得提供具有下列字首的秘密名稱： `/CodeBuild/`

步驟 6：建立運算機群

為您的專案建立運算機群。

1. 在 主控台中，前往 CodeBuild 並建立新的運算機群。
2. 選擇 macOS 做為作業系統，然後選取適當的運算類型和映像。

步驟 7：在 CodeBuild 中建立專案

在 CodeBuild 中建立您的專案。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
3. 設定您的來源提供者（例如 GitHub、CodeCommit）。這是 iOS 專案來源儲存庫，而不是憑證儲存庫。
4. 在 Environment (環境) 中：
 - 選擇預留容量。
 - 針對機群，選取上面建立的機群。

- 提供 CodeBuild 將為您建立的服務角色名稱。
- 提供下列環境變數。
 - 名稱：MATCH_PASSWORD、值：*<secrets arn>*、類型：Secrets Manager (在步驟 5 中為 MATCH_PASSWORD 建立的秘密 ARN)
 - 名稱：FASTLANE_SESSION、值：*<secrets arn>*、類型：Secrets Manager (在步驟 5 中為 FASTLANE_SESSION 建立的秘密 ARN)
 - 名稱：MATCH_GIT_BASIC_AUTHORIZATION、值：*<secrets ARN>*、類型：Secrets Manager Secrets ARN (在步驟 5 為 建立MATCH_GIT_BASIC_AUTHORIZATION)

5. 在 Buildspec 中，新增下列項目：

```
version: 0.2

phases:
  install:
    commands:
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo "Building and signing the app..."
      - bundle exec fastlane build
  post_build:
    commands:
      - echo "Build completed on date"

artifacts:
  files:
    - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

步驟 8：執行建置

執行組建。您可以在 CodeBuild 中檢閱建置狀態和日誌。

任務完成後，您就可以檢視任務的日誌。

疑難排解

- 如果您在存取 GitHub 儲存庫時遇到問題，請仔細檢查您的個人存取字符和 `MATCH_GIT_BASIC_AUTHORIZATION` 環境變數。
- 如果您在憑證解密時遇到問題，請確定您在 `MATCH_PASSWORD` 環境變數中設定正確的密碼短語。
- 對於程式碼簽署問題，請確認您的 Apple 開發人員帳戶具有必要的憑證和設定檔，而且 Xcode 專案中的套件識別符符合佈建設定檔中的項目。

安全考量

以下是本教學課程的安全性考量事項。

- 將憑證的 GitHub 儲存庫保持私有，並定期稽核存取權。
- 考慮使用 AWS Secrets Manager 來儲存敏感資訊，例如 `MATCH_PASSWORD` 和 `FASTLANE_SESSION`。

此範例提供使用 GitHub 進行憑證儲存的 CodeBuild 中 Fastlane 的 iOS 程式碼簽署設定。您可能需要根據您的特定專案需求和 CodeBuild 環境來調整一些步驟。這種方法利用 AWS 服務來增強 AWS 生態系統內的安全和整合。

使用語意版本控制在建置時間設定成品名稱

此範例包含 `buildspec` 範例檔案，示範如何指定建置時所建立的成品名稱。`buildspec` 檔案中指定的名稱可以包含 Shell 命令和環境變數，而變成唯一的名稱。您在 `buildspec` 檔案中指定的名稱，將會覆寫您建立專案時在主控台輸入的名稱。

如果您建立多次，則使用 `buildspec` 檔案中指定的成品名稱可確保輸出成品檔名是唯一的。例如，您可以使用建置時插入成品名稱中的日期和時間戳記。

如果您要以 `buildspec` 檔案中的名稱覆寫您在主控台輸入的成品名稱，請執行下列操作：

1. 將組建專案設定成以 `buildspec` 檔案中的名稱來覆寫成品名稱。
 - 如果使用主控台來建立組建專案，請選取 `Enable semantic versioning` (啟用語意版本控制)。如需詳細資訊，請參閱[建立組建專案 \(主控台\)](#)。
 - 如果您使用 AWS CLI，請在傳遞給的 JSON 格式檔案中將 `overrideArtifactName` 設定為 `truecreate-project`。如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。

- 如果您使用 AWS CodeBuild API，請在建立或更新專案或啟動組建時，在ProjectArtifacts物件上設定 overrideArtifactName旗標。

2. 在 buildspec 檔案中指定名稱。將下列範例 buildspec 檔案當作指南。

此 Linux 範例示範如何指定包含組建建立日期的成品名稱：

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

此 Linux 範例說明如何指定使用 CodeBuild 環境變數的成品名稱。如需詳細資訊，請參閱[建置環境中的環境變數](#)。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

此 Windows 範例示範如何指定包含組建建立日期和時間的成品名稱：

```
version: 0.2
env:
  variables:
    TEST_ENV_VARIABLE: myArtifactName
phases:
  build:
    commands:
      - cd samples/helloworld
      - dotnet restore
```

```
    - dotnet run
artifacts:
  files:
    - '**/*'
  name: $Env:TEST_ENV_VARIABLE-$(Get-Date -UFormat "%Y%m%d-%H%M%S")
```

此 Windows 範例說明如何指定成品名稱，使用 buildspec 檔案中宣告的變數和 CodeBuild 環境變數。如需詳細資訊，請參閱[建置環境中的環境變數](#)。

```
version: 0.2
env:
  variables:
    TEST_ENV_VARIABLE: myArtifactName
phases:
  build:
    commands:
      - cd samples/helloworld
      - dotnet restore
      - dotnet run
artifacts:
  files:
    - '**/*'
  name: $Env:TEST_ENV_VARIABLE-$Env:AWS_REGION
```

如需詳細資訊，請參閱[CodeBuild 的建置規格參考](#)。

執行 CodeBuild 的 Microsoft Windows 範例

這些範例使用執行 Microsoft Windows Server 2019、.NET Framework 和 .NET Core SDK 的 AWS CodeBuild 建置環境，從以 F# 和 Visual Basic 撰寫的程式碼建置執行期檔案。

Important

執行這些範例可能會導致 AWS 您的帳戶產生費用。這包括 CodeBuild 和與 Amazon S3 相關的 AWS 資源和動作 AWS KMS，以及 CloudWatch Logs 的可能費用。如需詳細資訊，請參閱 [CodeBuild 定價](#)、[Amazon S3 定價](#)、[AWS Key Management Service 定價](#) 和 [Amazon CloudWatch 定價](#)。

執行 Windows 範例

使用下列程序來執行 Windows 範例。

執行 Windows 範例

1. 如本主題的 [目錄結構](#) 和 [檔案](#) 章節所述建立檔案，然後將其上傳至 S3 輸入儲存貯體或 CodeCommit 或 GitHub 儲存庫。

Important

請勿上傳 (*root directory name*)，僅上傳 (*root directory name*) 內的檔案即可。

如果您使用的是 S3 輸入儲存貯體，請務必建立包含這些檔案的 ZIP 檔案，然後將其上傳至輸入儲存貯體。請勿將 (*root directory name*) 新增至 ZIP 檔案，僅新增 (*root directory name*) 內的檔案即可。

2. 建立建置專案。建置專案必須使用 `mcr.microsoft.com/dotnet/framework/sdk:4.8` 映像來建置 .NET Framework 專案。

如果您使用 AWS CLI 來建立建置專案，`create-project` 命令的 JSON 格式輸入可能會與此類似。(以您自己的值取代預留位置。)

```
{
  "name": "sample-windows-build-project",
```



```

"source": {
  "type": "S3",
  "location": "codebuild-region-ID-account-ID-input-bucket/windows-build-input-artifact.zip"
},
"artifacts": {
  "type": "S3",
  "location": "codebuild-region-ID-account-ID-output-bucket",
  "packaging": "ZIP",
  "name": "windows-build-output-artifact.zip"
},
"environment": {
  "type": "WINDOWS_SERVER_2019_CONTAINER",
  "image": "mcr.microsoft.com/dotnet/framework/sdk:4.8",
  "computeType": "BUILD_GENERAL1_MEDIUM"
},
"serviceRole": "arn:aws:iam::account-ID:role/role-name",
"encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}

```

3. 執行組建，並遵循 [中的步驟手動執行組建](#)。
4. 若要取得建置輸出成品，請在您的 S3 輸出儲存貯體中，將 *windows-build-output-artifact.zip* 檔案下載至您的本機電腦或執行個體。擷取內容以取得執行時間和其他檔案。
 - 使用 .NET Framework 的 F# 範例執行期檔案 FSharpHelloWorld.exe 可在 FSharpHelloWorld\bin\Debug 目錄中找到。
 - 使用 .NET Framework 的 Visual Basic 範例執行期檔案 VBHelloWorld.exe 可在 VBHelloWorld\bin\Debug 目錄中找到。

目錄結構

這些範例假設下列目錄結構。

F# 和 .NET Framework

```

(root directory name)
### buildspec.yml
### FSharpHelloWorld.sln
### FSharpHelloWorld
### App.config
### AssemblyInfo.fs

```

```
### FSharpHelloWorld.fsproj
### Program.fs
```

Visual Basic 和 .NET Framework

```
(root directory name)
### buildspec.yml
### VBHelloWorld.sln
### VBHelloWorld
  ### App.config
  ### HelloWorld.vb
  ### VBHelloWorld.vbproj
  ### My Project
    ### Application.Designer.vb
    ### Application.myapp
    ### AssemblyInfo.vb
    ### Resources.Designer.vb
    ### Resources.resx
    ### Settings.Designer.vb
    ### Settings.settings
```

檔案

這些範例使用下列檔案。

F# 和 .NET Framework

buildspec.yml (在 *(root directory name)* 中)：

```
version: 0.2

env:
  variables:
    SOLUTION: .\FSharpHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
```

```

- '& nuget restore $env:SOLUTION -PackagesDirectory $env:PACKAGE_DIRECTORY'
- '& msbuild -p:FrameworkPathOverride="C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
artifacts:
  files:
    - .\FSharpHelloWorld\bin\Debug\*
```

FSharpHelloWorld.sln (在 *(root directory name)* 中) :

```

Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F2A71F9B-5D33-465A-A702-920D77279786}") = "FSharpHelloWorld",
  "FSharpHelloWorld\FSharpHelloWorld.fsproj", "{D60939B6-526D-43F4-9A89-577B2980DF62}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.ActiveCfg = Release|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.Build.0 = Release|Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
EndGlobal
```

App.config (在 *(root directory name)*\FSharpHelloWorld 中) :

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

AssemblyInfo.fs (在 *(root directory name)*\FSharpHelloWorld 中) :

```
namespace FSharpHelloWorld.AssemblyInfo

open System.Reflection
open System.Runtime.CompilerServices
open System.Runtime.InteropServices

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[<assembly: AssemblyTitle("FSharpHelloWorld")>]
[<assembly: AssemblyDescription("")>]
[<assembly: AssemblyConfiguration("")>]
[<assembly: AssemblyCompany("")>]
[<assembly: AssemblyProduct("FSharpHelloWorld")>]
[<assembly: AssemblyCopyright("Copyright © 2017")>]
[<assembly: AssemblyTrademark("")>]
[<assembly: AssemblyCulture("")>]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[<assembly: ComVisible(false)>]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[<assembly: Guid("d60939b6-526d-43f4-9a89-577b2980df62")>]

// Version information for an assembly consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [<assembly: AssemblyVersion("1.0.*")>]
[<assembly: AssemblyVersion("1.0.0.0")>]
[<assembly: AssemblyFileVersion("1.0.0.0")>]

do
    ()
```

FSharpHelloWorld.fsproj (在 *(root directory name)*\FSharpHelloWorld 中) :

```

<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props"
  Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>d60939b6-526d-43f4-9a89-577b2980df62</ProjectGuid>
    <OutputType>Exe</OutputType>
    <RootNamespace>FSharpHelloWorld</RootNamespace>
    <AssemblyName>FSharpHelloWorld</AssemblyName>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
    <TargetFSharpCoreVersion>4.4.0.0</TargetFSharpCoreVersion>
    <Name>FSharpHelloWorld</Name>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <Tailcalls>false</Tailcalls>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <WarningLevel>3</WarningLevel>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DocumentationFile>bin\Debug\FSharpHelloWorld.XML</DocumentationFile>
    <Prefer32Bit>true</Prefer32Bit>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <Tailcalls>true</Tailcalls>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <WarningLevel>3</WarningLevel>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DocumentationFile>bin\Release\FSharpHelloWorld.XML</DocumentationFile>
    <Prefer32Bit>true</Prefer32Bit>
  </PropertyGroup>

```

```

<ItemGroup>
  <Reference Include="mscorlib" />
  <Reference Include="FSharp.Core, Version=$(TargetFSharpCoreVersion),
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a">
    <Private>True</Private>
  </Reference>
  <Reference Include="System" />
  <Reference Include="System.Core" />
  <Reference Include="System.Numerics" />
</ItemGroup>
<ItemGroup>
  <Compile Include="AssemblyInfo.fs" />
  <Compile Include="Program.fs" />
  <None Include="App.config" />
</ItemGroup>
<PropertyGroup>
  <MinimumVisualStudioVersion Condition="'$(MinimumVisualStudioVersion)' == ''">11</
MinimumVisualStudioVersion>
</PropertyGroup>
<Choose>
  <When Condition="'$(VisualStudioVersion)' == '11.0'">
    <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)\..\Microsoft SDKs\F#
\3.0\Framework\v4.0\Microsoft.FSharp.Targets')">
      <FSharpTargetsPath>$(MSBuildExtensionsPath32)\..\Microsoft SDKs\F#
\3.0\Framework\v4.0\Microsoft.FSharp.Targets</FSharpTargetsPath>
    </PropertyGroup>
  </When>
  <Otherwise>
    <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)\Microsoft
\VisualStudio\v$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets')">
      <FSharpTargetsPath>$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v
$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets</FSharpTargetsPath>
    </PropertyGroup>
  </Otherwise>
</Choose>
<Import Project="$(FSharpTargetsPath)" />
<!-- To modify your build process, add your task inside one of the targets below and
uncomment it.
    Other similar extension points exist, see Microsoft.Common.targets.
<Target Name="BeforeBuild">
</Target>
<Target Name="AfterBuild">
</Target>
-->

```

```
</Project>
```

Program.fs (在 *(root directory name)*\FSharpHelloWorld 中) :

```
// Learn more about F# at http://fsharp.org
// See the 'F# Tutorial' project for more help.

[<EntryPoint>]
let main argv =
    printfn "Hello World"
    0 // return an integer exit code
```

Visual Basic 和 .NET Framework

buildspec.yml (在 *(root directory name)* 中) :

```
version: 0.2

env:
  variables:
    SOLUTION: .\VBHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
      - '& "C:\ProgramData\chocolatey\bin\NuGet.exe" restore $env:SOLUTION -
        PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& "C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" -
        p:FrameworkPathOverride="C:\Program Files (x86)\Reference Assemblies\Microsoft
        \Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
    artifacts:
      files:
        - .\VBHelloWorld\bin\Debug\*
```

VBHelloWorld.sln (在 *(root directory name)* 中) :

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
```

```

MinimumVisualStudioVersion = 10.0.40219.1
Project("{F184B08F-C81C-45F6-A57F-5ABD9991F28F}") = "VBHelloWorld", "VBHelloWorld\VBHelloWorld.vbproj", "{4DCEC446-7156-4FE6-8CCC-219E34DD409D}"
EndProject
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Any CPU = Debug|Any CPU
        Release|Any CPU = Release|Any CPU
    EndGlobalSection
    GlobalSection(ProjectConfigurationPlatforms) = postSolution
        {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
        {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.Build.0 = Debug|Any CPU
        {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.ActiveCfg = Release|Any CPU
        {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.Build.0 = Release|Any CPU
    EndGlobalSection
    GlobalSection(SolutionProperties) = preSolution
        HideSolutionNode = FALSE
    EndGlobalSection
EndGlobal

```

App.config (在 *(root directory name)*\VBHelloWorld 中) :

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
    </startup>
</configuration>

```

HelloWorld.vb (在 *(root directory name)*\VBHelloWorld 中) :

```

Module HelloWorld

    Sub Main()
        MsgBox("Hello World")
    End Sub

End Module

```

VBHelloWorld.vbproj (在 *(root directory name)*\VBHelloWorld 中) :

```

<?xml version="1.0" encoding="utf-8"?>

```



```

<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props"
  Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{4DCEC446-7156-4FE6-8CCC-219E34DD409D}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <StartupObject>VBHelloWorld.HelloWorld</StartupObject>
    <RootNamespace>VBHelloWorld</RootNamespace>
    <AssemblyName>VBHelloWorld</AssemblyName>
    <FileAlignment>512</FileAlignment>
    <MyType>Console</MyType>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <DefineDebug>true</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <OutputPath>bin\Debug\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <DefineDebug>false</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  </PropertyGroup>
  <PropertyGroup>
    <OptionExplicit>On</OptionExplicit>
  </PropertyGroup>
  <PropertyGroup>
    <OptionCompare>Binary</OptionCompare>

```

```

</PropertyGroup>
<PropertyGroup>
  <OptionStrict>Off</OptionStrict>
</PropertyGroup>
<PropertyGroup>
  <OptionInfer>On</OptionInfer>
</PropertyGroup>
<ItemGroup>
  <Reference Include="System" />
  <Reference Include="System.Data" />
  <Reference Include="System.Deployment" />
  <Reference Include="System.Xml" />
  <Reference Include="System.Core" />
  <Reference Include="System.Xml.Linq" />
  <Reference Include="System.Data.DataSetExtensions" />
  <Reference Include="System.Net.Http" />
</ItemGroup>
<ItemGroup>
  <Import Include="Microsoft.VisualBasic" />
  <Import Include="System" />
  <Import Include="System.Collections" />
  <Import Include="System.Collections.Generic" />
  <Import Include="System.Data" />
  <Import Include="System.Diagnostics" />
  <Import Include="System.Linq" />
  <Import Include="System.Xml.Linq" />
  <Import Include="System.Threading.Tasks" />
</ItemGroup>
<ItemGroup>
  <Compile Include="HelloWorld.vb" />
  <Compile Include="My Project\AssemblyInfo.vb" />
  <Compile Include="My Project\Application.Designer.vb">
    <AutoGen>True</AutoGen>
    <DependentUpon>Application.myapp</DependentUpon>
  </Compile>
  <Compile Include="My Project\Resources.Designer.vb">
    <AutoGen>True</AutoGen>
    <DesignTime>True</DesignTime>
    <DependentUpon>Resources.resx</DependentUpon>
  </Compile>
  <Compile Include="My Project\Settings.Designer.vb">
    <AutoGen>True</AutoGen>
    <DependentUpon>Settings.settings</DependentUpon>
    <DesignTimeSharedInput>True</DesignTimeSharedInput>
  </Compile>

```

```

    </Compile>
  </ItemGroup>
  <ItemGroup>
    <EmbeddedResource Include="My Project\Resources.resx">
      <Generator>VbMyResourcesResXFileCodeGenerator</Generator>
      <LastGenOutput>Resources.Designer.vb</LastGenOutput>
      <CustomToolNamespace>My.Resources</CustomToolNamespace>
      <SubType>Designer</SubType>
    </EmbeddedResource>
  </ItemGroup>
  <ItemGroup>
    <None Include="My Project\Application.myapp">
      <Generator>MyApplicationCodeGenerator</Generator>
      <LastGenOutput>Application.Designer.vb</LastGenOutput>
    </None>
    <None Include="My Project\Settings.settings">
      <Generator>SettingsSingleFileGenerator</Generator>
      <CustomToolNamespace>My</CustomToolNamespace>
      <LastGenOutput>Settings.Designer.vb</LastGenOutput>
    </None>
    <None Include="App.config" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.VisualBasic.targets" />
  <!-- To modify your build process, add your task inside one of the targets below and
  uncomment it.
      Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>

```

Application.Designer.vb (在 *(root directory name)\VBHelloWorld\My Project* 中) :

```

' -----
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.

```

```
' </auto-generated>
'-----

Option Strict On
Option Explicit On
```

Application.myapp (在 *(root directory name)*\VBHelloWorld\My Project 中) :

```
<?xml version="1.0" encoding="utf-8"?>
<MyApplicationData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MySubMain>false</MySubMain>
  <SingleInstance>false</SingleInstance>
  <ShutdownMode>0</ShutdownMode>
  <EnableVisualStyles>true</EnableVisualStyles>
  <AuthenticationMode>0</AuthenticationMode>
  <ApplicationType>2</ApplicationType>
  <SaveMySettingsOnExit>true</SaveMySettingsOnExit>
</MyApplicationData>
```

AssemblyInfo.vb (在 *(root directory name)*\VBHelloWorld\My Project 中) :

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices

' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("VBHelloWorld")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("VBHelloWorld")>
<Assembly: AssemblyCopyright("Copyright © 2017")>
<Assembly: AssemblyTrademark("")>

<Assembly: ComVisible(False)>

'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("137c362b-36ef-4c3e-84ab-f95082487a5a")>
```

```
' Version information for an assembly consists of the following four values:
'
' Major Version
' Minor Version
' Build Number
' Revision
'
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
' <Assembly: AssemblyVersion("1.0.*")>

<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyFileVersion("1.0.0.0")>
```

Resources.Designer.vb (在 *(root directory name)\VBHelloWorld\My Project* 中):

```
' -----
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
' -----
```

```
Option Strict On
Option Explicit On
```

```
Namespace My.Resources
```

```
'This class was auto-generated by the StronglyTypedResourceBuilder
'class via a tool like ResGen or Visual Studio.
'To add or remove a member, edit your .ResX file then rerun ResGen
'with the /str option, or rebuild your VS project.
'''<summary>
'''   A strongly-typed resource class, for looking up localized strings, etc.
'''</summary>
```

```
<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedRe
"4.0.0.0"), _
Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
```

```

Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
Global.Microsoft.VisualBasic.HideModuleNameAttribute())> _
Friend Module Resources

    Private resourceMan As Global.System.Resources.ResourceManager

    Private resourceCulture As Global.System.Globalization.CultureInfo

    '''<summary>
    ''' Returns the cached ResourceManager instance used by this class.
    '''</summary>

    <Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState.Never)>
    Friend ReadOnly Property ResourceManager() As
        Global.System.Resources.ResourceManager
        Get
            If Object.ReferenceEquals(resourceMan, Nothing) Then
                Dim temp As Global.System.Resources.ResourceManager = New
                Global.System.Resources.ResourceManager("VBHelloWorld.Resources",
                GetType(Resources).Assembly)
                resourceMan = temp
            End If
            Return resourceMan
        End Get
    End Property

    '''<summary>
    ''' Overrides the current thread's CurrentUICulture property for all
    ''' resource lookups using this strongly typed resource class.
    '''</summary>

    <Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState.Never)>
    Friend Property Culture() As Global.System.Globalization.CultureInfo
        Get
            Return resourceCulture
        End Get
        Set(ByVal value As Global.System.Globalization.CultureInfo)
            resourceCulture = value
        End Set
    End Property
End Module

```

End Namespace

Resources.resx (在 *(root directory name)*\VBHelloWorld\My Project 中) :

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<root>
```

```
<!--
```

```
Microsoft ResX Schema
```

```
Version 2.0
```

The primary goals of this format is to allow a simple XML format that is mostly human readable. The generation and parsing of the various data types are done through the TypeConverter classes associated with the data types.

Example:

```
... ado.net/XML headers & schema ...
```

```
<resheader name="resmimetype">text/microsoft-resx</resheader>
```

```
<resheader name="version">2.0</resheader>
```

```
<resheader name="reader">System.Resources.ResXResourceReader,  
System.Windows.Forms, ...</resheader>
```

```
<resheader name="writer">System.Resources.ResXResourceWriter,  
System.Windows.Forms, ...</resheader>
```

```
<data name="Name1"><value>this is my long string</value><comment>this is a  
comment</comment></data>
```

```
<data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
```

```
<data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
```

```
<value>[base64 mime encoded serialized .NET Framework object]</value>
```

```
</data>
```

```
<data name="Icon1" type="System.Drawing.Icon, System.Drawing"  
mimetype="application/x-microsoft.net.object.bytearray.base64">
```

```
<value>[base64 mime encoded string representing a byte array form of the .NET  
Framework object]</value>
```

```
<comment>This is a comment</comment>
```

```
</data>
```

There are any number of "resheader" rows that contain simple name/value pairs.

Each data row contains a name, and value. The row also contains a type or mimetype. Type corresponds to a .NET class that support

text/value conversion through the TypeConverter architecture. Classes that don't support this are serialized and stored with the mimetype set.

The mimetype is used for serialized objects, and tells the ResXResourceReader how to depersist the object. This is currently not extensible. For a given mimetype the value must be set accordingly:

Note - application/x-microsoft.net.object.binary.base64 is the format that the ResXResourceWriter will generate, however the reader can read any of the formats listed below.

```
mimetype: application/x-microsoft.net.object.binary.base64
value    : The object must be serialized with
          : System.Serialization.Formatters.Binary.BinaryFormatter
          : and then encoded with base64 encoding.
```

```
mimetype: application/x-microsoft.net.object.soap.base64
value    : The object must be serialized with
          : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
          : and then encoded with base64 encoding.
```

```
mimetype: application/x-microsoft.net.object.bytearray.base64
value    : The object must be serialized into a byte array
          : using a System.ComponentModel.TypeConverter
          : and then encoded with base64 encoding.
```

```
-->
```

```
<xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xsd:element name="root" msdata:IsDataSet="true">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="metadata">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="value" type="xsd:string" minOccurs="0" />
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" />
            <xsd:attribute name="type" type="xsd:string" />
            <xsd:attribute name="mimetype" type="xsd:string" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="assembly">
          <xsd:complexType>
```



```

        <xsd:attribute name="alias" type="xsd:string" />
        <xsd:attribute name="name" type="xsd:string" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="data">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
            <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" msdata:Ordinal="1" />
        <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
        <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="resheader">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required" />
    </xsd:complexType>
</xsd:element>
</xsd:choice>
</xsd:complexType>
</xsd:element>
</xsd:schema>
<resheader name="resmimetype">
    <value>text/microsoft-resx</value>
</resheader>
<resheader name="version">
    <value>2.0</value>
</resheader>
<resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>
<resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>

```

```
</root>
```

Settings.Designer.vb (在 *(root directory name)*\VBHelloWorld\My Project 中) :

```
' -----
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.42000
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
' -----

Option Strict On
Option Explicit On

Namespace My

    <Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
Global.System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.Settings
"11.0.0.0"), _

Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsable
-
    Partial Friend NotInheritable Class MySettings
        Inherits Global.System.Configuration.ApplicationSettingsBase

        Private Shared defaultInstance As MySettings =
CType(Global.System.Configuration.ApplicationSettingsBase.Synchronized(New
MySettings), MySettings)

        #Region "My.Settings Auto-Save Functionality"
            #If _MyType = "WindowsForms" Then
                Private Shared addedHandler As Boolean

                Private Shared addedHandlerLockObject As New Object

                <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),
Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsable
-

```

```

        Private Shared Sub AutoSaveSettings(ByVal sender As Global.System.Object, ByVal
e As Global.System.EventArgs)
            If My.Application.SaveMySettingsOnExit Then
                My.Settings.Save()
            End If
        End Sub
    #End If
#End Region

Public Shared ReadOnly Property [Default]() As MySettings
    Get

        #If _MyType = "WindowsForms" Then
            If Not addedHandler Then
                SyncLock addedHandlerLockObject
                    If Not addedHandler Then
                        AddHandler My.Application.Shutdown, AddressOf AutoSaveSettings
                        addedHandler = True
                    End If
                End SyncLock
            End If
        #End If
        Return defaultInstance
    End Get
End Property
End Class
End Namespace

Namespace My

    <Global.Microsoft.VisualBasic.HideModuleNameAttribute(), _
Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(> _
Friend Module MySettingsProperty

    <Global.System.ComponentModel.Design.HelpKeywordAttribute("My.Settings")> _
Friend ReadOnly Property Settings() As Global.VBHelloWorld.My.MySettings
    Get
        Return Global.VBHelloWorld.My.MySettings.Default
    End Get
End Property
End Module
End Namespace

```

Settings.settings (在 (*root directory name*)\VBHelloWorld\My Project 中) :

```
<?xml version='1.0' encoding='utf-8'?>
<SettingsFile xmlns="http://schemas.microsoft.com/VisualStudio/2004/01/settings"
  CurrentProfile="(Default)" UseMySettingsClassName="true">
  <Profiles>
    <Profile Name="(Default)" />
  </Profiles>
  <Settings />
</SettingsFile>
```

在 中規劃建置 AWS CodeBuild

使用 之前 AWS CodeBuild，您必須回答下列問題：

1. 原始程式碼存放在哪裡？CodeBuild 目前支援從下列來源碼儲存庫提供者建置。原始程式碼必須包含建置規格 (Buildspec) 檔案。buildspec 是組建命令和相關設定的集合，採用 YAML 格式，可供 CodeBuild 用來執行組建。您可以在組建專案定義中宣告 buildspec。

儲存庫提供者	必要	文件
CodeCommit：	儲存庫名稱。 (選用) 與原始碼相關聯的遞交 ID。	請參閱 AWS CodeCommit 使用者指南中的這些主題： 建立 CodeCommit 儲存庫 在 CodeCommit 中建立遞交
Amazon S3	輸入儲存貯體名稱。 與包含原始碼的組建輸入 ZIP 檔案相對應的物件名稱。 (選用) 與組建輸入 ZIP 檔案相關聯的版本 ID。	請參閱《Amazon S3 入門指南》中的這些主題： 建立儲存貯體 將物件新增到儲存貯體
GitHub	儲存庫名稱。 (選用) 與原始碼相關聯的遞交 ID。	請參閱 GitHub 說明網站上的這個主題： 建立儲存庫
Bitbucket	儲存庫名稱。	請參閱 Bitbucket Cloud 文件網站上的這個主題： 建立儲存庫

儲存庫提供者	必要	文件
	(選用) 與原始碼相關聯的遞交 ID。	

2. 您需要執行哪些組建命令和依什麼順序執行？根據預設，CodeBuild 會從您指定的提供者下載建置輸入，並將建置輸出上傳至您指定的儲存貯體。您使用 `buildspec` 來指示如何將下載的組建輸入變成預期的組建輸出。如需更多資訊，請參閱[Buildspec 參考](#)。
3. 您執行組建需要哪些執行時間和工具？例如，您是為 Java、Ruby、Python 或 Node.js 而組建嗎？組建需要 Maven 或 Ant，還是 Java、Ruby 或 Python 的編譯器？建置是否需要 Git AWS CLI、或其他工具？

CodeBuild 會在使用 Docker 映像的建置環境中執行組建。這些 Docker 映像必須存放在 CodeBuild 支援的儲存庫類型中。其中包括 CodeBuild Docker 映像儲存庫、Docker Hub 和 Amazon Elastic Container Registry (Amazon ECR)。如需 CodeBuild Docker 映像儲存庫的詳細資訊，請參閱[CodeBuild 提供的 Docker 映像](#)。

4. 您需要非 CodeBuild 自動提供 AWS 的資源嗎？如果是這樣，這些資源需要哪些安全政策？例如，您可能需要修改 CodeBuild 服務角色，以允許 CodeBuild 使用這些資源。
5. 您希望 CodeBuild 使用您的 VPC 嗎？若是如此，您需要有 VPC 組態的 VPC ID、子網路 ID 和安全群組 ID。如需詳細資訊，請參閱[AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用](#)。

回答這些問題之後，您應該就具備成功執行組建所需的設定和資源。若要執行組建，您可以：

- 使用 AWS CodeBuild 主控台 AWS CLI，或 AWS SDKs。如需詳細資訊，請參閱[手動執行組建](#)。
- 在 中建立或識別管道 AWS CodePipeline，然後新增組建或測試動作，指示 CodeBuild 自動測試您的程式碼、執行組建或兩者。如需詳細資訊，請參閱[搭配 CodePipeline 使用 CodeBuild CodePipeline](#)。

CodeBuild 的建置規格參考

本主題提供關於建置規格 (Buildspec) 檔案的重要參考資訊。buildspec 是組建命令和相關設定的集合，採用 YAML 格式，可供 CodeBuild 用來執行組建。您可以在原始程式碼中包含 buildspec，也可以在建立建置專案時定義 buildspec。如需組建規格運作方式的詳細資訊，請參閱[CodeBuild 的運作方式](#)。

主題

- [Buildspec 檔案名稱和儲存位置](#)
- [Buildspec 語法](#)
- [Buildspec 範例](#)
- [Buildspec 版本](#)
- [批次建置 buildspec 參考](#)

Buildspec 檔案名稱和儲存位置

若您隨著來源碼併入組建規格，依預設，組建規格檔案的名稱會是 buildspec.yml，並且放置在來源目錄的根目錄中。

您可以覆寫預設的組建規格檔案名稱和位置。例如，您可以：

- 對相同儲存庫中的不同組建使用不同的組建規格檔案，例如 buildspec_debug.yml 和 buildspec_release.yml。
- 在來源目錄根目錄以外的位置儲存組建規格檔案，例如 config/buildspec.yml，或儲存在 S3 儲存貯體中。S3 儲存貯體必須與建置專案位於相同的 AWS 區域。使用其 ARN 指定 buildspec 檔案 (例如，arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml)。

您只可以為組建專案指定一個組建規格，而不論組建規格檔案的名稱。

若要覆寫預設的組建規格檔案名稱、位置或兩者，請執行下列其中一個動作：

- 執行 AWS CLI create-project 或 update-project 命令，將 buildspec 值設定為相對於內建環境變數 值的替代 buildspec 檔案路徑 CODEBUILD_SRC_DIR。您也可以在 AWS SDKs 中執行與 create project 操作相同的操作。如需詳細資訊，請參閱 [建立組建專案](#) 或 [變更建置專案設定](#)。

- 執行 AWS CLI `start-build` 命令，將 `buildspecOverride` 值設定為相對於內建環境變數 值的替代 `buildspec` 檔案路徑 `CODEBUILD_SRC_DIR`。您也可以在 AWS SDKs 中執行與 `start build` 操作相同的操作。如需詳細資訊，請參閱[手動執行組建](#)。
- 在 AWS CloudFormation 範本中，將 資源 `Source` 中的 `BuildSpec` 屬性設定為相對於內建環境變數 值的替代 `buildspec` 檔案 `AWS::CodeBuild::Project` 路徑 `CODEBUILD_SRC_DIR`。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的[AWS CodeBuild 專案來源](#)中的 `BuildSpec` 屬性。

Buildspec 語法

組建規格檔案必須以 [YAML](#) 格式表達。

如果命令包含 YAML 不支援的字元或字元字串，您必須以引號 (") 括住命令。下列命令用引號括起來，因為 YAML 中不允許冒號 (:) 後面跟著空格。命令中的引號被逸出 (\")。

```
"export PACKAGE_NAME=$(cat package.json | grep name | head -1 | awk -F: '{ print $2 }' | sed 's/[\\\",]//g')"
```

組建規格具有下列語法：

```
version: 0.2

run-as: Linux-user-name

env:
  shell: shell-tag
  variables:
    key: "value"
    key: "value"
  parameter-store:
    key: "value"
    key: "value"
  exported-variables:
    - variable
    - variable
  secrets-manager:
    key: secret-id:json-key:version-stage:version-id
  git-credential-helper: no | yes

proxy:
```


upload-artifacts: no | yes

logs: no | yes

batch:

fast-fail: false | true

build-list:

build-matrix:

build-graph:

build-fanout:

phases:

install:

run-as: *Linux-user-name*

on-failure: ABORT | CONTINUE | RETRY | RETRY-*count* | RETRY-*regex* |

RETRY-*count-regex*

runtime-versions:

runtime: *version*

runtime: *version*

commands:

- *command*

- *command*

finally:

- *command*

- *command*

pre_build:

run-as: *Linux-user-name*

on-failure: ABORT | CONTINUE | RETRY | RETRY-*count* | RETRY-*regex* |

RETRY-*count-regex*

commands:

- *command*

- *command*

finally:

- *command*

- *command*

build:

run-as: *Linux-user-name*

on-failure: ABORT | CONTINUE | RETRY | RETRY-*count* | RETRY-*regex* |

RETRY-*count-regex*

commands:

- *command*

- *command*

finally:

- *command*
- *command*

post_build:run-as: *Linux-user-name*on-failure: ABORT | CONTINUE | RETRY | RETRY-*count* | RETRY-*regex* |RETRY-*count-regex*commands:

- *command*
- *command*

finally:

- *command*
- *command*

reports:*report-group-name-or-arn*:files:

- *location*
- *location*

base-directory: *location*discard-paths: no | yesfile-format: *report-format*artifacts:files:

- *location*
- *location*

name: *artifact-name*discard-paths: no | yesbase-directory: *location*exclude-paths: *excluded paths*enable-symlinks: no | yess3-prefix: *prefix*secondary-artifacts:*artifactIdentifier*:files:

- *location*
- *location*

name: *secondary-artifact-name*discard-paths: no | yesbase-directory: *location**artifactIdentifier*:files:

- *location*
- *location*

```
  discard-paths: no | yes
  base-directory: location
cache:
  key: key
  fallback-keys:
    - fallback-key
    - fallback-key
  action: restore | save
  paths:
    - path
    - path
```

建置規格包含下列各項：

version

必要的映射。代表組建規格版本。建議您使用 0.2。

Note

雖然仍支援版本 0.1，建議您盡可能使用版本 0.2。如需詳細資訊，請參閱[Buildspec 版本](#)。

執行方式

選用的序列。僅供 Linux 使用者使用。指定在此 buildspec 檔案中執行命令的 Linux 使用者。會 run-as 授予指定的使用者讀取和執行許可。在 buildspec 檔案上方指定 run-as 時，它會全域套用到所有命令。如果您不想為所有 buildspec 檔案命令指定一個使用者，您可以在其中一個 phases 區塊中使用 run-as 來為階段中的命令指定一個使用者。如果未指定 run-as，則會以根使用者身分執行所有命令。

env

選用的序列。代表一或多個自訂環境變數的資訊。

Note

為了保護敏感資訊，CodeBuild 日誌中會隱藏下列項目：

- AWS 存取金鑰 IDs。如需詳細資訊，請參閱 AWS Identity and Access Management 《使用者指南》中的[管理 IAM 使用者的存取金鑰](#)。

- 使用參數存放區指定的字串。如需詳細資訊，請參閱《Amazon EC2 Systems Manager [Systems Manager 使用者指南](#)》中的 [Systems Manager 參數存放區](#)和 [Systems Manager 參數存放區主控台演練](#)。
- 使用 指定的字串 AWS Secrets Manager。如需詳細資訊，請參閱[金鑰管理](#)。

Note

遮罩符合存放在 Amazon EC2 Systems Manager 參數存放區 或 中的確切值 AWS Secrets Manager。如果組建命令在秘密出現在日誌之前轉換秘密，則結果會是未遮罩的不同字串。例如，使用 Base64 編碼、反轉、變更大小寫，或將其與其他文字串連，都會建立不同的值，並會出現在建置日誌中。

若要讓值保持遮罩，請以建置使用的格式存放。例如，如果您的建置需要 Base64-encoded 值，請將 Base64-encoded 字串直接存放在 AWS Secrets Manager 或 Amazon EC2 Systems Manager 參數存放區中。如此一來，記錄的值就會符合儲存的值。

env/shell

選用的序列。指定 Linux 或 Windows 作業系統支援的 Shell。

對於 Linux 作業系統，支援的 shell 標籤為：

- bash
- /bin/sh

對於 Windows 作業系統，支援的 shell 標籤為：

- powershell.exe
- cmd.exe

env/variables

如果指定 env，且您想要以純文字定義自訂環境變數，則為必要。包含 *key/value* 純量的映射，其中的每個映射代表純文字的單一自訂環境變數。*key* 為自訂環境變數的名稱，而 *value* 為該變數的值。

 Important

我們強烈建議不將敏感值儲存在環境變數中。環境變數可以使用 CodeBuild 主控台和等工具以純文字顯示 AWS CLI。針對機密值，建議您改為使用 `parameter-store` 或 `secrets-manager` 映射，如本節稍後所述。

任何您設定的環境變數都會取代現有環境變數。例如，如果 Docker 影像已包含名為 `MY_VAR` 且值為 `my_value` 的環境變數，而且您設定名為 `MY_VAR` 且值為 `other_value` 的環境變數，則 `my_value` 會取代為 `other_value`。同樣地，如果 Docker 影像已包含名為 `PATH` 且值為 `/usr/local/sbin:/usr/local/bin` 的環境變數，而且您設定名為 `PATH` 且值為 `$PATH:/usr/share/ant/bin` 的環境變數，則 `/usr/local/sbin:/usr/local/bin` 會取代為文字值 `$PATH:/usr/share/ant/bin`。

請不要設定名稱開頭為 `CODEBUILD_` 的任何環境變數。此字首保留供內部使用。

如果有多個位置定義同名的環境變數，則會決定值，如下所示：

- 開始建置操作呼叫中的值會採用最高優先順序。當您建立建置時，您可以新增或覆寫環境變數。如需詳細資訊，請參閱[手動執行 AWS CodeBuild 組建](#)。
- 組建專案定義中的值會採用下一個優先順序。當您建立或編輯專案時，您可以在專案層級新增環境變數。如需詳細資訊，請參閱[在中建立建置專案 AWS CodeBuild](#)及[在中變更建置專案設定 AWS CodeBuild](#)。
- `buildspec` 宣告中的值會採用最低優先順序。

env/parameter-store

如果已指定 `env`，且您想要擷取存放在 Amazon EC2 Systems Manager 參數存放區的自訂環境變數 `env`，則為必要項目。包含索引 `##` 純量的映射，其中每個映射代表存放在 Amazon EC2 Systems Manager 參數存放區中的單一自訂環境變數。`###`是您稍後在建置命令中用來參考此自訂環境變數的名稱，而 `#`是存放在 Amazon EC2 Systems Manager 參數存放區中的自訂環境變數名稱。若要存放敏感值，請參閱《Amazon EC2 Systems Manager 使用者指南》中的 [Systems Manager 參數存放區](#)和[逐步解說：建立和測試字串參數（主控台）](#)。

 Important

若要允許 CodeBuild 擷取存放在 Amazon EC2 Systems Manager 參數存放區中的自訂環境變數，您必須將 `ssm:GetParameters` 動作新增至 CodeBuild 服務角色。如需詳細資訊，請參閱[允許 CodeBuild 與其他 AWS 服務互動](#)。

您從 Amazon EC2 Systems Manager 參數存放區擷取的任何環境變數都會取代現有的環境變數。例如，如果 Docker 影像已包含名為 `MY_VAR` 且值為 `my_value` 的環境變

數，而且您擷取名為 MY_VAR 且值為 other_value 的環境變數，則 my_value 會取代為 other_value。同樣地，如果 Docker 影像已包含名為 PATH 且值為 /usr/local/sbin:/usr/local/bin 的環境變數，而且您擷取名為 PATH 且值為 \$PATH:/usr/share/ant/bin 的環境變數，則 /usr/local/sbin:/usr/local/bin 會取代為文字值 \$PATH:/usr/share/ant/bin。

請不要存放名稱開頭為 CODEBUILD_ 的任何環境變數。此字首保留供內部使用。

如果有多個位置定義同名的環境變數，則會決定值，如下所示：

- 開始建置操作呼叫中的值會採用最高優先順序。當您建立建置時，您可以新增或覆寫環境變數。如需詳細資訊，請參閱[手動執行 AWS CodeBuild 組建](#)。
- 組建專案定義中的值會採用下一個優先順序。當您建立或編輯專案時，您可以在專案層級新增環境變數。如需詳細資訊，請參閱[在 中建立建置專案 AWS CodeBuild](#)及[在 中變更建置專案設定 AWS CodeBuild](#)。
- buildspec 宣告中的值會採用最低優先順序。

env/secrets-manager

如果您想要擷取存放在 中的自訂環境變數，則為必要項目 AWS Secrets Manager。reference-key 使用以下模式指定 Secrets Manager：

```
<key>: <secret-id>:<json-key>:<version-stage>:<version-id>
```

```
<key>
```

(必要) 本機環境變數名稱。使用此名稱在建置期間存取 變數。

```
<secret-id>
```

(必要) 做為秘密唯一識別符的名稱或 Amazon Resource Name (ARN)。若要存取您 AWS 帳戶中的秘密，只需要指定秘密名稱。若要存取不同 AWS 帳戶中的秘密，請指定秘密 ARN。

```
<json-key>
```

(選用) 指定您要擷取其值之 Secrets Manager 金鑰/值對的金鑰名稱。如果您未指定 json-key，CodeBuild 會擷取整個秘密文字。

```
<####>
```

(選用) 透過附加至版本的預備標籤，指定您要擷取的秘密版本。預備標籤在輪換程序期間用來追蹤不同版本。如果您使用 version-stage，請不要指定 version-id。如果您不指定版本階段或版本 ID，則預設會擷取版本階段值為 AWSCURRENT 的版本。

`<version-id>`

(選用) 指定您要使用的秘密版本的唯一識別符。如果指定 `version-id`，則不要指定 `version-stage`。如果您不指定版本階段或版本 ID，則預設會擷取版本階段值為 `AWSCURRENT` 的版本。

在下列範例中，`TestSecret` 是存放在 `Secrets Manager` 中的鍵/值對名稱。的金鑰 `TestSecret` 是 `MY_SECRET_VAR`。您可以使用 `LOCAL_SECRET_VAR` 名稱在建置期間存取變數。

```
env:
  secrets-manager:
    LOCAL_SECRET_VAR: "TestSecret:MY_SECRET_VAR"
```

如需詳細資訊，請參閱 `AWS Secrets Manager` 使用者指南中的 [什麼是 AWS Secrets Manager](#)。

`env/exported-variables`

選用的映射。用於列出您想要匯出的環境變數。在 `exported-variables` 下以獨立行列指定您想要匯出的每個變數的名稱。建置期間您想要匯出的變數必須在容器中提供。您匯出的變數可為環境變數。

匯出的環境變數會與 `搭配使用 AWS CodePipeline`，以將環境變數從目前的建置階段匯出至管道中的後續階段。如需詳細資訊，請參閱 `AWS CodePipeline` 《使用者指南》中的 [使用變數](#)。

建置期間，變數的值在 `install` 階段即開始提供。它可以在 `install` 階段開始和 `post_build` 結束之間進行更新。`post_build` 階段結束後，匯出的變數值無法變更。

Note

無法匯出下列項目：

- 建置專案中指定的 `Amazon EC2 Systems Manager` 參數存放區秘密。
- 建置專案中指定的 `Secrets Manager` 秘密
- 以 `AWS_` 開頭的環境變數。

`env/git-credential-helper`

選用的映射。用於指示 `CodeBuild` 是否使用其 `Git` 登入資料協助程式來提供 `Git` 登入資料。`yes` 如果使用的話。否則為 `no` 或未指定。如需詳細資訊，請參閱 `Git` 網站上的 [gitcredentials](#)。

Note

由公有 Git 儲存庫的 Webhook 觸發的建置不支援 git-credential-helper。

proxy

選用的序列。如果在明確的代理伺服器中執行建置，則用於表示設定。如需詳細資訊，請參閱[在明確代理伺服器中執行 CodeBuild](#)。

proxy/upload-artifacts

選用的映射。如果您想要在明確的代理伺服器中的建置上傳成品，請設為 yes。預設值為 no。

proxy/logs

選用的映射。將明確代理伺服器中建置yes的 設定為 ，以建立 CloudWatch 日誌。預設值為 no。

階段

必要的序列。代表 CodeBuild 在建置的每個階段執行的命令。

Note

在 buildspec 0.1 版中，CodeBuild 會在建置環境中預設 Shell 的個別執行個體中執行每個命令。這表示每個命令會與所有其他命令隔離執行。因此，根據預設，如果單一命令倚賴任何之前命令的狀態 (例如，變更目錄或設定環境變數)，您就無法加以執行。為因應這個限制，我們建議您使用 0.2 版，它可解決這個問題。如果您必須使用組建規格版本 0.1，我們建議使用[建置環境中的 Shell 和命令](#)中的方法。

phases/*/run-as

選用的序列。在建置階段中用來指定可執行其命令的 Linux 使用者。如果也在 buildspec 檔案上方為所有命令全域指定 run-as，則階段層級的使用者具有優先權。例如，如果全域run-as指定 User-1，而對於install階段只有一個run-as陳述式指定 User-2，則 buildspec 檔案中的所有命令都會以 User-1 執行，但install階段中的命令除外，這些命令會以 User-2 執行。

phase/*/on-failure

選用的序列。指定在階段期間發生失敗時要採取的動作。這可以是下列其中一個值：

- ABORT - 中止建置。
- CONTINUE - 繼續下一個階段。
- RETRY - 重試建置最多 3 次，並顯示符合規則表達式的錯誤訊息。`*`。
- RETRY-*count* - 以符合規則表達式的錯誤訊息*##*表示，在指定的次數內重試組建。`*`。請注意，*##*必須介於 0 到 100 之間。例如，有效值包括 RETRY-4 和 RETRY-8。
- RETRY-*regex* - 重試建置最多 3 次，並使用 *regex* 包含規則運算式以符合指定的錯誤訊息。例如，有效值包括 Retry-.*Error: Unable to connect to database.* 和 RETRY-invalid+。
- RETRY-*count-regex* - 在指定的次數內重試組建，如*##*所示。請注意，*##*必須介於 0 到 100 之間。您也可以使用 *regex* 來包含規則表達式，以符合錯誤訊息。例如，有效值包括 Retry-3-.*connection timed out.* 和 RETRY-8-invalid+。

如果未指定此屬性，則失敗程序會遵循轉換階段，如所示[建置階段轉換](#)。

Important

使用 Lambda 運算或預留容量時，不支援 on-failure 屬性。此屬性僅適用於 CodeBuild 提供的 EC2 運算映像。

階段/*/最終

選用的區塊。finally 區塊中指定的命令會在 commands 區塊中的命令之後執行。即使 finally 區塊中的命令失敗，也會執行 commands 區塊中的命令。例如，如果 commands 區塊包含三個命令且第一個失敗，CodeBuild 會略過剩餘的兩個命令，並在 finally 區塊中執行任何命令。當 commands 和 finally 區塊中的所有命令成功執行時，此階段即成功。如果階段中的任何命令失敗，階段即失敗。

允許的建置階段名稱為：

phases/install

選用的序列。代表 CodeBuild 在安裝期間執行的任何命令。建議您僅針對在組建環境中安裝套件使用 install 階段。例如，您可能會使用此階段來安裝程式碼測試架構，例如 Mocha 或 RSpec。

phases/install/runtime-versions

選用的序列。Ubuntu 標準映像 5.0 或更新版本和 Amazon Linux 2 標準映像 4.0 或更新版本支援執行期版本。如已指定，此區段中至少要包含一個執行時間。使用特定版本指定執行時

間，主要版本後面接著 `.x`，以指定 CodeBuild 使用該主要版本搭配其最新的次要版本，或使用 `latest` 最新的主要和次要版本（例如 `ruby: 3.2`、`nodejs: 18.x` 或 `java: latest`）。您可以使用數字或環境變數指定執行時間。例如，如果您使用 Amazon Linux 2 標準映像 4.0，則以下指定已安裝 Java 版本 17、python 版本 3 的最新次要版本，以及 Ruby 環境變數中包含的版本。如需詳細資訊，請參閱 [CodeBuild 提供的 Docker 映像](#)。

```
phases:
  install:
    runtime-versions:
      java: corretto8
      python: 3.x
      ruby: "$MY_RUBY_VAR"
```

您可以在 `buildspec` 檔案的 `runtime-versions` 區段中指定一或多個執行時間。如果您的執行階段依存於另一個執行時間，您也可以在此檔案中指定其相依的執行時間。如果您未在 `buildspec` 檔案中指定任何執行時間，CodeBuild 會選擇您所使用的映像中可用的預設執行時間。如果您指定一或多個執行時間，CodeBuild 只會使用這些執行時間。如果未指定相依執行時間，CodeBuild 會嘗試為您選擇相依執行時間。

Note

如果未指定執行時間版本，CodeBuild 會使用預設版本。當先前的預設版本達到生命週期結束 (EOL) 時，預設版本可能會變更。為避免意外變更建置環境，建議您在 `buildspec` 檔案中指定執行時間版本。

如果兩個指定的執行時間發生衝突，則建置會失敗。例如，`android: 29` 和 `java: openjdk11` 相衝突，所以如果指定這兩個，則組建會失敗。

如需可用執行時間的詳細資訊，請參閱 [可用的執行時間](#)。

Note

如果您指定 `runtime-versions` 區段並使用 Ubuntu Standard Image 2.0 或更新版本以外的映像，或 Amazon Linux 2 (AL2) 標準映像 1.0 或更新版本，則建置會發出警告 "Skipping install of runtimes. Runtime version selection is not supported by this build image."。

phases/install/commands

選用的序列。包含一系列純量，其中每個純量代表 CodeBuild 在安裝期間執行的單一命令。CodeBuild 從頭到尾依序執行每個命令，一次一個。

phases/pre_build

選用的序列。如果有，代表 CodeBuild 在建置之前執行的命令。例如，您可以使用此階段登入 Amazon ECR，也可以安裝 npm 相依性。

phases/pre_build/commands

只有在指定 pre_build 時才為必要序列。包含一系列純量，其中每個純量代表 CodeBuild 在建置之前執行的單一命令。CodeBuild 從頭到尾依列出的順序一次執行每個命令。

phases/build

選用的序列。如果有，代表 CodeBuild 在建置期間執行的命令。例如，您可能會使用此階段來執行 Mocha、RSpec 或 sbt。

phases/build/commands

如果指定 build，則為必要項目。包含一系列純量，其中每個純量代表 CodeBuild 在建置期間執行的單一命令。CodeBuild 從頭到尾依列出的順序一次執行每個命令。

phases/post_build

選用的序列。如果有，代表 CodeBuild 在建置之後執行的命令。例如，您可以使用 Maven 將建置成品封裝到 JAR 或 WAR 檔案中，也可以將 Docker 映像推送到 Amazon ECR。然後，您可以透過 Amazon SNS 傳送建置通知。

phases/post_build/commands

如果指定 post_build，則為必要項目。包含一系列純量，其中每個純量代表 CodeBuild 在建置之後執行的單一命令。CodeBuild 從頭到尾依列出的順序一次執行每個命令。

報告

report-group-name-or-arn

選用的序列。指定傳送報告的目標報告群組。一個專案最多可以擁有五個報告群組。指定現有報告群組的 ARN，或新報告群組的名稱。如果您指定名稱，CodeBuild 會使用專案名稱和您以格式指定的名稱來建立報告群組<project-name>-<report-group-name>。您也可以使用 buildspec

中的環境變數來設定報告群組名稱，例如 `$REPORT_GROUP_NAME`。如需詳細資訊，請參閱[報告群組命名](#)。

`reports/<report-group>/files`

必要的序列。代表包含由建立產生之測試結果原始資料的位置。包含一系列純量，每個純量代表 CodeBuild 可以尋找測試檔案的個別位置，相對於原始建置位置，或如果設定，則為 `base-directory`。位置可能包括下列：

- 單一檔案 (例如，`my-test-report-file.json`)。
- 子目錄中的單一檔案 (例如，`my-subdirectory/my-test-report-file.json` 或 `my-parent-subdirectory/my-subdirectory/my-test-report-file.json`)。
- `'**/*'` 代表遞迴的所有檔案。
- `my-subdirectory/*` 代表名為 `my-subdirectory` 的子目錄中的所有檔案。
- `my-subdirectory/**/*` 代表從名為 `my-subdirectory` 開始的子目錄遞迴的所有檔案。

`reports/<report-group>/file-format`

選用的映射。代表報告檔案格式。如果未指定，則會使用 `JUNITXML`。此值不區分大小寫。可能值為：

測試報告

CUCUMBERJSON

Cucumber JSON

JUNITXML

JUnit XML

NUNITXML

NUnit XML

NUNIT3XML

NUnit 3 XML

TESTNGXML

TestNG XML

VISUALSTUDIOTRX

Visual Studio TRX

程式碼涵蓋範圍報告

CLOVERXML

Clover XML

COBERTURAXML

Cobertura XML

JACOCOXML

JaCoCo XML

SIMPLECOV

SimpleCov JSON

Note

CodeBuild 接受 [simplecov](#) 產生的 JSON 程式碼涵蓋範圍報告，而不是 [simplecov-json](#)。

reports/<report-group>/base-directory

選用的映射。代表一個或多個最上層目錄，相對於原始建置位置，CodeBuild 用來判斷在何處尋找原始測試檔案。

reports/<report-group>/discard-paths

選用。指定報告檔案目錄是否在輸出中平面化。如果未指定或包含 no，則報告檔案會以完整的目錄結構輸出。如果包含 yes，則所有的測試檔案都會放置在相同的輸出目錄中。例如，如果測試結果的路徑是 com/myapp/mytests/TestResult.xml，則指定 yes 會將此檔案置於 /TestResult.xml 中。

成品

選用的序列。代表 CodeBuild 尋找建置輸出的位置，以及 CodeBuild 如何準備上傳到 S3 輸出儲存體的相關資訊。例如，如果您正在建置 Docker 映像並將其推送至 Amazon ECR，或者正在對原始程式碼執行單元測試，但未建置它，則不需要此序列。

 Note

Amazon S3 中繼資料具有名為 `x-amz-meta-codebuild-buildarn` 的 CodeBuild 標頭，其中包含將成品發佈至 Amazon S3 `buildArn` 的 CodeBuild 組建的 `buildArn`。新增 `buildArn` 以允許來源追蹤通知，並參考產生成品的建置。

artifacts/files

必要的序列。代表包含組建環境中組建輸出成品的位置。包含一系列純量，每個純量代表一個單獨的位置，其中 CodeBuild 可以找到相對於原始組建位置的組建輸出成品，如果設定，則為基礎目錄。位置可能包括下列：

- 單一檔案 (例如，`my-file.jar`)。
- 子目錄中的單一檔案 (例如，`my-subdirectory/my-file.jar` 或 `my-parent-subdirectory/my-subdirectory/my-file.jar`)。
- `**/*` 代表遞迴的所有檔案。
- `my-subdirectory/*` 代表名為 `my-subdirectory` 的子目錄中的所有檔案。
- `my-subdirectory/**/*` 代表從名為 `my-subdirectory` 開始的子目錄遞迴的所有檔案。

當您指定建置輸出成品位置時，CodeBuild 可以在建置環境中找到原始建置位置。您不需要在組建成品輸出位置前方附加原始組建位置的路徑或指定 `./` 之類內容。如果要知道此位置的路徑，您可以在組建期間執行 `echo $CODEBUILD_SRC_DIR` 之類命令。每個組建環境的位置可能稍有不同。

artifacts/name

選用名稱。指定組建成品的名稱。當下列其中一項成立時，會使用此名稱。

- 您可以使用 CodeBuild API 來建立組建，並在更新專案、建立專案或啟動組建時，在 `ProjectArtifacts` 物件上設定 `overrideArtifactName` 旗標。
- 您可以使用 CodeBuild 主控台來建立組建、在 `buildspec` 檔案中指定名稱，並在建立或更新專案時選取啟用語意版本控制。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#)。

您可以在組建時計算的組建規格檔案中指定名稱。組建規格檔案中指定的名稱使用 Shell 命令語言。例如，您可以將日期和時間附加到成品名稱，讓它一律是唯一的。唯一成品名稱可防止覆寫成品。如需詳細資訊，請參閱 [Shell 命令語言](#)。

- 這是成品名稱附加了成品建立日期的範例。

```
version: 0.2
phases:
```

```
build:
  commands:
    - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

- 這是使用 CodeBuild 環境變數的成品名稱範例。如需詳細資訊，請參閱[建置環境中的環境變數](#)。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

- 這是成品名稱的範例，其使用 CodeBuild 環境變數並附加成品的建立日期。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: $AWS_REGION-$(date +%Y-%m-%d)
```

您可以將路徑資訊新增至名稱，以便根據名稱中的路徑將具名成品放置在目錄中。在此範例中，建置成品會放置在下的輸出中builds/<build number>/my-artifacts。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
```

```
name: builds/$CODEBUILD_BUILD_NUMBER/my-artifacts
```

artifacts/discard-paths

選用。指定組建成品目錄是否在輸出中平面化。如果未指定或包含 `no`，則會以完整的目錄結構輸出組建成品。如果包含 `yes`，則所有組建成品都會放置在相同的輸出目錄中。例如，如果組建輸出成品中的文件路徑是 `com/mycompany/app/HelloWorld.java`，則會指定 `yes` 將此文件放置在 `/HelloWorld.java` 中。

artifacts/base-directory

選用的映射。代表一個或多個最上層目錄，相對於原始建置位置，CodeBuild 用來決定要包含在建置輸出成品中的檔案和子目錄。有效值包含：

- 單一上層目錄 (例如，`my-directory`)。
- `'my-directory*'` 代表名稱開頭為 `my-directory` 的所有上層目錄。

相符的上層目錄不會包含在組建輸出成品中，只有其檔案和子目錄。

您可以使用 `files` 和 `discard-paths` 以進一步限制包含的檔案和子目錄。例如，針對下列目錄結構：

```
.  
### my-build-1  
#   ### my-file-1.txt  
### my-build-2  
    ### my-file-2.txt  
    ### my-subdirectory  
        ### my-file-3.txt
```

以及下列 `artifacts` 序列：

```
artifacts:  
  files:  
    - '*my-file-3.txt'  
  base-directory: my-build-2
```

下列子目錄和檔案會包含在組建輸出成品中：

```
.  
### my-subdirectory  
    ### my-file-3.txt
```


當下列 artifacts 序列：

```
artifacts:
  files:
    - '**/*'
  base-directory: 'my-build*'
  discard-paths: yes
```

下列檔案會包含在組建輸出成品中：

```
.
### my-file-1.txt
### my-file-2.txt
### my-file-3.txt
```

artifacts/exclude-paths

選用的映射。代表 CodeBuild 將從建置成品中排除base-directory的一或多個路徑。星號(*) 字元符合不超過資料夾邊界之名稱元件的零個或多個字元。雙星號(**) 會比對所有目錄中名稱元件的零個或多個字元。

exclude-paths 的範例包括下列項目：

- 若要從所有目錄排除檔案：`"**/file-name/**/*"`
- 若要排除所有點資料夾：`"**/.*/**/*"`
- 若要排除所有點檔案：`"**/.*"`

artifacts/enable-symlinks

選用。如果輸出類型為 ZIP，請指定內部符號連結是否保留在 ZIP 檔案中。如果其中包含 yes，則來源中的所有內部符號連結都會保留在成品 ZIP 檔案中。

artifacts/s3 字首

選用。指定當成品輸出至 Amazon S3 儲存貯體且命名空間類型為 時所使用的字首BUILD_ID。使用時，儲存貯體中的輸出路徑為 <s3-prefix>/<build-id>/<name>.zip。

artifacts/secondary-artifacts

選用的序列。代表一或多個成品定義，做為成品識別符與成品定義之間的映射。此區塊中的每個成品識別符必須符合專案secondaryArtifacts屬性中定義的成品。每個個別的定義具有與以上 artifacts 區塊相同的語法。

 Note

即使只有已定義的次要成品，也一律需要[artifacts/files](#)序列。

例如，如果您的專案具有以下結構：

```
{
  "name": "sample-project",
  "secondaryArtifacts": [
    {
      "type": "S3",
      "location": "<output-bucket1>",
      "artifactIdentifier": "artifact1",
      "name": "secondary-artifact-name-1"
    },
    {
      "type": "S3",
      "location": "<output-bucket2>",
      "artifactIdentifier": "artifact2",
      "name": "secondary-artifact-name-2"
    }
  ]
}
```

則您的 buildspec 如下所示：

```
version: 0.2

phases:
build:
  commands:
    - echo Building...
artifacts:
  files:
    - '**/*'
  secondary-artifacts:
    artifact1:
      files:
        - directory/file1
      name: secondary-artifact-name-1
    artifact2:
```

```
files:
  - directory/file2
name: secondary-artifact-name-2
```

快取

選用的序列。代表 CodeBuild 可以準備檔案以將快取上傳至 S3 快取儲存貯體的相關資訊。如果專案的快取類型為 No Cache，則此序列不是必要。

cache/key

選用的序列。代表搜尋或還原快取時所使用的主索引鍵。CodeBuild 會完全符合主索引鍵。

以下是 金鑰的範例：

```
key: npm-key-${codebuild-hash-files package-lock.json} }
```

cache/fallback-keys

選用的序列。當無法使用主索引鍵找到快取時，代表循序使用的備用索引鍵清單。最多支援五個備用金鑰，每個金鑰都使用字首搜尋進行比對。如果未提供金鑰，則會忽略此序列。

以下是備用金鑰的範例：

```
fallback-keys:
  - npm-key-${codebuild-hash-files package-lock.json} }
  - npm-key-
  - npm-
```

cache/action

選用的序列。指定要在快取上執行的動作。有效值包含：

- restore 只會還原快取，而不會儲存更新。
- save 只會儲存快取，而不會還原先前的版本。

如果未提供任何值，CodeBuild 會預設為同時執行還原和儲存。

cache/paths

必要的序列。代表快取的位置。包含一系列純量，每個純量代表一個單獨的位置，其中 CodeBuild 可以找到相對於原始組建位置的組建輸出成品，如果設定，則為基礎目錄。位置可能包括下列：

- 單一檔案 (例如, `my-file.jar`)。
- 子目錄中的單一檔案 (例如, `my-subdirectory/my-file.jar` 或 `my-parent-subdirectory/my-subdirectory/my-file.jar`)。
- `'**/*'` 代表遞迴的所有檔案。
- `my-subdirectory/*` 代表名為 `my-subdirectory` 的子目錄中的所有檔案。
- `my-subdirectory/**/*` 代表從名為 `my-subdirectory` 開始的子目錄遞迴的所有檔案。

Important

因為建置規格宣告必須為有效的 YAML，因此建置規格宣告中的間距相當重要。若您組建規格宣告中的空格數無效，組建會立即失敗。您可以使用 YAML 驗證程式來測試您的組建規格宣告是否為有效的 YAML。

如果您在建立或更新組建專案時使用 AWS CLI 或 AWS SDKs 宣告 `buildspec`，則 `buildspec` 必須是以 YAML 格式表示的單一字串，以及必要的空格和換行逸出字元。下一節提供一個範例。

如果您使用 CodeBuild 或 AWS CodePipeline 主控台而非 `buildspec.yml` 檔案，您只能插入 `build` 階段的命令。不要使用前述語法，您應該在單一行中列出要在組建階段期間執行的所有命令。針對多個命令，以 `&&` 區隔每個命令 (例如, `mvn test && mvn package`)。

您可以使用 CodeBuild 或 CodePipeline 主控台來指定組建環境中組建輸出成品的位置，而非 `buildspec.yml` 檔案。不要使用前述語法，而是應該在單一行中列出所有位置。針對多個位置，以逗號區隔每個位置 (例如, `buildspec.yml, target/my-app.jar`)。

Buildspec 範例

以下是 `buildspec.yml` 檔案的範例。

```
version: 0.2

env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword

phases:
  install:
    commands:
```

```
- echo Entered the install phase...
- apt-get update -y
- apt-get install -y maven
finally:
- echo This always runs even if the update or install command fails
pre_build:
  commands:
    - echo Entered the pre_build phase...
    - docker login -u User -p $LOGIN_PASSWORD
  finally:
    - echo This always runs even if the login command fails
build:
  commands:
    - echo Entered the build phase...
    - echo Build started on `date`
    - mvn install
  finally:
    - echo This always runs even if the install command fails
post_build:
  commands:
    - echo Entered the post_build phase...
    - echo Build completed on `date`

reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
  files:
    - "**/*"
  base-directory: 'target/tests/reports'
  discard-paths: no
reportGroupCucumberJson:
  files:
    - 'cucumber/target/cucumber-tests.xml'
  discard-paths: yes
  file-format: CUCUMBERJSON # default is JUNITXML
artifacts:
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
  secondary-artifacts:
    artifact1:
      files:
        - target/artifact-1.0.jar
      discard-paths: yes
    artifact2:
```

```
files:
  - target/artifact-2.0.jar
discard-paths: yes
cache:
  paths:
    - '/root/.m2/**/*'
```

以下是上述 buildspec 的範例，以單一字串表示，可與 AWS CLI 或 AWS SDKs 搭配使用。

```
"version: 0.2\n\nenv:\n  variables:\n    JAVA_HOME: \"/usr/lib/jvm/java-8-openjdk-  
amd64\n\nparameter-store:\n  LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword\n\nphases:\n\ninstall:\n  commands:\n    - echo Entered the install phase...\n    - apt-get update -y\n    - apt-get install -y maven\n  finally:\n    - echo This always runs even if the update or install command fails\n\npre_build:\n  commands:\n    - echo Entered the pre_build phase...\n    - docker login -u User -p $LOGIN_PASSWORD\n  finally:\n    - echo This always runs even if the login command fails\n\nbuild:\n  commands:\n    - echo Entered the build phase...\n    - echo Build started on `date`\n    - mvn install\n  finally:\n    - echo This always runs even if the install command fails\n\npost_build:\n  commands:\n    - echo Entered the post_build phase...\n    - echo Build completed on `date`\n\nreports:\n  reportGroupJUnitXml:\n    files:\n      - \"**/*\"\n    base-directory: 'target/tests/reports'\n    discard-paths: false\n  reportGroupCucumberJson:\n    files:\n      - 'cucumber/target/cucumber-tests.xml'\n    file-format: CUCUMBERJSON\n\nartifacts:\n  files:\n    - target/messageUtil-1.0.jar\n  discard-paths: yes\n  secondary-artifacts:\n    artifact1:\n      files:\n        - target/messageUtil-1.0.jar\n      discard-paths: yes\n    artifact2:\n      files:\n        - target/messageUtil-1.0.jar\n      discard-paths: yes\n  cache:\n    paths:\n      - '/root/.m2/**/*'"
```

以下是 build 階段中的命令範例，可與 CodeBuild 或 CodePipeline 主控台搭配使用。

```
echo Build started on `date` && mvn install
```

在這些範例中：

- 純文字的自訂環境變數，已設定 JAVA_HOME 的金鑰和 /usr/lib/jvm/java-8-openjdk-amd64 的值。
- dockerLoginPassword 您儲存在 Amazon EC2 Systems Manager 參數存放區中名為 的自訂環境變數，稍後會在建置命令中使用金鑰 進行參考 LOGIN_PASSWORD。
- 您無法變更這些組建階段名稱。在此範例中執行的命令為 apt-get update -y 和 apt-get install -y maven (安裝 Apache Maven)、mvn install (將原始程式碼編譯、測試和

封裝為建置輸出成品，並在其內部儲存庫中安裝建置輸出成品)、`docker login` (使用對應於`dockerLoginPassword`您在 Amazon EC2 Systems Manager 參數存放區中設定之自訂環境變數值的密碼登入 Docker) 和數個`echo`命令。此處包含這些`echo`命令，以顯示 CodeBuild 如何執行命令及其執行順序。

- `files` 代表要上傳至組建輸出位置的檔案。在此範例中，CodeBuild 會上傳單一檔案 `messageUtil-1.0.jar`。您可以在組建環境中名為 `target` 的相對目錄中找到 `messageUtil-1.0.jar` 檔案。因為已指定 `discard-paths: yes`，系統會直接上傳 `messageUtil-1.0.jar` (並且不會上傳到中繼 `target` 目錄)。檔案名稱 (`messageUtil-1.0.jar`) 及相對目錄名稱 (`target`) 是以 Apache Maven 建立及存放組建輸出成品的方式為基礎，僅適用於此範例。在您自己的案例中，檔案名稱及目錄可能會有所不同。
- `reports` 代表在建置期間產生報告的兩個報告群組：
 - `arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1` 指定報告群組的 ARN。以測試框架產生的測試結果位於 `target/tests/reports` 目錄。檔案格式為 `JunitXml`，且路徑不會從包含測試結果的檔案移除。
 - `reportGroupCucumberJson` 指定新的報告群組。如果專案的名稱為 `my-project`，則會在組件執行時會建立名為 `my-project-reportGroupCucumberJson` 的報告群組。以測試框架產生的測試結果位於 `cucumber/target/cucumber-tests.xml`。測試檔案格式為 `CucumberJson`，且路徑不會從包含測試結果的檔案移除。

Buildspec 版本

下表列出組建規格版本與版本之間的變更。

版本	變更
0.2	<ul style="list-style-type: none"> • <code>environment_variables</code> 已重新命名為 <code>env</code>。 • <code>plaintext</code> 已重新命名為 <code>variables</code>。 • 已捨棄 <code>artifacts</code> 的 <code>type</code> 屬性。 • 在 0.1 版中，會在建置環境中預設 Shell 的個別執行個體中 AWS CodeBuild 執行每個建置命令。在 0.2 版中，CodeBuild 會在組建環境中預設 Shell 的相同執行個體中執行所有組建命令。

版本	變更
0.1	這是組建規格格式的初始定義。

批次建置 buildspec 參考

本主題包含批次建置屬性的 buildspec 參考。

批次

選用的映射。專案的批次建置設定。

批次/快速失敗

選用。指定一或多個建置任務失敗時批次建置的行為。

false

預設值。所有執行中的組建都會完成。

true

當其中一個組建任務失敗時，所有執行中的組建都會停止。

根據預設，所有批次建置任務都會使用 buildspec 檔案中指定的建置設定執行 phases，例如 env 和 。您可以在 batch/<batch-type>/buildspec 參數中指定不同的 env 值或不同的 buildspec 檔案，以覆寫預設建置設定。

batch 屬性的內容會根據指定的批次建置類型而有所不同。可能的批次建置類型包括：

- [batch/build-graph](#)
- [batch/build-list](#)
- [batch/build-matrix](#)
- [batch/build-fanout](#)

batch/build-graph

定義建置圖表。建置圖表會定義一組任務，這些任務對批次中的其他任務具有相依性。如需詳細資訊，請參閱[組建圖表](#)。

此元素包含建置任務的陣列。每個建置任務都包含下列屬性。

識別符

必要. 任務的識別符。

buildspec

選用。用於此任務之 buildspec 檔案的路徑和檔案名稱。如果未指定此參數，則會使用目前的 buildspec 檔案。

除錯工作階段

選用。布林值，指出是否針對此批次建置啟用工作階段偵錯。如需工作階段偵錯的詳細資訊，請參閱 [使用 Session Manager 偵錯組建](#)。

false

工作階段偵錯已停用。

true

工作階段偵錯已啟用。

依存項目

選用。此任務所依賴的任務識別符陣列。完成這些任務之前，不會執行此任務。

env

選用。建置環境會覆寫任務。這可以包含下列屬性：

運算類型

用於任務之運算類型的識別符。如需可能的值 [the section called “建置環境運算模式和類型”](#)，請參閱 中的 computeType。

機群

用於任務的機群識別符。如需詳細資訊，請參閱 [the section called “在預留容量機群上執行組建”](#)。

映像

用於任務的映像識別符。如需可能的值 [the section called “CodeBuild 提供的 Docker 映像”](#)，請參閱 中的映像識別符。

特權模式

布林值，指出是否要在 Docker 容器內執行 Docker 協助程式。true 只有在建置專案用於建置 Docker 映像時，才將設定為。否則，嘗試與 Docker 協助程式互動的組建會失敗。預設設定為 false。

type

用於任務的環境類型識別符。如需可能的值[the section called “建置環境運算模式和類型”](#)，請參閱 中的環境類型。

變數

將出現在建置環境中的環境變數。如需詳細資訊，請參閱[env/variables](#)。

Note

請注意，運算類型和機群無法在單一建置的相同識別符中提供。

ignore-failure

選用。布林值，指出是否可以忽略此建置任務的失敗。

false

預設值。如果此建置任務失敗，批次建置將會失敗。

true

如果此建置任務失敗，批次建置仍然可以成功。

以下是建置圖表 buildspec 項目的範例：

```
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
        ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
```

```
env:
  variables:
    BUILD_ID: build2
depend-on:
  - build1
- identifier: build3
  env:
    variables:
      BUILD_ID: build3
  depend-on:
    - build2
- identifier: build4
  env:
    compute-type: ARM_LAMBDA_1GB
- identifier: build5
  env:
    fleet: fleet_name
```

batch/build-list

定義建置清單。建置清單用於定義一些平行執行的任務。如需詳細資訊，請參閱[組建清單](#)。

此元素包含建置任務的陣列。每個建置任務都包含下列屬性。

識別符

必要。任務的識別符。

buildspec

選用。用於此任務之 buildspec 檔案的路徑和檔案名稱。如果未指定此參數，則會使用目前的 buildspec 檔案。

除錯工作階段

選用。布林值，指出是否針對此批次建置啟用工作階段偵錯。如需工作階段偵錯的詳細資訊，請參閱 [使用 Session Manager 偵錯組建](#)。

false

工作階段偵錯已停用。

true

工作階段偵錯已啟用。

env

選用。建置環境會覆寫任務。這可以包含下列屬性：

運算類型

用於任務之運算類型的識別符。如需可能的值[the section called “建置環境運算模式和類型”](#)，請參閱 `computeType`。

機群

用於任務的機群識別符。如需詳細資訊，請參閱[the section called “在預留容量機群上執行組建”](#)。

映像

用於任務的映像識別符。如需可能的值[the section called “CodeBuild 提供的 Docker 映像”](#)，請參閱 `image` 中的映像識別符。

特權模式

布林值，指出是否要在 Docker 容器內執行 Docker 協助程式。`true` 只有在建置專案用於建置 Docker 映像時，才將 `privilegedMode` 設定為 `true`。否則，嘗試與 Docker 協助程式互動的組建會失敗。預設設定為 `false`。

type

用於任務的環境類型識別符。如需可能的值[the section called “建置環境運算模式和類型”](#)，請參閱 `environmentType` 中的環境類型。

變數

將出現在建置環境中的環境變數。如需詳細資訊，請參閱[env/variables](#)。

Note

請注意，運算類型和機群無法在單一建置的相同識別符中提供。

ignore-failure

選用。布林值，指出是否可以忽略此建置任務的失敗。

false

預設值。如果此建置任務失敗，批次建置將會失敗。

`true`

如果此建置任務失敗，批次建置仍然可以成功。

以下是組建清單 `buildspec` 項目的範例：

```
batch:
  fast-fail: false
  build-list:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      ignore-failure: true
    - identifier: build3
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build4
      env:
        fleet: fleet_name
    - identifier: build5
      env:
        compute-type: GENERAL_LINUX_XLAGRE
```

batch/build-matrix

定義建置矩陣。建置矩陣會定義具有平行執行之不同組態的任務。CodeBuild 會為每個可能的組態組合建立個別的組建。如需詳細資訊，請參閱[建置矩陣](#)。

靜態

靜態屬性會套用至所有建置任務。

ignore-failure

選用。布林值，指出是否可以忽略此建置任務的失敗。

false

預設值。如果此建置任務失敗，批次建置將會失敗。

true

如果此建置任務失敗，批次建置仍然可以成功。

env

選用。建置環境會覆寫所有任務。

特權模式

布林值，指出是否要在 Docker 容器內執行 Docker 協助程式。true 只有在建置專案用於建置 Docker 映像時，才將設定為。否則，嘗試與 Docker 協助程式互動的組建會失敗。預設設定為 false。

type

用於任務的環境類型識別符。如需可能的值[the section called “建置環境運算模式和類型”](#)，請參閱 中的環境類型。

動態

動態屬性定義建置矩陣。

buildspec

選用。陣列，其中包含用於這些任務之 buildspec 檔案的路徑和檔案名稱。如果未指定此參數，則會使用目前的 buildspec 檔案。

env

選用。建置環境會覆寫這些任務。

運算類型

陣列，其中包含用於這些任務之運算類型的識別符。如需可能的值[the section called “建置環境運算模式和類型”](#)，請參閱 中的 computeType。

映像

陣列，其中包含用於這些任務的影像識別符。如需可能的值[the section called “CodeBuild 提供的 Docker 映像”](#)，請參閱 中的映像識別符。

變數

陣列，其中包含這些任務在建置環境中存在的環境變數。如需詳細資訊，請參閱[env/variables](#)。

以下是建置矩陣 `buildspec` 項目的範例：

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
            - VALUE3
```

如需詳細資訊，請參閱[建置矩陣](#)。

batch/build-fanout

定義建置散發。建置散發用於定義任務，該任務會分割成多個平行執行的建置。如需詳細資訊，請參閱[在批次建置中執行平行測試](#)。

此元素包含可分割為多個組建的組建任務。`build-fanout` 區段包含下列屬性。

平行處理

必要。將平行執行測試的組建數目。

ignore-failure

選用。布林值，指出是否可以忽略任何扇出組建任務中的失敗。此 `ignore-failure` 值將套用至所有散發組建。

false

預設值。如果任何散發建置任務失敗，批次建置將會失敗。

true

如果任何散發建置任務失敗，批次建置仍然可以成功。

以下是建置散發 buildspec 項目的範例：

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - npm install
  build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run \
          --test-command 'npx jest --runInBand --coverage' \
          --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
          --sharding-strategy 'equal-distribution'
```

如需詳細資訊，請參閱[建置散發](#)及[使用 codebuild-tests-run CLI 命令](#)。

的建置環境參考 AWS CodeBuild

當您呼叫 AWS CodeBuild 來執行組建時，您必須提供組建環境的相關資訊。建置環境代表作業系統、程式設計語言執行時間，以及 CodeBuild 用來執行建置之工具的組合。如需建置環境運作方式的詳細資訊，請參閱 [CodeBuild 的運作方式](#)。

建置環境包含 Docker 影像。如需詳細資訊，請參閱 Docker Docs 網站上的 [Docker Glossary](#)。

當您將建置環境的相關資訊提供給 CodeBuild 時，您可以在支援的儲存庫類型中指定 Docker 映像的識別符。其中包括 CodeBuild Docker 映像儲存庫、Docker Hub 中公開可用的映像，以及 AWS 您的帳戶有權存取的 Amazon Elastic Container Registry (Amazon ECR) 儲存庫。

- 我們建議您使用存放在 CodeBuild Docker 映像儲存庫中的 Docker 映像，因為它們已針對與服務搭配使用進行最佳化。如需詳細資訊，請參閱[CodeBuild 提供的 Docker 映像](#)。
- 若要取得在 Docker Hub 中存放可公開取得的 Docker 影像識別符，請參閱 Docker Docs 網站上的[搜尋儲存貯體](#)。
- 若要了解如何使用儲存在帳戶中 AWS Amazon ECR 儲存庫中的 Docker 映像，請參閱 [Amazon ECR 範例](#)。

除了 Docker 影像識別符，您還可以指定組建環境使用的一組運算資源。如需詳細資訊，請參閱[建置環境運算模式和類型](#)。

主題

- [CodeBuild 提供的 Docker 映像](#)
- [建置環境運算模式和類型](#)
- [建置環境中的 Shell 和命令](#)
- [建置環境中的環境變數](#)
- [建置環境中的背景工作](#)

CodeBuild 提供的 Docker 映像

支援的影像是 CodeBuild 中可用影像的最新主要版本，並以次要和修補程式版本更新進行更新。CodeBuild 透過在機器的 [Amazon Machine Image \(AMI\)](#) 中快取支援的影像來最佳化組建的佈建持續時間。如果您想要從快取中獲益並將建置的佈建持續時間降至最低，請在 CodeBuild 主控台的映

像版本區段中選取永遠使用此執行時間版本的最新映像，而非更精細的版本，例如 `aws/codebuild/amazonlinux-x86_64-standard:4.0-1.0.0`。

主題

- [取得目前 Docker 映像的清單](#)
- [EC2 運算映像](#)
- [Lambda 運算映像](#)
- [已棄用 CodeBuild 映像](#)
- [可用的執行時間](#)
- [執行時間版本](#)

取得目前 Docker 映像的清單

CodeBuild 會經常更新 Docker 映像清單，以新增最新映像並棄用舊映像。若要取得最新的清單，請執行下列其中一項操作：

- 在 CodeBuild 主控台的建立建置專案精靈或編輯建置專案頁面中，針對環境映像選擇受管映像。從 Operating system (作業系統)、Runtime (執行時間) 和 Runtime version (執行時間版本) 下拉式清單中，進行選擇。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 或 [變更建置專案的設定 \(主控台\)](#)。
- 針對 AWS CLI，執行 `list-curated-environment-images` 命令：

```
aws codebuild list-curated-environment-images
```

- 對於 AWS SDKs，請呼叫目標程式設計語言的 `ListCuratedEnvironmentImages` 操作。如需更多資訊，請參閱 [AWS SDKs 和工具參考](#)。

EC2 運算映像

AWS CodeBuild 支援 CodeBuild 中可用於 EC2 運算的下列 Docker 映像。

Note

Windows Server Core 2019 平台的基礎映像僅適用於下列區域：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)

- 美國西部 (奧勒岡)
- 歐洲 (愛爾蘭)

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-standard:4.0	al/standard/4.0
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-standard:5.0	al/standard/5.0
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-standard:6.0	al/standard/6.0
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-standard:corretto8	al/standard/corretto8
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-standard:corretto11	al/standard/corretto11
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0
Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0

平台	映像識別符	定義
Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
Ubuntu 24.04	aws/codebuild/standard:8.0	ubuntu/standard/8.0
Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	N/A
Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A
Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A
Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	N/A

 **Note**

2024 年 11 月 22 日，Linux 型標準執行期映像的別名已從 更新amazonlinux2為 amazonlinux。不需要手動更新，因為先前的別名仍然有效。

Lambda 運算映像

AWS CodeBuild 支援下列可在 CodeBuild 中 AWS Lambda 運算的 Docker 映像。

 **Important**

使用 Lambda 運算或預留容量時，不支援 on-failure 屬性。此屬性僅適用於 CodeBuild 提供的 EC2 運算映像。

aarch64 架構

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet6	al-lambda/aarch64/dotnet6
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet8	al-lambda/aarch64/dotnet8
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet10	al-lambda/aarch64/dotnet10
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.21	al-lambda/aarch64/go1.21
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.24	al-lambda/aarch64/go1.24
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.25	al-lambda/aarch64/go1.25
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto11	al-lambda/aarch64/corretto11

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto17	al-lambda/aarch64/corretto17
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto21	al-lambda/aarch64/corretto21
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto25	al-lambda/aarch64/corretto25
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs18	al-lambda/aarch64/nodejs18
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs20	al-lambda/aarch64/nodejs20
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs22	al-lambda/aarch64/nodejs22
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs24	al-lambda/aarch64/nodejs24

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.11	al-lambda/aarch64/python3.11
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.12	al-lambda/aarch64/python3.12
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.13	al-lambda/aarch64/python3.13
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.14	al-lambda/aarch64/python3.14
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.2	al-lambda/aarch64/ruby3.2
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.4	al-lambda/aarch64/ruby3.4

x86_64 架構

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet6	al-lambda/x86_64/dotnet6

平台	映像識別符	定義
	bda-standard:dotnet6	
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet8	al-lambda/x86_64/dotnet8
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet10	al-lambda/x86_64/dotnet10
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.21	al-lambda/x86_64/go1.21
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.24	al-lambda/x86_64/go1.24
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.25	al-lambda/x86_64/go1.25
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto11	al-lambda/x86_64/corretto11
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto17	al-lambda/x86_64/corretto17

平台	映像識別符	定義
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21	al-lambda/x86_64/corretto21
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto25	al-lambda/x86_64/corretto25
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs18	al-lambda/x86_64/nodejs18
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20	al-lambda/x86_64/nodejs20
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs22	al-lambda/x86_64/nodejs22
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs24	al-lambda/x86_64/nodejs24
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.11	al-lambda/x86_64/python3.11

平台	映像識別符	定義
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12	al-lambda/x86_64/python3.12
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.13	al-lambda/x86_64/python3.13
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.14	al-lambda/x86_64/python3.14
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.2	al-lambda/x86_64/ruby3.2
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.4	al-lambda/x86_64/ruby3.4

已棄用 CodeBuild 映像

已棄用的影像是 CodeBuild 不再快取或更新的映像。已棄用的影像不會再收到次要版本更新或修補程式版本更新，而且由於不再更新，因此使用它們可能不安全。如果您的 CodeBuild 專案設定為使用較舊的映像版本，佈建程序會下載此 docker 映像，並使用它來建立容器化執行期環境，這可能會增加佈建持續時間和整體建置持續時間。

CodeBuild 已棄用下列 Docker 映像。您仍然可以使用這些映像，但這些映像不會快取在建置主機上，而且會導致較高的佈建時間。

平台	映像識別符	定義	取代日期
Amazon Linux 2	aws/codebuild/ amazonlinux2- x86_64-st andard:3.0	al2/standard/3.0	2023 年 5 月 9 日
Ubuntu 18.04	aws/codebuild/ standard:4.0	ubuntu/standard/4.0	2023 年 3 月 31 日
Amazon Linux 2	aws/codebuild/ amazonlinux2- aarch64-s tandard:1.0	al2/aarch64/standa rd/1.0	2023 年 3 月 31 日
Ubuntu 18.04	aws/codebuild/ standard:3.0	ubuntu/standard/3.0	2022 年 6 月 30 日
Amazon Linux 2	aws/codebuild/ amazonlinux2- x86_64-st andard:2.0	al2/standard/2.0	2022 年 6 月 30 日

主題

- [可用的執行時間](#)
- [執行時間版本](#)

可用的執行時間

您可以在 buildspec 檔案的 runtime-versions 區段中指定一或多個執行時間。如果您的執行階段依存於另一個執行時間，您也可以在建構 spec 檔案中指定其相依的執行時間。如果您未在 buildspec 檔案中指定任何執行時間，CodeBuild 會選擇您所使用的映像中可用的預設執行時間。如果您指定一或多個執行時間，CodeBuild 只會使用這些執行時間。如果未指定相依執行時間，CodeBuild 會嘗試為您選擇相依執行時間。

 Note

如果未指定執行時間版本，CodeBuild 會使用預設版本。當先前的預設版本達到生命週期結束 (EOL) 時，預設版本可能會變更。為避免意外變更建置環境，建議您在 buildspec 檔案中指定執行時間版本。

如需詳細資訊，請參閱[Specify runtime versions in the buildspec file](#)。

主題

- [Linux 映像執行時間](#)
- [macOS 映像執行時間](#)
- [Windows 映像執行時間](#)

Linux 映像執行時間

下表包含可用的執行時間，以及支援這些執行時間的標準 Linux 映像。

Ubuntu 和 Amazon Linux 平台執行時間

執行時間名稱	版本	映像
dotnet	6.0	Amazon Linux 2 x86_64 Lambda 標準：dotnet6
		Amazon Linux 2 AArch64 Lambda 標準：dotnet6
		Amazon Linux 2023 x86_64 標準：5.0
		Amazon Linux 2023 AArch64 標準：3.0
	8.0	Ubuntu 22.04 標準：7.0
		Amazon Linux 2023 x86_64 標準：5.0

執行時間名稱	版本	映像
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
	10.0	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
golang	1.20	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0

執行時間名稱	版本	映像
	1.21	Amazon Linux 2 x86_64 Lambda 標準 : go1.21
		Amazon Linux 2 AArch64 Lambda 標準 : go1.21
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0
	1.22	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0
	1.23	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0

執行時間名稱	版本	映像
	1.24	Amazon Linux 2023 x86_64 Lambda 標準 : go1.24
		Amazon Linux 2023 AArch64 Lambda 標準 : go1.24
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
	1.25	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	1.26	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
java	corretto8	Amazon Linux 2 x86_64 標準 : corretto8
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	corretto11	Amazon Linux 2 x86_64 標準 : corretto11
		Amazon Linux 2 x86_64 Lambda 標準 : corretto11
		Amazon Linux 2 AArch64 Lambda 標準 : corretto11
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	corretto17	Amazon Linux 2 x86_64 Lambda 標準 : corretto17
		Amazon Linux 2 AArch64 Lambda 標準 : corretto17
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	corretto21	Amazon Linux 2 x86_64 Lambda 標準 : corretto21
		Amazon Linux 2 AArch64 Lambda 標準 : corretto21
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
	corretto25	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
nodejs	18	Amazon Linux 2 x86_64 Lambda 標準 : nodejs18
		Amazon Linux 2 AArch64 Lambda 標準 : nodejs18
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0
	20	Amazon Linux 2 x86_64 Lambda 標準 : nodejs20
		Amazon Linux 2 AArch64 Lambda 標準 : nodejs20
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0

執行時間名稱	版本	映像
	22	Amazon Linux 2023 x86_64 Lambda 標準 : nodejs22
		Amazon Linux 2023 AArch64 Lambda 標準 : nodejs22
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
	24	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	26	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
php	8.1	Amazon Linux 2023 AArch64 標準 : 3.0
	8.2	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	8.3	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
	8.4	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	8.5	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
python	3.9	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0
	3.10	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	3.11	Amazon Linux 2 x86_64 Lambda 標準 : python3.11
		Amazon Linux 2 AArch64 Lambda 標準 : python3.11
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	3.12	Amazon Linux 2 x86_64 Lambda 標準 : python3.12
		Amazon Linux 2 AArch64 Lambda 標準 : python3.12
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	3.13	Amazon Linux 2023 x86_64 Lambda 標準 : python3.13
		Amazon Linux 2023 AArch64 Lambda 標準 : python3.13
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
	3.14	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
ruby	3.1	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0
	3.2	Amazon Linux 2 x86_64 Lambda 標準 : ruby3.2
		Amazon Linux 2 AArch64 Lambda 標準 : ruby3.2
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Ubuntu 22.04 標準 : 7.0
	3.3	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
	3.4	Amazon Linux 2023 x86_64 Lambda 標準 : ruby3.4
		Amazon Linux 2023 AArch64 Lambda 標準 : ruby3.4
		Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0
	4.0	Amazon Linux 2023 x86_64 標準 : 5.0
		Amazon Linux 2023 x86_64 標準 : 6.0
		Amazon Linux 2023 AArch64 標準 : 3.0
		Amazon Linux 2023 AArch64 標準 : 4.0
		Ubuntu 22.04 標準 : 7.0
		Ubuntu 24.04 標準 : 8.0

執行時間名稱	版本	映像
生鏽	1.94	Amazon Linux 2023 AArch64 標準：3.0
		Amazon Linux 2023 AArch64 標準：4.0
	1.97	Amazon Linux 2023 x86_64 標準：6.0
		Ubuntu 24.04 標準：8.0

macOS 映像執行時間

 Important

Mac 組建的 CodeBuild 精選映像包含預先安裝的 macOS 和 Xcode。使用 Xcode 軟體，即表示您認可、了解和同意 [Xcode 和 Apple SDKs協議](#)。如果您不接受協議的條款與條件，請勿使用 Xcode 軟體。反之，請提供您自己的 Amazon Machine Image (AMI)。如需詳細資訊，請參閱[如何設定預留容量 macOS 機群？](#)

下表包含 macOS 支援的可用執行時間。

macOS 平台執行時間

執行時間名稱	版本	映像	其他備註
bash	3.2.57	macos-arm-base：15	
		macos-arm-base：26	
clang	17.0.0	macos-arm-base：15	
		macos-arm-base：26	
dotnet sdk	8.0.416	macos-arm-base：15	
	8.0.417	macos-arm-base：26	

執行時間名稱	版本	映像	其他備註
gcc	10.0.101	macos-arm-base : 15	
	10.0.102	macos-arm-base : 26	
	11.5.0	macos-arm-base : 15 macos-arm-base : 26	使用gcc-11別名提供
	12.4.0	macos-arm-base : 15 macos-arm-base : 26	使用gcc-12別名提供
	13.4.0	macos-arm-base : 15 macos-arm-base : 26	使用gcc-13別名提供
	14.3.0	macos-arm-base : 15 macos-arm-base : 26	使用gcc-14別名提供
gnu	11.5.0	macos-arm-base : 15 macos-arm-base : 26	使用gfortran-11 別名提供
	12.4.0	macos-arm-base : 15 macos-arm-base : 26	使用gfortran-12 別名提供
	13.4.0	macos-arm-base : 15 macos-arm-base : 26	使用gfortran-13 別名提供
	14.3.0	macos-arm-base : 15 macos-arm-base : 26	使用gfortran-14 別名提供
golang	1.24.11	macos-arm-base : 15 macos-arm-base : 26	

執行時間名稱	版本	映像	其他備註
java	1.25.4	macos-arm-base : 15 macos-arm-base : 26	
	Corretto8	macos-arm-base : 15 macos-arm-base : 26	
	Corretto11	macos-arm-base : 15 macos-arm-base : 26	
	Corretto17	macos-arm-base : 15 macos-arm-base : 26	
	Corretto21	macos-arm-base : 15 macos-arm-base : 26	
	Corretto25	macos-arm-base : 15 macos-arm-base : 26	
	2.2.21	macos-arm-base : 15	
	2.3.0	macos-arm-base : 26	
	6.14.1	macos-arm-base : 15 macos-arm-base : 26	
	20.19.6	macos-arm-base : 15 macos-arm-base : 26	
nodejs	22.21.1	macos-arm-base : 15 macos-arm-base : 26	

執行時間名稱	版本	映像	其他備註
	24.11.1	macos-arm-base : 15	
		macos-arm-base : 26	
perl	5.34.1	macos-arm-base : 15	
		macos-arm-base : 26	
php	8.2.29	macos-arm-base : 15	
	8.2.30	macos-arm-base : 26	
	8.3.28	macos-arm-base : 15	
	8.3.30	macos-arm-base : 26	
	8.4.15	macos-arm-base : 15	
	8.4.17	macos-arm-base : 26	
	8.5.0	macos-arm-base : 15	
	8.5.2	macos-arm-base : 26	
	3.10.19	macos-arm-base : 15	
		macos-arm-base : 26	
python	3.11.14	macos-arm-base : 15	
		macos-arm-base : 26	
	3.12.12	macos-arm-base : 15	
		macos-arm-base : 26	
	3.13.10	macos-arm-base : 15	
		macos-arm-base : 26	

執行時間名稱	版本	映像	其他備註
ruby	3.14.1	macos-arm-base : 15	
		macos-arm-base : 26	
	3.2.9	macos-arm-base : 15	
		macos-arm-base : 26	
	3.3.10	macos-arm-base : 15	
		macos-arm-base : 26	
生鏽	3.4.7	macos-arm-base : 15	
		macos-arm-base : 26	
	1.91.1	macos-arm-base : 15	
		macos-arm-base : 26	
	1.92.0	macos-arm-base : 15	
		macos-arm-base : 26	
Swift	6.2.1	macos-arm-base : 15	
	6.2.3	macos-arm-base : 26	
Xcode	26.1.1	macos-arm-base : 15	
	26.2	macos-arm-base : 26	

Windows 映像執行時間

Windows Server Core 2019 的基礎映像包含下列執行時間。

Windows 平台執行時間

執行時間名稱	Windows Server Core 2019 標準 : 1.0 版	Windows Server Core 2019 標準 : 2.0 版	Windows Server Core 2019 標準 : 3.0 版本
dotnet	3.1	3.1	8.0
	5.0	6.0	

執行時間名稱	Windows Server Core 2019 標準 : 1.0 版	Windows Server Core 2019 標準 : 2.0 版	Windows Server Core 2019 標準 : 3.0 版本
		7.0	
dotnet sdk	3.1 5.0	3.1 6.0 7.0	8.0
golang	1.14	1.18	1.21 1.22 1.23
gradle	6.7	7.6	8.12
java	Corretto11	Corretto11 Corretto17	Corretto8 Corretto11 Corretto17 Corretto21
maven	3.6	3.8	3.9
nodejs	14.15	16.19	20.18 22.13
php	7.4	8.1	8.3 8.4
powershell	7.1	7.2	7.4

執行時間名稱	Windows Server Core 2019 標準：1.0 版	Windows Server Core 2019 標準：2.0 版	Windows Server Core 2019 標準：3.0 版本
python	3.8	3.10	3.10 3.11 3.12 3.13
ruby	2.7	3.1	3.2 3.3 3.4

執行時間版本

當您在 Buildspec 檔案的 [runtime-versions](#) 區段中指定執行時間時，您可以指定特定版本、特定主要版本和最新次要版本，或最新版本。下表列出可用的執行時間及其指定方法。並非所有執行時間版本都適用於所有映像。自訂映像也不支援執行期版本選擇。如需詳細資訊，請參閱[可用的執行時間](#)。如果您想要安裝並使用自訂執行期版本，而非預先安裝的執行期版本，請參閱[自訂執行時間版本](#)。

Ubuntu 和 Amazon Linux 2 平台執行時間版本

執行時間名稱	版本	特定版本	特定主要和最新次要版本	最新版本
dotnet	6.0	dotnet: 6.0	dotnet: 6.x	dotnet: latest
	8.0	dotnet: 8.0	dotnet: 8.x	
	10.0	dotnet: 10.0	dotnet: 10.x	
golang	1.20	golang: 1.20	golang: 1.x	golang: latest
	1.21	golang: 1.21		
	1.22	golang: 1.22		

執行時間名稱	版本	特定版本	特定主要和最新次要版本	最新版本
	1.23	golang: 1.23		
	1.24	golang: 1.24		
	1.25	golang: 1.25		
	1.26	golang: 1.26		
java	corretto8	java: corretto	java: corretto .x	java: latest
	corretto11	java: corretto 1	java: corretto 1.x	
	corretto17	java: corretto 7	java: corretto 7.x	
	corretto21	java: corretto 1	java: corretto 1.x	
	corretto25	java: corretto 5	java: corretto 5.x	
nodejs	18	nodejs: 18	nodejs: 18.x	nodejs: latest
	20	nodejs: 20	nodejs: 20.x	
	22	nodejs: 22	nodejs: 22.x	
	24	nodejs: 24	nodejs: 24.x	
php	8.1	php: 8.1	php: 8.x	php: latest
	8.2	php: 8.2		
	8.3	php: 8.3		
	8.4	php: 8.4		

執行時間名稱	版本	特定版本	特定主要和最新次要版本	最新版本
python	8.5	php: 8.5	python: 3.x	python: latest
	3.9	python: 3.9		
	3.10	python: 3.10		
	3.11	python: 3.11		
	3.12	python: 3.12		
	3.13	python: 3.13		
ruby	3.14	python: 3.14	ruby: 3.x	ruby: latest
	3.1	ruby: 3.1		
	3.2	ruby: 3.2		
	3.3	ruby: 3.3		
	3.4	ruby: 3.4		
	4.0	ruby: 4.0		
生鏽	1.94	rust: 1.94	rust: 1.x	rust: latest

您可以使用建置規格，在install建置階段安裝其他元件（例如，AWS CLI、Apache Maven、Apache Ant、Mocha、RSpec 等）。如需詳細資訊，請參閱[Buildspec 範例](#)。

自訂執行時間版本

您可以安裝和使用您選擇的自訂版本，而不是在 CodeBuild 受管映像中使用預先安裝的執行期版本。下表列出可用的自訂執行期，以及如何指定它們。

Note

只有 Ubuntu 和 Amazon Linux 映像才支援自訂執行時間版本選擇。

自訂執行時間版本

執行時間名稱	語法	範例
dotnet	<code><major>.<minor>.<patch></code>	5.0.408
golang	<code><major>.<minor></code>	1.19
	<code><major>.<minor>.<patch></code>	1.19.1
java	corretto <code><major></code>	corretto15
nodejs	<code><major></code>	14
	<code><major>.<minor></code>	14.21
	<code><major>.<minor>.<patch></code>	14.21.3
php	<code><major>.<minor>.<patch></code>	8.0.30
python	<code><major></code>	3
	<code><major>.<minor></code>	3.7
	<code><major>.<minor>.<patch></code>	3.7.16
ruby	<code><major>.<minor>.<patch></code>	3.0.6

自訂執行期 buildspec 範例

以下是指定自訂執行時間版本的 buildspec 範例。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: corretto15
      php: 8.0.30
      ruby: 3.0.6
      golang: 1.19
      python: 3.7
      nodejs: 14
```

`dotnet: 5.0.408`

建置環境運算模式和類型

在 CodeBuild 中，您可以指定 CodeBuild 用來執行建置的運算和執行期環境映像。運算是指由 CodeBuild 管理和維護的運算引擎 (CPU、記憶體和作業系統)。執行時間環境映像是在您選擇的運算平台上執行的容器映像，並包含建置可能需要的額外工具，例如 AWS CLI。

主題

- [關於運算](#)
- [關於預留容量環境類型](#)
- [關於隨需環境類型](#)

關於運算

CodeBuild 提供 EC2 和 AWS Lambda 運算模式。EC2 在建置期間提供最佳化的彈性，並提供 AWS Lambda 最佳化的啟動速度。AWS Lambda 支援更快的建置，因為啟動延遲較低。AWS Lambda 也會自動擴展，因此建置不會在佇列中等待執行。如需詳細資訊，請參閱[在 AWS Lambda 運算上執行組建](#)。

使用 EC2 運算模式，您可以使用隨需或預留容量機群來執行建置。對於隨需機群，您可以選取預先定義的運算類型，例如 BUILD_GENERAL1_SMALL 或 BUILD_GENERAL1_LARGE。如需詳細資訊，請參閱[關於隨需環境類型](#)。對於預留容量機群，您可以選擇運算組態，包括 vCPU、記憶體和磁碟空間。指定組態後，CodeBuild 會選擇符合您需求的支援運算類型。如需詳細資訊，請參閱[關於預留容量環境類型](#)。

關於預留容量環境類型

AWS CodeBuild 為預留容量機群提供 Linux x86、Arm、GPU、Windows 和 macOS 環境類型。下表顯示依區域排序的可用機器類型、記憶體、vCPUs 和磁碟空間：

US East (N. Virginia)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu. .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu. .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu. .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gib .nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
Linux GPU	64	256 GiB	1885 GB (SSD)	NVME	reserved.gpu.64cpu.256gib.nvme
Linux GPU	96	384 GiB	3785 GB (SSD)	NVME	reserved.gpu.96cpu.384gib.nvme

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
macOS	8	24 GiB	128 GB	GENERAL	reserved. arm.m2.8c pu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved. arm.m2.12 cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

如需定價識別符的詳細資訊，請參閱 [https : //https://aws.amazon.com/codebuild/pricing/](https://aws.amazon.com/codebuild/pricing/)。

US East (Ohio)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu. .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu. .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu. .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
macOS	8	24 GiB	128 GB	GENERAL	reserved.arm.m2.8cpu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gi b

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

US West (N. California)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu. .64gib
ARM	48	96 GiB	824 GB	GENERAL	reserved. arm.48cpu. .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Linux	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Linux	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Linux	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

US West (Oregon)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux GPU	64	256 GiB	1885 GB (SSD)	NVME	reserved.gpu.64cpu.256gib.nvme
macOS	8	24 GiB	128 GB	GENERAL	reserved.arm.m2.8cpu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved.x86-64.32cpu.64gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Asia Pacific (Tokyo)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu. .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu. .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu. .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gib .nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Asia Pacific (Malaysia)

環境類型	vCPUs	記憶體	機器類型	運算執行個體類型
ARM	2	4 GiB	GENERAL	reserved: arm.g1.sm all
ARM	4	8 GiB	GENERAL	reserved: arm.g1.me dium
ARM	8	16 GiB	GENERAL	reserved: arm.g1.la rge
ARM	32	64 GiB	GENERAL	reserved: arm.g1.xl arge
ARM	48	96 GiB	GENERAL	reserved: arm.g1.2x large
Linux	2	4 GiB	GENERAL	reserved: x86-64.g1 .small

環境類型	vCPUs	記憶體	機器類型	運算執行個體類型
Linux	4	8 GiB	GENERAL	reserved: x86-64.g1 .medium
Linux	8	16 GiB	GENERAL	reserved: x86-64.g1 .large

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Asia Pacific (New Zealand)

環境類型	vCPUs	記憶體	機器類型	運算執行個體類型
ARM	2	4 GiB	GENERAL	reserved: arm.g1.sm all
ARM	4	8 GiB	GENERAL	reserved: arm.g1.me dium
ARM	8	16 GiB	GENERAL	reserved: arm.g1.la rge
ARM	32	64 GiB	GENERAL	reserved: arm.g1.xl arge
ARM	48	96 GiB	GENERAL	reserved: arm.g1.2x large

環境類型	vCPUs	記憶體	機器類型	運算執行個體類型
Linux	2	4 GiB	GENERAL	reserved: x86-64.g1 .small
Linux	4	8 GiB	GENERAL	reserved: x86-64.g1 .medium
Linux	8	16 GiB	GENERAL	reserved: x86-64.g1 .large

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Asia Pacific (Thailand)

環境類型	vCPUs	記憶體	機器類型	運算執行個體類型
ARM	2	4 GiB	GENERAL	reserved: arm.g1.sm all
ARM	4	8 GiB	GENERAL	reserved: arm.g1.me dium
ARM	8	16 GiB	GENERAL	reserved: arm.g1.la rge
ARM	32	64 GiB	GENERAL	reserved: arm.g1.xl arge

環境類型	vCPUs	記憶體	機器類型	運算執行個體類型
ARM	48	96 GiB	GENERAL	reserved: arm.g1.2x large
Linux	2	4 GiB	GENERAL	reserved: x86-64.g1 .small
Linux	4	8 GiB	GENERAL	reserved: x86-64.g1 .medium
Linux	8	16 GiB	GENERAL	reserved: x86-64.g1 .large

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Asia Pacific (Mumbai)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu. .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu. .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu. .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gi b
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Asia Pacific (Singapore)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gib .nvme

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Asia Pacific (Sydney)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu. .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu. .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu. .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gib .nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
macOS	8	24 GiB	128 GB	GENERAL	reserved.arm.m2.8cpu.24gib
macOS	12	32 GiB	256 GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Europe (Frankfurt)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved. arm.16cpu. .32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved. arm.32cpu. .64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved. arm.48cpu. .96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
macOS	8	24 GiB	128 GB	GENERAL	reserved.arm.m2.8cpu.24gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

Europe (Ireland)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	64	128 GiB	824 GB	GENERAL	reserved. arm.64cpu. .128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved. arm.2cpu. 4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved. arm.4cpu. 8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved. arm.8cpu. 16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved. x86-64.48 cpu.96gib .nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved. x86-64.72 cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	2	4 GiB	64 GB	GENERAL	reserved. x86-64.2c pu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	64	128 GiB	512 GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

South America (São Paulo)

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	4	8 GiB	128 GB	GENERAL	reserved. x86-64.4c pu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved. x86-64.8c pu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gi b
Linux	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gi b

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows	16	32 GiB	256 GB	GENERAL	reserved. x86-64.16 cpu.32gib
Windows	32	64 GiB	256 GB	GENERAL	reserved. x86-64.32 cpu.64gib
Windows	36	72 GiB	256 GB	GENERAL	reserved. x86-64.36 cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved. x86-64.48 cpu.96gib
Windows	64	128 GiB	512 GB	GENERAL	reserved. x86-64.64 cpu.128gib
Windows	72	144 GiB	824 GB	GENERAL	reserved. x86-64.72 cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved. x86-64.96 cpu.192gib

環境類型	vCPUs	記憶體	磁碟空間	機器類型	運算執行個體類型
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

如需定價識別符的詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

若要選擇運算類型：

- 在 CodeBuild 主控台的運算機群組態頁面中，從 vCPUs、記憶體和磁碟中選擇其中一個選項。如需詳細資訊，請參閱[建立預留容量機群](#)。
- 對於 AWS CLI，執行 `create-fleet` 或 `update-fleet` 命令，將 `computeType` 的值指定為 `ATTRIBUTE_BASED_COMPUTE`。如需詳細資訊，請參閱 [create-fleet](#) 或 [update-fleet](#)。
- 對於 AWS SDKs，請為目標程式設計語言呼叫相當於 `CreateFleet` 或 `UpdateFleet` 操作的，並將 `computeType` 指定為 `ATTRIBUTE_BASED_COMPUTE`。如需更多資訊，請參閱[AWS SDKs 和工具參考](#)。

Note

對於 AWS CLI AWS SDKs，您仍然可以使用 `computeType` 輸入 `BUILD_GENERAL1_SMALL` 來選擇運算類型，而不是 `ATTRIBUTE_BASED_COMPUTE`。如需詳細資訊，請參閱[關於隨需環境類型](#)。

支援的執行個體系列

AWS CodeBuild 支援預留容量機群的下列執行個體：

- 一般用途：M5 | M5a | M5ad | M5d | M5dn | M5n | M5zn | M6a | M6g | M6gd | M6i | M6id | M6idn | M6in | M7a | M7g | M7gd | M7i | M7i-flex | M8g | T3 | T3a | T4g

- 運算最佳化：C5 | C5a | C5ad | C5d | C5n | C6a | C6g | C6gd | C6gn | C6i | C6id | C6in | C7a | C7g | C7gd | C7gn | C7i | C7i-flex | C8g
- 記憶體最佳化：R5 | R5a | R5ad | R5b | R5d | R5dn | R5n | R6a | R6g | R6gd | R6i | R6idn | R6in | R6id | R7a | R7g | R7gd | R7i | R7iz | R8g | U-3tb1 | U-6tb1 | U-9tb1 | U-12tb1 | U-18tb1 | U-24tb1 | U7i-6tb | U7i-8tb | U7i-12tb | U7in-16tb | U7in-16tb | U7in-24tb | U7in-32tb | X1X1e2 | X2gd2 | X2idn | X2iedn | X2iezn | X8g
- 儲存最佳化：D3 | D3en | I3 | I3en | I4g | I4i | I7ie | I8g | Im4gn | Is4gen
- 加速運算：DL1 | DL2q | F1 | F2 | G4ad | G4dn | G5 | G5g | G6 | G6e | Gr6 | Inf1 | Inf2 | P3 | P3dn | P4d | P5 | P5e | P5en | Trn1 | Trn1n | Trn2 | VT1
- 高效能運算：Hpc6a | Hpc6id | Hpc7a | Hpc7g
- 上一代：A1

若要建立具有特定執行個體類型的預留容量機群：

- 在 CodeBuild 主控台的運算機群組態頁面中，導覽至容量組態區段。在運算選取模式中，選取手動輸入，然後在運算執行個體類型中選擇其中一個執行個體類型。如需詳細資訊，請參閱[建立預留容量機群](#)。
- 對於 AWS CLI，請執行 `create-fleet` 或 `update-fleet` 命令，將 `computeType` 指定為 `CUSTOM_INSTANCE_TYPE`，將 `ComputeConfiguration.instanceType` 指定為指定的執行個體類型。如需詳細資訊，請參閱[create-fleet](#) 或 [update-fleet](#)。
- 對於 AWS SDKs，請為目標程式設計語言呼叫相當於 `CreateFleet` 或 `UpdateFleet` 操作的，將 `computeType` 指定為 `CUSTOM_INSTANCE_TYPE`，將 `ComputeConfiguration.instanceType` 指定為指定的執行個體類型。如需更多資訊，請參閱[AWS SDKs和工具參考](#)。

關於隨需環境類型

AWS CodeBuild 為 EC2 運算模式提供具有下列可用記憶體、vCPUs 和磁碟空間的建置環境：

運算類型	Environment computeType 值	環境類型值	記憶體	vCPUs	磁碟空間
ARM Small 1	BUILD_GENERAL1_SMALL	ARM_CONTAINER ARM_EC2	4 GiB	2	64 GB
ARM Medium 1	BUILD_GENERAL1_MEDIUM	ARM_CONTAINER ARM_EC2	8 GiB	4	128 GB
ARM 大型 1	BUILD_GENERAL1_LARGE	ARM_CONTAINER ARM_EC2	16 GiB	8	128 GB
ARM XLarge 1	BUILD_GENERAL1_XLARGE	ARM_CONTAINER	64 GiB	32	256 GB
ARM 2XLarge 1	BUILD_GENERAL1_2XLARGE	ARM_CONTAINER	96 GiB	48	824 GB
Linux 小型 1	BUILD_GENERAL1_SMALL	LINUX_CONTAINER LINUX_EC2	4 GiB	2	64 GB
Linux Medium 1	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER LINUX_EC2	8 GiB	4	128 GB
Linux 大型 1	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	16 GiB	8	128 GB

運算類型	Environment computeType 值	環境類型值	記憶體	vCPUs	磁碟空間
		LINUX_EC2			
Linux XLarge 1	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	72 GiB	36	256 GB
Linux 2XLarge	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	144 GiB	72	824 GB (SSD)
Linux GPU 小型	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GiB	4	235 GB (SSD)
Linux GPU 大型	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GiB	32	50 GB
Windows Medium 1	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER WINDOWS_SERVER_2022_CONTAINER WINDOWS_EC2	8 GiB	4	128 GB

運算類型	Environment computeType 值	環境類型值	記憶體	vCPUs	磁碟空間
Windows 大型 1	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER	16 GiB	8	128 GB
		WINDOWS_SERVER_2022_CONTAINER			
		WINDOWS_EC2			
Windows XLarg 1	BUILD_GENERAL1_XLARGE	WINDOWS_SERVER_2022_CONTAINER	72 GiB	36	256 GB
Windows 2XLarg 1	BUILD_GENERAL1_2XLARGE	WINDOWS_SERVER_2022_CONTAINER	144 GiB	72	824 GB

¹ 此映像類型的最新版本會快取。如果您指定更特定的版本，則 CodeBuild 會佈建該版本，而不是快取的版本。這可能會導致建置時間更長。例如，若要從快取中受益，請指定 `aws/codebuild/amazonlinux-x86_64-standard:5.0`，而非更精細的版本，例如 `aws/codebuild/amazonlinux-x86_64-standard:5.0-1.0.0`。

AWS CodeBuild 針對 AWS Lambda 運算模式提供具有下列可用記憶體和磁碟空間的 建置環境：

運算類型	Environment computeType 值	環境類型值	記憶體	磁碟空間
ARM Lambda 1GB	BUILD_LAMBDA_1GB	ARM_LAMBDA_CONTAINER	1 GiB	10 GB
ARM Lambda 2GB	BUILD_LAMBDA_2GB	ARM_LAMBDA_CONTAINER	2 GiB	10 GB
ARM Lambda 4GB	BUILD_LAMBDA_4GB	ARM_LAMBDA_CONTAINER	4 GiB	10 GB
ARM Lambda 8GB	BUILD_LAMBDA_8GB	ARM_LAMBDA_CONTAINER	8 GiB	10 GB
ARM Lambda 10GB	BUILD_LAMBDA_10GB	ARM_LAMBDA_CONTAINER	10 GiB	10 GB
Linux Lambda 1GB	BUILD_LAMBDA_1GB	LINUX_LAMBDA_CONTAINER	1 GiB	10 GB
Linux Lambda 2GB	BUILD_LAMBDA_2GB	LINUX_LAMBDA_CONTAINER	2 GiB	10 GB
Linux Lambda 4GB	BUILD_LAMBDA_4GB	LINUX_LAMBDA_CONTAINER	4 GiB	10 GB
Linux Lambda 8GB	BUILD_LAMBDA_8GB	LINUX_LAMBDA_CONTAINER	8 GiB	10 GB

運算類型	Environment computeType 值	環境類型值	記憶體	磁碟空間
Linux Lambda 10G	BUILD_LAMBDA_10GB	LINUX_LAMBDA_CONTAINER	10 GiB	10 GB

使用其他環境類型時，建議您使用快取的映像來縮短建置時間。

每個建置環境列出的磁碟空間只可在 CODEBUILD_SRC_DIR 環境變數指定的目錄中使用。

若要選擇運算類型：

- 在 CodeBuild 主控台的建立建置專案精靈或編輯建置專案頁面中，在環境展開其他組態，然後從運算類型中選擇其中一個選項。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 或 [變更建置專案的設定 \(主控台\)](#)。
- 對於 AWS CLI，執行 create-project 或 update-project 命令，指定 environment 物件 computeType 的值。如需詳細資訊，請參閱 [建立建置專案 \(AWS CLI\)](#) 或 [變更建置專案的設定 \(AWS CLI\)](#)。
- 對於 AWS SDKs，請為目標程式設計語言呼叫對等的 CreateProject 或 UpdateProject 操作，並指定對等的 environment 物件 computeType 值。如需更多資訊，請參閱 [AWS SDKs 和工具參考](#)。

有些環境和運算類型具有區域可用性限制：

- 運算類型 Linux GPU Small (LINUX_GPU_CONTAINER) 僅適用於下列區域：
 - 美國東部 (維吉尼亞北部)
 - 美國西部 (奧勒岡)
 - 亞太地區 (東京)
 - 加拿大 (中部)
 - 歐洲 (法蘭克福)
 - 歐洲 (愛爾蘭)
 - 歐洲 (倫敦)
- 運算類型 Linux GPU Large (LINUX_GPU_CONTAINER) 僅適用於下列區域：
 - 美國東部 (俄亥俄)

- 美國東部 (維吉尼亞北部)
- 美國西部 (奧勒岡)
- 亞太地區 (首爾)
- 亞太地區 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 運算類型BUILD_GENERAL1_2XLARGE僅適用於這些區域：
 - 美國東部 (俄亥俄)
 - 美國東部 (維吉尼亞北部)
 - 美國西部 (加利佛尼亞北部)
 - 美國西部 (奧勒岡)
 - 亞太地區 (海德拉巴)
 - 亞太地區 (香港)
 - 亞太地區 (雅加達)
 - 亞太地區 (墨爾本)
 - 亞太地區 (孟買)
 - 亞太區域 (首爾)
 - 亞太區域 (新加坡)
 - 亞太地區 (雪梨)
 - 亞太區域 (東京)
 - 加拿大 (中部)
 - 中國 (北京)
 - 中國 (寧夏)
 - 歐洲 (法蘭克福)
 - 歐洲 (愛爾蘭)

- 歐洲 (倫敦)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- 環境類型ARM_CONTAINER僅適用於這些區域：
 - 美國東部 (俄亥俄)
 - 美國東部 (維吉尼亞北部)
 - 美國西部 (加利佛尼亞北部)
 - 美國西部 (奧勒岡)
 - 亞太地區 (香港)
 - 亞太地區 (雅加達)
 - 亞太地區 (海德拉巴)
 - 亞太區域 (馬來西亞)
 - 亞太地區 (孟買)
 - 亞太區域 (紐西蘭)
 - 亞太地區 (大阪)
 - 亞太區域 (首爾)
 - 亞太區域 (新加坡)
 - 亞太地區 (雪梨)
 - 亞太區域 (泰國)
 - 亞太地區 (東京)
 - 加拿大 (中部)
 - 中國 (北京)
 - 中國 (寧夏)
 - 歐洲 (法蘭克福)

- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- 環境類型WINDOWS_SERVER_2022_CONTAINER僅適用於這些區域：
 - 美國東部 (俄亥俄)
 - 美國東部 (維吉尼亞北部)
 - 美國西部 (奧勒岡)
 - 亞太地區 (雪梨)
 - 亞太地區 (東京)
 - 歐洲 (法蘭克福)
 - 歐洲 (愛爾蘭)
 - 南美洲 (聖保羅)
- 環境類型
LINUX_EC2(BUILD_GENERAL1_SMALL、BUILD_GENERAL1_MEDIUM、BUILD_GENERAL1_LARGE)
僅適用於下列區域：
 - 美國東部 (俄亥俄)
 - 美國東部 (維吉尼亞北部)
 - 美國西部 (加利佛尼亞北部)
 - 美國西部 (奧勒岡)
 - 非洲 (開普敦)
 - 亞太地區 (香港)
 - 亞太地區 (雅加達)
 - 亞太地區 (墨爾本)
 - 亞太地區 (海德拉巴)

- 亞太區域 (馬來西亞)
- 亞太地區 (孟買)
- 亞太區域 (紐西蘭)
- 亞太地區 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太地區 (雪梨)
- 亞太區域 (泰國)
- 亞太地區 (東京)
- 加拿大 (中部)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (US-West)
- AWS GovCloud (US-East)
- 環境類型
ARM_EC2(BUILD_GENERAL1_SMALL、BUILD_GENERAL1_MEDIUM、BUILD_GENERAL1_LARGE)
僅適用於下列區域：
 - 美國東部 (俄亥俄)
 - 美國東部 (維吉尼亞北部)

- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 亞太地區 (香港)
- 亞太地區 (雅加達)
- 亞太地區 (海德拉巴)
- 亞太區域 (馬來西亞)
- 亞太地區 (孟買)
- 亞太區域 (紐西蘭)
- 亞太地區 (大阪)
- 亞太地區 (首爾)
- 亞太區域 (泰國)
- 亞太地區 (新加坡)
- 亞太地區 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 南美洲 (聖保羅)
- AWS GovCloud (US-West)
- AWS GovCloud (US-East)

- 環境類型 `WINDOWS_EC2(BUILD_GENERAL1_MEDIUM、BUILD_GENERAL1_LARGE)` 僅適用於下列區域：
 - 美國東部 (俄亥俄)
 - 美國東部 (維吉尼亞北部)
 - 美國西部 (奧勒岡)
 - 亞太地區 (雪梨)
 - 亞太地區 (東京)
 - 歐洲 (法蘭克福)
 - 歐洲 (愛爾蘭)
 - 南美洲 (聖保羅)
- 運算模式 `AWS Lambda (ARM_LAMBDA_CONTAINER 和 LINUX_LAMBDA_CONTAINER)` 僅適用於下列區域：
 - 美國東部 (維吉尼亞北部)
 - 美國東部 (俄亥俄)
 - 美國西部 (加利佛尼亞北部)
 - 美國西部 (奧勒岡)
 - 亞太地區 (孟買)
 - 亞太地區 (新加坡)
 - 亞太地區 (雪梨)
 - 亞太地區 (東京)
 - 歐洲 (法蘭克福)
 - 歐洲 (愛爾蘭)
 - 南美洲 (聖保羅)
- 運算模式 `MAC_ARM` 僅適用於下列區域：
 - 美國東部 (維吉尼亞北部)
 - 美國東部 (俄亥俄)
 - 美國西部 (奧勒岡)
 - 亞太地區 (悉尼)
 - 歐洲 (法蘭克福)

對於運算類型 `BUILD_GENERAL1_2XLARGE`，最多可支援 100 GB 未壓縮的 Docker 影像。

Note

對於自訂建置環境映像，CodeBuild 支援在 Linux 和 Windows 中未壓縮高達 50 GB 的 Docker 映像，無論運算類型為何。若要查看您的建置映像的大小，請使用 Docker 執行 `docker images REPOSITORY:TAG` 命令。

您可以使用 Amazon EFS 來存取建置容器中的更多空間。如需詳細資訊，請參閱[Amazon Elastic File System 範例 AWS CodeBuild](#)。如果您希望在建置期間操作容器的磁碟空間，則必須使用授權模式來執行組建。

Note

預設會針對非 VPC 建置啟用 Docker 協助程式。如果您想要將 Docker 容器用於 VPC 建置，請參閱 Docker 文件網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特殊權限模式。

建置環境中的 Shell 和命令

您可以提供一組命令 AWS CodeBuild，讓在建置生命週期期間於建置環境中執行（例如，安裝建置相依性，以及測試和編譯原始程式碼）。有幾種方式可以指定這些命令：

- 建立建置規格檔案，並將其包含在您的來源碼中。在這個檔案中，指定您想在每個建置生命週期階段執行的命令。如需更多資訊，請參閱[CodeBuild 的建置規格參考](#)。
- 使用 CodeBuild 主控台建立建置專案。在 Insert build commands (插入組建命令) 的 Build commands (組建命令) 中，輸入您想在 build 階段執行的命令。如需詳細資訊，請參閱[建立組建專案 \(主控台\)](#)。
- 使用 CodeBuild 主控台變更建置專案的設定。在 Insert build commands (插入組建命令) 的 Build commands (組建命令) 中，輸入您想在 build 階段執行的命令。如需詳細資訊，請參閱[變更建置專案的設定 \(主控台\)](#)。
- 使用 AWS CLI AWS SDKs 建立建置專案或變更建置專案的設定。參考來源碼 (其中包含 Builds spec 檔案與您的命令)，或者指定單一字串以包含相同 Builds spec 檔案的內容。如需詳細資訊，請參閱[建立組建專案](#) 或 [變更建置專案設定](#)。
- 使用 AWS CLI AWS SDKs 啟動組建，指定 builds spec 檔案或包含同等 builds spec 檔案內容的單一字串。如需詳細資訊，請參閱[手動執行組建](#) 中針對 `buildspecOverride` 值的描述。

您可以指定任何 Shell 命令語言 (sh) 命令。在 buildspec 0.1 版中，CodeBuild 會在建置環境中的個別執行個體中執行每個 Shell 命令。這表示每個命令會與所有其他命令隔離執行。因此，根據預設，如果單一命令倚賴任何之前命令的狀態 (例如，變更目錄或設定環境變數)，您就無法加以執行。為因應這個限制，我們建議您使用 0.2 版，它可解決這個問題。如果您必須使用 0.1 版，我們建議以下方法：

- 在您的來源碼中納入 shell 指令碼，以包含您希望在預設 shell 單一執行個體中執行的命令。例如，您可以在來源碼中包含名為 my-script.sh 的檔案，以包含 `cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd`；這類命令。接著，在您的 Buildspec 檔案中，指定 `./my-script.sh` 命令。
- 在您的 Buildspec 檔案或 Build commands (組建命令) 設定 (僅限 build 階段) 中，輸入單一命令，以包含您希望在預設 shell 單一執行個體中執行的所有命令 (例如 `cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd`)。

如果 CodeBuild 遇到錯誤，與在其預設 shell 的執行個體中執行單一命令相比，錯誤可能更難進行故障診斷。

在 Windows Server Core 映像中執行的命令會使用 PowerShell shell。

建置環境中的環境變數

AWS CodeBuild 提供數個環境變數，您可以在建置命令中使用：

AWS_DEFAULT_REGION

組建執行所在的 AWS 區域 (例如 us-east-1)。此環境變數主要由 AWS CLI 使用。

AWS_REGION

組建執行所在的 AWS 區域 (例如 us-east-1)。此環境變數主要由 AWS SDKs 使用。

CODEBUILD_BATCH_BUILD_IDENTIFIER

批次組建中組建的識別符。這是在批次建置規格中指定的。如需詳細資訊，請參閱[the section called “Batch buildspec 參考”](#)。

CODEBUILD_BUILD_ARN

組建的 Amazon Resource Name (ARN) (例如 `arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`)。

CODEBUILD_BUILD_ID

組建的 CodeBuild ID (例如 `codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`)。

CODEBUILD_BUILD_IMAGE

CodeBuild 組建映像識別符 (例如 `aws/codebuild/standard:2.0`)。

CODEBUILD_BUILD_NUMBER

專案目前的建置編號。

CODEBUILD_BUILD_SUCCEEDING

目前建置是否成功。設定為 0 表示建置失敗；設定為 1 表示建置成功。

CODEBUILD_INITIATOR

啟動建置的實體。如果 CodePipeline 已啟動建置，則這是管道的名稱 (例如，`codepipeline/my-demo-pipeline`)。如果使用者開始建置，這是使用者的名稱 (例如 `MyUserName`)。如果 CodeBuild 的 Jenkins 外掛程式已啟動建置，這是字串 `CodeBuild-Jenkins-Plugin`。

CODEBUILD_KMS_KEY_ID

CodeBuild 用來加密建置輸出成品的 AWS KMS 金鑰識別符 (例如 `arn:aws:kms:region-ID:account-ID:key/key-ID` 或 `alias/key-alias`)。

CODEBUILD_PROJECT_ARN

專案的 Amazon Resource Name (ARN) (例如，`arn:aws:codebuild:region-ID:account-ID:project/project-name`)。

CODEBUILD_PUBLIC_BUILD_URL

此組建在公有組建網站上的組建結果 URL。只有在建置專案已啟用公有建置時，才會設定此變數。如需詳細資訊，請參閱 [取得公有建置專案 URLs](#)。

CODEBUILD_RESOLVED_SOURCE_VERSION

組建原始碼的版本識別符。內容取決於原始程式碼儲存庫：

CodeCommit、GitHub、GitHub Enterprise Server 和 Bitbucket

此變數包含遞交 ID。

CodePipeline

此變數包含 CodePipeline 提供的來源修訂版。

如果 CodePipeline 無法解析來源修訂，例如當來源是未啟用版本控制的 Amazon S3 儲存貯體時，則不會設定此環境變數。

Amazon S3

未設定此變數。

適用時，CODEBUILD_RESOLVED_SOURCE_VERSION 變數只能在 DOWNLOAD_SOURCE 階段之後使用。

CODEBUILD_SOURCE_REPO_URL

輸入成品或原始程式碼儲存庫的 URL。對於 Amazon S3，s3:// 後面接著儲存貯體名稱和輸入成品路徑。對於 CodeCommit 和 GitHub，這是儲存庫的複製 URL。如果組建源自 CodePipeline，則此環境變數可能是空的。

對於次要來源，次要來源儲存庫 URL 的環境變數為

CODEBUILD_SOURCE_REPO_URL_*<sourceIdentifier>*，其中 *<sourceIdentifier>* 是您建立的來源識別符。

CODEBUILD_SOURCE_VERSION

值的格式取決於來源儲存庫。

- 對於 Amazon S3，它是與輸入成品相關聯的版本 ID。
- 對於 CodeCommit，它是與要建置的來源碼版本相關聯的遞交 ID 或分支名稱。
- 對於 GitHub、GitHub Enterprise Server 和 Bitbucket，它是與要建置的來源碼版本相關聯的遞交 ID、分支名稱或標籤名稱。

Note

對於由 Webhook 提取請求事件觸發的 GitHub 或 GitHub Enterprise Server 建置，它是 *pr/pull-request-number*。

對於次要來源，次要來源版本的環境變數為

CODEBUILD_SOURCE_VERSION_*<sourceIdentifier>*，其中 *<sourceIdentifier>* 是您建立的來源識別符。如需詳細資訊，請參閱 [多個輸入來源和輸出成品範例](#)。

CODEBUILD_SRC_DIR

CodeBuild 用於建置的目錄路徑（例如，/tmp/src123456789/src）。

對於次要來源，次要來源目錄路徑的環境變數為

CODEBUILD_SRC_DIR_*<sourceIdentifier>*，其中 *<sourceIdentifier>* 是您建立的來源識別符。如需詳細資訊，請參閱[多個輸入來源和輸出成品範例](#)。

CODEBUILD_START_TIME

指定為 Unix 時間戳記的組建開始時間，以毫秒為單位。

CODEBUILD_WEBHOOK_ACTOR_ACCOUNT_ID

觸發 Webhook 事件之使用者的帳戶 ID。

CODEBUILD_WEBHOOK_BASE_REF

觸發目前建置之 Webhook 事件的基本參考名稱。對於提取請求，這是分支參考。

CODEBUILD_WEBHOOK_EVENT

觸發目前建置的 Webhook 事件。

CODEBUILD_WEBHOOK_MERGE_COMMIT

用於建置之合併遞交的識別符。當 Bitbucket 提取請求與 squash 策略合併且提取請求分支關閉時，會設定此變數。在此情況下，原始提取請求遞交不再存在，因此此環境變數包含壓縮合併遞交的識別符。

CODEBUILD_WEBHOOK_PREV_COMMIT

觸發目前組建的 Webhook 推送事件之前的最近遞交 ID。

CODEBUILD_WEBHOOK_HEAD_REF

觸發目前建置之 Webhook 事件的前端參考名稱。它可以是分支參考或標籤參考。

CODEBUILD_WEBHOOK_TRIGGER

顯示觸發建置的 Webhook 事件。此變數僅適用 Webhook 觸發的建置。此值會從 GitHub Enterprise Server 或 Bitbucket 傳送至 CodeBuild 的承載剖析。該值的格式取決於觸發建置的事件類型。

- 對於提取請求觸發的建置，它是 pr/*pull-request-number*。
- 對於建立新分支或將遞交推送至分支而觸發的建置，它是 branch/*branch-name*。

- 對於將標籤推送到儲存庫而觸發的建置，它是 tag/*tag-name*。

首頁

此環境變數一律設定為 /root。

AWS CodeBuild 也支援一組自我託管執行器建置的環境變數。若要進一步了解 CodeBuild 自我託管執行器，請參閱 [教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。

CODEBUILD_RUNNER_OWNER

觸發自我託管執行器建置的儲存庫擁有者。

CODEBUILD_RUNNER_REPO

觸發自我託管執行器建置的儲存庫名稱。

CODEBUILD_RUNNER_REPO_DOMAIN

觸發自我託管執行器建置的儲存庫網域。只有指定的 GitHub Enterprise 組建。

CODEBUILD_WEBHOOK_LABEL

用於在建置期間設定建置覆寫和自我託管執行器的標籤。

CODEBUILD_WEBHOOK_RUN_ID

與組建相關聯的工作流程執行 ID。

CODEBUILD_WEBHOOK_JOB_ID

與組建相關聯之任務的任務 ID。

CODEBUILD_WEBHOOK_WORKFLOW_NAME

如果在 Webhook 請求承載中存在與組建相關聯的工作流程名稱。

CODEBUILD_RUNNER_WITH_BUILDSPEC

如果在自我託管執行器請求標籤中設定 buildspec 覆寫，這會設定為 true。

您也可以為建置環境提供您自己的環境變數。如需詳細資訊，請參閱下列主題：

- [搭配 CodePipeline 使用 CodeBuild CodePipeline](#)
- [建立組建專案](#)

- [變更建置專案設定](#)
- [手動執行組建](#)
- [Buildspec 參考](#)

若要列出建置環境中所有可用的環境變數，您可以在建置期間執行 `printenv` 命令 (適用於 Linux 建置環境) 或 `"Get-ChildItem Env:"` (適用於 Windows 建置環境)。除了上述之外，開頭 `CODEBUILD_` 為的環境變數僅供 CodeBuild 內部使用。因此，您不應將它們用在建置命令中。

Important

我們強烈建議不使用環境變數來存放敏感值，尤其是 AWS 存取金鑰 IDs。環境變數可以使用 CodeBuild 主控台和 等工具以純文字顯示 AWS CLI。

我們建議您將敏感值存放在 Amazon EC2 Systems Manager 參數存放區中，然後從 `buildspec` 中擷取這些值。若要存放敏感值，請參閱《Amazon EC2 Systems Manager 使用者指南》中的 [Systems Manager 參數存放區](#) 和 [逐步解說：建立和測試字串參數（主控台）](#)。若要擷取這些值，請參閱 [Buildspec 語法](#) 中的 `parameter-store` 映射。

CODEBUILD_BUILD_URL

此組建的組建結果 URL。

建置環境中的背景工作

您可以在建置環境中執行背景工作。若要執行此作業，請在您的建置規格中使用 `nohup` 命令，以背景工作的形式執行命令，即使建立程序退出 Shell 亦同。使用 `disown` 命令以強制停止執行中的背景工作。

範例：

- 開始背景處理程序，並等待它稍後完成：

```
|  
nohup sleep 30 & echo $! > pidfile  
...  
wait $(cat pidfile)
```

- 開始背景處理程序，但不要等待它完成：

```
|  
nohup sleep 30 & disown $!
```

- 開始背景處理程序，並於稍後刪除：

```
|  
nohup sleep 30 & echo $! > pidfile  
...  
kill $(cat pidfile)
```

組建專案

建置專案包含如何執行建置的相關資訊，包括取得原始程式碼的位置、要使用的建置環境、要執行的建置命令，以及儲存建置輸出的位置。

您可以在處理組建專案時，執行這些任務：

主題

- [在 中建立建置專案 AWS CodeBuild](#)
- [建立通知規則](#)
- [在 中變更建置專案設定 AWS CodeBuild](#)
- [CodeBuild 中的多個存取字](#)
- [在 中刪除組建專案 AWS CodeBuild](#)
- [取得公有建置專案 URLs](#)
- [共用建置專案](#)
- [標記建置專案](#)
- [搭配 使用執行器 AWS CodeBuild](#)
- [搭配 使用 Webhook AWS CodeBuild](#)
- [在 中檢視建置專案的詳細資訊 AWS CodeBuild](#)
- [在 中檢視建置專案名稱 AWS CodeBuild](#)

在 中建立建置專案 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台 AWS CLI或 AWS SDKs來建立建置專案。

主題

- [先決條件](#)
- [建立組建專案 \(主控台\)](#)
- [建立建置專案 \(AWS CLI\)](#)
- [建立建置專案AWS SDKs\)](#)
- [建立建置專案 \(CloudFormation\)](#)

先決條件

在建立建置專案之前，請回答 中的問題[規劃組建](#)。

建立組建專案 (主控台)

在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。

如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開建置，選擇建置專案，然後選擇建立建置專案。

選擇 Create build project (建立建置專案)。

填寫下列各節。完成後，選擇頁面底部的建立建置專案。

章節：

- [專案組態](#)
- [來源](#)
- [Environment](#)
- [Buildspec](#)
- [批次組態](#)
- [成品](#)
- [日誌](#)

專案組態

Project name (專案名稱)

輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。

Description

輸入建置專案的選用描述，以協助其他使用者了解此專案的用途。

建置徽章

(選用) 選取啟用建置徽章，讓專案的建置狀態可見且可內嵌。如需詳細資訊，請參閱[建置徽章範例](#)。

 Note

如果您的來源供應商是 Amazon S3，則建置徽章不適用。

啟用並行建置限制

(選用) 如果您想要限制此專案的並行建置數量，請執行下列步驟：

1. 選取限制此專案可以啟動的並行建置數量。
2. 在並行建置限制中，輸入此專案允許的並行建置數目上限。此限制不能大於為帳戶設定的並行建置限制。如果您嘗試輸入大於帳戶限制的數字，則會顯示錯誤訊息。

只有當目前的建置數量小於或等於此限制時，才會啟動新的建置。如果目前的建置計數符合此限制，則會調節新的建置且不會執行。

其他資訊

(選用) 對於標籤，輸入您希望支援服務 AWS 使用的任何標籤的名稱和值。使用 Add row (新增資料列) 來新增標籤。您最多可以新增 50 個標籤。

來源

來源提供者

選擇來源碼提供者類型。使用下列清單來選擇適合您的來源提供者：

 Note

CodeBuild 不支援 Bitbucket Server。

Amazon S3

儲存貯體

選擇包含來源碼的輸入儲存貯體名稱。

S3 物件金鑰或 S3 資料夾

輸入 ZIP 檔案名稱或包含原始碼的資料夾路徑。輸入正斜線 (/) 以下載 S3 儲存貯體中的所有項目。

來源版本

輸入代表輸入檔案建置之物件的版本 ID。如需更多詳細資訊，請參閱 [使用的來源版本範例 AWS CodeBuild](#)。

CodeCommit

儲存庫

選擇您要使用的儲存庫。

參考類型

選擇分支、Git 標籤或遞交 ID 來指定原始碼的版本。如需詳細資訊，請參閱 [使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇建立淺複製，將歷史記錄截斷為指定的遞交數量。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

Bitbucket

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

選擇 AWS CodeConnections、OAuth、應用程式密碼或個人存取字符以連線至 CodeBuild。
Connection (連線)

選取 Bitbucket 連線或 Secrets Manager 秘密，以透過您指定的連線類型進行連線。

儲存庫

在我的 Bitbucket 帳戶或公有儲存庫中選擇儲存庫，然後輸入儲存庫 URL。

來源版本

輸入分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

建置狀態

如果您希望向來源提供者回報建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

在狀態內容中，輸入要用於 Bitbucket 遞交狀態中 name 參數的值。如需詳細資訊，請參閱 Bitbucket API 文件中的[建置](#)。

對於目標 URL，輸入要用於 Bitbucket 遞交狀態中 url 參數的值。如需詳細資訊，請參閱 Bitbucket API 文件中的[建置](#)。

Webhook 觸發的組建狀態一律會回報給來源提供者。若要讓從主控台啟動的組建狀態或向來源提供者回報的 API 呼叫，您必須選取此設定。

如果您專案的建置是由 Webhook 觸發，您必須將新的遞交推送至儲存庫，此設定的變更才會生效。

在主要來源 Webhook 事件中，如果您希望 CodeBuild 每次將程式碼變更推送至此儲存庫時都建置原始碼，請選取在每次將程式碼變更推送至此儲存庫時重建。如需 Webhook 和篩選群組的詳細資訊，請參閱 [Bitbucket Webhook 事件](#)。

GitHub

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

選擇 GitHub 應用程式、OAuth 或個人存取字符以連線至 CodeBuild。

Connection (連線)

選取 GitHub 連線或 Secrets Manager 秘密，以透過您指定的連線類型進行連線。

儲存庫

在我的 GitHub 帳戶、公有儲存庫或 GitHub 範圍 Webhook 中選擇儲存庫，然後輸入儲存庫 URL。 GitHub

來源版本

輸入分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

建置狀態

如果您希望向來源提供者回報建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

在狀態內容中，輸入要用於 GitHub 遞交狀態中 context 參數的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

針對目標 URL，輸入 GitHub 遞交狀態 target_url 中參數要使用的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

Webhook 觸發的組建狀態一律會回報給來源提供者。若要讓從主控台啟動的組建狀態或向來源提供者回報的 API 呼叫，您必須選取此設定。

如果您專案的建置是由 Webhook 觸發，您必須將新的遞交推送至儲存庫，此設定的變更才會生效。

在主要來源 Webhook 事件中，如果您希望 CodeBuild 每次將程式碼變更推送至此儲存庫時都建置原始碼，請選取在每次將程式碼變更推送至此儲存庫時重建。如需 Webhook 和篩選群組的詳細資訊，請參閱[GitHub Webhook 事件](#)。

GitHub Enterprise Server

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

選擇 AWS CodeConnections 或個人存取字符以連線至 CodeBuild。

Connection (連線)

選取 GitHub Enterprise 連線或 Secrets Manager 秘密，以透過您指定的連線類型進行連線。

儲存庫

在我的 GitHub Enterprise 帳戶或 GitHub Enterprise 範圍 Webhook 中選擇儲存庫，然後輸入儲存庫 URL。GitHub

來源版本

輸入提取請求、分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

建置狀態

如果您希望向來源提供者報告建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

在狀態內容中，輸入要用於 GitHub 遞交狀態中 context 參數的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

針對目標 URL，輸入要用於 GitHub 遞交狀態中 target_url 參數的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

Webhook 觸發的組建狀態一律會回報給來源提供者。若要讓從主控台啟動的組建狀態或向來源提供者回報的 API 呼叫，您必須選取此設定。

如果您專案的建置是由 Webhook 觸發，您必須將新的遞交推送至儲存庫，此設定的變更才會生效。

不安全的 SSL

選取 Enable insecure SSL (啟用不安全 SSL)，在連線至 GitHub Enterprise 專案儲存庫時忽略 SSL 警告。

在主要來源 Webhook 事件中，如果您希望 CodeBuild 每次將程式碼變更推送至此儲存庫時都建置原始碼，請選取每次將程式碼變更推送至此儲存庫時重建。如需 Webhook 和篩選群組的詳細資訊，請參閱 [GitHub Webhook 事件](#)。

GitLab

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

AWS CodeConnections 用於將 GitLab 連線至 CodeBuild。

Connection (連線)

選取要連線的 GitLab 連線 AWS CodeConnections。

儲存庫

選擇您要使用的儲存庫。

來源版本

輸入提取請求 ID、分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

建置狀態

如果您想要向來源提供者報告建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

GitLab Self Managed

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

AWS CodeConnections 用於將 GitLab 自我管理連線至 CodeBuild。

Connection (連線)

選取要透過的 GitLab 自我管理連線 AWS CodeConnections。

儲存庫

選擇您要使用的儲存庫。

來源版本

輸入提取請求 ID、分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

建置狀態

如果您希望向來源提供者回報建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

Environment

佈建模型

執行以下任意一項：

- 若要使用由 管理的隨需機群 AWS CodeBuild，請選擇隨需。透過隨需機群，CodeBuild 為您的建置提供運算。這些機器會在建置完成時銷毀。隨需機群受到完全管理，並包含自動擴展功能來處理需求激增。
- 若要使用由 管理的預留容量機群 AWS CodeBuild，請選擇預留容量，然後選取機群名稱。使用預留容量機群，您可以為建置環境設定一組專用執行個體。這些機器保持閒置狀態，準備好立即處理組建或測試，並減少組建持續時間。使用預留容量機群時，您的機器一律在執行中，只要佈建，就會持續產生成本。

如需相關資訊，請參閱[在預留容量機群上執行組建](#)。

環境映像

執行以下任意一項：

- 若要使用由 管理的 Docker 映像 AWS CodeBuild，請選擇受管映像，然後從作業系統、執行期 (Runtime)、映像和映像版本進行選擇。若可用，請從 Environment type (環境類型) 進行選擇。
- 若要使用另一個 Docker 映像，請選擇 Custom image (自訂映像)。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。如果您選擇其他登錄檔，請針對外部登錄檔 URL，使用 格式，在 Docker Hub 中輸入 Docker 映像檔的名稱和標籤 *docker repository/docker image name*。如果您選擇 Amazon ECR，請使用 Amazon ECR 儲存庫和 Amazon ECR 映像來選擇 AWS 帳戶中的 Docker 映像。

- 若要使用私有 Docker 映像，請選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。針對 Image registry (映像登錄) 選擇 Other registry (其他登錄)，然後輸入私人 Docker 映像的憑證的 ARN。登入資料必須由 Secrets Manager 建立。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [What Is AWS Secrets Manager](#)。

 Note

CodeBuild 會覆寫自訂 Docker 映像 ENTRYPOINT 的。

運算

執行以下任意一項：

- 若要使用 EC2 運算，請選擇 EC2。EC2 運算可在動作執行期間提供最佳化的彈性。
- 若要使用 Lambda 運算，請選擇 Lambda。Lambda 運算為您的建置提供最佳化的啟動速度。由於啟動延遲較低，Lambda 支援更快速的建置。Lambda 也會自動擴展，因此組建不會在佇列中等待執行。如需詳細資訊，請參閱 [在 AWS Lambda 運算上執行組建](#)。

服務角色

執行以下任意一項：

- 如果您沒有 CodeBuild 服務角色，請選擇新服務角色。在角色名稱中，輸入新角色的名稱。
- 如果您有 CodeBuild 服務角色，請選擇現有服務角色。在角色 ARN 中，選擇服務角色。

 Note

當您使用 主控台 建立建置專案時，您可以同時建立 CodeBuild 服務角色。根據預設，此角色只能與該建置專案搭配運作。如果您使用主控台將此服務角色與另一個建置專案建立關聯，則會更新此角色以與其他建置專案搭配運作。服務角色最多可以與 10 個組建專案搭配運作。

其他組態

自動重試限制

指定建置失敗後的其他自動重試次數。例如，如果自動重試限制設為 2，CodeBuild 會呼叫 RetryBuild API 以自動重試您的建置最多額外 2 次。

Timeout (逾時)

指定介於 5 分鐘到 36 小時之間的值，之後 CodeBuild 會在未完成時停止建置。如果 hours (小時) 和 minutes (分鐘) 空白，則會使用預設值 60 分鐘。

特殊權限

(選用) 如果您想要建置 Docker 映像，或希望建置在計劃使用此建置專案建置 Docker 映像時取得更高的權限，請選取啟用此旗標。否則，所有嘗試與 Docker 協助程式互動的相關建置都會失敗。您也必須啟動 Docker 協助程式，以讓您的建置與其互動。其中一種方法是執行下列建置命令，在 `buildspec install` 階段初始化 Docker 協助程式。如果您選擇 CodeBuild 透過 Docker 支援提供的建置環境映像，請勿執行這些命令。

Note

根據預設，非 VPC 建置會啟用 Docker 協助程式。如果您想要將 Docker 容器用於 VPC 建置，請參閱 Docker 文件網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特殊權限模式。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --  
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &  
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

VPC

如果您希望 CodeBuild 使用您的 VPC：

- 針對 VPC，選擇 CodeBuild 使用的 VPC ID。
- 對於 VPC 子網路，選擇包含 CodeBuild 使用之資源的子網路。
- 針對 VPC 安全群組，選擇 CodeBuild 用來允許存取 VPCs 中資源的安全群組。

如需詳細資訊，請參閱[AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用](#)。

運算

選擇其中一個可用的選項。

登錄檔登入資料

使用非私有登錄檔映像設定專案時，指定登錄檔登入資料。

 Note

只有在映像被私有登錄檔的映像覆寫時，才會使用此登入資料。

環境變數

輸入名稱和值，然後選擇組建要使用的每個環境變數類型。

 Note

CodeBuild 會自動為您的 AWS 區域設定環境變數。如果您尚未在 `buildspec.yml` 中加入環境變數，您必須加以設定：

- `AWS_ACCOUNT_ID`
- `IMAGE_REPO_NAME`
- `IMAGE_TAG`

主控台和 AWS CLI 使用者可以查看環境變數。如果您不在意環境變數的可見性，請設定 Name (名稱) 和 Value (值) 欄位，然後將 Type (類型) 設定為 Plaintext (純文字)。

我們建議您存放具有敏感值的環境變數，例如 AWS 存取金鑰 ID、AWS 私密存取金鑰或密碼，做為 Amazon EC2 Systems Manager 參數存放區或 中的參數 AWS Secrets Manager。

如果您使用 Amazon EC2 Systems Manager 參數存放區，則針對類型，選擇參數。針對名稱，輸入 CodeBuild 要參考的識別符。針對值，輸入儲存在 Amazon EC2 Systems Manager 參數存放區中的參數名稱。使用名為 `/CodeBuild/dockerLoginPassword` 的參數做為範例，針對 Type (類型)，選擇 Parameter (參數)。對於名稱，輸入 `LOGIN_PASSWORD`。針對數值，輸入 `/CodeBuild/dockerLoginPassword`。

 Important

如果您使用 Amazon EC2 Systems Manager 參數存放區，建議您使用開頭為 `/CodeBuild/` 的參數名稱來存放參數 `/CodeBuild/` (例如，`/CodeBuild/dockerLoginPassword`)。您可以使用 CodeBuild 主控台在 Amazon EC2 Systems Manager 中建立參數。選擇 Create parameter (建立參數)，然後遵循對話方塊中的說明。(在該對話方塊中，對於 KMS 金鑰，您可以在帳戶中指定 AWS KMS 金鑰的 ARN。Amazon EC2 Systems Manager 使用此金鑰在儲存期間加密參數的值，並在擷取期間解密。) 如果您使用 CodeBuild 主

控制台來建立參數，主控台會在儲存參數時以 `/CodeBuild/` 啟動參數名稱。如需詳細資訊，請參閱《Amazon EC2 Systems Manager [Systems Manager 使用者指南](#)》中的 [Systems Manager 參數存放區](#) 和 [Systems Manager 參數存放區主控台演練](#)。

如果您的建置專案參考 Amazon EC2 Systems Manager 參數存放區中存放的參數，則建置專案的服務角色必須允許 `ssm:GetParameters` 動作。如果您先前選擇新服務角色，CodeBuild 會將此動作包含在建置專案的預設服務角色中。不過，如果您選擇 Existing service role (現有服務角色)，則您必須個別將此動作包含在服務角色中。

如果您的建置專案參考存放在 Amazon EC2 Systems Manager 參數存放區中的參數，其參數名稱開頭不是 `/CodeBuild/`，而且您選擇了新服務角色，則必須更新該服務角色，以允許存取開頭不是的參數名稱 `/CodeBuild/`。這是因為該服務角色僅允許存取開頭為 `/CodeBuild/` 的參數名稱。

如果您選擇新增服務角色，服務角色會包含許可，以解密 Amazon EC2 Systems Manager 參數存放區中 `/CodeBuild/` 命名空間下的所有參數。

您設定的環境變數會取代現有環境變數。例如，如果 Docker 影像已包含名為 `MY_VAR` 且值為 `my_value` 的環境變數，而且您設定名為 `MY_VAR` 且值為 `other_value` 的環境變數，則 `my_value` 會取代為 `other_value`。同樣地，如果 Docker 影像已包含名為 `PATH` 且值為 `/usr/local/sbin:/usr/local/bin` 的環境變數，而且您設定名為 `PATH` 且值為 `$PATH:/usr/share/ant/bin` 的環境變數，則 `/usr/local/sbin:/usr/local/bin` 會取代為文字值 `$PATH:/usr/share/ant/bin`。

請不要設定名稱開頭為 `CODEBUILD_` 的任何環境變數。此字首保留供內部使用。

如果有多個位置定義同名的環境變數，則會決定值，如下所示：

- 開始建置操作呼叫中的值會採用最高優先順序。
- 組建專案定義中的值會採用下一個優先順序。
- `buildspec` 宣告中的值會採用最低優先順序。

如果您使用 Secrets Manager，請針對類型選擇 Secrets Manager。針對名稱，輸入 CodeBuild 要參考的識別符。針對 Value (值)，請使用 `secret-id:json-key:version-stage:version-id` 模式輸入 reference-key。如需相關資訊，請參閱 [Secrets Manager reference-key in the buildspec file](#)。

Important

如果您使用 Secrets Manager，我們建議您存放名稱開頭為 `/CodeBuild/` 的秘密 `/CodeBuild/` (例如 `/CodeBuild/dockerLoginPassword`)。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [What Is AWS Secrets Manager](#)。

如果您的建置專案參考存放在 Secrets Manager 中的秘密，則建置專案的服務角色必須允許 `secretsmanager:GetSecretValue` 動作。如果您先前選擇新服務角色，CodeBuild 會將此動作包含在建置專案的預設服務角色中。不過，如果您選擇 Existing service role (現有服務角色)，則您必須個別將此動作包含在服務角色中。如果您的建置專案參考存放在 Secrets Manager 中的秘密，其秘密名稱開頭不是 `/CodeBuild/`，而且您選擇了新服務角色，您必須更新服務角色，以允許存取開頭不是 `/CodeBuild/` 的秘密名稱。這是因為服務角色僅允許存取開頭為 `/CodeBuild/` 的秘密名稱。`/CodeBuild/`。

如果您選擇新增服務角色，服務角色會包含許可，以解密 Secrets Manager 中 `/CodeBuild/` 命名空間下的所有秘密。

Buildspec

組建規格

執行以下任意一項：

- 如果您的來源碼包含 buildspec 檔案，請選擇 Use a buildspec file (使用 buildspec 檔案)。根據預設，CodeBuild buildspec.yml 會在來源碼根目錄中尋找名為 `buildspec.yml` 的檔案。如果您的 buildspec 檔案使用不同的名稱或位置，請在 Buildspec 名稱中輸入來源根的路徑（例如，`buildspec-two.yml` 或 `configuration/buildspec.yml`。如果 buildspec 檔案位於 S3 儲存貯體中，它必須與您的建置專案位於相同的 AWS 區域。使用其 ARN 指定 buildspec 檔案（例如，`arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`）。
- 如果您的來源碼未包含 buildspec 檔案，或者，您要執行的組建命令與針對 build 階段（位於來源碼根目錄的 buildspec.yml 檔案中）所指定的組建命令不同，則請選擇 Insert build commands (插入組建命令)。針對 Build commands (組建命令)，在 build 階段中輸入您要執行的命令。針對多個命令，以 `&&` 區隔每個命令（例如，`mvn test && mvn package`）。若要在其他階段執行命令，或者如果您有該 build 階段的長命令清單，請將 buildspec.yml 檔案新增至來源碼根目錄，將命令新增至檔案，然後在來源碼根目錄中選擇使用 buildspec.yml。

如需更多資訊，請參閱 [Buildspec 參考](#)。

批次組態

您可以執行組建群組做為單一操作。如需詳細資訊，請參閱 [批次執行組建](#)。

定義批次組態

選取以允許此專案中的批次建置。

批次服務角色

提供批次建置的服務角色。

選擇下列其中一項：

- 如果您沒有批次服務角色，請選擇新服務角色。在服務角色中，輸入新角色的名稱。
- 如果您有批次服務角色，請選擇現有服務角色。在服務角色中，選擇服務角色。

批次建置在批次組態中導入新的安全性角色。此新角色是必要的，因為 CodeBuild 必須能夠代表您呼叫 StartBuild、StopBuild 和 RetryBuild 動作，才能在批次處理中執行組建。客戶應該使用新的角色，而不是與他們在組建中使用的角色相同，原因有兩個：

- 賦予建置角色 StartBuild、StopBuild 和 RetryBuild 權限，此將允許單一組建透過 buildspect 啟動更多組建。
- CodeBuild 批次建置提供限制，限制可用於批次中建置的建置和運算類型數目。如果組建角色具有這些權限，組建本身可能會略過這些限制。

批次允許的運算類型

選取批次允許的運算類型。選取所有適用項目。

批次允許的機群

選取批次允許的機群。選取所有適用項目。

批次允許的建置上限

輸入批次中允許的組建數目上限。如果批次超過此限制，批次將會失敗。

批次逾時

輸入批次建置完成的時間上限。

合併成品

選取將所有成品從批次合併到單一位置，讓批次中的所有成品合併到單一位置。

批次報告模式

選取批次組建所需的組建狀態報告模式。

 Note

只有在專案來源為 Bitbucket、GitHub 或 GitHub Enterprise 時，才可使用此欄位，並在來源下選取建置開始和完成時，向來源提供者報告建置狀態。

彙總組建

選取 `Aggregate`，讓批次中所有組建的狀態合併為單一狀態報告。

個別組建

選取 `Individual`，讓批次中所有組建的組建狀態分別報告。

成品

類型

執行以下任意一項：

- 如果您不要建立任何建置輸出成品，則請選擇 `No artifacts` (無成品)。如果您只執行建置測試，或想要將 Docker 映像推送至 Amazon ECR 儲存庫，建議您執行此操作。
- 若要將建置輸出儲存在 S3 儲存貯體中，請選擇 `Amazon S3`，然後執行下列動作：
 - 如果您想要使用專案名稱做為組建輸出 ZIP 檔案或資料夾名稱，則請將 `Name` (名稱) 保留空白。否則請輸入名稱。(如果您想要輸出 ZIP 檔案，並且想要 ZIP 檔案有副檔名，則請務必將其包含在 ZIP 檔案名稱後面。)
 - 如果您想要 `buildspec` 檔案中所指定的名稱來覆寫主控台中所指定的任何名稱，請選取 `Enable semantic versioning` (啟用語意版本控制)。`buildspec` 檔案中的名稱是在建置時計算，並使用 Shell 命令語言。例如，您可以將日期和時間附加到成品名稱，讓它一律是唯一的。唯一成品名稱可防止覆寫成品。如需詳細資訊，請參閱[Buildspec 語法](#)。
 - 針對 `Bucket name` (儲存貯體名稱)，選擇輸出儲存貯體的名稱。
 - 如果您在本程序稍早選擇 `Insert build commands` (插入組建命令)，然後針對 `Output files` (輸出檔案)，輸入要放入組建輸出 ZIP 檔案或資料夾之組建中的檔案位置。針對多個位置，以逗號區隔每個位置 (例如，`appspec.yml`，`target/my-app.jar`)。如需詳細資訊，請參閱[Buildspec 語法](#)中的 `files` 描述。
 - 如果您不想要加密建置成品，請選取 `Remove artifacts encryption` (移除成品加密)。

針對您想要的每組次要成品：

1. 針對 Artifact identifier (成品識別符)，輸入的值少於 128 個字元，並且只包含英數字元和底線。
2. 選擇 Add artifact (新增成品)。
3. 遵循先前的步驟來設定您的次要成品。
4. 選擇 Save artifact (儲存成品)。

其他組態

加密金鑰

(選擇性) 執行下列操作：

- 若要在帳戶中使用 AWS 受管金鑰 適用於 Amazon S3 的 來加密建置輸出成品，請將加密金鑰保留空白。這是預設值。
- 若要使用客戶受管金鑰來加密建置輸出成品，請在加密金鑰中輸入 KMS 金鑰的 ARN。使用 `arn:aws:kms:region-ID:account-ID:key/key-ID` 格式。

快取類型

針對 Cache type (快取類型)，選擇以下其中一項：

- 如果您不想要使用快取，請選擇 No cache (無快取)。
- 如果您想要使用 Amazon S3 快取，請選擇 Amazon S3，然後執行下列動作：
 - 針對 Bucket (儲存貯體)，選擇存放快取的 S3 儲存貯體名稱。
 - (選用) 對於快取路徑字首，輸入 Amazon S3 路徑字首。Cache path prefix (快取路徑字首) 值類似目錄名稱。它可讓您將快取存放至儲存貯體的相同目錄下方。

Important

請不要在路徑字首結尾附加尾端斜線 (/)。

- 如果您想要使用本機快取，請選擇 Local (本機)，然後選擇一或多個本機快取模式。

Note

「Docker layer cache」(Docker 層快取) 模式僅適用於 Linux。如果您選擇此模式，您的專案必須以特殊權限模式執行。

使用快取可節省大量建置時間，因為建置環境的可重複使用部分存放在快取中，並用於各建置。如需在 buildspec 檔案中指定快取的詳細資訊，請參閱[Buildspec 語法](#)。如需快取的詳細資訊，請參閱「[快取建置以改善效能](#)」。

日誌

選擇您要建立的日誌。您可以建立 Amazon CloudWatch Logs、Amazon S3 Logs 或兩者。

CloudWatch

如果您想要 Amazon CloudWatch Logs 日誌：

CloudWatch 日誌

選取 CloudWatch 日誌。

Group name (群組名稱)

輸入 Amazon CloudWatch Logs 日誌群組的名稱。

串流名稱

輸入您的 Amazon CloudWatch Logs 日誌串流名稱。

S3

如果您想要 Amazon S3 日誌：

S3 日誌

選取 S3 logs (S3 日誌)。

儲存貯體

選擇日誌的 S3 儲存貯體名稱。

路徑字首

輸入日誌的字首。

停用 S3 日誌加密

如果您不希望 S3 日誌加密，請選取。

建立建置專案 (AWS CLI)

如需 AWS CLI 搭配 CodeBuild 使用的詳細資訊，請參閱[命令列參考](#)。

若要使用 建立 CodeBuild 組建專案 AWS CLI，您可以建立 JSON 格式的[專案](#)結構、填入結構，然後呼叫 `create-project` 命令來建立專案。

建立 JSON 檔案

使用 `--generate-cli-skeleton` 選項，使用 `create-project` 命令建立骨架 JSON 檔案：

```
aws codebuild create-project --generate-cli-skeleton > <json-file>
```

這會建立具有 *<json-file> ##### JSON ##*。

填寫 JSON 檔案

如下所示修改 JSON 資料並儲存結果。

```
{
  "name": "<project-name>",
  "description": "<description>",
  "source": {
    "type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" | "GITLAB" |
    "GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
    "location": "<source-location>",
    "gitCloneDepth": "<git-clone-depth>",
    "buildspec": "<buildspec>",
    "InsecureSsl": "<insecure-ssl>",
    "reportBuildStatus": "<report-build-status>",
    "buildStatusConfig": {
      "context": "<context>",
      "targetUrl": "<target-url>"
    },
    "gitSubmodulesConfig": {
      "fetchSubmodules": "<fetch-submodules>"
    },
    "auth": {
      "type": "<auth-type>",
      "resource": "<auth-resource>"
    },
    "sourceIdentifier": "<source-identifier>"
  },
  "secondarySources": [
    {
      "type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" |
      "GITLAB" | "GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
```

```

    "location": "<source-location>",
    "gitCloneDepth": "<git-clone-depth>",
    "buildspec": "<buildspec>",
    "InsecureSsl": "<insecure-ssl>",
    "reportBuildStatus": "<report-build-status>",
    "auth": {
        "type": "<auth-type>",
        "resource": "<auth-resource>"
    },
    "sourceIdentifier": "<source-identifier>"
}
],
"secondarySourceVersions": [
{
    "sourceIdentifier": "<secondary-source-identifier>",
    "sourceVersion": "<secondary-source-version>"
}
],
"sourceVersion": "<source-version>",
"artifacts": {
    "type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
    "location": "<artifacts-location>",
    "path": "<artifacts-path>",
    "namespaceType": "<artifacts-namespacetype>",
    "name": "<artifacts-name>",
    "overrideArtifactName": "<override-artifact-name>",
    "packaging": "<artifacts-packaging>"
},
"secondaryArtifacts": [
{
    "type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
    "location": "<secondary-artifact-location>",
    "path": "<secondary-artifact-path>",
    "namespaceType": "<secondary-artifact-namespaceType>",
    "name": "<secondary-artifact-name>",
    "packaging": "<secondary-artifact-packaging>",
    "artifactIdentifier": "<secondary-artifact-identifier>"
}
],
"cache": {
    "type": "<cache-type>",
    "location": "<cache-location>",
    "mode": [
        "<cache-mode>"
    ]
}

```



```

    ]
  },
  "environment": {
    "type": "LINUX_CONTAINER" | "LINUX_GPU_CONTAINER" | "ARM_CONTAINER" |
    "WINDOWS_SERVER_2019_CONTAINER" | "WINDOWS_SERVER_2022_CONTAINER",
    "image": "<image>",
    "computeType": "BUILD_GENERAL1_SMALL" | "BUILD_GENERAL1_MEDIUM" |
    "BUILD_GENERAL1_LARGE" | "BUILD_GENERAL1_2XLARGE",
    "certificate": "<certificate>",
    "environmentVariables": [
      {
        "name": "<environmentVariable-name>",
        "value": "<environmentVariable-value>",
        "type": "<environmentVariable-type>"
      }
    ],
    "registryCredential": [
      {
        "credential": "<credential-arn-or-name>",
        "credentialProvider": "<credential-provider>"
      }
    ],
    "imagePullCredentialsType": "CODEBUILD" | "SERVICE_ROLE",
    "privilegedMode": "<privileged-mode>"
  },
  "serviceRole": "<service-role>",
  "autoRetryLimit": <auto-retry-limit>,
  "timeoutInMinutes": <timeout>,
  "queuedTimeoutInMinutes": <queued-timeout>,
  "encryptionKey": "<encryption-key>",
  "tags": [
    {
      "key": "<tag-key>",
      "value": "<tag-value>"
    }
  ],
  "vpcConfig": {
    "securityGroupIds": [
      "<security-group-id>"
    ],
    "subnets": [
      "<subnet-id>"
    ],
    "vpcId": "<vpc-id>"
  }
}

```

```

},
"badgeEnabled": "<badge-enabled>",
"logsConfig": {
  "cloudWatchLogs": {
    "status": "<cloudwatch-logs-status>",
    "groupName": "<group-name>",
    "streamName": "<stream-name>"
  },
  "s3Logs": {
    "status": "<s3-logs-status>",
    "location": "<s3-logs-location>",
    "encryptionDisabled": "<s3-logs-encryption-disabled>"
  }
},
"fileSystemLocations": [
  {
    "type": "EFS",
    "location": "<EFS-DNS-name-1>:/<directory-path>",
    "mountPoint": "<mount-point>",
    "identifier": "<efs-identifier>",
    "mountOptions": "<efs-mount-options>"
  }
],
"buildBatchConfig": {
  "serviceRole": "<batch-service-role>",
  "combineArtifacts": <combine-artifacts>,
  "restrictions": {
    "maximumBuildsAllowed": <max-builds>,
    "computeTypesAllowed": [
      "<compute-type>"
    ],
    "fleetsAllowed": [
      "<fleet-name>"
    ]
  },
  "timeoutInMins": <batch-timeout>,
  "batchReportMode": "REPORT_AGGREGATED_BATCH" | "REPORT_INDIVIDUAL_BUILDS"
},
"concurrentBuildLimit": <concurrent-build-limit>
}

```

取代以下項目：

name

必要. 此建置專案的名稱。此名稱在您 AWS 帳戶中的所有建置專案中必須是唯一的。

description

選用。此建置專案的描述。

source

必要. [ProjectSource](#) 物件，其中包含此建置專案的原始程式碼設定的相關資訊。在您新增 source 物件之後，即可使用 新增最多 12 個以上的來源。這些設定包含下列項目：

來源/類型

必要. 包含要建置之來源碼的儲存庫類型。有效值包含：

- CODECOMMIT
- CODEPIPELINE
- GITHUB
- GITHUB_ENTERPRISE
- GITLAB
- GITLAB_SELF_MANAGED
- BITBUCKET
- S3
- NO_SOURCE

若您使用 NO_SOURCE，則 Buildspec 不能是檔案，因為專案沒有任何來源。反之，您必須使用 buildspec 屬性來為您的 buildspec 指定 YAML 格式字串。如需詳細資訊，請參閱[建立不含來源的建置專案](#)。

來源/位置

除非您將 *<source-type>* 設定為 `CODEPIPELINE`，否則為必要。所指定儲存庫類型的來源碼位置。

- 對於 CodeCommit，HTTPS 會將 URL 複製到包含原始程式碼和 buildspec 檔案的儲存庫（例如 `https://git-codecommit.<region-id>.amazonaws.com/v1/repos/<repo-name>`）。

- 對於 Amazon S3，建置輸入儲存貯體名稱，後面接著包含原始碼和 buildspec 的 ZIP 檔案的路徑和名稱。例如：
 - 對於位於輸入儲存貯體根目錄的 ZIP 檔案：`<bucket-name>/<object-name>.zip`。
 - 對於位於輸入儲存貯體中子資料夾中的 ZIP 檔案：`<bucket-name>/<subfolder-path>/<object-name>.zip`。
- 針對 GitHub，HTTPS 會將 URL 複製到包含來源碼和 buildspec 檔案的儲存庫。URL 必須包含 github.com。您必須將 AWS 帳戶連接到 GitHub 帳戶。若要這樣做，請使用 CodeBuild 主控台來建立建置專案。
 - 選擇 Authorize application (授權應用程式)。(連線至 GitHub 帳戶後，您不需要完成建立建置專案。您可以關閉 CodeBuild 主控台。)
- 針對 GitHub Enterprise Server，HTTP 或 HTTPS 會將 URL 複製到包含來源碼和 buildspec 檔案的儲存庫。您還必須將 AWS 帳戶連接到 GitHub Enterprise Server 帳戶。若要這樣做，請使用 CodeBuild 主控台來建立建置專案。
 1. 在 GitHub Enterprise Server 中建立個人存取字符。
 2. 將此權杖複製到剪貼簿，以便在建立 CodeBuild 專案時使用它。如需詳細資訊，請參閱 GitHub Help 網站上的 [Creating a personal access token for the command line](#)。
 3. 當您使用主控台建立 CodeBuild 專案時，在來源供應商的來源中，選擇 GitHub Enterprise。
 4. 針對 Personal Access Token (個人存取字符)，貼上已複製至剪貼簿的字符。選擇 Save Token (儲存字符)。您的 CodeBuild 帳戶現在已連線至 GitHub Enterprise Server 帳戶。
- 對於 GitLab 和 GitLab 自我管理，HTTPS 會將 URL 複製到包含原始程式碼和 buildspec 檔案的儲存庫。請注意，如果您使用 GitLab，URL 必須包含 gitlab.com。如果您使用 GitLab 自我管理，則 URL 不需要包含 gitlab.com。您必須將 AWS 帳戶連接到 GitLab 或 GitLab 自我管理帳戶。若要這樣做，請使用 CodeBuild 主控台來建立建置專案。
 - 在開發人員工具導覽窗格中，選擇設定、連線，然後選擇建立連線。在此頁面上，建立 GitLab 或 GitLab 自我管理連線，然後選擇連線至 GitLab。
- 針對 Bitbucket，HTTPS 會將 URL 複製到包含來源碼和 buildspec 檔案的儲存庫。URL 必須包含 bitbucket.org。您還必須將 AWS 帳戶連接到 Bitbucket 帳戶。若要這樣做，請使用 CodeBuild 主控台來建立建置專案。
 1. 當您使用主控台連線 (或重新連線) Bitbucket 時，請在 Bitbucket 的 Confirm access to your account (確認帳戶存取) 頁面上，選擇 Grant access (授予存取權)。(連線至 Bitbucket 帳戶後，您不需要完成建立建置專案。您可以關閉 CodeBuild 主控台。)
- 對於 AWS CodePipeline，請勿指定 location 的值 source。CodePipeline 會忽略此值，因為當您在 CodePipeline 中建立管道時，您會在管道的來源階段指定來源碼位置。

source/gitCloneDepth

選用。要下載的歷史記錄深度。最小值為 0。如果此值為 0、大於 25 或未提供，則會下載每個建置專案的完整歷史記錄。如果您的來源類型是 Amazon S3，則不支援此值。

source/buildspec

選用。要使用的組建規格定義或檔案。如果此值未提供，或設定為空字串，則來源碼必須包含其根目錄中的 `buildspec.yml` 檔案。如果設定此值，可以是內嵌 `buildspec` 定義、相對於主要來源根目錄的替代 `buildspec` 檔案路徑，或是 S3 儲存貯體的路徑。儲存貯體必須與建置專案位於相同的 AWS 區域。使用其 ARN 指定 `buildspec` 檔案 (例如，`arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`)。如需詳細資訊，請參閱[Buildspec 檔案名稱和儲存位置](#)。

來源/身分驗證

包含 CodeBuild 存取要建置之原始碼的授權設定相關資訊。

source/auth/type

必要。要使用的授權類型。有效的值如下：

- OAUTH
- CODECONNECTIONS
- SECRETS_MANAGER

source/auth/resource

選用。套用至指定授權類型的資源值。這可以是 Secrets Manager ARN 或 AWS CodeConnections ARN。

source/reportBuildStatus

指定是否將建置的啟動和完成狀態傳送給來源提供者。如果您搭配 GitHub、GitHub Enterprise Server 或 Bitbucket 以外的來源供應商設定此項，則會擲出 `invalidInputException`。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

source/buildStatusConfig

包含定義 CodeBuild 組建專案如何向來源提供者報告組建狀態的資訊。此選項僅在來源類型為 GITHUB、GITHUB_ENTERPRISE 或 時使用 BITBUCKET。

source/buildStatusConfig/context

對於 Bitbucket 來源，此參數用於 Bitbucket 遞交狀態中的 `name` 參數。對於 GitHub 來源，此參數用於 GitHub 遞交狀態中的 `context` 參數。

例如，您可以使用 CodeBuild 環境變數讓 context 包含建置編號和 Webhook 觸發：

```
AWS CodeBuild sample-project Build #${CODEBUILD_BUILD_NUMBER} -  
${CODEBUILD_WEBHOOK_TRIGGER}
```

這會導致 Webhook 提取請求事件觸發的建置 #24 的內容出現如下：

```
AWS CodeBuild sample-project Build #24 - pr/8
```

source/buildStatusConfig/targetUrl

對於 Bitbucket 來源，此參數用於 Bitbucket 遞交狀態中的 url 參數。對於 GitHub 來源，此參數用於 GitHub 遞交狀態中的 target_url 參數。

例如，您可以將 targetUrl 設定為 `https://aws.amazon.com/codebuild/<path to build>`，遞交狀態將連結至此 URL。

您也可以在中包含 CodeBuild 環境變數 targetUrl，將其他資訊新增至 URL。例如，若要將建置區域新增至 URL，請將 targetUrl 設定為：

```
"targetUrl": "https://aws.amazon.com/codebuild/<path to build>?region=  
$AWS_REGION"
```

如果建置區域為 us-east-2，這會擴展至：

```
https://aws.amazon.com/codebuild/<path to build>?region=us-east-2
```

source/gitSubmodulesConfig

選用。Git 子模組組態的相關資訊。僅與 CodeCommit、GitHub、GitHub Enterprise Server 和 Bitbucket 搭配使用。

source/gitSubmodulesConfig/fetchSubmodules

如果您想要將 Git 子模組包含在您的儲存庫中，請將 fetchSubmodules 設定為 true。包含的 Git 子模組必須設定為 HTTPS。

來源/InsecureSsl

選用。只能與 GitHub Enterprise Server 搭配使用。將此數值設定為 true，忽略連線至 GitHub Enterprise Server 專案儲存庫時出現的 TLS 警告。預設值為 false。InsecureSsl 應僅用於測試目的。不應用於生產環境。

source/sourceIdentifier

專案來源的使用者定義識別符。主要來源的選用。次要來源為必要。

secondarySources

選用。[ProjectSource](#) 物件陣列，其中包含組建專案次要來源的相關資訊。您最多可以新增 12 個次要來源。secondarySources 物件使用與物件相同的屬性。在次要來源物件中，sourceIdentifier 是必要的。

secondarySourceVersions

選用。[ProjectSourceVersion](#) 物件的陣列。如果在組建層級指定 secondarySourceVersions，則其優先順序會高於此。

sourceVersion

選用。要為此專案建置的建置輸入版本。如果未指定，則會使用最新的版本。如果指定，則必須是以下其中一個：

- 對於 CodeCommit，要使用的遞交 ID、分支或 Git 標籤。
- 針對 GitHub，為遞交 ID、提取請求 ID、分支名稱，或與您想要建置之來源碼版本對應的標籤名稱。如果指定提取請求 ID，其格式必須為 pr/pull-request-ID (例如，pr/25)。如果指定分支名稱，則會使用分支的 HEAD 遞交 ID。如果未指定，則會使用預設分支的 HEAD 遞交 ID。
- 對於 GitLab，遞交 ID、提取請求 ID、分支名稱、標籤名稱或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。
- 針對 Bitbucket，為遞交 ID、分支名稱，或與您想要建置之來源碼版本對應的標籤名稱。如果指定分支名稱，則會使用分支的 HEAD 遞交 ID。如果未指定，則會使用預設分支的 HEAD 遞交 ID。
- 對於 Amazon S3：物件的版本 ID，代表要使用的建置輸入 ZIP 檔案。

如果在組建層級指定 sourceVersion，則該版本的優先順序會高於此 sourceVersion (在專案層級)。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。

成品

必要。[ProjectArtifacts](#) 物件，其中包含此建置專案輸出成品設定的相關資訊。在您新增 artifacts 物件之後，即可使用 新增最多 12 個以上的成品。這些設定包含下列項目：

成品/類型

必要. 建置輸出成品類型。有效的值如下：

- CODEPIPELINE
- NO_ARTIFACTS
- S3

成品/位置

僅與S3成品類型搭配使用。不適用於其他成品類型。

您在先決條件中建立或識別的輸出儲存貯體名稱。

成品/路徑

僅與S3成品類型搭配使用。不適用於其他成品類型。

要放置 ZIP 檔案或資料夾的輸出儲存貯體路徑。如果您未指定的值path，CodeBuild 會使用 namespaceType (如果指定) 和 name 來判斷建置輸出 ZIP 檔案或資料夾的路徑和名稱。例如，如果您MyPath為 和 path MyArtifact.zip 指定 name，則路徑和名稱將為 MyPath/MyArtifact.zip。

artifacts/namespaceType

僅與S3成品類型搭配使用。不適用於其他成品類型。

組建輸出 ZIP 檔案或資料夾的命名空間。有效值包括 BUILD_ID 與 NONE。使用 BUILD_ID，將建置 ID 插入至建置輸出 ZIP 檔案或資料夾的路徑。否則，請使用 NONE。如果您未指定的值namespaceType，CodeBuild 會使用 path (如果指定) 和 name 來判斷建置輸出 ZIP 檔案或資料夾的路徑和名稱。例如，如果您MyPath為 指定 path、BUILD_ID為 指定 namespaceType，MyArtifact.zip並為 指定 name，則路徑和名稱將為 MyPath/*build-ID*/MyArtifact.zip。

artifacts/name

僅與S3成品類型搭配使用。不適用於其他成品類型。

內部建置輸出 ZIP 檔案或資料夾的名稱location。例如，如果您MyPath為 path和 MyArtifact.zip 指定 name，則路徑和名稱將為 MyPath/MyArtifact.zip。

artifacts/overrideArtifactName

僅與 S3 成品類型搭配使用。不適用於其他成品類型。

選用。如果設定為 `true`，在 `buildspec` 檔案的 `artifacts` 區塊中指定的名稱會覆寫 `name`。如需詳細資訊，請參閱[CodeBuild 的建置規格參考](#)。

成品/封裝

僅與 S3 成品類型搭配使用。不適用於其他成品類型。

選用。指定如何封裝成品。允許的值為：

NONE

建立包含建置成品的資料夾。這是預設值。

ZIP

建立包含建置成品的 ZIP 檔案。

secondaryArtifacts

選用。[ProjectArtifacts](#) 物件陣列，其中包含組建專案次要成品設定的相關資訊。您可以新增最多 12 個次要成品。`secondaryArtifacts` 使用物件所使用的許多相同設定。

快取

必要。包含此建置專案快取設定相關資訊的 [ProjectCache](#) 物件。如需詳細資訊，請參閱[快取建置](#)。

環境

必要。[ProjectEnvironment](#) 物件，其中包含此專案建置環境設定的相關資訊。這些設定包括：

environment/type

必要。建置環境類型。如需詳細資訊，請參閱 CodeBuild API 參考中的[類型](#)。

environment/image

必要。此建置環境所使用的 Docker 影像識別符。一般而言，此識別符以 *image-name:tag* 表示。例如，在 CodeBuild 用來管理其 Docker 映像的 Docker 儲存庫中，這可能是 `aws/codebuild/standard:5.0`。在 Docker Hub 中，為 `maven:3.3.9-jdk-8`。在 Amazon ECR 中，*account-id.dkr.ecr.region-id.amazonaws.com/your-Amazon-ECR-repo-name:tag*。如需詳細資訊，請參閱[CodeBuild 提供的 Docker 映像](#)。

environment/computeType

必要。指定此建置環境使用的運算資源。如需詳細資訊，請參閱 CodeBuild API 參考中的[computeType](#)。

環境/憑證

選用。Amazon S3 儲存貯體的 ARN、路徑字首和包含 PEM 編碼憑證的物件金鑰。物件金鑰可以是包含 PEM 編碼憑證的 .pem 檔案或 .zip 檔案。例如，如果您的 Amazon S3 儲存貯體名稱為 *<my-bucket>*，您的路徑字首為 *<cert>*，而您的物件金鑰名稱為 *<certificate.pem>*，則可接受的格式 certificate 為 *<my-bucket/cert/certificate.pem>* 或 *arn:aws:s3:::<my-bucket/cert/certificate.pem>*。

environment/environmentVariables

選用。[EnvironmentVariable](#) 物件陣列，其中包含您要為此建置環境指定的環境變數。每個環境變數都以包含 name、和 value、namevalue 和 type 的物件表示 type。

主控台和 AWS CLI 使用者可以查看所有環境變數。如果您對環境變數的可見性沒有疑慮，請將 name 和 設定為 value，並將 type 設定為 PLAINTEXT。

我們建議您使用敏感值存放環境變數，例如 AWS 存取金鑰 ID、AWS 私密存取金鑰或密碼，做為 Amazon EC2 Systems Manager 參數存放區或 中的參數 AWS Secrets Manager。對於 name，針對該預存參數，設定 CodeBuild 的識別符以供參考。

如果您使用 Amazon EC2 Systems Manager 參數存放區，請針對 value 設定參數存放區中存放的參數名稱。將 type 設定為 PARAMETER_STORE。使用名為 的參數/CodeBuild/dockerLoginPassword 做為範例，將 name 設定為 LOGIN_PASSWORD。將 value 設定為 /CodeBuild/dockerLoginPassword。將 type 設定為 PARAMETER_STORE。

Important

如果您使用 Amazon EC2 Systems Manager 參數存放區，建議您使用開頭為 的參數名稱來存放參數 /CodeBuild/ (例如，/CodeBuild/dockerLoginPassword)。您可以使用 CodeBuild 主控台在 Amazon EC2 Systems Manager 中建立參數。選擇 Create parameter (建立參數)，然後遵循對話方塊中的說明。(在該對話方塊中，對於 KMS 金鑰，您可以在帳戶中指定 AWS KMS 金鑰的 ARN。Amazon EC2 Systems Manager 使用此金鑰在儲存期間加密參數的值，並在擷取期間解密。) 如果您使用 CodeBuild 主控台來建立參數，主控台會在儲存參數/CodeBuild/時以 啟動參數名稱。如需詳細資訊，請參閱《Amazon EC2 Systems Manager [Systems Manager 使用者指南](#) 中的 [Systems Manager 參數存放區](#) 和 [Systems Manager 參數存放區主控台演練](#)。

如果您的建置專案參考 Amazon EC2 Systems Manager 參數存放區中存放的參數，則建置專案的服務角色必須允許 ssm:GetParameters 動作。如果您先前選擇新服務角色，CodeBuild 會將此動作包含在建置專案的預設服務角色中。不過，如果您選擇 Existing service role (現有服務角色)，則您必須個別將此動作包含在服務角色中。

如果您的建置專案參考存放在 Amazon EC2 Systems Manager 參數存放區中的參數，其參數名稱開頭不是 `/CodeBuild/`，而且您選擇了新服務角色，則必須更新該服務角色，以允許存取開頭不是 `/CodeBuild/` 的參數名稱。這是因為該服務角色僅允許存取開頭為 `/CodeBuild/` 的參數名稱。

如果您選擇新增服務角色，服務角色會包含 Amazon EC2 Systems Manager 參數存放區中 `/CodeBuild/` 命名空間下所有參數的解密許可。

您設定的環境變數會取代現有環境變數。例如，如果 Docker 影像已包含名為 `MY_VAR` 且值為 `my_value` 的環境變數，而且您設定名為 `MY_VAR` 且值為 `other_value` 的環境變數，則 `my_value` 會取代為 `other_value`。同樣地，如果 Docker 影像已包含名為 `PATH` 且值為 `/usr/local/sbin:/usr/local/bin` 的環境變數，而且您設定名為 `PATH` 且值為 `$PATH:/usr/share/ant/bin` 的環境變數，則 `/usr/local/sbin:/usr/local/bin` 會取代為文字值 `$PATH:/usr/share/ant/bin`。

請不要設定名稱開頭為 `CODEBUILD_` 的任何環境變數。此字首保留供內部使用。

如果有多個位置定義同名的環境變數，則會決定值，如下所示：

- 開始建置操作呼叫中的值會採用最高優先順序。
- 組建專案定義中的值會採用下一個優先順序。
- `buildspec` 宣告中的值會採用最低優先順序。

如果您使用 Secrets Manager，請針對 `value` 設定儲存在 Secrets Manager 中的參數名稱。將 `type` 設定為 `SECRETS_MANAGER`。使用名為 `/CodeBuild/dockerLoginPassword` 做為範例，將 `name` 設定為 `LOGIN_PASSWORD`。將 `value` 設定為 `/CodeBuild/dockerLoginPassword`。將 `type` 設定為 `SECRETS_MANAGER`。

Important

如果您使用 Secrets Manager，我們建議您存放名稱開頭為 `/CodeBuild/` 的秘密（例如 `/CodeBuild/dockerLoginPassword`）。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [What Is AWS Secrets Manager](#)。

如果您的建置專案參考存放在 Secrets Manager 中的秘密，則建置專案的服務角色必須允許 `secretsmanager:GetSecretValue` 動作。如果您先前選擇新服務角色，CodeBuild 會將此動作包含在建置專案的預設服務角色中。不過，如果您選擇 Existing service role（現有服務角色），則您必須個別將此動作包含在服務角色中。

如果您的建置專案參考存放在 Secrets Manager 中的秘密，其秘密名稱不是以開頭 `/CodeBuild/`，而且您選擇了新服務角色，則您必須更新服務角色，以允許存取不是

以開頭的秘密名稱/CodeBuild/。這是因為服務角色僅允許存取開頭為 的秘密名稱/CodeBuild/。
如果您選擇新服務角色，服務角色會包含許可，以解密 Secrets Manager 中/CodeBuild/命名空間下的所有秘密。

environment/registryCredential

選用。[RegistryCredential](#) 物件，指定提供私有 Docker 登錄檔存取權的登入資料。

environment/registryCredential/credential

指定使用 建立的登入資料的 ARN 或名稱 AWS Managed Services。您只能在登入資料存在於您目前區域時才能使用其名稱

environment/registryCredential/credentialProvider

唯一有效的值為 SECRETS_MANAGER。

當此值設為：

- `imagePullCredentials` 必須設定為 `SERVICE_ROLE`。
- 影像不能是精選影像或 Amazon ECR 影像。

environment/imagePullCredentialsType

選用。CodeBuild 用來在建置中提取映像的登入資料類型。兩種有效值如下：

程式碼建置

CODEBUILD 指定 CodeBuild 使用自己的登入資料。您必須編輯 Amazon ECR 儲存庫政策，才能信任 CodeBuild 服務主體。

SERVICE_ROLE

指定 CodeBuild 使用您建置專案的服務角色。

當您使用跨帳戶或私有登錄影像，您必須使用 `SERVICE_ROLE` 登入資料。當您使用 CodeBuild 策劃的映像時，您必須使用CODEBUILD登入資料。

environment/privilegedMode

`true` 只有在您計劃使用此建置專案建置 Docker 映像時，才將 設定為 。否則，所有嘗試與 Docker 協助程式互動的相關建置都會失敗。您也必須啟動 Docker 協助程式，以讓您的建置與其互動。
執行這項操作的一種方式是執行下列建置命令，以在 `buildspec` 檔案的 `install` 階段中初始化

Docker 協助程式。如果您使用 Docker 支援指定 CodeBuild 提供的建置環境映像，請勿執行這些命令。

 Note

根據預設，非 VPC 建置會啟用 Docker 協助程式。如果您想要將 Docker 容器用於 VPC 建置，請參閱 Docker 文件網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特殊權限模式。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --  
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &  
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

serviceRole

必要。CodeBuild 用來代表使用者與 服務互動的服務角色 ARN（例如 `arn:aws:iam::account-id:role/role-name`）。

autoRetryLimit

選用。建置失敗後的其他自動重試次數。例如，如果自動重試限制設為 2，CodeBuild 將呼叫 RetryBuild API 以自動重試您的建置最多額外 2 次。

timeoutInMinutes

選用。介於 5 到 2160 (36 小時) 之間的分鐘數，之後 CodeBuild 會在組建未完成時停止組建。如果未指定，則會使用預設值 60。若要判斷 CodeBuild 是否和何時因逾時而停止建置，請執行 `batch-get-builds` 命令。若要判斷是否已停止建置，請查看 `buildStatus` 值 `FAILED` 的輸出。若要判斷組建何時逾時，請查看與 `phaseStatus` 值 `TIMED_OUT` 建立關聯之 `endTime` 值的輸出。

queuedTimeoutInMinutes

選用。介於 5 到 480 (8 小時) 之間的分鐘數，之後 CodeBuild 會在仍在佇列中時停止建置。如果未指定，則會使用預設值 60。

encryptionKey

選用。CodeBuild AWS KMS key 用來加密建置輸出之 的別名或 ARN。如果您指定別名，則請使用 `arn:aws:kms:region-ID:account-ID:key/key-ID` 格式，或者，如果別名已存在，則請使用 `alias/key-alias` 格式。如果未指定，則會使用 Amazon S3 的 AWS 受管 KMS 金鑰。

標籤

選用。[標籤](#)物件陣列，提供您要與此建置專案建立關聯的標籤。您可以指定最多 50 個標籤。支援 CodeBuild 組建專案標籤的任何 AWS 服務都可以使用這些標籤。每個標籤會以具有 key 和 的物件表示 value。

vpcConfig

選用。包含專案 VPC 組態相關資訊的 [VpcConfig](#) 物件。如需詳細資訊，請參閱[AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用](#)。

這些屬性包括：

vpclId

必要。CodeBuild 使用的 VPC ID。執行此命令，以取得您區域中的所有 VPC ID 清單：

```
aws ec2 describe-vpcs --region <region-ID>
```

子網路

必要。子網路 IDs 陣列，其中包含 CodeBuild 使用的資源。執行此命令，以取得這些 ID：

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region <region-ID>
```

securityGroupIds

必要。CodeBuild 用來允許存取 VPC 中資源的安全群組 IDs 陣列。執行此命令，以取得這些 ID：

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --<region-ID>
```

badgeEnabled

選用。指定是否要在 CodeBuild 專案中包含組建徽章。設為 `true` 以啟用建置徽章，`false` 否則設為 `0`。如需詳細資訊，請參閱[使用 CodeBuild 建置徽章範例](#)。

logsConfig

[LogsConfig](#) 物件，其中包含此組建日誌所在位置的相關資訊。

logsConfig/cloudWatchLogs

[CloudWatchLogsConfig](#) 物件，其中包含將日誌推送至 CloudWatch Logs 的相關資訊。

logsConfig/s3Logs

[S3LogsConfig](#) 物件，其中包含將日誌推送至 Amazon S3 的相關資訊。

fileSystemLocations

選用。[ProjectFileSystemsLocation](#) 物件陣列，其中包含 Amazon EFS 組態的相關資訊。

buildBatchConfig

選用。`buildBatchConfig` 物件是 [ProjectBuildBatchConfig](#) 結構，其中包含專案的批次建置組態資訊。

buildBatchConfig/serviceRole

批次建置專案的服務角色 ARN。

buildBatchConfig/combineArtifacts

布林值，指定是否要將批次組建的組建成品合併為單一成品位置。

buildBatchConfig/restrictions/maximumBuildsAllowed

允許的最大組建數量。

buildBatchConfig/restrictions/computeTypesAllowed

字串陣列，指定批次建置允許的運算類型。如需這些值，請參閱[建置環境運算類型](#)。

buildBatchConfig/restrictions/fleetsAllowed

字串陣列，指定批次建置允許的機群。如需詳細資訊，請參閱[在預留容量機群上執行建置](#)。

buildBatchConfig/timeoutInMinutes

批次建置必須完成的時間上限，以分鐘為單位。

buildBatchConfig/batchReportMode

指定如何將建置狀態報告傳送至批次建置的來源提供者。有效值包含：

REPORT_AGGREGATED_BATCH

(預設) 將所有建置狀態彙總到單一狀態報告中。

REPORT_INDIVIDUAL_BUILDS

針對每個個別建置傳送單獨的狀態報告。

concurrentBuildLimit

此專案允許並行建置的最大數量。

只有當目前的建置數量小於或等於此限制時，才會啟動新的建置。如果目前的建置計數符合此限制，則會調節新的建置且不會執行。

建立專案

若要建立專案，請再次執行 [create-project](#) 命令，並傳遞您的 JSON 檔案：

```
aws codebuild create-project --cli-input-json file://<json-file>
```

如果成功，[專案](#)物件的 JSON 表示法會出現在主控台輸出中。如需此資料的範例，請參閱 [CreateProject 回應語法](#)。

除了建置專案名稱之外，您稍後可以變更任何建置專案設定。如需詳細資訊，請參閱[變更建置專案的設定 \(AWS CLI\)](#)。

若要開始執行建置，請參閱[執行建置 \(AWS CLI\)](#)。

如果您的原始程式碼存放在 GitHub 儲存庫中，而且您希望 CodeBuild 每次將程式碼變更推送到儲存庫時重建原始程式碼，請參閱[自動開始執行建置 \(AWS CLI\)](#)。

建立建置專案AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

建立建置專案 (CloudFormation)

如需 AWS CodeBuild 搭配使用的詳細資訊 CloudFormation，請參閱AWS CloudFormation 《使用者指南》中的 [CodeBuild CloudFormation 範本](#)。

建立通知規則

您可以使用通知規則，以在發生重要變更 (例如建置成功和失敗) 時通知使用者。通知規則會指定用於傳送通知的事件和 Amazon SNS 主題。如需詳細資訊，請參閱[什麼是通知？](#)

您可以使用 主控台或 AWS CLI 來建立的通知規則 AWS CodeBuild。

建立通知規則 (主控台)

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 選擇 Build (建置)，選擇 Build projects (建置專案)，然後選擇您要新增通知的建置專案。
3. 在建置專案頁面上，選擇 Notify (通知)，然後選擇 Create notification rule (建立通知規則)。您也可以前往建置專案的 Settings (設定) 頁面，然後選擇 Create notification rule (建立通知規則)。
4. 在 Notification name (通知名稱) 中，輸入規則的名稱。
5. 如果您只希望提供給 Amazon EventBridge 的資訊包含在通知中，請在 Detail type (詳細資訊類型) 中，選擇 Basic (基本)。如果您想要包含提供給 Amazon EventBridge 的資訊，以及 CodeBuild 或通知管理員可能提供的資訊，請選擇完整。

如需詳細資訊，請參閱[了解通知內容與安全性](#)。

6. 在 Events that trigger notifications (觸發通知的事件) 中，選取您要傳送通知的事件。如需詳細資訊，請參閱[組建專案上通知規則的事件](#)。
7. 在 Targets (目標) 中，執行下列其中一個動作：
 - 如果您已將資源設定為搭配通知使用，請在選擇目標類型中選擇聊天應用程式 (Slack) 中的 Amazon Q Developer 或 SNS 主題。在選擇目標中，選擇用戶端的名稱 (適用於聊天應用程式中在 Amazon Q Developer 中設定的 Slack 用戶端) 或 Amazon SNS 主題的 Amazon Resource Name (ARN) (適用於已設定通知所需政策的 Amazon SNS 主題)。
 - 如果您尚未設定要與通知搭配使用的資源，請選擇 Create target (建立目標)，然後選擇 SNS topic (SNS 主題)。在 codestar-notifications- 之後，提供主題名稱，然後選擇 Create (建立)。

 Note

- 如果您在建立通知規則的過程中建立 Amazon SNS 主題，將會為您套用允許通知功能將事件發佈至主題的政策。使用針對通知規則建立的主題，有助於確保您只訂閱需要接收此資源相關通知的使用者。
- 您無法在聊天應用程式用戶端中建立 Amazon Q Developer，作為建立通知規則的一部分。如果您在聊天應用程式 (Slack) 中選擇 Amazon Q Developer，您會看到一個按鈕，指示您在聊天應用程式中設定 Amazon Q Developer 中的用戶端。選擇此選項會在聊天應用程式主控台中開啟 Amazon Q Developer。如需詳細資訊，請參閱[聊天應用程式中設定通知與 Amazon Q Developer 之間的整合](#)。
- 如果您想要使用現有的 Amazon SNS 主題做為目標，除了該主題可能存在的任何其他政策之外，還必須新增AWS CodeStar通知所需的政策。如需詳細資訊，請參閱[為通知設定 Amazon SNS 主題](#)和[了解通知內容與安全性](#)。

8. 若要完成建立規則，請選擇 Submit (提交)。
9. 您必須為使用者訂閱規則的 Amazon SNS 主題，才能接收通知。如需詳細資訊，請參閱[訂閱使用者成為目標的 Amazon SNS 主題](#)。您也可以聊天應用程式中設定通知與 Amazon Q Developer 之間的整合，將通知傳送至 Amazon Chime 聊天室。如需詳細資訊，請參閱[聊天應用程式中設定通知與 Amazon Q Developer 之間的整合](#)。

建立通知規則 (AWS CLI)

1. 在終端機或命令提示字元中，執行 create-notification rule 命令以產生 JSON 架構：

```
aws codestarnotifications create-notification-rule --generate-cli-skeleton  
> rule.json
```

您可以將檔案命名為任何您想要的名稱。在此範例中，檔案命名為 *rule.json*。

2. 在純文字編輯器中開啟 JSON 檔案，並編輯成包含您想要用於規則的資源、事件類型和目標。下列範例顯示 ID **MyNotificationRule** 為 **123456789012** AWS 的帳戶中名為 **MyBuildProject** 之組建專案的通知規則。當組建成功時，通知會以完整詳細資訊類型傳送至名為 *codestar-notifications-MyNotificationTopic* 的 Amazon SNS 主題：

```
{  
  "Name": "MyNotificationRule",
```

```
"EventTypeId": [
  "codebuild-project-build-state-succeeded"
],
"Resource": "arn:aws:codebuild:us-east-2:123456789012:MyBuildProject",
"Targets": [
  {
    "TargetType": "SNS",
    "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
  }
],
"Status": "ENABLED",
"DetailType": "FULL"
}
```

儲存檔案。

3. 在終端機或命令列中，再次執行 `create-notification-rule` 命令，使用您剛編輯的檔案建立通知規則：

```
aws codestarnotifications create-notification-rule --cli-input-json
file://rule.json
```

4. 如果成功，此命令會傳回通知規則的 ARN，如下所示：

```
{
  "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

在中變更建置專案設定 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台、AWS CLI 或 AWS SDKs 來變更建置專案的設定。

如果您將測試報告新增至建置專案，請確定您的 IAM 角色具有 [中所述的許可測試報告許可](#)。

主題

- [變更建置專案的設定 \(主控台\)](#)
- [變更建置專案的設定 \(AWS CLI\)](#)
- [變更建置專案的設定 \(AWS SDKs\)](#)

變更建置專案的設定 (主控台)

若要變更建置專案的設定，請執行下列程序：

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 執行以下任意一項：
 - 選擇您想要變更的組建專案連結，然後選擇 Build details (組建詳細資訊)。
 - 選擇您想要變更之組建專案旁的選項按鈕，選擇 View details (檢視詳細資訊)，然後選擇 Build details (組建詳細資訊)。

您可以修改下列區段：

章節

- [專案組態](#)
- [來源](#)
- [Environment](#)
- [Buildspec](#)
- [批次組態](#)
- [成品](#)
- [日誌](#)

專案組態

在專案組態區段中，選擇編輯。變更完成後，請選擇更新組態以儲存新組態。

您可以修改下列屬性。

Description

輸入建置專案的選用描述，以協助其他使用者了解此專案的用途。

組建徽章

選取 Enable build badge (啟用組建徽章)，讓專案的組建狀態變成可見且可嵌入。如需詳細資訊，請參閱[建置徽章範例](#)。

 Note

如果您的來源供應商是 Amazon S3，則建置徽章不適用。

啟用並行建置限制

如果您想要限制此專案的並行建置數量，請執行下列步驟：

1. 選取限制此專案可以啟動的並行建置數量。
2. 在並行建置限制中，輸入此專案允許的並行建置數目上限。此限制不能大於為帳戶設定的並行建置限制。如果您嘗試輸入大於帳戶限制的數字，則會顯示錯誤訊息。

只有當目前的建置數量小於或等於此限制時，才會啟動新的建置。如果目前的建置計數符合此限制，則會調節新的建置且不會執行。

啟用公有建置存取

若要讓專案的建置結果可供大眾使用，包括無法存取 AWS 帳戶的使用者，請選取啟用公有建置存取權，並確認您想要公開建置結果。下列屬性用於公有建置專案：

公有建置服務角色

如果您想要讓 CodeBuild 為您建立新的服務角色，請選取新服務角色；如果您想要使用現有的服務角色，請選取現有服務角色。

公有建置服務角色可讓 CodeBuild 讀取 CloudWatch Logs 並下載專案建置的 Amazon S3 成品。這是讓專案的建置日誌和成品可供大眾使用的必要項目。

服務角色

輸入新服務角色或現有服務角色的名稱。

若要將專案的建置結果設為私有，請清除啟用公有建置存取。

如需詳細資訊，請參閱[取得公有建置專案 URLs](#)。

 Warning

將專案的建置結果設為公開時，請注意下列事項：

- 所有專案的建置結果、日誌和成品，包括在專案私有時執行的建置，都可供大眾使用。

- 所有建置日誌和成品都可供大眾使用。環境變數、原始程式碼和其他敏感資訊可能已輸出至建置日誌和成品。您必須小心將哪些資訊輸出到建置日誌。一些最佳實務如下：
 - 請勿在環境變數中存放敏感值，尤其是 AWS 存取金鑰 IDs 和私密存取金鑰。建議您使用 Amazon EC2 Systems Manager 參數存放區或 AWS Secrets Manager 來存放敏感值。
 - 遵循 [使用 Webhook 的最佳實務](#) 來限制哪些實體可以觸發組建，並且不要將 buildspec 存放在專案本身，以確保您的 Webhook 盡可能安全。
- 惡意使用者可以使用公有組建來分發惡意成品。我們建議專案管理員檢閱所有提取請求，以確認提取請求是合法的變更。我們也建議您使用其檢查總和驗證任何成品，以確保下載正確的成品。

其他資訊

對於標籤，輸入您希望支援服務 AWS 使用的任何標籤的名稱和值。使用 Add row (新增資料列) 來新增標籤。您最多可以新增 50 個標籤。

來源

在來源區段中，選擇編輯。變更完成後，請選擇更新組態以儲存新組態。

您可以修改下列屬性：

來源提供者

選擇來源碼提供者類型。使用下列清單來選擇適合您的來源提供者：

Note

CodeBuild 不支援 Bitbucket Server。

Amazon S3

儲存貯體

選擇包含來源碼的輸入儲存貯體名稱。

S3 物件金鑰或 S3 資料夾

輸入 ZIP 檔案名稱或包含原始碼的資料夾路徑。輸入正斜線 (/) 以下載 S3 儲存貯體中的所有項目。

來源版本

輸入代表輸入檔案建置之物件的版本 ID。如需更多詳細資訊，請參閱 [使用的來源版本範例 AWS CodeBuild](#)。

CodeCommit

儲存庫

選擇您要使用的儲存庫。

參考類型

選擇分支、Git 標籤或遞交 ID 來指定原始程式碼的版本。如需詳細資訊，請參閱 [使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇建立淺複製，其歷史記錄會截斷為指定的遞交數量。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

Bitbucket

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

選擇 AWS CodeConnections、OAuth、應用程式密碼或個人存取字符以連線至 CodeBuild。
Connection (連線)

選取 Bitbucket 連線或 Secrets Manager 秘密，以透過您指定的連線類型進行連線。

儲存庫

在我的 Bitbucket 帳戶或公有儲存庫中選擇儲存庫，然後輸入儲存庫 URL。

來源版本

輸入分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

建置狀態

如果您希望向來源提供者報告建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

在狀態內容中，輸入要用於 Bitbucket 遞交狀態中 name 參數的值。如需詳細資訊，請參閱 Bitbucket API 文件中的[建置](#)。

針對目標 URL，輸入要用於 Bitbucket 遞交狀態中 url 參數的值。如需詳細資訊，請參閱 Bitbucket API 文件中的[建置](#)。

Webhook 觸發的組建狀態一律會回報給來源提供者。若要讓從主控台啟動的組建狀態或向來源提供者回報的 API 呼叫，您必須選取此設定。

如果您專案的建置是由 Webhook 觸發，您必須將新的遞交推送至儲存庫，此設定的變更才會生效。

在主要來源 Webhook 事件中，如果您希望 CodeBuild 每次將程式碼變更推送至此儲存庫時都建置原始碼，請選取在每次將程式碼變更推送至此儲存庫時重建。如需 Webhook 和篩選群組的詳細資訊，請參閱 [Bitbucket Webhook 事件](#)。

GitHub

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

選擇 GitHub 應用程式、OAuth 或個人存取字符以連線至 CodeBuild。

Connection (連線)

選取 GitHub 連線或 Secrets Manager 秘密，以透過您指定的連線類型進行連線。

儲存庫

在我的 GitHub 帳戶、公有儲存庫或 GitHub 範圍 Webhook 中選擇儲存庫，然後輸入儲存庫 URL。 GitHub

來源版本

輸入分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

建置狀態

如果您希望向來源提供者回報建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

在狀態內容中，輸入要用於 GitHub 遞交狀態中 context 參數的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

針對目標 URL，輸入要用於 GitHub 遞交狀態中 target_url 參數的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

Webhook 觸發的組建狀態一律會回報給來源提供者。若要讓從主控台啟動的組建狀態或向來源提供者回報的 API 呼叫，您必須選取此設定。

如果您專案的建置是由 Webhook 觸發，您必須將新的遞交推送至儲存庫，此設定的變更才會生效。

在主要來源 Webhook 事件中，如果您希望 CodeBuild 每次將程式碼變更推送至此儲存庫時都建置原始碼，請選取在每次將程式碼變更推送至此儲存庫時重建。如需 Webhook 和篩選群組的詳細資訊，請參閱[GitHub Webhook 事件](#)。

GitHub Enterprise Server

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

選擇 AWS CodeConnections 或個人存取字符以連線至 CodeBuild。

Connection (連線)

選取 GitHub Enterprise 連線或 Secrets Manager 秘密，以透過您指定的連線類型進行連線。

儲存庫

在我的 GitHub Enterprise 帳戶或 GitHub Enterprise 範圍 Webhook 中選擇儲存庫，然後輸入儲存庫 URL。GitHub

來源版本

輸入提取請求、分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模組

若您要將 Git 子模組包含在您的儲存庫中，請選擇 Use Git submodules (使用 Git 子模組)。

建置狀態

如果您希望向來源提供者回報建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

針對狀態內容，輸入 GitHub 遞交狀態中 context 參數要使用的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

針對目標 URL，輸入要用於 GitHub 遞交狀態中 target_url 參數的值。如需詳細資訊，請參閱《GitHub 開發人員指南》中的[建立認可狀態](#)。

Webhook 觸發的組建狀態一律會回報給來源提供者。若要讓從主控台啟動的組建狀態或向來源提供者回報的 API 呼叫，您必須選取此設定。

如果您專案的建置是由 Webhook 觸發，您必須將新的遞交推送至儲存庫，此設定的變更才會生效。

不安全的 SSL

選取 Enable insecure SSL (啟用不安全 SSL)，在連線至 GitHub Enterprise 專案儲存庫時忽略 SSL 警告。

在主要來源 Webhook 事件中，如果您希望 CodeBuild 每次將程式碼變更推送至此儲存庫時都建置原始碼，請選取在每次將程式碼變更推送至此儲存庫時重建。如需 Webhook 和篩選群組的詳細資訊，請參閱 [GitHub Webhook 事件](#)。

GitLab

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

AWS CodeConnections 用於將 GitLab 連線至 CodeBuild。

Connection (連線)

選取要透過的 GitLab 連線 AWS CodeConnections。

儲存庫

選擇您要使用的儲存庫。

來源版本

輸入提取請求 ID、分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

建置狀態

如果您希望向來源提供者回報建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

GitLab Self Managed

登入資料

選擇預設來源憑證或自訂來源憑證，然後依照指示管理預設來源憑證或自訂來源憑證。

連線類型

AWS CodeConnections 用於將 GitLab 自我管理連線至 CodeBuild。

Connection (連線)

選取要透過的 GitLab 自我管理連線 AWS CodeConnections。

儲存庫

選擇您要使用的儲存庫。

來源版本

輸入提取請求 ID、分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱[使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

Git 複製深度

選擇 Git clone depth (Git 複製深度)，建立記錄截取至指定遞交數的 Shallow 複製。如果您想要完整複製，請選擇 Full (完整)。

建置狀態

如果您希望向來源提供者報告建置的開始和完成狀態，請在建置開始和完成時，選取向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

Environment

在環境區段中，選擇編輯。變更完成後，請選擇更新組態以儲存新組態。

您可以修改下列屬性：

佈建模型

若要變更佈建模型，請選擇變更佈建模型，然後執行下列其中一項操作：

- 若要使用由 管理的隨需機群 AWS CodeBuild，請選擇隨需。透過隨需機群，CodeBuild 為您的建置提供運算。這些機器會在建置完成時銷毀。隨需機群是全受管的，並包含自動擴展功能來處理需求激增。
- 若要使用由 管理的預留容量機群 AWS CodeBuild，請選擇預留容量，然後選取機群名稱。使用預留容量機群，您可以為建置環境設定一組專用執行個體。這些機器會保持閒置，準備好立即處理組建或測試，並減少組建持續時間。使用預留容量機群時，您的機器一律在執行中，只要佈建，就會持續產生成本。

如需相關資訊，請參閱[在預留容量機群上執行組建](#)。

環境映像

若要變更建置映像，請選擇覆寫映像並執行下列其中一項操作：

- 若要使用 管理的 Docker 映像 AWS CodeBuild，請選擇受管映像，然後從作業系統、執行時間 (s)、映像和映像版本進行選擇。若可用，請從 Environment type (環境類型) 進行選擇。
- 若要使用另一個 Docker 映像，請選擇 Custom image (自訂映像)。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。如果您選擇其他登錄檔，請針對外部登錄檔 URL，使用 格式在 Docker Hub 中輸入 Docker 映像檔的名稱和標籤 *docker repository/docker*

image name。如果您選擇 Amazon ECR，請使用 Amazon ECR 儲存庫和 Amazon ECR 映像來選擇 AWS 帳戶中的 Docker 映像。

- 若要使用私有 Docker 映像，請選擇自訂映像。針對環境類型，選擇 ARM、Linux、Linux GPU 或 Windows。針對 Image registry (映像登錄) 選擇 Other registry (其他登錄)，然後輸入私人 Docker 映像的憑證的 ARN。登入資料必須由 Secrets Manager 建立。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [What Is AWS Secrets Manager](#)。

 Note

CodeBuild 會覆寫自訂 Docker 映像 ENTRYPOINT 的。

服務角色

執行以下任意一項：

- 如果您沒有 CodeBuild 服務角色，請選擇新服務角色。在角色名稱中，輸入新角色的名稱。
- 如果您有 CodeBuild 服務角色，請選擇現有服務角色。在角色 ARN 中，選擇服務角色。

 Note

當您使用 主控台 建立建置專案時，您可以同時建立 CodeBuild 服務角色。根據預設，此角色只能與該建置專案搭配運作。如果您使用主控台將此服務角色與另一個建置專案建立關聯，則會更新此角色以與其他建置專案搭配運作。服務角色最多可以與 10 個組建專案搭配運作。

其他組態

Timeout (逾時)

指定介於 5 分鐘到 36 小時之間的值，之後 CodeBuild 會在未完成時停止建置。如果 hours (小時) 和 minutes (分鐘) 空白，則會使用預設值 60 分鐘。

特權

如果您想要建置 Docker 映像或希望建置獲得更高的權限，請選取啟用此旗標。只有在您打算使用此建置專案建置 Docker 映像時。否則，所有嘗試與 Docker 協助程式互動的相關建置都會失敗。您也必須啟動 Docker 協助程式，以讓您的建置與其互動。其中一種方法是執行下列建置命令，在 buildspec 的 install 階段初始化 Docker 協助程式。如果您選擇 CodeBuild 透過 Docker 支援提供的建置環境映像，請勿執行這些命令。

Note

根據預設，非 VPC 建置會啟用 Docker 協助程式。如果您想要將 Docker 容器用於 VPC 建置，請參閱 Docker 文件網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特殊權限模式。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --  
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &  
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

VPC

如果您希望 CodeBuild 使用您的 VPC：

- 針對 VPC，選擇 CodeBuild 使用的 VPC ID。
- 針對 VPC 子網路，選擇包含 CodeBuild 使用之資源的子網路。
- 針對 VPC 安全群組，選擇 CodeBuild 用來允許存取 VPCs 中資源的安全群組。

如需詳細資訊，請參閱[AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用](#)。

運算

選擇其中一個可用的選項。

登錄檔登入資料

使用非私有登錄檔映像設定專案時，指定登錄檔登入資料。

Note

只有在映像被私有登錄檔的映像覆寫時，才會使用此登入資料。

環境變數

輸入名稱和值，然後選擇要用於組建的每個環境變數類型。

Note

CodeBuild 會自動為您的 AWS 區域設定環境變數。如果您尚未在 `buildspec.yml` 中加入環境變數，您必須加以設定：

- AWS_ACCOUNT_ID
- IMAGE_REPO_NAME
- IMAGE_TAG

主控台和 AWS CLI 使用者可以查看環境變數。如果您不在意環境變數的可見性，請設定 Name (名稱) 和 Value (值) 欄位，然後將 Type (類型) 設定為 Plaintext (純文字)。

建議您存放具有敏感值的環境變數，例如 AWS 存取金鑰 ID、AWS 私密存取金鑰或密碼，做為 Amazon EC2 Systems Manager 參數存放區或 中的參數 AWS Secrets Manager。

如果您使用 Amazon EC2 Systems Manager 參數存放區，則針對類型，選擇參數。針對名稱，輸入 CodeBuild 要參考的識別符。針對值，輸入儲存在 Amazon EC2 Systems Manager 參數存放區中的參數名稱。使用名為 /CodeBuild/dockerLoginPassword 的參數做為範例，針對 Type (類型)，選擇 Parameter (參數)。對於名稱，輸入 LOGIN_PASSWORD。針對數值，輸入 /CodeBuild/dockerLoginPassword。

Important

如果您使用 Amazon EC2 Systems Manager 參數存放區，建議您使用開頭為 的參數名稱來存放參數 /CodeBuild/ (例如，/CodeBuild/dockerLoginPassword)。您可以使用 CodeBuild 主控台在 Amazon EC2 Systems Manager 中建立參數。選擇 Create parameter (建立參數)，然後遵循對話方塊中的說明。(在該對話方塊中，對於 KMS 金鑰，您可以在帳戶中指定 AWS KMS 金鑰的 ARN。Amazon EC2 Systems Manager 使用此金鑰在儲存期間加密參數的值，並在擷取期間解密。) 如果您使用 CodeBuild 主控台建立參數，主控台會在儲存參數時以 /CodeBuild/ 啟動參數名稱。如需詳細資訊，請參閱《Amazon EC2 Systems Manager 使用者指南》中的 [Systems Manager 參數存放區](#) 和 [Systems Manager 參數存放區主控台演練](#)。

如果您的建置專案參考 Amazon EC2 Systems Manager 參數存放區中存放的參數，則建置專案的服務角色必須允許 ssm:GetParameters 動作。如果您先前選擇新服務角色，CodeBuild 會將此動作包含在建置專案的預設服務角色中。不過，如果您選擇 Existing service role (現有服務角色)，則您必須個別將此動作包含在服務角色中。如果您的建置專案參考存放在 Amazon EC2 Systems Manager 參數存放區中的參數，其參數名稱不是以 開頭 /CodeBuild/，而且您選擇了新服務角色，則必須更新該服務角色，以允許存取不是以 開頭的參數名稱 /CodeBuild/。這是因為該服務角色僅允許存取開頭為 /CodeBuild/ 的參數名稱。

如果您選擇新服務角色，服務角色會包含許可，以解密 Amazon EC2 Systems Manager 參數存放區中/CodeBuild/命名空間下的所有參數。

您設定的環境變數會取代現有環境變數。例如，如果 Docker 影像已包含名為 MY_VAR 且值為 my_value 的環境變數，而且您設定名為 MY_VAR 且值為 other_value 的環境變數，則 my_value 會取代為 other_value。同樣地，如果 Docker 影像已包含名為 PATH 且值為 /usr/local/sbin:/usr/local/bin 的環境變數，而且您設定名為 PATH 且值為 \$PATH:/usr/share/ant/bin 的環境變數，則 /usr/local/sbin:/usr/local/bin 會取代為文字值 \$PATH:/usr/share/ant/bin。

請不要設定名稱開頭為 CODEBUILD_ 的任何環境變數。此字首保留供 內部使用。

如果有多個位置定義同名的環境變數，則會決定值，如下所示：

- 開始建置操作呼叫中的值會採用最高優先順序。
- 組建專案定義中的值會採用下一個優先順序。
- buildspec 宣告中的值會採用最低優先順序。

如果您使用 Secrets Manager，請針對類型選擇 Secrets Manager。針對名稱，輸入 CodeBuild 要參考的識別符。針對 Value (值)，請、使用 *secret-id:json-key:version-stage:version-id* 模式輸入 reference-key。如需相關資訊，請參閱 [Secrets Manager reference-key in the buildspec file](#)。

Important

如果您使用 Secrets Manager，我們建議您存放名稱開頭為 的秘密 /CodeBuild/ (例如 /CodeBuild/dockerLoginPassword)。如需詳細資訊，請參閱《AWS Secrets Manager使用者指南》中的 [What Is AWS Secrets Manager](#)。

如果您的建置專案參考存放在 Secrets Manager 中的秘密，則建置專案的服務角色必須允許 secretsmanager:GetSecretValue 動作。如果您先前選擇新服務角色，CodeBuild 會將此動作包含在建置專案的預設服務角色中。不過，如果您選擇 Existing service role (現有服務角色)，則您必須個別將此動作包含在服務角色中。

如果您的建置專案參考存放在 Secrets Manager 中的秘密，其秘密名稱不是以 開頭/CodeBuild/，而且您選擇了新服務角色，則您必須更新服務角色，以允許存取不是以 開頭的秘密名稱/CodeBuild/。這是因為服務角色僅允許存取開頭為 的秘密名稱/CodeBuild/。

如果您選擇新服務角色，服務角色會包含許可，以解密 Secrets Manager 中/CodeBuild/命名空間下的所有秘密。

Buildspec

在 Buildspec 區段中，選擇編輯。變更完成後，請選擇更新組態以儲存新組態。

您可以修改下列屬性：

組建規格

執行以下任意一項：

- 如果您的來源碼包含 buildspec 檔案，請選擇 Use a buildspec file (使用 buildspec 檔案)。根據預設，CodeBuild buildspec.yml 會在來源碼根目錄中尋找名為 的檔案。如果您的 buildspec 檔案使用不同的名稱或位置，請在 Buildspec 名稱中輸入來源根的路徑（例如，buildspec-two.yml 或 configuration/buildspec.yml。如果 buildspec 檔案位於 S3 儲存貯體中，它必須與您的建置專案位於相同的 AWS 區域。使用其 ARN 指定 buildspec 檔案（例如，arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml）。
- 如果您的來源碼未包含 buildspec 檔案，或者，您要執行的組建命令與針對 build 階段 (位於來源碼根目錄的 buildspec.yml 檔案中) 所指定的組建命令不同，則請選擇 Insert build commands (插入組建命令)。針對 Build commands (組建命令)，在 build 階段中輸入您要執行的命令。針對多個命令，以 && 區隔每個命令 (例如，mvn test && mvn package)。若要在其他階段執行命令，或者如果您有該 build 階段的長命令清單，請將 buildspec.yml 檔案新增至來源碼根目錄，將命令新增至檔案，然後在來源碼根目錄中選擇使用 buildspec.yml。

如需更多資訊，請參閱[Buildspec 參考](#)。

批次組態

在批次組態區段中，選擇編輯。變更完成後，請選擇更新組態以儲存新組態。如需詳細資訊，請參閱[批次執行組建](#)。

您可以修改下列屬性：

批次服務角色

提供批次建置的服務角色。

選擇下列其中一項：

- 如果您沒有批次服務角色，請選擇新增服務角色。在服務角色中，輸入新角色的名稱。
- 如果您有批次服務角色，請選擇現有服務角色。在服務角色中，選擇服務角色。

批次建置在批次組態中導入新的安全性角色。此新角色是必要的，因為 CodeBuild 必須能夠代表您呼叫 StartBuild、StopBuild 和 RetryBuild 動作，才能在批次處理中執行組建。客戶應該使用新的角色，而不是與他們在組建中使用的角色相同，原因有兩個：

- 賦予建置角色 StartBuild、StopBuild 和 RetryBuild 權限，此將允許單一組建透過 buildspec 啟動更多組建。
- CodeBuild 批次建置提供限制，限制可用於批次中建置的建置和運算類型數目。如果組建角色具有這些權限，組建本身可能會略過這些限制。

批次允許的運算類型

選取批次允許的運算類型。選取所有適用項目。

批次允許的機群

選取批次允許的機群。選取所有適用項目。

批次允許的建置上限

輸入批次中允許的組建數目上限。如果批次超過此限制，批次將會失敗。

批次逾時

輸入批次建置完成的時間上限。

合併成品

選取將所有成品從批次合併到單一位置，讓批次中的所有成品合併到單一位置。

批次報告模式

選取批次組建所需的組建狀態報告模式。

Note

只有當專案來源為 Bitbucket、GitHub 或 GitHub Enterprise，且在來源下選取建置開始和完成時，向來源提供者報告建置狀態時，才能使用此欄位。

彙總組建

選取 ☐，讓批次中所有組建的狀態合併為單一狀態報告。

個別組建

選取 ☐，讓批次中所有組建的組建狀態分別報告。

成品

在成品區段中，選擇編輯。變更完成後，請選擇更新組態以儲存新組態。

您可以修改下列屬性：

類型

執行以下任意一項：

- 如果您不要建立任何建置輸出成品，則請選擇 No artifacts (無成品)。如果您只執行建置測試，或想要將 Docker 映像推送至 Amazon ECR 儲存庫，建議您執行此操作。
- 若要將建置輸出儲存在 S3 儲存貯體中，請選擇 Amazon S3，然後執行下列動作：
 - 如果您想要使用專案名稱做為組建輸出 ZIP 檔案或資料夾名稱，則請將 Name (名稱) 保留空白。否則請輸入名稱。(如果您想要輸出 ZIP 檔案，並且想要 ZIP 檔案有副檔名，則請務必將其包含在 ZIP 檔案名稱後面。)
 - 如果您想要 buildspec 檔案中所指定的名稱來覆寫主控台中所指定的任何名稱，請選取 Enable semantic versioning (啟用語意版本控制)。buildspec 檔案中的名稱是在建置時計算，並使用 Shell 命令語言。例如，您可以將日期和時間附加到成品名稱，讓它一律是唯一的。唯一成品名稱可防止覆寫成品。如需詳細資訊，請參閱[Buildspec 語法](#)。
 - 針對 Bucket name (儲存貯體名稱)，選擇輸出儲存貯體的名稱。
 - 如果您在本程序稍早選擇 Insert build commands (插入組建命令)，然後針對 Output files (輸出檔案)，輸入要放入組建輸出 ZIP 檔案或資料夾之組建中的檔案位置。針對多個位置，以逗號區隔每個位置 (例如，appspect.yml, target/my-app.jar)。如需詳細資訊，請參閱[Buildspec 語法](#)中的 files 描述。
 - 如果您不想要加密建置成品，請選取 Remove artifacts encryption (移除成品加密)。

針對您想要的每組次要成品：

1. 針對 Artifact identifier (成品識別符)，輸入的值少於 128 個字元，並且只包含英數字元和底線。
2. 選擇 Add artifact (新增成品)。
3. 遵循先前的步驟來設定您的次要成品。
4. 選擇 Save artifact (儲存成品)。

其他組態

加密金鑰

執行以下任意一項：

- 若要使用帳戶中的 AWS 受管金鑰 Amazon S3 來加密建置輸出成品，請將加密金鑰保留空白。這是預設值。
- 若要使用客戶受管金鑰來加密建置輸出成品，請在加密金鑰中輸入客戶受管金鑰的 ARN。使用 `arn:aws:kms:region-ID:account-ID:key/key-ID` 格式。

快取類型

針對 Cache type (快取類型)，選擇以下其中一項：

- 如果您不想要使用快取，請選擇 No cache (無快取)。
- 如果您想要使用 Amazon S3 快取，請選擇 Amazon S3，然後執行下列動作：
 - 針對 Bucket (儲存貯體)，選擇存放快取的 S3 儲存貯體名稱。
 - (選用) 對於快取路徑字首，輸入 Amazon S3 路徑字首。Cache path prefix (快取路徑字首) 值類似目錄名稱。它可讓您將快取存放至儲存貯體的相同目錄下方。

Important

請不要在路徑字首結尾附加尾端斜線 (/)。

- 如果您想要使用本機快取，請選擇 Local (本機)，然後選擇一或多個本機快取模式。

Note

「Docker layer cache」(Docker 層快取) 模式僅適用於 Linux。如果您選擇此模式，您的專案必須以特殊權限模式執行。

使用快取可節省大量建置時間，因為建置環境的可重複使用部分存放在快取中，並用於各建置。如需在 buildspec 檔案中指定快取的詳細資訊，請參閱[Buildspec 語法](#)。如需快取的詳細資訊，請參閱「[快取建置以改善效能](#)」。

日誌

在日誌區段中，選擇編輯。變更完成後，請選擇更新組態以儲存新組態。

您可以修改下列屬性：

選擇您要建立的日誌。您可以建立 Amazon CloudWatch Logs、Amazon S3 Logs 或兩者。

CloudWatch

如果您想要 Amazon CloudWatch Logs 日誌：

CloudWatch 日誌

選取 CloudWatch 日誌。

Group name (群組名稱)

輸入 Amazon CloudWatch Logs 日誌群組的名稱。

串流名稱

輸入您的 Amazon CloudWatch Logs 日誌串流名稱。

S3

如果您想要 Amazon S3 日誌：

S3 日誌

選取 S3 logs (S3 日誌)。

儲存貯體

選擇日誌的 S3 儲存貯體名稱。

路徑字首

輸入日誌的字首。

停用 S3 日誌加密

如果您不希望 S3 日誌加密，請選取。

變更建置專案的設定 (AWS CLI)

如需 AWS CLI 搭配使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

若要使用更新 CodeBuild 專案 AWS CLI，您可以建立具有更新屬性的 JSON 檔案，並將該檔案傳遞至 [update-project](#) 命令。更新檔案中未包含的任何屬性保持不變。

在更新 JSON 檔案中，只需要 name 屬性和修改過的屬性。name 屬性可識別要修改的專案。對於任何修改的結構，也必須包含這些結構的必要參數。例如，若要修改專案的環境，需要 environment/type 和 environment/computeType 屬性。以下是更新環境映像的範例：

```
{
  "name": "<project-name>",
  "environment": {
    "type": "LINUX_CONTAINER",
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/amazonlinux-x86_64-standard:4.0"
  }
}
```

如果您需要取得專案的目前屬性值，請使用 [batch-get-projects](#) 命令來取得您正在修改之專案的目前屬性，並將輸出寫入檔案。

```
aws codebuild batch-get-projects --names "<project-name>" > project-info.json
```

project-info.json 檔案包含一系列專案，因此無法直接用於更新專案。不過，您可以從 *project-info.json* 檔案複製要修改的屬性，並將其貼到您的更新檔案中，做為您要修改屬性的基準。如需詳細資訊，請參閱[檢視建置專案的詳細資訊 \(AWS CLI\)](#)。

如中所述修改更新 JSON 檔案[建立建置專案 \(AWS CLI\)](#)，並儲存結果。當您完成修改更新 JSON 檔案時，請執行 [update-project](#) 命令，傳遞更新 JSON 檔案。

```
aws codebuild update-project --cli-input-json file://<update-project-file>
```

如果成功，更新的專案 JSON 會出現在輸出中。如果缺少任何必要的參數，輸出中會顯示錯誤訊息，以識別缺少的參數。例如，如果缺少 environment/type 參數，則會顯示以下錯誤訊息：

```
aws codebuild update-project --cli-input-json file://update-project.json

Parameter validation failed:
Missing required parameter in environment: "type"
```

變更建置專案的設定 (AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs 和工具參考](#)。

CodeBuild 中的多個存取字符

CodeBuild 支援從 AWS Secrets Manager 中或透過 AWS CodeConnections 連線的秘密，將存取權杖來源至第三方供應商。您可以將秘密或連線設定為與指定第三方供應商互動的預設登入資料，例如 GitHub、GitHub Enterprise 或 Bitbucket。

您可以將來源登入資料設定為三個不同的層級：

1. 所有專案的帳戶層級登入資料：這些是 AWS 帳戶中所有專案的預設登入資料。當未指定專案或來源層級登入資料時，它們將用於專案。
2. 特定儲存庫的來源層級登入資料：這是在專案來源上定義 Secrets Manager 秘密或 AWS CodeConnections 連線時。這些登入資料只會用於指定來源儲存庫上的操作。這可讓您在相同專案中設定具有不同許可範圍的多個存取字符，而不使用預設帳戶層級登入資料。
3. 專案層級備用憑證：您可以使用 NO_SOURCE 做為主要來源類型來設定專案層級備用憑證，並在其中定義秘密或連線。當您在專案上有多個來源，但想要為它們使用相同的登入資料，或者您不想為專案使用預設帳戶層級登入資料時，請使用此選項。

主題

- [步驟 1：建立 Secrets Manager 秘密或 AWS CodeConnections 連線](#)
- [步驟 2：授予 CodeBuild 專案 IAM 角色對 Secrets Manager 秘密的存取權](#)
- [步驟 3：設定 Secrets Manager 或 AWS CodeConnections 權杖](#)
- [其他設定選項](#)

步驟 1：建立 Secrets Manager 秘密或 AWS CodeConnections 連線

使用下列指示來建立 Secrets Manager 秘密或 AWS CodeConnections 連線：

- [在 Secrets Manager 秘密中建立和存放權杖。](#)
- [建立 GitHub 的連線](#)
- [建立 GitHub Enterprise Server 的連線](#)
- [建立與 Bitbucket 的連線](#)

步驟 2：授予 CodeBuild 專案 IAM 角色對 Secrets Manager 秘密的存取權

Note

在繼續之前，您必須擁有在 Secrets Manager 或 中建立的權杖的存取權 AWS CodeConnections。

若要授予 CodeBuild 專案 IAM 角色對 Secrets Manager 或 的存取權 AWS CodeConnections，您必須新增下列 IAM 政策。

授予 CodeBuild 專案 IAM 角色存取權

1. 遵循 CodeBuild 專案的指示，為您的 CodeBuild 專案建立 [允許 CodeBuild 與其他 AWS 服務互動](#) IAM 角色。
2. 執行以下任意一項：
 - 將下列 IAM 政策新增至 CodeBuild 專案角色，以授予對秘密的存取權。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:iam::*:role/Service*"
      ]
    }
  ]
}
```

(選用) 如果您使用 AWS KMS 客戶受管金鑰來加密 Secrets Manager 秘密，您可以新增下列政策陳述式來授予存取權。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:iam::*:role/Service*",
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:SecretARN":
            "arn:aws:iam::*:role/Service*"
        }
      }
    }
  ]
}
```

- 將下列 IAM 政策新增至 CodeBuild 專案角色，以授予連線的存取權。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:GetConnectionToken",
        "codeconnections:GetConnection"
      ],
      "Resource": [
        "arn:aws:iam::*:role/Service*"
      ]
    }
  ]
}
```

步驟 3：設定 Secrets Manager 或 AWS CodeConnections 權杖

您可以使用 Secrets Manager 或 AWS CodeConnections 權杖，將來源登入資料設定為三個不同的層級。

將 Secrets Manager 或 AWS CodeConnections 權杖設定為帳戶層級登入資料

您可以將 Secrets Manager 秘密或 AWS CodeConnections 連線設定為帳戶層級登入資料，並在專案中使用。

AWS 管理主控台

若要在 中將連線設定為帳戶層級登入資料 AWS 管理主控台

1. 針對來源提供者，選擇 Bitbucket、GitHub 或 GitHub Enterprise。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇預設來源登入資料，以使用您帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線到來源提供者，請選擇管理預設來源憑證。
 - b. 針對登入資料類型，選擇登入資料類型。
 - c. 如果您選擇 AWS CodeConnections，請選擇使用現有的連線或建立新的連線。

如果您選擇不同的登入資料類型，對於 Service，請選擇您要用來存放字符的服務，並執行下列動作：

- 如果您選擇使用 Secrets Manager，您可以選擇使用現有的秘密連線或建立新的秘密，然後選擇儲存。如需如何建立新的秘密的詳細資訊，請參閱 [在 Secrets Manager 秘密中建立和存放權杖](#)。
- 如果您選擇使用 CodeBuild，請輸入您的字符或使用者名稱和應用程式密碼，然後選擇儲存。對於 Bitbucket API 字符，請改為輸入您的 Atlassian 帳戶電子郵件地址和 API 字符。
- 選擇自訂來源登入資料，以使用自訂來源登入資料覆寫您帳戶的預設設定。
 - a. 針對登入資料類型，選擇登入資料類型。
 - b. 在連線中，選擇使用現有的連線或建立新的連線。

AWS CLI

若要在 中將連線設定為帳戶層級登入資料 AWS CLI

- 開啟終端機 (Linux、macOS 或 Unix) 或命令提示 (Windows)。使用 AWS CLI 執行 `import-source-credentials` 命令。

使用下列命令來設定 Secrets Manager 秘密：

```
aws codebuild import-source-credentials \  
  --token "<secret-arn>" \  
  --server-type <source-provider> \  
  --auth-type SECRETS_MANAGER \  
  --region <aws-region>
```

使用下列命令來設定 AWS CodeConnections 連線：

```
aws codebuild import-source-credentials \  
  --token "<connection-arn>" \  
  --server-type <source-provider> \  
  --auth-type CODECONNECTIONS \  
  --region <aws-region>
```

此命令可讓您匯入字符做為帳戶層級預設來源憑證。當您使用 [ImportSourceCredentials](#) API 匯入登入資料時，除非已在專案中設定更具體的登入資料集，否則 CodeBuild 將使用權杖與來源提供者的所有互動，例如 Webhook、建置狀態報告和 git 複製操作。

您現在可以在建置專案中使用權杖並執行它。如需詳細資訊，請參閱 [在 中建立建置專案 AWS CodeBuild](#) 及 [手動執行 AWS CodeBuild 組建](#)。

將多個字符設定為來源層級登入資料

若要使用 Secrets Manager 秘密或 AWS CodeConnections 連線做為來源層級登入資料，請直接參考 CodeBuild 專案中的字符，然後啟動組建。

AWS 管理主控台

若要在 中將多個字符設定為來源層級登入資料 AWS 管理主控台

- 針對 Source provider (來源供應商)，選擇 GitHub。

2. 對於登入資料，請執行下列其中一項操作：
 - 選擇預設來源登入資料，以使用您帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線至 GitHub，請選擇管理預設來源憑證。
 - b. 針對登入資料類型，選擇 GitHub 應用程式。
 - c. 在連線中，選擇使用現有的連線或建立新的連線。
 - 選擇自訂來源登入資料，以使用自訂來源登入資料覆寫您帳戶的預設設定。
 - a. 針對登入資料類型，選擇 GitHub 應用程式。
 - b. 在連線中，選擇使用現有的連線或建立新的連線。
3. 選擇新增來源，然後重複選擇來源提供者和登入資料的程序。

AWS CLI

若要在 中將多個字符設定為來源層級登入資料 AWS CLI

- 開啟終端機 (Linux、macOS 或 Unix) 或命令提示 (Windows)。使用 AWS CLI 執行 create-project 命令。

使用下列命令：

```
aws codebuild create-project --region <aws-region> \
  --name <project-name> \
  --artifacts type=NO_ARTIFACTS \
  --environment "type=LINUX_CONTAINER,
                  computeType=BUILD_GENERAL1_SMALL,
                  image=aws/codebuild/amazonlinux-x86_64-standard:5.0" \
  --service-role <service-role-name> \
  --source "type=GITHUB,
            location=<github-repository-1>,
            auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-1>}" \
  --secondary-sources "type=GITHUB,
                       location=<github-repository-2>,
                       auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-2>},
                       sourceIdentifier=secondary"

aws codebuild start-build --region <aws-region> --project-name <project-name>
```

設定專案層級來源憑證備用

若要設定專案層級來源憑證備用，請將 NO_SOURCE 用於專案的主要來源，並參考字符。

```
aws codebuild create-project \  
  --name <project-name> \  
  --service-role <service-role-name> \  
  --artifacts type=NO_ARTIFACTS \  
  --environment "type=LINUX_CONTAINER,  
                 computeType=BUILD_GENERAL1_SMALL,  
                 image=aws/codebuild/amazonlinux-x86_64-standard:5.0" \  
  --service-role <service-role-name> \  
  --source "type=NO_SOURCE,  
            auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn>},  
            buildspec=<buildspec>"  
  --secondary-sources "type=GITHUB,  
                      location=<github-repository>,  
                      sourceIdentifier=secondary"  
  
aws codebuild start-build --region <aws-region> --project-name <project_name>
```

使用時 NO_SOURCE，通常會在來源模型內提供 buildspec，因為它不會直接設定為使用外部來源來擷取 [buildspec](#)。通常，NO_SOURCE 來源會處理從 buildspec 內複製所有相關儲存庫。為了確保設定的登入資料可供這些操作使用，您可以在 buildspec 中啟用 git-credential-helper 選項。

```
env:  
  git-credential-helper: yes
```

在建置期間，CodeBuild 會從設定的權杖讀取 AuthServer 欄位，並針對對該特定第三方來源提供者的所有 git 請求使用權杖憑證。

其他設定選項

您可以使用 CloudFormation 範本來設定 Secrets Manager 帳戶層級登入資料。您可以使用下列 CloudFormation 範本來設定帳戶層級登入資料：

```
Parameters:  
  GitHubToken:  
    Type: String  
    NoEcho: true  
    Default: placeholder  
Resources:
```

```
CodeBuildAuthTokenSecret:
  Type: AWS::SecretsManager::Secret
  Properties:
    Description: CodeBuild auth token
    Name: codebuild-auth-token
    SecretString:
      !Join
      - '-'
      - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":'
        - '!Ref GitHubToken
        - '"}'
```

Note

如果您也要在相同的堆疊中建立專案，請使用 CloudFormation 屬性 [DependsOn](#)，以確保 AccountCredential 在專案之前建立。

您也可以使用 CloudFormation 範本設定 Secrets Manager 多個來源層級登入資料。您可以使用下列 CloudFormation 範本，使用多個字符來提取多個來源：

```
Parameters:
  GitHubTokenOne:
    Type: String
    NoEcho: true
    Default: placeholder
  GitHubTokenTwo:
    Type: String
    NoEcho: true
    Default: placeholder
```



```
Resources:
  CodeBuildSecretsManagerProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Name: codebuild-multitoken-example
      ServiceRole: <service-role>
      Environment:
        Type: LINUX_CONTAINER
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux-x86_64-standard:5.0
      Source:
        Type: GITHUB
        Location: <github-repository-one>
        Auth:
          Type: SECRETS_MANAGER
          Resource: !Ref CodeBuildAuthTokenSecretOne
      SecondarySources:
        - Type: GITHUB
          Location: <github-repository-two>
          Auth:
            Type: SECRETS_MANAGER
            Resource: !Ref CodeBuildAuthTokenSecretTwo
          SourceIdentifier: secondary
      Artifacts:
        Type: NO_ARTIFACTS
      LogsConfig:
        CloudWatchLogs:
          Status: ENABLED
  CodeBuildProjectIAMRoleSecretAccess:
    Type: AWS::IAM::RolePolicy
    Properties:
      RoleName: <role-name>
      PolicyName: CodeBuildProjectIAMRoleSecretAccessPolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action:
              - secretsmanager:GetSecretValue
            Resource:
              - !Ref CodeBuildAuthTokenSecretOne
              - !Ref CodeBuildAuthTokenSecretTwo
  CodeBuildAuthTokenSecretOne:
    Type: AWS::SecretsManager::Secret
```

```

Properties:
  Description: CodeBuild auth token one
  Name: codebuild-auth-token-one
  SecretString:
    !Join
    - ''
    - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":""'
      - !Ref GitHubTokenOne
      - '"}'
  Tags:
    - Key: codebuild:source:provider
      Value: github
    - Key: codebuild:source:type
      Value: personal_access_token
CodeBuildAuthTokenSecretTwo:
  Type: AWS::SecretsManager::Secret
  Properties:
    Description: CodeBuild auth token two
    Name: codebuild-auth-token-two
    SecretString:
      !Join
      - ''
      - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":""'
        - !Ref GitHubTokenTwo
        - '"}'
    Tags:
      - Key: codebuild:source:provider
        Value: github
      - Key: codebuild:source:type
        Value: personal_access_token

```

在 中刪除組建專案 AWS CodeBuild

您可以使用 CodeBuild 主控台、AWS CLI、AWS SDKs 來刪除 CodeBuild 中的組建專案。如果您刪除專案，不會刪除其建置。

Warning

您無法刪除包含建置與資源政策的專案。若要刪除包含資源政策與建置的專案，您必須先移除資源政策並刪除其建置。

主題

- [刪除建置專案 \(主控台\)](#)
- [刪除建置專案 \(AWS CLI\)](#)
- [刪除組建專案AWS SDKs\)](#)

刪除建置專案 (主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 執行以下任意一項：
 - 選擇您希望刪除之建置專案旁的選項按鈕，然後選擇 Delete (刪除)。
 - 選擇您希望刪除的組建專案連結，然後選擇 Delete (刪除)。

Note

根據預設，只會顯示最新的 10 個組建專案。若要檢視更多組建專案，請在 Projects per page (每頁顯示專案數) 中選擇其他值，或選擇用於檢視專案的向前和向後箭頭。

刪除建置專案 (AWS CLI)

1. 執行 delete-project 命令：

```
aws codebuild delete-project --name name
```

取代下列預留位置：

- *name*：必要字串。欲刪除的建置專案名稱。若要取得可用的建置專案清單，請執行 list-projects 命令。如需詳細資訊，請參閱[檢視建置專案名稱清單 \(AWS CLI\)](#)。
2. 如果成功，則在輸出中不會出現任何資料或錯誤。

如需 AWS CLI 搭配使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

刪除組建專案AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

取得公有建置專案 URLs

AWS CodeBuild 可讓您將建置專案的建置結果、日誌和成品提供給一般大眾。這可讓來源儲存庫的參與者檢視結果並下載組建的成品，而不需要他們存取 AWS 帳戶。

當您讓專案的組建可供公有使用時，所有專案的組建結果、日誌和成品，包括在專案私有時執行的組建都會提供給公有。同樣地，當您將公有建置專案設為私有時，該專案的建置結果將不再可供公有使用。

如需如何變更專案建置結果公有可見性的詳細資訊，請參閱 [啟用公有建置存取](#)。

CodeBuild 為您的專案提供專屬於您專案的公有建置 URL。

若要取得建置專案的公有 URL，請使用下列程序。

取得公有建置專案的 URL

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 選擇您要取得公有 URL 之建置專案的連結。
4. 公有 URL 會顯示在組態區段的公有專案 URL 欄位中。您可以選擇連結來開啟 URL，或使用複製按鈕複製 URL。

Warning

在公開專案的建置結果時，請注意下列事項：

- 所有專案的建置結果、日誌和成品，包括在專案私有時執行的建置，都可供大眾使用。
- 所有建置日誌和成品都可供大眾使用。環境變數、原始程式碼和其他敏感資訊可能已輸出至建置日誌和成品。您必須小心將哪些資訊輸出到建置日誌。一些最佳實務如下：
 - 請勿在環境變數中存放敏感值，特別是 AWS 存取金鑰 IDs和私密存取金鑰。建議您使用 Amazon EC2 Systems Manager 參數存放區或 AWS Secrets Manager 來存放敏感值。
 - 遵循 [使用 Webhook 的最佳實務](#) 來限制哪些實體可以觸發組建，並且不要將 buildspec 存放在專案本身中，以確保您的 Webhook 盡可能安全。

- 惡意使用者可以使用公有組建來分發惡意成品。我們建議專案管理員檢閱所有提取請求，以確認提取請求合法的變更。我們也建議您使用其檢查總和驗證任何成品，以確保下載正確的成品。

共用建置專案

專案共用可讓專案擁有者與其他 AWS 帳戶或使用者共用其 AWS CodeBuild 專案。在此模型中，擁有專案的帳戶 (擁有者) 將專案分享給其他帳戶 (消費者)。消費者無法編輯或執行專案。

主題

- [共用專案](#)
- [相關服務](#)
- [存取與您共用的 CodeBuild 專案](#)
- [取消共用共用專案](#)
- [識別共用專案](#)
- [共用的專案許可](#)

共用專案

消費者可以使用 AWS CLI 和 AWS CodeBuild 主控台來檢視您共用的專案和組建。消費者無法編輯或執行專案。

您可以將專案新增至現有的資源分享，或在 [AWS RAM 主控台](#) 中建立資源分享。

Note

您無法刪除其建置已新增至資源分享的專案。

若要與組織單位或整個組織共用專案，必須啟用透過 AWS Organizations 來共用的功能。如需詳細資訊，請參閱《AWS RAM 使用者指南》中的[透過 AWS Organizations 啟用共用](#)。

您可以使用 AWS CodeBuild 主控台、AWS RAM 主控台或 AWS CLI 來共用您擁有的專案。

共用專案的先決條件

開始共用專案之前，請確定 AWS 您的帳戶擁有該專案。您無法將已分享給您的專案再分享出去。

共用您擁有的專案 (CodeBuild 主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。

Note

根據預設，只會顯示最新的 10 個組建專案。若要檢視更多組建專案，請選擇齒輪圖示，然後針對 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

3. 選擇您要共用的專案，然後選擇 Share (共用)。如需詳細資訊，請參閱《AWS RAM 使用者指南》中的 [建立資源共用](#)。

共用您擁有的專案AWS RAM (主控台)

請參閱 AWS RAM 使用者指南中的 [建立資源共用](#)。

共用您擁有的專案 (AWS RAM 命令)

使用 [create-resource-share](#) 命令。

共用您擁有的專案 (CodeBuild 命令)

使用 [put-resource-policy](#) 命令：

1. 建立名為 policy.json 的檔案，並將以下命令複製到其中。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "codebuild:BatchGetProjects",
      "codebuild:BatchGetBuilds",
      "codebuild:ListBuildsForProject"],
    "Resource": "arn:aws:iam::*:role/Service*"
  ]
}
```

```
}
```

2. 使用專案 ARN 和識別符更新 `policy.json`，以與之共用。下列範例會將唯讀存取權授予 123456789012 所識別 AWS 帳戶的根使用者。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "123456789012"
      ]
    },
    "Action": [
      "codebuild:BatchGetProjects",
      "codebuild:BatchGetBuilds",
      "codebuild:ListBuildsForProject"],
    "Resource": "arn:aws:codebuild:us-west-2:123456789012:project/my-project"
  ]
}
```

3. 執行 [put-resource-policy](#) 命令。

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. 取得 AWS RAM 資源共享 ARN。

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

這將傳回類似以下的回應：

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
```

```
    "creationTime": "<creation-time>",
    "lastUpdatedTime": "<last-update-time>"
  }
]
```

從回應中，複製要在下一個步驟中使用的 *<resource-share-arn>* 值。

5. 執行 AWS RAM [promote-resource-share-created-from-policy](#) 命令。

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-
share-arn>
```

相關服務

專案共用與 AWS Resource Access Manager (AWS RAM) 整合，這項服務可讓您與任何 AWS 帳戶或透過 共用 AWS 資源 AWS Organizations。有了 AWS RAM，您可以建立資源共用，指定資源及要與之分享的消費者，以分享資源。消費者可以是 中的個別 AWS 帳戶 AWS Organizations、組織單位或整個組織 AWS Organizations。

如需詳細資訊，請參閱「[AWS RAM 使用者指南](#)」。

存取與您共用的 CodeBuild 專案

若要存取共用專案，取用者的 IAM 角色需要 BatchGetProjects 許可。您可以將下列政策連接至其 IAM 角色：

```
{
  "Effect": "Allow",
  "Resource": [
    "*"
  ],
  "Action": [
    "codebuild:BatchGetProjects"
  ]
}
```

如需詳細資訊，請參閱[使用的身分型政策 AWS CodeBuild](#)。

取消共用專案

只有其擁有者可以存取已取消共享的專案 (包括其建置)。如果您取消共用專案，您之前與其共用的任何 AWS 帳戶或使用者都無法存取專案或其組建。

若要取消共享您擁有的已共用專案，您必須從資源共享中移除它。您可以使用 AWS CodeBuild 主控台、AWS RAM 主控台或 AWS CLI 來執行此操作。

取消共用您擁有的共用專案AWS RAM (主控台)

請參閱《AWS RAM 使用者指南》中的[更新資源共享](#)。

取消共享您擁有的共用專案 (AWS CLI)

使用 [disassociate-resource-share](#) 命令。

取消共用您擁有的專案 (CodeBuild 命令)

執行 [delete-resource-policy](#) 命令並指定您要取消共享之專案的 ARN：

```
aws codebuild delete-resource-policy --resource-arn project-arn
```

識別共用專案

擁有者和消費者可以使用 AWS CLI 來識別共用專案。

識別與 AWS 您的帳戶或使用者共用的專案 (AWS CLI)

使用 [list-shared-projects](#) 命令執行已與您共用的專案。

共用的專案許可

擁有者的許可

專案擁有者可以編輯專案，並用它執行建置。

消費者的許可

專案消費者可以檢視專案及其建置，但無法編輯專案或使用它來執行建置。

標記建置專案

標籤是您或 AWS 指派給 AWS 資源的自訂屬性標籤。每個 AWS 標籤有兩個部分：

- 標籤鍵 (例如, CostCenter、Environment、Project 或 Secret)。標籤鍵會區分大小寫。
- 一個名為標籤值 (例如, 111122223333、Production 或團隊名稱) 的選用欄位。忽略標籤值基本上等同於使用空字串。與標籤鍵相同, 標籤值會區分大小寫。

這些合稱為鍵值組。如需專案可擁有的標籤數目, 以及標籤金鑰和值的限制的相關資訊, 請參閱[Tags \(標籤\)](#)。

標籤可協助您識別和組織 AWS 資源。許多 AWS 服務支援標記, 因此您可以將相同的標籤指派給來自不同服務的資源, 以指出資源相關。例如, 您可以將相同的標籤指派給指派給 S3 儲存貯體的 CodeBuild 專案。如需使用標籤的詳細資訊, 請參閱[標記最佳實務](#)。

在 CodeBuild 中, 主要資源是專案和報告群組。您可以使用 CodeBuild 主控台、AWS CLI、CodeBuild APIs 或 AWS SDKs 來新增、管理和移除專案的標籤。除了使用標籤識別、組織和追蹤專案之外, 您還可以在 IAM 政策中使用標籤, 以協助控制誰可以檢視專案並與之互動。如需以標籤為基礎的存取政策範例, 請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

Important

使用預留容量功能時, 同一帳戶中的其他專案可以存取機群執行個體上快取的資料, 包括來源檔案、Docker layer 和 buildspect 中指定的快取目錄。這是根據設計, 並允許相同帳戶中的專案共用機群執行個體。

主題

- [將標籤新增至專案](#)
- [檢視專案的標籤](#)
- [編輯專案的標籤](#)
- [從專案移除標籤](#)

將標籤新增至專案

將標籤新增至專案可協助您識別和組織資源, 並管理對這些 AWS 資源的存取。首先, 將一或多個標籤 (金鑰值對) 新增到專案。請記住, 專案的標籤數目有所限制。金鑰和值欄位可使用的字數有所限制。如

需詳細資訊，請參閱[Tags \(標籤\)](#)。擁有標籤後，您可以建立 IAM 政策，以根據這些標籤管理對專案的存取。您可以使用 CodeBuild 主控台或 AWS CLI 將標籤新增至專案。

Important

使用預留容量功能時，同一帳戶中的其他專案可以存取機群執行個體上快取的資料，包括來源檔案、Docker layer 和 buildspec 中指定的快取目錄。這是根據設計，並允許相同帳戶中的專案共用機群執行個體。

如需在建立政策時，將標籤新增至專案的詳細資訊，請參閱[將標籤新增至專案 \(主控台\)](#)。

Important

將標籤新增至專案之前，請務必檢閱任何可能使用標籤來控制資源存取的 IAM 政策，例如建置專案。如需以標籤為基礎的存取政策範例，請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

主題

- [將標籤新增至專案 \(主控台\)](#)
- [將標籤新增至專案 \(AWS CLI\)](#)

將標籤新增至專案 (主控台)

您可以使用 CodeBuild 主控台，將一或多個標籤新增至 CodeBuild 專案。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Build projects (組建專案) 中，選擇您要新增標籤的專案名稱。
3. 在導覽窗格中，選擇設定。選擇 Build project tags (組建專案標籤)。
4. 如果沒有任何標籤新增至專案，請選擇 Add tag (新增標籤)。否則，選擇 Edit (編輯)，然後選擇 Add tag (新增標籤)。
5. 在 Key (金鑰) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。
6. (選用) 若要新增另一個標籤，再選擇 Add tag (新增標籤) 一次。
7. 當您完成新增標籤的作業時，請選擇 Submit (提交)。

將標籤新增至專案 (AWS CLI)

若要在建立專案時，將標籤新增到專案，請參閱[建立建置專案 \(AWS CLI\)](#)。在 `create-project.json` 中，新增您的標籤。

在這些步驟中，我們假設您已經安裝新版 AWS CLI 或更新到最新版本。如需詳細資訊，請參閱[安裝 AWS Command Line Interface](#)。

若成功，此命令不會傳回任何內容。

檢視專案的標籤

標籤可協助您識別和組織 AWS 資源，並管理對資源的存取。如需使用標籤的詳細資訊，請參閱[標記最佳實務](#)白皮書。如需以標籤為基礎的存取政策範例，請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

檢視專案的標籤 (主控台)

您可以使用 CodeBuild 主控台來檢視與 CodeBuild 專案相關聯的標籤。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Build projects (組建專案) 中，選擇您要檢視標籤的專案名稱。
3. 在導覽窗格中，選擇設定。選擇 Build project tags (組建專案標籤)。

檢視專案的標籤 (AWS CLI)

若要檢視組建專案的標籤，請執行以下命令。針對 `--names` 參數使用您的專案名稱。

```
aws codebuild batch-get-projects --names your-project-name
```

如果成功，此命令會傳回 JSON 格式的資訊，其中包含類似如下的組建專案相關資訊：

```
{
  "tags": {
    "Status": "Secret",
    "Team": "JanesProject"
  }
}
```

如果專案沒有標籤，則 tags 區段為空白：

```
"tags": []
```

編輯專案的標籤

您可以變更與專案相關聯的標籤值。您也可以變更金鑰的名稱，這相當於移除目前的標籤，並新增具有新名稱和與其他金鑰相同值的不同標籤。請記住金鑰和值欄位可使用的字數有所限制。如需詳細資訊，請參閱[Tags \(標籤\)](#)。

Important

編輯專案的標籤可能會影響該專案的存取權。在編輯專案標籤的名稱（索引鍵）或值之前，請務必檢閱任何可能使用標籤索引鍵或值的 IAM 政策，以控制對建置專案等資源的存取。如需以標籤為基礎的存取政策範例，請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

編輯專案的標籤 (主控台)

您可以使用 CodeBuild 主控台來編輯與 CodeBuild 專案相關聯的標籤。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Build projects (組建專案) 中，選擇您要編輯標籤的專案名稱。
3. 在導覽窗格中，選擇設定。選擇 Build project tags (組建專案標籤)。
4. 選擇編輯。
5. 執行以下任意一項：
 - 若要變更標籤，請在 Key (金鑰) 輸入新的名稱。變更標籤名稱等於移除標籤並使用新的金鑰名稱新增一個新的標籤。
 - 若要變更標籤的值，請輸入新的值。如果您想要變更值為沒有，請刪除目前的值並保留欄位空白。
6. 當您完成編輯標籤，選擇 Submit (提交)。

編輯專案的標籤 (AWS CLI)

若要新增、變更或刪除組建專案的標籤，請參閱[變更建置專案的設定 \(AWS CLI\)](#)。更新您用於更新專案之 JSON 格式資料中的 tags 區段。

從專案移除標籤

您可以移除一或多個與專案相關聯的標籤。移除標籤不會從與該標籤相關聯的其他 AWS 資源中刪除標籤。

Important

移除專案的標籤可能會影響該專案的存取權。從專案移除標籤之前，請務必檢閱任何可能使用標籤金鑰或值的 IAM 政策，以控制對建置專案等資源的存取。如需以標籤為基礎的存取政策範例，請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

從專案移除標籤 (主控台)

您可以使用 CodeBuild 主控台來移除標籤與 CodeBuild 專案之間的關聯。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Build projects (組建專案) 中，選擇您要移除標籤的專案名稱。
3. 在導覽窗格中，選擇設定。選擇 Build project tags (組建專案標籤)。
4. 選擇編輯。
5. 尋找您想要移除的標籤，然後選擇 Remove tag (移除標籤)。
6. 當您完成移除標籤，請選擇 Submit (提交)。

從專案移除標籤 (AWS CLI)

若要將一或多個標籤從組建專案中刪除，請參閱[變更建置專案的設定 \(AWS CLI\)](#)。使用未包含您要刪除之標籤的已更新標籤清單，來更新 JSON 格式資料中的 tags 區段。如果您要刪除所有標籤，請將 tags 區段更新為：

```
"tags: []"
```

Note

如果您刪除 CodeBuild 組建專案，所有標籤關聯都會從已刪除的組建專案中移除。您不需要在刪除組建專案之前移除標籤。

搭配 使用執行器 AWS CodeBuild

AWS CodeBuild 支援與 GitHub Actions 執行器、自我管理的 GitLab 執行器和 Buildkite 執行器整合。

主題

- [中的自我託管 GitHub 動作執行器 AWS CodeBuild](#)
- [中的自我管理 GitLab 執行器 AWS CodeBuild](#)
- [中的自我管理 Buildkite 執行器 AWS CodeBuild](#)

中的自我託管 GitHub 動作執行器 AWS CodeBuild

您可以設定專案在 CodeBuild 容器中設定自我託管的 GitHub Actions 執行器，以處理您的 GitHub Actions 工作流程任務。這可以透過使用 CodeBuild 專案設定 Webhook，並更新您的 GitHub Actions 工作流程 YAML 來使用 CodeBuild 機器上託管的自我託管執行器來完成。

設定 CodeBuild 專案以執行 GitHub Actions 任務的高階步驟如下：

1. 如果您尚未這麼做，請建立個人存取字符或連線至 OAuth 應用程式，將您的專案連線至 GitHub。
2. 導覽至 CodeBuild 主控台，並使用 Webhook 建立 CodeBuild 專案，並設定您的 Webhook 篩選條件。
3. 在 GitHub 中更新您的 GitHub 動作工作流程 YAML，以設定您的建置環境。

如需更詳細的程序，請參閱 [教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。

此功能可讓您的 GitHub Actions 工作流程任務取得與 的原生整合 AWS，透過 IAM AWS Secrets Manager AWS CloudTrail、整合和 Amazon VPC 等功能提供安全性和便利性。您可以存取最新的執行個體類型，包括以 ARM 為基礎的執行個體。

主題

- [關於 CodeBuild 託管的 GitHub 動作執行器](#)
- [教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)
- [Webhook 故障診斷](#)
- [CodeBuild 託管的 GitHub 動作執行器支援標籤覆寫](#)
- [CodeBuild 託管的 GitHub Actions Runner 支援運算映像](#)

關於 CodeBuild 託管的 GitHub 動作執行器

以下是有關 CodeBuild 託管的 GitHub 動作執行器的一些常見問題。

我應該何時在標籤中包含映像和執行個體覆寫？

您可以在 標籤 中包含映像和執行個體覆寫，以便為每個 GitHub 動作工作流程任務指定不同的建置環境。這可以完成，無需建立多個 CodeBuild 專案或 Webhook。例如，當您需要 [為工作流程任務使用矩陣](#) 時，這很有用。

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      image:${{ matrix.os }}
      instance-size:${{ matrix.size }}
    strategy:
      matrix:
        include:
          - os: arm-3.0
            size: small
          - os: linux-5.0
            size: large
    steps:
      - run: echo "Hello World!"
```

Note

如果 runs-on 有多個包含 GitHub 動作內容的標籤，則可能需要引號。

我可以將 CloudFormation 用於此功能嗎？

是，您可以在 CloudFormation 範本中包含篩選條件群組，在專案 Webhook 中指定 GitHub 動作工作流程任務事件篩選條件。

```
Triggers:
  Webhook: true
  FilterGroups:
```



```
- - Type: EVENT
  Pattern: WORKFLOW_JOB_QUEUED
```

如需詳細資訊，請參閱[篩選 GitHub Webhook 事件 \(CloudFormation\)](#)。

如果您需要在 CloudFormation 範本中設定專案登入資料的說明，請參閱AWS CloudFormation 《使用者指南》中的 [AWS::CodeBuild::SourceCredential](#) 以取得詳細資訊。

使用此功能時，如何遮罩秘密？

根據預設，不會遮罩在日誌中列印的秘密。如果您想要遮罩秘密，您可以使用下列語法：`::add-mask::value`。以下是如何在 YAML 中使用此語法的範例：

```
name: Secret Job
on: [push]
jobs:
  Secret-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    env:
      SECRET_NAME: "secret-name"
    steps:
      - run: echo "::add-mask::$SECRET_NAME"
```

如需詳細資訊，請參閱在 GitHub 上[遮罩日誌中的值](#)。

我可以從單一專案中的多個儲存庫接收 GitHub Actions Webhook 事件嗎？

CodeBuild 支援從指定組織或企業接收事件的組織和全域層級 Webhook。如需詳細資訊，請參閱[GitHub 全域和組織 Webhook](#)。

哪些區域支援使用 CodeBuild 託管的 GitHub Actions 執行器？

所有 CodeBuild 區域都支援 CodeBuild 託管的 GitHub Actions 執行器。如需 CodeBuild 可用 AWS 區域位置的詳細資訊，請參閱[AWS 依區域提供服務](#)。

哪些平台支援使用 CodeBuild 託管的 GitHub Actions 執行器？

Amazon EC2 和[AWS Lambda](#)運算都支援 CodeBuild 託管的 GitHub Actions 執行器。您可以使用下列平台：Amazon Linux 2、Amazon Linux 2023、Ubuntu 和 Windows Server Core 2019。如需詳細資訊，請參閱[EC2 運算映像](#)及[Lambda 運算映像](#)。

教學課程：設定 CodeBuild 託管的 GitHub 動作執行器

本教學課程說明如何設定 CodeBuild 專案以執行 GitHub Actions 任務。如需搭配 CodeBuild 使用 GitHub 動作的詳細資訊，請參閱 [教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。

若要完成此教學課程，您必須先：

- 使用個人存取字符、Secrets Manager 秘密、OAuth 應用程式或 GitHub 應用程式連線。如果您想要與 OAuth 應用程式連線，您必須使用 CodeBuild 主控台來執行此操作。如果您想要建立個人存取字符，您可以使用 CodeBuild 主控台或使用 [ImportSourceCredentials API](#)。如需更多說明，請參閱 [CodeBuild 中的 GitHub 和 GitHub Enterprise Server 存取](#)。
- 將 CodeBuild 連接至您的 GitHub 帳戶。若要這麼做，您可以執行下列其中一項操作：
 - 您可以在主控台中新增 GitHub 做為來源提供者。您可以使用個人存取字符、Secrets Manager 秘密、OAuth 應用程式或 GitHub 應用程式來連線。如需說明，請參閱 [CodeBuild 中的 GitHub 和 GitHub Enterprise Server 存取](#)。
 - 您可以透過 [ImportSourceCredentials API](#) 匯入 GitHub 登入資料。這只能使用個人存取字符來完成。如果您使用 OAuth 應用程式進行連線，則必須改用主控台進行連線。如需說明，請參閱 [使用存取字符 \(CLI\) 連接 GitHub](#)。

Note

只有在您尚未連線到您帳戶的 GitHub 時，才需要這麼做。

步驟 1：使用 Webhook 建立 CodeBuild 專案

在此步驟中，您將使用 Webhook 建立 CodeBuild 專案，並在 GitHub 主控台中檢閱。您也可以選擇 GitHub Enterprise 做為來源提供者。若要進一步了解如何在 GitHub Enterprise 中建立 Webhook，請參閱 [GitHub 手動 Webhook](#)。

使用 Webhook 建立 CodeBuild 專案

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
3. 在專案類型中，選擇執行器專案。

在執行器中：

- a. 針對執行器提供者，選擇 GitHub。
- b. 針對執行器位置，選擇儲存庫。
- c. 針對儲存庫下的儲存庫 URL，選擇 `https://github.com/user-name/repository-name`。

Note

根據預設，您的專案只會接收單一儲存庫 `WORKFLOW_JOB_QUEUED` 的事件。如果您想要接收組織或企業內所有儲存庫的事件，請參閱 [GitHub 全域和組織 Webhook](#)。

4.
 - 在 Environment (環境) 中：
 - 選擇支援的環境映像和運算。請注意，您可以選擇使用 GitHub Actions 工作流程 YAML 中的標籤覆寫映像和執行個體設定。如需詳細資訊，請參閱 [步驟 2：更新您的 GitHub 動作工作流程 YAML](#)
 - 在 Builds spec (建置規格) 中：
 - 請注意，除非新增為標籤，否則您的 `buildspec buildspec-override:true` 將被忽略。反之，CodeBuild 會覆寫它，以使用將設定自我託管執行器的命令。
5. 繼續執行預設值，然後選擇建立建置專案。
6. 在開啟 GitHub 主控台 `https://github.com/user-name/repository-name/settings/hooks`，確認 webhook 已建立並啟用，以交付工作流程任務事件。

步驟 2：更新您的 GitHub 動作工作流程 YAML

在此步驟中，您將在 [中更新 GitHub Actions 工作流程 YAML 檔案](#) [GitHub](#)，以設定您的建置環境，並在 CodeBuild 中使用 GitHub Actions 自我託管執行器。如需詳細資訊，請參閱 [搭配自我託管執行器使用標籤](#)和 [CodeBuild 託管的 GitHub 動作執行器支援標籤覆寫](#)。

更新您的 GitHub 動作工作流程 YAML

導覽 [GitHub](#) 至 GitHub 動作工作流程 YAML 並更新 `runs-on` 設定，以設定您的建置環境。若要這麼做，您可以執行下列其中一項操作：

- 您可以指定專案名稱和執行 ID，在這種情況下，組建將使用您現有的專案組態進行運算、映像、映像版本和執行個體大小。需要專案名稱，才能將 GitHub Actions 任務的 AWS 相關設定連結至特定的 CodeBuild 專案。透過在 YAML 中包含專案名稱，CodeBuild 可以使用正確的專案設定调用任務。透過提供執行 ID，CodeBuild 會將您的組建映射至特定工作流程執行，並在工作流程執行取消時停止組建。如需詳細資訊，請參閱 [github 內容](#)。

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
```

Note

請確定您的 *<project-name>* 符合您在上一個步驟中建立的專案名稱。如果不相符，CodeBuild 不會處理 Webhook，GitHub 動作工作流程可能會停止。

以下是 GitHub 動作工作流程 YAML 的範例：

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    steps:
      - run: echo "Hello World!"
```

- 您也可以在此標籤中覆寫映像和運算類型。[CodeBuild 託管的 GitHub Actions Runner 支援運算映像](#) 如需精選影像的清單，請參閱。如需使用自訂映像，請參閱 [CodeBuild 託管的 GitHub 動作執行器支援標籤覆寫](#)。標籤中的運算類型和映像會覆寫專案上的環境設定。若要覆寫 CodeBuild EC2 或 Lambda 運算組建的环境設定，請使用下列語法：

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    image:<environment-type>-<image-identifier>
    instance-size:<instance-size>
```

以下是 GitHub 動作工作流程 YAML 的範例：

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        image:arm-3.0
        instance-size:small
```

```
steps:
  - run: echo "Hello World!"
```

- 您可以在 標籤 中覆寫用於組建的機群。這將覆寫在專案上設定的機群設定，以使用指定的機群。如需詳細資訊，請參閱 [在預留容量機群上執行組建](#)。若要覆寫 Amazon EC2 運算建置的機群設定，請使用下列語法：

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
  fleet:<fleet-name>
```

若要覆寫用於建置的機群和映像，請使用下列語法：

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
  fleet:<fleet-name>
  image:<environment-type>-<image-identifier>
```

以下是 GitHub 動作工作流程 YAML 的範例：

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      fleet:myFleet
      image:arm-3.0
    steps:
      - run: echo "Hello World!"
```

- 若要在自訂映像上執行 GitHub 動作任務，您可以在 CodeBuild 專案中設定自訂映像，並避免提供映像覆寫標籤。如果未提供映像覆寫標籤，CodeBuild 將使用專案中設定的映像。
- 或者，您可以在 CodeBuild 支援的標籤之外提供標籤。為了覆寫組建的屬性，將忽略這些標籤，但 Webhook 請求不會失敗。例如，將新增 testLabel 為標籤不會阻止組建執行。

 Note

如果在 CodeBuild 環境中無法使用 GitHub 託管執行器提供的相依性，您可以在工作流程執行中使用 GitHub 動作安裝相依性。例如，您可以使用 [setup-python](#) 動作為您的建置環境安裝 Python。

在 INSTALL、PRE_BUILD 和 POST_BUILD 階段執行 `buildspec` 命令

根據預設，CodeBuild 會在執行自我託管的 GitHub Actions 組建時忽略任何 `buildspec` 命令。若要在建置期間執行 `buildspec` 命令，`buildspec-override:true` 可以新增為標籤的尾碼：

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    buildspec-override:true
```

透過使用此命令，CodeBuild `actions-runner` 將在容器的主要來源資料夾中建立名為 `actions-runner` 的資料夾。當 GitHub Actions 執行器在 BUILD 階段期間啟動時，執行器將在 `actions-runner` 目錄中執行。

在自我託管 GitHub 動作建置中使用 `buildspec` 覆寫時，有幾個限制：

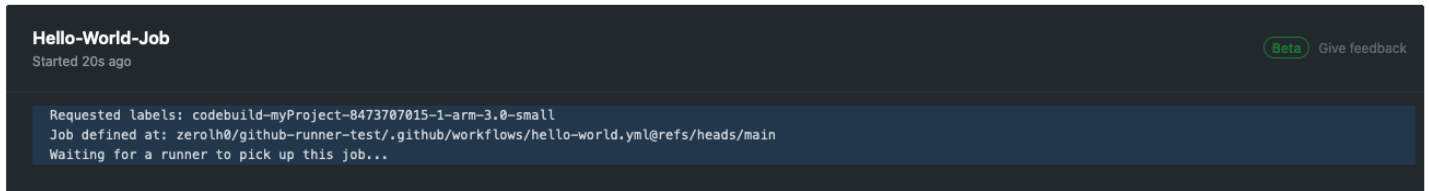
- CodeBuild 不會在 BUILD 階段期間執行 `buildspec` 命令，因為自我託管執行器會在 BUILD 階段中執行。
- CodeBuild 不會在 DOWNLOAD_SOURCE 階段期間下載任何主要或次要來源。如果您已設定 `buildspec` 檔案，則只會從專案的主要來源下載該檔案。
- 如果建置命令在 PRE_BUILD 或 INSTALL 階段失敗，CodeBuild 將不會啟動自我託管執行器，而且需要手動取消 GitHub Actions 工作流程任務。
- CodeBuild 會在 DOWNLOAD_SOURCE 階段期間擷取執行器字符，其過期時間為一小時。如果您的 PRE_BUILD 或 INSTALL 階段超過一小時，執行器字符可能會在 GitHub 自我託管執行器啟動之前過期。

步驟 3：檢閱您的結果

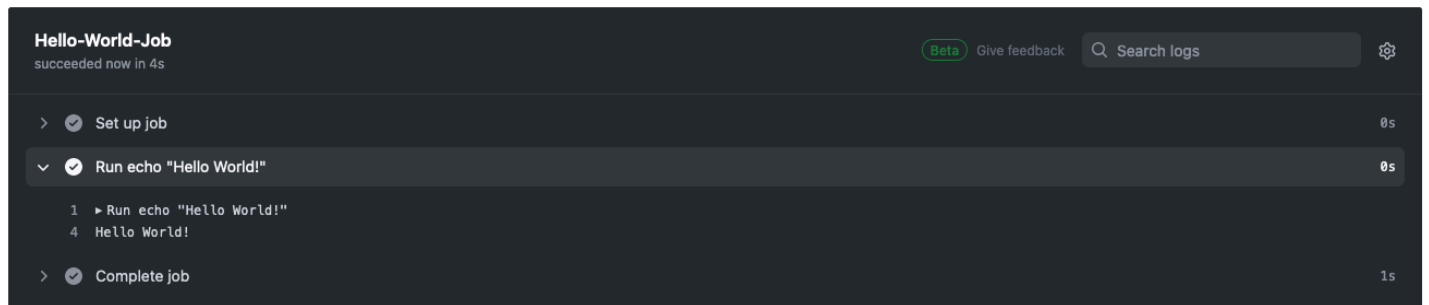
每當 GitHub Actions 工作流程執行時，CodeBuild 會透過 Webhook 接收工作流程任務事件。對於工作流程中的每個任務，CodeBuild 會啟動組建以執行暫時性 GitHub Actions 執行器。執行器負責執行單一工作流程任務。任務完成後，執行器和相關聯的建置程序將立即終止。

若要檢視工作流程任務日誌，請導覽至 GitHub 中的儲存庫、選擇動作、選擇所需的工作流程，然後選擇您要檢閱日誌的特定任務。

您可以在任務等待 CodeBuild 中的自我託管執行器提取時，檢閱日誌中請求的標籤。



任務完成後，您將能夠檢視任務的日誌。



GitHub Actions 執行器組態選項

您可以在專案組態中指定下列環境變數，以修改自我託管執行器的設定組態。

`CODEBUILD_CONFIG_GITHUB_ACTIONS_ORG_REGISTRATION_NAME`

CodeBuild 會將自我託管執行器註冊到指定為此環境變數值的組織名稱。如需在組織層級註冊執行器和必要許可的詳細資訊，請參閱[為組織的just-in-time執行器建立組態](#)。

`CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME`

CodeBuild 會將自我託管執行器註冊到指定為此環境變數值的企業名稱。如需在企業層級註冊執行器和必要許可的詳細資訊，請參閱[為企業的just-in-time執行器建立組態](#)。

Note

根據預設，企業執行器不適用於組織儲存庫。若要讓自我託管的執行器接收工作流程任務，您可能需要設定執行器群組存取設定。如需詳細資訊，請參閱[讓儲存庫可以使用企業執行器](#)。

CODEBUILD_CONFIG_GITHUB_ACTIONS_RUNNER_GROUP_ID

CodeBuild 會將自我託管的執行器註冊到整數執行器群組 ID，此群組 ID 會儲存為此環境變數的值。根據預設，此值為 1。如需自我託管執行器群組的詳細資訊，請參閱[使用群組管理對自我託管執行器的存取](#)。

CODEBUILD_CONFIG_GITHUB_ACTIONS_ORG_REGISTRATION_NAME

若要使用 GitHub Actions 工作流程 YAML 檔案設定組織層級執行器註冊，您可以使用下列語法：

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        organization-registration-name:myOrganization
    steps:
      - run: echo "Hello World!"
```

CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME

若要使用 GitHub Actions 工作流程 YAML 檔案設定企業層級執行器註冊，您可以使用下列語法：

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        enterprise-registration-name:myEnterprise
    steps:
      - run: echo "Hello World!"
```

CODEBUILD_CONFIG_GITHUB_ACTIONS_RUNNER_GROUP_ID

若要使用 GitHub Actions 工作流程 YAML 檔案將註冊執行器設定為特定執行器群組 ID，您可以使用下列語法：

```
name: Hello World
on: [push]
jobs:
```



```
Hello-World-Job:
  runs-on:
    - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    registration-group-id:3
  steps:
    - run: echo "Hello World!"
```

篩選 GitHub 動作 Webhook 事件 (CloudFormation)

範本的 CloudFormation 下列 YAML 格式部分會建立篩選條件群組，在評估為 true 時觸發組建。下列篩選條件群組會指定工作流程名稱符合規則表達式的 GitHub Actions 工作流程任務請求\[CI-CodeBuild\]。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: organization-name
        Scope: GITHUB_ORGANIZATION
      FilterGroups:
        - - Type: EVENT
          Pattern: WORKFLOW_JOB_QUEUED
        - Type: WORKFLOW_NAME
          Pattern: \[CI-CodeBuild\]
```

篩選 GitHub 動作 Webhook 事件 (AWS CDK)

下列 AWS CDK 範本會建立篩選條件群組，其會在評估為 true 時觸發組建。下列篩選條件群組會指定 GitHub 動作工作流程任務請求。

```
import { aws_codebuild as codebuild } from 'aws-cdk-lib';
import { EventAction, FilterGroup } from "aws-cdk-lib/aws-codebuild";

const source = codebuild.Source.gitHub({
  owner: 'owner',
  repo: 'repo',
  webhook: true,
  webhookFilters: [FilterGroup.inEventOf(EventAction.WORKFLOW_JOB_QUEUED)],
});
```

篩選 GitHub 動作 Webhook 事件 (Terraform)

下列 Terraform 範本會建立篩選條件群組，其會在評估為 true 時觸發組建。下列篩選條件群組會指定 GitHub 動作工作流程任務請求。

```
resource "aws_codebuild_webhook" "example" {
  project_name = aws_codebuild_project.example.name
  build_type   = "BUILD"
  filter_group {
    filter {
      type      = "EVENT"
      pattern   = "WORKFLOW_JOB_QUEUED"
    }
  }
}
```

篩選 GitHub 動作 Webhook 事件 (AWS CLI)

下列 AWS CLI 命令會使用 GitHub Actions 工作流程任務請求篩選條件群組來建立自我託管的 GitHub Actions 執行器專案，該群組會在評估為 true 時觸發組建。

```
aws codebuild create-project \
--name <project name> \
--source "{\"type\":\"GITHUB\",\"location\":\"<repository location>\",\"buildspec\":\"\\\"\\\"}\" \
--artifacts "{\"type\":\"NO_ARTIFACTS\"}\" \
--environment "{\"type\":\"LINUX_CONTAINER\",\"image\":\"aws/codebuild/amazonlinux-x86_64-standard:5.0\",\"computeType\":\"BUILD_GENERAL1_MEDIUM\"}\" \
--service-role "<service role ARN>"
```

```
aws codebuild create-webhook \
```

```
--project-name <project name> \  
--filter-groups "[[{\"type\":\"EVENT\",\"pattern\":\"WORKFLOW_JOB_QUEUED\"}]]"
```

Webhook 故障診斷

問題：您在 中設定的 Webhook [教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#) 無法運作，或您的工作流程任務在 GitHub 上懸置。

可能的原因：

- 您的 Webhook 工作流程任務事件可能無法觸發組建。檢閱回應日誌以檢視回應或錯誤訊息。
- 您的任務因為標籤組態而指派給不正確的執行器代理程式。當您在單一工作流程執行中的其中一個任務的標籤少於另一個任務時，可能會發生此問題。例如，如果您在相同的工作流程執行中有兩個具有下列標籤的任務：
 - 任務 1：codebuild-myProject-\${{ github.run_id }}-\${{ github.run_attempt }}
 - 任務 2：codebuild-myProject-\${{ github.run_id }}-\${{ github.run_attempt }}、instance-size:medium

路由自我託管的 GitHub 動作任務時，GitHub 會將任務路由到具有所有任務指定標籤的任何執行器。此行為表示任務 1 可以由為任務 1 或任務 2 建立的執行器挑選，但任務 2 只能由為任務 2 建立的執行器挑選，因為它具有額外的標籤。如果任務 1 是由為任務 2 建立的執行器接管，則任務 2 將會停滯，因為任務 1 執行器沒有instance-size:medium標籤。

建議的解決方案：

在相同的工作流程執行中建立多個任務時，請為每個任務使用相同數量的標籤覆寫，或為每個任務指派自訂標籤，例如 job1或 job2。

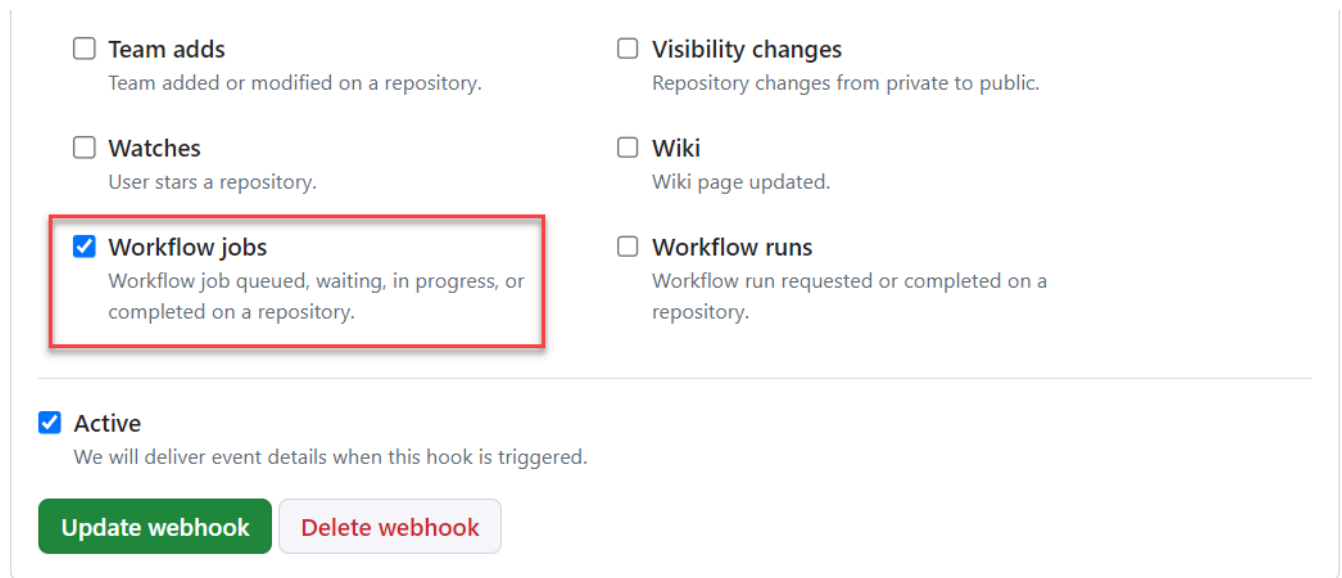
指派每個任務自己的唯一自訂標籤是最可靠的選項。它可確保沒有任何任務的一組標籤是另一個任務的子集。當每個任務都有沒有其他任務的標籤時，GitHub 只會將每個任務比對至為其建立的執行器，無論每個任務使用多少其他覆寫。在下列工作流程中，每個任務都有唯一的標籤 (job1 和 job2)。雖然任務 2 具有額外的instance-size:medium覆寫，但任務 1 無法再符合為任務 2 建立的執行器，因此兩個任務都無法停滯：

```
name: Hello World  
on: [push]  
jobs:
```

```
job1:
  runs-on:
    - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    - job1
  steps:
    - run: echo "Hello from Job 1!"
job2:
  runs-on:
    - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    - instance-size:medium
    - job2
  steps:
    - run: echo "Hello from Job 2!"
```

如果錯誤仍然存在，請使用下列指示來偵錯問題。

1. 在開啟 GitHub 主控台<https://github.com/user-name/repository-name/settings/hooks>，以檢視儲存庫的 Webhook 設定。在此頁面上，您會看到為儲存庫建立的 Webhook。
2. 選擇編輯並確認已啟用 Webhook 以交付工作流程任務事件。



☐ Team adds
Team added or modified on a repository.

☐ Watches
User stars a repository.

☒ Workflow jobs
Workflow job queued, waiting, in progress, or completed on a repository.

☐ Visibility changes
Repository changes from private to public.

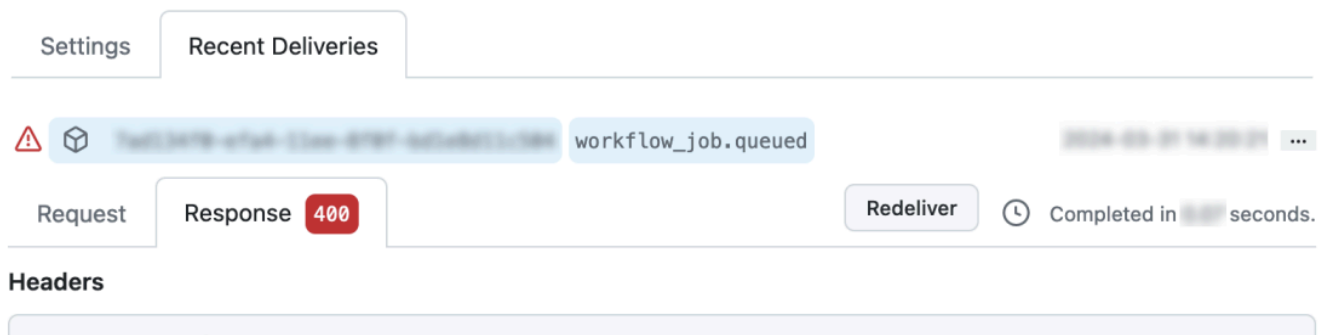
☐ Wiki
Wiki page updated.

☐ Workflow runs
Workflow run requested or completed on a repository.

☒ Active
We will deliver event details when this hook is triggered.

Update webhook Delete webhook

3. 導覽至最近傳送索引標籤，尋找對應的workflow_job.queued事件，然後展開事件。
4. 檢閱承載中的標籤欄位，並確認其如預期。
5. 最後，檢閱回應索引標籤，因為這包含 CodeBuild 傳回的回應或錯誤訊息。



6. 或者，您可以使用 GitHub 的 APIs 偵錯 Webhook 失敗。您可以使用[儲存庫 Webhook API 的列出交付來檢視 Webhook 的最新交付](#)：

```
gh api \
  -H "Accept: application/vnd.github+json" \
  -H "X-GitHub-API-Version: 2022-11-28" \
  /repos/owner/repo/hooks/hook-id/deliveries
```

找到您要偵錯的 Webhook 交付並記下交付 ID 之後，您可以使用[取得儲存庫 Webhook API 的交付](#)。CodeBuild 對 Webhook 交付承載的回應可在 response 區段中找到：

```
gh api \
  -H "Accept: application/vnd.github+json" \
  -H "X-GitHub-API-Version: 2022-11-28" \
  /repos/owner/repo/hooks/hook-id/deliveries/delivery-id
```

問題：啟用[部署保護](#)規則的 GitHub 動作觸發會在 CodeBuild 中建置，然後再核准部署。

可能原因：CodeBuild 會擷取與 GitHub 動作任務相關聯的部署和環境，如果它們存在，以驗證部署是否已核准。如果 CodeBuild 無法擷取部署或環境，可能會提早觸發 CodeBuild 組建。

建議的解決方案：確認與 CodeBuild 專案相關聯的登入資料具有 GitHub 內部署和動作的讀取許可。

CodeBuild 託管的 GitHub 動作執行器支援標籤覆寫

在 GitHub Actions 工作流程 YAML 中，您可以提供各種標籤覆寫，以修改您的自我託管執行器建置。CodeBuild 無法辨識的任何建置都會遭到忽略，但不會失敗您的 Webhook 請求。下列工作流程 YAML 定義兩個任務：會針對映像、執行個體大小、機群和 buildspec job2 job1 使用覆寫，並僅使用映像覆寫。為每個任務提供一個唯一的標籤 (job1 和 job2)，以便 GitHub 將每個任務路由到為其建立的執行器：

```
name: Hello World
on: [push]
jobs:
  job1:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:arm-3.0
      - instance-size:small
      - fleet:myFleet
      - buildspec-override:true
      - job1
    steps:
      - run: echo "Hello from Job 1!"
  job2:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:linux-5.0
      - job2
    steps:
      - run: echo "Hello from Job 2!"
```

您也可以使用矩陣策略跨多個組態執行單一任務。單一矩陣任務的擴展始終具有相同數量的標籤，因此執行器比對問題不會影響它們。不過，如果工作流程執行包含多個矩陣任務，請將自己的唯一自訂標籤（例如 `matrix-job-1` 和 `matrix-job-2`）提供給每個矩陣任務，讓 GitHub 不會將任務與為不同矩陣任務建立的執行器比對：

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:${{ matrix.os }}
      - instance-size:${{ matrix.size }}
      - fleet:myFleet
      - buildspec-override:true
      - matrix-job-1
    strategy:
      matrix:
        include:
          - os: arm-3.0
            size: small
```

```
- os: linux-5.0
  size: large
steps:
- run: echo "Hello World!"
```

Note

如果您的工作流程任務在 GitHub 上懸置，請參閱[Webhook 故障診斷](#)和[使用自訂標籤來路由任務](#)。

codebuild-*<project-name>*-\${{github.run_id}}-\${{github.run_attempt}} (必要)

- 範例：codebuild-fake-project-\${{ github.run_id }}-\${{ github.run_attempt }}
- 所有 GitHub Actions 工作流程 YAMLs 都需要。*<project name>* 應等於設定自我託管執行器 Webhook 的專案名稱。

image:*<environment-type>*-*<image-identifier>*

- 範例：image:arm-3.0
- 覆寫使用精選映像啟動自我託管執行器建置時所使用的映像和環境類型。若要了解支援的值，請參閱[CodeBuild 託管的 GitHub Actions Runner 支援運算映像](#)。
 - 若要覆寫與自訂映像搭配使用的映像和環境類型，請使用 image:custom-*<environment-type>*-*<custom-image-identifier>*
 - 範例：image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0

Note

如果自訂映像位於私有登錄檔中，請參閱[設定自我託管執行器的私有登錄登入資料](#)。

instance-size:*<instance-size>*

- 範例：instance-size:medium

- 覆寫啟動自我託管執行器建置時所使用的執行個體類型。若要了解支援的值，請參閱 [CodeBuild 託管的 GitHub Actions Runner 支援運算映像](#)。

`fleet:<fleet-name>`

- 範例：fleet:myFleet
- 覆寫專案上設定的機群設定，以使用指定的機群。如需詳細資訊，請參閱[在預留容量機群上執行組建](#)。

`buildspec-override:<boolean>`

- 範例：buildspec-override:true
- 如果設為 PRE_BUILD，允許組建在 INSTALL、和 POST_BUILD 階段中執行 buildspec 命令 true。

`<unique-label>`（當工作流程執行有多個任務時建議使用）

- 範例：job1
- 您定義且相同工作流程執行中沒有其他任務使用的自訂標籤。為每個任務指派唯一的標籤，可確保 GitHub 只會將每個任務路由到為其建立的執行器。如果沒有它，當相同工作流程執行中的任務具有不同數量的標籤覆寫時，為具有更多標籤的任務建立的執行器可以拾取具有較少標籤的任務，這會離開沒有執行器的第二個任務。如需詳細資訊，請參閱[Webhook 故障診斷](#)。

單一標籤覆寫

CodeBuild 可讓您使用下列項目，在單一標籤中提供多個覆寫：

Note

由於所有覆寫都會合併為單一標籤，因此每個任務的執行器只會註冊一個標籤。這表示 中所述的多工作執行器比對問題[Webhook 故障診斷](#)無法使用此格式發生，即使每個工作上沒有唯一的標籤也一樣。如果您偏好此行為，而且您的任務只需要覆寫其支援的，請使用單一標籤格式。當您需要不支援的覆寫時，請勿使用它。例如，自訂映像（包括私有登錄檔中的映像）只能使用 `image:custom-<environment-type>-<custom-image-identifier>` 標籤格式來指定。對於這些覆寫，請使用本主題先前所述的個別標籤格式。

- 若要覆寫 Amazon EC2/Lambda 運算組建的環境設定，請使用下列語法：


```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<environment-type>-<image-identifier>-<instance-size>
```

- 若要覆寫 Amazon EC2 運算建置的機群設定，請使用下列語法：

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}-
fleet-<fleet-name>
```

- 若要覆寫用於建置的機群和映像，請使用下列語法：

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-image-<image-version>-fleet-<fleet-name>
```

- 若要在建置期間執行 buildspec 命令，-with-buildspec 可以新增為標籤的尾碼：

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<image>-<image-version>-<instance-size>-with-buildspec
```

- 或者，您可以提供執行個體大小覆寫，而無需覆寫映像。對於 Amazon EC2 組建，您可以排除環境類型和映像識別符。對於 Lambda 組建，您可以排除影像識別符。

CodeBuild 託管的 GitHub Actions Runner 支援運算映像

在您在 中設定的標籤中[教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)，您可以使用前三欄中的值覆寫 Amazon EC2 環境設定。CodeBuild 提供下列 Amazon EC2 運算映像。如需關於

環境類型	映像識別符	執行個體大小	平台	已解析的映像	定義
linux	4.0	small	Amazon Linux 2	aws/codeb	al/standa rd/4.0
		medium		uild/amaz	
		large		onlinux-x	
		xlarge		86_64-sta	
linux	5.0	2xlarge	Amazon Linux 2023	andard:4.0	al/standa rd/5.0
		gpu_small		aws/codeb	
				uild/amaz	
				onlinux-x	

環境類型	映像識別符	執行個體大小	平台	已解析的映像	定義
		gpu_large		86_64-standard:5.0	
linux-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-x86_64-base:latest	無
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0
arm-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-arm-base:latest	無

環境類型	映像識別符	執行個體大小	平台	已解析的映像	定義
ubuntu	5.0	small medium	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0	large xlarge	Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0	2xlarge gpu_small gpu_large	Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	N/A
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	N/A
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A

環境類型	映像識別符	執行個體大小	平台	已解析的映像	定義
windows-ec2	2022	medium large	Windows Server Core 2022	aws/codebuild/ami/windows-base:2022	無

此外，您可以使用下列值覆寫 Lambda 環境設定。如需 CodeBuild Lambda 運算的詳細資訊，請參閱在 [AWS Lambda 運算上執行組建](#)。CodeBuild 支援下列 Lambda 運算映像：

環境類型	映像識別符	執行個體大小			
linux-lambda	dotnet6	1GB			
	go1.21	2GB			
arm-lambda	corretto11	4GB			
		8GB			
	corretto17	10GB			
	corretto21				
	nodejs18				
	nodejs20				
	python3.11				
	python3.12				
	ruby3.2				

如需詳細資訊，請參閱[建置環境運算模式和類型](#)及[CodeBuild 提供的 Docker 映像](#)。

中的自我管理 GitLab 執行器 AWS CodeBuild

GitLab 提供兩種執行模式，可在 CI/CD 管道中執行 GitLab 任務。一種模式是 GitLab 託管的執行器，由 GitLab 管理並與 GitLab 完全整合。另一個模式是自我管理的執行器，可讓您將自己的自訂環境帶入 GitLab CI/CD 管道中執行任務。

設定 CodeBuild 專案以執行 GitLab CI/CD 管道任務的高階步驟如下：

1. 如果您尚未這麼做，請使用 OAuth 應用程式將專案連線至 GitLab。
2. 導覽至 CodeBuild 主控台，並使用 Webhook 建立 CodeBuild 專案，並設定您的 Webhook 篩選條件。
3. 在 GitLab 中更新您的 GitLab CI/CD 管道 YAML，以設定您的建置環境。

如需更詳細的程序，請參閱 [教學課程：設定 CodeBuild 託管的 GitLab 執行器](#)。

此功能可讓您的 GitLab CI/CD 管道任務與原生整合 AWS，透過 IAM、和 Amazon VPC 等功能提供安全性 AWS CloudTrail 和便利性。您可以存取最新的執行個體類型，包括以 ARM 為基礎的執行個體。

主題

- [關於 CodeBuild 託管的 GitLab Runner](#)
- [教學課程：設定 CodeBuild 託管的 GitLab 執行器](#)
- [CodeBuild 託管的 GitLab Runner 支援標籤覆寫](#)
- [CodeBuild 託管的 GitLab Runner 支援運算映像](#)

關於 CodeBuild 託管的 GitLab Runner

以下是有關 CodeBuild 託管的 GitLab 執行器的一些常見問題。

CodeBuild 託管的 GitLab 執行器支援哪些來源類型？

GITLAB 和 GITLAB_SELF_MANAGED 來源類型支援 CodeBuild 託管的 GitLab 執行器。

我應該在標籤中何時包含映像和執行個體覆寫？

您可以在標籤中包含映像和執行個體覆寫，以便為每個 GitLab CI/CD 管道任務指定不同的建置環境。這可以完成，而無需建立多個 CodeBuild 專案或 Webhook。

我可以將 CloudFormation 用於此功能嗎？

是，您可以在 CloudFormation 範本中包含篩選條件群組，在專案 Webhook 中指定 GitLab 工作流程任務事件篩選條件。

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
      Pattern: WORKFLOW_JOB_QUEUED
```

如需詳細資訊，請參閱[篩選 GitLab Webhook 事件 \(CloudFormation\)](#)。

如果您需要在 CloudFormation 範本中設定專案登入資料的說明，請參閱AWS CloudFormation 《使用者指南》中的 [AWS::CodeBuild::SourceCredential](#) 以取得詳細資訊。

使用此功能時，如何遮罩秘密？

根據預設，不會遮罩在日誌中列印的秘密。如果您想要遮罩秘密，您可以透過更新 CI/CD 環境變數設定來執行此操作：

在 GitLab 中遮罩秘密

1. 在您的 GitLab 設定中，選擇 CI/CD。
2. 在變數中，針對您要遮罩的秘密選擇編輯。
3. 在可見性中，選取遮罩變數，然後選擇更新變數以儲存變更。

我可以從單一群組中的多個專案接收 GitLab Webhook 事件嗎？

CodeBuild 支援群組 Webhook，從指定的 GitLab 群組接收事件。如需詳細資訊，請參閱[GitLab 群組 Webhook](#)。

我可以在 Docker 執行器中為自我管理的執行器執行任務嗎？例如，我想要在特定映像上執行管道任務，以在個別隔離的容器中維護相同的建置環境。

您可以使用特定映像在 CodeBuild 中執行 GitLab 自我管理執行器，[方法是使用自訂映像建立專案](#)，或[覆寫檔案中的映像](#)。`.gitlab-ci.yml`

CodeBuild 中的自我管理執行器會搭配哪些執行器執行？

CodeBuild 中的自我管理執行器會使用 shell 執行器執行，其中建置會在本機執行，並搭配在 Docker 容器內執行的 GitLab 執行器。

我可以提供 buildspec 命令與自我管理的執行器嗎？

是，您可以新增 buildspec 命令以及自我管理的執行器。您可以在 GitLab 儲存庫中提供 buildspec.yml 檔案，並在任務的標籤區段中使用 buildspec-override:true 標籤。如需詳細資訊，請參閱 [Buildspec 檔案名稱和儲存位置](#)。

哪些區域支援使用 CodeBuild 託管的 GitLab 執行器？

所有 CodeBuild 區域都支援 CodeBuild 託管的 GitLab 執行器。如需 CodeBuild 可用 AWS 區域 位置的詳細資訊，請參閱 [AWS 依區域提供服務](#)。

哪些平台支援使用 CodeBuild 託管的 GitLab 執行器？

Amazon EC2 和 [AWS Lambda](#) 運算都支援 CodeBuild 託管的 GitLab 執行器。您可以使用下列平台：Amazon Linux 2、Amazon Linux 2023、Ubuntu 和 Windows Server Core 2019。如需詳細資訊，請參閱 [EC2 運算映像](#) 及 [Lambda 運算映像](#)。

教學課程：設定 CodeBuild 託管的 GitLab 執行器

本教學課程說明如何設定 CodeBuild 專案以執行 GitLab CI/CD 管道任務。如需使用 GitLab 或 GitLab Self Managed with CodeBuild 的詳細資訊，請參閱 [中的自我管理 GitLab 執行器 AWS CodeBuild](#)。

若要完成此教學課程，您必須先：

- 使用與 OAuth 應用程式連線 AWS CodeConnections。請注意，連線至 OAuth 應用程式時，您必須使用 CodeBuild 主控台來執行此操作。如需更多說明，請參閱 [CodeBuild 中的 GitLab 存取](#)。
- 將 CodeBuild 連接至您的 GitLab 帳戶。若要這麼做，您可以在主控台中新增 GitLab 做為來源提供者。如需說明，請參閱 [CodeBuild 中的 GitLab 存取](#)。

Note

只有在您尚未連線到您帳戶的 GitLab 時，才需要這麼做。

透過此功能，CodeBuild 需要額外的許可，例如 `manage_runner` 來自 GitLab OAuth 應用程式的 `create_runner` 和 `manage_runner`。如果特定 GitLab AWS CodeConnections 帳戶已有，則不會自動請求更新許可。若要這麼做，您可以前往 AWS CodeConnections 主控台，並建立相同 GitLab 帳戶的虛擬連線，以觸發重新授權以取得其他許可。完成此步驟後，所有現有的連線都可以使用執行器功能。完成後，您可以刪除虛擬連線。

步驟 1：使用 Webhook 建立 CodeBuild 專案

在此步驟中，您將使用 Webhook 建立 CodeBuild 專案，並在 GitLab 主控台中檢閱。

使用 Webhook 建立 CodeBuild 專案

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。

在專案類型中，選擇執行器專案。

- 在執行器中：
 - 針對 Runner 提供者，選擇 GitLab。
 - 針對登入資料，選擇下列其中一項：
 - 選擇預設來源憑證。預設連線會將預設 GitLab 連線套用至所有專案。
 - 選擇自訂來源憑證。自訂連線會套用自訂 GitLab 連線，覆寫您帳戶的預設設定。

Note

如果您尚未建立與供應商的連線，則必須建立新的 GitLab 連線。如需說明，請參閱[將 CodeBuild 連接至 GitLab](#)。

- 針對執行器位置，選擇儲存庫。
 - 針對儲存庫，使用命名空間指定專案路徑，在 GitLab 中選擇專案的名稱。
 - 在 Environment (環境) 中：
 - 選擇支援的環境映像和運算。請注意，您可以選擇使用 GitLab CI/CD 管道 YAML 中的標籤覆寫映像和執行個體設定。如需詳細資訊，請參閱[步驟 2：在您的儲存庫中建立 .gitlab-ci.yml 檔案](#)。
 - 在 Builds spec (建置規格) 中：
 - 請注意，除非新增為標籤，否則您的 builds spec `buildspec-override:true` 將被忽略。反之，CodeBuild 會覆寫它，以使用將設定自我管理執行器的命令。
 -
3. 繼續執行預設值，然後選擇建立建置專案。

- 開啟位於的 GitLab 主控台<https://gitlab.com/user-name/repository-name/-/hooks>，確認 webhook 已建立並啟用，以交付工作流程任務事件。

步驟 2：在您的儲存庫中建立 .gitlab-ci.yml 檔案

在此步驟中，您將在 中建立 .gitlab-ci.yml 檔案[GitLab](#)，以設定您的建置環境，並在 CodeBuild 中使用 GitLab 自我管理執行器。如需詳細資訊，請參閱[使用自我管理的執行器](#)。

更新您的 GitLab CI/CD 管道 YAML

導覽至 <https://gitlab.com/user-name/project-name/-/tree/branch-name>並在您的儲存庫中建立 .gitlab-ci.yml 檔案。您可以執行下列其中一項操作來設定建置環境：

- 您可以指定 CodeBuild 專案名稱，在這種情況下，組建將使用您現有的專案組態進行運算、映像、映像版本和執行個體大小。需要專案名稱，才能將 GitLab 任務的 AWS 相關設定連結至特定 CodeBuild 專案。透過在 YAML 中包含專案名稱，CodeBuild 可以使用正確的專案設定叫用任務。

```
tags:
  - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
```

`$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME` 需要將組建映射至特定的管道任務執行，並在管道執行取消時停止組建。

Note

請確定您的 `<project-name>` 符合您在 CodeBuild 中建立的專案名稱。如果不相符，CodeBuild 不會處理 Webhook，GitLab CI/CD 管道可能會停止運作。

以下是 GitLab CI/CD 管道 YAML 的範例：

```
workflow:
  name: HelloWorld
stages:          # List of stages for jobs, and their order of execution
  - build

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  script:
    - echo "Hello World!"
```

```
tags:
  - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
```

- 您也可以 在 標籤中覆寫映像和運算類型。 [CodeBuild 託管的 GitLab Runner 支援運算映像](#) 如需精選影像的清單，請參閱。如需使用自訂映像，請參閱 [CodeBuild 託管的 GitLab Runner 支援標籤覆寫](#)。標籤中的運算類型和映像會覆寫專案上的環境設定。若要覆寫 Amazon EC2 運算組建的環境設定，請使用下列語法：

```
tags:
  - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
  - image:<environment-type>-<image-identifier>
  - instance-size:<instance-size>
```

以下是 GitLab CI/CD 管道 YAML 的範例：

```
stages:
  - build

build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:arm-3.0
    - instance-size:small
```

- 您可以在 標籤中覆寫用於建置的機群。這將覆寫在專案上設定的機群設定，以使用指定的機群。如需詳細資訊，請參閱[在預留容量機群上執行組建](#)。若要覆寫 Amazon EC2 運算建置的機群設定，請使用下列語法：

```
tags:
  - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
  - fleet:<fleet-name>
```

若要同時覆寫用於建置的機群和映像，請使用下列語法：

```
tags:
  - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
  - fleet:<fleet-name>
```

```
- image:<environment-type>-<image-identifier>
```

以下是 GitLab CI/CD 管道 YAML 的範例：

```
stages:
  - build

build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - fleet:myFleet
    - image:arm-3.0
```

- 若要在自訂映像上執行 GitLab CI/CD 管道任務，您可以在 CodeBuild 專案中設定自訂映像，並避免提供映像覆寫標籤。如果未提供映像覆寫標籤，CodeBuild 將使用專案中設定的映像。

將變更遞交至後 `.gitlab-ci.yml`，將觸發 GitLab 管道，而 `build-job` 將傳送 Webhook 通知，以在 CodeBuild 中開始建置。

在 `INSTALL`、`PRE_BUILD` 和 `POST_BUILD` 階段中執行 `buildspec` 命令

根據預設，CodeBuild 會在執行自我管理的 GitLab 組建時忽略任何 `buildspec` 命令。若要在建置期間執行 `buildspec` 命令，`buildspec-override:true` 可以新增為的尾碼 `tags`：

```
tags:
  - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
  - buildspec-override:true
```

透過使用此命令，CodeBuild `gitlab-runner` 將在容器的主要來源資料夾中建立名為 `gitlab-runner` 的資料夾。當 GitLab Runner 在 `BUILD` 階段期間啟動時，執行器將在 `gitlab-runner` 目錄中執行。

在自我管理的 GitLab 建置中使用 `buildspec` 覆寫時，有幾個限制：

- CodeBuild 不會在 `BUILD` 階段期間執行 `buildspec` 命令，因為自我管理的執行器會在 `BUILD` 階段中執行。
- CodeBuild 不會在 `DOWNLOAD_SOURCE` 階段期間下載任何主要或次要來源。如果您已設定 `buildspec` 檔案，則只會從專案的主要來源下載該檔案。

- 如果建置命令在 PRE_BUILD或 INSTALL階段失敗，CodeBuild 將不會啟動自我管理的執行器，而且需要手動取消 GitLab CI/CD 管道任務。
- CodeBuild 會在DOWNLOAD_SOURCE階段期間擷取執行器字符，其過期時間為一小時。如果您的 PRE_BUILD或 INSTALL階段超過一小時，執行器權杖可能會在 GitLab 自我管理的執行器啟動之前過期。

步驟 3：檢閱您的結果

每當 GitLab CI/CD 管道執行時，CodeBuild 會透過 Webhook 接收 CI/CD 管道任務事件。對於 CI/CD 管道中的每個任務，CodeBuild 會啟動組建以執行暫時性 GitLab 執行器。執行器負責執行單一 CI/CD 管道任務。任務完成後，執行器和相關聯的建置程序將立即終止。

若要檢視 CI/CD 管道任務日誌，請導覽至 GitLab 中的儲存庫，選擇建置、任務，然後選擇您要檢閱日誌的特定任務。

您可以在任務等待 CodeBuild 中的自我管理執行器提取時，檢閱日誌中請求的標籤。

篩選 GitLab Webhook 事件 (CloudFormation)

範本的 CloudFormation 下列 YAML 格式部分會建立篩選條件群組，在評估為 true 時觸發組建。下列篩選條件群組會使用符合規則表達式的 CI/CD 管道名稱，指定 GitLab CI/CD 管道任務請求\[CI-CodeBuild\]。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITLAB
      Location: CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: group-name
```

```

Scope: GITLAB_GROUP
FilterGroups:
  - - Type: EVENT
    Pattern: WORKFLOW_JOB_QUEUED
  - Type: WORKFLOW_NAME
    Pattern: \[CI-CodeBuild\]

```

CodeBuild 託管的 GitLab Runner 支援標籤覆寫

在 GitLab CI/CD 管道 YAML 中，您可以提供各種標籤覆寫，以修改自我管理的執行器建置。CodeBuild 無法辨識的任何建置都會遭到忽略，但不會讓您的 Webhook 請求失敗。例如，下列 YAML 包含映像、執行個體大小、機群和 buildspec 的覆寫：

```

workflow:
  name: HelloWorld
stages:
  - build

build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:arm-3.0
    - instance-size:small
    - fleet:myFleet
    - buildspec-override:true

```

codebuild-*<project-name>*-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME (必要)

- 範例：codebuild-myProject-\$CI_PROJECT_ID-\$CI_PIPELINE_IID-\$CI_JOB_NAME
- 所有 GitLab CI/CD 管道 YAMLs 都需要。*<project name>* 應等於設定自我管理執行器 Webhook 的專案名稱。

image:*<environment-type>*-*<image-identifier>*

- 範例：image:arm-3.0
- 覆寫啟動自我管理執行器建置時所使用的映像和環境類型。若要了解支援的值，請參閱 [CodeBuild 託管的 GitLab Runner 支援運算映像](#)。

- 若要覆寫與自訂映像搭配使用的映像和環境類型，請使用 `image:custom-<environment-type>-<custom-image-identifier>`
- 範例：`image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0`

Note

如果自訂映像位於私有登錄檔中，請參閱 [設定自我託管執行器的私有登錄登入資料](#)。

`instance-size:<instance-size>`

- 範例：`instance-size:small`
- 覆寫啟動自我管理執行器建置時所使用的執行個體類型。若要了解支援的值，請參閱 [CodeBuild 託管的 GitLab Runner 支援運算映像](#)。

`fleet:<fleet-name>`

- 範例：`fleet:myFleet`
- 覆寫專案上設定的機群設定，以使用指定的機群。如需詳細資訊，請參閱 [在預留容量機群上執行組建](#)。

`buildspec-override:<boolean>`

- 範例：`buildspec-override:true`
- 如果設為 `PRE_BUILD`，允許組建在 `INSTALL`、和 `POST_BUILD` 階段中執行 `buildspec` 命令 `true`。

CodeBuild 託管的 GitLab Runner 支援運算映像

在您在 中設定的標籤中 [教學課程：設定 CodeBuild 託管的 GitLab 執行器](#)，您可以使用前三欄中的值覆寫 Amazon EC2 環境設定。CodeBuild 提供下列 Amazon EC2 運算映像。如需關於

環境類型	映像識別符	執行個體大小	平台	影像	定義
linux	4.0	small	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/standard/4.0

環境類型	映像識別符	執行個體大小	平台	影像	定義
		medium large xlarge		onlinux-x86_64-standard:4.0	
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-standard:5.0	al/standard/5.0
linux-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-x86_64-base:latest	無
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0

環境類型	映像識別符	執行個體大小	平台	影像	定義
arm-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-arm-base:latest	無
ubuntu	5.0	small medium	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0	large xlarge 2xlarge	Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0	gpu_small gpu_large	Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	N/A
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	N/A
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A

環境類型	映像識別符	執行個體大小	平台	影像	定義
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A
windows-ec2	2022	medium large	Windows Server Core 2022	aws/codebuild/ami/windows-base:2022	無

此外，您可以使用下列值覆寫 Lambda 環境設定。如需 CodeBuild Lambda 運算的詳細資訊，請參閱在 [AWS Lambda 運算上執行組建](#)。CodeBuild 支援下列 Lambda 運算映像：

環境類型	執行時間版本	執行個體大小			
linux-lambda	dotnet6	1GB			
	go1.21	2GB			
arm-lambda	corretto1.1	4GB			
		8GB			
	corretto1.7	10GB			
	corretto2.1				
	nodejs18				
	nodejs20				
	python3.11				
	python3.12				

環境類型	執行時間版本	執行個體大小			
	ruby3.2				

如需詳細資訊，請參閱[建置環境運算模式和類型](#)及[CodeBuild 提供的 Docker 映像](#)。

中的自我管理 Buildkite 執行器 AWS CodeBuild

您可以設定專案在 CodeBuild 容器中設定自我託管的 Buildkite 執行器，以處理您的 Buildkite 任務。這可以透過使用 CodeBuild 專案設定 Webhook，並更新您的 Buildkite 管道 YAML 步驟來使用 CodeBuild 機器上託管的自我託管執行器來完成。

設定 CodeBuild 專案以執行 Buildkite 任務的高階步驟如下：

- 導覽至 CodeBuild 主控台，並使用 Buildkite 執行器專案執行器類型組態建立 CodeBuild 專案
- 將 `job.scheduled` Webhook 新增至您的 Buildkite 組織。
- 在 Buildkite 中更新您的 Buildkite 管道 YAML 步驟，以設定您的建置環境。

如需更詳細的程序，請參閱 [教學課程：設定 CodeBuild 託管的 Buildkite 執行器](#)。此功能可讓您的 Buildkite 任務與進行原生整合 AWS，透過 IAM AWS Secrets Manager AWS CloudTrail、和 Amazon VPC 等功能提供安全性和便利性。您可以存取最新的執行個體類型，包括以 ARM 為基礎的執行個體。

關於 CodeBuild 託管的 Buildkite 執行器

以下是有關 CodeBuild 託管的 Buildkite 執行器的一些常見問題。

我應該何時在標籤中包含映像和執行個體覆寫？

您可以在 標籤中包含映像和執行個體覆寫，以便為每個 Buildkite 任務指定不同的建置環境。這可以完成，無需建立多個 CodeBuild 專案或 Webhook。例如，當您需要使用 [Buildkite 任務的矩陣](#)時，這很有用。

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
```

```
project: "codebuild-myProject"
image: "${matrix.os}"
instance-size: "${matrix.size}"
matrix:
  setup:
    os:
      - "arm-3.0"
      - "al2-5.0"
    size:
      - "small"
      - "large"
```

CodeBuild 是否可以在 Buildkite 中自動建立 Webhook？

目前，Buildkite 要求使用其主控台手動建立所有 Webhook。您可以遵循的教學課程[教學課程：設定 CodeBuild 託管的 Buildkite 執行器](#)，在 Buildkite 主控台中手動建立 Buildkite Webhook。

我可以使用 CloudFormation 來建立 Buildkite Webhook 嗎？

CloudFormation 目前不支援 Buildkite 執行器 Webhook，因為 Buildkite 需要使用其主控台手動建立 Webhook。

哪些區域支援使用 CodeBuild 託管的 Buildkite 執行器？

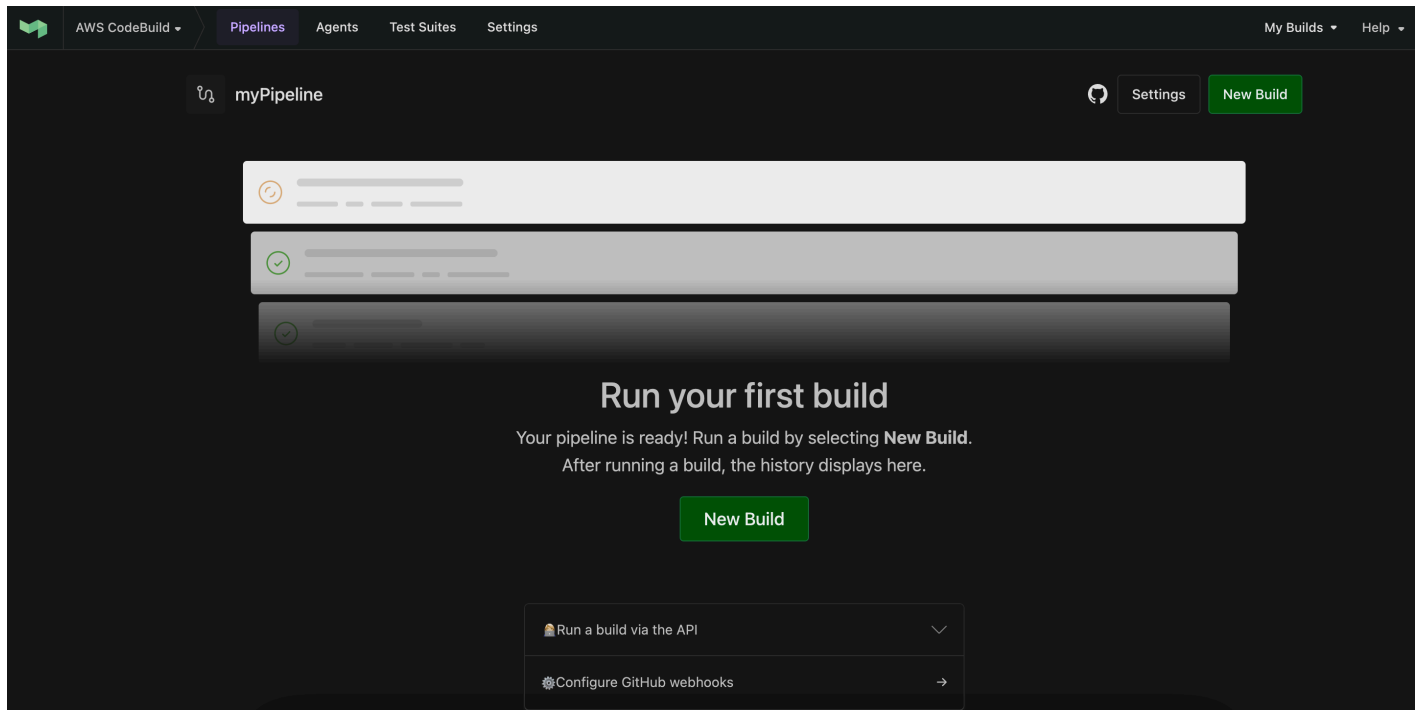
所有 CodeBuild 區域都支援 CodeBuild 託管的 Buildkite 執行器。如需 CodeBuild 可用 AWS 區域的詳細資訊，請參閱[AWS 依區域提供服務](#)。

教學課程：設定 CodeBuild 託管的 Buildkite 執行器

本教學課程說明如何設定 CodeBuild 專案以執行 Buildkite 任務。如需搭配 CodeBuild 使用 Buildkite 的詳細資訊，請參閱 [中的自我管理 Buildkite 執行器 AWS CodeBuild](#)。

若要完成此教學課程，您必須先：

- 有權存取 Buildkite 組織。如需有關設定 Buildkite 帳戶和組織的詳細資訊，您可以遵循此[入門教學課程](#)。
- 建立設定為使用自我託管執行器的 Buildkite 管道、叢集和佇列。如需有關設定這些資源的詳細資訊，您可以參考 [Buildkite 管道設定教學課程](#)。



步驟 1：產生 Buildkite 代理程式字符

在此步驟中，您將在 Buildkite 中產生代理程式字符，用於驗證 CodeBuild 自我託管執行器。如需此資源的詳細資訊，請參閱 [Buildkite Agent Tokens](#)。

產生 Buildkite 代理程式字符

1. 在 Buildkite 叢集中，選擇客服人員字符，然後選擇新字符。
2. 將描述新增至字符，然後按一下建立字符。
3. 儲存代理程式字符值，因為它將在 CodeBuild 專案設定期間稍後使用。

Agent Tokens > New

Description — Required

myToken

Describe the set of agents this token is for

Allowed IP Addresses

0.0.0.0/0 ::/0

Restrict which network addresses are allowed to use this agent token. Use space-separated, IPv4 CIDR notation, for example:
192.0.2.0/24 198.51.100.12 25c7:c056:9943:1e01::/64 .

Create Token

步驟 2：使用 Webhook 建立 CodeBuild 專案

使用 Webhook 建立 CodeBuild 專案

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立自我託管建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
 - 在專案組態中，選取執行器專案。在執行器中：
 - 針對 Runner 提供者，選擇 Buildkite。
 - 對於 Buildkite 代理程式字符，使用建立秘密頁面選擇建立新的代理程式字符。系統會提示您在 中建立新的秘密 AWS Secrets Manager，秘密值等於您在上面產生的 Buildkite 代理程式字符。
 - (選用) 如果您想要為任務使用 CodeBuild 受管登入資料，請在 Buildkite 來源登入資料選項下選取任務的來源儲存庫提供者，並確認已為您的帳戶設定登入資料。此外，請確認您的 Buildkite 管道使用 HTTPS 進行結帳。

Note

Buildkite 需要建置環境中的來源憑證，才能提取任務的來源。如需可用的來源登入資料選項 [向私有儲存庫驗證 Buildkite](#)，請參閱。

- (選用) 在環境中：

- 選擇支援的環境映像和運算。

請注意，您可以選擇使用 Buildkite YAML 步驟中的標籤覆寫映像和執行個體設定。如需詳細資訊，請參閱[步驟 4：更新您的 Buildkite 管道步驟](#)。

- (選用) 在 Buildspec 中：
- 除非 `buildspec-override: "true"` 新增為標籤，否則預設會忽略您的 `buildspec`。反之，CodeBuild 會覆寫它，以使用將設定自我託管執行器的命令。

 Note

CodeBuild 不支援 Buildkite 自我託管執行器建置的 `buildspec` 檔案。對於內嵌 `buildspec`，如果您已設定 CodeBuild 受管來源憑證，則需要在 `buildspec` 中啟用 [git-credential-helper](#)

3. 繼續執行預設值，然後選擇建立建置專案。
4. 從建立 Webhook 快顯視窗中儲存承載 URL 和秘密值。請依照快顯視窗中的指示建立新的 Buildkite 組織 Webhook，或繼續下一節。

步驟 3：在 Buildkite 中建立 CodeBuild Webhook

在此步驟中，您將使用 CodeBuild Webhook 中的承載 URL 和秘密值，在 Buildkite 中建立新的 Webhook。當有效的 Buildkite 任務啟動時，此 Webhook 將用於觸發 CodeBuild 中的組建。

在 Buildkite 中建立新的 Webhook

1. 開啟 Buildkite 組織的設定頁面。
2. 在整合下，選取通知服務。
3. 選擇 Webhook 方塊旁的新增。在新增 Webhook 通知頁面中，使用下列組態：
 - a. 在 Webhook URL 下，新增已儲存的承載 URL 值。
 - b. 在字符下，確認已選取以 X-Buildkite-Token 傳送字符。將 Webhook Secret 值新增至 Token 欄位。
 - c. 在下，確認已選取以 X-Buildkite-Token 傳送權杖。將 Webhook Secret 值新增至 Token 欄位。
 - d. 在事件下，選取 `job.scheduled` Webhook 事件。
 - e. (選用) 在管道下，您可以選擇只觸發特定管道的建置。

4. 選擇新增 Webhook 通知。

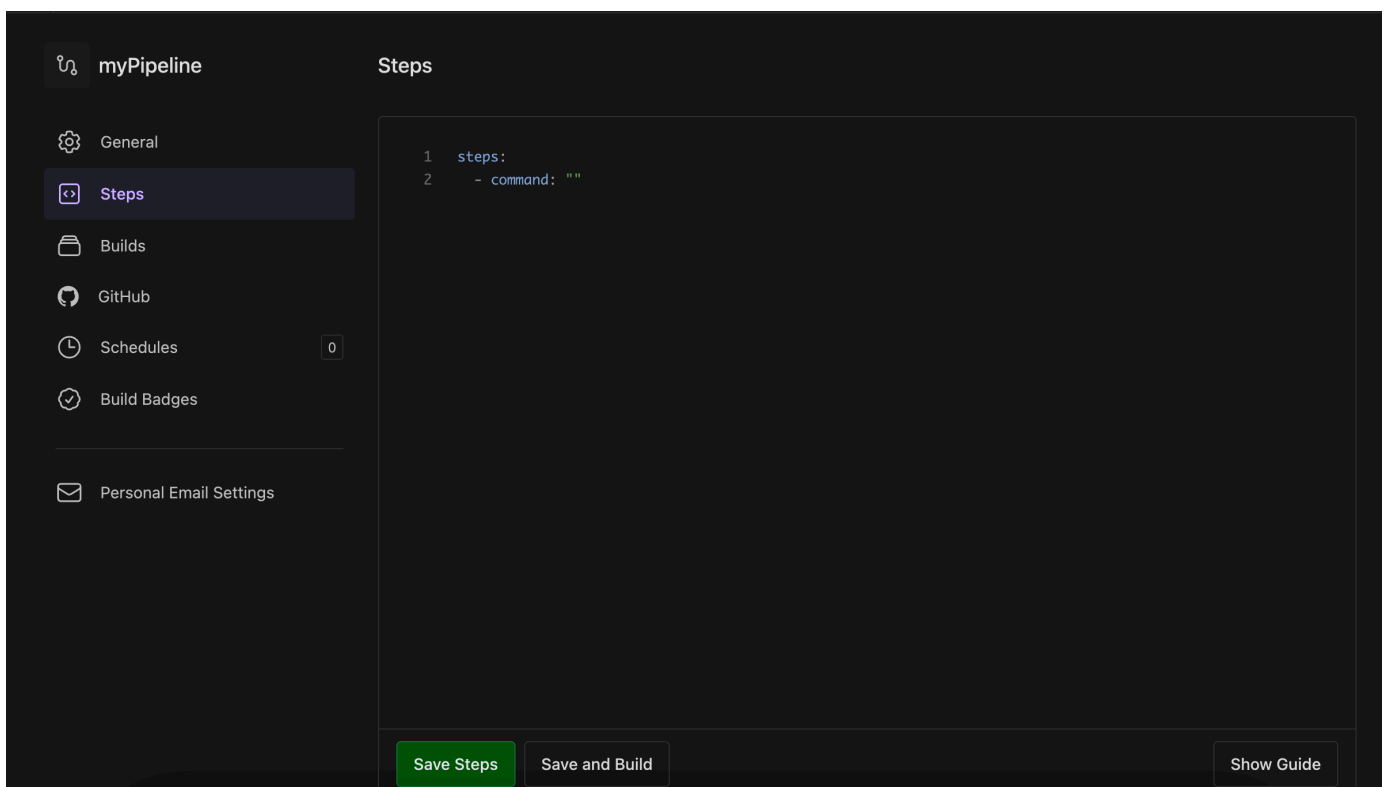
步驟 4：更新您的 Buildkite 管道步驟

在此步驟中，更新 Buildkite 管道的步驟，以新增必要的標籤和選用覆寫。如需支援標籤覆寫的完整清單，請參閱 [CodeBuild 託管的 Buildkite 執行器支援標籤覆寫](#)。

更新您的管道步驟

1. 選擇您的 Buildkite 管道、選擇設定，然後選擇步驟，以開啟 Buildkite 管道步驟頁面。

如果您尚未進行，請選擇轉換為 YAML 步驟。



2. 您至少需要指定參考 CodeBuild 管道名稱的 [Buildkite 代理程式標籤](#)。需要專案名稱，才能將 Buildkite 任務 AWS 的相關設定連結至特定 CodeBuild 專案。透過在 YAML 中包含專案名稱，CodeBuild 可以使用正確的專案設定叫用任務。

```
agents:
  project: "codebuild-<project name>"
```

以下是只有專案標籤的 Buildkite 管道步驟範例：

```
agents:
  project: "codebuild-myProject"
steps:
  - command: "echo \"Hello World\""
```

您也可以在 標籤中覆寫映像和運算類型。[CodeBuild 託管的 Buildkite 執行器支援運算映像](#) 如需可用映像的清單，請參閱 。標籤中的運算類型和映像會覆寫專案上的環境設定。若要覆寫 CodeBuild EC2 或 Lambda 運算組建的環境設定，請使用下列語法：

```
agents:
  project: "codebuild-<project name>"
  image: "<environment-type>-<image-identifier>"
  instance-size: "<instance-size>"
```

以下是具有映像和執行個體大小覆寫的 Buildkite 管道步驟範例：

```
agents:
  project: "codebuild-myProject"
  image: "arm-3.0"
  instance-size: "small"
steps:
  - command: "echo \"Hello World\""
```

您可以在 標籤中覆寫用於組建的機群。這將覆寫在專案上設定的機群設定，以使用指定的機群。如需詳細資訊，請參閱[在預留容量機群上執行組建](#)。

若要覆寫 Amazon EC2 運算組建的機群設定，請使用下列語法：

```
agents:
  project: "codebuild-<project name>"
  fleet: "<fleet-name>"
```

若要覆寫用於建置的機群和映像，請使用下列語法：

```
agents:
  project: "codebuild-<project name>"
  fleet: "<fleet-name>"
  image: "<environment-type>-<image-identifier>"
```


以下是具有機群和映像覆寫的 Buildkite 管道步驟範例：

```
agents:
  project: "codebuild-myProject"
  fleet: "myFleet"
  image: "arm-3.0"
steps:
  - command: "echo \"Hello World\""
```

3. 您可以選擇在自我託管的 Buildkite 執行器建置期間執行內嵌 `buildspec` 命令（如需詳細資訊[針對 `INSTALL`、`PRE\_BUILD` 和 `POST\_BUILD` 階段執行 `buildspec` 命令](#)，請參閱）。若要指定 CodeBuild 組建應在 Buildkite 自我託管執行器組建期間執行 `buildspec` 命令，請使用下列語法：

```
agents:
  project: "codebuild-<project name>"
  buildspec-override: "true"
```

以下是具有 `buildspec` 覆寫的 Buildkite 管道範例：

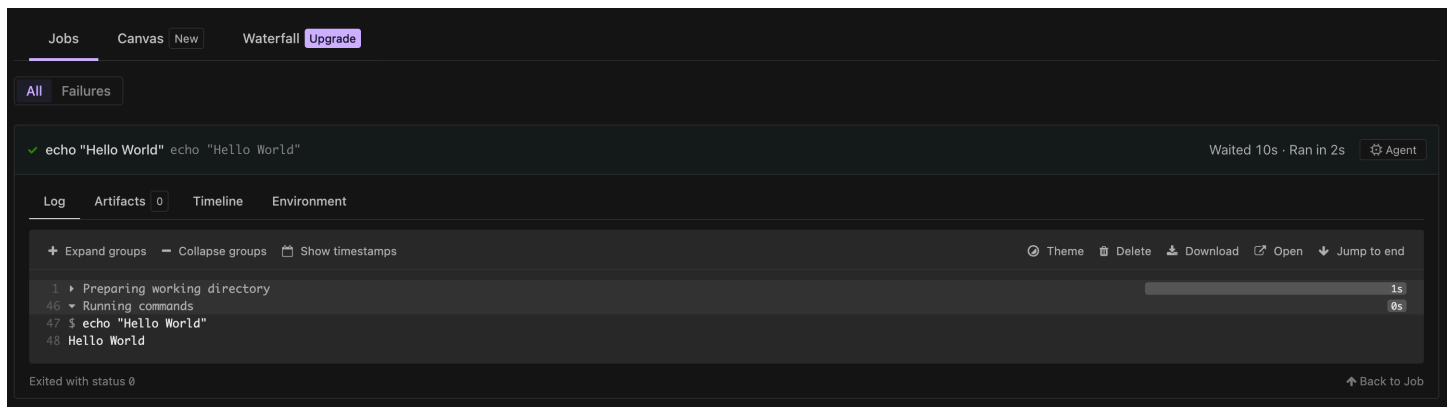
```
agents:
  project: "codebuild-myProject"
  buildspec-override: "true"
steps:
  - command: "echo \"Hello World\""
```

4. 或者，您可以在 CodeBuild 支援的標籤之外提供標籤。為了覆寫組建的屬性，將忽略這些標籤，但 Webhook 請求不會失敗。例如，將新增 `myLabel: "testLabel"` 為標籤不會阻止組建執行。

步驟 5：檢閱您的結果

每當 Buildkite 任務在您的管道中啟動時，CodeBuild 將透過 Buildkite `job.scheduled` Webhook 接收 Webhook 事件。對於 Buildkite 組建中的每個任務，CodeBuild 會啟動組建以執行暫時性的 Buildkite 執行器。執行器負責執行單一 Buildkite 任務。任務完成後，執行器和相關聯的建置程序將立即終止。

若要檢視工作流程任務日誌，請導覽至 Buildkite 管道，然後選取最新的組建（您可以選擇新組建來觸發新組建）。每個任務的相關聯 CodeBuild 組建開始並挑選任務後，您應該會在 Buildkite 主控台中查看任務的日誌



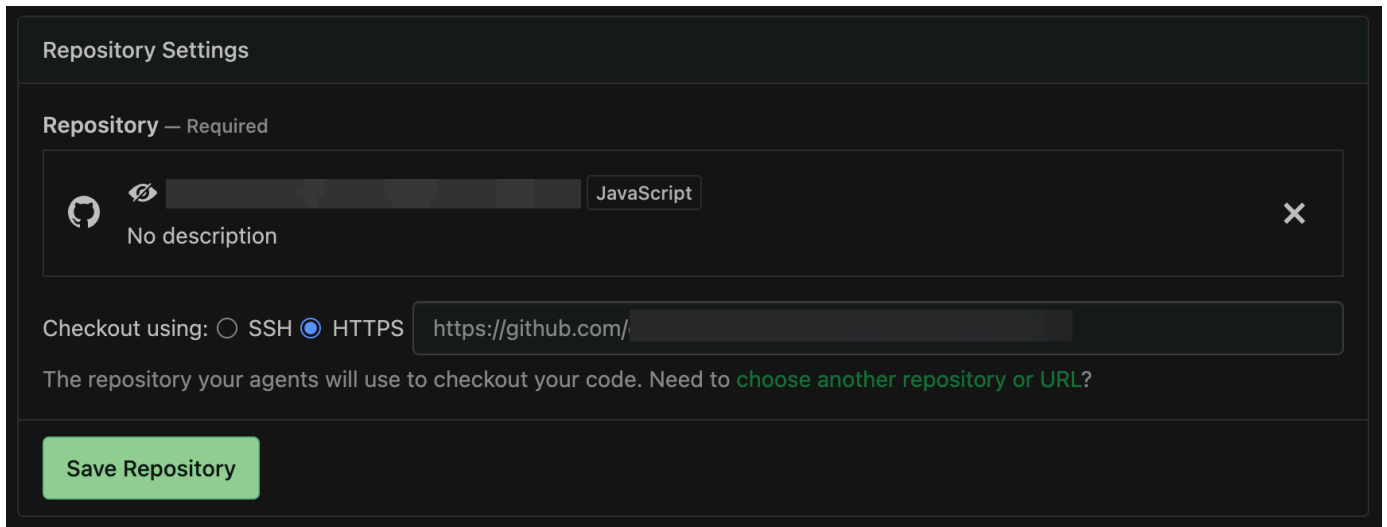
向私有儲存庫驗證 Buildkite

如果您在 Buildkite 管道中設定了私有儲存庫，則 Buildkite [需要建置環境中的其他許可](#)才能提取儲存庫，因為 Buildkite 不會將登入資料提供給自我託管的執行器，以便從私有儲存庫提取。若要向外部私有來源儲存庫驗證 Buildkite 自我託管執行器代理程式，您可以使用下列其中一個選項。

使用 CodeBuild 驗證

CodeBuild 為支援的來源類型提供受管憑證處理。若要使用 CodeBuild 來源登入資料提取任務的來源儲存庫，您可以使用下列步驟：

1. 在 CodeBuild 主控台中，導覽至編輯專案，或使用 [中的步驟](#)建立新的 CodeBuild 專案[步驟 2：使用 Webhook 建立 CodeBuild 專案](#)。
2. 在 Buildkite 來源登入資料選項下，選取任務的來源儲存庫提供者。
 1. 如果您想要使用帳戶層級的 CodeBuild 登入資料，請確認它們已正確設定。此外，如果您的專案已設定內嵌 buildspect，請確認 [git-credential-helper](#) 已啟用。
 2. 如果您想要使用專案層級 CodeBuild 登入資料，請選取僅針對此專案使用覆寫登入資料，並為您的專案設定登入資料。
3. 在 Buildkite 管道設定中，導覽至儲存庫設定。將來源儲存庫結帳設定設定為使用 HTTPS 結帳



Repository Settings

Repository — Required

  No description 

Checkout using: ☐ SSH ☒ HTTPS

The repository your agents will use to checkout your code. Need to [choose another repository or URL?](#)

使用 Buildkite 秘密進行身分驗證

Buildkite 會維護 [ssh-checkout 外掛程式](#)，可用來使用 ssh 金鑰向外部來源儲存庫驗證自我託管執行器。金鑰值會儲存為 [Buildkite 秘密](#)，並在嘗試提取私有儲存庫時由 Buildkite 自我託管執行器代理程式自動擷取。若要為您的 Buildkite 管道設定 ssh-checkout 外掛程式，您可以使用下列步驟：

1. 使用您的電子郵件地址產生私有和公有 SSH 金鑰，例如 `ssh-keygen -t rsa -b 4096 -C "myEmail@address.com"`
2. 將公有金鑰新增至私有來源儲存庫。例如，您可以遵循[本指南](#)，將金鑰新增至 GitHub 帳戶。
3. 將新的 SSH 金鑰秘密新增至您的 Buildkite 叢集。在 Buildkite 叢集中，選取秘密 → 新秘密。在金鑰欄位中新增秘密的名稱，並將私有 SSH 金鑰新增至值欄位：

New Secret

Key — Required

SOURCE_SSH_KEY

Keys are case insensitive, can only contain alphanumeric and underscore characters, and can't start with BUILDKITE or BK

Value — Required

-----BEGIN OPENSSH PRIVATE KEY-----

4. 在 Buildkite 管道中，導覽至您的儲存庫設定，並將結帳設定為使用 SSH。

Repository Settings

Repository — Required

  [redacted] 

No description

Checkout using: ☒ SSH ☐ HTTPS

The repository your agents will use to checkout your code. Need to [choose another repository or URL?](#)

[Save Repository](#)

5. 更新您的管道 YAML 步驟以使用 `git-ssh-checkout` 外掛程式。例如，下列管道 YAML 檔案會將結帳動作與上述 Buildkite 私密金鑰搭配使用：

```
agents:
  project: "codebuild-myProject"
steps:
  - command: "npm run build"
    plugins:
      - git-ssh-checkout#v0.4.1:
          ssh-secret-key-name: 'SOURCE_SSH_KEY'
```

6. 在 CodeBuild 中執行 Buildkite 自我託管執行器任務時，Buildkite 現在會在提取私有儲存庫時自動使用您設定的秘密值

執行器組態選項

您可以在專案組態中指定下列環境變數，以修改自我託管執行器的設定組態：

- `CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN`：CodeBuild 會從擷取設定為此環境變數 AWS Secrets Manager 值的秘密值，以註冊 Buildkite 自我託管執行器代理程式。此環境變數的類型必須是 `SECRETS_MANAGER`，且值應該是 Secrets Manager 中的秘密名稱。所有 Buildkite 執行器專案都需要 Buildkite 代理程式字環境變數。
- `CODEBUILD_CONFIG_BUILDKITE_CREDENTIAL_DISABLE`：根據預設，CodeBuild 會將帳戶或專案層級來源登入資料載入建置環境，因為 Buildkite 代理程式會使用這些登入資料來提取任務的來源儲存庫。若要停用此行為，您可以將此環境變數新增至值設為的專案 `true`，以防止來源登入資料載入建置環境。

針對 `INSTALL`、`PRE_BUILD` 和 `POST_BUILD` 階段執行 `buildspec` 命令

根據預設，CodeBuild 會在執行自我託管的 Buildkite 執行器建置時忽略任何 `buildspec` 命令。若要在建置期間執行 `buildspec` 命令，

```
buildspec-override: "true"
```

可以新增為標籤的尾碼：

```
agents:
  project: "codebuild-<project name>"
  buildspec-override: "true"
```

透過使用此命令，CodeBuild `buildkite-runner` 將在容器的主要來源資料夾中建立名為 `buildkite-runner` 的資料夾。當 Buildkite 執行器在 `BUILD` 階段期間啟動時，執行器會在 `buildkite-runner` 目錄中執行。

在自我託管的 Buildkite 建置中使用 `buildspec` 覆寫時，有幾個限制：

- Buildkite 代理程式要求來源憑證存在於建置環境中，以提取任務的來源儲存庫。如果您使用 CodeBuild 來源登入資料進行身分驗證，則需要在 `buildspec git-credential-helper` 中啟用。例如，您可以使用下列 `buildspec` 為 `git-credential-helper` Buildkite 組建啟用：

```
version: 0.2
env:
  git-credential-helper: yes
phases:
  pre_build:
    commands:
      - echo "Hello World"
```

- CodeBuild 不會在 BUILD 階段期間執行 buildspec 命令，因為自我託管執行器會在 BUILD 階段中執行。
- CodeBuild 不支援 Buildkite Runner 組建的 buildspec 檔案。Buildkite 自我託管執行器僅支援內嵌 buildspec
- 如果建置命令在 PRE_BUILD 或 INSTALL 階段失敗，CodeBuild 將不會啟動自我託管執行器，而且需要手動取消 Buildkite 任務。

以程式設計方式設定 Buildkite 執行器

若要以程式設計方式設定 Buildkite 執行器專案，您需要設定下列資源：

以程式設計方式建立 Buildkite 執行器

1. 建立 Buildkite 代理程式字符，並在 中以純文字儲存字符 AWS Secrets Manager。
2. 使用您偏好的組態設定 CodeBuild 專案。您需要設定下列其他屬性：
 1. 名稱為 的環境值 CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN，類型為 SECRETS_MANAGER，值等於與 Buildkite 叢集相關聯的 Buildkite 代理程式字符。
 2. 來源類型等於 NO_SOURCE
 3. 存取專案服務角色中步驟 1 中建立之秘密的許可

例如，您可以使用下列命令，透過 CLI 建立有效的 Buildkite 執行器專案：

```
aws codebuild create-project \
  --name buildkite-runner-project \
  --source "{\"type\": \"NO_SOURCE\", \"buildspec\": \"\"}" \
  --environment "{\"image\": \"aws/codebuild/amazonlinux-x86_64-standard:5.0\", \"type\": \"LINUX_CONTAINER\", \"computeType\": \"BUILD_GENERAL1_MEDIUM\", \"environmentVariables\": [{\"name\": \"CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN\", \"type\": \"SECRETS_MANAGER\", \"value\": \"<buildkite-secret-name>\"}]}" \
```

```
--artifacts "{\"type\": \"NO_ARTIFACTS\"}" \
--service-role <service-role>
```

3. 在步驟 2 中建立的專案上建立 Buildkite 執行器 Webhook。建立 Webhook 時，您將需要使用下列組態選項：

1. build-type 應等於 RUNNER_BUILDKITE_BUILD
2. 類型為 EVENT 且模式等於 的篩選條件 WORKFLOW_JOB_QUEUED

例如，您可以使用下列命令，透過 CLI 建立有效的 Buildkite 執行器 Webhook：

```
aws codebuild create-webhook \
--project-name buildkite-runner-project \
--filter-groups "[[{"type\":\"EVENT\",\"pattern\":\"WORKFLOW_JOB_QUEUED\"}]]" \
--build-type RUNNER_BUILDKITE_BUILD
```

4. 儲存 create-webhook 呼叫傳回的承載 URL 和秘密值，並使用登入資料在 Buildkite 主控台中建立 Webhook。您可以在 中參考步驟 3：在 Buildkite 中建立 CodeBuild Webhook，[教學課程：設定 CodeBuild 託管的 Buildkite 執行器](#)以取得如何設定此資源的指南。

針對失敗的建置或懸置任務進行 Webhook 故障診斷

問題：

您在 中設定的 Webhook [教學課程：設定 CodeBuild 託管的 Buildkite 執行器](#) 無法運作，或您的工作流程任務在 Buildkite 中懸置。

可能的原因：

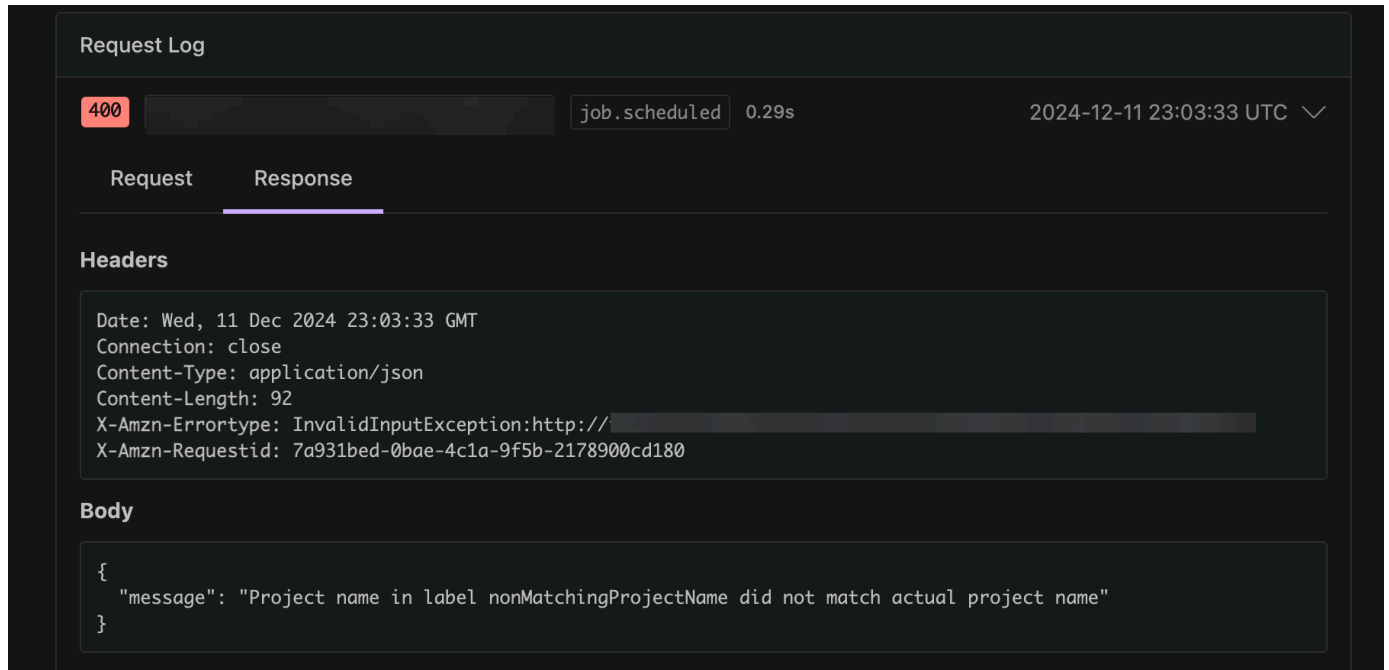
- 您的 Webhook job.scheduled 事件可能無法觸發組建。檢閱回應日誌以檢視回應或錯誤訊息。
- 您的 CodeBuild 組建會在啟動 Buildkite 自我託管執行器代理程式來處理任務之前失敗。

建議的解決方案：

若要偵錯失敗的 Buildkite Webhook 事件：

1. 在您的 Buildkite 組織設定中，導覽至 Notification Services，選取您的 CodeBuild Webhook，然後尋找請求日誌。

2. 尋找與您停滯的 Buildkite 任務相關聯的 `job.scheduled` Webhook 事件。您可以使用 Webhook 承載中的任務 ID 欄位，將 Webhook 事件與 Buildkite 任務建立關聯。
3. 選取回應索引標籤並檢查回應內文。確認回應狀態碼為 `200` 且回應內文不包含任何非預期的訊息。



The screenshot displays the 'Request Log' interface in the AWS CodeBuild console. It shows a single log entry with a status of 400 (highlighted in red) and a duration of 0.29s. The event type is 'job.scheduled' and it occurred on '2024-12-11 23:03:33 UTC'. Below the log entry, the 'Response' tab is selected, showing the following headers and body:

Headers

```
Date: Wed, 11 Dec 2024 23:03:33 GMT
Connection: close
Content-Type: application/json
Content-Length: 92
X-Amzn-ErrorType: InvalidInputException:http://
X-Amzn-Requestid: 7a931bed-0bae-4c1a-9f5b-2178900cd180
```

Body

```
{
  "message": "Project name in label nonMatchingProjectName did not match actual project name"
}
```

故障診斷 Webhook 許可問題

問題：

由於許可問題，Buildkite 任務無法簽出任務的來源儲存庫。

可能的原因：

- CodeBuild 沒有足夠的許可來簽出任務的來源儲存庫。
- 管道的儲存庫設定設定為使用 SSH for CodeBuild 受管憑證進行簽出。

建議的解決方案：

- 確認 CodeBuild 已設定足夠的許可來簽出任務的來源儲存庫。此外，請確認 CodeBuild 專案的服務角色具有足夠的許可，可存取設定的來源許可選項。
- 如果您使用 CodeBuild 受管來源儲存庫憑證，請確認您的 Buildkite 管道已設定為使用 HTTPS 結帳。

CodeBuild 託管的 Buildkite 執行器支援標籤覆寫

在 Buildkite 管道步驟代理程式標籤標籤中，您可以提供各種標籤覆寫，以修改您的自我託管執行器建置。CodeBuild 無法辨識的任何建置都會遭到忽略，但不會讓您的 Webhook 請求失敗。例如，下列工作流程 YAML 包含映像、執行個體大小、機群和 buildspec 的覆寫：

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
      project: "codebuild-myProject"
      image: "{{matrix.os}}"
      instance-size: "{{matrix.size}}"
      buildspec-override: "true"
    matrix:
      setup:
        os:
          - "arm-3.0"
          - "al2-5.0"
        size:
          - "small"
          - "large"
```

project:codebuild-*<project-name>* (必要)

- 範例：project: "codebuild-myProject"
- 所有 Buildkite 管道步驟組態都需要。*<project name>* 應等於設定自我託管執行器 Webhook 的專案名稱。

queue: "*<queue-name>*"

- 範例：queue: "*<queue-name>*"
- 用來將 Buildkite 任務路由到特定佇列。如需詳細資訊，請參閱 [Buildkite 代理程式佇列標籤](#)。

image: "*<environment-type>-<image-identifier>*"

- 範例：image: "arm-3.0"

- 覆寫使用精選映像啟動自我託管執行器建置時所使用的映像和環境類型。若要了解支援的值，請參閱 [CodeBuild 託管的 Buildkite 執行器支援運算映像](#)。

1. 若要覆寫與自訂映像搭配使用的映像和環境類型，請使用 `image: "custom-<environment-type>-<custom-image-identifier>"`
2. 範例：

```
image:
  "custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0"
```

Note

如果自訂映像位於私有登錄檔中，您必須在 CodeBuild 專案中設定適當的登錄檔登入資料。

`instance-size: "<instance-size>"`

- 範例：`instance-size: "medium"`
- 覆寫啟動自我託管執行器建置時所使用的執行個體類型。若要了解支援的值，請參閱 [CodeBuild 託管的 Buildkite 執行器支援運算映像](#)。

`fleet: "<fleet-name>"`

- 範例：`fleet: "myFleet"`
- 覆寫專案上設定的機群設定，以使用指定的機群。如需詳細資訊，請參閱 [在預留容量機群上執行組建](#)。

`buildspec-override: "<boolean>"`

- 範例：`buildspec-override: "true"`
- 如果設為 `PRE_BUILD`，允許組建在 `INSTALL`、和 `POST_BUILD` 階段中執行 `buildspec` 命令 `true`。

CodeBuild 託管的 Buildkite 執行器支援運算映像

在您在 中設定的標籤中 [中的自我管理 Buildkite 執行器 AWS CodeBuild](#)，您可以使用前三欄中的值覆寫 Amazon EC2 環境設定。CodeBuild 提供下列 Amazon EC2 運算映像。如需關於

環境類型	映像識別符	執行個體大小	平台	已解析的映像	定義
linux	4.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-standard:4.0	al/standard/4.0
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-standard:5.0	al/standard/5.0
linux-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-x86_64-base:latest	無
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0

環境類型	映像識別符	執行個體大小	平台	已解析的映像	定義
arm-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-arm-base:latest	無
ubuntu	5.0	small medium	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0	large xlarge 2xlarge	Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0	gpu_small gpu_large	Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	N/A
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	N/A
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A

環境類型	映像識別符	執行個體大小	平台	已解析的映像	定義
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A
windows-ec2	2022	medium large	Windows Server Core 2022	aws/codebuild/ami/windows-base:2022	無

此外，您可以使用下列值覆寫 Lambda 環境設定。如需 CodeBuild Lambda 運算的詳細資訊，請參閱在 [AWS Lambda 運算上執行組建](#)。CodeBuild 支援下列 Lambda 運算映像：

環境類型	映像識別符	執行個體大小			
linux-lambda	dotnet6	1GB			
	go1.21	2GB			
arm-lambda	corretto11	4GB			
		8GB			
	corretto17	10GB			
	corretto21				
	nodejs18				
	nodejs20				
	python3.11				
	python3.12				

環境類型	映像識別符	執行個體大小			
	ruby3.2				

如需詳細資訊，請參閱[建置環境運算模式和類型](#)及[CodeBuild 提供的 Docker 映像](#)。

搭配 使用 Webhook AWS CodeBuild

AWS CodeBuild 支援與 GitHub、GitHub Enterprise Server、GitLab、GitLab Self Managed 和 Bitbucket 的 Webhook 整合。

主題

- [搭配 使用 Webhook 的最佳實務 AWS CodeBuild](#)
- [Bitbucket Webhook 事件](#)
- [GitHub 全域和組織 Webhook](#)
- [GitHub 手動 Webhook](#)
- [GitHub Webhook 事件](#)
- [GitLab 群組 Webhook](#)
- [GitLab 手動 Webhook](#)
- [GitLab Webhook 事件](#)
- [Buildkite 手動 Webhook](#)
- [提取請求評論核准](#)

搭配 使用 Webhook 的最佳實務 AWS CodeBuild

對於使用公有儲存庫來設定 Webhook 的專案，我們建議使用下列選項：

設定ACTOR_ACCOUNT_ID篩選條件

將ACTOR_ACCOUNT_ID篩選條件新增至專案的 Webhook 篩選條件群組，以指定哪些使用者可以觸發組建。交付至 CodeBuild 的每個 Webhook 事件都會隨附寄件者資訊，以指定演員的識別符。CodeBuild 會根據篩選條件中提供的規則表達式模式來篩選 Webhook。您可以使用此篩選條件指定允許觸發組建的特定使用者。如需詳細資訊，請參閱[GitHub Webhook 事件](#)及[Bitbucket Webhook 事件](#)。

設定FILE_PATH篩選條件

將FILE_PATH篩選條件新增至專案的 Webhook 篩選條件群組，以包含或排除可在變更時觸發組建的檔案。例如，您可以使用規則表達式模式，例如 `includeMatchedPattern` 屬性`^buildspec.yml$`，拒絕`buildspec.yml`檔案變更的建置請求。如需詳細資訊，請參閱[GitHub Webhook 事件](#)及[Bitbucket Webhook 事件](#)。

縮小建置 IAM 角色的許可範圍

Webhook 觸發的組建會使用專案中指定的 IAM 服務角色。我們建議將服務角色中的許可設定為執行建置所需的最低許可集。例如，在測試和部署案例中，建立一個專案用於測試，另一個專案用於部署。測試專案接受來自儲存庫的 Webhook 組建，但不會對您的資源提供寫入許可。部署專案會為您的資源提供寫入許可，而 Webhook 篩選條件已設定為僅允許信任的使用者觸發組建。

使用內嵌或 Amazon S3 儲存的 buildspec

如果您在專案本身中內嵌定義 buildspec，或將 buildspec 檔案存放在 Amazon S3 儲存貯體中，則只有專案擁有者可以看到 buildspec 檔案。這可防止提取請求對 buildspec 檔案進程式碼變更，並觸發不需要的組建。如需詳細資訊，請參閱 CodeBuild API 參考中的 [ProjectSource.buildspec](#)。

Bitbucket Webhook 事件

您可以使用 Webhook 篩選群組來指定哪些 Bitbucket Webhook 事件會觸發組建。例如，您可以指定僅針對特定分支的變更觸發組建。

您可以建立一或多個 Webhook 篩選群組來指定哪些 Webhook 事件會觸發組建。如果任何篩選條件群組評估為 true，則會觸發組建，這會在群組中的所有篩選條件評估為 true 時發生。當您建立篩選群組時，您可以指定這些項目：

事件

對於 Bitbucket，您可以選擇下列一或多個事件：

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED
- PULL_REQUEST_CLOSED

Webhook 的事件類型位在 X-Event-Key 欄位的標頭中。下表顯示 X-Event-Key 標頭值如何映射到事件類型。

Note

如果您建立會使用 `PULL_REQUEST_MERGED` 事件類型的 Webhook 篩選群組，則必須在您的 Bitbucket Webhook 設定中啟用 `merged` 事件。如果您建立使用 `declined` 事件類型的 Webhook 篩選條件群組，您還必須在 Bitbucket Webhook 設定中啟用 `PULL_REQUEST_CLOSED` 事件。

X-Event-Key 標頭值	事件類型
<code>repo:push</code>	<code>PUSH</code>
<code>pullrequest:created</code>	<code>PULL_REQUEST_CREATED</code>
<code>pullrequest:updated</code>	<code>PULL_REQUEST_UPDATED</code>
<code>pullrequest:fulfilled</code>	<code>PULL_REQUEST_MERGED</code>
<code>pullrequest:rejected</code>	<code>PULL_REQUEST_CLOSED</code>

對於 `PULL_REQUEST_MERGED`，如果提取請求與 `squash` 策略合併，且提取請求分支已關閉，則原始提取請求遞交不再存在。在此情況下，`CODEBUILD_WEBHOOK_MERGE_COMMIT` 環境變數會包含壓縮合併遞交的識別符。

一或多個選用篩選條件

使用規則表達式來指定篩選條件。若要讓事件觸發組建，與其相關聯的群組中的每個篩選條件都必須評估為 `true`。

CodeBuild 使用 RE2 規則表達式語法評估篩選條件模式，不支援 `lookahead`、`look behind`、`backreferences` 或 `atomic group`。如需詳細資訊，請參閱 GitHub 網站上的 [RE2 語法](#) 頁面。

`ACTOR_ACCOUNT_ID` (`ACTOR_ID` 主控台中的)

當 Bitbucket 帳戶 ID 符合規則表達式模式時，Webhook 事件會觸發組建。這個值會出現在 Webhook 篩選條件承載之 `actor` 物件的 `account_id` 屬性中。

HEAD_REF

當頭部參考符合規則表達式模式（例如，`refs/heads/branch-name`和）時，Webhook 事件會觸發組建`refs/tags/tag-name`。HEAD_REF 篩選條件會評估分支或標籤的 Git 參考名稱。分支或標籤名稱會出現在 Webhook 承載 `push` 物件之 `new` 物件的 `name` 欄位中。針對提取請求事件，分支名稱會出現在 Webhook 承載 `source` 物件之 `branch` 物件的 `name` 欄位中。

BASE_REF

當基本參考符合規則表達式模式時，Webhook 事件會觸發組建。BASE_REF 篩選條件僅適用於提取請求事件（例如 `refs/heads/branch-name`）。BASE_REF 篩選條件會評估分支的 Git 參考名稱。分支名稱會出現在 Webhook 承載 `destination` 物件之 `branch` 物件的 `name` 欄位中。

FILE_PATH

當變更的檔案路徑符合規則表達式模式時，Webhook 會觸發組建。

Note

FILE_PATH 篩選條件會根據推送或提取請求中前 3,000 個變更的檔案進行評估。若要確保篩選條件依預期套用，請將變更的檔案數量保持在此限制內。對於提取請求事件，我們也建議使用提取請求建置政策來控制哪些提取請求可以啟動建置。

COMMIT_MESSAGE

當前端遞交訊息符合規則表達式模式時，Webhook 會觸發組建。

WORKFLOW_NAME

當工作流程名稱符合規則表達式模式時，Webhook 會觸發組建。

Note

您可以在 Bitbucket 儲存庫的 Webhook 設定中找到 Webhook 承載。

主題

- [篩選 Bitbucket Webhook 事件 \(主控台\)](#)
- [篩選 Bitbucket Webhook 事件 \(開發套件\)](#)
- [篩選 Bitbucket Webhook 事件 \(CloudFormation\)](#)

篩選 Bitbucket Webhook 事件 (主控台)

若要使用 AWS 管理主控台 篩選 Webhook 事件：

1. 當您建立專案時，請選取 Rebuild every time a code change is pushed to this repository (在每次將程式碼變更推送至此儲存庫時重建)。
2. 從 Event type (事件類型)，選擇一或多個事件。
3. 若要篩選事件觸發組建的時間，請在 Start a build under these conditions (在這些情況下開始組建) 下新增一或多個選用的篩選條件。
4. 若要篩選何時不觸發事件，請在 Don't start a build under these conditions (在這些情況下不開始組建) 下新增一或多個選用的篩選條件。
5. 選擇 Add filter group (新增篩選群組) 新增另一個篩選群組。

如需詳細資訊，請參閱《AWS CodeBuild API 參考》中的 [建立組建專案 \(主控台\)](#)和 [WebhookFilter](#)。

在這個範例中，Webhook 篩選群組僅針對提取請求觸發組建：

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PULL_REQUEST_CREATED](#) ✕[PULL_REQUEST_UPDATED](#) ✕[PULL_REQUEST_MERGED](#) ✕[PULL_REQUEST_CLOSED](#) ✕

► Start a build under these conditions - optional

► Don't start a build under these conditions - optional

在有兩個篩選群組的範例中，當一個或兩個篩選群組評估為 true 時，就會觸發組建：

- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而標頭參考符合 `^refs/heads/branch1!`。
- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/branch1$`。

Webhook event filter group 1

Event type

Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PULL_REQUEST_CREATED X

PULL_REQUEST_UPDATED X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

^refs/heads/branch1\$

BASE_REF - optional

^refs/heads/main\$

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

Webhook event filter group 2

Remove filter group

Event type

Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

^refs/heads/branch1\$

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

在這個範例中，Webhook 篩選群組針對所有請求 (標籤事件除外) 觸發組建。

Filter group 1

Remove filter group

Event type
Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH X PULL_REQUEST_CREATED X PULL_REQUEST_UPDATED X

PULL_REQUEST_MERGED X PULL_REQUEST_CLOSED X

► Start a build under these conditions - optional

▼ Don't start a build under these conditions - optional

Add filter

Filter 1

Type

HEAD_REF ▼

Pattern

^refs/tags/.*

在這個範例中，只有在檔案名稱符合規則表達式 `^buildspec.*` 的檔案變更時，Webhook 篩選群組才會觸發組建。

Webhook event filter group 1

Event type

PUSH X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

在此範例中，Webhook 篩選群組只會在 src 或 test 資料夾變更檔案時觸發組建。

Webhook event filter group 1

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

在這個範例中，只有當 Bitbucket 使用者進行變更，而且其帳戶 ID 不符合規則表達式 actor-account-id 時，Webhook 篩選群組才會觸發建置。

Note

有關如何尋找您的 Bitbucket 帳戶 ID 的資訊，請參閱 <https://api.bitbucket.org/2.0/users/user-name>，其中 *user-name* 是您的 Bitbucket 使用者名稱。

Filter group 1[Remove filter group](#)**Event type**

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH ✕

PULL_REQUEST_CREATED ✕

PULL_REQUEST_UPDATED ✕

PULL_REQUEST_MERGED ✕

PULL_REQUEST_CLOSED ✕

▼ Start a build under these conditions - optional[Add filter](#)**Filter 2****Type**

ACTOR_ACCOUNT_ID ▼

Pattern

actor-account-id

在此範例中，當 head 提交訊息符合規則運算式 `\[CodeBuild\]` 時，webhook 篩選群組會觸發推送事件的建置。

Webhook event filter group 1

Event type

PUSH X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

篩選 Bitbucket Webhook 事件 (開發套件)

若要使用 AWS CodeBuild SDK 篩選 Webhook 事件，請使用 `CreateWebhook` 或 `UpdateWebhook` API 方法請求語法中的 `filterGroups` 欄位。如需詳細資訊，請參閱 CodeBuild API 參考中的 [WebhookFilter](#)。

若要建立 Webhook 篩選條件來只針對提取請求觸發組建，請在請求語法插入以下程式碼：

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED,  
PULL_REQUEST_CLOSED"  
    }  
  ]  
]
```

若要建立 Webhook 篩選條件來只針對指定的分支觸發組建，請使用 `pattern` 參數指定規則表達式來篩選分支名稱。在有兩個篩選群組的範例中，當一個或兩個篩選群組評估為 `true` 時，就會觸發組建：

- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而標頭參考符合 `^refs/heads/myBranch$`。

- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/myBranch$`。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_CLOSED"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    },  
    {  
      "type": "BASE_REF",  
      "pattern": "^refs/heads/main$"  
    }  
  ],  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    }  
  ]  
]
```

您可以使用 `excludeMatchedPattern` 參數來指定哪些事件不會觸發組建。在這個範例中，將針對所有請求 (標籤事件除外) 觸發組建。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,  
PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"  
    },  
    {  
      "type": "HEAD_REF",  

```

```
    "pattern": "^refs/tags/.*",
    "excludeMatchedPattern": true
  }
]
```

您可以建立篩選條件來指定只有在帳戶 ID 為 `actor-account-id` 的 Bitbucket 使用者進行變更時，才觸發組建。

Note

有關如何尋找您的 Bitbucket 帳戶 ID 的資訊，請參閱 <https://api.bitbucket.org/2.0/users/user-name>，其中 *user-name* 是您的 Bitbucket 使用者名稱。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
]
```

您可以建立篩選條件來指定只有在檔案名稱符合 `pattern` 引數中的規則表達式的檔案變更時，才觸發組建。在這個範例中，篩選群組指定只有在檔案名稱符合規則表達式 `^buildspec.*` 的檔案變更時，才觸發組建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
```

```
    "pattern": "^buildspec.*"
  }
]
]
```

在此範例中，篩選條件群組指定只有在 `src` 或 `test` 資料夾中變更檔案時，才會觸發組建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^src/.+|^test/.+"
    }
  ]
]
```

您可以建立篩選條件，只有在 `head` 提交訊息符合模式引數中的規則運算式時，才觸發組建。在這個範例中，篩選群組指定只有在推送事件的 `head` 遞交訊息符合規則表達式 `\[CodeBuild\]` 時，才觸發組建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "COMMIT_MESSAGE",
      "pattern": "\[CodeBuild\]"
    }
  ]
]
```

篩選 Bitbucket Webhook 事件 (CloudFormation)

若要使用 CloudFormation 範本篩選 Webhook 事件，請使用 AWS CodeBuild 專案的 `FilterGroups` 屬性。在 CloudFormation 範本中，以下 YAML 格式的部分建立兩個篩選群組。當其中一個或兩個評估為 `true` 時，它們會一起觸發組建：

- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而且是由帳戶 ID 不是 12345 的 Bitbucket 使用者所建立或更新。
- 第二個篩選群組指定分支上建立的推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/.*`。
- 第三個篩選群組會指定其 head 提交訊息符合規則運算式 `\[CodeBuild\]` 的推送請求。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: BITBUCKET
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*          - Type: FILE_PATH
            Pattern: READ_ME
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: COMMIT_MESSAGE
```

```
Pattern: \[CodeBuild\  
- Type: FILE_PATH  
Pattern: ^src/.+|^test/.+
```

GitHub 全域和組織 Webhook

您可以使用 CodeBuild GitHub 全域或組織 Webhook，從 GitHub 組織或企業內的任何儲存庫開始建置 Webhook 事件。全域和組織 Webhook 可與任何現有的 GitHub Webhook 事件類型搭配使用，而且可以在建立 CodeBuild Webhook 時新增範圍組態來設定。您也可以使用全域和組織 Webhook 在 [CodeBuild 中設定自我託管的 GitHub 動作執行器](#)，以便在單一專案中接收來自多個儲存庫 WORKFLOW_JOB_QUEUED 的事件。

主題

- [設定全域或組織 GitHub Webhook](#)
- [篩選 GitHub 全域或組織 Webhook 事件（主控台）](#)
- [篩選 GitHub 組織 Webhook 事件 \(CloudFormation\)](#)

設定全域或組織 GitHub Webhook

設定全域或組織 GitHub Webhook 的高階步驟如下所示。如需全域和組織 GitHub Webhook 的詳細資訊，請參閱 [GitHub 全域和組織 Webhook](#)。

1. 將專案的來源位置設定為 CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION。
2. 在 Webhook 的範圍組態中，將範圍設定為 GITHUB_ORGANIZATION 或 GITHUB_GLOBAL，取決於該範圍是組織還是 [全域 Webhook](#)。如需詳細資訊，請參閱 [Webhook 的類型](#)。
3. 指定名稱做為 Webhook 範圍組態的一部分。對於組織 Webhook，這是組織名稱，而對於全域 Webhook，這是企業名稱。

Note

如果專案的來源類型為 GITHUB_ENTERPRISE，您也需要指定網域做為 Webhook 範圍組態的一部分。

4. （選用）如果您只想要接收組織或企業內特定儲存庫的 Webhook 事件，您可以在建立 Webhook 時指定 REPOSITORY_NAME 做為篩選條件。

5. 如果您要建立組織 Webhook，請確保 CodeBuild 具有在 GitHub 中建立組織層級 Webhook 的許可。您可以建立具有組織 Webhook 許可的 GitHub 個人存取字符，或使用 CodeBuild OAuth。如需詳細資訊，請參閱[GitHub 和 GitHub Enterprise Server 存取字符](#)。

請注意，組織 Webhook 可與任何現有的 GitHub Webhook 事件類型搭配使用。

6. 如果您要建立全域 Webhook，則需要手動建立 Webhook。如需如何在 GitHub 中手動建立 Webhook 的詳細資訊，請參閱[GitHub 手動 Webhook](#)。

請注意，全域 Webhook 僅支援 WORKFLOW_JOB_QUEUED 事件類型。如需詳細資訊，請參閱[教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。

篩選 GitHub 全域或組織 Webhook 事件（主控台）

透過主控台建立 GitHub 專案時，請選取下列選項，以在專案內建立 GitHub 全域或組織 Webhook。如需全域和組織 GitHub Webhook 的詳細資訊，請參閱[GitHub 全域和組織 Webhook](#)。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱[建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
 - 在 Source (來源) 中：
 - 針對來源提供者，選擇 GitHub 或 GitHub Enterprise。
 - 針對儲存庫，選擇 GitHub 範圍 Webhook。

GitHub 儲存庫會自動設定為 CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION，這是全域和組織 Webhook 所需的來源位置。

Note

如果您使用的是組織 Webhook，請確定 CodeBuild 具有在 GitHub 中建立組織層級 Webhook 的許可。如果您使用的是[現有的 OAuth 連線](#)，您可能需要重新產生連線，才能授予 CodeBuild 此許可。或者，您可以使用[CodeBuild 手動 Webhook 功能手動建立 Webhook](#)。請注意，如果您有現有的 GitHub OAuth 權杖，並想要新增其他組織許可，則可以[撤銷 OAuth 權杖的許可](#)，並透過 CodeBuild 主控台重新連接權杖。

Source

Add source

Source 1 - Primary

Source provider

GitHub ▼

Repository

☐ Repository in my GitHub account☐ Public repository☒ GitHub scoped webhook

GitHub repository

CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION

Connection status

You are connected to GitHub using a personal access token.

Disconnect from GitHub

- 在主要來源 Webhook 事件中：
 - 對於範圍類型，如果您要建立組織 Webhook，請選擇組織層級；如果您要建立全域 Webhook，請選擇企業層級。
 - 對於名稱，輸入企業或組織名稱，取決於 Webhook 是全域還是組織 Webhook。

如果專案的來源類型為 GITHUB_ENTERPRISE，您也需要指定網域做為 Webhook 組織組態的一部分。例如，如果您組織的 URL 為 **https://domain.com/orgs/org-name**，則網域為 **https://domain.com**。

 Note

建立 Webhook 之後，就無法變更此名稱。若要變更名稱，您可以刪除並重新建立 Webhook。如果您想要完全移除 Webhook，您也可以將專案來源位置更新為 GitHub 儲存庫。

Primary source webhook events [Info](#)[Add filter group](#)Webhook - *optional* [Info](#) ☒ Rebuild every time a code change is pushed to this repository

Scope type

☒ Organization level☐ Enterprise level

Organization name

Your GitHub organization name.

Build type

☒ Single build
Triggers single build☐ Batch build
Triggers multiple builds as single execution**► Additional configuration**

- (選用) 在 Webhook 事件篩選條件群組中，您可以指定[要觸發新組建的事件](#)。您也可以指定 REPOSITORY_NAME 做為篩選條件，以僅觸發來自特定儲存庫的 Webhook 事件建置。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

[Remove filter group](#)

Event type - *optional*

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

WORKFLOW_JOB_QUEUED ✕

▼ Start a build under these conditions - *optional*

[Add filter](#)

Filter 1

Type

REPOSITORY_NAME ▼

Pattern

repository-name

[Remove](#)

您也可以將事件類型設定為 WORKFLOW_JOB_QUEUED，以設定自我託管的 GitHub 動作執行器。如需詳細資訊，請參閱[教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。

3. 繼續執行預設值，然後選擇建立建置專案。

篩選 GitHub 組織 Webhook 事件 (CloudFormation)

若要使用 CloudFormation 範本篩選組織 Webhook 事件，請使用 AWS CodeBuild 專案的 ScopeConfiguration 屬性。如需全域和組織 GitHub Webhook 的詳細資訊，請參閱 [GitHub 全域和組織 Webhook](#)。

Note

不支援全域 Webhook 和 GitHub Enterprise Webhook CloudFormation。

範本的下列 YAML 格式部分 CloudFormation 會建立四個篩選群組。當一個或全部評估為 true 時，它們會一起觸發組建：

- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而且是由帳戶 ID 不是 12345 的 GitHub 使用者所建立或更新。
- 第二個篩選群組指定在分支中的檔案上建立的推送請求，並且這些檔案的名稱符合規則表達式 `README`，而分支的 Git 參考名稱符合規則表達式 `^refs/heads/.*`。
- 第三個篩選群組會指定其 head 提交訊息符合規則運算式 `\[CodeBuild\]` 的推送請求。
- 第四個篩選條件群組會指定工作流程名稱符合規則表達式的 GitHub Actions 工作流程任務請求 `\[CI-CodeBuild\]`。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: source-location
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: organization-name
        Scope: GITHUB_ORGANIZATION
    FilterGroups:
      - - Type: EVENT
        Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
      - Type: BASE_REF
        Pattern: ^refs/heads/main$
        ExcludeMatchedPattern: false
      - Type: ACTOR_ACCOUNT_ID
        Pattern: 12345
        ExcludeMatchedPattern: true
      - - Type: EVENT
        Pattern: PUSH
      - Type: HEAD_REF
        Pattern: ^refs/heads/.*      - Type: FILE_PATH
```

```
    Pattern: READ_ME
    ExcludeMatchedPattern: true
- - Type: EVENT
    Pattern: PUSH
- - Type: COMMIT_MESSAGE
    Pattern: \[CodeBuild\]
- - Type: FILE_PATH
    Pattern: ^src/.+|^test/.+
- - Type: EVENT
    Pattern: WORKFLOW_JOB_QUEUED
- - Type: WORKFLOW_NAME
    Pattern: \[CI-CodeBuild\]
```

GitHub 手動 Webhook

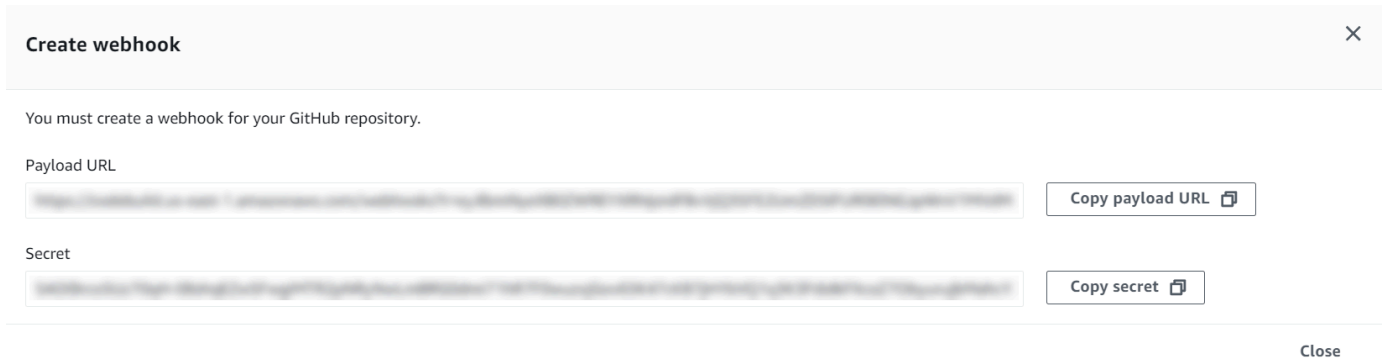
您可以設定手動 GitHub Webhook，以防止 CodeBuild 自動嘗試在 GitHub 中建立 Webhook。CodeBuild 會在呼叫中傳回承載 URL 來建立 Webhook，可用於在 GitHub 中手動建立 Webhook。即使 CodeBuild 不允許在 GitHub 帳戶中建立 Webhook，您仍然可以手動為建置專案建立 Webhook。

使用下列程序來建立 GitHub 手動 Webhook。

建立 GitHub 手動 Webhook

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
 - 在 Source (來源) 中：
 - 針對 Source provider (來源供應商)，選擇 GitHub。
 - 在儲存庫中，選擇 GitHub 帳戶中的儲存庫。
 - 針對 Repository URL (儲存庫 URL)，輸入 **`https://github.com/user-name/repository-name`**。
 - 在主要來源 Webhook 事件中：
 - 對於 Webhook - 選用，每次將程式碼變更推送至此儲存庫時，請選擇重建。
 - 選擇其他組態，對於手動建立 - 選用，選擇在 GitHub 主控台中手動為此儲存庫建立 Webhook。

- 繼續執行預設值，然後選擇建立建置專案。請記下承載 URL 和秘密值，因為您稍後會使用這些值。



- 在開啟 GitHub 主控台 <https://github.com/user-name/repository-name/settings/hooks>，然後選擇新增 Webhook。
 - 針對承載 URL，輸入您先前記下的承載 URL 值。
 - 針對內容類型，選擇應用程式/json。
 - 針對機密，輸入您先前記下的機密值。
 - 設定將 Webhook 承載傳送至 CodeBuild 的個別事件。對於您要觸發此 Webhook 的事件有哪些？，請選擇讓我選取個別事件，然後從下列事件中選擇：推送、提取請求和發行。如果您想要啟動 WORKFLOW_JOB_QUEUED 事件的組建，請選擇工作流程任務。若要進一步了解 GitHub Actions 執行器，請參閱 [教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。若要進一步了解 CodeBuild 支援的事件類型，請參閱 [GitHub Webhook 事件](#)。
- 選擇新增 Webhook。

GitHub Webhook 事件

您可以使用 Webhook 篩選群組來指定哪些 GitHub Webhook 事件會觸發組建。例如，您可以指定僅針對特定分支的變更觸發組建。

您可以建立一或多個 Webhook 篩選群組來指定哪些 Webhook 事件會觸發組建。如果任何篩選條件群組評估為 true，則會觸發組建，這會在群組中的所有篩選條件評估為 true 時發生。當您建立篩選群組時，您可以指定這些項目：

事件

對於 GitHub，您可以選擇下列一或多個事

件：PUSH、PULL_REQUEST_CREATED、PULL_REQUEST_UPDATED、PULL_REQUEST_REOPENED、PULL_REQUEST_RELEASED 和 WORKFLOW_JOB_QUEUED。Webhook 事件類型位在 Webhook 承載的 X-

GitHub-Event 標頭中。在 X-GitHub-Event 標頭中，您可能會看到 pull_request 或 push。若為提取請求事件，類型位在 Webhook 事件承載的 action 欄位中。下表顯示 X-GitHub-Event 標頭值和 Webhook 提取請求承載 action 欄位值如何映射到可用的事件類型。

X-GitHub-Event 標頭值	Webhook 事件承載 action 值	事件類型
pull_request	opened	PULL_REQUEST_CREATED
pull_request	reopened	PULL_REQUEST_REOPENED
pull_request	synchronize	PULL_REQUEST_UPDATED
pull_request	closed，而 merged 欄位為 true	PULL_REQUEST_MERGED
pull_request	closed，而 merged 欄位為 false	PULL_REQUEST_CLOSED
push	N/A	PUSH
release	已發行	RELEASED
release	預先發行	PRERELEASED
workflow_job	queued	WORKFLOW_JOB_QUEUED

Note

PULL_REQUEST_REOPENED 事件類型僅能搭配 GitHub 和 GitHub Enterprise Server 使用。RELEASED 和 PRERELEASED 事件類型只能與 GitHub 搭配使用。如需 WORKFLOW_JOB_QUEUED 的詳細資訊，請參閱[教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。

一或多個選用篩選條件

使用規則表達式來指定篩選條件。若要讓事件觸發組建，與其相關聯的群組中的每個篩選條件都必須評估為 true。

CodeBuild 使用 RE2 規則表達式語法來評估篩選條件模式，該語法不支援 lookahead、look behind、backreferences 或 atomic group。如需詳細資訊，請參閱 GitHub 網站上的 RE2 [語法](#) 頁面。

ACTOR_ACCOUNT_ID (ACTOR_ID 主控台中的)

當 GitHub 或 GitHub Enterprise Server 帳戶 ID 符合規則表達式模式時，Webhook 事件會觸發組建。此值位於 Webhook 承載之 sender 物件的 id 屬性中。

HEAD_REF

當頭部參考符合規則表達式模式（例如，refs/heads/branch-name 或）時，Webhook 事件會觸發組建 refs/tags/tag-name。若為推送事件，參考名稱位在 Webhook 承載的 ref 屬性中。若為提取請求事件，分支名稱位在 Webhook 承載之 head 物件的 ref 屬性中。

BASE_REF

當基本參考符合規則表達式模式（例如，）時，Webhook 事件會觸發組建 refs/heads/branch-name。BASE_REF 篩選條件僅可搭配提取請求事件使用。分支名稱位在 Webhook 承載之 base 物件的 ref 屬性中。

FILE_PATH

當變更的檔案路徑符合規則表達式模式時，Webhook 會觸發組建。FILE_PATH 篩選條件可以用於 GitHub 推送和提取請求事件和 GitHub Enterprise Server 推送事件。它不能與 GitHub Enterprise Server 提取請求事件一起使用。

Note

FILE_PATH 篩選條件會根據推送或提取請求中前 100 個變更的檔案進行評估。為確保篩選條件如預期套用，請將變更的檔案數量保持在此限制內。對於提取請求事件，我們也建議使用提取請求建置政策來控制哪些提取請求可以啟動建置。

COMMIT_MESSAGE

當前端遞交訊息符合規則表達式模式時，Webhook 會觸發組建。COMMIT_MESSAGE 篩選條件可以用於 GitHub 推送和提取請求事件和 GitHub Enterprise Server 推送事件。它不能與 GitHub Enterprise Server 提取請求事件一起使用。

TAG_NAME

當版本的標籤名稱符合規則表達式模式時，Webhook 會觸發組建。TAG_NAME 篩選條件可與 GitHub 發行和預先發行的請求事件搭配使用。

RELEASE_NAME

當版本名稱符合規則表達式模式時，Webhook 會觸發組建。RELEASE_NAME 篩選條件可與 GitHub 發行和預先發行的請求事件搭配使用。

REPOSITORY_NAME

當儲存庫名稱符合規則表達式模式時，Webhook 會觸發組建。REPOSITORY_NAME 篩選條件只能與 GitHub 全域或組織 Webhook 搭配使用。

ORGANIZATION_NAME

當組織名稱符合規則表達式模式時，Webhook 會觸發組建。ORGANIZATION_NAME 篩選條件只能與 GitHub 全域 Webhook 搭配使用。

WORKFLOW_NAME

當工作流程名稱符合規則表達式模式時，Webhook 會觸發組建。WORKFLOW_NAME 篩選條件可與 GitHub Actions 工作流程任務佇列請求事件搭配使用。

Note

您可以在您 GitHub 儲存庫的 Webhook 設定中找到 Webhook 承載。

主題

- [篩選 GitHub Webhook 事件 \(主控台\)](#)
- [篩選 GitHub Webhook 事件 \(開發套件\)](#)
- [篩選 GitHub Webhook 事件 \(CloudFormation\)](#)

篩選 GitHub Webhook 事件 (主控台)

使用下列指示來篩選使用的 GitHub Webhook 事件 AWS 管理主控台。如需 GitHub Webhook 事件的詳細資訊，請參閱 [GitHub Webhook 事件](#)。

在主要來源 Webhook 事件中，選取下列項目。只有當您在您的 GitHub 帳戶中為來源儲存庫選擇儲存庫時，才能使用本節。

1. 當您建立專案時，請選取 Rebuild every time a code change is pushed to this repository (在每次將程式碼變更推送至此儲存庫時重建)。
2. 從 Event type (事件類型)，選擇一或多個事件。
3. 若要篩選事件觸發組建的時間，請在 Start a build under these conditions (在這些情況下開始組建) 下新增一或多個選用的篩選條件。
4. 若要篩選何時不觸發事件，請在 Don't start a build under these conditions (在這些情況下不開始組建) 下新增一或多個選用的篩選條件。
5. 如有需要，請選擇新增篩選條件群組以新增另一個篩選條件群組。

如需詳細資訊，請參閱《AWS CodeBuild API 參考》中的 [建立組建專案 \(主控台\)](#)和 [WebhookFilter](#)。

在這個範例中，Webhook 篩選群組僅針對提取請求觸發組建：

The screenshot displays the 'Filter group 1' configuration in the AWS CodeBuild console. At the top right is a 'Remove filter group' button. Below the title, the 'Event type' section includes a descriptive text: 'Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.' A dropdown menu is present, followed by five event type buttons: 'PULL_REQUEST_CREATED', 'PULL_REQUEST_UPDATED', 'PULL_REQUEST_REOPENED', 'PULL_REQUEST_MERGED', and 'PULL_REQUEST_CLOSED', each with a close icon. At the bottom, there are two expandable sections: 'Start a build under these conditions - optional' and 'Don't start a build under these conditions - optional'.

在有兩個 Webhook 篩選群組的範例中，當一個或兩個篩選群組評估為 true 時，就會觸發組建：

- 第一個篩選群組指定在分支上建立、更新或重新開啟的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而標頭參考符合 `^refs/heads/branch1$`。
- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/branch1$`。

Webhook event filter group 1

Event type

Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PULL_REQUEST_CREATED ✕

PULL_REQUEST_UPDATED ✕

PULL_REQUEST_REOPENED ✕

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

^refs/heads/branch1\$

BASE_REF - optional

^refs/heads/main\$

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

Webhook event filter group 2

Remove filter group

Event type

Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH ✕

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

^refs/heads/branch1\$

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

在這個範例中，Webhook 篩選群組針對所有請求 (標籤事件除外) 觸發組建。

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PUSH](#) ✕[PULL_REQUEST_CREATED](#) ✕[PULL_REQUEST_UPDATED](#) ✕[PULL_REQUEST_REOPENED](#) ✕[PULL_REQUEST_MERGED](#) ✕[PULL_REQUEST_CLOSED](#) ✕

► Start a build under these conditions - *optional*

▼ Don't start a build under these conditions - *optional*

[Add filter](#)

Filter 1

Type

Pattern

在這個範例中，只有在檔案名稱符合規則表達式 `^buildspec.*` 的檔案變更時，Webhook 篩選群組才會觸發組建。

Webhook event filter group 1

Event type

PUSH X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

在此範例中，Webhook 篩選群組只會在 `src` 或 `test` 資料夾中變更檔案時觸發組建。

Webhook event filter group 1

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

► Don't start a build under these conditions

在這個範例中，只有當指定的 GitHub 或 GitHub Enterprise Server 使用者進行變更，而且其帳戶 ID 符合規則表達式 `actor-account-id` 時，Webhook 篩選群組才會觸發建置。

Note

有關如何尋找您的 GitHub 帳戶 ID 的資訊，請參閱 <https://api.github.com/users/user-name>，其中 *user-name* 是您的 GitHub 使用者名稱。

Filter group 1[Remove filter group](#)**Event type**

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH ✕

PULL_REQUEST_CREATED ✕

PULL_REQUEST_UPDATED ✕

PULL_REQUEST_REOPENED ✕

PULL_REQUEST_MERGED ✕

PULL_REQUEST_CLOSED ✕

▼ Start a build under these conditions - optional[Add filter](#)**Filter 2****Type**

ACTOR_ACCOUNT_ID ▼

Pattern

actor-account-id

[Remove](#)**► Don't start a build under these conditions - optional**

在此範例中，當 head 提交訊息符合規則運算式 `\[CodeBuild\]` 時，webhook 篩選群組會觸發推送事件的建置。

Webhook event filter group 1

Event type

PUSH X

▼ Start a build under these conditions

ACTOR_ID - optional

HEAD_REF - optional

BASE_REF - optional

FILE_PATH - optional

COMMIT_MESSAGE -
optional

\[CodeBuild\]

► Don't start a build under these conditions

在此範例中，Webhook 篩選群組只會觸發 GitHub Actions 工作流程任務事件的組建。

 Note

CodeBuild 只有在 Webhook 具有包含 WORKFLOW_JOB_QUEUED 事件篩選條件的篩選條件群組時，才會處理 GitHub Actions 工作流程任務。

Filter group 1

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

WORKFLOW_JOB_QUEUED X

► Start a build under these conditions - optional

► Don't start a build under these conditions - optional

在此範例中，Webhook 篩選群組會為符合規則表達式 的工作流程名稱觸發組建CI-CodeBuild。

Filter group 1

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

WORKFLOW_JOB_QUEUED X

▼ Start a build under these conditions - optional

Add filter

Filter 1

Type

WORKFLOW_NAME

Pattern

CI-CodeBuild

Remove

► Don't start a build under these conditions - optional

篩選 GitHub Webhook 事件 (開發套件)

若要使用 AWS CodeBuild SDK 篩選 Webhook 事件，請使用 `CreateWebhook` 或 `UpdateWebhook` API 方法請求語法中的 `filterGroups` 欄位。如需詳細資訊，請參閱 CodeBuild API 參考中的 [WebhookFilter](#)。

如需 GitHub Webhook 事件的詳細資訊，請參閱 [GitHub Webhook 事件](#)。

若要建立 Webhook 篩選條件來只針對提取請求觸發組建，請在請求語法插入以下程式碼：

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,  
PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"  
    }  
  ]  
]
```

```
]
```

若要建立 Webhook 篩選條件來只針對指定的分支觸發組建，請使用 `pattern` 參數指定規則表達式來篩選分支名稱。在有兩個篩選群組的範例中，當一個或兩個篩選群組評估為 `true` 時，就會觸發組建：

- 第一個篩選群組指定在分支上建立、更新或重新開啟的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而標頭參考符合 `^refs/heads/myBranch$`。
- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/myBranch$`。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,  
PULL_REQUEST_REOPENED"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    },  
    {  
      "type": "BASE_REF",  
      "pattern": "^refs/heads/main$"  
    }  
  ],  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    }  
  ]  
]
```

您可以使用 `excludeMatchedPattern` 參數來指定哪些事件不會觸發組建。例如，在這個範例中，將針對所有請求 (標籤事件除外) 觸發組建。


```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,  
PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/tags/.*",  
      "excludeMatchedPattern": true  
    }  
  ]  
]
```

您可以建立篩選條件來指定只有在檔案名稱符合 pattern 引數中的規則表達式的檔案變更時，才觸發組建。在這個範例中，篩選群組指定只有在檔案名稱符合規則表達式 `^buildspec.*` 的檔案變更時，才觸發組建。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "FILE_PATH",  
      "pattern": "^buildspec.*"  
    }  
  ]  
]
```

在此範例中，篩選條件群組指定只有在 `src` 或 `test` 資料夾中變更檔案時，才會觸發組建。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "FILE_PATH",
```

```
        "pattern": "^src/.+|^test/.+"
      }
    ]
  ]
```

您可以建立篩選條件來指定只有在指定的 GitHub 或 GitHub Enterprise Server 使用者進行變更，並且其帳戶 ID 為 actor-account-id 時，才觸發組建。

 Note

有關如何尋找您的 GitHub 帳戶 ID 的資訊，請參閱 <https://api.github.com/users/user-name>，其中 *user-name* 是您的 GitHub 使用者名稱。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
]
```

您可以建立篩選條件，只有在 head 提交訊息符合模式引數中的規則運算式時，才觸發組建。在這個範例中，篩選群組指定只有在推送事件的 head 遞交訊息符合規則表達式 `\[CodeBuild\]` 時，才觸發組建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "COMMIT_MESSAGE",
      "pattern": "\\[CodeBuild\\]"
    }
  ]
]
```

```
    }  
  ]  
]
```

若要建立僅觸發 GitHub 動作工作流程任務建置的 Webhook 篩選條件，請將以下內容插入請求語法：

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "WORKFLOW_JOB_QUEUED"  
    }  
  ]  
]
```

篩選 GitHub Webhook 事件 (CloudFormation)

若要使用 CloudFormation 範本篩選 Webhook 事件，請使用 AWS CodeBuild 專案的 `FilterGroups` 屬性。

如需 GitHub Webhook 事件的詳細資訊，請參閱 [GitHub Webhook 事件](#)。

在 CloudFormation 範本中，以下 YAML 格式的部分建立兩個篩選群組。當其中一個或兩個評估為 `true` 時，它們會一起觸發組建：

- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而且是由帳戶 ID 不是 12345 的 GitHub 使用者所建立或更新。
- 第二個篩選群組指定在分支中的檔案上建立的推送請求，並且這些檔案的名稱符合規則表達式 `README`，而分支的 Git 參考名稱符合規則表達式 `^refs/heads/.*`。
- 第三個篩選群組會指定其 head 提交訊息符合規則運算式 `\[CodeBuild\]` 的推送請求。
- 第四個篩選條件群組會指定工作流程名稱符合規則表達式 `\[CI-CodeBuild\]` 的 GitHub Actions 工作流程任務請求。

```
CodeBuildProject:  
  Type: AWS::CodeBuild::Project  
  Properties:  
    Name: MyProject  
    ServiceRole: service-role  
    Artifacts:
```

```
Type: NO_ARTIFACTS
Environment:
  Type: LINUX_CONTAINER
  ComputeType: BUILD_GENERAL1_SMALL
  Image: aws/codebuild/standard:5.0
Source:
  Type: GITHUB
  Location: source-location
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
      Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
    - - Type: BASE_REF
      Pattern: ^refs/heads/main$
      ExcludeMatchedPattern: false
    - - Type: ACTOR_ACCOUNT_ID
      Pattern: 12345
      ExcludeMatchedPattern: true
    - - Type: EVENT
      Pattern: PUSH
    - - Type: HEAD_REF
      Pattern: ^refs/heads/.+
    - - Type: FILE_PATH
      Pattern: README
      ExcludeMatchedPattern: true
    - - Type: EVENT
      Pattern: PUSH
    - - Type: COMMIT_MESSAGE
      Pattern: \[CodeBuild\]
    - - Type: FILE_PATH
      Pattern: ^src/.+|^test/.+
    - - Type: EVENT
      Pattern: WORKFLOW_JOB_QUEUED
    - - Type: WORKFLOW_NAME
      Pattern: \[CI-CodeBuild\]
```

GitLab 群組 Webhook

您可以使用 CodeBuild GitLab 群組 Webhook，從 GitLab 群組中的任何儲存庫開始建置 Webhook 事件。群組 Webhook 可與任何現有的 GitLab Webhook 事件類型搭配使用，而且可以在建立 CodeBuild Webhook 時新增範圍組態來設定。您也可以使用群組 Webhook 在 [CodeBuild 中設定自我託管的 GitLab 執行器](#)，以便在單一專案中接收來自多個儲存庫WORKFLOW_JOB_QUEUED的事件。

主題

- [設定群組 GitLab Webhook](#)
- [篩選 GitLab 群組 Webhook 事件 \(主控台\)](#)
- [篩選 GitLab 群組 Webhook 事件 \(CloudFormation\)](#)

設定群組 GitLab Webhook

設定群組 GitLab Webhook 的高階步驟如下所示。如需群組 GitLab Webhook 的詳細資訊，請參閱 [GitLab 群組 Webhook](#)。

1. 將專案的來源位置設定為 CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION。
2. 在 Webhook 的範圍組態中，將範圍設定為 GITLAB_GROUP。
3. 指定名稱做為 Webhook 範圍組態的一部分。對於群組 Webhook，這是群組名稱。

Note

如果專案的來源類型為 GITLAB_SELF_MANAGED，您也需要指定網域做為 Webhook 範圍組態的一部分。

4. (選用) 如果您只想要接收組織或企業內特定儲存庫的 Webhook 事件，您可以在建立 Webhook 時指定 REPOSITORY_NAME 做為篩選條件。
5. 建立群組 Webhook 時，請確定 CodeBuild 具有在 GitLab 中建立群組層級 Webhook 的許可。若要這樣做，您可以使用 CodeBuild OAuth AWS CodeConnections。如需詳細資訊，請參閱 [CodeBuild 中的 GitLab 存取](#)。

請注意，群組 Webhook 可與任何現有的 GitLab Webhook 事件類型搭配使用。

篩選 GitLab 群組 Webhook 事件 (主控台)

透過主控台建立 GitLab 專案時，請選取下列選項以在專案中建立 GitLab 群組 Webhook。如需群組 GitLab Webhook 的詳細資訊，請參閱 [GitLab 群組 Webhook](#)。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
 - 在 Source (來源) 中：

- 針對來源提供者，選擇 GitLab 或 GitLab 自我管理。
- 針對儲存庫，選擇 GitLab 範圍 Webhook。

GitLab 儲存庫會自動設定為 CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION，這是群組 Webhook 所需的來源位置。

 Note

使用群組 Webhook 時，請確定 CodeBuild 具有在 GitLab 中建立群組層級 Webhook 的許可。如果您使用的是[現有的 OAuth 連線](#)，您可能需要重新產生連線，才能授予 CodeBuild 此許可。

Source

[Add source](#)

Source 1 - Primary

Source provider

GitLab

Credential



Default source credential

Use your account's default source credential to apply to all projects



Custom source credential

Use a custom source credential to override your account's default settings

 Successfully connected through CodeConnections - [open resource](#)

[Manage default source credential](#)

Repository



Repository in my GitLab account



GitLab scoped webhook

Repository

CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION

- 在主要來源 Webhook 事件中：
- 針對群組名稱，輸入群組名稱。

如果專案的來源類型為 GITLAB_SELF_MANAGED，您也需要在 Webhook 群組組態中指定網域。例如，如果您群組的 URL 為 **https://domain.com/group/group-name**，則網域為 **https://domain.com**。

Note

建立 Webhook 之後，就無法變更此名稱。若要變更名稱，您可以刪除並重新建立 Webhook。如果您想要完全移除 Webhook，您也可以將專案來源位置更新為 GitLab 儲存庫。

Primary source webhook events [Info](#)[Add filter group](#)Webhook - *optional* [Info](#) ☒ Rebuild every time a code change is pushed to this repository

Group name

Your GitLab group name.

Build type

☒ Single build

Triggers single build

☐ Batch build

Triggers multiple builds as single execution

► Additional configuration

- (選用) 在 Webhook 事件篩選條件群組中，您可以指定[要觸發新組建的事件](#)。您也可以指定 REPOSITORY_NAME 做為篩選條件，以僅觸發來自特定儲存庫的 Webhook 事件建置。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

[Remove filter group](#)

Event type - *optional*

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

WORKFLOW_JOB_QUEUED ✕

▼ Start a build under these conditions - *optional*

[Add filter](#)

Filter 1

Type

REPOSITORY_NAME ▼

Pattern

repository-name

[Remove](#)

您也可以將事件類型設定為 WORKFLOW_JOB_QUEUED，以設定自我託管的 GitLab 執行器。
如需詳細資訊，請參閱[中的自我管理 GitLab 執行器 AWS CodeBuild](#)。

3. 繼續執行預設值，然後選擇建立建置專案。

篩選 GitLab 群組 Webhook 事件 (CloudFormation)

若要使用 CloudFormation 範本篩選群組 Webhook 事件，請使用 AWS CodeBuild 專案的 ScopeConfiguration 屬性。如需群組 GitLab Webhook 的詳細資訊，請參閱[GitLab 群組 Webhook](#)。

範本的下列 YAML 格式部分 CloudFormation 會建立四個篩選群組。當一個或全部評估為 true 時，它們會一起觸發組建：

- 第一個篩選條件群組指定在分支上建立或更新提取請求，其 Git 參考名稱符合沒有帳戶 ID 的 `^refs/heads/main$` GitLab 使用者的規則表達式 12345。
- 第二個篩選群組指定在分支中的檔案上建立的推送請求，並且這些檔案的名稱符合規則表達式 `README`，而分支的 Git 參考名稱符合規則表達式 `^refs/heads/.*`。
- 第三個篩選群組會指定其 head 提交訊息符合規則運算式 `\[CodeBuild\]` 的推送請求。

- 第四個篩選條件群組指定 GitLab CI/CD 管道任務請求，其 CI/CD 管道名稱符合規則表達式 `\[CI-CodeBuild\]`。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITLAB
      Location: source-location
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: group-name
        Scope: GITLAB_GROUP
    FilterGroups:
      - - Type: EVENT
          Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
        - Type: BASE_REF
          Pattern: ^refs/heads/main$
          ExcludeMatchedPattern: false
        - Type: ACTOR_ACCOUNT_ID
          Pattern: 12345
          ExcludeMatchedPattern: true
      - - Type: EVENT
          Pattern: PUSH
        - Type: HEAD_REF
          Pattern: ^refs/heads/. *
        - Type: FILE_PATH
          Pattern: READ_ME
          ExcludeMatchedPattern: true
      - - Type: EVENT
          Pattern: PUSH
        - Type: COMMIT_MESSAGE
          Pattern: \[CodeBuild\]
```

- Type: FILE_PATH
Pattern: ^src/.+|^test/.+
- - Type: EVENT
Pattern: WORKFLOW_JOB_QUEUED
- Type: WORKFLOW_NAME
Pattern: \[CI-CodeBuild\]

GitLab 手動 Webhook

您可以設定手動 GitLab Webhook，以防止 CodeBuild 自動嘗試在 GitLab 中建立 Webhook。CodeBuild 會在呼叫中傳回承載 URL 來建立 Webhook，可用於在 GitLab 中手動建立 Webhook。即使 CodeBuild 不允許在 GitLab 帳戶中建立 Webhook，您仍然可以手動為建置專案建立 Webhook。

使用下列程序來建立 GitLab 手動 Webhook。

建立 GitLab 手動 Webhook

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
 - 在 Source (來源) 中：
 - 針對來源提供者，選擇 GitLab。
 - 對於儲存庫，在我的 GitLab 帳戶中選擇儲存庫。
 - 針對 Repository URL (儲存庫 URL)，輸入 **`https://gitlab.com/user-name/repository-name`**。
 - 在主要來源 Webhook 事件中：
 - 對於 Webhook - 選用，每次將程式碼變更推送至此儲存庫時，請選擇重建。
 - 選擇其他組態，對於手動建立 - 選用，選擇在 GitLab 主控台中手動建立此儲存庫的 Webhook。
3. 繼續執行預設值，然後選擇建立建置專案。請記下承載 URL 和秘密值，因為您稍後會使用這些值。
4. 在開啟 GitLab 主控台 <https://gitlab.com/user-name/repository-name/-/hooks>，然後選擇新增 Webhook。
 - 針對 URL，輸入您先前記下的承載 URL 值。

- 針對機密字符，輸入您先前記下的機密值。
- 設定將 Webhook 承載傳送至 CodeBuild 的個別事件。針對觸發，從下列事件中選擇：推送事件、合併請求事件、發行事件和任務事件。若要進一步了解 CodeBuild 支援的事件類型，請參閱 [GitLab Webhook 事件](#)。

5. 選擇新增 Webhook。

GitLab Webhook 事件

您可以使用 Webhook 篩選群組來指定觸發組建的 GitLab Webhook 事件。例如，您可以指定僅針對特定分支的變更觸發組建。

您可以建立一或多個 Webhook 篩選群組來指定哪些 Webhook 事件會觸發組建。如果任何篩選條件群組評估為 true，則會觸發組建，這會在群組中的所有篩選條件評估為 true 時發生。當您建立篩選群組時，您可以指定這些項目：

事件

對於 GitLab，您可以選擇下列一或多個事

件：PUSH、PULL_REQUEST_CREATED、PULL_REQUEST_UPDATED、PULL_REQUEST_MERGED、PULL_REQUEST_REJECTED和 WORKFLOW_JOB_QUEUED。

Webhook 的事件類型位在 X-GitLab-Event 欄位的標頭中。下表顯示 X-GitLab-Event 標頭值如何映射到事件類型。對於 Merge Request Hook Webhook 事件，承載 `object_attributes.action` 將包含合併請求類型的其他資訊。

X-GitLab-Event 標頭值	object_attributes.action	事件類型
Push Hook	N/A	PUSH
Merge Request Hook	開啟	PULL_REQUEST_CREATED
Merge Request Hook	update	PULL_REQUEST_UPDATED
Merge Request Hook	merge	PULL_REQUEST_MERGED
Merge Request Hook	重新開啟	PULL_REQUEST_REOPENED

X-GitLab-Event 標頭值	object_attributes.action	事件類型
Merge Request Hook	關閉	PULL_REQUEST_CLOSED
Release Hook	建立、更新	RELEASED
Job Hook	N/A	WORKFLOW_JOB_QUEUED

對於 `PULL_REQUEST_MERGED`，如果提取請求與 squash 策略合併，且提取請求分支已關閉，則原始提取請求遞交不再存在。在此情況下，`CODEBUILD_WEBHOOK_MERGE_COMMIT` 環境變數會包含壓縮合併遞交的識別符。

一或多個選用篩選條件

使用規則表達式來指定篩選條件。若要讓事件觸發組建，與其相關聯的群組中的每個篩選條件都必須評估為 `true`。

CodeBuild 使用 RE2 規則表達式語法來評估篩選條件模式，該語法不支援 lookahead、look behind、backreferences 或 atomic group。如需詳細資訊，請參閱 GitHub 網站上的 RE2 [語法](#) 頁面。

`ACTOR_ACCOUNT_ID` (`ACTOR_ID` 主控台中的)

當 GitLab 帳戶 ID 符合規則表達式模式時，Webhook 事件會觸發組建。這個值會出現在 Webhook 篩選條件承載之 actor 物件的 `account_id` 屬性中。

`HEAD_REF`

當前端參考符合規則表達式模式（例如 `refs/heads/branch-name` 和）時，Webhook 事件會觸發組建 `refs/tags/tag-name`。`HEAD_REF` 篩選條件會評估分支或標籤的 Git 參考名稱。分支或標籤名稱會出現在 Webhook 承載 push 物件之 new 物件的 `name` 欄位中。針對提取請求事件，分支名稱會出現在 Webhook 承載 source 物件之 branch 物件的 `name` 欄位中。

`BASE_REF`

當基本參考符合規則表達式模式時，Webhook 事件會觸發組建。`BASE_REF` 篩選條件僅適用於提取請求事件（例如 `refs/heads/branch-name`）。`BASE_REF` 篩選條件會評估分支的 Git 參考名稱。分支名稱會出現在 Webhook 承載 destination 物件之 branch 物件的 `name` 欄位中。

FILE_PATH

當變更的檔案路徑符合規則表達式模式時，Webhook 會觸發組建。

Note

FILE_PATH 篩選條件會根據推送或提取請求中前 100 個變更的檔案進行評估。為了確保篩選條件如預期般套用，請將變更的檔案數量保持在此限制內。對於提取請求事件，我們也建議使用提取請求建置政策來控制哪些提取請求可以啟動建置。

COMMIT_MESSAGE

當前端遞交訊息符合規則表達式模式時，Webhook 會觸發組建。

WORKFLOW_NAME

當工作流程名稱符合規則表達式模式時，Webhook 會觸發組建。

Note

您可以在 GitLab 儲存庫的 Webhook 設定中找到 Webhook 承載。

主題

- [Filter GitLab Webhook 事件 \(主控台\)](#)
- [Filter GitLab Webhook 事件 \(SDK\)](#)
- [篩選 GitLab Webhook 事件 \(CloudFormation\)](#)

Filter GitLab Webhook 事件 (主控台)

使用下列指示來使用 AWS 管理主控台 來篩選 Webhook 事件。如需 GitLab Webhook 事件的詳細資訊，請參閱 [GitLab Webhook 事件](#)。

1. 當您建立專案時，請選取 Rebuild every time a code change is pushed to this repository (在每次將程式碼變更推送至此儲存庫時重建)。
2. 從 Event type (事件類型)，選擇一或多個事件。

3. 若要篩選事件觸發組建的時間，請在 Start a build under these conditions (在這些情況下開始組建) 下新增一或多個選用的篩選條件。
4. 若要篩選何時不觸發事件，請在 Don't start a build under these conditions (在這些情況下不開始組建) 下新增一或多個選用的篩選條件。
5. 選擇 Add filter group (新增篩選群組) 新增另一個篩選群組。

如需詳細資訊，請參閱《AWS CodeBuild API 參考》中的 [建立組建專案 \(主控台\)](#) 和 [WebhookFilter](#)。

在這個範例中，Webhook 篩選群組僅針對提取請求觸發組建：

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PULL_REQUEST_CREATED](#) ✕[PULL_REQUEST_UPDATED](#) ✕[PULL_REQUEST_MERGED](#) ✕

► Start a build under these conditions - *optional*

► Don't start a build under these conditions - *optional*

在有兩個篩選群組的範例中，當一個或兩個篩選群組評估為 true 時，就會觸發組建：

- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而標頭參考符合 `^refs/heads/branch1!`。
- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/branch1$`。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PULL_REQUEST_CREATED](#) ✕[PULL_REQUEST_UPDATED](#) ✕

▼ **Start a build under these conditions - optional**

[Add filter](#)

Filter 1

Type

Pattern

[Remove](#)

Filter 2

Type

Pattern

[Remove](#)

► **Don't start a build under these conditions - optional**

Filter group 2

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PUSH](#) ✕

Filter 1

Type

在這個範例中，Webhook 篩選群組針對所有請求 (標籤事件除外) 觸發組建。

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH ×PULL_REQUEST_CREATED ×PULL_REQUEST_UPDATED ×PULL_REQUEST_MERGED ×

► **Start a build under these conditions - optional**

▼ **Don't start a build under these conditions - optional**

[Add filter](#)

Filter 1

Type

Pattern

在這個範例中，只有在檔案名稱符合規則表達式 `^buildspec.*` 的檔案變更時，Webhook 篩選群組才會觸發組建。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PUSH X](#)

▼ Start a build under these conditions - *optional*

[Add filter](#)

Filter 1

Type

Pattern

[Remove](#)

► Don't start a build under these conditions - *optional*

在此範例中，Webhook 篩選群組只會在 `src` 或 `test` 資料夾變更檔案時觸發組建。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PUSH](#) [×](#)

▼ Start a build under these conditions - optional

[Add filter](#)

Filter 1

Type

Pattern

[Remove](#)

► Don't start a build under these conditions - optional

在此範例中，Webhook 篩選條件群組只會在 GitLab 使用者所做的變更中觸發組建，而該使用者的帳戶 ID 不符合規則表達式 `actor-account-id`。

Note

如需如何尋找 GitLab 帳戶 ID 的資訊，請參閱 <https://api.github.com/users/user-name>，其中 *user-name* 是您的 GitLab 使用者名稱。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PUSH](#) [×](#)

▼ Start a build under these conditions - optional

[Add filter](#)

Filter 1

Type

Pattern

[Remove](#)

► Don't start a build under these conditions - optional

在此範例中，當 head 提交訊息符合規則運算式 `\[CodeBuild\]` 時，webhook 篩選群組會觸發推送事件的建置。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

[Remove filter group](#)

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

[PUSH](#) [×](#)

▼ Start a build under these conditions - optional

[Add filter](#)

Filter 1

Type

Pattern

[Remove](#)

► Don't start a build under these conditions - optional

Filter GitLab Webhook 事件 (SDK)

若要使用 AWS CodeBuild SDK 篩選 Webhook 事件，請使用 `CreateWebhook` 或 `UpdateWebhook` API 方法請求語法中的 `filterGroups` 欄位。如需詳細資訊，請參閱 CodeBuild API 參考中的 [WebhookFilter](#)。

如需 GitLab Webhook 事件的詳細資訊，請參閱 [GitLab Webhook 事件](#)。

若要建立 Webhook 篩選條件來只針對提取請求觸發組建，請在請求語法插入以下程式碼：

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
```

```
    }  
  ]  
]
```

若要建立 Webhook 篩選條件來只針對指定的分支觸發組建，請使用 `pattern` 參數指定規則表達式來篩選分支名稱。在有兩個篩選群組的範例中，當一個或兩個篩選群組評估為 `true` 時，就會觸發組建：

- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而標頭參考符合 `^refs/heads/myBranch$`。
- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/myBranch$`。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    },  
    {  
      "type": "BASE_REF",  
      "pattern": "^refs/heads/main$"  
    }  
  ],  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    }  
  ]  
]
```

您可以使用 `excludeMatchedPattern` 參數來指定哪些事件不會觸發組建。在這個範例中，將針對所有請求 (標籤事件除外) 觸發組建。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,  
PULL_REQUEST_MERGED"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/tags/.*",  
      "excludeMatchedPattern": true  
    }  
  ]  
]
```

您可以建立篩選條件，只有在帳戶 ID 為 的 GitLab 使用者進行變更時，才會觸發組建actor-account-id。

 Note

如需如何尋找 GitLab 帳戶 ID 的資訊，請參閱 <https://api.github.com/users/user-name>，其中 *user-name* 是您的 GitLab 使用者名稱。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,  
PULL_REQUEST_MERGED"  
    },  
    {  
      "type": "ACTOR_ACCOUNT_ID",  
      "pattern": "actor-account-id"  
    }  
  ]  
]
```

您可以建立篩選條件來指定只有在檔案名稱符合 pattern 引數中的規則表達式的檔案變更時，才觸發組建。在這個範例中，篩選群組指定只有在檔案名稱符合規則表達式 `^buildspec.*` 的檔案變更時，才觸發組建。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "FILE_PATH",  
      "pattern": "^buildspec.*"  
    }  
  ]  
]
```

在此範例中，篩選條件群組指定只有在 `src` 或 `test` 資料夾中變更檔案時，才會觸發組建。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "FILE_PATH",  
      "pattern": "^src/.+|^test/.+"  
    }  
  ]  
]
```

您可以建立篩選條件，只有在 `head` 提交訊息符合模式引數中的規則運算式時，才觸發組建。在這個範例中，篩選群組指定只有在推送事件的 `head` 遞交訊息符合規則表達式 `\[CodeBuild\]` 時，才觸發組建。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PUSH"  
    },  
    {  
      "type": "COMMIT_MESSAGE",  
      "pattern": "\[CodeBuild\<]"  
    }  
  ]  
]
```

```
]
]
```

篩選 GitLab Webhook 事件 (CloudFormation)

若要使用 CloudFormation 範本篩選 Webhook 事件，請使用 AWS CodeBuild 專案的 `FilterGroups` 屬性。如需 GitLab Webhook 事件的詳細資訊，請參閱 [GitLab Webhook 事件](#)。

在 CloudFormation 範本中，以下 YAML 格式的部分建立兩個篩選群組。當其中一個或兩個評估為 `true` 時，它們會一起觸發組建：

- 第一個篩選條件群組指定在分支上建立或更新提取請求，其 Git 參考名稱符合沒有帳戶 ID 的 `^refs/heads/main$` GitLab 使用者的規則表達式 12345。
- 第二個篩選群組指定分支上建立的推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/.*`。
- 第三個篩選群組會指定其 head 提交訊息符合規則運算式 `\[CodeBuild\]` 的推送請求。
- 第四個篩選條件群組會指定工作流程名稱符合規則表達式的 GitHub Actions 工作流程任務請求 `\[CI-CodeBuild\]`。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITLAB
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - Type: EVENT
          Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
        - Type: BASE_REF
          Pattern: ^refs/heads/main$
```



```
ExcludeMatchedPattern: false
- Type: ACTOR_ACCOUNT_ID
  Pattern: 12345
  ExcludeMatchedPattern: true
- - Type: EVENT
    Pattern: PUSH
- Type: HEAD_REF
  Pattern: ^refs/heads/. *
- - Type: EVENT
    Pattern: PUSH
- Type: COMMIT_MESSAGE
  Pattern: \[CodeBuild\]
- - Type: EVENT
    Pattern: WORKFLOW_JOB_QUEUED
- Type: WORKFLOW_NAME
  Pattern: \[CI-CodeBuild\]
```

Buildkite 手動 Webhook

目前，CodeBuild 需要手動建立所有 Buildkite Webhook。CodeBuild 會在呼叫中傳回承載 URL 來建立 Webhook，可用於在 Buildkite 中手動建立 Webhook。

使用下列程序來建立 Buildkite 手動 Webhook。

使用 Webhook 建立 CodeBuild 專案

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 建立建置專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
3. 在專案組態中，選擇執行器專案。

在執行器中：

- 針對 Runner 供應商，選擇 Buildkite。
- 對於 Buildkite 代理程式字符，使用建立秘密頁面選擇建立新的代理程式字符。系統會提示您在 AWS Secrets Manager 中建立新的秘密，其秘密值等於您在上面產生的 Buildkite 代理程式字符。
- (選用) 如果您想要為任務使用 CodeBuild 受管登入資料，請在 Buildkite 來源登入資料選項下選取任務的來源儲存庫提供者，並確認已為您的帳戶設定登入資料。此外，請確認您的 Buildkite 管道使用 HTTPS 結帳。

4.
 - 在 Environment (環境) 中：
 - 選擇支援的環境映像和運算。請注意，您可以選擇使用 GitHub Actions 工作流程 YAML 中的標籤覆寫映像和執行個體設定。如需詳細資訊，請參閱[步驟 2：更新您的 GitHub 動作工作流程 YAML](#)
 - 在 Buildspec (建置規格) 中：
 - 請注意，除非buildspec-override:true新增為標籤，否則您的 buildspec 將被忽略。反之，CodeBuild 會覆寫它，以使用將設定自我託管執行器的命令。
5. 繼續執行預設值，然後選擇建立建置專案。
6. 從建立 Webhook 快顯視窗中儲存承載 URL 和秘密值。依照彈出視窗中的指示建立新的 Buildkite 組織 Webhook。

提取請求評論核准

CodeBuild 支援提取請求建置政策，可對提取請求觸發的建置提供額外控制。您可能不想自動建立來自未知使用者的提取請求，直到可以檢閱變更為止。此功能可讓您要求其中一位團隊成員先檢閱程式碼，然後執行管道。在建置未知參與者提交的程式碼時，這通常用作安全措施。

提取請求建置政策可讓您根據參與者的許可和核准狀態，控制 CodeBuild 何時觸發提取請求的建置。這對接受外部協作者貢獻的公有儲存庫或儲存庫來說特別重要。

啟用時，此功能可確保只有在下列任一情況下才會針對提取請求觸發組建：

- 提取請求是由信任的貢獻者建立。
- 信任的貢獻者會透過張貼特定註解來核准提取請求。

運作方式

信任的貢獻者

信任的貢獻者是使用者，其來源控制系統中的目前角色在提取請求型政策中設定為核准者角色。當信任的貢獻者建立提取請求時，CodeBuild 會自動觸發建置，以維護目前的行為。

不受信任的貢獻者

不受信任的貢獻者是使用者，其角色未在核准者角色清單中設定。當不受信任的貢獻者建立提取請求時：

1. CodeBuild 會將建置狀態標記為「失敗」，並顯示「啟動建置所需的提取請求核准」訊息。

2. 信任的貢獻者必須檢閱變更，並發佈評論給 `/codebuild_run(<SHA_OF_THE_LATEST_COMMIT>)` 以觸發建置。例如 `/codebuild_run(046e8b67481d53bdc86c3f6affdd5d1afae6d369)`。
3. CodeBuild 驗證評論者的許可，並在核准時觸發組建。
4. 建置結果會在提取請求頁面上回報。

註解核准語法

信任的貢獻者可以使用下列評論格式核准組建：

- `/codebuild_run(046e8b67481d53bdc86c3f6affdd5d1afae6d369)` - 根據指定的遞交 SHA 觸發建置。

組態

預設行為

預設會為所有新建立的 CodeBuild 專案啟用提取請求建置政策。

API 參數

提取請求建置政策是在下列動作中使用 `PullRequestBuildPolicy` 參數設定：

- `CreateWebhook`
- `UpdateWebhook`

`PullRequestBuildPolicy` 結構

```
{
  "requiresCommentApproval": "string",
  "approverRoles": ["string", ...]
}
```

`requiresCommentApproval`

指定在觸發提取請求建置之前，何時需要以註解為基礎的核准。此設定會決定建置是否自動執行，或透過註解要求明確核准。

類型：字串

有效值：

- `DISABLED` - 自動建置觸發程序，而不需要註解核准。

- `FORK_PULL_REQUESTS` - 僅從叉儲存庫提取請求需要評論核准（除非參與者是核准者角色之一）。
- `ALL_PULL_REQUESTS` - 所有提取請求都需要註解核准才能執行組建（除非參與者是核准者角色之一）。這是預設值。

approverRoles

需要評論核准時，具有提取請求組建核准權限的儲存庫角色清單。只有具有這些角色的使用者才能提供有效的評論核准。如果提取請求參與者是這些角色之一，其提取請求建置將自動觸發。

類型：字串陣列

GitHub 專案的有效值（這些值對應至 GitHub 角色）：

- `GITHUB_ADMIN` - 儲存庫管理員
- `GITHUB_MAINTAIN` - 儲存庫維護程式
- `GITHUB_WRITE` - 具有寫入許可的使用者
- `GITHUB_TRIAGE` - 具有分類許可的使用者
- `GITHUB_READ` - 具有讀取許可的使用者
- 預設：["`GITHUB_ADMIN`", "`GITHUB_MAINTAIN`", "`GITHUB_WRITE`"]

GitLab 專案的有效值（這些值會映射至 GitLab 角色）：

- `GITLAB_OWNER` - 儲存庫擁有者
- `GITLAB_MAINTAINER` - 儲存庫維護器
- `GITLAB_DEVELOPER` - 具有開發人員許可的使用者
- `GITLAB_REPORTER` - 具有報告者許可的使用者
- `GITLAB_PLANNER` - 具有規劃器許可的使用者
- `GITLAB_GUEST` - 具有訪客許可的使用者
- 預設：["`GITLAB_OWNER`", "`GITLAB_MAINTAINER`", "`GITLAB_DEVELOPER`"]

Bitbucket 專案的有效值（這些值對應至 Bitbucket 角色）：

- `BITBUCKET_ADMIN` - 儲存庫管理員
- `BITBUCKET_WRITE` - 具有寫入許可的使用者
- `BITBUCKET_READ` - 具有讀取許可的使用者
- 預設：["`BITBUCKET_ADMIN`", "`BITBUCKET_WRITE`"]

自訂 GitHub Enterprise 角色

CodeBuild 會根據授予自訂角色的最高許可層級，將自訂 GitHub Enterprise (GHE) 儲存庫角色映射至標準 GITHUB_* 值。CodeBuild 會評估從最高到最低權限的許可，而第一個啟用的許可會決定映射的角色。

- admin → GITHUB_ADMIN
- maintain → GITHUB_MAINTAIN
- push (寫入) → GITHUB_WRITE
- triage → GITHUB_TRIAGE
- pull (讀取) → GITHUB_READ

例如，授予 push 和 triage 許可映射至 的自訂角色，GITHUB_WRITE 因為 push 具有比 更高的權限 triage。

如果自訂角色沒有可辨識的許可（管理員、維護、推送、分類、提取所有停用），則使用者不符合任何核准者角色，並被視為不受信任的貢獻者。

標準 GitHub 角色（管理員、維護、寫入、分類、讀取）會直接映射至其對應的 GITHUB_* 值，無需以許可為基礎的解析。

範例

為所有提取請求啟用註解核准

若要使用 AWS CodeBuild SDK 啟用或停用 Webhook 的提取請求建置政策，請使用 CreateWebhook 或 UpdateWebhook API 方法請求語法中的 pullRequestBuildPolicy 欄位。如需詳細資訊，請參閱 CodeBuild API 參考中的 [WebhookFilter](#)。

具有 GitHub 角色的使用者管理、維護和寫入將被視為信任的貢獻者。

```
"pullRequestBuildPolicy": {
  "requiresCommentApproval": "ALL_PULL_REQUESTS",
  "approverRoles": ["GITHUB_ADMIN", "GITHUB_MAINTAIN", "GITHUB_WRITE"]
}
```

僅針對儲存庫管理員和維護程式啟用註解核准

具有 GitHub 角色管理員和維護的使用者將被視為信任的貢獻者。

```
"pullRequestBuildPolicy": {
```

```
"requiresCommentApproval": "FORK_PULL_REQUESTS",
"approverRoles": ["GITHUB_ADMIN", "GITHUB_MAINTAIN"]
}
```

停用註解核准

```
"pullRequestBuildPolicy": {
  "requiresCommentApproval": "DISABLED"
}
```

AWS CloudFormation

若要使用 AWS CloudFormation 範本啟用或停用 Webhook 的提取請求建置政策，請使用 PullRequestBuildPolicy 屬性。下列 an AWS CloudFormation 範本的 YAML 格式部分會建立具有 Webhook 的專案，該 Webhook 已為所有提取請求啟用提取請求建置政策。維護和管理角色指定為核准者。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: BITBUCKET
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - Type: EVENT
          Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
        - Type: BASE_REF
          Pattern: ^refs/heads/main$
          ExcludeMatchedPattern: false
    PullRequestBuildPolicy:
      RequiresCommentApproval: ALL_PULL_REQUESTS
```

```
ApproverRoles:
  - GITHUB_MAINTAIN
  - GITHUB_ADMIN
```

主控台組態

若要使用 AWS 管理主控台篩選 Webhook 事件：

1. 針對註解核准，請針對所有提取請求 (ALL_PULL_REQUEST) 選取停用或啟用，或僅針對又 () 的提取請求 FORK_PULL_REQUEST 選取。
2. 對於核准者角色，選取在需要評論核准時具有提取請求組建核准權限的儲存庫角色。

如需詳細資訊，請參閱 CodeBuild API 參考中的 [建立組建專案 \(主控台\)](#) 和 [WebhookFilter](#)。

▼ **Primary source webhook events** [Info](#)

Webhook - optional [Info](#) [↗](#)

☒ Rebuild every time a code change is pushed to this repository

Build type

☒ **Single build**
Triggers single build

☐ **Batch build**
Triggers multiple builds as single execution

Comment approval

ALL_PULL_REQUESTS ▼

Approver roles

▼

BITBUCKET_WRITE ✕ BITBUCKET_ADMIN ✕

► **Webhook event filter groups** [Add filter group](#)

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

在 中檢視建置專案的詳細資訊 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台 AWS CLI 或 AWS SDKs 在 CodeBuild 中檢視組建專案的詳細資訊。

主題

- [檢視建置專案的詳細資訊 \(主控台\)](#)
- [檢視建置專案的詳細資訊 \(AWS CLI\)](#)
- [檢視建置專案的詳細資訊 \(AWS SDKs\)](#)

檢視建置專案的詳細資訊 (主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。

Note

根據預設，只會顯示最新的 10 個組建專案。若要檢視更多組建專案，請選擇齒輪圖示，然後針對 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

3. 在組建專案清單中的 Name (名稱) 欄中，選擇組建專案的連結。
4. 在建置專案：*project-name* 頁面上，選擇建置詳細資訊。

檢視建置專案的詳細資訊 (AWS CLI)

執行 batch-get-projects 命令：

```
aws codebuild batch-get-projects --names names
```

在上述命令中，取代下列預留位置：

- *names*：必要字串，用來表示要檢視詳細資訊的一個或多個組建專案名稱。若要指定一個以上的組建專案，請以空格將每個組建專案名稱分開。您最多可以指定 100 個組建專案名稱。若要取得組建專案清單，請參閱[檢視建置專案名稱清單 \(AWS CLI\)](#)。

例如，如果您執行此命令：

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-project2  
my-other-demo-project
```


類似下列的結果可能會顯示於輸出。省略符號 (...) 用來代表為了簡潔起見而省略的資料。

```
{
  "projectsNotFound": [
    "my-other-demo-project"
  ],
  "projects": [
    {
      ...
      "name": codebuild-demo-project,
      ...
    },
    {
      ...
      "name": codebuild-demo-project2",
      ...
    }
  ]
}
```

在先前的輸出中，`projectsNotFound` 陣列會列出已指定但找不到的任何組建專案名稱。`projects` 陣列會列出每個找到資訊的組建專案詳細資訊。為簡潔起見，組建專案詳細資訊已在前述輸出中省略。如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#) 的輸出。

`batch-get-projects` 命令不支援篩選特定屬性值，但您可以撰寫指令碼來列舉專案的屬性。例如，下列 Linux shell 指令碼會列舉目前帳戶目前區域中的專案，並列印每個專案所使用的映像。

```
#!/usr/bin/sh

# This script enumerates all of the projects for the current account
# in the current region and prints out the image that each project is using.

imageName=""

function getImageName(){
  local environmentValues=(${1//'\t'/ })
  imageName=${environmentValues[1]}
}

function processProjectInfo() {
  local projectInfo=$1
```

```
while IFS=$'\t' read -r section value; do
    if [[ "$section" == *"ENVIRONMENT"* ]]; then
        getImageName "$value"
    fi
done <<< "$projectInfo"
}

# Get the list of projects.
projectList=$(aws codebuild list-projects --output=text)

for projectName in $projectList
do
    if [[ "$projectName" != *"PROJECTS"* ]]; then
        echo "=====

        # Get the detailed information for the project.
        projectInfo=$(aws codebuild batch-get-projects --output=text --names
"$projectName")

        processProjectInfo "$projectInfo"

        printf 'Project "%s" has image "%s"\n' "$projectName" "$imageName"
    fi
done
```

如需 AWS CLI 搭配使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

檢視建置專案的詳細資訊 (AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

在 中檢視建置專案名稱 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台 AWS CLI或 AWS SDKs CodeBuild 中檢視組建專案的清單。

主題

- [檢視建置專案名稱清單 \(主控台\)](#)
- [檢視建置專案名稱清單 \(AWS CLI\)](#)
- [檢視組建專案名稱 \(AWS SDKs的清單\)](#)

檢視建置專案名稱清單 (主控台)

您可以在 主控台中檢視 AWS 區域中的建置專案清單。資訊包括名稱、來源提供者、儲存庫、最新建置狀態和說明 (如果有的話)。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。

Note

根據預設，只會顯示最新的 10 個組建專案。若要檢視更多組建專案，請選擇齒輪圖示，然後針對 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

檢視建置專案名稱清單 (AWS CLI)

執行 list-projects 命令：

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-token next-token
```

在上述命令中，取代下列預留位置：

- *sort-by*：選用字串，用於表示列出組建專案名稱時所依據的準則。有效值包含：
 - CREATED_TIME：根據每個組建專案的建立時間，列出組建專案名稱。
 - LAST_MODIFIED_TIME：根據每個組建專案上次變更資訊的時間，列出組建專案名稱。
 - NAME：根據每個組建專案的名稱，列出組建專案名稱。
- *sort-order*：選用字串，用於表示依據 *sort-by* 列出組建專案時所按照的順序。有效值包括 ASCENDING 與 DESCENDING。
- *next-token*：選用字串。在先前的執行中，如果清單具有超過 100 個項目，則只會傳回前 100 個項目，以及稱為 next token 的唯一字串。若要取得清單中下一個批次中的項目，請再次執行此命令，將 next token 新增至呼叫。若要取得清單中的所有項目，請繼續對每個後續的 next token 執行此命令，直到不再傳回 next token 為止。

例如，如果您執行此命令：

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

類似下列的結果可能會顯示於輸出：

```
{
  "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=",
  "projects": [
    "codebuild-demo-project",
    "codebuild-demo-project2",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project99"
  ]
}
```

如果您再次執行此命令：

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING --next-token
Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
```

類似下列的結果可能會顯示於輸出：

```
{
  "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
  ]
}
```

檢視組建專案名稱 (AWS SDKs的清單)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

在 中建置 AWS CodeBuild

組建代表執行的一組動作，以根據一組輸入成品（例如 Java 類別檔案的集合）AWS CodeBuild 建立輸出成品（例如 JAR 檔案）。

當您執行多個建置時，適用以下規則：

- 如果可能，建置會並行執行。並行執行的建置數量上限可能不同。如需詳細資訊，請參閱[的配額 AWS CodeBuild](#)。
- 如果組建專案已設定並行組建限制，當執行中的組建數量達到專案的並行組建限制時，組建會傳回錯誤。如需詳細資訊，請參閱[啟用並行建置限制](#)。
- 如果建置專案未設定並行建置限制，則如果執行中的建置數量達到平台和運算類型的並行建置限制，則會將建置排入佇列。佇列中建置的數量上限為並行建置限制的五倍。如需詳細資訊，請參閱[的配額 AWS CodeBuild](#)。

佇列中的建置若未在其指定的逾時值分鐘數之後開始，則會從佇列中移除。預設逾時值為八小時。執行建置時，您可以以介於五分鐘到八個小時之間的值覆寫建置佇列逾時。如需詳細資訊，請參閱[手動執行 AWS CodeBuild 組建](#)。

我們無法預測佇列建置的開始順序。

Note

您可以存取一年的組建歷史紀錄。

您可以在處理建置時，執行這些任務：

主題

- [手動執行 AWS CodeBuild 組建](#)
- [在 AWS Lambda 運算上執行組建](#)
- [在預留容量機群上執行組建](#)
- [批次執行組建](#)
- [在批次建置中執行平行測試](#)
- [快取建置以改善效能](#)

- [在中對組建進行偵錯 AWS CodeBuild](#)
- [在中刪除組建 AWS CodeBuild](#)
- [在中手動重試建置 AWS CodeBuild](#)
- [在中自動重試建置 AWS CodeBuild](#)
- [在中停止建置 AWS CodeBuild](#)
- [在中停止批次建置 AWS CodeBuild](#)
- [自動觸發 AWS CodeBuild 建置](#)
- [在中檢視建置詳細資訊 AWS CodeBuild](#)
- [檢視 中的組建 IDs清單 AWS CodeBuild](#)
- [在中檢視組建專案的組建 IDs 清單 AWS CodeBuild](#)

手動執行 AWS CodeBuild 組建

您可以使用 AWS CodeBuild 主控台 AWS CLI或 AWS SDKs CodeBuild 中執行組建。

主題

- [使用 AWS CodeBuild 代理程式在本機執行組建](#)
- [執行建置 \(主控台\)](#)
- [執行建置 \(AWS CLI\)](#)
- [執行批次建置 \(AWS CLI\)](#)
- [自動開始執行建置 \(AWS CLI\)](#)
- [自動停止執行建置 \(AWS CLI\)](#)
- [執行組建AWS SDKs\)](#)

使用 AWS CodeBuild 代理程式在本機執行組建

您可以使用 AWS CodeBuild 代理程式在本機電腦上執行 CodeBuild 組建。有適用於 x86_64 和 ARM 平台的代理程式。

您也可以訂閱，以在發佈新版本的代理程式時接收通知。

先決條件

開始之前，您需要執行下列動作：

- 在您的本機電腦上安裝 Git。
- 在本機電腦上安裝和設定 [Docker](#)。

設定建置映像

您只需要在第一次執行代理程式或映像變更時設定建置映像。

設定建置映像

1. 如果您想要使用精選的 Amazon Linux 2 映像，您可以使用下列命令，從位於 https://gallery.ecr.aws/codebuild/amazonlinux-x86_64-standard 的 CodeBuild 公有 Amazon ECR 儲存庫中提取映像：

```
$ docker pull public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0
```

或者，如果您想要使用另一個 Linux 映像，請執行下列步驟：

- a. 複製 CodeBuild 映像儲存庫：

```
$ git clone https://github.com/aws/aws-codebuild-docker-images.git
```

- b. 變更為映像目錄。在此範例中，請使用 `aws/codebuild/standard:5.0` 映像：

```
$ cd aws-codebuild-docker-images/ubuntu/standard/5.0
```

- c. 建置映像。這將需要數分鐘的時間。

```
$ docker build -t aws/codebuild/standard:5.0 .
```

2. 下載 CodeBuild 代理程式。

若要下載 x86_64 版本的代理程式，請執行下列命令：

```
$ docker pull public.ecr.aws/codebuild/local-builds:latest
```

若要下載代理程式的 ARM 版本，請執行下列命令：

```
$ docker pull public.ecr.aws/codebuild/local-builds:aarch64
```

3. CodeBuild 代理程式可從 <https://gallery.ecr.aws/codebuild/local-builds> 取得。

x86_64 版代理程式的安全雜湊演算法 (SHA) 簽章為：

```
sha256:ccb19bdd7af94e4dc761e4c58c267e9455c28ec68d938086b4dc1cf8fe6b0940
```

代理程式 ARM 版本的 SHA 簽章為：

```
sha256:7d7b5d35d2ac4e062ae7ba8c662ffed15229a52d09bd0d664a7816c439679192
```

您可以使用 SHA 來識別代理程式的版本。若要查看代理程式的 SHA 簽章，請執行下列命令，並在下尋找 SHARepoDigests：

```
$ docker inspect public.ecr.aws/codebuild/local-builds:latest
```

執行 CodeBuild 代理程式

執行 CodeBuild 代理程式

1. 變更至包含您建置專案來源的目錄。
2. 下載 [codebuild_build.sh](#) 指令碼：

```
$ curl -O https://raw.githubusercontent.com/aws/aws-codebuild-docker-images/master/local_builds/codebuild_build.sh
$ chmod +x codebuild_build.sh
```

3. 執行codebuild_build.sh指令碼並指定您的容器映像和輸出目錄。

若要執行 x86_64 組建，請執行下列命令：

```
$ ./codebuild_build.sh -i <container-image> -a <output directory>
```

若要執行 ARM 組建，請執行下列命令：

```
$ ./codebuild_build.sh -i <container-image> -a <output directory> -l
public.ecr.aws/codebuild/local-builds:aarch64
```

將 *<container-image>* 取代為容器映像的名稱，例如 aws/codebuild/standard:5.0 或 public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0。

指令碼會啟動建置映像，並在目前目錄中的專案上執行建置。若要指定建置專案的位置，請將 `-s <build project directory>` 選項新增至指令碼命令。

接收新 CodeBuild 代理程式版本的通知

您可以訂閱 Amazon SNS 通知，以便在發行新版本的 AWS CodeBuild 代理程式時收到通知。

訂閱 CodeBuild 代理程式通知

1. 在 <https://console.aws.amazon.com/sns/v3/home> 開啟 Amazon SNS 主控台。
2. 在導覽列中，如果尚未選取，請將 AWS 區域變更為美國東部（維吉尼亞北部）。您必須選取此 AWS 區域，因為您訂閱的 Amazon SNS 通知會在此區域中建立。
3. 在導覽窗格中，選擇訂閱。
4. 選擇建立訂閱。
5. 在建立訂閱中，執行下列動作：

- a. 對於 Topic ARN (主題 ARN)，請使用下列 Amazon Resource Name (ARN)：

```
arn:aws:sns:us-east-1:850632864840:AWS-CodeBuild-Local-Agent-Updates
```

- b. 針對通訊協定，選擇電子郵件 或簡訊。
- c. 針對 Endpoint (端點)，選擇要接收通知的位置 (電子郵件或簡訊)。輸入電子郵件或地址或電話號碼，包括區碼。
- d. 選擇建立訂閱。
- e. 選擇電子郵件以接收要求您確認訂閱的電子郵件。遵循電子郵件中的指示來完成訂閱。

如果您不想再接收這些通知，請使用下列程序來取消訂閱。

取消訂閱 CodeBuild 代理程式通知

1. 在 <https://console.aws.amazon.com/sns/v3/home> 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇訂閱。
3. 選取訂閱，然後從 Actions (動作)，選擇 Delete subscriptions (刪除訂閱)。出現確認提示時，選擇 Delete (刪除)。

執行建置 (主控台)

若要使用 AWS CodePipeline 搭配 CodeBuild 執行組建，請略過這些步驟並遵循 [中的指示](#) [搭配 CodePipeline 使用 CodeBuild CodePipeline](#)。

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 在建置專案清單中，選擇建置專案。
4. 您可以使用預設建置專案設定來執行建置，或僅覆寫此建置的建置設定。
 - a. 如果您想要使用預設建置專案設定執行建置，請選擇開始建置。建置會立即開始。
 - b. 如果您想要覆寫預設建置專案設定，請選擇使用覆寫開始建置。在開始建置頁面中，您可以覆寫下列項目：
 - 組建組態
 - 來源
 - 環境變數覆寫

如果您需要選取更多進階覆寫，請選擇進階建置覆寫。在此頁面上，您可以覆寫下列項目：

- 組建組態
- 來源
- Environment (環境)
- Buildspec
- 成品
- 日誌

當您進行覆寫選擇後，請選擇開始建置。

如需此建置的詳細資訊，請參閱 [檢視建置的詳細資訊 \(主控台\)](#)。

執行建置 (AWS CLI)

Note

若要使用 CodePipeline 搭配 執行組建 AWS CodeBuild，請略過這些步驟並遵循 中的指示[建立使用 CodeBuild 的管道 \(AWS CLI\)](#)。

如需 AWS CLI 搭配 CodeBuild 使用 的詳細資訊，請參閱 [命令列參考](#)。

1. 使用下列其中一種方式，執行 start-build 命令：

```
aws codebuild start-build --project-name <project-name>
```

如果您想要執行的建置使用建置輸入成品和組建專案現有設定的最新版本，則請使用此命令。

```
aws codebuild start-build --generate-cli-skeleton
```

如果您想要執行的建置使用建置輸入成品的舊版本，或者想要覆寫建置輸出成品、環境變數、buildspec 或預設建置逾時期間的設定，則請使用此命令。

2. 如果您使用 --project-name 選項執行 start-build 命令，請將 <project-name> 取代為建置專案的名稱，然後跳至此程序的步驟 6。若要取得組建專案清單，請參閱[檢視組建專案名稱](#)。
3. 如果您使用 --idempotency-token 選項執行 start-build 命令，start-build 則請求中會包含唯一的區分大小寫識別符或字符。此字符在 請求之後的 5 分鐘內有效。如果您使用相同的字符重複 start-build 請求，但變更參數，CodeBuild 會傳回參數不相符錯誤。
4. 如果您搭配執行 start-build 命令與 --generate-cli-skeleton 選項，則會在輸出中顯示 JSON 格式化資料。將資料複製到 AWS CLI 安裝 的本機電腦或執行個體位置中的檔案（例如 *start-build.json*）。修改複製的資料以符合下列格式，並儲存您的結果：

```
{
  "projectName": "projectName",
  "sourceVersion": "sourceVersion",
  "artifactsOverride": {
    "type": "type",
    "location": "location",
    "path": "path",
    "namespaceType": "namespaceType",
    "name": "artifactsOverride-name",
    "packaging": "packaging"
```

```

},
"buildspecOverride": "buildspecOverride",
"cacheOverride": {
  "location": "cacheOverride-location",
  "type": "cacheOverride-type"
},
"certificateOverride": "certificateOverride",
"computeTypeOverride": "computeTypeOverride",
"environmentTypeOverride": "environmentTypeOverride",
"environmentVariablesOverride": {
  "name": "environmentVariablesOverride-name",
  "value": "environmentVariablesValue",
  "type": "environmentVariablesOverride-type"
},
"gitCloneDepthOverride": "gitCloneDepthOverride",
"imageOverride": "imageOverride",
"idempotencyToken": "idempotencyToken",
"insecureSslOverride": "insecureSslOverride",
"privilegedModeOverride": "privilegedModeOverride",
"queuedTimeoutInMinutesOverride": "queuedTimeoutInMinutesOverride",
"reportBuildStatusOverride": "reportBuildStatusOverride",
"timeoutInMinutesOverride": "timeoutInMinutesOverride",
"sourceAuthOverride": "sourceAuthOverride",
"sourceLocationOverride": "sourceLocationOverride",
"serviceRoleOverride": "serviceRoleOverride",
"sourceTypeOverride": "sourceTypeOverride"
}

```

取代下列預留位置：

- *projectName*：必要字串。用於此建置的組建專案名稱。
- *sourceVersion*：選用字串。要建置的來源碼版本，如下所示：
 - 對於 Amazon S3，對應至您要建置之輸入 ZIP 檔案版本的版本 ID。如果未指定 *sourceVersion*，則會使用最新版本。
 - 對於 CodeCommit，對應至您要建置之來源碼版本的遞交 ID。如果未指定 *sourceVersion*，則會使用預設分支的 HEAD 遞交 ID。(您無法指定 *sourceVersion* 的標籤名稱，但可以指定標籤的遞交 ID)。
 - 針對 GitHub，為遞交 ID、提取請求 ID、分支名稱，或與您想要建置之來源碼版本對應的標籤名稱。如果指定提取請求 ID，其格式必須為 *pr/pull-request-ID* (例如，*pr/25*)。如果

指定分支名稱，則會使用分支的 HEAD 遞交 ID。如果未指定 *sourceVersion*，則會使用預設分支的 HEAD 遞交 ID。

- 針對 Bitbucket，為遞交 ID、分支名稱，或與您想要建置之來源碼版本對應的標籤名稱。如果指定分支名稱，則會使用分支的 HEAD 遞交 ID。如果未指定 *sourceVersion*，則會使用預設分支的 HEAD 遞交 ID。
- 下列是 *artifactsOverride* 的預留位置。
 - *type*：選用。此建置的建置輸出成品類型，可覆寫組建專案中所定義的建置輸出成品類型。
 - *location*：選用。此建置的建置輸出成品位置，可覆寫組建專案中所定義的建置輸出成品位置。
 - *##*：選用。此建置的建置輸出成品路徑，可覆寫組建專案中所定義的建置輸出成品路徑。
 - *namespaceType*：選用。此建置的建置輸出成品路徑類型，可覆寫組建專案中所定義的建置輸出成品路徑類型。
 - *name*：選用。此建置的建置輸出成品名稱，可覆寫組建專案中所定義的建置輸出成品名稱。
 - *##*：選用。此建置的建置輸出成品封裝類型，可覆寫組建專案中所定義的建置輸出成品封裝類型。
- *buildspecOverride*：選用。此建置的 Buildspec 宣告，可覆寫建置專案中所定義的 Buildspec 宣告。如果已設定此值，它可以是內嵌 buildspec 定義，或相對於內建 CODEBUILD_SRC_DIR 環境變數值之替代 buildspec 檔案的路徑，或 S3 儲存貯體的路徑。S3 儲存貯體必須位於與建置專案相同的 AWS 區域。使用其 ARN 指定 buildspec 檔案 (例如，arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml)。如果此值未提供，或設定為空字串，則來源碼必須包含其根目錄中的 buildspec.yml 檔案。如需詳細資訊，請參閱[Buildspec 檔案名稱和儲存位置](#)。
- 下列是 *cacheOverride* 的預留位置。
 - *cacheOverride-location*：選用。此建置的 ProjectCache 物件位置，可覆寫組建專案中所指定的 ProjectCache 物件。cacheOverride 是選用項目，並採用 ProjectCache 物件。ProjectCache 物件中需要 location。
 - *cacheOverride-type*：選用。此建置的 ProjectCache 物件類型，可覆寫組建專案中所指定的 ProjectCache 物件。cacheOverride 是選用項目，並採用 ProjectCache 物件。ProjectCache 物件中需要 type。
- *certificateOverride*：選用。此建置的憑證名稱，可覆寫組建專案中所指定的憑證名稱。
- *environmentTypeOverride*：選用。此建置的容器名稱，可覆寫組建專案中所指定的容器名稱。目前有效字串為 LINUX_CONTAINER。
- 下列是 *environmentVariablesOverride* 的預留位置。

- *environmentVariablesOverride-name* : 選用。您想要針對此建置覆寫其值之組建專案中的環境變數名稱。
- *environmentVariablesOverride-type* : 選用。您想要針對此建置覆寫其值之組建專案中的環境變數類型。
- *environmentVariablesValue* : 選用。您想要針對此建置覆寫組建專案中所定義的環境變數值。
- *gitCloneDepthOverride* : 選用。您想要針對此建置覆寫其值之組建專案中的 Git clone depth (Git 複製深度) 值。如果您的來源類型是 Amazon S3，則不支援此值。
- *imageOverride* : 選用。此建置的映像名稱，可覆寫組建專案中所指定的映像名稱。
- *idempotencyToken* : 選用。字串，可做為指定建置請求為等冪的字符。您可以選擇長度為 64 個或更少字元的任何字串。字符在 start-build 請求後 5 分鐘內有效。如果您使用相同的字符重複 start-build 請求，但變更參數，CodeBuild 會傳回參數不相符錯誤。
- *insecureSslOverride* : 選用布林值，指定是否覆寫建置專案中所指定的不安全 TLS 設定。不安全 TLS 設定決定是否忽略連線至專案來源碼時的 TLS 警告。只有在建置的來源是 GitHub Enterprise Server 時，才會套用此覆寫。
- *privilegedModeOverride* : 選用布林值。如果設定為 true，則建置會覆寫建置專案中的特殊權限模式。
- *queuedTimeoutInMinutesOverride* : 選用整數，指定在建置逾時之前，允許將建置排入佇列的分鐘數。其最小值為五分鐘，其最大值為 480 分鐘 (八小時)。
- *reportBuildStatusOverride* : 選用布林值，指定是否將建置的啟動和完成狀態傳送給您的來源提供者。如果您搭配 GitHub、GitHub Enterprise Server 或 Bitbucket 以外的來源供應商設定此項，則會擲出 `invalidInputException`。
- *sourceAuthOverride* : 選用字串。此建置的授權類型，可覆寫組建專案中所定義的授權類型。只有在建置專案的來源是 Bitbucket 或 GitHub 時，才會套用此覆寫。
- *sourceLocationOverride* : 選用字串。此建置的位置，可覆寫組建專案中所定義位置的來源位置。
- *serviceRoleOverride* : 選用字串。此建置的服務角色名稱，可覆寫組建專案中所指定的服務角色名稱。
- *sourceTypeOverride* : 選用字串。此建置的來源輸入類型，可覆寫組建專案中所定義的來源輸入。有效字串為 NO_SOURCE、CODECOMMIT、CODEPIPELINE、GITHUB、S3、BITBUCKET 和 GITHUB_ENTERPRISE。

- `timeoutInMinutesOverride`：選用數字。此建置的建置逾時分鐘數目，可覆寫組建專案中所定義的建置逾時分鐘數目。

我們建議您存放具有敏感值的環境變數，例如 AWS 存取金鑰 ID、AWS 私密存取金鑰或密碼，做為 Amazon EC2 Systems Manager 參數存放區中的參數。CodeBuild 只有在參數的名稱開頭為 `/CodeBuild/`（例如）時，才能使用存放在 Amazon EC2 Systems Manager 參數存放區中的參數 `/CodeBuild/dockerLoginPassword`。您可以使用 CodeBuild 主控台在 Amazon EC2 Systems Manager 中建立參數。選擇 **Create a parameter**（建立參數），然後遵循指示進行。（在該對話方塊中，對於 KMS 金鑰，您可以選擇指定帳戶中金鑰 AWS KMS 的 ARN。Amazon EC2 Systems Manager 使用此金鑰在儲存期間加密參數的值，並在擷取期間解密。）如果您使用 CodeBuild 主控台建立參數，主控台會在儲存參數 `/CodeBuild/` 時以啟動參數。不過，如果您使用 Amazon EC2 Systems Manager 參數存放區主控台來建立參數，您必須使用啟動參數的名稱 `/CodeBuild/`，而且必須將 Type 設定為 **Secure String**。如需詳細資訊，請參閱《Amazon EC2 Systems Manager 使用者指南》中的 [AWS Systems Manager 參數存放區](#) 和 [逐步解說：建立和測試字串參數（主控台）](#)。

如果您的建置專案參考 Amazon EC2 Systems Manager 參數存放區中存放的參數，則建置專案的服務角色必須允許 `ssm:GetParameters` 動作。如果您先前選擇在帳戶中建立新的服務角色，則 CodeBuild 會自動在建置專案的預設服務角色中包含此動作。不過，如果您選擇 **Choose an existing service role from your account**（從您的帳戶中選擇現有服務角色），則必須將此動作單獨包含在服務角色中。

您設定的環境變數會取代現有環境變數。例如，如果 Docker 影像已包含名為 `MY_VAR` 且值為 `my_value` 的環境變數，而且您設定名為 `MY_VAR` 且值為 `other_value` 的環境變數，則 `my_value` 會取代為 `other_value`。同樣地，如果 Docker 影像已包含名為 `PATH` 且值為 `/usr/local/sbin:/usr/local/bin` 的環境變數，而且您設定名為 `PATH` 且值為 `$PATH:/usr/share/ant/bin` 的環境變數，則 `/usr/local/sbin:/usr/local/bin` 會取代為文字值 `$PATH:/usr/share/ant/bin`。

請不要設定名稱開頭為 `CODEBUILD_` 的任何環境變數。此字首保留供內部使用。

如果有多個位置定義同名的環境變數，則會決定環境變數的值，如下所示：

- 開始建置操作呼叫中的值會採用最高優先順序。
- 組建專案定義中的值會採用下一個優先順序。
- `buildspec` 檔案宣告中的值會採用最低優先順序。

如需這些預留位置有效值的資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。如需組建專案的最新設定清單，請參閱[檢視組建專案詳細資訊](#)。

5. 切換到包含您剛儲存之檔案的目錄，然後再次執行 `start-build` 命令。

```
aws codebuild start-build --cli-input-json file://start-build.json
```

6. 如果成功，則會在輸出中顯示[執行建置](#) 程序所述資料類似的資料。

若要使用此建置的詳細資訊，請記下輸出中的 `id` 值，然後參閱[檢視建置的詳細資訊 \(AWS CLI\)](#)。

執行批次建置 (AWS CLI)

1. 使用下列其中一種方式，執行 `start-build-batch` 命令：

```
aws codebuild start-build-batch --project-name <project-name>
```

如果您想要執行的建置使用建置輸入成品和組建專案現有設定的最新版本，則請使用此命令。

```
aws codebuild start-build-batch --generate-cli-skeleton > <json-file>
```

如果您想要執行的建置使用建置輸入成品的舊版本，或者想要覆寫建置輸出成品、環境變數、`buildspec` 或預設建置逾時期間的設定，則請使用此命令。

2. 如果您使用 `--project-name` 選項執行 `start-build-batch` 命令，請將 *<project-name>* 取代為建置專案的名稱，然後跳至此程序的步驟 6。若要取得組建專案清單，請參閱[檢視組建專案名稱](#)。
3. 如果您使用 `--idempotency-token` 選項執行 `start-build-batch` 命令，`start-build-batch` 則請求中會包含唯一的區分大小寫識別符或字符。此字符在 請求之後的 5 分鐘內有效。如果您使用相同的字符重複 `start-build-batch` 請求，但變更參數，CodeBuild 會傳回參數不相符錯誤。
4. 如果您使用 `--generate-cli-skeleton` 選項執行 `start-build-batch` 命令，JSON 格式的資料會輸出到 *<json-file>* 檔案。此檔案類似於 `start-build` 命令產生的骨架，並新增下列物件。如需常見物件的詳細資訊，請參閱 [執行建置 \(AWS CLI\)](#)。

修改此檔案以新增任何建置覆寫，並儲存結果。

```
"buildBatchConfigOverride": {  
  "combineArtifacts": combineArtifacts,  
  "restrictions": {
```



```
"computeTypesAllowed": [  
    allowedComputeTypes  
],  
"maximumBuildsAllowed": maximumBuildsAllowed  
},  
"serviceRole": "batchServiceRole",  
"timeoutInMins": batchTimeout  
}
```

buildBatchConfigOverride 物件是 [ProjectBuildBatchConfig](#) 結構，其中包含此建置的批次建置組態覆寫。

combineArtifacts

布林值，指定是否應將批次組建的組建成品合併為單一成品位置。

allowedComputeTypes

字串陣列，指定批次建置允許的運算類型。如需這些值，請參閱[建置環境運算類型](#)。

maximumBuildsAllowed

指定允許的建置數目上限。

batchServiceRole

指定批次建置專案的服務角色 ARN。

batchTimeout

指定批次建置必須完成的時間上限 (以分鐘為單位)。

5. 切換到包含您剛儲存之檔案的目錄，然後再次執行 start-build-batch 命令。

```
aws codebuild start-build-batch --cli-input-json file://start-build.json
```

6. 如果成功，則 [BuildBatch](#) 物件的 JSON 表示法會出現在主控台輸出中。如需此資料的範例，請參閱 [StartBuildBatch 回應語法](#)。

自動開始執行建置 (AWS CLI)

如果您的原始碼存放在 GitHub 或 GitHub Enterprise Server 儲存庫中，則只要將程式碼變更推送至儲存庫，您就可以使用 GitHub Webhook AWS CodeBuild 來重建原始碼。

執行 `create-webhook` 命令，如下所示：

```
aws codebuild create-webhook --project-name <project-name>
```

`<project-name>` 是包含要重建之原始碼的建置專案名稱。

針對 GitHub，則會在輸出中顯示與下列內容類似的資訊：

```
{
  "webhook": {
    "url": "<url>"
  }
}
```

`<url>` 是 GitHub Webhook 的 URL。

針對 GitHub Enterprise Server，則會在輸出中顯示與下列內容類似的資訊：

```
{
  "webhook": {
    "secret": "YRV4JYAGFsekjiirp5ytx86oZpyhUdySNSDTLNUxoXX1c7aZ6XYDF37-ZFyY02rs4JSE70mLW3w-gh-ryoVB80SSSC1aAtBtuPkHwYuncCCmdogCVCfniQ7ukYX2_xM--n1Dma5Engig_Bi_N465yi33zyTUNPoQ1xCpL0-BwghcVa91AurwR77-uY7i-_XCJFahwMx1f4ub0gBBsMT2A16apqjqQJoKSb61XVkyZy1Giuy4nliAXFv9WnN76CaCsndb3fVIE78fpygfo41xYxSQ6vpo6LRTKtPzbyeTHbVXGda1PJvnk8lnKmJDo0RTgI1m2oYr17dWziQlrrvoCoNgy1S00_7LKfA-nNXFc_f1SiFy0AqeMB43-d00cdkzybHncE81QTRwEUCFfmX-AJCwmLXV0kg0G67T925jbpz0fRlkh5pwIF193_bB_jOHDinK6i0iPpf2dIDAIZgGMagqZeWb-axDeTAbopoU8J6gFI1yKo5aq9q151zC1PERUsMgJFtJr_a-Z-L_kylr-4hSSxasSJNuJ43_XOBRWqT51xqvH-A69bV07KbVT_Kc6wxkSHyYCEMoa_Pfa7ZQgyfY6B00ogMNj31yFbjthORNL1cDo6-3J-McDLOYrRtSEOV9QnxvsG5zu1N5-z20rkJtg_MOFnwocfUutFXb7vrGTduH1R1dzXLusHuxOVVuDUWm9vhwMr-hUkeGo_1kDKyk4E2QFvZxpjYw0vFv-dwxRFR_mifzxW1wyfmt2iFtLkp_YZj_4WeFAckGefr-ilNaYvsZpzXj78Ae1adVoLf48AmDdN2pWswJjatU9zt942gLiSFFmKakcvJuy5yxXHaxxbhUyC8NHYiESUWPfcfnqrMsr8op3P4AUCHlpiZCYyuiwI_cac-pIUB00Xaur_lu_fyFghgOJc7cfTnA36rv5X5DnFDM8P3HNBELjaF9QZ6AijegPEWTHIKJON3AUDwpkz_hwTXyUoAU8MdZfPTXbBoT6N5Z5THBHsYxR",
    "payloadUrl": "https://codebuild.us-east-2.amazonaws.com/webhooks?t=eyJlbmNyeXB0ZWREYXRhIjoiaUmfqMmJERGRQbGhwLzNTNld3R0VGRjZzOTNwLz1lZGV6N21pIR1E0RUxdzhGeWnnVFFqWTR0WEFwT2dJRnNmRHc3S3RNC0xYMEhNcXFTagk1cE1nSy9zPSIsIm12UGFyYW1ldGVyU3B1YyI6IndS01Qrc2VpQjBCZzhPeVYiLCJtYXR1cm1hbFNldFNlcm1hbCI6MX0%3D&v=1"
  }
}
```

1. 複製輸出中的秘密金鑰和有效負載 URL。您需要它們，才能在 GitHub Enterprise Server 中新增 Webhook。
2. 在 GitHub Enterprise Server 中，選擇儲存 CodeBuild 專案的儲存庫。依序選擇 Settings (設定)、Hooks & services (關聯和服務) 和 Add webhook (新增 Webhook)。
3. 輸入有效負載 URL 和秘密金鑰，並接受其他欄位的預設值，然後選擇 Add webhook (新增 Webhook)。

自動停止執行建置 (AWS CLI)

如果您的原始碼存放在 GitHub 或 GitHub Enterprise Server 儲存庫中，您可以設定 GitHub Webhook，以便在將程式碼變更推送至儲存庫時 AWS CodeBuild 重建原始碼。如需詳細資訊，請參閱 [自動開始執行建置 \(AWS CLI\)](#)。

如果您已啟用此行為，則可以執行 `delete-webhook` 命令予以關閉，如下所示：

```
aws codebuild delete-webhook --project-name <project-name>
```

- 其中 *<project-name>* 是包含要重建之原始碼的建置專案名稱。

如果此命令成功，則輸入中不會顯示任何資訊和錯誤。

Note

這只會從您的 CodeBuild 專案中刪除 Webhook。您也應該刪除 GitHub 或 GitHub Enterprise Server 儲存庫中的 Webhook。

執行組建AWS SDKs)

若要使用 CodePipeline 執行組建 AWS CodeBuild，請略過這些步驟，並[AWS CodeBuild 搭配使用 AWS CodePipeline 來測試程式碼和執行組建](#)改為遵循 中的指示。

如需搭配 AWS SDKs 使用 CodeBuild 的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

在 AWS Lambda 運算上執行組建

AWS Lambda 運算為您的組建提供最佳化的啟動速度。AWS Lambda 支援更快的組建，因為啟動延遲較低。AWS Lambda 也會自動擴展，因此組建不會在佇列中等待執行。不過，有些使用案例 AWS Lambda 不支援，如果它們影響到您，請使用 EC2 運算。如需詳細資訊，請參閱[AWS Lambda 運算的限制](#)。

主題

- [在上執行的精選執行時間環境Docker 映像中，會包含哪些工具和執行時間 AWS Lambda？](#)
- [如果策劃的映像不包含我需要的工具，該怎麼辦？](#)
- [哪些區域支援 CodeBuild 中的 AWS Lambda 運算？](#)
- [AWS Lambda 運算的限制](#)
- [使用 AWS SAM 搭配 CodeBuild Lambda Java 部署 Lambda 函數](#)
- [使用 CodeBuild Lambda Node.js 建立單一頁面 React 應用程式](#)
- [使用 CodeBuild Lambda Python 更新 Lambda 函數組態](#)

在 上執行的精選執行時間環境 Docker 映像中，會包含哪些工具和執行時間 AWS Lambda？

AWS Lambda 支援下列工具：AWS CLI v2、AWS SAM CLI、git、go、Java、Node.js、Python、pip、Ruby 和 .NET。

如果策劃的映像不包含我需要的工具，該怎麼辦？

如果策劃的映像不包含您需要的工具，您可以提供包含必要工具的自訂環境 Docker 映像。

Note

Lambda 不支援使用多架構容器映像的函數。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[使用容器映像建立 Lambda 函數](#)。

請注意，您需要下列 Amazon ECR 許可，才能使用自訂映像進行 Lambda 運算：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/image-repo"
    }
  ]
}
```

```
}
```

另請注意，`wget`必須安裝 `curl`或 才能使用自訂映像。

哪些區域支援 CodeBuild 中的 AWS Lambda 運算？

在 CodeBuild 中，以下支援 AWS Lambda 運算 AWS 區域：美國東部（維吉尼亞北部）、美國東部（俄亥俄）、美國西部（奧勒岡）、亞太區域（孟買）、亞太區域（新加坡）、亞太區域（雪梨）、亞太區域（東京）、歐洲（法蘭克福）、歐洲（愛爾蘭）和南美洲（聖保羅）。如需 CodeBuild 可用位置的詳細資訊 AWS 區域，請參閱[AWS 依區域的服務](#)。

AWS Lambda 運算的限制

有些使用案例 AWS Lambda 不支援，如果它們影響到您，請使用 EC2 運算：

- AWS Lambda 不支援需要根許可的工具。對於 `yum`或 等工具`rpm`，請使用 EC2 運算類型或其他不需要根許可的工具。
- AWS Lambda 不支援 Docker 組建或執行。
- AWS Lambda 不支援寫入 外部的檔案/`tmp`。包含的套件管理員預設會設定為使用 `/tmp`目錄來下載和參考套件。
- AWS Lambda 不支援 環境類型，`LINUX_GPU_CONTAINER``Windows Server Core 2019` 也不支援。
- AWS Lambda 不支援快取、自訂建置逾時、佇列逾時、建置徽章、特殊權限模式、自訂執行時間環境或超過 15 分鐘的執行時間。
- AWS Lambda 不支援使用 Session Manager 進行 VPC 連線、固定範圍的 CodeBuild 來源 IP 地址、EFS、安裝憑證或 SSH 存取。

使用 AWS SAM 搭配 CodeBuild Lambda Java 部署 Lambda 函數

AWS Serverless Application Model (AWS SAM) 是用於建置無伺服器應用程式的開放原始碼架構。如需詳細資訊，請參閱 GitHub 上的儲存[AWS Serverless Application Model 庫](#)。下列 Java 範例使用 Gradle 來建置和測試 AWS Lambda 函數。之後，會使用 AWS SAM CLI 來部署 CloudFormation 範本和部署套件。透過使用 CodeBuild Lambda，系統會自動處理建置、測試和部署步驟，以便快速更新基礎設施，而無需在單一建置中手動介入。

設定您的 AWS SAM 儲存庫

使用 CLI AWS SAM 建立 AWS SAM Hello World專案。

建立您的 AWS SAM 專案

1. 請遵循在本機電腦上[安裝 CLI AWS SAM](#)的 AWS Serverless Application Model 開發人員指南中的指示。
2. 執行 `sam init` 並選取下列專案組態。

```
Which template source would you like to use?: 1 - AWS Quick Start Templates
Choose an AWS Quick Start application template: 1 - Hello World Example
Use the most popular runtime and package type? (Python and zip) [y/N]: N
Which runtime would you like to use?: 8 - java21
What package type would you like to use?: 1 - Zip
Which dependency manager would you like to use?: 1 - gradle
Would you like to enable X-Ray tracing on the function(s) in your application? [y/N]: N
Would you like to enable monitoring using CloudWatch Application Insights? [y/N]: N
Would you like to set Structured Logging in JSON format on your Lambda functions?
[y/N]: N
Project name [sam-app]: <insert project name>
```

3. 將 AWS SAM 專案資料夾上傳至支援的來源儲存庫。如需支援的來源類型清單，請參閱 [ProjectSource](#)。

建立 CodeBuild Lambda Java 專案

建立 AWS CodeBuild Lambda Java 專案並設定建置所需的 IAM 許可。

建立 CodeBuild Lambda Java 專案

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。
3. 在 Project name (專案名稱) 中，輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。
4. 在來源中，選取 AWS SAM 專案所在的來源儲存庫。
5. 在 Environment (環境) 中：
 - 針對運算，選取 Lambda。
 - 針對 Runtime(s)，選取 Java。

- 針對影像，選取 `aws/codebuild/amazonlinux-x86_64-lambda-standard : corretto21`。
 - 對於服務角色，保留選取新的服務角色。記下角色名稱。當您稍後在此範例中更新專案的 IAM 許可時，您需要此項目。
6. 選擇 Create build project (建立建置專案)。
 7. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
 8. 在導覽窗格中，選擇角色，然後選取與您的專案相關聯的服務角色。您可以透過選取建置專案、選擇編輯、環境，然後選擇服務角色，在 CodeBuild 中尋找專案角色。
 9. 選擇 Trust Relationships (信任關係) 標籤，然後選擇 Edit Trust Relationship (編輯信任政策)。
 10. 將下列內嵌政策新增至您的 IAM 角色。這將用於稍後部署您的 AWS SAM 基礎設施。如需詳細資訊，請參閱《IAM 使用者指南》中的 [新增和移除 IAM 身分許可](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "lambda:*",
        "iam:*",
        "apigateway:*",
        "s3:*"
      ],
      "Resource": "arn:aws:iam::*:role/Service*"
    }
  ]
}
```

設定專案 buildspec

為了建置、測試和部署 Lambda 函數，CodeBuild 會從 buildspec 讀取和執行建置命令。

設定您的專案 buildspec

1. 在 CodeBuild 主控台中，選取您的建置專案，然後選擇編輯和建置規格。

2. 在 Buildspec 中，選擇插入建置命令，然後切換到編輯器。
3. 刪除預先填入的建置命令，並貼到下列 buildspec 中。

```
version: 0.2
env:
  variables:
    GRADLE_DIR: "HelloWorldFunction"
phases:
  build:
    commands:
      - echo "Running unit tests..."
      - cd $GRADLE_DIR; gradle test; cd ..
      - echo "Running build..."
      - sam build --template-file template.yaml
      - echo "Running deploy..."
      - sam package --output-template-file packaged.yaml --resolve-s3 --template-file template.yaml
      - yes | sam deploy
```

4. 選擇 Update buildspec (更新 buildspec)。

部署您的 AWS SAM Lambda 基礎設施

使用 CodeBuild Lambda 自動部署 Lambda 基礎設施

部署 Lambda 基礎設施

1. 選擇 Start build (開始組建)。這會自動建置、測試和部署您的 AWS SAM 應用程式以 AWS Lambda 使用 CloudFormation。
2. 建置完成後，導覽至 AWS Lambda 主控台，並在 AWS SAM 專案名稱下搜尋新的 Lambda 函數。
3. 在函數概觀下選取 API Gateway，然後按一下 API 端點 URL，以測試 Lambda 函數。您應該會看到開啟的頁面，其中包含訊息 "message": "hello world"。

清除您的基礎設施

為了避免您在本教學課程中使用的資源進一步收費，請刪除 AWS SAM 範本和 CodeBuild 建立的資源。

清除您的基礎設施

1. 導覽至 CloudFormation 主控台，然後選取 `aws-sam-cli-managed-default`。
2. 在 資源中，清空部署儲存貯體 `SamCliSourceBucket`。
3. 刪除 `aws-sam-cli-managed-default` 堆疊。
4. 刪除與您的 AWS SAM 專案相關聯的 CloudFormation 堆疊。此堆疊應與 AWS SAM 專案具有相同的名稱。
5. 導覽至 CloudWatch 主控台，並刪除與 CodeBuild 專案相關聯的 CloudWatch 日誌群組。
6. 導覽至 CodeBuild 主控台，然後選擇刪除組建專案來刪除您的 CodeBuild 專案。

使用 CodeBuild Lambda Node.js 建立單一頁面 React 應用程式

[Create React 應用程式](#) 是建立單頁 React 應用程式的一種方式。下列 Node.js 範例使用 Node.js 從 Create React 應用程式建置來源成品，並傳回建置成品。

設定您的來源儲存庫和成品儲存貯體

使用 yarn 和 Create React 應用程式為您的專案建立來源儲存庫。

設定來源儲存庫和成品儲存貯體

1. 在本機電腦上，執行 `yarn create react-app <app-name>` 以建立簡單的 React 應用程式。
2. 將 React 應用程式專案資料夾上傳至支援的來源儲存庫。如需支援的來源類型清單，請參閱 [ProjectSource](#)。

建立 CodeBuild Lambda Node.js 專案

建立 AWS CodeBuild Lambda Node.js 專案。

建立 CodeBuild Lambda Node.js 專案

1. 在 `https://console.aws.amazon.com/codesuite/codebuild/home` 開啟 AWS CodeBuild 主控台。
2. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開組建，選擇組建專案，然後選擇建立組建專案。
3. 在 Project name (專案名稱) 中，輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。

4. 在來源中，選取 AWS SAM 專案所在的來源儲存庫。
5. 在 Environment (環境) 中：
 - 針對運算，選取 Lambda。
 - 針對 Runtime(s)，選取 Node.js。
 - 針對影像，選取 aws/codebuild/amazonlinux-x86_64-lambda-standard : nodejs20。
6. 在 Artifacts (成品) 中：
 - 針對類型，選取 Amazon S3。
 - 針對儲存貯體名稱，選取您先前建立的專案成品儲存貯體。
 - 針對成品封裝，選取 Zip。
7. 選擇 Create build project (建立建置專案)。

設定專案 buildspec

為了建置您的 React 應用程式，CodeBuild 會從 buildspec 檔案讀取並執行建置命令。

設定您的專案 buildspec

1. 在 CodeBuild 主控台中，選取您的建置專案，然後選擇編輯和建置規格。
2. 在 Buildspec 中，選擇插入建置命令，然後切換到編輯器。
3. 刪除預先填入的建置命令，並貼到下列 buildspec 中。

```
version: 0.2
phases:
  build:
    commands:
      - yarn
      - yarn add --dev jest-junit @babel/plugin-proposal-private-property-in-object
      - yarn run build
      - yarn run test -- --coverage --watchAll=false --testResultsProcessor="jest-junit" --detectOpenHandles
artifacts:
  name: "build-output"
  files:
    - "**/*"
reports:
  test-report:
    files:
```

```
- 'junit.xml'
file-format: 'JUNITXML'
coverage-report:
  files:
    - 'coverage/clover.xml'
file-format: 'CLOVERXML'
```

4. 選擇 Update buildspec (更新 buildspec)。

建置並執行您的 React 應用程式

在 CodeBuild Lambda 上建置 React 應用程式、下載建置成品，並在本機執行 React 應用程式。

建置和執行您的 React 應用程式

1. 選擇 Start build (開始組建)。
2. 建置完成後，導覽至 Amazon S3 專案成品儲存貯體並下載 React 應用程式成品。
3. 解壓縮專案資料夾中的 React 組建成品 和 `run npm install -g serve && serve -s build`。
4. `serve` 命令會在本機連接埠上為靜態網站提供服務，並將輸出列印到您的終端機。您可以在終端機輸出 Local: 中造訪 下的 localhost URL，以檢視您的 React 應用程式。

若要進一步了解如何處理以 React 為基礎的伺服器的部署，請參閱[建立 React 應用程式部署](#)。

清除您的基礎設施

若要避免您在本教學課程中使用的資源進一步收費，請刪除為 CodeBuild 專案建立的資源。

清除您的基礎設施

1. 刪除專案成品 Amazon S3 儲存貯體
2. 導覽至 CloudWatch 主控台，並刪除與 CodeBuild 專案相關聯的 CloudWatch 日誌群組。
3. 導覽至 CodeBuild 主控台，然後選擇刪除組建專案來刪除您的 CodeBuild 專案。

使用 CodeBuild Lambda Python 更新 Lambda 函數組態

下列 Python 範例使用 [Boto3](#) 和 CodeBuild Lambda Python 來更新 Lambda 函數的組態。您可以延伸此範例，以程式設計方式管理其他 AWS 資源。如需詳細資訊，請參閱 [Boto3 文件](#)。

先決條件

在您的帳戶中建立或尋找 Lambda 函數。

此範例假設您已在帳戶中建立 Lambda 函數，並將使用 CodeBuild 更新 Lambda 函數的環境變數。如需透過 CodeBuild 設定 Lambda 函數的詳細資訊，請參閱[使用 AWS SAM 搭配 CodeBuild Lambda Java 部署 Lambda 函數範例](#)或造訪 [AWS Lambda](#)。

設定您的來源儲存庫

建立來源儲存庫以存放 Boto3 python 指令碼。

設定來源儲存庫

1. 將下列 python 指令碼複製到名為 `update_lambda_environment_variables.py` 的新檔案。

```
import boto3
from os import environ

def update_lambda_env_variable(lambda_client):
    lambda_function_name = environ['LAMBDA_FUNC_NAME']
    lambda_env_variable = environ['LAMBDA_ENV_VARIABLE']
    lambda_env_variable_value = environ['LAMBDA_ENV_VARIABLE_VALUE']
    print("Updating lambda function " + lambda_function_name + " environment
variable "
        + lambda_env_variable + " to " + lambda_env_variable_value)
    lambda_client.update_function_configuration(
        FunctionName=lambda_function_name,
        Environment={
            'Variables': {
                lambda_env_variable: lambda_env_variable_value
            }
        },
    )

if __name__ == "__main__":
    region = environ['AWS_REGION']
    client = boto3.client('lambda', region)
    update_lambda_env_variable(client)
```

2. 將 python 檔案上傳至支援的來源儲存庫。如需支援的來源類型清單，請參閱 [ProjectSource](#)。

建立 CodeBuild Lambda Python 專案

建立 CodeBuild Lambda Python 專案。

建立 CodeBuild Lambda Java 專案

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 如果顯示 CodeBuild 資訊頁面，請選擇建立組建專案。否則，請在導覽窗格中展開建置，選擇建置專案，然後選擇建立建置專案。
3. 在 Project name (專案名稱) 中，輸入此建置專案的名稱。組建專案名稱在每個 AWS 帳戶中必須是唯一的。您也可以包含建置專案的選用描述，以協助其他使用者了解此專案的用途。
4. 在來源中，選取 AWS SAM 專案所在的來源儲存庫。
5. 在 Environment (環境) 中：
 - 針對運算，選取 Lambda。
 - 針對 Runtime(s)，選取 Python。
 - 針對影像，選取 aws/codebuild/amazonlinux-x86_64-lambda-standard : python3.12。
 - 對於服務角色，保留選取新的服務角色。記下角色名稱。當您稍後在此範例中更新專案的 IAM 許可時，您需要此項目。
6. 選擇 Create build project (建立建置專案)。
7. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
8. 在導覽窗格中，選擇角色，然後選取與您的專案相關聯的服務角色。您可以透過選取建置專案、選擇編輯、環境，然後選擇服務角色，在 CodeBuild 中尋找專案角色。
9. 選擇 Trust Relationships (信任關係) 標籤，然後選擇 Edit Trust Relationship (編輯信任政策)。
10. 將下列內嵌政策新增至您的 IAM 角色。這將用於稍後部署您的 AWS SAM 基礎設施。如需詳細資訊，請參閱《IAM 使用者指南》中的 [新增和移除 IAM 身分許可](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateLambdaPermissions",
      "Effect": "Allow",
```

```
        "Action": [  
            "lambda:UpdateFunctionConfiguration"  
        ],  
        "Resource": [  
            "*"   
        ]  
    }  
]  
}
```

設定專案 buildspec

若要更新 Lambda 函數，指令碼會從 buildspec 讀取環境變數，以尋找 Lambda 函數的名稱、環境變數名稱和環境變數值。

設定您的專案 buildspec

1. 在 CodeBuild 主控台中，選取您的建置專案，然後選擇編輯和建置規格。
2. 在 Buildspec 中，選擇插入建置命令，然後切換到編輯器。
3. 刪除預先填入的建置命令，並貼到下列 buildspec 中。

```
version: 0.2  
env:  
  variables:  
    LAMBDA_FUNC_NAME: "<lambda-function-name>"  
    LAMBDA_ENV_VARIABLE: "FEATURE_ENABLED"  
    LAMBDA_ENV_VARIABLE_VALUE: "true"  
phases:  
  install:  
    commands:  
      - pip3 install boto3  
  build:  
    commands:  
      - python3 update_lambda_environment_variables.py
```

4. 選擇 Update buildspec (更新 buildspec)。

更新您的 Lambda 組態

使用 CodeBuild Lambda Python 自動更新 Lambda 函數的組態。

更新 Lambda 函數的組態

1. 選擇 Start build (開始組建)。
2. 建置完成後，導覽至 Lambda 函數。
3. 選取組態，然後選取環境變數。您應該會看到具有索引鍵 `FEATURE_ENABLED` 和值 `true` 的新環境變數。

清除您的基礎設施

為了避免您在本教學課程中使用的資源進一步收費，請刪除為 CodeBuild 專案建立的資源。

清除您的基礎設施

1. 導覽至 CloudWatch 主控台，並刪除與 CodeBuild 專案相關聯的 CloudWatch 日誌群組。
2. 導覽至 CodeBuild 主控台，然後選擇刪除組建專案來刪除您的 CodeBuild 專案。
3. 如果您為此範例建立 Lambda 函數，請選擇動作和刪除函數來清除 Lambda 函數。

延伸模組

如果您想要延伸此範例以使用 AWS CodeBuild Lambda Python 管理其他 AWS 資源：

- 更新 Python 指令碼以使用 Boto3 修改新資源。
- 更新與 CodeBuild 專案相關聯的 IAM 角色，以取得新資源的許可。
- 將與新資源相關聯的任何新環境變數新增至您的 `buildspec`。

在預留容量機群上執行組建

CodeBuild 提供下列運算機群：

- 隨需機群
- 預留容量機群

透過隨需機群，CodeBuild 為您的建置提供運算。這些機器會在建置完成時銷毀。隨需機群受到完全管理，並包含自動擴展功能來處理需求激增。

 Note

隨需機群不支援 macOS。

CodeBuild 也提供預留容量機群，其中包含由 CodeBuild 維護的 Amazon EC2 提供支援的執行個體。使用預留容量機群，您可以為建置環境設定一組專用執行個體。這些機器會保持閒置，準備好立即處理組建或測試，並減少組建持續時間。使用預留容量機群時，您的機器一律在執行中，只要佈建，就會持續產生成本。

 Important

無論您執行執行個體的時間有多長，預留容量機群都會產生每個執行個體的初始費用，之後可能會產生額外的相關費用。如需詳細資訊，請參閱<https://aws.amazon.com/codebuild/pricing/>。

主題

- [建立預留容量機群](#)
- [最佳實務](#)
- [我可以跨多個 CodeBuild 專案共用預留容量機群嗎？](#)
- [屬性型運算如何運作？](#)
- [我可以手動指定機群的 Amazon EC2 執行個體嗎？](#)
- [哪些區域支援預留容量機群？](#)
- [如何設定預留容量 macOS 機群？](#)
- [如何為預留容量機群設定自訂 Amazon Machine Image \(AMI\)？](#)
- [預留容量機群的限制](#)
- [預留容量機群屬性](#)
- [使用的預留容量範例 AWS CodeBuild](#)

建立預留容量機群

使用下列指示來建立預留容量機群。

建立預留容量機群

1. 登入 AWS 管理主控台 並在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇運算機群，然後選擇建立機群。
3. 在運算機群名稱文字欄位中，輸入機群的名稱。
4. 從作業系統下拉式功能表中，選擇作業系統。
5. 從架構下拉式功能表中，選擇架構。
6. (選用) 選取使用執行模式的執行個體 - 選擇性地直接在 Amazon EC2 執行個體上執行，而非 Docker 容器。然後選擇主要版本和次要版本。
7. (選用) 在其他組態中，執行下列動作：
 - 選取設定 VPC - 選用，將機群連線至 VPC，以在使用期間存取私有資源。
 - 從 VPC 下拉式選單中，選取 CodeBuild 機群將存取的 VPC。
 - 從子網路下拉式功能表中，選取 CodeBuild 應該用來設定 VPC 組態的子網路。
 - 從安全群組下拉式功能表中，選取 CodeBuild 應用於 VPC 的安全群組。
 - 在機群服務角色欄位中，選擇現有的服務角色。

Note

請確定您的機群角色具有必要的許可。如需詳細資訊，請參閱[允許使用者新增機群服務角色的許可政策](#)。

- 如果您選擇 Amazon Linux 作業系統，請選取定義代理組態 - 選用，以套用預留容量執行個體的網路存取控制。
- 針對預設行為，選擇預設允許或拒絕所有目的地的傳出流量。
- 針對 Proxy 規則，選擇新增 Proxy 規則以指定目的地網域或 IPs，以允許或拒絕網路存取控制。
- 選取設定自訂 AMI - 選用以使用自訂 Amazon Machine Image (AMI)。
 - 從 AMI 下拉式功能表中，選取機群的 Amazon Machine Image (AMI)。
 - 在機群服務角色欄位中，選擇現有的服務角色。

Note

請確定您的機群角色具有必要的許可。如需詳細資訊，請參閱[允許使用者新增機群服務角色的許可政策](#)。

8. 在容量組態中，從運算選擇模式中，選擇下列其中一項：
 - 如果您選擇引導式選擇，請執行下列動作：
 - 針對運算，選擇此機群中包含的執行個體類型。
 - 在容量文字欄位中，輸入機群中執行個體的最小數量。
 - (選用) 在其他組態中，請執行下列動作：
 - 選取設定擴展 - 選用，以根據此組態自動擴展機群。從擴展模式 - 選用下拉式功能表中，選擇需求超過機群容量時的行為。
 - 如果您選擇自訂執行個體，請執行下列動作：
 - 從運算執行個體類型下拉式選單中，選取此機群中包含的執行個體類型。
 - 在其他 EBS 磁碟區大小 - 選用的文字欄位中，輸入所提供的 64GB 磁碟空間以外的磁碟區。
 - 在容量文字欄位中，輸入機群中執行個體的最小數量。
 - (選用) 在其他組態中，執行下列動作：
 - 選取設定擴展 - 選用，以根據此組態自動擴展機群。從擴展模式 - 選用下拉式功能表中，選擇需求超過機群容量時的行為。
9. 選擇建立運算機群。
10. 建立運算機群後，請建立新的 CodeBuild 專案或編輯現有的專案。在環境中，選擇佈建模型下的預留容量，然後在機群名稱下選擇指定的機群。

最佳實務

使用預留容量機群時，建議您遵循這些最佳實務。

- 我們建議您使用來源快取模式，透過快取來源來協助改善建置效能。
- 我們建議您使用 Docker layer 快取，透過快取現有的 Docker layer 來協助改善建置效能。

我可以跨多個 CodeBuild 專案共用預留容量機群嗎？

可以，您可以將機群容量的使用率最大化，方法是在多個專案中使用。

Important

使用預留容量功能時，同一帳戶中的其他專案可以存取機群執行個體上快取的資料，包括來源檔案、Docker layer 和 buildspec 中指定的快取目錄。這是根據設計，並允許相同帳戶中的專案共用機群執行個體。

屬性型運算如何運作？

如果您選擇 `ATTRIBUTE_BASED_COMPUTE` 做為機群的 `computeType`，您可以在名為 `computeConfiguration` 的新欄位中指定屬性 `computeConfiguration`。這些屬性包括 `vCPUs`、記憶體、磁碟空間和 `machineType`。`machineType` 這是 `GENERAL` 或 `NVME`。指定一或多個可用屬性之後，CodeBuild 會從可用的支援執行個體類型中選擇運算類型，做為最終的 `computeConfiguration`。

Note

CodeBuild 將選擇符合所有輸入需求最便宜的執行個體。所選執行個體的記憶體、`vCPUs` 和磁碟空間都會大於或等於輸入需求。您可以在建立或更新的機群 `computeConfiguration` 中檢查已解析的。

如果您輸入無法在 CodeBuild 中滿足 `computeConfiguration` 的，您將會收到驗證例外狀況。另請注意，如果 `computeConfiguration` 不適用於隨需，則會覆寫隨需機群溢位行為以佇列行為。

我可以手動指定機群的 Amazon EC2 執行個體嗎？

可以，您可以透過選取自訂執行個體或設定 API 參數，在主控台中直接輸入所需的 Amazon EC2 執行個體 `InstanceType`。此欄位用於下列 APIs：`CreateFleet`、`UpdateFleet`、`CreateProject`、`UpdateProject` 和 `StartBuild`。如需詳細資訊，請參閱 [Compute instance type](#)。

哪些區域支援預留容量機群？

預留容量 Amazon Linux 和 Windows 機群支援以下項目 AWS 區域：美國東部（維吉尼亞北部）、美國東部（俄亥俄）、美國西部（奧勒岡）、亞太區域（孟買）、亞太區域（新加坡）、亞太區域（雪梨）、亞太區域（東京）、歐洲（法蘭克福）、歐洲（愛爾蘭）和南美洲（聖保羅）。如需 CodeBuild 可用位置的詳細資訊 AWS 區域，請參閱 [AWS 依區域提供服務](#)。

下列支援預留容量 macOS Medium 機群 AWS 區域：美國東部（維吉尼亞北部）、美國東部（俄亥俄）、美國西部（奧勒岡）、亞太區域（雪梨）和歐洲（法蘭克福）。預留容量 macOS 支援大型機群 AWS 區域：美國東部（維吉尼亞北部）、美國東部（俄亥俄）、美國西部（奧勒岡）和亞太區域（雪梨）。

如何設定預留容量 macOS 機群？

設定預留容量 macOS 機群

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/codesuite/codebuild/home> 的 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇運算機群，然後選擇建立機群。
3. 在運算機群名稱文字欄位中，輸入機群的名稱。
4. 從作業系統下拉式功能表中，選擇 macOS。
5. 在運算欄位中，選擇下列其中一種運算機器類型：Apple M2、24 GB 記憶體、8 個 vCPUs 或 Apple M2、32 GB 記憶體、12 個 vCPUs。
6. 在容量文字欄位中，輸入機群中執行個體的最小數量。
7. （選用）若要為您的機群使用自訂映像，請參閱 [如何為預留容量機群設定自訂 Amazon Machine Image \(AMI\)？](#) 以確保您的 Amazon Machine Image (AMI) 具有必要的先決條件。
8. （選用）若要使用機群設定 VPC，請在其他組態中執行下列動作：
 - 從 VPC - 選用下拉式功能表中，選取 CodeBuild 機群將存取的 VPC。
 - 從子網路下拉式功能表中，選取 CodeBuild 應該用來設定 VPC 組態的子網路。
 - 從安全群組下拉式功能表中，選取 CodeBuild 應用於 VPC 的安全群組。
 - 在機群服務角色欄位中，選擇現有的服務角色。

Note

請確定您的機群角色具有必要的許可。如需詳細資訊，請參閱 [允許使用者新增機群服務角色的許可政策](#)。

9. 選擇建立運算機群，並等待機群執行個體啟動。啟動後，容量將為 n/n ，其中 n 是提供的容量。
10. 運算機群啟動後，建立新的 CodeBuild 專案或編輯現有的專案。在環境中，選擇佈建模型下的預留容量，然後在機群名稱下選擇指定的機群。

如何為預留容量機群設定自訂 Amazon Machine Image (AMI) ?

為預留容量機群設定自訂 Amazon Machine Image (AMI)

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/codesuite/codebuild/home> 的 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇運算機群，然後選擇建立機群。
3. 在運算機群名稱文字欄位中，輸入機群的名稱。
4. 選擇機群的自訂映像，並確保您的 Amazon Machine Image (AMI) 具有下列先決條件：
 - 如果您的環境類型是 MAC_ARM，請確定您的 AMI 架構是 64 位元 Mac-Arm。
 - 如果您的環境類型是 LINUX_EC2，請確定您的 AMI 架構是 64 位元 x86。
 - 如果您的環境類型是 ARM_EC2，請確定您的 AMI 架構是 64 位元 Arm。
 - 如果您的環境類型是 WINDOWS_EC2，請確定您的 AMI 架構是 64 位元 x86。
 - AMI 允許 CodeBuild 服務組織 ARN。如需組織 ARNs 的清單，請參閱 [Amazon Machine Images \(AMI\)](#)。
 - 如果 AMI 使用 AWS KMS 金鑰加密，則 AWS KMS 金鑰也必須允許 CodeBuild 服務組織 ID。如需組織 IDs 的清單，請參閱 [Amazon Machine Images \(AMI\)](#)。如需 AWS KMS 金鑰的詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [允許組織和 OUs 使用 KMS 金鑰](#)。若要授予 CodeBuild 組織使用 KMS 金鑰的許可，請將下列陳述式新增至金鑰政策：

```
{
  "Sid": "Allow access for organization root",
  "Effect": "Allow",
  "Principal": "*",
  "Action": [
    "kms:Describe*",
    "kms:List*",
    "kms:Get*",
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
```

```
        "aws:PrincipalOrgID": "o-123example"
      }
    }
  }
```

- 在機群服務角色欄位中，授予下列 Amazon EC2 許可：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeImages",
        "ec2:DescribeSnapshots"
      ],
      "Resource": "*"
    }
  ]
}
```

預留容量機群的限制

有些使用案例不支援預留容量機群，如果它們影響到您，請改用隨需機群：

- 預留容量機群不支援建置使用率指標。
- 預留容量 macOS 機群不支援偵錯工作階段。

如需限制和配額的詳細資訊，請參閱 [運算機群](#)。

預留容量機群屬性

預留容量機群包含下列屬性。如需預留容量機群的詳細資訊，請參閱 [在預留容量機群上執行組建](#)。

作業系統

作業系統。下列作業系統可供使用：

- Amazon Linux

- macOS
- Windows Server 2019
- Windows Server 2022

Architecture

處理器架構。下列架構可供使用：

- x86_64
- Arm64

環境類型

選取 Amazon Linux 時可用的環境類型。下列環境類型可供使用：

- Linux EC2
- Linux GPU

運算執行個體類型

機群執行個體的運算組態。

引導式選取

選取 vCPU、記憶體和磁碟空間設定來指定不同的運算類型。如需依區域劃分的運算類型可用性資訊，請參閱 [關於預留容量環境類型](#)。

自訂執行個體

手動指定所需的執行個體類型。

Capacity

配置給機群的機器初始數量，定義可平行執行的組建數量。

溢位行為

定義建置數量超過機群容量時的行為。

隨需

溢位建置會隨需在 CodeBuild 上執行。

Note

如果您在建立 VPC 連線機群時選擇將溢位行為設定為隨需，請務必將所需的 VPC 許可新增至專案服務角色。如需詳細資訊，請參閱[範例政策陳述式，以允許 CodeBuild AWS 存取建立 VPC 網路介面所需的服務](#)。

Important

如果您選擇將溢位行為設定為隨需，請注意，溢位建置將分別計費，類似於隨需 Amazon EC2。如需詳細資訊，請參閱<https://aws.amazon.com/codebuild/pricing/>。

佇列

建置執行會放置在佇列中，直到機器可用為止。這可限制額外的成本，因為沒有配置額外的機器。

Amazon Machine Image (AMI)

機群的 Amazon Machine Image (AMI) 屬性。CodeBuild 支援下列屬性：

AWS 區域	組織 ARN	組織 ID
us-east-1	arn:aws:organizations::851725618577:organization/o-c6wcu152r1	o-c6wcu152r1
us-east-2	arn:aws:organizations::992382780434:organization/o-seufr2suvq	o-seufr2suvq
us-west-2	arn:aws:organizations::381491982620:organization/o-0412o99a4r	o-0412o99a4r

AWS 區域	組織 ARN	組織 ID
ap-northeast-1	arn:aws:organizations::891376993293:organization/o-b6k3sjqavm	o-b6k3sjqavm
ap-south-1	arn:aws:organizations::891376924779:organization/o-krtah1lkeg	o-krtah1lkeg
ap-southeast-1	arn:aws:organizations::654654522137:organization/o-mcn8uvc3tp	o-mcn8uvc3tp
ap-southeast-2	arn:aws:organizations::767398067170:organization/o-6crt0f6bu4	o-6crt0f6bu4
eu-central-1	arn:aws:organizations::590183817084:organization/o-lb2lne3te6	o-lb2lne3te6
eu-west-1	arn:aws:organizations::891376938588:organization/o-ullrrg5qf0	o-ullrrg5qf0
sa-east-1	arn:aws:organizations::533267309133:organization/o-db63c45ozw	o-db63c45ozw

其他組態

VPC - 選用

CodeBuild 機群將存取的 VPC。如需詳細資訊，請參閱[AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用](#)。

Note

如果在呼叫 StartBuild API 時指定機群覆寫，CodeBuild 會忽略專案 VPC 組態。

子網路

CodeBuild 用來設定 VPC 組態的 VPC 子網路。請注意，預留容量機群在單一可用區域中僅支援一個子網路。此外，請確定子網路包含 NAT 閘道。

安全群組

CodeBuild 搭配 VPC 使用的 VPC 安全群組。確保您的安全群組允許傳出連線。

機群服務角色

從您帳戶中的現有服務角色定義機群的服務角色。

定義代理組態 - 選用

將網路存取控制套用至預留容量執行個體的代理組態。如需詳細資訊，請參閱[AWS CodeBuild 搭配受管代理伺服器使用](#)。

Note

Proxy 組態不支援 VPC、Windows 或 macOS。

預設行為

定義傳出流量的行為。

Allow

根據預設，允許傳出流量到所有目的地。

拒絕

根據預設，拒絕所有目的地的傳出流量。

Proxy 規則

指定要允許或拒絕網路存取控制的目的地網域或 IPs。

使用的預留容量範例 AWS CodeBuild

這些範例可用於實驗 CodeBuild 中的預留容量機群。

主題

- [使用預留容量範例進行快取](#)

使用預留容量範例進行快取

快取可以存放組建環境的可重複使用部分，並在多個組建間使用這些部分。此範例示範如何使用預留容量在建置專案中啟用快取。如需詳細資訊，請參閱[快取建置以改善效能](#)。

您可以先在專案設定中指定一或多個快取模式：

Cache:

Type: LOCAL

Modes:

- LOCAL_CUSTOM_CACHE
- LOCAL_DOCKER_LAYER_CACHE
- LOCAL_SOURCE_CACHE

Note

請務必啟用特權模式，才能使用 Docker layer 快取。

您的專案 buildspec 設定應如下所示：

```
version: 0.2
phases:
  build:
    commands:
```

```

- echo testing local source cache
- touch /codebuild/cache/workspace/foobar.txt
- git checkout -b cached_branch
- echo testing local docker layer cache
- docker run alpine:3.14 2>&1 | grep 'Pulling from' || exit 1
- echo testing local custom cache
- touch foo
- mkdir bar && ln -s foo bar/foo2
- mkdir bar/bar && touch bar/bar/foo3 && touch bar/bar/foo4
- "[ -f foo ] || exit 1"
- "[ -L bar/foo2 ] || exit 1"
- "[ -f bar/bar/foo3 ] || exit 1"
- "[ -f bar/bar/foo4 ] || exit 1"
cache:
  paths:
    - './foo'
    - './bar/**/*'
    - './bar/bar/foo3'

```

您可以從使用新專案執行組建開始，以植入快取。完成後，您應該使用覆寫的 `buildspec` 啟動另一個組建，如下所示：

```

version: 0.2
  phases:
    build:
      commands:
        - echo testing local source cache
        - git branch | if grep 'cached_branch'; then (exit 0); else (exit 1); fi
        - ls /codebuild/cache/workspace | if grep 'foobar.txt'; then (exit 0); else
(exit 1); fi
        - echo testing local docker layer cache
        - docker run alpine:3.14 2>&1 | if grep 'Pulling from'; then (exit 1); else
(exit 0); fi
        - echo testing local custom cache
        - "[ -f foo ] || exit 1"
        - "[ -L bar/foo2 ] || exit 1"
        - "[ -f bar/bar/foo3 ] || exit 1"
        - "[ -f bar/bar/foo4 ] || exit 1"
      cache:
        paths:
          - './foo'
          - './bar/**/*'
          - './bar/bar/foo3'

```

批次執行組建

您可以使用 AWS CodeBuild 執行具有批次建置之專案的並行和協調建置。

主題

- [安全角色](#)
- [批次建置類型](#)
- [批次報告模式](#)
- [其他資訊](#)

安全角色

批次建置在批次組態中導入新的安全性角色。此新角色是必要的，因為 CodeBuild 必須能夠代表您呼叫 StartBuild、StopBuild 和 RetryBuild 動作，才能在批次處理中執行組建。客戶應該使用新的角色，而不是與他們在組建中使用的角色相同，原因有兩個：

- 賦予建置角色 StartBuild、StopBuild 和 RetryBuild 權限，此將允許單一組建透過 buildspec 啟動更多組建。
- CodeBuild 批次建置提供限制，限制可用於批次中建置的建置和運算類型數目。如果組建角色具有這些權限，組建本身可能會略過這些限制。

批次建置類型

CodeBuild 支援下列批次建置類型：

批次建置類型

- [組建圖表](#)
- [組建清單](#)
- [建置矩陣](#)
- [建置分發](#)

組建圖表

建置圖表會定義一組任務，這些任務對批次中的其他任務具有相依性。

下列範例會定義建立相依性鏈的建置圖表。

```
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1
    - identifier: build3
      env:
        variables:
          BUILD_ID: build3
      depend-on:
        - build2
    - identifier: build4
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build5
      env:
        fleet: fleet_name
```

在此範例中：

- build1 會先執行，因為它沒有相依性。
- build2 具有 的相依性build1，因此 會在 build1完成後build2執行。
- build3 具有 的相依性build2，因此 會在 build2完成後build3執行。

如需建置圖表 buildspec 語法的詳細資訊，請參閱 [batch/build-graph](#)。

組建清單

建置清單會定義一些平行執行的任務。

下列範例定義建置清單。build1 和 build2建置會平行執行。

```
batch:
  fast-fail: false
  build-list:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      ignore-failure: true
    - identifier: build3
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build4
      env:
        fleet: fleet_name
    - identifier: build5
      env:
        compute-type: GENERAL_LINUX_XLAGRE
```

如需建置清單 buildspec 語法的詳細資訊，請參閱 [batch/build-list](#)。

建置矩陣

建置矩陣會定義具有平行執行之不同組態的任務。CodeBuild 會為每個可能的組態組合建立個別的組建。

下列範例顯示具有兩個 buildspec 檔案和三個環境變數值的建置矩陣。

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
```

```
variables:
  MY_VAR:
    - VALUE1
    - VALUE2
    - VALUE3
```

在此範例中，CodeBuild 會建立六個組建：

- matrix1.yml 取代為 \$MY_VAR=VALUE1
- matrix1.yml 取代為 \$MY_VAR=VALUE2
- matrix1.yml 取代為 \$MY_VAR=VALUE3
- matrix2.yml 取代為 \$MY_VAR=VALUE1
- matrix2.yml 取代為 \$MY_VAR=VALUE2
- matrix2.yml 取代為 \$MY_VAR=VALUE3

每個組建都會有下列設定：

- ignore-failure 設定為 false
- env/type 設定為 LINUX_CONTAINER
- env/image 設定為 aws/codebuild/amazonlinux-x86_64-standard:4.0
- env/privileged-mode 設定為 true

這些建置會平行執行。

如需建置矩陣 buildspec 語法的詳細資訊，請參閱 [batch/build-matrix](#)。

建置分發

建置散發會定義任務，該任務將在批次中分割為多個建置。這可用於平行執行測試。CodeBuild 會根據 parallelism 欄位中設定的值，為每個測試案例碎片建立個別的組建。

下列範例會定義建置散發，以建立五個平行執行的建置。

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
```



```
parallelism: 5
ignore-failure: false

phases:
  install:
    commands:
      - npm install
  build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run \
          --test-command 'npx jest --runInBand --coverage' \
          --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
          --sharding-strategy 'equal-distribution'
```

在此範例中，假設需要執行 100 個測試，CodeBuild 會建立五個組建，每個組建會平行執行 20 個測試。

如需建置圖表 buildspec 語法的詳細資訊，請參閱 [batch/build-fanout](#)。

批次報告模式

如果專案的來源提供者是 Bitbucket、GitHub 或 GitHub Enterprise，且您的專案已設定為向來源提供者報告建置狀態，您可以選取要將批次建置狀態傳送至來源提供者的方式。您可以選擇讓狀態作為批次的單一彙總狀態報告傳送，或讓批次中每個組建的狀態個別報告。

如需詳細資訊，請參閱下列主題：

- [批次組態（建立）](#)
- [批次組態（更新）](#)

其他資訊

如需詳細資訊，請參閱下列主題：

- [批次建置 buildspec 參考](#)
- [批次組態](#)
- [執行批次建置 \(AWS CLI\)](#)

- [在中停止批次建置 AWS CodeBuild](#)

在批次建置中執行平行測試

您可以使用 AWS CodeBuild 在批次建置中執行平行測試。平行測試執行是一種測試方法，其中多個測試案例會同時跨不同的環境、機器或瀏覽器執行，而不是依序執行。這種方法可以大幅縮短整體測試執行時間，並改善測試效率。在 CodeBuild 中，您可以將測試分割到多個環境，並同時執行。

平行測試執行的主要優點包括：

1. 縮短執行時間 - 循序需要數小時的測試可以在幾分鐘內完成。
2. 更好的資源使用率 - 有效率地使用可用的運算資源。
3. 早期意見回饋 - 更快完成測試意味著對開發人員提供更快的意見回饋。
4. 經濟實惠 - 長期下來可節省時間和運算成本。

實作平行測試執行時，通常會考慮兩種主要方法：獨立環境和多執行緒。雖然這兩種方法都旨在實現並行測試執行，但它們的實作和有效性有很大的不同。不同的環境會建立隔離的執行個體，其中每個測試套件都會獨立執行，而多執行緒會使用不同的執行緒在相同的程序空間內同時執行多個測試。

相較於多執行緒，不同環境的主要優點包括：

1. 隔離 - 每個測試都會在完全隔離的環境中執行，以防止測試之間的干擾。
2. 資源衝突 - 不競爭多執行緒中經常發生的共用資源。
3. 穩定性 - 較不容易發生競爭條件和同步問題。
4. 更輕鬆地偵錯 - 當測試失敗時，更容易識別原因，因為每個環境都是獨立的。
5. 狀態管理 - 輕鬆管理困擾多執行緒測試的共用狀態問題。
6. 更好的可擴展性 - 可以輕鬆新增更多環境，而無需複雜性。

主題

- [中的支援 AWS CodeBuild](#)
- [在批次建置中啟用平行測試執行](#)
- [使用 codebuild-tests-run CLI 命令](#)
- [使用 codebuild-glob-search CLI 命令](#)
- [關於測試分割](#)

- [自動合併個別建置報告](#)
- [各種測試架構範例的平行測試執行](#)

中的支援 AWS CodeBuild

AWS CodeBuild 透過其批次建置功能提供對平行測試執行的強大支援，專為利用個別環境執行而設計。此實作完美符合隔離測試環境的優點。

具有測試分佈的批次建置

CodeBuild 的批次建置功能可建立同時執行的多個建置環境。每個環境都以完全隔離的單位運作，具有自己的運算資源、執行期環境和相依性。透過批次建置組態，您可以指定他們需要多少平行環境，以及如何將測試分散到各個環境。

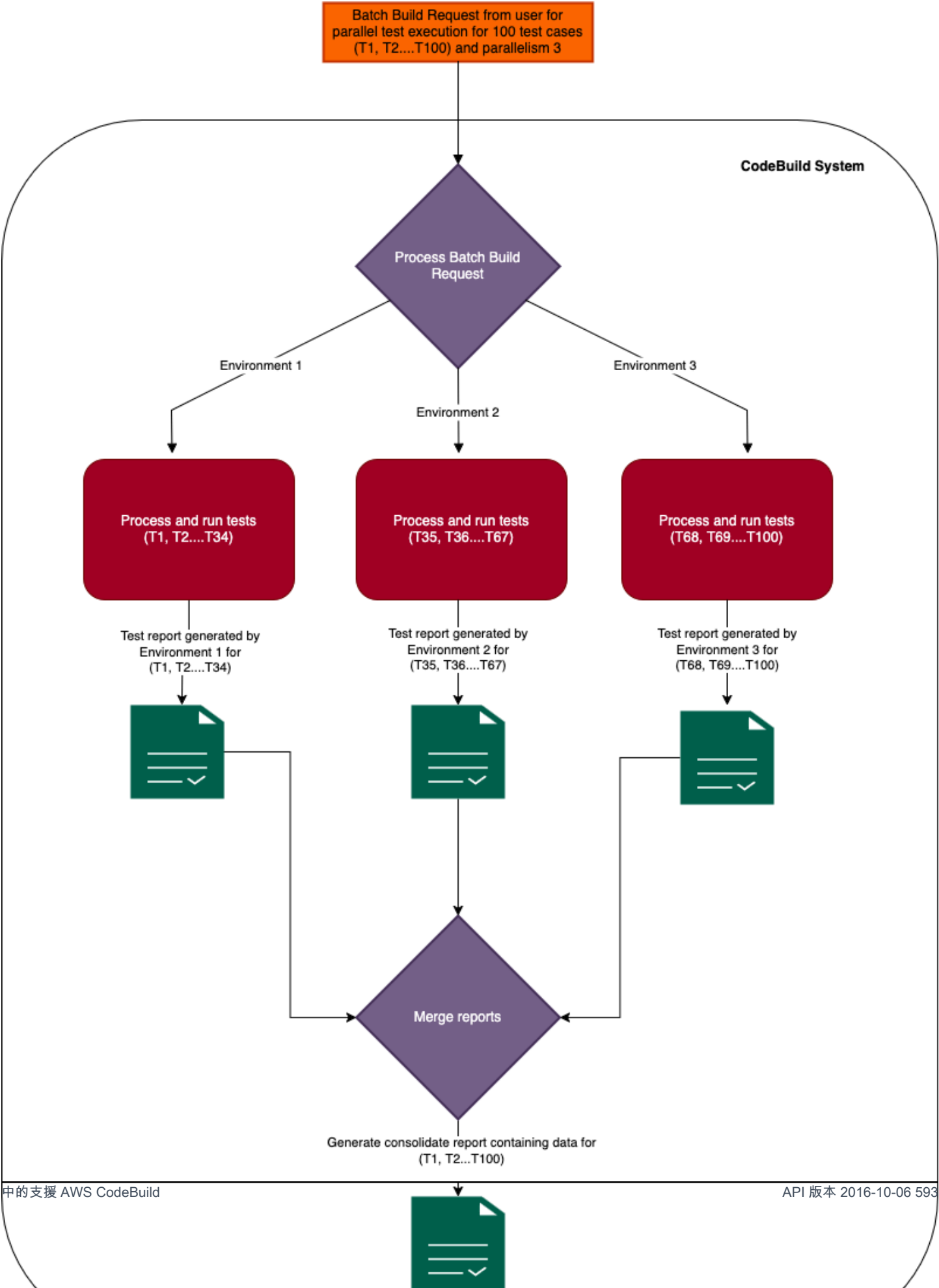
測試碎片 CLI

CodeBuild 透過其 CLI 工具 包含內建測試分佈機制 `codebuild-tests-run`，其會自動將測試劃分為不同的環境。

報告彙總

CodeBuild 實作的主要優勢之一是能夠順暢地處理測試結果彙總。當測試在不同的環境中執行時，CodeBuild 會自動收集每個環境的測試報告，並將其合併為批次建置層級的統一測試報告。此整合提供測試結果的完整檢視，同時維持平行執行的效率優勢。

下圖說明 中平行測試執行的完整概念 AWS CodeBuild。



在批次建置中啟用平行測試執行

若要平行執行測試，請更新批次建置 `buildspec` 檔案，以包含 `build-fanout` 欄位和平行建置數目，以在 `parallelism` 欄位中分割測試套件，如下所示。`parallelism` 欄位指定要設定多少獨立執行器來執行測試套件。

若要在多個平行執行環境中執行測試，請將 `parallelism` 欄位設定為大於零的值。在下面的範例中，`parallelism` 設定為 5，這表示 CodeBuild 會啟動五個相同的組建，並平行執行測試套件的一部分。

您可以使用 [codebuild-tests-run](#) CLI 命令來分割和執行測試。您的測試檔案會分割，而您的部分測試會在每次建置中執行。這可減少執行完整測試套件所需的整體時間。在下列範例中，測試將分割成五個，並根據測試的名稱計算分割點。

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - npm install jest-junit --save-dev
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - |
        codebuild-tests-run \
          --test-command 'npx jest --runInBand --coverage' \
          --files-search "codebuild-glob-search '**/_tests_/**/*.test.js'" \
          --sharding-strategy 'equal-distribution'

  post_build:
    commands:
      - codebuild-glob-search '**/*.xml'
      - echo "Running post-build steps..."
      - echo "Build completed on `date`"
```

```
reports:
  test-reports:
    files:
      - '**/junit.xml'
    base-directory: .
    discard-paths: yes
    file-format: JUNITXML
```

如果報告已設定為建置集組建，則每個組建都會分別產生測試報告，這些報告可以在 AWS CodeBuild 主控台中對應組建的報告索引標籤下檢視。

如需如何批次執行平行測試的詳細資訊，請參閱 [各種測試架構範例的平行測試執行](#)。

使用 **codebuild-tests-run** CLI 命令

AWS CodeBuild 提供 CLI，將測試命令和測試檔案位置做為輸入。具有此輸入的 CLI 會根據測試檔案名稱，將測試分割為 parallelism 欄位中指定的碎片數量。將測試檔案指派給碎片是由碎片策略決定。

```
codebuild-tests-run \
  --files-search "codebuild-glob-search '**/__tests__/*.js'" \
  --test-command 'npx jest --runInBand --coverage' \
  --sharding-strategy 'equal-distribution'
```

下表說明 CLI **codebuild-tests-run** 命令的欄位。

欄位名稱	Type	必要或選用	定義
test-command	String	必要	此命令用於執行測試。
files-search	String	必要	此命令會提供測試檔案的清單。您可以使用 AWS CodeBuild 提供的 codebuild-glob-search CLI 命令或您選擇的任何其他檔案搜尋工具。

欄位名稱	Type	必要或選用	定義
			<div>  Note 確保files-search 命令輸出檔案名稱，每個名稱都以新行分隔。 </div>
sharding-strategy	列舉	選用	<p>有效值：equal-distribution (預設)、stability</p> <ul style="list-style-type: none"> equal-distribution：根據測試檔案名稱平均碎片測試檔案。 stability：使用檔案名稱的一致雜湊來碎片測試檔案。 <p>如需詳細資訊，請參閱關於測試分割。</p>

CLI `codebuild-tests-run` 會先使用 `files-search` 參數中提供的命令來識別測試檔案的清單。然後，它會使用指定的碎片策略，決定指定給目前碎片（環境）的測試檔案子集。最後，此測試檔案子集會格式化為以空格分隔的清單，並在執行之前附加至 `test-command` 參數中提供的命令結尾。

對於不接受空格分隔清單的測試架構，`codebuild-tests-run` CLI 透過 `CODEBUILD_CURRENT_SHARD_FILES` 環境變數提供彈性的替代方案。此變數包含指派給目前建置碎片之測試檔案路徑的新行分隔清單。透過利用此環境變數，您可以輕鬆適應各種測試架構需求，容納預期輸入格式與空間分隔清單不同的測試架構需求。此外，您也可以根據測試架構的需求格式化測試檔案

名稱。以下是在 Linux CODEBUILD_CURRENT_SHARD_FILES 上使用 搭配 Django 架構的範例。以下是CODEBUILD_CURRENT_SHARD_FILES用來取得 Django 支援的點表示法檔案路徑：

```
codebuild-tests-run \
  -files-search "codebuild-glob-search '/tests/test.py'" \
  -test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
-E "s/\//__g; s/\.py$/; s/__/./g")' \
  -sharding-strategy 'equal-distribution'
```

Note

請注意，CODEBUILD_CURRENT_SHARD_FILES環境變數只能在 CLI codebuild-tests-run 的範圍內使用。

此外，如果您在 test-mand CODEBUILD_CURRENT_SHARD_FILES內使用，請放入雙引號CODEBUILD_CURRENT_SHARD_FILES內，如上述範例所示。

使用 **codebuild-glob-search** CLI 命令

AWS CodeBuild 提供稱為 的內建 CLI 工具codebuild-glob-search，可讓您根據一或多個 glob 模式搜尋工作目錄中的檔案。當您想要對專案中的特定檔案或目錄執行測試時，此工具特別有用。

Usage

codebuild-glob-search CLI 具有下列用量語法：

```
codebuild-glob-search <glob_pattern1> [<glob_pattern2> ...]
```

- *<glob_pattern1>*、*<glob_pattern2>*等：與工作目錄中的檔案相符的一或多個 glob 模式。
- *：符合任何字元序列（路徑分隔符號除外）。
- **：符合任何字元序列（包括路徑分隔符號）。

Note

確定 glob 字串有引號。若要檢查模式比對的結果，請使用 echo命令。

```
version: 0.2
```



```
phases:
  build:
    commands:
      - echo $(codebuild-glob-search '**/__tests__/*.js')
      - codebuild-glob-search '**/__tests__/*.js' | xargs -n 1 echo
```

Output

CLI 將輸出符合所提供 glob 模式的檔案路徑換行分隔清單。傳回的檔案路徑將相對於工作目錄。

如果找不到符合所提供模式的檔案，CLI 會輸出一則訊息，指出找不到檔案。

請注意，由於任何指定模式而找到的目錄將從搜尋結果中排除。

範例

如果您只想要搜尋測試目錄及其子目錄中具有 .js 副檔名的檔案，您可以搭配 CLI `codebuild-glob-search` 使用下列命令：

```
codebuild-glob-search '**/__tests__/*.js'
```

此命令將搜尋 `__tests__` 目錄及其子目錄中具有 .js 副檔名的所有檔案，如 模式所示。

關於測試分割

AWS CodeBuild 的測試分割功能可讓您在多個運算執行個體間平行化測試套件執行，進而縮短整體測試執行時間。此功能是透過 CodeBuild 專案設定中的批次組態和 `buildspec` 檔案中的 `codebuild-tests-run` 公用程式來啟用。

測試會根據指定的碎片策略進行分割。CodeBuild 提供兩種碎片策略，如下所示：

相等分佈

`equal-distribution` 碎片策略會根據測試檔案名稱的字母順序，將測試分割為平行組建。此方法會先排序測試檔案，然後採用以區塊為基礎的方法來分配它們，確保類似的檔案分組在一起進行測試。建議在處理一組相對較小的測試檔案時。雖然此方法旨在為每個碎片配置大約相等數量的檔案，但最大差異為 1，但不保證穩定性。在後續建置中新增或移除測試檔案時，現有檔案的分佈可能會變更，從而導致碎片之間的重新指派。

穩定性

`stability` 碎片策略採用一致的雜湊演算法，在碎片之間分割測試，確保檔案分佈保持穩定。新增或移除新檔案時，此方法可確保現有的file-to-shard指派大致保持不變。對於大型測試套件，建議使用穩定性選項，將測試平均分散到碎片。此機制旨在提供近乎相等的分佈，確保每個碎片都會收到相似數量的檔案，且只有最小的差異。雖然穩定性策略不保證理想的相等分佈，但它提供近乎相等的分佈，即使在新增或移除檔案時，也能維持跨組建的檔案指派一致性。

若要啟用測試分割，您需要在 CodeBuild 專案設定中設定批次區段，指定所需的parallelism層級和其他相關參數。此外，您需要在 buildspec 檔案中包含 `codebuild-tests-run` 公用程式，以及適當的測試命令和分割方法。

自動合併個別建置報告

在廣發批次建置中，AWS CodeBuild 支援將個別建置報告自動合併為合併批次層級報告。此功能提供批次中所有組建的測試結果和程式碼涵蓋範圍的完整檢視。

運作方式

執行fanout批次建置時，每個個別建置都會產生[測試報告](#)。CodeBuild 接著會自動將來自不同組建的相同報告合併為統一報告，該報告會連接到批次組建。這些合併報告可透過 [BatchGetBuildBatches](#) API 的 `reportArns` 欄位輕鬆存取，也可以在 主控台的報告索引標籤中檢視。此合併功能也延伸至自動探索的報告。

合併的報告是在 buildspec 中指定的[報告群組](#)下建立，或由 CodeBuild 自動探索。您可以直接在這些報告群組下分析合併報告的趨勢，提供對相同建置批次專案歷史組建的整體組建效能和品質指標的寶貴見解。

對於批次中的每個個別組建，CodeBuild 會自動建立個別的報告群組。這些遵循特定的命名慣例，將批次建置報告群組名稱與尾碼 結合BuildFanoutShard<shard_number>，其中 shard_number代表建立報告群組的碎片數量。此組織可讓您追蹤和分析合併和個別建置層級的趨勢，提供監控和評估其建置程序的彈性。

批次建置報告遵循與[個別建置報告](#)相同的結構。報告索引標籤中的下列金鑰欄位專屬於批次建置報告：

批次建置報告狀態

批次建置報告的狀態會遵循特定規則，視報告類型而定：

- 測試報告：
 - 成功：當所有個別建置報告都成功時，狀態會設定為成功。

- 失敗：如果任何個別建置報告失敗，狀態會設定為失敗。
- 未完成：如果任何個別建置報告遺失或狀態不完整，狀態會標記為未完成。
- 程式碼涵蓋範圍報告：
 - 完成：當所有個別建置報告完成時，狀態設定為完成。
 - 失敗：如果任何個別建置報告失敗，狀態會設定為失敗。
 - 未完成：如果任何個別建置報告遺失或狀態不完整，狀態會標記為未完成。

測試摘要

合併的測試報告會合併所有個別建置報告的下列欄位：

- duration-in-nano-seconds：所有個別建置報告之間的測試持續時間上限，以奈秒為單位。
- 總計：所有測試案例的合併計數，加總每個組建的測試總數。
- status-counts：透過彙總所有個別組建中每個狀態類型的計數，來提供測試狀態的合併檢視，例如通過、失敗或略過。

程式碼涵蓋範圍摘要

合併程式碼涵蓋範圍報告使用下列計算，結合所有個別組建中的欄位：

- 分支涵蓋：個別報告中所有涵蓋分支的總和。
- 分支遺失：個別報告中所有遺失分支的總和。
- branch-coverage-percentage： $(\text{Total covered branches} / \text{Total branches}) * 100$
- 涵蓋的行數：個別報告中所有涵蓋行數的總和。
- 遺漏行：個別報告中所有遺漏行的總和。
- lines-coverage-percentage： $(\text{Total covered lines} / \text{Total lines}) * 100$

執行 ID

批次建置 ARN。

測試案例

合併的報告包含來自個別建置之所有測試案例的合併清單，可透過 [DescribeTestCases](#) API 和主控台下的批次建置報告存取。

程式碼涵蓋範圍

合併程式碼涵蓋範圍報告提供所有個別組建中每個檔案的合併行和分支涵蓋範圍資訊，可透過 主控台下的 [DescribeCodeCoverages](#) API 和批次組建報告存取。注意：對於分佈在不同碎片的多個測試檔案所涵蓋的檔案，合併的報告會使用下列選擇條件：

1. 主要選擇是以碎片之間的最高線涵蓋範圍為基礎。
2. 如果多個碎片的折線涵蓋範圍相等，則會選取分支涵蓋範圍最高的碎片。

各種測試架構範例的平行測試執行

您可以使用 `codebuild-tests-run` CLI 命令在平行執行環境中分割和執行測試。下一節提供各種架構 `buildspec.yml` 的範例，說明 `codebuild-tests-run` 命令的使用方式。

- 以下每個範例都包含五個 `parallelism` 層級，這表示將建立五個相同的執行環境來分割您的測試。您可以透過修改 `build-fanout` 區段中的 `parallelism` 值來選擇適合您專案的 `parallelism` 關卡。
- 以下每個範例顯示將您的測試設定為依測試檔案名稱分割，這是預設值。這會將測試平均分配到平行執行環境。

開始使用之前，請參閱 [在批次建置中執行平行測試](#) 以取得詳細資訊。

如需使用 CLI `codebuild-tests-run` 命令時選項的完整清單，請參閱 [使用 `codebuild-tests-run` CLI 命令](#)。

主題

- [使用 Django 設定平行測試](#)
- [使用 Elixir 設定平行測試](#)
- [使用 Go 設定平行測試](#)
- [使用 Java 設定平行測試 \(Maven\)](#)
- [使用 Javascript 設定平行測試 \(Jest\)](#)
- [使用 Kotlin 設定平行測試](#)
- [使用 PHPUnit 設定平行測試](#)
- [使用 Pytest 設定平行測試](#)
- [使用 Ruby \(Cucumber\) 設定平行測試](#)
- [使用 Ruby \(RSpec\) 設定平行測試](#)

使用 Django 設定平行測試

以下是在 Ubuntu 平台上 `buildspec.yml` 使用 Django 顯示平行測試執行的 範例：

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5

phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - sudo yum install -y python3 python3-pip
      - python3 -m ensurepip --upgrade
      - python3 -m pip install django
  pre_build:
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Django Tests'
      - |
        codebuild-tests-run \
          --test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES"
| sed -E "s/\//__/g; s/\.py$/;/; s/__/./g")' \
          --files-search "codebuild-glob-search '**/tests/*test_*.py'" \
          --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo 'Test execution completed'
```

上述範例顯示環境變數 的使用情

況CODEBUILD_CURRENT_SHARD_FILES。CODEBUILD_CURRENT_SHARD_FILES 這裡用於擷取 Django 支援的點表示法檔案路徑。使用CODEBUILD_CURRENT_SHARD_FILES內部雙引號，如上所示。

使用 Elixir 設定平行測試

以下是在 Ubuntu 平台上buildspec.yml使用 Elixir 顯示平行測試執行的 範例：

```
version: 0.2
```

```
batch:
```

```
fast-fail: false
build-fanout:
  parallelism: 5

phases:
  install:
    commands:
      - echo 'Installing Elixir dependencies'
      - sudo apt update
      - sudo DEBIAN_FRONTEND=noninteractive apt install -y elixir
      - elixir --version
      - mix --version
  pre_build:
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Elixir Tests'
      - |
        codebuild-tests-run \
          --test-command 'mix test' \
          --files-search "codebuild-glob-search '**/test/**/*_test.exs'" \
          --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

使用 Go 設定平行測試

以下是buildspec.yml在 Linux 平台上使用 Go 顯示平行測試執行的 範例：

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Fetching Go version'
```

```
- go version
pre_build:
  commands:
    - echo 'prebuild'
build:
  commands:
    - echo 'Running go Tests'
    - go mod init calculator
    - cd calc
    - |
      codebuild-tests-run \
        --test-command "go test -v calculator.go" \
        --files-search "codebuild-glob-search '**/*test.go'"
post_build:
  commands:
    - echo "Test execution completed"
```

在上述範例中，calculator.go函數包含要測試的簡單數學函數，且所有測試檔案和calculator.go檔案都在 calc 資料夾內。

使用 Java 設定平行測試 (Maven)

以下是在 Linux 平台上buildspec.yml使用 Java 顯示平行測試執行的 範例：

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo "Running mvn test"
      - |
        codebuild-tests-run \
          --test-command 'mvn test -Dtest=$(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
"s|src/test/java/||g; s/\..java//g; s|/|.|g; s/ /,/g" | tr "\n" "," | sed "s/,,$//")' \
```

```
--files-search "codebuild-glob-search '**/test/**/*.java'"

post_build:
  commands:
    - echo "Running post-build steps..."
    - echo "Test execution completed"
```

在指定的範例中，環境變數CODEBUILD_CURRENT_SHARD_FILES包含目前碎片中的測試檔案，並以換行分隔。這些檔案會以 Maven -Dtest 參數接受的格式轉換為以逗號分隔的類別名稱清單。

使用 Javascript 設定平行測試 (Jest)

以下是在 Ubuntu 平台上buildspec.yml使用 Javascript 顯示平行測試執行的 範例：

```
version: 0.2

batch:
  fast-fail: true
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Node.js dependencies'
      - apt-get update
      - apt-get install -y nodejs
      - npm install
      - npm install --save-dev jest-junit
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running JavaScript Tests'
      - |
        codebuild-tests-run \
          --test-command "npx jest" \
          --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
          --sharding-strategy 'stability'
  post_build:
    commands:
```



```
- echo 'Test execution completed'
```

使用 Kotlin 設定平行測試

以下是在 Linux 平台上buildspec.yml使用 Kotlin 顯示平行測試執行的 範例：

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 2
    ignore-failure: false

phases:
  install:
    runtime-versions:
      java: corretto11
    commands:
      - echo 'Installing dependencies'
      - KOTLIN_VERSION="1.8.20" # Replace with your desired version
      - curl -o kotlin-compiler.zip -L "https://github.com/JetBrains/kotlin/releases/download/v${KOTLIN_VERSION}/kotlin-compiler-${KOTLIN_VERSION}.zip"
      - unzip kotlin-compiler.zip -d /usr/local
      - export PATH=$PATH:/usr/local/kotlinc/bin
      - kotlin -version
      - curl -O https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-standalone/1.8.2/junit-platform-console-standalone-1.8.2.jar
    pre_build:
      commands:
        - echo 'prebuild'
  build:
    commands:
      - echo 'Running Kotlin Tests'
      - |
        codebuild-tests-run \
          --test-command 'kotlinc src/main/kotlin/*.kt $(echo
"$CODEBUILD_CURRENT_SHARD_FILES" | tr "\n" " ") -d classes -cp junit-platform-console-standalone-1.8.2.jar' \
          --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
      - |
        codebuild-tests-run \
          --test-command '
```

```

        java -jar junit-platform-console-standalone-1.8.2.jar --class-path classes
\
        $(for file in $CODEBUILD_CURRENT_SHARD_FILES; do
            class_name=$(basename "$file" .kt)
            echo "--select-class $class_name"
        done)
    ' \
    --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
post_build:
  commands:
    - echo "Test execution completed"

```

在上述範例中，會使用兩次 `codebuild-tests-run` CLI。在第一次執行期間，`kotlinc` 會編譯檔案。`CODEBUILD_CURRENT_SHARD_FILES` 變數會擷取指派給目前碎片的測試檔案，然後轉換為以空格分隔的清單。在第二次執行中，JUnit 會執行測試。同樣地，`CODEBUILD_CURRENT_SHARD_FILES` 會擷取指派給目前碎片的測試檔案，但這次會轉換為類別名稱。

使用 PHPUnit 設定平行測試

以下是在 Linux 平台上 `buildspec.yml` 使用 PHPUnit 顯示平行測試執行的範例：

```

version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Install dependencies'
      - composer require --dev phpunit/phpunit
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running phpunit Tests'
      - composer dump-autoload
      - |

```

```
codebuild-tests-run \  
  --test-command "./vendor/bin/phpunit --debug" \  
  --files-search "codebuild-glob-search '**/tests/*Test.php'"  
post_build:  
  commands:  
    - echo 'Test execution completed'
```

使用 Pytest 設定平行測試

以下是在 Ubuntu 平台上buildspec.yml使用 Pytest 顯示平行測試執行的 範例：

```
version: 0.2  
  
batch:  
  fast-fail: false  
  build-fanout:  
    parallelism: 5  
    ignore-failure: false  
  
phases:  
  install:  
    commands:  
      - echo 'Installing Python dependencies'  
      - apt-get update  
      - apt-get install -y python3 python3-pip  
      - pip3 install --upgrade pip  
      - pip3 install pytest  
  build:  
    commands:  
      - echo 'Running Python Tests'  
      - |  
        codebuild-tests-run \  
          --test-command 'python -m pytest' \  
          --files-search "codebuild-glob-search 'tests/test_*.py'" \  
          --sharding-strategy 'equal-distribution'  
  post_build:  
    commands:  
      - echo "Test execution completed"
```

以下是在 Windows 平台上buildspec.yml使用 Pytest 顯示平行測試執行的 範例：

```
version: 0.2
```

```

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - pip install pytest
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running pytest'
      - |
        & codebuild-tests-run `
        --test-command 'pytest @("$env:CODEBUILD_CURRENT_SHARD_FILES" -split \"`r?`n
        \")' `
        --files-search "codebuild-glob-search '**/test_*.py' '**/*_test.py'" `
        --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"

```

在上述範例中，CODEBUILD_CURRENT_SHARD_FILES環境變數用於擷取指派給目前碎片並做為陣列傳遞至 pytest 命令的測試檔案。

使用 Ruby (Cucumber) 設定平行測試

以下是在 Linux 平台上buildspec.yml使用 Cucumber 顯示平行測試執行的 範例：

```

version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:

```

```
install:
  commands:
    - echo 'Installing Ruby dependencies'
    - gem install bundler
    - bundle install
pre_build:
  commands:
    - echo 'prebuild'
build:
  commands:
    - echo 'Running Cucumber Tests'
    - cucumber --init
    - |
      codebuild-tests-run \
        --test-command "cucumber" \
        --files-search "codebuild-glob-search '**/*.feature'"
post_build:
  commands:
    - echo "Test execution completed"
```

使用 Ruby (RSpec) 設定平行測試

以下是在 Ubuntu 平台上buildspec.yml使用 RSpec 顯示平行測試執行的 範例：

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Ruby dependencies'
      - apt-get update
      - apt-get install -y ruby ruby-dev build-essential
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo 'Running Ruby Tests'
```

```
- |
  codebuild-tests-run \
    --test-command 'bundle exec rspec' \
    --files-search "codebuild-glob-search 'spec/**/*.rb'" \
    --sharding-strategy 'equal-distribution'
post_build:
  commands:
    - echo "Test execution completed"
```

快取建置以改善效能

當您的專案使用快取來建置時，可以為您省下時間。快取可以存放組建環境的可重複使用部分，並在多個組建間使用這些部分。您的建置專案可以使用兩種快取類型的其中之一：Amazon S3 或本機。如果您使用本機快取，您必須從三種快取模式中選擇一或多種：來源快取、Docker 層快取和自訂快取。

Note

Docker 層快取模式僅適用於 Linux 環境。如果您選擇此模式，您必須以特殊權限模式執行您的組建。CodeBuild 專案授予特權模式會授予其容器存取所有裝置的權限。如需詳細資訊，請參閱 Docker Docs 網站上的[執行期權限和 Linux 功能](#)。

主題

- [Amazon S3 快取](#)
- [本機快取](#)
- [指定本機快取](#)

Amazon S3 快取

Amazon S3 快取會將快取存放在跨多個建置主機可用的 Amazon S3 儲存貯體中。對於較下載更昂貴的中小型建置成品而言，這是不錯的選擇。

若要在組建中使用 Amazon S3，您可以指定要在 中快取之檔案的路徑 `buildspec.yml`。CodeBuild 會自動將快取儲存並更新至專案上設定的 Amazon S3 位置。如果您未指定檔案路徑，CodeBuild 會盡最大努力快取常見的語言相依性，以協助您加速建置。您可以在建置日誌中檢視快取詳細資訊。

此外，如果您想要有多個版本的快取，您可以在 `buildspec.yml` 中定義快取金鑰。CodeBuild 會將快取存放在此快取金鑰的內容下，並建立唯一的快取複本，一旦建立就不會更新。快取金鑰也可以跨專案共用。只有在指定金鑰時，才能使用動態金鑰、快取版本控制和跨組建快取共用等功能。

若要進一步了解 `buildspec` 檔案中的快取語法，請參閱 `buildspec` 參考[快取](#)中的。

主題

- [產生動態金鑰](#)
- [codebuild-hash-files](#)
- [快取版本](#)
- [專案之間的快取共用](#)
- [Buildspec 範例](#)

產生動態金鑰

快取金鑰可以包含 Shell 命令和環境變數，使其獨一無二，以便在金鑰變更時啟用自動快取更新。例如，您可以使用 `package-lock.json` 檔案的雜湊來定義金鑰。當該檔案中的相依性變更時，雜湊 - 因此是快取金鑰 - 會變更，觸發自動建立新的快取。

```
cache:
  key: npm-key-$(codebuild-hash-files package-lock.json)
```

CodeBuild 將評估表達式 `$(codebuild-hash-files package-lock.json)` 以取得最終金鑰：

```
npm-key-abc123
```

您也可以使用環境變數定義快取金鑰，例如 `CODEBUILD_RESOLVED_SOURCE_VERSION`。這可確保每當來源變更時，都會產生新的金鑰，進而自動儲存新的快取：

```
cache:
  key: npm-key-$CODEBUILD_RESOLVED_SOURCE_VERSION
```

CodeBuild 將評估表達式並取得最終索引鍵：

```
npm-key-046e8b67481d53bdc86c3f6affdd5d1afae6d369
```

codebuild-hash-files

codebuild-hash-files 是一種 CLI 工具，可使用 glob 模式計算 CodeBuild 來源目錄中一組檔案的 SHA-256 雜湊：

```
codebuild-hash-files <glob-pattern-1> <glob-pattern-2> ...
```

以下是使用 的一些範例codebuild-hash-files：

```
codebuild-hash-files package-lock.json
codebuild-hash-files '**/*.md'
```

快取版本

快取版本是從快取目錄路徑產生的雜湊。如果兩個快取有不同的版本，則在比對程序期間會將它們視為不同的快取。例如，以下兩個快取會被視為不同，因為它們參考不同的路徑：

```
version: 0.2

phases:
  build:
    commands:
      - pip install pandas==2.2.3 --target pip-dependencies
cache:
  key: pip-dependencies
  paths:
    - "pip-dependencies/**/*"
```

```
version: 0.2

phases:
  build:
    commands:
      - pip install pandas==2.2.3 --target tmp/pip-dependencies
cache:
  key: pip-dependencies
  paths:
    - "tmp/pip-dependencies/**/*"
```


專案之間的快取共用

您可以使用 `cache` 區段下的 `cacheNamespace` API 欄位，跨多個專案共用快取。此欄位定義快取的範圍。若要共用快取，您必須執行下列動作：

- 使用相同的 `cacheNamespace`。
- 指定相同的快取 key。
- 定義相同的快取路徑。
- `pathPrefix` 如果設定，請使用相同的 Amazon S3 儲存貯體和。

這可確保一致性，並啟用跨專案的快取共用。

指定快取命名空間（主控台）

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 選擇建立專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
3. 在成品中，選擇其他組態。
4. 針對快取類型，選擇 Amazon S3。
5. 針對快取命名空間 - 選用，輸入命名空間值。

▼ Additional configuration

Cache, encryption key

Encryption key - optional

Provide the AWS KMS customer master key used to encrypt this build's output artifacts. The default is your AWS-managed customer master key for S3.

arn:aws:kms:<region-ID>:<account-ID>:key/<key-ID>

Cache type

Amazon S3 ▼

Cache bucket**Cache path prefix - optional****Cache lifecycle (days) - optional**

You can apply a lifecycle expiration action to all or a subset of objects in the cache bucket based on the path prefix.

[+ Add expiration](#)**Cache namespace - optional**

test-cache-namespace

Provide a cache namespace if you want to share caches across projects.

6. 繼續執行預設值，然後選擇建立建置專案。

指定快取命名空間 (AWS CLI)

您可以使用 中的 `--cache` 參數 AWS CLI 來指定快取命名空間。

```
--cache '{"type": "S3", "location": "your-s3-bucket", "cacheNamespace": "test-cache-namespace"}'
```

Buildspec 範例

以下是幾個常見語言的 buildspec 範例：

主題

- [快取 Node.js 相依性](#)

- [快取 Python 相依性](#)
- [快取 Ruby 相依性](#)
- [快取 Go 相依性](#)

快取 Node.js 相依性

如果您的專案包含 `package-lock.json` 檔案並使用 `npm` 來管理 Node.js 相依性，下列範例會示範如何設定快取。根據預設，會將相依性 `npm` 安裝到 `node_modules` 目錄中。

```
version: 0.2

phases:
  build:
    commands:
      - npm install
cache:
  key: npm-$(codebuild-hash-files package-lock.json)
  paths:
    - "node_modules/**/*"
```

快取 Python 相依性

如果您的專案包含 `requirements.txt` 檔案，並使用 `pip` 來管理 Python 相依性，下列範例會示範如何設定快取。根據預設，`pip` 會將套件安裝到系統的 `site-packages` 目錄中。

```
version: 0.2

phases:
  build:
    commands:
      - pip install -r requirements.txt
cache:
  key: python-$(codebuild-hash-files requirements.txt)
  paths:
    - "/root/.pyenv/versions/${python_version}/lib/python${python_major_version}/site-packages/**/*"
```

此外，您可以在特定目錄中安裝相依性，並設定該目錄的快取。

```
version: 0.2
```

```
phases:
  build:
    commands:
      - pip install -r requirements.txt --target python-dependencies
cache:
  key: python-$(codebuild-hash-files requirements.txt)
  paths:
    - "python-dependencies/**/*"
```

快取 Ruby 相依性

如果您的專案包含 `Gemfile.lock` 檔案並使用 Bundler 來管理 Gem 相依性，以下範例示範如何有效地設定快取。

```
version: 0.2

phases:
  build:
    commands:
      - bundle install --path vendor/bundle
cache:
  key: ruby-$(codebuild-hash-files Gemfile.lock)
  paths:
    - "vendor/bundle/**/*"
```

快取 Go 相依性

如果您的專案包含 `go.sum` 檔案，並使用 Go 模組來管理相依性，下列範例會示範如何設定快取。根據預設，Go 模組會下載並存放在 `${GOPATH}/pkg/mod` 目錄中。

```
version: 0.2

phases:
  build:
    commands:
      - go mod download
cache:
  key: go-$(codebuild-hash-files go.sum)
  paths:
    - "/go/pkg/mod/**/*"
```

本機快取

本機快取將快取存放在組建主機的本機，而僅供該組建主機使用。這是中大型建置成品的好選項，因為快取可立即在建置主機上使用。如果是不常存取的組建，這不是最佳選項。這表示組建效能不會受到網路傳輸時間所影響。

如果您選擇本機快取，您必須選擇以下一或多個快取模式：

- 來源快取模式會快取主要和次要來源的 Git 中繼資料。建立快取之後，後續組建只會提取遞交之間的變更。如果專案有全新的工作目錄，且來源是大型的 Git 儲存庫，此模式是很好的選擇。如果您選擇此選項，且您的專案不使用 Git 儲存庫 (AWS CodeCommit、GitHub、GitHub Enterprise Server 或 Bitbucket)，則會忽略此選項。
- Docker 層快取模式會快取現有的 Docker 層。如果專案會建置或提取大型 Docker 影像，此模式是很好的選擇。它可以避免從網路提取大型 Docker 影像所造成的效能問題。

Note

- 您只能在 Linux 環境中使用 Docker 層快取。
- 您必須設定 privileged 旗標，專案才能具備所需的 Docker 許可。

根據預設，非 VPC 建置會啟用 Docker 協助程式。如果您想要將 Docker 容器用於 VPC 建置，請參閱 Docker 文件網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特權模式。

- 使用 Docker 層快取之前，您應該考慮安全隱憂。

- 自訂快取模式會快取您在 buildspec 檔案中指定的目錄。如果您的組建案例不適合其他兩個本機快取模式，此模式是很好的選擇。如果您使用自訂快取：
 - 只能指定目錄來快取。您不能指定個別檔案。
 - 符號連結用來參考快取的目錄。
 - 在您的組建下載其專案來源之前，快取的目錄會連結到您的組建。如果快取項目具有相同的名稱，則快取項目會覆寫來源項目。目錄的指定方式是使用 buildspec 檔案中的快取路徑。如需詳細資訊，請參閱[Buildspec 語法](#)。
 - 避免來源和快取中的目錄名稱相同。本機快取的目錄可能會覆寫或刪除具有相同名稱之來源儲存庫中的目錄內容。

Note

LINUX_GPU_CONTAINER 環境類型和BUILD_GENERAL1_2XLARGE運算類型不支援本機快取。如需詳細資訊，請參閱[建置環境運算模式和類型](#)。

Note

當您設定 CodeBuild 使用 VPC 時，不支援本機快取。如需搭配 CodeBuild 使用 VPCs 的詳細資訊，請參閱[AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用](#)。

指定本機快取

您可以使用 AWS CLI、主控台、SDK 或 CloudFormation 來指定本機快取。如需本機快取的詳細資訊，請參閱[本機快取](#)。

主題

- [指定本機快取 \(CLI\)](#)
- [指定本機快取 \(主控台\)](#)
- [指定本機快取 \(CloudFormation\)](#)

指定本機快取 (CLI)

您可以使用 中的 `--cache` 參數 AWS CLI 來指定三種本機快取類型的每一個。

- 若要指定來源快取：

```
--cache type=LOCAL,mode=[LOCAL_SOURCE_CACHE]
```

- 若要指定 Docker 層快取：

```
--cache type=LOCAL,mode=[LOCAL_DOCKER_LAYER_CACHE]
```

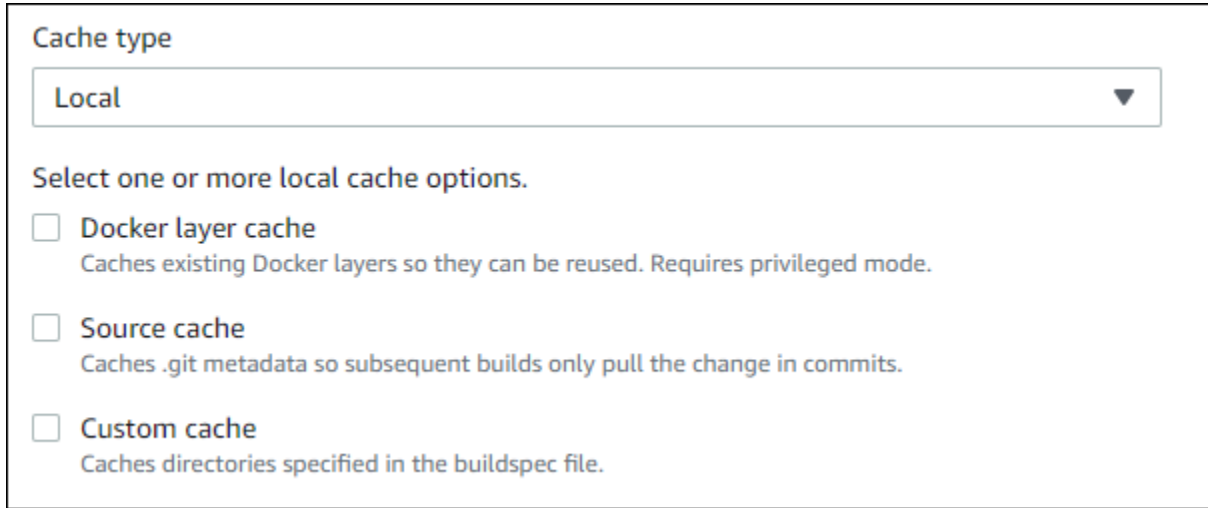
- 若要指定自訂快取：

```
--cache type=LOCAL,mode=[LOCAL_CUSTOM_CACHE]
```

如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。

指定本機快取 (主控台)

您可以在主控台的 Artifacts (成品) 區段中指定快取。針對快取類型，選擇 Amazon S3 或 Local。如果您選擇 Local (本機)，請從三種本機快取選項中選擇一或多種。



Cache type

Local ▼

Select one or more local cache options.

- ☐ Docker layer cache
Caches existing Docker layers so they can be reused. Requires privileged mode.
- ☐ Source cache
Caches .git metadata so subsequent builds only pull the change in commits.
- ☐ Custom cache
Caches directories specified in the buildspec file.

如需詳細資訊，請參閱[建立組建專案 \(主控台\)](#)。

指定本機快取 (CloudFormation)

如果您使用 CloudFormation 指定本機快取，請在 Cache 屬性上指定 TypeLOCAL。下列 YAML 格式 CloudFormation 程式碼範例會指定所有三種本機快取類型。您可以指定這些類型的任意組合。如果您使用 Docker 層快取，在 Environment 下，您必須將 PrivilegedMode 設定為 true，將 Type 設定為 LINUX_CONTAINER。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: <service-role>
    Artifacts:
      Type: S3
      Location: <bucket-name>
      Name: myArtifact
      EncryptionDisabled: true
      OverrideArtifactName: true
    Environment:
      Type: LINUX_CONTAINER
```

```
ComputeType: BUILD_GENERAL1_SMALL
Image: aws/codebuild/standard:5.0
Certificate: <bucket/cert.zip>
# PrivilegedMode must be true if you specify LOCAL_DOCKER_LAYER_CACHE
PrivilegedMode: true
Source:
  Type: GITHUB
  Location: <github-location>
  InsecureSsl: true
  GitCloneDepth: 1
  ReportBuildStatus: false
TimeoutInMinutes: 10
Cache:
  Type: LOCAL
  Modes: # You can specify one or more cache mode,
    - LOCAL_CUSTOM_CACHE
    - LOCAL_DOCKER_LAYER_CACHE
    - LOCAL_SOURCE_CACHE
```

Note

根據預設，非 VPC 建置會啟用 Docker 協助程式。如果您想要將 Docker 容器用於 VPC 建置，請參閱 Docker Docs 網站上的[執行期權限和 Linux 功能](#)，並啟用特權模式。此外，Windows 不支援特權模式。

如需詳細資訊，請參閱[建立建置專案 \(CloudFormation\)](#)。

在中對組建進行偵錯 AWS CodeBuild

AWS CodeBuild 提供兩種在開發和疑難排解期間對組建進行偵錯的方法。您可以使用 CodeBuild 沙盒環境來調查問題並即時驗證修正，也可以使用 AWS Systems Manager Session Manager 連線到建置容器並檢視容器狀態。

使用 CodeBuild 沙盒對組建進行偵錯

CodeBuild 沙盒環境在安全且隔離的環境中提供互動式偵錯工作階段。您可以透過 AWS 管理主控台 或直接與環境互動 AWS CLI、執行命令，並逐步驗證您的建置程序。它使用經濟實惠的每秒計費模型，並支援與建置環境的來源提供者 AWS 和服務相同的原生整合。您也可以使用 SSH 用戶端或從整合式開發環境 (IDEs) 連線到沙盒環境。

若要進一步了解 CodeBuild 沙盒定價，請造訪 [CodeBuild 定價文件](#)。如需詳細說明，請參閱 [使用 CodeBuild 沙盒對組建進行偵錯](#) 文件。

使用 Session Manager 偵錯組建

AWS Systems Manager Session Manager 可讓您直接存取在實際執行環境中執行的組建。此方法可讓您連線至作用中的建置容器，並即時檢查建置程序。您可以檢查檔案系統、監控執行中的程序，並在問題發生時進行故障診斷。

如需詳細說明，請參閱 [使用 Session Manager 偵錯組建](#) 文件。

使用 CodeBuild 沙盒對組建進行偵錯

在 中 AWS CodeBuild，您可以使用 CodeBuild 沙盒對組建進行偵錯，以執行自訂命令並對組建進行疑難排解。

主題

- [先決條件](#)
- [使用 CodeBuild 沙盒對組建進行偵錯（主控台）](#)
- [使用 CodeBuild 沙盒對組建進行偵錯 \(AWS CLI\)](#)
- [教學課程：使用 SSH 連線至沙盒](#)
- [AWS CodeBuild 對沙盒 SSH 連線問題進行故障診斷](#)

先決條件

使用 CodeBuild 沙盒之前，請確定您的 CodeBuild 服務角色具有下列 SSM 政策：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssmmessages:CreateControlChannel",
        "ssmmessages:CreateDataChannel",
        "ssmmessages:OpenControlChannel",
```

```

        "ssmmessages:OpenDataChannel"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:StartSession"
    ],
    "Resource": [
      "arn:aws:codebuild:us-east-1:111122223333:build/*",
      "arn:aws:ssm:us-east-1::document/AWS-StartSSHSession"
    ]
  }
]
}

```

使用 CodeBuild 沙盒對組建進行偵錯（主控台）

使用下列指示在 主控台中執行命令，並將 SSH 用戶端連線至 CodeBuild 沙盒。

使用 CodeBuild 沙盒執行命令（主控台）

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。選擇建置專案，然後選擇偵錯建置。

sandbox-project Actions ▼ Create trigger Edit Clone Clear cache Debug build Start build with overrides **Start build**

Configuration

Source provider No source	Primary repository -	Artifacts upload location -	Service role arn:aws:iam::[redacted]:role/service-role/codebuild-sandbox-project-service-role
Public builds Disabled			

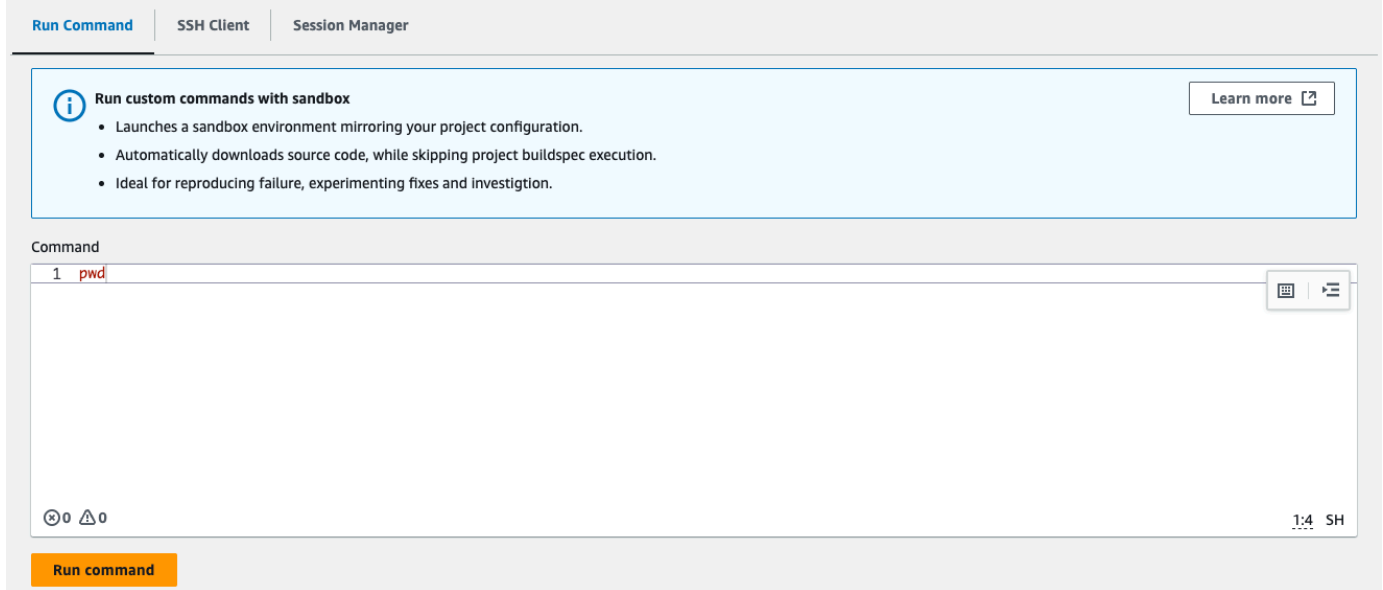
Build history Batch history **Project details** Build triggers Metrics Debug sessions

Project configuration Edit

Name sandbox-project	Description -
Project ARN arn:aws:codebuild:us-east-1:[redacted]:project/sandbox-project	Build badge Disabled

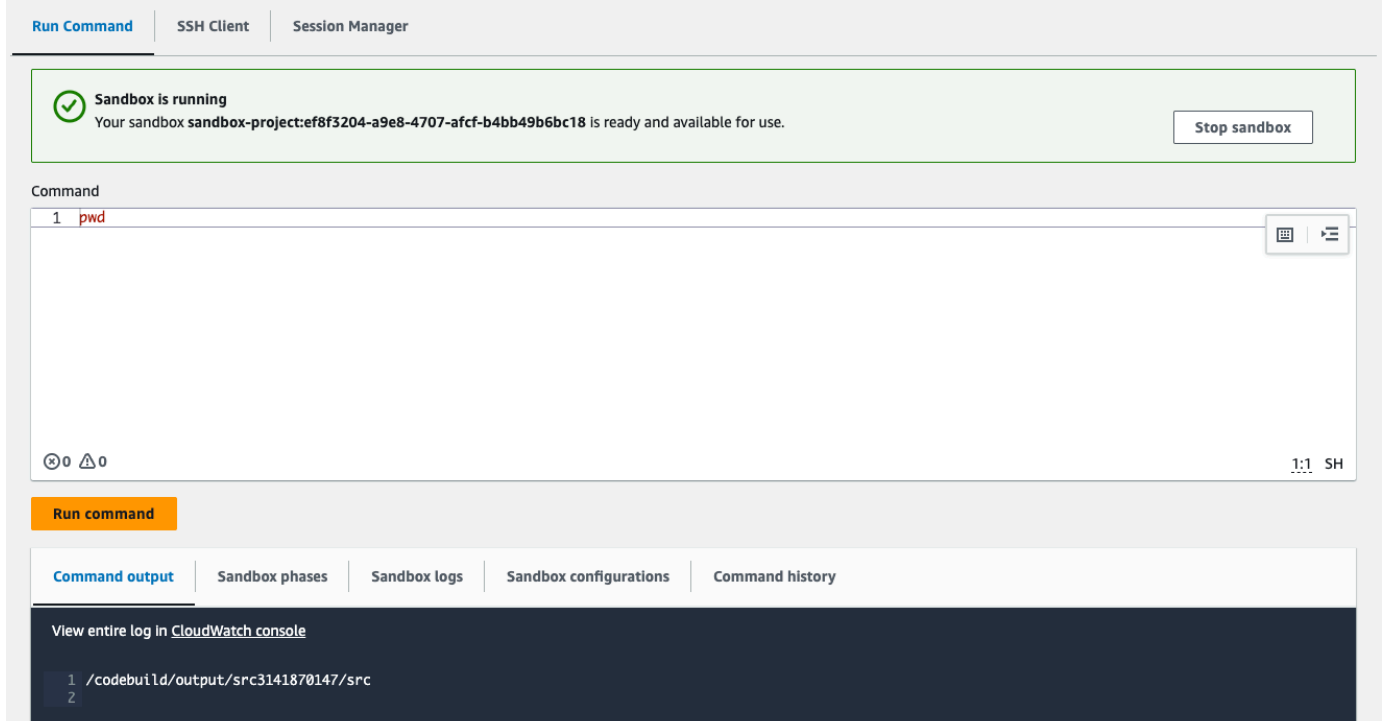
3. 在執行命令索引標籤中，輸入您的自訂命令，然後選擇執行命令。

Debug build

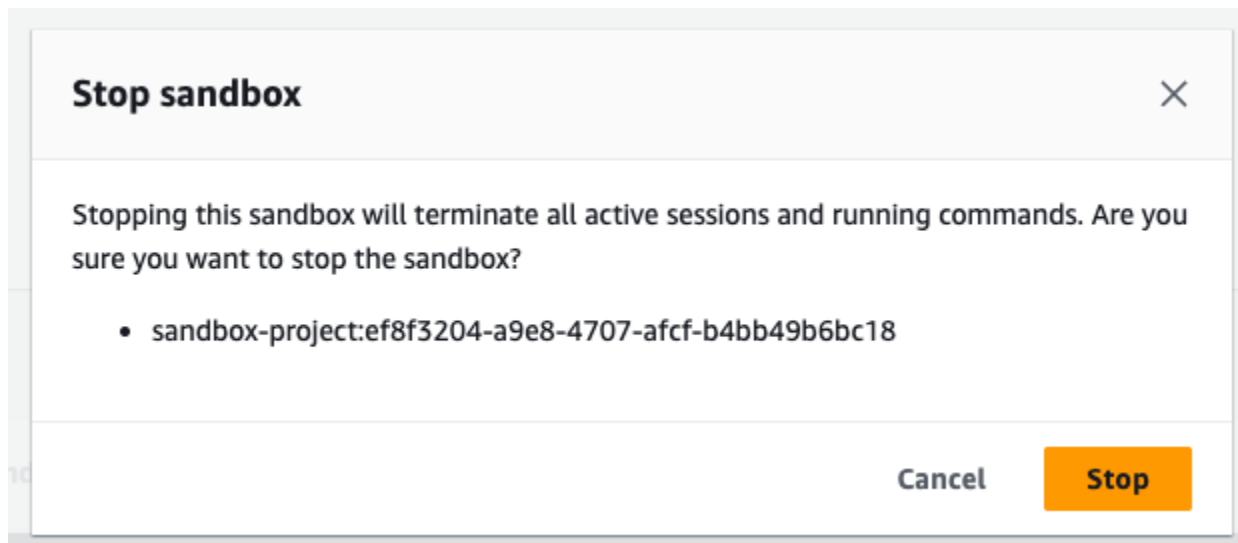


4. 然後，您的 CodeBuild 沙盒將初始化並開始執行您的自訂命令。輸出完成後，輸出會顯示在輸出索引標籤中。

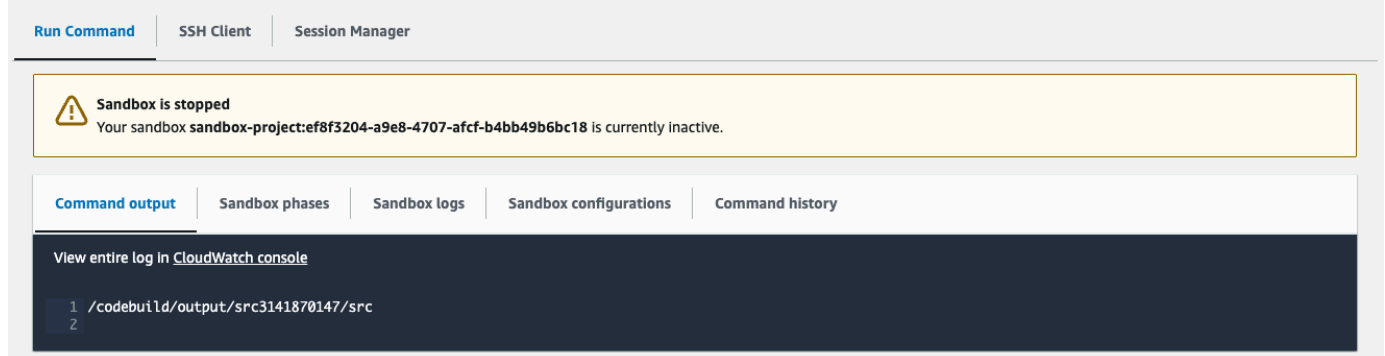
Debug build



5. 疑難排解完成時，您可以選擇停止沙盒來停止沙盒。然後選擇停止，確認您的沙盒將停止。

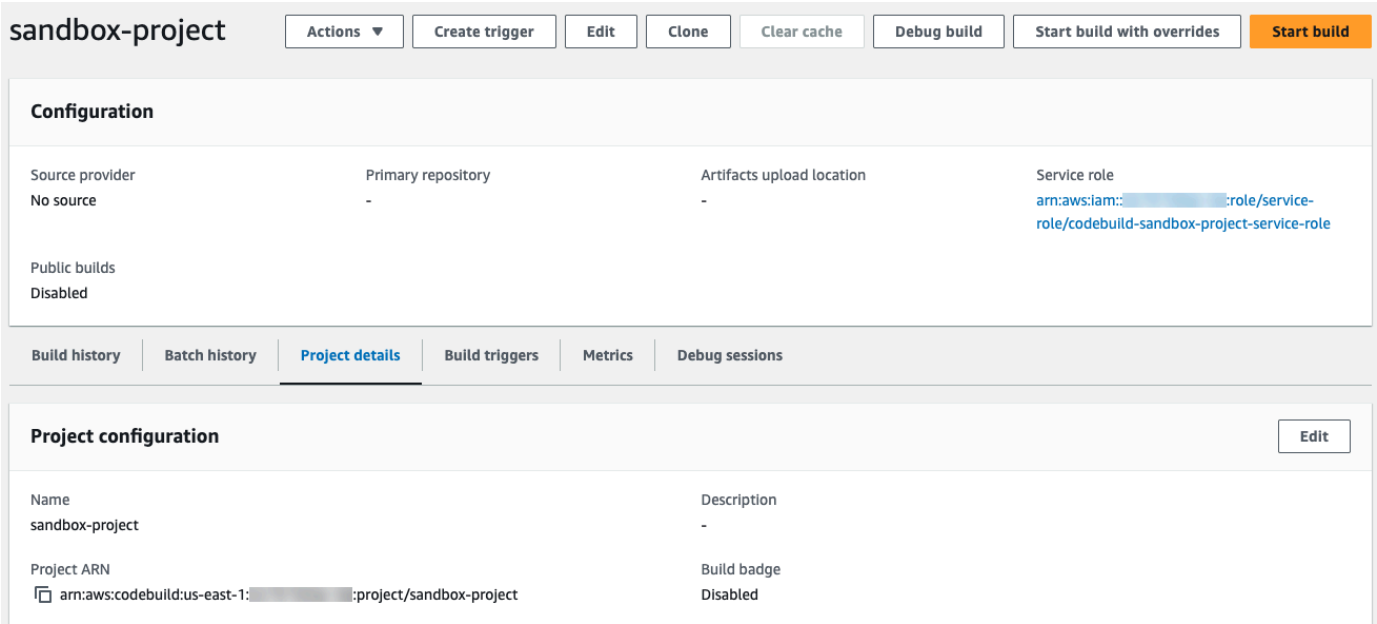


Debug build

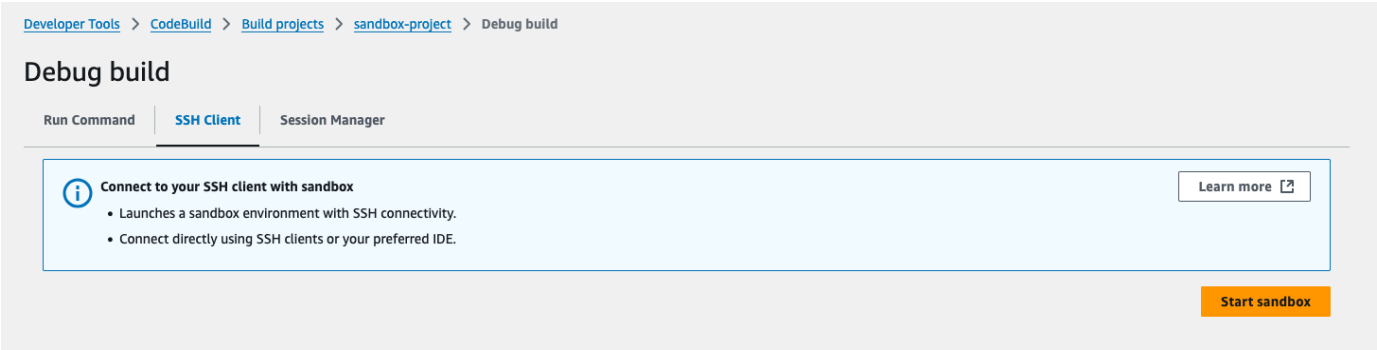


使用 CodeBuild 沙盒連線至 SSH 用戶端（主控台）

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。選擇建置專案，然後選擇偵錯建置。



3. 在 SSH 用戶端索引標籤中，選擇啟動沙盒。



4. CodeBuild 沙盒開始執行後，請依照主控台指示，將 SSH 用戶端連線至沙盒。

Debug build

Run Command

SSH Client

Session Manager

**Sandbox is running**Your sandbox `sandbox-project:80b80de0-6a4d-4e0c-9af2-45917603b1a8` is ready and available for use.

Stop sandbox

Terminal

Visual Studio Code

IntelliJ IDEA

Linux

macOS

Windows

If you haven't done so already, paste and execute the following command in macOS Terminal. For more information about using SSH, see [documentation page](#).

```
curl -O https://codefactory-us-east-1-prod-default-build-agent-executor.s3.us-east-1.amazonaws.com/mac-sandbox-ssh.sh
chmod +x mac-sandbox-ssh.sh
./mac-sandbox-ssh.sh
rm mac-sandbox-ssh.sh
```

Make sure your CLI user has the `codebuild:StartSandboxConnection` permission. For more information, see [AWS CLI authentication](#) documentation.

Connect to your sandbox environment with following command:

```
ssh codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:; :sandbox/sandbox-project:80b80de0-6a4d-4e0c-9af2-45917603b1a8
```

5. 疑難排解完成後，您可以選擇停止沙盒來停止沙盒。然後選擇停止，確認您的沙盒將停止。

Stop sandbox



Stopping this sandbox will terminate all active sessions and running commands. Are you sure you want to stop the sandbox?

- `sandbox-project:80b80de0-6a4d-4e0c-9af2-45917603b1a8`

Cancel

Stop

Debug build

Run Command

SSH Client

Session Manager

 **Sandbox is stopped**
Your sandbox `sandbox-project:80b80de0-6a4d-4e0c-9af2-45917603b1a8` is currently inactive.

Sandbox phases

Sandbox logs

Sandbox configurations

Name	Status	Context	Duration	Start time	End time
SUBMITTED	✔ Succeeded	-	<1 sec	Apr 8, 2025 1:33 PM (UTC-7:00)	Apr 8, 2025 1:33 PM (UTC-7:00)
QUEUED	✔ Succeeded	-	<1 sec	Apr 8, 2025 1:33 PM (UTC-7:00)	Apr 8, 2025 1:33 PM (UTC-7:00)
PROVISIONING	✔ Succeeded		5 secs	Apr 8, 2025 1:33 PM (UTC-7:00)	Apr 8, 2025 1:33 PM (UTC-7:00)
DOWNLOAD_SOURCE	✔ Succeeded		8 secs	Apr 8, 2025 1:33 PM (UTC-7:00)	Apr 8, 2025 1:33 PM (UTC-7:00)
RUN_SANDBOX	✔ Succeeded		213 secs	Apr 8, 2025 1:33 PM (UTC-7:00)	Apr 8, 2025 1:36 PM (UTC-7:00)
UPLOAD_ARTIFACTS	✔ Succeeded		<1 sec	Apr 8, 2025 1:36 PM (UTC-7:00)	Apr 8, 2025 1:36 PM (UTC-7:00)
FINALIZING	✔ Succeeded		<1 sec	Apr 8, 2025 1:36 PM (UTC-7:00)	Apr 8, 2025 1:36 PM (UTC-7:00)
COMPLETED	✔ Succeeded	-	-	Apr 8, 2025 1:36 PM (UTC-7:00)	-

使用 CodeBuild 沙盒對組建進行偵錯 (AWS CLI)

使用下列指示來執行命令，並將 SSH 用戶端連線至 CodeBuild 沙盒。

啟動 CodeBuild 沙盒 (AWS CLI)

CLI command

```
aws codebuild start-sandbox --project-name $PROJECT_NAME
```

- `--project-name` : CodeBuild 專案名稱

Sample request

```
aws codebuild start-sandbox --project-name "project-name"
```

Sample response

```
{
  "id": "project-name",
  "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
  "projectName": "project-name",
  "requestTime": "2025-02-06T11:24:15.560000-08:00",
}
```

```
"status": "QUEUED",
"source": {
  "type": "S3",
  "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
  "insecureSsl": false
},
"environment": {
  "type": "LINUX_CONTAINER",
  "image": "aws/codebuild/standard:6.0",
  "computeType": "BUILD_GENERAL1_SMALL",
  "environmentVariables": [{
    "name": "foo",
    "value": "bar",
    "type": "PLAINTEXT"
  },
  {
    "name": "bar",
    "value": "baz",
    "type": "PLAINTEXT"
  }
],
  "privilegedMode": false,
  "imagePullCredentialsType": "CODEBUILD"
},
"timeoutInMinutes": 10,
"queuedTimeoutInMinutes": 480,
"logConfig": {
  "cloudWatchLogs": {
    "status": "ENABLED",
    "groupName": "group",
    "streamName": "stream"
  },
  "s3Logs": {
    "status": "ENABLED",
    "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
    "encryptionDisabled": false
  }
},
"encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
"serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
"currentSession": {
  "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
  "currentPhase": "QUEUED",
```



```

    "status": "QUEUED",
    "startTime": "2025-02-06T11:24:15.626000-08:00",
    "logs": {
      "groupName": "group",
      "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
      "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
      "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz?region=us-west-2",
      "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
      "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
      "cloudWatchLogs": {
        "status": "ENABLED",
        "groupName": "group",
        "streamName": "stream"
      },
      "s3Logs": {
        "status": "ENABLED",
        "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
        "encryptionDisabled": false
      }
    }
  }
}

```

取得沙盒狀態的相關資訊 (AWS CLI)

CLI command

```
aws codebuild batch-get-sandboxes --ids $SANDBOX_IDS
```

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name"
```

- `--ids` : 以逗號分隔的 `sandboxIds` 或 清單 `sandboxArns`。

您可以提供沙盒 ID 或沙盒 ARN：

- 沙盒 ID：`<codebuild-project-name>:<UUID>`

例如 `project-name:d25be134-05cb-404a-85da-ac5f85d2d72c`。

- 沙盒 ARN：`arn:aws:codebuild:<region>:<account-id>:sandbox/<codebuild-project-name>:<UUID>`

例如 `arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name:d25be134-05cb-404a-85da-ac5f85d2d72c`。

Sample response

```
{
  "sandboxes": [{
    "id": "project-name",
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
    "requestTime": "2025-02-06T11:24:15.560000-08:00",
    "endTime": "2025-02-06T11:39:21.587000-08:00",
    "status": "STOPPED",
    "source": {
      "type": "S3",
      "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-sources/eb-sample-jetty-v4.zip",
      "insecureSsl": false
    },
    "environment": {
      "type": "LINUX_CONTAINER",
      "image": "aws/codebuild/standard:6.0",
      "computeType": "BUILD_GENERAL1_SMALL",
      "environmentVariables": [{
        "name": "foo",
        "value": "bar",
        "type": "PLAINTEXT"
      },
      {
        "name": "bar",
        "value": "baz",
        "type": "PLAINTEXT"
      }
    ]
  }
],
```

```

        "privilegedMode": false,
        "imagePullCredentialsType": "CODEBUILD"
    },
    "timeoutInMinutes": 10,
    "queuedTimeoutInMinutes": 480,
    "logConfig": {
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    },
    "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
    "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
    "currentSession": {
        "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "currentPhase": "COMPLETED",
        "status": "STOPPED",
        "startTime": "2025-02-06T11:24:15.626000-08:00",
        "endTime": "2025-02-06T11:39:21.600000-08:00",
        "phases": [{
            "phaseType": "SUBMITTED",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-06T11:24:15.577000-08:00",
            "endTime": "2025-02-06T11:24:15.606000-08:00",
            "durationInSeconds": 0
        },
        {
            "phaseType": "QUEUED",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-06T11:24:15.606000-08:00",
            "endTime": "2025-02-06T11:24:16.067000-08:00",
            "durationInSeconds": 0
        },
        {
            "phaseType": "PROVISIONING",
            "phaseStatus": "SUCCEEDED",

```

```

        "startTime": "2025-02-06T11:24:16.067000-08:00",
        "endTime": "2025-02-06T11:24:20.519000-08:00",
        "durationInSeconds": 4,
        "contexts": [{
            "statusCode": "",
            "message": ""
        }]
    },
    {
        "phaseType": "DOWNLOAD_SOURCE",
        "phaseStatus": "SUCCEEDED",
        "startTime": "2025-02-06T11:24:20.519000-08:00",
        "endTime": "2025-02-06T11:24:22.238000-08:00",
        "durationInSeconds": 1,
        "contexts": [{
            "statusCode": "",
            "message": ""
        }]
    },
    {
        "phaseType": "RUNNING_SANDBOX",
        "phaseStatus": "TIMED_OUT",
        "startTime": "2025-02-06T11:24:22.238000-08:00",
        "endTime": "2025-02-06T11:39:21.560000-08:00",
        "durationInSeconds": 899,
        "contexts": [{
            "statusCode": "BUILD_TIMED_OUT",
            "message": "Build has timed out. "
        }]
    },
    {
        "phaseType": "COMPLETED",
        "startTime": "2025-02-06T11:39:21.560000-08:00"
    }
],
"logs": {
    "groupName": "group",
    "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
    "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
    "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz?region=us-west-2",

```

```

        "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
            "encryptionDisabled": false
        }
    }
}],
"sandboxesNotFound": []
}

```

停止沙盒 (AWS CLI)

CLI command

```
aws codebuild stop-sandbox --id $SANDBOX-ID
```

- `--id` : sandboxId 或 sandboxArn。

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/
project-name"
```

Sample response

```

{
    "id": "project-name",
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
    "requestTime": "2025-02-06T11:24:15.560000-08:00",

```

```
"status": "STOPPING",
"source": {
  "type": "S3",
  "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
  "insecureSsl": false
},
"environment": {
  "type": "LINUX_CONTAINER",
  "image": "aws/codebuild/standard:6.0",
  "computeType": "BUILD_GENERAL1_SMALL",
  "environmentVariables": [{
    "name": "foo",
    "value": "bar",
    "type": "PLAINTEXT"
  },
  {
    "name": "bar",
    "value": "baz",
    "type": "PLAINTEXT"
  }
],
  "privilegedMode": false,
  "imagePullCredentialsType": "CODEBUILD"
},
"timeoutInMinutes": 10,
"queuedTimeoutInMinutes": 480,
"logConfig": {
  "cloudWatchLogs": {
    "status": "ENABLED",
    "groupName": "group",
    "streamName": "stream"
  },
  "s3Logs": {
    "status": "ENABLED",
    "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
    "encryptionDisabled": false
  }
},
"encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
"serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
"currentSession": {
  "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
  "currentPhase": "RUN_SANDBOX",
```

```
"status": "STOPPING",
"startTime": "2025-02-06T11:24:15.626000-08:00",
"phases": [{
  "phaseType": "SUBMITTED",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2025-02-08T14:33:26.144000-08:00",
  "endTime": "2025-02-08T14:33:26.173000-08:00",
  "durationInSeconds": 0
},
{
  "phaseType": "QUEUED",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2025-02-08T14:33:26.173000-08:00",
  "endTime": "2025-02-08T14:33:26.702000-08:00",
  "durationInSeconds": 0
},
{
  "phaseType": "PROVISIONING",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2025-02-08T14:33:26.702000-08:00",
  "endTime": "2025-02-08T14:33:30.530000-08:00",
  "durationInSeconds": 3,
  "contexts": [{
    "statusCode": "",
    "message": ""
  }]
},
{
  "phaseType": "DOWNLOAD_SOURCE",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2025-02-08T14:33:30.530000-08:00",
  "endTime": "2025-02-08T14:33:33.478000-08:00",
  "durationInSeconds": 2,
  "contexts": [{
    "statusCode": "",
    "message": ""
  }]
},
{
  "phaseType": "RUN_SANDBOX",
  "startTime": "2025-02-08T14:33:33.478000-08:00"
}
],
"logs": {
```

```

        "groupName": "group",
        "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
        "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
            "encryptionDisabled": false
        }
    }
}

```

啟動命令執行 (AWS CLI)

CLI command

```
aws codebuild start-command-execution --command $COMMAND --type $TYPE --sandbox-id
$SANDBOX-ID
```

- `--command` : 需要執行的命令。
- `--sandbox-id` : `sandboxId`或 `sandboxArn`。
- `--type` : 命令類型 SHELL。

Sample request

```
aws codebuild start-command-execution --command "echo \"Hello World\"" --type SHELL --
sandbox-id "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name"
```

Sample response

```
{
  "id": "e1c658c2-02bb-42a8-9abb-94835241fcd6",
  "sandboxId": "f7126a4a-b0d5-452f-814c-fea73718f805",
  "submitTime": "2025-02-06T20:12:02.683000-08:00",
  "status": "SUBMITTED",
  "command": "echo \"Hello World\"",
  "type": "SHELL",
  "logs": {
    "groupName": "group",
    "streamName": "stream",
    "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
    "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/codefactory-test-
pool-1-us-west-2-beta-default-build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz?
region=us-west-2",
    "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
    "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-default-
build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz",
    "cloudWatchLogs": {
      "status": "ENABLED",
      "groupName": "group",
      "streamName": "stream"
    },
    "s3Logs": {
      "status": "ENABLED",
      "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
      "encryptionDisabled": false
    }
  }
}
```

取得命令執行的相關資訊 (AWS CLI)

CLI command

```
aws codebuild batch-get-command-executions --command-execution-ids $COMMAND-IDS --
sandbox-id $SANDBOX-IDS
```

- `--command-execution-ids` : 逗號分隔的 清單 `commandExecutionIds`。
- `--sandbox-id` : `sandboxId` 或 `sandboxArn`。

Sample request

```
aws codebuild batch-get-command-executions --command-execution-
ids "c3c085ed-5a8f-4531-8e95-87d547f27ffd" --sandbox-id "arn:aws:codebuild:us-
west-2:962803963624:sandbox/project-name"
```

Sample response

```
{
  "commandExecutions": [{
    "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
    "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
    "submitTime": "2025-02-10T20:18:17.118000-08:00",
    "startTime": "2025-02-10T20:18:17.939000-08:00",
    "endTime": "2025-02-10T20:18:17.976000-08:00",
    "status": "SUCCEEDED",
    "command": "echo \"Hello World\"",
    "type": "SHELL",
    "exitCode": "0",
    "standardOutputContent": "Hello World\n",
    "logs": {
      "groupName": "group",
      "streamName": "stream",
      "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
      "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
      "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
```

```
    "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-  
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",  
    "cloudWatchLogs": {  
        "status": "ENABLED",  
        "groupName": "group",  
        "streamName": "stream"  
    },  
    "s3Logs": {  
        "status": "ENABLED",  
        "location": "codefactory-test-pool-1-us-west-2-beta-default-build-  
logs",  
        "encryptionDisabled": false  
    }  
  },  
  "commandExecutionsNotFound": []  
}
```

列出沙盒的命令執行 (AWS CLI)

CLI command

```
aws codebuild list-command-executions-for-sandbox --sandbox-id $SANDBOX-ID --next-  
token $NEXT_TOKEN --max-results $MAX_RESULTS --sort-order $SORT_ORDER
```

- `--next-token` : 下一個字符，如果有的話，以取得分頁結果。您將從先前執行的清單沙盒取得此值。
- `--max-results` : (選用) 要擷取的沙盒記錄數目上限。
- `--sort-order` : 應擷取沙盒記錄的順序。

Sample request

```
aws codebuild list-command-executions-for-sandbox --sandbox-id  
"arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name"
```

Sample response

```
{  
  "commandExecutions": [{  
    "id": "aad6687e-07bc-45ab-a1fd-f5440229b528",
```

```

    "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
    "submitTime": "2025-02-10T20:18:35.304000-08:00",
    "startTime": "2025-02-10T20:18:35.615000-08:00",
    "endTime": "2025-02-10T20:18:35.651000-08:00",
    "status": "FAILED",
    "command": "fail command",
    "type": "SHELL",
    "exitCode": "127",
    "standardErrContent": "/codebuild/output/tmp/script.sh: 4: fail: not
found\n",
    "logs": {
        "groupName": "group",
        "streamName": "stream",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
        "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
        "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
        "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
            "encryptionDisabled": false
        }
    },
    {
        "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
        "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
        "submitTime": "2025-02-10T20:18:17.118000-08:00",
        "startTime": "2025-02-10T20:18:17.939000-08:00",
        "endTime": "2025-02-10T20:18:17.976000-08:00",
        "status": "SUCCEEDED",
        "command": "echo \"Hello World\"",
        "type": "SHELL",

```

```

        "exitCode": "0",
        "standardOutputContent": "Hello World\n",
        "logs": {
            "groupName": "group",
            "streamName": "stream",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                "encryptionDisabled": false
            }
        }
    }
]
}

```

列出沙盒 (AWS CLI)

CLI command

```
aws codebuild list-sandboxes --next-token $NEXT_TOKEN --max-results $MAX_RESULTS --
sort-order $SORT_ORDER
```

Sample request

```
aws codebuild list-sandboxes
```

Sample response

```
{
  "ids": [
    "s3-log-project-integ-test-temp173925062814985d64e0f-7880-41df-9a3c-fb6597a266d2:827a5243-0841-4b69-a720-4438796f6967",
    "s3-log-project-integ-test-temp1739249999716bbd438dd-8bb8-47bd-ba6b-0133ac65b3d3:e2fa4eab-73af-42e3-8903-92fddaf9f378",
    "s3-log-project-integ-test-temp17392474779450fbdacc2-2d6e-4190-9ad5-28f891bb7415:cd71e456-2a4c-4db4-ada5-da892b0bba05",
    "s3-log-project-integ-test-temp17392246284164301421c-5030-4fa1-b4d3-ca15e44771c5:9e26ab3f-65e4-4896-a19c-56b1a95e630a",
    "s3-log-project-integ-test-temp173921367319497056d8d-6d8e-4f5a-a37c-a62f5686731f:22d91b06-df1e-4e9c-a664-c0abb8d5920b",
    "s3-log-project-integ-test-temp1739213439503f6283f19-390c-4dc8-95a9-c8480113384a:82cc413e-fc46-47ab-898f-ae23c83a613f",
    "s3-log-project-integ-test-temp1739054385570b1f1ddc2-0a23-4062-bd0c-24e9e4a99b99:c02562f3-2396-42ec-98da-38e3fe5da13a",
    "s3-log-project-integ-test-temp173905400540237dab1ac-1fde-4dfb-a8f5-c0114333dc89:d2f30493-f65e-4fa0-a7b6-08a5e77497b9",
    "s3-log-project-integ-test-temp17390534055719c534090-7bc4-48f1-92c5-34acaec5bf1e:df5f1c8a-f017-43b7-91ba-ad2619e2c059",
    "s3-log-project-integ-test-temp1739052719086a61813cc-ebb9-4db4-9391-7f43cc984ee4:d61917ec-8037-4647-8d52-060349272c4a",
    "s3-log-project-integ-test-temp173898670094078b67edb-c42f-42ed-9db2-4b5c1a5fc66a:ce33dfbc-beeb-4466-8c99-a3734a0392c7",
    "s3-log-project-integ-test-temp17389863425584d21b7cd-32e2-4f11-9175-72c89ecaffef:046dadf0-1f3a-4d51-a2c0-e88361924acf",
    "s3-log-project-integ-test-temp1738985884273977ccd23-394b-46cc-90d3-7ab94cf764dc:0370dc41-9339-4b0a-91ed-51929761b244",
    "s3-log-project-integ-test-temp1738985365972241b614f-8e41-4387-bd25-2b8351fbc9e0:076c392a-9630-47d8-85a9-116aa34edfff",
    "s3-log-project-integ-test-temp1738985043988a51a9e2b-09d6-4d24-9c3c-1e6e21ac9fa8:6ea3949c-435b-4177-aa4d-614d5956244c",
    "s3-log-project-integ-test-temp1738984123354c68b31ad-49d1-4f4b-981d-b66c00565fff:6c3fff6c-815b-48b5-ada3-737400a6dee8",
    "s3-log-project-integ-test-temp1738977263715d4d5bf6c-370a-48bf-8ea6-905358a6cf92:968a0f54-724a-42d1-9207-6ed854b2fae8",
  ]
}
```

```
"s3-log-project-integ-test-
temp173897358796816ce8d7d-2a5e-41ef-855b-4a94a8d2795d:80f9a7ce-930a-402e-934e-
d8b511d68b04",
"s3-log-project-integ-test-temp17389730633301af5e452-0966-467c-
b684-4e36d47f568c: cabbe989-2e8a-473c-af25-32edc8c28646",
"s3-log-project-integ-test-temp1738901503813173fd468-
b723-4d7b-9f9f-82e88d17f264:f7126a4a-b0d5-452f-814c-fea73718f805",
"s3-log-project-integ-test-temp1738890502472c13616fb-
bd0f-4253-86cc-28b74c97a0ba:c6f197e5-3a53-45b6-863e-0e6353375437",
"s3-log-project-integ-test-
temp17388903044683610daf3-8da7-43c6-8580-9978432432ce:d20aa317-8838-4966-
bbfc-85b908213df1",
"s3-log-project-integ-test-temp173888857196780b5ab8b-e54b-44fd-a222-
c5a374fffe96:ab4b9970-ffae-47a0-b3a8-7b6790008cad",
"s3-log-project-integ-test-temp1738888336931c11d378d-e74d-49a4-
a723-3b92e6f7daac:4922f0e8-9b7d-4119-9c9f-115cd85e703e",
"s3-log-project-integ-test-temp17388881717651612a397-c23f-4d88-
ba87-2773cd3fc0c9:be91c3fc-418e-4feb-8a3a-ba58ff8f4e8a",
"s3-log-project-integ-test-
temp17388879727174c3c62ed-6195-4afb-8a03-59674d0e1187:a48826a8-3c0d-43c5-
a1b5-1c98a0f978e9",
"s3-log-project-integ-test-temp1738885948597cef305e4-b8b4-46b0-a65b-
e2d0a7b83294:c050e77d-e3f8-4829-9a60-46149628fe96",
"s3-log-project-integ-test-temp173888561463001a7d2a8-
e4e4-4434-94db-09d3da9a9e17:8c3ac3f5-7111-4297-aec9-2470d3ead873",
"s3-log-project-integ-test-
temp1738869855076eb19cafd-04fe-41bd-8aa0-40826d0c0d27:d25be134-05cb-404a-85da-
ac5f85d2d72c",
"s3-project-integ-test-temp1738868157467148eacfc-d39b-49fc-a137-
e55381cd2978:4909557b-c221-4814-b4b6-7d9e93d37c35",
"s3-project-integ-test-temp1738820926895abec0af2-
e33d-473c-9cf4-2122dd9d6876:8f5cf218-71d6-40a4-a4be-6cacebd7765f",
"s3-project-integ-test-temp173881998877574f969a6-1c2e-4441-b463-
ab175b45ce32:04396851-c901-4986-9117-585528e3877f",
"s3-project-integ-test-temp17388189812309abd2604-29ba-4cf6-
b6bf-073207b7db9c:540075c7-f5ec-41e8-9341-2233c09247eb",
"s3-project-integ-test-temp1738818843474d3ea9ac1-b609-461b-
bbdb-2da245c9bc96:865d4c3c-fbfe-4ece-9c92-d0c928341404",
"s3-project-integ-test-temp1738818542236006e9169-e6d9-4344-9b59-
f557e7aec619:1f9ffa87-da15-4290-83e2-eebdd877497b",
"s3-project-integ-test-
temp173881809557486ad11fd-7931-48d7-81d5-499cea52a6bc:c4c2efc4-685f-4e13-8b0f-1ef85ec300b1",
"s3-project-integ-test-temp173881794103322941020-3f0b-49c3-b836-
fcd818ec9484:0344cfba-de48-456d-b2a8-6566bd4a5d6e",
```

```
"s3-project-integ-test-temp1738817680747b93d0d0b-ea16-497f-9559-af25ee6dcfdf:654a3a55-d92a-4dc6-8da8-56fd4d40d7e1",
"s3-project-integ-test-temp17388174027191255c3da-086c-4270-b047-acac0b7bee0d:b7e82740-2c69-42fc-ab5a-dbf15bc016a1",
"s3-project-integ-test-temp1738817099799016e7fa3-b9b5-46a2-bcd5-0888c646743f:8705a6a4-79ff-427a-a1c3-85c4e8fe462e",
"s3-project-integ-test-temp1738816479281bb0c3606-5ebf-4623-bed5-12b60e9d3512:f23fc74b-a981-4835-8e28-375fcd4c99e4",
"s3-project-integ-test-temp1738816263585c939a133-4d37-482c-9238-1dbff34b7674:ca28e234-0045-4ae6-8732-938b17597f50",
"s3-project-integ-test-temp173881580873072d18733-8fe4-43b1-83f7-95f25bb27ccf:c6f0f55b-5736-47c7-a3aa-1b8461a6d5ed"
]
```

教學課程：使用 SSH 連線至沙盒

本教學課程說明如何使用 SSH 用戶端連線至 CodeBuild 沙盒。

若要完成此教學課程，您必須先：

- 確保您有現有的 AWS CodeBuild 專案。
- 為您的 CodeBuild 專案角色設定適當的 IAM 許可。
- AWS CLI 在本機電腦上安裝和設定。

步驟 1：啟動沙盒

在主控台中啟動 CodeBuild 沙盒

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。選擇建置專案，然後選擇偵錯建置。

Developer Tools > CodeBuild > Build projects > sandbox-project

sandbox-project

Actions ▼ Create trigger Edit Clone Debug build Start build with overrides **Start build**

Configuration

Source provider	Primary repository	Artifacts upload location	Service role
No source	-	-	arn:aws:iam::012345678910:role/service-role/codebuild-sandbox-project-service-role
Public builds			
Disabled			

Build history Batch history **Project details** Build triggers Metrics Debug sessions

Project configuration Edit

Name	Description
sandbox-project	-
Project ARN	Build badge
arn:aws:codebuild:us-east-1:012345678910:project/sandbox-project	Disabled

3. 在 SSH 用戶端索引標籤中，選擇啟動沙盒。

Developer Tools > CodeBuild > Build projects > sandbox-project > Debug build

Debug build

Run Command **SSH Client** Session Manager

Connect to your SSH client with sandbox

- Launches a sandbox environment with SSH connectivity.
- Connect directly using SSH clients or your preferred IDE.

Learn more

Start sandbox

4. 沙盒初始化程序可能需要一些時間。當沙盒的狀態變更為 時，您可以連線到沙盒RUN_SANDDBOX。

Developer Tools > CodeBuild > Build projects > sandbox-project > Debug build

Debug build

Run Command | **SSH Client** | Session Manager


Sandbox is running
 Your sandbox `sandbox-project:253616fd-9624-434e-bb9a-bbe52620d256` is ready and available for use.
 Stop sandbox

Terminal | Visual Studio Code | IntelliJ IDEA

Linux | **macOS** | Windows

If you haven't done so already, paste and execute the following command in macOS Terminal. For more information about using SSH, see [documentation page](#).

```
curl -O https://codefactory-us-east-1-prod-default-build-agent-executor.s3.us-east-1.amazonaws.com/mac-sandbox-ssh.sh
chmod +x mac-sandbox-ssh.sh
./mac-sandbox-ssh.sh
rm mac-sandbox-ssh.sh
```

Make sure your CLI user has the `codebuild:StartSandboxConnection` permission. For more information, see [AWS CLI authentication](#) documentation.

Connect to your sandbox environment with following command:

```
ssh codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:012345678910:sandbox/sandbox-project:253616fd-9624-434e-bb9a-bbe52620d256
```

Sandbox phases		Sandbox logs		Sandbox configurations	
Name	Status	Context	Duration	Start time	End time
SUBMITTED	✔ Succeeded	-	<1 sec	Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 PM (UTC-7:00)
QUEUED	✔ Succeeded	-	<1 sec	Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 PM (UTC-7:00)
PROVISIONING	✔ Succeeded	-	4 secs	Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 PM (UTC-7:00)
DOWNLOAD_SOURCE	✔ Succeeded	-	6 secs	Apr 1, 2025 4:33 PM (UTC-7:00)	Apr 1, 2025 4:33 PM (UTC-7:00)
RUN_SANDBOX	-	-	-	Apr 1, 2025 4:33 PM (UTC-7:00)	-

步驟 2：修改本機 SSH 組態

如果您是第一次連線到沙盒，則需要使用下列步驟執行一次性設定程序：

在主控台中修改本機 SSH 組態

1. 找到您作業系統的設定命令。
2. 開啟您的本機終端機，然後複製並執行提供的命令，以下載並執行指令碼來設定本機 SSH 組態。例如，如果您的作業系統是 macOS，請使用下列命令：

Linux | **macOS** | Windows

If you haven't done so already, paste and execute the following command in macOS Terminal. For more information about using SSH, see [documentation page](#).

```
curl -O https://codefactory-us-east-1-prod-default-build-agent-executor.s3.us-east-1.amazonaws.com/mac-sandbox-ssh.sh
chmod +x mac-sandbox-ssh.sh
./mac-sandbox-ssh.sh
rm mac-sandbox-ssh.sh
```

3. 組態指令碼會新增連線至沙盒所需的組態。系統會提示您接受這些變更。

- 成功組態後，將建立新的 CodeBuild 沙盒 SSH 組態項目。

```
Host codebuild-sandbox-ssh*
  StrictHostKeyChecking no
  LogLevel INFO
  ForwardAgent yes
  ControlMaster auto
  ControlPersist 10m
  ProxyCommand sh -c "/Users/[redacted]/.aws/codebuild-dev-env/codebuild-sandbox-connect.sh %n"
```

步驟 3：連線至沙盒

在主控台中修改本機 SSH 組態

- 設定 AWS CLI 身分驗證，並確保您的 AWS CLI 使用者具有 `codebuild:StartSandboxConnection` 許可。如需詳細資訊，請參閱《第 AWS 1 版命令列界面使用者指南》中的[使用的 IAM 使用者憑證進行身分驗證 AWS CLI](#)。
- 使用下列命令連線至沙盒：

```
ssh codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:<account-id>:sandbox/<sandbox-id>
```

Note

若要對連線失敗進行故障診斷，請使用 `-v` 旗標來啟用詳細輸出。例如 `ssh -v codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:<account-id>:sandbox/<sandbox-id>`。

如需其他故障診斷指引，請參閱 [AWS CodeBuild 對沙盒 SSH 連線問題進行故障診斷](#)。

步驟 4：檢閱您的結果

連線後，您就可以偵錯建置失敗、測試建置命令、試驗組態變更，以及使用沙盒驗證環境變數和相依性。

AWS CodeBuild 對沙盒 SSH 連線問題進行故障診斷

使用本主題中的資訊來協助您識別、診斷和解決 CodeBuild 沙盒 SSH 連線問題。

主題

- [StartSandboxConnection InvalidInputException SSH 進入 CodeBuild 沙盒環境時發生錯誤](#)

- [錯誤：當 SSH 進入 CodeBuild 沙盒環境時，「找不到登入資料」](#)
- [StartSandboxConnection AccessDeniedException SSH 進入 CodeBuild 沙盒環境時發生錯誤](#)
- [錯誤：當 SSH 進入 CodeBuild 沙盒環境時，「ssh：無法解析主機名稱」](#)

StartSandboxConnection InvalidInputException SSH 進入 CodeBuild 沙盒環境時發生錯誤

問題：嘗試使用命令連線到 CodeBuild 沙盒環境時 `ssh codebuild-sandbox-ssh=<sandbox-arn>`，您可能會遇到 `InvalidInputException` 錯誤，例如：

```
An error occurred (InvalidInputException) when calling the StartSandboxConnection operation: Failed to start SSM session for {sandbox-arn}
User: arn:aws:sts::<account-ID>:assumed-role/<service-role-name>/AWSCodeBuild-<UUID>
is not authorized to perform: ssm:StartSession on resource.
```

```
An error occurred (InvalidInputException) when calling the StartSandboxConnection operation: Failed to start SSM session for
sandbox <sandbox-arn>: codebuild:<UUID> is not connected.
```

可能原因：

- 缺少 Amazon EC2 Systems Manager 代理程式：建置映像未正確安裝或設定 SSM 代理程式。
- 許可不足：CodeBuild 專案服務角色缺少必要的 SSM 許可。

建議的解決方案：如果您使用建置的自訂映像，請執行下列動作。

1. 安裝 SSM Agent。如需詳細資訊，請參閱《》中的在 [Linux 的 Amazon EC2 執行個體上手動安裝和解除安裝 SSM 代理](#) 程式。SSM Agent 版本必須為 3.0.1295.0 或更新版本。
2. 將檔案 <https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/7.0/amazon-ssm-agent.json> 複製到映像中的 `/etc/amazon/ssm/` 目錄。這會在 SSM 代理程式中啟用容器模式。
3. 確保您的 CodeBuild 專案的服務角色具有下列許可，然後重新啟動沙盒環境：

```
{
  "Effect": "Allow",
  "Action": [
    "ssmmessages:CreateControlChannel",
    "ssmmessages:CreateDataChannel",
    "ssmmessages:OpenControlChannel",
```

```
        "ssmmessages:OpenDataChannel"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "ssm:StartSession"
    ],
    "Resource": [
        "arn:aws:codebuild:region:account-id:build/*",
        "arn:aws:ssm:region::document/AWS-StartSSHSession"
    ]
}
```

錯誤：當 SSH 進入 CodeBuild 沙盒環境時，「找不到登入資料」

問題：嘗試使用命令 連線至 CodeBuild 沙盒環境時 `ssh codebuild-sandbox-ssh=<sandbox-arn>`，您可能會遇到下列登入資料錯誤：

```
Unable to locate credentials. You can configure credentials by running
"aws configure".
```

可能的原因：AWS 憑證尚未在您的本機環境中正確設定。

建議的解決方案：遵循 第 AWS 2 版命令列界面使用者指南中的官方文件：[設定 的設定 AWS CLI](#)，以設定您的 AWS CLI 登入資料。

StartSandboxConnection AccessDeniedException SSH 進入 CodeBuild 沙盒環境時發生錯誤

問題：嘗試使用命令 連線至 CodeBuild 沙盒環境時 `ssh codebuild-sandbox-ssh=<sandbox-arn>`，您可能會遇到下列許可錯誤：

```
An error occurred (AccessDeniedException) when calling the StartSandboxConnection
operation:
User: arn:aws:sts::account-id:assumed-role/role-name
is not authorized to perform: codebuild:StartSandboxConnection on resource:
sandbox-arn
because no identity-based policy allows the codebuild:StartSandboxConnection action
```

可能原因：您的 AWS 登入資料缺少執行此操作所需的 CodeBuild 許可。

建議的解決方案：確保與您的 AWS CLI 登入資料相關聯的 IAM 使用者或角色具有下列許可：

```
{
  "Effect": "Allow",
  "Action": [
    "codebuild:StartSandboxConnection"
  ],
  "Resource": [
    "arn:aws:codebuild:region:account-id:sandbox/*"
  ]
}
```

錯誤：當 SSH 進入 CodeBuild 沙盒環境時，「ssh：無法解析主機名稱」

問題：嘗試使用命令 連線至 CodeBuild 沙盒環境時 `ssh codebuild-sandbox-ssh=<sandbox-arn>`，您遇到下列主機名稱解析錯誤：

```
ssh: Could not resolve hostname
```

可能原因：當在您的本機環境中未正確執行必要的 CodeBuild 沙盒連線指令碼時，通常會發生此錯誤。

建議解決方案：

1. 下載 CodeBuild 沙盒連線指令碼。
2. 在終端機中執行指令碼，以建立必要的 SSH 組態。
3. 重試 SSH 連線至沙盒環境。

使用 Session Manager 偵錯組建

在 中 AWS CodeBuild，您可以暫停執行中的組建，然後使用 AWS Systems Manager Session Manager 連線到組建容器，並檢視容器的狀態。

Note

此功能不適用於 Windows 環境。

主題

- [先決條件](#)
- [暫停建置](#)
- [啟動建置](#)
- [連線至建置容器](#)
- [繼續建置](#)

先決條件

若要允許 Session Manager 與建置工作階段搭配使用，您必須啟用建置的工作階段連線。有兩個先決條件：

- CodeBuild Linux 標準精選映像已安裝 SSM 代理程式並啟用 SSM 代理程式 ContainerMode。

如果您使用建置的自訂映像，請執行下列動作：

1. 安裝 SSM Agent。如需詳細資訊，請參閱 AWS Systems Manager 《使用者指南》中的在 [Linux 的 EC2 執行個體上手動安裝 SSM 代理](#) 程式。SSM Agent 版本必須為 3.0.1295.0 或更新版本。
2. 將檔案 <https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/5.0/amazon-ssm-agent.json> 複製到映像中的 /etc/amazon/ssm/ 目錄。這會在 SSM 代理程式中啟用容器模式。

Note

自訂映像需要最新 SSM 代理程式，此功能才能如預期運作。

- CodeBuild 服務角色必須具有下列 SSM 政策：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssmmessages:CreateControlChannel",
        "ssmmessages:CreateDataChannel",
```

```

        "ssmmessages:OpenControlChannel",
        "ssmmessages:OpenDataChannel"
    ],
    "Resource": "*"
}
]
}

```

您可以在啟動建置時，讓 CodeBuild 主控台自動將此政策連接至您的服務角色。或者，您可以手動將此政策連接至您的服務角色。

- 如果您在 Systems Manager 偏好設定中已啟用稽核和記錄工作階段活動，則 CodeBuild 服務角色也必須具有其他許可。許可不同，取決於日誌的存放位置。

CloudWatch Logs

如果使用 CloudWatch Logs 存放日誌，請將下列許可新增至 CodeBuild 服務角色：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:DescribeLogGroups",
      "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:MyLogGroup:*"
    }
  ]
}

```

Amazon S3

如果使用 Amazon S3 存放日誌，請將下列許可新增至 CodeBuild 服務角色：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetEncryptionConfiguration",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket-name>",
        "arn:aws:s3:::<bucket-name>/*"
      ]
    }
  ]
}
```

如需詳細資訊，請參閱AWS Systems Manager 《使用者指南》中的[稽核和記錄工作階段活動](#)。

暫停建置

若要暫停建置，請在 buildspec 檔案中的任何建置階段插入 codebuild-breakpoint 命令。建置將在此時暫停，這可讓您連線至建置容器，並檢視其目前狀態的容器。

例如，將下列項目新增至 buildspec 檔案中的建置階段。

```
phases:
  pre_build:
    commands:
      - echo Entered the pre_build phase...
      - echo "Hello World" > /tmp/hello-world
      - codebuild-breakpoint
```

此程式碼會建立 /tmp/hello-world 檔案，然後在此時暫停建置。

啟動建置

若要允許 Session Manager 與建置工作階段搭配使用，您必須啟用建置的工作階段連線。若要這樣做，在開始建置時，請依照下列步驟執行：

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。選擇建置專案，然後選擇使用覆寫開始建置。
3. 選擇 Advanced build overrides (進階組建覆寫)。
4. 在環境區段中，選擇啟用工作階段連線選項。如果未選取此選項，則會忽略所有 codebuild-breakpoint 和 codebuild-resume 命令。
5. 進行任何其他所需的變更，然後選擇開始建置。
6. 在主控台中監控建置狀態。當工作階段可用時，AWS 工作階段管理員連結會出現在建置狀態區段中。

連線至建置容器

您可以透過下列兩種方式之一連線到建置容器：

CodeBuild 主控台

在 Web 瀏覽器中，開啟 AWS Session Manager 連結以連線至建置容器。終端機工作階段隨即開啟，可讓您瀏覽和控制建置容器。

AWS CLI

Note

您的本機電腦必須安裝此程序的 Session Manager 外掛程式。如需詳細資訊，請參閱 AWS Systems Manager 《使用者指南》中的 [安裝 CLI AWS 的 Session Manager 外掛程式](#)。

1. 使用組建 ID 呼叫 batch-get-builds api 以取得組建的相關資訊，包括工作階段目標識別符。工作階段目標識別符屬性名稱會根據aws命令的輸出類型而有所不同。這就是--output json將 新增至命令的原因。

```
aws codebuild batch-get-builds --ids <buildID> --region <region> --output json
```

2. 複製 sessionTarget 屬性值。sessionTarget 屬性名稱可能會因aws命令的輸出類型而有所不同。這就是為什麼 --output json 會新增至上一個步驟中的 命令。
3. 使用下列命令來連線至建置容器。

```
aws ssm start-session --target <sessionTarget> --region <region>
```

在此範例中，請確認/tmp/hello-world檔案存在且包含文字 Hello World。

繼續建置

完成檢查建置容器後，請從容器 Shell 發出 codebuild-resume 命令。

```
$ codebuild-resume
```

在 中刪除組建 AWS CodeBuild

您可以使用 AWS CLI 或 AWS SDKs 來刪除 中的組建 AWS CodeBuild。

主題

- [刪除建置 \(AWS CLI\)](#)
- [刪除組建 \(AWS SDKs\)](#)

刪除建置 (AWS CLI)

執行 batch-delete-builds 命令：

```
aws codebuild batch-delete-builds --ids ids
```

在上述命令中，取代下列預留位置：

- *ids*：必要的字串。要刪除的組建 ID。若要指定多個組建，請以空格將每個組建 ID 分開。若要取得建置 ID 的清單，請參閱下列主題：
 - [檢視建置 ID 清單 \(AWS CLI\)](#)
 - [檢視建置專案的建置 ID 清單 \(AWS CLI\)](#)

如果成功，輸出中會出現 buildsDeleted 陣列，其中包含成功刪除之每個組建的 Amazon Resource Name (ARN)。未成功刪除之組建的相關資訊會出現在輸出中的 buildsNotDeleted 陣列內。

例如，如果您執行此命令：

```
aws codebuild batch-delete-builds --ids my-demo-build-project:f8b888d2-5e1e-4032-8645-b115195648EX my-other-demo-build-project:a18bc6ee-e499-4887-b36a-8c90349c7eEX
```

類似下列內容的資訊會顯示在輸出中：

```
{
  "buildsNotDeleted": [
    {
      "id": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-build-
project:f8b888d2-5e1e-4032-8645-b115195648EX",
      "statusCode": "BUILD_IN_PROGRESS"
    }
  ],
  "buildsDeleted": [
    "arn:aws:codebuild:us-west-2:123456789012:build/my-other-demo-build-
project:a18bc6ee-e499-4887-b36a-8c90349c7eEX"
  ]
}
```

刪除組建 (AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用 的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

在 中手動重試建置 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台 AWS CLI或 AWS SDKs手動重試單一建置或批次建置 AWS CodeBuild。

主題

- [手動重試組建 \(主控台 \)](#)
- [手動重試組建 \(AWS CLI\)](#)
- [手動重試組建 \(AWS SDKs\)](#)

手動重試組建 (主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 執行以下任意一項：
 - 如果顯示 **build-project-name : build-ID** 頁面，請選擇重試建置。
 - 在導覽窗格中，選擇 Build history (組建歷史記錄)。在組建清單中，選取組建的方塊，然後選擇重試組建。

- 在導覽窗格中，選擇 Build projects (建置專案)。在組建專案清單中的 Name (名稱) 欄中，選擇組建專案名稱的連結。在組建清單中，選取組建的方塊，然後選擇重試組建。

Note

根據預設，只會顯示最新 100 個組建或組建專案。若要檢視更多組建或組建專案，請選擇齒輪圖示，然後針對 Builds per page (每頁顯示組建數) 或 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

手動重試組建 (AWS CLI)

- 執行 retry-build 命令：

```
aws codebuild retry-build --id <build-id> --idempotency-token <idempotencyToken>
```

在上述命令中，取代下列預留位置：

- <build-id>**：必要字串。要重試的建置或批次建置 ID。若要取得建置 ID 的清單，請參閱下列主題：
 - [檢視建置 ID 清單 \(AWS CLI\)](#)
 - [檢視批次建置 IDs 的清單 \(AWS CLI\)](#)
 - [檢視建置專案的建置 ID 清單 \(AWS CLI\)](#)
 - [檢視組建專案的批次組建 IDs 清單 \(AWS CLI\)](#)
- idempotency-token**：選用。如果您使用 選項執行 retry-build 命令，retry-build 則請求中會包含唯一的區分大小寫識別符或字符。此字符在 請求之後的 5 分鐘內有效。如果您使用相同的字符重複 retry-build 請求，但變更參數，CodeBuild 會傳回參數不相符錯誤。

手動重試組建 (AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs 和工具參考](#)。

在中自動重試建置 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台、AWS CLI 或 AWS SDKs 自動重試建置 AWS CodeBuild。啟用自動重試後，CodeBuild 會在失敗的建置達到指定的限制之後 `RetryBuild`，使用專案的服務角色自動呼叫。例如，如果自動重試限制設為兩個，CodeBuild 將呼叫 `RetryBuild` API 以自動重試您的組建最多兩次。

 Note

CodeBuild 不支援 CodePipeline 的自動重試。

主題

- 自動重試建置 (主控台)
- 自動重試建置 (AWS CLI)
- 自動重試組建 (AWS SDKs)

自動重試建置（主控台）

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 選擇建立專案。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 及 [執行建置 \(主控台\)](#)。
 - 在 Environment (環境) 中：
 - 針對自動重試限制，輸入建置失敗後所需的自動重試次數上限。
3. 在環境中，選擇其他組態。
4. 繼續執行預設值，然後選擇建立建置專案。

自動重試建置 (AWS CLI)

- 執行 create-project 命令：

```
aws codebuild create-project \
  --name "<project-name>" \
  --auto-retry-limit <auto-retry-limit> \
```

```
--source "<source>" \  
--artifacts {<artifacts>} \  
--environment "{\"type\": \"<environment-type>\", \"image\": \"<image-type>\",  
\"computeType\": \"<compute-type>\"}" \  
--service-role "<service-role>"
```

在上述命令中，取代下列預留位置：

- **<auto-retry-limit>**：將自動重試限制設定為建置失敗後所需的自動重試次數上限。
- **<project-name>**、**<source>**、**<artifacts>**、**<environment-type>**、**<image-type>**、**<compute-type>** 和 **<service-role>**：設定所需的專案組態設定。

自動重試組建 (AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

在 中停止建置 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台、AWS CLI、AWS SDKs來停止 中的建置 AWS CodeBuild。

主題

- [停止建置 \(主控台\)](#)
- [停止建置 \(AWS CLI\)](#)
- [停止建置 \(AWS SDKs\)](#)

停止建置 (主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 執行以下任意一項：
 - 如果顯示 **build-project-name:build-ID** 頁面，請選擇 Stop build (停止組建)。
 - 在導覽窗格中，選擇 Build history (組建歷史記錄)。在組建清單中，選取組建的方塊，然後選擇 Stop build (停止組建)。
 - 在導覽窗格中，選擇 Build projects (建置專案)。在組建專案清單中的 Name (名稱) 欄中，選擇組建專案名稱的連結。在組建清單中，選取組建的方塊，然後選擇 Stop build (停止組建)。

 Note

根據預設，只會顯示最新 100 個組建或組建專案。若要檢視更多組建或組建專案，請選擇齒輪圖示，然後針對 Builds per page (每頁顯示組建數) 或 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

如果 AWS CodeBuild 無法成功停止建置（例如，如果建置程序已完成），則會停用停止按鈕，否則可能不會顯示。

停止建置 (AWS CLI)

- 執行 stop-build 命令：

```
aws codebuild stop-build --id id
```

在上述命令中，取代下列預留位置：

- id*：必要的字串。要停止的組建 ID。若要取得建置 ID 的清單，請參閱下列主題：
 - [檢視建置 ID 清單 \(AWS CLI\)](#)
 - [檢視建置專案的建置 ID 清單 \(AWS CLI\)](#)

如果 AWS CodeBuild 成功停止建置，則輸出中build物件中的buildStatus值為 STOPPED。

如果 CodeBuild 無法成功停止建置（例如，如果建置已完成），則輸出中build物件中的buildStatus值為最終建置狀態（例如，SUCCEEDED）。

停止建置 (AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

在 中停止批次建置 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台 AWS CLI或 AWS SDKs來停止 中的批次建置 AWS CodeBuild。

Note

如果您在批次建置中使用 Lambda 運算，則無法停止進行中的 Lambda 建置。

主題

- [停止批次建置（主控台）](#)
- [停止批次建置 \(AWS CLI\)](#)
- [停止批次建置AWS SDKs](#)

停止批次建置（主控台）

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 執行以下任意一項：
 - 如果顯示 **build-project-name:build-ID** 頁面，請選擇 Stop build (停止組建)。
 - 在導覽窗格中，選擇 Build history (組建歷史記錄)。在組建清單中，選取組建的方塊，然後選擇 Stop build (停止組建)。
 - 在導覽窗格中，選擇 Build projects (建置專案)。在組建專案清單中的 Name (名稱) 欄中，選擇組建專案名稱的連結。在組建清單中，選取組建的方塊，然後選擇 Stop build (停止組建)。

Note

根據預設，只會顯示最新 100 個組建或組建專案。若要檢視更多組建或組建專案，請選擇齒輪圖示，然後針對 Builds per page (每頁顯示組建數) 或 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

停止批次建置 (AWS CLI)

- 執行 [stop-build-batch](#) 命令：

```
aws codebuild stop-build-batch --id <batch-build-id>
```

在上述命令中，取代下列預留位置：

- `<batch-build-id>`：必要字串。要停止的批次建置識別符。若要取得批次建置識別符的清單，請參閱下列主題：
 - [檢視批次建置 IDs 的清單 \(AWS CLI\)](#)
 - [檢視組建專案的批次組建 IDs 清單 \(AWS CLI\)](#)

停止批次建置AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

自動觸發 AWS CodeBuild 建置

您可以在專案上建立觸發來排定每小時、每天或每週執行一次組建。您也可以編輯觸發條件，以使用具有 Amazon CloudWatch Cron 表達式的自訂規則。例如，您可以使用 cron 表達式來排定在每個工作日的特定時間執行組建。如需建立和編輯觸發程序的資訊，請參閱 [建立 AWS CodeBuild 觸發條件](#)和 [編輯 AWS CodeBuild 觸發條件](#)。

主題

- [建立 AWS CodeBuild 觸發條件](#)
- [編輯 AWS CodeBuild 觸發條件](#)

建立 AWS CodeBuild 觸發條件

您可以在專案上建立觸發來排定每小時、每天或每週執行一次組建。您也可以使用具有 Amazon CloudWatch Cron 表達式的自訂規則來建立觸發。例如，您可以使用 cron 表達式來排定在每個工作日的特定時間執行組建。

Note

您無法從組建觸發條件、Amazon EventBridge 事件或 AWS Step Functions 任務啟動批次組建。

主題

- [建立 AWS CodeBuild 觸發（主控台）](#)

- [以程式設計方式建立 AWS CodeBuild 觸發](#)

建立 AWS CodeBuild 觸發 (主控台)

使用下列程序，使用 建立觸發 AWS 管理主控台。

建立觸發

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 選擇您想要新增觸發的組建專案的連結，然後選擇 Build triggers (組建觸發) 索引標籤。

Note

根據預設，會顯示 100 個最新的建置專案。若要檢視更多組建專案，請選擇齒輪圖示，然後針對 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

4. 選擇 Create trigger (建立觸發)。
5. 在 Trigger name (觸發名稱) 中輸入名稱。
6. 從 Frequency (頻率) 下拉式清單中，選擇觸發的頻率。如果想要使用 Cron 表達式建立頻率，請選擇 Custom (自訂)。
7. 指定觸發的頻率參數。您可以在文字方塊中鍵入選項的前幾個字元來篩選下拉式選單項目。

Note

開始時間和分鐘為零。開始分鐘是介於零到 59 之間的數字。開始時間是介於零到 23 之間的數字。例如，每天下午 12：15 開始的每日觸發。開始時間為 12，開始時間為 15。每日午夜開始的觸發，開始時間為零，開始分鐘為零。每日觸發會在每天下午 11：59 開始。開始時間為 23，開始時間為 59。

Frequency (頻率)	必要參數	詳細資訊
每小時	起始分鐘	使用 Start minute (起始分鐘) 下拉式選單。

Frequency (頻率)	必要參數	詳細資訊
每日	起始分鐘 起始小時	使用 Start minute (起始分鐘) 下拉式選單。 使用 Start hour (起始小時) 下拉式選單。
每週	起始分鐘 起始小時 起始日	使用 Start minute (起始分鐘) 下拉式選單。 使用 Start hour (起始小時) 下拉式選單。 使用 Start day (起始日) 下拉式選單。
Custom	Cron 表達式	在 Cron expression (Cron 表達式) 中輸入 cron 表達式。Cron 表達式有六個必要欄位，以空格隔開。這些欄位指定了分鐘、小時、日、月、星期和年的起始值。您也可以使用萬用字元來指定一個範圍、其他值等。例如， <code>9:00 0 9 ? * MON-FRI</code> 排程組建。如需詳細資訊，請參閱《Amazon CloudWatch Events 使用者指南》中的 Cron 表達式 。

8. 選取 Enable this trigger (啟用此觸發)。
9. (選用) 展開 Advanced (進階) 區段。在 Source version (來源版本) 中，輸入來源的版本。
 - 針對 Amazon S3，輸入對應至您要建置之輸入成品版本的版本 ID。如果 Source version (來源版本) 為空白，將使用最新版本。

- 對於 AWS CodeCommit，輸入遞交 ID。如果 Source version (來源版本) 為空白，將使用預設分支的 HEAD 遞交 ID。
 - 針對 GitHub 或 GitHub Enterprise，輸入遞交 ID、提取要求 ID、分支名稱，或與您想要建置之原始碼版本對應的標籤名稱。如果指定提取要求 ID，其格式必須為 `pr/pull-request-ID` (例如，`pr/25`)。如果指定分支名稱，將使用分支的 HEAD 遞交 ID。如果 Source version (來源版本) 空白，則會使用預設分支的 HEAD 遞交 ID。
 - 針對 Bitbucket，輸入遞交 ID、分支名稱，或與您想要建置之原始碼版本對應的標籤名稱。如果指定分支名稱，將使用分支的 HEAD 遞交 ID。如果 Source version (來源版本) 空白，則會使用預設分支的 HEAD 遞交 ID。
10. (選用) 指定介於 5 分鐘到 2160 分鐘 (36 小時) 之間的逾時。此值指定建置在停止之前 AWS CodeBuild 嘗試建置的時間長度。如果 Hours (小時) 和 Minutes (分鐘) 空白，則會使用專案中指定的預設逾時值。
11. 選擇 Create trigger (建立觸發)。

以程式設計方式建立 AWS CodeBuild 觸發

CodeBuild 使用 Amazon EventBridge 規則進行建置觸發。您可以使用 EventBridge API，以程式設計方式為您的 CodeBuild 專案建立建置觸發。如需詳細資訊，請參閱 [Amazon EventBridge API 參考](#)。

編輯 AWS CodeBuild 觸發條件

您可以在專案上編輯觸發來排定每小時、每天或每週執行一次組建。您也可以編輯觸發條件，以使用具有 Amazon CloudWatch Cron 表達式的自訂規則。例如，您可以使用 cron 表達式來排定在每個工作日的特定時間執行組建。如需有關建立觸發的資訊，請參閱[建立 AWS CodeBuild 觸發條件](#)。

主題

- [編輯 AWS CodeBuild 觸發條件 \(主控台\)](#)
- [以程式設計方式編輯 AWS CodeBuild 觸發](#)

編輯 AWS CodeBuild 觸發條件 (主控台)

使用下列程序來編輯使用的觸發條件 AWS 管理主控台。

如何編輯觸發條件

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。

2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 選擇您想要變更的組建專案的連結，然後選擇 Build triggers (組建觸發) 索引標籤。

 Note

根據預設，會顯示 100 個最新的建置專案。若要檢視更多組建專案，請選擇齒輪圖示，然後針對 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

4. 選擇您想要變更的觸發旁邊的選項按鈕，然後選擇 Edit (編輯)。
5. 從 Frequency (頻率) 下拉式清單中，選擇觸發的頻率。如果想要使用 Cron 表達式建立頻率，請選擇 Custom (自訂)。
6. 指定觸發的頻率參數。您可以在文字方塊中鍵入選項的前幾個字元來篩選下拉式選單項目。

 Note

開始時間和分鐘為零。開始分鐘是介於零到 59 之間的數字。開始時間是介於零到 23 之間的數字。例如，每天下午 12：15 開始的每日觸發。開始時間為 12，開始時間為 15。每日午夜開始的觸發，開始時間為零，開始分鐘為零。每日觸發會在每天下午 11：59 開始。開始時間為 23，開始時間為 59。

Frequency (頻率)	必要參數	詳細資訊
每小時	起始分鐘	使用 Start minute (起始分鐘) 下拉式選單。
每日	起始分鐘	使用 Start minute (起始分鐘) 下拉式選單。
	起始小時	使用 Start hour (起始小時) 下拉式選單。
每週	起始分鐘	使用 Start minute (起始分鐘) 下拉式選單。
	起始小時	
	起始日	使用 Start hour (起始小時) 下拉式選單。

Frequency (頻率)	必要參數	詳細資訊
		使用 Start day (起始日) 下拉式選單。
Custom	Cron 表達式	在 Cron expression (Cron 表達式) 中輸入 cron 表達式。Cron 表達式有六個必要欄位，以空格隔開。這些欄位指定了分鐘、小時、日、月、星期和年的起始值。您也可以使用萬用字元來指定一個範圍、其他值等。例如，cron 表達式會在每個工作日上午 9:00 0 9 ? * MON-FRI * 排程組建。如需詳細資訊，請參閱《Amazon CloudWatch Events 使用者指南》中的 Cron 表達式 。

7. 選取 Enable this trigger (啟用此觸發)。

Note

您可以使用 Amazon CloudWatch 主控台，網址為 <https://console.aws.amazon.com/cloudwatch/>：// AWS CodeBuild。

以程式設計方式編輯 AWS CodeBuild 觸發

CodeBuild 使用 Amazon EventBridge 規則進行建置觸發。您可以使用 EventBridge API，以程式設計方式編輯 CodeBuild 專案的建置觸發條件。如需詳細資訊，請參閱 [Amazon EventBridge API 參考](#)。

在中檢視建置詳細資訊 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台 AWS CLI 或 AWS SDKs 來檢視 CodeBuild 管理之組建的詳細資訊。

主題

- [檢視建置的詳細資訊 \(主控台\)](#)
- [檢視建置的詳細資訊 \(AWS CLI\)](#)
- [檢視建置詳細資訊AWS SDKs\)](#)
- [建置階段轉換](#)

檢視建置的詳細資訊 (主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 執行以下任意一項：
 - 在導覽窗格中，選擇 Build history (組建歷史記錄)。在組建清單中的 Build run (組建執行) 欄中，選擇組建的連結。
 - 在導覽窗格中，選擇 Build projects (建置專案)。在組建專案清單中的 Name (名稱) 欄中，選擇組建專案名稱的連結。然後，在組建清單中的 Build run (組建執行) 欄中，選擇組建的連結。

Note

根據預設，只會顯示最新 10 個組建或組建專案。若要檢視更多組建或組建專案，請選擇齒輪圖示，然後針對 Builds per page (每頁顯示組建數) 或 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

檢視建置的詳細資訊 (AWS CLI)

如需 AWS CLI 搭配使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

執行 batch-get-builds 命令：

```
aws codebuild batch-get-builds --ids ids
```

取代下列預留位置：

- *ids*：必要的字串。要檢視詳細資訊的一或多個組建 ID。若要指定一個以上的組建 ID，請以空格將每個組建 ID 分開。您最多可以指定 100 個組建 ID。若要取得建置 ID 的清單，請參閱下列主題：

- 檢視建置 ID 清單 (AWS CLI)
- 檢視建置專案的建置 ID 清單 (AWS CLI)

例如，如果您執行此命令：

```
aws codebuild batch-get-builds --ids codebuild-demo-project:e9c4f4df-3f43-41d2-  
ab3a-60fe2EXAMPLE codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE my-other-  
project:813bb6c6-891b-426a-9dd7-6d8a3EXAMPLE
```

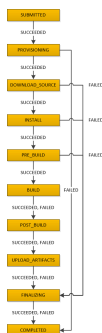
如果命令成功，輸出中會顯示類似於 [檢視摘要建置資訊](#) 所述的資料。

檢視建置詳細資訊AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs 和工具參考](#)。

建置階段轉換

逐步建置 AWS CodeBuild :



 Important

即使 BUILD 階段失敗，也一定會嘗試 UPLOAD_ARTIFACTS 階段。

檢視 中的組建 IDs清單 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台、AWS CLI 或 AWS SDKs 來檢視 CodeBuild 管理之組建 IDs 清單。

主題

- 檢視建置 ID 清單 (主控台)

- [檢視建置 ID 清單 \(AWS CLI\)](#)
- [檢視批次建置 IDs 的清單 \(AWS CLI\)](#)
- [檢視組建 IDs \(AWS SDKs 的清單\)](#)

檢視建置 ID 清單 (主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build history (組建歷史記錄)。

Note

根據預設，只會顯示最新 10 個組建。若要檢視更多組建，請選擇齒輪圖示，然後針對 Builds per page (每頁顯示組建數) 選擇不同的值，或使用向前和向後箭頭。

檢視建置 ID 清單 (AWS CLI)

如需 AWS CLI 搭配 CodeBuild 使用的詳細資訊，請參閱 [命令列參考](#)。

- 執行 list-builds 命令：

```
aws codebuild list-builds --sort-order sort-order --next-token next-token
```

在上述命令中，取代下列預留位置：

- *sort-order*：用於表示如何列出組建 ID 的選用字串。有效值包括 ASCENDING 與 DESCENDING。
- *next-token*：選用字串。在先前的執行中，如果清單具有超過 100 個項目，則只會傳回前 100 個項目，以及稱為 next token 的唯一字串。若要取得清單中下一個批次中的項目，請再次執行此命令，將 next token 新增至呼叫。若要取得清單中的所有項目，請繼續對每個後續的 next token 執行此命令，直到不再傳回 next token 為止。

例如，如果您執行此命令：

```
aws codebuild list-builds --sort-order ASCENDING
```

類似下列的結果可能會顯示於輸出：

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

如果您再次執行此命令：

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full
token has been omitted for brevity...MzY2OA==
```

類似下列的結果可能會顯示於輸出：

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

檢視批次建置 IDs 的清單 (AWS CLI)

如需 AWS CLI 搭配 CodeBuild 使用的詳細資訊，請參閱 [命令列參考](#)。

- 執行 list-build-batches 命令：

```
aws codebuild list-build-batches --sort-order sort-order --next-token next-token
```

在上述命令中，取代下列預留位置：

- **sort-order** : 選用字串，用於指示如何列出批次建置 IDs。有效值包括 ASCENDING 與 DESCENDING。
- **next-token** : 選用字串。在先前的執行中，如果清單具有超過 100 個項目，則只會傳回前 100 個項目，以及稱為 next token 的唯一字串。若要取得清單中下一個批次中的項目，請再次執行此命令，將 next token 新增至呼叫。若要取得清單中的所有項目，請繼續對每個後續的 next token 執行此命令，直到不再傳回 next token 為止。

例如，如果您執行此命令：

```
aws codebuild list-build-batches --sort-order ASCENDING
```

類似下列的結果可能會顯示於輸出：

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY20A==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE",
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

如果您再次執行此命令：

```
aws codebuild list-build-batches --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for brevity...MzY20A==
```

類似下列的結果可能會顯示於輸出：

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

檢視組建 IDs(AWS SDKs的清單)

如需搭配 AWS SDKs 使用 CodeBuild 的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

在 中檢視組建專案的組建 IDs 清單 AWS CodeBuild

您可以使用 AWS CodeBuild 主控台 AWS CLI或 AWS SDKs在 CodeBuild 中檢視組建IDs 清單。

主題

- [檢視建置專案的建置 ID 清單 \(主控台\)](#)
- [檢視建置專案的建置 ID 清單 \(AWS CLI\)](#)
- [檢視組建專案的批次組建 IDs 清單 \(AWS CLI\)](#)
- [檢視組建專案的組建 IDs 清單 \(AWS SDKs\)](#)

檢視建置專案的建置 ID 清單 (主控台)

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。在組建專案清單中，於 Name (名稱) 欄中，選擇組建專案。

Note

根據預設，只會顯示最新 100 個組建或組建專案。若要檢視更多組建或組建專案，請選擇齒輪圖示，然後針對 Builds per page (每頁顯示組建數) 或 Projects per page (每頁顯示專案數) 選擇不同的值，或使用向前和向後箭頭。

檢視建置專案的建置 ID 清單 (AWS CLI)

如需 AWS CLI 搭配 使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

執行 list-builds-for-project 命令，如下所示：

```
aws codebuild list-builds-for-project --project-name project-name --sort-order sort-order --next-token next-token
```

在上述命令中，取代下列預留位置：

- **project-name**：必要字串，用於表示要列出組建 ID 的組建專案名稱。若要取得組建專案清單，請參閱[檢視建置專案名稱清單 \(AWS CLI\)](#)。
- **sort-order**：用於表示如何列出組建 ID 的選用字串。有效值包括 ASCENDING 與 DESCENDING。
- **next-token**：選用字串。在先前的執行中，如果清單具有超過 100 個項目，則只會傳回前 100 個項目，以及稱為 next token 的唯一字串。若要取得清單中下一個批次中的項目，請再次執行此命令，將 next token 新增至呼叫。若要取得清單中的所有項目，請繼續對每個後續傳回的 next token 執行此命令，直到不再傳回 next token 為止。

例如，如果您執行類似以下的命令：

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-order ASCENDING
```

輸出中可能會顯示與下列類似的結果：

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
  ]
}
```

如果您再次執行此命令：

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

您在輸出中可能會看到如下的結果：

```
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
    "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
  ]
}
```

```
... The full list of build IDs has been omitted for brevity ...  
"codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"  
]  
}
```

檢視組建專案的批次組建 IDs 清單 (AWS CLI)

如需 AWS CLI 搭配使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

執行 `list-build-batches-for-project` 命令，如下所示：

```
aws codebuild list-build-batches-for-project --project-name project-name --sort-  
order sort-order --next-token next-token
```

在上述命令中，取代下列預留位置：

- *project-name*：必要字串，用於表示要列出組建 ID 的組建專案名稱。若要取得組建專案清單，請參閱[檢視建置專案名稱清單 \(AWS CLI\)](#)。
- *sort-order*：用於表示如何列出組建 ID 的選用字串。有效值包括 ASCENDING 與 DESCENDING。
- *next-token*：選用字串。在先前的執行中，如果清單具有超過 100 個項目，則只會傳回前 100 個項目，以及稱為 next token 的唯一字串。若要取得清單中下一個批次中的項目，請再次執行此命令，將 next token 新增至呼叫。若要取得清單中的所有項目，請繼續對每個後續傳回的 next token 執行此命令，直到不再傳回 next token 為止。

例如，如果您執行類似以下的命令：

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project --  
sort-order ASCENDING
```

輸出中可能會顯示與下列類似的結果：

```
{  
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",  
  "ids": [  
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"  
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"  
    ... The full list of build IDs has been omitted for brevity ...  
    "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"  
  ]  
}
```

```
}
```

如果您再次執行此命令：

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project
--sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for
brevity...MzY2OA==
```

您在輸出中可能會看到如下的結果：

```
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
    "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

檢視組建專案的組建 IDs 清單 (AWS SDKs)

如需 AWS CodeBuild 搭配 AWS SDKs 使用的詳細資訊，請參閱 [AWS SDKs和工具參考](#)。

中的測試報告 AWS CodeBuild

您可以在 CodeBuild 中建立報告，其中包含建置期間執行之測試的詳細資訊。您可以建立如單位測試、組態測試和功能測試等測試。

支援下列測試報告檔案格式：

- 小黃瓜 JSON (.json)
- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX (.trx)
- Visual Studio TRX XML (.xml)

Note

最新支援的 版本cucumber-js是 7.3.2。

使用可以用其中一種格式 (例如，Surefire JUnit 外掛程式、TestNG 或 Cucumber) 建立報告檔案的任何測試框架，來建立您的測試案例。

若要建立測試報告，請將報告群組名稱新增至內有測試案例之建置專案的 buildspec 檔案。當您執行建置專案時，便會執行測試案例並建立測試報告。每次測試案例執行時，都會在報告群組中建立新的測試報告。執行測試之前，無須建立報告群組。如果您指定報告群組名稱，CodeBuild 會在您執行報告時為您建立報告群組。如果您想要使用已存在的報告群組，請在 buildspec 檔案中指定其 ARN。

您可以使用測試報告在建置執行時協助針對問題進行故障診斷。如果您有許多測試報告來自於建置專案的多個建置，則可以使用測試報告檢視趨勢和測試，以及失敗率，以協助您最佳化建置。

報告會在建立 30 天後過期。您無法檢視已過期的測試報告。如果您想要保留測試報告超過 30 天，您可以將測試結果的原始資料檔案匯出至 Amazon S3 儲存貯體。匯出的測試檔案不會過期。當您建立測試群組時，就會指定 S3 儲存貯體的相關資訊。

 Note

專案中指定的 CodeBuild 服務角色用於上傳到 S3 儲存貯體的許可。

主題

- [建立測試報告](#)
- [建立程式碼涵蓋範圍報告](#)
- [在 CodeBuild 中自動探索報告](#)
- [報告群組](#)
- [測試架構](#)
- [檢視測試報告](#)
- [測試報告許可](#)
- [測試報告狀態](#)

建立測試報告

若要建理測試報告，請執行設定為在其 buildspec 檔案中有一到五個報告群組的建置專案。測試報告會在執行期間建立。它包含報告群組指定的測試案例結果。使用相同 buildspec 檔案的各個後續建置都會產生新的測試報告。

建立測試報告

1. 建立建置專案。如需相關資訊，請參閱[在 中建立建置專案 AWS CodeBuild](#)。
2. 請使用新的測試報告資訊設定您專案的 buildspec 檔案：

- a. 新增reports:區段並指定現有報告群組的 ARN，或報告群組的名稱。

如果您指定 ARN，CodeBuild 會使用該報告群組。

如果您指定名稱，CodeBuild 會使用專案名稱和您指定的名稱，以 *<project-name>-<report-group-name>* 格式為您建立報告群組。如果已命名的報告群組已存在，CodeBuild 會使用該報告群組。

- b. 在報告群組下，指定包含測試結果的檔案位置。如果您不只使用一個報告群組，請指定每個報告群組的測試結果檔案位置。您的建置專案每次執行時都會建立新的測試報告。如需詳細資訊，請參閱[指定測試檔案](#)。

- c. 在 `build` 或 `post_build` 序列的 `commands` 區段中，請指定命令，藉以執行您為自己的報告群組指定的測試案例。如需詳細資訊，請參閱[指定測試命令](#)。

以下是 `buildspec reports` 區段的範例：

```
reports:
  php-reports:
    files:
      - "reports/php/*.xml"
    file-format: "JUNITXML"
  nunit-reports:
    files:
      - "reports/nunit/*.xml"
    file-format: "NUNITXML"
```

3. 執行建置專案的建置。如需詳細資訊，請參閱[手動執行 AWS CodeBuild 組建](#)。
4. 當建置完成時，請從專案頁面上的 Build history (建置歷史記錄) 選擇新的建置執行。選擇 Reports (報告) 以檢視測試報告。如需詳細資訊，請參閱[檢視建置的測試報告](#)。

建立程式碼涵蓋範圍報告

CodeBuild 可讓您為測試產生程式碼涵蓋範圍報告。提供下列程式碼涵蓋範圍報告：

線路涵蓋範圍

線路涵蓋範圍會測量您的測試涵蓋多少陳述式。陳述式是單一的指示，不包括備註或條件。

$$\text{line coverage} = (\text{total lines covered}) / (\text{total number of lines})$$

分支涵蓋範圍

分支涵蓋範圍會衡量您的測試在控制結構的每個可能分支中涵蓋了多少個分支，比如 `if` 或 `case` 陳述式。

$$\text{branch coverage} = (\text{total branches covered}) / (\text{total number of branches})$$

支援下列程式碼涵蓋範圍報告檔案格式：

- JaCoCo XML
- SimpleCov JSON¹

- Clover XML
- Cobertura XML
- LCOV 資訊

¹ CodeBuild 接受由 [simplecov](#) 產生的 JSON 程式碼涵蓋範圍報告，而非 [simplecov-json](#)。

建立程式碼涵蓋範圍報告

若要建立程式碼涵蓋範圍報告，請執行建置專案，該專案在其 `buildspec` 檔案中至少設定一個程式碼涵蓋範圍報告群組。CodeBuild 會解譯程式碼涵蓋範圍結果，並提供執行的程式碼涵蓋範圍報告。使用相同 `buildspec` 檔案的各個後續建置都會產生新的測試報告。

建立測試報告

1. 建立建置專案。如需相關資訊，請參閱[在 中建立建置專案 AWS CodeBuild](#)。
2. 使用測試報告資訊設定專案的 `buildspec` 檔案：
 - a. 新增 `reports` 區段，並指定報告群組的名稱。CodeBuild 會使用專案名稱和您以 `project-name-` 格式指定的名稱，為您建立報告群組 `report-group-name-in-buildspec`。如果您已有要使用的報告群組，請指定其 ARN。如果您使用名稱而非 ARN，CodeBuild 會建立新的報告群組。如需詳細資訊，請參閱[Reports syntax in the buildspec file](#)。
 - b. 在報告群組下，指定包含程式碼涵蓋範圍結果的檔案位置。如果您使用多個報告群組，請為每個報告群組指定結果檔案位置。每次建置專案執行時，都會建立新的程式碼涵蓋範圍報告。如需詳細資訊，請參閱[指定測試檔案](#)。

這是為位於 `test-` 中的 JaCoCo XML 結果檔案產生程式碼涵蓋範圍報告的範例 `results/jacoco-coverage-report.xml`。

```
reports:
  jacoco-report:
    files:
      - 'test-results/jacoco-coverage-report.xml'
    file-format: 'JACOCOXML'
```

- c. 在 `build` 或 `post_build` 序列的 `commands` 區段中，指定執行程式碼涵蓋範圍分析的命令。如需詳細資訊，請參閱[指定測試命令](#)。
3. 執行建置專案的建置。如需詳細資訊，請參閱[手動執行 AWS CodeBuild 組建](#)。

4. 當建置完成時，請從專案頁面上的 Build history (建置歷史記錄) 選擇新的建置執行。選擇報告以檢視程式碼涵蓋範圍報告。如需詳細資訊，請參閱[檢視建置的測試報告](#)。

在 CodeBuild 中自動探索報告

透過自動探索，CodeBuild 會在建置階段完成後搜尋所有建置檔案、搜尋任何支援的報告檔案類型，並自動建立新的測試和程式碼涵蓋範圍報告群組和報告。對於任何發現的報告類型，CodeBuild 會使用下列模式建立新的報告群組：

```
<project-name>-<report-file-format>-AutoDiscovered
```

Note

如果探索的報告檔案具有相同的格式類型，則會將其放入相同的報告群組或報告。

報告自動探索是由您的專案環境變數設定：

CODEBUILD_CONFIG_AUTO_DISCOVER

此變數會判斷是否在建置期間停用報告自動探索。根據預設，會為所有組建啟用報告自動探索。若要停用此功能，請將 CODEBUILD_CONFIG_AUTO_DISCOVER 設定為 false。

CODEBUILD_CONFIG_AUTO_DISCOVER_DIR

(選用) 此變數決定 CodeBuild 搜尋潛在報告檔案的位置。請注意，**/* 根據預設，CodeBuild 會在 中搜尋。

這些環境變數可在建置階段期間修改。例如，如果您只想為 git main 分支上的組建啟用報告自動探索，您可以在組建過程中檢查 git 分支，如果組建不在 main 分支上，則 CODEBUILD_CONFIG_AUTO_DISCOVER 設定為 false。您可以使用 主控台 或使用專案環境變數來停用報告自動探索。

主題

- [使用主控台設定報告自動探索](#)
- [使用專案環境變數設定報告自動探索](#)

使用主控台設定報告自動探索

使用下列程序，使用主控台設定報告自動探索。

使用主控台設定報告自動探索

1. 建立建置專案或選擇要編輯的建置專案。如需詳細資訊，請參閱 [在中建立建置專案 AWS CodeBuild](#) 或 [在中變更建置專案設定 AWS CodeBuild](#)。
2. 在環境中，選取其他組態。
3. 若要停用報告自動探索，請在報告自動探索中，選取停用報告自動探索。
4. （選用）在自動探索目錄 - 選用中，輸入 CodeBuild 的目錄模式來搜尋支援的報告格式檔案。請注意，`**/*` 根據預設，CodeBuild 會在 `中` 搜尋。

使用專案環境變數設定報告自動探索

使用下列程序，使用專案環境變數設定報告自動探索。

使用專案環境變數設定報告自動探索

1. 建立建置專案或選擇要編輯的建置專案。如需詳細資訊，請參閱 [在中建立建置專案 AWS CodeBuild](#) 或 [在中變更建置專案設定 AWS CodeBuild](#)。
2. 在環境變數中，執行下列動作：
 - a. 若要停用報告自動探索，請在名稱中輸入 `CODEBUILD_CONFIG_AUTO_DISCOVER` 在值中輸入 `false`。這會停用報告自動探索。
 - b. （選用）對於名稱，輸入 `CODEBUILD_CONFIG_AUTO_DISCOVER_DIR`，對於值，輸入 CodeBuild 應搜尋支援報告格式檔案的目錄。例如，`output/*xml` 會在 `output` 目錄中搜尋 `.xml` 檔案

報告群組

報告群組包含測試報告並指定共用設定。您可以使用 `buildspec` 檔案指定要執行的測試案例，並執行其建置時要執行的命令。對於在建置專案中設定的每個報告群組，執行建置專案可以建立測試報告。多次執行使用報告群組設定的建置專案時，則會在該報告群組中建立多個測試報告，每個報告都包含針對該報告群組所指定之相同測試案例的結果。

在建置專案的 `buildspec` 檔案中，會為報告群組指定測試案例。您可以在一個建置專案中最多指定五個報告群組。當您執行建置時，所有測試案例都會執行。新的測試報告會以報告群組指定的各個測試案例結果建立。每次執行新的建置時，測試案例便會執行，並以新的測試結果建立。

報告群組可以用於多個建置專案。使用一個報告群組建立的所有測試報告都可以共用相同的組態，例如其匯出選項和許可，即使是使用不同的建置專案建立測試報告。如果測試報告是使用在多個建置專案中的一個報告群組建立，該測試報告可以包含執行多組不同的測試案例後取得的結果 (每個建置專案一組測試案例)。這是因為您可以為每個專案 `buildspec` 檔案中的報告群組，指定不同的測試案例檔案。您也可以編輯其 `buildspec` 檔案，變更建置專案中報告群組的測試案例檔案。後續建置執行時會建立新的測試報告，而且報告包含在已更新 `buildspec` 中的測試案例檔案結果。

主題

- [建立報告群組](#)
- [報告群組命名](#)
- [共用報告群組](#)
- [指定測試檔案](#)
- [指定測試命令](#)
- [在 中標記報告群組 AWS CodeBuild](#)
- [更新報告群組](#)

建立報告群組

您可以使用 CodeBuild 主控台 AWS CLI、或 `buildspec` 檔案來建立報告群組。您的 IAM 角色必須擁有建立報告群組所需的許可。如需詳細資訊，請參閱[測試報告許可](#)。

主題

- [建立報告群組 \(Buildspec\)](#)
- [建立報告群組 \(主控台\)](#)
- [建立報告群組 \(CLI\)](#)
- [建立報告群組 \(CloudFormation\)](#)

建立報告群組 (Buildspec)

使用 `buildspec` 建立的報告群組不會匯出原始測試結果檔案。您可以檢視報告群組，並指定匯出設定。如需詳細資訊，請參閱[更新報告群組](#)。

使用 buildspec 檔案建立報告群組

1. 選擇與您 AWS 帳戶中的報告群組沒有關聯的報告群組名稱。
2. 使用此名稱設定 buildspec 檔案的 reports 區段。在此範例中，報告群組名稱是 new-report-group 並使用 JUnit 框架建立使用測試案例：

```
reports:
  new-report-group: #surefire junit reports
    files:
      - '**/*'
    base-directory: 'surefire/target/surefire-reports'
```

您也可以在 buildspec 中使用環境變數來指定報告群組名稱：

```
version: 0.2
env:
  variables:
    REPORT_GROUP_NAME: "new-report-group"
phases:
  build:
    commands:
      - ...
...
reports:
  $REPORT_GROUP_NAME:
    files:
      - '**/*'
    base-directory: 'surefire/target/surefire-reports'
```

如需詳細資訊，請參閱[指定測試檔案](#)及[Reports syntax in the buildspec file](#)。

3. 在 commands 區段中指定執行測試的命令。如需詳細資訊，請參閱[指定測試命令](#)。
4. 執行組建。當建置完成時，便會以使用格式 project-name-report-group-name 的名稱建立新的報告群組。如需詳細資訊，請參閱[報告群組命名](#)。

建立報告群組 (主控台)

使用下列程序，使用 建立報告群組 AWS 管理主控台。

建立報告群組

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Report groups (報告群組)。
3. 選擇 Create report group (建立報告群組)。
4. 在 Report group name (報告群組名稱) 中，輸入報告群組的名稱。
5. (選用) 對於標籤，輸入您希望支援服務 AWS 使用的任何標籤的名稱和值。使用 Add row (新增資料列) 來新增標籤。您最多可以新增 50 個標籤。
6. 如果您想要將測試報告結果的原始資料上傳至 Amazon S3 儲存貯體：
 - a. 選取匯出至 Amazon S3。
 - b. 針對 S3 bucket name (S3 儲存貯體名稱)，輸入 S3 儲存貯體的名稱。
 - c. (選用) 對於 S3 儲存貯體擁有者，輸入擁有 S3 儲存貯體之帳戶的帳戶識別符。AWS 這可讓報告資料匯出到由執行組建帳戶以外的帳戶所擁有的 Amazon S3 儲存貯體。
 - d. 針對 Path prefix (路徑前置詞)，輸入您想要在 S3 儲存貯體中上傳測試結果的路徑。
 - e. 選取 Compress test result data in a zip file (以 zip 檔案壓縮測試結果資料)，以壓縮原始測試結果資料檔案。
 - f. 展開 Additional configuration (其他組態) 以顯示加密選項。選擇下列其中一項：
 - 用於 AWS 受管金鑰 Amazon S3 的預設 AWS 受管金鑰。如需詳細資訊，請參閱AWS Key Management Service 《使用者指南》中的[客戶受管 CMKs](#)。這是預設加密選項。
 - 選擇自訂金鑰以使用您建立和設定的客戶受管金鑰。針對AWS KMS 加密金鑰，輸入加密金鑰的 ARN。格式為 `arn:aws:kms:<region-id>:<aws-account-id>:key/<key-id>`。如需詳細資訊，請參閱AWS Key Management Service 《使用者指南》中的[建立 KMS 金鑰](#)。
 - Disable artifact encryption (停用成品) 可讓您停用加密。如果您想要共享測試結果，或發佈至靜態網站，您可以選擇此選項。(動態網站可以執行程式碼，以解密測試結果。)

如需加密待用資料的詳細資訊，請參閱[資料加密](#)。

Note

專案中指定的 CodeBuild 服務角色用於上傳到 S3 儲存貯體的許可。

7. 選擇 Create report group (建立報告群組)。

建立報告群組 (CLI)

使用下列程序，使用 建立報告群組 AWS CLI。

建立報告群組

1. 建立名為 CreateReportGroup.json 的檔案。
2. 根據您的需求，將下列其中一個 JSON 程式碼片段複製到 CreateReportGroup.json：
 - 使用以下 JSON 指定您的測試報告群組將原始測試結果檔案匯出至 Amazon S3 儲存貯體。

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<bucket-name>",
      "bucketOwner": "<bucket-owner>",
      "path": "<path>",
      "packaging": "NONE | ZIP",
      "encryptionDisabled": "false",
      "encryptionKey": "<your-key>"
    },
    "tags": [
      {
        "key": "tag-key",
        "value": "tag-value"
      }
    ]
  }
}
```

- 將 **<bucket-name>** 取代為您的 Amazon S3 儲存貯體名稱，並將 **<path>** 取代為您儲存貯體中要匯出檔案的路徑。
- 如果您想要壓縮匯出的檔案，請為 packaging 指定 ZIP。否則請指定 NONE。
- bucketOwner 是選用的，只有在 Amazon S3 儲存貯體為執行建置之帳戶以外的帳戶所擁有時才需要。

- 使用 `encryptionDisabled` 指定是否要將匯出的檔案加密。如果您加密匯出的檔案，請輸入您的客戶受管金鑰。如需詳細資訊，請參閱[更新報告群組](#)。
- 若不要讓您的測試報告匯出原始測試檔案，請使用下列 JSON。

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "NO_EXPORT"
  }
}
```

Note

專案中指定的 CodeBuild 服務角色用於上傳到 S3 儲存貯體的許可。

3. 執行以下命令：

```
aws codebuild create-report-group --cli-input-json file://
CreateReportGroupInput.json
```

建立報告群組 (CloudFormation)

使用下列指示來使用 CloudFormation 範本建立報告群組

使用 CloudFormation 範本建立報告群組

您可以使用 CloudFormation 範本檔案來建立和佈建報告群組。如需詳細資訊，請參閱[CloudFormation 使用者指南](#)。

下列 CloudFormation YAML 範本會建立不會匯出原始測試結果檔案的報告群組。

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
```

```
ExportConfigType: NO_EXPORT
```

下列 CloudFormation YAML 範本會建立報告群組，將原始測試結果檔案匯出至 Amazon S3 儲存貯體。

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: S3
        S3Destination:
          Bucket: amzn-s3-demo-bucket
          Path: path-to-folder-for-exported-files
          Packaging: ZIP
          EncryptionKey: my-KMS-encryption-key
          EncryptionDisabled: false
```

Note

專案中指定的 CodeBuild 服務角色用於上傳到 S3 儲存貯體的許可。

報告群組命名

當您使用 AWS CLI 或 AWS CodeBuild 主控台建立報告群組時，您可以指定報告群組的名稱。如果您使用 buildspec 建立新的報告群組，則會使用格式 *project-name-report-group-name-specified-in-buildspec* 命名。如果報告是透過執行建置專案的建置方式建立，則此類報告皆屬於有新名稱的新報告群組。

如果您不希望 CodeBuild 建立新的報告群組，請在組建專案的 buildspec 檔案中指定報告群組的 ARN。您可以在多個建置專案中指定報告群組的 ARN。每個建置專案執行後，報告群組包含每個建置專案建立的測試報告。

例如，如果您以名稱 my-report-group 建立一個報告群組，然後在兩個名為 my-project-1 和 my-project-2 的不同建置專案中使用其名稱，並建立這兩個專案的一個建置，則會建立兩個新的報告群組。結果是採用下列名稱的三個報告群組：

- `my-report-group`：沒有任何測試報告。
- `my-project-1-my-report-group`：包含的報告擁有由名為 `my-project-1` 之建置專案所執行測試的結果。
- `my-project-2-my-report-group`：包含的報告擁有由名為 `my-project-2` 之建置專案所執行測試的結果。

如果您在這兩個專案中使用名為 `my-report-group` 之報告群組的 ARN，然後執行各個專案的建置，則仍有一個報告群組 (`my-report-group`)。該報告群組包含的測試報告含有由這兩個建置專案所執行之測試的結果。

如果您選擇的報告群組名稱不屬於您 AWS 帳戶中的報告群組，然後將該名稱用於 `buildspec` 檔案中的報告群組並執行其建置專案的建置，則會建立新的報告群組。新的報告群組的名稱格式為 *project-name-new-group-name*。例如，如果 AWS 您的帳戶中沒有名為 `new-report-group` 的報告群組，並在名為 `test-project` 的組建專案中指定它，組建執行會以名稱 `test-project-new-report-group` 建立新的報告群組。

共用報告群組

報告群組共用可讓多個 AWS 帳戶或使用者檢視報告群組、其未過期的報告，以及其報告的測試結果。在此模型中，擁有報告群組的帳戶 (擁有者) 會與其他帳戶 (消費者) 共用報告群組。消費者無法編輯報告群組。報告會在建立 30 天後過期。

主題

- [共用報告群組](#)
- [相關服務](#)
- [存取與您共用的報告群組](#)
- [取消共用報告群組](#)
- [識別共用報告群組](#)
- [共用報告群組的許可](#)

共用報告群組

當您共用報告群組時，就會授予消費者報告群組及其報告的唯讀存取權。取用者可以使用 AWS CLI 來檢視報告群組、其報告，以及每個報告的測試案例結果。消費者無法：

- 在 CodeBuild 主控台中檢視共用報告群組或其報告。

- 編輯共用報告群組。
- 使用專案中共用報告群組的 ARN 來執行報告。指定共用報告群組的專案建置會失敗。

您可以使用 CodeBuild 主控台將報告群組新增至現有的資源共享。如果您想要將報告群組新增至新的資源共用，需先在 [AWS RAM 主控台](#) 中建立它。

若要與組織單位或整個組織共用報告群組，必須啟用透過 AWS Organizations 來共用的功能。如需詳細資訊，請參閱《AWS RAM 使用者指南》中的 [透過 AWS Organizations 啟用共用](#)。

您可以使用 CodeBuild AWS RAM 主控台、主控台或 AWS CLI 來共用您擁有的報告群組。

先決條件

若要共用報告群組，AWS 您的帳戶必須擁有該群組。您無法將已共用給您的報告群組再共用出去。

共用您擁有的報告群組 (CodeBuild 主控台)

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Report groups (報告群組)。
3. 選擇您要共用的專案，然後選擇 Share (共用)。如需詳細資訊，請參閱《AWS RAM 使用者指南》中的 [建立資源共用](#)。

共用您擁有的報告群組 AWS RAM (主控台)

請參閱 AWS RAM 使用者指南中的 [建立資源共用](#)。

共用您擁有的報告群組 (AWS RAM 命令)

使用 [create-resource-share](#) 命令。

共用您擁有的報告群組 (CodeBuild 命令)

使用 [put-resource-policy](#) 命令：

1. 建立名為 policy.json 的檔案，並將以下命令複製到其中。

JSON

```
{
```

```

"Version":"2012-10-17",
"Statement":[{"
  "Effect":"Allow",
  "Principal":{"
    "AWS":"111122223333"
  },
  "Action":[
    "codebuild:BatchGetReportGroups",
    "codebuild:BatchGetReports",
    "codebuild:ListReportsForReportGroup",
    "codebuild:DescribeTestCases"],
  "Resource":"arn:aws:iam::*:role/Service*"
}]
}

```

2. 使用報告群組 ARN 和識別符更新 policy.json，以與之共用。下列範例使用 Alice 的 ARN 和 123456789012 所識別 AWS 帳戶的根使用者arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group，授予報告群組的唯讀存取權。

JSON

```

{
  "Version":"2012-10-17",
  "Statement":[{"
    "Effect":"Allow",
    "Principal":{"
      "AWS": [
        "arn:aws:iam::123456789012:user/Alice",
        "123456789012"
      ]
    },
    "Action":[
      "codebuild:BatchGetReportGroups",
      "codebuild:BatchGetReports",
      "codebuild:ListReportsForReportGroup",
      "codebuild:DescribeTestCases"],
    "Resource":"arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group"
  ]
}

```

3. 執行下列命令。

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://  
policy.json
```

相關服務

報告群組共用與 AWS Resource Access Manager (AWS RAM) 整合，這項服務可讓您與任何 AWS 帳戶或透過 共用 AWS 資源 AWS Organizations。與 共用您擁有的資源 AWS RAM，方法是建立資源共用，指定要共用的資源和取用者。消費者可以是 中的個別 AWS 帳戶 AWS Organizations、組織單位或整個組織 AWS Organizations。

如需詳細資訊，請參閱「[AWS RAM 使用者指南](#)」。

存取與您共用的報告群組

若要存取共用報告群組，消費者的 IAM 角色需要 BatchGetReportGroups 許可。您可以將下列政策連接至其 IAM 角色：

```
{  
  "Effect": "Allow",  
  "Resource": [  
    "*"   
  ],  
  "Action": [  
    "codebuild:BatchGetReportGroups"  
  ]  
}
```

如需詳細資訊，請參閱[使用的身分型政策 AWS CodeBuild](#)。

取消共用共用報告群組

唯有其擁有者才能存取已取消共享的報告群組，包括其報告及測試案例結果。如果您取消共用報告群組，您之前與其共用的任何 AWS 帳戶或使用者都無法存取報告群組、其報告或報告中測試案例的結果。

若要取消共享您擁有的已共用報告存取，您必須從資源共享中移除它。您可以使用 AWS RAM 主控台或 AWS CLI 來執行此操作。

取消共用您擁有的共用報告群組AWS RAM（主控台）

請參閱《AWS RAM 使用者指南》中的[更新資源共享](#)。

取消共用您擁有的共用報告群組 (AWS RAM 命令)

使用 [disassociate-resource-share](#) 命令。

取消共用您擁有的報告群組 (CodeBuild 命令)

執行 [delete-resource-policy](#) 命令並指定您要取消共享之報告群組的 ARN：

```
aws codebuild delete-resource-policy --resource-arn report-group-arn
```

識別共用報告群組

擁有者和取用者可以使用 AWS CLI 來識別共用報告群組。

若要識別並取得共用報告群組及其報告的資訊，請使用下列命令：

- 若要查看與您共用的報告群組 ARN，請執行 [list-shared-report-groups](#)：

```
aws codebuild list-shared-report-groups
```

- 若要查看在報告存取中的報告 ARN，請使用報告群組 ARN 執行 [list-reports-for-report-group](#)：

```
aws codebuild list-reports-for-report-group --report-group-arn report-group-arn
```

- 若要查看報告中測試案例的相關資訊，請使用報告 ARN 執行 [describe-test-cases](#)：

```
aws codebuild describe-test-cases --report-arn report-arn
```

輸出看起來如下：

```
{
  "testCases": [
    {
      "status": "FAILED",
      "name": "Test case 1",
      "expired": 1575916770.0,
      "reportArn": "report-arn",
      "prefix": "Cucumber tests for agent",
```

```
        "message": "A test message",
        "durationInNanoSeconds": 1540540,
        "testRawDataPath": "path-to-output-report-files"
    },
    {
        "status": "SUCCEEDED",
        "name": "Test case 2",
        "expired": 1575916770.0,
        "reportArn": "report-arn",
        "prefix": "Cucumber tests for agent",
        "message": "A test message",
        "durationInNanoSeconds": 1540540,
        "testRawDataPath": "path-to-output-report-files"
    }
]
```

共用報告群組的許可

擁有者的許可

報告群組的擁有者可以編輯報告群組，並在專案中指定該報告群組，以執行報告。

消費者的許可

報告群組消費者可以檢視報告群組、其報告，以及其報告的測試案例結果。消費者無法編輯報告群組或其報告，也無法用它來建立報告。

指定測試檔案

您可以在建置專案 `buildspec` 檔案的 `reports` 區段中，指定每個報告群組的測試結果檔案及其位置。如需詳細資訊，請參閱[Reports syntax in the buildspec file](#)。

以下是指定建置專案的兩個報告群組的範例 `reports` 區段。一個以其 ARN 指定，另一個則以名稱指定。`files` 區段指定包含測試案例結果的檔案。選用 `base-directory` 區段指定測試案例檔案所在的目錄。選用 `discard-paths` 區段指定是否捨棄上傳至 Amazon S3 儲存貯體之測試結果檔案的路徑。

```
reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
    #surefire junit reports
```

```
files:
  - '**/*'
base-directory: 'surefire/target/surefire-reports'
discard-paths: false

sampleReportGroup: #Cucumber reports from json plugin
files:
  - 'cucumber-json/target/cucumber-json-report.json'
file-format: CUCUMBERJSON #Type of the report, defaults to JUNITXML
```

指定測試命令

您可以在 `buildspec` 檔案的 `commands` 區段中指定執行測試案例的命令。這些命令會在 `buildspec` 檔案的 `reports` 區段中執行為報告群組指定的測試案例。以下範例 `commands` 區段包括要執行測試檔案中測試的命令：

```
commands:
  - echo Running tests for surefire junit
  - mvn test -f surefire/pom.xml -fn
  - echo
  - echo Running tests for cucumber with json plugin
  - mvn test -Dcucumber.options="--plugin json:target/cucumber-json-report.json" -f
    cucumber-json/pom.xml -fn
```

如需詳細資訊，請參閱[Buildspec 語法](#)。

在中標記報告群組 AWS CodeBuild

標籤是您或 AWS 指派給 AWS 資源的自訂屬性標籤。每個 AWS 標籤有兩個部分：

- 標籤鍵 (例如，`CostCenter`、`Environment`、`Project` 或 `Secret`)。標籤鍵會區分大小寫。
- 一個名為標籤值 (例如，`111122223333`、`Production` 或團隊名稱) 的選用欄位。忽略標籤值基本上等同於使用空字串。與標籤鍵相同，標籤值會區分大小寫。

這些合稱為鍵值組。對於報告群組上標籤數目的限制，以及標籤金鑰和值的限制，請參閱[Tags \(標籤\)](#)。

標籤可協助您識別和組織 AWS 資源。許多 AWS 服務支援標記，因此您可以將相同的標籤指派給來自不同服務的資源，以指出資源相關。例如，您可以將相同的標籤指派給指派給 Amazon S3 儲存貯體的 CodeBuild 報告群組。如需使用標籤的詳細資訊，請參閱[標記最佳實務](#)白皮書。

在 CodeBuild 中，主要資源是報告群組和專案。您可以使用 CodeBuild 主控台、AWS CLI、CodeBuild APIs 或 AWS SDKs 來新增、管理和移除報告群組的標籤。除了識別、組織和追蹤具有標籤的報告群組之外，您還可以在 IAM 政策中使用標籤，以協助控制誰可以檢視報告群組並與之互動。如需以標籤為基礎的存取政策範例，請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

主題

- [將標籤新增至報告群組](#)
- [檢視報告群組的標籤](#)
- [編輯報告群組的標籤](#)
- [從報告群組移除標籤](#)

將標籤新增至報告群組

將標籤新增至報告群組可協助您識別和組織資源，並管理對這些 AWS 資源的存取。首先，將一或多個標籤 (金鑰值對) 新增到報告群組。請記住，報告群組的標籤數目有所限制。金鑰和值欄位可使用的字數有所限制。如需詳細資訊，請參閱[Tags \(標籤\)](#)。擁有標籤後，您可以建立 IAM 政策，以根據這些標籤管理對報告群組的存取。您可以使用 CodeBuild 主控台或 AWS CLI 將標籤新增至報告群組。

Important

將標籤新增到報告群組可能會影響該報告群組的存取權。將標籤新增至報告群組之前，請務必檢閱任何可能使用標籤來控制對報告群組等資源的存取的 IAM 政策。如需以標籤為基礎的存取政策範例，請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

如需在建立政策時，將標籤新增至報告群組的詳細資訊，請參閱[建立報告群組 \(主控台\)](#)。

主題

- [將標籤新增至報告群組 \(主控台\)](#)
- [將標籤新增至報告群組 \(AWS CLI\)](#)

將標籤新增至報告群組 (主控台)

您可以使用 CodeBuild 主控台，將一或多個標籤新增至 CodeBuild 報告群組。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Report groups (報告群組) 中，選擇您要新增標籤的報告群組名稱。

3. 在導覽窗格中，選擇設定。
4. 如果沒有任何標籤新增至報告群組，請選擇 Add tag (新增標籤)。您也可以選擇 Edit (編輯)，然後選擇 Add tag (新增標籤)。
5. 在 Key (金鑰) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。
6. (選用) 若要新增另一個標籤，再選擇 Add tag (新增標籤) 一次。
7. 當您完成新增標籤的作業時，請選擇 Submit (提交)。

將標籤新增至報告群組 (AWS CLI)

若要在建立政策時，將標籤新增至報告群組，請參閱[建立報告群組 \(CLI\)](#)。在 `CreateReportGroup.json` 中，新增您的標籤。

若要將標籤新增至現有的報告群組，請參閱[更新報告群組 \(CLI\)](#)，並在 `UpdateReportGroupInput.json` 中新增標籤。

在這些步驟中，我們假設您已經安裝新版 AWS CLI 或更新到最新版本。如需詳細資訊，請參閱[安裝 AWS Command Line Interface](#)。

檢視報告群組的標籤

標籤可協助您識別和組織 AWS 資源，並管理對資源的存取。如需使用標籤的詳細資訊，請參閱[標記最佳實務](#)白皮書。如需以標籤為基礎的存取政策範例，請參閱[Deny or allow actions on report groups based on resource tags](#)。

檢視報告群組的標籤 (主控台)

您可以使用 CodeBuild 主控台來檢視與 CodeBuild 報告群組相關聯的標籤。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Report groups (報告群組) 中，選擇您要檢視標籤的報告群組名稱。
3. 在導覽窗格中，選擇設定。

檢視報告群組的標籤 (AWS CLI)

請依照下列步驟使用 AWS CLI 檢視報告群組的 AWS 標籤。如果未新增任何標籤，會傳回空白的標籤清單。

1. 使用 主控台或 AWS CLI 來尋找報告群組的 ARN。記下該 ARN。

AWS CLI

執行下列命令。

```
aws list-report-groups
```

此命令會傳回類似如下的 JSON 格式資訊：

```
{
  "reportGroups": [
    "arn:aws:codebuild:region:123456789012:report-group/report-group-1",
    "arn:aws:codebuild:region:123456789012:report-group/report-group-2",
    "arn:aws:codebuild:region:123456789012:report-group/report-group-3"
  ]
}
```

報告群組 ARN 以其名稱結尾，您可以用來識別報告群組的 ARN。

Console

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
 2. 在 Report groups (報告群組) 中，選擇帶有您要檢視之標籤的報告群組名稱。
 3. 在 Configuration (組態) 中，找出報告群組的 ARN。
2. 執行下列命令。使用您針對 `--report-group-arns` 參數記下的 ARN。

```
aws codebuild batch-get-report-groups --report-group-arns
arn:aws:codebuild:region:123456789012:report-group/report-group-name
```

如果成功，此命令會傳回 JSON 格式的資訊，其包含類似如下的 tags 區段：

```
{
  ...
  "tags": {
    "Status": "Secret",
    "Project": "TestBuild"
  }
  ...
}
```

編輯報表群組的標籤

您可以變更與報告群組相關聯標籤的值。您也可以變更金鑰的名稱，這相當於移除目前的標籤，並新增具有新名稱和與其他金鑰相同值的不同標籤。請記住金鑰和值欄位可使用的字元有所限制。如需詳細資訊，請參閱[Tags \(標籤\)](#)。

Important

編輯報告群組的標籤可能會影響該報告群組的存取權。在編輯報告群組標籤的名稱（索引鍵）或值之前，請務必檢閱任何可能使用標籤索引鍵或值的 IAM 政策，以控制對報告群組等資源的存取。如需以標籤為基礎的存取政策範例，請參閱[Deny or allow actions on report groups based on resource tags](#)。

編輯報告群組的標籤 (主控台)

您可以使用 CodeBuild 主控台來編輯與 CodeBuild 報告群組相關聯的標籤。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Report groups (報告群組) 中，選擇您要編輯標籤的報告群組名稱。
3. 在導覽窗格中，選擇設定。
4. 選擇編輯。
5. 執行以下任意一項：
 - 若要變更標籤，請在 Key (金鑰) 輸入新的名稱。變更標籤名稱等於移除標籤並使用新的金鑰名稱新增一個新的標籤。
 - 若要變更標籤的值，請輸入新的值。如果您想要變更值為沒有，請刪除目前的值並保留欄位空白。
6. 當您完成編輯標籤，選擇 Submit (提交)。

編輯報表群組的標籤 (AWS CLI)

若要新增、變更或刪除報告群組的標籤，請參閱[更新報告群組 \(CLI\)](#)。更新 UpdateReportGroupInput.json 中的標籤。

從報告群組移除標籤

您可以移除一或多個與報告群組相關聯的標籤。移除標籤不會從與該標籤相關聯的其他 AWS 資源中刪除標籤。

Important

移除報表群組的標籤可能會影響該報表群組的存取權。從報告群組移除標籤之前，請務必檢閱任何可能使用標籤金鑰或值的 IAM 政策，以控制對報告群組等資源的存取。如需以標籤為基礎的存取政策範例，請參閱[使用標籤控制對 AWS CodeBuild 資源的存取](#)。

從報告群組移除標籤 (主控台)

您可以使用 CodeBuild 主控台來移除標籤與 CodeBuild 報告群組之間的關聯。

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在 Report groups (報告群組) 中，選擇您要移除標籤的報告群組名稱。
3. 在導覽窗格中，選擇設定。
4. 選擇編輯。
5. 尋找您想要移除的標籤，然後選擇 Remove tag (移除標籤)。
6. 當您完成移除標籤，請選擇 Submit (提交)。

從報告群組移除標籤 (AWS CLI)

請依照下列步驟，使用 從 CodeBuild 報告群組 AWS CLI 中移除標籤。移除標籤並不會將其刪除，只會移除標籤和報告群組之間的關聯。

Note

如果您刪除 CodeBuild 報告群組，所有標籤關聯都會從已刪除的報告群組中移除。您不需要在刪除報告群組之前移除標籤。

若要將一或多個標籤從報告群組中刪除，請參閱[編輯報表群組的標籤 \(AWS CLI\)](#)。使用未包含您要刪除之標籤的已更新標籤清單，來更新 JSON 格式資料中的 tags 區段。如果您要刪除所有標籤，請將 tags 區段更新為：


```
"tags: []"
```

更新報告群組

當您更新報告群組時，您可以指定是否將原始測試結果資料匯出至 Amazon S3 儲存貯體中的檔案的相關資訊。如果您選擇匯出至 S3 儲存貯體，您可以針對報告群組指定下列項目：

- 原始測試結果檔案是否以 ZIP 檔案壓縮。
- 原始測試結果檔案是否加密。您可以使用下列其中一種形式來指定加密：
 - AWS 受管金鑰 適用於 Amazon S3 的。
 - 您建立和設定的客戶受管金鑰。

如需詳細資訊，請參閱[資料加密](#)。

如果您使用 AWS CLI 更新報告群組，您也可以更新或新增標籤。如需詳細資訊，請參閱[在中標記報告群組 AWS CodeBuild](#)。

Note

專案中指定的 CodeBuild 服務角色用於上傳到 S3 儲存貯體的許可。

主題

- [更新報告群組 \(主控台\)](#)
- [更新報告群組 \(CLI\)](#)

更新報告群組 (主控台)

使用下列程序來更新使用的報告群組 AWS 管理主控台。

更新報告群組

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Report groups (報告群組)。
3. 選擇您想要更新的報告群組。

4. 選擇編輯。
5. 選取或清除備份至 Amazon S3。如果選取此選項，請指定匯出設定：
 - a. 針對 S3 bucket name (S3 儲存貯體名稱)，輸入 S3 儲存貯體的名稱。
 - b. 針對 Path prefix (路徑前置詞)，輸入您想要在 S3 儲存貯體中上傳測試結果的路徑。
 - c. 選取 Compress test result data in a zip file (以 zip 檔案壓縮測試結果資料)，以壓縮原始測試結果資料檔案。
 - d. 展開 Additional configuration (其他組態) 以顯示加密選項。選擇下列其中一項：
 - 用於 AWS 受管金鑰 Amazon S3 的預設 AWS 受管金鑰。如需詳細資訊，請參閱AWS Key Management Service 《使用者指南》中的[客戶受管 CMKs](#)。這是預設加密選項。
 - 選擇自訂金鑰，以使用您建立和設定的客戶受管金鑰。針對AWS KMS 加密金鑰，輸入加密金鑰的 ARN。格式為 `arn:aws:kms:<region-id>: <aws-account-id>:key/<key-id>`。如需詳細資訊，請參閱AWS Key Management Service 《使用者指南》中的[建立 KMS 金鑰](#)。
 - Disable artifact encryption (停用成品) 可讓您停用加密。如果您想要共享測試結果，或發佈至靜態網站，您可以選擇此選項。(動態網站可以執行程式碼，以解密測試結果。)

更新報告群組 (CLI)

使用下列程序來更新使用 的報告群組 AWS CLI。

更新報告群組

1. 建立名為 UpdateReportGroupInput.json 的檔案。
2. 將以下內容複製到 UpdateReportGroupInput.json：

```
{
  "arn": "",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "bucket-name",
      "path": "path",
      "packaging": "NONE | ZIP",
      "encryptionDisabled": "false",
      "encryptionKey": "your-key"
    }
  }
}
```

```
    },
    "tags": [
      {
        "key": "tag-key",
        "value": "tag-value"
      }
    ]
  }
}
```

3. 在 `arn` 行中輸入您報告群組的 ARN (例如, "`arn:aws:codebuild:region:123456789012:report-group/report-group-1`").
4. 使用您要套用至報告群組的更新來更新 `UpdateReportGroupInput.json`.
 - 如果您要更新報告群組, 讓原始測試結果檔案匯出至 S3 儲存貯體, 請更新 `exportConfig` 區段。以您的 S3 儲存貯體名稱取代 `bucket-name`, 並以您想要在 S3 儲存貯體中匯出檔案的路徑取代 `path`。如果您想要壓縮匯出的檔案, 請為 `packaging` 指定 `ZIP`。否則請指定 `NONE`。使用 `encryptionDisabled` 指定是否要將匯出的檔案加密。如果您加密匯出的檔案, 請輸入您的客戶受管金鑰。
 - 如果您要更新報告群組, 讓報告群組不會匯出原始測試結果檔案至 S3 儲存貯體, 請使用下列 JSON 更新 `exportConfig` 區段:

```
{
  "exportConfig": {
    "exportConfigType": "NO_EXPORT"
  }
}
```

- 如果您要更新報告群組的標籤, 請更新 `tags` 區段。您可以變更、新增或移除標籤。如果您想移除所有標籤, 請使用以下 JSON 更新之:

```
"tags": []
```

5. 執行以下命令:

```
aws codebuild update-report-group \
--cli-input-json file://UpdateReportGroupInput.json
```

測試架構

本節中的主題示範如何 AWS CodeBuild 針對各種測試架構在 中設定測試報告。

主題

- [使用 Jasmine 設定測試報告](#)
- [使用 Jest 設定測試報告](#)
- [使用 pytest 設定測試報告](#)
- [使用 RSpec 設定測試報告](#)

使用 Jasmine 設定測試報告

下列程序示範如何使用 JasmineBDD 測試架構 AWS CodeBuild 在 中設定測試報告。 [JasmineBDD](#)

此程序需要下列先決條件：

- 您有現有的 CodeBuild 專案。
- 您的專案是設定為使用 Jasmine 測試框架的 Node.js 專案。

將 [jasmine-reporters](#) 套件新增至您專案的 `package.json` 檔案的 `devDependencies` 區段。此套件具備 JavaScript 報告程式類別的集合，可與 Jasmine 搭配使用。

```
npm install --save-dev jasmine-reporters
```

如果尚未存在，請將 `test` 指令碼新增至專案的 `package.json` 檔案中。`test` 指令碼可確保 `npm test` 在執行時呼叫 Jasmine。

```
{
  "scripts": {
    "test": "npx jasmine"
  }
}
```

CodeBuild 支援下列 Jasmine 測試報告程式：

JUnitXmlReporter

用於產生 JUnitXml 格式的報告。

NUnitXmlReporter

用於產生 NUnitXml 格式的報告。

預設情況下，使用 Jasmine 的 Node.js 專案會有 spec 子目錄，其包含 Jasmine 組態和測試指令碼。

若要將 Jasmine 設為產生 JUnitXML 格式的報告，請將以下程式碼新增至您的測試，以執行個體化 JUnitXmlReporter 報告程式。

```
var reporters = require('jasmine-reporters');

var junitReporter = new reporters.JUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(junitReporter);
```

若要將 Jasmine 設為產生 NUnitXML 格式的報告，請將以下程式碼新增至您的測試，以執行個體化 NUnitXmlReporter 報告程式。

```
var reporters = require('jasmine-reporters');

var nunitReporter = new reporters.NUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(nunitReporter)
```

測試報告會匯出至 <test report directory>/<report filename> 指定的檔案。

在您的 buildspec.yml 檔案中，新增/更新以下區段。

```
version: 0.2

phases:
  pre_build:
    commands:
```

```
- npm install
build:
  commands:
    - npm build
    - npm test

reports:
  jasmine_reports:
    files:
      - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```

如果您使用的是NunitXml報告格式，請將 file-format 值變更為以下內容。

```
file-format: NUNITXML
```

使用 Jest 設定測試報告

下列程序示範如何使用 Jest 測試架構 AWS CodeBuild 在 中設定測試報告。 <https://jestjs.io/>

此程序需要下列先決條件：

- 您有現有的 CodeBuild 專案。
- 您的專案是設定為使用 Jest 測試框架的 Node.js 專案。

將 [jest-junit](#) 套件新增至您專案的 package.json 檔案的 devDependencies 區段。CodeBuild 使用此套件來產生 JunitXml 格式的報告。

```
npm install --save-dev jest-junit
```

如果尚未存在，請將 test 指令碼新增至專案的 package.json 檔案中。test 指令碼可確保在 npm test 執行時呼叫 Jest。

```
{
  "scripts": {
    "test": "jest"
  }
}
```

將以下內容新增至您的 Jest 組態檔，以將 Jest 設為使用 JunitXml 報告程式。如果您的專案沒有 Jest 組態檔，請在您專案的根目錄中，建立一個名為 `jest.config.js` 的檔案，並新增以下內容。測試報告會匯出至 `<test report directory>/<report filename>` 指定的檔案。

```
module.exports = {
  reporters: [
    'default',
    [ 'jest-junit', {
      outputDirectory: <test report directory>,
      outputName: <report filename>,
    } ]
  ]
};
```

在您的 `buildspec.yml` 檔案中，新增/更新以下區段。

```
version: 0.2

phases:
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - npm build
      - npm test

reports:
  jest_reports:
    files:
      - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```

使用 pytest 設定測試報告

下列程序示範如何使用 pytest 測試架構 AWS CodeBuild 在 中設定測試報告。 <https://docs.pytest.org/>

此程序需要下列先決條件：

- 您有現有的 CodeBuild 專案。
- 您的專案是設定為使用 pytest 測試框架的 Python 專案。

將以下項目新增至 build 或 buildspec.yml 檔案的 post_build 階段。此程式碼會自動探索目前目錄中的測試，並將測試報告匯出至 *<test report directory>/<report filename>* 指定的檔案。報告使用 JunitXml 格式。

```
- python -m pytest --junitxml=<test report directory>/<report filename>
```

在您的 buildspec.yml 檔案中，新增/更新以下區段。

```
version: 0.2

phases:
  install:
    runtime-versions:
      python: 3.7
    commands:
      - pip3 install pytest
  build:
    commands:
      - python -m pytest --junitxml=<test report directory>/<report filename>

reports:
  pytest_reports:
    files:
      - <report filename>
    base-directory: <test report directory>
    file-format: JUNITXML
```

使用 RSpec 設定測試報告

下列程序示範如何使用 RSpec 測試架構 AWS CodeBuild 在 中設定測試報告。 [RSpec](#)

此程序需要下列先決條件：

- 您有現有的 CodeBuild 專案。
- 您的專案是設定為使用 RSpec 測試框架的 Ruby 專案。

在您的 buildspec.yml 檔案中，新增/更新以下內容。此程式碼會在 *<test source directory>* 目錄中執行測試，並將測試報告匯出至 *<test report directory>/<report filename>* 指定的檔案。報告使用 JunitXml 格式。


```
version: 0.2

phases:
  install:
    runtime-versions:
      ruby: 2.6
  pre_build:
    commands:
      - gem install rspec
      - gem install rspec_junit_formatter
  build:
    commands:
      - rspec <test source directory>/* --format RspecJunitFormatter --out <test report
        directory>/<report filename>
  reports:
    rspec_reports:
      files:
        - <report filename>
      base-directory: <test report directory>
      file-format: JUNITXML
```

檢視測試報告

您可以檢視測試報告的詳細資訊，例如其測試案例、成功和失敗次數，以及其執行時間的相關資訊。您可以檢視依建置執行、報告群組或 AWS 您的帳戶分組的測試報告。選擇主控台內的測試報告，以查看其詳細資訊及其測試案例的結果。

您可以查看尚未過期的測試報告。測試報告會在建立的 30 天後過期。您無法在 CodeBuild 中檢視過期的報告。

主題

- [檢視建置的測試報告](#)
- [檢視報告群組的測試報告](#)
- [檢視您 AWS 帳戶中的測試報告](#)

檢視建置的測試報告

檢視建置的測試報告

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 找到您想要檢視的建置。如果您知道是哪個專案執行建立測試報告的建置：
 1. 在導覽窗格中，選擇 Build projects (建置專案)，然後選擇其建置會執行您想要檢視之測試報告的專案。
 2. 選擇 Build history (建置歷史記錄)，然後選擇執行您要檢視之報告的建置。

您也可以在建置歷史記錄中，找到您 AWS 帳戶的建置。

1. 在導覽窗格中，選擇 Build history (建置歷史記錄)，然後選擇建立您想要檢視之測試報告的建置。
3. 在建置頁面中，選擇 Reports (報告)，然後選擇測試報告以查看其詳細資訊。

檢視報告群組的測試報告

檢視報告群組中的測試報告

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Report groups (報告群組)。
3. 選擇包含您想要檢視之測試報告的報告群組。
4. 選擇測試報告以查看其詳細資訊。

檢視您 AWS 帳戶中的測試報告

檢視您 AWS 帳戶中的測試報告

1. 在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Report history (報告歷史記錄)。

3. 選擇測試報告以查看其詳細資訊。

測試報告許可

本主題說明與測試報告相關的重要許可資訊。

主題

- [測試報告的 IAM 角色](#)
- [測試報告操作的許可](#)
- [測試報告許可範例](#)

測試報告的 IAM 角色

若要測試執行報告，並更新專案以包含測試報告，您的 IAM 角色需要下列許可。這些許可包含在預先定義的 AWS 受管政策中。如果您想要將測試報告新增至現有的建置專案，您必須自行新增這些許可。

- CreateReportGroup
- CreateReport
- UpdateReport
- BatchPutTestCases

若要執行程式碼涵蓋範圍報告，您的 IAM 角色也必須包含 BatchPutCodeCoverages 許可。

Note

BatchPutTestCases、UpdateReport、CreateReport 和 BatchPutCodeCoverages 不是公有許可。您無法針對這些許可呼叫對應的 AWS CLI 命令或 SDK 方法。

為了確保您擁有這些許可，您可以將下列政策連接至您的 IAM 角色：

```
{
  "Effect": "Allow",
  "Resource": [
    "*"
  ]
}
```

```
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

建議您將此政策限制為您必須使用的這些報告群組。在以下範例中，僅限具有政策中兩個 ARN 的報告群組才擁有許可：

```
{
  "Effect": "Allow",
  "Resource": [
    "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1",
    "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-2"
  ],
  "Action": [
    "codebuild:CreateReportGroup",
    "codebuild:CreateReport",
    "codebuild:UpdateReport",
    "codebuild:BatchPutTestCases",
    "codebuild:BatchPutCodeCoverages"
  ]
}
```

在以下範例中，僅限在執行名為 my-project 之專案的建置時建立的報告群組才擁有許可：

```
{
  "Effect": "Allow",
  "Resource": [
    "arn:aws:codebuild:your-region:your-aws-account-id:report-group/my-project-*"
  ],
  "Action": [
    "codebuild:CreateReportGroup",
    "codebuild:CreateReport",
    "codebuild:UpdateReport",
    "codebuild:BatchPutTestCases",
    "codebuild:BatchPutCodeCoverages"
  ]
}
```

```
]
}
```

Note

專案中指定的 CodeBuild 服務角色用於上傳到 S3 儲存貯體的許可。

測試報告操作的許可

您可以指定下列測試報告 CodeBuild API 操作的許可：

- BatchGetReportGroups
- BatchGetReports
- CreateReportGroup
- DeleteReportGroup
- DeleteReport
- DescribeTestCases
- ListReportGroups
- ListReports
- ListReportsForReportGroup
- UpdateReportGroup

如需詳細資訊，請參閱[AWS CodeBuild 許可參考](#)。

測試報告許可範例

如需與測試報告相關的範例政策資訊，請參閱以下相關資訊：

- [允許使用者變更報告群組](#)
- [允許使用者建立報告群組](#)
- [允許使用者刪除報告](#)
- [允許使用者刪除報告群組](#)
- [允許使用者取得報告群組的相關資訊](#)
- [允許使用者取得報告的相關資訊](#)

- [允許使用者取得報告群組的清單](#)
- [允許使用者取得報告的清單](#)
- [允許使用者取得報告群組的報告清單](#)
- [允許使用者取得報告的測試案例清單](#)

測試報告狀態

測試報告的狀態可以是下列其中一個：

- GENERATING：測試案例仍在執行中。
- DELETING：正在刪除測試報告。刪除測試報告時，也會刪除其測試案例。不會刪除匯出至 S3 儲存貯體的原始測試結果資料檔案。
- INCOMPLETE：測試報告未完成。傳回此狀態的原因可能是以下其中之一：
 - 指定報告測試案例的報告組態發生問題。例如，在 buildspec 檔案中的報告群組下方，測試案例的路徑可能不正確。
 - 執行建置的 IAM 使用者可能沒有執行測試的許可。如需詳細資訊，請參閱[測試報告許可](#)。
 - 建置因與測試無關的錯誤而未完成。
- SUCCEEDED：所有測試案例皆成功。
- FAILED：部分測試案例未成功。

每個測試案例都會傳送狀態。，測試案例的狀態可以是下列其中一個：

- SUCCEEDED：測試案例成功。
- FAILED：測試案例失敗。
- ERROR：測試案例導致意外的錯誤。
- SKIPPED：測試案例未執行。
- UNKNOWN：測試案例傳回 SUCCEEDED、FAILED、ERROR 或 SKIPPED 以外的狀態。

測試報告的測試案例結果的上限為 500 個。如果執行超過 500 個測試案例，CodeBuild 會以 狀態排定測試的優先順序，FAILED並截斷測試案例結果。

AWS CodeBuild 搭配 Amazon Virtual Private Cloud 使用

一般而言，AWS CodeBuild 無法存取 VPC 中的資源。若要啟用存取，您必須在 CodeBuild 專案組態中提供其他 VPC 特定的組態資訊。其中包括 VPC ID、VPC 子網路 ID 和 VPC 安全群組 ID。然後，已啟用 VPC 的建置就能夠存取 VPC 內的資源。如需在 Amazon VPC 中設定 VPC 的詳細資訊，請參閱 [Amazon VPC 使用者指南](#)。

主題

- [使用案例](#)
- [VPC 最佳實務](#)
- [VPCs的限制](#)
- [允許 CodeBuild 專案中的 Amazon VPC 存取](#)
- [對 VPC 設定進行故障診斷](#)
- [使用 VPC 端點](#)
- [AWS CodeBuild 搭配受管代理伺服器使用](#)
- [AWS CodeBuild 搭配代理伺服器使用](#)
- [CloudFormation VPC 範本](#)

使用案例

AWS CodeBuild 建置的 VPC 連線可讓您：

- 針對私有子網路上隔離的 Amazon RDS 資料庫中的資料，從您的組建執行整合測試。
- 直接從測試查詢 Amazon ElastiCache 叢集中的資料。
- 與 Amazon EC2、Amazon ECS 上託管的內部 Web 服務，或使用內部 Elastic Load Balancing 的服務互動。
- 從自我託管的內部成品儲存庫擷取相依性，例如 PyPI for Python、Maven for Java 和 npm for Node.js。
- 在設定為僅允許透過 Amazon VPC 端點存取的 S3 儲存貯體中存取物件。
- 透過與子網路相關聯的 NAT 閘道或 NAT 執行個體的彈性 IP 地址，查詢需要固定 IP 地址的外部 Web 服務。

您的組建可以存取您的 VPC 中託管的任何資源。

VPC 最佳實務

當您設定 VPC 以使用 CodeBuild 時，請使用此檢查清單。

- 使用公有和私有子網路以及 NAT 閘道來設定 VPC。NAT 閘道必須位於公有子網路中。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[含公有和私有子網路 \(NAT\) 的 VPC](#)。

Important

您需要 NAT 閘道或 NAT 執行個體，才能搭配 VPC 使用 CodeBuild，以便 CodeBuild 可以連線到公有端點（例如，在執行組建時執行 CLI 命令）。您無法使用網際網路閘道，而不是 NAT 閘道或 NAT 執行個體，因為 CodeBuild 不支援將彈性 IP 地址指派給其建立的網路介面，Amazon EC2 不支援針對在 Amazon EC2 執行個體啟動之外建立的任何網路介面自動指派公有 IP 地址。

- 將多個可用區域加入您的 VPC。
- 請確定您的安全群組不允許流量傳入 (進入) 您的組建。CodeBuild 對傳出流量沒有特定要求，但您必須允許存取建置所需的任何網際網路資源，例如 GitHub 或 Amazon S3。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[安全群組規則](#)。

- 為您的組建設定個別子網路。
- 當您設定 CodeBuild 專案以存取 VPC 時，請僅選擇私有子網路。

如需在 Amazon VPC 中設定 VPC 的詳細資訊，請參閱[Amazon VPC 使用者指南](#)。

如需使用 CloudFormation 設定 VPC 以使用 CodeBuild VPC 功能的詳細資訊，請參閱[CloudFormation VPC 範本](#)。

VPCs的限制

- 共用 VPC 不支援 CodeBuild VPCs 連線。

允許 CodeBuild 專案中的 Amazon VPC 存取

在您的 VPC 組態中包含這些設定：

- 針對 VPC ID，選擇 CodeBuild 使用的 VPC ID。

- 針對子網路，選擇具有 NAT 轉譯的私有子網路，其中包含或具有 CodeBuild 所用資源的路由。
- 針對安全群組，選擇 CodeBuild 用來允許存取 VPCs 中資源的安全群組。

若要使用主控台來建立建置專案，請參閱[建立組建專案 \(主控台\)](#)。當您建立或變更 CodeBuild 專案時，請在 VPC 中選擇您的 VPC ID、子網路和安全群組。

若要使用 AWS CLI 建立建置專案，請參閱[建立建置專案 \(AWS CLI\)](#)。如果您使用 AWS CLI 搭配 CodeBuild，則 CodeBuild 用來代表 IAM 使用者與服務互動的服務角色必須連接政策。如需相關資訊，請參閱[允許 CodeBuild 存取建立 VPC 網路介面所需的 AWS 服務](#)。

vpcConfig 物件應該包含您的 *vpcId*、*securityGroupIds* 和 *subnets*。

- *vpcId*：必要的。CodeBuild 使用的 VPC ID。執行此命令以取得您區域中所有 Amazon VPC IDs 的清單：

```
aws ec2 describe-vpcs
```

- *subnets*：必要。包含 CodeBuild 所用資源 IDs。執行此命令以取得這些 ID：

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

 Note

將 us-east-1 取代為您的區域。

- *securityGroupIds*：必要。CodeBuild 用來允許存取 VPCs 中資源的安全群組 IDs。執行此命令以取得這些 ID：

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

 Note

將 us-east-1 取代為您的區域。

對 VPC 設定進行故障診斷

請使用錯誤訊息中出現的資訊來協助您查明、診斷和解決問題。

以下是一些準則，可協助您疑難排解常見的 CodeBuild VPC 錯誤：Build does not have internet connectivity. Please check subnet network configuration.

1. [確定您的網際網路閘道連接至 VPC。](#)
2. [確定公有子網路的路由表指向網際網路閘道。](#)
3. [確定您的網路 ACL 允許流量流動。](#)
4. [確定您的安全群組允許流量流動。](#)
5. [為您的 NAT 閘道進行故障診斷。](#)
6. [確定私有子網路的路由表指向 NAT 閘道。](#)
7. 請確定 CodeBuild 用來代表 IAM 使用者與服務互動的服務角色具有[此政策](#)中的許可。如需詳細資訊，請參閱[允許 CodeBuild 與其他 AWS 服務互動](#)。

如果 CodeBuild 缺少許可，您可能會收到錯誤，指出 Unexpected EC2 error: UnauthorizedOperation。如果 CodeBuild 沒有使用 VPC 所需的 Amazon EC2 許可，則可能會發生此錯誤。

使用 VPC 端點

您可以將設定為 AWS CodeBuild 使用介面 VPC 端點，以改善建置的安全性。介面端點採用 PrivateLink，這項技術可讓您使用私有 IP 地址來私下存取 Amazon EC2 和 CodeBuild。PrivateLink 會將受管執行個體、CodeBuild 和 Amazon EC2 之間的所有網路流量限制在 Amazon 網路。(受管執行個體無法存取網際網路。) 此外，您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。您不需要 (但建議) 設定 PrivateLink。如需 PrivateLink 和 VPC 端點的詳細資訊，請參閱[什麼是 AWS PrivateLink？](#)。

建立 VPC 端點之前

設定 VPC 端點之前 AWS CodeBuild，請注意下列限制。

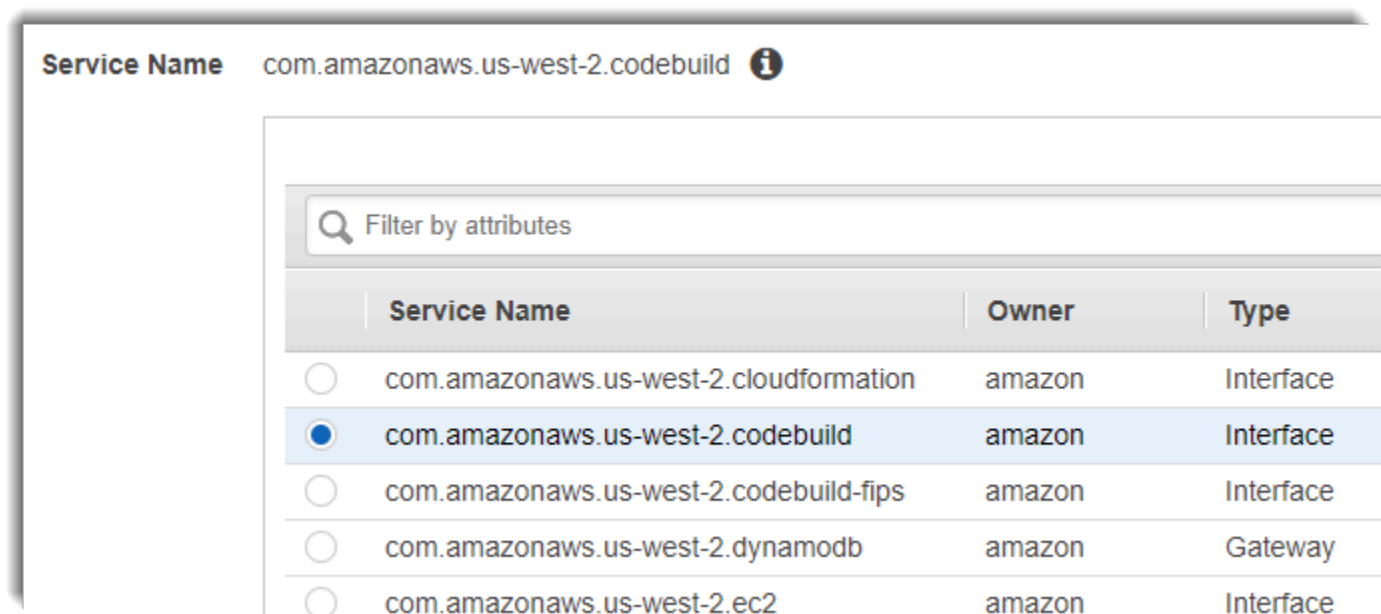
Note

如果您想要搭配不支援 Amazon VPC PrivateLink 連線 AWS 的服務使用 CodeBuild，請使用[NAT 閘道](#)。

- VPC 端點僅支援 Amazon 透過 Amazon Route 53 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 [DHCP 選項集](#)。
- VPC 端點目前不支援跨區域請求。請確定您在與存放建置輸入和輸出的任何 S3 儲存貯體相同的 AWS 區域中建立端點。您可以使用 Amazon S3 主控台或 [get-bucket-location](#) 命令來尋找儲存貯體的位置。使用區域特定的 Amazon S3 端點來存取您的儲存貯體（例如 `<bucket-name>.s3-us-west-2.amazonaws.com`）。如需 Amazon S3 區域特定端點的詳細資訊，請參閱《》中的 [Amazon Simple Storage Service](#) Amazon Web Services 一般參考。如果您使用 AWS CLI 向 Amazon S3 提出請求，請將預設區域設定為建立儲存貯體的相同區域，或在請求中使用 `--region` 參數。

為 CodeBuild 建立 VPC 端點

按照[建立界面端點](#)中的指示建立 `com.amazonaws.region.codebuild` 端點。這是的 VPC 端點 AWS CodeBuild。



region 代表 CodeBuild 支援的 AWS 區域的區域識別符，例如 `us-east-2` 美國東部（俄亥俄）區域。如需支援 AWS 區域的清單，請參閱《AWS 一般參考》中的 [CodeBuild](#)。端點會預先填入您在登入時指定的區域 AWS。如果變更您的區域，VPC 端點會隨之更新。

為 CodeBuild 建立 VPC 端點政策

您可以為 Amazon VPC 端點建立政策，您可以在 AWS CodeBuild 其中指定：

- 可執行動作的主體。

- 可執行的動作。
- 可對其執行動作的資源。

以下範例政策會指定所有委託人只能開始和檢視 project-name 專案的建置。

```
{
  "Statement": [
    {
      "Action": [
        "codebuild:ListBuildsForProject",
        "codebuild:StartBuild",
        "codebuild:BatchGetBuilds"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:codebuild:region-ID:account-ID:project/project-name",
      "Principal": "*"
    }
  ]
}
```

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[使用 VPC 端點控制對服務的存取](#)。

AWS CodeBuild 搭配受管代理伺服器使用

若要在受管代理伺服器中執行 AWS CodeBuild 預留容量機群，您必須設定代理伺服器，以允許或拒絕使用代理規則往返外部網站的流量。請注意，VPC、Windows 或 MacOS 不支援在受管代理伺服器中執行預留容量機群。

Important

根據機群中存在代理組態的持續時間，會產生額外的成本。如需詳細資訊，請參閱 <https://aws.amazon.com/codebuild/pricing/>。

主題

- [設定預留容量機群的受管代理組態](#)
- [執行 CodeBuild 預留容量機群](#)

設定預留容量機群的受管代理組態

若要為預留容量機群設定受管代理伺服器，您必須在主控制台或使用 [建立機群](#) 時啟用此功能 AWS CLI。您需要定義幾個屬性：

定義代理組態 - 選用

將網路存取控制套用至預留容量執行個體的代理組態。

預設行為

定義傳出流量的行為。

Allow

根據預設，允許傳出流量到所有目的地。

拒絕

根據預設，拒絕所有目的地的傳出流量。

Proxy 規則

指定要限制網路存取控制的目的地網域。

若要在主控制台中定義代理組態，請參閱 [建立預留容量機群](#) 以取得指示。若要使用 定義代理組態 AWS CLI，您可以修改下列 JSON 語法並儲存結果來執行此操作：

```
"proxyConfiguration": {
  "defaultBehavior": "ALLOW_ALL" | "DENY_ALL",
  "orderedProxyRules": [
    {
      "type": "DOMAIN" | "IP",
      "effect": "ALLOW" | "DENY",
      "entities": [
        "destination"
      ]
    }
  ]
}
```

您的 JSON 檔案看起來可能會類似以下內容：

```
"proxyConfiguration": {
```

```
"defaultBehavior": "DENY_ALL",
"orderedProxyRules": [
  {
    "type": "DOMAIN",
    "effect": "ALLOW",
    "entities": [
      "github.com"
    ]
  }
]
```

執行 CodeBuild 預留容量機群

使用受管代理伺服器執行 AWS CodeBuild 預留容量機群時，CodeBuild 會自動使用受管代理地址設定其 HTTP_PROXY 和 HTTPS_PROXY 環境變數。如果您的相依性軟體有自己的組態，且不遵守環境變數，您可以參考這些值，並在建置命令中更新您的軟體組態，以透過受管代理正確路由建置流量。如需詳細資訊，請參閱 [在 中建立建置專案 AWS CodeBuild](#) 及 [在 中變更建置專案設定 AWS CodeBuild](#)。

AWS CodeBuild 搭配代理伺服器使用

您可以使用 AWS CodeBuild 搭配代理伺服器來調節進出網際網路的 HTTP 和 HTTPS 流量。若要使用代理伺服器執行 CodeBuild，請在公有子網路中安裝代理伺服器，並在 VPC 中的私有子網路中安裝 CodeBuild。

在代理伺服器中執行 CodeBuild 有兩種主要使用案例：

- 不需要在您的 VPC 中使用 NAT 閘道或 NAT 執行個體。
- 可讓您指定代理伺服器中的執行個體可以存取的 URL，以及代理伺服器拒絕存取的 URL。

您可以搭配兩種類型的代理伺服器使用 CodeBuild。對於兩者，代理伺服器在公有子網路中執行，而 CodeBuild 在私有子網路中執行。

- 明確代理：如果您使用明確代理伺服器，您必須在專案層級的 CodeBuild 中設定 HTTP_PROXY、NO_PROXY 和 HTTPS_PROXY 環境變數。如需詳細資訊，請參閱 [在 中變更建置專案設定 AWS CodeBuild](#) 及 [在 中建立建置專案 AWS CodeBuild](#)。
- 透明代理：如果您使用透明代理伺服器，不需要特殊組態。

主題

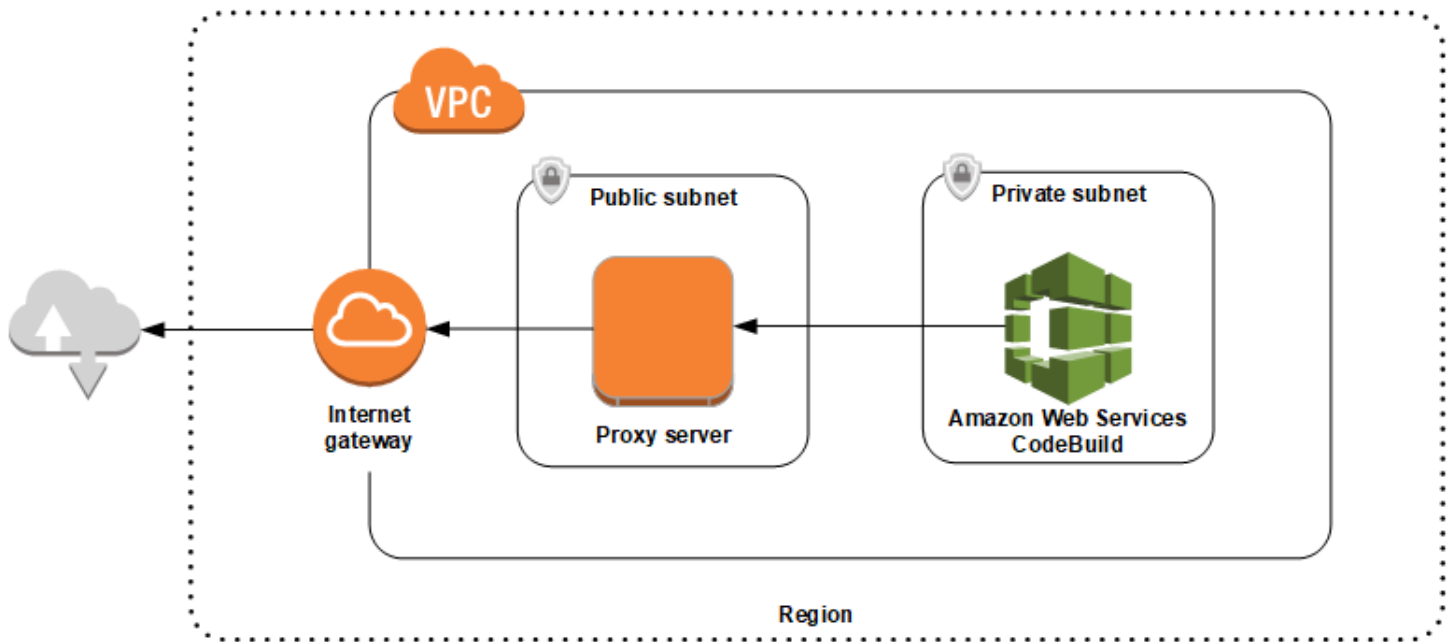
- [設定在代理伺服器中執行 CodeBuild 所需的元件](#)
- [在明確代理伺服器中執行 CodeBuild](#)
- [在透明代理伺服器中執行 CodeBuild](#)
- [在代理伺服器中執行套用管理員和其他工具](#)

設定在代理伺服器中執行 CodeBuild 所需的元件

您需要這些元件才能在透明或明確的代理伺服器 AWS CodeBuild 中執行：

- VPC。
- 您的 VPC 中一個用於代理伺服器的公有子網路。
- VPC for CodeBuild 中的一個私有子網路。
- 允許 VPC 與網際網路之間通訊的網際網路閘道。

下圖顯示元件如何互動。



設定 VPC、子網路和網路閘道

在透明或明確代理伺服器 AWS CodeBuild 中執行 需要下列步驟。

1. 建立 VPC。如需詳細資訊，請參閱《Amazon [VPC 使用者指南](#)》中的[建立 VPC](#)。

2. 在 VPC 中建立兩個子網路。一個是名為 Public Subnet 的公有子網路，您的代理伺服器在其中執行。另一個是名為 Private Subnet 的私有子網路，CodeBuild 在其中執行。

如需相關資訊，請參閱[在您的 VPC 中建立子網路](#)。

3. 建立網際網路閘道並連接到您的 VPC。如需詳細資訊，請參閱[建立和連接網際網路閘道](#)。
4. 在預設路由表中新增規則，將從 VPC (0.0.0.0/0) 傳出的流量路由傳送到網際網路閘道。如需相關資訊，請參閱[從路由表新增和移除路由](#)。
5. 在 VPC 的預設安全群組中新增規則，以允許來自 VPC (0.0.0.0/0) 的輸入 SSH 流量 (TCP 22)。
6. 請遵循《Amazon EC2 使用者指南》中的[使用啟動執行個體精靈](#)啟動執行個體中的指示來啟動 Amazon Linux 執行個體。當您執行精靈時，請選擇以下選項：
 - 在選擇執行個體類型中，選擇 Amazon Linux Amazon Machine Image (AMI)。
 - 在 Subnet (子網路) 中，選擇您先前在此主題中建立的公有子網路。如果您使用建議的名稱，則為公有子網路 (Public Subnet)。
 - 在 Auto-assign Public IP (自動指派公有 IP) 中，選擇 Enable (啟用)。
 - 在 Configure Security Group (設定安全群組) 頁面，針對 Assign a security group (指派安全群組)，選擇 Select an existing security group (選取現有的安全群組)。接下來，選擇預設安全群組。
 - 當您選擇 Launch (啟動) 之後，選擇現有的金鑰對或建立金鑰對。

對於其他所有選項，選擇預設設定。

7. 在您的 EC2 執行個體執行之後，停用來源/目的地檢查。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[停用來源/目的地檢查](#)。
8. 在您 VPC 中建立路由表。在路由表中新增規則，將流向網際網路的流量路由傳送到您的代理伺服器。將此路由表與您的私有子網路建立關聯。這是必要的，因此來自您私有子網路中執行 CodeBuild 之執行個體的傳出請求一律會透過代理伺服器路由。

安裝和設定代理伺服器

有許多代理伺服器可供選擇。這裡使用開放原始碼代理伺服器 Squid 來示範如何在代理伺服器中 AWS CodeBuild 執行。您可以將相同的概念套用到其他代理伺服器。

若要安裝 Squid，請執行下列命令來使用 yum 儲存庫：

```
sudo yum update -y
sudo yum install -y squid
```


安裝 Squid 之後，請按照本主題稍後的指示編輯其 `squid.conf` 檔案。

針對 HTTPS 流量來設定 Squid

對於 HTTPS，HTTP 流量會封裝在 Transport Layer Security (TLS) 連線中。Squid 使用稱為 [SslPeekAndSplice](#) 的功能，從包含所要求網際網路主機的 TLS 初始中，擷取伺服器名稱指示 (SNI)。需要如此，Squid 就不需要解密 HTTPS 流量。若要啟用 `SslPeekAndSplice`，Squid 需要憑證。使用 OpenSSL 建立此憑證：

```
sudo mkdir /etc/squid/ssl
cd /etc/squid/ssl
sudo openssl genrsa -out squid.key 2048
sudo openssl req -new -key squid.key -out squid.csr -subj "/C=XX/ST=XX/L=squid/O=squid/CN=squid"
sudo openssl x509 -req -days 3650 -in squid.csr -signkey squid.key -out squid.crt
sudo cat squid.key squid.crt | sudo tee squid.pem
```

Note

對於 HTTP，Squid 不需要設定。它可以從所有 HTTP/1.1 請求訊息中擷取主機標頭欄位，此欄位指定所要求的網際網路主機。

在明確代理伺服器中執行 CodeBuild

若要 AWS CodeBuild 在明確代理伺服器中執行，您必須設定代理伺服器以允許或拒絕進出外部網站的流量，然後設定 `HTTP_PROXY` 和 `HTTPS_PROXY` 環境變數。

主題

- [將 Squid 設定為明確代理伺服器](#)
- [建立 CodeBuild 專案](#)
- [明確代理伺服器 squid.conf 檔案範例](#)

將 Squid 設定為明確代理伺服器

若要將 Squid 代理伺服器設定為明確，您必須將其 `/etc/squid/squid.conf` 檔案修改如下：

- 移除以下預設的存取控制清單 (ACL) 規則。

```
acl localnet src 10.0.0.0/8
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
acl localnet src fc00::/7
acl localnet src fe80::/10
```

新增以下規則來取代您移除的預設 ACL 規則。第一行允許來自 VPC 的請求。接下來兩行授予許可給您的代理伺服器存取 AWS CodeBuild 可能使用的目的地 URL。編輯最後一行中的規則表達式，以在 AWS 區域中指定 S3 儲存貯體或 CodeCommit 儲存庫。例如：

- 如果您的來源是 Amazon S3，請使用命令 `acl download_src dstdom_regex .*s3\.us-west-1\.amazonaws\.com` 授予區域中 S3 儲存貯體的存取權 `us-west-1`。
- 如果您的來源是 AWS CodeCommit，請使用 `git-codecommit.<your-region>.amazonaws.com` 將 AWS 區域新增至允許清單。

```
acl localnet src 10.1.0.0/16 #Only allow requests from within the VPC
acl allowed_sites dstdomain .github.com #Allows to download source from GitHub
acl allowed_sites dstdomain .bitbucket.com #Allows to download source from Bitbucket
acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from
Amazon S3 or CodeCommit
```

- 將 `http_access allow localnet` 換成下列項目：

```
http_access allow localnet allowed_sites
http_access allow localnet download_src
```

- 如果您想要建置上傳日誌和成品，請執行以下其中一項：
- 在 `http_access deny all` 陳述式之前，插入以下陳述式。它們允許 CodeBuild 存取 CloudWatch 和 Amazon S3。需要存取 CloudWatch，以便 CodeBuild 可以建立 CloudWatch 日誌。需要存取 Amazon S3 才能上傳成品和 Amazon S3 快取。

```
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
```

```
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
```

- 儲存 之後 `squid.conf`，請執行下列命令：

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service squid restart
```

2. 將 `proxy` 新增至您的 `buildspec` 檔案。如需詳細資訊，請參閱[Buildspec 語法](#)。

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
      - command
```

Note

如果您收到 `RequestError` 逾時錯誤，請參閱[在代理伺服器中執行 CodeBuild 時發生 RequestError 逾時錯誤](#)。

如需詳細資訊，請參閱本主題後面部分的 [明確代理伺服器 squid.conf 檔案範例](#)。

建立 CodeBuild 專案

若要 AWS CodeBuild 使用明確代理伺服器執行，請使用您為代理伺服器建立的 EC2 執行個體私有 IP 地址 `HTTP_PROXY` 和專案層級的連接埠 3128 來設定其 `HTTP_PROXY` 環境變數。私有 IP 地址看起來像是 `http://your-ec2-private-ip-address:3128`。如需詳細資訊，請參閱[在 中建立建置專案 AWS CodeBuild](#)及[在 中變更建置專案設定 AWS CodeBuild](#)。

使用以下命令來檢視 Squid 代理存取日誌：

```
sudo tail -f /var/log/squid/access.log
```

明確代理伺服器 **squid.conf** 檔案範例

以下是為明確代理伺服器設定的 squid.conf 檔案的範例。

```
acl localnet src 10.0.0.0/16 #Only allow requests from within the VPC
# add all URLs to be whitelisted for download source and commands to be run in build
environment
acl allowed_sites dstdomain .github.com      #Allows to download source from github
acl allowed_sites dstdomain .bitbucket.com #Allows to download source from bitbucket
acl allowed_sites dstdomain ppa.launchpad.net #Allows to run apt-get in build
environment
acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from S3
or CodeCommit
acl SSL_ports port 443
acl Safe_ports port 80  # http
acl Safe_ports port 21  # ftp
acl Safe_ports port 443  # https
acl Safe_ports port 70  # gopher
acl Safe_ports port 210  # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280  # http-mgmt
acl Safe_ports port 488  # gss-http
acl Safe_ports port 591  # filemaker
acl Safe_ports port 777  # multiling http
acl CONNECT method CONNECT
#
# Recommended minimum Access Permission configuration:
#
# Deny requests to certain unsafe ports
http_access deny !Safe_ports
# Deny CONNECT to other than secure SSL ports
http_access deny CONNECT !SSL_ports
# Only allow cachemgr access from localhost
http_access allow localhost manager
http_access deny manager
# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
```

```
# from where browsing should be allowed
http_access allow localnet allowed_sites
http_access allow localnet download_src
http_access allow localhost
# Add this for CodeBuild to access CWL end point, caching and upload artifacts S3
bucket end point
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
# And finally deny all other access to this proxy
http_access deny all
# Squid normally listens to port 3128
http_port 3128
# Uncomment and adjust the following to add a disk cache directory.
#cache_dir ufs /var/spool/squid 100 16 256
# Leave coredumps in the first cache dir
coredump_dir /var/spool/squid
#
# Add any of your own refresh_pattern entries above these.
#
refresh_pattern ^ftp: 1440 20% 10080
refresh_pattern ^gopher: 1440 0% 1440
refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
refresh_pattern . 0 20% 4320
```

在透明代理伺服器中執行 CodeBuild

若要 AWS CodeBuild 在透明代理伺服器中執行，您必須將代理伺服器設定為可存取其互動的網站和網域。

主題

- [將 Squid 設定為透明代理伺服器](#)
- [建立 CodeBuild 專案](#)

將 Squid 設定為透明代理伺服器

若要將代理伺服器設定為透明，您必須授與它您希望其存取之網域和網站的存取權。若要 AWS CodeBuild 使用透明代理伺服器執行，您必須將其存取權授予 `amazonaws.com`。您也必須將存取權授予 CodeBuild 使用的其他網站。視您建立 CodeBuild 專案的方式而定，這些項目會有所不同。例如，GitHub、Bitbucket、Yum 和 Maven 等儲存庫的網站。若要授與 Squid 存取特定網域和網站，請使用類似以下的命令來更新 `squid.conf` 檔案。此命令範例會授予對 `amazonaws.com`、`github.com` 和 `bitbucket.com` 的存取權。您可以編輯此範例來授與存取其他網站。

```
cat | sudo tee /etc/squid/squid.conf #EOF
visible_hostname squid
#Handling HTTP requests
http_port 3129 intercept
acl allowed_http_sites dstdomain .amazonaws.com
#acl allowed_http_sites dstdomain domain_name [uncomment this line to add another
domain]
http_access allow allowed_http_sites
#Handling HTTPS requests
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl allowed_https_sites ssl::server_name .github.com
acl allowed_https_sites ssl::server_name .bitbucket.com
#acl allowed_https_sites ssl::server_name [uncomment this line to add another website]
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
http_access deny all
EOF
```

從私有子網路中的執行個體傳入的請求必須重新導向到 Squid 連接埠。Squid 會在連接埠 3129 (而不是 80) 接聽 HTTP 流量，在 3130 (而不是 443) 接聽 HTTPS 流量。使用 `iptables` 命令來路由流量：

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3129
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
```

```
sudo service iptables save
sudo service squid start
```

建立 CodeBuild 專案

設定代理伺服器後，您可以在私有子網路 AWS CodeBuild 中搭配使用代理伺服器，無需更多組態。每個 HTTP 和 HTTPS 請求會通過公有代理伺服器。使用以下命令來檢視 Squid 代理存取日誌：

```
sudo tail -f /var/log/squid/access.log
```

在代理伺服器中執行套用管理員和其他工具

使用下列程序在代理伺服器中執行套件管理員和其他工具。

在代理伺服器中執行工具，例如套件管理員

1. 在 squid.conf 檔案中新增陳述式，以將工具新增至代理伺服器的允許清單。
2. 在 buildspec 檔案中新增一行，指向您代理伺服器的私有端點。

下列範例示範如何對 apt-get、curl 和 maven 這樣做。如果您使用不同的工具，也適用同樣的原則。將其新增至 squid.conf 檔案中的允許清單，並將命令新增至您的 buildspec 檔案，讓 CodeBuild 了解代理伺服器的端點。

如何在代理伺服器中執行 **apt-get**

1. 在 squid.conf 檔案中新增以下陳述式，將 apt-get 新增至代理伺服器的允許清單。前三行允許在建置環境中 apt-get 執行。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required for apt-get to run in the
build environment
acl apt_get dstdom_regex .*\.launchpad.net # Required for CodeBuild to run apt-get
in the build environment
acl apt_get dstdom_regex .*\.ubuntu.com      # Required for CodeBuild to run apt-get
in the build environment
http_access allow localnet allowed_sites
http_access allow localnet apt_get
```

2. 在 buildspec 檔案中新增以下陳述式，讓 apt-get 命令在 /etc/apt/apt.conf.d/00proxy 中尋找代理組態。

```
echo 'Acquire::http::Proxy "http://<private-ip-of-proxy-server>:3128";  
Acquire::https::Proxy "http://<private-ip-of-proxy-server>:3128";  
Acquire::ftp::Proxy "http://<private-ip-of-proxy-server>:3128";' > /etc/apt/  
apt.conf.d/00proxy
```

如何在代理伺服器中執行 **curl**

1. 在 `squid.conf` 檔案中新增以下陳述式，將 `curl` 新增至建置環境中的允許清單。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the  
build environment  
acl allowed_sites dstdomain google.com # Required for access to a website. This  
example uses www.google.com.  
http_access allow localnet allowed_sites  
http_access allow localnet apt_get
```

2. 在 `buildspec` 檔案中新增以下陳述式，讓 `curl` 使用私有代理伺服器來存取您新增到 `squid.conf` 的網站。在此範例中，網站為 `google.com`。

```
curl -x <private-ip-of-proxy-server>:3128 https://www.google.com
```

如何在代理伺服器中執行 **maven**

1. 在 `squid.conf` 檔案中新增以下陳述式，將 `maven` 新增至建置環境中的允許清單。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the  
build environment  
acl maven dstdom_regex .*\.maven.org # Allows access to the maven repository in the  
build environment  
http_access allow localnet allowed_sites  
http_access allow localnet maven
```

2. 在您的 `buildspec` 檔案中新增以下陳述式。

```
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -  
DproxyPort=3128
```


CloudFormation VPC 範本

CloudFormation 可讓您以可預測且重複的方式建立和佈建 AWS 基礎設施部署，方法是使用範本檔案建立和刪除資源集合，做為單一單位 (堆疊)。如需詳細資訊，請參閱「[CloudFormation 使用者指南](#)」。

以下是 CloudFormation YAML 範本，用於設定要使用的 VPC AWS CodeBuild。此檔案也可在 [sample.zip](#) 中使用。

```
Description: This template deploys a VPC, with a pair of public and private subnets spread
              across two Availability Zones. It deploys an internet gateway, with a default
              route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
              and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the
    first Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the
    second Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in
    the first Availability Zone
    Type: String
    Default: 10.192.20.0/24
```

PrivateSubnet2CIDR:

Description: Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone

Type: String

Default: 10.192.21.0/24

Resources:**VPC:**

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: !Ref EnvironmentName

InternetGateway:

Type: AWS::EC2::InternetGateway

Properties:**Tags:**

- Key: Name

Value: !Ref EnvironmentName

InternetGatewayAttachment:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

InternetGatewayId: !Ref InternetGateway

VpcId: !Ref VPC

PublicSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

AvailabilityZone: !Select [0, !GetAZs '']

CidrBlock: !Ref PublicSubnet1CIDR

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: !Sub \${EnvironmentName} Public Subnet (AZ1)

PublicSubnet2:

Type: AWS::EC2::Subnet

Properties:

```
VpcId: !Ref VPC
AvailabilityZone: !Select [ 1, !GetAZs '' ]
CidrBlock: !Ref PublicSubnet2CIDR
MapPublicIpOnLaunch: true
Tags:
  - Key: Name
    Value: !Sub ${EnvironmentName} Public Subnet (AZ2)
```

```
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet1CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ1)
```

```
PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet2CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
```

```
NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
```

```
NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
```

```
NatGateway1:
  Type: AWS::EC2::NatGateway
```

Properties:

AllocationId: !GetAtt NatGateway1EIP.AllocationId
SubnetId: !Ref PublicSubnet1

NatGateway2:

Type: AWS::EC2::NatGateway

Properties:

AllocationId: !GetAtt NatGateway2EIP.AllocationId
SubnetId: !Ref PublicSubnet2

PublicRouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name
Value: !Sub \${EnvironmentName} Public Routes

DefaultPublicRoute:

Type: AWS::EC2::Route

DependsOn: InternetGatewayAttachment

Properties:

RouteTableId: !Ref PublicRouteTable
DestinationCidrBlock: 0.0.0.0/0
GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PublicRouteTable
SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PublicRouteTable
SubnetId: !Ref PublicSubnet2

PrivateRouteTable1:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name
Value: !Sub \${EnvironmentName} Private Routes (AZ1)

DefaultPrivateRoute1:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable1

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PrivateRouteTable1

SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name
Value: !Sub \${EnvironmentName} Private Routes (AZ2)

DefaultPrivateRoute2:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable2

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PrivateRouteTable2

SubnetId: !Ref PrivateSubnet2

NoIngressSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupName: "no-ingress-sg"

GroupDescription: "Security group with no ingress rule"

VpcId: !Ref VPC

Outputs:**VPC:**

Description: A reference to the created VPC

Value: !Ref VPC

PublicSubnets:

Description: A list of the public subnets

Value: !Join [",", [!Ref PublicSubnet1, !Ref PublicSubnet2]]

PrivateSubnets:

Description: A list of the private subnets

Value: !Join [",", [!Ref PrivateSubnet1, !Ref PrivateSubnet2]]

PublicSubnet1:

Description: A reference to the public subnet in the 1st Availability Zone

Value: !Ref PublicSubnet1

PublicSubnet2:

Description: A reference to the public subnet in the 2nd Availability Zone

Value: !Ref PublicSubnet2

PrivateSubnet1:

Description: A reference to the private subnet in the 1st Availability Zone

Value: !Ref PrivateSubnet1

PrivateSubnet2:

Description: A reference to the private subnet in the 2nd Availability Zone

Value: !Ref PrivateSubnet2

NoIngressSecurityGroup:

Description: Security group with no ingress rule

Value: !Ref NoIngressSecurityGroup

在中記錄和監控 AWS CodeBuild

記錄和監控是維護和 AWS 解決方案的可靠性、可用性 AWS CodeBuild 和效能的重要部分。您應該從 AWS 解決方案的所有部分收集監控資料，以便在發生多點失敗時更輕鬆地偵錯。AWS 提供下列工具來監控 CodeBuild 資源和組建，以及回應潛在事件。

主題

- [使用記錄 AWS CodeBuild API 呼叫 AWS CloudTrail](#)
- [使用 CloudWatch 監控 CodeBuild 建置](#)

使用記錄 AWS CodeBuild API 呼叫 AWS CloudTrail

AWS CodeBuild 已與服務整合 AWS CloudTrail，此服務提供由使用者、角色或 CodeBuild 中的 AWS 服務所採取之動作的記錄。CloudTrail 會將 CodeBuild 的所有 API 呼叫擷取為事件，包括從 CodeBuild 主控台以及從程式碼呼叫到 CodeBuild APIs 呼叫。如果您建立線索，則可以將 CloudTrail 事件持續交付至 S3 儲存貯體，包括 CodeBuild 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台的事件歷史記錄檢視最新事件。您可以使用 CloudTrail 所收集的資訊，判斷對 CodeBuild 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱「[AWS CloudTrail 使用者指南](#)」。

主題

- [關於 CloudTrail 中 AWS CodeBuild 的資訊](#)
- [關於 AWS CodeBuild 日誌檔案項目](#)

關於 CloudTrail 中 AWS CodeBuild 的資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 CodeBuild 中發生時，該活動會與事件歷史記錄中的其他服務 AWS 事件一起記錄在 CloudTrail 事件中。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱 AWS CloudTrail 使用者指南中的 [使用 CloudTrail 事件歷史記錄檢視事件](#)。

若要不持續記錄您 AWS 帳戶中的事件，包括 CodeBuild 的事件，請建立追蹤。線索可讓 CloudTrail 將日誌檔案傳遞到 S3 儲存貯體。根據預設，當您在主控台建立追蹤時，權杖會套用到所有區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案傳送到您指定的 S3 儲存貯體。您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中收集的事件資料。如需詳細資訊，請參閱：

- [建立追蹤的概觀](#)
- [CloudTrail 支援的服務和整合](#)
- [設定 CloudTrail 的 Amazon SNS 通知](#)
- [接收多個區域的 CloudTrail 日誌檔案](#)和[接收多個帳戶的 CloudTrail 日誌檔案](#)

CloudTrail 會記錄所有 CodeBuild 動作，並記錄在 [CodeBuild API 參考](#)中。例如，對 CreateProject (在 `create-project`) AWS CLI、StartBuild (在 `start-build`) AWS CLI、start-project) 和 UpdateProject (在 `update-project`) AWS CLI 動作的呼叫會在 CloudTrail 日誌檔案中產生項目。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或使用者憑證提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱AWS CloudTrail 《使用者指南》中的 [CloudTrail userIdentity 元素](#)。

關於 AWS CodeBuild 日誌檔案項目

權杖是一種組態，能讓事件以日誌檔案的形式交付至您指定的 S3 儲存貯體。CloudTrail 日誌檔案包含一個或多個日誌項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

Note

為了保護敏感資訊，CodeBuild 日誌中會隱藏下列項目：

- AWS 存取金鑰 IDs。如需詳細資訊，請參閱AWS Identity and Access Management 《使用者指南》中的[管理 IAM 使用者的存取金鑰](#)。
- 使用參數存放區指定的字串。如需詳細資訊，請參閱《Amazon EC2 Systems Manager Systems Manager 使用者指南》中的 [Systems Manager 參數存放區](#)和 [Systems Manager 參數存放區主控台演練](#)。
- 使用 指定的字串 AWS Secrets Manager。如需詳細資訊，請參閱[金鑰管理](#)。

下列範例顯示 CloudTrail 日誌項目，示範在 CodeBuild 中建立組建專案。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "FederatedUser",
    "principalId": "account-ID:user-name",
    "arn": "arn:aws:sts::account-ID:federated-user/user-name",
    "accountId": "account-ID",
    "accessKeyId": "access-key-ID",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2016-09-06T17:59:10Z"
      },
      "sessionIssuer": {
        "type": "IAMUser",
        "principalId": "access-key-ID",
        "arn": "arn:aws:iam::account-ID:user/user-name",
        "accountId": "account-ID",
        "userName": "user-name"
      }
    }
  },
  "eventTime": "2016-09-06T17:59:11Z",
  "eventSource": "codebuild.amazonaws.com",
  "eventName": "CreateProject",
  "awsRegion": "region-ID",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "user-agent",
  "requestParameters": {
    "awsActId": "account-ID"
  },
  "responseElements": {
    "project": {
      "environment": {
        "image": "image-ID",
        "computeType": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "name": "codebuild-demo-project",
      "description": "This is my demo project",
    }
  }
}
```

```
{
  "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project:project-ID",
  "encryptionKey": "arn:aws:kms:region-ID:key-ID",
  "timeoutInMinutes": 10,
  "artifacts": {
    "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket",
    "type": "S3",
    "packaging": "ZIP",
    "outputName": "MyOutputArtifact.zip"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
  "lastModified": "Sep 6, 2016 10:59:11 AM",
  "source": {
    "type": "GITHUB",
    "location": "https://github.com/my-repo.git"
  },
  "created": "Sep 6, 2016 10:59:11 AM"
}
},
"requestID": "9d32b228-745b-11e6-98bb-23b67EXAMPLE",
"eventID": "581f7dd1-8d2e-40b0-aeee-0dbf7EXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "account-ID"
}
```

使用 CloudWatch 監控 CodeBuild 建置

您可以使用 Amazon CloudWatch 來監看您的建置、在發生錯誤時回報，以及適時採取自動動作。您可以在兩個層級監控組建：

專案層級

這些指標適用於指定專案中的所有組建。若要查看專案的指標，請在 CloudWatch 中 ProjectName 為維度指定。

AWS 帳戶層級

這些指標適用於帳戶中的所有組建。若要查看帳戶層級的 AWS 指標，請勿在 CloudWatch 中輸入維度。AWS 帳戶層級無法使用建置資源使用率指標。

CloudWatch 指標會顯示隨著時間的推移，您建置的行為。例如，您可以監控：

- 隨著時間的推移，在組建專案或 AWS 帳戶中嘗試了多少組建。
- 隨著時間的推移，組建專案或 AWS 帳戶中有多少組建成功。
- 隨著時間的推移，組建專案或 AWS 帳戶中有多少組建失敗。
- CodeBuild 隨著時間在組建專案或 AWS 帳戶中執行組建所花費的時間。
- 建置建置或整個建置專案的資源使用率。建置資源使用率指標包括 CPU、記憶體和儲存使用率等指標。

如需詳細資訊，請參閱[檢視 CodeBuild 指標](#)。

CodeBuild CloudWatch 指標

您可以追蹤每個 AWS 帳戶或組建專案的下列指標。如需搭配 CodeBuild 使用 CloudWatch 的詳細資訊，請參閱[使用 CloudWatch 監控 CodeBuild 建置](#)。

BuildDuration

測量組建的 BUILD 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

Builds

測量已觸發的組建數目。

單位：Count

有效的 CloudWatch 統計資料：總和

DownloadSourceDuration

測量組建的 DOWNLOAD_SOURCE 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

持續時間

測量隨著時間所有組建的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

FailedBuilds

測量由於用戶端錯誤或逾時而失敗的建置數目。

單位：Count

有效的 CloudWatch 統計資料：總和

FinalizingDuration

測量組建的 FINALIZING 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

InstallDuration

測量組建的 INSTALL 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

PostBuildDuration

測量組建的 POST_BUILD 階段的持續時間

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

PreBuildDuration

測量組建的 PRE_BUILD 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

ProvisioningDuration

測量組建的 PROVISIONING 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

QueuedDuration

測量組建的 QUEUED 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

SubmittedDuration

測量組建的 SUBMITTED 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

SucceededBuilds

測量成功的組建數目。

單位：Count

有效的 CloudWatch 統計資料：總和

UploadArtifactsDuration

測量組建的 UPLOAD_ARTIFACTS 階段的持續時間。

單位：秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

CodeBuild CloudWatch 資源使用率指標

Note

CodeBuild 資源使用率指標僅適用於下列區域：

- 亞太區域 (東京) 區域
- 亞太區域 (首爾) 區域
- 亞太地區 (孟買) 區域
- 亞太區域 (新加坡) 區域
- 亞太區域 (雪梨) 區域

- 加拿大 (中部) 區域
- 歐洲 (法蘭克福) 區域
- 歐洲 (愛爾蘭) 區域
- 歐洲 (倫敦) 區域
- 歐洲 (巴黎) 區域
- 南美洲 (聖保羅) 區域
- 美國東部 (維吉尼亞北部) 區域
- 美國東部 (俄亥俄) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 美國西部 (奧勒岡) 區域

您可以追蹤下列資源使用率指標。如需搭配 CodeBuild 使用 CloudWatch 的詳細資訊，請參閱 [使用 CloudWatch 監控 CodeBuild 建置](#)。

CPUUtilized

建置容器使用的已配置處理 CPU 單位數量。

單位：CPU 單位

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

CPUUtilizedPercent

建置容器使用的已配置處理百分比。

單位：百分比

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

MemoryUtilized

組建容器使用的記憶體數。

單位：MB

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

MemoryUtilizedPercent

建置容器使用的配置記憶體百分比。

單位：百分比

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

StorageReadBytes

組建容器使用的儲存讀取速度。

單位：位元組/秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

StorageWriteBytes

組建容器使用的儲存寫入速度。

單位：位元組/秒

有效的 CloudWatch 統計資料：平均（建議）、最大值、最小值

CodeBuild CloudWatch 維度

CodeBuild 提供下列 CloudWatch 指標維度。如果未指定這些指標，則指標適用於目前 AWS 帳戶。

BuildId、BuildNumber、ProjectName

針對組建識別符、組建編號和專案名稱提供指標。

ProjectName

為專案名稱提供指標。

CodeBuild CloudWatch 警示

您可以使用 CloudWatch 主控台根據 CodeBuild 指標建立警示，以便在建置發生錯誤時做出反應。下列各點說明兩個對警示最有用的指標。如需搭配 CodeBuild 使用 CloudWatch 的詳細資訊，請參閱 [使用 CloudWatch 監控 CodeBuild 建置](#)。

- **FailedBuild。** 您可以在預定秒數內偵測到特定數量的失敗組建時，建立觸發的警示。在 CloudWatch 中，您可以指定觸發警示的秒數和失敗組建次數。
- **Duration。** 您可以建立在建置時間超過預期時觸發的警示。您可以指定於觸發警示之前，在組建開始後與組建完成前，必須經過的秒數。

如需如何為 CodeBuild 指標建立警示的資訊，請參閱 [使用 CloudWatch 警示監控 CodeBuild 組建](#)。如需警示的詳細資訊，請參閱《[Amazon CloudWatch 使用者指南](#)》中的 [建立 Amazon CloudWatch 警示](#)。Amazon CloudWatch

檢視 CodeBuild 指標

AWS CodeBuild 會代表您監控 函數，並透過 Amazon CloudWatch 報告指標。這些指標包含建置總數、失敗建置和成功建置，以及建置持續時間。

您可以使用 CodeBuild 主控台或 CloudWatch 主控台來監控 CodeBuild 的指標。下列程序說明如何檢視指標。

主題

- [檢視建置指標 \(CodeBuild 主控台\)](#)
- [檢視建置指標 \(Amazon CloudWatch 主控台\)](#)

檢視建置指標 (CodeBuild 主控台)

Note

您無法自訂用於在 CodeBuild 主控台中顯示指標的指標或圖形。如果您想要自訂顯示，請使用 Amazon CloudWatch 主控台來檢視您的建置指標。

帳戶層級指標

檢視 AWS 帳戶層級指標

1. 登入 AWS 管理主控台 並在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Account metrics (帳戶指標)。

專案層級指標

檢視專案層級指標

1. 登入 AWS 管理主控台 並在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。

2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 在組建專案清單中，於 Name (名稱) 欄中，選擇您想要檢視指標所在的專案。
4. 選擇指標標籤。

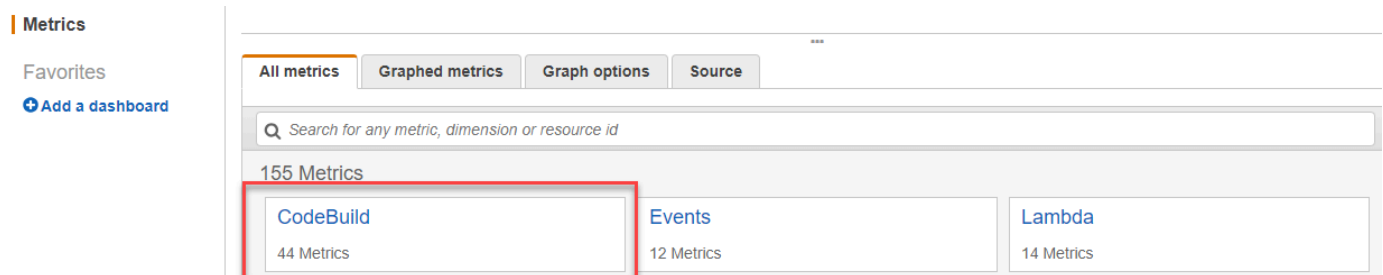
檢視建置指標 (Amazon CloudWatch 主控台)

您可以使用 CloudWatch 主控台自訂指標和用來顯示指標的圖形。

帳戶層級指標

檢視帳戶層級指標

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 指標。
3. 在 All metrics (所有指標) 標籤上，選擇 CodeBuild。

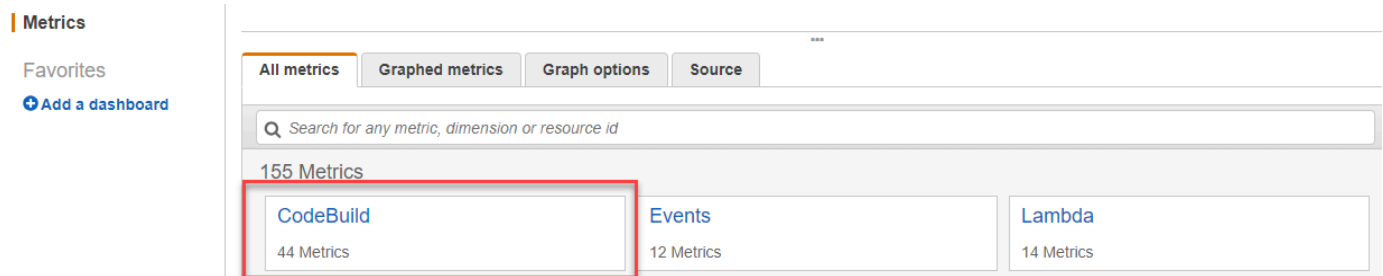


4. 選擇帳戶指標。
5. 選擇一或多個專案和指標。針對每個專案，您可以選擇 SucceededBuilds、FailedBuilds、Builds 和 Duration 指標。所有選取的專案和指標組合都會顯示在頁面的圖形中。

專案層級指標

檢視專案層級指標

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 指標。
3. 在 All metrics (所有指標) 標籤上，選擇 CodeBuild。



4. 選擇 By Project (依專案)。
5. 選擇一或多個專案和指標組合。針對每個專案，您可以選擇 SucceededBuilds、FailedBuilds、Builds 和 Duration 指標。所有選取的專案和指標組合都會顯示在頁面的圖形中。
6. (選用) 您可以自訂指標和圖形。例如，從統計資料欄中的下拉式清單中，您可以選擇要顯示的不同統計資料。或者，從下拉式功能表中，於 Period (期間) 欄中，您可以選擇用來監控指標的不同時間期間。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[圖形指標](#)和[檢視可用的指標](#)。

檢視 CodeBuild 資源使用率指標

AWS CodeBuild 會代表您監控建置資源使用率，並透過 Amazon CloudWatch 報告指標。這些包括 CPU、記憶體和儲存使用率等指標。

Note

CodeBuild 資源使用率指標只會針對執行超過一分鐘的組建進行記錄。

您可以使用 CodeBuild 主控台或 CloudWatch 主控台來監控 CodeBuild 的資源使用率指標。

Note

CodeBuild 資源使用率指標僅適用於下列區域：

- 亞太區域 (東京) 區域
- 亞太區域 (首爾) 區域
- 亞太地區 (孟買) 區域
- 亞太區域 (新加坡) 區域

- 亞太區域 (雪梨) 區域
- 加拿大 (中部) 區域
- 歐洲 (法蘭克福) 區域
- 歐洲 (愛爾蘭) 區域
- 歐洲 (倫敦) 區域
- 歐洲 (巴黎) 區域
- 南美洲 (聖保羅) 區域
- 美國東部 (維吉尼亞北部) 區域
- 美國東部 (俄亥俄) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 美國西部 (奧勒岡) 區域

下列程序說明如何存取資源使用率指標。

主題

- [存取資源使用率指標 \(CodeBuild 主控台 \)](#)
- [存取資源使用率指標 \(Amazon CloudWatch 主控台 \)](#)

存取資源使用率指標 (CodeBuild 主控台)

Note

您無法自訂用於在 CodeBuild 主控台中顯示指標的指標或圖形。如果您想要自訂顯示，請使用 Amazon CloudWatch 主控台來檢視您的建置指標。

專案層級資源使用率指標

存取專案層級的資源使用率指標

1. 登入 AWS 管理主控台 並在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 在建置專案清單中的名稱欄中，選擇您要檢視其使用率指標的專案。

4. 選擇指標標籤。資源使用率指標會顯示在資源使用率指標區段中。
5. 若要在 CloudWatch 主控台中檢視專案層級資源使用率指標，請在資源使用率指標區段中選擇在 CloudWatch 中檢視。

建置層級資源使用率指標

存取建置層級資源使用率指標

1. 登入 AWS 管理主控台 並在 <https://console.aws.amazon.com/codesuite/codebuild/home> 開啟 AWS CodeBuild 主控台。
2. 在導覽窗格中，選擇 Build history (組建歷史記錄)。
3. 在組建清單中的組建執行欄中，選擇您要檢視其使用率指標的組建。
4. 選擇資源使用率索引標籤。
5. 若要在 CloudWatch 主控台中檢視建置層級資源使用率指標，請在資源使用率指標區段中選擇在 CloudWatch 中檢視。

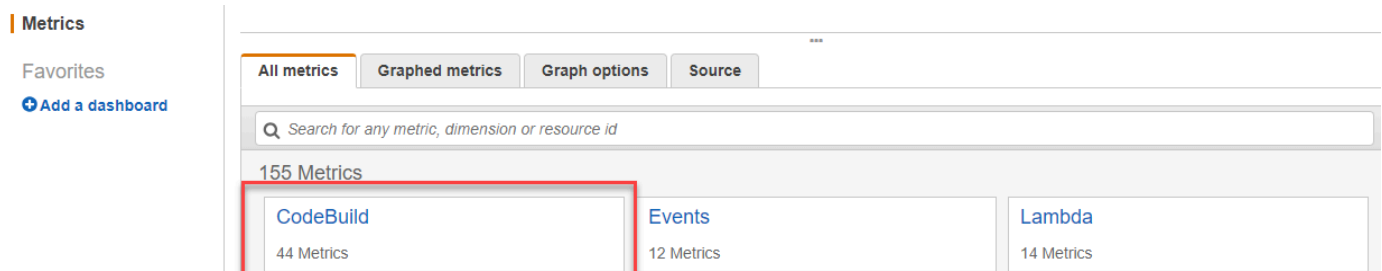
存取資源使用率指標 (Amazon CloudWatch 主控台)

Amazon CloudWatch 主控台可用來存取 CodeBuild 資源使用率指標。

專案層級資源使用率指標

存取專案層級的資源使用率指標

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 指標。
3. 在 All metrics (所有指標) 標籤上，選擇 CodeBuild。



4. 選擇 By Project (依專案)。

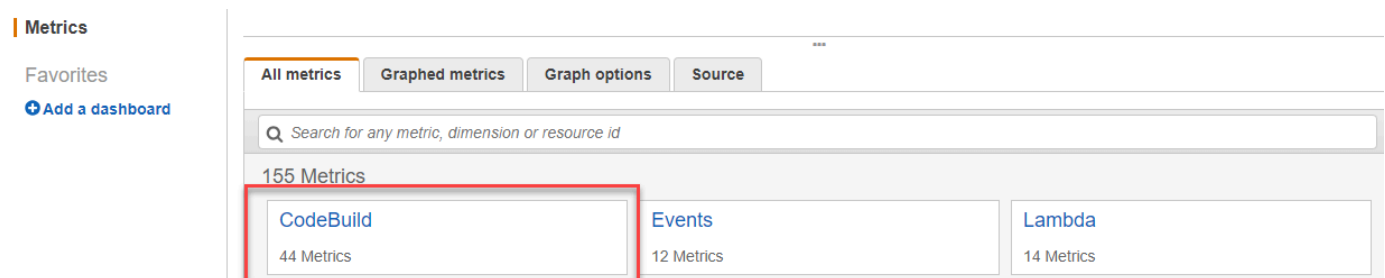
5. 選擇要新增至圖形的一或多個專案和指標組合。所有選取的專案和指標組合都會顯示在頁面的圖形中。
6. (選用) 您可以從圖形化指標索引標籤自訂指標和圖形。例如，從統計資料欄中的下拉式清單中，您可以選擇要顯示的不同統計資料。或者，從下拉式功能表中，於 Period (期間) 欄中，您可以選擇用來監控指標的不同時間期間。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[繪製指標](#)圖形和[檢視可用的指標](#)。

建置層級資源使用率指標

存取建置層級資源使用率指標

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 指標。
3. 在 All metrics (所有指標) 標籤上，選擇 CodeBuild。



4. 選擇 BuildId、BuildNumber、ProjectName。
5. 選擇要新增至圖形的一或多個組建和指標組合。所有選取的組建和指標組合都會顯示在頁面上的圖形中。
6. (選用) 您可以從圖形化指標索引標籤自訂指標和圖形。例如，從統計資料欄中的下拉式清單中，您可以選擇要顯示的不同統計資料。或者，從下拉式功能表中，於 Period (期間) 欄中，您可以選擇用來監控指標的不同時間期間。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[繪製指標](#)圖形和[檢視可用的指標](#)。

使用 CloudWatch 警示監控 CodeBuild 組建

您可以為組建建立 CloudWatch 警示。警示會監看您指定時間段的單個指標，然後根據幾個時間段內與指定閾值相關的指標值來執行一或多個動作。使用原生 CloudWatch 警示功能，您可以在超過閾值時指定 CloudWatch 支援的任何動作。例如，您可以指定在帳戶中有三個以上的組建在十五分鐘內失敗時傳送 Amazon SNS 通知。

為 CodeBuild 指標建立 CloudWatch 警示

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Alarms (警示)。
3. 選擇建立警示。
4. 在 CloudWatch Metrics by Category (依類別的 CloudWatch 指標) 下，選擇 CodeBuild Metrics (CodeBuild 指標)。如果您只需要專案層級指標，請選擇 By Project (依專案)。如果您只需要帳戶層級指標，請選擇 Account Metrics (帳戶指標)。
5. 在 Create Alarm (建立警示) 上，如果尚未選取指標，請選擇 Select Metric (選擇指標)。
6. 選擇您要為其建立警示的指標。選項為 By Project (依專案) 或 Account Metrics (帳戶指標)。
7. 選擇 Next (下一步) 或 Define Alarm (定義警示)，然後建立警示。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的[建立 CloudWatch 警示](#)。如需在觸發警示時設定 Amazon SNS 通知的詳細資訊，請參閱《[Amazon SNS 開發人員指南](#)》中的[設定 Amazon SNS 通知](#)。Amazon SNS
8. 選擇建立警示。

中的安全性 AWS CodeBuild

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構是為了符合最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模型](#) 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 – AWS 負責保護在 中執行 AWS 服務的基礎設施 AWS 雲端。AWS 也為您提供可安全使用的服務。在 [AWS 合規計劃](#) 中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用的合規計劃 AWS CodeBuild，請參閱[AWS 合規計劃範圍內的服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 CodeBuild 時套用共同責任模型。下列主題說明如何設定 CodeBuild 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 CodeBuild 資源。

主題

- [中的資料保護 AWS CodeBuild](#)
- [中的身分和存取管理 AWS CodeBuild](#)
- [的合規驗證 AWS CodeBuild](#)
- [中的彈性 AWS CodeBuild](#)
- [中的基礎設施安全 AWS CodeBuild](#)
- [在 CodeBuild 中存取您的來源提供者](#)
- [預防跨服務混淆代理人](#)

中的資料保護 AWS CodeBuild

將 AWS [共同責任模式](#) 應用於 AWS CodeBuild 中的資料保護。如此模型所述，AWS 負責保護執行所有的全球基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱[資料隱私權常見問答集](#)。如需歐洲資料保護的詳細資訊，請參閱[一般資料保護規則 \(GDPR\) 中心](#)。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《使用者指南》中的[使用 CloudTrail 追蹤](#)。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱[聯邦資訊處理標準 \(FIPS\) 140-3](#)。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 CodeBuild 或使用主控台、API AWS CLI或其他 AWS 服務 AWS SDKs 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

為了保護敏感資訊，CodeBuild 日誌中會隱藏下列項目：

- 使用 CodeBuild 專案環境變數中的參數存放區或 buildspec env/parameter-store區段指定的字串。如需詳細資訊，請參閱《Amazon EC2 Systems Manager [Systems Manager 使用者指南](#)》中的 [Systems Manager 參數存放區](#)和 [Systems Manager 參數存放區主控台演練](#)。
- 在 CodeBuild 專案環境變數或 buildspec env/secrets-manager區段 AWS Secrets Manager 中使用 指定的字串。如需詳細資訊，請參閱[金鑰管理](#)。

Note

遮罩符合存放在 Amazon EC2 Systems Manager 參數存放區 或 中的確切值 AWS Secrets Manager。如果組建命令在秘密出現在日誌之前轉換秘密，則結果會是未遮罩的不同字串。例如，使用 Base64 編碼、反轉、變更大小寫，或將其與其他文字串連，都會建立不同的值，並會出現在建置日誌中。

若要讓值保持遮罩，請以建置使用的格式存放。例如，如果您的建置需要 Base64-encoded 的值，請將 Base64-encoded字串直接存放在 AWS Secrets Manager 或 Amazon EC2 Systems Manager 參數存放區中。如此一來，記錄的值就會符合儲存的值。

如需資料保護的詳細資訊，請參閱安全部落格上的[AWS 共同責任模型和 GDPR](#) AWS 部落格文章。

主題

- [資料加密](#)
- [金鑰管理](#)
- [流量隱私權](#)

資料加密

加密是 CodeBuild 安全性的重要部分。有些加密，例如傳輸中的資料，會預設提供，並且您不需要採取任何動作。其他加密，例如靜態資料，則可以在建立您的專案或組建時設定。

- 靜態資料加密 - 依預設會使用 加密建置成品，例如快取、日誌、匯出的原始測試報告資料檔案和建置結果 AWS 受管金鑰。如果您不想使用這些 KMS 金鑰，則必須建立和設定客戶受管金鑰。如需詳細資訊，請參閱AWS Key Management Service 《使用者指南》中的[建立 KMS 金鑰](#)和[AWS 金鑰管理服務概念](#)。
- 您可以存放 CodeBuild 用來加密CODEBUILD_KMS_KEY_ID環境變數中建置輸出成品的 AWS KMS 金鑰識別符。如需詳細資訊，請參閱[建置環境中的環境變數](#)
- 您可以在建立建置專案時指定客戶受管金鑰。如需詳細資訊，請參閱 [Set the Encryption Key Using the Console](#)和[使用 CLI 設定加密金鑰](#)。

根據預設，建置機群的 Amazon Elastic Block Store 磁碟區會使用 加密 AWS 受管金鑰。

- 加密傳輸中的資料 - 客戶與 CodeBuild 之間以及 CodeBuild 與其下游相依性之間的所有通訊都會使用使用 Signature 第 4 版簽署程序簽署的 TLS 連線進行保護。所有 CodeBuild 端點都使用由管理的 SHA-256 憑證 AWS 私有憑證授權單位。如需詳細資訊，請參閱[簽章版本 4 簽署程序](#)和[什麼是 ACM PCA](#)。
- 建置成品加密 - 與建置專案相關聯的 CodeBuild 服務角色需要存取 KMS 金鑰，才能加密其建置輸出成品。根據預設，CodeBuild 會在您的帳戶中使用 AWS 受管金鑰 適用於 Amazon S3 的 AWS 。如果您不想使用此 AWS 受管金鑰，則必須建立和設定客戶受管金鑰。如需詳細資訊，請參閱《AWS KMS 開發人員指南》中的 [加密建置輸出](#)和[建立金鑰](#)。

金鑰管理

您可以藉由加密保護您的內容不遭到未經授權的使用。將您的加密金鑰存放在 中 AWS Secrets Manager，然後授予與建置專案相關聯的 CodeBuild 服務角色，以從您的 Secrets Manager 帳戶取得

加密金鑰的許可。如需詳細資訊，請參閱[使用客戶受管金鑰加密建置輸出](#)、[在中建立建置專案 AWS CodeBuild](#)、[手動執行 AWS CodeBuild 組建](#)和[教學：儲存並擷取秘密](#)。

在建置命令中使用CODEBUILD_KMS_KEY_ID環境變數來取得 AWS KMS 金鑰識別符。如需詳細資訊，請參閱[建置環境中的環境變數](#)。

您可以使用 Secrets Manager 保護私有登錄檔的登入資料，該登錄檔存放用於執行期環境的 Docker 映像。如需詳細資訊，請參閱[具有 CodeBuild AWS Secrets Manager 範例的私有登錄檔](#)。

流量隱私權

您可以將 CodeBuild 設定為使用介面 VPC 端點，以改善建置的安全性。若要執行此動作，您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。您也不需要設定 PrivateLink，不過這是建議的行為。如需詳細資訊，請參閱[使用 VPC 端點](#)。如需 PrivateLink 和 VPC 端點的詳細資訊，請參閱 [AWS PrivateLink](#) 和[透過 PrivateLink AWS 存取服務](#)。

中的身分和存取管理 AWS CodeBuild

存取 AWS CodeBuild 需要登入資料。這些登入資料必須具有存取 AWS 資源的許可，例如在 S3 儲存貯體中儲存和擷取組建成品，以及檢視組建的 Amazon CloudWatch Logs。下列各節說明如何使用 [AWS Identity and Access Management](#)(IAM) 和 CodeBuild 來協助安全存取您的 資源：

管理 AWS CodeBuild 資源存取許可的概觀

每個 AWS 資源都由 AWS 帳戶擁有，而建立或存取資源的許可是由許可政策管理。帳戶管理員可以將許可政策連接到 IAM 身分 (即使用者、群組和角色)。

Note

帳戶管理員 (或管理員使用者) 是具有管理員權限的使用者。如需詳細資訊，請參 [《IAM 使用者指南》](#) 中的 IAM 最佳實務。

當您授予許可時，您要決定取得許可的人員、其可存取的資源、以及可對這些資源執行的動作。

主題

- [AWS CodeBuild 資源和操作](#)
- [了解資源所有權](#)
- [管理 資源的存取](#)

- [指定政策元素：動作、效果和委託人](#)

AWS CodeBuild 資源和操作

在 中 AWS CodeBuild，主要資源是建置專案。在政策中，您使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。組建也是資源，並且有與其關聯的 ARN。如需詳細資訊，請參閱《》中的 [Amazon Resource Names \(ARN\) AWS 和服務命名空間](#) Amazon Web Services 一般參考。

Resource Type (資源類型)	ARN 格式
組建專案	arn:aws:codebuild: <i>region-ID</i> : <i>account-ID</i> :project/ <i>project-name</i>
建置	arn:aws:codebuild: <i>region-ID</i> : <i>account-ID</i> :build/ <i>build-ID</i>
報告群組	arn:aws:codebuild: <i>region-ID</i> : <i>account-ID</i> :report-group/ <i>report-group-name</i>
報告	arn:aws:codebuild: <i>region-ID</i> : <i>account-ID</i> :report/ <i>report-ID</i>
機群	arn:aws:codebuild: <i>region-ID</i> : <i>account-ID</i> :fleet/ <i>fleet-ID</i>
所有 CodeBuild 資源	arn:aws:codebuild:*
指定區域中指定帳戶擁有的所有 CodeBuild 資源 AWS	arn:aws:codebuild: <i>region-ID</i> : <i>account-ID</i> :*

Important

使用預留容量功能時，同一帳戶中的其他專案可以存取機群執行個體上快取的資料，包括來源檔案、Docker layer 和 buildspec 中指定的快取目錄。這是根據設計，並允許相同帳戶中的專案共用機群執行個體。

 Note

大多數 AWS 服務會將冒號 (:) 或正斜線 (/) 視為 ARNs 中的相同字元。不過，CodeBuild 在資源模式和規則中使用完全相符的項目。在建立事件模式時，請務必使用正確的字元，使這些字元符合資源中的 ARN 語法。

例如，您可以在陳述式中使用其 ARN 指定特定的組建專案 (*myBuildProject*)，如下所示：

```
"Resource": "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject"
```

若要指定所有資源，或如果 API 動作不支援 ARN，請在 Resource 元素中使用萬用字元 (*)，如下所示：

```
"Resource": "*"
```

有些 CodeBuild API 動作接受多個資源（例如 BatchGetProjects）。若要在單一陳述式中指定多個資源，請用逗號分隔他們的 ARN，如下所示：

```
"Resource": [  
  "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject",  
  "arn:aws:codebuild:us-east-2:123456789012:project/myOtherBuildProject"  
]
```

CodeBuild 提供一組操作來使用 CodeBuild 資源。如需清單，請參閱[AWS CodeBuild 許可參考](#)。

了解資源所有權

無論誰建立資源，AWS 帳戶都會擁有在帳戶中建立的資源。具體而言，資源擁有者是驗證資源建立請求的[委託人實體](#) AWS 帳戶（即根帳戶、使用者或 IAM 角色）。下列範例說明其如何運作：

- 如果您使用 AWS 帳戶的根帳戶登入資料來建立規則，AWS 您的帳戶即為 CodeBuild 資源的擁有者。
- 如果您在 AWS 帳戶中建立使用者，並將建立 CodeBuild 資源的許可授予該使用者，則使用者可以建立 CodeBuild 資源。不過，使用者所屬 AWS 的帳戶擁有 CodeBuild 資源。
- 如果您在 AWS 帳戶中建立具有建立 CodeBuild 資源許可的 IAM 角色，則任何可以擔任該角色的人都可以建立 CodeBuild 資源。該角色所屬 AWS 的帳戶擁有 CodeBuild 資源。

管理 資源的存取

許可政策說明誰可以存取哪些資源。

Note

本節討論如何在 AWS CodeBuild 中使用 IAM。它不提供 IAM 服務的詳細資訊。如需完整的 IAM 文件，請參閱 IAM 使用者指南中的 [什麼是 IAM](#)。如需有關 IAM 政策語法和說明的資訊，請參閱 IAM 使用者指南中的 [AWS IAM 政策參考](#)。

連接到 IAM 身分的政策稱為身分類型政策 (IAM 政策)。連接到資源的政策稱為資源型政策。CodeBuild 支援特定唯讀 APIs 的身分型政策和資源型政策，以用於跨帳戶資源共用。

安全存取 S3 儲存貯體

我們強烈建議您在 IAM 角色中包含下列許可，以確認與您 CodeBuild 專案相關聯的 S3 儲存貯體是由您或您信任的人所擁有。這些許可不包含在 AWS 受管政策和角色中。您必須自行新增這些許可。

- `s3:GetBucketAcl`
- `s3:GetBucketLocation`

如果專案使用的 S3 儲存貯體擁有者變更，您必須驗證您仍然擁有儲存貯體，如果沒有，則更新 IAM 角色中的許可。如需詳細資訊，請參閱 [允許使用者與 CodeBuild 互動](#) 及 [允許 CodeBuild 與其他 AWS 服務互動](#)。

指定政策元素：動作、效果和委託人

對於每個 AWS CodeBuild 資源，服務會定義一組 API 操作。若要授予這些 API 操作的許可，CodeBuild 會定義一組您可以在政策中指定的動作。為了執行 API 操作，某些 API 操作可能需要多個動作的許可。如需詳細資訊，請參閱 [AWS CodeBuild 資源和操作](#) 及 [AWS CodeBuild 許可參考](#)。

以下是基本的政策元素：

- 資源 - 您使用 Amazon Resource Name (ARN) 識別欲套用政策的資源。
- 動作 - 您可以使用動作關鍵字來識別您要允許或拒絕的資源操作。例如，`codebuild:CreateProject` 許可授予使用者執行 `CreateProject` 操作的許可。

- 效果 – 當使用者請求動作時，您可以指定允許或拒絕的效果。如果您未明確授予存取 (允許) 資源，則隱含地拒絕存取。您也可以明確拒絕存取資源。您可以明確拒絕以確定即使其他政策授予存取權，使用者仍無法存取資源。
- 主體 – 在以身分為基礎的政策 (IAM 政策) 中，附加政策的使用者是隱含主體。對於以資源為基礎的政策，您可以指定希望獲得許可的使用者、帳戶、服務或其他實體。

如需進一步了解有關 IAM 政策語法和說明的詳細資訊，請參閱《IAM 使用者指南》中的 [AWS IAM 政策參考](#)。

如需顯示所有 CodeBuild API 動作及其適用的資源的資料表，請參閱 [AWS CodeBuild 許可參考](#)。

使用的身分型政策 AWS CodeBuild

本主題提供身分類型政策範例，示範帳戶管理員如何將許可政策連接至 IAM 身分 (即使用者、群組和角色)，並藉此授予許可，以對 AWS CodeBuild 資源執行操作。

Important

建議您先檢閱簡介主題，說明可用於管理 CodeBuild 資源存取權的基本概念和選項。如需詳細資訊，請參閱[管理 AWS CodeBuild 資源存取許可的概觀](#)。

主題

- [使用 AWS CodeBuild 主控台所需的許可](#)
- [AWS CodeBuild 連線至 Amazon Elastic Container Registry 所需的許可](#)
- [AWS CodeBuild 主控台連線至來源提供者所需的許可](#)
- [AWS 的受管 \(預先定義\) 政策 AWS CodeBuild](#)
- [CodeBuild 受管政策和通知](#)
- [CodeBuild 受 AWS 管政策的更新](#)
- [客戶受管政策範例](#)

以下顯示的許可政策範例，可讓使用者取得組建專案的相關資訊，僅針對 us-east-2 區域的帳戶 123456789012 中名稱開頭為 my 的組建專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

使用 AWS CodeBuild 主控台所需的許可

使用 AWS CodeBuild 主控台的使用者必須擁有一組最低許可，允許使用者描述 AWS 帳戶的其他 AWS 資源。您必須具備下列服務的許可：

- AWS CodeBuild
- Amazon CloudWatch
- CodeCommit (如果您將原始程式碼存放在 AWS CodeCommit 儲存庫)
- Amazon Elastic Container Registry (Amazon ECR) (如果您使用的建置環境依賴 Amazon ECR 儲存庫中的 Docker 映像)

Note

截至 2022 年 7 月 26 日，預設 IAM 政策已更新。如需詳細資訊，請參閱[AWS CodeBuild 連線至 Amazon Elastic Container Registry 所需的許可](#)。

- Amazon Elastic Container Service (Amazon ECS) (如果您使用的是依賴 Amazon ECR 儲存庫中 Docker 映像的建置環境)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

如果您建立比最低必要許可更嚴格的 IAM 政策，主控台將無法如預期般運作。

AWS CodeBuild 連線至 Amazon Elastic Container Registry 所需的許可

自 2022 年 7 月 26 日起，AWS CodeBuild 已更新其 Amazon ECR 許可的預設 IAM 政策。下列許可已從預設政策中移除：

```
"ecr:PutImage",  
"ecr:InitiateLayerUpload",  
"ecr:UploadLayerPart",  
"ecr:CompleteLayerUpload"
```

對於 2022 年 7 月 26 日之前建立的 CodeBuild 專案，我們建議您使用以下 Amazon ECR 政策更新您的政策：

```
"Action": [  
    "ecr:BatchCheckLayerAvailability",  
    "ecr:GetDownloadUrlForLayer",  
    "ecr:BatchGetImage"  
]
```

如需更新政策的詳細資訊，請參閱 [允許使用者與 CodeBuild 互動](#)。

AWS CodeBuild 主控台連線至來源提供者所需的許可

AWS CodeBuild 主控台使用以下 API 動作連線至來源提供者（例如 GitHub 儲存庫）。

- `codebuild:ListConnectedOAuthAccounts`
- `codebuild:ListRepositories`
- `codebuild:PersistOAuthToken`
- `codebuild:ImportSourceCredentials`

您可以使用 AWS CodeBuild 主控台將來源提供者（例如 GitHub 儲存庫）與建置專案建立關聯。若要這樣做，您必須先將上述 API 動作新增至與您用來存取 AWS CodeBuild 主控台之使用者相關聯的 IAM 存取政策。

`ListConnectedOAuthAccounts`、`ListRepositories` 和 `PersistOAuthToken` API 動作不是為了供您的程式碼呼叫。因此，這些 API 動作不包含在 AWS CLI 和 AWS SDKs 中。

AWS 的 受管（預先定義）政策 AWS CodeBuild

AWS 透過提供由 建立和管理的獨立 IAM 政策，解決許多常見的使用案例 AWS。這些 AWS 受管政策會授予常見使用案例的必要許可，因此您可以避免調查需要哪些許可。CodeBuild 的受管政策也提供在其他 服務中執行操作的許可，例如 IAM AWS CodeCommit、Amazon EC2、Amazon ECR、Amazon SNS 和 Amazon CloudWatch Events，以滿足已授予相關政策之使用者的責任所需。例如，AWSCodeBuildAdminAccess政策是一種管理層級使用者政策，可讓使用者使用此政策建立和管理專案組建的 CloudWatch Events 規則，以及針對專案相關事件（名稱字首為 的主題arn:aws:codebuild:) 通知的 Amazon SNS 主題，以及管理 CodeBuild 中的專案和報告群組。如需詳細資訊，請參閱《IAM 使用者指南》中的 [AWS 受管政策](#)。

下列 AWS 受管政策是特定的，您可以連接到您帳戶中的使用者 AWS CodeBuild。

AWSCodeBuildAdminAccess

提供 CodeBuild 的完整存取權，包括管理 CodeBuild 組建專案的許可。

AWSCodeBuildDeveloperAccess

提供 CodeBuild 的存取權，但不允許組建專案管理。

AWSCodeBuildReadOnlyAccess

提供 CodeBuild 的唯讀存取權。

若要存取 CodeBuild 建立的建置輸出成品，您還必須連接名為 的 AWS 受管政策AmazonS3ReadOnlyAccess。

若要建立和管理 CodeBuild 服務角色，您還必須連接名為 的 AWS 受管政策IAMFullAccess。

您也可以建立自己的自訂 IAM 政策，以允許 CodeBuild 動作和資源的許可。您可以將這些自訂政策連接至需要這些許可的使用者或群組。

主題

- [AWSCodeBuildAdminAccess](#)
- [AWSCodeBuildDeveloperAccess](#)
- [AWSCodeBuildReadOnlyAccess](#)

AWSCodeBuildAdminAccess

此AWSCodeBuildAdminAccess政策提供 CodeBuild 的完整存取權，包括管理 CodeBuild 組建專案的許可。僅將此政策套用至管理層級使用者，以授予他們對 AWS 帳戶中 CodeBuild 專案、報告群組和相關資源的完整控制權，包括刪除專案和報告群組的能力。

如需完整的受管政策，請參閱 IAM 受管政策參考中的 [AWSCodeBuildAdminAccess](#)。

AWSCodeBuildDeveloperAccess

此AWSCodeBuildDeveloperAccess政策允許存取 CodeBuild 的所有功能，以及專案和報告群組相關資源。此政策不允許使用者刪除 CodeBuild 專案或報告群組，或其他 AWS 服務中的相關資源，例如 CloudWatch Events。建議將此政策套用到大多數使用者。

如需完整的受管政策，請參閱 IAM 受管政策參考中的 [AWSCodeBuildDeveloperAccess](#)。

AWSCodeBuildReadOnlyAccess

此AWSCodeBuildReadOnlyAccess政策會授予 CodeBuild 和其他 AWS 服務中相關資源的唯讀存取權。將此政策套用至可檢視和執行組建、檢視專案，以及檢視報告群組，但無法對其進行任何變更的使用者。

如需完整的受管政策，請參閱 IAM 受管政策參考中的 [AWSCodeBuildReadOnlyAccess](#)。

CodeBuild 受管政策和通知

CodeBuild 支援通知，可通知使用者建置專案的重要變更。CodeBuild 的受管政策包含通知功能的政策陳述式。如需詳細資訊，請參閱[什麼是通知？](#)。

唯讀受管政策中的通知相關許可

AWSCodeBuildReadOnlyAccess 受管政策包含下列陳述式，允許對通知的唯讀存取權限。適用此受管政策的使用者可以檢視資源的通知，但無法建立、管理或訂閱通知。

```
{
  "Sid": "CodeStarNotificationsPowerUserAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-notifications:DescribeNotificationRule"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {"codestar-notifications:NotificationsForResource" :
      "arn:aws:codebuild:*:*:project/*"}
  }
}
```

```

    }
  },
  {
    "Sid": "CodeStarNotificationsListAccess",
    "Effect": "Allow",
    "Action": [
      "codestar-notifications:ListNotificationRules",
      "codestar-notifications:ListEventTypes",
      "codestar-notifications:ListTargets"
    ],
    "Resource": "*"
  }
}

```

其他受管政策中的通知相關許可

`AWSCodeBuildDeveloperAccess` 受管政策包含下列陳述式，允許使用者建立、編輯和訂閱通知。使用者無法刪除通知規則或管理資源的標籤。

```

{
  "Sid": "CodeStarNotificationsReadWriteAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-notifications:CreateNotificationRule",
    "codestar-notifications:DescribeNotificationRule",
    "codestar-notifications:UpdateNotificationRule",
    "codestar-notifications:Subscribe",
    "codestar-notifications:Unsubscribe"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild:*:*:project/*"}
  }
},
{
  "Sid": "CodeStarNotificationsListAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-notifications:ListNotificationRules",
    "codestar-notifications:ListTargets",
    "codestar-notifications:ListTagsForResource",
    "codestar-notifications:ListEventTypes"
  ],
}

```

```
    "Resource": "*"
  },
  {
    "Sid": "SNSTopicListAccess",
    "Effect": "Allow",
    "Action": [
      "sns:ListTopics"
    ],
    "Resource": "*"
  },
  {
    "Sid": "CodeStarNotificationsChatbotAccess",
    "Effect": "Allow",
    "Action": [
      "chatbot:DescribeSlackChannelConfigurations",
      "chatbot:ListMicrosoftTeamsChannelConfigurations"
    ],
    "Resource": "*"
  }
}
```

如需 IAM 和通知的詳細資訊，請參閱 [Identity and Access Management](#) 以取得AWS CodeStar通知。

CodeBuild 受 AWS 管政策的更新

檢視自此服務開始追蹤這些變更以來 CodeBuild AWS 受管政策更新的詳細資訊。如需此頁面變更的自動提醒，請訂閱 上的 RSS 摘要[AWS CodeBuild 使用者指南文件歷史記錄](#)。

變更	描述	日期
AWSCodeBuildAdminAccess 和 AWSCodeBuildDeveloperAccess 更新現有政策	<div>CodeBuild 新增了這些政策的ssmmessages:OpenDataChannel 許可，以支援Session Manager 互動式建置偵錯。</div> <div>AWSCodeBuildAdminAccess 和 AWSCodeBuildDeveloperAccess 政策現在包含 Session Manager 工作階段資源 (arn:aws:s</div>	2025 年 12 月 1 日

變更	描述	日期
	<p>sm:*:*:session/*)</p> <p>ssmmessages:OpenDataChannel 的動作，以支援此 WebSocket API 上的 SigV4 強制執行。</p>	
AWSCodeBuildAdminAccess、AWSCodeBuildDeveloperAccess 和 AWSCodeBuildReadOnlyAccess – 更新現有政策	<p>CodeBuild 已更新這些政策的資源。</p> <p>AWSCodeBuildAdminAccess、AWSCodeBuildDeveloperAccess 和 AWSCodeBuildReadOnlyAccess 政策已變更為更新現有資源。原始資源arn:aws:codebuild:*:*:project/* 。</p>	2024 年 11 月 15 日
AWSCodeBuildAdminAccess、AWSCodeBuildDeveloperAccess 和 AWSCodeBuildReadOnlyAccess – 更新現有政策	<p>CodeBuild 已將資源新增至這些政策，以支援品牌 AWS CodeConnections 重塑。</p> <p>AWSCodeBuildAdminAccess、AWSCodeBuildDeveloperAccess 和 AWSCodeBuildReadOnlyAccess 政策已變更為新增資源arn:aws:codeconnections:*:*:* 。</p>	2024 年 4 月 18 日

變更	描述	日期
AWSCodeBuildAdminAccess 和 AWSCodeBuildDeveloperAccess – 更新現有政策	<p>CodeBuild 新增了這些政策的許可，以支援在聊天應用程式中使用 Amazon Q Developer 的其他通知類型。</p> <p>AWSCodeBuildAdminAccess 和 AWSCodeBuildDeveloperAccess 政策已變更為新增許可 chatbot:ListMicrosoftTeamsChannelConfigurations 。</p>	2023 年 5 月 16 日
CodeBuild 開始追蹤變更	CodeBuild 開始追蹤其 AWS 受管政策的變更。	2021 年 5 月 16 日

客戶受管政策範例

在本節中，您可以找到授予 AWS CodeBuild 動作許可的使用者政策範例。當您使用 CodeBuild API、AWS SDKs 或 時，這些政策會運作 AWS CLI。當您使用主控台時，您必須授予其他主控台特定的許可。如需相關資訊，請參閱[使用 AWS CodeBuild 主控台所需的許可](#)。

您可以使用下列範例 IAM 政策來限制使用者和角色的 CodeBuild 存取。

主題

- [允許使用者取得建置專案的相關資訊](#)
- [允許使用者取得機群的相關資訊](#)
- [允許使用者取得報告群組的相關資訊](#)
- [允許使用者取得報告的相關資訊](#)
- [允許使用者建立建置專案](#)
- [允許使用者建立機群](#)
- [允許使用者建立報告群組](#)
- [允許使用者刪除機群](#)
- [允許使用者刪除報告群組](#)

- [允許使用者刪除報告](#)
- [允許使用者刪除建置專案](#)
- [允許使用者取得建置專案名稱的清單](#)
- [允許使用者變更建置專案的相關資訊](#)
- [允許使用者變更機群](#)
- [允許使用者變更報告群組](#)
- [允許使用者取得建置的相關資訊](#)
- [允許使用者取得建置專案之建置 ID 的清單](#)
- [允許使用者取得建置 ID 的清單](#)
- [允許使用者取得機群清單](#)
- [允許使用者取得報告群組的清單](#)
- [允許使用者取得報告的清單](#)
- [允許使用者取得報告群組的報告清單](#)
- [允許使用者取得報告的測試案例清單](#)
- [允許使用者開始執行建置](#)
- [允許使用者嘗試停止建置](#)
- [允許使用者嘗試刪除建置](#)
- [允許使用者取得 CodeBuild 管理之 Docker 映像的相關資訊](#)
- [允許使用者新增機群服務角色的許可政策](#)
- [允許 CodeBuild 存取建立 VPC 網路介面所需的 AWS 服務](#)
- [使用拒絕陳述式來 AWS CodeBuild 防止 中斷與來源提供者的連線](#)

允許使用者取得建置專案的相關資訊

以下顯示的政策陳述式範例，會允許使用者取得組建專案的相關資訊，針對 us-east-2 區域的帳戶 123456789012 中名稱開頭為 my 的組建專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Action": "codebuild:BatchGetProjects",
  "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
}
```

允許使用者取得機群的相關資訊

下列範例政策陳述式允許使用者取得帳戶 us-east-2 區域中機群的相關資訊 123456789012：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetFleets",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
    }
  ]
}
```

允許使用者取得報告群組的相關資訊

以下顯示的政策陳述式範例，會允許使用者取得帳戶 123456789012 在 us-east-2 區域的相關報告群組資訊：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReportGroups",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```



```
    }  
  ]  
}
```

允許使用者取得報告的相關資訊

以下顯示的政策陳述式範例，會允許使用者取得帳戶 123456789012 在 us-east-2 區域的相關報告資訊：

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "codebuild:BatchGetReports",  
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"  
    }  
  ]  
}
```

允許使用者建立建置專案

下列範例政策陳述式允許使用者建立任何名稱的組建專案，但僅限於 帳戶的 us-east-2 區域中123456789012，且只能使用指定的 CodeBuild 服務角色：

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "codebuild:CreateProject",  
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/*"  
    },  
    {  
      "Effect": "Allow",
```

```
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/CodeBuildServiceRole"
  }
]
```

下列範例政策陳述式允許使用者建立具有任何名稱的組建專案，但僅限於帳戶 us-east-2 區域中 123456789012，且僅使用指定的 CodeBuild 服務角色。它也會強制使用者只能搭配使用指定的服務角色 AWS CodeBuild，而非任何其他 AWS 服務。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/CodeBuildServiceRole",
      "Condition": {
        "StringEquals": {"iam:PassedToService": "codebuild.amazonaws.com"}
      }
    }
  ]
}
```

允許使用者建立機群

下列範例政策陳述式允許使用者在 us-east-2 區域中為帳戶 建立機群 123456789012：

JSON

```
{
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": "codebuild:CreateFleet",
    "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
  }
]
```

允許使用者建立報告群組

以下顯示的政策陳述式範例，會允許使用者建立帳戶 123456789012 在 us-east-2 區域中的報告群組：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

允許使用者刪除機群

下列範例政策陳述式允許使用者刪除帳戶 us-east-2 區域中的機群 123456789012：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteFleet",
```

```
    "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
  }
]
}
```

允許使用者刪除報告群組

以下顯示的政策陳述式範例，會允許使用者刪除帳戶 123456789012 在 us-east-2 區域中的報告群組：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

允許使用者刪除報告

以下顯示的政策陳述式範例，會允許使用者刪除帳戶 123456789012 在 us-east-2 區域中的報告：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReport",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

允許使用者刪除建置專案

以下顯示的政策陳述式範例，會允許使用者刪除組建專案，針對 us-east-2 區域的帳戶 123456789012 中名稱開頭為 my 的組建專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteProject",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

允許使用者取得建置專案名稱的清單

下列政策陳述式的範例，會允許使用者取得相同帳戶之組建專案名稱的清單：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListProjects",
      "Resource": "*"
    }
  ]
}
```

允許使用者變更建置專案的相關資訊

以下顯示的政策陳述式範例，會允許使用者變更使用任何名稱組建專案的相關資訊，但只能針對 us-east-2 區域中的帳戶 123456789012，並只能使用指定的 AWS CodeBuild 服務角色：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateProject",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/CodeBuildServiceRole"
    }
  ]
}
```

允許使用者變更機群

下列範例政策陳述式允許使用者變更帳戶 us-east-2 區域中的機群 123456789012：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateFleet",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
    }
  ]
}
```

允許使用者變更報告群組

以下顯示的政策陳述式範例，會允許使用者變更帳戶 123456789012 在 us-east-2 區域中的報告群組：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

允許使用者取得建置的相關資訊

以下顯示的政策陳述式範例，會允許使用者取得組建的相關資訊，針對 us-east-2 區域的帳戶 123456789012 中名為 my-build-project 和 my-other-build-project 的組建專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetBuilds",
      "Resource": [
        "arn:aws:codebuild:us-east-2:111122223333:project/my-build-project",
        "arn:aws:codebuild:us-east-2:111122223333:project/my-other-build-project"
      ]
    }
  ]
}
```

允許使用者取得建置專案之建置 ID 的清單

以下顯示的政策陳述式範例，會允許使用者取得組建 ID 的清單，針對 us-east-2 區域的帳戶 123456789012 中名為 my-build-project 和 my-other-build-project 的組建專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListBuildsForProject",
      "Resource": [
        "arn:aws:codebuild:us-east-2:111122223333:project/my-build-project",
        "arn:aws:codebuild:us-east-2:111122223333:project/my-other-build-project"
      ]
    }
  ]
}
```

允許使用者取得建置 ID 的清單

下列政策陳述式的範例，會允許使用者取得相同帳戶之所有組建 ID 的清單：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListBuilds",
      "Resource": "*"
    }
  ]
}
```

允許使用者取得機群清單

下列範例政策陳述式允許使用者取得帳戶 us-east-2 區域中機群的清單 123456789012：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListFleets",
      "Resource": "*"
    }
  ]
}
```

允許使用者取得報告群組的清單

以下顯示的政策陳述式範例，會允許使用者取得帳戶 123456789012 在 us-east-2 區域的報告群組清單：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReportGroups",
      "Resource": "*"
    }
  ]
}
```

允許使用者取得報告的清單

以下顯示的政策陳述式範例，會允許使用者取得帳戶 123456789012 在 us-east-2 區域的報告清單：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReports",
      "Resource": "*"
    }
  ]
}
```

允許使用者取得報告群組的報告清單

以下顯示的政策陳述式範例，會允許使用者取得帳戶 123456789012 在 us-east-2 區域中報告群組的報告清單：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReportsForReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

允許使用者取得報告的測試案例清單

以下顯示的政策陳述式範例，會允許使用者取得帳戶 123456789012 在 us-east-2 區域中報告的測試案例清單：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DescribeTestCases",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

允許使用者開始執行建置

以下顯示的政策陳述式範例，會允許使用者執行組建，針對 us-east-2 區域的帳戶 123456789012 中名稱開頭為 my 的組建專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

允許使用者嘗試停止建置

以下顯示的政策陳述式範例，會允許使用者嘗試停止執行中的組建，僅針對 us-east-2 區域的帳戶 123456789012 中名稱開頭為 my 的組建專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StopBuild",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

允許使用者嘗試刪除建置

以下顯示的政策陳述式範例，會允許使用者嘗試刪除建置，僅針對 us-east-2 區域的帳戶 123456789012 中名稱開頭為 my 的建置專案：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchDeleteBuilds",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

允許使用者取得 CodeBuild 管理之 Docker 映像的相關資訊

下列範例政策陳述式允許使用者取得 CodeBuild 管理之所有 Docker 映像的相關資訊：

JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "codebuild:ListCuratedEnvironmentImages",
    "Resource": "*"
  }
]
```

允許使用者新增機群服務角色的許可政策

下列範例資源政策陳述式允許使用者為機群服務角色新增 VPC 許可政策：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildFleetVpcCreateNI",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-1",
        "arn:aws:ec2:us-west-2:111122223333:security-group/security-  
group-id-1",
        "arn:aws:ec2:us-west-2:111122223333:network-interface/*"
      ]
    },
    {
      "Sid": "CodeBuildFleetVpcPermission",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
```

```

        "ec2:DeleteNetworkInterface"
    ],
    "Resource": "*"
  },
  {
    "Sid": "CodeBuildFleetVpcNIPermission",
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "arn:aws:ec2:us-west-2:111122223333:network-interface/*",
    "Condition": {
      "StringEquals": {
        "ec2:Subnet": [
          "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-1"
        ]
      }
    }
  }
]
}

```

下列範例資源政策陳述式允許使用者為機群服務角色新增自訂 Amazon Managed Image (AMI) 許可政策：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:DescribeImages",
      "Resource": "*"
    }
  ]
}

```

下列信任政策陳述式範例允許使用者新增機群服務角色的許可政策：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildFleetVPCTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

允許 CodeBuild 存取建立 VPC 網路介面所需的 AWS 服務

下列範例政策陳述式授予在具有兩個子網路的 VPC 中建立網路介面的 AWS CodeBuild 許可：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2>DeleteNetworkInterface",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
      ]
    }
  ]
}
```

```

        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateNetworkInterfacePermission"
        ],
        "Resource": "arn:aws:ec2:us-west-2:111122223333:network-interface/*",
        "Condition": {
            "StringEquals": {
                "ec2:AuthorizedService": "codebuild.amazonaws.com"
            },
            "ArnEquals": {
                "ec2:Subnet": [
                    "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-1",
                    "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-2"
                ]
            }
        }
    }
]
}

```

使用拒絕陳述式來 AWS CodeBuild 防止 中斷與來源提供者的連線

以下範例政策陳述式會使用拒絕陳述式，以防止 AWS CodeBuild 中斷與來源提供者的連接。它使用 `codebuild:DeleteOAuthToken` (與 `codebuild:PersistOAuthToken` 和 `codebuild:ImportSourceCredentials` 相反) 與來源供應商連接。如需詳細資訊，請參閱[AWS CodeBuild 主控台連線至來源提供者所需的許可](#)。

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "codebuild:DeleteOAuthToken",
            "Resource": "*"
        }
    ]
}

```



```
}
```

AWS CodeBuild 許可參考

您可以在 AWS CodeBuild 政策中使用 AWS 整個條件索引鍵來表達條件。如需清單，請參閱《IAM 使用者指南》中的 [可用金鑰](#)。

您可以在政策的 Action 欄位中指定動作。若要指定動作，請使用 codebuild: 字首，後面接著 API 操作名稱 (例如 codebuild:CreateProject 和 codebuild:StartBuild)。若要在單一陳述式中指定多個動作，請用逗號加以分隔 (例如 "Action": ["codebuild:CreateProject", "codebuild:StartBuild"])。

使用萬用字元

您可以使用或不使用萬用字元 (*)，指定 ARN 做為政策之 Resource 欄位中的資源值。您可以使用萬用字元指定多個動作或資源。例如，會 codebuild:* 指定所有 CodeBuild 動作，並 codebuild:Batch* 指定以字詞 開頭的所有 CodeBuild 動作 Batch。以下範例會對名稱開頭為 my 之所有組建專案授予存取權：

```
arn:aws:codebuild:us-east-2:123456789012:project/my*
```

CodeBuild API 操作和動作的必要許可

BatchDeleteBuilds

動作：codebuild:BatchDeleteBuilds

必須具備才能刪除組建。

資源：arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetBuilds

動作：codebuild:BatchGetBuilds

必須具備才能取得組建的相關資訊。

資源：arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetProjects

動作：codebuild:BatchGetProjects

必須具備才能取得組建專案的相關資訊。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

BatchGetReportGroups

動作 : `codebuild:BatchGetReportGroups`

必須具備才能取得報告群組的相關資訊。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

BatchGetReports

動作 : `codebuild:BatchGetReports`

必須具備才能取得報告的相關資訊。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

BatchPutTestCases 1

動作 : `codebuild:BatchPutTestCases`

建立或更新測試報告時需要。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

CreateProject

動作 : `codebuild:CreateProject`、`iam:PassRole`

必須具備才能建立組建專案。

資源 :

- `arn:aws:codebuild:region-ID:account-ID:project/project-name`
- `arn:aws:iam::account-ID:role/role-name`

CreateReport 1

動作 : `codebuild:CreateReport`

建立測試報告時需要。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

CreateReportGroup

動作 : `codebuild:CreateReportGroup`

必須具備才能建立報告群組。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

CreateWebhook

動作 : `codebuild:CreateWebhook`

必須具備才能建立 Webhook。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

DeleteProject

動作 : `codebuild>DeleteProject`

刪除 CodeBuild 專案時需要。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

DeleteReport

動作 : `codebuild>DeleteReport`

必須具備才能刪除報告。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

DeleteReportGroup

動作 : `codebuild>DeleteReportGroup`

必須具備才能刪除報告群組。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

DeleteSourceCredentials

動作 : `codebuild:DeleteSourceCredentials`

必須具備才能刪除包含 GitHub、GitHub Enterprise Server 或 Bitbucket 儲存庫適用登入資料的一組 `SourceCredentialsInfo` 物件。

資源 : *

DeleteWebhook

動作 : `codebuild:DeleteWebhook`

必須具備才能建立 Webhook。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

DescribeTestCases

動作 : `codebuild:DescribeTestCases`

必須具備才能傳回測試案例的分頁清單。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

ImportSourceCredentials

動作 : `codebuild:ImportSourceCredentials`

必須具備才能匯入包含 GitHub、GitHub Enterprise 或 Bitbucket 儲存庫適用登入資料的一組 `SourceCredentialsInfo` 物件。

資源 : *

InvalidateProjectCache

動作 : `codebuild:InvalidateProjectCache`

必須具備才能重設專案的快取。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

ListBuildBatches

動作：codebuild:ListBuildBatches

取得組建批次 IDs清單時需要。

資源：*

ListBuildBatchesForProject

動作：codebuild:ListBuildBatchesForProject

取得特定專案的建置批次 IDs清單時需要。

資源：arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

ListBuilds

動作：codebuild:ListBuilds

必須具備才能取得組建 ID 的清單。

資源：*

ListBuildsForProject

動作：codebuild:ListBuildsForProject

必須具備才能取得組建專案的組建 ID 清單。

資源：arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

ListCuratedEnvironmentImages

動作：codebuild:ListCuratedEnvironmentImages

必須具備才能取得 AWS CodeBuild 管理之所有 Docker 影像的相關資訊。

資源：* (必要，但並未參考可定址的 AWS 資源)

ListProjects

動作：codebuild:ListProjects

必須具備才能取得組建專案名稱的清單。

資源：*

ListReportGroups

動作 : `codebuild:ListReportGroups`

必須具備才能取得報告群組的清單。

資源 : *

ListReports

動作 : `codebuild:ListReports`

必須具備才能取得報告清單。

資源 : *

ListReportsForReportGroup

動作 : `codebuild:ListReportsForReportGroup`

必須具備才能取得報告群組的報告清單。

資源 : `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

RetryBuild

動作 : `codebuild:RetryBuild`

重試組建時需要。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

StartBuild

動作 : `codebuild:StartBuild`

必須具備才能開始執行組建。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

StopBuild

動作 : `codebuild:StopBuild`

必須具備才能停止執行組建。

資源 : `arn:aws:codebuild:region-ID:account-ID:project/project-name`

UpdateProject

動作 : codebuild:UpdateProject、 iam:PassRole

必須具備才能變更組建的相關資訊。

資源 :

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

UpdateProjectVisibility

動作 : codebuild:UpdateProjectVisibility、 iam:PassRole

變更專案建置的公有可見性時需要。

資源 :

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

UpdateReport 1

動作 : codebuild:UpdateReport

建立或更新測試報告時需要。

資源 : arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

UpdateReportGroup

動作 : codebuild:UpdateReportGroup

必須具備才能更新報告群組。

資源 : arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

UpdateWebhook

動作 : codebuild:UpdateWebhook

必須具備才能更新 Webhook。

資源 : arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

1 僅用於許可。此動作沒有 API。

使用標籤控制對 AWS CodeBuild 資源的存取

IAM 政策陳述式中的條件是語法的一部分，可用來指定 CodeBuild 專案型動作的許可。您可以建立政策，允許或拒絕根據與這些專案相關聯的標籤對專案執行動作，然後將這些政策套用到您為管理使用者所設定的 IAM 群組。如需使用 主控台或 將標籤套用至專案的資訊 AWS CLI，請參閱 [在中建立建置專案 AWS CodeBuild](#)。如需有關使用 CodeBuild SDK 套用標籤的資訊，請參閱 CodeBuild API 參考中的 [CreateProject](#) 和 [標籤](#)。如需使用標籤控制 AWS 資源存取的資訊，請參閱《IAM 使用者指南》中的 [使用資源標籤控制 AWS 資源存取](#)。

Important

使用預留容量功能時，同一帳戶中的其他專案可以存取機群執行個體上快取的資料，包括來源檔案、Docker layer 和 buildspec 中指定的快取目錄。這是根據設計，並允許相同帳戶中的專案共用機群執行個體。

Example範例 1：根據資源標籤限制 CodeBuild 專案動作

以下範例會在以索引鍵為 Environment 且索引鍵值為 Production 標記的專案上，拒絕所有 BatchGetProjects 動作：除了受管使用者政策外，使用者的管理員還必須將此 IAM 政策連接到未經授權的使用者。aws:ResourceTag 條件索引鍵用於依據其標籤來控制資源的存取。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetProjects"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:ResourceTag/Environment": "Production"
        }
      }
    }
  ]
}
```



```
    }  
  }  
]  
}
```

Example 範例 2：根據請求標籤限制 CodeBuild 專案動作

如果請求包含索引鍵為 `Environment` 且索引鍵值為 `Production` 的標籤，則下列政策會拒絕使用者對 `CreateProject` 動作的許可。此外，如果請求包含索引鍵為 `Environment` 的標籤，則政策會使用 `aws:TagKeys` 條件索引鍵來不允許 `UpdateProject`，防止未授權的使用者修改專案。除了受管使用者政策之外，管理員還必須將此 IAM 政策連接到未獲授權執行這些動作的使用者。`aws:RequestTag` 條件索引鍵用於控制哪些標籤可以在 IAM 請求中傳遞

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": [  
        "codebuild:CreateProject"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "ForAnyValue:StringEquals": {  
          "aws:RequestTag/Environment": "Production"  
        }  
      }  
    },  
    {  
      "Effect": "Deny",  
      "Action": [  
        "codebuild:UpdateProject"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "ForAnyValue:StringEqualsIgnoreCase": {  
          "aws:TagKeys": ["Environment"]  
        }  
      }  
    }  
  ]  
}
```

```

    }
  ]
}
```

Example範例 3：根據資源標籤拒絕或允許報告群組上的動作

您可以根據與這些資源相關聯的 AWS 標籤，建立允許或拒絕 CodeBuild 資源（專案和報告群組）動作的政策，然後將這些政策套用至您為管理使用者所設定的 IAM 群組。例如，您可以建立一個政策，以拒絕具有 AWS 標籤索引鍵 `Status` 和 索引鍵值的任何報告群組上的所有 CodeBuild 動作 `Secret`，然後將該政策套用到您為一般開發人員 (####) 建立的 IAM 群組。然後，您需要確保處理這些標記報告群組的開發人員不是該一般 #### 群組的成員，而是屬於未套用限制性政策的不同 IAM 群組 (`SecretDevelopers`)。

下列範例會拒絕標記 索引鍵 `Status` 和 索引鍵值之報告群組的所有 CodeBuild 動作 `Secret`：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetReportGroups",
        "codebuild:CreateReportGroup",
        "codebuild>DeleteReportGroup",
        "codebuild:ListReportGroups",
        "codebuild:ListReportsForReportGroup",
        "codebuild:UpdateReportGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestedRegion": "us-east-1"
        }
      }
    }
  ]
}
```

Example範例 4：根據資源標籤將 CodeBuild 動作限制為 AWSCodeBuildDeveloperAccess

您可以建立政策，允許 CodeBuild 對所有未加上特定標籤的報告群組和專案執行動作。例如，以下政策允許對所有報告群組和專案擁有同等的 [AWSCodeBuildDeveloperAccess](#) 許可，但以特定標籤標記的專案除外：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:BatchGet*",
        "codebuild:GetResourcePolicy",
        "codebuild:DescribeTestCases",
        "codebuild:List*",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "codecommit:ListBranches",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "logs:GetLogEvents",
        "s3:GetBucketLocation",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/Status": "Secret",
          "aws:ResourceTag/Team": "Saanvi"
        }
      }
    }
  ]
}
```

在主控台檢視資源

AWS CodeBuild 主控台需要 `ListRepositories` 許可，才能顯示您登入 AWS 區域中 AWS 帳戶的儲存庫清單。主控台還包含 `Go to resource` (移至資源) 功能，可快速執行不區分大小寫的資源搜尋。此搜尋會在您登入的 AWS 區域中的帳戶 AWS 中執行。將會跨以下服務來顯示以下資源：

- AWS CodeBuild：組建專案
- AWS CodeCommit：儲存庫
- AWS CodeDeploy：應用程式
- AWS CodePipeline：管道

若要跨所有服務中的資源執行此搜尋，您必須擁有以下許可：

- CodeBuild：`ListProjects`
- CodeCommit：`ListRepositories`
- CodeDeploy：`ListApplications`
- CodePipeline：`ListPipelines`

如果您沒有某項服務的許可，則不會傳回該服務之資源的結果。即使您有許可來檢視資源，但如果有任何明確的 `Deny` 而無法檢視部分資源，則不會傳回這些資源。

的合規驗證 AWS CodeBuild

我們的新 AWS 註冊體驗並非針對受管制的工作負載而設計。如果您正在使用我們新的 AWS 註冊體驗，但想要將 AWS 用於受管制的工作負載，您可以[註冊 AWS（進階）](#)或為您的 AWS 環境[啟用進階功能](#)。

若要了解是否 AWS 服務在特定合規計劃範圍內，請參閱[AWS 服務合規計劃範圍內](#)然後選擇您感興趣的合規計劃。如需一般資訊，請參閱[AWS 合規計劃](#)。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱[下載報告下載 AWS Artifact](#)。

您使用時的合規責任 AWS 服務取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用時合規責任的詳細資訊 AWS 服務，請參閱[AWS 安全文件](#)。

中的彈性 AWS CodeBuild

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。AWS 區域提供多個實體隔離和隔離的可用區域，這些區域以低延遲、高輸送量和高度備援聯網連接。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

中的基礎設施安全 AWS CodeBuild

作為受管服務，AWS CodeBuild 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 CodeBuild。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

在 CodeBuild 中存取您的來源提供者

對於 GitHub 或 GitHub Enterprise Server，您可以使用個人存取字符、Secrets Manager 秘密、連線或 OAuth 應用程式來存取來源提供者。對於 Bitbucket，您可以使用存取字符、應用程式密碼、API 字符、Secrets Manager 秘密、連線或 OAuth 應用程式來存取來源提供者。

主題

- [在 Secrets Manager 秘密中建立和存放權杖](#)
- [CodeBuild 中的 GitHub 和 GitHub Enterprise Server 存取](#)
- [CodeBuild 中的 Bitbucket 存取](#)
- [CodeBuild 中的 GitLab 存取](#)

在 Secrets Manager 秘密中建立和存放權杖

如果您選擇使用 Secrets Manager 存放存取字符，您可以使用現有的秘密連線或建立新的秘密。若要建立新的秘密，請執行下列動作：

AWS 管理主控台

在 中建立 Secrets Manager 秘密 AWS 管理主控台

1. 針對來源提供者，選擇 Bitbucket、GitHub 或 GitHub Enterprise。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇預設來源登入資料，以使用您帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線到來源提供者，請選擇管理預設來源憑證。
 - b. 針對登入資料類型，選擇 以外的登入資料類型AWS CodeConnections。
 - c. 針對服務，選擇 Secrets Manager，針對 Secrets 選擇新秘密。
 - d. 在秘密名稱中，輸入秘密的名稱。
 - e. 在秘密描述 - 選用中，輸入秘密的描述。
 - f. 根據您選擇的來源提供者，輸入您的字符或使用者名稱和應用程式密碼，然後選擇儲存。對於 Bitbucket API 字符，請改為輸入您的 Atlassian 帳戶電子郵件地址和 API 字符。
 - 選擇自訂來源登入資料，以使用自訂來源登入資料覆寫您帳戶的預設設定。
 - a. 針對登入資料類型，選擇 以外的登入資料類型AWS CodeConnections。
 - b. 在連線中，選擇建立秘密。
 - c. 在秘密名稱中，輸入秘密的名稱。
 - d. 在秘密描述 - 選用中，輸入秘密的描述。
 - e. 根據您選擇的來源提供者，輸入您的字符或使用者名稱和應用程式密碼，然後選擇建立。對於 Bitbucket API 字符，請改為輸入您的 Atlassian 帳戶電子郵件地址和 API 字符。

AWS CLI

在 中建立 Secrets Manager 秘密 AWS CLI

- 開啟終端機 (Linux、macOS 或 Unix) 或命令提示 (Windows)。使用 AWS CLI 執行 Secrets Manager create-secret命令。

```
aws secretsmanager create-secret --region <aws-region> \  
    --name '<secret-name>' \  
    --description '<secret-description>' \  
    --
```

```
--secret-string '{
    "ServerType": "<server-type>",
    "AuthType": "<auth-type>",
    "Token": "<token>"
}' \
--tags Key=codebuild:source,Value='' \
      Key=codebuild:source:type,Value=<type> \
      Key=codebuild:source:provider,Value=<provider>
```

CodeBuild 接受的 Secrets Manager 秘密必須與 CodeBuild 專案位於相同的帳戶和 AWS 區域，且必須是下列 JSON 格式：

```
{
    "ServerType": ServerType,
    "AuthType": AuthType,
    "Token": string,
    "Username": string // Optional and is only used for Bitbucket app
    password or API token
}
```

欄位	有效值	說明
ServerType	GITHUB	Secrets Manager 秘密的第三方來源提供者。
	GITHUB_ENTERPRISE	
	位元儲存貯體	
AuthType	Personal_ACCESS_TOKEN	登入資料使用的存取字符類型。對於 GitHub，只有 Personal_ACCESS_TOKEN 有效。BASIC_AUTH 僅適用於 Bitbucket 應用程式密碼或 API 字符。
	BASIC_AUTH	

欄位	有效值	說明
權杖	<i>string</i>	對於 GitHub 或 GitHub Enterprise，這是個人存取字符。對於 Bitbucket，這是存取字符、Bitbucket 應用程式密碼或 Bitbucket API 字符。
使用者名稱	<i>string</i>	AuthType 為 BASIC_AUTH 時的 Bitbucket 使用者名稱。如果您使用 Bitbucket API 字符，這是與您的 Atlassian 帳戶相關聯的電子郵件地址。此參數不適用於其他類型的來源提供者。

此外，CodeBuild 會在秘密上使用下列資源標籤，以確保在建立或編輯專案時可輕鬆選取秘密。

標籤鍵	標籤值	說明
codebuild : source : provider	GitHub	告知 CodeBuild 此秘密的用途提供者。
	github_enterprise	
	bitbucket	
codebuild : source : type	personal_access_token	告知 CodeBuild 此秘密中的存取字符類型。
	basic_auth	

CodeBuild 中的 GitHub 和 GitHub Enterprise Server 存取

對於 GitHub，您可以使用個人存取字符、OAuth 應用程式、Secrets Manager 秘密或 GitHub 應用程式連線來存取來源提供者。對於 GitHub Enterprise Server，您可以使用個人存取字符、Secrets Manager 秘密或 GitHub 應用程式連線來存取來源提供者。

主題

- [GitHub 和 GitHub Enterprise Server 的 GitHub 應用程式連線](#)
- [GitHub 和 GitHub Enterprise Server 存取字符](#)
- [GitHub OAuth 應用程式](#)

GitHub 和 GitHub Enterprise Server 的 GitHub 應用程式連線

您可以使用 GitHub 應用程式與 CodeBuild 連線。透過 支援 GitHub 應用程式連線[AWS CodeConnections](#)。

來源提供者存取可讓您[GitHub Webhook 事件](#)使用 [CreateWebhook](#) 訂閱 或在 CodeBuild [教學課程](#)：設定 CodeBuild 託管的 GitHub 動作執行器中使用 來觸發組建。

Note

AWS CodeConnections 可在比 CodeBuild 更少的區域中使用。您可以在 CodeBuild 中使用跨區域連線。在選擇加入區域中建立的連線無法用於其他區域。如需詳細資訊，請參閱 [AWS CodeConnections 端點和配額](#)。

主題

- [步驟 1：建立 GitHub 應用程式的連線（主控台）](#)
- [步驟 2：授予 CodeBuild 專案 IAM 角色存取權以使用連線](#)
- [步驟 3：設定 CodeBuild 以使用新的連線](#)
- [對 GitHub 應用程式的問題進行故障診斷](#)

步驟 1：建立 GitHub 應用程式的連線（主控台）

使用這些步驟來使用 CodeBuild 主控台為 GitHub 中的專案新增連線。

建立連至 GitHub 的連線

- 請遵循開發人員工具使用者指南中[建立 GitHub 連線](#)的說明。

Note

您可以使用從另一個帳戶共用的連線，而不是在帳戶中建立或使用現有的連線 AWS。如需詳細資訊，請參閱[與 AWS 帳戶共用連線](#)。

步驟 2：授予 CodeBuild 專案 IAM 角色存取權以使用連線

您可以授予 CodeBuild 專案 IAM 角色存取權，以使用連線提供的 GitHub 字符。

授予 CodeBuild 專案 IAM 角色存取權

1. 遵循 CodeBuild 專案的指示，為您的 CodeBuild 專案建立 [允許 CodeBuild 與其他 AWS 服務互動](#) IAM 角色。
2. 遵循指示時，將下列 IAM 政策新增至 CodeBuild 專案角色，以授予連線的存取權。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:GetConnectionToken",
        "codeconnections:GetConnection"
      ],
      "Resource": [
        "arn:aws:iam::*:role/Service*"
      ]
    }
  ]
}
```

步驟 3：設定 CodeBuild 以使用新的連線

您可以將連線設定為帳戶層級登入資料，並在專案中使用它。

AWS 管理主控台

若要在 中將連線設定為帳戶層級登入資料 AWS 管理主控台

1. 針對 Source provider (來源供應商)，選擇 GitHub。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇預設來源登入資料，以使用您帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線至 GitHub，請選擇管理預設來源憑證。
 - b. 針對登入資料類型，選擇 GitHub 應用程式。
 - c. 在連線中，選擇使用現有的連線或建立新的連線。
 - 選擇自訂來源登入資料，以使用自訂來源登入資料覆寫您帳戶的預設設定。
 - a. 針對登入資料類型，選擇 GitHub 應用程式。
 - b. 在連線中，選擇使用現有的連線或建立新的連線。

AWS CLI

若要在 中將連線設定為帳戶層級登入資料 AWS CLI

- 開啟終端機 (Linux、macOS 或 Unix) 或命令提示 (Windows)。使用 AWS CLI 執行 import-source-credentials 命令，--token 為您的連線指定 --auth-type、--server-type 和。

使用下列命令：

```
aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-type GITHUB --token <connection-arn>
```

您也可以為 CodeBuild 專案設定多個權杖。如需詳細資訊，請參閱[將多個字符設定為來源層級登入資料](#)。

對 GitHub 應用程式的問題進行故障診斷

下列資訊可協助您對 GitHub 應用程式的常見問題進行疑難排解。

主題

- [在不需要的區域安裝 AWS Connector for GitHub 應用程式](#)
- [GitHub 應用程式連線無法存取儲存庫](#)

- [AWS 服務的 IAM 角色缺少必要的 IAM 許可。](#)

在不需要的區域安裝 AWS Connector for GitHub 應用程式

問題：您從 GitHub Marketplace 安裝 AWS 了 Connector for GitHub，但連線是在不需要的區域建立的。如果您嘗試在 GitHub 網站上重新設定應用程式，則無法運作，因為應用程式已安裝在您的 GitHub 帳戶。

可能原因：應用程式已安裝在您的 GitHub 帳戶中，因此您只能重新設定應用程式許可。

建議的解決方案：您可以使用所需區域中的安裝 ID 建立新的連線。

1. 在 AWS CodeConnections <https://console.aws.amazon.com/codesuite/settings/connections> 開啟主控台，並使用 AWS 主控台導覽列中的區域選擇器導覽至所需區域。
2. 請遵循開發人員工具使用者指南中[建立 GitHub 連線](#)的說明。

 Note

由於您已安裝 AWS Connector for GitHub 應用程式，因此可以選擇它，而不是安裝新的應用程式。

GitHub 應用程式連線無法存取儲存庫

問題：使用連線 AWS 的服務，例如 CodeBuild 或 CodePipeline，報告其無法存取儲存庫或儲存庫不存在。一些可能的錯誤訊息包括：

- Authentication required for primary source.
- Unable to create webhook at this time. Please try again later.
- Failed to create webhook. GitHub API limit reached. Please try again later.

可能原因：您可能一直使用 GitHub 應用程式，但尚未授予 Webhook 許可範圍。

建議的解決方案：若要授予所需的許可範圍，請遵循[導覽至您要檢閱或修改的 GitHub 應用程式](#)中的指示，以設定已安裝的應用程式。在許可區段下，您會看到應用程式沒有 Webhook 許可，而且您可以選擇檢閱新請求的許可。檢閱並接受新的許可。如需詳細資訊，請參閱[核准 GitHub 應用程式的更新許可](#)。

可能原因：連線如預期般運作，但突然無法存取儲存庫。

可能的解決方案：從檢閱您的[授權](#)和[安裝](#)開始，然後驗證 GitHub 應用程式是否已授權並安裝。如果 GitHub 應用程式安裝已暫停，則需要將其取消暫停。如果 GitHub 應用程式未獲授權進行 [UAT（使用者存取字符）](#) 連線，或未針對 [IAT（安裝存取字符）](#) 連線安裝，則現有的連線將無法再使用，而且您將需要建立新的連線。請注意，重新安裝 GitHub 應用程式不會復原與舊安裝相關聯的先前連線。

可能的解決方案：如果連線是 UAT 連線，請確定未同時使用連線，例如在多個 CodeBuild 並行執行建置中使用。這是因為如果連線重新整理過期權杖，GitHub 會立即使先前發行的 UAT 失效。如果您需要將 UAT 連線用於多個並行 CodeBuild 組建，您可以建立多個連線並獨立使用每個連線。

可能的解決方案：如果在過去 6 個月內未使用 UAT 連線，GitHub 會將連線失效。若要修正此問題，請建立新的連線。

可能原因：您可能已在未安裝應用程式的情況下使用 UAT 連線。

建議的解決方案：雖然建立 UAT 連線不需要將連線與 GitHub 應用程式安裝建立關聯，但需要安裝才能存取儲存庫。依照指示[檢閱安裝](#)，以確保已安裝 GitHub 應用程式。如果未安裝，請導覽至 [GitHub 應用程式頁面](#) 以安裝應用程式。如需 UAT 存取的詳細資訊，請參閱[關於使用者存取字符](#)。

AWS 服務的 IAM 角色缺少必要的 IAM 許可。

問題：您會看到下列任何錯誤訊息：

- Access denied to connection *<connection-arn>*
- Failed to get access token from *<connection-arn>*

建議的解決方案：通常您使用 AWS 服務的連線，例如 CodePipeline 或 CodeBuild。當您為 AWS 服務提供 IAM 角色時，該 AWS 服務可以使用該角色的許可來代表您採取行動。確定 IAM 角色具有必要的許可。如需必要 IAM 許可的詳細資訊，請參閱《開發人員工具主控台使用者指南》中的[授予 CodeBuild 專案 IAM 角色存取權，以使用通知的連線和身分和存取管理 AWS CodeStarAWS CodeConnections](#)。

GitHub 和 GitHub Enterprise Server 存取字符

存取字符先決條件

開始之前，您必須將適當的許可範圍新增至 GitHub 存取字符。

對於 GitHub，您的個人存取字符必須具有以下範圍。

- repo：授予私有儲存庫的完全控制。
- repo：status：授予公有和私有儲存庫遞交狀態的讀取/寫入存取權。
- admin:repo_hook：授予儲存庫勾點的完全控制。如果您的字符有 repo 範圍，則不需要此範圍。
- admin：org_hook：授予組織勾點的完整控制權。只有在您使用組織 Webhook 功能時，才需要此範圍。

如需詳細資訊，請參閱 GitHub 網站上的 [了解 OAuth 應用程式的範圍](#)。

如果您使用的是精細的個人存取字符，根據您的使用案例，您的個人存取字符可能需要以下許可：

- 內容：唯讀：授予私有儲存庫的存取權。如果您使用私有儲存庫做為來源，則需要此許可。
- 遞交狀態：讀取和寫入：授予建立遞交狀態的許可。如果您的專案已設定 Webhook，或者您已啟用報告建置狀態功能，則需要此許可。
- Webhooks：讀取和寫入：授予管理 Webhook 的許可。如果您的專案已設定 Webhook，則需要此許可。
- 提取請求：唯讀：授予存取提取請求的許可。如果您的 Webhook 具有提取請求事件的 FILE_PATH 篩選條件，則需要此許可。
- 管理：讀取和寫入：如果您搭配 CodeBuild 使用自我託管的 GitHub 動作執行器功能，則需要此許可。如需詳細資訊，請參閱 [建立儲存庫的註冊字符](#) 和 [教學課程：設定 CodeBuild 託管的 GitHub 動作執行器](#)。

Note

如果您想要存取組織儲存庫，請務必將組織指定為存取權杖的資源擁有者。

如需詳細資訊，請參閱 GitHub [網站上的微調個人存取字符所需的許可](#)。

使用存取字符連接 GitHub（主控台）

若要使用 主控台，使用存取字符將專案連線至 GitHub，請在建立專案時執行下列動作。如需相關資訊，請參閱 [建立組建專案 \(主控台\)](#)。

1. 針對 Source provider (來源供應商)，選擇 GitHub。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇使用帳戶登入資料，將帳戶的預設來源登入資料套用至所有專案。

- a. 如果您未連線至 GitHub，請選擇管理帳戶登入資料。
- b. 針對登入資料類型，選擇個人存取字符。
- 如果您選擇使用 Service 的帳戶層級登入資料，請選擇您要用來存放字符的服務，然後執行下列動作：
 - a. 如果您選擇使用 Secrets Manager，您可以選擇使用現有的秘密連線或建立新的秘密，然後選擇儲存。如需如何建立新秘密的詳細資訊，請參閱 [在 Secrets Manager 秘密中建立和存放權杖](#)。
 - b. 如果您選擇使用 CodeBuild，請輸入您的 GitHub 個人存取字符，然後選擇儲存。
- 選取僅使用此專案的覆寫登入資料來使用自訂來源登入資料來覆寫帳戶的登入資料設定。
 - a. 從填入的登入資料清單中，選擇個人存取字符下的其中一個選項。
 - b. 您也可以描述中選取建立新的個人存取字符連線，以建立新的個人存取字符。

使用存取字符 (CLI) 連接 GitHub

請依照下列步驟使用 AWS CLI，使用存取字符將您的專案連線至 GitHub。如需 AWS CLI 搭配使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

1. 執行 `import-source-credentials` 命令：

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

即會在輸出中顯示 JSON 格式化資料。將資料複製到 AWS CLI 安裝的本機電腦或執行個體位置中的檔案（例如 `import-source-credentials.json`）。如下所示修改複製的資料，並儲存您的結果。

```
{
  "serverType": "server-type",
  "authType": "auth-type",
  "shouldOverwrite": "should-overwrite",
  "token": "token",
  "username": "username"
}
```

取代以下項目：

- ***server-type*** : 必要值。用於此登入資料的來源供應商。有效值為 GITHUB、BITBUCKET、GITHUB_ENTERPRISE、GITLAB 和 GITLAB_SELF_MANAGED。
 - ***auth-type*** : 必要值。用於連線至儲存庫的身分驗證類型。有效值為 OAUTH、BASIC_AUTH、Personal_ACCESS_TOKEN、CODECONNECTIONS 和 SECRETS_MANAGER。對於 GitHub，僅允許使用 Personal_ACCESS_TOKEN。僅允許使用 Bitbucket 應用程式密碼或 API 字串進行 BASIC_AUTH。
 - ***should-overwrite*** : 選用值。設為 false 可防止覆寫儲存庫來源登入資料。設為 true 可覆寫儲存庫來源登入資料。預設值為 true。
 - ***token*** : 必要值。對於 GitHub 或 GitHub Enterprise Server，這是個人存取字串。對於 Bitbucket，這是個人存取字串、應用程式密碼或 API 字串。對於身分類型 CODECONNECTIONS，這是連線 ARN。對於身分驗證類型 SECRETS_MANAGER，這是秘密 ARN。
 - ***username*** : 選用值。GitHub 和 GitHub Enterprise Server 來源提供者會忽略此參數。對於 Bitbucket 應用程式密碼，這是您的 Bitbucket 使用者名稱。對於 Bitbucket API 字串，這是與您的 Atlassian 帳戶相關聯的電子郵件地址。
2. 若要使用存取字串連接您的帳戶，請切換到包含您在步驟 1 中儲存的 `import-source-credentials.json` 檔案的目錄，並再次執行 `import-source-credentials` 命令。

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 格式的資料會出現在 Amazon Resource Name (ARN) 的輸出中。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

 Note

如果您使用相同的伺服器類型和身分驗證類型執行 `import-source-credentials` 命令第二次，即會更新存放的存取字串。

您的帳戶與存取字串連線後，您可以使用 `create-project` 來建立 CodeBuild 專案。如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。

- 若要檢視連接的存取字符，請執行 `list-source-credentials` 命令。

```
aws codebuild list-source-credentials
```

輸出中即會顯示 JSON 格式的 `sourceCredentialsInfos` 物件：

```
{
  "sourceCredentialsInfos": [
    {
      "authType": "auth-type",
      "serverType": "server-type",
      "arn": "arn"
    }
  ]
}
```

`sourceCredentialsObject` 包含連接的來源登入資料資訊的清單：

- `authType` 是登入資料使用的身分驗證類型。這可以是 `OAUTH`、`BASIC_AUTH`、`CODECONNECTIONS`、`PERSONAL_ACCESS_TOKEN`或 `SECRETS_MANAGER`。
 - `serverType` 是來源供應商的類型。這可以是 `GITHUB`、`GITHUB_ENTERPRISE`、`GITLAB`、`BITBUCKET`或 `GITLAB_SELF_MANAGED`。
 - `arn` 為字符的 ARN。
- 若要與來源供應商中斷連接並移除其存取字符，請使用其 ARN 執行 `delete-source-credentials` 命令。

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON 格式的資料會隨著所刪除登入資料的 ARN 傳回。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

GitHub OAuth 應用程式

使用 OAuth 連接 GitHub (主控台)

若要使用主控台，使用 OAuth 應用程式將專案連線至 GitHub，請在建立專案時執行下列動作。如需相關資訊，請參閱[建立組建專案 \(主控台\)](#)。

1. 針對 Source provider (來源供應商)，選擇 GitHub。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇使用帳戶登入資料，將帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線至 GitHub，請選擇管理帳戶登入資料。
 - b. 針對登入資料類型，選擇 OAuth 應用程式。
 - 如果您選擇使用 Service 的帳戶層級登入資料，請選擇您要用來存放字符的服務，並執行下列動作：
 - a. 如果您選擇使用 Secrets Manager，您可以選擇使用現有的秘密連線或建立新的秘密，然後選擇儲存。如需如何建立新秘密的詳細資訊，請參閱 [在 Secrets Manager 秘密中建立和存放權杖](#)。
 - b. 如果您選擇使用 CodeBuild，然後選擇儲存。
 - 選取僅使用此專案的覆寫登入資料來使用自訂來源登入資料來覆寫帳戶的登入資料設定。
 - a. 從填入的登入資料清單中，選擇 OAuth 應用程式下的其中一個選項。
 - b. 您也可以描述中選取建立新的 OAuth 應用程式字符連線，以建立新的 OAuth 應用程式字符。

若要檢閱授權的 OAuth 應用程式，請導覽至 GitHub 上的[應用程式](#)，並確認已列出由 [aws-codesuite](#) AWS CodeBuild (*region*)擁有且名為 的應用程式。

CodeBuild 中的 Bitbucket 存取

對於 Bitbucket，您可以使用存取字符、應用程式密碼、API 字符、OAuth 應用程式或 Bitbucket 連線來存取來源提供者。

主題

- [Bitbucket 應用程式連線](#)
- [Bitbucket 應用程式密碼、API 字符或存取字符](#)

- [Bitbucket OAuth 應用程式](#)

Bitbucket 應用程式連線

您可以使用 Bitbucket 與 CodeBuild 連線。透過 支援 Bitbucket 應用程式連線[AWS CodeConnections](#)。

Note

AWS CodeConnections 可在比 CodeBuild 更少的區域中使用。您可以在 CodeBuild 中使用跨區域連線。在選擇加入區域中建立的連線無法用於其他區域。如需詳細資訊，請參閱 [AWS CodeConnections 端點和配額](#)。

主題

- [步驟 1：建立與 Bitbucket 的連線（主控台）](#)
- [步驟 2：授予 CodeBuild 專案 IAM 角色存取權以使用連線](#)
- [步驟 3：設定 CodeBuild 以使用新的連線](#)
- [Bitbucket OAuth 輪換重新整理權杖](#)

步驟 1：建立與 Bitbucket 的連線（主控台）

使用這些步驟來使用 CodeBuild 主控台為 Bitbucket 中的專案新增連線。

建立連至 Bitbucket 的連線

- 請遵循開發人員工具使用者指南中[建立 Bitbucket 連線](#)的說明。

Note

您可以使用從另一個帳戶共用的連線，而不是在帳戶中建立或使用現有的連線 AWS。如需詳細資訊，請參閱[與 AWS 帳戶共用連線](#)。

步驟 2：授予 CodeBuild 專案 IAM 角色存取權以使用連線

您可以授予 CodeBuild 專案 IAM 角色存取權，以使用連線提供的 Bitbucket 權杖。

授予 CodeBuild 專案 IAM 角色存取權

1. 遵循 CodeBuild 專案的指示，為您的 CodeBuild 專案建立 [允許 CodeBuild 與其他 AWS 服務互動](#) IAM 角色。
2. 遵循指示時，將下列 IAM 政策新增至 CodeBuild 專案角色，以授予連線的存取權。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:GetConnectionToken",
        "codeconnections:GetConnection"
      ],
      "Resource": [
        "arn:aws:iam::*:role/Service*"
      ]
    }
  ]
}
```

步驟 3：設定 CodeBuild 以使用新的連線

您可以將連線設定為帳戶層級登入資料，並在專案中使用它。

AWS 管理主控台

若要在 中將連線設定為帳戶層級登入資料 AWS 管理主控台

1. 對於 Source provider (來源供應商)，選擇 Bitbucket。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇預設來源登入資料，以使用您帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線至 Bitbucket，請選擇管理預設來源憑證。
 - b. 針對登入資料類型，選擇 AWS CodeConnections。
 - c. 在連線中，選擇使用現有的連線或建立新的連線。

- 選擇自訂來源登入資料，以使用自訂來源登入資料覆寫您帳戶的預設設定。
 - a. 針對登入資料類型，選擇 AWS CodeConnections。
 - b. 在連線中，選擇使用現有的連線或建立新的連線。

AWS CLI

若要在 中將連線設定為帳戶層級登入資料 AWS CLI

- 開啟終端機 (Linux、macOS 或 Unix) 或命令提示 (Windows)。使用 AWS CLI 執行 `import-source-credentials` 命令，`--token` 為您的連線指定 `--auth-type`、`--server-type` 和 `--token`。

使用下列命令：

```
aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-type BITBUCKET --token <connection-arn>
```

如需在 CodeBuild 專案中設定多個字符的詳細資訊，請參閱 [將多個字符設定為來源層級登入資料](#)。

Bitbucket OAuth 輪換重新整理權杖

Atlassian 現在會強制執行 Bitbucket OAuth 的一次性輪換重新整理權杖。每次使用重新整理權杖時，Bitbucket 都會將其失效並傳回新的權杖。如需詳細資訊，請參閱 Atlassian 開發人員網站上的 [Bitbucket OAuth 單次使用重新整理權杖 \(CHANGE-3052\)](#)。

Secrets Manager 儲存的登入資料所需的動作

如果您在 Secrets Manager 中存放 Bitbucket OAuth 登入資料，請將 `secretsmanager:PutSecretValue` 許可新增至 CodeBuild 服務角色。使用此許可，CodeBuild 可以在每次權杖重新整理後，將更新的重新整理權杖寫回您的秘密。如果沒有此許可，字符重新整理後的下一個組建會失敗。

下列範例 IAM 政策陳述式會授予必要的許可：

```
{
  "Effect": "Allow",
  "Action": "secretsmanager:PutSecretValue",
  "Resource": "arn:aws:secretsmanager:*:*:secret:codebuild/*"
}
```

如果您使用 CodeBuild 受管選項，則不需要變更。CodeBuild 會自動處理字符輪換。

遷移至 AWS CodeConnections (建議)

我們建議遷移至 AWS CodeConnections 以進行 Bitbucket 整合。AWS CodeConnections 使用單獨的授權機制，輪換重新整理權杖不會影響。您不需要管理字符或新增許可。

若要遷移至 AWS CodeConnections，請完成下列步驟：

1. 開啟開發人員工具主控台。選擇設定、連線，然後選擇建立連線。選取 Bitbucket 做為提供者。
2. 在 Bitbucket 工作區中授權 AWS CodeConnections 應用程式。這需要在 Bitbucket 中管理工作區許可。
3. 將下列許可新增至 CodeBuild 服務角色：`codeconnections:GetConnectionToken`和 `codeconnections:GetConnection`。
4. 將連線設定為您的預設登入資料：

```
aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-type BITBUCKET --token <connection-arn>
```

5. 所有使用預設登入資料切換的專案都會自動執行。個別更新使用自訂每個專案登入資料的專案。使用 主控台或 `UpdateProject` API。

Bitbucket 應用程式密碼、API 字符或存取字符

Bitbucket 應用程式密碼已棄用

Atlassian 已棄用 Bitbucket 應用程式密碼，以支持 API 字符。若要繼續搭配 CodeBuild 使用基本身分驗證，請在 Atlassian 帳戶設定中建立 API 字符。當您將其連接到 CodeBuild 時，請使用與您的 Atlassian 帳戶相關聯的電子郵件地址做為使用者名稱。CodeBuild 會自動使用正確的登入資料進行 Git 操作和 Bitbucket API 呼叫。

如需詳細資訊，請參閱 Atlassian 網站上的[使用 API 字符](#)。

Bitbucket OAuth 字符輪換

如果您使用 Bitbucket OAuth 登入資料，建議您遷移至 [Bitbucket 應用程式連線](#)。如需繼續搭配 Secrets Manager 使用 OAuth 時所需變更的相關資訊，請參閱 [Bitbucket OAuth 應用程式](#)。

先決條件

開始之前，您必須將適當的許可範圍新增至 Bitbucket 應用程式密碼、API 字符或存取字符。

對於 Bitbucket，您的應用程式密碼、API 字符或存取權杖必須具有下列範圍。

- repository:read：授予獲授權使用者可存取的所有儲存庫之讀取存取權。
- pullrequest:read：授予提取請求的讀取存取權。如果您的專案具有 Bitbucket Webhook，則您的應用程式密碼或存取權杖必須具有此範圍。
- webhook：授予 Webhook 的存取權。如果您的專案具有 Webhook 操作，則您的應用程式密碼或存取權杖必須具有此範圍。
- 帳戶：授予使用者帳戶資訊的讀取存取權。

如需詳細資訊，請參閱 Bitbucket 網站上的 [Bitbucket Cloud REST API 的範圍](#) 和 [Bitbucket Cloud 上的 OAuth](#)。

使用應用程式密碼或 API 字符（主控台）連接 Bitbucket

若要使用主控台，使用應用程式密碼或 API 字符將專案連線至 Bitbucket，請在建立專案時執行下列動作。如需相關資訊，請參閱 [建立組建專案 \(主控台\)](#)。

Note

如果您使用 API 字符，請輸入與您的 Atlassian 帳戶相關聯的電子郵件地址做為使用者名稱。

1. 對於 Source provider (來源供應商)，選擇 Bitbucket。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇使用帳戶登入資料，將帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線至 Bitbucket，請選擇管理帳戶登入資料。
 - b. 針對登入資料類型，選擇 API 字符（先前顯示為應用程式密碼）。
 - 如果您選擇使用 Service 的帳戶層級登入資料，請選擇您要用來存放字符的服務，並執行下列動作：
 - a. 如果您選擇使用 Secrets Manager，您可以選擇使用現有的秘密連線或建立新的秘密，然後選擇儲存。如需如何建立新秘密的詳細資訊，請參閱 [在 Secrets Manager 秘密中建立和存放權杖](#)。

- b. 如果您選擇使用 CodeBuild，請輸入您的 Bitbucket 使用者名稱和密碼，然後選擇儲存。對於 Bitbucket API 字符，請改為輸入您的 Atlassian 帳戶電子郵件地址和 API 字符。
- 選取僅使用此專案的覆寫登入資料來使用自訂來源登入資料來覆寫帳戶的登入資料設定。
 - a. 從填入的登入資料清單中，選擇應用程式密碼或 API 字符下的其中一個選項。
 - b. 您也可以在此描述中選取建立連線連結，以建立新的應用程式密碼或 API 字符連線。

使用存取字符連接 Bitbucket（主控台）

若要使用主控台，使用存取字符將專案連線至 Bitbucket，請在建立專案時執行下列動作。如需相關資訊，請參閱[建立組建專案 \(主控台\)](#)。

1. 對於 Source provider (來源供應商)，選擇 Bitbucket。
2. 對於登入資料，請執行下列其中一項操作：
 - 選擇使用帳戶登入資料，將帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線至 Bitbucket，請選擇管理帳戶登入資料。
 - b. 針對登入資料類型，選擇個人存取字符。
 - 如果您選擇使用 Service 的帳戶層級登入資料，請選擇您要用來存放字符的服務，並執行下列動作：
 - a. 如果您選擇使用 Secrets Manager，您可以選擇使用現有的秘密連線或建立新的秘密，然後選擇儲存。如需如何建立新秘密的詳細資訊，請參閱 [在 Secrets Manager 秘密中建立和存放權杖](#)。
 - b. 如果您選擇使用 CodeBuild，請輸入您的 Bitbucket 個人存取字符，然後選擇儲存。
 - 選取僅使用此專案的覆寫登入資料來使用自訂來源登入資料來覆寫帳戶的登入資料設定。
 - a. 從填入的登入資料清單中，選擇個人存取字符下的其中一個選項。
 - b. 您也可以在此描述中選取建立新的個人存取權杖連線，以建立新的個人存取權杖。

使用應用程式密碼或存取權杖 (CLI) 連接 Bitbucket

請依照下列步驟使用 AWS CLI，使用應用程式密碼或存取權杖將您的專案連線至 Bitbucket。如需 AWS CLI 搭配使用的詳細資訊 AWS CodeBuild，請參閱 [命令列參考](#)。

1. 執行 import-source-credentials 命令：


```
aws codebuild import-source-credentials --generate-cli-skeleton
```

即會在輸出中顯示 JSON 格式化資料。將資料複製到 AWS CLI 安裝的本機電腦或執行個體位置中的檔案（例如 *import-source-credentials.json*）。如下所示修改複製的資料，並儲存您的結果。

```
{
  "serverType": "BITBUCKET",
  "authType": "auth-type",
  "shouldOverwrite": "should-overwrite",
  "token": "token",
  "username": "username"
}
```

取代以下項目：

- *server-type*：必要值。用於此登入資料的來源供應商。有效值為 GITHUB、BITBUCKET、GITHUB_ENTERPRISE、GITLAB 和 GITLAB_SELF_MANAGED。
 - *auth-type*：必要值。用於連線至儲存庫的身分驗證類型。有效值為 OAUTH、BASIC_AUTH、Personal_ACCESS_TOKEN、CODECONNECTIONS 和 SECRETS_MANAGER。對於 GitHub，僅允許使用 Personal_ACCESS_TOKEN。僅允許使用 Bitbucket 應用程式密碼或 API 字符進行 BASIC_AUTH。
 - *should-overwrite*：選用值。設為 false 可防止覆寫儲存庫來源登入資料。設為 true 可覆寫儲存庫來源登入資料。預設值為 true。
 - *token*：必要值。對於 GitHub 或 GitHub Enterprise Server，這是個人存取字符。對於 Bitbucket，這是個人存取字符、應用程式密碼或 API 字符。對於身分驗證類型 CODECONNECTIONS，這是連線 ARN。對於身分驗證類型 SECRETS_MANAGER，這是秘密 ARN。
 - *username*：選用值。GitHub 和 GitHub Enterprise Server 來源提供者會忽略此參數。對於 Bitbucket 應用程式密碼，這是您的 Bitbucket 使用者名稱。對於 Bitbucket API 字符，這是與您的 Atlassian 帳戶相關聯的電子郵件地址。
2. 若要使用應用程式密碼、API 字符或存取字符來連接您的帳戶，請切換到包含您在步驟 1 中所儲存 *import-source-credentials.json* 檔案的目錄，然後再次執行 *import-source-credentials* 命令。

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 格式的資料會出現在 Amazon Resource Name (ARN) 的輸出中。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

 Note

如果您使用相同的伺服器類型和身分驗證類型執行 import-source-credentials 命令第二次，即會更新存放的存取字符。

在您的帳戶與應用程式密碼或 API 字符連線後，您可以使用 create-project 來建立 CodeBuild 專案。如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)。

- 若要檢視連線的應用程式密碼或存取權杖，請執行 list-source-credentials 命令。

```
aws codebuild list-source-credentials
```

輸出中即會顯示 JSON 格式的 sourceCredentialsInfos 物件：

```
{
  "sourceCredentialsInfos": [
    {
      "authType": "auth-type",
      "serverType": "BITBUCKET",
      "arn": "arn"
    }
  ]
}
```

sourceCredentialsObject 包含連接的來源登入資料資訊的清單：

- `authType` 是登入資料使用的身份驗證類型。這可以是 `OAUTH`、`BASIC_AUTH`、`CODECONNECTIONS`、`PERSONAL_ACCESS_TOKEN` 或 `SECRETS_MANAGER`。
 - `serverType` 是來源供應商的類型。這可以是 `GITHUB`、`GITHUB_ENTERPRISE`、`GITLAB`、`BITBUCKET` 或 `GITLAB_SELF_MANAGED`。
 - `arn` 為字符的 ARN。
4. 若要中斷與來源提供者的連線並移除其應用程式密碼或存取權杖，請使用其 ARN 執行 `delete-source-credentials` 命令。

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON 格式的資料會隨著所刪除登入資料的 ARN 傳回。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

Bitbucket OAuth 應用程式

Bitbucket OAuth 權杖輪換所需的動作

Atlassian 現在會強制執行 Bitbucket OAuth 的一次性輪換重新整理權杖。建議您遷移至 [Bitbucket 應用程式連線](#)，這會完全消除字符管理。如需遷移的指示，請參閱 [遷移至 AWS CodeConnections \(建議\)](#)。

如果您偏好繼續使用 OAuth 搭配 Secrets Manager，請將 `secretsmanager:PutSecretValue` 許可新增至 CodeBuild 服務角色。如需詳細資訊，請參閱 [Secrets Manager 儲存的登入資料所需的動作](#)。

如果您使用 CodeBuild 受管選項，則不需要變更。CodeBuild 會自動處理字符輪換。

使用 OAuth 連接 Bitbucket (主控台)

若要使用 主控台，使用 OAuth 應用程式將專案連線至 Bitbucket，請在建立專案時執行下列動作。如需相關資訊，請參閱 [建立組建專案 \(主控台\)](#)。

1. 對於 Source provider (來源供應商)，選擇 Bitbucket。

2. 對於登入資料，請執行下列其中一項操作：

- 選擇使用帳戶登入資料，將帳戶的預設來源登入資料套用至所有專案。
 - a. 如果您未連線至 Bitbucket，請選擇管理帳戶登入資料。
 - b. 針對登入資料類型，選擇 OAuth 應用程式。
- 如果您選擇使用 Service 的帳戶層級登入資料，請選擇您要用來存放字符的服務，然後執行下列動作：
 - a. 如果您選擇使用 Secrets Manager，您可以選擇使用現有的秘密連線或建立新的秘密，然後選擇儲存。如需建立新秘密的詳細資訊，請參閱 [在 Secrets Manager 秘密中建立和存放權杖](#)。
 - b. 如果您選擇使用 CodeBuild，然後選擇儲存。
- 選取僅使用此專案的覆寫登入資料來使用自訂來源登入資料來覆寫帳戶的登入資料設定。
 - a. 從填入的登入資料清單中，選擇 OAuth 應用程式下的其中一個選項。
 - b. 您也可以描述中選取建立新的 OAuth 應用程式字符連線，以建立新的 OAuth 應用程式字符。

若要檢閱授權的 OAuth 應用程式，請導覽至 Bitbucket [上的應用程式授權](#)，並確認 AWS CodeBuild (*region*) 已列出名為 的應用程式。

CodeBuild 中的 GitLab 存取

對於 GitLab，您可以使用 GitLab 連線來存取來源提供者。

主題

- [將 CodeBuild 連接至 GitLab](#)

將 CodeBuild 連接至 GitLab

連線可讓您授權和建立組態，將第三方供應商與使用 AWS 的資源建立關聯 AWS CodeConnections。若要將第三方儲存庫關聯為建置專案的來源，請使用 連線。

若要在 CodeBuild 中新增 GitLab 或 GitLab 自我管理來源提供者，您可以選擇：

- 使用 CodeBuild 主控台建立建置專案精靈或編輯來源頁面，選擇 GitLab 或 GitLab 自我管理提供者選項。請參閱 [建立連至 GitLab 的連線 \(主控台\)](#) 以新增來源提供者。主控台可協助您建立連線資源。

- 使用 CLI 建立連線資源，請參閱[建立連至 GitLab 的連線 \(CLI\)](#)使用 CLI 建立連線資源。

 Note

您也可以使用設定下的開發人員工具主控台建立連線。請參閱[建立連線](#)。

 Note

在 GitLab 中授權此連線安裝，即表示您授予我們的服務透過存取您的帳戶處理您的資料的許可，您隨時可以解除安裝應用程式以撤銷許可。

建立連至 GitLab 的連線

本節說明如何將 GitLab 連線至 CodeBuild。如需 GitLab 連線的詳細資訊，請參閱[將 CodeBuild 連接至 GitLab](#)。

開始之前：

- 您必須已在 GitLab 建立帳戶。

 Note

連線只能存取用於建立和授權連線之帳戶擁有的儲存庫。

 Note

您可以建立與 GitLab 中擁有擁有者角色之儲存庫的連線，然後該連線可與具有 CodeBuild 等資源的儲存庫搭配使用。如果是群組中的儲存庫，您不需要為群組擁有者。

- 若要指定建置專案的來源，您必須已在 GitLab 上建立儲存庫。

主題

- [建立連至 GitLab 的連線 \(主控台\)](#)
- [建立連至 GitLab 的連線 \(CLI\)](#)

建立連至 GitLab 的連線 (主控台)

使用這些步驟來使用 CodeBuild 主控台，在 GitLab 中為您的專案（儲存庫）新增連線。

Note

您可以使用從另一個帳戶共用的連線，而不是在帳戶中建立或使用現有的連線 AWS。如需詳細資訊，請參閱[與 AWS 帳戶共用連線](#)。

建立或編輯您的建置專案

1. 登入 CodeBuild 主控台。
2. 選擇下列其中一項。
 - 選擇以建立建置專案。依照 中的步驟[建立組建專案 \(主控台\)](#)完成第一個畫面，然後在來源提供者下的來源區段中，選擇 GitLab。
 - 選擇以編輯現有的建置專案。選擇編輯，然後選擇來源。在編輯來源頁面的來源提供者下，選擇 GitLab。
3. 選擇下列其中一項：
 - 在連線下，選擇預設連線。預設連線會將預設 GitLab 連線套用至所有專案。
 - 在連線下，選擇自訂連線。自訂連線會套用自訂 GitLab 連線，覆寫您帳戶的預設設定。
4. 執行以下任意一項：
 - 在預設連線或自訂連線下，如果您尚未建立與提供者的連線，請選擇建立新的 GitLab 連線。繼續進行步驟 5 以建立連線。
 - 在連線下，如果您已建立與供應商的連線，請選擇連線。繼續步驟 10。

Note

如果您在建立 GitLab 連線之前關閉快顯視窗，則需要重新整理頁面。

5. 若要建立連至 GitLab 儲存庫的連線，請在選取供應商底下選擇 GitLab。在 Connection name (連線名稱) 底下，輸入您要建立的連線名稱。選擇連線至 GitLab。

[Developer Tools](#) > [Connections](#) > [Create connection](#)

Create a connection [Info](#)

Create GitLab connection [Info](#)

Connection name

► **Tags - optional**

Connect to GitLab

6. 當 GitLab 的登入頁面顯示時，使用您的登入資料登入，然後選擇登入。
7. 如果這是您第一次授權連線，則會顯示授權頁面，其中包含請求連線授權以存取 GitLab 帳戶的訊息。

選擇 Authorize (授權)。

Authorize **AWS Connector for GitLab** to use your account?

An application called **AWS Connector for GitLab** is requesting access to your GitLab account. This application was created by **Amazon AWS**. Please note that this application is not provided by GitLab and you should verify its authenticity before allowing access.

This application will be able to:

- **Access the authenticated user's API**

Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.

- **Read the authenticated user's personal information**

Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

- **Read Api**

Grants read access to the API, including all groups and projects, the container registry, and the package registry.

- **Allows read-only access to the repository**

Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

- **Allows read-write access to the repository**

Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

8. 瀏覽器會返回連線主控台頁面。在 GitLab 連線設定下，新的連線會顯示在連線名稱中。
9. 選擇連線。

成功建立 GitLab 連線後，頂端會顯示成功橫幅。

10. 在建立建置專案頁面的預設連線或自訂連線下拉式清單中，確定已列出您的連線 ARN。如果沒有，請選擇重新整理按鈕使其出現。
11. 在儲存庫中，使用命名空間指定專案路徑，以選擇 GitLab 中的專案名稱。例如，對於群組層級儲存庫，以下列格式輸入儲存庫名稱：group-name/repository-name。如需路徑和命名空間的詳細資訊，請參閱 <https://docs.gitlab.com/ee/api/projects.html#get-single-project> 中的 path_with_namespace 欄位。如需 GitLab 中命名空間的詳細資訊，請參閱 <https://docs.gitlab.com/ee/user/namespace/>。

Note

對於 GitLab 中的群組，您必須使用命名空間手動指定專案路徑。例如，針對群組 myrepo 中名為 mygroup 的儲存庫，輸入下列項目：mygroup/myrepo。您可以在 GitLab 的 URL 中找到具有命名空間的專案路徑。

12. 在來源版本 - 選用中，輸入提取請求 ID、分支、遞交 ID、標籤或參考和遞交 ID。如需詳細資訊，請參閱 [使用的來源版本範例 AWS CodeBuild](#)。

Note

我們建議您選擇看起來不像遞交 IDs 的 Git 分支名稱，例如 811dd1ba1aba14473856cee38308caed7190c0d 或 5392f7。這可協助您避免 Git 結帳與實際遞交的衝突。

13. 在 Git 複製深度 - 選用中，您可以使用截斷至指定遞交數量的歷史記錄來建立淺複製。如果您想要完整複製，請選擇 Full (完整)。
14. 在建置狀態 - 選用中，如果您希望向來源提供者報告建置的開始和完成狀態，請選取在建置開始和完成時向來源提供者報告建置狀態。

若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱 [來源提供者存取](#)。

建立連至 GitLab 的連線 (CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來建立連線。

若要這麼做，請使用 `create-connection` 命令。

Important

根據預設，透過 AWS CLI 或 建立的連線 AWS CloudFormation 處於 PENDING 狀態。建立與 CLI 或 的連線後 CloudFormation，請使用 主控台來編輯連線，使其成為狀態 AVAILABLE。

建立連線

- 請遵循開發人員工具主控台使用者指南中[建立 GitLab \(CLI\) 連線](#)的說明。

預防跨服務混淆代理人

混淆代理人問題屬於安全性問題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 AWS 中，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況，AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

我們建議在資源政策中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容索引鍵，以限制將另一個服務 AWS CodeBuild 提供給資源的許可。如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 `aws:SourceArn`。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 `aws:SourceAccount`。

防範混淆代理人問題的最有效方法是使用 `aws:SourceArn` 全域條件內容索引鍵，以及資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 `aws:SourceArn` 全域內容條件索引鍵搭配萬用字元 (*) 來表示 ARN 的未知部分。例如 `arn:aws:codebuild:*:123456789012:*`。

如果 `aws:SourceArn` 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 ARN)，您必須使用這兩個全域條件內容索引鍵來限制許可。

的值 `aws:SourceArn` 必須是 CodeBuild 專案 ARN。

下列範例示範如何使用 CodeBuild 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容索引鍵，以防止混淆代理人問題。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:codebuild:us-east-1:111122223333:project/MyProject"
        }
      }
    }
  ]
}
```

進階主題

本小節包含數個進階主題，對於較熟悉 AWS CodeBuild 的使用者極為實用。

主題

- [允許使用者與 CodeBuild 互動](#)
- [允許 CodeBuild 與其他 AWS 服務互動](#)
- [使用客戶受管金鑰加密建置輸出](#)
- [使用與 CodeBuild 互動 AWS CLI](#)
- [的命令列參考 AWS CodeBuild](#)
- [的AWS SDKs和工具參考 AWS CodeBuild](#)
- [搭配 AWS SDK 使用此服務](#)
- [指定 AWS CodeBuild 端點](#)
- [AWS CodeBuild 搭配使用 AWS CodePipeline 來測試程式碼和執行組建](#)
- [AWS CodeBuild 搭配 Codecov 使用](#)
- [AWS CodeBuild 與 Jenkins 搭配使用](#)
- [AWS CodeBuild 搭配無伺服器應用程式使用](#)
- [AWS CodeBuild 適用於 Windows 的第三方通知](#)
- [使用 CodeBuild 條件索引鍵做為 IAM 服務角色變數，以控制組建存取](#)
- [AWS CodeBuild 條件索引鍵](#)

允許使用者與 CodeBuild 互動

如果您 AWS CodeBuild 第一次遵循 中的步驟[開始使用主控台](#)來存取，則很可能不需要本主題中的資訊。不過，當您繼續使用 CodeBuild 時，您可能想要執行一些動作，例如讓組織中的其他使用者和群組能夠與 CodeBuild 互動。

若要允許 IAM 使用者或群組與之互動 AWS CodeBuild，您必須授予他們 CodeBuild 的存取許可。本節說明如何使用 IAM 主控台或執行此操作 AWS CLI。

如果您將使用 AWS 根帳戶（不建議）或 AWS 帳戶中的管理員使用者存取 CodeBuild，則不需要遵循這些指示。

如需有關 AWS 根帳戶和管理員使用者的資訊，請參閱 [《使用者指南》中的 AWS 帳戶 根使用者和建立您的第一個 AWS 帳戶 根使用者和群組](#)。

將 CodeBuild 存取許可新增至 IAM 群組或使用者（主控台）

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。

您應該已使用下列其中一項 AWS 管理主控台 登入：

- 您的 AWS 根帳戶。此為不建議的選項。如需詳細資訊，請參閱 [《使用者指南》中的 AWS 帳戶 根使用者](#)。
- 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱 [《使用者指南》中的 建立您的第一個 AWS 帳戶 根使用者和群組](#)。
- 您 AWS 帳戶中具有執行下列最低動作集之許可的使用者：

```
iam:AttachGroupPolicy
iam:AttachUserPolicy
iam:CreatePolicy
iam>ListAttachedGroupPolicies
iam>ListAttachedUserPolicies
iam>ListGroups
iam>ListPolicies
iam>ListUsers
```

如需詳細資訊，請參閱 [《使用者指南》中的 IAM 政策概觀](#)。

2. 在導覽窗格中，選擇政策。
3. 若要將一組自訂 AWS CodeBuild 存取許可新增至 IAM 群組或 IAM 使用者，請跳至此程序中的步驟 4。

若要將一組預設的 CodeBuild 存取許可新增至 IAM 群組或 IAM 使用者，請選擇政策類型、AWS 受管，然後執行下列動作：

- 若要新增 CodeBuild 的完整存取許可，請選取名為 `AWSCodeBuildAdminAccess` 的方塊，選擇政策動作，然後選擇連接。選取目標 IAM 群組或使用者旁的方塊，然後選擇連接政策。對名為 `AmazonS3ReadOnlyAccess` 和 `IAMFullAccess` 的政策，重複此步驟。
- 若要針對組建專案管理以外的所有項目新增 CodeBuild 存取許可，請選取名為 `AWSCodeBuildDeveloperAccess` 的方塊，選擇政策動作，然後選擇連接。選取目標 IAM 群組或使用者旁的方塊，然後選擇連接政策。對名為 `AmazonS3ReadOnlyAccess` 的政策，重複此步驟。

- 若要將唯讀存取許可新增至 CodeBuild，請選取名為 `AWSCodeBuildReadOnlyAccess` 的方塊。選取目標 IAM 群組或使用者旁的方塊，然後選擇連接政策。對名為 `AmazonS3ReadOnlyAccess` 的政策，重複此步驟。

您現在已將一組預設的 CodeBuild 存取許可新增至 IAM 群組或使用者。略過此程序的其餘步驟。

- 選擇建立政策。
- 在建立政策頁面的建立您自己的政策旁，選擇選取。
- 在 Review Policy (檢閱政策) 頁面的 Policy Name (政策名稱) 中，輸入政策的名稱 (例如 **CodeBuildAccessPolicy**)。如果您使用不同的名稱，請務必在本程序中一律使用該名稱。
- 針對 Policy Document (政策文件)，輸入下列項目，然後選擇 Create Policy (建立政策)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::111122223333:role/role-name"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
```

```

        "Resource": "*"
    },
    {
        "Sid": "S3AccessPolicy",
        "Effect": "Allow",
        "Action": [
            "s3:CreateBucket",
            "s3:GetObject",
            "s3:List*",
            "s3:PutObject"
        ],
        "Resource": "*"
    },
    {
        "Sid": "S3BucketIdentity",
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketAcl",
            "s3:GetBucketLocation"
        ],
        "Resource": "*"
    }
]
}

```

Note

此政策允許存取所有 CodeBuild 動作和可能大量的 AWS 資源。若要限制特定 CodeBuild 動作的許可，請在 CodeBuild 政策陳述式 `codebuild:*` 中變更的值。如需詳細資訊，請參閱 [身分與存取管理](#)。若要限制對特定 AWS 資源的存取，請變更 `Resource` 物件的值。如需詳細資訊，請參閱 [身分與存取管理](#)。

8. 在導覽窗格中，選擇 Groups (群組) 或 Users (使用者)。
9. 在群組或使用者清單中，選擇您要新增 CodeBuild 存取許可的 IAM 群組或 IAM 使用者名稱。
10. 如果是群組，請在群組設定頁面的 Permissions (許可) 索引標籤上，展開 Managed Policies (受管政策)，然後選擇 Attach Policy (連接政策)。

如果是使用者，請在使用者設定頁面的 Permissions (許可) 索引標籤上，選擇 Add permissions (新增許可)。

11. 對於群組，在連接政策頁面上，選取 CodeBuildAccessPolicy，然後選擇連接政策。

對於使用者，在新增許可頁面上，選擇直接連接現有政策。選取 CodeBuildAccessPolicy，選擇下一步：檢閱，然後選擇新增許可。

將 CodeBuild 存取許可新增至 IAM 群組或使用者 (AWS CLI)

1. 請確定您已 AWS CLI 使用對應至其中一個 IAM 實體的 AWS 存取金鑰和 AWS 私密存取金鑰來設定，如先前程序所述。如需詳細資訊，請參閱AWS Command Line Interface 《使用者指南》中的[設定 AWS Command Line Interface](#)。
2. 若要將一組自訂 AWS CodeBuild 存取許可新增至 IAM 群組或 IAM 使用者，請跳至此程序的步驟 3。

若要將一組預設的 CodeBuild 存取許可新增至 IAM 群組或 IAM 使用者，請執行下列動作：

執行下列其中一個命令，取決於您要將許可新增至 IAM 群組或使用者：

```
aws iam attach-group-policy --group-name group-name --policy-arn policy-arn  
  
aws iam attach-user-policy --user-name user-name --policy-arn policy-arn
```

您必須執行 命令三次，以 IAM 群組名稱或使用者名稱取代 *group-name* 或 *user-name*，並針對下列每個政策 Amazon Resource Name (ARNs) 取代 *policy-arn* 一次：

- 若要將完整存取許可新增至 CodeBuild，請使用下列政策 ARNs：
 - arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess
 - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
 - arn:aws:iam::aws:policy/IAMFullAccess
- 若要針對組建專案管理以外的所有項目新增 CodeBuild 的存取許可，請使用下列政策 ARNs：
 - arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess
 - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
- 若要將唯讀存取許可新增至 CodeBuild，請使用下列政策 ARNs：
 - arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess
 - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess

您現在已將一組預設的 CodeBuild 存取許可新增至 IAM 群組或使用者。略過此程序的其餘步驟。

3. 在 AWS CLI 安裝的本機工作站或執行個體上的空目錄中，建立名為 `put-group-policy.json` 或的檔案 `put-user-policy.json`。如果您使用不同的檔案名稱，請務必在本程序中一律使用該檔案名稱。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::111122223333:role/role-name"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3AccessPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:GetObject",
        "s3:List*",
        "s3:PutObject"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "*"
  },
  {
    "Sid": "S3BucketIdentity",
    "Effect": "Allow",
    "Action": [
      "s3:GetBucketAcl",
      "s3:GetBucketLocation"
    ],
    "Resource": "*"
  }
]
}

```

Note

此政策允許存取所有 CodeBuild 動作和可能大量的 AWS 資源。若要限制特定 CodeBuild 動作的許可，請在 CodeBuild 政策陳述式 `codebuild:*` 中變更 的值。如需詳細資訊，請參閱 [身分與存取管理](#)。若要限制對特定 AWS 資源的存取，請變更相關 `Resource` 物件的值。如需詳細資訊，請參閱 [身分與存取管理](#) 或特定 AWS 服務的安全文件。

4. 切換到您已儲存檔案的目錄，然後執行下列其中一個命令。您可以對 `CodeBuildGroupAccessPolicy` 和 `CodeBuildUserAccessPolicy` 使用不同的值。如果您使用不同的值，請務必在此處使用這些值。

對於 IAM 群組：

```
aws iam put-group-policy --group-name group-name --policy-name
CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
```

對於使用者：

```
aws iam put-user-policy --user-name user-name --policy-name
CodeBuildUserAccessPolicy --policy-document file://put-user-policy.json
```

在上述命令中，將 *group-name* 或 *user-name* 取代為目標 IAM 群組或使用者名稱。

允許 CodeBuild 與其他 AWS 服務互動

如果您 AWS CodeBuild 第一次遵循 中的步驟[開始使用主控台](#)來存取，則很可能不需要本主題中的資訊。不過，當您繼續使用 CodeBuild 時，您可能想要執行一些動作，例如允許 CodeBuild 與其他 AWS 服務互動。

若要允許 CodeBuild 代表您與相依 AWS 服務互動，您需要 AWS CodeBuild 服務角色。您可以使用 CodeBuild 或 AWS CodePipeline 主控台建立 CodeBuild 服務角色。如需相關資訊，請參閱：

- [建立組建專案 \(主控台\)](#)
- [建立使用 CodeBuild \(CodePipeline 主控台\) 的管道](#)
- [將 CodeBuild 組建動作新增至管道 \(CodePipeline 主控台\)](#)
- [變更建置專案的設定 \(主控台\)](#)

如果您不打算使用這些主控台，本節說明如何使用 IAM 主控台或 建立 CodeBuild 服務角色 AWS CLI。

Important

CodeBuild 會將 服務角色用於代表您執行的所有操作。如果該角色包含使用者不該有的許可，您可能在無意中提升使用者的許可。確定該角色授予[最低權限](#)。

此頁面所述的服務角色包含政策，授予使用 CodeBuild 所需的最低許可。視您的使用案例而定，您可能需要新增其他許可。

建立 CodeBuild 服務角色 (主控台)

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。

您應該已使用下列其中一項登入主控台：

- 您的 AWS 根帳戶。此為不建議的選項。如需詳細資訊，請參閱 [《使用者指南》中的 AWS 帳戶根使用者](#)。
- 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱 [《使用者指南》中的建立您的第一個 AWS 帳戶 根使用者和群組](#)。
- 您 AWS 帳戶中具有執行下列最低動作集之許可的使用者：

```
iam:AddRoleToInstanceProfile
```

```
iam:AttachRolePolicy
iam:CreateInstanceProfile
iam:CreatePolicy
iam:CreateRole
iam:GetRole
iam:ListAttachedRolePolicies
iam:ListPolicies
iam:ListRoles
iam:PassRole
iam:PutRolePolicy
iam:UpdateAssumeRolePolicy
```

如需詳細資訊，請參閱《使用者指南》中的 [IAM 政策概觀](#)。

2. 在導覽窗格中，選擇政策。
3. 選擇建立政策。
4. 在 Create Policy (建立政策) 頁面上，選擇 JSON。
5. 針對 JSON 政策，輸入下列內容，然後選擇 Review Policy (檢閱政策)：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ],
}
```

```
{
  "Sid": "S3GetObjectPolicy",
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:GetObjectVersion"
  ],
  "Resource": "*"
},
{
  "Sid": "S3PutObjectPolicy",
  "Effect": "Allow",
  "Action": [
    "s3:PutObject"
  ],
  "Resource": "*"
},
{
  "Sid": "ECRPullPolicy",
  "Effect": "Allow",
  "Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
  ],
  "Resource": "*"
},
{
  "Sid": "ECRAuthPolicy",
  "Effect": "Allow",
  "Action": [
    "ecr:GetAuthorizationToken"
  ],
  "Resource": "*"
},
{
  "Sid": "S3BucketIdentity",
  "Effect": "Allow",
  "Action": [
    "s3:GetBucketAcl",
    "s3:GetBucketLocation"
  ],
  "Resource": "*"
}
```

```
]
}
```

Note

此政策包含允許存取可能大量 AWS 資源的陳述式。若要限制 AWS CodeBuild 存取特定 AWS 資源，請變更Resource陣列的值。如需詳細資訊，請參閱 AWS 服務的安全文件。

- 在 Review Policy (檢閱政策) 頁面上，針對 Policy Name (政策名稱)，輸入政策的名稱 (例如，**CodeBuildServiceRolePolicy**)，然後選擇 Create policy (建立政策)。

Note

如果您使用不同的名稱，請務必在本程序中一律使用該名稱。

- 在導覽窗格中，選擇 Roles (角色)。
- 選擇 Create Role (建立角色)。
- 在建立角色頁面上，在已選取AWS 服務的情況下，選擇 CodeBuild，然後選擇下一步：許可。
- 在連接許可政策頁面上，選取 CodeBuildServiceRolePolicy，然後選擇下一步：檢閱。
- 在 Create role and review (建立角色和檢閱) 頁面上，針對 Role name (角色名稱)，輸入角色的名稱 (例如 **CodeBuildServiceRole**)，然後選擇 Create role (建立角色)。

建立 CodeBuild 服務角色 (AWS CLI)

- 請確定您已 AWS CLI 使用對應至其中一個 IAM 實體的 AWS 存取金鑰和 AWS 私密存取金鑰來設定，如先前程序所述。如需詳細資訊，請參閱AWS Command Line Interface 《使用者指南》中的[設定 AWS Command Line Interface](#)。
- 在 AWS CLI 安裝的本機工作站或執行個體上的空目錄中，建立兩個名為 create-role.json和的檔案put-role-policy.json。如果您選擇不同的檔案名稱，請務必在本程序中一律使用這些檔案名稱。

create-role.json:

JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }
]
}

```

Note

建議您使用 `aws:SourceAccount` 和 `aws:SourceArn` 條件索引鍵，保護自己免受[混淆代理人問題](#)的困擾。例如，您可以使用下列條件區塊編輯先前的信任政策。`aws:SourceAccount` 是 CodeBuild 專案的擁有者，而 `aws:SourceArn` 是 CodeBuild 專案 ARN。

如果您想要將您的服務角色限制為 AWS 帳戶，看起來 `create-role.json` 可能會類似：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "111122223333"
          ]
        }
      }
    }
  ]
}

```

```
    }
  ]
}
```

如果您想要將您的服務角色限制為特定的 CodeBuild 專案，`create-role.json` 看起來可能會類似：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:codebuild:us-east-1:111122223333:project/MyProject"
        }
      }
    }
  ]
}
```

Note

如果您不知道或尚未決定 CodeBuild 專案的名稱，並想要對特定 ARN 模式設定信任政策限制，您可以使用萬用字元 (*) 取代 ARN 的該部分。建立專案之後，您就可以更新信任政策。

`put-role-policy.json`:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
```

```
"Effect": "Allow",
"Action": [
    "s3:GetBucketAcl",
    "s3:GetBucketLocation"
],
"Resource": "*"
}
]
```

Note

此政策包含允許存取可能大量 AWS 資源的陳述式。若要限制 AWS CodeBuild 存取特定 AWS 資源，請變更Resource陣列的值。如需詳細資訊，請參閱 AWS 服務的安全文件。

3. 切換到您已儲存上述檔案的目錄，然後執行下列兩個命令，按此順序一次執行一個。您可以對 CodeBuildServiceRole 和 CodeBuildServiceRolePolicy 使用不同的值，但務必在此處使用這些值。

```
aws iam create-role --role-name CodeBuildServiceRole --assume-role-policy-document
file://create-role.json
```

```
aws iam put-role-policy --role-name CodeBuildServiceRole --policy-name
CodeBuildServiceRolePolicy --policy-document file://put-role-policy.json
```

使用客戶受管金鑰加密建置輸出

如果您 AWS CodeBuild 第一次遵循 中的步驟[開始使用主控台](#)來存取，則很可能不需要本主題中的資訊。不過，當您繼續使用 CodeBuild 時，您可能想要執行一些動作，例如加密建置成品。

為了 AWS CodeBuild 讓 加密其建置輸出成品，它需要存取 KMS 金鑰。根據預設，CodeBuild 會使用您 AWS 帳戶中 Amazon S3 AWS 受管金鑰的。

如果您不想使用 AWS 受管金鑰，則必須自行建立和設定客戶受管金鑰。本節說明如何使用 IAM 主控台執行此操作。

如需客戶受管金鑰的相關資訊，請參閱《AWS KMS 開發人員指南》中的[AWS Key Management Service 概念](#)和[建立金鑰](#)。

若要設定客戶受管金鑰供 CodeBuild 使用，請遵循《AWS KMS 開發人員指南》中修改金鑰政策》的「如何[修改金鑰政策](#)」一節中的指示。然後，將下列陳述式 (在 **### BEGIN ADDING STATEMENTS HERE ###** 和 **### END ADDING STATEMENTS HERE ###** 之間) 新增至金鑰政策。省略符號 (...) 是為了簡潔起見，也有助於您找到要新增陳述式的位置。請不要移除任何陳述式，也不要將金鑰政策中輸入這些省略符號。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.us-east-1.amazonaws.com",
          "kms:CallerAccount": "111122223333"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

- **region-ID** 代表與 CodeBuild 相關聯的 Amazon S3 儲存貯體所在 AWS 區域的 ID（例如 us-east-1）。
- **account-ID** 代表擁有客戶受管金鑰之 AWS 帳戶的 ID。
- **CodeBuild-service-role** 代表您先前在本主題中建立或識別的 CodeBuild 服務角色名稱。

Note

若要透過 IAM 主控台建立或設定客戶受管金鑰，您必須先 AWS 管理主控台 使用下列其中一項登入：

- 您的 AWS 根帳戶。此為不建議的選項。如需詳細資訊，請參閱 [《使用者指南》中的帳戶根使用者](#)。
- 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱《使用者指南》中的[建立您的第一個 AWS 帳戶根使用者和群組](#)。
- 您 AWS 帳戶中具有建立或修改客戶受管金鑰許可的使用者。如需詳細資訊，請參閱《AWS KMS 開發人員指南》中的[使用 AWS KMS 主控台所需的許可](#)。

使用 與 CodeBuild 互動 AWS CLI

如果您 AWS CodeBuild 第一次遵循 中的步驟[開始使用主控台](#)來存取，則很可能不需要本主題中的資訊。不過，當您繼續使用 CodeBuild 時，您可能想要執行一些動作，例如允許使用者使用 與 CodeBuild 互動 AWS CLI，而不是（或除了）CodeBuild 主控台、CodePipeline 主控台或 AWS SDKs。

若要安裝和設定 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的[設定 AWS Command Line Interface](#)。

安裝 之後 AWS CLI，請完成下列任務：

1. 執行下列命令，以確認您的 安裝 AWS CLI 是否支援 CodeBuild：

```
aws codebuild list-builds
```

如果成功，輸出中會顯示類似如下的資訊：

```
{
```

```
"ids": []  
}
```

空白方括號表示您尚未執行任何組建。

2. 如果輸出錯誤，您必須解除安裝目前的 AWS CLI 版本，然後安裝最新版本。如需詳細資訊，請參閱 [AWS Command Line Interface 《使用者指南》](#) 中的 [解除安裝 AWS CLI](#) 和 [安裝 AWS Command Line Interface](#)。

的命令列參考 AWS CodeBuild

AWS CLI 提供用於自動化的命令 AWS CodeBuild。使用本主題中的資訊做為 [AWS Command Line Interface 使用者指南](#) 和 [AWS CLI 參考 AWS CodeBuild](#) 的補充。

不是您想找的內容嗎？如果您想要使用 AWS SDKs 呼叫 CodeBuild，請參閱 [AWS SDKs 和工具參考](#)。

若要使用本主題中的資訊，您應該已安裝 [AWS CLI](#)，並將其設定為與 CodeBuild 搭配使用，如中所述 [使用與 CodeBuild 互動 AWS CLI](#)。

若要使用 AWS CLI 指定 CodeBuild 的端點，請參閱 [指定 AWS CodeBuild 端點 \(AWS CLI\)](#)。

執行此命令以取得 CodeBuild 命令的清單。

```
aws codebuild help
```

執行此命令以取得 CodeBuild 命令的相關資訊，其中 *command-name* 是命令的名稱。

```
aws codebuild command-name help
```

CodeBuild 命令包括：

- `batch-delete-builds`：刪除 CodeBuild 中的一個或多個組建。如需詳細資訊，請參閱 [刪除建置 \(AWS CLI\)](#)。
- `batch-get-builds`：取得 CodeBuild 中多個組建的相關資訊。如需詳細資訊，請參閱 [檢視建置的詳細資訊 \(AWS CLI\)](#)。
- `batch-get-projects`：取得一或多個指定組建專案的相關資訊。如需詳細資訊，請參閱 [檢視建置專案的詳細資訊 \(AWS CLI\)](#)。
- `create-project`：建立組建專案。如需詳細資訊，請參閱 [建立建置專案 \(AWS CLI\)](#)。

- `delete-project`：刪除組建專案。如需詳細資訊，請參閱[刪除建置專案 \(AWS CLI\)](#)。
- `list-builds`：列出 CodeBuild 中建置的 Amazon Resource Name (ARNs)。如需詳細資訊，請參閱[檢視建置 ID 清單 \(AWS CLI\)](#)。
- `list-builds-for-project`：取得與指定組建專案相關聯的組建 ID 的清單。如需詳細資訊，請參閱[檢視建置專案的建置 ID 清單 \(AWS CLI\)](#)。
- `list-curated-environment-images`：取得 CodeBuild 管理的 Docker 映像清單，可用於您的建置。如需詳細資訊，請參閱[CodeBuild 提供的 Docker 映像](#)。
- `list-projects`：取得組建專案名稱的清單。如需詳細資訊，請參閱[檢視建置專案名稱清單 \(AWS CLI\)](#)。
- `start-build`：開始執行組建。如需詳細資訊，請參閱[執行建置 \(AWS CLI\)](#)。
- `stop-build`：嘗試停止指定的組建，使其無法執行。如需詳細資訊，請參閱[停止建置 \(AWS CLI\)](#)。
- `update-project`：變更指定組建專案的相關資訊。如需詳細資訊，請參閱[變更建置專案的設定 \(AWS CLI\)](#)。

的AWS SDKs和工具參考 AWS CodeBuild

若要使用其中一個 AWS SDKs或工具來自動化 AWS CodeBuild，請參閱下列資源。

如果您想要使用 AWS CLI 執行 CodeBuild，請參閱 [命令列參考](#)。

的支援 AWS SDKs和工具 AWS CodeBuild

下列 AWS SDKs和工具支援 CodeBuild：

- [適用於 C++ 的AWS 開發套件](#)。如需詳細資訊，請參閱適用於 AWS C++ 的 SDK API 參考的 [Aws::CodeBuild](#) 命名空間一節。
- [適用於 Go 的AWS 開發套件](#)。如需詳細資訊，請參閱適用於 AWS Go 的 SDK API 參考的[程式碼建置](#)一節。
- [適用於 Java 的AWS 開發套件](#)。如需詳細資訊，請參閱適用於 [AWS Java 的 SDK API 參考](#)的 `com.amazonaws.services.codebuild`和 `com.amazonaws.services.codebuild.model`章節。
- [瀏覽器中適用於 JavaScript 的AWS 開發套件](#)和 [Node.js 中適用於 JavaScript 的AWS 開發套件](#)。如需詳細資訊，請參閱 [類別：AWS。適用於 JavaScript 的 SDK API 參考的 CodeBuild](#) 區段。AWS JavaScript

- [適用於 .NET 的 AWS 開發套件](#)。如需詳細資訊，請參閱適用於 [.NET API 參考的 SDK 的 Amazon.CodeBuild](#) 和 [Amazon.CodeBuild.Model](#) 命名空間區段。AWS
- [適用於 PHP 的 AWS 開發套件](#)。如需詳細資訊，請參閱適用於 AWS PHP 的 SDK API 參考中的[命名空間 Aws\CodeBuild](#) 一節。
- [適用於 Python 的 AWS 開發套件 \(Boto3\)](#)。如需詳細資訊，請參閱 Boto 3 文件的 [CodeBuild](#) 一節。
- [適用於 Ruby 的 AWS 開發套件](#)。如需詳細資訊，請參閱適用於 AWS Ruby 的 SDK API 參考的 [Module : Aws : : CodeBuild](#) 一節。
- [適用於 PowerShell 的 AWS 工具](#)。如需詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference 的 [AWS CodeBuild](#) 一節。

搭配 AWS SDK 使用此服務

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
適用於 C++ 的 AWS SDK	適用於 C++ 的 AWS SDK 程式碼範例
AWS CLI	AWS CLI 程式碼範例
適用於 Go 的 AWS SDK	適用於 Go 的 AWS SDK 程式碼範例
適用於 Java 的 AWS SDK	適用於 Java 的 AWS SDK 程式碼範例
適用於 JavaScript 的 AWS SDK	適用於 JavaScript 的 AWS SDK 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
適用於 .NET 的 AWS SDK	適用於 .NET 的 AWS SDK 程式碼範例
適用於 PHP 的 AWS SDK	適用於 PHP 的 AWS SDK 程式碼範例
AWS Tools for PowerShell	AWS Tools for PowerShell 程式碼範例
AWS 適用於 Python v4 的 SDK	AWS 適用於 Python v4 的 SDK 程式碼範例

SDK 文件	代碼範例
適用於 Python (Boto3) 的 AWS SDK	適用於 Python (Boto3) 的 AWS SDK 程式碼範例
適用於 Ruby 的 AWS SDK	適用於 Ruby 的 AWS SDK 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

如需此服務的特定範例，請參閱 [CodeBuild AWS SDKs 的程式碼範例](#)。

可用性範例

找不到所需的內容嗎？請使用本頁面底部的提供意見回饋連結申請程式碼範例。

指定 AWS CodeBuild 端點

您可以使用 AWS Command Line Interface (AWS CLI) 或其中一個 AWS SDKs 來指定使用的端點 AWS CodeBuild。CodeBuild 可用的每個區域都有端點。除了區域端點，四個區域也會有聯邦資訊處理標準 (FIPS) 端點。如需 FIPS 端點的詳細資訊，請參閱 [FIPS 140-2 概觀](#)。

指定端點是選用的。如果您未明確告知 CodeBuild 要使用的端點，服務會使用與您 AWS 帳戶使用的區域相關聯的端點。CodeBuild 永遠不會預設為 FIPS 端點。如果您想要使用 FIPS 端點，您必須使用下列其中一種方法將 CodeBuild 與其建立關聯。

Note

您可以使用別名或區域名稱，使用 AWS SDK 指定端點。如果您使用 AWS CLI，則必須使用完整的端點名稱。

如需可與 CodeBuild 搭配使用的端點，請參閱 [CodeBuild 區域和端點](#)。

主題

- [指定 AWS CodeBuild 端點 \(AWS CLI\)](#)
- [指定 AWS CodeBuild 端點 \(AWS SDK\)](#)

指定 AWS CodeBuild 端點 (AWS CLI)

您可以使用 AWS CLI 在任何 CodeBuild 命令中使用 `--endpoint-url` 引數來指定透過其 AWS CodeBuild 存取的端點。例如，執行此命令來取得在美國東部（維吉尼亞北部）區域中使用聯邦資訊處理標準 (FIPS) 端點的專案建置名稱清單：

```
aws codebuild list-projects --endpoint-url https://codebuild-fips.us-east-1.amazonaws.com
```

在端點的 `https://` 開頭包含。

引 `--endpoint-url` AWS CLI 數適用於所有 AWS 服務。如需此引數和其他 AWS CLI 引數的詳細資訊，請參閱 [AWS CLI 命令參考](#)。

指定 AWS CodeBuild 端點 (AWS SDK)

您可以使用 AWS SDK 來指定透過 AWS CodeBuild 存取的端點。雖然此範例使用 [AWS 適用於 Java 的開發套件](#)，但您可以使用其他 AWS SDKs 指定端點。

建構 `AWSCodeBuild` 用戶端時，請使用 `withEndpointConfiguration` 方法。以下是要使用的格式：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",
    "region")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

如需有關 `AWSCodeBuildClientBuilder` 的詳細資訊，請參閱 [Class AWSCodeBuildClientBuilder](#)。

`withCredentials` 中使用之登入資料的類型必須是 `AWSCredentialsProvider`。如需詳細資訊，請參閱 [使用 AWS 登入](#) 資料。

請勿在端點 `https://` 開頭包含。

如果您想要指定非 FIPS 端點，您可以使用區域而非實際端點。例如，若要在美國東部（維吉尼亞北部）區域指定端點，您可以使用 `us-east-1`，而不是完整的端點名稱 `codebuild.us-east-1.amazonaws.com`。

如果您想要指定 FIPS 端點，您可以使用別名來簡化程式碼。只有 FIPS 端點具有別名。其他端點必須使用其區域或完整名稱來指定。

下表列出四個可用之 FIPS 端點的別名：

區域名稱	區域	Endpoint	Alias (別名)
美國東部 (維吉尼亞北部)	us-east-1	codebuild-fips.us-east-1.amazonaws.com	us-east-1-fips
美國東部 (俄亥俄)	us-east-2	codebuild-fips.us-east-2.amazonaws.com	us-east-2-fips
美國西部 (加利佛尼亞北部)	us-west-1	codebuild-fips.us-west-1.amazonaws.com	us-west-1-fips
美國西部 (奧勒岡)	us-west-2	codebuild-fips.us-west-2.amazonaws.com	us-west-2-fips

若要使用別名指定在美國西部（奧勒岡）區域中使用 FIPS 端點：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-
fips", "us-west-2")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

若要指定在美國東部（維吉尼亞北部）區域中使用非 FIPS 端點：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
```

```
withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1",  
"us-east-1")).  
withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).  
build();
```

若要指定在亞太區域（孟買）區域中使用非 FIPS 端點：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().  
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1",  
"ap-south-1")).  
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).  
    build();
```

AWS CodeBuild 搭配使用 AWS CodePipeline 來測試程式碼和執行組建

您可以使用來測試程式碼並使用執行組建 AWS CodePipeline，以自動化發行程序 AWS CodeBuild。

下表列出可用於執行它們的任務和方法。使用 AWS 開發套件完成這些任務不在本主題的範圍內。

任務	可用方法	本主題中說明的方法
使用 CodePipeline 建立持續交付 (CD) 管道，以 CodeBuild 自動化建置	<ul style="list-style-type: none">CodePipeline 主控台AWS CLIAWS SDKs	<ul style="list-style-type: none">使用 CodePipeline 主控台使用 AWS CLI您可以配合本主題中的資訊使用 AWS 軟體開發套件。如需詳細資訊，請參閱《適用於 Amazon Web Services 的工具 SDKs》一節中的程式設計語言 create-pipeline 的動作文件，或參閱《AWS CodePipeline API 參考 CreatePipeline》中的。
使用 CodeBuild 將測試和建置自動化新增至	<ul style="list-style-type: none">CodePipeline 主控台AWS CLIAWS SDKs	<ul style="list-style-type: none">使用 CodePipeline 主控台新增建置自動化使用 CodePipeline 主控台新增測試自動化對於 AWS CLI，您可以調整本主題中的資訊，以建立包含 CodeBuild 組建動作或測試動作的管道。如需詳細資

任務	可用方法	本主題中說明的方法
CodePipeline 中的現有管道		<p>訊，請參閱AWS CodePipeline 《使用者指南》中的編輯管道 (AWS CLI) 和 CodePipeline 管道結構參考。</p> <ul style="list-style-type: none"> 您可以配合本主題中的資訊使用 AWS 軟體開發套件。如需詳細資訊，請參閱透過 Tools for Amazon Web Services 的 SDKs 一節的程式設計語言update-pipeline 動作文件，或參閱 AWS CodePipeline API 參考UpdatePipeline 中的。

主題

- [先決條件](#)
- [建立使用 CodeBuild \(CodePipeline 主控台 \) 的管道](#)
- [建立使用 CodeBuild 的管道 \(AWS CLI\)](#)
- [將 CodeBuild 組建動作新增至管道 \(CodePipeline 主控台 \)](#)
- [將 CodeBuild 測試動作新增至管道 \(CodePipeline 主控台 \)](#)

先決條件

1. 回答[規劃組建](#)中的問題。
2. 如果您使用使用者存取 CodePipeline，而不是 AWS 根帳戶或管理員使用者，請將名為 AWSCodePipelineFullAccess 的受管政策連接至使用者（或使用者所屬的 IAM 群組）。不建議使用 AWS 根帳戶。此政策授予使用者在 CodePipeline 中建立管道的許可。如需詳細資訊，請參閱《使用者指南》中的[連接受管政策](#)。

Note

將政策連接至使用者（或使用者所屬的 IAM 群組）的 IAM 實體必須在 IAM 中具有連接政策的許可。如需詳細資訊，請參閱《使用者指南》中的[委派管理 IAM 使用者、群組和登入資料的許可](#)。

3. 如果您的 AWS 帳戶中還沒有可用的 CodePipeline 服務角色，請建立該角色。CodePipeline 會使用此服務角色來代表您與其他 AWS 服務互動 AWS CodeBuild，包括。例如，若要使用 AWS CLI 建立 CodePipeline 服務角色，請執行 IAM create-role 命令：

針對 Linux、macOS 或 Unix：

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role --assume-role-policy-document '{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": "codepipeline.amazonaws.com"}, "Action": "sts:AssumeRole"}]}'
```

針對 Windows：

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role --assume-role-policy-document "{\"Version\":\"2012-10-17\", \"Statement\": [{\"Effect\": \"Allow\", \"Principal\": {\"Service\": \"codepipeline.amazonaws.com\"}, \"Action\": \"sts:AssumeRole\"}]}"
```

 Note

建立此 CodePipeline 服務角色的 IAM 實體必須在 IAM 中具有建立服務角色的許可。

4. 建立 CodePipeline 服務角色或識別現有服務角色之後，您必須將預設 CodePipeline 服務角色政策新增至服務角色，如AWS CodePipeline 《使用者指南》中的[檢閱預設 CodePipeline 服務角色政策](#)中所述，如果該政策尚未成為角色政策的一部分。

 Note

新增此 CodePipeline 服務角色政策的 IAM 實體必須具有 IAM 中的許可，才能將服務角色政策新增至服務角色。

5. 建立原始碼並將其上傳至 CodeBuild 和 CodePipeline 支援的儲存庫類型，例如 CodeCommit、Amazon S3、Bitbucket 或 GitHub。來源碼應該包含 Builds spec 檔案，但您可以在本主題稍後定義組建專案時宣告一個。如需更多資訊，請參閱[Builds spec 參考](#)。

 Important

若您計劃使用管道來部署建置的原始程式碼，則建置輸出成品必須與您使用的部署系統相容。

- 如需 OpsWorks，請參閱 OpsWorks 《使用者指南》中的[應用程式來源](#)和[搭配使用 CodePipeline OpsWorks](#)。

建立使用 CodeBuild (CodePipeline 主控台) 的管道

使用下列程序來建立使用 CodeBuild 建置和部署原始程式碼的管道。

若要建立僅測試來源碼的管道：

- 使用下列程序建立管道，然後從管道刪除「組建」及「Beta」階段。然後使用本主題中的[將 CodeBuild 測試動作新增至管道 \(CodePipeline 主控台\)](#) 程序，將使用 CodeBuild 的測試動作新增至管道。
- 使用本主題中的其他程序之一來建立管道，然後使用本主題中的[將 CodeBuild 測試動作新增至管道 \(CodePipeline 主控台\)](#) 程序，將使用 CodeBuild 的測試動作新增至管道。

若要使用 CodePipeline 中的建立管道精靈來建立使用 CodeBuild 的管道

1. 使用 AWS 管理主控台 登入：

- 您的 AWS 根帳戶。此為不建議的選項。如需詳細資訊，請參閱[《使用者指南》中的帳戶根使用者](#)。
- 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱《使用者指南》中的[建立您的第一個 AWS 帳戶 根使用者和群組](#)。
- 您 AWS 帳戶中具有使用下列最低動作集之許可的使用者：

```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
```

```
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. 在 <https://console.aws.amazon.com/codesuite/codepipeline/home> 開啟 AWS CodePipeline 主控台。
3. 在 AWS 區域選取器中，選擇建置專案 AWS 資源所在的 AWS 區域。這必須是支援 CodeBuild AWS 的區域。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 [AWS CodeBuild](#)。
4. 建立管道。如果顯示 CodePipeline 資訊頁面，請選擇建立管道。若顯示 Pipelines (管道) 頁面，選擇 Create pipeline (建立管道)。
5. 在 Step 1: Choose pipeline settings (步驟 1：選擇管道設定) 頁面，針對 Pipeline name (管道名稱)，輸入管道的名稱 (例如 **CodeBuildDemoPipeline**)。如果您選擇不同名稱，請在此程序中都使用此名稱。
6. 針對 Role name (角色名稱)，執行下列其中一項操作：

選擇 New service role (新服務角色)，並在 Role Name (角色名稱) 中輸入新服務角色的名稱。

選擇現有的服務角色，然後選擇您建立或識別為本主題先決條件一部分的 CodePipeline 服務角色。

7. 針對 Artifact store (成品存放區) 中，執行下列其中一項操作：
 - 針對您為管道選取的 AWS 區域中的管道，選擇預設位置以使用預設成品存放區，例如指定為預設的 S3 成品儲存貯體。
 - 如果您已在管道的相同 AWS 區域中建立現有的成品存放區，例如 S3 成品儲存貯體，請選擇自訂位置。

Note

這不是管道原始碼的來源儲存貯體。這是管道的成品存放區。在與管道相同的 AWS 區域中，每個管道都需要單獨的成品存放區，例如 S3 儲存貯體。

8. 選擇下一步。

9. 在 Step 2: Add source stage (步驟 2：新增來源階段) 頁面，針對 Source provider (來源提供者)，執行下列其中一項操作：
 - 如果您的原始程式碼存放在 S3 儲存貯體中，請選擇 Amazon S3。針對 Bucket (儲存貯體)，選取含有原始程式碼的 S3 儲存貯體。針對 S3 物件金鑰，輸入包含來源碼的檔案名稱（例如，`file-name.zip`）。選擇下一步。
 - 如果您的原始碼存放在 AWS CodeCommit 儲存庫中，請選擇 CodeCommit。針對 Repository name (儲存庫名稱)，請選擇包含來源碼的儲存庫名稱。針對 Branch name (分支名稱)，請選擇分支名稱，包含您希望建立的原始程式碼版本。選擇下一步。
 - 若您的來源碼存放於 GitHub 儲存庫，請選擇 GitHub。選擇 Connect to GitHub (連線到 GitHub)，然後依照說明向 GitHub 進行身分驗證。針對 Repository (儲存庫)，請選擇包含來源碼的儲存庫名稱。針對 Branch (分支)，請選擇分支名稱，包含您希望建立的原始程式碼版本。

選擇下一步。

10. 在步驟 3：新增建置階段頁面上，針對建置提供者，選擇 CodeBuild。
11. 如果您已有想要使用的建置專案，請在專案名稱中選擇建置專案的名稱，然後跳至此程序的下一個步驟。

如果您需要建立新的 CodeBuild 組建專案，請遵循 中的指示[建立組建專案 \(主控台\)](#)並返回此程序。

如果您選擇現有的建置專案，它必須已定義建置輸出成品設定（即使 CodePipeline 覆寫它們）。如需詳細資訊，請參閱[變更建置專案的設定 \(主控台\)](#)。

Important

如果您為 CodeBuild 專案啟用 Webhook，且該專案用作 CodePipeline 中的建置步驟，則會為每個遞交建立兩個相同的建置。一個建置會透過 Webhook 觸發，另一個則透過 CodePipeline 觸發。因為帳單是以每次組建為基礎，因此您將必須同時為兩次組建支付費用。因此，如果您使用的是 CodePipeline，建議您在 CodeBuild 中停用 Webhook。在 AWS CodeBuild 主控台中，清除 Webhook 方塊。如需詳細資訊，請參閱[變更建置專案的設定 \(主控台\)](#)。

12. 在 Step 4: Add deploy stage (步驟 4：新增部署階段) 頁面，執行下列其中一項操作：
 - 若您不想要部署組建輸出成品，請在系統出現提示時選擇 Skip (略過)，並確認此選擇。

- 若您想要部署組建輸出成品，則針對 Deploy provider (部署提供者)，請選擇部署提供者，然後在系統出現提示時指定設定。

選擇下一步。

13. 在 Review (檢閱) 頁面上，檢閱您的選擇，然後選擇 Create pipeline (建立管道)。
14. 在管道執行成功之後，您便會取得組建輸出成品。使用 CodePipeline 主控台中顯示的管道，在建置動作中，選擇工具提示。記下 Output artifact (輸出成品) 的值 (例如，MyAppBuild)。

Note

您也可以選擇 CodeBuild 主控台中建置詳細資訊頁面上的建置成品連結，以取得建置輸出成品。若要抵達此頁面，請跳過此程序中的其餘步驟，並參閱[檢視建置的詳細資訊 \(主控台\)](#)。

15. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
16. 在儲存貯體清單中，開啟管線使用的儲存貯體。儲存貯體名稱必須遵循以下格式：`codepipeline-region-ID-random-number`。您可以使用 AWS CLI 執行 CodePipeline `get-pipeline` 命令來取得儲存貯體的名稱，其中 *my-pipeline-name* 是管道的顯示名稱：

```
aws codepipeline get-pipeline --name my-pipeline-name
```

在輸出中，pipeline 物件包含 artifactStore 物件，其中包含帶有儲存貯體名稱的 location 值。

17. 開啟與您管道名稱相符的資料 (根據管道名稱的長度，資料夾的名稱可能會遭到截斷)，然後開啟與您稍早記下的 Output artifact (輸出成品) 值相符的資料夾。
18. 解壓縮 檔案的內容。若該資料夾中有多個檔案，請解壓縮 Last Modified (最後修改) 時間戳記最新的檔案內容。(您可能需要為檔案加上 .zip 副檔名，才能搭配您系統的 ZIP 公用程式使用。) 組建輸出成品位於檔案的解壓縮內容中。
19. 如果您指示 CodePipeline 部署建置輸出成品，請使用部署提供者的指示來取得部署目標上的建置輸出成品。

建立使用 CodeBuild 的管道 (AWS CLI)

使用下列程序建立使用 CodeBuild 建置原始程式碼的管道。

若要使用 AWS CLI 建立管道來部署建置的原始程式碼，或只測試原始程式碼，您可以調整 AWS CodePipeline 《使用者指南》中的[編輯管道 \(AWS CLI\)](#) 和 [CodePipeline 管道結構參考](#)中的說明。

1. 在 CodeBuild 中建立或識別組建專案。如需詳細資訊，請參閱[建立組建專案](#)。

 Important

建置專案必須定義建置輸出成品設定（即使 CodePipeline 覆寫它們）。如需詳細資訊，請參閱[建立建置專案 \(AWS CLI\)](#)中的 artifacts 描述。

2. 請確定您已 AWS CLI 使用對應至本主題所述其中一個 IAM 實體的 AWS 存取金鑰和 AWS 私密存取金鑰來設定。如需詳細資訊，請參閱 AWS Command Line Interface 《使用者指南》中的[設定 AWS Command Line Interface](#)。
3. 建立代表管線結構的 JSON 格式檔案。將檔案命名為 `create-pipeline.json` 或相似名稱。例如，此 JSON 格式結構會建立管道，其中包含參考 S3 輸入儲存貯體的來源動作，以及使用 CodeBuild 的建置動作：

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::<account-id>:role/<AWS-CodePipeline-service-role-name>",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "configuration": {
              "S3Bucket": "<bucket-name>",
```

```

        "S3objectKey": "<source-code-file-name.zip>"
    },
    "runOrder": 1
  }
]
},
{
  "name": "Build",
  "actions": [
    {
      "inputArtifacts": [
        {
          "name": "MyApp"
        }
      ],
      "name": "Build",
      "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "version": "1",
        "provider": "CodeBuild"
      },
      "outputArtifacts": [
        {
          "name": "default"
        }
      ],
      "configuration": {
        "ProjectName": "<build-project-name>"
      },
      "runOrder": 1
    }
  ]
}
],
"artifactStore": {
  "type": "S3",
  "location": "<CodePipeline-internal-bucket-name>"
},
"name": "<my-pipeline-name>",
"version": 1
}
}

```

在此 JSON 格式資料中：

- 的值 `roleArn` 必須符合您建立或識別為先決條件一部分之 CodePipeline 服務角色的 ARN。
- `configuration` 中 `S3Bucket` 和 `S3ObjectKey` 的值會假設原始程式碼存放於 S3 儲存貯體中。如需其他來源碼儲存庫類型的設定，請參閱 AWS CodePipeline 《使用者指南》中的 [CodePipeline 管道結構參考](#)。
- 的值 `ProjectName` 是您在此程序中稍早建立的 CodeBuild 建置專案的名稱。
- `location` 的值為此管線使用的 S3 儲存貯體名稱。如需詳細資訊，請參閱 AWS CodePipeline 《使用者指南》中的 [建立 S3 儲存貯體的政策以用作 CodePipeline 的成品存放區](#)。
- `name` 的值為此管線的名稱。所有管線名稱對您的帳戶來說都必須是獨一無二的。

雖然此資料僅描述來源動作和建置動作，但您可以為與測試、部署建置輸出成品、叫用 AWS Lambda 函數等相關的活動新增動作。如需詳細資訊，請參閱 AWS CodePipeline 《使用者指南》中的 [AWS CodePipeline 管道結構參考](#)。

4. 切換到包含 JSON 檔案的資料夾，然後執行 CodePipeline [create-pipeline](#) 命令，指定檔案名稱：

```
aws codepipeline create-pipeline --cli-input-json file://create-pipeline.json
```

Note

您必須在支援 CodeBuild 的 AWS 區域中建立管道。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 [AWS CodeBuild](#)。

JSON 格式的資料會出現在輸出中，CodePipeline 會建立管道。

5. 若要取得管道狀態的相關資訊，請執行 CodePipeline [get-pipeline-state](#) 命令，指定管道的名稱：

```
aws codepipeline get-pipeline-state --name <my-pipeline-name>
```

在輸出中，尋找確認建置成功的資訊。省略符號 (...) 用於顯示為求簡化而省略的資料。

```
{
  ...
  "stageStates": [
    ...
```

```
{
  "actionStates": [
    {
      "actionName": "CodeBuild",
      "latestExecution": {
        "status": "SUCCEEDED",
        ...
      },
      ...
    }
  ]
}
```

若您太早執行此命令，您可能會無法看到任何與建置動作有關的資訊。您可能需要執行此命令數次，直到管線完成執行建置動作。

6. 在成功建置之後，請遵循這些說明取得建置輸出成品。開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。

Note

您也可以選擇 CodeBuild 主控台中相關組建詳細資訊頁面上的組建成品連結，以取得組建輸出成品。若要抵達此頁面，請跳過此程序中的其餘步驟，並參閱[檢視建置的詳細資訊 \(主控台\)](#)。

7. 在儲存貯體清單中，開啟管線使用的儲存貯體。儲存貯體名稱必須遵循以下格式：`codepipeline-<region-ID>-<random-number>`。您可以從 `create-pipeline.json` 檔案取得儲存貯體名稱，也可以執行 CodePipeline `get-pipeline` 命令來取得儲存貯體的名稱。

```
aws codepipeline get-pipeline --name <pipeline-name>
```

在輸出中，`pipeline` 物件包含 `artifactStore` 物件，其中包含帶有儲存貯體名稱的 `location` 值。

8. 開啟與您管線名稱相符的資料夾 (例如，*<pipeline-name>*)。
9. 在該資料夾中，開啟名為 `default` 的資料夾。

10. 解壓縮 檔案的內容。若該資料夾中有多個檔案，請解壓縮 Last Modified (最後修改) 時間戳記最新的檔案內容。(您可能需要為檔案加上 .zip 副檔名，才能與您系統的 ZIP 公用程式搭配使用。) 組建輸出成品位於檔案的解壓縮內容中。

將 CodeBuild 組建動作新增至管道 (CodePipeline 主控台)

1. 使用 AWS 管理主控台 登入：

- 您的 AWS 根帳戶。此為不建議的選項。如需詳細資訊，請參閱 [《使用者指南》中的帳戶根使用者](#)。
- 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱 [《使用者指南》中的建立您的第一個 AWS 帳戶 根使用者和群組](#)。
- 您 AWS 帳戶中具有執行下列最低動作集之許可的使用者：

```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. 在 <https://console.aws.amazon.com/codesuite/codepipeline/home> 開啟 CodePipeline 主控台。

3. 在 AWS 區域選擇器中，選擇管道所在的 AWS 區域。這必須是支援 CodeBuild 的區域。如需詳細資訊，請參閱《》中的 [CodeBuild](#) Amazon Web Services 一般參考。
4. 在 Pipelines (管道) 頁面上，選擇管道名稱。
5. 在管道詳細資訊頁面上，於 Source (來源) 動作中，選擇工具提示。記下 Output artifact (輸出成品) 的值 (例如，MyApp)。

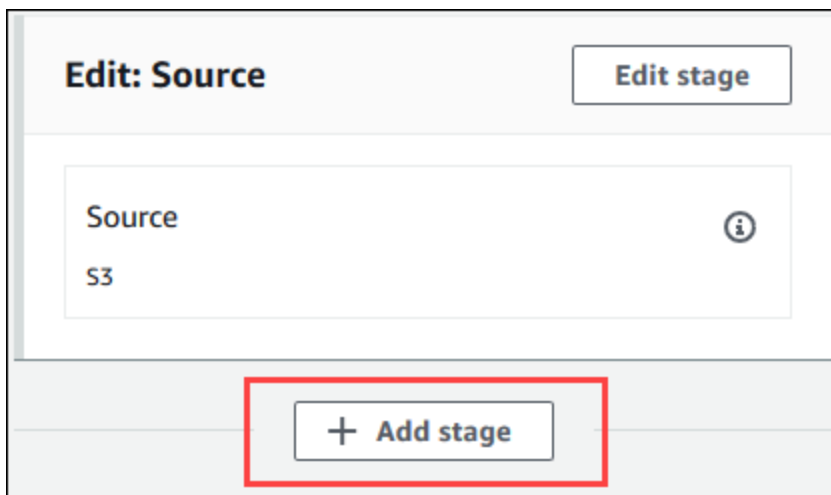
Note

此程序說明如何在 Source (來源) 和 Beta 階段間的組建階段內新增組建動作。若您希望將組建動作新增至其他地方，請選擇動作上的工具提示 (就在您要新增組建動作之位置前)，然後記下 Output artifact (輸出成品) 的值。

6. 選擇編輯。
7. 在 Source (來源) 和 Beta 階段間，選擇 Add stage (新增階段)。

Note

此程序說明如何在 Source (來源) 和 Beta 階段之間將建置階段新增至您的管道。若要將組建動作新增至現有的階段，請選擇階段中的 Edit stage (編輯階段) 圖示，然後跳到此程序的步驟 8。若要在別處新增組建階段，請在所需位置選擇 Add stage (新增階段)。



8. 針對 Stage name (階段名稱)，輸入組建階段的名稱 (例如，**Build**)。如果您選擇其他名稱，請在此程序中皆使用它。
9. 在選取的階段內部，選擇 Add action (新增動作)。

 Note

此程序說明您如何在組建階段內新增組建動作。若要在別處新增組建動作，請在所需位置選擇 Add action (新增動作)。您可能必須先選擇您欲新增組建動作之現有階段內的 Edit stage (編輯階段)。

10. 在 Edit action (編輯動作) 中，針對 Action name (動作名稱)，輸入動作的名稱 (例如 **CodeBuild**)。如果您選擇其他名稱，請在此程序中皆使用它。
11. 針對動作提供者，選擇 CodeBuild。
12. 如果您已有想要使用的建置專案，請在專案名稱中選擇建置專案的名稱，然後跳至此程序的下一個步驟。

如果您需要建立新的 CodeBuild 組建專案，請遵循 中的指示[建立組建專案 \(主控台\)](#)並返回此程序。

如果您選擇現有的建置專案，則必須定義建置輸出成品設定 (即使 CodePipeline 覆寫它們)。如需詳細資訊，請參閱 [建立組建專案 \(主控台\)](#) 或 中的成品描述[變更建置專案的設定 \(主控台\)](#)。

 Important

如果您為 CodeBuild 專案啟用 Webhook，且該專案用作 CodePipeline 中的建置步驟，則會為每個遞交建立兩個相同的建置。一個建置會透過 Webhook 觸發，另一個則透過 CodePipeline 觸發。因為帳單是以每次組建為基礎，因此您將必須同時為兩次組建支付費用。因此，如果您使用的是 CodePipeline，建議您在 CodeBuild 中停用 Webhook。在 CodeBuild 主控台中，清除 Webhook 方塊。如需詳細資訊，請參閱[變更建置專案的設定 \(主控台\)](#)

13. 針對 Input artifacts (輸入成品)，選擇您稍早在本程序中記下的輸出成品。
14. 針對 Output artifacts (輸出成品)，輸入輸出成品名稱 (例如，**MyAppBuild**)。
15. 選擇新增動作。
16. 選擇 Save (儲存)，然後選擇 Save (儲存) 以儲存對管道的變更。
17. 選擇 Release change (版本變更)。
18. 在管道執行成功之後，您便會取得組建輸出成品。使用 CodePipeline 主控台中顯示的管道，在建置動作中，選擇工具提示。記下 Output artifact (輸出成品) 的值 (例如，MyAppBuild)。

 Note

您也可以選擇 CodeBuild 主控台建置詳細資訊頁面上的建置成品連結，以取得建置輸出成品。若要抵達此頁面，請參閱[檢視建置的詳細資訊 \(主控台\)](#)，然後跳到此程序的步驟 31。

19. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
20. 在儲存貯體清單中，開啟管線使用的儲存貯體。儲存貯體名稱必須遵循以下格式：`codepipeline-region-ID-random-number`。您可以使用 AWS CLI 執行 CodePipeline `get-pipeline` 命令，以取得儲存貯體的名稱：

```
aws codepipeline get-pipeline --name my-pipeline-name
```

在輸出中，`pipeline` 物件包含 `artifactStore` 物件，其中包含帶有儲存貯體名稱的 `location` 值。

21. 開啟與您管道名稱相符的資料 (根據管道名稱的長度，資料夾的名稱可能會遭到截斷)，然後開啟與您稍早在本程序中記下的 Output artifact (輸出成品) 值相符的資料夾。
22. 解壓縮 檔案的內容。若該資料夾中有多個檔案，請解壓縮 Last Modified (最後修改) 時間戳記最新的檔案內容。(您可能需要為檔案加上 `.zip` 副檔名，才能搭配您系統的 ZIP 公用程式使用。) 組建輸出成品位於檔案的解壓縮內容中。
23. 如果您指示 CodePipeline 部署建置輸出成品，請使用部署提供者的指示來取得部署目標上的建置輸出成品。

將 CodeBuild 測試動作新增至管道 (CodePipeline 主控台)

1. 使用 AWS 管理主控台 登入：
 - 您的 AWS 根帳戶。此為不建議的選項。如需詳細資訊，請參閱 [《使用者指南》中的帳戶根使用者](#)。
 - 您 AWS 帳戶中的管理員使用者。如需詳細資訊，請參閱 [《使用者指南》中的建立您的第一個 AWS 帳戶 根使用者和群組](#)。
 - 您 AWS 帳戶中具有執行下列最低動作集之許可的使用者：

```
codepipeline:*
iam:ListRoles
iam:PassRole
```

```
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. 在 <https://console.aws.amazon.com/codesuite/codepipeline/home> 開啟 CodePipeline 主控台。
3. 在 AWS 區域選擇器中，選擇管道所在的 AWS 區域。這必須是支援 CodeBuild AWS 的區域。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 [AWS CodeBuild](#)。
4. 在 Pipelines (管道) 頁面上，選擇管道名稱。
5. 在管道詳細資訊頁面上，於 Source (來源) 動作中，選擇工具提示。記下 Output artifact (輸出成品) 的值 (例如，MyApp)。

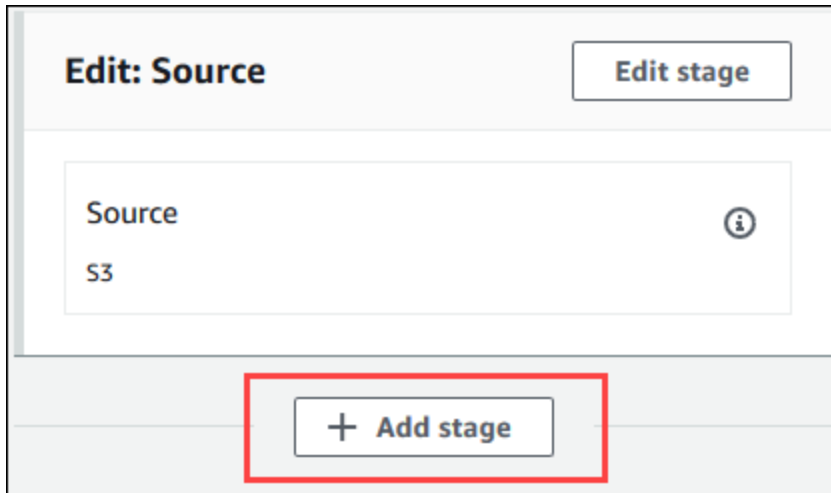
 Note

此程序說明您如何在 Source (來源) 和 Beta 階段的測試階段內新增測試動作。若您希望將測試動作新增至其他地方，請將滑鼠指標停留在前方的動作上，然後記下 Output artifact (輸出成品) 的值。

6. 選擇編輯。
7. 緊接在 Source (來源) 階段之後，選擇 Add stage (階段)。

Note

此程序說明如何於緊接在 Source (來源) 階段之後將測試階段新增至您的管道。若要將測試動作新增至現有的階段，請選擇階段中的 Edit stage (編輯階段) 圖示，然後跳到此程序的步驟 8。若要在別處新增測試階段，請在所需位置選擇 Add stage (新增階段)。



8. 針對 Stage name (階段名稱)，輸入測試階段的名稱 (例如，**Test**)。如果您選擇其他名稱，請在此程序中皆使用它。
9. 在選取的階段中，選擇 Add action (新增動作)。

Note

此程序會說明如何在測試階段中新增測試動作。若要在別處新增測試動作，請在所需位置選擇 Add action (新增動作)。您可能必須先選擇您欲新增測試動作之現有階段內的 Edit (編輯)。

10. 在 Edit action (編輯動作) 中，針對 Action name (動作名稱)，輸入動作的名稱 (例如 **Test**)。如果您選擇其他名稱，請在此程序中皆使用它。
11. 對於動作提供者，在測試下，選擇 CodeBuild。
12. 如果您已有想要使用的建置專案，請在專案名稱中選擇建置專案的名稱，然後跳至此程序的下一個步驟。

如果您需要建立新的 CodeBuild 組建專案，請遵循 [中的指示](#) [建立組建專案 \(主控台\)](#) 並返回此程序。

⚠ Important

如果您為 CodeBuild 專案啟用 Webhook，且該專案用作 CodePipeline 中的建置步驟，則會為每個遞交建立兩個相同的建置。一個建置會透過 Webhook 觸發，另一個則透過 CodePipeline 觸發。因為帳單是以每次組建為基礎，因此您將必須同時為兩次組建支付費用。因此，如果您使用的是 CodePipeline，建議您在 CodeBuild 中停用 Webhook。在 CodeBuild 主控台中，清除 Webhook 方塊。如需詳細資訊，請參閱 [變更建置專案的設定 \(主控台\)](#)

13. 針對 Input artifacts (輸入成品)，選取您稍早在本程序中記下的 Output artifact (輸出成品) 值。
14. (選用) 若您希望測試動作產生輸出成品，並且也已將您的 Builds spec 適當設定完畢，則針對 Output artifact (輸出成品)，輸入您希望指派給輸出成品的值。
15. 選擇儲存。
16. 選擇 Release change (版本變更)。
17. 在管道執行成功之後，您便會取得測試結果。在管道的測試階段中，選擇 CodeBuild 超連結，以在 CodeBuild 主控台中開啟相關的建置專案頁面。
18. 在組建專案頁面上，於 Build history (組建歷史記錄) 中選擇 Build run (組建執行) 超連結。
19. 在建置執行頁面的建置日誌中，選擇檢視整個日誌超連結，以在 Amazon CloudWatch 主控台中開啟建置日誌。
20. 捲動組建日誌，檢視測試結果。

AWS CodeBuild 搭配 Codecov 使用

Codecov 是一種用於測量程式碼測試涵蓋範圍的工具。Codecov 可識別程式碼中的哪些方法和陳述式尚未經過測試。使用結果來決定要將測試寫入哪個位置，提升程式碼的品質。Codecov 適用於 CodeBuild 支援的三個來源儲存庫：GitHub、GitHub Enterprise Server 和 Bitbucket。如果您的建置專案使用 GitHub Enterprise Server，則必須使用 Codecov Enterprise。

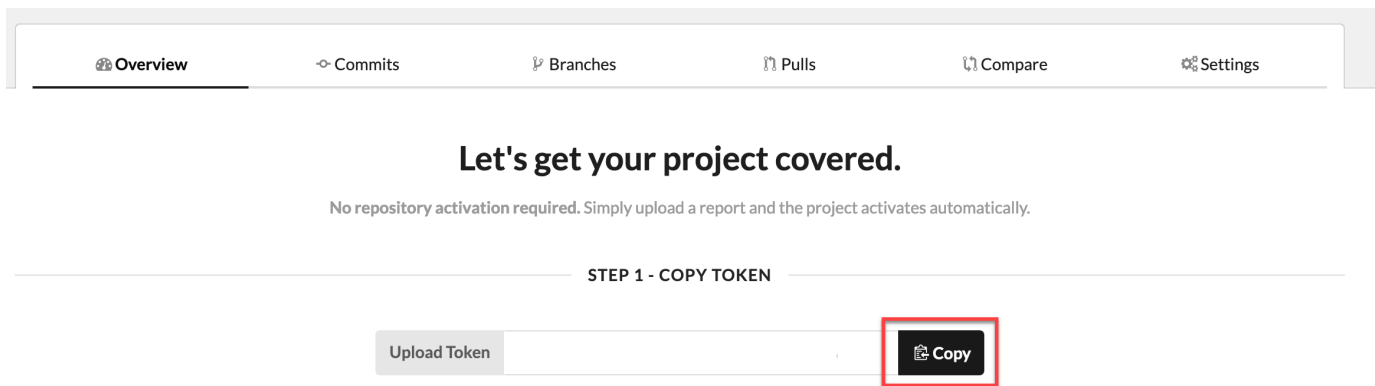
當您執行與 Codecov 整合的 CodeBuild 專案建置時，分析儲存庫中程式碼的 Codecov 報告會上傳至 Codecov。建置日誌包含報告的連結。此範例示範如何將 Python 和 Java 建置專案與 Codecov 加以整合。如需 Codecov 支援語言清單，請參閱 Codecov 網站上的 [Codecov 支援語言](#)。

將 Codecov 整合至建置專案

使用下列程序將 Codecov 整合到建置專案中。

整合 Codecov 與建置專案

1. 前往 <https://codecov.io/signup> 並註冊 GitHub 或 Bitbucket 來源儲存庫。如果您使用的是 GitHub Enterprise，則請參閱 Codecov 網站上的 [Codecov Enterprise](#)。
2. 在 Codecov 中，新增儲存庫以用於您需要的涵蓋範圍。
3. 當畫面上顯示字符資訊時，請選擇 Copy (複製)。



4. 將複製的字符以名為 CODECOV_TOKEN 環境變數的形式新增至建置專案。如需詳細資訊，請參閱 [變更建置專案的設定 \(主控台\)](#)。
5. 在儲存庫中建立名為 my_script.sh 的文字檔案。在檔案中輸入下列內容：

```
#!/bin/bash
bash <(curl -s https://codecov.io/bash) -t $CODECOV_TOKEN
```

6. 針對適合您建置專案使用的情況，選擇 Python 或 Java 標籤，然後按照下列步驟進行。

Java

1. 將下列 JaCoCo 外掛程式新增至儲存庫中的 pom.xml。

```
<build>
  <plugins>
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.2</version>
      <executions>
```

```
<execution>
  <goals>
    <goal>prepare-agent</goal>
  </goals>
</execution>
<execution>
  <id>report</id>
  <phase>test</phase>
  <goals>
    <goal>report</goal>
  </goals>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

2. 在 Buildspec 檔案中輸入下列命令。如需詳細資訊，請參閱[Buildspec 語法](#)。

```
build:
  - mvn test -f pom.xml -fn
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

Python

- 在 Buildspec 檔案中輸入下列命令。如需詳細資訊，請參閱[Buildspec 語法](#)。

```
build:
  - pip install coverage
  - coverage run -m unittest discover
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

7. 執行建置專案的建置。針對專案產生的 Codecov 報告連結會顯示在建置日誌中。使用連結檢視 Codecov 報告。如需詳細資訊，請參閱[手動執行 AWS CodeBuild 組建及使用記錄 AWS CodeBuild API 呼叫 AWS CloudTrail](#)。建置日誌中的 Codecov 資訊看起來類似下列內容：

```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh
```

/ — | | |
 | | — — | | — — — — —
 | | / — \ / — ` | / — ∨ — / — \ \ / /
 | | — | () | (| | — / (| () \ ∨ /
 \ — \ — / \ — , — | \ — | \ — \ — / \ — /

Bash-20200303-bc4d7e6

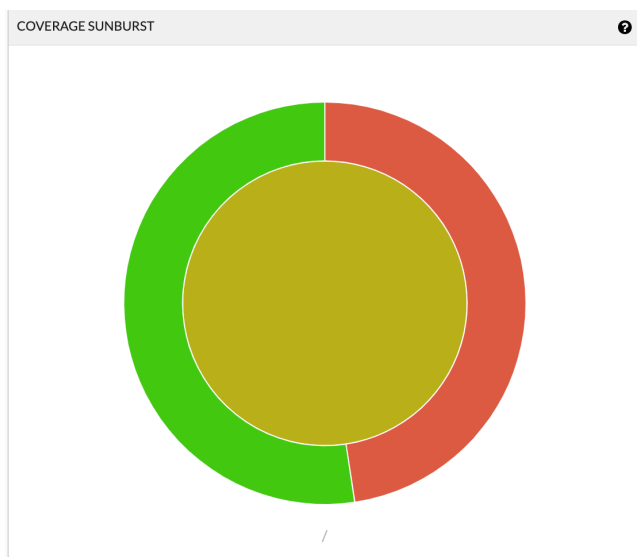
```

·[0;90m==>·[0m AWS CodeBuild detected.
... The full list of Codecov log entries has been omitted for brevity ...
·
·[0;32m->·[0m View reports at ·[0;36mhttps://codecov.io/github/user/test\_py/commit/commit-id·[0m

```

```
[Container] 2020/03/09 16:31:07 Phase complete: POST_BUILD State: SUCCEEDED
```

報告看起來類似下列內容：



						
Files					Coverage	
 code.py	10	7	0	3	<div></div> 70.00%	
 tests.py	11	11	0	0	<div></div> 100.00%	
Project Totals (2 files)	21	18	0	3	<div></div> 85.71%	

AWS CodeBuild 與 Jenkins 搭配使用

您可以使用的 Jenkins 外掛程式 AWS CodeBuild，將 CodeBuild 與您的 Jenkins 組建任務整合。您可以使用外掛程式將組建任務傳送至 CodeBuild，而不是將組建任務傳送至 Jenkins 組建節點。透過此方式，您便不需佈建、設定和管理 Jenkins 組建節點。

主題

- [設定 Jenkins](#)
- [安裝外掛程式。](#)
- [使用 外掛程式](#)

設定 Jenkins

如需使用 AWS CodeBuild 外掛程式設定 Jenkins 的資訊，以及下載外掛程式原始程式碼，請參閱 [https : //github.com/aws-labs/aws-codebuild-jenkins-plugin](https://github.com/aws-labs/aws-codebuild-jenkins-plugin)。

安裝外掛程式。

如果您已設定 Jenkins 伺服器，並且只想要安裝 AWS CodeBuild 外掛程式，則請在您的 Jenkins 執行個體上的 Plugin Manager 中搜尋 **CodeBuild Plugin for Jenkins**。

使用 外掛程式

AWS CodeBuild 搭配 VPC 外部的來源使用

1. 在 CodeBuild 主控台中建立專案。如需詳細資訊，請參閱[建立組建專案 \(主控台\)](#)。
 - 選擇您要執行建置 AWS 的區域。
 - (選用) 設定 Amazon VPC 組態，以允許 CodeBuild 組建容器存取 VPC 中的資源。
 - 寫下專案的名稱。您在步驟 3 會用到。
 - (選用) 如果 CodeBuild 原生不支援您的來源儲存庫，您可以將 Amazon S3 設定為專案的輸入來源類型。
2. 在 IAM 主控台中，建立 Jenkins 外掛程式要使用的使用者。
 - 建立使用者的登入資料時，請選擇 Programmatic Access (程式設計存取)。
 - 建立類似以下的政策，然後將政策連接至您的使用者。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:/aws/codebuild/{{projectName}}:*"
      ],
      "Action": [
        "logs:GetLogEvents"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::{{inputBucket}}"
      ],
      "Action": [
        "s3:GetBucketVersioning"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::{{inputBucket}}/{{inputObject}}"
      ],
      "Action": [
        "s3:PutObject"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::{{outputBucket}}/*"
      ],
      "Action": [
        "s3:GetObject"
      ]
    }
  ]
}
```

```
        "Effect": "Allow",
        "Resource": [
            "arn:aws:codebuild:us-east-1:111122223333:project/
{{projectName}}"
        ],
        "Action": [
            "codebuild:StartBuild",
            "codebuild:BatchGetBuilds",
            "codebuild:BatchGetProjects"
        ]
    }
}
```

3. 在 Jenkins 中建立自由形式專案。

- 在設定頁面上，選擇新增建置步驟，然後選擇在 CodeBuild 上執行建置。
- 設定組建步驟。
 - 提供 Region (區域)、Credentials (登入資料) 和 Project Name (專案名稱) 的值。
 - 選擇 Use Project source (使用專案來源)。
 - 儲存組態，並從 Jenkins 執行組建。

4. 針對 Source Code Management (來源碼管理)，選擇要擷取來源的方式。您可能需要在 Jenkins 伺服器上安裝 GitHub 外掛程式 (或 Jenkins 外掛程式做為您的來源儲存庫提供者)。

- 在設定頁面上，選擇新增建置步驟，然後選擇執行建置。AWS CodeBuild
- 設定組建步驟。
 - 提供 Region (區域)、Credentials (登入資料) 和 Project Name (專案名稱) 的值。
 - 選擇 Use Jenkins source (使用 Jenkins 來源)。
 - 儲存組態，並從 Jenkins 執行組建。

將 AWS CodeBuild 外掛程式與 Jenkins 管道外掛程式搭配使用

- 在 Jenkins 管道專案頁面上，使用程式碼片段產生器產生管道指令碼，將 CodeBuild 新增為管道中的步驟。它應該會產生類似以下的指令碼：

```
awsCodeBuild projectName: 'project', credentialsType: 'keys', region: 'us-west-2',
sourceControlType: 'jenkins'
```

AWS CodeBuild 搭配無伺服器應用程式使用

AWS Serverless Application Model (AWS SAM) 是用於建置無伺服器應用程式的開放原始碼架構。如需詳細資訊，請參閱 GitHub 上的 [AWS Serverless Application Model](#) 儲存庫。

您可以使用 AWS CodeBuild 來封裝和部署遵循標準的 AWS SAM 無伺服器應用程式。針對部署步驟，CodeBuild 可以使用 AWS CloudFormation。若要使用 CodeBuild 和 自動化無伺服器應用程式的建置和部署 CloudFormation，您可以使用 AWS CodePipeline。

如需詳細資訊，請參閱 [《開發人員指南》中的部署無伺服器應用程式](#)。AWS Serverless Application Model

相關資源

- 如需 入門的詳細資訊 AWS CodeBuild，請參閱 [AWS CodeBuild 開始使用 主控台](#)。
- 如需 CodeBuild 中問題的疑難排解資訊，請參閱 [故障診斷 AWS CodeBuild](#)。
- 如需 CodeBuild 中配額的相關資訊，請參閱 [的配額 AWS CodeBuild](#)。

AWS CodeBuild 適用於 Windows 的第三方通知

當您使用 CodeBuild for Windows 組建時，您可以選擇使用某些第三方套件和模組，讓您的建置應用程式能夠在 Microsoft Windows 作業系統上執行，並與某些第三方產品互通。下列清單包含適用的第三方法律條款，規範您如何使用指定的第三方套件和模組。

主題

- [1\) 基本 Docker 影像—windowsservercore](#)
- [2\) Windows 基底 Docker 影像—choco](#)
- [3\) Windows-base Docker 影像 -- 2.16.2 版](#)
- [4\) Windows-base Docker 影像 -microsoft-build-tools -- 15.0.26320.2 版](#)
- [5\) Windows-base Docker 影像—nuget.commandline -- 4.5.1 版](#)
- [7\) Windows-base Docker 影像—netfx-4.6.2-devpack](#)
- [8\) Windows-base Docker 映像—visualfsharptools，4.0 版](#)
- [9\) Windows-base Docker 影像—netfx-pcl-reference-assemblies-4.6](#)
- [10\) Windows 基礎 Docker 映像—visualcppbuildtools v 14.0.25420.1](#)
- [11\) windows-base Docker 影像—microsoft-windows-netfx3-ondemand-package.cab](#)

- [12\) Windows-base Docker 影像—dotnet-sdk](#)

1) 基本 Docker 影像—windowsservercore

(授權條款請參閱：https://hub.docker.com/_/microsoft-windows-servercore)

授權：要求和使用此 Container OS Image for Windows 容器，即表示您認可、了解和同意下列增補授權條款：

MICROSOFT 軟體增補授權條款

容器作業系統映像

Microsoft Corporation (或以您的居住地、其附屬公司之一為準) (以下簡稱「我們」或 "Microsoft") 授權您使用此「容器作業系統映像」增補 (以下簡稱「增補」)。僅授權您使用此增補連同基礎主機作業系統軟體 (以下簡稱「主機軟體」) 來輔助在主機軟體中執行容器功能。您使用此增補須受主機軟體授權條款之規範。若您沒有主機軟體的授權，請勿使用此增補。您可以搭配主機軟體的每個有效授權副本使用此增補。

其他授權要求和/或使用權利

您依前段規定使用此增補時可能建立或修改含有某些增補元件的容器映像 (以下簡稱「容器映像」)。需要釐清，容器映像與虛擬機器或虛擬應用裝置映像不同。根據這些授權條款，我們授予您有限權利遵守下列條件再散佈這些增補元件：

- (i) 您只能在容器映像中使用並視同其一部分來使用增補元件，
- (ii) 只要您的容器映像中有顯著的主要功能實質上有別於增補，您得使用這些增補元件；以及
- (iii) 您同意在您的容器映像中附上這些授權條款 (或我們或託管服務提供者要求的類似條款) 以適當授權最終使用者可能使用增補元件。

我們保留本文未明確授予的其他所有權利。

使用此增補即表示您接受這些條款。如不接受，請勿使用此增補。

根據此 Container OS Image for Windows 容器的增補授權條款，您亦受基礎 Windows Server 主機軟體授權條款 (<https://www.microsoft.com/en-us/usetems>) 的約束。

2) Windows 基底 Docker 影像—choco

(授權條款請參閱：<https://github.com/chocolatey/choco/blob/master/LICENSE>)

Copyright 2011 - Present RealDimensions Software, LLC

依據 Apache License 2.0 版授權 (以下簡稱「授權」)，除非符合授權，否則您不得使用這些檔案。您可以在此處取得授權副本：

<http://www.apache.org/licenses/LICENSE-2.0>

除非準據法所要求或經書面同意，依據授權散佈之軟體係以「現狀」散佈，不附帶任何明示或默示之保證或條件。請參閱授權了解其中特定的語言規範許可及限制。

3) Windows-base Docker 影像 -- 2.16.2 版

(授權條款位於：<https://chocolatey.org/packages/git/2.16.2>)

依據 GNU 一般公有授權第 2 版授權，位於：<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html>。

4) Windows-base Docker 影像 -microsoft-build-tools -- 15.0.26320.2 版

(授權條款位於：<https://www.visualstudio.com/license-terms/mt171552/>)

MICROSOFT VISUAL STUDIO 2015 擴充功能、VISUAL STUDIO SHELL 和 C++ 可轉散發套件

這些授權條款是 Microsoft Corporation (或以您的居住地、其附屬公司之一為準) 與您之間的協議。適用於上述軟體。這些條款亦適用於本軟體的任何 Microsoft 服務或更新，惟尚有其他條款的服務或更新則屬例外。

若您遵守這些授權條款，則您享有下列權利。

1. 安裝和使用權利。您得安裝和使用任意數量的軟體副本。
2. 特定元件的條款。
 - a. 公用程式。軟體可能包含位於 <https://docs.microsoft.com/en-us/visualstudio/productinfo/2015-redistribution-vs> 的公用程式清單中的一些項目。如果軟體包含這些項目，您可以在您的或其他第三方機器上複製並安裝這些項目，以偵錯和部署您使用軟體開發的應用程式和資料庫。請注意，公用程式的設計是臨時用途、Microsoft 可能無法於本軟體其餘部分之外另行修補或更新公用程式，以及某些公用程式本質上可能允許其他人存取其安裝所在的機器。因此，當您偵錯或部署完畢您的應用程式和資料庫之後，請刪除您已安裝的所有公用程式。對於第三方如何使用或存取您安裝於任何機器上的公用程式，Microsoft 概不負責。

- b. Microsoft 平台。本軟體可能包含來自 Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office 和 Microsoft SharePoint 的元件。這些元件受另外合約及其各自產品支援政策所規範，詳述於該元件之安裝目錄或本軟體隨附之 "Licenses" 資料夾中的授權條款。
- c. 第三方元件。本軟體可能包含另有法律聲明或受其他合約所規範的第三方元件，詳述於本軟體隨附的 ThirdPartyNotices 檔案中。即使這些元件受其他合約所規範，但以下損害之免責聲明及限制和免除仍適用。本軟體亦可能包含依據開放原始碼授權所授權且負有原始碼可得性義務的元件。ThirdPartyNotices 檔案包含這些授權的副本 (如適用)。依據相關開放原始碼授權，如有必要，您得向我們索取此原始碼，請將 5.00 USD 的匯票或支票寄到此地址：Source Code Compliance Team, Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052。請於付款備忘欄註明以下一或多個元件的原始碼：
- Remote Tools for Visual Studio 2015;
 - Standalone Profiler for Visual Studio 2015;
 - IntelliTraceCollector for Visual Studio 2015;
 - Microsoft VC++ Redistributable 2015;
 - Multibyte MFC Library for Visual Studio 2015;
 - Microsoft Build Tools 2015;
 - Feedback Client;
 - Visual Studio 2015 Integrated Shell ; 或
 - Visual Studio 2015 Isolated Shell。

您亦可於 <http://thirdpartysource.microsoft.com> 取得原始碼副本。

3. 資料。本軟體可能收集有關於您和您如何使用本軟體的資訊，並傳送到 Microsoft。Microsoft 得使用此資訊來提供服務及改善我們的產品與服務。依產品文件所述，您得退出其中許多方案，但非全部。本軟體亦有某些功能可讓您收集應用程式使用者的資料。若您使用這些功能在應用程式中收集資料，則須遵循準據法，包括向應用程式的使用者提供適當聲明。您可以在說明文件和隱私權聲明中進一步了解資料收集和使用，請參閱 <https://privacy.microsoft.com/en-us/privacystatement>。您須同意這些慣例方可使用本軟體。
4. 授權範圍。本軟體係授權使用，非經銷售取得。本合約僅授予您使用本軟體的某些權利。Microsoft 保留其他所有權利。除非準據法授予您超出此限制的權利，否則您僅可依本合約明確許可的方式使用本軟體。這樣做時，您須遵守本軟體中僅允許您以特定方式使用本軟體的任何技術限制。您不得
- 規避本軟體的任何技術限制；
 - 對本軟體進行逆向工程、反編譯或反組譯，或企圖從事此行為，除非規範使用本軟體可能隨附之特定開放原始碼元件的第三方授權條款有此必要，則屬例外；

- 移除、限縮、遮蓋或修改 Microsoft 或其供應商於本軟體中的任何聲明；
 - 以任何違法形式使用本軟體；或
 - 分享、發行、出租或租賃本軟體，或以獨立託管解決方案提供本軟體供他人使用。
5. 出口限制。您須遵守適用於本軟體的所有國內和國際出口法規，包括限制目的地、最終使用者和最終用途。如需出口限制的進一步資訊，請造訪 (aka.ms/exporting)。
6. 支援服務。因為本軟體「依現狀」提供，我們可能不提供支援服務。
7. 完整合約。本合約、您使用的增補、更新、網際網路服務與支援服務的條款，構成軟體與支援服務的完整合約。
8. 準據法。若您於美國取得本軟體，則違反本合約之解釋與主張以華盛頓法律為準，其他所有主張依您居住地之州法為準。若您於其他任何國家取得本軟體，則依其法律為準。
9. 消費者權利；區域性差異。本合約描述特定法律權利。依據您所在地區或國家的法律，您可能享有其他權利，包括消費者權利。姑且不論您與 Microsoft 的關係，就供您取得本軟體的那一方而言，您亦可能享有權利。若您所在地區或國家不允許變更這些權利，本合約亦同。假設您於下列地區之一取得本軟體，或國家強制法律實施，您須遵守下列規定：
- a. 澳大利亞。您依據澳大利亞消費者法律而享有法定保障，本合約絕不影響那些權利。
 - b. 加拿大。若您於加拿大取得本軟體，您得關閉自動更新功能、中斷裝置與網際網路的連線 (然而，若您重新連上網際網路，本軟體會繼續檢查是否有更新並安裝) 或解除安裝本軟體，以停止接收更新。產品文件 (若有) 亦可能指明如何對您的特定裝置或軟體關閉更新。
 - c. 德國和奧地利。
 - i. 保固。適當授權軟體的運作大致上符合本軟體隨附之任何 Microsoft 資料所述。然而，Microsoft 對於授權軟體不提供任何契約保證。
 - ii. 有限責任。在故意行為、重大過失、依據產品責任法而請求賠償的情況下，以及在引起死亡或人身傷害的情況下，Microsoft 依成文法負起法律責任。依據前述條款 (ii)，倘若 Microsoft 違反上述重要契約義務、因履行而抵觸本合約、因違約而危及本合約之宗旨及一方一貫信任之承諾 (即所謂「基本義務」)，Microsoft 僅負起輕微過失之責任。在其他輕微過失情況下，Microsoft 不負輕微過失之責任。
10. 免責聲明。本軟體「依現狀」授權。您承擔使用風險。MICROSOFT 不提供明確保固、保證或條件。在當地法律允許的範圍內，MICROSOFT 排除適銷性、符合特定目的和未侵權的隱含保證。
11. 限制和排除損害。您得向 MICROSOFT 及其供應商僅索取直接損害賠償最多 5.00 USD。您無法復原任何其他損害，包括衍生性損害、利潤損失、特殊損害、間接損害或附隨性損害。此限制適用於 (a) 第三方網站或第三方應用程式之軟體、服務、內容 (包括程式碼) 的任何相關情事；(b) 對違約、未履行保固、保證或條件、嚴格責任、過失或準據法允許範圍內之其他侵權行為所提起之索賠。

即使 Microsoft 已知或應知損害發生之可能性，亦適用。上述限制或免除可能不適用於您，因為您的國家可能不允許排除或限制附帶、間接或其他損害賠償。

EULA ID: VS2015_Update3_ShellsRedist_<ENU>

5) Windows-base Docker 影像—nuget.commandline -- 4.5.1 版

(授權條款位於：<https://github.com/NuGet/Home/blob/dev/LICENSE.txt>)

Copyright (c) .NET Foundation. 保留所有權利。

依據 Apache License 2.0 版授權 (以下簡稱「授權」)，除非符合授權，否則您不得使用這些檔案。您可以在此處取得授權副本：

<http://www.apache.org/licenses/LICENSE-2.0>

除非準據法所要求或經書面同意，依據授權散佈之軟體係以「現狀」散佈，不附帶任何明示或默示之保證或條件。請參閱授權了解其中特定的語言規範許可及限制。

7) Windows-base Docker 影像—netfx-4.6.2-devpack

MICROSOFT 軟體增補授權條款

.NET FRAMEWORK 和 MICROSOFT WINDOWS 作業系統相關的語言套件

Microsoft Corporation (或以您的居住地、其附屬公司之一為準) 授權您使用此增補。若您獲授權使用 Microsoft Windows 作業系統軟體 (以下簡稱「軟體」)，您得使用此增補。若您沒有本軟體的授權，請勿使用此增補。您得將此增補用於本軟體的每個有效授權副本。

下列授權條款描述此增補的其他使用條款。您使用此增補須受這些條款和本軟體的授權條款所規範。若有衝突，以這些增補授權條款為準。

使用此增補，即表示您接受這些條款。如果您不接受，請勿使用此增補。

若您遵守這些授權條款，則您享有下列權利。

1. 可分發程式碼。 補充包含可分發程式碼。「可散發程式碼」係指若您遵守下列條款，則允許在您開發的程式中散發的程式碼。

- a. 使用和散發權利。
 - 您得以目的碼形式複製和散發此增補。
 - 第三方散發。您得允許您程式的散發者隨著這些程式複製和散發可散發程式碼。
 - b. 分佈要求。對於您分發的任何可分發程式碼，您必須
 - 在您的程式中對它新增顯著的主要功能；
 - 對於副檔名為 .lib 的任何可散發程式碼，僅透過您程式的連結器來散發該可散發程式碼的執行結果；
 - 直接散發隨附於安裝程式的可散發程式碼，不得修改安裝程式；
 - 要求散發者和外部最終使用者至少同意相當於本合約的保護條款；
 - 在您的程式上顯示有效的著作權聲明；以及
 - 保障、維護和免除 Microsoft 免於承擔因散發或使用您的程式而引起之任何索賠，包括律師費。
 - c. 分佈限制。您可能不會
 - 於可散發程式碼中變更任何著作權、商標或專利聲明；
 - 在您的程式名稱中或以暗示您的程式來自或由 Microsoft 背書的任何形式使用 Microsoft 的商標；
 - 將可散發程式碼散發到 Windows 平台以外的平台上執行；
 - 在惡意、詐欺或非法律程式中包含可散發程式碼；或
 - 因修改或散發任何可散發程式碼的原始碼，而致使其任何部分受到「排除授權」的約束。排除授權是指依據使用、修改或散發的條件，要求
 - 以原始碼形式來公開或散發程式碼；或
 - 其他人有修改的權利。
2. 增補的支援服務。Microsoft 提供本軟體的支援服務，請參閱 www.support.microsoft.com/common/international.aspx。

8) Windows-base Docker 映像—visualfsharpools，4.0 版

(授權條款請參閱：<https://github.com/dotnet/fsharp/blob/main/License.txt>)

Copyright (c) Microsoft Corporation. 保留所有權利。

依據 Apache License 2.0 版授權 (以下簡稱「授權」)，除非符合授權，否則您不得使用這些檔案。您

<http://www.apache.org/licenses/LICENSE-2.0>

除非準據法所要求或經書面同意，依據授權散佈之軟體係以「現狀」散佈，不附帶任何明示或默示之保證或條件。請參閱授權了解其中特定的語言規範許可及限制。

9) Windows-base Docker 影像—netfx-pcl-reference-assemblies-4.6

MICROSOFT 軟體授權條款

MICROSOFT .NET 可攜式類別庫參考組件 – 4.6

這些授權條款是 Microsoft Corporation (或以您的居住地、其附屬公司之一為準) 與您之間的協議。請閱讀。適用於上述軟體。這些條款亦適用於本軟體的任何 Microsoft

- 更新、
- 增補、
- 網際網路服務及
- 支援服務，

除非那些項目還隨附其他條款。若是如此，以那些條款為準。

使用該軟體，即表示您接受這些條款。如果您不接受，請勿使用軟體。

若您遵守這些授權條款，則您享有下列永久權利。

1. 安裝和使用權利。您得安裝和使用本軟體任意數量的副本來設計、開發和測試您的程式。
2. 其他授權要求和/或使用權利。
 - a. 可散發程式碼。若您遵守下列條款，則可以在您開發的開發人員工具程式中散發本軟體，讓您程式的客戶能夠開發適用於任何裝置或作業系統的可攜式程式庫。
 - i. 使用和散發權利。軟體是「可散發程式碼」。
 - 可散發程式碼。您得以目的碼形式複製和散發本軟體。
 - 第三方散發。您得允許您程式的散發者隨著這些程式複製和散發可散發程式碼。
 - ii. 分佈要求。對於您分發的任何可散發程式碼，您必須
 - 在您的程式中對它新增顯著的主要功能；

- 要求散發者和您的客戶至少同意相當於本合約的保護條款；
- 在您的程式上顯示有效的著作權聲明；以及
- 保障、維護和免除 Microsoft 免於承擔因散發或使用您的程式而引起之任何索賠，包括律師費。

iii. 分佈限制。您可能不會

- 於可散發程式碼中變更任何著作權、商標或專利聲明；
- 在您的程式名稱中或以暗示您的程式來自或由 Microsoft 背書的任何形式使用 Microsoft 的商標；
- 在惡意、詐欺或非法律程式中包含可散發程式碼；或
- 因修改或散發任何可散發程式碼，而致使其任何部分受到「排除授權」的約束。排除授權是指依據使用、修改或散發的條件，要求
 - 以原始碼形式來公開或散發程式碼；或
 - 其他人有修改的權利。

3. 授權範圍。本軟體係授權使用，非經銷售取得。本合約僅授予您使用本軟體的某些權利。Microsoft 保留其他所有權利。除非準據準授予您超出此限制的權利，否則您僅可依本合約明確許可的方式使用本軟體。這樣做時，您須遵守本軟體中僅允許您以特定方式使用本軟體的任何技術限制。您不得

- 規避本軟體的任何技術限制；
- 對本軟體進行逆向工程、反編譯或反組譯，除非且僅限於準據準明確允許的範圍內，而不論此限制；
- 發佈本軟體供他人複製；或
- 出租、租賃或出借本軟體。

4. 意見回饋。您得提供有關本軟體的意見回饋。若您將有關本軟體的意見回饋提供給 Microsoft，即表示您免費授權 Microsoft 以任何方式和出於任何目的使用、分享和商品化您的意見回饋。您亦免費授權第三方在其產品、技術和服務上所需的任何專利權，以使用或銜接 Microsoft 軟體或服務中包含此意見回饋的任何特定部分。因為我們將您的意見回饋納入 Microsoft 的軟體或文件中，所以您提供的意見回饋不會受到授權的約束，而致使 Microsoft 必須將軟體或文件授權給第三方。這些權利在本合約終止後仍然有效。

5. 轉移給第三方。本軟體的首位使用者得將本軟體和本合約直接轉移給第三方。轉移之前，該方須同意本合約適用於本軟體之轉移與使用。首位使用者將本軟體轉移脫離裝置之前，須解除安裝本軟體。首位使用者不得保留任何副本。

6. 出口限制。本軟體受美國出口法規的約束。您須遵守適用於本軟體的所有國內和國際出口法規。這些法律包括限制目的地、最終使用者和最終用途。如需其他資訊，請參閱 www.microsoft.com/exporting。

7. 支援服務。因為本軟體「依現狀」提供，我們可能不提供支援服務。
8. 完整合約。本合約、您使用的增補、更新、網際網路服務與支援服務的條款，構成本軟體與我們提供之任何支援服務的完整合約。
9. 準據法。
 - a. 美國。若您於美國取得本軟體，則違反本合約之解釋與主張，無論是否與法律原則衝突，概以華盛頓法律為準。其他所有主張以您居住地州法為準，包括依據消費者保護法、不公平競爭法和侵權行為而提起之主張。
 - b. 美國境外。若您於其他任何國家取得本軟體，則依該國法律為準。
10. 法律效力。本合約描述特定法律權利。依據您所在國家的法律，您可能享有其他權利。就供您取得本軟體的那一方而言，您亦可能享有權利。若貴國法律不允許變更您依據貴國法律而享有的權利，本合約亦同。
11. 免責聲明。本軟體「依現狀」授權。您承擔使用風險。MICROSOFT 不提供明確保固、保證或條件。依據當地法律，您得享有本合約不得變更之其他消費者權利或法定保障。在當地法律允許的範圍內，MICROSOFT 排除適銷性、符合特定目的和未侵權的隱含保證。

FOR AUSTRALIA - 根據澳洲消費者法，您有法定保證，這些條款中的任何內容都無意影響這些權利。

12. 對補救措施和損害的限制和排除。您得向 MICROSOFT 及其供應商僅索取直接損害賠償最多 5.00 USD。您無法復原任何其他損害，包括衍生性損害、利潤損失、特殊損害、間接損害或附隨性損害。

此限制適用於

- 第三方網站或第三方應用程式之軟體、服務、內容 (包括程式碼) 的任何相關情事；以及
- 對違約、未履行保固、保證或條件、嚴格責任、過失或準據法允許範圍內之其他侵權行為所提起之索賠。

即使 Microsoft 已知或應知損害發生之可能性，亦適用。上述限制或免除可能不適用於您，因為您的國家可能不允許排除或限制附帶、間接或其他損害賠償。

10) Windows 基礎 Docker 映像—visualcppbuildtools v 14.0.25420.1

(授權條款位於：<https://www.visualstudio.com/license-terms/mt644918/>)

MICROSOFT VISUAL C++ BUILD TOOLS

MICROSOFT 軟體授權條款

MICROSOFT VISUAL C++ BUILD TOOLS

這些授權條款是 Microsoft Corporation (或以您的居住地、其附屬公司之一為準) 與您之間的協議。適用於上述軟體。這些條款亦適用於本軟體的任何 Microsoft 服務或更新，惟另有條款的服務或更新則屬例外。

若您遵守這些授權條款，則您享有下列權利。

1. 安裝和使用權利。

a. 一位使用者得使用本軟體的副本來設計和測試其應用程式。

2. 資料。本軟體可能收集有關於您和您如何使用本軟體的資訊，並傳送到 Microsoft。Microsoft 得使用此資訊來提供服務及改善我們的產品與服務。依產品文件所述，您得退出其中許多方案，但非全部。本軟體亦有某些功能可讓您收集應用程式使用者的資料。若您使用這些功能在應用程式中收集資料，則須遵循準據法，包括向應用程式的使用者提供適當聲明。您可以在說明文件和隱私權聲明中進一步了解資料收集和使用，請參閱 <http://go.microsoft.com/fwlink/?LinkID=528096>。您須同意這些慣例方可使用本軟體。

3. 特定元件的條款。

a. 組建伺服器。本軟體可能包含 BuildServer.TXT 檔案中列出的一些組建伺服器元件，和/或此 Microsoft 軟體授權條款後面的 BuildServer 清單上列出的任何檔案。您得將這些項目 (若隨附於本軟體) 複製和安裝到您的組建機器上。您與貴組織中的其他人得在你們的組建機器上使用這些項目，純粹只為了編譯、組建、驗證和封存你們的應用程式，或在組建過程中執行品質或效能測試。

b. Microsoft 平台。本軟體可能包含來自 Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office 和 Microsoft SharePoint 的元件。這些元件受另外合約及其各自產品支援政策所規範，詳述於該元件之安裝目錄或本軟體隨附之 "Licenses" 資料夾中的授權條款。

c. 第三方元件。本軟體可能包含另有法律聲明或受其他合約所規範的第三方元件，詳述於本軟體隨附的 ThirdPartyNotices 檔案中。即使這些元件受其他合約所規範，但以下損害之免責聲明及限制和免除仍適用。

d. 封裝管理員。本軟體可能包含封裝管理員，例如 Nuget，可讓您選擇下載其他 Microsoft 和第三方軟體套件來用於您的應用程式。這些套件受其本身授權所規範，而非本合約。Microsoft 對任何第三方套件不散發、授權或提供任何保證。

4. 授權範圍。本軟體係授權使用，非經銷售取得。本合約僅授予您使用本軟體的某些權利。Microsoft 保留其他所有權利。除非準據準授予您超出此限制的權利，否則您僅可依本合約明確許可的方式使用本軟體。這樣做時，您須遵守本軟體中僅允許您以特定方式使用本軟體的任何技術限制。如需詳細資訊，請參閱 <https://docs.microsoft.com/en-us/legal/information-protection/software-license-terms#1-installation-and-use-rights>。您不得
 - 規避本軟體的任何技術限制；
 - 對本軟體進行逆向工程、反編譯或反組譯，或企圖從事此行為，除非規範使用本軟體可能隨附之特定開放原始碼元件的第三方授權條款有此必要，則屬例外；
 - 移除、限縮、遮蓋或修改 Microsoft 或其供應商的任何聲明；
 - 以任何違法形式使用本軟體；或
 - 分享、發行、出租或租賃本軟體，或以獨立託管解決方案提供本軟體供他人使用。
5. 出口限制。您須遵守適用於本軟體的所有國內和國際出口法規，包括限制目的地、最終使用者和最終用途。如需出口限制的進一步資訊，請造訪 (aka.ms/exporting)。
6. 支援服務。因為本軟體「依現狀」提供，我們可能不提供支援服務。
7. 完整合約。本合約、您使用的增補、更新、網際網路服務與支援服務的條款，構成軟體與支援服務的完整合約。
8. 準據法。若您於美國取得本軟體，則違反本合約之解釋與主張以華盛頓法律為準，其他所有主張依您居住地之州法為準。若您於其他任何國家取得本軟體，則依其法律為準。
9. 消費者權利；區域性差異。本合約描述特定法律權利。依據您所在地區或國家的法律，您可能享有其他權利，包括消費者權利。姑且不論您與 Microsoft 的關係，就供您取得本軟體的那一方而言，您亦可能享有權利。若您所在地區或國家不允許變更這些權利，本合約亦同。假設您於下列地區之一取得本軟體，或國家強制法律實施，您須遵守下列規定：
 - 澳大利亞。您依據澳大利亞消費者法律而享有法定保障，本合約絕不影響那些權利。
 - 加拿大。若您於加拿大取得本軟體，您得關閉自動更新功能、中斷裝置與網際網路的連線 (然而，若您重新連上網際網路，本軟體會繼續檢查是否有更新並安裝) 或解除安裝本軟體，以停止接收更新。產品文件 (若有) 亦可能指明如何對您的特定裝置或軟體關閉更新。
 - 德國和奧地利。
 - 保固。適當授權軟體的運作大致上符合本軟體隨附之任何 Microsoft 資料所述。然而，Microsoft 對於授權軟體不提供任何契約保證。
 - 有限責任。在故意行為、重大過失、依據產品責任法而請求賠償的情況下，以及在引起死亡或人身傷害的情況下，Microsoft 依成文法負起法律責任。

依據前述條款 (ii)，倘若 Microsoft 違反上述重要契約義務、因履行而抵觸本合約、因違約而危及本合約之宗旨及一方一貫信任之承諾 (即所謂「基本義務」)，Microsoft 僅負起輕微過失之責任。在其他輕微過失情況下，Microsoft 不負輕微過失之責任。

10 法律效力。本合約描述特定法律權利。依據您所在地區或國家的法律，您可能享有其他權利。若您所在地區或國家的法律不允許變更您依據所在地區或法律而享有的權利，本合約亦同。不受前述條款所限制，在澳大利亞，您依據澳大利亞消費者法律而享有法定保障，本合約絕不影響那些權利

11 免責聲明。本軟體「依現狀」授權。您承擔使用風險。MICROSOFT 不提供明確保固、保證或條件。在當地法律允許的範圍內，MICROSOFT 排除適銷性、符合特定目的和未侵權的隱含保證。

12 限制和排除損害。您得向 MICROSOFT 及其供應商僅索取直接損害賠償最多 5.00 USD。您無法復原任何其他損害，包括衍生性損害、利潤損失、特殊損害、間接損害或附隨性損害。

此限制適用於 (a) 第三方網站或第三方應用程式之軟體、服務、內容 (包括程式碼) 的任何相關情事；(b) 對違約、未履行保固、保證或條件、嚴格責任、過失或準據法允許範圍內之其他侵權行為所提起之索賠。

即使 Microsoft 已知或應知損害發生之可能性，亦適用。上述限制或免除可能不適用於您，因為您的國家可能不允許排除或限制附帶、間接或其他損害賠償。

11) windows-base Docker 影像—microsoft-windows-netfx3-ondemand-package.cab

MICROSOFT 軟體增補授權條款

.NET FRAMEWORK 和 MICROSOFT WINDOWS 作業系統相關的語言套件

Microsoft Corporation (或以您的居住地、其附屬公司之一為準) 授權您使用此增補。若您獲授權使用 Microsoft Windows 作業系統軟體 (適用此增補) (以下簡稱「軟體」)，您得使用此增補。若您沒有本軟體的授權，請勿使用此增補。您得將此增補的副本用於本軟體的每個有效授權副本。

下列授權條款描述此增補的其他使用條款。您使用此增補須受這些條款和本軟體的授權條款所規範。若有衝突，以這些增補授權條款為準。

使用此增補，即表示您接受這些條款。如果您不接受，請勿使用此增補。

若您遵守這些授權條款，則您享有下列權利。

1. 增補的支援服務。Microsoft 提供本軟體的支援服務，請參閱 www.support.microsoft.com/common/international.aspx。
2. MICROSOFT .NET 基準測試。本軟體包含 Windows 作業系統的 .NET Framework、Windows Communication Foundation、Windows Presentation Foundation 和 Windows Workflow Foundation 元件 (.NET 元件)。您得進行 .NET 元件的內部基準測試。倘若您遵守 <http://go.microsoft.com/fwlink/?LinkID=66406> 列舉的條件，您得公開 .NET 元件之任何基準測試的結果。

無論您與 Microsoft 之間是否有其他任何協議，若您公開這些基準測試結果，對於您與適用之 .NET 元件競爭的產品，Microsoft 應有權利公開其對您的產品所進行之基準測試的結果，前提是遵守 <http://go.microsoft.com/fwlink/?LinkID=66406> 列舉的相同條件。

12) Windows-base Docker 影像—dotnet-sdk

(可在 <https://github.com/dotnet/core/blob/main/LICENSE.TXT> 取得)

MIT 授權 (MIT)

Copyright (c) Microsoft Corporation

在此免費准許任何取得本軟體之副本及相關文件檔案 (簡稱「本軟體」) 的任何人不受限制地運用本軟體，權利包括不限於使用、複製、修改、合併、發佈、散發、再授權和/或銷售本軟體的副本，以及允許取得本軟體的人如此做，惟受下列條件的約束：

本軟體所有副本或重要部分應包含上述版權聲明和此許可聲明。

本軟體「依現狀」提供，不作任何明示或暗示的保證，包括但不限於對適銷性、特定用途適用性和不侵權的保證。不論起因於、出自或涉及本軟體，或因使用或其他買賣本軟體之行為而發生之契約訴訟、侵權行為或其他情事，無論如何，作者或著作權持有人不承擔任何索賠、損害或其他責任。

使用 CodeBuild 條件索引鍵做為 IAM 服務角色變數，以控制組建存取

透過 CodeBuild 組建 ARN，您可以使用內容索引鍵來限制 CodeBuild 服務角色中的資源存取範圍，以限制 CodeBuild 資源存取。對於 CodeBuild，可用於控制建置存取行為的金鑰為 `codebuild:buildArn` 和 `codebuild:projectArn`。使用建置專案 ARN，您可以驗證對資源的呼叫是否來自特定的建置專案。若要驗證這一點，請在 IAM 身分型政策中使用 `codebuild:buildArn` 或 `codebuild:projectArn` 條件金鑰。

若要在政策中使用 `codebuild:buildArn` 或 `codebuild:projectArn` 條件索引鍵，請在任何 ARN 條件運算子中包含它做為條件。金鑰的值必須是解析為有效 ARN 的 IAM 變數。在下面的範例政策中，允許的唯一存取權是使用 `${codebuild:projectArn}` IAM 變數的專案 ARN 存取建置專案。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket-name/${codebuild:projectArn}/*"
    }
  ]
}
```

AWS CodeBuild 條件索引鍵

AWS CodeBuild 提供一組條件金鑰，可用於 IAM 政策，以在專案和機群等 CodeBuild 資源上強制執行組織政策。條件索引鍵涵蓋大部分 CodeBuild API 請求內容，包括網路設定、登入資料組態和運算限制。

主題

- [在專案和機群上強制執行 VPC 連線設定](#)
- [防止對專案 `buildspec` 進行未經授權的修改](#)
- [限制您建置的運算類型](#)
- [控制環境變數設定](#)
- [在條件索引鍵名稱中使用變數](#)
- [檢查 API 請求中是否存在屬性](#)

在專案和機群上強制執行 VPC 連線設定

此政策允許發起人在建立 CodeBuild 專案和機群時，使用選取的 VPCs、子網路和安全群組。如需多值內容索引鍵的詳細資訊，請參閱[單一值與多值內容索引鍵](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "codebuild:CreateProject",
      "codebuild:CreateFleet"
    ],
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "codebuild:vpcConfig.vpcId": [
          "vpc-01234567890abcdef",
          "vpc-abcdef01234567890"
        ],
        "codebuild:vpcConfig.subnets": [
          "subnet-1234abcd",
          "subnet-5678abcd"
        ],
        "codebuild:vpcConfig.securityGroupIds": [
          "sg-12345678abcdefghij",
          "sg-01234567abcdefghij"
        ]
      }
    }
  ]
}
```

防止對專案 buildspec 進行未經授權的修改

此政策不允許發起人覆寫 buildspecOverride 欄位中的 buildspec。

Note

codebuild:source.buildspec 條件索引鍵僅支援 Null 運算子來檢查 API 欄位是否存在。它不會評估 buildspec 的內容。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "codebuild:StartBuild",
    "Resource": "*"
  }, {
    "Effect": "Deny",
    "Action": "codebuild:StartBuild",
    "Resource": "*",
    "Condition": {
      "Null": {
        "codebuild:source.buildspec": "false"
      }
    }
  }
]}
}
```

限制您建置的運算類型

此政策允許建立只能使用 c5.large 或 m5.large [運算執行個體類型](#) 建置的機群。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "codebuild:CreateFleet",
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "codebuild:computeConfiguration.instanceType": ["c5.large",
"m5.large"]
      }
    }
  }
]}
}
```

控制環境變數設定

此政策允許發起人覆寫STAGE環境變數為 BETA或 GAMMA。它也明確拒絕覆寫STAGE為 PRODUCTION，並拒絕覆寫MY_APP_VERSION環境變數。如需多個值內容索引鍵，請參閱[單一值與多值內容索引鍵](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "codebuild:environment.environmentVariables/STAGE.value": [
            "BETA",
            "GAMMA"
          ]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:StartBuild"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "codebuild:environment.environmentVariables/STAGE.value":
            "PRODUCTION"
        },
        "ForAnyValue:StringEquals": {
          "codebuild:environment.environmentVariables.name": [
            "MY_APP_VERSION"
          ]
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

在條件索引鍵名稱中使用變數

您可以在條件索引鍵名稱中使用變數，例如 `secondarySources/${sourceIdentifier}.location` 和 `secondaryArtifacts/${artifactIdentifier}.location`，您可以在 IAM 政策中指定次要[來源](#)或次要[成品](#)識別符。以下政策允許發起人建立具有次要來源之特定來源位置的專案 `mySecondSource`。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "codebuild:secondarySources/mySecondSource.location": "my-source-location"
        }
      }
    }
  ]
}

```

檢查 API 請求中是否存在屬性

CodeBuild 支援條件索引鍵，以檢查 API 請求中是否存在某些欄位。政策會在建立或更新專案時強制執行 VPC 要求。

```

{
  "Version": "2012-10-17",

```

```
    "Statement": [{
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild:UpdateProject"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "codebuild:vpcConfig.vpcId": "false"
        }
      }
    }]
  }
}
```

CodeBuild AWS SDKs的程式碼範例

下列程式碼範例示範如何使用 CodeBuild 搭配 AWS 軟體開發套件 (SDK)。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

程式碼範例

- [CodeBuild AWS SDKs的基本範例](#)
 - [CodeBuild using AWS SDKs 的動作](#)
 - [CreateProject 搭配 AWS SDK 或 CLI 使用](#)
 - [ListBuilds 搭配 AWS SDK 或 CLI 使用](#)
 - [ListProjects 搭配 AWS SDK 或 CLI 使用](#)
 - [StartBuild 搭配 AWS SDK 或 CLI 使用](#)

CodeBuild AWS SDKs的基本範例

下列程式碼範例示範如何 AWS CodeBuild 搭配 AWS SDKs 使用 的基本概念。

範例

- [CodeBuild using AWS SDKs 的動作](#)
 - [CreateProject 搭配 AWS SDK 或 CLI 使用](#)
 - [ListBuilds 搭配 AWS SDK 或 CLI 使用](#)
 - [ListProjects 搭配 AWS SDK 或 CLI 使用](#)
 - [StartBuild 搭配 AWS SDK 或 CLI 使用](#)

CodeBuild using AWS SDKs 的動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 CodeBuild 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

下列範例僅包含最常使用的動作。如需完整清單，請參閱 [《AWS CodeBuild API 參考》](#)。

範例

- [CreateProject 搭配 AWS SDK 或 CLI 使用](#)
- [ListBuilds 搭配 AWS SDK 或 CLI 使用](#)
- [ListProjects 搭配 AWS SDK 或 CLI 使用](#)
- [StartBuild 搭配 AWS SDK 或 CLI 使用](#)

CreateProject 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateProject。

CLI

AWS CLI

範例 1：建立 AWS CodeBuild 組建專案

下列 create-project 範例使用來自 S3 儲存貯體的來源檔案，建立 CodeBuild 組建專案

```
aws codebuild create-project \
  --name "my-demo-project" \
  --source '{"type": "S3","location": "codebuild-us-west-2-123456789012-input-bucket/my-source.zip"}' \
  --artifacts '{"type": "S3","location": "codebuild-us-west-2-123456789012-output-bucket"}' \
  --environment '{"type": "LINUX_CONTAINER","image": "aws/codebuild/standard:1.0","computeType": "BUILD_GENERAL1_SMALL"}' \
  --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-service-role"
```

輸出：

```
{
  "project": {
    "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-project",
    "name": "my-cli-demo-project",
    "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-service-role",
    "lastModified": 1556839783.274,
    "badge": {
```



```
        "badgeEnabled": false
    },
    "queuedTimeoutInMinutes": 480,
    "environment": {
        "image": "aws/codebuild/standard:1.0",
        "computeType": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "imagePullCredentialsType": "CODEBUILD",
        "privilegedMode": false,
        "environmentVariables": []
    },
    "artifacts": {
        "location": "codebuild-us-west-2-123456789012-output-bucket",
        "name": "my-cli-demo-project",
        "namespaceType": "NONE",
        "type": "S3",
        "packaging": "NONE",
        "encryptionDisabled": false
    },
    "source": {
        "type": "S3",
        "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip",
        "insecureSsl": false
    },
    "timeoutInMinutes": 60,
    "cache": {
        "type": "NO_CACHE"
    },
    "created": 1556839783.274
}
}
```

範例 2：使用參數的 JSON 輸入檔案建立 AWS CodeBuild 組建專案

下列 create-project 範例透過在 JSON 輸入檔案中傳遞所有必要參數，建立 CodeBuild 組建專案。僅使用 --generate-cli-skeleton parameter 執行命令，藉此建立輸入檔案範本。

```
aws codebuild create-project --cli-input-json file://create-project.json
```

輸入 JSON create-project.json 檔案包含下列內容：

```
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:1.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```

輸出：

```
{
  "project": {
    "name": "codebuild-demo-project",
    "serviceRole": "serviceIAMRole",
    "tags": [],
    "artifacts": {
      "packaging": "NONE",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "message-util.zip"
    },
    "lastModified": 1472661575.244,
    "timeoutInMinutes": 60,
    "created": 1472661575.244,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:1.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
```

```
        "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
      },
      "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
      "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project"
    }
  }
}
```

如需詳細資訊，請參閱 AWS CodeBuild 使用者指南中的[建立建置專案 \(AWS CLI\)](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[CreateProject](#)。

JavaScript

適用於 JavaScript (v3) 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

建立專案。

```
import {
  ArtifactsType,
  CodeBuildClient,
  ComputeType,
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";

// Create the AWS CodeBuild project.
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam:xxxxxxxxxxxx:role/CodeBuildAdmin",
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
) => {
  const codeBuildClient = new CodeBuildClient({});
```

```
const response = await codeBuildClient.send(
  new CreateProjectCommand({
    artifacts: {
      // The destination of the build artifacts.
      type: ArtifactsType.S3,
      location: buildOutputBucket,
    },
    // Information about the build environment. The combination of
    "computeType" and "type" determines the
    // requirements for the environment such as CPU, memory, and disk space.
    environment: {
      // Build environment compute types.
      // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-compute-types.html
      computeType: ComputeType.BUILD_GENERAL1_SMALL,
      // Docker image identifier.
      // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-available.html
      image: "aws/codebuild/standard:7.0",
      // Build environment type.
      type: EnvironmentType.LINUX_CONTAINER,
    },
    name: projectName,
    // A role ARN with permission to create a CodeBuild project, write to the
    artifact location, and write CloudWatch logs.
    serviceRole: roleArn,
    source: {
      // The type of repository that contains the source code to be built.
      type: SourceType.GITHUB,
      // The location of the repository that contains the source code to be
      built.
      location: githubUrl,
    },
  }),
);
console.log(response);
// {
//   '$metadata': {
//     httpStatusCode: 200,
//     requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
//     extendedRequestId: undefined,
//     cfId: undefined,
//     attempts: 1,
```

```
//      totalRetryDelay: 0
//    },
//    project: {
//      arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxxxxx:project/MyCodeBuilder',
//      artifacts: {
//        encryptionDisabled: false,
//        location: 'xxxxxx-xxxxxx-xxxxxx',
//        name: 'MyCodeBuilder',
//        namespaceType: 'NONE',
//        packaging: 'NONE',
//        type: 'S3'
//      },
//      badge: { badgeEnabled: false },
//      cache: { type: 'NO_CACHE' },
//      created: 2023-08-18T14:46:48.979Z,
//      encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxxxx:alias/aws/s3',
//      environment: {
//        computeType: 'BUILD_GENERAL1_SMALL',
//        environmentVariables: [],
//        image: 'aws/codebuild/standard:7.0',
//        imagePullCredentialsType: 'CODEBUILD',
//        privilegedMode: false,
//        type: 'LINUX_CONTAINER'
//      },
//      lastModified: 2023-08-18T14:46:48.979Z,
//      name: 'MyCodeBuilder',
//      projectVisibility: 'PRIVATE',
//      queuedTimeoutInMinutes: 480,
//      serviceRole: 'arn:aws:iam:xxxxxxxxxxxx:role/CodeBuildAdmin',
//      source: {
//        insecureSsl: false,
//        location: 'https://...',
//        reportBuildStatus: false,
//        type: 'GITHUB'
//      },
//      timeoutInMinutes: 60
//    }
//  }
return response;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。

- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [CreateProject](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

ListBuilds 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListBuilds。

C++

SDK for C++

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
#!/ List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }
    }
```

```
    Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
codeBuildClient.ListBuilds(
    listBuildsRequest);

    if (listBuildsOutcome.IsSuccess()) {
        const Aws::Vector<Aws::String> &ids =
listBuildsOutcome.GetResult().GetIds();
        if (!ids.empty()) {

            std::cout << "Information about each build:" << std::endl;
            Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
            getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
            Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
                getBuildsRequest);

            if (getBuildsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
                std::cout << builds.size() << " build(s) found." <<
std::endl;

                for (auto val: builds) {
                    std::cout << val.GetId() << std::endl;
                }
            } else {
                std::cerr << "Error getting builds"
                    << getBuildsOutcome.GetError().GetMessage() <<
std::endl;

                return false;
            }
        } else {
            std::cout << "No builds found." << std::endl;
        }

        // Get the next token for pagination.

        nextToken = listBuildsOutcome.GetResult().GetNextToken();
    } else {
        std::cerr << "Error listing builds"
            << listBuildsOutcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}
```

```
    } while (!nextToken.  
        empty()  
        );  
    return true;  
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [ListBuilds](#)。

CLI

AWS CLI

若要取得 AWS CodeBuild 組建 IDs 的清單。

下列 `list-builds` 範例會取得依遞增順序排序的 CodeBuild ID 清單。

```
aws codebuild list-builds --sort-order ASCENDING
```

輸出包含 `nextToken` 值，指出有更多輸出可用。

```
{  
  "nextToken": "4AEA6u7J...The full token has been omitted for  
brevity...MzY2OA==",  
  "ids": [  
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"  
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"  
    ... The full list of build IDs has been omitted for brevity ...  
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"  
  ]  
}
```

再次執行此命令，並提供上一個回應中的 `nextToken` 值做為參數，以取得下一個輸出部分。重複此動作，直到您未在回應中收到 `nextToken` 值為止。

```
aws codebuild list-builds --sort-order ASCENDING --next-  
token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```


下一個輸出部分：

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

如需詳細資訊，請參閱 AWS CodeBuild 使用者指南中的[檢視組建 IDs 清單 \(AWS CLI\)](#)

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[ListBuilds](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

ListProjects 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListProjects。

C++

SDK for C++

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType
    sortType,
                                     const Aws::Client::ClientConfiguration
    &clientConfiguration) {
```

```
Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
listProjectsRequest.SetSortOrder(sortType);

Aws::String nextToken; // Next token for pagination.
Aws::Vector<Aws::String> allProjects;

do {
    if (!nextToken.empty()) {
        listProjectsRequest.SetNextToken(nextToken);
    }

    Aws::CodeBuild::Model::ListProjectsOutcome outcome =
codeBuildClient.ListProjects(
    listProjectsRequest);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::String> &projects =
outcome.GetResult().GetProjects();
        allProjects.insert(allProjects.end(), projects.begin(),
projects.end());
        nextToken = outcome.GetResult().GetNextToken();
    }

    else {
        std::cerr << "Error listing projects" <<
outcome.GetError().GetMessage()
        << std::endl;
    }

} while (!nextToken.empty());

std::cout << allProjects.size() << " project(s) found." << std::endl;
for (auto project: allProjects) {
    std::cout << project << std::endl;
}

return true;
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [ListProjects](#)。

CLI

AWS CLI

取得 AWS CodeBuild 組建專案名稱的清單。

下列 `list-projects` 範例會取得依名稱遞增順序排序的 CodeBuild 組建專案清單。

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

輸出包含 `nextToken` 值，指出有更多輸出可用。

```
{
  "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=",
  "projects": [
    "codebuild-demo-project",
    "codebuild-demo-project2",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project99"
  ]
}
```

再次執行此命令，並提供上一個回應的 `nextToken` 值做為參數，以取得下一個輸出部分。重複此動作，直到您未在回應中收到 `nextToken` 值為止。

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING --next-token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=

{
  "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",

    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
  ]
}
```

如需詳細資訊，請參閱 AWS CodeBuild 使用者指南中的[檢視組建專案名稱清單 \(AWS CLI\)](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ListProjects](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

StartBuild 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 StartBuild。

C++

SDK for C++

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
#!/ Start an AWS CodeBuild project build.
/*!
    \param projectName: A CodeBuild project name.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome =
codeBuildClient.StartBuild(
    startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
<< std::endl;
    }
}
```

```
    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [StartBuild](#)。

CLI

AWS CLI

開始執行建置 an AWS CodeBuild 組建專案。

下列 start-build 範例啟動指定的 CodeBuild 專案的組建。組建會覆寫專案的設定，以允許組建在逾時前排入佇列的分鐘數，以及專案的成品設定。

```
aws codebuild start-build \
  --project-name "my-demo-project" \
  --queued-timeout-in-minutes-override 5 \
  --artifacts-override {"\"type\": \"S3\", \"location\":  
  \"arn:aws:s3:::artifacts-override\", \"overrideArtifactName\": true"}
```

輸出：

```
{
  "build": {
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-  
service-role",
    "buildStatus": "IN_PROGRESS",
    "buildComplete": false,
    "projectName": "my-demo-project",
    "timeoutInMinutes": 60,
    "source": {
      "insecureSsl": false,
      "type": "S3",
      "location": "codebuild-us-west-2-123456789012-input-bucket/my-  
source.zip"
    }
  }
}
```

```

    },
    "queuedTimeoutInMinutes": 5,
    "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
    "currentPhase": "QUEUED",
    "startTime": 1556905683.568,
    "environment": {
        "computeType": "BUILD_GENERAL1_MEDIUM",
        "environmentVariables": [],
        "type": "LINUX_CONTAINER",
        "privilegedMode": false,
        "image": "aws/codebuild/standard:1.0",
        "imagePullCredentialsType": "CODEBUILD"
    },
    "phases": [
        {
            "phaseStatus": "SUCCEEDED",
            "startTime": 1556905683.568,
            "phaseType": "SUBMITTED",
            "durationInSeconds": 0,
            "endTime": 1556905684.524
        },
        {
            "startTime": 1556905684.524,
            "phaseType": "QUEUED"
        }
    ],
    "logs": {
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-west-2#logEvent:group=null;stream=null"
    },
    "artifacts": {
        "encryptionDisabled": false,
        "location": "arn:aws:s3:::artifacts-override/my-demo-project",
        "overrideArtifactName": true
    },
    "cache": {
        "type": "NO_CACHE"
    },
    "id": "my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE",
    "initiator": "my-aws-account-name",
    "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE"
}

```

```
}
```

如需詳細資訊，請參閱 AWS CodeBuild 使用者指南中的[執行組建 \(AWS CLI\)](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [StartBuild](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

公有建置 API

主控台 AWS CodeBuild 和公有建置結果網站會使用公有建置 API，以取得公有建置專案的相關資訊。如需詳細資訊，請參閱[the section called “取得公有建置專案 URLs”](#)。

Note

CodeBuild 公有建置 API 不包含在 AWS CLI AWS SDKs 中。

主題

- [公有建置動作](#)
- [公有建置資料類型](#)

公有建置動作

Note

CodeBuild 公有建置 API 不包含在 AWS CLI AWS SDKs 中。

CodeBuild 公有建置 API 支援下列動作：

動作

- [DescribeBuildBatchesForPublicProject](#)
- [DescribeBuildsForPublicProject](#)
- [GetCloudWatchLogsForPublicBuild](#)
- [GetPresignedUrlsForPublicBuild](#)
- [GetPublicBuild](#)
- [GetPublicBuildBatch](#)
- [GetPublicProject](#)

DescribeBuildBatchesForPublicProject

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

請求語法

```
{
  "filter": {
    "status": "string"
  },
  "maxResults": number,
  "nextToken": "string",
  "publicProjectAlias": "string",
  "sortOrder": "string"
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《[Common Parameters](#)》。

請求接受採用 JSON 格式的下列資料。

Note

在下列清單中，會先描述必要的參數。

publicProjectAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+$`

必要：是

篩選條件

擷取批次建置時指定篩選條件。

類型：[BuildBatchFilter](#) 物件

必要：否

maxResults

類型：整數

有效範圍：最小值為 1。最大值為 100。

必要：否

nextToken

類型：字串

必要：否

sortOrder

類型：字串

有效值：ASCENDING | DESCENDING

必要：否

回應語法

```
{
  "nextToken": "string",
  "publicBuildBatches": [
    {
      "buildBatchNumber": number,
      "buildBatchStatus": "string",
      "endTime": number,
      "publicBuildBatchAlias": "string",
      "sourceVersion": "string",
      "startTime": number
    }
  ]
}
```

```
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

[nextToken](#)

類型：字串

[publicBuildBatches](#)

類型：[BuildBatchForDescribeBuildBatchesPublic](#) 物件陣列

錯誤

如需所有動作常見錯誤的相關資訊，請參閱[常見錯誤](#)。

InvalidInputException

HTTP 狀態碼：400

ResourceNotFoundException

HTTP 狀態碼：400

DescribeBuildsForPublicProject

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

請求語法

```
{
  "maxResults": number,
  "nextToken": "string",
  "publicProjectAlias": "string",
  "sortOrder": "string"
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《[Common Parameters](#)》。

請求接受採用 JSON 格式的下列資料。

Note

在下列清單中，會先描述必要的參數。

publicProjectAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+$`

必要：是

maxResults

類型：整數

有效範圍：最小值為 1。最大值為 100。

必要：否

[nextToken](#)

類型：字串

必要：否

[sortOrder](#)

類型：字串

有效值：ASCENDING | DESCENDING

必要：否

回應語法

```
{
  "builds": [
    {
      "artifacts": {
        "artifactIdentifier": "string",
        "packaging": "string",
        "type": "string"
      },
      "buildComplete": boolean,
      "buildNumber": number,
      "buildStatus": "string",
      "endTime": number,
      "environment": {
        "computeType": "string",
        "environmentVariables": [
          {
            "name": "string",
            "type": "string",
            "value": "string"
          }
        ],
        "image": "string",
        "type": "string"
      },
      "id": "string",
      "initiator": "string",
```

```
"logsStatus": {
  "cloudWatchLogsStatus": "string",
  "s3LogsStatus": "string"
},
"phases": [
  {
    "contexts": [
      {
        "message": "string",
        "statusCode": "string"
      }
    ],
    "durationInSeconds": number,
    "endTime": number,
    "phaseStatus": "string",
    "phaseType": "string",
    "startTime": number
  }
],
"projectName": "string",
"queuedTimeoutInMinutes": number,
"resolvedSourceVersion": "string",
"secondaryArtifacts": [
  {
    "artifactIdentifier": "string",
    "packaging": "string",
    "type": "string"
  }
],
"secondarySources": [
  {
    "buildspec": "string",
    "gitCloneDepth": number,
    "gitSubmodulesConfig": {
      "fetchSubmodules": boolean
    },
    "location": "string",
    "sourceIdentifier": "string",
    "type": "string"
  }
],
"secondarySourceVersions": [
  {
    "sourceIdentifier": "string",
```

```
        "sourceVersion": "string"
      },
    ],
    "source": {
      "buildspec": "string",
      "gitCloneDepth": number,
      "gitSubmodulesConfig": {
        "fetchSubmodules": boolean
      },
      "location": "string",
      "sourceIdentifier": "string",
      "type": "string"
    },
    "sourceVersion": "string",
    "startTime": number,
    "timeoutInMinutes": number
  }
],
"nextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

組建

類型：[PublicBuild](#) 物件陣列

nextToken

類型：字串

錯誤

如需所有動作常見錯誤的相關資訊，請參閱[常見錯誤](#)。

InvalidInputException

HTTP 狀態碼：400

ResourceNotFoundException

HTTP 狀態碼 : 400

GetCloudWatchLogsForPublicBuild

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

請求語法

```
{
  "maxResult": number,
  "nextToken": "string",
  "publicBuildAlias": "string"
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《[Common Parameters](#)》。

請求接受採用 JSON 格式的下列資料。

Note

在下列清單中，會先描述必要的參數。

publicBuildAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+:\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

必要：是

maxResult

類型：整數

有效範圍：最小值為 1。最大值為 10000。

必要：否

[nextToken](#)

類型：字串

必要：否

回應語法

```
{
  "logs": [ "string" ],
  "nextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

[日誌](#)

類型：字串陣列

[nextToken](#)

類型：字串

長度限制：長度下限為 1。

錯誤

如需所有動作常見錯誤的相關資訊，請參閱[常見錯誤](#)。

InvalidInputException

HTTP 狀態碼：400

ResourceNotFoundException

HTTP 狀態碼：400

GetPresignedUrlsForPublicBuild

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

請求語法

```
{  
  "publicBuildAlias": "string"  
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《[Common Parameters](#)》。

請求接受採用 JSON 格式的下列資料。

Note

在下列清單中，會先描述必要的參數。

publicBuildAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

必要：是

回應語法

```
{  
  "artifacts": {
```

```
    "expiredAt": number,
    "identifier": "string",
    "md5Checksum": "string",
    "presignedUrl": "string",
    "s3Arn": "string",
    "sha256Checksum": "string"
  },
  "log": {
    "expiredAt": number,
    "identifier": "string",
    "md5Checksum": "string",
    "presignedUrl": "string",
    "s3Arn": "string",
    "sha256Checksum": "string"
  },
  "secondaryArtifacts": [
    {
      "expiredAt": number,
      "identifier": "string",
      "md5Checksum": "string",
      "presignedUrl": "string",
      "s3Arn": "string",
      "sha256Checksum": "string"
    }
  ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

成品

類型：[S3Downloadable](#) 物件

日誌

類型：[S3Downloadable](#) 物件

secondaryArtifacts

類型：[S3Downloadable](#) 物件陣列

錯誤

如需所有動作常見錯誤的相關資訊，請參閱[常見錯誤](#)。

InvalidInputException

HTTP 狀態碼：400

ResourceNotFoundException

HTTP 狀態碼：400

GetPublicBuild

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

請求語法

```
{
  "publicBuildAlias": "string"
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《[Common Parameters](#)》。

請求接受採用 JSON 格式的下列資料。

Note

在下列清單中，會先描述必要的參數。

publicBuildAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+:\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

必要：是

回應語法

```
{
  "build": {
    "artifacts": {
      "artifactIdentifier": "string",
      "packaging": "string",
```

```
    "type": "string"
  },
  "buildComplete": boolean,
  "buildNumber": number,
  "buildStatus": "string",
  "endTime": number,
  "environment": {
    "computeType": "string",
    "environmentVariables": [
      {
        "name": "string",
        "type": "string",
        "value": "string"
      }
    ],
    "image": "string",
    "type": "string"
  },
  "id": "string",
  "initiator": "string",
  "logsStatus": {
    "cloudWatchLogsStatus": "string",
    "s3LogsStatus": "string"
  },
  "phases": [
    {
      "contexts": [
        {
          "message": "string",
          "statusCode": "string"
        }
      ],
      "durationInSeconds": number,
      "endTime": number,
      "phaseStatus": "string",
      "phaseType": "string",
      "startTime": number
    }
  ],
  "projectName": "string",
  "queuedTimeoutInMinutes": number,
  "resolvedSourceVersion": "string",
  "secondaryArtifacts": [
    {
```



```

        "artifactIdentifier": "string",
        "packaging": "string",
        "type": "string"
    }
],
"secondarySources": [
    {
        "buildspec": "string",
        "gitCloneDepth": number,
        "gitSubmodulesConfig": {
            "fetchSubmodules": boolean
        },
        "location": "string",
        "sourceIdentifier": "string",
        "type": "string"
    }
],
"secondarySourceVersions": [
    {
        "sourceIdentifier": "string",
        "sourceVersion": "string"
    }
],
"source": {
    "buildspec": "string",
    "gitCloneDepth": number,
    "gitSubmodulesConfig": {
        "fetchSubmodules": boolean
    },
    "location": "string",
    "sourceIdentifier": "string",
    "type": "string"
},
"sourceVersion": "string",
"startTime": number,
"timeoutInMinutes": number
}
}

```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

組建

類型：[PublicBuild](#) 物件

錯誤

如需所有動作常見錯誤的相關資訊，請參閱[常見錯誤](#)。

InvalidInputException

HTTP 狀態碼：400

ResourceNotFoundException

HTTP 狀態碼：400

GetPublicBuildBatch

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

請求語法

```
{  
  "publicBuildBatchAlias": "string"  
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《[Common Parameters](#)》。

請求接受採用 JSON 格式的下列資料。

Note

在下列清單中，會先描述必要的參數。

publicBuildBatchAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+:[a-z0-9]{8}(-[a-z0-9]{4}){3}-[a-z0-9]{12}$`

必要：是

回應語法

```
{  
  "publicBuildBatch": {  
    "artifacts": {  
      "artifactIdentifier": "string",  
      "packaging": "string",
```

```
    "type": "string"
  },
  "buildBatchConfig": {
    "combineArtifacts": boolean,
    "restrictions": {
      "computeTypesAllowed": [ "string" ],
      "maximumBuildsAllowed": number
    },
    "timeoutInMins": number
  },
  "buildBatchNumber": number,
  "buildBatchStatus": "string",
  "buildGroups": [
    {
      "currentBuildSummary": {
        "buildStatus": "string",
        "primaryArtifact": {
          "identifier": "string",
          "location": "string",
          "type": "string"
        },
        "publicBuildAlias": "string",
        "requestedOn": number,
        "secondaryArtifacts": [
          {
            "identifier": "string",
            "location": "string",
            "type": "string"
          }
        ]
      },
      "dependsOn": [ "string" ],
      "identifier": "string",
      "ignoreFailure": boolean,
      "priorBuildSummaryList": [
        {
          "buildStatus": "string",
          "primaryArtifact": {
            "identifier": "string",
            "location": "string",
            "type": "string"
          },
          "publicBuildAlias": "string",
          "requestedOn": number,
```

```

        "secondaryArtifacts": [
            {
                "identifier": "string",
                "location": "string",
                "type": "string"
            }
        ]
    },
    ],
    "buildTimeoutInMinutes": number,
    "complete": boolean,
    "currentPhase": "string",
    "endTime": number,
    "environment": {
        "computeType": "string",
        "environmentVariables": [
            {
                "name": "string",
                "type": "string",
                "value": "string"
            }
        ],
        "image": "string",
        "type": "string"
    },
    "id": "string",
    "initiator": "string",
    "logsStatus": {
        "cloudWatchLogsStatus": "string",
        "s3LogsStatus": "string"
    },
    "phases": [
        {
            "contexts": [
                {
                    "message": "string",
                    "statusCode": "string"
                }
            ],
            "durationInSeconds": number,
            "endTime": number,
            "phaseStatus": "string",

```

```

        "phaseType": "string",
        "startTime": number
    }
],
"projectName": "string",
"publicBuildBatchAlias": "string",
"queuedTimeoutInMinutes": number,
"resolvedSourceVersion": "string",
"secondaryArtifacts": [
    {
        "artifactIdentifier": "string",
        "packaging": "string",
        "type": "string"
    }
],
"secondarySources": [
    {
        "buildspec": "string",
        "gitCloneDepth": number,
        "gitSubmodulesConfig": {
            "fetchSubmodules": boolean
        },
        "location": "string",
        "sourceIdentifier": "string",
        "type": "string"
    }
],
"secondarySourceVersions": [
    {
        "sourceIdentifier": "string",
        "sourceVersion": "string"
    }
],
"source": {
    "buildspec": "string",
    "gitCloneDepth": number,
    "gitSubmodulesConfig": {
        "fetchSubmodules": boolean
    },
    "location": "string",
    "sourceIdentifier": "string",
    "type": "string"
},
"sourceVersion": "string",

```

```
    "startTime": number
  }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

[publicBuildBatch](#)

類型：[PublicBuildBatch](#) 物件

錯誤

如需所有動作常見錯誤的相關資訊，請參閱[常見錯誤](#)。

InvalidInputException

HTTP 狀態碼：400

ResourceNotFoundException

HTTP 狀態碼：400

GetPublicProject

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

請求語法

```
{  
  "publicProjectAlias": "string"  
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《[Common Parameters](#)》。

請求接受採用 JSON 格式的下列資料。

Note

在下列清單中，會先描述必要的參數。

publicProjectAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+$`

必要：是

回應語法

```
{  
  "project": {  
    "artifacts": {  
      "artifactIdentifier": "string",  
      "location": "string",  
    }  
  }  
}
```



```
    "type": "string"
  },
  "buildBatchConfig": {
    "combineArtifacts": boolean,
    "restrictions": {
      "computeTypesAllowed": [ "string" ],
      "maximumBuildsAllowed": number
    },
    "timeoutInMins": number
  },
  "concurrentBuildLimit": number,
  "description": "string",
  "environment": {
    "computeType": "string",
    "environmentVariables": [
      {
        "name": "string",
        "type": "string",
        "value": "string"
      }
    ],
    "image": "string",
    "type": "string"
  },
  "name": "string",
  "queuedTimeoutInMinutes": number,
  "secondaryArtifacts": [
    {
      "artifactIdentifier": "string",
      "location": "string",
      "type": "string"
    }
  ],
  "secondarySources": [
    {
      "buildspec": "string",
      "gitCloneDepth": number,
      "gitSubmodulesConfig": {
        "fetchSubmodules": boolean
      },
      "location": "string",
      "sourceIdentifier": "string",
      "type": "string"
    }
  ]
}
```

```
    ],
    "secondarySourceVersions": [
      {
        "sourceIdentifier": "string",
        "sourceVersion": "string"
      }
    ],
    "source": {
      "buildspec": "string",
      "gitCloneDepth": number,
      "gitSubmodulesConfig": {
        "fetchSubmodules": boolean
      },
      "location": "string",
      "sourceIdentifier": "string",
      "type": "string"
    },
    "sourceVersion": "string",
    "timeoutInMinutes": number,
    "webhook": {
      "branchFilter": "string",
      "buildType": "string",
      "filterGroups": [
        [
          {
            "excludeMatchedPattern": boolean,
            "pattern": "string",
            "type": "string"
          }
        ]
      ],
      "payloadUrl": "string",
      "url": "string"
    }
  }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

專案

類型：[PublicProject](#) 物件

錯誤

如需所有動作常見錯誤的相關資訊，請參閱[常見錯誤](#)。

InvalidInputException

HTTP 狀態碼：400

ResourceNotFoundException

HTTP 狀態碼：400

公有建置資料類型

Note

CodeBuild 公有建置 API 不包含在 AWS CLI AWS SDKs 中。

CodeBuild 公有建置 API 支援下列資料類型：

資料類型

- [BuildBatchForDescribeBuildBatchesPublic](#)
- [PublicBuild](#)
- [PublicBuildArtifacts](#)
- [PublicBuildBatch](#)
- [PublicBuildGroup](#)
- [PublicBuildSummary](#)
- [PublicLogsStatus](#)
- [PublicProject](#)
- [PublicProjectArtifacts](#)
- [PublicProjectBuildBatchConfig](#)

- [PublicProjectEnvironment](#)
- [PublicProjectSource](#)
- [PublicWebhook](#)
- [S3Downloadable](#)

BuildBatchForDescribeBuildBatchesPublic

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

buildBatchNumber

類型：Long

必要：否

buildBatchStatus

類型：字串

有效值： PENDING | SUCCEEDED | FAILED | FAULT | TIMED_OUT | IN_PROGRESS | STOPPED

必要：否

endTime

類型：Timestamp

必要：否

publicBuildBatchAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+:[a-z0-9]{8}(-[a-z0-9]{4}){3}-[a-z0-9]{12}$`

必要：否

sourceVersion

類型：字串

長度限制：長度下限為 1。

必要：否

startTime

類型：Timestamp

必要：否

PublicBuild

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

成品

類型：[PublicBuildArtifacts](#) 物件

必要：否

buildComplete

類型：布林值

必要：否

buildNumber

類型：Long

必要：否

buildStatus

類型：字串

有效值： PENDING | SUCCEEDED | FAILED | FAULT | TIMED_OUT | IN_PROGRESS | STOPPED

必要：否

endTime

類型：Timestamp

必要：否

環境

類型：[PublicProjectEnvironment](#) 物件

必要：否

id

類型：字串

長度限制：長度下限為 1。

必要：否

啟動器

類型：字串

必要：否

logsStatus

類型：[PublicLogsStatus](#) 物件

必要：否

階段

類型：[BuildPhase](#) 物件的陣列

必要：否

projectName

類型：字串

長度限制：長度下限為 1。

必要：否

queuedTimeoutInMinutes

類型：整數

必要：否

resolvedSourceVersion

類型：字串

長度限制：長度下限為 1。

必要：否

secondaryArtifacts

類型：[PublicBuildArtifacts](#) 物件陣列

必要：否

secondarySources

類型：[PublicProjectSource](#) 物件陣列

陣列成員：項目數下限為 0。最多 12 個項目。

必要：否

secondarySourceVersions

類型：[ProjectSourceVersion](#) 物件陣列

陣列成員：項目數下限為 0。最多 12 個項目。

必要：否

source

類型：[PublicProjectSource](#) 物件

必要：否

sourceVersion

類型：字串

長度限制：長度下限為 1。

必要：否

startTime

類型：Timestamp

必要 : 否

timeoutInMinutes

類型 : 整數

必要 : 否

PublicBuildArtifacts

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

artifactIdentifier

類型：字串

必要：否

封裝

類型：字串

必要：否

type

類型：字串

有效值： CODEPIPELINE | S3 | NO_ARTIFACTS | DOCKER_IMAGE

必要：否

PublicBuildBatch

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

成品

類型：[PublicBuildArtifacts](#) 物件

必要：否

buildBatchConfig

類型：[PublicProjectBuildBatchConfig](#) 物件

必要：否

buildBatchNumber

類型：Long

必要：否

buildBatchStatus

類型：字串

有效值： PENDING | SUCCEEDED | FAILED | FAULT | TIMED_OUT | IN_PROGRESS | STOPPED

必要：否

buildGroups

類型：[PublicBuildGroup](#) 物件陣列

必要：否

buildTimeoutInMinutes

類型：整數

必要：否

完成

類型：布林值

必要：否

currentPhase

類型：字串

必要：否

endTime

類型：Timestamp

必要：否

環境

類型：[PublicProjectEnvironment](#) 物件

必要：否

id

類型：字串

長度限制：長度下限為 1。

必要：否

啟動器

類型：字串

必要：否

logsStatus

類型：[PublicLogsStatus](#) 物件

必要：否

階段

類型：[BuildBatchPhase](#) 物件的陣列

必要：否

projectName

類型：字串

長度限制：長度下限為 1。

必要：否

publicBuildBatchAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式： $^[\text{0-9a-zA-Z\%+=}]+:[\text{a-z0-9}]\{8\}(-[\text{a-z0-9}]\{4\})\{3\}-[\text{a-z0-9}]\{12\}\$$

必要：否

queuedTimeoutInMinutes

類型：整數

必要：否

resolvedSourceVersion

類型：字串

長度限制：長度下限為 1。

必要：否

secondaryArtifacts

類型：[PublicBuildArtifacts](#) 物件陣列

必要：否

secondarySources

類型：[PublicProjectSource](#) 物件陣列

陣列成員：項目數下限為 0。最多 12 個項目。

必要：否

secondarySourceVersions

類型：[ProjectSourceVersion](#) 物件陣列

陣列成員：項目數下限為 0。最多 12 個項目。

必要：否

source

類型：[PublicProjectSource](#) 物件

必要：否

sourceVersion

類型：字串

長度限制：長度下限為 1。

必要：否

startTime

類型：Timestamp

必要：否

PublicBuildGroup

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

currentBuildSummary

類型：[PublicBuildSummary](#) 物件

必要：否

dependsOn

類型：字串陣列

長度限制：長度下限為 1。

必要：否

識別符

類型：字串

必要：否

ignoreFailure

類型：布林值

必要：否

priorBuildSummaryList

類型：[PublicBuildSummary](#) 物件陣列

必要：否

PublicBuildSummary

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

buildStatus

類型：字串

必要：否

primaryArtifact

代表已解析的建置成品。已解決的成品是建置並部署到目的地的成品，例如 Amazon S3。

類型：[ResolvedArtifact](#) 物件

必要：否

publicBuildAlias

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`^[0-9a-zA-Z%+=]+:\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

必要：否

requestedOn

類型：Timestamp

必要：否

secondaryArtifacts

類型：[ResolvedArtifact](#) 物件的陣列

必要：否

PublicLogsStatus

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

cloudWatchLogsStatus

類型：字串

有效值： ENABLED | DISABLED

必要：否

s3LogsStatus

類型：字串

有效值： ENABLED | DISABLED

必要：否

PublicProject

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

成品

類型：[PublicProjectArtifacts](#) 物件

必要：否

buildBatchConfig

類型：[PublicProjectBuildBatchConfig](#) 物件

必要：否

concurrentBuildLimit

類型：整數

必要：否

description

類型：字串

長度限制：長度下限為 0。長度上限為 255。

必要：否

環境

類型：[PublicProjectEnvironment](#) 物件

必要：否

name

類型：字串

長度限制：長度下限為 2。長度上限為 150。

模式：`[A-Za-z0-9][A-Za-z0-9\-\_]{1,254}`

必要：否

queuedTimeoutInMinutes

類型：整數

有效範圍：最小值為 5。最大值為 480。

必要：否

secondaryArtifacts

類型：[PublicProjectArtifacts](#) 物件陣列

陣列成員：項目數下限為 0。最多 12 個項目。

必要：否

secondarySources

類型：[PublicProjectSource](#) 物件陣列

陣列成員：項目數下限為 0。最多 12 個項目。

必要：否

secondarySourceVersions

類型：[ProjectSourceVersion](#) 物件陣列

陣列成員：項目數下限為 0。最多 12 個項目。

必要：否

source

類型：[PublicProjectSource](#) 物件

必要：否

sourceVersion

類型：字串

長度限制：長度下限為 1。

必要：否

timeoutInMinutes

類型：整數

有效範圍：最小值為 5。最大值為 480。

必要：否

Webhook

類型：[PublicWebhook](#) 物件

必要：否

PublicProjectArtifacts

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

type

類型：字串

有效值： CODEPIPELINE | S3 | NO_ARTIFACTS | DOCKER_IMAGE

必要：是

artifactIdentifier

類型：字串

必要：否

位置

類型：字串

必要：否

PublicProjectBuildBatchConfig

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

combineArtifacts

類型：布林值

必要：否

限制

指定批次建置的限制。

類型：[BatchRestrictions](#) 物件

必要：否

timeoutInMins

類型：整數

必要：否

PublicProjectEnvironment

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

computeType

類型：字串

有效值： BUILD_GENERAL1_SMALL | BUILD_GENERAL1_MEDIUM |
BUILD_GENERAL1_LARGE | BUILD_GENERAL1_2XLARGE

必要：否

environmentVariables

類型：[EnvironmentVariable](#) 物件的陣列

必要：否

映像

類型：字串

長度限制：長度下限為 1。

必要：否

type

類型：字串

有效值： WINDOWS_CONTAINER | WINDOWS_SERVER_2019_CONTAINER |
LINUX_CONTAINER | LINUX_GPU_CONTAINER | ARM_CONTAINER | MAC

必要：否

PublicProjectSource

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

type

類型：字串

有效值： NO_SOURCE | CODECOMMIT | CODEPIPELINE | GITHUB | S3 | BITBUCKET
| GITHUB_ENTERPRISE | GITLAB | GITLAB_SELF_MANAGED

必要：是

buildspec

類型：字串

必要：否

gitCloneDepth

類型：整數

有效範圍：最小值為 0。

必要：否

gitSubmodulesConfig

AWS CodeBuild 建置專案的 Git 子模組組態相關資訊。

類型：[GitSubmodulesConfig](#) 物件

必要：否

位置

類型：字串

必要：否

sourceIdentifier

類型：字串

必要：否

PublicWebhook

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

branchFilter

類型：字串

必要：否

buildType

類型：字串

必要：否

filterGroups

類型：[WebhookFilter](#) 物件陣列

必要：否

payloadUrl

類型：字串

長度限制：長度下限為 1。

必要：否

url

類型：字串

長度限制：長度下限為 1。

必要：否

S3Downloadable

Note

此 API 元素不包含在 AWS CLI AWS SDKs 中。

目錄

Note

在下列清單中，會先描述必要的參數。

expiredAt

類型：Timestamp

必要：否

識別符

類型：字串

必要：否

md5Checksum

類型：字串

必要：否

presignedUrl

類型：字串

必要：否

s3Arn

類型：字串

必要：否

sha256Checksum

類型：字串

必要：否

故障診斷 AWS CodeBuild

使用本主題的資訊來協助您查明、診斷和解決問題。若要了解如何記錄和監控 CodeBuild 組建以疑難排解問題，請參閱 [日誌記錄和監控](#)。

主題

- [Apache Maven 建置參考錯誤儲存庫中的成品](#)
- [建置命令預設以 root 身分執行](#)
- [當檔案名稱具有非美國 時，組建可能會失敗。英文字元](#)
- [從 Amazon EC2 參數存放區取得參數時，組建可能會失敗](#)
- [無法存取 CodeBuild 主控台的分支篩選條件](#)
- [無法檢視組建成功或失敗](#)
- [未向來源提供者報告建置狀態](#)
- [找不到並選取 Windows Server Core 2019 平台的基礎映像](#)
- [在 Buildspec 檔案中後來的命令無法辨識先前的命令](#)
- [錯誤：嘗試下載快取時出現「存取遭拒」](#)
- [使用自訂建置映像時發生錯誤："BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE"](#)
- [錯誤：「完成建置之前，發現建置容器失效。因為記憶體不足，或不支援 Docker 映像，建置容器失效。ErrorCode: 500」](#)
- [執行組建時發生錯誤：「無法連接到 Docker 協助程式」](#)
- [錯誤：建立或更新組建專案時，「CodeBuild 無權執行：sts：AssumeRole」](#)
- [錯誤：「呼叫 GetBucketAcl 時發生錯誤：儲存貯體擁有者已變更，或服務角色不再有權呼叫 s3:GetBucketAcl」](#)
- [執行建置時發生錯誤：「無法上傳成品：無效的 arn」](#)
- [錯誤：「Git 複製失敗：無法存取 'your-repository-URL'：SSL 憑證問題：自我簽署憑證」](#)
- [執行建置時發生錯誤：「必須使用指定的端點來定址您嘗試存取的儲存貯體」](#)
- [錯誤：「此建置映像需要選取至少一個執行時間版本。」](#)
- [錯誤："QUEUED: INSUFFICIENT_SUBNET"，當建置佇列中的建置失敗時](#)
- [錯誤：「無法下載快取：RequestError: 傳送請求失敗，原因：x509: 無法載入系統根目錄，未提供根目錄」](#)

- [錯誤：「無法從 S3 下載憑證。AccessDenied」](#)
- [錯誤：「找不到登入資料」](#)
- [在代理伺服器中執行 CodeBuild 時發生 RequestError 逾時錯誤](#)
- [Bourne Shell \(sh\) 必須存在於建置映像中](#)
- [執行建置時出現警告：「正在略過執行時間的安裝。此建置映像不支援執行時間版本選項」](#)
- [錯誤：開啟 CodeBuild 主控台時「無法驗證 JobWorker 身分」](#)
- [組建無法啟動](#)
- [在本機快取建置中存取 GitHub 中繼資料](#)
- [AccessDenied：報告群組的儲存貯體擁有者不符合 S3 儲存貯體的擁有者...](#)
- [錯誤：使用 建立 CodeBuild 專案時，「您的登入資料缺少一或多個必要的權限範圍」 AWS CodeConnections](#)
- [錯誤：使用 Ubuntu 安裝命令建置時，「很抱歉，完全沒有終端請求 - 無法取得輸入」](#)
- [Bitbucket 組建在使用 Secrets Manager OAuth 後失敗並出現身分驗證錯誤](#)

Apache Maven 建置參考錯誤儲存庫中的成品

問題：當您搭配 AWS CodeBuild 提供的 Java 組建環境使用 Maven 時，Maven 會從安全中央 Maven 儲存庫提取組建和外掛程式相依性，網址為 <https://repo1.maven.org/maven2>。即使組建專案的 pom.xml 檔案明確宣告改用其他位置，還是會發生此情況。

可能原因：CodeBuild 提供的 Java 建置環境包含一個名為 `settings.xml` 的檔案，該檔案預先安裝在建置環境的 `/root/.m2` 目錄中。`settings.xml` 檔案包含下列宣告，指示 Maven 一律從位於 <https://repo1.maven.org/maven2> 的安全中央 Maven 儲存庫提取組建和外掛程式相依性。

```
<settings>
  <activeProfiles>
    <activeProfile>securecentral</activeProfile>
  </activeProfiles>
  <profiles>
    <profile>
      <id>securecentral</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
```

```
        <enabled>true</enabled>
      </releases>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>central</id>
      <url>https://repo1.maven.org/maven2</url>
      <releases>
        <enabled>true</enabled>
      </releases>
    </pluginRepository>
  </pluginRepositories>
</profile>
</profiles>
</settings>
```

建議解決方案：執行下列操作：

1. 將 settings.xml 檔案新增至您的原始碼。
2. 在此 settings.xml 檔案中，將先前的 settings.xml 格式當作指南，宣告您希望 Maven 改以從哪些儲存庫中提取組建和外掛程式相依性。
3. 在您建置專案的 install 階段，指示 CodeBuild 將您的 settings.xml 檔案複製到建置環境的 /root/.m2 目錄。例如，buildspec.yml 檔案中有下列程式碼片段示範此行為。

```
version 0.2

phases:
  install:
    commands:
      - cp ./settings.xml /root/.m2/settings.xml
```

建置命令預設以 root 身分執行

Issue：AWS CodeBuild 以根使用者身分執行建置命令。即使相關組建映像的 Dockerfile 將 USER 指示設為不同使用者，還是會發生此情況。

原因：根據預設，CodeBuild 會以根使用者身分執行所有建置命令。

建議解決方案：無。

當檔案名稱具有非美國 時，組建可能會失敗。 英文字元

問題：當您執行的組建使用檔案的檔案名稱包含非美國 英文字元（例如中文字元），建置失敗。

可能原因：建置 提供的環境 AWS CodeBuild，其預設地區設定設為 POSIX。POSIX地區化設定與 CodeBuild 和包含非美國 的檔案名稱較不相容。 英文字元 和 可能會導致相關組建失敗。

建議解決方案：將下列命令新增至 buildspec 檔案的 pre_build 區段。這些命令可讓建置環境使用美式英文 UTF-8 進行當地語系化設定，這更相容於 CodeBuild 和包含非美式的檔案名稱。 英文字元。

以 Ubuntu 為基礎的組建環境：

```
pre_build:
  commands:
    - export LC_ALL="en_US.UTF-8"
    - locale-gen en_US en_US.UTF-8
    - dpkg-reconfigure -f noninteractive locales
```

針對以 Amazon Linux 為基礎的建置環境：

```
pre_build:
  commands:
    - export LC_ALL="en_US.utf8"
```

從 Amazon EC2 參數存放區取得參數時，組建可能會失敗

問題：當組建嘗試取得存放在 Amazon EC2 參數存放區中一或多個參數的值時，組建會在 DOWNLOAD_SOURCE階段失敗，並顯示錯誤 Parameter does not exist。

可能原因：建置專案依賴的服務角色沒有呼叫 ssm:GetParameters動作的許可，或者建置專案使用產生的服務角色 AWS CodeBuild，並允許呼叫 ssm:GetParameters動作，但參數的名稱開頭不是 /CodeBuild/。

建議解決方案：

- 如果服務角色不是由 CodeBuild 產生，請更新其定義以允許 CodeBuild 呼叫 ssm:GetParameters動作。例如，下列政策陳述式允許呼叫 ssm:GetParameters 動作，以取得名稱開頭為 /CodeBuild/ 的參數：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/CodeBuild/*"
    }
  ]
}
```

- 如果服務角色是由 CodeBuild 產生，請更新其定義，以允許 CodeBuild 使用開頭為 `CodeBuild/` 的名稱以外的名稱存取 Amazon EC2 參數存放區中的參數 `/CodeBuild/`。例如，下列政策陳述式允許呼叫 `ssm:GetParameters` 動作，以取得指定名稱的參數：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/PARAMETER_NAME"
    }
  ]
}
```

無法存取 CodeBuild 主控台下的分支篩選條件

問題：當您建立或更新 AWS CodeBuild 專案時，主控台中無法使用分支篩選選項。

可能的原因：分支篩選選項已棄用。它已由 Webhook 篩選條件群組取代，該群組可進一步控制觸發 CodeBuild 中新建置的 Webhook 事件。

建議的解決方案：若要遷移在 Webhook 篩選條件推出之前建立的分支篩選條件，請使用 `HEAD_REF` 篩選條件和規則表達式 `^refs/heads/branchName$` 建立 Webhook 篩選群組。例如，如果分

支篩選條件規則表達式為 `^branchName$`，則您放在 `HEAD_REF` 篩選條件中已更新的規則表達式為 `^refs/heads/branchName$`。如需詳細資訊，請參閱[Bitbucket Webhook 事件](#)及[篩選 GitHub Webhook 事件 \(主控台\)](#)。

無法檢視組建成功或失敗

問題：您看不到重試的建置是成功或失敗。

可能的原因：建置狀態的報告選項未啟用。

建議的解決方案：建立或更新 CodeBuild 專案時啟用報告建置狀態。此選項會指示 CodeBuild 在您觸發組建時回報狀態。如需詳細資訊，請參閱 AWS CodeBuild API 參考中的 [reportBuildStatus](#)。

未向來源提供者報告建置狀態

問題：允許 GitHub 或 Bitbucket 等來源提供者的建置狀態報告之後，不會更新建置狀態。

可能原因：與來源提供者相關聯的使用者沒有儲存庫的寫入存取權。

建議的解決方案：若要能夠向來源提供者報告建置狀態，與來源提供者相關聯的使用者必須具有儲存庫的寫入存取權。如果使用者沒有寫入存取權，則無法更新建置狀態。如需詳細資訊，請參閱[來源提供者存取](#)。

找不到並選取 Windows Server Core 2019 平台的基礎映像

問題：找不到或選取 Windows Server Core 2019 平台的基礎映像。

可能原因：您使用 AWS 的區域不支援此映像。

建議的解決方案：使用下列 AWS 其中一個支援 Windows Server Core 2019 平台基礎映像的區域：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (奧勒岡)
- 歐洲 (愛爾蘭)

在 Buildspec 檔案中後來的命令無法辨識先前的命令

問題：Buildspec 檔案中後來的命令無法辨識同一 buildspec 檔案中先前一個或多命令的結果。例如，命令可能設定本機環境變數，但後來執行的命令可能無法取得該本機環境變數的值。

可能原因：在 buildspec 檔案 0.1 版中，AWS CodeBuild 會在建置環境的預設 shell 個別執行個體中執行每個命令。這表示每個命令會與其他命令隔離執行。所以，根據預設，如果單一命令倚賴任何先前命令的狀態，則無法執行此命令。

建議的解決方案：我們建議您使用 buildspec 0.2 版，以解決此問題。如果您必須使用 Buildspec 0.1 版，建議使用 Shell 命令鏈結運算子 (例如 Linux 中的 &&)，將多個命令合併成單一命令。或者，在您的原始碼中加進含有多個命令的 shell 指令碼，然後在 buildspec 檔案中從單一命令呼叫該 shell 指令碼。如需詳細資訊，請參閱[建置環境中的 Shell 和命令](#)及[建置環境中的環境變數](#)。

錯誤：嘗試下載快取時出現「存取遭拒」

問題：在已啟用快取的建置專案上嘗試下載快取時，您收到 Access denied 錯誤。

可能原因：

- 您剛在組建專案中設定快取。
- 快取最近才透過 InvalidateProjectCache API 而變成失效。
- 對於持有快取的 S3 儲存貯體，CodeBuild 使用的服務角色沒有 s3:GetObject 和 s3:PutObject 許可。

建議的解決方案：第一次使用時，在更新快取組態之後看到此訊息很正常。如果此錯誤持續發生，對於持有快取的 S3 儲存貯體，您應該檢查服務角色是否有 s3:GetObject 和 s3:PutObject 許可。如需詳細資訊，請參閱《Amazon [S3 開發人員指南](#)》中的[指定 S3 許可](#)。Amazon S3

使用自訂建置映像時發生錯

誤："BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE"

問題：當您嘗試執行的建置使用自訂建置映像時，建置失敗，發生錯誤 BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE。

可能原因：建置映像的整體未壓縮大小大於建置環境運算類型的可用磁碟空間。若要查看您的建置映像的大小，請使用 Docker 執行 `docker images REPOSITORY:TAG` 命令。如需依運算類型列出的可用磁碟空間清單，請參閱[建置環境運算模式和類型](#)。

建議的解決方案：使用具有更多可用磁碟空間的較大運算類型，或減少自訂建置映像的大小。

可能原因：AWS CodeBuild 沒有從您的 Amazon Elastic Container Registry (Amazon ECR) 提取建置映像的許可。

建議的解決方案：更新 Amazon ECR 中儲存庫的許可，以便 CodeBuild 可以將自訂建置映像提取至建置環境。如需更多資訊，請參閱[Amazon ECR 範例](#)。

可能原因：您請求的 Amazon ECR 映像不適用於 AWS 您的帳戶使用的 AWS 區域。

建議的解決方案：使用與您 AWS 帳戶使用的 位於相同 AWS 區域的 Amazon ECR 映像。

可能原因：您在沒有公有網際網路存取的 VPC 中使用私有登錄檔。CodeBuild 無法從 VPC 中的私有 IP 地址提取映像。如需詳細資訊，請參閱[具有 CodeBuild AWS Secrets Manager 範例的私有登錄檔](#)。

建議的解決方案：如果您在 VPC 中使用私有登錄檔，請確定 VPC 具有公有網際網路存取。

可能原因：如果錯誤訊息包含「toomanyrequests」，而且映像是從 Docker Hub 取得，則此錯誤表示已達到 Docker Hub 提取限制。

建議的解決方案：使用 Docker Hub 私有登錄檔，或從 Amazon ECR 取得映像。如需使用私有登錄檔的詳細資訊，請參閱 [具有 CodeBuild AWS Secrets Manager 範例的私有登錄檔](#)。如需使用 Amazon ECR 的詳細資訊，請參閱 [CodeBuild 的 Amazon ECR 範例](#)。

錯誤：「完成建置之前，發現建置容器失效。因為記憶體不足，或不支援 Docker 映像，建置容器失效。ErrorCode: 500」

問題：當您嘗試在 中使用 Microsoft Windows 或 Linux 容器時 AWS CodeBuild，此錯誤會在 PROVISIONING 階段發生。

可能原因：

- CodeBuild 不支援容器作業系統版本。
- 在容器中指定 HTTP_PROXY、HTTPS_PROXY，或兩者都指定。

建議解決方案：

- 針對 Microsoft Windows，請使用容器作業系統版本為 microsoft/windowsservercore:10.0.x 的 Windows 容器 (例如 microsoft/windowsservercore:10.0.14393.2125)。
- 針對 Linux，請清除 Docker 映像中的 HTTP_PROXY 和 HTTPS_PROXY 設定，或指定您建置專案的 VPC 組態。

執行組建時發生錯誤：「無法連接到 Docker 協助程式」

問題：您的建置失敗，並在建置日誌中收到類似 Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running? 錯誤。

可能的原因：您不是在特殊權限模式中執行您的建置。

建議的解決方案：若要修正此錯誤，您必須啟用特殊權限模式，並使用下列指示更新您的 buildspec。

若要在特殊權限模式下執行建置，請依照下列步驟執行：

1. 前往 <https://console.aws.amazon.com/codebuild/> 開啟 CodeBuild 主控台。
2. 在導覽窗格中，選擇建置專案，然後選擇您的建置專案。
3. 從 Edit (編輯) 選擇 Environment (環境)。
4. 選擇 Additional configuration (其他組態)。
5. 如果您想要建置 Docker 映像或希望建置獲得更高的權限，請從特權中選取啟用此旗標。
6. 選擇 Update environment (更新環境)。
7. 選擇 Start build (啟動組建) 來重試組建。

您也需要在容器內啟動 Docker 協助程式。buildspec 的 install 階段可能看起來像這樣。

```
phases:
  install:
    commands:
      - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
        host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
      - timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

如需 Buildspec 檔案中參考的 OverlayFS 儲存驅動程式的詳細資訊，請參閱 Docker 網站上的[使用 OverlayFS 儲存驅動程式](#)。

Note

如果基本作業系統是 Alpine Linux，請在 `buildspec.yml` 中將 `-t` 引數新增至 `timeout`：

```
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

若要進一步了解如何使用 建置和執行 Docker 映像 AWS CodeBuild，請參閱 [CodeBuild 自訂映像範例中的 Docker](#)。

錯誤：建立或更新組建專案時，「CodeBuild 無權執行：sts：AssumeRole」

問題：當您嘗試建立或更新建置專案時，您收到 `Code:InvalidInputException`，
`Message:CodeBuild is not authorized to perform: sts:AssumeRole on arn:aws:iam::account-ID:role/service-role-name` 錯誤。

可能原因：

- 您嘗試建立或更新建置專案 AWS 的區域已停用 AWS Security Token Service (AWS STS)。
- 與組建專案相關聯的 AWS CodeBuild 服務角色不存在，或沒有足夠的許可來信任 CodeBuild。
- 與建置專案相關聯的 AWS CodeBuild 服務角色大小寫與實際的 IAM 角色不相符。

建議解決方案：

- 確定已為您嘗試建立或更新建置專案 AWS 的區域 AWS STS 啟用。如需詳細資訊，請參閱《IAM 使用者指南》中的 [AWS STS 在 AWS 區域中啟用和停用](#)。
- 請確定 AWS 目標 CodeBuild 服務角色存在於您的帳戶中。如果您不是使用主控台，請確定您在建立或更新組建專案時沒有拼錯服務角色的 Amazon Resource Name (ARN)。請注意，IAM 角色區分大小寫，因此請檢查 IAM 角色的大小寫是否正確。
- 確定目標 CodeBuild 服務角色具有足夠的許可來信任 CodeBuild。如需詳細資訊，請參閱 [允許 CodeBuild 與其他 AWS 服務互動](#) 中的信任關係政策陳述式。

錯誤：「呼叫 GetBucketAcl 時發生錯誤：儲存貯體擁有者已變更，或服務角色不再有權呼叫 s3:GetBucketAcl」

問題：當您執行建置時，您會收到有關 S3 儲存貯體擁有權和 GetBucketAcl 許可變更的錯誤。

可能原因：您已將 s3:GetBucketAcl 和 s3:GetBucketLocation 許可新增至 IAM 角色。這些許可會保護您專案的 S3 儲存貯體，並確保只有您可加以存取。新增這些許可之後，S3 儲存貯體的擁有者會變更。

建議的解決方案：確認您是 S3 儲存貯體的擁有者，然後再次將許可新增至您的 IAM 角色。如需詳細資訊，請參閱[安全存取 S3 儲存貯體](#)。

執行建置時發生錯誤：「無法上傳成品：無效的 arn」

問題：當您執行建置時，UPLOAD_ARTIFACTS 建置階段失敗，並出現 Failed to upload artifacts: Invalid arn 錯誤。

可能原因：您的 S3 輸出儲存貯體（存放其建置輸出的 AWS CodeBuild 儲存貯體）位於與 CodeBuild 建置專案不同的 AWS 區域中。

建議的解決方案：更新組建專案的設定，以指向與組建專案位於相同 AWS 區域的輸出儲存貯體。

錯誤：「Git 複製失敗：無法存取 '**your-repository-URL**'：SSL 憑證問題：自我簽署憑證」

問題：當您嘗試執行建置專案時，建置失敗並出現此錯誤。

可能的原因：您的來源儲存庫有自我簽署憑證，但您在建置專案中未選擇從 S3 儲存貯體來安裝憑證。

建議解決方案：

- 編輯您的專案。針對 Certificate (憑證)，選擇 Install certificate from S3 (從 S3 安裝憑證)。針對 Bucket of certificate (憑證的儲存貯體)，選擇存放 SSL 憑證的 S3 儲存貯體。針對 Object key of certificate (憑證的物件金鑰)，輸入 S3 物件金鑰的名稱。
- 編輯您的專案。選取 Insecure SSL (不安全 SSL)，忽略連線至 GitHub Enterprise Server 專案儲存庫時出現的 SSL 警告。

 Note

建議您只使用 Insecure SSL (不安全 SSL) 進行測試。不應用於生產環境。

執行建置時發生錯誤：「必須使用指定的端點來定址您嘗試存取的儲存貯體」

問題：當您執行建置時，DOWNLOAD_SOURCE 建置階段失敗，並出現 The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint 錯誤。

可能原因：您的預先建置原始程式碼會存放在 S3 儲存貯體中，且該儲存貯體位於與 AWS CodeBuild 建置專案不同的 AWS 區域中。

建議的解決方案：更新建置專案的設定來指向您預先建置的原始程式碼所在的儲存貯體。請確定儲存貯體與建置專案位於相同的 AWS 區域。

錯誤：「此建置映像需要選取至少一個執行時間版本。」

問題：當您執行建置時，DOWNLOAD_SOURCE 建置階段失敗，並出現 YAML_FILE_ERROR: This build image requires selecting at least one runtime version 錯誤。

可能的原因：您的建置使用 Amazon Linux 2 (AL2) 標準映像版本 1.0 或更新版本，或 Ubuntu 標準映像版本 2.0 或更新版本和未於 buildspec 檔案中指定的執行時間。

建議的解決方案：如果您使用 aws/codebuild/standard:2.0 CodeBuild 受管映像，您必須在 buildspec 檔案的 runtime-versions 區段中指定執行時間版本。例如，對於使用 PHP 的專案，您可以使用以下 buildspec 檔案：

```
version: 0.2

phases:
  install:
    runtime-versions:
      php: 7.3
  build:
```

```
commands:
  - php --version
artifacts:
  files:
    - README.md
```

Note

如果您指定 `runtime-versions` 區段並使用 Ubuntu Standard Image 2.0 或更新版本以外的映像，或 Amazon Linux 2 (AL2) 標準映像 1.0 或更新版本，建置會發出警告 "Skipping install of runtimes. Runtime version selection is not supported by this build image."。

如需詳細資訊，請參閱 [Specify runtime versions in the buildspec file](#)。

錯誤："QUEUED: INSUFFICIENT_SUBNET"，當建置佇列中的建置失敗時

問題：建置佇列中的建置失敗，錯誤類似 `QUEUED: INSUFFICIENT_SUBNET`。

可能的原因：針對您 VPC 指定的 IPv4 CIDR 區塊使用預留的 IP 地址。您無法使用每個子網路 CIDR 區塊中的前四個 IP 地址和最後一個 IP 地址，也無法將這些 IP 地址指派給執行個體。例如，在使用 CIDR 區塊 `10.0.0.0/24` 的子網中，會預留下列五個 IP 地址：

- `10.0.0.0`：網路地址。
- `10.0.0.1`：由 AWS 為 VPC 路由器預留。
- `10.0.0.2`：預留者 AWS。DNS 伺服器的 IP 地址一律為 VPC 網路範圍的基礎加二；但是，我們也會預留每個子網路範圍的基礎加二。針對使用多個 CIDR 區塊的 VPC，DNS 伺服器的 IP 地址會位於主要 CIDR。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 [Amazon DNS 伺服器](#)。
- `10.0.0.3`：由預留 AWS 以供日後使用。
- `10.0.0.255`：網路廣播地址。VPC 中不支援播送。此為預留的地址。

建議的解決方案：檢查您的 VPC 是否使用預留 IP 地址。請將所有預留 IP 地址替換為非預留的 IP 地址。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 [VPC 和子網路規模調整](#)。

錯誤：「無法下載快取：RequestError: 傳送請求失敗，原因：x509: 無法載入系統根目錄，未提供根目錄」

問題：當您嘗試執行建置專案時，建置失敗並出現此錯誤。

可能的原因：您已在建置專案中設定快取，而使用的 Docker 影像較舊且包含過期的根憑證。

建議的解決方案：更新 AWS CodeBuild 專案中使用的 Docker 映像。如需詳細資訊，請參閱[CodeBuild 提供的 Docker 映像](#)。

錯誤：「無法從 S3 下載憑證。AccessDenied」

問題：當您嘗試執行建置專案時，建置失敗並出現此錯誤。

可能原因：

- 您選擇錯誤的 S3 儲存貯體來取得您的憑證。
- 您為憑證輸入的物件金鑰錯誤。

建議解決方案：

- 編輯您的專案。針對 Bucket of certificate (憑證的儲存貯體)，選擇存放 SSL 憑證的 S3 儲存貯體。
- 編輯您的專案。針對 Object key of certificate (憑證的物件金鑰)，輸入 S3 物件金鑰的名稱。

錯誤：「找不到登入資料」

問題：當您嘗試執行 AWS CLI、使用 AWS SDK 或呼叫另一個類似的元件做為組建的一部分時，您會收到與 AWS CLI、AWS SDK 或元件直接相關的組建錯誤。例如，您可能會收到 Unable to locate credentials 等建置錯誤。

可能原因：

- 建置環境中的 AWS CLI、AWS SDK 或元件版本不相容 AWS CodeBuild。
- 您正在使用 Docker 的建置環境中執行 Docker 容器，而且容器預設無法存取 AWS 登入資料。

建議解決方案：

- 請確定您的建置環境具有以下版本或更新版本的 AWS CLI、AWS SDK 或元件。

- AWS CLI: 1.10.47
 - AWS 適用於 C++ 的 SDK : 0.2.19
 - AWS 適用於 Go 的 SDK : 1.2.5
 - AWS 適用於 Java 的 SDK : 1.11.16
 - AWS 適用於 JavaScript 的 SDK : 2.4.7
 - AWS 適用於 PHP 的 SDK : 3.18.28
 - AWS 適用於 Python 的 SDK (Boto3) : 1.4.0
 - AWS 適用於 Ruby 的 SDK : 2.3.22
 - Botocore: 1.4.37
 - CoreCLR: 3.2.6-beta
 - Node.js: 2.4.7
- 如果您需要在建置環境中執行 Docker 容器，且容器需要 AWS 登入資料，則必須將登入資料從建置環境傳遞至容器。在您的 buildspec 檔案中，加進如下的 Docker run 命令。此範例使用 `aws s3 ls` 命令列出可用的 S3 儲存貯體。`-e` 選項會傳遞容器存取 AWS 登入資料所需的環境變數。

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag aws s3 ls
```

- 如果您要建置 Docker 映像，且建置需要 AWS 登入資料（例如，從 Amazon S3 下載檔案），則必須將登入資料從建置環境傳遞至 Docker 建置程序，如下所示。

1. 在 Docker 影像的原始碼 Dockerfile 中，指定下列 ARG 指示。

```
ARG AWS_DEFAULT_REGION
ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

2. 在您的 buildspec 檔案中，加進如下的 Docker build 命令。這些 `--build-arg` 選項會設定 Docker 建置程序存取 AWS 登入資料所需的環境變數。

```
docker build --build-arg AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION --build-arg
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI -
t your-image-tag .
```

在代理伺服器中執行 CodeBuild 時發生 RequestError 逾時錯誤

問題：您收到類似以下其中一項的 RequestError 錯誤：

- `RequestError: send request failed caused by: Post https://logs.<your-region>.amazonaws.com/: dial tcp 52.46.158.105:443: i/o timeout` 來自 CloudWatch Logs。
- `Error uploading artifacts: RequestError: send request failed caused by: Put https://<your-bucket>.s3.<your-aws-region>.amazonaws.com/*: dial tcp 52.219.96.208:443: connect: connection refused` 來自 Amazon S3。

可能原因：

- `ssl-bump` 未正確設定。
- 您的組織的安全政策不允許您使用 `ssl_bump`。
- 您的 `buildspec` 檔案無使用 `proxy` 元素指定的代理設定。

建議解決方案：

- 請確定 `ssl-bump` 已正確設定。如果您使用 Squid 作為代理伺服器，請參閱[將 Squid 設定為明確代理伺服器](#)。
- 請依照下列步驟，將私有端點用於 Amazon S3 和 CloudWatch Logs：
 1. 在您的私有子網路路由表中，移除您為了將流向網際網路的流量路由傳送到代理伺服器而新增的規則。如需詳細資訊，請參閱《Amazon [VPC 使用者指南](#)》中的[在 VPC 中建立子網路](#)。
 2. 建立私有 Amazon S3 端點和 CloudWatch Logs 端點，並將其與 Amazon VPC 的私有子網路建立關聯。如需詳細資訊，請參閱《Amazon [VPC 使用者指南](#)》中的[VPC 端點服務](#)。
 3. 確認已選取 Amazon VPC 中的啟用私有 DNS 名稱。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[建立介面端點](#)。
- 如果您未將 `ssl-bump` 用於明確的代理伺服器，請使用 `proxy` 元素將代理組態新增至您的 `buildspec` 檔案。如需詳細資訊，請參閱[在明確代理伺服器中執行 CodeBuild](#)及[Buildspec 語法](#)。

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
```

Bourne Shell (sh) 必須存在於建置映像中

問題：您使用的建置映像不是由提供 AWS CodeBuild，而且您的建置失敗並顯示訊息 Build container found dead before completing the build。

可能原因：Bourne shell (sh) 不包含在您的建置映像中。CodeBuild sh 需要執行建置命令和指令碼。

建議的解決方案：如果 sh 不存在於建置映像中，請務必在啟動任何其他使用映像的建置之前將其納入。(CodeBuild 已包含在其建置映像 sh 中。)

執行建置時出現警告：「正在略過執行時間的安裝。此建置映像不支援執行時間版本選項」

問題：執行建置時，建置日誌包含此警告。

可能原因：您的建置不會使用 Amazon Linux 2 (AL2) 標準映像的 1.0 版或更新版本，或 Ubuntu 標準映像的 2.0 版或更新版本，而且執行時間會在 buildspec 檔案中的 runtime-versions 區段中指定。

建議的解決方案：確保 buildspec 檔案不包含 runtime-versions 區段。只有在您使用 Amazon Linux 2 (AL2) 標準映像或更新版本，或 Ubuntu 標準映像 2.0 版或更新版本時，才需要 runtime-versions 本節。

錯誤：開啟 CodeBuild 主控台時「無法驗證 JobWorker 身分」

問題：當您開啟 CodeBuild 主控台時，會顯示「無法驗證 JobWorker 身分」錯誤訊息。

可能原因：用於主控台存取的 IAM 角色具有標籤 jobId 做為金鑰。此標籤金鑰保留給 CodeBuild，如果存在則會導致此錯誤。

建議的解決方案：將任何具有金鑰的自訂 IAM 角色標籤變更為 jobId 具有不同金鑰，例如 jobIdIdentifier。

組建無法啟動

問題：啟動組建時，您會收到組建無法啟動錯誤訊息。

可能原因：已達到並行建置的數量。

建議的解決方案：等到其他組建完成，或提高專案的並行組建限制，然後再次啟動組建。如需詳細資訊，請參閱[專案組態](#)。

在本機快取建置中存取 GitHub 中繼資料

問題：在某些情況下，快取組建中的 .git 目錄是文字檔案，而不是目錄。

可能原因：啟用建置的本機來源快取時，CodeBuild 會為 .git 目錄建立 gitlink。這表示 .git 目錄實際上是包含目錄路徑的文字檔案。

建議的解決方案：在所有情況下，請使用下列命令來取得 Git 中繼資料目錄。無論 的格式為何，此命令都可以運作 .git：

```
git rev-parse --git-dir
```

AccessDenied：報告群組的儲存貯體擁有者不符合 S3 儲存貯體的擁有者...

問題：將測試資料上傳至 Amazon S3 儲存貯體時，CodeBuild 無法將測試資料寫入儲存貯體。

可能原因：

- 為報告群組儲存貯體擁有者指定的帳戶與 Amazon S3 儲存貯體的擁有者不相符。
- 服務角色沒有對儲存貯體的寫入存取權。

建議解決方案：

- 變更報告群組儲存貯體擁有者，以符合 Amazon S3 儲存貯體的擁有者。
- 修改服務角色以允許對 Amazon S3 儲存貯體的寫入存取。

錯誤：使用 建立 CodeBuild 專案時，「您的登入資料缺少一或多個必要的權限範圍」 AWS CodeConnections

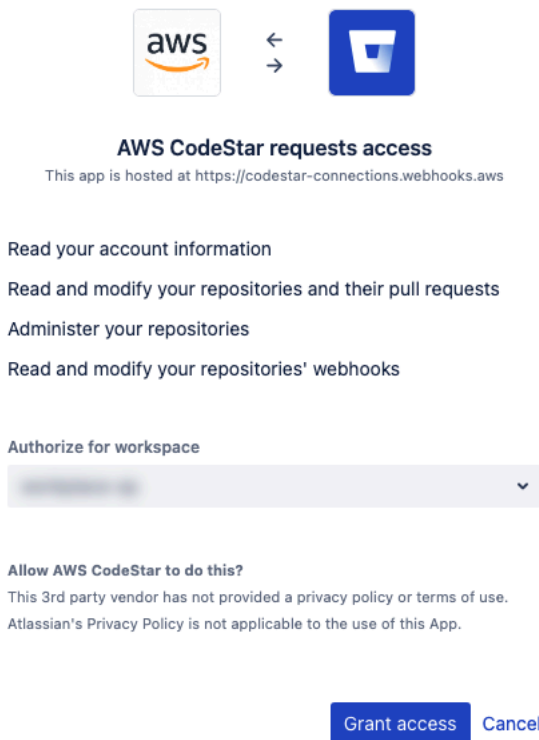
問題：使用 建立 CodeBuild 專案時 AWS CodeConnections，您沒有安裝 Bitbucket Webhook 的許可。

可能原因：

- 您的 Bitbucket 帳戶中可能尚未接受新的許可範圍。

建議解決方案：

- 若要接受新的許可，您應該已收到一封電子郵件，其中包含標題為必要動作 - 適用於 AWS CodeStar 的範圍已由 Bitbucket 傳送的變更notifications-noreply@bitbucket.org。電子郵件包含連結，可將 Webhook 許可授予您現有的 AWS CodeConnections Bitbucket 應用程式安裝。
- 如果您找不到電子郵件，您可以透過導覽至 https://bitbucket.org/site/addons/reauthorize?account=<workspace-name>&addon_key=aws-codestar 或 https://bitbucket.org/site/addons/reauthorize?addon_key=aws-codestar 選取您要授予 Webhook 許可的工作區來授予許可。



錯誤：使用 Ubuntu 安裝命令建置時，「很抱歉，完全沒有終端請求 - 無法取得輸入」

問題：如果您正在執行 GPU 容器特殊權限組建，則可能會依照[此程序](#)安裝 NVIDIA Container Toolkit。在最新的 CodeBuild 映像版本中，CodeBuild 會在最新和精選的映像 `nvidia-container-toolkit` 中使用 `ubuntu` 預先安裝 `amazonlinux` 和設定 Docker。遵循此程序會導致使用 Ubuntu 安裝命令的組建失敗，並出現下列錯誤：

```
Running command curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | gpg --
dearmor --no-tty -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
gpg: Sorry, no terminal at all requested - can't get input
curl: (23) Failed writing body
```

可能的原因：gpg 金鑰已存在於相同的位置。

建議的解決方案：nvidia-container-toolkit映像中已安裝。如果您看到此錯誤，您可以略過buildspec 中的安裝並重新啟動 docker 程序。

Bitbucket 組建在使用 Secrets Manager OAuth 後失敗並出現身分驗證錯誤

問題：使用 Bitbucket 做為來源提供者的建置失敗，並出現來源憑證錯誤。Bitbucket 連線是使用 OAuth 搭配 AWS Secrets Manager 做為儲存服務來設定。

可能原因：Atlassian 會強制執行 Bitbucket OAuth 的一次性輪換重新整理權杖。每次使用重新整理權杖時，Bitbucket 都會將其失效並傳回新的權杖。如需詳細資訊，請參閱 Atlassian 開發人員網站上的 [Bitbucket OAuth 單次使用重新整理權杖 \(CHANGE-3052\)](#)。

建議的解決方案：將 secretsmanager:PutSecretValue 許可新增至 CodeBuild 服務角色。如需所需 IAM 許可的詳細資訊，請參閱 [Secrets Manager 儲存的登入資料所需的動作](#)。

的配額 AWS CodeBuild

下表列出目前的配額 AWS CodeBuild。除非另有指定，否則這些配額適用於每個 AWS 帳戶的每個支援 AWS 區域。

Service Quotas

以下是 AWS CodeBuild 服務的預設配額。

名稱	預設	可調整	說明
每個專案的相關標籤	每個受支援的區域：50	否	您可以與組建專案相關聯的標籤數目上限
組建專案	每個受支援的區域：5,000	是	建置專案上限數量
建置逾時以分鐘為單位	每個支援的區域：2,160	否	以分鐘為單位的建置逾時上限
有關組建資訊的並行請求	每個受支援的區域：100	否	您可以使用 CLI 或 AWS SDK 隨時請求有關 AWS 資訊的組建數量上限。
有關建置專案資訊的並行請求	每個受支援的區域：100	否	您可以使用 CLI 或 AWS SDK 隨時請求有關 AWS 資訊的建置專案數量上限。
同時執行 ARM Lambda/10GB 環境的建置	每個受支援的區域：1	是	ARM Lambda/10GB 環境同時執行建置的數量上限
同時執行 ARM Lambda/1GB 環境的建置	每個受支援的區域：1	是	ARM Lambda/1GB 環境同時執行組建的數量上限

名稱	預設	可調整	說明
同時執行 ARM Lambda/2GB 環境的建置	每個受支援的區域：1	是	ARM Lambda/2GB 環境同時執行建置的數量上限
同時執行 ARM Lambda/4GB 環境的建置	每個受支援的區域：1	是	ARM Lambda/4GB 環境同時執行組建的數量上限
同時執行 ARM Lambda/8GB 環境的建置	每個受支援的區域：1	是	ARM Lambda/8GB 環境同時執行組建的數量上限
同時執行 ARM/2XLarge 環境的建置	每個受支援的區域：1	是	ARM/2XLarge 環境同時執行建置的數量上限
同時執行 ARM/Large 環境的建置	每個受支援的區域：1	是	ARM/Large 環境同時執行組建的數量上限
同時執行 ARM/Medium 環境的建置	每個受支援的區域：1	是	ARM/Medium 環境同時執行建置的數量上限
同時執行 ARM/小型環境的建置	每個受支援的區域：1	是	ARM/小型環境並行執行組建的數量上限
同時執行 ARM/XLarge 環境的建置	每個受支援的區域：1	是	ARM/XLarge 環境同時執行建置的數量上限
同時執行 Linux GPU 大型環境的組建	每個支援的區域：0	是	Linux GPU/大型環境同時執行組建的數量上限
同時執行 Linux GPU Small 環境的組建	每個支援的區域：0	是	Linux GPU/小型環境並行執行組建的數量上限
同時執行 Linux Lambda/10GB 環境的建置	每個受支援的區域：1	是	Linux Lambda/10GB 環境同時執行建置的數量上限
同時執行 Linux Lambda/1GB 環境的建置	每個受支援的區域：1	是	Linux Lambda/1GB 環境同時執行建置的數量上限

名稱	預設	可調整	說明
同時執行 Linux Lambda/2GB 環境的建置	每個受支援的區域：1	是	Linux Lambda/2GB 環境同時執行的組建數目上限
同時執行 Linux Lambda/4GB 環境的建置	每個受支援的區域：1	是	Linux Lambda/4GB 環境同時執行建置的數量上限
同時執行 Linux Lambda/8GB 環境的建置	每個受支援的區域：1	是	Linux Lambda/8GB 環境同時執行建置的數量上限
同時執行 Linux/2XLarge 環境的建置	每個支援的區域：0	是	Linux/2XLarge 環境同時執行建置的數量上限
同時執行 Linux/Large 環境的建置	每個受支援的區域：1	是	Linux/Large 環境同時執行組建的數量上限
同時執行 Linux/Medium 環境的建置	每個受支援的區域：1	是	Linux/Medium 環境同時執行建置的數量上限
同時執行 Linux/小型環境的建置	每個受支援的區域：1	是	Linux/小型環境並行執行組建的數量上限
同時執行 Linux/XLarge 環境的建置	每個受支援的區域：1	是	Linux/XLarge 環境同時執行建置的數量上限
同時執行 Windows Server 2019/Large 環境的組建	每個受支援的區域：1	是	同時執行的 Windows Server 2019/Large 環境組建數量上限
同時執行 Windows Server 2019/Medium 環境的組建	每個受支援的區域：1	是	同時執行的 Windows Server 2019/Medium 環境組建數量上限
同時執行 Windows Server 2022/2XLarge 環境的組建	每個受支援的區域：1	是	Windows Server 2022/2XLarge 環境並行執行組建的數量上限

名稱	預設	可調整	說明
同時執行 Windows Server 2022/Large 環境的組建	每個受支援的區域：1	是	Windows Server 2022/Large 環境同時執行組建的數量上限
同時執行 Windows Server 2022/Medium 環境的組建	每個受支援的區域：1	是	針對 Windows Server 2022/Medium 環境同時執行組建的數量上限
同時執行 Windows Server 2022/XLarge 環境的組建	每個受支援的區域：1	是	針對 Windows Server 2022/XLarge 環境同時執行組建的數量上限
同時執行適用於 Windows/Large 環境的組建	每個受支援的區域：1	是	Windows/Large 環境同時執行建置的數量上限
同時執行 Windows/Medium 環境的建置	每個受支援的區域：1	是	Windows/Medium 環境同時執行建置的數量上限
建置逾時的最短期間，以分鐘為單位	每個受支援的區域：5	否	最短建置逾時，以分鐘為單位
VPC 組態下的安全群組	每個受支援的區域：5	否	可用於 VPC 組態的安全群組
VPC 組態下的子網路	每個受支援的區域：16	否	適用於 VPC 組態的子網路

 Note

內部指標將決定並行執行組建的預設配額。

並行執行組建數目上限的配額會根據運算類型而有所不同。對於某些平台和運算類型，預設值為 20。若要請求更高的並行組建配額，或者如果您收到「帳戶不能有超過 X 個作用中組建」錯誤，請使用上面的連結提出請求。如需定價的詳細資訊，請參閱 [AWS CodeBuild 定價](#)。

其他限制

組建專案

資源	預設
組建專案描述中允許的字元	任何
組建專案名稱中允許的字元	字母 A-Z 和 a-z、數字 0-9，以及特殊字元 - 和 _
建置專案名稱的長度	2 到 150 個字元，包含
組建專案描述的長度上限	255 個字元
您可以新增至專案的報告上限數量	5
您可以在組建專案中針對所有相關組建的組建逾時指定的分鐘數	5 到 2160 (36 小時)

Builds

資源	預設
保留的建置歷史記錄時間上限	1 年
您可以針對單一組建的組建逾時指定的分鐘數	5 到 2160 (36 小時)

運算機群

資源	預設
運算機群的並行數量	10
同時執行 ARM/小型環境機群的執行個體	1
同時執行 ARM/Large 環境機群的執行個體	1
同時執行 Linux/小型環境機群的執行個體	1
同時執行 Linux/Medium 環境機群的執行個體	1
同時執行 Linux/大型環境機群的執行個體	1
同時執行 Linux/XLarge 環境機群的執行個體	1
同時執行 Linux/2XLarge 環境機群的執行個體	0
同時執行 Linux GPU/小型環境機群的執行個體	0
同時執行 Linux GPU/大型環境機群的執行個體	0
同時執行 Windows Server 2019/Medium 環境機群的執行個體	1
同時執行 Windows Server 2019/大型環境機群的執行個體	1
同時執行 Windows Server 2022/Medium 環境機群的執行個體	1
同時執行 Windows Server 2022/Large 環境機群的執行個體	1
同時執行 Mac ARM/Medium 環境機群的執行個體	1
同時執行 Mac ARM/大型環境機群的執行個體	1

報告

資源	預設
測試報告建立後可用的最大期間	30 天
測試案例訊息的長度上限	5,000 個字元
測試案例名稱的長度上限	1,000 個字元
每個 AWS 帳戶的報告群組數量上限	5,000
每個報告的測試案例上限數量	500

Tags (標籤)

標籤限制適用於 CodeBuild 組建專案和 CodeBuild 報告群組資源上的標籤。

資源	預設
資源標籤金鑰名稱	<p>任何 Unicode 字母、數字、空格，以及允許的 UTF-8 字元的組合，長度在 1 到 127 個字元之間。允許的字元為 + - = . _ : / @</p> <p>標籤金鑰名稱必須是唯一的，而且每個金鑰只能有一個值。標籤金鑰名稱不可：</p> <ul style="list-style-type: none">• 以 aws：開頭• 只包含空格• 以空格結尾• 包含表情圖示或以下任何字元：? ^ * [\ ~ ! # \$ % & * () > < " ' ` [] { } ;
資源標籤值	任何 Unicode 字母、數字、空格，以及允許的 UTF-8 字元的組合，長度在 0 到 255 個字元之間。允許的字元為 + - = . _ : / @

資源	預設
	金鑰只能有一個值，但多個金鑰可以有相同的值。標籤金鑰值不可包含表情圖示或以下任何字元： ? ^ * [\ ~ ! # \$ % & * () > < " ' ` [] { } ;

AWS CodeBuild 使用者指南文件歷史記錄

下表說明自上次發行 以來文件的重要變更 AWS CodeBuild。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

- 最新 API 版本：2016-10-06

變更	描述	日期
更新內容：CodeBuild 標準映像的可用執行時間	新增適用於 x86_64 標準映像的 Node.js 26 執行期和 Rust 1.97。	2026 年 9 月 22 日
更新內容：CodeBuild 標準映像的可用執行時間	已將 Amazon Linux 2023 AArch64 標準：4.0 映像新增至支援的執行時間。	2026 年 9 月 22 日
更新內容：Webhook 篩選條件規則表達式語法	釐清 Webhook 篩選模式使用 RE2 規則表達式語法，以及不支援哪些建構。	2026 年 8 月 3 日
已更新內容：Lambda 運算映像	新增 .NET 10、Go 1.25、Node.js 24 和的 Lambda Corretto 25 運算映像 Python 3.14。	2026 年 7 月 29 日
已更新內容：Bitbucket 應用程式密碼、API 字符或存取字符	新增對 Bitbucket API 字符的支援，以取代已取代的應用程式密碼。	2026 年 7 月 19 日
自訂 GitHub Enterprise 角色映射和 MAINTAIN 角色名稱修正	新增文件，說明 GitHub Enterprise 自訂儲存庫角色如何根據許可階層對應至標準角色。在 GITHUB_MAINTAIN 預設值和範例中，將維護層級角色名稱更正為。	2026 年 4 月 13 日

更新內容：CodeBuild 標準映像的可用執行時間	更新可用的執行時間版本並新增新映像。	2026 年 4 月 7 日
已更新內容：EC2 運算映像	新增 Amazon Linux 2023 x86_64 標準：6.0 和 Ubuntu 24.04 標準：8.0 運算映像。	2026 年 4 月 7 日
更新內容：執行時間版本選取行為	新增了當先前預設版本達到生命週期結束 (EOL) 時，預設執行時間版本可能會變更的指引。	2026 年 3 月 25 日
更新內容：CodeBuild 標準映像的可用執行時間	更新可用的執行時間版本並新增新映像。	2026 年 3 月 25 日
已更新的 content：AWS managed (predefined) 政策 AWS CodeBuild	AWSCodeBuildAdminAccess 和 AWSCodeBuildDeveloperAccess 政策已更新，以包含 Session Manager 互動式建置偵錯所需的 <code>ssmmessages:OpenDataChannel</code> 許可。	2025 年 12 月 1 日
提取請求評論核准	CodeBuild 支援提取請求建置政策，可對提取請求觸發的建置提供額外的控制。	2025 年 8 月 7 日
已更新的 content：AWS managed (predefined) 政策 AWS CodeBuild	AWSCodeBuildAdminAccess、AWSCodeBuildDeveloperAccess 和 AWSCodeBuildReadOnlyAccess 政策已更新。原始資源 <code>arn:aws:codeconnections:*:*:connection/*</code> 已更新為 <code>arn:aws:codeconnections:*:*:*</code> 。	2025 年 6 月 10 日

CodeBuild 條件索引鍵的新參考	已新增參考頁面，其中包含使用 CodeBuild 條件金鑰的範例。請參閱 AWS CodeBuild 條件索引鍵 。	2025 年 5 月 15 日
新內容：CodeBuild 的 Docker 映像建置伺服器範例	CodeBuild 現在支援將 Docker 組建卸載至受管映像組建伺服器。	2025 年 5 月 15 日
新的運算類型：CUSTOM_INSTANCE_TYPE	CodeBuild 現在可讓您使用建立具有特定執行個體類型的預留容量機群CUSTOM_INSTANCE_TYPE 。	2025 年 4 月 23 日
CodeBuild 沙盒的新支援	新增使用新 CodeBuild 沙盒的相關資訊。請參閱 使用 CodeBuild 沙盒偵錯組建 。	2025 年 4 月 7 日
新的 Windows 環境類型	CodeBuild 現在支援 Windows XL 和 2XL 環境類型。如需詳細資訊，請參閱 建置環境運算類型 。	2025 年 3 月 31 日
已更新 Amazon S3 快取	CodeBuild 現在支援 Amazon S3 快取的新快取行為。	2025 年 3 月 28 日
新內容：GitHub Actions Runner 組態選項	CodeBuild 現在支援在企業層級CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME 註冊。	2025 年 3 月 11 日
新內容：新增 Webhook 篩選條件類型	新增對新 Webhook 篩選條件類型的支援 (ORGANIZATION_NAME)。	2025 年 3 月 11 日

新內容：使用具有 S3 憑證儲存的 Fastlane 進行 Apple 程式碼簽署教學課程	在 CodeBuild 中使用 S3 進行憑證儲存的 Fastlane 新增 Apple 程式碼簽署教學課程	2025 年 2 月 5 日
新內容：GitHub 憑證儲存的 Fastlane Apple 程式碼簽署教學課程	在 CodeBuild 中使用 GitHub 進行憑證儲存的 Fastlane 新增 Apple 程式碼簽署教學課程	2025 年 2 月 5 日
新內容：Buildkite Runner	新增 Buildkite 執行器的新內容	2025 年 1 月 31 日
新內容：Buildkite 手動 Webhook	新增對 Buildkite 手動 Webhook 的支援。	2025 年 1 月 31 日
新內容：批次建置 buildspec 參考	新增對預留容量機群和 Lambda 環境中批次建置的支援。	2025 年 1 月 8 日
新內容：在批次建置中執行平行測試	在批次建置中新增平行測試的新內容。	2025 年 1 月 2 日
新內容：自動重試建置	CodeBuild 現在支援 Webhook 組建的自動重試。	2024 年 12 月 18 日
新內容：設定自我託管執行器的私有登錄登入資料	新增使用來自非私有登錄檔的自訂映像時，設定登錄檔登入資料的支援。	2024 年 12 月 13 日
新內容：GitHub Actions Runner 組態選項	CodeBuild GitHub Actions 自我託管執行器現在可讓您在組織層級註冊執行器，並設定特定的執行器群組 ID。	2024 年 12 月 12 日
新增內容：新增失敗時屬性 RETRY	CodeBuild 現在可讓您在 buildspec RETRY 中設定失敗時屬性至。	2024 年 12 月 12 日
新內容：GitLab 手動 Webhook	新增對 GitLab 手動 Webhook 的支援。	2024 年 12 月 11 日

更新內容：更新別名	更新 Linux 型標準執行期映像的別名。	2024 年 11 月 22 日
更新內容：CodeBuild 託管的 GitLab 執行器支援標籤覆寫	新增對 GitLab 執行器自訂映像標籤覆寫的支援。	2024 年 11 月 22 日
更新內容：CodeBuild 託管的 GitHub 動作執行器支援標籤覆寫	新增對 GitHub Actions 執行器自訂映像標籤覆寫的支援。	2024 年 11 月 22 日
已更新的 content：AWS managed (predefined) 政策 AWS CodeBuild	AWSCodeBuildAdminAccess、AWSCodeBuildDeveloperAccess 和 AWSCodeBuildReadOnlyAccess 政策已更新。原始資源 <code>arn:aws:codebuild:</code> * 已更新為 <code>arn:aws:codebuild:*:*:project/*</code> 。	2024 年 11 月 15 日
更新內容：預留容量	預留容量機群現在支援非容器建置：ARM EC2、Linux EC2 和 Windows EC2。	2024 年 11 月 12 日
更新內容：預留容量	預留容量機群現在支援屬性型運算。	2024 年 11 月 6 日
新內容：自動重試建置	CodeBuild 現在可讓您為組建啟用自動重試。	2024 年 10 月 25 日
新內容：在預留容量機群的受管代理伺服器中執行 CodeBuild	新增預留容量機群的代理組態支援。	2024 年 10 月 15 日
新內容：自我管理的 GitLab 執行器	新增自我管理 GitLab 執行器的新內容	2024 年 9 月 17 日
新內容：GitLab 群組 Webhook	新增對 GitLab 群組 Webhook 的支援。	2024 年 9 月 17 日

新內容：在 INSTALL、PRE_BUILD 和 POST_BUILD 階段執行 buildspec 命令	新增對的支援-with-buildspec。	2024 年 8 月 20 日
更新內容：預留容量	預留容量機群現在支援 macOS。	2024 年 8 月 19 日
新內容：GitHub 應用程式連線	新增對 GitHub 應用程式連線的支援。	2024 年 8 月 14 日
新內容：Bitbucket 應用程式連線	新增對 Bitbucket 應用程式連線的支援。	2024 年 8 月 14 日
新內容：CodeBuild 中的多個存取字符	新增支援從 中的秘密 AWS Secrets Manager 或透過 AWS CodeConnections 連線將存取權杖來源給第三方供應商。	2024 年 8 月 14 日
更新內容：預留容量	預留容量機群現在支援 ARM Medium、ARM XLarge 和 ARM 2XLarge 運算類型。	2024 年 8 月 5 日
更新內容：預留容量	CodeBuild 現在支援 Windows 上預留容量機群的 VPC 連線能力。	2024 年 8 月 1 日
新的 ARM 運算類型	CodeBuild 現在支援 ARM Medium、ARM XLarge 和 ARM 2XLarge 運算類型。如需詳細資訊，請參閱 建置環境運算類型 。	2024 年 7 月 10 日
更新內容：SHA 簽章	更新 x86_64 和 ARM 的安全雜湊演算法 (SHA) 簽章。	2024 年 6 月 19 日
新內容：GitHub 全域和組織 Webhook	新增對 GitHub 全域和組織 Webhook 的支援。	2024 年 6 月 17 日

新內容：新增 Webhook 篩選條件類型	新增對新 Webhook 篩選條件類型的支援 (REPOSITORY_NAME)。	2024 年 6 月 17 日
已更新磁碟空間	ARM Small 和 ARM Large 運算類型現在已增加磁碟空間。	2024 年 6 月 4 日
新內容：GitHub 手動 Webhook	新增對 GitHub 手動 Webhook 的支援。	2024 年 5 月 23 日
更新內容：預留容量	CodeBuild 現在支援 Amazon Linux 上預留容量機群的 VPC 連線能力。	2024 年 5 月 15 日
已更新內容：Lambda 運算映像	新增 .NET 8 (a1-lambda/aarch64/dotnet8 和 a1-lambda/x86_64/dotnet8) 的 Lambda 支援	2024 年 5 月 8 日
已更新配額：建置逾時	將建置逾時配額上限更新為 2160 分鐘 (36 小時)。	2024 年 5 月 1 日
已更新的 content：AWS managed (predefined) 政策 AWS CodeBuild	AWSCodeBuildAdminAccess、AWSCodeBuildDeveloperAccess 和 AWSCodeBuildReadOnlyAccess 政策已更新，以反映品牌 AWS CodeConnections 重塑。	2024 年 4 月 30 日
新內容：Bitbucket 應用程式密碼、API 字符或存取字符	新增對 Bitbucket 存取權杖的支援。	2024 年 4 月 11 日
新內容：在 CodeBuild 中自動探索報告	CodeBuild 現在支援報告自動探索。	2024 年 4 月 4 日
新內容：自我託管的 GitHub 動作執行器	新增自我託管 GitHub Actions 執行器的新內容	2024 年 4 月 2 日

新內容：GitLab 連線	新增對 GitLab 和 GitHub 自我管理連線的支援。	2024 年 3 月 25 日
新內容：新增 Webhook 事件和篩選條件類型	新增對新 Webhook 事件 (RELEASED 和 PRERELEASED) 和篩選條件類型 (TAG_NAME 和) 的支援RELEASE_NAME 。	2024 年 3 月 15 日
新內容：新增 Webhook 事件：PULL_REQUEST_CLOSED	新增對新 Webhook 事件的支援：PULL_REQUEST_CLOSED 。	2024 年 2 月 20 日
已更新內容：CodeBuild 提供的 Docker 映像	新增對 Windows Server Core 2019 的支援 (windows-base:2019-3.0)	2024 年 2 月 7 日
已更新內容：CodeBuild 提供的 Docker 映像	新增對 Amazon Linux 2023 新執行時間的支援 (a12/aarch64/standard/3.0)	2024 年 1 月 29 日
新內容：預留容量	CodeBuild 現在支援 CodeBuild 中的預留容量機群。	2024 年 1 月 18 日
新的運算類型	CodeBuild 現在支援 Linux XLarge 運算類型。如需詳細資訊，請參閱 建置環境運算類型 。	2024 年 1 月 8 日
已更新內容：CodeBuild 提供的 Docker 映像	新增對 Amazon Linux 2 (a12/standard/5.0) 和 Ubuntu (ubuntu/standard/7.0) 新執行時間的支援	2023 年 12 月 14 日
已更新內容：CodeBuild 提供的 Docker 映像	新增對新 Lambda 運算映像的支援	2023 年 12 月 8 日

新內容：AWS Lambda 運算	新增 AWS Lambda 運算的新內容	2023 年 11 月 6 日
已更新內容：CodeBuild 提供的 Docker 映像	新增對 Amazon Linux 2 的支援 (a12/standard/5.0)	2023 年 5 月 17 日
CodeBuild 受管政策的變更	CodeBuild AWS 受管政策更新的詳細資訊現已提供。如需詳細資訊，請參閱 CodeBuild AWS 受管政策的更新 。	2023 年 5 月 16 日
已更新內容：CodeBuild 提供的 Docker 映像	移除對 Amazon Linux 2 (a12/standard/3.0) 的支援，並新增對 Amazon Linux 2 (a12/standard/corretto8) 和 Amazon Linux 2 (a12/standard/corretto11) 的支援	2023 年 5 月 9 日
已更新內容：CodeBuild 提供的 Docker 映像	新增對 Ubuntu 22.04 的支援 (ubuntu/standard/7.0)	2023 年 4 月 13 日
已更新內容：CodeBuild 提供的 Docker 映像	移除對 Ubuntu 18.04 (ubuntu/standard/4.0) 和 Amazon Linux 2 (a12/aarch64/standard/1.0) 的支援	2023 年 3 月 31 日
更新內容：移除 VPC 限制	移除下列限制：如果您將 CodeBuild 設定為使用 VPC，則不支援本機快取。從 02/28/22 開始，您的 VPC 組建需要更長的時間，因為每個組建將使用新的 Amazon EC2 執行個體。	2023 年 3 月 1 日

已更新內容：CodeBuild 提供的 Docker 映像	移除對 Ubuntu 18.04 (ubuntu/standard/3.0) 和 Amazon Linux 2 (al2/standard/2.0) 的支援	2022 年 6 月 30 日
Amazon ECR 範例：限制映像存取	使用 CodeBuild 登入資料提取 Amazon ECR 映像時，您可以限制對特定 CodeBuild 專案的映像存取。如需詳細資訊，請參閱 Amazon ECR 範例 。	2022 年 3 月 10 日
已新增的區域支援	下列額外區域現在支援 ARM_CONTAINER 運算類型：亞太區域（首爾）、加拿大（中部）、歐洲（倫敦）和歐洲（巴黎）。如需詳細資訊，請參閱 建置環境運算類型 。	2022 年 3 月 10 日
新的 VPC 限制	如果您將 CodeBuild 設定為使用 VPC，則不支援本機快取。從 02/28/22 開始，您的 VPC 組建需要更長的時間，因為每個組建將使用新的 Amazon EC2 執行個體。	2022 年 2 月 25 日
批次報告模式	CodeBuild 現在可讓您選取批次建置狀態如何傳送至專案的來源提供者。如需詳細資訊，請參閱 批次報告模式 。	2021 年 10 月 4 日
新的運算類型	CodeBuild 現在支援小型 ARM 運算類型。如需詳細資訊，請參閱 建置環境運算類型 。	2021 年 9 月 13 日

公有建置專案	CodeBuild 現在可讓您將建置專案的建置結果提供給大眾，而不需要存取 AWS 帳戶。如需詳細資訊，請參閱 公有建置專案 。	2021 年 8 月 11 日
批次建置的工作階段偵錯	CodeBuild 現在支援批次建置的工作階段偵錯。如需詳細資訊，請參閱 建置圖表 和 建置清單 。	2021 年 3 月 3 日
專案層級並行建置限制	CodeBuild 現在可讓您限制組建專案的並行組建數量。如需詳細資訊，請參閱 專案組態 和 concurrentBuildLimit 。	2021 年 2 月 16 日
新的 buildspec 屬性：s3-prefix	CodeBuild 現在提供成品的 s3-prefix buildspec 屬性，可讓您為上傳至 Amazon S3 的成品指定路徑字首。如需詳細資訊，請參閱 s3 字首 。	2021 年 2 月 9 日
新的 buildspec 屬性：故障時	CodeBuild 現在提供建置階段的 on-failure buildspec 屬性，可讓您判斷建置階段失敗時會發生什麼情況。如需詳細資訊，請參閱 失敗時 。	2021 年 2 月 9 日
新的 buildspec 屬性：exclude-paths	CodeBuild 現在提供成品的 exclude-paths buildspec 屬性，可讓您從建置成品中排除路徑。如需詳細資訊，請參閱 exclude-paths 。	2021 年 2 月 9 日

[新的 buildspec 屬性：enable-symlinks](#)

CodeBuild 現在提供成品的 enable-symlinks buildspec 屬性，可讓您在 ZIP 成品中保留符號連結。如需詳細資訊，請參閱 [enable-symlinks](#)。

2021 年 2 月 9 日

[Buildspec 成品名稱增強功能](#)

CodeBuild 現在允許 artifacts/name 屬性包含路徑資訊。如需詳細資訊，請參閱 [名稱](#)。

2021 年 2 月 9 日

[程式碼涵蓋範圍報告](#)

CodeBuild 現在提供程式碼涵蓋範圍報告。如需詳細資訊，請參閱 [程式碼涵蓋範圍報告](#)。

2020 年 7 月 30 日

[批次建置](#)

CodeBuild 現在支援執行專案的並行和協調組建。如需詳細資訊，請參閱 [CodeBuild 中的批次建置](#)。

2020 年 7 月 30 日

[Windows Server 2019 映像](#)

CodeBuild 現在提供 Windows Server Core 2019 建置映像。如需詳細資訊，請參閱 [CodeBuild 提供的 Docker 映像](#)。

2020 年 7 月 20 日

[工作階段管理員](#)

CodeBuild 現在可讓您暫停執行中的組建，然後使用 AWS Systems Manager Session Manager 連線到組建容器並檢視容器的狀態。如需詳細資訊，請參閱 [工作階段管理員](#)。

2020 年 7 月 20 日

更新主題	CodeBuild 現在支援在 buildspec 檔案中指定要在其建置環境中使用的 Shell。如需詳細資訊，請參閱 建置規格參考 。	2020 年 6 月 25 日
使用測試架構測試報告	新增數個主題，說明如何使用數個測試架構產生 CodeBuild 測試報告。如需詳細資訊，請參閱 使用測試框架測試報告 。	2020 年 5 月 29 日
更新主題	CodeBuild 現在支援將標籤新增至報告群組。如需詳細資訊，請參閱 ReportGroup 。	2020 年 5 月 21 日
支援測試報告	CodeBuild 支援測試報告現已正式推出。	2020 年 5 月 21 日
更新主題	CodeBuild 現在支援為 Github 和 Bitbucket 建立 Webhook 篩選條件，該篩選條件僅在前端遞交訊息符合指定的表達式時觸發建置。如需詳細資訊，請參閱 GitHub 提取請求和 Webhook 篩選條件範例 和 Bitbucket 提取請求和 Webhook 篩選條件範例 。	2020 年 5 月 6 日
新主題	CodeBuild 現在支援共用建置專案和報告群組資源。如需詳細資訊，請參閱 使用共用的專案 和 使用共用的報告群組 。	2019 年 12 月 13 日
新的和更新的 主題	CodeBuild 現在支援在建置專案執行期間測試報告。如需詳細資訊，請參閱 使用測試報告 、 建立測試報告 和 使用 AWS CLI 範例建立測試報告 。	2019 年 11 月 25 日

更新主題	CodeBuild 現在支援 Linux GPU 和 Arm 環境類型，以及 2xlarge 運算類型。如需詳細資訊，請參閱 建置環境運算類型 。	2019 年 11 月 19 日
已更新主題	CodeBuild 現在支援所有組建的組建編號、匯出環境變數和 AWS Secrets Manager 整合。如需詳細資訊，請參閱 Buildspec 語法 中的 匯出變數 和 Secrets Manager 。	2019 年 11 月 6 日
新主題	CodeBuild 現在支援通知規則。您可以使用通知規則，向使用者通知建置專案中的重要變更。如需詳細資訊，請參閱 建立通知規則 。	2019 年 11 月 5 日
已更新主題	CodeBuild 現在支援 Android 29 版和 Go 1.13 版執行時間。如需詳細資訊，請參閱 CodeBuild 和 Buildspec 語法提供的 Docker 映像 。 https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html#build-spec-ref-syntax	2019 年 9 月 10 日
更新主題	建立專案時，您現在可以選擇 Amazon Linux 2 (AL2) 受管的映像。如需詳細資訊，請參閱 CodeBuild 提供的 Docker 映像 和 CodeBuild 的 buildspec 檔案範例中的執行期版本 。	2019 年 8 月 16 日

[更新主題](#)

建立專案時，您現在可以選擇停用 S3 日誌的加密，而且如果您使用以 Git 為基礎的來源儲存庫，請包括 Git 子模組。如需詳細資訊，請參閱在 [CodeBuild 中建立建置專案](#)。

2019 年 3 月 8 日

[新主題](#)

CodeBuild 現在支援本機快取。當您建立建置時，您可以從四種模式中的一或多種指定本機快取。如需詳細資訊，請參閱在 [CodeBuild 中建置快取](#)。

2019 年 2 月 21 日

[新主題](#)

CodeBuild 現在支援 Webhook 篩選群組，以指定觸發組建的事件。如需詳細資訊，請參閱 [篩選 GitHub Webhook 事件](#) 和 [篩選 Bitbucket Webhook 事件](#)。

2019 年 2 月 8 日

[新主題](#)

CodeBuild 使用者指南現在說明如何搭配代理伺服器使用 CodeBuild。如需詳細資訊，請參閱 [搭配代理伺服器使用 CodeBuild](#)。

2019 年 2 月 4 日

[更新主題](#)

CodeBuild 現在支援使用另一個 AWS 帳戶中的 Amazon ECR 映像。已更新數個主題以反映此變更，包括 [CodeBuild 的 Amazon ECR 範例](#)、[建立建置專案](#) 和 [建立 CodeBuild 服務角色](#)。

2019 年 1 月 24 日

[支援私有 Docker 登錄檔](#)

CodeBuild 現在支援使用存放在私有登錄檔中的 Docker 映像做為您的執行時間環境。如需詳細資訊，請參閱[具有 AWS Secrets Manager 範例的私有登錄檔](#)。

2019 年 1 月 24 日

[更新主題](#)

CodeBuild 現在支援使用存取字符連接到 GitHub（使用個人存取字符）和 Bitbucket（使用應用程式密碼）儲存庫。如需詳細資訊，請參閱[建立建置專案 \(主控台\)](#)和[使用存取字符搭配您的來源供應商](#)。

2018 年 12 月 6 日

[更新主題](#)

CodeBuild 現在支援測量組建中每個階段持續時間的新組建指標。如需詳細資訊，請參閱[CodeBuild CloudWatch 指標](#)。

2018 年 11 月 15 日

[VPC 端點政策主題](#)

CodeBuild 的 Amazon VPC 端點現在支援政策。如需詳細資訊，請參閱為[CodeBuild 建立 VPC 端點政策](#)。

2018 年 11 月 9 日

[已更新內容](#)

已更新主題以反映新的主控台體驗。

2018 年 10 月 30 日

[Amazon EFS 範例](#)

CodeBuild 可以在建置期間使用專案的 buildspec 檔案中的命令來掛載 Amazon EFS 檔案系統。如需詳細資訊，請參閱[CodeBuild 的 Amazon EFS 範例](#)。

2018 年 10 月 26 日

Bitbucket Webhook	當您為儲存庫使用 Bitbucket 時，CodeBuild 現在支援 Webhook。如需詳細資訊，請參閱 CodeBuild 的 Bitbucket 提取請求範例 。	2018 年 10 月 2 日
S3 日誌	CodeBuild 現在支援在 S3 儲存貯體中建置日誌。先前，您只能使用 CloudWatch Logs 建置日誌。如需詳細資訊，請參閱 建立專案 。	2018 年 9 月 17 日
多個輸入來源和多個輸出成品	CodeBuild 現在支援使用多個輸入來源並發佈多個成品集的專案。如需詳細資訊，請參閱 多個輸入來源和輸入成品範例 ，以及 與 CodeBuild 的 CodePipeline 整合，以及多個輸入來源和輸出成品範例 CodeBuild 。	2018 年 8 月 30 日
語意版本控制範例	CodeBuild 使用者指南現在提供以使用案例為基礎的範例，示範如何使用語意版本控制在建置時建立成品名稱。如需詳細資訊，請參閱 使用語意版本控制為建置成品範例命名 。	2018 年 8 月 14 日
新的靜態網站範例	CodeBuild 使用者指南現在提供以使用案例為基礎的範例，示範如何在 S3 儲存貯體中託管建置輸出。此範例利用對未加密建置成品的最新支援。如需詳細資訊，請參閱 使用在 S3 儲存貯體中託管的建置輸出建立靜態網站 。	2018 年 8 月 14 日

[支援使用語意版本控制覆寫成品名稱](#)

您現在可以使用語意版本控制來指定 CodeBuild 用於命名建置成品的格式。因為具有硬式編碼名稱的組建成品會覆寫先前使用相同硬式編碼名稱的組建成品，這很實用。例如，如果組建在某天觸發多次，您現在可以將時間戳記新增至成品名稱。每個組建成品名稱是唯一的，並且不會覆寫先前組建的成品。

2018 年 8 月 7 日

[支援未加密的建置成品](#)

CodeBuild 現在支援具有未加密建置成品的建置。如需詳細資訊，請參閱[建立建置專案 \(主控台\)](#)。

2018 年 7 月 26 日

[支援 Amazon CloudWatch 指標和警示](#)

CodeBuild 現在提供與 CloudWatch 指標和警示的整合。您可以使用 CodeBuild 或 CloudWatch 主控台，在專案和帳戶層級監控組建。如需詳細資訊，請參閱[監控建置](#)。

2018 年 7 月 19 日

[支援報告組建的狀態](#)

CodeBuild 現在可以向您的來源提供者報告組建的開始和完成狀態。如需詳細資訊，請參閱在 [CodeBuild 中建立建置專案](#)。

2018 年 7 月 10 日

[新增至 CodeBuild 文件的環境變數](#)

[建置環境中的環境變數](#)頁面已更新，加上 CODEBUILD_BUILD_ID、CODEBUILD_LOG_PATH 和 CODEBUILD_START_TIME 環境變數。

2018 年 7 月 9 日

[支援 buildspec 檔案中的 finally 區塊](#)

CodeBuild 文件已更新，其中包含 buildspec 檔案中選用 finally 區塊的詳細資訊。最後區塊中的命令一律會在其對應命令區塊中的命令之後執行。如需詳細資訊，請參閱 [Buildspec 語法](#)。

2018 年 6 月 20 日

[CodeBuild 代理程式更新通知](#)

CodeBuild 文件已更新，詳細說明如何在發行 CodeBuild 代理程式的新版本時，使用 Amazon SNS CodeBuild 接收通知。如需詳細資訊，請參閱 [接收新 AWS CodeBuild 代理程式版本的通知](#)。

2018 年 6 月 15 日

舊版更新

下表描述 2018 年 6 月前，每個 AWS CodeBuild 使用者指南版本的重要變更。

變更	描述	日期
支援 Windows 組建	CodeBuild 現在支援 Microsoft Windows Server 平台的建置，包括 Windows 上 .NET Core 2.0 的預先封裝建置環境。如需詳細資訊，請參閱 執行 CodeBuild 的 Microsoft Windows 範例 。	2018 年 5 月 25 日
支援組建幕等	當您使用 AWS Command Line Interface (AWS CLI) 執行 start-build 命令時，您可以指定建置為等幕。如需詳細資訊，請參閱 執行建置 (AWS CLI) 。	2018 年 5 月 15 日

變更	描述	日期
支援對更多組建專案設定的覆寫	您現在可以在建立組建時覆寫更多組建專案設定。覆寫僅針對該建置。如需詳細資訊，請參閱 手動執行 AWS CodeBuild 組建 。	2018 年 5 月 15 日
VPC 端點支援	您現在可以使用 VPC 端點來改善組建的安全性。如需詳細資訊，請參閱 使用 VPC 端點 。	2018 年 3 月 18 日
支援觸發	您現在可以建立觸發來以一般頻率排定建置。如需詳細資訊，請參閱 建立 AWS CodeBuild 觸發條件 。	2018 年 3 月 28 日
FIPS 端點文件	您現在可以了解如何使用 AWS Command Line Interface (AWS CLI) 或 AWS 開發套件，指示 CodeBuild 使用四個聯邦資訊處理標準 (FIPS) 端點的其中一個。如需詳細資訊，請參閱 指定 AWS CodeBuild 端點 。	2018 年 3 月 28 日
AWS CodeBuild 適用於亞太區域（孟買）、歐洲（巴黎）和南美洲（聖保羅）	AWS CodeBuild 現可於亞太區域（孟買）、歐洲（巴黎）和南美洲（聖保羅）區域使用。如需詳細資訊，請參閱 Amazon Web Services 一般參考中的 AWS CodeBuild 。	2018 年 3 月 28 日
GitHub Enterprise Server 支援	CodeBuild 現在可以從存放在 GitHub Enterprise Server 儲存庫中的原始程式碼建置。如需詳細資訊，請參閱 執行 GitHub Enterprise Server 範例 。	2018 年 1 月 25 日

變更	描述	日期
Git 複製深度支援	CodeBuild 現在支援建立淺複製，並將歷史記錄截斷為指定的遞交數量。如需詳細資訊，請參閱 建立組建專案 。	2018 年 1 月 25 日
VPC 支援	已啟用 VPC 的組建現在能夠存取 VPC 內的資源。如需詳細資訊，請參閱 VPC 支援 。	2017 年 11 月 27 日
相依性快取支援	CodeBuild 現在支援相依性快取。這可讓 CodeBuild 在快取中儲存建置環境的特定可重複使用部分，並在建置間使用此部分。	2017 年 11 月 27 日
組建識別證支援	CodeBuild 現在支援使用組建徽章，該徽章提供可嵌入、動態產生的映像（徽章），顯示專案的最新組建狀態。如需詳細資訊，請參閱 建置徽章範例 。	2017 年 11 月 27 日
AWS Config 整合	AWS Config 現在支援 CodeBuild 做為 AWS 資源，這表示服務可以追蹤您的 CodeBuild 專案。如需的詳細資訊 AWS Config，請參閱 AWS Config 範例 。	2017 年 10 月 20 日
自動重新組建 GitHub 儲存庫中更新的來源碼	如果您的來源碼存放在 GitHub 儲存庫，您可以讓 AWS CodeBuild 在每次將程式碼變更推送至儲存庫時重建來源碼。如需詳細資訊，請參閱 執行 GitHub 提取請求和 Webhook 篩選條件範例 。	2017 年 9 月 21 日

變更	描述	日期
在 Amazon EC2 Systems Manager 參數存放區中存放和擷取敏感或大型環境變數的新方法	您現在可以使用 AWS CodeBuild 主控台或 AWS CLI 來擷取存放在 Amazon EC2 Systems Manager 參數存放區中的敏感或大型環境變數。您現在可以使用 AWS CodeBuild 主控台，將這些類型的環境變數存放在 Amazon EC2 Systems Manager 參數存放區中。在過去，您只可以透過將環境變數併入 Buildspect 中或執行組建命令，才能自動化 AWS CLI來擷取這些類型的環境變數。您只能使用 Amazon EC2 Systems Manager 參數存放區主控台來存放這些類型的環境變數。如需詳細資訊，請參閱 建立組建專案 、 變更建置專案設定 和 手動執行組建 。	2017 年 9 月 14 日
組建刪除支援	您現在可以在 AWS CodeBuild 中刪除建置。如需詳細資訊，請參閱 刪除建置 。	2017 年 8 月 31 日

變更	描述	日期
更新使用 buildspec 擷取存放在 Amazon EC2 Systems Manager 參數存放區中的敏感或大型環境變數的方法	AWS CodeBuild 現在可讓您更輕鬆地使用 buildspec 擷取存放在 Amazon EC2 Systems Manager 參數存放區中的敏感或大型環境變數。在過去，您只可以透過執行組建命令，才能自動化 AWS CLI來擷取這些類型的環境變數。如需詳細資訊，請參閱 parameter-store 映射 Buildspec 語法 。	2017 年 8 月 10 日
AWS CodeBuild 支援 Bitbucket	CodeBuild 現在可以從存放在 Bitbucket 儲存庫中的原始程式碼建置。如需詳細資訊，請參閱 建立組建專案 及 手動執行組建 。	2017 年 8 月 10 日
AWS CodeBuild 在美國西部（加利佛尼亞北部）、歐洲（倫敦）和加拿大（中部）提供	AWS CodeBuild 現已在美國西部（加利佛尼亞北部）、歐洲（倫敦）和加拿大（中部）區域提供。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 AWS CodeBuild 。	2017 年 6 月 29 日
支援的替代 Buildspect 檔案名稱和位置	您現在可以指定要對組建專案使用的 Buildspect 檔案的替代檔案名稱或位置，而非位於來源碼根目錄名為 buildspec.yml 的預設 Buildspect 檔案。如需詳細資訊，請參閱 Buildspec 檔案名稱和儲存位置 。	2017 年 6 月 27 日

變更	描述	日期
更新的組建通知範例	CodeBuild 現在透過 Amazon CloudWatch Events 和 Amazon Simple Notification Service (Amazon SNS) 提供建置通知的內建支援。先前的 建置通知範例 已更新，以示範這個新行為。	2017 年 6 月 22 日
已新增自訂映像中的 Docker 範例	已新增範例，說明如何使用 CodeBuild 和自訂 Docker 組建映像來建置和執行 Docker 映像。如需詳細資訊，請參閱 自訂映像中的 Docker 範例 。	2017 年 6 月 7 日
擷取用於 GitHub 提取請求的來源碼	當您使用依賴存放在 GitHub 儲存庫中的原始程式碼的 CodeBuild 執行組建時，您現在可以指定要組建的 GitHub 提取請求 ID。您也可以改為指定遞交 ID、分支名稱或標籤名稱。如需詳細資訊，請參閱 中的來源版本值 執行建置 (主控台) 或 執行建置 (AWS CLI) 中的sourceVersion 值。	2017 年 6 月 6 日
組建規格版本已更新	已釋出新版本的 Buildspect 格式。0.2 版解決 CodeBuild 在預設 Shell 的個別執行個體中執行每個建置命令的問題。同時在 0.2 版中，environment_variables 重新命名為 env，而 plaintext 重新命名為 variables。如需詳細資訊，請參閱 CodeBuild 的建置規格參考 。	2017 年 5 月 9 日

變更	描述	日期
GitHub 中提供用於組建映像的 Dockerfile	提供的許多建置映像的定義 AWS CodeBuild 可在 GitHub 中做為 Dockerfiles 使用。如需詳細資訊，請參閱 CodeBuild 提供的 Docker 映像 中表格的「定義」一欄。	2017 年 5 月 2 日
AWS CodeBuild 在歐洲（法蘭克福）、亞太區域（新加坡）、亞太區域（雪梨）和亞太區域（東京）提供	AWS CodeBuild 現已在歐洲（法蘭克福）、亞太區域（新加坡）、亞太區域（雪梨）和亞太區域（東京）區域提供。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 AWS CodeBuild 。	2017 年 3 月 21 日
CodeBuild 的 CodePipeline 測試動作支援 CodeBuild	您現在可以使用 CodeBuild 將測試動作新增至 CodePipeline 中的管道。CodeBuild 如需詳細資訊，請參閱 將 CodeBuild 測試動作新增至管道 (CodePipeline 主控台) 。	2017 年 3 月 8 日
Buildspec 檔案支援從選取的上層目錄內擷取組建輸出	Buildspec 檔案現在可讓您指定個別最上層目錄，其內容可指示 CodeBuild 包含在組建輸出成品中。您可以使用 base-directory 映射來執行此動作。如需詳細資訊，請參閱 Buildspec 語法 。	2017 年 2 月 8 日

變更	描述	日期
內建的環境變數	AWS CodeBuild 提供額外的內建環境變數，供您的建置使用。這些包括描述開始組建的實體、來源碼儲存庫的 URL、來源碼的版本 ID 等等環境變數。如需詳細資訊，請參閱 建置環境中的環境變數 。	2017 年 1 月 30 日
AWS CodeBuild 美國東部（俄亥俄）提供	AWS CodeBuild 現已在美國東部（俄亥俄）區域提供。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 AWS CodeBuild 。	2017 年 1 月 19 日
Shell 和命令行為資訊	CodeBuild 會在建置環境預設 Shell 的個別執行個體中執行您指定的每個命令。此預設行為可能對您的命令產生非預期的副作用。我們建議了一些方法，可在需要時處理這個預設行為。如需詳細資訊，請參閱 建置環境中的 Shell 和命令 。	2016 年 12 月 9 日
環境變數資訊	CodeBuild 提供數個可在建置命令中使用的環境變數。您也可以定義自己的環境變數。如需詳細資訊，請參閱 建置環境中的環境變數 。	2016 年 7 月 12 日
故障診斷主題	故障診斷資訊現已提供。如需詳細資訊，請參閱 故障診斷 AWS CodeBuild 。	2016 年 12 月 5 日

變更	描述	日期
Jenkins 外掛程式初始版本	這是 CodeBuild Jenkins 外掛程式的初始版本。如需詳細資訊，請參閱 AWS CodeBuild 與 Jenkins 搭配使用 。	2016 年 12 月 5 日
使用者指南初始版本	這是 CodeBuild 使用者指南的初始版本。	2016 年 12 月 1 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。