使用者指南

AWS CodeBuild

API 版本 2016-10-06

AWS CodeBuild: 使用者指南

Table of Contents

什麼是 AWS CodeBuild？

AWS CodeBuild是雲端中完全受控的建置服務。 CodeBuild 編譯您的原始程式碼、執行單元測試，並產生準備好部署的成品。 CodeBuild 無需佈建、管理及擴充您自己的建置伺服器。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。您也可以在中自訂建置環境， CodeBuild 以使用您自己的建置工具。 CodeBuild 自動擴充以符合尖峰建置要求。

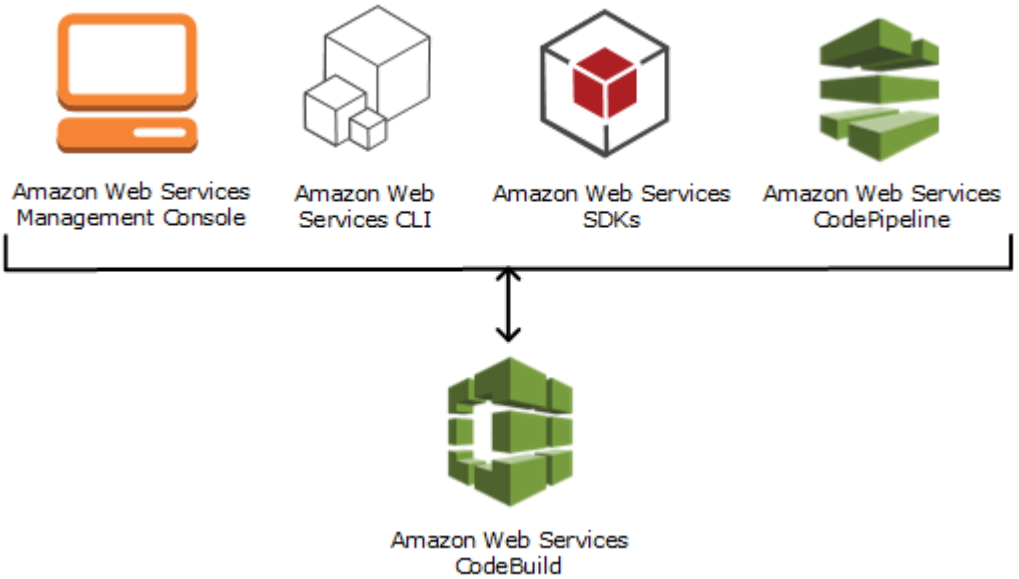
CodeBuild 提供以下好處：

- 完全受控 — CodeBuild 無需設定、修補、更新及管理自己的建置伺服器。
- 隨需 — 依需求 CodeBuild 擴充，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 — 為最流行的程式設計語言 CodeBuild 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。

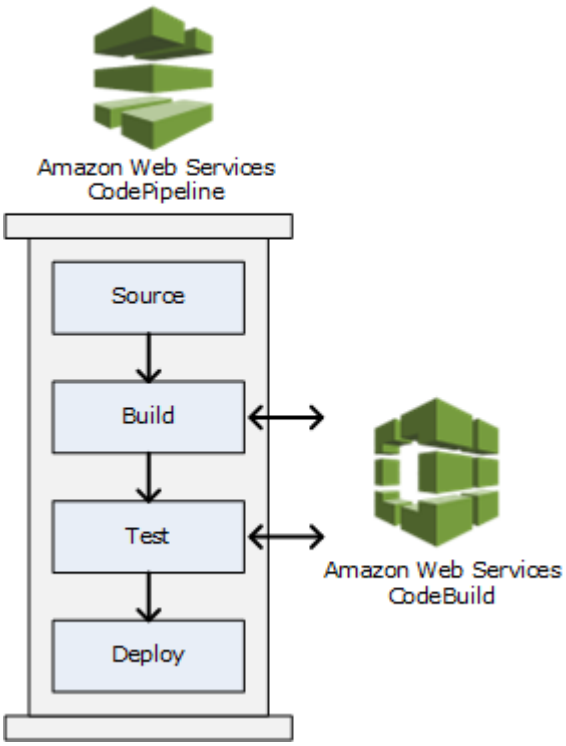
如需詳細資訊，請參閱 AWS CodeBuild。

如何運行 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 SDK CodeBuild 來自動執行。



如下圖所示，您可以將建置或測試動 CodeBuild 作新增至中管線的建置或測試階段AWS CodePipeline。 AWS CodePipeline是一種持續交付服務，可用於建立模型、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。



若要用 CodePipeline 來建立管線，然後新增 CodeBuild 建置或測試動作，請參閱 CodePipeline 搭配使用 CodeBuild。若要取得有關的更多資訊 CodePipeline，請參閱AWS CodePipeline使用者指南。

主 CodeBuild 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。

定價 CodeBuild

如需詳細資訊，請參閱CodeBuild 定價。

我該如何開始使用 CodeBuild？

建議您完成下列步驟：

1. 若要進一步瞭 CodeBuild 解，請閱讀中的資訊概念。

2. 依照 CodeBuild 中的指示在範例案例中進行實驗開始使用主控台。
3. 按照 CodeBuild 中的指示在您自己的案例中使用規劃組建。

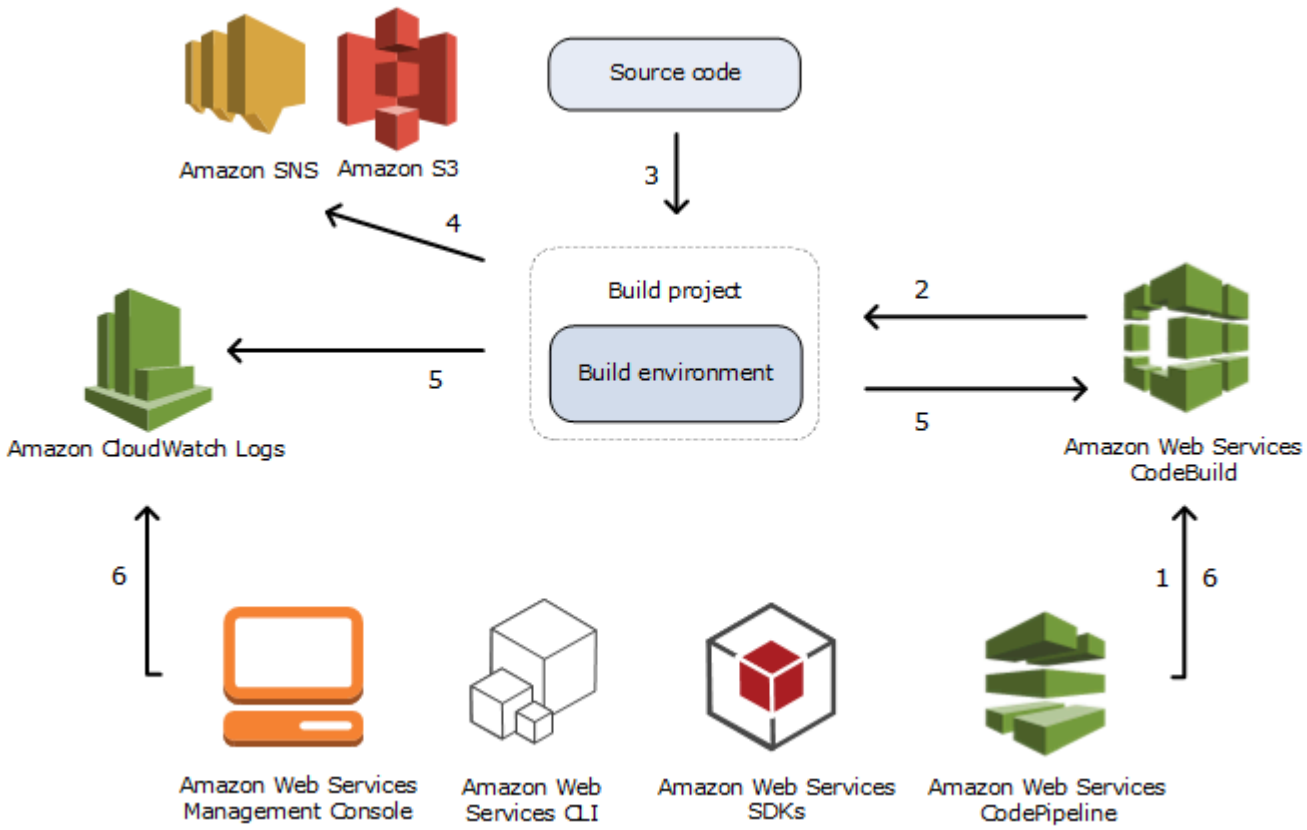
AWS CodeBuild 概念

以下是了解 CodeBuild 的運作方式的重要概念。

主題

CodeBuild 運作方式

下圖顯示使用 CodeBuild 執行組建時會發生什麼情況：



1. 作為輸入，您必須為 CodeBuild 提供組建專案。一個組建專案包含有關如何執行組建的資訊，其中包括來源碼位置、要使用哪個建置環境、要執行哪個建置命令、在何處存儲建置輸出。一個組建

環境表示 CodeBuild 用以執行組建的作業系統、程式語言執行時間及其他工具的組合。如需更多詳細資訊，請參閱：

- 建立組建專案
- 建置環境參考

2. CodeBuild 使用組建專案來建立組建環境。
3. CodeBuild 會將來源碼下載到組建環境，然後使用該組建規格 (buildspec，如組建專案中所定義)，或在來源碼中直接將其包含。一個組建規格是 CodeBuild 用以執行組建的一組組建命令與相關設定 (使用 YAML 格式)。如需詳細資訊，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。組建環境也可以執行您在組建規格中指定的任務 (例如，將組建通知傳送至 Amazon SNS 主題)。如需範例，請參閱 建置通知範例。
5. 組建執行時，組建環境會將資訊傳送至 CodeBuild 和 Amazon CloudWatch Logs。
6. 組建執行時，您可以使用AWS CodeBuild主控台、AWS CLI, 或AWS開發套件：從 CodeBuild 獲取摘要組建資訊，以及從 Amazon CloudWatch Logs 取得詳細建置資訊。如果您使用AWS CodePipeline來執行組建，則可以從 CodePipeline 獲取有限的組建資訊。

下一步驟

現在您已更加了解 AWS CodeBuild，建議您執行下列後續步驟：

1. 實驗在示例場景中使用 CodeBuild，請按照開始使用主控台。
2. 使用在您自己的案例中執行規劃組建。

開始使用

在下列教學課程中，您會使用 AWS CodeBuild 將範例來源碼輸入檔案的集合建置到可部署的來源碼版本。

兩個教學課程具有相同的輸入和結果，但一個使用 AWS CodeBuild 主控台，另一個則使用 AWS CLI。

> **Important**
>
> 不建議使用 AWS 根帳戶來完成本教學課程。

使用主控台開始使用 AWS CodeBuild

在本教學課程中，您會使用 AWS CodeBuild 將一組範例來源碼輸入檔案 (建置輸入成品或建置輸入) 建置成可部署版本的來源碼 (建置輸出成品或建置輸出)。具體來說，您指示 CodeBuild 使用 Apache Maven，一種通用的構建工具，將一組 Java 類文件構建到 Java 存檔（JAR）文件中。您不需要熟悉 Apache Maven 或 Java，也能完成本教學課程。

您可以透 CodeBuild 過主 CodeBuild 控台AWS CodePipeline、AWS CLI、或 AWS SDK 使用。本教程演示如何使用控 CodeBuild 制台。如需使用 CodePipeline 的相關資訊，請參閱 CodePipeline 搭配使用 CodeBuild。

> **Important**
>
> 本教學課程中的步驟需要您建立可能會對 AWS 帳戶產生費用的資源 (例如 S3 儲存貯體)。其中包括 CodeBuild 與 Amazon S3 相關的AWS資源和動作以及 CloudWatch 日誌可能的費用。AWS KMS如需詳細資訊，請參閱AWS CodeBuild定價、Amazon S3 定AWS Key Management Service價、定價和 Amazon CloudWatch 定價。

步驟

- 步驟 1：建立原始程式碼
- 第 2 步：創建構建規格文件
- 步驟 3：建立兩個 S3 儲存貯體

步驟 1：建立原始程式碼

(部分：使用主控台開始使用 AWS CodeBuild)

在此步驟中，您會建立要建置 CodeBuild 到輸出值區的原始程式碼。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
```

下一步驟

[第 2 步：創建構建規格文件]

第 2 步：創建構建規格文件

(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 是建置命令和相關設定的集合 (YAML 格式)， CodeBuild 用來執行組建。如果沒有組建規格，就 CodeBuild 無法成功將組建輸入轉換為組建輸出，或在建置環境中找到組建輸出成品，以便上傳至輸出值區。

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` 代表您可以指示 CodeBuild 執行命令的建置階段期間。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 執行`mvn install`指令。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。當您在本教學課程後文檢視詳細建置資訊時，這些 `echo` 命令的輸出可協助您更進一步了解 CodeBuild 執行命令的方式及順序。(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。) 對於每個構建階段，從開始到結束 CodeBuild 運行每個指定的命令，按列出的順序一次執行一個命令。
- `artifacts`代表上 CodeBuild載至輸出值區的組建輸出成品集。 `files`代表要包含在組建輸出中的檔案。 CodeBuild 上傳在構建環境中`target`相對目錄中找到的單個`messageUtil-1.0.jar`文件。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
```

下一步驟

步驟 3：建立兩個 S3 儲存貯體

步驟 3：建立兩個 S3 儲存貯體

(上一個步驟：第 2 步：創建構建規格文件)

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* 是儲存貯體的 AWS 區域，*account-ID* 則是您的 AWS 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位於與您組建相同的 AWS 區域內。例如，如果您指示 CodeBuild 在美國東部 (俄亥俄) 區域執行組建，這些值區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 儲存庫中的組建輸入，但本教學課程並未向您展示如何使用它們。如需詳細資訊，請參閱 規劃組建。

下一步驟

步驟 4：上傳原始程式碼和 Buildspec 檔案

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 存儲庫，按照慣例，您必須存儲在每個存儲庫的根目錄（頂層）`buildspec.yml`中命名的構建規格文件，或將構建規範聲明作為構建項目定義的一部分包含在內。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。

下一步驟

步驟 5：建立建置專案

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 用來執行組建的組建專案。組建專案包含如何執行組建的相關資訊，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，以及儲存組建輸出的位置。建置環境代表作業系統、程式設計語言執行階段，以及 CodeBuild 用來執行組建的工具的組合。構建環境以 Docker 映像表示。如需詳細資訊，請參閱 Docker Docs 網站上的 Docker 概觀主題。

對於這個構建環境，您指示 CodeBuild 使用包含 Java 開發工具包（JDK）和 Apache Maven 版本的碼頭映像。

建立建置專案

1. 請登入AWS Management Console並開啟AWS CodeBuild主控台，網址為 https://console.aws.amazon.com/codesuite/codebuild/home。
2. 使用AWS區域選擇器來選擇支援 CodeBuild 的AWS地區。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，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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 帳戶中都必須是唯一的。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權限建立建置專案的使用者AWS Management Console身分登入。若要修正此問題，請登出AWS Management Console，然後使用屬於下列其中一個 IAM 實體的認證重新登入：
>
> - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的「建立您的第一個 AWS 帳戶 root 使用者和群組」。
> - 您AWS帳戶中的使用者 `AWSCodeBuildAdminAccess`、`AmazonS3ReadOnlyAccess`，且`IAMFullAccess`受管政策附加至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權限的使用者或群組，而且無法將這些權限新增至您的使用者或群組，請聯絡您的AWS帳戶管理員以尋求協助。如需詳細資訊，請參閱 AWS 的管理 (預先定義) 策略 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 權限參考。

5. 在來源中，針對來源供應商，選擇 Amazon S3。
6. ***############-## ID-## ID-######***
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 Operating system (作業系統)，請選擇 Amazon Linux 2。
10. 針對 Runtime(s) (執行時間)，選擇 Standard (標準)。
11. 對於圖像，請選擇 AWS /代碼構建/ 亞馬遜鏈 2-x86_64 標準：4.0。
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 Buildspec，將 Use a buildspec file (使用 buildspec 檔案) 維持在選取狀態。
14. 在成品中，對於類型，選擇 Amazon S3。
15. ***##################-### ID#-### ID#-#######***
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)。

下一步驟

步驟 6：執行建置

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，您將指示 AWS CodeBuild 使用組建專案中的設定執行組建。

執行建置

1. 請在以下位置開啟AWS CodeBuild主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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 [開始組建]。建置會立即開始。

下一步驟

步驟 7：檢視摘要建置資訊

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<build-ID>未顯示codebuild-demo-project:頁面，請在導覽列中選擇 [建置歷程記錄]。接下來，在組建專案清單中，針對 Project，選擇的 [建置執行] 連結codebuild-demo-project。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] 頁面的 [階段詳細資料] 中，應該會顯示下列建置階段，且 [狀態] 資料行中的 [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。

下一步驟

步驟 8：檢視詳細建置資訊

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，您可以在 CloudWatch 日誌中查看有關構建的詳細信息。

> **Note**
>
> 為了保護敏感資訊，記 CodeBuild 錄檔中會隱藏下列項目：
>
> - AWS 存取金鑰 ID。如需詳細資訊，請參閱使用AWS Identity and Access Management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 使用指南中的 Systems Manager 參數存放區和系統管理員參數存放主控台逐步解說。

- 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 記錄檔中查看整個組建記錄，請選擇 [檢視整個記錄檔] 連結。
2. 在 CloudWatch 記錄檔資料流中，您可以瀏覽記錄事件。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，大多數的日誌事件都包含 CodeBuild 下載及將建置依存性檔案安裝到建置環境的詳細資訊，而您可能不太重視這類資訊。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，則只會顯示包含[INFO]的事件。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的篩選器和模式語法。

下一步驟

步驟 9：取得建置輸出成品

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 建置並上傳至輸出值區的`messageUtil-1.0.jar`檔案。

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

取得建置輸出成品 (AWS CodeBuild 主控台)

1. 在 CodeBuild 主控台仍然開啟的情況下，並且上一步仍顯示組建詳細資料頁面，請選擇 [組建詳細資料] 索引標籤，然後向下捲動至 [成品] 區段。

> **Note**
>
> 如果未顯示組建詳細資料頁面，請在導覽列中選擇 [組建歷程記錄]，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。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。

獲取構建輸出工件（Amazon S3 控制台）

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (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。

下一步驟

步驟 10：刪除 S3 儲存貯體

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)

為了避免AWS帳戶持續收費，您可以刪除本教學課程中使用的輸入和輸出值區。如需指示，請參閱 Amazon 簡易儲存服務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 使用者刪除這些值區，則該使用者必須擁有更多存取權限。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。

```
{
  "Version": "2012-10-17",
  "Id": "...",
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
```

```
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
    ### END ADDING STATEMENT HERE ###
  ]
}
```

下一步驟

包裝

包裝

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

您現在可以嘗試在自己的情況下使用 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。如需更多詳細資訊，請參閱 範例。

使用 AWS CLI 開始使用 AWS CodeBuild

在本教學課程中，您會使用 AWS CodeBuild 將一組範例來源碼輸入檔案 (稱為建置輸入成品或建置輸入) 建置成可部署版本的來源碼 (稱為建置輸出成品或建置輸出)。具體來說，您指示 CodeBuild 使用 Apache Maven，一種通用的構建工具，將一組 Java 類文件構建到 Java 存檔（JAR）文件中。您不需要熟悉 Apache Maven 或 Java，也能完成本教學課程。

您可以透 CodeBuild 過主 CodeBuild 控台AWS CodePipeline、AWS CLI、或 AWS SDK 使用。本自學課程將示範如何 CodeBuild 與AWS CLI. 如需使用的資訊 CodePipeline，請參閱 CodePipeline 搭配使用 CodeBuild。

> **Important**
>
> 本教學課程中的步驟需要您建立可能會對 AWS 帳戶產生費用的資源 (例如 S3 儲存貯體)。其中包括 CodeBuild 與 Amazon S3 相關的AWS資源和動作以及 CloudWatch 日誌可能的費用。AWS KMS如需詳細資訊，請參閱CodeBuild定價、Amazon S3 定AWS Key Management Service價、定價和 Amazon CloudWatch 定價。

步驟

步驟 1：建立原始程式碼

(部分：使用 AWS CLI 開始使用 AWS CodeBuild)

在此步驟中，您會建立要建置 CodeBuild 到輸出值區的原始程式碼。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
```

下一步驟

[第 2 步：創建構建規格文件]

第 2 步：創建構建規格文件

(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 是建置命令和相關設定的集合 (YAML 格式)， CodeBuild 用來執行組建。如果沒有組建規格，就 CodeBuild 無法成功將組建輸入轉換為組建輸出，或在建置環境中找到組建輸出成品，以便上傳至輸出值區。

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` 代表您可以指示 CodeBuild 執行命令的建置階段期間。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 執行`mvn install`指令。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。當您在本教學課程後文檢視詳細建置資訊時，這些 `echo` 命令的輸出可協助您更進一步了解 CodeBuild 執行命令的方式及順序。(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。) 對於每個構建階 CodeBuild 段，按照列出的順序從頭到尾一次運行一個指定的命令。
- `artifacts`代表上 CodeBuild載至輸出值區的組建輸出成品集。 `files`代表要包含在組建輸出中的檔案。 CodeBuild 上傳在構建環境中`target`相對目錄中找到的單個`messageUtil-1.0.jar`文件。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
```

下一步驟

步驟 3：建立兩個 S3 儲存貯體

步驟 3：建立兩個 S3 儲存貯體

(上一個步驟：第 2 步：創建構建規格文件)

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`* 是儲存貯體的 AWS 區域，*`account-ID`* 則是您的 AWS 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位於與您組建相同的 AWS 區域內。例如，如果您指示 CodeBuild 在美國東部 (俄亥俄) 區域執行組建，這些值區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 儲存庫中的組建輸入，但本教學課程並未向您展示如何使用它們。如需詳細資訊，請參閱 規劃組建。

下一步驟

步驟 4：上傳原始程式碼和 Buildspec 檔案

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 存儲庫，按照慣例，您必須存儲在每個存儲庫的根目錄（頂層）`buildspec.yml`中命名的構建規格文件，或將構建規範聲明作為構建項目定義的一部分包含在內。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。

下一步驟

步驟 5：建立建置專案

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 用來執行組建的組建專案。組建專案包含如何執行組建的相關資訊，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，以及儲存組建輸出的位置。建置環境代表作業系統、程式設計語言執行階段，以及 CodeBuild 用來執行組建的工具的組合。構建環境以 Docker 映像表示。如需詳細資訊，請參閱 Docker Docs 網站上的 Docker 概觀主題。

對於這個構建環境，您指示 CodeBuild 使用包含 Java 開發工具包（JDK）和 Apache Maven 版本的碼頭映像。

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。將資料複製到本機電腦或執行個體上 AWS CLI 安裝位置中名為 `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"
}
```

以 service *IAMRole* 的 Amazon 資源名稱 (ARN) 取代 CodeBuild 服務角色 (例如)。arn:aws:iam::*account-ID*:role/*role-name*若要建立服務角色，請參閱建立 CodeBuild 服務角色。

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是代表原始程式碼儲存庫類型的必要值 (在此範例中，S3針對 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存放庫類型的必要值 (在此範例中，S3針對 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是代表此構建項目使用的 Docker 映像名稱和標籤組合的必要值，由 Docker 映像存儲庫類型指定 (在本示例中，aws/codebuild/standard:5.0對於 Docker 映像存儲庫中的 Docker 映像庫中的 Doc CodeBuild ker 映像) 。 aws/codebuild/standard是碼頭圖像的名稱。 5.0是碼頭圖像的標籤。

  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是代表運算資源 CodeBuild 使用的必要值 (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`代表如果組建尚未完成，則會在這段時間後 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在建置期間使 CodeBuild 用的環境變數。
  - `encryptionKey`代表用來加密組建輸出成品之客戶管理金鑰的 ARN。 CodeBuild
  - `arn` 代表組建專案的 ARN。

> **Note**
>
> 執行create-project命令之後，可能會輸出類似下列內容的錯誤訊息：使用***#:User arn*** 無法執行:codebuild:. CreateProject 這很可能是因為您使用AWS CLI沒有足夠權限可用來建立組建專案的使用者認證 CodeBuild 進行設定。若要修正此問題，請使AWS CLI用屬於下列其中一個 IAM 實體的登入資料進行設定：
>
> - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的「建立您的第一個 AWS 帳戶 root 使用者和群組」。
> - 您AWS帳戶中的使用者 `AWSCodeBuildAdminAccess`、`AmazonS3ReadOnlyAccess`，且`IAMFullAccess`受管政策附加至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權限的使用者或群組，而且無法將這些權限新增至您的使用者或群組，請聯絡您的AWS帳戶管理員以尋求協助。如需詳細資訊，請參閱 AWS 的管理 (預先定義) 策略 AWS CodeBuild。

下一步驟

步驟 6：執行建置

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，您將指示 AWS CodeBuild 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。

下一步驟

步驟 7：檢視摘要建置資訊

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* 替換id為上一步輸出中顯示的值。

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`代表在構建過程中 CodeBuild 運行的一組構建階段。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 日誌中有關組建日誌的資訊。
  - `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 [屬性] 窗格中，展開 [中繼資料]，然後檢視 [內容-x-amz-meta-codebuildmd5] 和 [內容- sha256] 的值。x-amz-meta-codebuild(在 Amazon S3 主控台中，不應將建置輸出成品的 ETag 值解譯為 MD5 或 SHA-256 雜湊值。)

若您使用 AWS 開發套件取得這些雜湊，則這些值會命名為 `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 中繼資料有一個名為的 CodeBuild 標頭，`x-amz-meta-codebuild-buildarn`其`buildArn`中包含將成品發佈到 Amazon S3 的 CodeBuild 組建。`buildArn`已新增以允許來源追蹤通知，以及參考從中產生成品的組建。

下一步驟

步驟 8：檢視詳細建置資訊

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，您可以在 CloudWatch 日誌中查看有關構建的詳細信息。

> **Note**
>
> 為了保護敏感資訊，記 CodeBuild 錄檔中會隱藏下列項目：
>
> - AWS 存取金鑰 ID。如需詳細資訊，請參閱使用AWS Identity and Access Management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 使用指南中的 Systems Manager 參數存放區和系統管理員參數存放主控台逐步解說。
> - 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 記錄檔資料流中，您可以瀏覽記錄事件。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，大多數的日誌事件都包含 CodeBuild 下載及將建置依存性檔案安裝到建置環境的詳細資訊，而您可能不太重視這類資訊。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，則只會顯示包含[INFO]的事件。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的篩選器和模式語法。

CloudWatch 記錄檔資料流的這些部分與本教學課程有關。

```
...
[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 成功完成建置前、建置和建置後的建置階段。它執行了單位測試並成功建置了 `messageUtil-1.0.jar` 檔案。

下一步驟

步驟 9：取得建置輸出成品

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 建置並上傳至輸出值區的`messageUtil-1.0.jar`檔案。

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

取得建置輸出成品 (AWS CodeBuild 主控台)

1. 在 CodeBuild 主控台仍然開啟的情況下，並且上一步仍顯示組建詳細資料頁面，請選擇 [組建詳細資料] 索引標籤，然後向下捲動至 [成品] 區段。

> **Note**
>
> 如果未顯示組建詳細資料頁面，請在導覽列中選擇 [組建歷程記錄]，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。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。

獲取構建輸出工件（Amazon S3 控制台）

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (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。

下一步驟

步驟 10：刪除 S3 儲存貯體

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)

為了避免AWS帳戶持續收費，您可以刪除本教學課程中使用的輸入和輸出值區。如需指示，請參閱 Amazon 簡易儲存服務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 使用者刪除這些值區，則該使用者必須擁有更多存取權限。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。

```
{
  "Version": "2012-10-17",
  "Id": "...",
```

```
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
    ### END ADDING STATEMENT HERE ###
  ]
}
```

下一步驟

包裝

包裝

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

您現在可以嘗試在自己的情況下使用 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。如需更多詳細資訊，請參閱 範例。

CodeBuild 樣本

這些樣本組可用於實驗 AWS CodeBuild：

主題

使用以案例為基礎的範例 CodeBuild

您可以使用這些基於案例的 AWS CodeBuild示例進行實驗：

跨服務範例

要試驗的跨服務範例清單。 AWS CodeBuild

建置徽章範例

說明如何 CodeBuild 使用組建徽章進行設定。

使用 AWS CLI 範例建立測試報告

使用 AWS CLI 建立、執行和檢視測試報告的結果。

碼頭工人樣品 CodeBuild

示範如何使用自訂 Docker 映像、將 Docker 映像發佈到 Amazon ECR 中的儲存庫，以及如何在私有登錄中使用 Docker 映像檔。

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。

多個輸入來源和輸出成品範例

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。

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

說明如何在 CodeBuild 組建專案中使用特定版本的原始碼。

[的第三方來源儲存庫範例 CodeBuild]

示範如何使 CodeBuild用 Webhook 建立 BitBucket、 GitHub 企業伺服器及 GitHub 提取要求。

[使用語意版本控制來命名建置成品範例]

示範在建置時如何使用語意版本控制來建立成品名稱。

跨服務範例 CodeBuild

您可以使用這些跨服務範例進行試驗： AWS CodeBuild

[Amazon ECR 樣本]

在 Amazon ECR 存儲庫中使用碼頭映像來使用阿帕奇 Maven 生成一個 JAR 文件。

[Amazon EFS 樣本]

說明如何設定建置規格檔案，讓 CodeBuild 專案掛載並建置在 Amazon EFS 檔案系統上。

[AWS CodePipeline 樣本]

說明如 AWS CodePipeline 何使用建立含有批次組建的組建，以及多個輸入來源和多個輸出成品。

[AWS Config 樣本]

顯示如何設定 AWS Config。列出要追蹤的 CodeBuild 資源，並說明如何在中查詢 CodeBuild 專案 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。

Amazon ECR 樣品 CodeBuild

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

> **Important**
>
> 執行此範例可能會導致您的 AWS 帳戶收取費用。其中包括與 Amazon S3、 AWS KMS CloudWatch 日誌和 Amazon ECR 相關的 AWS 資源和動作的可能收費和費用。 AWS

CodeBuild 如需詳細資訊，請參閱CodeBuild 定價、Amazon S3 定AWS Key Management Service價、定價、Amazon CloudWatch 定價和 Amazon 彈性容器登錄定價。

執行範例

如何執行此範例

1. 若要在 Amazon ECR 中建立 Docker 映像檔並將其推送至您的映像儲存庫，請完成「執行範例」一節中的步驟。將碼頭映像發佈到 Amazon ECR 映像儲存庫範例
2. 建立 Go 專案：

   a. 依照本主題Go 專案結構和Go 專案檔案章節中所述建立檔案，然後將它們上傳到 S3 輸入儲存貯體或 AWS CodeCommit GitHub、或 Bitbucket 存放庫。

   > **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 建立組建專案，`create-project`指令的 JSON 格式輸入可能會類似於此。(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. 如果下列其中一項成立，則必須將許可新增至 Amazon ECR 中的映像儲存庫， AWS CodeBuild 以便將其 Docker 映像提取到建置環境中。

   - 您的專案會使用 CodeBuild 登入資料來提取 Amazon ECR 映像檔。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 映像檔。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 映像檔，對於AWS 帳戶 ID，請輸入您要授予存取權的 AWS 帳戶 ID。
   7. 略過 All IAM entities (所有 IAM 實體) 清單。
   8. 針對「動作」，選取僅提取動作：ecr: GetDownloadUrlForLayer、ecr: BatchGetImage 和 ecr:。BatchCheckLayerAvailability
   9. 對於「條件」，請新增下列項目：

```
{
   "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 登入資料提取 Amazon ECR 映像，則"codebuild.amazonaws.com"會出現在服務主體下。
- 如果您的專案使用跨帳戶 Amazon ECR 映像檔，您要授予存取權的 AWS 帳戶 ID 會顯示在「AWS 帳戶 ID」下方。

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

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccessPrincipal",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
         ],
         "Condition":{
            "StringEquals":{
               "aws:SourceArn":"arn:aws:codebuild:<region>:<aws-account-
id>:project/<project-name>",
               "aws:SourceAccount":"<aws-account-id>"
            }
         }
      },
```

```
      {
         "Sid":"CodeBuildAccessCrossAccount",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::<AWS-account-ID>:root"
         },
         "Action":[
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
         ]
      }
   ]
}
```

- 如果您的專案使用 CodeBuild 登入資料，而且您希望 CodeBuild 專案擁有 Amazon ECR 儲存庫的開放存取權，則可以省略這些Condition金鑰並新增下列範例政策。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CodeBuildAccessPrincipal",
      "Effect":"Allow",
      "Principal":{
        "Service":"codebuild.amazonaws.com"
      },
      "Action":[
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    },
    {
      "Sid":"CodeBuildAccessCrossAccount",
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::<AWS-account-ID>:root"
      },
      "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 建立組建專案，`create-project`指令的 JSON 格式輸入可能會類似於此。(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)
}
```

相關資源

- 如需開始使用的相關資訊 AWS CodeBuild，請參閱使用主控台開始使用 AWS CodeBuild。

- 如需有關疑難排解中問題的資訊 CodeBuild，請參閱疑難排 AWS CodeBuild。
- 如需中配額的相關資訊 CodeBuild，請參閱AWS CodeBuild 的配額。

Amazon Elastic File System 範例 AWS CodeBuild

您可能想要在 Amazon Elastic File System 上 AWS CodeBuild 建立組建，這是一種適用於 Amazon EC2 執行個體的可擴展共用檔案服務。Amazon EFS 的儲存容量具有彈性，因此會隨著新增和移除檔案而增加或縮小。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？ 在 Amazon Elastic File System 使用者指南中。

此範例說明如何設定 CodeBuild 專案以便將 Java 應用程式掛接到 Amazon EFS 檔案系統，然後建立一個 Java 應用程式。在開始之前，您必須準備好要建置的 Java 應用程式，該應用程式已上傳到 S3 輸入儲存貯體或 AWS CodeCommit、 GitHub、 GitHub 企業伺服器或 Bitbucket 存放庫。

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。

高階步驟

本範例涵蓋將 Amazon EFS 搭配使用所需的三個高階步驟 AWS CodeBuild：

1. 在您的 AWS 帳戶中建立虛擬私有雲 (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。該 CodeBuild 項目使用以下內容來識別文件系統：
    - 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，就會顯示 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。

> **Note**
>
> 只有在 Linux 平台上才支援在 Amazon EFS 中建立的檔案系統。

使用建立 VPC AWS CloudFormation

使用 AWS CloudFormation 範本建立您的 VPC。

1. 按照中的AWS CloudFormation VPC 範本說明 AWS CloudFormation 建立 VPC。

   > **Note**
   >
   > 此 AWS CloudFormation 範本建立的 VPC 具有兩個私有子網路和兩個公用子網路。只有在用於掛接在 Amazon EFS 中建立的檔案系統時，才能使用 AWS CodeBuild 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 Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
3. 選擇您使用 AWS CloudFormation建立的 VPC。
4. 在 Description (描述) 標籤上，記下 VPC 的名稱及其 ID。當您稍後在本示例中創建 AWS CodeBuild 項目時，兩者都是必需的。

使用您的 VPC 建立 Amazon 彈性檔案系統檔案系統

使用您之前建立的 VPC 為此範例建立簡單的 Amazon EFS 檔案系統。

1. 登入 AWS Management Console 並開啟 Amazon EFS 主控台，網址為 https://console.aws.amazon.com/efs/。
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，對於預設名稱金鑰的值中，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. (選擇性) 建議您在 Amazon EFS 檔案系統新增政策，以強制對傳輸中的資料加密。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]。

建立要搭配 Amazon EFS 使用的 CodeBuild 專案

AWS CodeBuild 建立使用您先前在此範例中建立的 VPC 的專案。執行組建時，它會掛接先前建立的 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. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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. 輸入用於尋找應 CodeBuild 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. 從映像中，選擇 AWS /代碼構建/ 亞馬遜鏈 2-x86_64 標準：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，輸入為您 CodeBuild 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**
    >
    > 依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 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。您必須在掛接 Amazon EFS 檔案系統的組建中使用私有子網路。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 使用此識別碼建立可識別彈性檔案系統的環境變數。環境變量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。
- (選擇性) 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，則會 CodeBuild掛載整個檔案系統。此路徑相對於檔案系統的根目錄。
- 對於掛載點，請輸入掛載檔案系統之組建容器中目錄的絕對路徑。如果此目錄不存在，請在建置期間建 CodeBuild 立該目錄。
- (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 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. 在編輯器中輸入以下構建規範命令。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 -Dgpg.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)。

CodeBuild 和 Amazon EFS 範例摘要

在您的 AWS CodeBuild 項目構建後：

- 您有一個由 Java 應用程式建立的 .jar 檔案，該檔案是在掛載點目錄下的 Amazon EFS 檔案系統中建立的。
- 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 彈性檔案系統使用者指南中的掛接檔案系統。

故障診斷

以下是您在使用設定 Amazon EFS 時可能會遇到的錯誤 CodeBuild。

主題

- 客戶端錯誤：掛載 '127.0.0.1：/'失敗。權限被拒絕
- 客戶端錯誤：掛載 '127.0.0.1：/'失敗。通過對等方重置連接
- 客戶端錯誤：意外的 EC2 錯誤： UnauthorizedOperation

客戶端錯誤：掛載 '127.0.0.1：/'失敗。權限被拒絕

使用掛接 Amazon EFS 時不支援 IAM 授權 CodeBuild。如果您使用自訂的 Amazon EFS 檔案系統政策，則需要授與所有 IAM 主體的讀取和寫入存取權限。例如：

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

客戶端錯誤：掛載 '127.0.0.1：/'失敗。通過對等方重置連接

此錯誤有兩個可能的原因：

- 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。您可以透過新增輸入規則來允許來自 VPC (為 VPC 新增主要 CIDR 區塊) 或安全性群組本身的所有流量來解決此問題。

客戶端錯誤：意外的 EC2 錯誤： UnauthorizedOperation

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

CodePipeline 樣品 CodeBuild

主題

- AWS CodePipeline 與整合 CodeBuild 和批次建置
- AWS CodePipeline 與多個輸入來源 CodeBuild 和輸出假影範例整合

AWS CodePipeline 與整合 CodeBuild 和批次建置

AWS CodeBuild 現在支援批次建置。此範例示範如 AWS CodePipeline 何使用建立使用批次建置的建置專案。

您可以使用 JSON 格式的檔案 AWS CLI 來定義管線的結構，然後將它與建立管線搭配使用。如需詳細資訊，請參閱AWS CodePipeline 使用指南中的AWS CodePipeline 配管結構參考。

使用個別成品 Batch 建置

使用下列 JSON 檔案做為管線結構的範例，該管線結構可建立具有不同成品的批次建置。若要啟用批次建置 CodePipeline，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
  }
}
```

以下是可搭配此管線組態使用的 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
  secondary-artifacts:
    artifact1:
      files:
        - output_file
    artifact2:
      files:
        - output_file
```

管道的 JSON 檔案中指定的輸出成品名稱必須與 buildspec 檔案中定義的組建和成品的識別碼相符。########################## _ ########

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的次要工件上傳build1到的位置build1_artifact1，依此類推。如果只指定一個輸出位置，則該名稱應該只是####碼。

建立 JSON 檔案之後，您可以建立管道。使用執 AWS CLI 行建立管線命令，並將檔案傳遞給參數。--cli-input-json如需詳細資訊，請參閱AWS CodePipeline 使用指南中的建立管線 (CLI)。

使用組合成品的 Batch 建置

使用下列 JSON 檔案做為管線結構的範例，該管線結構可建立包含合併成品的批次建置。若要啟用批次建置 CodePipeline，請將configuration物件的BatchEnabled參數設定為true。若要將組建加工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
 }
}
```

以下是可搭配此管線組態使用的 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
```

如果已針對批次建置啟用合併的加工品，則僅允許一個輸出。 CodeBuild 將所有構建的主要工件合併到一個 ZIP 文件中。

建立 JSON 檔案之後，您可以建立管道。使用執 AWS CLI 行建立管線命令，並將檔案傳遞給參數。`--cli-input-json`如需詳細資訊，請參閱AWS CodePipeline 使用指南中的建立管線 (CLI)。

AWS CodePipeline 與多個輸入來源 CodeBuild 和輸出假影範例整合

一個 AWS CodeBuild 項目可以採用多個輸入源。也可以建立不只一個輸出成品。此範例示範如 AWS CodePipeline 何使用建立組建專案，該專案使用多個輸入來源建立多個輸出成品。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 JSON 格式的檔案 AWS CLI 來定義管線的結構，然後將它與建立管線搭配使用。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 使用指南》中的CodePipeline 配管結構參考。

```
{
 "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",
    "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。此源代碼是查 CodeBuild 找和運行 buildspec 文件的目錄。關鍵字用PrimarySource於在 JSON 檔案的階段configuration區 CodeBuild 段中指定主要來源。
- 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環境變數，$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。使用執 AWS CLI 行建立管線命令，並將檔案傳遞給參數。`--cli-input-json`如需詳細資訊，請參閱AWS CodePipeline 使用指南中的建立管線 (CLI)。

AWS Config 與 CodeBuild 樣品一起使用

AWS Config 提供資源清單，以及這些 AWS 資源的組態變更歷程記錄。 AWS Config 現在支持 AWS CodeBuild 作為 AWS 資源，這意味著該服務可以跟踪您的 CodeBuild 項目。如需有關的詳細資訊 AWS Config，請參閱什麼是 AWS Config？ 在AWS Config 開發人員指南中。

您可以在主控台的 [資 CodeBuild 源清查] 頁面上看到下列有關資源的資 AWS Config 訊：

- CodeBuild 組態變更的時間表。
- 每個 CodeBuild 項目的配置詳細信息。
- 與其他 AWS 資源的關係。
- 您 CodeBuild 專案的變更清單。

本主題中的程序向您展示如何設置 AWS Config 和查詢和檢視 CodeBuild專案。

主題

- 必要條件
- 設定 AWS Config
- 查找 AWS CodeBuild 項目
- 在 AWS Config 主控台中檢視 AWS CodeBuild 組態詳細資訊

必要條件

創建您的 AWS CodeBuild 項目。如需說明，請參閱建立組建專案。

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。

查找 AWS CodeBuild 項目

1. 登入 AWS 管理主控台，然後開啟 AWS Config 主控台，網址為 https://console.aws.amazon.com/config。
2. 在 [資源清查] 頁面上，選取 [資源類型] 下的 [AWS CodeBuild 專案] 向下捲動並選取CodeBuild 專案勾選方塊。
3. 選擇 Look up (查閱)。
4. 新增 CodeBuild 專案清單後，在「Config 時間軸」欄中選擇 CodeBuild 專案名稱連結。

在 AWS Config 主控台中檢視 AWS CodeBuild 組態詳細資訊

當您在 [資源清查] 頁面上查詢資源時，您可以選擇時 AWS Config 間表來檢視 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AWS Config 開發人員指南中的在 AWS Config 主控台中檢視設定詳細資料。

建立通知範例 CodeBuild

Amazon CloudWatch 活動內置了 AWS CodeBuild. CloudWatch 事件是描述 AWS 資源變更的系統事件串流。透過 E CloudWatch vents，您可以撰寫宣告式規則，將感興趣的事件與要採取的自動化動作產生關聯。此範例使用 Amazon E CloudWatch vents 和 Amazon Simple Notification Service (Amazon SNS)，在建置成功、失敗、從一個建置階段移至另一個階段或這些事件的任何組合時，傳送建置通知給訂閱者。

> **Important**
>
> 執行此範例可能會導致您的 AWS 帳戶收取費用。其中包括與 Amazon CodeBuild 和 Amazon SNS 相關的 AWS 資源和動作的可能收費 CloudWatch 和費用。如需詳細資訊，請參閱CodeBuild 定價、亞馬遜 CloudWatch定價和 Amazon SNS 定價。

執行範例

如何執行此範例

1. 如果您已經在 Amazon SNS 中設定並訂閱了要用於此範例的主題，請跳到步驟 4。否則，如果您使用 IAM 使用者而不是 AWS 根帳戶或管理員使用者使用 Amazon SNS，請將下列陳述式 (介於 *### ############ ### # ### END ######## ###*) 給使用者 (或與使用者關聯的 IAM 群組)。不建議使用 AWS 根帳號。此聲明可讓您檢視、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，也不要在現有政策中輸入這些省略符號。

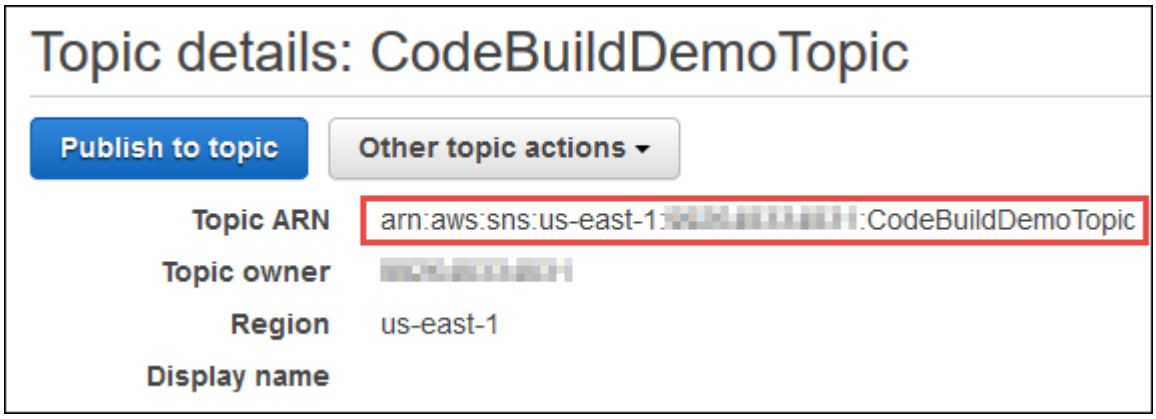
```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "sns:CreateTopic",
        "sns:GetTopicAttributes",
        "sns:List*",
        "sns:Publish",
        "sns:SetTopicAttributes",
        "sns:Subscribe"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> 修改此政策的 IAM 實體必須具有 IAM 中的許可，才能修改政策。
> 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。

2. 在 Amazon SNS 中建立或識別主題。 AWS CodeBuild 使用 CloudWatch 事件透過 Amazon SNS 傳送組建通知至此主題。

建立主題：

1. 開啟 Amazon SNS 主控台，網址為 https://console.aws.amazon.com/sns。
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 [主題詳細資料: 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。

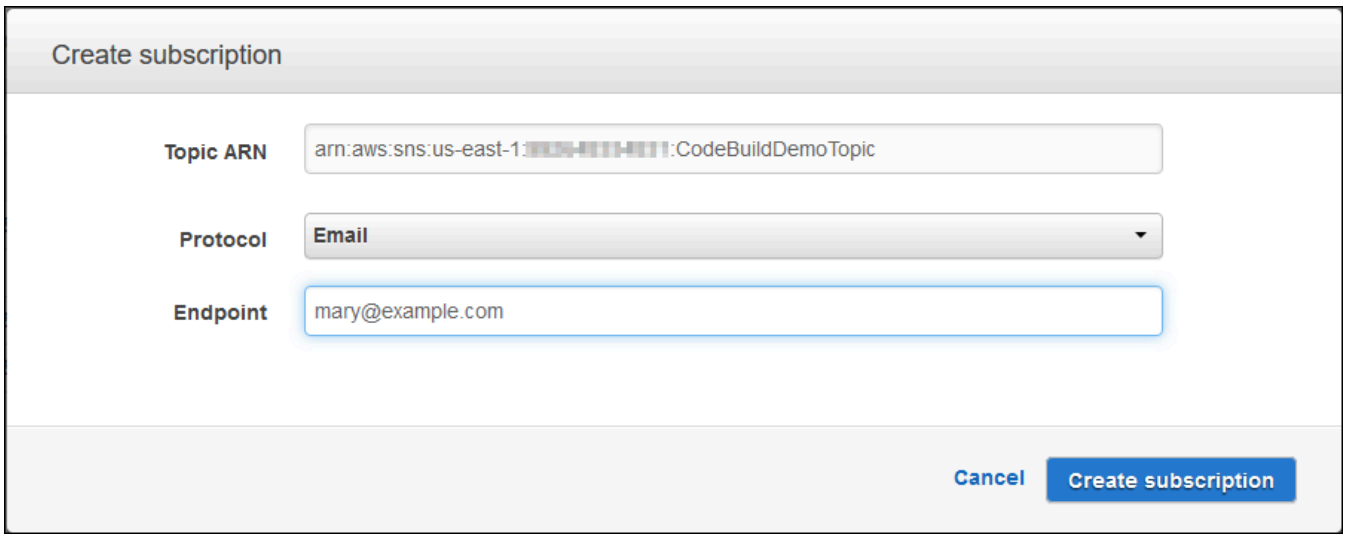


如需詳細資訊，請參閱 Amazon SNS 開發人員指南中的建立主題。

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. 在上一個步驟開啟 Amazon SNS 主控台的情況下，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者。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。收件者按一下連結後，如果訂閱成功，Amazon SNS 會在收件者的網頁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. 如果您使用的是使用者而非 AWS 根帳戶或管理員使用者來處理 CloudWatch 事件，請將下列陳述式 (介於 *### ############ ### # ### END ADD ### HERE ###*) 給使用者 (或與使用者關聯的 IAM 群組)。不建議使用 AWS 根帳號。該語句用於允許用戶與 CloudWatch 事件的工作。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，也不要在現有政策中輸入這些省略符號。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "events:*",
        "iam:PassRole"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
```

```
  "Version": "2012-10-17"
}
```

> **Note**
>
> 修改此政策的 IAM 實體必須具有 IAM 中的許可，才能修改政策。
> 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。

5. 在 CloudWatch 事件中建立規則。若要這麼做，請開啟主 CloudWatch 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. 在 Service Name (服務名稱) 中，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。

如需有關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。

如需使用事件模式篩選的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的以事件模式進行內容篩選。

> **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. 確認現在 CodeBuild 已成功發送構建通知。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)。

若要停止使用規則傳送組建通知，請在 CloudWatch 主控台中選擇要停止使用的規則，選擇 [動作]，然後選擇 [停用]。

若要完全刪除規則，請在 CloudWatch 主控台中選擇要刪除的規則，選擇 [動作]，然後選擇 [刪除]。

相關資源

- 如需開始使用的相關資訊 AWS CodeBuild，請參閱使用主控台開始使用 AWS CodeBuild。
- 如需有關疑難排解中問題的資訊 CodeBuild，請參閱疑難排 AWS CodeBuild。
- 如需中配額的相關資訊 CodeBuild，請參閱AWS CodeBuild 的配額。

建置通知輸入格式參考

CloudWatch 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"
  }
}
```

建立徽章範例 CodeBuild

AWS CodeBuild 現在支持使用構建徽章，它提供了可嵌入的動態生成的圖像（徽章），該圖像（徽章）顯示項目的最新版本的狀態。此圖像可以通過為您的 CodeBuild 項目生成的公開可用 URL 訪問。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。

建立啟用建置徽章的建置專案 (主控台)

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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。每個 AWS 帳戶的組建專案名稱必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
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 Amazon S3 來源供應商的建置徽章。由於 AWS CodePipeline 使用 Amazon S3 進行成品傳輸，因此屬於中建立管道一部分的建置專案不支援建置徽章 CodePipeline。

- 如果您選擇 CodeCommit，請選擇存放庫的名稱做為「存放庫」。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 GitHub，請依照指示進行連線 (或重新連線) GitHub。在 [GitHub授權應用程式] 頁面上，對於 [組織存取]，選擇您想要存取的每個儲存庫旁邊的 [AWS CodeBuild 要求存取權]。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)。選擇授與存取權後，請返回 AWS CodeBuild 主控台，對於「儲存庫」，選擇包含原始程式碼的儲存庫名稱。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 映像檔 AWS CodeBuild，請選擇 [受管理的映像檔]，然後從 [作業系統]、[執行階段]、[映像] 和 [映像檔版本] 中進行選取。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)。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。如果您選擇 [其他登錄]，對於 [外部登錄 URL]，請使用格式在 Docker Hub 中輸入 Docker 映像的名稱和標記。*docker repository*/*docker image name*如果您選擇 Amazon ECR，請使用 Amazon ECR 存儲庫和 Amazon ECR 映像在您的帳戶中選擇碼頭映像。 AWS
   - 若要使用私人 Docker 映像檔，請選擇 [自訂映像檔]。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。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。認證必須由 Secrets Manager 建立。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

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 [新增服務角色]。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**
   >
   > 使用主控台建立或更新組建專案時，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. 在建構規格中，執行下列其中一項作業：

   - 選擇「使用建置規格檔案」，在原始程式碼根目錄中使用 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)。
   - 若要將組建輸出存放在 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。
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。

建立啟用建置徽章的建置專案 (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 專案中包含組建徽章，您必須指定 *`badgeEnabled`* 的值為。`true`

存取您的 AWS CodeBuild 組建徽章

您可以使用 AWS CodeBuild 控制台或 AWS CLI 訪問構建徽章。

- 在 CodeBuild 主控台的建置專案清單的 [名稱] 欄中，選擇與建置專案對應的連結。在 [建立***##:####***] 頁面的 [設定] 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，執行命`batch-get-projects`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)。

組建徽章要求 URL 是使用通用預設分支產生的，但是您可以在來源存放庫中指定用來執行組建的任何分支。例如：

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

您也可以在來源儲存庫中指定標籤，方法是將branch參數取代為標記 URL 中的tag參數。例如：

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

發佈您的 CodeBuild 組建徽章

您可以使用降價圖像中的構建徽章 URL 在降價文件中顯示最新版本的狀態。這對於在源存儲庫的 readme.md 文件中顯示最新構建的狀態非常有用（例如， GitHub 或）。 CodeCommit例如：

```
![](<build badge URL>)
```

CodeBuild 徽章狀態

- 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。

CodeBuild 使用 AWS CLI 範例建立測試報告

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 將測試併入組建中 CodeBuild。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 控制台訪問測試結果。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

必要條件

- 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：

  - 黃瓜
  - 六月 XML（的 .xml）
  - 單位 XML（.xml）
  - nUnit3 XML (.xml)
  - 。TestNG。。XML。
  - 視覺工作室 TRX (.trx)
  - 可視化工作室 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？ 在 Amazon S3 用戶指南中。
- 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。

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
  }
}
```

使用報告群組設定專案

若要執行報表，請先 CodeBuild 建立使用報表群組設定的組建專案。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。這些文件存儲您可以訪問的測試結果 CodeBuild。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。如果您指定名稱而不是 ARN，則 CodeBuild 會在執行組建時建立報表群組。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
```

執行並檢視報告的結果

在本節中，您會執行先前建立專案的建置。在構建過程中， CodeBuild 創建一個包含測試用例結果的報告。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是上面創建的構建項目的名稱。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。記下輸出reportArns屬性中的報告 ARN。

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

3. 執行下列命令以取得有關報表的詳細資料。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": "<your-S3-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>"
    }
  ]
}
```

碼頭工人樣品 CodeBuild

主題

- 自訂影像範例中的泊塢視窗 CodeBuild
- 將 Docker 映像發佈到 Amazon 彈性容器登錄映像儲存庫範例， CodeBuild
- 帶有 AWS Secrets Manager 示例的私人註冊表 CodeBuild

自訂影像範例中的泊塢視窗 CodeBuild

此示例通過使用和自定義 Docker 構建映像（`docker:dind`在 Docker Hub 中）構建 AWS CodeBuild 和運行 Docker 映像。

要了解如何通過使用 Docker 支持提供的構建映像來 CodeBuild 構建 Docker 映像，請參閱我們的. 將碼頭映像發佈到 Amazon ECR 映像儲存庫範例

> **Important**
>
> 執行此範例可能會導致您的 AWS 帳戶收取費用。其中包括 CodeBuild 與 Amazon S3 相關的 AWS 資源和動作以及 CloudWatch 日誌可能的費用。 AWS KMS如需詳細資訊，請參閱CodeBuild 定價、Amazon S3 定AWS Key Management Service價、定價和 Amazon CloudWatch 定價。

主題

- 執行範例
- 目錄結構
- 檔案
- 相關資源

執行範例

如何執行此範例

1. 按照本主題的「目錄結構」和「檔案」一節中所述建立檔案，然後將它們上傳到 S3 輸入儲存貯體 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 建立組建專案，`create-project`指令的 JSON 格式輸入可能會類似於此。(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**
>
> 依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 Linux 功能並啟用特權模式。此外，Windows 不支援特殊權限模式。

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:
  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"
```

相關資源

- 如需開始使用的相關資訊 AWS CodeBuild，請參閱使用主控台開始使用 AWS CodeBuild。
- 如需有關疑難排解中問題的資訊 CodeBuild，請參閱疑難排 AWS CodeBuild。
- 如需中配額的相關資訊 CodeBuild，請參閱AWS CodeBuild 的配額。

將 Docker 映像發佈到 Amazon 彈性容器登錄映像儲存庫範例， CodeBuild

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 影像推送至 Docker Hub。如需詳細資訊，請參閱 調整範例將映像推送至 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 影像作為組建輸出。然後，它將碼頭映像推送到 Amazon ECR 映像存儲庫。多階段 Docker 影像組建有助於縮小最終 Docker 影像。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您的 AWS 帳戶收取費用。其中包括與 Amazon S3、 AWS KMS CloudWatch 日誌和 Amazon ECR 相關的 AWS 資源和動作的可能收費和費用。 AWS CodeBuild 如需詳細資訊，請參閱CodeBuild 定價、Amazon S3 定AWS Key Management Service價、定價、Amazon CloudWatch 定價和 Amazon 彈性容器登錄定價。

主題

- 執行範例
- 目錄結構
- 檔案
- 調整範例將映像推送至 Docker Hub
- 相關資源

執行範例

如何執行此範例

1. 如果您想要使用的 Amazon ECR 中已有映像儲存庫，請跳至步驟 3。否則，如果您使用的是使用者而非 AWS 根帳戶或管理員使用者使用 Amazon ECR，請將此陳述式 (介於 *### ######## ### ### # ### END ######## ###*) 給使用者 (或與使用者關聯的 IAM 群組)。不建議使用 AWS 根帳戶。此聲明允許建立用於存放 Docker 映像的 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 AWS Management Console中的使用內嵌政策。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "ecr:CreateRepository"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> 修改此政策的 IAM 實體必須具有 IAM 中的許可，才能修改政策。

2. 在 Amazon ECR 中創建一個映像存儲庫。請務必在建立組建環境並執行組建的相同 AWS 區域中建立存放庫。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 儲存庫政策授予您 CodeBuild 服務 IAM 角色的映像推送存取權限。

3. 將此聲明 (*# ### ########## ### # ### ######## ###*) 到您附加到服務角色的策略中。 AWS CodeBuild 此聲明允許 CodeBuild 將碼頭圖像上傳到 Amazon ECR 存儲庫。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，也不要在政策中輸入這些省略符號。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:CompleteLayerUpload",
        "ecr:GetAuthorizationToken",
        "ecr:InitiateLayerUpload",
        "ecr:PutImage",
        "ecr:UploadLayerPart"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> 修改此政策的 IAM 實體必須具有 IAM 中的許可，才能修改政策。

4. 按照本主題的「目錄結構」和「檔案」一節中所述建立檔案，然後將它們上傳到 S3 輸入儲存貯體 AWS CodeCommit、 GitHub、或 Bitbucket 儲存庫。若要取得更多資訊，請參閱AWS CodePipeline 使用指南中的「影像定義檔案參考」。

> **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 建立組建專案，`create-project`指令的 JSON 格式輸入可能會類似於此。(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 映像檔推送至儲存庫：

   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ENTRYPOINT訂泊塢視窗影像的。

調整範例將映像推送至 Docker Hub

要將碼頭映像推送到碼頭集線器而不是 Amazon ECR，請編輯此示例的代碼。

> **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. 替換文件buildspec.yml中的這些 Amazon ECR 特定行代碼：

```
...
  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. 將編輯過的程式碼上傳至 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. 建立組建環境、執行組建，以及檢視相關的組建資訊。
5. 確認 AWS CodeBuild 已成功將 Docker 映像檔推送至儲存庫。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。

相關資源

- 如需開始使用的相關資訊 AWS CodeBuild，請參閱使用主控台開始使用 AWS CodeBuild。
- 如需有關疑難排解中問題的資訊 CodeBuild，請參閱疑難排 AWS CodeBuild。
- 如需中配額的相關資訊 CodeBuild，請參閱AWS CodeBuild 的配額。

帶有 AWS Secrets Manager 示例的私人註冊表 CodeBuild

此範例說明如何使用儲存在私人登錄中的 Docker 映像檔做為 AWS CodeBuild 執行階段環境。私有登錄檔的登入資料會存放在 AWS Secrets Manager。任何私人註冊表都可以使用 CodeBuild. 這個範例使用 Docker Hub。

> **Note**
>
> 密碼在動作中可見，並且在寫入檔案時不會遮罩。

私有登錄檔範例需求

若要搭配使用私人登錄 AWS CodeBuild，您必須具備下列項目：

- 儲存您的 Docker Hub 認證的秘 Secrets Manager 碼。這些登入資料會用來存取您的私有儲存庫。

  > **Note**
  >
  > 您將需要為您建立的秘密付費。

- 私有儲存庫或帳戶。

- 授予秘密管理員密碼存取權的 CodeBuild 服務角色與存取權管理員政策。

請依照下列步驟建立這些資源，然後使用儲存在私人登錄中的 Docker 映像檔建立 CodeBuild 建置專案。

使用私人登錄建立 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。您也可以在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 密碼」中的步驟進行。

   a. 在步驟 3 的 [選擇密碼類型] 中，選擇 [其他密碼類型]。

   b. 在鍵/值對中，為您的 Docker Hub 用戶名創建一個鍵值對，為您的 Docker Hub 密碼創建一個鍵值對。

   c. 繼續執行建立 AWS Secrets Manager 密碼中的步驟。

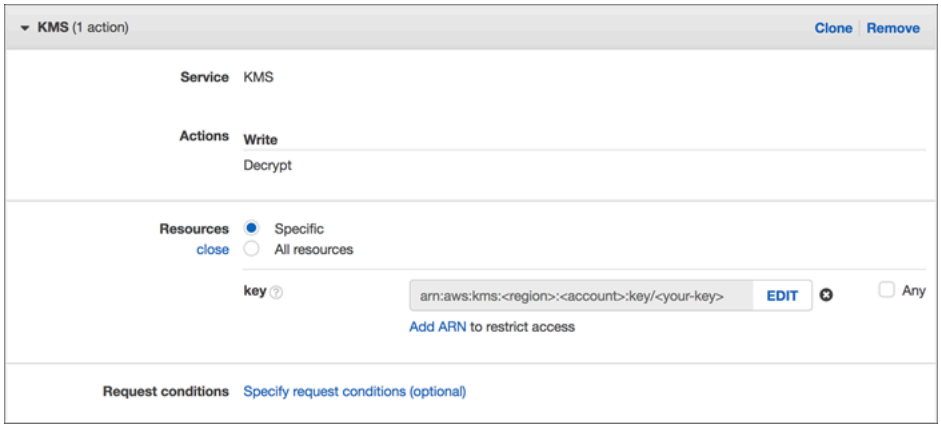
   d. 在步驟 5 中，在 [設定自動輪換] 頁面上，將其關閉，因為金鑰對應於您的 Docker Hub 認證。

   e. 完成依照建立 AWS Secrets Manager 密碼中的步驟操作。

   如需詳細資訊，請參閱什麼是 AWS Secrets Manager？

3. 當您在控制台中創建 AWS CodeBuild 項目時，為您 CodeBuild 附加所需的權限。如果您使用的 AWS KMS 金鑰不是`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)。

   您的服務角色必須至少具有權限，才能與 Secrets Manager 搭配使`secretsmanager:GetSecretValue`用。



4. 若要使用主控台搭配私有登錄檔中所儲存的環境來建立專案，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**
>
> 如果您的私人註冊表位於 VPC 中，則必須具有公共互聯網訪問權限。 CodeBuild 無法從 VPC 中的私有 IP 地址提取映像。

   a. 在環境影像中，選擇 [自訂影像]。
   b. 針對 Environment type (環境類型)，選擇 Linux 或 Windows。
   c. 對於映像登錄，請選擇其他登錄。
   d. 在外部登錄 URL 中，輸入映像位置，並在登錄憑證中輸入-選擇性地輸入 ARN 或 Secrets Manager 認證的名稱。

> **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。

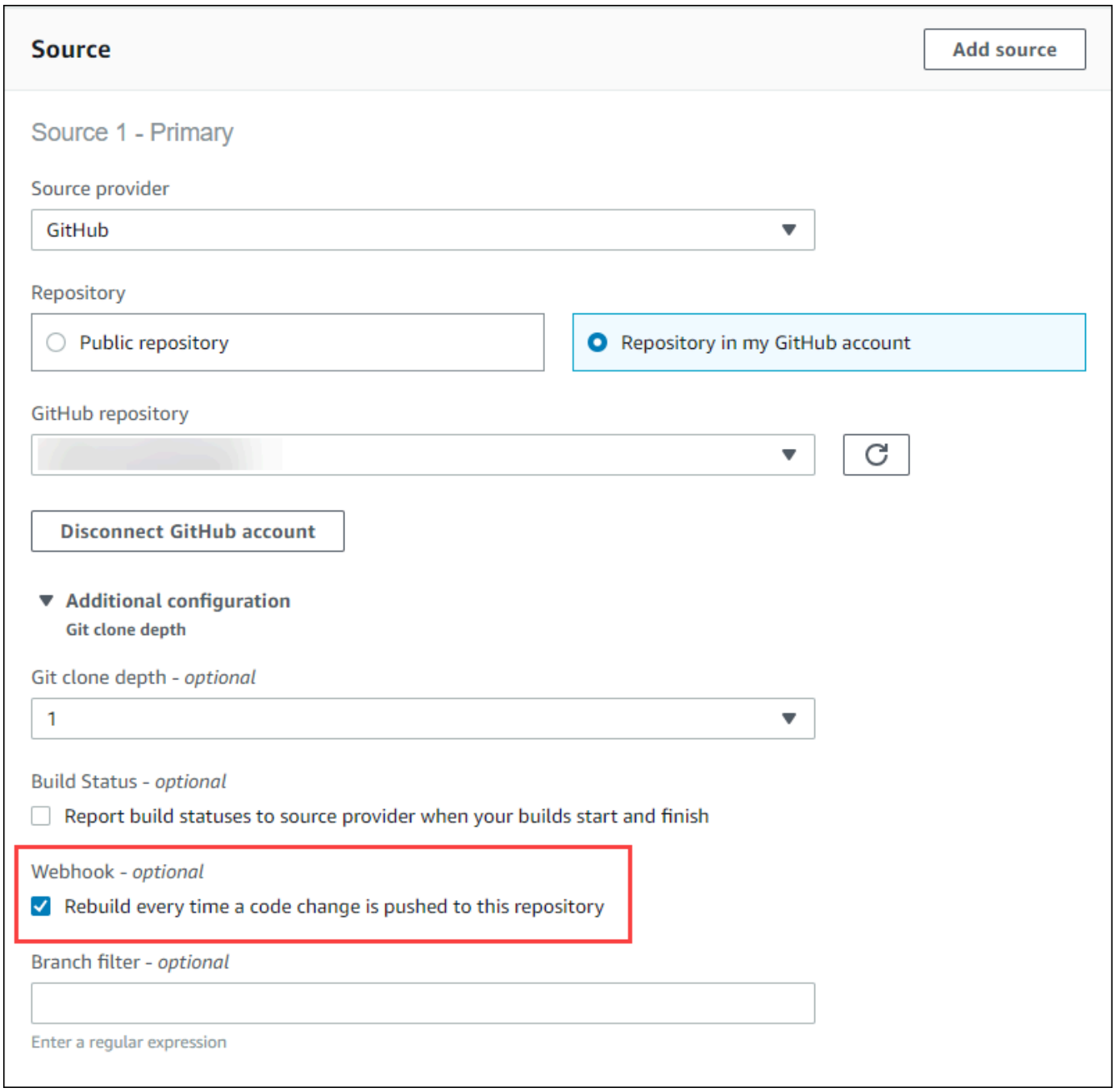
使用託管於 S3 儲存貯體中的建置輸出建立靜態網站

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. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
3. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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。每個 AWS 帳戶的組建專案名稱必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
5. 在來源中，對於來源提供者，選擇GitHub。依照指示進行連線 (或重新連線) GitHub，然後選擇 [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。



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 映像檔 AWS CodeBuild，請選擇 [受管理的映像檔]，然後從 [作業系統]、[執行階段]、[映像] 和 [映像檔版本] 中進行選取。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)。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。如果您選擇 [其他登錄]，對於 [外部登錄 URL]，請使用格式在 Docker Hub 中輸入 Docker 映像的名稱和標記。*docker repository*/*docker image name*如果您選擇 Amazon ECR，請使用 Amazon ECR 存儲庫和 Amazon ECR 映像在您的帳戶中選擇碼頭映像。 AWS
- 若要使用私人 Docker 映像檔，請選擇 [自訂映像檔]。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。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。認證必須由 Secrets Manager 建立。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

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 [新增服務角色]。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**
   >
   > 使用主控台建立或更新組建專案時，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. 在建構規格中，執行下列其中一項作業：

   - 選擇「使用建置規格檔案」，在原始程式碼根目錄中使用 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，對於類型，選擇 Amazon S3 將組建輸出存放在 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. 如果您在環境中選擇 [插入建置命令]，則對於 [輸出檔案]，請輸入組建中要放入輸出值區的檔案位置。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. （可選）遵循 Amazon Amazon S3 開發人員指南中範例：使用 Amazon CloudFront 加快您的網站速度。

多個輸入來源和輸出成品範例

您可以 AWS CodeBuild 建立包含多個輸入來源和多組輸出成品的組建專案。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。

在此範例中，您會建立組建專案並用來執行組建。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。

1. 將您的來源上傳到一或多個 S3 儲存貯體 CodeCommit、 GitHub、 GitHub 企業伺服器或 Bitbucket 儲存庫。
2. 選擇哪個來源是主要來源。這是查 CodeBuild 找並運行構建規格文件的源代碼。
3. 建立建置專案。如需詳細資訊，請參閱 在 AWS CodeBuild 中建立建置專案。
4. 創建您的構建項目，運行構建，並獲取有關構建的信息。
5. 如果您使 AWS CLI 用建立組建專案，`create-project`指令的 JSON 格式輸入可能會類似下列內容：

```
{
  "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_`*`sourceIdentifer`* 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`上是和中的必要屬`CreateProject`性`UpdateProject`。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指令的 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"
      },
   ]
```

```
}
```

無來源的專案範例

您可以在設定來源時選擇**NO_SOURCE**來源類型來配置 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)。

若要瞭解如何建立使用多個來源輸入來 CodeBuild 建立多個輸出成品的管道，請參閱AWS CodePipeline 與多個輸入來源 CodeBuild 和輸出假影範例整合 。

構建規格文件示例中的運行時版本 CodeBuild

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，則可以在構建規格文件的`runtime-versions`部分中指定一個或多個運行時。這個範例說明如何變更您的專案執行時間、指定一個以上的執行時間，以及指定一個取決於另一個執行時間的執行時間。如需支援的執行時間資訊，請參閱碼頭圖片提供 CodeBuild。

> **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 在 AWS CodeBuild 中執行建置 及 在 AWS CodeBuild 中建立建置專案。

更新您的執行時間版本

您可以通過更新 buildspec 文件的`runtime-versions`部分將項目使用的運行時修改為新版本。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
```

下面的例子演示了如何使用 Ubuntu 的標準圖像 5.0 或 Amazon Linux 2 標準圖像 3.0 指定不同版本的 Python：

- 指`runtime-versions`定 3.7 版本 Python 部分：

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

- 指`runtime-versions`定 3.8 版本 Python 部分：

```
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。

1. 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。
2. 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
```

4. 將my-web-app目錄的內容上傳到 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。

5. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home

6. 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)。
     - 對於圖像，請選擇 AWS /代碼構建/ 亞馬遜鏈 2-x86_64 標準：4.0。

7. 選擇 Start build (開始組建)。
8. 在 Build configuration (組建組態) 上，接受預設值，然後選擇 Start build (開始組建)。
9. 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

您可以在同一個 CodeBuild 構建項目中指定多個運行時間。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。此runtime-versions區段會指定 Node.js 版本 12 和 Go 版本 1.13 執行階段。

```
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`目錄的內容上傳到 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. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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)。
      - 對於圖像，請選擇 AWS /代碼構建/ 亞馬遜鏈 2-x86_64 標準：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
```

原始碼版本範例 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，請使用代表組建輸入 ZIP 檔案之物件的版本識別碼。
- 對於 CodeCommit、Bitbucket 和 GitHub 企業伺服器，請使用下列其中一項： GitHub
  - 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**
>
> 只有在儲存庫為 GitHub 或 GitHub 企業伺服器時，才能指定提取要求來源的版本。

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。這是因為當您添加引用時， CodeBuild 不需要下載整個存儲庫即可查找提交。

- 您可以指定僅具有遞交 ID 的來源版本，例如 `12345678901234567890123467890123456789`。如果這樣做， CodeBuild 必須下載整個存儲庫以查找版本。

- 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^{1234567890123456789012346789012345678 9}) 。如果您這樣做，只會 CodeBuild 下載指定的分支以尋找版本。

> **Note**
>
> 為了加快構建DOWNLOAD_SOURCE階段，您還可以將 Git 克隆深度設置為較低的數字。CodeBuild 下載較少版本的儲存庫。

使用提交 ID 指定 GitHub 儲存庫版本

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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：
     - 對於來源提供者，請選擇GitHub。如果您沒有連接到 GitHub，請按照說明進行連接。
     - 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)。
     - 對於圖像，請選擇 AWS /代碼構建/ 亞馬遜鏈 2-x86_64 標準：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。

這些步驟說明如何使用相同版本的來源建立組建。這時，來源的版本是使用具有遞交 ID 的參考指定。

使用提交 ID 和參照指定 GitHub 儲存庫版本

1. 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。
2. 選擇 Start build (開始組建)。
3. 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`*}。
4. 選擇 Start build (開始組建)。

5. 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。

的第三方來源儲存庫範例 CodeBuild

主題

比特桶拉請求和網絡掛鉤過濾器樣本 CodeBuild

AWS CodeBuild 當源存儲庫是比特桶時支持網絡掛鉤。這意味著對於將其源代碼存儲在 Bitbucket 存儲庫中的 CodeBuild 構建項目，每次將代碼更改推送到存儲庫時，都可以使用 webhook 來重建源代碼。如需詳細資訊，請參閱 比特桶網絡鉤事件。

此範例示範如何使用 Bitbucket 儲存庫建立提取請求。它還向您展示瞭如何使用 Bitbucket webhook CodeBuild 來觸發創建項目的構建。

> **Note**
>
> 使用 Webhook 時，用戶可能會觸發意外的構建。若要降低此風險，請參閱使用網路掛鉤的最佳做法。

主題

- 必要條件
- 建立以 Bitbucket 為來源儲存庫的建置專案並啟用 Webhook
- 以 Bitbucket Webhook 觸發建置

必要條件

要運行此示例，您必須將 AWS CodeBuild 項目與您的 Bitbucket 帳戶連接起來。

> **Note**
>
> CodeBuild 已經更新了它的權限與比特桶。如果您之前將項目連接到 Bitbucket，現在收到 Bitbucket 連接錯誤，則必須重新連接以授予管理 Webhook 的 CodeBuild 權限。

建立以 Bitbucket 為來源儲存庫的建置專案並啟用 Webhook

以下步驟描述了如何使用 Bitbucket 作為源存儲庫創建 AWS CodeBuild 項目並啟用 Webhook。

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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)

   輸入此組建專案的名稱。每個 AWS 帳戶的組建專案名稱必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
5. 在 Source (來源) 中：

   來源提供者

   選擇比特桶。按照說明與 Bitbucket 連接（或重新連接），然後選擇授權。

   儲存庫

   在我的 Bitbucket 帳戶中選擇存儲庫。

如果您之前沒有連接到您的 Bitbucket 帳戶，請輸入您的 Bitbucket 用戶名和應用程序密碼，然後選擇保存 Bit bucket 憑據。

比特桶存儲庫

輸入您的比特桶存儲庫的 URL。

6. 在主要來源 Webhook 事件中，選取下列項目。

> **Note**
>
> 僅當您在上一步中選擇了 Bitbucket 帳戶中的存儲庫時，主要源 webhook 事件部分才可見。

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「新增過濾器群組」以新增其他過濾器群組

如需 Bitbucket Webhook 事件類型和篩選器的詳細資訊，請參閱。比特桶網絡鉤事件

7. 在 Environment (環境) 中：

環境影像

選擇下列其中一項：

若要使用由 AWS CodeBuild下列項目管理的 Docker 映像檔：

選擇 [受管理的映像檔]，然後從 [作業系統]、[執行階段]、[映像] 和 [映像檔版本] 中進行選取。若可用，請從 Environment type (環境類型) 進行選擇。

要使用另一個 Docker 圖像：

選擇「自訂影像」。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。如果您選擇 [其他登錄]，對於 [外部登錄 URL]，請使用格式在 Docker Hub 中輸入 Docker 映像的名稱

和標記。*docker repository*/*docker image name*如果您選擇 Amazon ECR，請使用 Amazon ECR 存儲庫和 Amazon ECR 映像在您的帳戶中選擇碼頭映像。 AWS

若要使用私人泊塢視窗映像檔：

選擇「自訂影像」。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。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。認證必須由 Secrets Manager 建立。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

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 [新增服務角色]。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**
>
> 使用主控台建立或更新組建專案時，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. 在建構規格中，執行下列其中一項作業：

   - 選擇「使用建置規格檔案」，在原始程式碼根目錄中使用 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)。
   - 若要將組建輸出存放在 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。

以 Bitbucket Webhook 觸發建置

對於使用 Bitbucket Webhook 的項目，請在 Bitbucket 存儲庫檢測到源代碼中的更改時 AWS CodeBuild 創建一個構建。

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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 網路掛接專案的相關資訊，請參閱。the section called “建立以 Bitbucket 為來源儲存庫的建置專案並啟用 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. 選擇檢視詳細資料，查看傳回之回應的詳細資訊 CodeBuild。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。

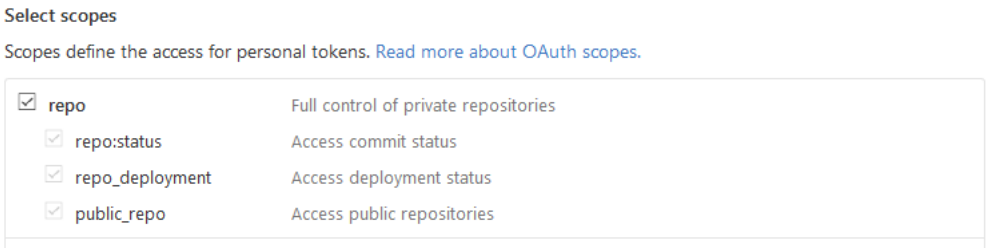
GitHub 企業伺服器範例 CodeBuild

AWS CodeBuild 支援 GitHub 企業伺服器作為來源儲存庫。此範例顯示當 GitHub 企業伺服器存放庫已安裝憑證時，如何設定 CodeBuild 專案。它還顯示瞭如何啟用 webhook，以便每次將代碼更改推送到 GitHub 企業服務器存儲庫時 CodeBuild 重建源代碼。

必要條件

1. 為您的 CodeBuild 項目生成個人訪問令牌。我們建議您建立 GitHub Enterprise 使用者，並為此使用者產生個人存取權杖。將其複製到剪貼板，以便在創建 CodeBuild 項目時可以使用它。如需詳細資訊，請參閱 GitHub 說明網站上的命令列建立個人存取權杖。

   當您建立個人存取字符時，請在定義中包含 repo 範圍。

2. 從 GitHub 企業伺服器下載您的憑證。 CodeBuild 使用憑證與存放庫建立受信任的 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 企業伺服器存放庫的 IP 位址。

   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 企業伺服器下載憑證。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**
>
> 此值區必須與您的組建位於相同的 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，則值區必須位於美國東部 (俄亥俄) 區域。

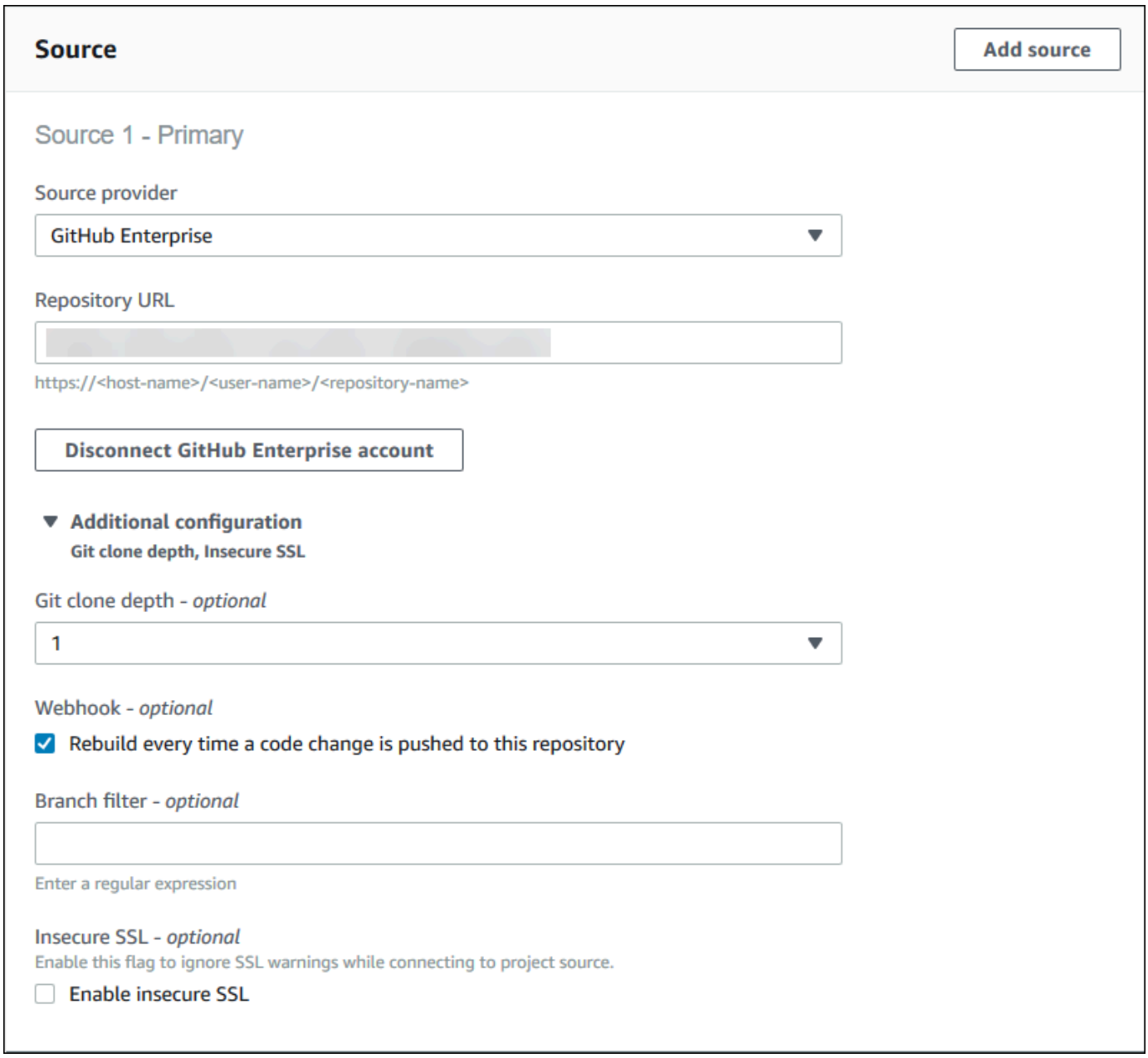
使用 GitHub企業服務器作為源存儲庫創建構建項目並啟用 webhook（控制台）

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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。每個 AWS 帳戶的組建專案名稱必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
4. 在來源的來源提供者中，選擇GitHub 企業。
   - 針對 Personal Access Token (個人存取字符)，貼上您已複製至剪貼簿的字符，然後選擇 Save Token (儲存字符)。在存放庫 URL 中，輸入 GitHub 企業伺服器儲存庫的 URL。

> **Note**
>
> 您只需要輸入及儲存個人存取字符一次。所有 future 的 AWS CodeBuild 項目都使用此令牌。

- 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 [啟用不安全的 SSL] 可在您連線至 GitHub 企業伺服器專案存放庫時忽略 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。



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 映像檔 AWS CodeBuild，請選擇 [受管理的映像檔]，然後從 [作業系統]、[執行階段]、[映像] 和 [映像檔版本] 中進行選取。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)。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。如果您選擇 [其他登錄]，對於 [外部登錄 URL]，請使用格式

在 Docker Hub 中輸入 Docker 映像的名稱和標記。*docker repository*/*docker image name*如果您選擇 Amazon ECR，請使用 Amazon ECR 存儲庫和 Amazon ECR 映像在您的帳戶中選擇碼頭映像。 AWS

- 若要使用私人 Docker 映像檔，請選擇 [自訂映像檔]。在「環境類型」中，選擇 ARM、Linux、GPU 或視窗。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。認證必須由 Secrets Manager 建立。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

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 [新增服務角色]。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**
   >
   > 使用主控台建立或更新組建專案時，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)。

   如果您想 CodeBuild 要使用 VPC：

   - 對於 VPC，請選擇使 CodeBuild 用的 VPC 識別碼。
   - 對於 VPC 子網路，請選擇包含使用之資源的子網路。 CodeBuild
   - 對於 VPC 安全性群組，請選擇 CodeBuild 用來允許存取 VPC 中資源的安全性群組。

   如需詳細資訊，請參閱 搭 AWS CodeBuild 配 Amazon Virtual Private Cloud 使用。

8. 在建構規格中，執行下列其中一項作業：

   - 選擇「使用建置規格檔案」，在原始程式碼根目錄中使用 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)。
- 若要將組建輸出存放在 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。

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。
  - (選擇性) 對於快取路徑前置詞，請輸入 Amazon S3 路徑前置詞。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。

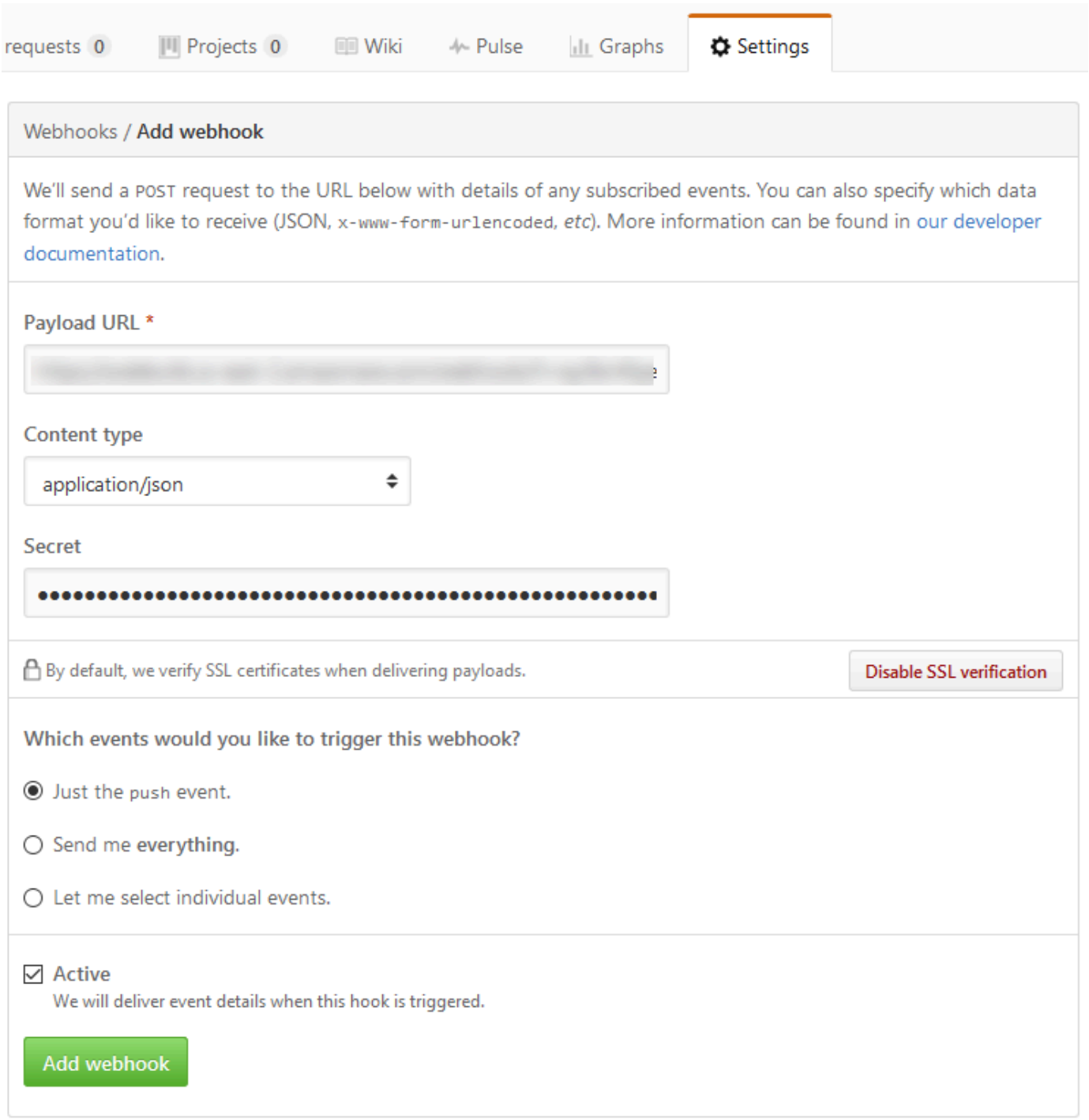
使用快取可節省大量建置時間，因為建置環境的可重複使用部分存放在快取中，並用於各建置。如需在 buildspec 檔案中指定快取的詳細資訊，請參閱Buildspec 語法。如需快取的詳細資訊，請參閱「在 AWS CodeBuild 中建立快取」。

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)。

12. 如果您已在 Source (來源) 中啟用 Webhook，則會出現 Create webhook (建立 Webhook) 對話方塊，其中包含 Payload URL (有效負載 URL) 和 Secret (秘密) 的值。

    > **Important**
    >
    > Create webhook (建立 Webhook) 對話方塊只會出現一次。複製有效負載 URL 和秘密金鑰。當您在 GitHub 企業服務器中添加 webhook 時，您需要它們。
    > 如果您需要再次產生承載 URL 和密鑰，則必須先從 GitHub 企業服務器存儲庫中刪除 webhook。在您的 CodeBuild 專案中，清除 [Webhook] 核取方塊，然後選擇 [儲存]。然後，您可以在選取「Webhook」(Webhook) 核取方塊的情況下建立或更新 CodeBuild專案。Create webhook (建立 Webhook) 對話方塊會再次出現。

13. 在 GitHub 企業伺服器中，選擇儲存 CodeBuild 專案的存放庫。
14. 依序選擇 Settings (設定)、Hooks & services (關聯和服務) 和 Add webhook (新增 Webhook)。
15. 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)。



16. 返回到您的 CodeBuild 項目。關閉 Create webhook (建立 Webhook) 對話方塊，然後選擇 Start build (開始建置)。

GitHub 拉請求和網絡掛鉤過濾器樣本 CodeBuild

AWS CodeBuild 當源存儲庫是 GitHub時支持網絡掛鉤。這意味著對於將源代碼存儲在存儲 GitHub 庫中的 CodeBuild 構建項目，每次將代碼更改推送到存儲庫時，都可以使用 webhook 重建源代碼。如需 CodeBuild 範例，請參閱AWS CodeBuild 範例。

> **Note**
>
> 使用 Webhook 時，用戶可能會觸發意外的構建。若要降低此風險，請參閱使用網路掛鉤的最佳做法。

創建一個構建項目 GitHub作為源存儲庫並啟用 webhook（控制台）

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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)

   輸入此組建專案的名稱。每個 AWS 帳戶的組建專案名稱必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
5. 在 Source (來源) 中：

   來源提供者

   選擇GitHub。依照指示進行連線 (或重新連線)， GitHub 然後選擇 [授權]。

   儲存庫

   在我的 GitHub帳戶中選擇儲存庫。

   GitHub 儲存庫

   輸入 GitHub 存放庫的 URL。
6. 在主要來源 Webhook 事件中，選取下列項目。

> **Note**
>
> 僅當您在上一步中選擇了我的 GitHub 帳戶中的存儲庫時，主要源 webhook 事件部分才可見。

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「新增過濾器群組」以新增其他過濾器群組

如需有關 GitHub webhook 事件類型和篩選器的詳細資訊，請參閱GitHub 網絡掛鉤事件。

7. 在 Environment (環境) 中：

環境影像

選擇下列其中一項：

若要使用由 AWS CodeBuild下列項目管理的 Docker 映像檔：

選擇 [受管理的映像檔]，然後從 [作業系統]、[執行階段]、[映像] 和 [映像檔版本] 中進行選取。若可用，請從 Environment type (環境類型) 進行選擇。

要使用另一個 Docker 圖像：

選擇「自訂影像」。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。如果您選擇 [其他登錄]，對於 [外部登錄 URL]，請使用格式在 Docker Hub 中輸入 Docker 映像的名稱和標記。*docker repository*/*docker image name*如果您選擇 Amazon ECR，請使用 Amazon ECR 存儲庫和 Amazon ECR 映像在您的帳戶中選擇碼頭映像。 AWS

若要使用私人泊塢視窗映像檔：

選擇「自訂影像」。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。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。認證必須由 Secrets Manager 建立。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

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 [新增服務角色]。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**
>
> 使用主控台建立或更新組建專案時，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. 在建構規格中，執行下列其中一項作業：

- 選擇「使用建置規格檔案」，在原始程式碼根目錄中使用 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)。
- 若要將組建輸出存放在 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。

驗證檢查

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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)。
   - 選擇您要驗證的 Webhook 構建項目旁邊的按鈕，選擇查看詳細信息，然後選擇構建詳細信息選項卡。
4. 在主要來源網路掛接事件中，選擇 Webhook 網址連結。
5. 在 GitHub 儲存庫的「設定」 頁面的「Webhook」下，確認已選取「提取要求」和「推送」。
6. 在您的個人資 GitHub 料設置中，在個人設置，應用程序，授權的 OAuth 應用程序下，您應該看到您的應用程序已被授權訪問您選擇的 AWS 區域。

使用語意版本控制來命名建置成品範例

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 JSON 格式的檔案中將設定`overrideArtifactName`為 true。`create-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")
```

這個 Windows 範例會示範如何指定使用 buildspec 檔案中宣告的變數和 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
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 建立的規格參考 CodeBuild。

Microsoft 視窗樣本 CodeBuild

這些範例會使用執行 Microsoft 視窗伺服器 2019、.NET 架 AWS CodeBuild 構和 .NET 核心 SDK 的建置環境，以 F# 和 Visual Basic 撰寫的程式碼建置執行階段檔案。

> **Important**
>
> 執行這些範例可能會向您的 AWS 帳戶收取費用。其中包括 CodeBuild 與 Amazon S3 相關的 AWS 資源和動作以及 CloudWatch 日誌可能的費用。 AWS KMS如需詳細資訊，請

參閱CodeBuild定價、Amazon S3 定AWS Key Management Service價、定價和 Amazon CloudWatch 定價。

執行範例

如何執行這些範例

1. 按照本主題的「目錄結構」和「檔案」一節中所述建立檔案，然後將它們上傳到 S3 輸入儲存貯體或 CodeCommit 或 GitHub存放庫。

   > **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。構建項目必須使用映mcr.microsoft.com/dotnet/framework/sdk:4.8像來構建 .NET 框架項目。

   如果您使用 AWS CLI 建立組建專案，create-project指令的 JSON 格式輸入可能會類似於此。(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，並遵循中的步驟執行建置。
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。解壓縮內容以進入執行階段和其他檔案。

   - 您可以在FSharpHelloWorld\bin\Debug目錄中找到使用 .NET Framework 之 F# 範例的執行階段檔案。FSharpHelloWorld.exe
   - 您可以在VBHelloWorld\bin\Debug目錄中找到使用 .NET 架構之 Visual Basic 範例的執行階段檔案。VBHelloWorld.exe

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>
  </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.EditorBrow
 _
    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.EditorBrow
 _
    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.EditorBrows
 _
  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.EditorBrows
 _
        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 AWS CodeBuild 中規劃建置

使用 AWS CodeBuild 之前，您必須先回答下列問題：

1. 源代碼存儲在哪裡？ CodeBuild 目前支援從下列原始程式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 是建置命令和相關設定的集合 (YAML 格式)， CodeBuild 用來執行組建。您可以在構建項目定義中聲明構建規格。

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： 創建一個 CodeCommit 存儲庫 創建一個提交 CodeCommit
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 入門指南中的下列主題： 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」網站上的此主題： 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，會從您指定的提供者 CodeBuild 下載組建輸入，並將組建輸出上傳到您指定的值區。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？ 建置需要 Git、AWS CLI 或其他工具？

   CodeBuild 在使用 Docker 映像檔的建置環境中執行組建。這些 Docker 影像必須存放在 CodeBuild 支援的儲存庫類型中。這些包括 CodeBuild 碼頭映像存儲庫，碼頭集線器和 Amazon Elastic Container Registry (Amazon ECR)。如需 CodeBuild Docker 影像儲存庫的詳細資訊，請參閱〈〉碼頭圖片提供 CodeBuild。
4. 您是否需要未自動提供的AWS資源 CodeBuild？ 如果是這樣，那些資源需要哪些安全策略？ 例如，您可能需要修改 CodeBuild 服務角色，才能使用這些資源。 CodeBuild
5. 您想要使 CodeBuild 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 搭 AWS CodeBuild 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 開發套件。如需詳細資訊，請參閱 執行建置。
- 在中建立或識別管道AWS CodePipeline，然後新增組建或測試動作， CodeBuild 以指示自動測試程式碼、執行組建或兩者。如需更多詳細資訊，請參閱 CodePipeline 搭配使用 CodeBuild。

建立的規格參考 CodeBuild

本主題提供關於建置規格 (Buildspec) 檔案的重要參考資訊。Buildspec 是建置命令和相關設定的集合 (YAML 格式)， CodeBuild 用來執行組建。您可以包含 buildspec 作為源代碼的一部分，也可以在創建構建項目時定義 buildspec。如需組建規格運作方式的詳細資訊，請參閱CodeBuild 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
- Batch 量生成構建規範參考

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)。

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值設定為替代 buildspec 檔案的路徑，相對於內建環境變數的值。CODEBUILD_SRC_DIR您也可以對 AWS SDK 中的create project操作進行等效操作。如需詳細資訊，請參閱 建立組建專案 或 變更建置專案的設定 。
- 執行 AWS CLI start-build指令，將buildspecOverride值設定為替代 buildspec 檔案的路徑，相對於內建環境變數的值。CODEBUILD_SRC_DIR您也可以對 AWS SDK 中的start build操作進行等效操作。如需詳細資訊，請參閱 執行建置。
- 在 AWS CloudFormation 範本中，將資源類型Source中的BuildSpec屬性設定AWS::CodeBuild::Project為替代 buildspec 檔案的路徑 (相對於內建環境變數的值)。CODEBUILD_SRC_DIR若要取得更多資訊，請參閱《AWS CloudFormation 使用指南》中的AWS CodeBuild 專案來源中的 BuildSpec性質。

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:

phases:
  install:
```

```
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    runtime-versions:
      runtime: version
      runtime: version
    commands:
      - command
      - command
    finally:
      - command
      - command
    # steps:
  pre_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    commands:
      - command
      - command
    finally:
      - command
      - command
    # steps:
  build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    commands:
      - command
      - command
    finally:
      - command
      - command
    # steps:
  post_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    commands:
      - command
      - command
    finally:
      - command
      - command
    # steps:
reports:
  report-group-name-or-arn:
```

```
    files:
      - location
      - location
    base-directory: location
    discard-paths: no | yes
    file-format: report-format
artifacts:
  files:
    - location
    - location
  name: artifact-name
  discard-paths: no | yes
  base-directory: location
  exclude-paths: excluded paths
  enable-symlinks: no | yes
  s3-prefix: prefix
  secondary-artifacts:
    artifactIdentifier:
      files:
        - location
        - location
      name: secondary-artifact-name
      discard-paths: no | yes
      base-directory: location
    artifactIdentifier:
      files:
        - location
        - location
      discard-paths: no | yes
      base-directory: location
cache:
  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。指定在此組建規格檔案中執行指令的 Linux 使用者。 `run-as`授與指定的使用者讀取和執行權限。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，記 CodeBuild 錄檔中會隱藏下列項目：
>
> - AWS 存取金鑰 ID。如需詳細資訊，請參閱使用AWS Identity and Access Management 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 使用指南中的 Systems Manager 參數存放區和系統管理員參數存放主控台逐步解說。
> - 使用指定的字串 AWS Secrets Manager。如需詳細資訊，請參閱 金鑰管理。

外殼/殼

選用的序列。指定 Linux 或視窗作業系統支援的殼層。

對於 Linux 作業系統，支援的殼層標籤為：

- `bash`
- `/bin/sh`

對於 Windows 作業系統，支援的殼層標籤為：

- `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**
>
> 我們強烈建議在環境變量中存儲敏感值。環境變數可以使用 CodeBuild 主控台和 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 在 AWS CodeBuild 中執行建置。
> - 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 AWS CodeBuild 中建立建置專案 及 在 AWS CodeBuild 中變更建置專案的設定。
> - buildspec 宣告中的值會採用最低優先順序。

env/parameter-store

如果env已指定，且您想要擷取存放在 Amazon EC2 Systems Manager 參數存放區中的自訂環境變數，則需要此選項。包含#/#標量的對應，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。 *key* 是您稍後在建置命令中用來參照此自訂環境變數的名稱，而 *value* 是儲存在 Amazon EC2 Systems Manager 參數存放區中的自訂環境變數名稱。若要存放機密值，請參閱 Amazon EC2 Systems Manager 使用者指南中的系統管理員參數存放區和逐步解說：建立和測試字串參數 (主控台)。

> **Important**
>
> 若 CodeBuild 要允許擷取存放在 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 建立 CodeBuild 服務角色。
> 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 在 AWS CodeBuild 中執行建置。
> - 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 AWS CodeBuild 中建立建置專案 及 在 AWS CodeBuild 中變更建置專案的設定。
> - buildspec 宣告中的值會採用最低優先順序。

env/secrets-manager

如果您要擷取儲存在中的自訂環境變數，則為必要項目 AWS Secrets Manager。`reference-key`使用下列病毒碼指定密碼管理員：

<key>: *<secret-id>*:*<json-key>*:*<version-stage>*:*<version-id>*

<key>

(必要) 本機環境變數名稱。使用此名稱可在建置期間存取變數。

<secret-id>

(必要) 作為密碼唯一識別碼的名稱或 Amazon 資源名稱 (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 帳戶中的密碼，請指定秘密 ARN。

<json-key>

(選擇性) 指定您要擷取其值之 Secrets Manager 金鑰值配對的金鑰名稱。如果未指定json-key， CodeBuild 會擷取整個密碼文字。

<version-stage>

(選擇性) 指定您要透過附加至版本的暫存標籤擷取的密碼版本。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>

(選擇性) 指定您要使用之密碼版本的唯一識別碼。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是儲存在 Secrets Manager 中的索引鍵值配對名稱。的關鍵TestSecret是MY_SECRET_VAR。您可以在建置期間使用LOCAL_SECRET_VAR名稱存取變數。

```
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。

匯出的環境變數可與搭配使用， AWS CodePipeline 將環境變數從目前的建置階段匯出至管線中的後續階段。若要取得更多資訊，請參閱《使用指南》中的〈AWS CodePipeline 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 參數存放區機密。

- 建置專案中指定的密碼 Secrets Manager 密碼
- 以 AWS_ 開頭的環境變數。

環/git-credential-helper

選用的映射。用於指示是否 CodeBuild 使用其 Git 憑證助手來提供 Git 憑據。 yes如果使用它。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。在明確的 Proxy 伺服器中，yes針對您的組建設定為，以建立 CloudWatch 記錄檔。預設值為 no。

階段

必要的序列。表示在構建的每個階段 CodeBuild 運行的命令。

> **Note**
>
> 在 buildspec 版本 0.1 中， CodeBuild 在構建環境中的默認 shell 的單獨實例中運行每個命令。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，如果全局`run-as`指定了用戶 1，並且對於`install`階段只有一個`run-as`語句指定了用戶 2，那麼 buildspec 文件中的所有命令都將作為 User 1 運行，除了在`install`階段中運行的命令，這是作為用戶 2 運行。

階段 /*/ 失敗

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`-中止構建。
- `CONTINUE`-繼續下一階段。

如果未指定此屬性，失敗流程會遵循轉移階段，如中所示建置階段轉換。

階段 /*/ 最後

選用的區塊。圖塊中指定的指令會在`finally`圖塊中的`commands`指令之後執行。即使`finally`區塊中的指令失敗，區`commands`塊中的指令也會執行。例如，如果`commands`圖塊包含三個指令，而第一個指令失敗，則 CodeBuild 略過剩餘的兩個指令並執行`finally`圖塊中的任何指令。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。代表在安裝期間 CodeBuild 執行的指令 (如果有的話)。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階段版本。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階段，主要版本後跟著指定 CodeBuild使用該主要版本及其最新次`latest`要版本，或使用最新的主要和次要版本 (例如`ruby: 3.2`、`nodejs: 18.x`、、或`java: latest`)。`.x`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 標準映像檔 4.0，則以下內容指定安裝 Java 版本 17、Python 版本 3 的最新次要版本，以及包含在 Ruby 環境變數中的版本。如需詳細資訊，請參閱 碼頭圖片提供 CodeBuild。

```
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 buildspec 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階段，請 CodeBuild 選擇您使用的映像中可用的預設執行階段。如果您指定一或多個執行階段，則只 CodeBuild 會使用這些執行階段。如果未指定相依執行階段，會 CodeBuild 嘗試為您選擇相依的執行階段。

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。

如需可用執行階段的詳細資訊，請參閱可用的執行階段。

> **Note**
>
> 如果您指定`runtime-versions`區段並使用 Ubuntu 標準映像 2.0 或更新版本以外的映像檔，或 Amazon Linux 2 (AL2) 標準映像 1.0 或更新版本，則組建會發出警告「`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，其中每個純量代表在安裝期間 CodeBuild 執行的單一命令。CodeBuild 依列出的順序從開始到結束執行每個指令，一次執行一個指令。

phases/pre_build

選用的序列。表示在構建之前 CodeBuild 運行的命令（如果有的話）。例如，您可以使用此階段登入 Amazon ECR，或者您可以安裝 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，其中每個純量代表在構建之前 CodeBuild 運行的單個命令。 CodeBuild依列出的順序從開始到結束執行每個指令，一次執行一個指令。

phases/build

選用的序列。代表建置期間 CodeBuild 執行的指令 (如果有的話)。例如，您可能會使用此階段來執行 Mocha、RSpec 或 sbt。

phases/build/commands

如果`build`已指定，則為必要。包含純量序列，其中每個純量代表在構建期間 CodeBuild 運行的單個命令。 CodeBuild 依列出的順序從開始到結束執行每個指令，一次執行一個指令。

phases/post_build

選用的序列。表示在構建後 CodeBuild 運行的命令（如果有的話）。例如，您可以使用 Maven 將構建成品打包到 JAR 或 WAR 文件中，或者您可以將 Docker 映像推送到 Amazon ECR 中。然後，您可以透過 Amazon SNS 傳送建置通知。

phases/post_build/commands

如果`post_build`已指定，則為必要。包含一系列純量，其中每個純量代表在構建後 CodeBuild 運行的單個命令。 CodeBuild 依列出的順序從開始到結束執行每個指令，一次執行一個指令。

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，請使用您的專案名稱和您在格式中指定的名稱來 CodeBuild 建立報表群組`<project-name>-<report-group-name>`。您也可以使用 buildspec 中的環境變數來設定報表群組名稱，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，每個標量代表一個單獨的位置，其中 CodeBuild 可以找到測試文件，相對於原始構建位置或（如果設置）。`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。代表報表檔案格式。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

單位 3

TESTNGXML

TestNG XML

VISUALSTUDIOTRX

Visual Studio TRX

代碼覆蓋率報告

CLOVERXML

三叶草

COBERTURAXML

XML 編輯器

JACOCOXML

JaCoCo XML

SIMPLECOV

SimpleCov JSON

> **Note**
>
> CodeBuild 接受簡單生成的 JSON 代碼覆蓋報告，而不是簡單的 JSON 代碼覆蓋率。

reports/<report-group>/base-directory

選用的映射。代表相對於原始構建位置的一個或多個頂級目錄， CodeBuild 用於確定在哪裡找到原始測試文件。

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。

工藝品

選用的序列。代表在何處 CodeBuild 可以找到組建輸出，以及如何 CodeBuild 準備上傳至 S3 輸出儲存貯體的資訊。例如，如果您正在建立 Docker 映像並將其推送到 Amazon ECR，或者您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 中繼資料有一個名為的 CodeBuild 標頭，x-amz-meta-codebuild-buildarn其buildArn中包含將成品發佈到 Amazon S3 的 CodeBuild 組建。buildArn已新增以允許來源追蹤通知，以及參考從中產生成品的組建。

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可以找到構建輸出加工品，相對於原始構建位置或基本目錄（如果設置）。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。

當您指定組建輸出加工品位置時，CodeBuild 可以在建置環境中尋找原始建置位置。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 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 控制台來創建構建，在 buildspec 文件中指定名稱，並在創建或更新項目時選擇啟用語義版本控制。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
```

- 這是一個人工因素名稱的範例，它使用 CodeBuild 環境變數並附加了人工因素的建立日期。

```
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的人工因素放置在目錄中。在此範例中，組建加工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。代表相對於原始組建位置的一或多個頂層目錄， CodeBuild 用來決定組建輸出加工品中要包含哪些檔案和子目錄。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
```

人工品/排除路徑

選用的映射。代表一個或多個相對於， CodeBuild 將從組建加工品中排除的路徑。base-directory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 (**) 會在所有目錄中比對名稱元件的零個或多個字元。

排除路徑的範例包括：

- 若要從所有目錄中排除檔案："**/*file-name*/**/*"
- 若要排除所有點資料夾："**/.*/**/*"
- 若要排除所有點檔案："**/.*"

人工/啟用符號鏈接

選用。如果輸出類型為ZIP，則指定是否在 ZIP 檔案中保留內部符號連結。如果包含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。

人工/S3 前綴

選用。指定當成品輸出到 Amazon S3 儲存貯體且命名空間類型為時使用的前置詞BUILD_ID。使用時，值區中的輸出路徑為<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的 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**
>
> 即使只有定義次要加工品，此artifacts/files序列始終是必要的。

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 可在何處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/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 可以找到構建輸出加工品，相對於原始構建位置或基本目錄（如果設置）。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建置專案時使用或 AWS SDK 來宣告 Buildspec，則組建規格必須是以 YAML 格式表示的單一字串，以及必要的空白和換行符逸出字元。 AWS CLI下一節提供一個範例。
>
> 如果您使用 CodeBuild 或 AWS CodePipeline 控制台而不是 buildspec.yml 檔案，則只能插入階段的命令。build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 控制台而不是 buildspec.yml 文件來指定構建環境中構建輸出成品的位置。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/**/*'
```

以下是前述 buildspec 的範例，以單一字串表示，以與 AWS CLI、或 SDK 搭配使用。 AWS

```
"version: 0.2\n\nenv:\n  variables:\n    JAVA_HOME: \"/usr/lib/jvm/java-8-openjdk-
amd64\\"\n  parameter-store:\n    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword\n
 phases:\n\n  install:\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  pre_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  build:\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  post_build:\n    commands:\n      - echo
 Entered the post_build phase...\n      - echo Build completed on `date`\n\n reports:
\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階段中指令的範例，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。
- 稍後會在建置命令中使用金鑰參數存放區參考dockerLoginPassword您儲存在 Amazon EC2 Systems Manager 參數存放區中的自訂環境變數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是apt-get update -y和 apt-get install -y maven (安裝 Apache Maven)、mvn install (編譯、測試和封裝原始程式碼至建置輸出成品，並在其內部儲存庫中安裝建置輸出成品)、docker login (使用

與dockerLoginPassword您在 Amazon EC2 Systems Manager 參數存放區中設定的自訂環境變數值對應的密碼登入 Docker) 以及數個echo命令。這裡包含這些echo指令，以顯示 CodeBuild 執行命令的方式以及執行命令的順序。

- files 代表要上傳至組建輸出位置的檔案。在此範例中， CodeBuild 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稱為 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	• environment_variables 已重新命名為 env。 • plaintext 已重新命名為 variables 。 • 已捨棄 artifacts 的 type 屬性。 • 在版本 0.1 中， AWS CodeBuild 在構建環境中的默認 shell 的單獨實例中運行每個構建命令。在 0.2 版中， CodeBuild 在構建環境中的默認 shell 的相同實例中運行所有構建命令。
0.1	這是組建規格格式的初始定義。

Batch 量生成構建規範參考

本主題包含批處理構建屬性的構建規範參考。

批次

選用的映射。項目的批量生成設置。

批次/快速失敗

選用。指定一個或多個生成任務失敗時批處理生成的行為。

`false`

預設值。所有正在運行的構建都將完成。

`true`

當其中一個構建任務失敗時，所有正在運行的構建都將停止。

默認情況下，所有批處理構建任務都使用生成設置（如env和phases（在 Buildspec 檔案中指定）。您可以通過指定不同的env值或不同的構建規範文件batch/*<batch-type>*/buildspec參數。

的內容batch屬性會因指定的批次構建類型而異。可能的批處理構建類型有：

- batch/build-graph
- batch/build-list
- batch/build-matrix

batch/build-graph

定義建置圖表。生成圖形定義了一組與批處理中的其他任務具有依賴關係的任務。如需詳細資訊，請參閱 構建圖。

此元素包含一系列的建置任務。每個生成任務都包含下列屬性。

標識符

必要。任務的識別符。

BuildSpec

選用。用於此任務的 Buildspec 檔案的路徑和名稱。如果未指定此參數，將使用當前 Buildspec 檔案。

偵錯會話

選用。布林值類型，用以指示是否啟用此批次生成的會話偵錯。如需工作階段調試的詳細資訊，請參在工作階段管理員中檢視正在執行。

`false`

會話調試已禁用。

`true`

會話調試已啟用。

依

選用。此任務所依賴的任務標識符數組。在完成這些任務之前，此任務將不會運行。

env

選用。任務的構建環境將覆蓋。此程式碼可包含下列屬性：

運算類型

用於任務的計算類型的標識符。請參閱computeType在the section called “建置環境運算模式和類型”以獲取可能的值。

映像

用於任務的映像識別符。請參閱映像識別符在the section called “碼頭圖片提供 CodeBuild”以獲取可能的值。

特權模式

布林值類型，用以指示是否在 Docker 容器中執行 Docker 協助程式。設定為`true`只有使用 Builker 項目來建置 Docker 映像時。否則，嘗試與 Docker 協助程式互動的組建會失敗。預設設定為 `false`。

類型

用於任務的環境類型的標識符。請參閱環境類型在the section called “建置環境運算模式和類型”以獲取可能的值。

變數

構建環境中將出現的環境變量。如需詳細資訊，請參閱 env/variables。

忽略失敗

選用。一個布爾值，指示是否可以忽略此構建任務的失敗。

`false`

預設值。如果此生成任務失敗，批處理構建將失敗。

`true`

如果此生成任務失敗，批處理構建仍然可以成功。

以下是建置圖形 Buildspec 項目範例：

```
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
```

batch/build-list

定義構建列表。構建列表用於定義多個並行運行的任務。如需詳細資訊，請參閱 組建清單。

此元素包含一系列的建置任務。每個生成任務都包含下列屬性。

標識符

必要。任務的識別符。

BuildSpec

選用。用於此任務的 Buildspec 檔案的路徑和名稱。如果未指定此參數，將使用當前 Buildspec 檔案。

偵錯會話

選用。布林值類型，用以指示是否啟用此批次生成的會話偵錯。如需工作階段調試的詳細資訊，請參[在工作階段管理員中檢視正在執行]。

`false`

會話調試已禁用。

`true`

會話調試已啟用。

env

選用。任務的構建環境將覆蓋。此程式碼可包含下列屬性：

運算類型

用於任務的計算類型的標識符。請參閱computeType在[the section called “建置環境運算模式和類型”]以獲取可能的值。

映像

用於任務的映像識別符。請參閱映像識別符在[the section called “碼頭圖片提供 CodeBuild”]以獲取可能的值。

特權模式

布林值類型，用以指示是否在 Docker 容器中執行 Docker 協助程式。設定為`true`只有使用 Builker 項目來建置 Docker 映像時。否則，嘗試與 Docker 協助程式互動的組建會失敗。預設設定為 `false`。

類型

用於任務的環境類型的標識符。請參閱環境類型在[the section called “建置環境運算模式和類型”]以獲取可能的值。

變數

構建環境中將出現的環境變量。如需詳細資訊，請參閱 env/variables。

忽略失敗

選用。一個布爾值，指示是否可以忽略此構建任務的失敗。

`false`

預設值。如果此生成任務失敗，批處理構建將失敗。

`true`

如果此生成任務失敗，批處理構建仍然可以成功。

以下是建置列表 Buildspec 項目範例：

```
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
```

batch/build-matrix

定義建置矩陣。構建矩陣定義了具有並行運行的不同配置的任務。CodeBuild 為每個可能的配置組合創建一個單獨的構建。如需詳細資訊，請參閱 建立矩陣。

靜態的

靜態屬性適用於所有構建任務。

忽略失敗

選用。一個布爾值，指示是否可以忽略此構建任務的失敗。

`false`

預設值。如果此生成任務失敗，批處理構建將失敗。

`true`

如果此生成任務失敗，批處理構建仍然可以成功。

env

選用。構建環境將覆蓋所有任務。

特權模式

布林值類型，用以指示是否在 Docker 容器中執行 Docker 協助程式。設定為`true`只有使用 Builker 項目來建置 Docker 映像時。否則，嘗試與 Docker 協助程式互動的組建會失敗。預設設定為 `false`。

類型

用於任務的環境類型的標識符。請參閱環境類型在the section called “建置環境運算模式和類型”以獲取可能的值。

動態

動態屬性定義構建矩陣。

BuildSpec

選用。一個數組，其中包含要用於這些任務的 buildspec 文件的路徑和文件名。如果未指定此參數，將使用當前 Buildspec 檔案。

env

選用。這些任務的構建環境將覆蓋。

運算類型

一個數組，其中包含要用於這些任務的計算類型的標識符。請參閱computeType在the section called “建置環境運算模式和類型”以獲取可能的值。

映像

包含用於這些任務的圖像標識符的數組。請參閱映像識別符在the section called “碼頭圖片提供 CodeBuild”以獲取可能的值。

變數

一個數組，其中包含將出現在這些任務的構建環境中的環境變量。如需詳細資訊，請參閱 env/variables。

以下是建置矩陣 Buildspec 項目範例：

```
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，請參閱建立矩陣。

AWS CodeBuild 建置環境參考

當您呼叫 AWS CodeBuild 以執行組建時，您必須提供組建環境的相關資訊。建置環境代表作業系統、程式設計語言執行階段，以及CodeBuild用來執行組建的工具的組合。如需建置環境如何運作的資訊，請參閱CodeBuild 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。

當您向 CodeBuild 提供建置環境資訊時，您需要指定支援儲存庫類型的 Docker 影像識別符。其中包括CodeBuild Docker 映像儲存庫、Docker Hub 中公開可用的映像檔，以及您的AWS帳戶有權存取的 Elastic Container Registry (Amazon ECR) 儲存庫。

- 我們建議您使用 CodeBuild Docker 影像儲存庫中存放的 Docker 影像，因為它們已經過最佳化而適合搭配使用相關服務。如需詳細資訊，請參閱碼頭圖片提供 CodeBuild。
- 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。

- 若要了解如何使用存放在AWS帳戶中 Amazon ECR 儲存庫中的 Docker 映像，請參閱Amazon ECR 樣本。

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

碼頭圖片提供 CodeBuild

支援的映像是中可用映像的最新主要版本，並以次要 CodeBuild 和修補程式版本更新進行更新。CodeBuild 透過在機器的 Amazon 機器映像 (AMI) 中快取所支援映像的組建，將組建的佈建持續時間最佳化。如果您想要從快取中獲益，並將組建的佈建持續時間降至最低，請在 CodeBuild 主控台的 Image version 區段中選取 [永遠使用此執行階段版本的最新映像檔，而不是更精細的版本，例如`aws/codebuild/amazonlinux2-x86_64-standard:4.0-1.0.0`。

CodeBuild 經常更新 Docker 映像列表以添加最新圖像並棄用舊映像。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 [建立組建專案精靈] 或 [編輯組建專案] 頁面中，針對 [環境映像] 選擇 [受管理的映像]。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) 。
- 對於 AWS CLI，請執行下`list-curated-environment-images`列命令：

```
aws codebuild list-curated-environment-images
```

- 對於 AWS SDK，請調用目標編程語言的`ListCuratedEnvironmentImages`操作。如需更多資訊，請參閱AWS 開發套件和工具參考。

Windows 伺服器核心 2019 平台的基本映像檔僅適用於下列地區：

- 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)

EC2 運算映像檔

AWS CodeBuild 支援中適用於 EC2 運算的下列 Docker 映像檔。 CodeBuild

平台	映像識別符	定義
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:4.0`	AL2/標準/4.0
Amazon Linux 2023	`aws/codebuild/amazonlinux2-x86_64-standard:5.0`	AL2/標準/5.0
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:corretto8`	AL2/標準/相關 8
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:corretto11`	AL2/標準/相關 11
Amazon Linux 2	`aws/codebuild/amazonlinux2-aarch64-standard:2.0`	AL2/aarch64/標準/2.0
Amazon Linux 2023	`aws/codebuild/amazonlinux2-aarch64-standard:3.0`	AL2/aarch64/標準/3.0
Ubuntu 20.04	`aws/codebuild/standard:5.0`	支持/標準/5.0

平台	映像識別符	定義
Ubuntu	aws/codebuild/standard:6.0	支持/標準/6.0
Ubuntu	aws/codebuild/standard:7.0	支持/標準 /7.0
視窗伺服器核心	aws/codebuild/windows-base:2019-1.0	N/A
視窗伺服器核心	aws/codebuild/windows-base:2019-2.0	N/A
視窗伺服器核心	aws/codebuild/windows-base:2019-3.0	N/A
視窗伺服器核心	aws/codebuild/windows-base:2022-1.0	N/A

Lambda 運算映像

AWS CodeBuild 支援下列可用於中 AWS Lambda CodeBuild運算的 Docker 映像檔。

`aarch64` 架構

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet6	阿尔羊达/阿尔64/網6
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet8	阿尔羊达/阿尔64/網8

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.21	阿尔兰布达/阿尔奇 64/go1.21
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto11	阿尔羊达/阿阿尔64/科雷特托11
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto17	阿尔羊达/阿阿尔64/科雷特托17
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto21	阿尔羊达/阿阿尔64/科雷特托21
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs18	阿尔羊达/阿尔64/nodejs18
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs20	阿尔羊达/阿尔64/nodejs20
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.11	阿爾蘭達/阿施 64 /3.11

平台	映像識別符	定義
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.12	阿爾蘭達/阿施 64 /3.12
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.2	阿尔兰达/阿阿尔盖 64 /紅寶石 3.2

x86_64 架構

平台	映像識別符	定義
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet6	阿爾羊達/x86_64/網路6
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet8	阿爾羊達/x86_64/網路8
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.21	阿尔兰布达
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto11	阿爾羊田/x86_64/科雷特托11
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lam	阿爾羊達/x86_64/科雷特托17

平台	映像識別符	定義
	bda-standard:corretto17	
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21	阿爾羊田/x86_64/科雷特托21
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs18	阿尔羊田/x86_64/nodejs18
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20	阿爾羊達/x86_64/nodejs20
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.11	阿爾蘭達/x86_64 /3.11
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12	阿爾羔羊達/x86_64 /3.12
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.2	阿爾蘭布達/x86_64 /紅寶石3.2

已過時的圖

已取代的影像是不再由快取或更新的影像 CodeBuild。已取代的映像檔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 項目配置為使用較舊的映像版本，則配置過程將下載此 docker 映像並使用它來創建容器化運行時環境，這可能會增加佈建持續時間和整體構建持續時間。

CodeBuild 已棄用以下 Docker 圖像。您仍然可以使用這些映像，但它們不會在構建主機上緩存，並且會導致更長的佈建時間。

平台	映像識別符	定義	取代日期
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard: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-standard:1.0`	al2/aarch64/standard/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-standard:2.0`	al2/standard/2.0	2022 年 6 月 30 日

主題

- 可用的執行階段
- 執行時間版本

可用的執行階段

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 buildspec 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階段，請 CodeBuild 選擇您使用的映像中可用的預設執行階段。如果您指定一或多個執行階段，則只 CodeBuild 會使用這些執行階段。如果未指定相依執行階段，會 CodeBuild 嘗試為您選擇相依的執行階段。如需詳細資訊，請參閱 Specify runtime versions in the buildspec file。

主題

映像執行階段

下表包含可用的執行階段以及支援這些執行階段的標準 Linux 映像檔。

Ubuntu 和 Amazon Linux 平台運行時

執行時間名稱	版本	映像
dotnet	3.1	Amazon Linux 2 年級 64 標準:2.0 标准:5.0
	5.0	标准:5.0
	6.0	Amazon Linux 2 x86_64 Lambda 標準:網站 Amazon Linux 2 年級 64 Lambda 標準:網站 6 亚 Amazon 标准 亚 Amazon 标准 亚 Amazon 标准:3.0 標準:6.0

執行時間名稱	版本	映像
		標準:7.0
	8.0	亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0
golang	1.12	Amazon Linux 2 年級 64 標準:2.0
	1.13	Amazon Linux 2 年級 64 標準:2.0
	1.14	Amazon Linux 2 年級 64 標準:2.0
	1.15	标准:5.0
	1.16	标准:5.0
	1.18	亞 Amazon 标准 標準:6.0
	1.20	亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0

執行時間名稱	版本	映像
	1.21	Amazon 2 x86_64 Lambda 標準: Amazon Linux 2 年級 64 Lambda 標準：去 1.21 亚 Amazon 标准 亚 Amazon 标准:3.0 標準:7.0
	1.22	亚 Amazon 标准 標準:7.0
java	corretto8	Amazon Linux 2 x86_64 標準:相關 亚 Amazon 标准 Amazon Linux 2 年級 64 標準:2.0 标准:5.0 標準:7.0

執行時間名稱	版本	映像
	corretto11	Amazon Linux 2 x86_64 標準:更新 Amazon Linux 2 x86_64 Lambda 標準:相關 亚 Amazon 标准 Amazon Linux 2 升級 64 Lambda 標準:相關 11 Amazon Linux 2 年級 64 標準:2.0 标准:5.0 標準:7.0
	科雷特托 17	Amazon Linux 2 x86_64 Lambda 標準:相關 Amazon Linux 2 年級 64 Lambda 標準:相關 17 亚 Amazon 标准 亚 Amazon 标准 亚 Amazon 标准:3.0 標準:6.0 標準:7.0

執行時間名稱	版本	映像
	科雷特托 21	Amazon Linux 2 x86_64 Lambda 標準:公司 Amazon Linux 2 年級 64 Lambda 標準:相關 21 亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0
nodejs	10	Amazon Linux 2 年級 64 標準:2.0
	12	Amazon Linux 2 年級 64 標準:2.0 标准:5.0
	14	标准:5.0
	16	亞 Amazon 标准 標準:6.0
	18	Amazon 2 x86_64 Lambda 標準: Amazon Linux 2 年級 64 Lambda 標準:9 月 18 亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0

執行時間名稱	版本	映像
	20	Amazon 2 x86_64 Lambda 標準: Amazon Linux 2 年級 64 Lambda 標準:9 月 20 亚 Amazon 标准 亚 Amazon 标准:3.0 標準:7.0
php	7.3	Amazon Linux 2 年級 64 標準:2.0 标准:5.0
	7.4	Amazon Linux 2 年級 64 標準:2.0 标准:5.0
	8.0	标准:5.0
	8.1	亚 Amazon 标准 亚 Amazon 标准:3.0 標準:6.0
	8.2	亚 Amazon 标准 亚 Amazon 标准:3.0 標準:7.0

執行時間名稱	版本	映像
	8.3	亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0
python	3.7	Amazon Linux 2 年級 64 標準:2.0 标准:5.0
	3.8	Amazon Linux 2 年級 64 標準:2.0 标准:5.0
	3.9	亞 Amazon 标准 亞 Amazon 标准 Amazon Linux 2 年級 64 標準:2.0 标准:5.0 標準:7.0
	3.10	亞 Amazon 标准 標準:6.0 標準:7.0

執行時間名稱	版本	映像
	3.11	Amazon Linux 2 x86_64 Lambda 標準:蟒蛇 Amazon Linux 2 AARCH64 Lambda 標準:蟒蛇 3.11 亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0
	3.12	Amazon Linux 2 x86_64 Lambda 標準:蟒蛇 Amazon Linux 2 AARCH64 Lambda 標準:蟒蛇 3.12 亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0
ruby	2.6	Amazon Linux 2 年級 64 標準:2.0 标准:5.0
	2.7	Amazon Linux 2 年級 64 標準:2.0 标准:5.0

執行時間名稱	版本	映像
	3.1	亞 Amazon 标准 亞 Amazon 标准 標準:6.0 標準:7.0
	3.2	Amazon Linux 2 x86_64 Lambda 標準:紅寶石 3.2 Amazon Linux 2 阿爾查 64 Lambda 標準:紅寶石 3.2 亞 Amazon 标准 亞 Amazon 标准:3.0 標準:7.0
	3.3	亞 Amazon 标准 標準:7.0

視窗映像執行階段

Windows 伺服器核心 2019 年的基本映像包含下列執行階段。

視窗平台執行階段

執行時間名稱	視窗服務器核心 2019 標準:1.0 版本	視窗伺服器核心 2019 標準:2.0 版本	視窗服務器核心 2019 標準:3.0 版本
dotnet	3.1 5.0	3.1 6.0 7.0	6.0 7.0 8.0
網點網 SDK	3.1	3.1	8.0

執行時間名稱	視窗服務器核心 2019 標準:1.0 版本	視窗伺服器核心 2019 標準:2.0 版本	視窗服務器核心 2019 標準:3.0 版本
	5.0	6.0 7.0	
golang	1.14	1.18	1.21
搖籃	6.7	7.6	8.5
java	科雷特托 11	科雷特托 11 科雷特托 17	科雷特托 21 號
Maven	3.6	3.8	3.9
nodejs	14.15	16.19	20.11
php	7.4	8.1	8.3
動力外殼	7.1	7.2	7.4
python	3.8	3.10	3.12
ruby	2.7	3.1	3.3

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。並非所有執行階段版本都適用於所有影像。自訂影像也不支援執行階段版本選擇。如需詳細資訊，請參閱 可用的執行階段。如果您想要安裝並使用自訂執行階段版本，而不是預先安裝的執行階段版本，請參閱自訂運行時版本。

Ubuntu 和 Amazon Linux 2 平台運行時版本

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
android	28	`android: 28`	`android: 28.x`	`android: latest`

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
	29	`android: 29`	`android: 29.x`	
dotnet	3.1	`dotnet: 3.1`	`dotnet: 3.x`	`dotnet: latest`
	5.0	`dotnet: 5.0`	`dotnet: 5.x`	
	6.0	`dotnet: 6.0`	`dotnet: 6.x`	
	8.0	`dotnet: 8.0`	`dotnet: 8.x`	
golang	1.12	`golang: 1.12`	`golang: 1.x`	`golang: latest`
	1.13	`golang: 1.13`		
	1.14	`golang: 1.14`		
	1.15	`golang: 1.15`		
	1.16	`golang: 1.16`		
	1.18	`golang: 1.18`		
	1.20	`golang: 1.20`		
	1.21	`golang: 1.21`		
	1.22	`golang: 1.22`		
java	corretto8	`java: corretto`	`java: corretto .x`	`java: latest`
	corretto11	`java: corretto 1`	`java: corretto 1.x`	
	科雷特托 17	`java: corretto 7`	`java: corretto 7.x`	

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
	科雷特托 21	java: corretto 1	java: corretto 1.x	
nodejs	10	nodejs: 10	nodejs: 10.x	nodejs: latest
	12	nodejs: 12	nodejs: 12.x	
	14	nodejs: 14	nodejs: 14.x	
	16	nodejs: 16	nodejs: 16.x	
	18	nodejs: 18	nodejs: 18.x	
	20	nodejs: 20	nodejs: 20.x	
php	7.3	php: 7.3	php: 7.x	php: latest
	7.4	php: 7.4		
	8.0	php: 8.0	php: 8.x	
	8.1	php: 8.1		
	8.2	php: 8.2		
	8.3	php: 8.3		
python	3.7	python: 3.7	python: 3.x	python: latest
	3.8	python: 3.8		
	3.9	python: 3.9		
	3.10	python: 3.10		
	3.11	python: 3.11		
	3.12	python: 3.12		

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
ruby	2.6	`ruby: 2.6`	`ruby: 2.x`	`ruby: latest`
	2.7	`ruby: 2.7`		
	3.1	`ruby: 3.1`	`ruby: 3.x`	
	3.2	`ruby: 3.2`		
	3.3	`ruby: 3.3`		

您可以使用構建規範來安裝其他組件（例如，阿帕奇 Maven 的，阿帕奇螞蟻，摩卡，RSpec 的，或類似的）在`install`構建階段。 AWS CLI如需詳細資訊，請參閱 Buildspec 範例。

自訂運行時版本

您可以安裝並使用您選擇的自訂版本，而不是在 CodeBuild-managed 映像檔中使用預先安裝的執行階段版本。下表列出可用的自訂執行階段以及如何指定它們。

> **Note**
>
> 只有 Ubuntu 和 Amazon Linux 映像檔才支援自訂執行階段版本選擇。

自訂運行時版本

執行時間名稱	語法	範例
dotnet	`<major>.<minor>.<patch>`	`5.0.408`
golang	`<major>.<minor>`	`1.19`
	`<major>.<minor>.<patch>`	`1.19.1`
java	`corretto<major>`	`corretto15`
nodejs	`<major>`	`14`
	`<major>.<minor>`	`14.21`

執行時間名稱	語法	範例
	<major>.*<minor>*.*<patch>*	14.21.3
php	*<major>*.*<minor>*.*<patch>*	8.0.30
python	*<major>*	3
	<major>.*<minor>*	3.7
	<major>.*<minor>*.*<patch>*	3.7.16
ruby	*<major>*.*<minor>*.*<patch>*	3.0.6

自定義運行時構建規範示例

以下是指定自定義運行時版本的 buildspec 的示例。

```
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

在中 CodeBuild，您可以指定用來執行組建的計算和執行階段環境映像檔。 CodeBuild 計算是指由管理及維護的運算引擎 (CPU、記憶體和作業系統) CodeBuild。執行階段環境映像檔是在您選擇的運算平台之上執行的容器映像檔，其中包含組建可能需要的額外工具，例如 AWS CLI.

主題

- 關於運算模式
- 關於環境類型

關於運算模式

CodeBuild 提供下列運算模式：

- EC2
- AWS Lambda

EC2 在建置期間提供最佳化的彈性，並 AWS Lambda 提供最佳化的啟動速度。 AWS Lambda 由於較低的啟動延遲，因此支援更快的組建。 AWS Lambda 也會自動擴展，因此構建不會在隊列中等待運行。如需詳細資訊，請參閱 在中使用 AWS Lambda 計算 AWS CodeBuild。

關於環境類型

AWS CodeBuild 為建置環境提供下列可用記憶體、vCPUs 和磁碟空間，適用於 EC2 運算模式：

運算類型	環境 computeType 值	環境類型值	記憶體	vCPU	磁碟空間
小臂	BUILD_GENERAL1_SMALL	ARM_CONTAINER	4 GB	2	50 GB
手臂大	BUILD_GENERAL1_LARGE	ARM_CONTAINER	16 GB	8	50 GB
小型	BUILD_GENERAL1_SMALL	LINUX_CONTAINER	3 GB	2	64 GB
中型	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER	7 GB	4	128 GB
大型	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	15 GB	8	128 GB

運算類型	環境 computeType 值	環境類型值	記憶體	vCPU	磁碟空間
大型	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	70 英鎊	36	256 GB
大型	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	145 GB	72	824 GB (SSD)
小型 GPU	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GB	4	220 GB
大型 GPU	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GB	32	50 GB
視窗中	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER	7 GB	4	128 GB
大型窗戶	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER	15 GB	8	128 GB

[1] 會快取此映像類型的最新版本。如果您指定更具體的版本，則 CodeBuild 佈建該版本而不是快取版本。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/amazonlinux2-x86_64-standard:5.0`，而非更精細的版本，例如 `aws/codebuild/amazonlinux2-x86_64-standard:5.0-1.0.0`。

AWS CodeBuild 為建置環境提供下列可用記憶體和磁碟空間，以供 AWS Lambda 運算模式使用：

運算類型	環境 computeType 值	環境類型值	記憶體	磁碟空間
手臂 Lambda 1GB	BUILD_LAMBDA_1GB	ARM_LAMBDA_CONTAINER	1 GB	10 GB
手臂 Lambda	BUILD_LAMBDA_2GB	ARM_LAMBDA_CONTAINER	2 GB	10 GB
手臂 Lambda	BUILD_LAMBDA_4GB	ARM_LAMBDA_CONTAINER	4 GB	10 GB
手臂 Lambda 8GB	BUILD_LAMBDA_8GB	ARM_LAMBDA_CONTAINER	8 GB	10 GB
手臂 Lambda	BUILD_LAMBDA_10GB	ARM_LAMBDA_CONTAINER	10 GB	10 GB
Lambda	BUILD_LAMBDA_1GB	LINUX_LAMBDA_CONTAINER	1 GB	10 GB
Lambda	BUILD_LAMBDA_2GB	LINUX_LAMBDA_CONTAINER	2 GB	10 GB
Lambda	BUILD_LAMBDA_4GB	LINUX_LAMBDA_CONTAINER	4 GB	10 GB
Lambda	BUILD_LAMBDA_8GB	LINUX_LAMBDA_CONTAINER	8 GB	10 GB

運算類型	環境 computeType 值	環境類型值	記憶體	磁碟空間
Lambda	BUILD_LAMBDA_10GB	LINUX_LAMBDA_CONTAINER	10 GB	10 GB

使用其他環境類型時，建議您使用快取的映像來減少建置時間。

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 [建立組建專案精靈] 或 [編輯組建專案] 頁面的 [環境] 中展開 [其他設定]，然後從 [計算類型] 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物件的computeType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 SDK，請針對目標程式設計語言呼叫相當於CreateProject或UpdateProject作業，並指定environment物件computeType值的對等值。如需更多資訊，請參閱 AWS 開發套件和工具參考。

某些環境和運算類型有區域可用性限制：

- Linux GPU 小型 (LINUX_GPU_CONTAINER) 的運算類型僅適用於下列區域：
  - 美國東部 (維吉尼亞北部)
  - 美國西部 (奧勒岡)
  - 亞太區域 (東京)
  - 加拿大 (中部)
  - 歐洲 (法蘭克福)
  - 歐洲 (愛爾蘭)
  - 歐洲 (倫敦)
- Linux GPU 大型 (LINUX_GPU_CONTAINER) 的運算類型僅適用於下列區域：
  - 美國東部 (俄亥俄)

- 美國東部 (維吉尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (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)

- 環境類型ARM_CONTAINER僅適用於以下區域：
  - 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)
  - 亞太區域 (香港)
  - 亞太區域 (雅加達)
  - 亞太區域 (海德拉巴)
  - 亞太區域 (孟買)
  - 亞太區域 (大阪)
  - 亞太區域 (首爾)
  - 亞太區域 (新加坡)
  - 亞太區域 (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)
- 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)

- 運算類型`BUILD_GENERAL1_2XLARGE`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)
  - 亞太區域 (海德拉巴)
  - 亞太區域 (香港)
  - 亞太區域 (雅加達)
  - 亞太區域 (墨爾本)
  - 亞太區域 (孟買)
  - 亞太區域 (首爾)
  - 亞太區域 (新加坡)
  - 亞太區域 (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)
- 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 (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)
  - 南美洲 (聖保羅)

對於運算類型 `BUILD_GENERAL1_2XLARGE`，最多可支援 100 GB 未壓縮的 Docker 影像。

> **Note**
>
> 對於自定義構建環境映像，無論運算類型如何，都 CodeBuild 支持 Linux 和 Windows 中最多 50 GB 未壓縮的 Docker 映像。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 存取組建容器中的更多空間。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**
>
> 依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 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，讓 AWS CodeBuild 在組建生命週期期間 (例如，安裝組建相依性，並測試及編譯您的來源碼) 於組建環境中執行。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。如需詳細資訊，請參閱 建立的規格參考 CodeBuild。
- 使用 CodeBuild 主控台來建立組建專案。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 或 AWS 軟體開發套件來建立組建專案，或變更組建專案的設定。參考來源碼 (其中包含 Buildspec 檔案與您的命令)，或者指定單一字串以包含相同 Buildspec 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的設定。
- 使用 AWS CLI 或 AWS 軟體開發套件開始組建，並指定 Buildspec 檔案，或單一字串以包含相同 Buildspec 檔案的內容。如需詳細資訊，請參閱 執行建置 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 會在組建環境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稱為 `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，錯誤可能會比在默認 shell 的自身執行個體中執行個體中執行個體中執行個體中執行個體中執行個體時更難故障診斷。

在 Windows Server Core 映像中執行的命令會使用 PowerShell 殼層。

建置環境中的環境變數

AWS CodeBuild 提供多種環境變數，可讓您在組建命令中使用：

AWS_DEFAULT_REGION

執行組建的AWS區域 (例如`us-east-1`)。此環境變數主要由 AWS CLI 使用。

AWS_REGION

執行組建的AWS區域 (例如`us-east-1`)。此環境變數主要由 AWS 軟體開發套件使用。

代碼生成批處理構建標識符

批次建置中組建的識別碼。這在批處理構建規格中指定。如需詳細資訊，請參閱the section called "Batch 次 BuildSpec 參考"。

程式碼建置程式碼

建置的 Amazon Resource Name () (ARN`arn:aws:codebuild:`*region-ID*`:`*account-ID*`:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`)。

程式碼建置 ID

組建的CodeBuild識別碼 (例如，`codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`)。

程式碼建置映像檔

組CodeBuild建映像檔識別碼 (例如，`aws/codebuild/standard:2.0`)。

程式碼建置編號

專案的建置。

程式碼建置成功

目前的組建是否成功。設定為 0 表示建置失敗；設定為 1 表示建置成功。

程式碼建置啟動器

啟動組建的實體。如果是由 CodePipeline 開始建置，則此為管道的名稱 (例如 codepipeline/my-demo-pipeline)。如果使用者啟動組建，這就是使用者的名稱 (例如MyUserName)。如果是由 CodeBuild 的 Jenkins 外掛程式開始建置，則此為 CodeBuild-Jenkins-Plugin 字串。

程式碼建置程式碼

用來加密組建輸出成品的AWS KMS金鑰識別碼 (例如，arn:aws:kms:*region-ID*:*account-ID*:key/*key-ID*或alias/*key-alias*)。CodeBuild

代碼生成日誌路徑

組建的CloudWatch記錄檔資料流名稱。

代碼生成器公共構建網址

在公共構建網站上此版本的構建結果的 URL。只有在建置專案已啟用公開組建時，才會設定此變數。如需詳細資訊，請參閱中的公共組建專案AWS CodeBuild。

程式碼建置 _ 解析來源版本

組建原始程式碼的版本識別碼。內容取決於源代碼存儲庫：

CodeCommit,GitHub,GitHub 企業伺服器和比特桶

此變數包含提交 ID。

CodePipeline

此變數包含由提供的來源修訂版本CodePipeline。

如果CodePipeline無法解析來源修訂，例如來源是未啟用版本控制的 Amazon S3 儲存貯體，則不會設定此環境變數。

Amazon S3

未設定此變數。

如果適用，CODEBUILD_RESOLVED_SOURCE_VERSION變數只能在DOWNLOAD_SOURCE階段之後使用。

程式碼建置來源網址

輸入成品或原始程式碼儲存庫的 URL。對於 Amazon S3，s3://其後是存儲桶名稱和輸入成品的路徑。對於CodeCommit和GitHub，這是存放庫的複製 URL。如果構建源於CodePipeline，則此環境變量可能是空的。

對於次要來源，次要來源存放庫 URL 的環境變數
為CODEBUILD_SOURCE_REPO_URL_*<sourceIdentifier>*，其中*<sourceIdentifier>*是您建立的來源識別碼。

程式碼建置來源版本

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識別碼。
- 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。
- 針對GitHubGitHub，為遞交 ID、和 Bitbucket，或與您想要建置之原始程式碼版本關聯的標籤名稱。

> **Note**
>
> 對於由 webhook 提取請求事件觸發的GitHub或GitHub企業服務器構建，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元。如需詳細資訊，請參閱多個輸入來源和輸出成品範例。

程式碼建置程式碼 _ 目錄

CodeBuild用於組建的目錄路徑 (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碼。如需詳細資訊，請參閱多個輸入來源和輸出成品範例。

程式碼建置啟動時間

指定為 Unix 時間戳記的構建的開始時間，以毫秒為單位。

代碼生成 _ 網絡掛鉤 _ 帳戶 ID

觸發 webhook 事件之使用者的帳戶識別碼。

代碼生成 _ 網絡掛鉤 _ 基礎 _ 參考

觸發當前構建的 webhook 事件的基本引用名稱。對於提取請求，這是分支參考。

代碼生成網絡掛鉤事件

觸發當前構建的 webhook 事件。

代碼生成 _ 合併 _ 提交

用於構建的合併提交的標識符。當 Bitbucket 提取請求與壁球策略合併並且拉取請求分支被關閉時設置此變量。在這種情況下，原始的提取請求提交不再存在，因此此環境變量包含壓縮合併提交的標識符。

代碼生成器提交網絡掛鉤

觸發當前構建的 webhook 推送事件之前的最新提交的 ID。

代碼生成 _ 網絡掛鉤 _ 引用

觸發當前構建的 webhook 事件的頭引用名稱。它可以是分支參考或標籤參考。

程式碼建置 _ 網頁掛鉤觸發器

顯示觸發組建的 Webhook 事件。此變數僅適用 Webhook 觸發的建置。該值是從發送到 GitHub，GitHub企業服務器或 BitbucketCodeBuild 的有效負載進行解析。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`*。

家

此環境變數一律設定為`/root`。

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：

- CodePipeline 搭配使用 CodeBuild
- 建立組建專案
- 變更建置專案的設定
- 執行建置

- 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)。除了上方列出的例外，以 `CODEBUILD_` 開頭的環境變數僅供 CodeBuild 內部使用。因此，您不應將它們用在建置命令中。

> **Important**
>
> 我們強烈建議使用環境變數來儲存敏感值，特別是AWS存取金鑰 ID。您可以使用 CodeBuild 主控台和 AWS CLI 等工具，以純文字顯示環境變數。
> 我們建議您將敏感值存放在 Amazon EC2 Systems Manager 參數存放區，然後從您的組建規格擷取這些值。若要存放機密值，請參閱 Amazon EC2 Systems Manager 使用者指南中的系統管理員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。

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)
```

在本機執行組建AWS CodeBuild代理人

您可以使用AWS CodeBuild要執行的代理CodeBuild建置在本機電腦上。有可用於 x86_64 和 ARM 平台的代理程式。

您也可以訂閱以在發佈新版代理程式時接收通知。

先決條件

在開始之前，您需要執行以下操作：

- 在您的本機電腦上安裝 Git。
- 安裝與設定碼頭工人在您的本地計算機上。

設置構建映像

您只需要在第一次執行代理程式或映像變更時設定組建映像。

若要設定組建映像

1. 如果你想使用一個精選的亞馬遜 Linux 2 映像，你可以從CodeBuild公共亞馬遜 ECR 存儲庫https://gallery.ecr.aws/codebuild/amazonlinux2-x86_64-standard使用以下命令：

   ```
   $ docker pull public.ecr.aws/codebuild/amazonlinux2-x86_64-standard:4.0
   ```

   或者，如果您想要使用其他 Linux 映像檔，請執行下列步驟：

   a. 克隆CodeBuild圖片回購：

      ```
      $ git clone https://github.com/aws/aws-codebuild-docker-images.git
      ```

   b. 切換到圖像目錄。在此範例中，請使用`aws/codebuild/standard:5.0`圖像：

      ```
      $ cd aws-codebuild-docker-images/ubuntu/standard/5.0
      ```

   c. 建立映像檔。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 ARM 版本的代理程式，請執行下列命令：

   ```
   $ docker pull public.ecr.aws/codebuild/local-builds:aarch64
   ```

3. 該CodeBuild代理程式可從https://gallery.ecr.aws/codebuild/local-builds。

   x86_64 版代理程式的安全雜湊演算法 (SHA) 簽章為：

   ```
   sha256:fac17c6d6c3cb500f6e7975887de1e41d29a9e70a86d6f49f76a2beacfcf967e
   ```

   ARM 版本的代理程式的 SHA 簽名為：

   ```
   sha256:57a5dfda63be50edce13dea16dcd5e73e8d8559029658ba08b793c9a7adc68c7
   ```

   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，請執行下列命令並尋找下方的 SHARepoDigests:

   ```
   $ docker inspect public.ecr.aws/codebuild/local-builds:latest
   ```

運行CodeBuild代理人

若要執行CodeBuild代理人

1. 切換到包含構建項目源代碼的目錄。
2. 下載codebuild_build.sh腳本：

   ```
   $ curl -O  https://raw.githubusercontent.com/aws/aws-codebuild-docker-images/
   master/local_builds/codebuild_build.sh
   $ chmod +x codebuild_build.sh
   ```

3. 運行codebuild_build.sh腳本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
   ```

   取代*<container-image>*與容器映像的名稱，例如aws/codebuild/standard:5.0或者public.ecr.aws/codebuild/amazonlinux2-x86_64-standard:4.0。

   指令碼會啟動組建映像，並在目前目錄中的專案上執行組建。若要指定建置專案的位置，請加入-s *<build project directory>*腳本命令的選項。

接收新 CodeBuild 代理程式版本的通知

您可以訂閱 Amazon SNS 通知，以便在新版本的AWS CodeBuild已釋放代理程式。

若要訂閱CodeBuild代理通知

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，請變更AWS區域至美國東部 (維吉尼亞北部)。您必須選擇AWS區域，因為您訂閱的 Amazon SNS 通知是在此區域中建立的。
3. 在導覽窗格中，選擇 Subscriptions (訂閱)。
4. 選擇 Create subscription (建立訂閱)。
5. 在建立訂閱，請執行下列動作：

   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 Protocol (通訊協定)，選擇 Email (電子郵件) 或 SMS (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. 選擇電郵以收到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。

如何取消訂閱 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 Subscriptions (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)。

搭 AWS CodeBuild 配 Amazon Virtual Private Cloud 使用

一般而言， AWS CodeBuild 無法存取 VPC 中的資源。若要啟用存取權，您必須在 CodeBuild 專案組態中提供其他 VPC 特定組態資訊。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

來自 AWS CodeBuild 組建的 VPC 連線能力讓您能夠：

- 針對隔離在私有子網路上的 Amazon RDS 資料庫中的資料，從您的組建執行整合測試。
- 直接從測試中查詢 Amazon ElastiCache 叢集中的資料。
- 與 Amazon EC2、Amazon ECS 或使用內部 Elastic Load Balancing 的服務上託管的內部網路服務互動。
- 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。
- 存取設定為僅允許透過 Amazon VPC 端點存取的 S3 儲存貯體中的物件。
- 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。

允許在您 CodeBuild的專案中存取 Amazon VPC

在您的 VPC 組態中包含這些設定：

- 對於 VPC ID，請選擇使 CodeBuild 用的 VPC 識別碼。
- 對於子網路，請選擇具有 NAT 轉譯的私有子網路，該子網路包含或具有通往所使用資源的路由 CodeBuild。
- 對於安全群組，請選擇 CodeBuild 用來允許存取 VPC 中資源的安全性群組。

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)。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 and CodeBuild，AWS CLI 用於代表 IAM 使 CodeBuild 用者與服務互動的服務角色必須附加政策。如需相關資訊，請參閱允許 CodeBuild 存取建立虛擬私人 VPC 網路介面所需的 AWS 服務。

vpcConfig ###### vpcId #####securityGroupIds

- *vpcId*：必要的。 CodeBuild 使用的 VPC 識別碼。執行此命令以取得您所在區域中所有 Amazon VPC ID 的清單：

```
aws ec2 describe-vpcs
```

- *subnets*：必要。包含所使用資源的子網路 ID CodeBuild。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。用於允許存取 VPC 中資源的安全性群組 ID。 CodeBuild 執行此命令以去得這些 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。

VPC 最佳實務

設定要使用的 VPC 時，請使用此檢查清單。 CodeBuild

- 使用公用和私有子網路以及 NAT 閘道來設定您的 VPC。NAT 閘道必須位於公用子網路中。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 CodeBuild 與 VPC 搭配使用， CodeBuild 以便連線到公用端點 (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非 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界面，而且 Amazon EC2 不支援在 Amazon EC2 執行個體啟動之外建立的任何網路界面自動指派公用 IP 地址。

- 將多個可用區域加入您的 VPC。
- 請確定您的安全性群組沒有允許組建的入站 (輸入) 流量。 CodeBuild 對輸出流量沒有特定需求，但是您必須允許存取組建所需的任何網際網路資源，例如 GitHub 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。
- 當您將 CodeBuild 專案設定為存取 VPC 時，請選擇僅私有子網路。

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 AWS CloudFormation 來設定 VPC 以使用 CodeBuild VPC 功能的詳細資訊，請參閱。AWS CloudFormation VPC 範本

為您的 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。

以下是在疑難排解常見 CodeBuild VPC 錯誤時協助您的一些準則：`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. 確保代表 IAM 使用者 CodeBuild 與服務互動時所使用的服務角色具有此政策中的許可。如需詳細資訊，請參閱 建立 CodeBuild 服務角色。

   如果 CodeBuild 缺少權限，您可能會收到錯誤訊息，顯示`Unexpected EC2 error: UnauthorizedOperation`。如果 CodeBuild 沒有使用 VPC 所需的 Amazon EC2 許可，就會發生此錯誤。

VPC 的局限性

- 視窗中不支援虛擬私人 VPC 連線。 CodeBuild
- 共用 VPC 不支援來自 CodeBuild 的 VPC 連線。

使用 VPC 端點

您可以將 AWS CodeBuild 設定成使用界面 VPC 端點，以提升建置的安全性。介面端點由 PrivateLink，您可以用來私有存取 Amazon EC2 的技術，以及CodeBuild通過使用私有 IP 地址。PrivateLink限制代管執行個體之間的所有網路流量，CodeBuild和亞馬遜 EC2 到亞馬遜網絡。(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。有關更多信息PrivateLink和 VPC 端點，請參閱什麼是AWS PrivateLink?。

建立 VPC 端點之前

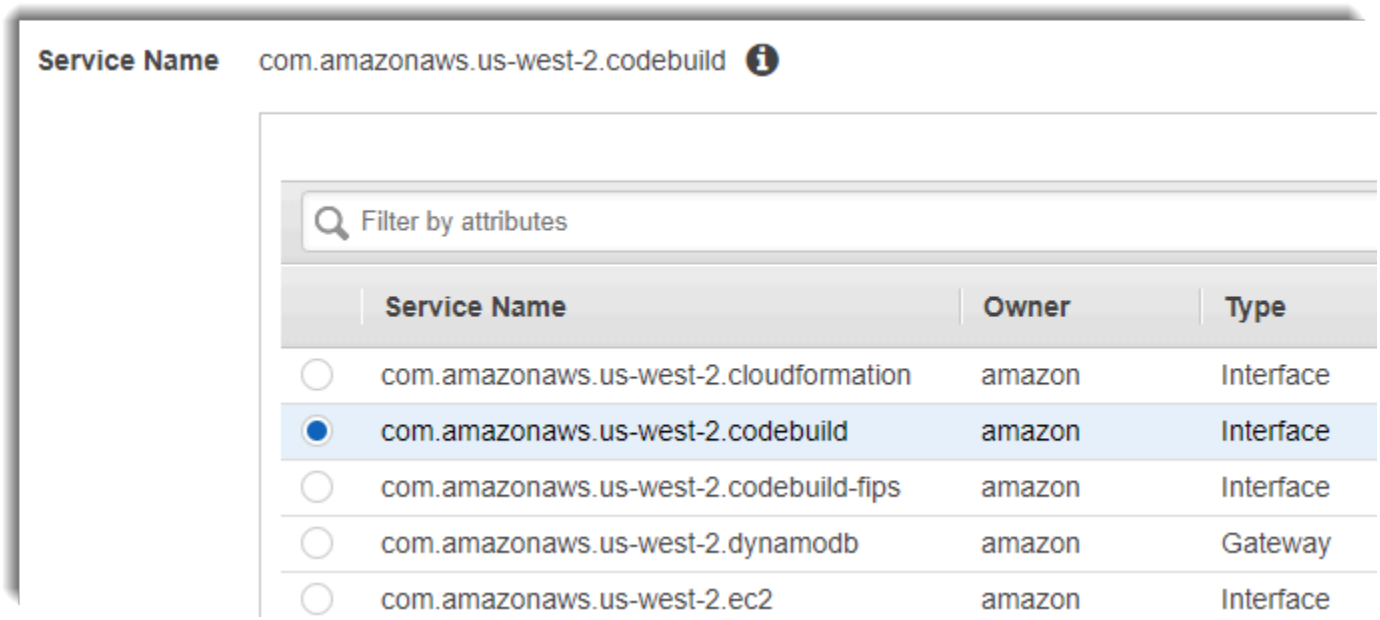
在您設定 AWS CodeBuild 的 VPC 端點之前，請注意以下的約束與限制。

> **Note**
>
> 使用一個NAT 閘道如果你想使用CodeBuild與AWS不支援亞馬遜 VPC 的服務PrivateLink連接。

- VPC 端點僅支援透過亞馬遜路線 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。確保您在相同的端點中創建了端點AWS區域為任何存放組建輸入和輸出的 S3 儲存貯體。您可以使用亞馬遜 S3 主控台或get-bucket-location命令以查找存儲桶的位置。使用區域特定的 Amazon S3 端點存取儲存貯體 (例如，`<bucket-name>.s3-us-west-2.amazonaws.com`). 如需 Amazon S3 特定區域端點的詳細資訊，請參閱亞馬遜簡單存儲服務在Amazon Web Services 一般參考。如果您使用AWS CLI若要向 Amazon S3 發出請求，請將您的預設區域設定為建立儲存貯體的相同區域，或使用`--region`您的請求中的參數。

建立適用於 CodeBuild 的 VPC 端點

按照建立界面端點中的指示建立 `com.amazonaws.region.codebuild` 端點。這是用於 AWS CodeBuild 的 VPC 端點。



region 代表 CodeBuild 支援之 AWS 區域的區域識別符，例如 `us-east-2` 代表美國東部 (俄亥俄) 區域。如需支援的清單AWS區域，請參閱CodeBuild在 AWS一般參考。端點會預先填入您登入 AWS 時所指定的區域。如果變更您的區域，VPC 端點會隨之更新。

建立 CodeBuild 的 VPC 端點政策

您可以為下列項目建立 Amazon VPC 端點的政策AWS CodeBuild您可以在其中指定：

- 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 CloudFormation VPC 範本

AWS CloudFormation可讓您建立和佈建AWS可預期和重複的基礎設施部署，使用範本檔案，將資源集合視為一個單位（堆。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》。

以下是用來將 VPC 設定為使用 AWS CodeBuild 的 AWS CloudFormation YAML 範本。這個檔案也可在中使用samples.zip。

```
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
```

使用 AWS CodeBuild 搭配代理伺服器

您可以使用 AWS CodeBuild 搭配代理伺服器來調節往返網際網路的 HTTP 和 HTTPS 流量。若要搭配代理伺服器執行 CodeBuild，請將代理伺服器安裝在公有子網路，在 VPC 中的私有子網路。

在代理伺服器中執行 CodeBuild 有兩個主要的使用案例：

- 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。

您可以使用 CodeBuild 搭配兩種類型的代理伺服器。兩者都一樣，代理伺服器在公有子網路中執行，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，您必須將NO_PROXY、HTTP_PROXY，和HTTPS_PROXY環境變量。如需詳細資訊，請參閱 在 AWS CodeBuild 中變更建置專案的設定 及 在 AWS CodeBuild 中建立建置專案。
- 透明代理：如果您使用透明代理伺服器，不需要特殊配置。

主題

- 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

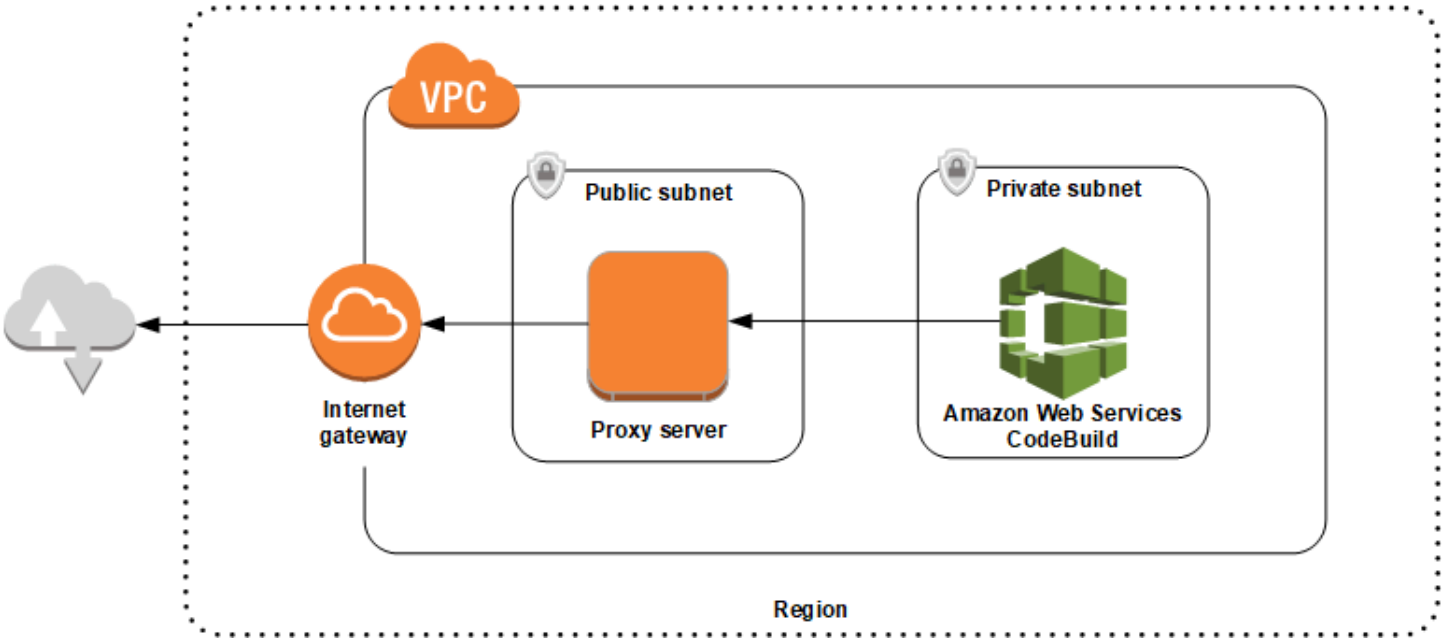
- 在透明代理伺服器中執行 CodeBuild
- 在代理伺服器中執行套用管理員和其他工具

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代理伺服器中執行 AWS CodeBuild：

- VPC。
- 您的 VPC 中一個用於代理伺服器的公有子網路。
- VPC 中一個用於 CodeBuild 的私有子網路。
- 允許 VPC 和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中執行 AWS CodeBuild 需要以下步驟。

1. 建立 VPC。如需相關資訊，請參閱「」。建立 VPC中的Amazon VPC User Guide。
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`在其中運行 CodeBuild。

   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. 請遵循使用啟動執行個體精靈啟動執行個體中的Amazon EC2 使用者指南啟動 Amazon Linux 執行個體。當您執行精靈時，請選擇以下選項：
   - In選擇執行個體類型下，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。如需相關資訊，請參閱「」。停用來源/目標檢查中的Amazon VPC User Guide。
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。需要如此，從私有子網路中的執行個體 (CodeBuild) 傳出的請求才會一律透過代理伺服器來路由傳送。

安裝和設定代理伺服器

有許多代理伺服器可供選擇。此處使用開放來源碼代理伺服器 Squid，以示範 AWS CodeBuild 如何在代理伺服器中執行。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

主題

- 將 Squid 設定為明確代理伺服器
- 創建 CodeBuild 專案
- 明確代理伺服器 squid.conf 檔案範例

若要在明確代理伺服器中執行 AWS CodeBuild，您必須設定代理伺服器，以允許或拒絕往返外部網站的流量，然後設定 HTTP_PROXY 和 HTTPS_PROXY 環境變數。

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。編輯最後一行的規則表達式，以指定 S3 儲存貯體，或在AWS區域。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，以授權存取us-west-1區域。
- 如果您的來源是AWS CodeCommit，使用git-codecommit.<*your-region*>.amazonaws.com新增AWS區域設定為允許清單。

```
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：

1. 在 http_access deny all 陳述式之前，插入以下陳述式。它們允許 CodeBuild 訪問 CloudWatch 和 Amazon S3。需要訪問 CloudWatch，以便 CodeBuild 可以創建 CloudWatch 日誌。需要能夠存取 Amazon S3 才能上傳工件和 Amazon S3 緩存。

   - ```
     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
     ```

   - 保存之後squid.conf，請執行以下命令：
     ```

```
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 RequestError 在代理服務器 CodeBuild 中運行時出現超時錯誤。

如需詳細資訊，請參閱本主題稍後的 明確代理伺服器 `squid.conf` 檔案範例。

創建 CodeBuild 專案

若要搭配明確代理伺服器來執行 AWS CodeBuild，請在專案層級將其 HTTP_PROXY 與 HTTPS_PROXY 環境變數設定為您為代理伺服器所建立之 EC2 執行個體的私有 IP 地址，以及連接埠 3128。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 AWS CodeBuild 中建立建置專案 及 在 AWS CodeBuild 中變更建置專案的設定。

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

若要在透明代理伺服器中執行 AWS CodeBuild，您必須將代理伺服器設定為可存取與之互動的網站和網域。

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。若要搭配透明代理伺服器來執行 AWS CodeBuild，您必須授與它 `amazonaws.com` 的存取權。您還必須授權存取使用 CodeBuild 使用的其他網路。這些都根據您如何建立 CodeBuild 專案而有所不同。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
```

創建 CodeBuild 專案

設定您的代理伺服器後，您就可以在私有子網路中使用它並搭配 AWS CodeBuild，而不需要再設定。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

若要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。將其添加到squid.conf檔案，並在 buildspec 檔案中新增命令，讓 CodeBuild 知道代理伺服器的端點。

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apt-get在構建環境中運行。

   ```
   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 webiste. 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
```

在 AWS CodeBuild 中使用建置專案和建置

若要開始，請遵循建立組建專案，然後遵循執行建置。如需組建專案和組建的詳細資訊，請參閱下列主題。

主題

- 使用組建專案
- 使用 AWS CodeBuild 中的建置

使用組建專案

組建專案包含如何執行組建的相關資訊，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，以及儲存組建輸出的位置。

您可以在處理組建專案時，執行這些任務：

主題

- 在 AWS CodeBuild 中建立建置專案
- 建立通知規則
- 在 AWS CodeBuild 中檢視建置專案名稱清單
- 在 AWS CodeBuild 中檢視建置專案的詳細資訊
- 在 AWS CodeBuild 中建立快取
- 在中構建觸發器 AWS CodeBuild
- GitLab 連接
- 使用網路掛鉤 AWS CodeBuild
- 在 AWS CodeBuild 中變更建置專案的設定
- 在 AWS CodeBuild 中刪除建置專案
- 使用共用專案
- 在 AWS CodeBuild 中標記專案
- Batch 建置 AWS CodeBuild
- GitHub 動作亞軍 AWS CodeBuild

- 中的公共組建專案AWS CodeBuild

在 AWS CodeBuild 中建立建置專案

您可以使用 AWS CodeBuild 主控台、AWS CLI 或 AWS 開發套件，建立建置專案。

先決條件

在創建構建項目之前，請回答規劃組建。

主題

- 建立組建專案 (主控台)
- 建立建置專案 (AWS CLI)
- 建立建置專案 (AWS 開發套件)
- 建立建置專案 (AWS CloudFormation)

建立組建專案 (主控台)

開啟主 AWS CodeBuild 控台，網址為 https://console.aws.amazon.com/codesuite/codebuild/home。

如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，然後選擇 [建立組建專案]。

選擇 Create build project (建立建置專案)。

填寫以下各節。完成後，選擇頁面底部的 [建立建置專案]。

章節：

- 項目配置
- 來源
- 環境
- 建置規格
- Batch 設定
- 成品
- 日誌

項目配置

Project name (專案名稱)

輸入此組建專案的名稱。每個 AWS 帳戶的組建專案名稱必須是唯一的。

Description

輸入建置專案的選用描述，以協助其他使用者瞭解此專案的用途。

建立徽章

(選擇性) 選取 [啟用組建徽章]，讓專案的建置狀態可見且可嵌入。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，則不適用建立徽章。

啟用並行建置限制

(選擇性) 如果您要限制此專案的並行建置數目，請執行下列步驟：

1. 選取「限制此專案可啟動的並行建構數目」。
2. 在並行建構限制中，輸入此專案允許的並行建構數目上限。此限制不得大於針對帳戶設定的並行建置限制。如果您嘗試輸入的數字大於帳號限制，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

(選擇性) 針對標籤，輸入您希望支援 AWS 服務使用之任何標籤的名稱和值。使用 Add row (新增資料列) 來新增標籤。您最多可新增 50 個標籤。

來源

來源提供者

選擇源代碼提供者類型。使用下列清單來選取適合您的來源提供者的選項：

> **Note**
>
> CodeBuild 不支援比特桶伺服器。

Amazon S3

儲存貯體

選擇包含原始程式碼之輸入值區的名稱。

S3 物件金鑰或 S3 資料夾

輸入 ZIP 檔案的名稱或包含原始程式碼之資料夾的路徑。輸入正斜線 (/) 以下載 S3 儲存貯體中的所有項目。

來源版本

輸入代表輸入檔案組建之物件的版本 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

CodeCommit

儲存庫

選擇您要使用的存放庫。

參考類型

選擇「分支」、「Git 標籤」或「提交 ID」來指定原始程式碼的版本。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

選擇此選項可建立歷程記錄截斷為指定數目的簡易複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模塊

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

儲存庫

選擇 [使用 OAuth Connec t] 或 [使用 Bitbucket 應用程式密碼連線]，然後依照指示進行連線 (或重新連線) 至 Bitbucket。

在您的帳戶中選擇一個公共存儲庫或存儲庫。

來源版本

輸入分支、提交 ID、標籤或參照，以及提交 ID。如需更多資訊，請參閱原始碼版本範例 AWS CodeBuild

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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 子模塊

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

針對狀態內容，請在 Bitbucket 認可狀態中輸入要用於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，請在 Bitbucket 認可狀態中輸入要用於url參數的值。如需詳細資訊，請參閱 Bitbucket API 文件中的建置。

由 webhook 觸發的構建狀態始終報告給源提供程序。若要讓從主控台啟動的組建狀態，或向來源提供者報告 API 呼叫，您必須選取此設定。

如果您的項目的構建是由 webhook 觸發的，則必須將新的提交推送到存儲庫，以使對此設置的更改生效。

如果您想要在每次將程式碼變更推送至此儲存庫時建立原始程式碼，請在 [主 CodeBuild 要來源 webhook 事件] 中選取 [每次程式碼變更推送至此儲存庫時重建]。如需有關 Webhook 和篩選器群組的詳細資訊，請參閱比特桶網絡鉤事件。

GitHub

儲存庫

選擇 [使用 OAuth Connect] 或 [使用個 GitHub 人存取權杖連線]，然後依照指示進行連線 (或重新連線) GitHub 並授權存取 AWS CodeBuild權。

在您的帳戶中選擇一個公共存儲庫或存儲庫。

來源版本

輸入分支、提交 ID、標籤或參照，以及提交 ID。如需更多資訊，請參閱原始碼版本範例 AWS CodeBuild

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如811dd1ba1aba14473856cee38308caed7190c0d或5392f7. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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 子模塊

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。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 確認狀態中`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提交狀態。

在「目標 URL」中，輸入 GitHub 確認狀態中要用於`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提交狀態。

由 webhook 觸發的構建狀態始終報告給源提供程序。若要讓從主控台啟動的組建狀態，或向來源提供者報告 API 呼叫，您必須選取此設定。

如果您的項目的構建是由 webhook 觸發的，則必須將新的提交推送到存儲庫，以使對此設置的更改生效。

如果您想要在每次將程式碼變更推送至此儲存庫時建立原始程式碼，請在 [主 CodeBuild 要來源 webhook 事件] 中選取 [每次程式碼變更推送至此儲存庫時重建]。如需有關 Webhook 和篩選器群組的詳細資訊，請參閱GitHub 網絡掛鉤事件。

GitHub Enterprise Server

GitHub 企業個人存取權杖

如需有GitHub 企業伺服器範例關如何將個人存取權杖複製到剪貼簿的資訊，請參閱。將字符貼入文字欄位，接著選擇 Save Token (儲存字符)。

> **Note**
>
> 您只需要輸入及儲存個人存取字符一次。 CodeBuild 在 future 的所有項目中使用此令牌。

來源版本

輸入提取請求、分支、提交 ID、標籤或參照，以及提交 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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 子模塊

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。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 確認狀態中`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提交狀態。

在「目標 URL」中，輸入 GitHub 確認狀態中要用於`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提交狀態。

由 webhook 觸發的構建狀態始終報告給源提供程序。若要讓從主控台啟動的組建狀態，或向來源提供者報告 API 呼叫，您必須選取此設定。

如果您的項目的構建是由 webhook 觸發的，則必須將新的提交推送到存儲庫，以使對此設置的更改生效。

不安全的 SSL

選取「啟用不安全的 SSL」可在連線至 GitHub 企業專案存放庫時忽略 SSL 警告。

如果您想要在每次將程式碼變更推送至此儲存庫時建立原始程式碼，請在 [主 CodeBuild 要來源 webhook 事件] 中選取 [每次程式碼變更推送至此儲存庫時重建]。如需有關 Webhook 和篩選器群組的詳細資訊，請參閱GitHub 網絡掛鉤事件。

GitLab

Connection (連線)

使用 Connect 連接您的 GitLab 帳戶 AWS CodeConnections，並使用連接將第三方存儲庫關聯為構建項目的源。

選擇預設連線或自訂連線。

預設連線會在所有專案中套用預設 GitLab 連線。自訂連線會套用覆寫帳戶預設設定的自訂 GitLab 連線。

預設連線

與您的帳戶相關聯的預設連線名稱。

如果您尚未建立與提供者的連線，請參閱以建立連線至 GitLab (主控台)取得指示。

自訂連線

選擇您要使用的自訂連線名稱。

如果您尚未建立與提供者的連線，請參閱以建立連線至 GitLab (主控台)取得指示。

儲存庫

選擇您要使用的存放庫。

來源版本

輸入提取請求 ID、分支、提交 ID、標籤或參照，以及提交 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

GitLab Self Managed

Connection (連線)

使用 Connect 連接您的 GitLab 帳戶 AWS CodeConnections，並使用連接將第三方存儲庫關聯為構建項目的源。

選擇預設連線或自訂連線。

預設連線會在所有專案中套用預設的「 GitLab 自我管理」連線。自訂連線會套用 GitLab 自訂「自我管理」連線，該連線會覆寫您帳戶的預設設定。

預設連線

與您的帳戶相關聯的預設連線名稱。

如果您尚未建立與提供者的連線，請參閱開發人員工具主控台使用者指南中的建立與 GitLab 自我管理的連線以取得指示。

自訂連線

選擇您要使用的自訂連線名稱。

如果您尚未建立與提供者的連線，請參閱開發人員工具主控台使用者指南中的建立與 GitLab 自我管理的連線以取得指示。

儲存庫

選擇您要使用的存放庫。

來源版本

輸入提取請求 ID、分支、提交 ID、標籤或參照，以及提交 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

環境

佈建模式

執行以下任意一項：

- 若要使用由管理的隨選叢集 AWS CodeBuild，請選擇 [隨選]。透過隨選叢集，為您的組建 CodeBuild 提供運算。當構建完成時，機器將被銷毀。隨需叢集是全受管的，並包含自動擴充功能，可處理尖峰的需求。

- 若要使用由管理的保留容量叢集 AWS CodeBuild，請選擇保留容量，然後選取叢集名稱。使用預留容量叢集，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，可立即處理構建或測試，並減少構建持續時間。使用保留容量叢集時，您的機器會一直在執行，而且只要佈建機器就會繼續產生成本。

如需相關資訊，請參閱使用保留容量 AWS CodeBuild。

環境影像

執行以下任意一項：

- 若要使用由管理的 Docker 映像檔 AWS CodeBuild，請選擇 [受管理的映像檔]，然後從 [作業系統]、[執行階段]、[映像] 和 [映像檔版本] 中進行選取。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)。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。如果您選擇 [其他登錄]，對於 [外部登錄 URL]，請使用格式在 Docker Hub 中輸入 Docker 映像的名稱和標記。*docker repository*/*docker image name*如果您選擇 Amazon ECR，請使用 Amazon ECR 存儲庫和 Amazon ECR 映像在您的帳戶中選擇碼頭映像。 AWS
- 若要使用私人 Docker 映像檔，請選擇 [自訂映像檔]。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。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。認證必須由 Secrets Manager 建立。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

> **Note**
>
> CodeBuild 會覆寫自`ENTRYPOINT`訂泊塢視窗影像的。

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 運算為您的組建提供最佳化的啟動速度。由於啟動延遲較低，Lambda 支援更快的建置速度。Lambda 也會自動擴展，因此組建不會在佇列中等待執行。如需相關資訊，請參閱在中使用 AWS Lambda 計算 AWS CodeBuild。

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 [新增服務角色]。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**
>
> 使用主控台建立組建專案時，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 8 小時之間的值，如果建置未完成，則 CodeBuild 會停止建置。如果 hours (小時) 和 minutes (分鐘) 空白，則會使用預設值 60 分鐘。

特權

(選擇性) 如果您想要建置 Docker 映像檔，或是希望組建只有在您打算使用此建置專案建置 Docker 映像時才能取得提升的權限，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。執行這項操作的一種方式是執行下列建置命令，以在建置規格的 `install` 階段中初始化 Docker 協助程式。如果您選擇由 CodeBuild Docker 支援提供的建置環境映像檔，請勿執行這些命令。

> **Note**
>
> 依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 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

如果您想 CodeBuild 要使用 VPC：

- 對於 VPC，請選擇使 CodeBuild 用的 VPC 識別碼。
- 對於 VPC 子網路，請選擇包含使用之資源的子網路。 CodeBuild
- 對於 VPC 安全性群組，請選擇 CodeBuild 用來允許存取 VPC 中資源的安全性群組。

如需詳細資訊，請參閱 搭 AWS CodeBuild 配 Amazon Virtual Private Cloud 使用。

運算

選擇其中一個可用選項。

環境變數

輸入名稱和值，然後選擇要使用之組建之每個環境變數的類型。

> **Note**
>
> CodeBuild 自動設定「 AWS 區域」的環境變數。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

控制台和 AWS CLI 用戶可以看到環境變量。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)。

我們建議您將具有敏感值的環境變數 (例如 AWS 存取金鑰 ID、 AWS 秘密存取金鑰或密碼) 存放在 Amazon EC2 Systems Manager 參數存放區或 AWS Secrets Manager。

如果您使用 Amazon EC2 Systems Manager 參數存放區，請選擇「參數」做為「類型」。在「名稱」中，輸入 CodeBuild 要參考的識別碼。在值中，輸入存放在 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)。針對名稱，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，您可以指定帳戶中金 AWS KMS 鑰的 ARN。 Amazon EC2 Systems Manager 使用此金鑰在儲存期間加密參數的值，並在擷取期間將其解密。) 如果您使用 CodeBuild 主控台建立參數，則主控台會在儲存參數名稱時以其開始。/CodeBuild/如需詳細資訊，請參閱 Amazon EC2 Systems Manager 使用指南中的 Systems Manager 參數存放區和系統管理員參數存放主控台逐步解說。

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，則建置專案的服務角色必須允許該ssm:GetParameters動作。如果您先前選擇 [新增服務角色]，請將此動作 CodeBuild 包含在組建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照存放在 Amazon EC2 Systems Manager 參數存放區中，參數名稱不是開頭為的參數/CodeBuild/，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以的參數名稱/CodeBuild/。這是因為該服務角色僅允許存取開頭為 /CodeBuild/ 的參數名稱。

如果選擇 [新增服務角色]，服務角色會包含解密 Amazon EC2 Systems Manager 參數存放區中/CodeBuild/命名空間下所有參數的權限。

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，請選擇 Secrets Manager 做為類型。在「名稱」中，輸入 CodeBuild 要參考的識別碼。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」，建議您以 `/CodeBuild/` (例如`/CodeBuild/dockerLoginPassword`) 開頭的名稱來儲存密碼。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。
> 如果您的組建專案參照儲存在 Secrets Manager 中的密碼，則組建專案的服務角色必須允許該`secretsmanager:GetSecretValue`動作。如果您先前選擇 [新增服務角色]，請將此動作 CodeBuild 包含在組建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。
> 如果您的組建專案參照儲存在 Secrets Manager 中的密碼`/CodeBuild/`，而且您選擇了新服務角色，則必須更新服務角色，以允許存取開頭不是以的密碼名稱`/CodeBuild/`。這是因為服務角色只允許存取以開頭的密碼名稱`/CodeBuild/`。
> 如果您選擇 [新增服務角色]，服務角色會包含解密 Secrets Manager 中`/CodeBuild/`命名空間下所有密碼的權限。

建置規格

建置規格

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)。依預設，會 CodeBuild尋找在原始程式碼根目錄`buildspec.yml`中名為的檔案。如果您的 buildspec 檔案使用不同的名稱或位置，請在 Buildspec 名稱中從來源根目錄輸入路徑 (例如，或. `buildspec-two.yml` `configuration/buildspec.yml` 如果 buildspec 檔案位於 S3 儲存貯體中，則該檔案必須與您的建置專案位於相同的 AWS 區域。 使用其 ARN 指定構建規格文件（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，或者您有一長串的`build`階段指令，請將`buildspec.yml`檔案新增至原始程式碼根目錄，將指令新增至檔案，然後在原始程式碼根目錄中選擇 [使用 buildspec.yml]。

如需更多資訊，請參閱Buildspec 參考。

Batch 設定

您可以將組建群組當做單一作業執行。如需詳細資訊，請參閱 Batch 建置 AWS CodeBuild。

定義批次組態

選取此選項可允許批次建置在此專案中。

Batch 服務角色

提供批次組建的服務角色。

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。

Batch 組建會在批次設定中引入新的資訊安全角色。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，才能在批次中執行組建。客戶應該使用新角色，而不是在組建中使用的角色，原因有兩個：

- 賦予構建角色 `StartBuild`、`StopBuild`，和`RetryBuild`權限將允許單個構建通過 buildspec 啟動更多構建。
- CodeBuild 批次組建會提供限制，限制可用於批次中組建的組建數量和運算類型。如果組建角色具有這些權限，則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。選取所有適用的項目。

批次允許的最大組建

輸入批次中允許的最大建構數目。如果批次超過此限制，批次就會失敗。

Batch 逾時

輸入批次建置完成的時間上限。

合併成品

選取「將批次中的所有人工因素合併至單一位置」，以將批次中的所有人工因素合併至單一位置。

Batch 報告模式

為批次組建選取所需的建置狀態報告模式。

> **Note**
>
> 只有當專案來源為 Bitbucket 或 GitHub企業 GitHub，並且在 [來源] 下選取 [組建開始和結束] 時，才能使用此欄位。

彙總組建

選取此選項可將批次中所有建構的狀態合併為單一狀態報表。

個別組建

選取此選項可分別報告批次中所有組建的建構狀態。

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，或者想要將 Docker 映像推送到 Amazon ECR 存放庫，則可能需要執行此操作。
- 若要將組建輸出存放在 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 AWS 受管金鑰 用 Amazon S3 加密建置輸出成品，請將加密金鑰保留空白。此為預設值。
- 若要使用客戶受管金鑰來加密組建輸出成品，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。
  - (選擇性) 對於快取路徑前置詞，請輸入 Amazon S3 路徑前置詞。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「在 AWS CodeBuild 中建立快取」。

日誌

選擇您要建立的記錄檔。您可以建立 Amazon CloudWatch 日誌、Amazon S3 日誌，或兩者兼而有之。

CloudWatch

如果你想要 Amazon CloudWatch 日誌日誌：

CloudWatch 日誌

選取CloudWatch 記錄檔。

Group name (群組名稱)

輸入您的 Amazon CloudWatch 日誌日誌群組的名稱。

串流名稱

輸入您的 Amazon CloudWatch 日誌日誌流名稱。

S3

如果你想要 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。

路徑前綴

輸入記錄檔的前置字元。

停用 S3 日誌加密

如果您不想要加密 S3 日誌，請選取此選項。

建立建置專案 (AWS CLI)

如需有關使用 AWS CLI 與的詳細資訊 CodeBuild，請參閱命令列參考。

若要使用 CodeBuild 建立組建專案 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指令來建立專案。

建立 JSON 檔案

使用以下選項，使用以下create-project命令創建骨架 JSON 文--generate-cli-skeleton件：

```
aws codebuild create-project --generate-cli-skeleton > <json-file>
```

這將創建一個 JSON 文件，其中包含指定的路徑和文件名<json-file>。

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>",
  "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>"
      ]
    },
    "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。包含此組建專案原始程式碼設定相關資訊的ProjectSource物件。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 無來源的專案範例。

來源/位置

除非您設定<source-type>為，否則為必要CODEPIPELINE。所指定儲存庫類型的來源碼位置。

- 對於 CodeCommit，HTTPS 克隆 URL 到包含源代碼和構建規格文件的存儲庫（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，後面接著包含原始程式碼和組建規格的 ZIP 檔案的路徑和名稱。例如：
  - 如果是位於輸入值區根目錄的 ZIP 檔案，請執行下列動作：*<bucket-name>*/*<object-name>*.zip
  - 對於位於輸入值區中子資料夾中的 ZIP 檔案：*<bucket-name>*/*<subfoler-path>*/*<object-name>*.zip。

- 對於 GitHub，HTTPS 克隆 URL 到包含源代碼和構建規格文件的存儲庫。URL 必須包含 github.com。您必須將您的 AWS 帳戶連接到您的 GitHub 帳戶。若要這麼做，請使用 CodeBuild 主控台建立建置專案。
  - 選擇 Authorize application (授權應用程式)。(在您連線到您的 GitHub 帳戶之後，您不需要完成建置專案的建立作業。 您可以關閉主 CodeBuild 控台。)
- 對於 GitHub 企業服務器，HTTP 或 HTTPS 克隆 URL 到包含源代碼和構建規格文件的存儲庫。您也必須將您的 AWS 帳戶連線到您的 GitHub企業伺服器帳戶。若要這麼做，請使用 CodeBuild 主控台建立建置專案。
  1. 在 GitHub 企業伺服器中建立個人存取權杖。
  2. 將此令牌複製到剪貼板，以便在創建 CodeBuild 項目時可以使用它。如需詳細資訊，請參閱 GitHub 說明網站上的命令列建立個人存取權杖。
  3. 當您使用主控台建立 CodeBuild 專案時，請在 [來源] 中，針對 [來源提供者] 選擇 [GitHub企業]。
  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 帳戶現在已連接到您的 GitHub 企業伺服器帳戶。
- 對於 GitLab 和 GitLab 自我管理，HTTPS 克隆 URL 到包含源代碼和 buildspec 文件的存儲庫。請注意，如果您使用 GitLab，網址必須包含 gitlab.com。如果您使用 GitLab 自我管理，則該網址不需要包含 gitlab.com。您必須將您的 AWS 帳戶連接到您的帳戶 GitLab或 GitLab 自我管理帳戶。若要這麼做，請使用 CodeBuild主控台建立建置專案。
  - 在開發人員工具導覽窗格中，選擇 [設定]、[連線]，然後選取 [建立連線] 在此頁面上，建立 GitLab 或 GitLab 自我管理的連線，然後選擇 [Connect 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。您還必須將您的 AWS 帳戶連接到您的 Bitbucket 帳戶。若要這麼做，請使用 CodeBuild 主控台建立建置專案。
  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)。 (在您連接到您的 Bitbucket 帳戶之後，您不需要完成構建項目的創建。 您可以關閉主 CodeBuild 控台。)
- 對於 AWS CodePipeline，請勿指定的`location`值`source`。 CodePipeline 會忽略此值，因為當您在中建立管線時 CodePipeline，會在管線的「來源」階段中指定原始程式碼位置。

來源/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。

來源/構建規格

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，它可以是內嵌 buildspec 定義、相對於主要來源根目錄的替代 buildspec 檔案路徑，或 S3 儲存貯體的路徑。值區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/驗證

請勿使用。此物件僅供 CodeBuild 主控台使用。

來源/reportBuildStatus

指定是否將建置的啟動和完成狀態傳送給來源提供者。如果您使用 GitHub 企業伺服器或 Bitbucket 以外 GitHub的來源提供者進行設定，則會擲回一`invalidInputException`個。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

來源/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。只有當來源類型為、或時`GITHUB`，`GITHUB_ENTERPRISE`才會使用此選項`BITBUCKET`。

來源//buildStatusConfig上下文

對於 Bitbucket 來源，此參數會用於 Bitbucket 認可狀態中的`name`參數。對於 GitHub 來源，此參數會用於提 GitHub 交狀態中的`context`參數。

例如，您可以使用 CodeBuild環境變量來`context`包含內部版本號和 webhook 觸發器：

```
AWS CodeBuild sample-project Build #$CODEBUILD_BUILD_NUMBER -
 $CODEBUILD_WEBHOOK_TRIGGER
```

這會導致由 webhook 提取請求事件觸發的構建 #24 的上下文如下所示：

```
AWS CodeBuild sample-project Build #24 - pr/8
```

來源//buildStatusConfig目標網站

對於 Bitbucket 來源，此參數會用於 Bitbucket 認可狀態中的`url`參數。對於 GitHub 來源，此參數會用於提 GitHub 交狀態中的`target_url`參數。

例如，您可以將設定targetUrl為https://aws.amazon.com/codebuild/*<path to build>*，提交狀態將連結至此 URL。

您也可以在中包含 CodeBuild 環境變數，targetUrl以將其他資訊新增至 URL。例如，要將構建區域添加到 URL，請將設置targetUrl為：

```
"targetUrl": "https://aws.amazon.com/codebuild/<path to build>?region=
$AWS_REGION"
```

如果構建區域是us-east-2，這將擴展到：

```
https://aws.amazon.com/codebuild/<path to build>?region=us-east-2
```

來源/gitSubmodulesConfig

選用。Git 子模組組態的相關資訊。僅與 CodeCommit、 GitHub、 GitHub 企業伺服器和 Bitbucket 搭配使用。

來源//獲gitSubmodulesConfig取子模塊

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。僅適用於 GitHub 企業伺服器。將此值設定為可在連線true至您的 GitHub 企業伺服器專案存放庫時忽略 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。

來源/來源識別碼

專案來源的使用者定義識別碼。主要來源的選用項目。次要來源需要。

secondarySources

選用。ProjectSource物件陣列，其中包含組建專案之次要來源的相關資訊。您最多可以新增 12 個次要來源。secondarySources物件使用物件所使用的相同性質。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。要為此專案建置的組建輸入版本。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，要使用的提交 ID、分支或 Git 標籤。
- 針 GitHub對您要建置的原始程式碼版本對應的提交 ID、提取要求 ID、分支名稱或標籤名稱。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，提交 ID，提取請求 ID，分支名稱，標籤名稱或引用和提交 ID。如需詳細資訊，請參閱 原始碼版本範例 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 原始碼版本範例 AWS CodeBuild。

文物

必要。包含此組建專案輸出成品設定相關資訊的ProjectArtifacts物件。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：

人工/類型

必要。建置輸出成品類型。有效的 值如下：

- `CODEPIPELINE`
- `NO_ARTIFACTS`
- `S3`

人工/位置

僅與S3人工因素類型搭配使用。不適用於其他人工因素類型。

您在先決條件中建立或識別的輸出值區的名稱。

人工/路徑

僅與S3人工因素類型搭配使用。不適用於其他人工因素類型。

要放置 ZIP 檔案或資料夾的輸出值區中的路徑。如果您未指定的值path， CodeBuild 會使用namespaceType (若有指定)，並決name定組建輸出 ZIP 檔案或資料夾的路徑和名稱。例如，如果您指定MyPathMyArtifact.zip為path和name，路徑和名稱將為MyPath/MyArtifact.zip。

人工/命名空間類型

僅與S3人工因素類型搭配使用。不適用於其他人工因素類型。

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 (若有指定)，並決name定組建輸出 ZIP 檔案或資料夾的路徑和名稱。例如，如果您指定MyPath為pathnamespaceType、BUILD_ID用MyArtifact.zip於和 ofname，則路徑和名稱將為MyPath/*build-ID*/MyArtifact.zip。

artifacts/name

僅與S3人工因素類型搭配使用。不適用於其他人工因素類型。

構建輸出 ZIP 文件或其中文件夾的名稱location。例如，如果您指定MyPathMyArtifact.zip為path和name，路徑和名稱將為MyPath/MyArtifact.zip。

人工藝品/overrideArtifactName

僅與 S3 成品類型搭配使用。不適用於其他人工因素類型。

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 建立的規格參考 CodeBuild。

工藝/包裝

僅與S3人工因素類型搭配使用。不適用於其他人工因素類型。

選用。指定如何封裝成品。允許的值為：

NONE

建立包含組建加工品的資料夾。這是預設值。

ZIP

建立包含組建加工品的 ZIP 檔案。

secondaryArtifacts

選用。ProjectArtifacts物件陣列，其中包含組建專案之次要加工品設定的相關資訊。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。包含有關此構建項目緩存設置信息的ProjectCache對象。如需詳細資訊，請參閱 建立快取。

環境

必要。包含此專案之建置環境設定相關資訊的ProjectEnvironment物件。這些設定包括：

環境/類型

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。

環境/影像

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 映像檔的 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,*account-id*.dkr.ecr.*region-id*.amazonaws.com/*your-Amazon-ECR-repo-name*:*tag*. 如需詳細資訊，請參閱 碼頭圖片提供 CodeBuild。

環境/computeType

必要。指定此建置環境所使用的計算資源。如需詳細資訊，請參閱 CodeBuild API 參考資料中的 computeType。

環境/證書

選用。Amazon S3 儲存貯體的 ARN、路徑前置詞以及包含 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 儲存貯體名稱是*<my-bucket>*，您的路徑前綴為*<cert>*，而您的物件金鑰名稱為*<certificate.pem>*certificate，則可接受的 is *<my-bucket/cert/certificate.pem>* 或格式arn:aws:s3:::*<my-bucket/cert/certificate.pem>*。

環境/環境變量

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、和name、value、和type的name物件來表示type。value

控制台和 AWS CLI 用戶可以看到所有環境變量。如果您對環境變數的可見性沒有任何疑慮value，請將和設定type為PLAINTEXT。name

我們建議您將具有敏感值的環境變數 (例如 AWS 存取金鑰 ID、 AWS 秘密存取金鑰或密碼) 存放為 Amazon EC2 Systems Manager 參數存放區中的參數或 AWS Secrets Manager。對於name，針對該儲存的參數，設定 CodeBuild 要參照的識別元。

如果您使用 Amazon EC2 Systems Manager 參數存放區，則將參數名稱設定為存放在參數存放區中。value將 type 設定為 PARAMETER_STORE。使用名/CodeBuild/dockerLoginPassword為範例的參數，name將設定為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，您可以指定帳戶中金 AWS KMS 鑰的 ARN。 Amazon EC2 Systems Manager 使用此金鑰在儲存期間加密參數的值，並在擷取期間將其解密。) 如果您使用 CodeBuild 主控台建立參數，則主控台會在儲存參數名稱時以其開始。/CodeBuild/如需詳細資訊，請參閱 Amazon EC2 Systems Manager 使用指南中的 Systems Manager 參數存放區和系統管理員參數存放主控台逐步解說。
>
> 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，則建置專案的服務角色必須允許該ssm:GetParameters動作。如果您先前選擇 [新增服務角色]，請將此動作 CodeBuild 包含在組建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照存放在 Amazon EC2 Systems Manager 參數存放區中，參數名稱不是開頭為的參數/CodeBuild/，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以的參數名稱/CodeBuild/。這是因為該服務角色僅允許存取開頭為 /CodeBuild/ 的參數名稱。
>
> 如果選擇 [新增服務角色]，服務角色會包含解密 Amazon EC2 Systems Manager 參數存放區中/CodeBuild/命名空間下所有參數的權限。
>
> 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 Managervalue，則設定參數名稱儲存在 Secrets Manager 中。將 type 設定為 SECRETS_MANAGER。使用名/CodeBuild/dockerLoginPassword為的密碼作為範例，name將設定為LOGIN_PASSWORD。將 value 設定為 /CodeBuild/dockerLoginPassword。將 type 設定為 SECRETS_MANAGER。

Important

如果您使用「Secrets Manager」，建議您以 /CodeBuild/ (例如/CodeBuild/dockerLoginPassword) 開頭的名稱來儲存密碼。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。
如果您的組建專案參考儲存在 Secrets Manager 中的密碼，則組建專案的服務角色必須允許該secretsmanager:GetSecretValue動作。如果您先前選擇 [新增服務角色]，請將此動作 CodeBuild 包含在組建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。
如果您的組建專案參照儲存在 Secrets Manager 中的密碼/CodeBuild/，而且您選擇了新服務角色，則必須更新服務角色，以允許存取開頭不是以的密碼名稱/CodeBuild/。這是因為服務角色只允許存取以開頭的密碼名稱/CodeBuild/。
如果您選擇 [新增服務角色]，服務角色會包含解密 Secrets Manager 中/CodeBuild/命名空間下所有密碼的權限。

環境/註冊證書

選用。指RegistryCredential定提供私人 Docker 登錄存取權之認證的物件。

環境/註冊證書/認證

指定 ARN 或使用 AWS Managed Services建立的認證名稱。您只能在登入資料存在於您目前區域時才能使用其名稱

環境/註冊證書/憑證提供者

唯一有效的值為 SECRETS_MANAGER。

當此值設為：

- `imagePullCredentials` 必須設定為 `SERVICE_ROLE`。
- 該映像不能是精選圖像或 Amazon ECR 映像。

環境/類型 imagePullCredentials

選用。認證類型 CodeBuild 用來提取組建中的映像檔。兩種有效值如下：

代碼生成

`CODEBUILD`指定使 CodeBuild 用自己的認證。您必須編輯 Amazon ECR 儲存庫政策以信任 CodeBuild 服務主體。

服務角色

指定使 CodeBuild 用組建專案的服務角色。

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。使用 CodeBuild 策劃映像時，必須使用`CODEBUILD`認證。

環境/privilegedMode

`true`只有當您打算使用此構建項目構建 Docker 映像時才設置為。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。如果您指定了由 Docker 支援提供的建置環境映像檔，請勿執 CodeBuild 行這些命令。

> **Note**
>
> 依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 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。服務角色的 ARN CodeBuild 用來代表使用者與服務互動 (例如，`arn:aws:iam::`*account-id*`:role/`*role-name*)。

timeoutInMinutes

選用。5 到 480（8 小時）之間的分鐘數，如果構建未完成，則 CodeBuild停止構建。如果未指定，則會使用預設值 60。若要判斷是否因逾時而 CodeBuild 停止組建，以及何時停止組建，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。

queuedTimeoutIn分鐘

選用。5 到 480 (8 小時) 之間的分鐘數，如果建置仍處於佇列狀態，則 CodeBuild會停止建置。如果未指定，則會使用預設值 60。

encryptionKey

選用。用 CodeBuild 來加密組建輸出 AWS KMS key 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，則會使用適用於 Amazon S3 的 AWS受管 KMS 金鑰。

標籤

選用。Tag 物件陣列，提供您想要與此建置專案相關聯的標籤。您可以指定最多 50 個標籤。任何支援 CodeBuild 建置專案標籤的 AWS 服務都可以使用這些標記。每個標籤都以帶有key和的物件表示value。

vpcConfig

選用。包VpcConfig含專案 VPC 組態相關資訊的物件。如需詳細資訊，請參閱 搭 AWS CodeBuild 配 Amazon Virtual Private Cloud 使用。

這些屬性包括：

vpcId

必要。 CodeBuild 使用的 VPC 識別碼。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 ID 陣列，其中包含使用的資源 CodeBuild。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。用於允許存取 VPC 中資源的安全群組 ID 陣列。 CodeBuild 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。設定為以啟`true`用組建徽章，或以`false`其他方式啟用。如需詳細資訊，請參閱 建立徽章範例 CodeBuild。

日誌配置

包含此組建記錄檔所在位置之相關資訊的LogsConfig物件。

日誌配置/cloudWatchLogs

包含將記錄檔推送至 CloudWatch 記錄檔的相關資訊的CloudWatchLogsConfig物件。

日誌配置/S3 日誌

包含將日誌推送到 Amazon LogsConfigS3 的相關資訊的 S3 物件。

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`物件是包含專案之批次建置組態資訊的ProjectBuildBatchConfig結構。

buildBatchConfig/serviceRo le

批次建置專案的服務角色 ARN。

buildBatchConfig/組合事實

Boolean 值，指定是否要將批次組建的建置加工品合併至單一人工因素位置。

buildBatchConfig/限制/ maximumBuildsAllowed

允許的最大組建數。

buildBatchConfig/限制/ computeTypesAllowed

字串陣列，指定批次建置允許的運算類型。有關這些值，請參閱構建環境計算類型。

buildBatchConfig/timeoutInMinutes

批次建置必須在中完成的時間上限 (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

要創建項目，請再次運行**create-project**命令，傳遞您的 JSON 文件：

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

如果成功，Pro ject 物件的 JSON 表示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儲存在儲存 GitHub 庫中，而且您想 CodeBuild 要在每次將程式碼變更推送至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 開發套件)

如需搭配使用 AWS CodeBuild 與 AWS 開發套件的資訊，請參閱 AWS 開發套件和工具參考。

建立建置專案 (AWS CloudFormation)

關於使用AWS CodeBuild取代為AWS CloudFormation，請參該AWS CloudFormationCodeBuild 模板中的AWS CloudFormation使用者指南。

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 為 AWS CodeBuild 建立通知規則。

建立通知規則 (主控台)

1. 請登入AWS Management Console並開啟 CodeBuild 主控台，網址為 https://console.aws.amazon.com/codebuild/。
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. 如果您只想要提供給 Amazon 的資訊 EventBridge 包含在通知中，請在 [詳細資料類型] 中選擇 [基本]。如果您想要包含提供給 Amazon 的資訊以 EventBridge 及可能由 CodeBuild 或通知管理員提供的資訊，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：

   - 如果您已設定要與通知搭配使用的資源，請在 Choose target type (選擇目標類型) 中，選擇 AWS Chatbot (Slack) 或 SNS topic (SNS 主題)。在選擇目標中，選擇 Amazon SNS 主題的用戶端名稱 (針對在中設定的 Slack 用戶端AWS Chatbot) 或 Amazon 資源名稱 (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。
> - 您無法在建立通知規則時建立 AWS Chatbot 用戶端。如果您選擇 AWS Chatbot (Slack)，您會看到一個按鈕，指示您在 AWS Chatbot 中設定用戶端。選擇該選項會開啟 AWS Chatbot 主控台。如需詳細資訊，請參閱設定通知和 AWS Chatbot 之間的整合。
> - 如果您想要使用現有的 Amazon SNS 主題作為目標，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通知的必要政策。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. 您必須先訂閱使用者訂閱規則的 Amazon SNS 主題，才能收到通知。如需詳細資訊，請參閱訂閱使用者訂閱屬於目標的 Amazon SNS 主題。您也可以設定通知之間的整合，並AWS Chatbot將通知傳送到 Amazon Chime 聊天室。如需詳細資訊，請參閱設定通知和 AWS Chatbot 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。下列範例會顯示AWS針對識別碼**MyNotificationRule**為 *123* 456789012 的帳戶*MyBuildProject*中名為的建置專案命名的通知規則。建置成功MyNotificationTopic時，會將通知與完整詳細資料類型一起傳送至名為 *codestar-##*的 Amazon SNS 主題：

```
{
```

```
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "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 AWS CodeBuild 中檢視建置專案名稱清單

您可以使用AWS CodeBuild主控台、AWS CLI, 或AWS開發套件，以檢視 CodeBuild 中組建專案清單。

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)
- 檢視建置專案名稱清單 (AWS 開發套件)

檢視建置專案名稱清單 (主控台)

您可以在主控台的 AWS 區域中檢視建置專案的清單。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. 開啟AWS CodeBuild主控台，位於https://console.aws.amazon.com/codesuite/codebuild/home。
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：

- *####*：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-sort-date*：選用字串，用於表示依據*####*。有效值包括 ASCENDING 與 DESCENDING。
- *#####*：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"
  ]
}
```

檢視建置專案名稱清單 (AWS 開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

在 AWS CodeBuild 中檢視建置專案的詳細資訊

您可以使用AWS CodeBuild主控台、AWS CLI, 或AWS要在 CodeBuild 中組建專案的詳細資訊的開發套件。

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 開發套件)

檢視建置專案的詳細資訊 (主控台)

1. 開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
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. 在組建專案：*####*頁面中，選擇組建詳細資訊。

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：

- *##*：必要字串，用來表示要審視詳細資訊的一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。

所以此batch-get-projects命令不支持對某些屬性值進行過濾，但您可以編寫枚舉項目屬性的腳本。例如，以下 Linux shell 腳本枚舉當前帳户當前區域中的項目，並打印每個項目使用的映像。

```
#!/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
```

如需搭配使用 AWS CLI 與 AWS CodeBuild 的詳細資訊，請參閱命令列參考。

檢視建置專案的詳細資訊 (AWS 開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

在 AWS CodeBuild 中建立快取

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以下兩種快取類型之一：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，則必須以授權模式執行組建。CodeBuild 授予特權模式的專案會將其容器存取權授予所有裝置。如需詳細資訊，請參閱 Docker 文件網站上的執行階段權限和 Linux 功能。

主題

- Amazon S3 緩存

- 本機快取

Amazon S3 緩存

Amazon S3 快取會將快取存放在可跨多個建置主機使用的 Amazon S3 儲存貯體中。這是一個很好的選擇，對於建立比下載更昂貴的小型到中間大小的構建成品。這不是大型建置成品的最佳選項，因為它們在網路上傳輸所需的時間很長，而會影響建置效能。如果您使用 Docker 層，這也不是最佳選項。

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。對於中級到大型組建構件來說，這是一個不錯的選擇，因為快取會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，而您的專案並未使用 Git 儲存庫 (AWS CodeCommit GitHub、 GitHub 企業伺服器或 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。
  >
  >   依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 Linux 功能並啟用特權模式。此外，Windows 不支援特殊權限模式。
  > - 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名稱，則會覆寫來源項目。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計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 為使用 VPC 時，不支援本機快取。如需搭配使用 VPC 的詳細資訊 CodeBuild，請參閱搭 AWS CodeBuild 配 Amazon Virtual Private Cloud 使用。

主題

您可以使用AWS CLI、主控台、SDK 或指AWS CloudFormation定本機快取。

指定本機快取 (CLI)

您可以在 AWS CLI 中使用 `--cache` 參數指定三種本機快取類型的每一種。

- 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]
```

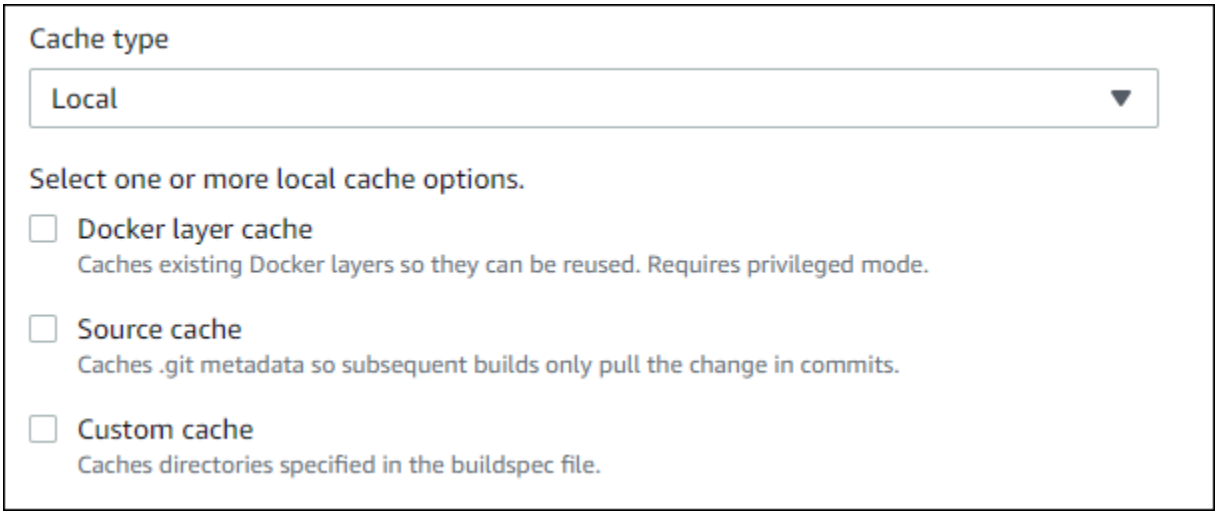
- 若要指定自訂快取：

```
--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。對於快取類型，請選擇 Amazon S3 或本機。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。



如需詳細資訊，請參閱 建立組建專案 (主控台)。

指定本機快取 (AWS CloudFormation)

如果您使用 AWS CloudFormation 指定本機快取，對於 Cache 屬性的 Type，請指定 LOCAL。以下 YAML 格式的 AWS CloudFormation 程式碼範例將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**
>
> 依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 Linux 功能並啟用特權模式。此外，Windows 不支援特殊權限模式。

如需更多詳細資訊，請參閱 建立建置專案 (AWS CloudFormation)。

在中構建觸發器 AWS CodeBuild

主題

- 建立 AWS CodeBuild 觸發
- 編輯 AWS CodeBuild 觸發條件

建立 AWS CodeBuild 觸發

創建AWS CodeBuild觸發器（控制台）

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 運算式的自訂規則來建立觸發器。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**
>
> 無法從建置觸發器、Amazon EventBridge 事件或AWS Step Functions任務啟動批次建置。

建立觸發

1. 請在以下位置開啟AWS CodeBuild主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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**
   >
   > 依預設，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**
   >
   > 開始時數和分鐘是從零開始的。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 開始的每日觸發器的開始小時為 12，開始分鐘為 15。每天午夜開始的每日觸發器的開始小時為零，開始分鐘為零。每天晚上 11:59 開始的每日觸發器的開始時間為 23，開始分鐘為 59。

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 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。
自訂	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 運算式會在每個工作日上午 9:00 **`0 9 ? * MON-FRI *`** 排程建置，如需詳細資訊，請參閱 Amazon CloudWatch 事件使用者指南中的 Cron 運算式。

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。

- 對於 Amazon S3，請輸入與您要建立之輸入成品版本對應的版本 ID。如果 Source version (來源版本) 為空白，將使用最新版本。
- 針對 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，請輸入對應至您要建置之原始程式碼版本的提交 ID、提取要求識別碼、分支名稱或標籤名稱。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 480 分鐘 (8 小時) 的逾時。此值指定 AWS CodeBuild 嘗試組建時多久之後停止。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)。

編程創建AWS CodeBuild觸發器

CodeBuild 使用 Amazon EventBridge 規則進行構建觸發器。您可以使用 EventBridge API 以程式設計方式建立 CodeBuild 專案的建置觸發程式。如需詳細資訊，請參閱 Amazon EventBridge API 參考。

編輯 AWS CodeBuild 觸發條件

編輯AWS CodeBuild觸發器（控制台）

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。您也可以編輯觸發器，將自訂規則與 Amazon CloudWatch cron 運算式搭配使用。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。

如何編輯觸發條件

1. 請在以下位置開啟AWS CodeBuild主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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**
>
> 依預設，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**
>
> 開始時數和分鐘是從零開始的。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 開始的每日觸發器的開始小時為 12，開始分鐘為 15。每天午夜開始的每日觸發器的開始小時為零，開始分鐘為零。每天晚上 11:59 開始的每日觸發器的開始時間為 23，開始分鐘為 59。

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 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。

頻率	必要參數	詳細資訊
自訂	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 運算式會在每個工作日上午 9:00 **`0 9 ? * MON-FRI *`** 排程建置，如需詳細資訊，請參閱 Amazon CloudWatch 事件使用者指南中的 Cron 運算式。

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/ 編輯中未提供的來源版本、逾時和其他選項AWS CodeBuild。

編輯AWS CodeBuild觸發程式

CodeBuild 使用 Amazon EventBridge 規則進行構建觸發器。您可以使用 EventBridge API 以程式設計方式編輯 CodeBuild 專案的建置觸發程式。如需詳細資訊，請參閱 Amazon EventBridge API 參考。

GitLab 連接

連線可讓您使用授權和建立將第三方供應商與您的 AWS 資源相關聯的組態 AWS CodeConnections。要將第三方存儲庫關聯為構建項目的源，請使用連接。

若要在中新增 GitLab 或 GitLab 自行管理來源提供者 CodeBuild，您可以選擇：

- 使用 CodeBuild 主控台 [建立組建專案] 精靈或 [編輯來源] 頁面來選擇GitLab或 [GitLab 自我管理的提供者] 選項。請參建立連線至 GitLab (主控台)閱新增來源提供者。主控台可協助您建立連線資源。

- 使用 CLI 建立連線資源，請參建立與 GitLab (CLI) 的連線閱使用 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**
>
> 通過授權此連接安裝 GitLab，您授予我們的服務權限通過訪問您的帳戶來處理您的數據，並且您可以通過卸載應用程序隨時撤消權限。

開始之前：

- 您必須已經在建立帳戶 GitLab。

  > **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**
  >
  > 您可以建立與具有 Owner 角色之儲存庫的連線 GitLab，然後連線可與具有諸如此類資源的存放庫一起使用 CodeBuild。如果是群組中的儲存庫，您不需要為群組擁有者。

- 若要指定組建專案的來源，您必須已在上建立儲存庫 GitLab。

主題

- 建立連線至 GitLab (主控台)
- 建立與 GitLab (CLI) 的連線

建立連線至 GitLab (主控台)

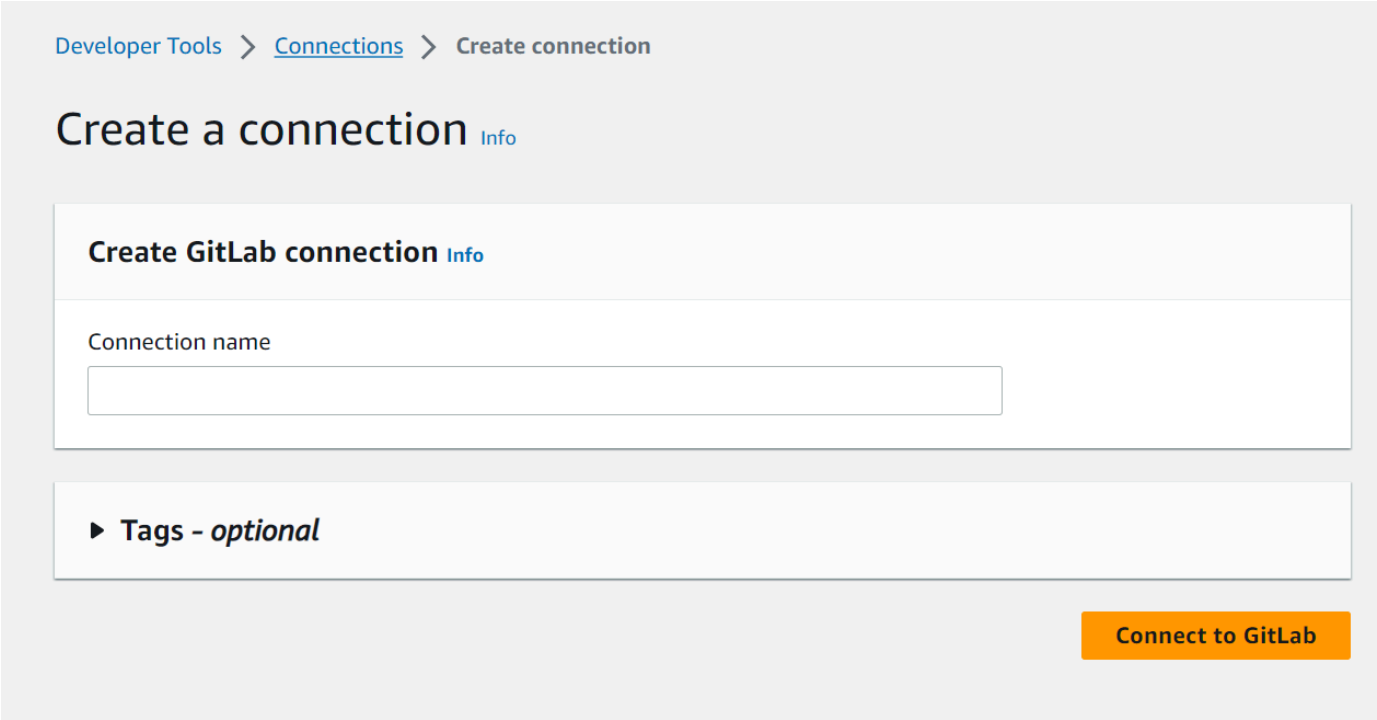
使用以下步驟使用 CodeBuild 控制台在中為您的專案 (存放庫) 新增連線 GitLab。

若要建立或編輯您的組建專案

1. 登入 CodeBuild 主控台。
2. 選擇下列其中一項。
   - 選擇建立建置專案。按照中的步驟完建立組建專案 (主控台)成第一個畫面，然後在「來源」區段的「來源提供者」下選擇GitLab。
   - 選擇編輯現有的組建專案。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。預設連線會在所有專案中套用預設 GitLab連線。
   - 在連線下，選擇自訂連線。自訂連線會套用覆寫帳戶預設設定的自訂 GitLab連線。
4. 執行以下任意一項：
   - 在 [預設連線] 或 [自訂連線] 底下，如果您尚未建立與提供者的連線，請選擇 [建立新 GitLab連線]。繼續執行步驟 5 以建立連線。
   - 在 [連線] 下方，如果您已建立與提供者的連線，請選擇連線。繼續執行步驟 10。

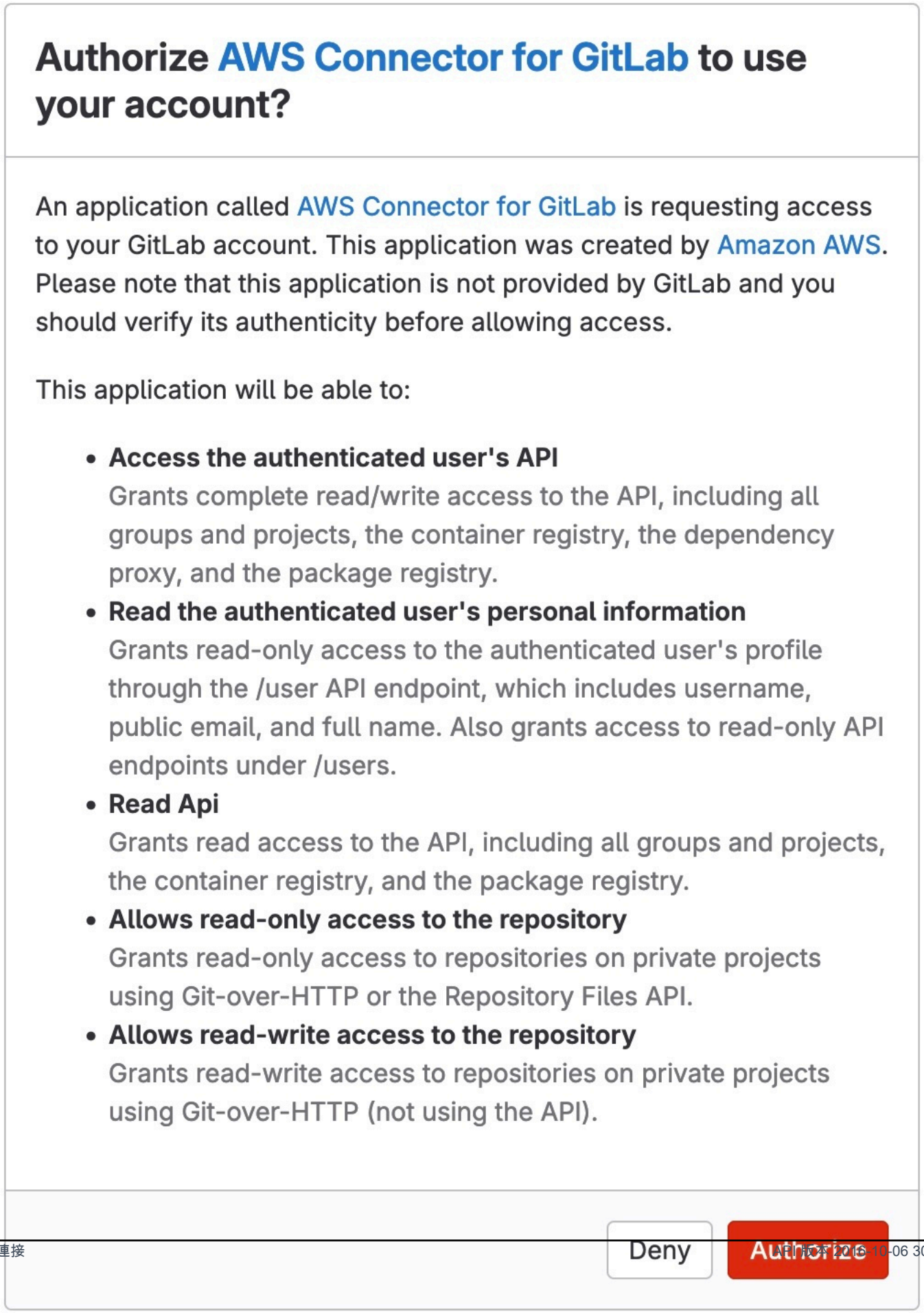
   > **Note**
   >
   > 如果您在建立 GitLab 連線之前關閉快顯視窗，則需要重新整理頁面。

5. 若要建立 GitLab 存放庫的連線，請在 [選取提供者] 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「Connect 至」 GitLab。



6. GitLab 顯示的登入頁面時，使用您的認證登入，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，其中會顯示一則訊息，要求授權連線存取您的 GitLab 帳戶。

   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).
Deny
Authorize

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，上方會顯示成功橫幅。
10. 在 [建立組建專案] 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，請選擇刷新按鈕以顯示它。
11. 在 Repository 中，透過使用命名空間指定專案路徑 GitLab來選擇中的專案名稱。例如，對於群組層級存放庫，請以下列格式輸入存放庫名稱：`group-name/repository-name`如需有關路徑和命名空間的詳細資訊，請參閱 https://docs.gitlab.com/ee/api/projects.html #中的欄`path_with_namespace`位get-single-project。如需有關中命名空間的詳細資訊 GitLab，請參閱 https://docs.gitlab.com/ee/user/namespace/。

    > **Note**
    >
    > 對於中的群組 GitLab，您必須使用命名空間手動指定專案路徑。例如，對於群組`myrepo`中名為的存放庫`mygroup`，請輸入以下內容：`mygroup/myrepo`您可以在中的 URL 中找到具有命名空間的專案路徑 GitLab。

12. 在來源版本-選用中，輸入提取要求識別碼、分支、提交 ID、標籤或參照，以及提交 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

    > **Note**
    >
    > 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

13. 在 Git clone 深度-可選中，您可以創建一個歷史記錄截斷為指定數量的提交淺層克隆。如果您想要完整複製，請選擇 Full (完整)。
14. 如果您想要將組建的開始和完成狀態報告給來源提供者，請在 [組建狀態-選用] 中，選取 [在組建開始和完成時向來源提供者報告組建狀態給來源提供者]。

    若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

建立與 GitLab (CLI) 的連線

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**
>
> 依預設，透過 AWS CLI 或建立 AWS CloudFormation 的連線處於PENDING狀態。建立與 CLI 的連線之後 AWS CloudFormation，或使用主控台編輯連線以顯示其狀態AVAILABLE。

建立連線

- 請依照開發人員工具主控台使用者指南中的建立連線 GitLab (CLI) 中的指示進行。

使用網路掛鉤 AWS CodeBuild

AWS CodeBuild 支持 Webhook 集成 GitHub， GitHub 企業服務器 GitLab， GitLab 自我管理和比特桶。

主題

- 搭配使用網路掛接的最佳做法 AWS CodeBuild
- 比特桶網絡鉤事件
- GitHub 網絡掛鉤事件
- GitLab 網絡掛鉤事件

搭配使用網路掛接的最佳做法 AWS CodeBuild

對於使用公共存儲庫來設置 webhook 的項目，我們建議您使用以下選項：

設置ACTOR_ACCOUNT_ID過濾器

將過ACTOR_ACCOUNT_ID濾器添加到項目的 webhook 過濾器組中，以指定哪些用戶可以觸發構建。傳遞給的每個 webhook 事件都會 CodeBuild 隨附指定演員標識符的發送者信息。 CodeBuild 將根據過濾器中提供的正則表達式模式過濾 Webhook。您可以指定允許使用此篩選器觸發組建的特定使用者。如需詳細資訊，請參閱 GitHub 網絡掛鉤事件 及 比特桶網絡鉤事件。

設置FILE_PATH過濾器

將過FILE_PATH濾器添加到項目的 webhook 過濾器組中，以包含或排除可以在更改時觸發構建的文件。例如，您可以使用規則運算式模式 (例如^buildspec.yml$，連同excludeMatchedPattern屬性) 拒絕buildspec.yml檔案變更的建置要求。如需詳細資訊，請參閱 GitHub 網絡掛鉤事件 及 比特桶網絡鉤事件。

縮小組建 IAM 角色的許可範圍

由 webhook 觸發的組建會使用專案中指定的 IAM 服務角色。我們建議您將服務角色中的權限設定為執行組建所需的最低權限集。例如，在測試和部署案例中，建立一個專案進行測試，另一個專案進行部署。測試項目接受來自存儲庫的 webhook 構建，但不提供對資源的寫入權限。部署專案會為您的資源提供寫入權限，而 webhook 篩選器設定為只允許受信任的使用者觸發組建。

使用內嵌或 Amazon S3 存儲的構建規格

如果您在專案本身內嵌定義組建規格，或將 Buildspec 檔案存放在 Amazon S3 儲存貯體中，則只有專案擁有者才能看到該建置規格檔案。這樣可以防止提取請求對 buildspec 文件進行代碼更改並觸發不需要的構建。如需詳細資訊，請參閱 API 參ProjectSource考資料中的 CodeBuild .buildspec。

比特桶網絡鉤事件

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，您可以指定只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。如果任何過濾器組評估為 true，則會觸發構建，當組中的所有過濾器評估為 true 時發生。當您建立篩選群組時，您可以指定這些項目：

一個事件

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。如果您建立使用該declined事件類型的 Webhook 篩選器群組，您也必須在 Bitbucket Webhook 設定中啟用該PULL_REQUEST_CLOSED事件。

X-Event-Key 標頭值	事件類型
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED
pullrequest:rejected	PULL_REQUEST_CLOSED

對於PULL_REQUEST_MERGED，如果拉取請求與壓縮策略合併，並且拉取請求分支關閉，則原始提取請求提交將不再存在。在這種情況下，CODEBUILD_WEBHOOK_MERGE_COMMIT環境變量包含壓縮合併提交的標識符。

一個或多個可選的過濾器

使用規則表達式來指定篩選條件。對於觸發構建的事件，與其關聯的組中的每個過濾器都必須評估為 true。

ACTOR_ACCOUNT_ID（ACTOR_ID在控制台中）

當 Bitbucket 帳戶 ID 與正則表達式模式匹配時，webhook 事件會觸發構建。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

當頭部引用匹配正則表達式模式（例如，refs/heads/branch-name和refs/tags/tag-name）時，webhook 事件觸發構建。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`

當基本引用與正則表達式模式匹配時，webhook 事件會觸發構建。`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`

當更改文件的路徑與正則表達式模式匹配時，webhook 觸發構建。

`COMMIT_MESSAGE`

當頭提交消息與正則表達式模式匹配時，webhook 觸發構建。

`WORKFLOW_NAME`

當工作流程名稱符合規則運算式模式時，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 (AWS CloudFormation)

篩選 Bitbucket Webhook 事件 (主控台)

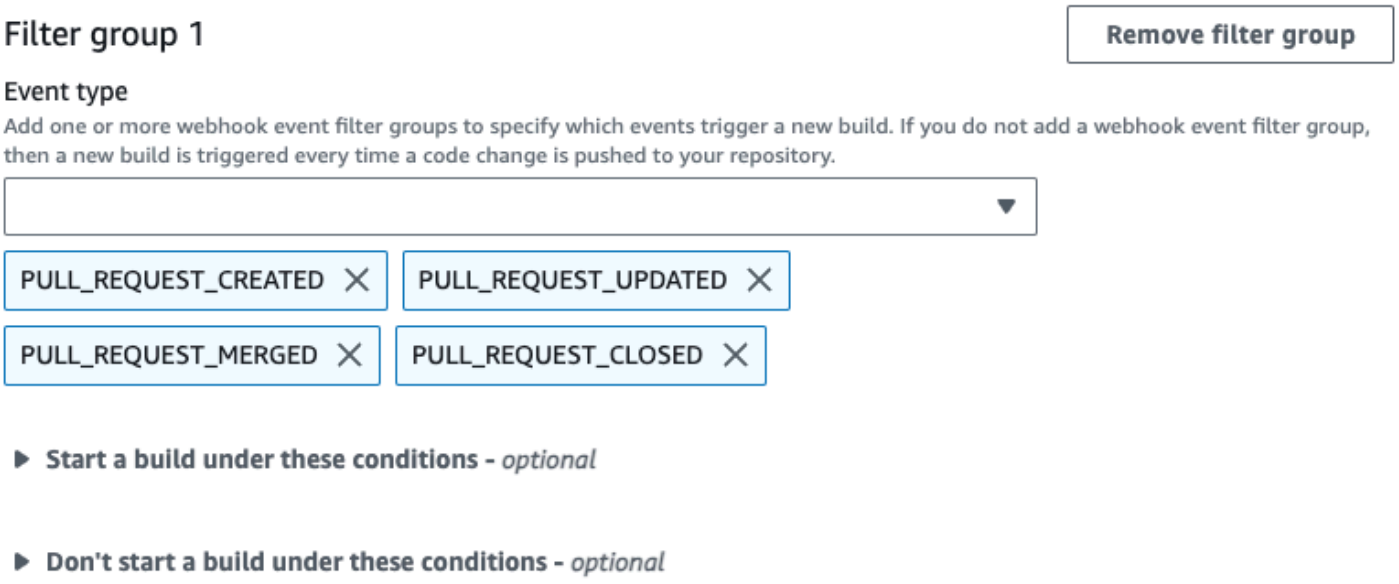
若要使用 AWS Management Console 來篩選網路掛接事件：

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。

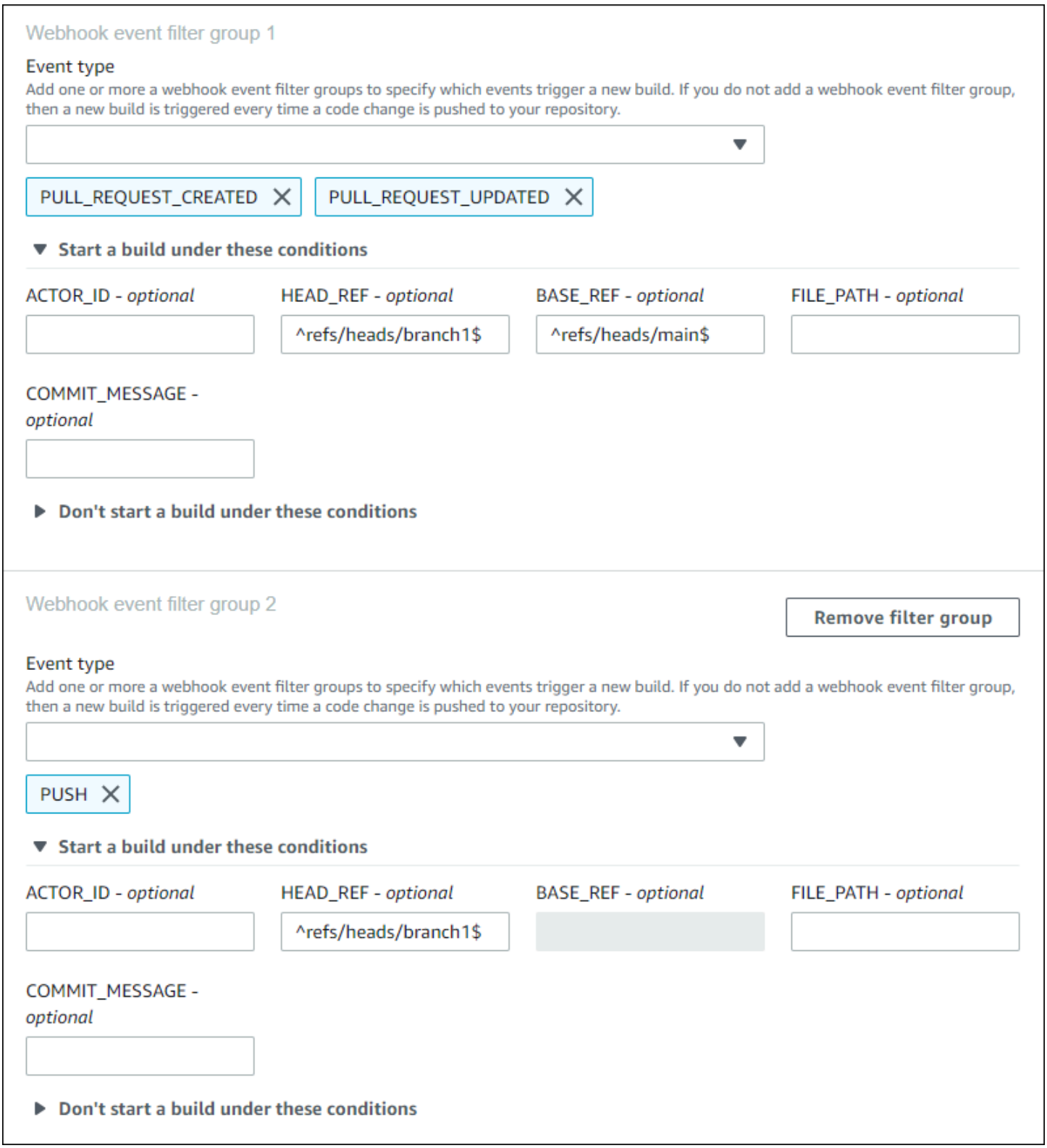
如需詳細資訊，請參閱 AWS CodeBuild API 參考WebhookFilter中的建立組建專案 (主控台)和。

在這個範例中，Webhook 篩選群組僅針對提取請求觸發組建：

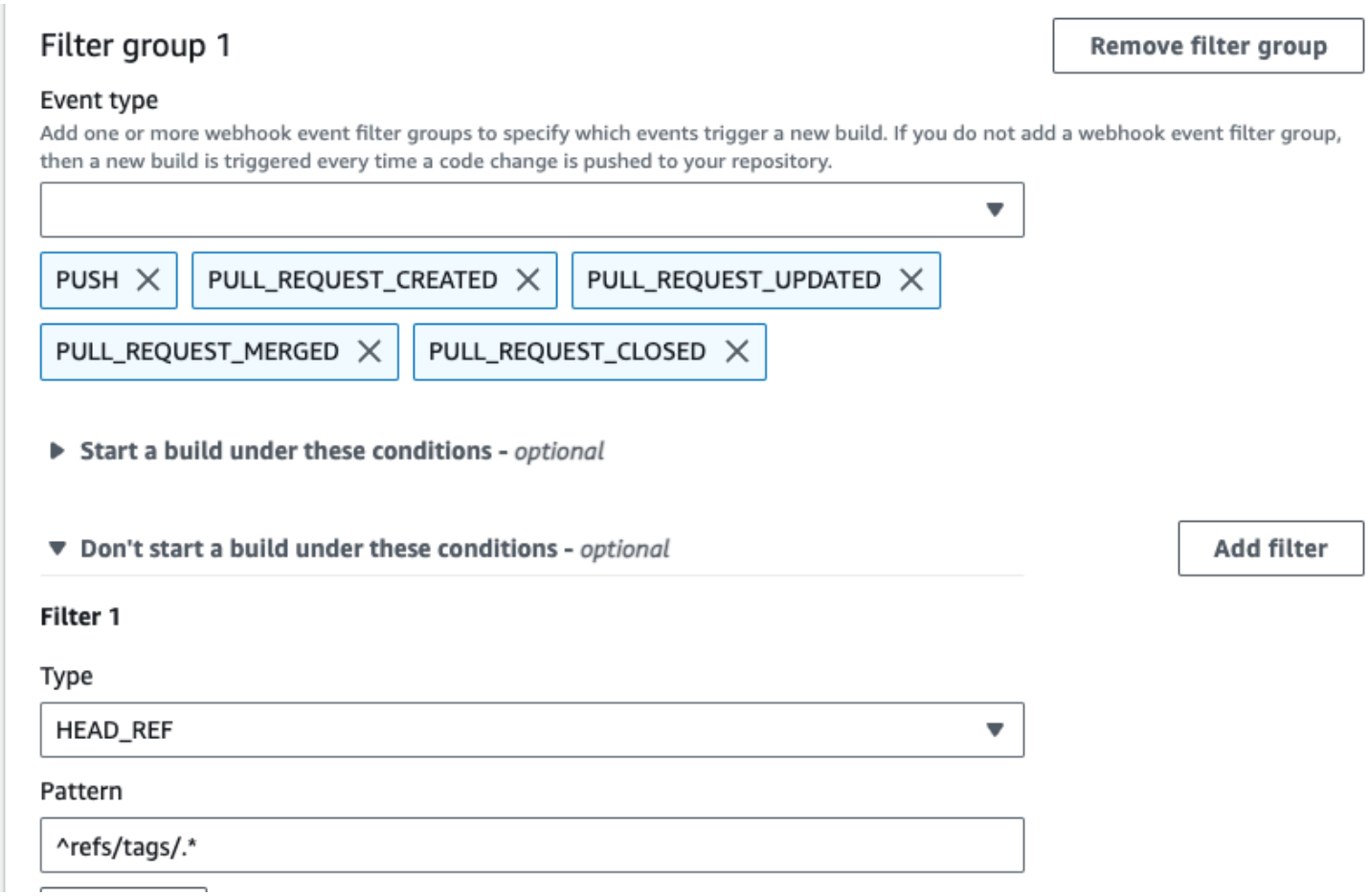


在有兩個篩選群組的範例中，當一個或兩個篩選群組評估為 true 時，就會觸發組建：

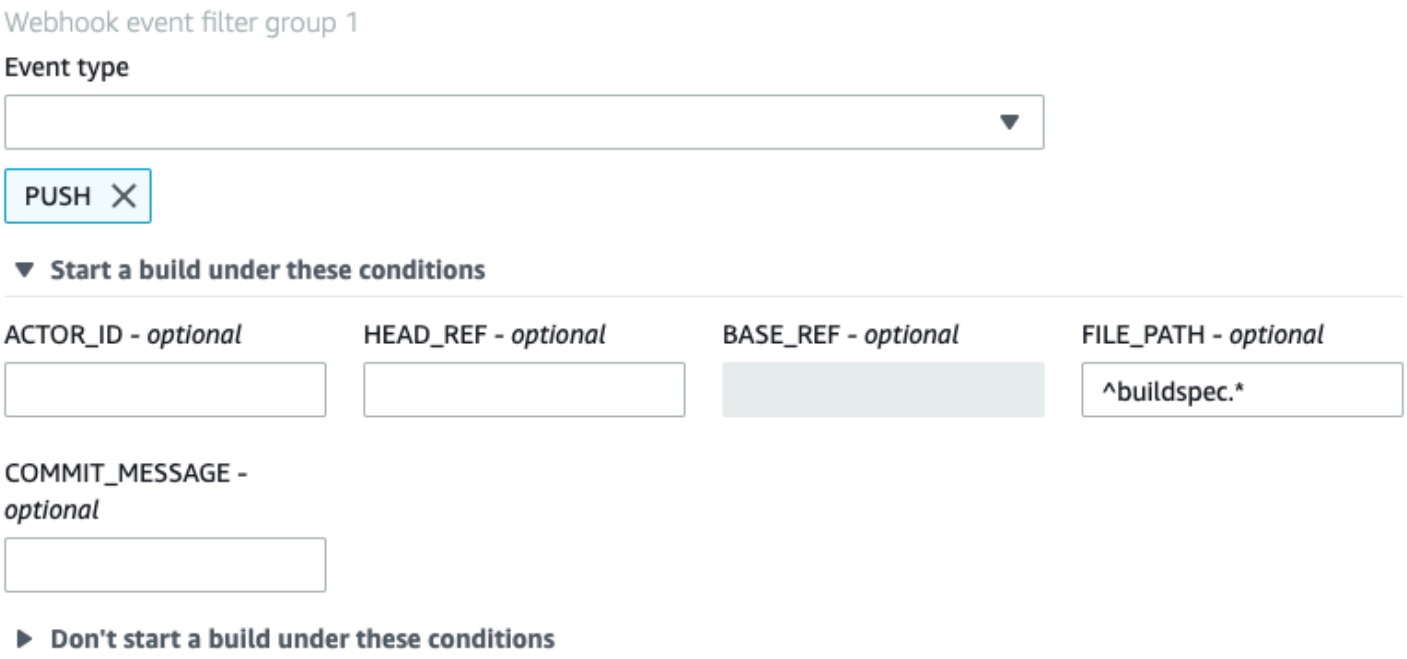
- 第一個篩選群組指定在分支上建立或更新的提取請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/main$`，而標頭參考符合 `^refs/heads/branch1!`。
- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/branch1$`。



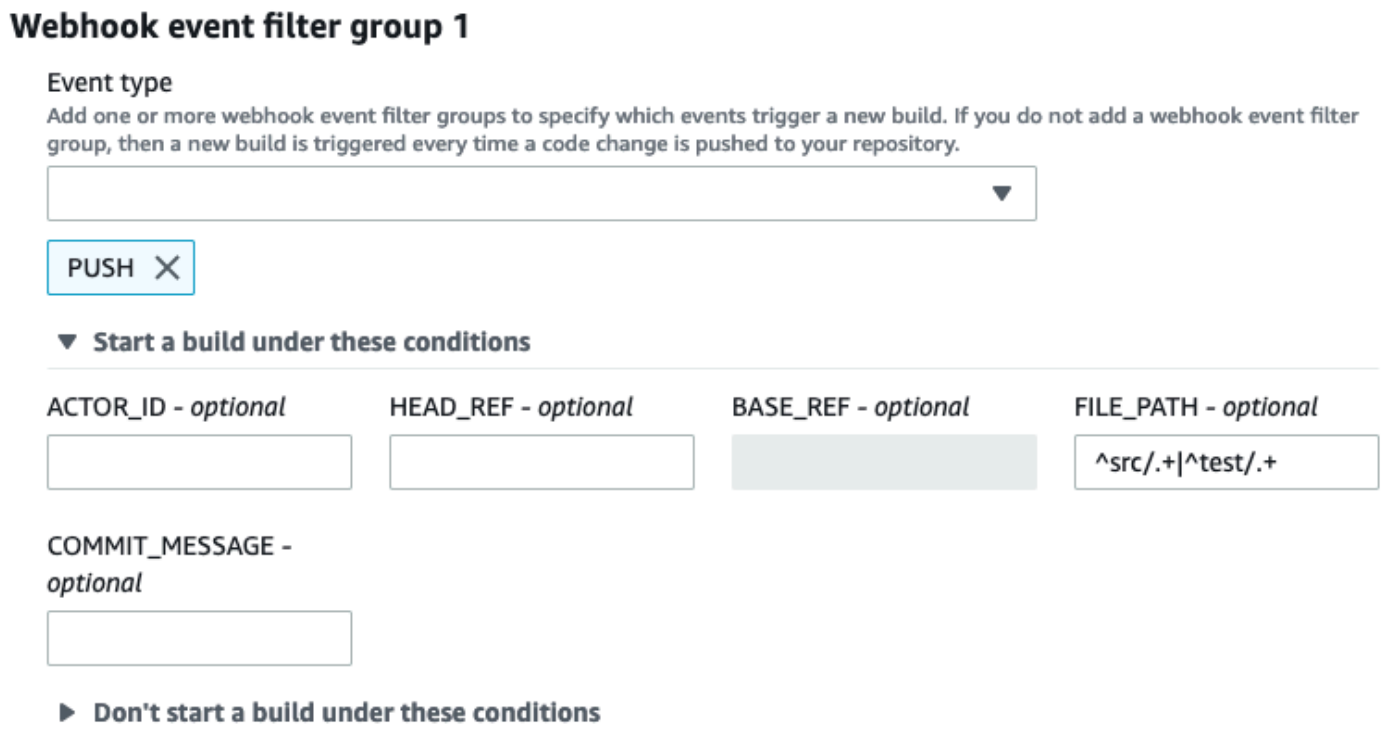
在這個範例中，Webhook 篩選群組針對所有請求 (標籤事件除外) 觸發組建。



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。

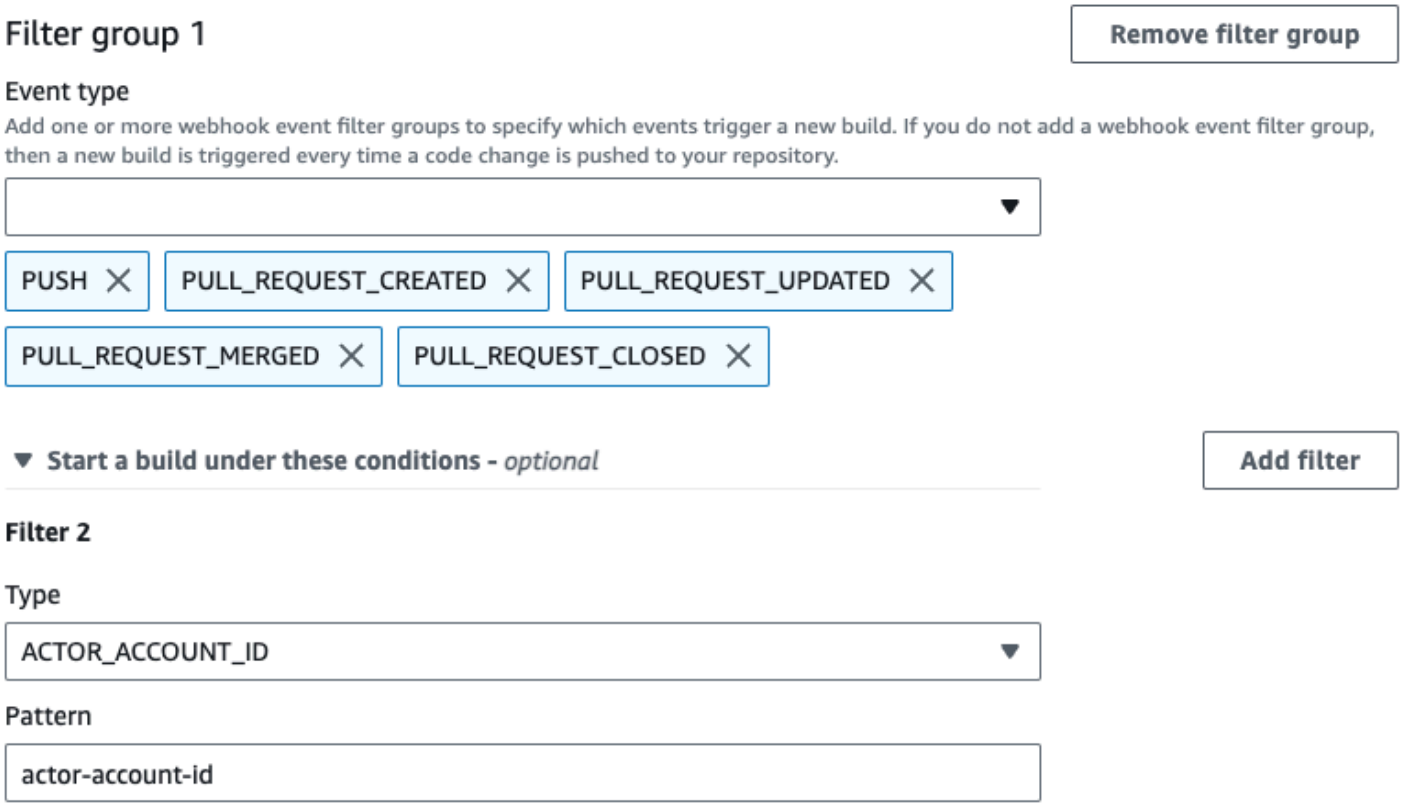


在此範例中，Webhook 篩選器群組只會在`src`或`test`資料夾中變更檔案時觸發組建。

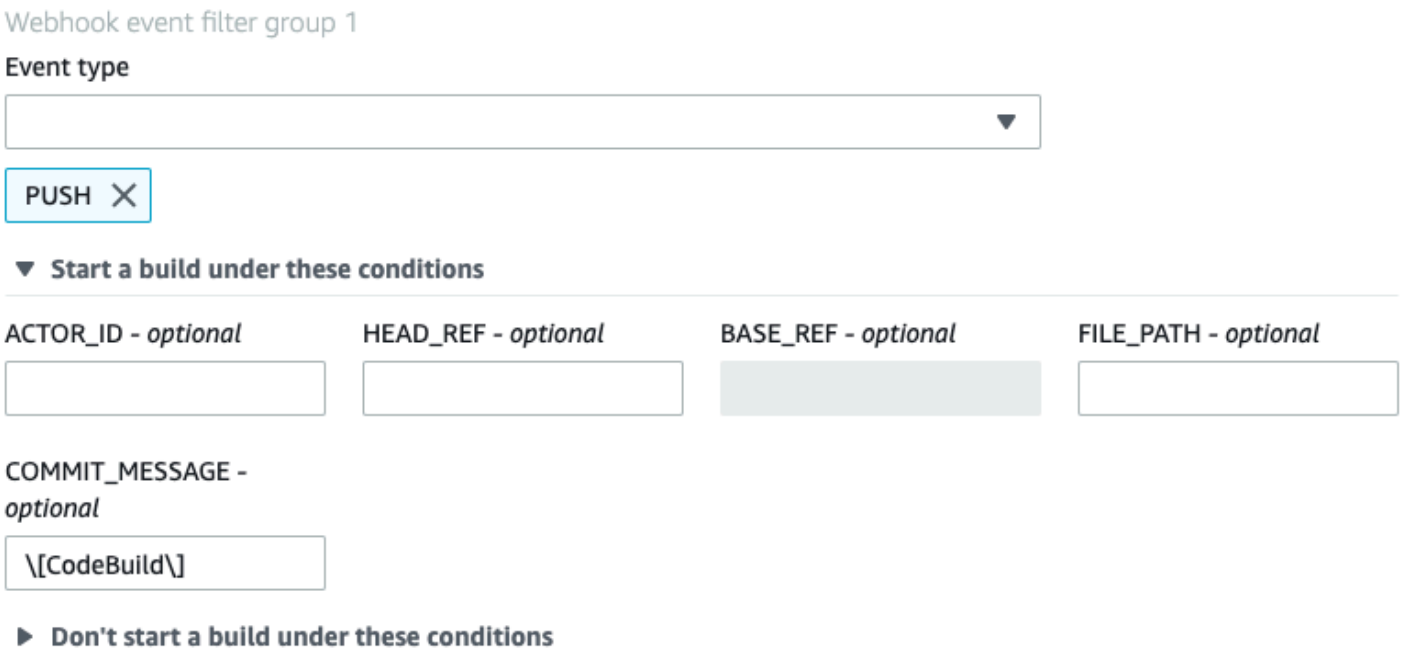


在這個範例中，只有當 Bitbucket 使用者進行變更，而且其帳戶 ID 不符合規則表達式 `actor-account-id` 時，Webhook 篩選群組才會觸發建置。

> **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。



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。



篩選 Bitbucket Webhook 事件 (開發套件)

若要使用 AWS CodeBuild SDK 來篩選網路掛接事件，請在`CreateWebhook`或 `UpdateWebhook` API 方法的要求語法中使用`filterGroups`欄位。如需詳細資訊，請參閱 CodeBuild API 參考WebhookFilter中的。

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，篩選器群組會指定只有在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 (AWS CloudFormation)

若要使用 AWS CloudFormation 範本來篩選 webhook 事件，請使用 AWS CodeBuild 專案的FilterGroups屬性。在 AWS 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 網絡掛鉤事件

您可以使用 webhook 過濾器組來指定哪些 GitHub webhook 事件觸發構建。例如，您可以指定只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。如果任何過濾器組評估為 true，則會觸發構建，當組中的所有過濾器評估為 true 時發生。當您建立篩選群組時，您可以指定這些項目：

一個事件

對於 GitHub，您可以選擇下列一或多個事件：PUSHPULL_REQUEST_CREATED、PULL_REQUEST_UPDATED、、PULL_REQUEST_REOPENED、PUL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

X-GitHub-Event 標頭值	Webhook 事件承載 **action** 值	事件類型
release	發布	RELEASED
release	預發行	PRERELEASED
workflow_job	queued	WORKFLOW_JOB_QUEUED

> **Note**
>
> PULL_REQUEST_REOPENED事件類型只能與 GitHub GitHub 企業伺服器搭配使用。RELEASEDPRERELEASED、和WORKFLOW_JOB_QUEUED事件類型 GitHub 只能搭配使用。如需 WORKFLOW_JOB_QUEUED 的詳細資訊，請參閱自學課程：設定 CodeBuild 自我託管的 GitHub 動作亞軍。

一個或多個可選的過濾器

使用規則表達式來指定篩選條件。對於觸發構建的事件，與其關聯的組中的每個過濾器都必須評估為 true。

ACTOR_ACCOUNT_ID（ACTOR_ID在控制台中）

當 GitHub 或 GitHub企業伺服器帳戶識別碼符合規則運算式模式時，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

當頭部引用匹配正則表達式模式（例如，refs/heads/branch-name或refs/tags/tag-name）時，webhook 事件觸發構建。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

當基底參考符合規則運算式模式 (例如refs/heads/branch-name) 時，webhook 事件會觸發組建。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`

當更改文件的路徑與正則表達式模式匹配時，webhook 觸發構建。`FILE_PATH`篩選器可與 GitHub 推送和提取要求事件和 GitHub企業伺服器推送事件搭配使用。它不能與 GitHub企業服務器提取請求事件一起使用。

`COMMIT_MESSAGE`

當頭提交消息與正則表達式模式匹配時，webhook 觸發構建。`COMMIT_MESSAGE`篩選器可與 GitHub 推送和提取要求事件和 GitHub企業伺服器推送事件搭配使用。它不能與 GitHub企業服務器提取請求事件一起使用。

`TAG_NAME`

當釋放的標籤名稱與正則表達式模式匹配時，webhook 觸發構建。`TAG_NAME`篩選器可與 GitHub 已發行和預先發行的要求事件搭配使用。

`RELEASE_NAME`

當發行名稱與正則表達式模式匹配時，webhook 觸發構建。`RELEASE_NAME`篩選器可與 GitHub 已發行和預先發行的要求事件搭配使用。

`WORKFLOW_NAME`

當工作流程名稱符合規則運算式模式時，webhook 會觸發組建。`WORKFLOW_NAME`篩選器可與 GitHub 「動作」工作流程工作排入佇列的要求事件搭配使用。

> **Note**
>
> 您可以在存儲庫的 webhook 設置中找到 webhook 有效負載。 GitHub

主題

- 過濾 GitHub 網絡掛鉤事件（控制台）
- 篩選 GitHub 網路掛鉤事件 (SDK)
- 篩選 GitHub 網路掛鉤事件 ()AWS CloudFormation

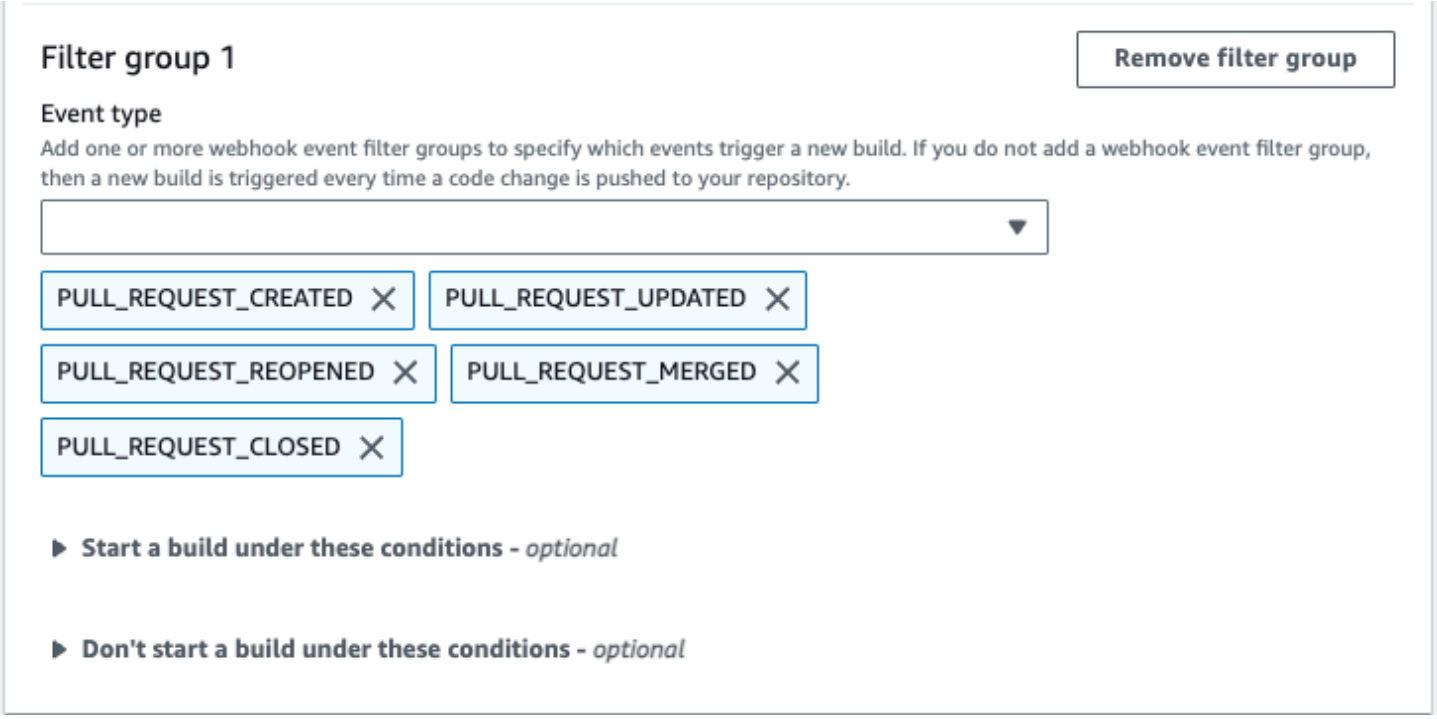
過濾 GitHub 網絡掛鉤事件（控制台）

在主要來源 Webhook 事件中，選取下列項目。只有當您針對來源儲存庫選擇 [我的 GitHub帳戶中的儲存庫] 時，才能使用此區段。

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「新增過濾器群組」以新增其他過濾器群組

如需詳細資訊，請參閱 AWS CodeBuild API 參考WebhookFilter中的建立組建專案 (主控台)和。

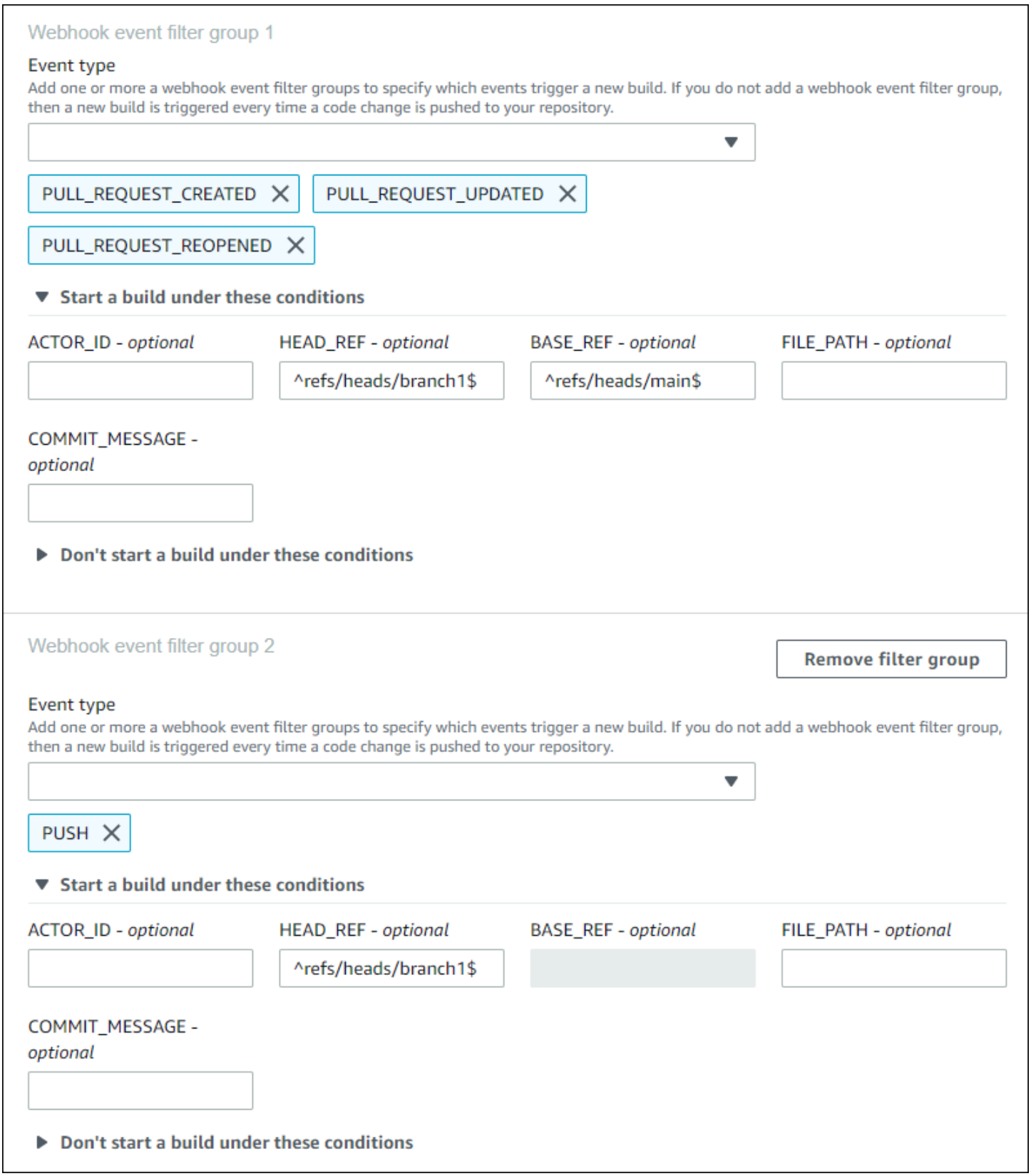
在這個範例中，Webhook 篩選群組僅針對提取請求觸發組建：



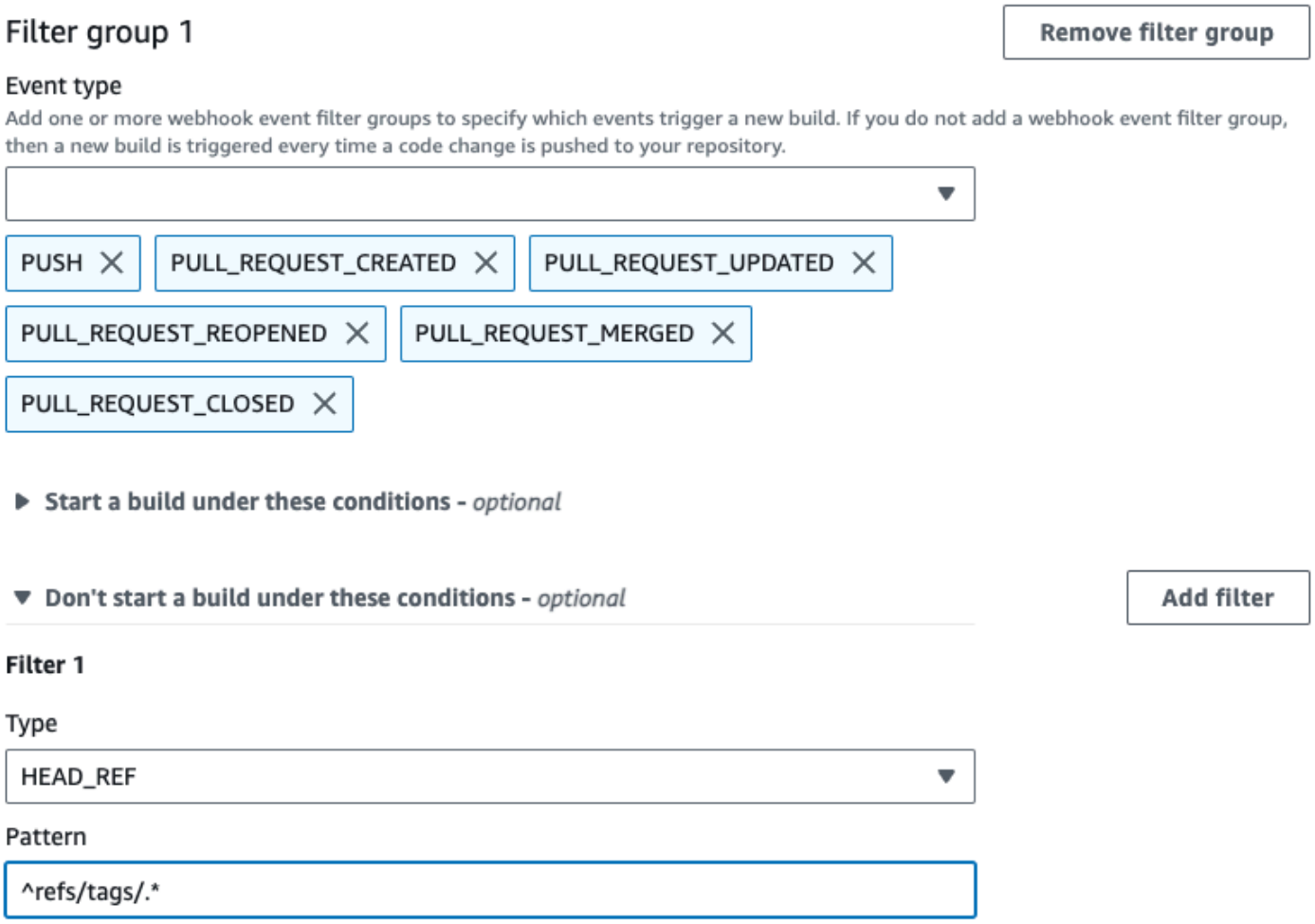
在有兩個 Webhook 篩選群組的範例中，當一個或兩個篩選群組評估為 true 時，就會觸發組建：

- 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$`，而標頭參考符合 `^refs/heads/branch1$`。

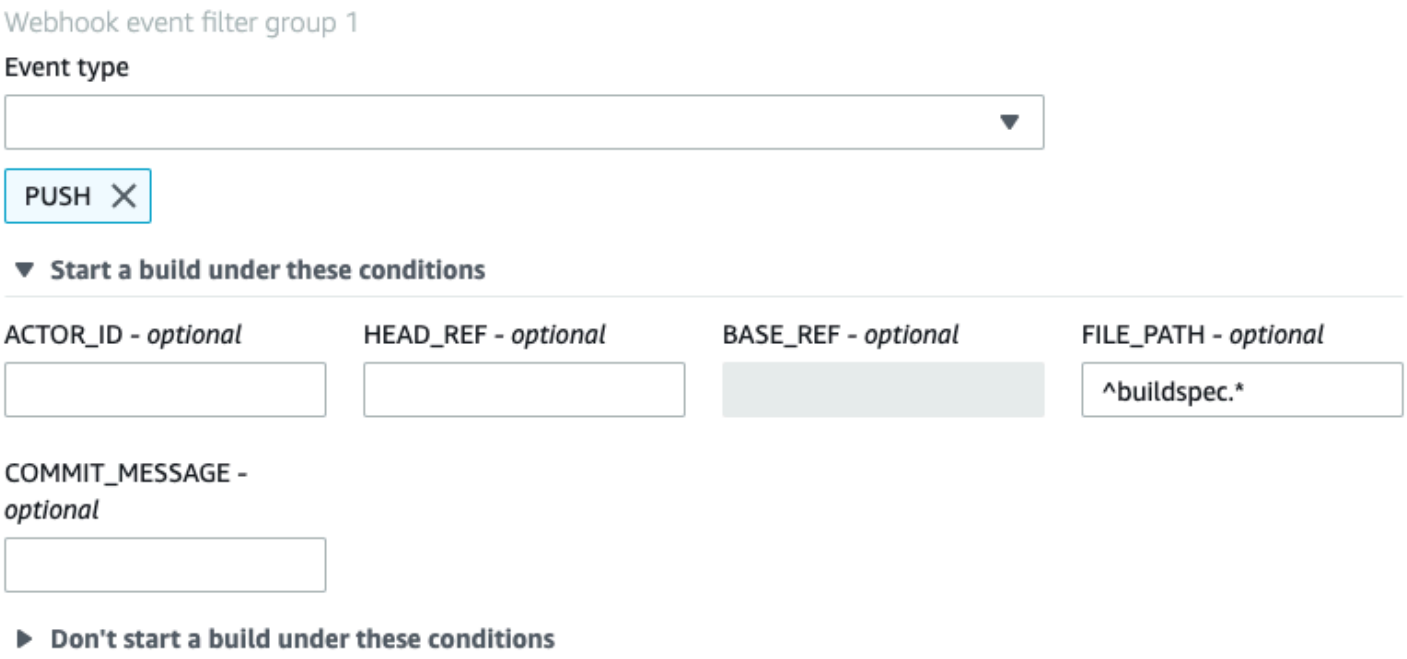
- 第二個篩選群組在分支上指定推送請求，並且這些分支的 Git 參考名稱符合規則表達式 `^refs/heads/branch1$`。



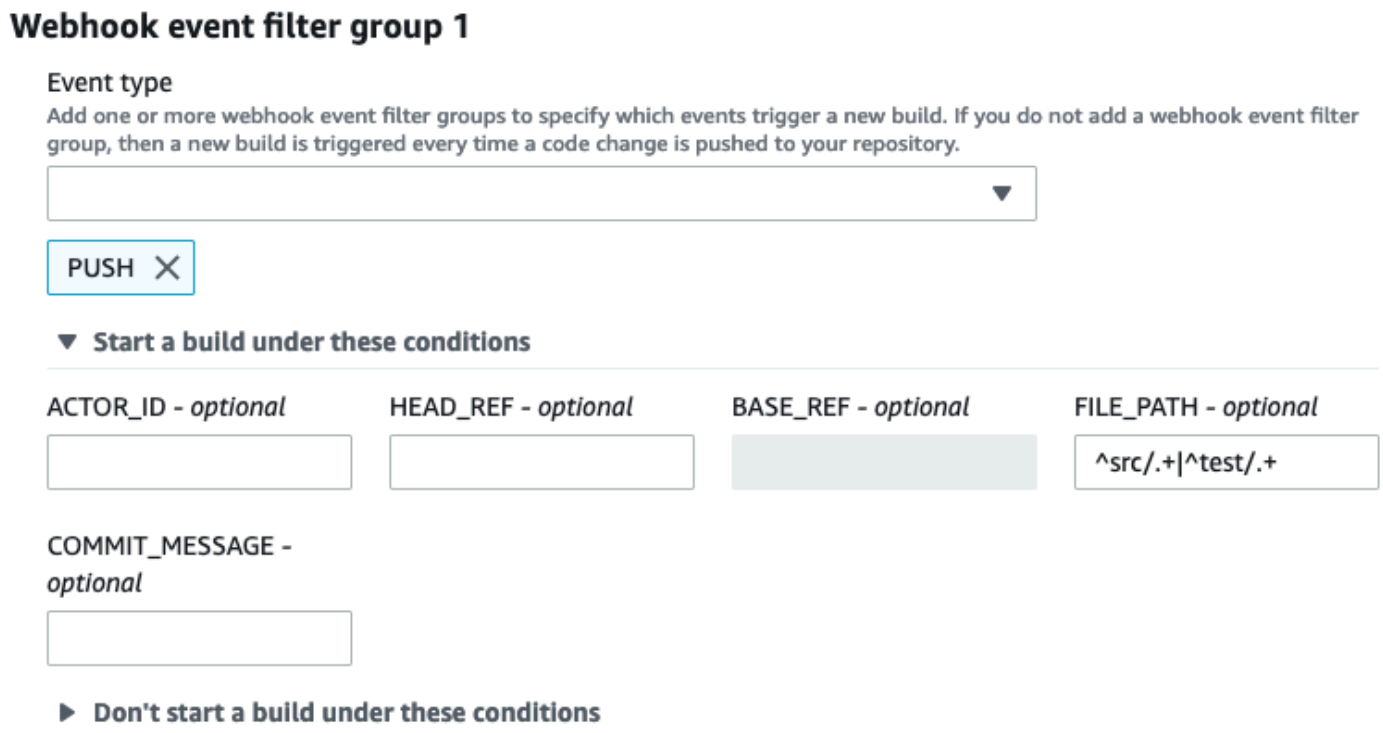
在這個範例中，Webhook 篩選群組針對所有請求 (標籤事件除外) 觸發組建。



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。

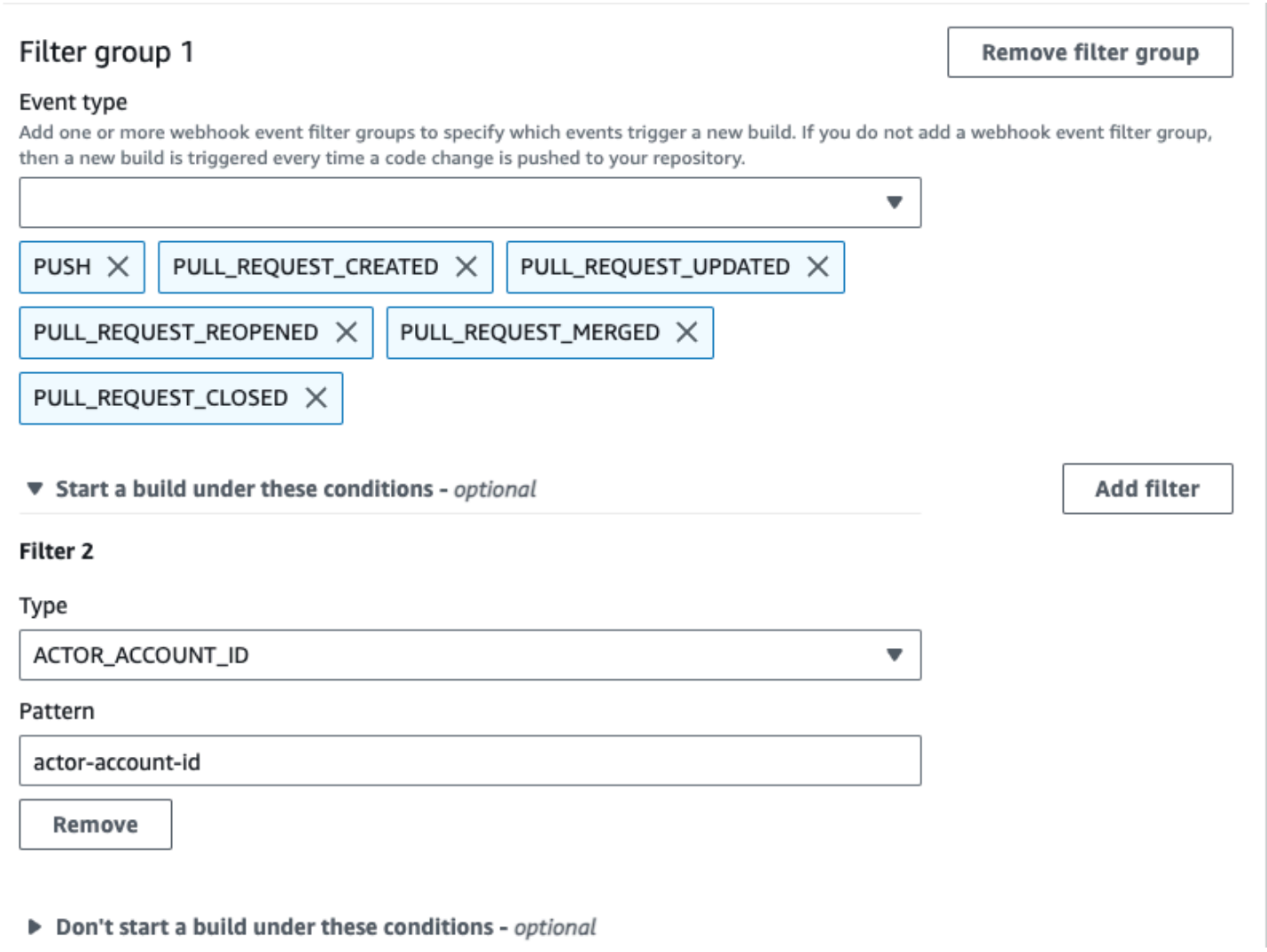


在此範例中，Webhook 篩選器群組只會在`src`或`test`資料夾中變更檔案時觸發組建。

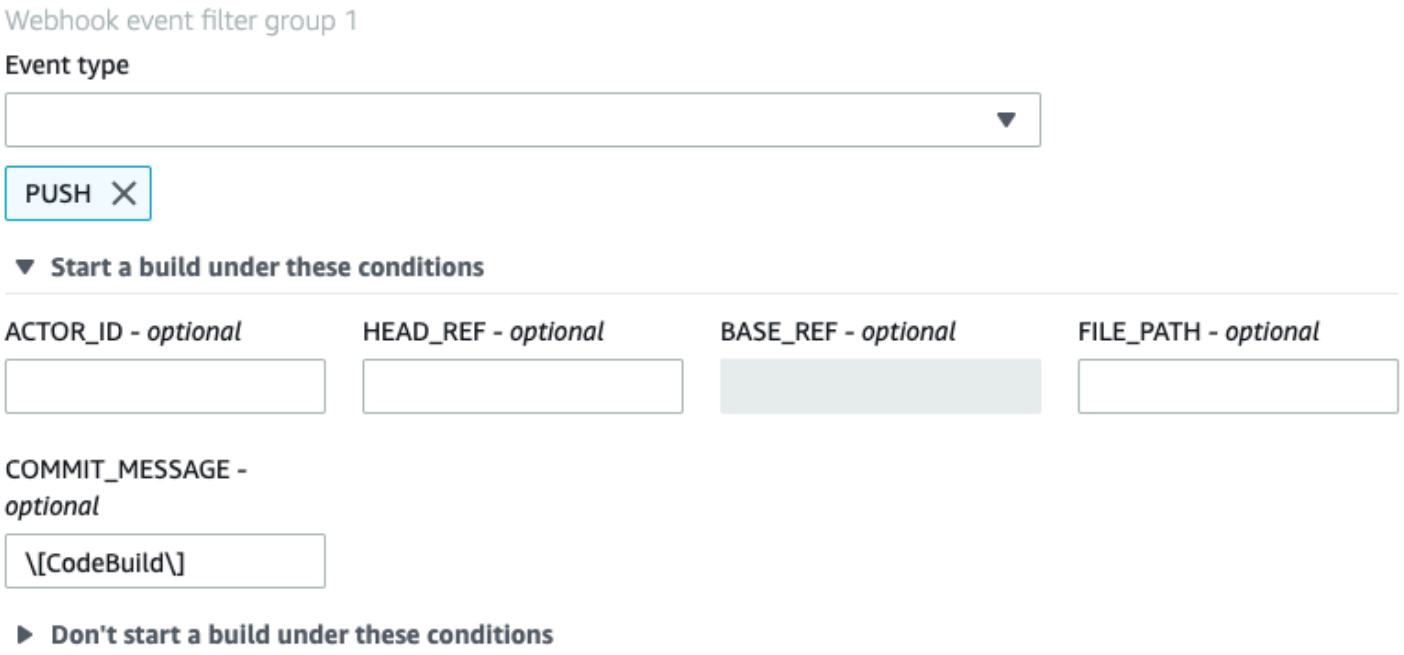


在此範例中，只有當指定的 GitHub 或 GitHub Enterprise Server 使用者進行變更時，Webhook 篩選器群組才會觸發組建，且帳戶識別碼符合規則運算式`actor-account-id`。

> **Note**
>
> 如需如何尋找 GitHub 帳戶 ID 的詳細資訊，請參閱 https://api.github.com/users/ #####，其中使用###是您的 GitHub 使用者名稱。

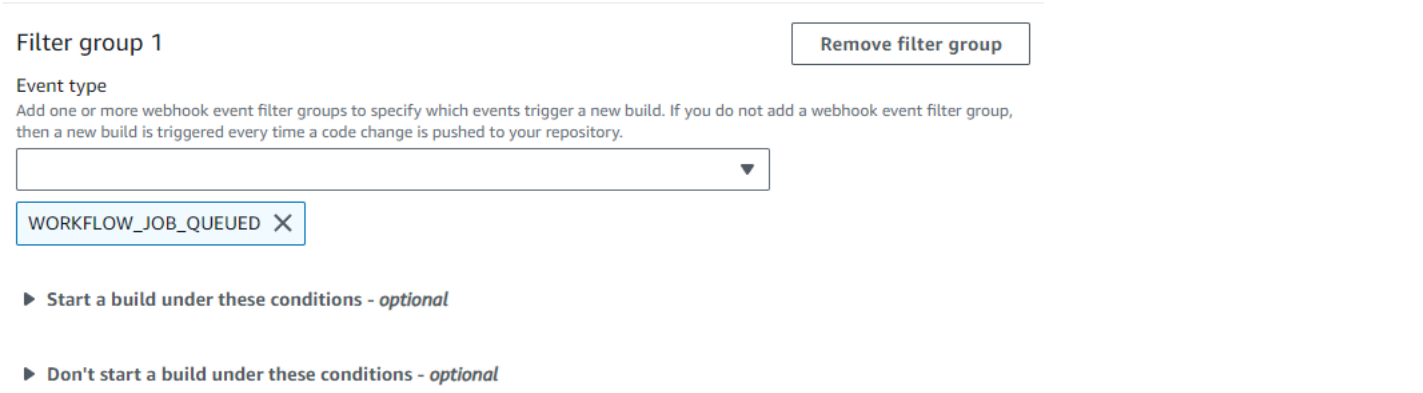


在此範例中，當 head 提交訊息符合規則運算式 `\[CodeBuild\]` 時，webhook 篩選群組會觸發推送事件的建置。

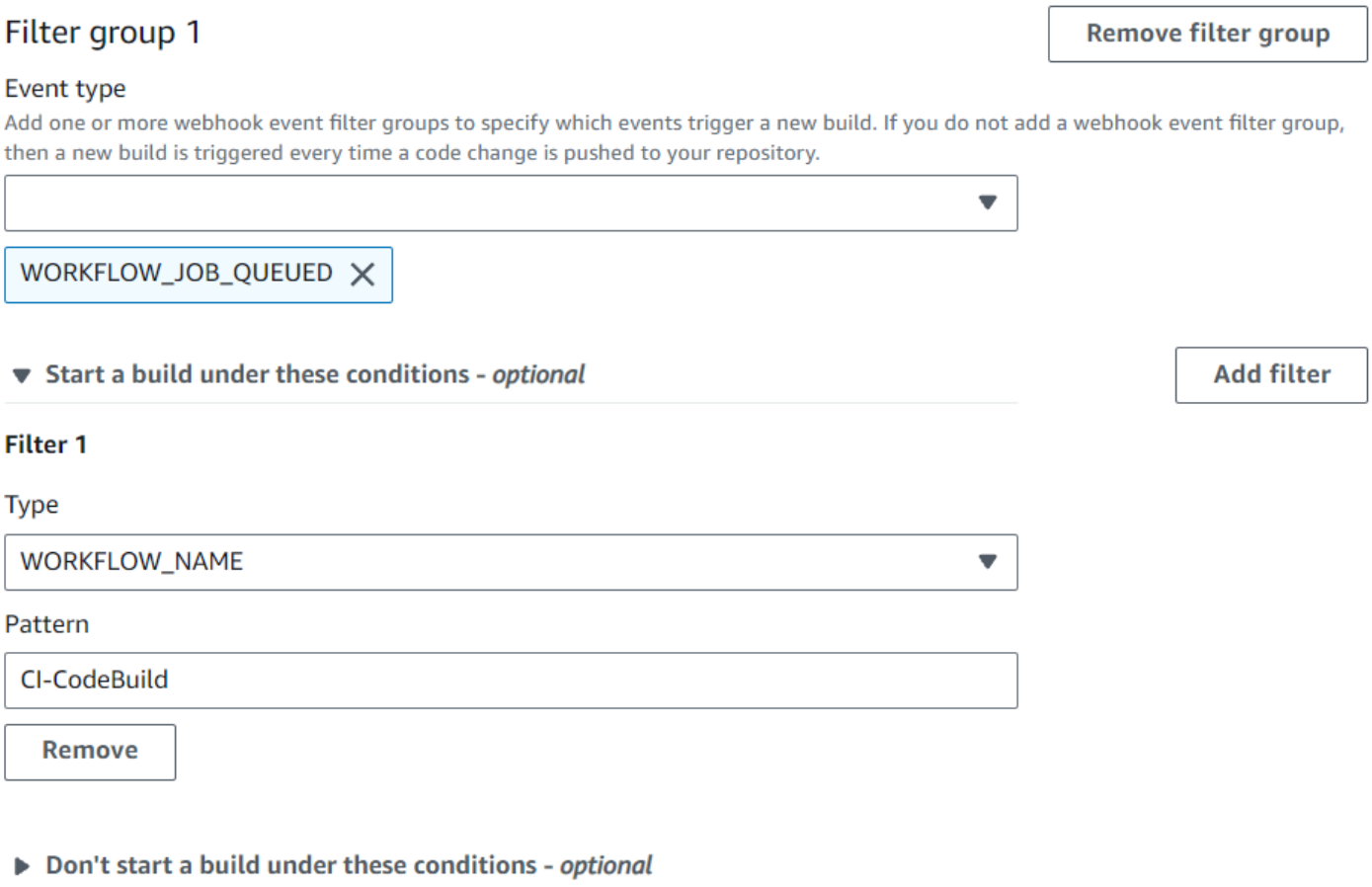


在此範例中，Webhook 篩選器群組只會觸發 GitHub 「動作」工作流程工作事件的組建。

> **Note**
>
> CodeBuild 只有當網路勾點具有包含 WORKFLOW_JO B_QUEED 事件篩選器的篩選器群組時，才會處理 GitHub 「動作」工作流程工作。



在此範例中，webhook 篩選器群組會針對符合規則運算式`CI-CodeBuild`的工作流程名稱觸發組建。



篩選 GitHub 網路掛鉤事件 (SDK)

若要使用 AWS CodeBuild SDK 來篩選網路掛接事件，請在`CreateWebhook`或 `UpdateWebhook` API 方法的要求語法中使用`filterGroups`欄位。如需詳細資訊，請參閱 CodeBuild API 參考[WebhookFilter](#)中的。

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，篩選器群組會指定只有在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/.+"
        }
```

```
    ]
]
```

您可以建立篩選器，僅在指定 GitHub 或具有帳戶 ID 的 GitHub 企業伺服器使用者進行變更時觸發組建`actor-account-id`。

> **Note**
>
> 如需如何尋找 GitHub 帳戶 ID 的詳細資訊，請參閱 https://api.github.com/users/ ##### ，其中使用###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": "\[CodeBuild\]"
        }
    ]
```

```
]
```

若要建立僅針對「動作」工作流程 GitHub 作業觸發組建的 Webhook 篩選器，請將下列內容插入要求語法中：

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "WORKFLOW_JOB_QUEUED"
        }
    ]
]
```

篩選 GitHub 網路掛鉤事件 ()AWS CloudFormation

若要使用 AWS CloudFormation 範本來篩選 webhook 事件，請使用 AWS CodeBuild 專案的`FilterGroups`屬性。在 AWS 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：

- 第一個過濾器組指定在具有 Git 引 GitHub 用名稱匹配正則表達式的分支上創建或更新提取請求，`^refs/heads/main$`由沒有帳戶 ID 的用戶`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 READ_ME，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。
- 第四個篩選群組會指定 GitHub 「動作」工作流程工作要求，其工作流程名稱與一般運算式相符`\[CI-CodeBuild\]`。

```
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
      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 網絡掛鉤事件

您可以使用 webhook 過濾器組來指定哪些 GitLab webhook 事件觸發構建。例如，您可以指定只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。如果任何過濾器組評估為 true，則會觸發構建，當組中的所有過濾器評估為 true 時發生。當您建立篩選群組時，您可以指定這些項目：

一個事件

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

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**
>
> 如果您建立使用該merged事件類型的 GitLab webhook 篩選器群組，則必須在 webhook 設定中啟用該PULL_REQUEST_MERGED事件。

X-Event-Key 標頭值	事件類型
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED

對於PULL_REQUEST_MERGED，如果拉取請求與壓縮策略合併，並且拉取請求分支關閉，則原始提取請求提交將不再存在。在這種情況下，CODEBUILD_WEBHOOK_MERGE_COMMIT環境變量包含壓縮合併提交的標識符。

一個或多個可選的過濾器

使用規則表達式來指定篩選條件。對於觸發構建的事件，與其關聯的組中的每個過濾器都必須評估為 true。

ACTOR_ACCOUNT_ID（ACTOR_ID在控制台中）

當 GitLab 帳戶 ID 與規則運算式模式相符時，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`

當頭部引用匹配正則表達式模式（例如，`refs/heads/branch-name`和`refs/tags/tag-name`）時，webhook 事件觸發構建。`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`

當基本引用與正則表達式模式匹配時，webhook 事件會觸發構建。`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`

當更改文件的路徑與正則表達式模式匹配時，webhook 觸發構建。

`COMMIT_MESSAGE`

當頭提交消息與正則表達式模式匹配時，webhook 觸發構建。

> **Note**
>
> 您可以在存儲庫的 webhook 設置中找到 webhook 有效負載。 GitLab

主題

- 過濾 GitLab 網絡掛鉤事件（控制台）
- 篩選 GitLab 網路掛鉤事件 (SDK)
- 篩選 GitLab 網路掛鉤事件 ()AWS CloudFormation

過濾 GitLab 網絡掛鉤事件（控制台）

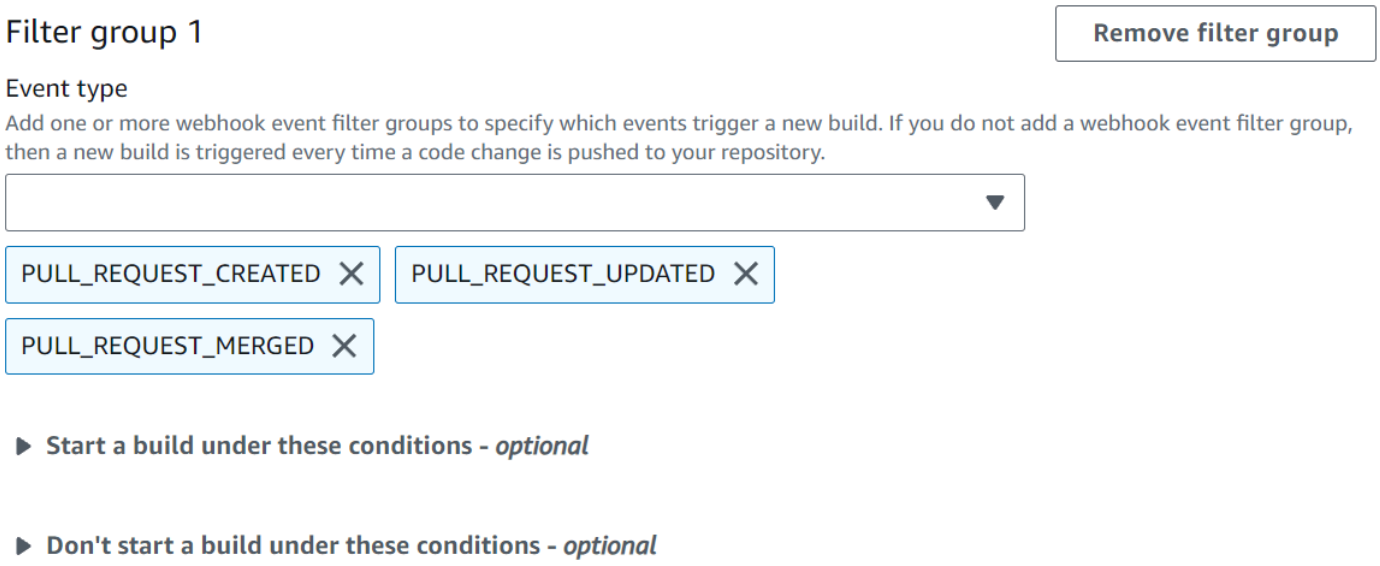
若要使用 AWS Management Console 來篩選網路掛接事件：

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WebhookFilter中的建立組建專案 (主控台)和。

在這個範例中，Webhook 篩選群組僅針對提取請求觸發組建：



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

HEAD_REF

Pattern

^refs/heads/branch1$

Remove

Filter 2

Type

BASE_REF

Pattern

^refs/heads/main

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.

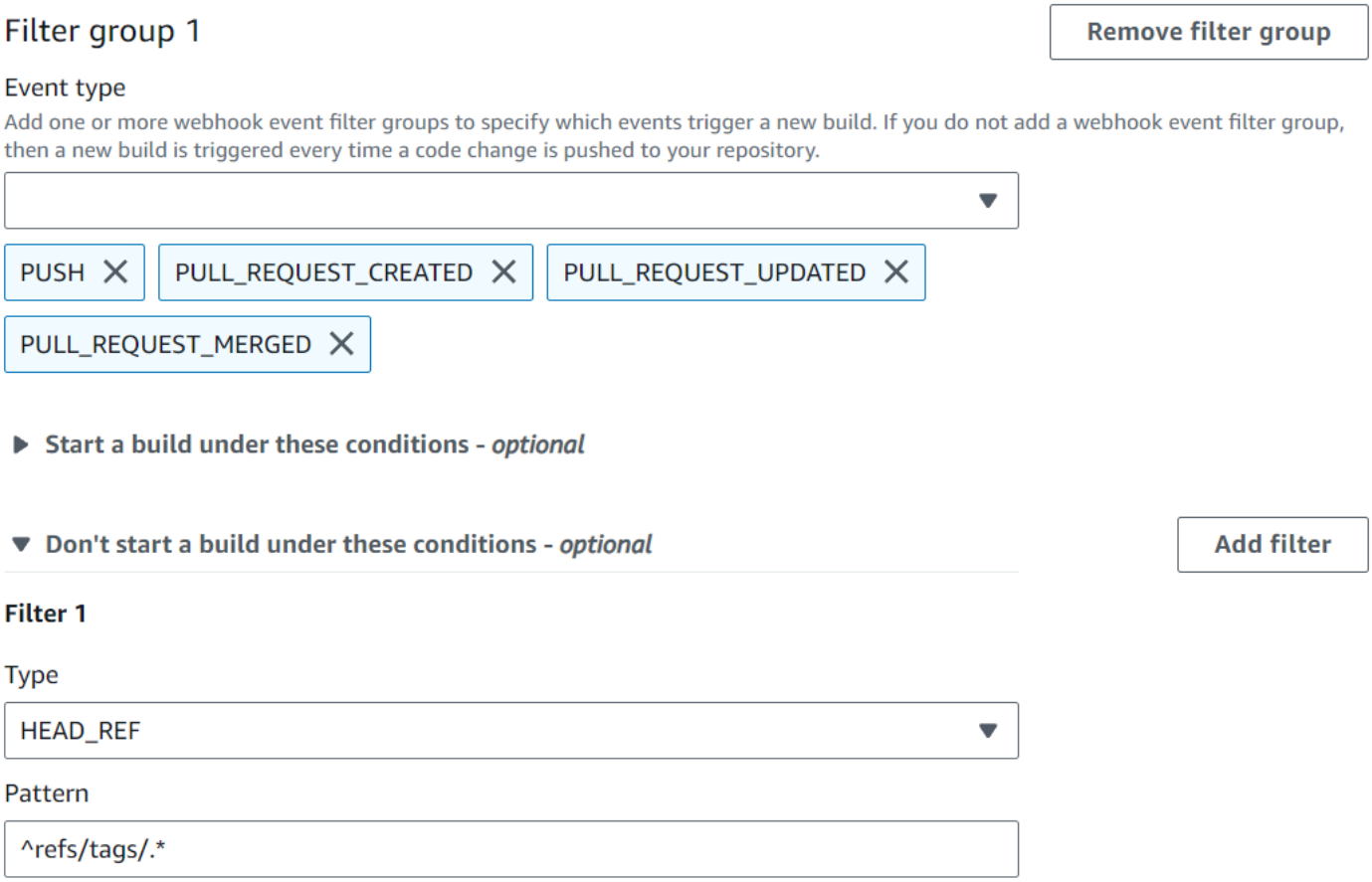
PUSH ✕

▼ **Start a build under these conditions - *optional***

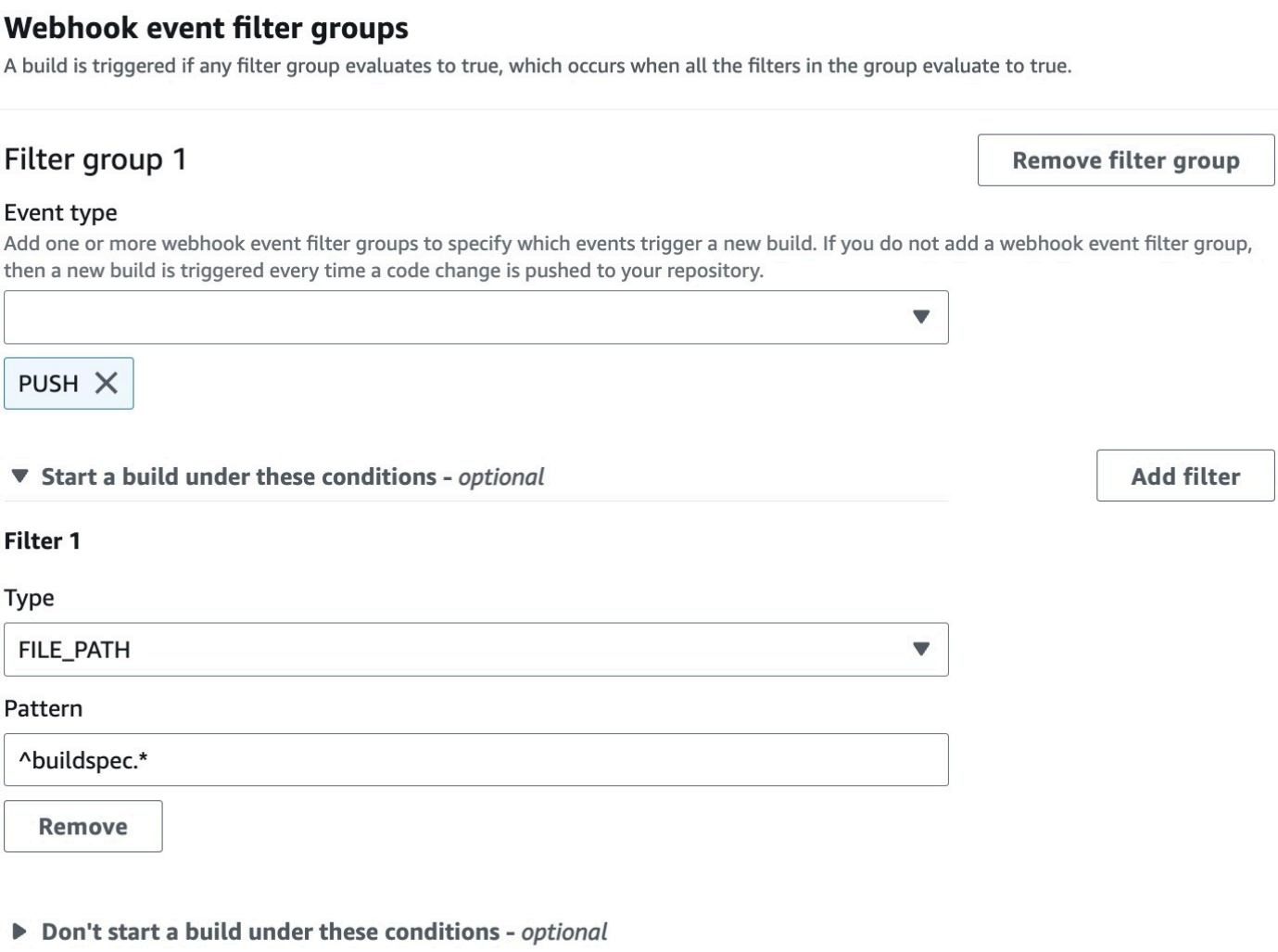
Add filter

Filter 1

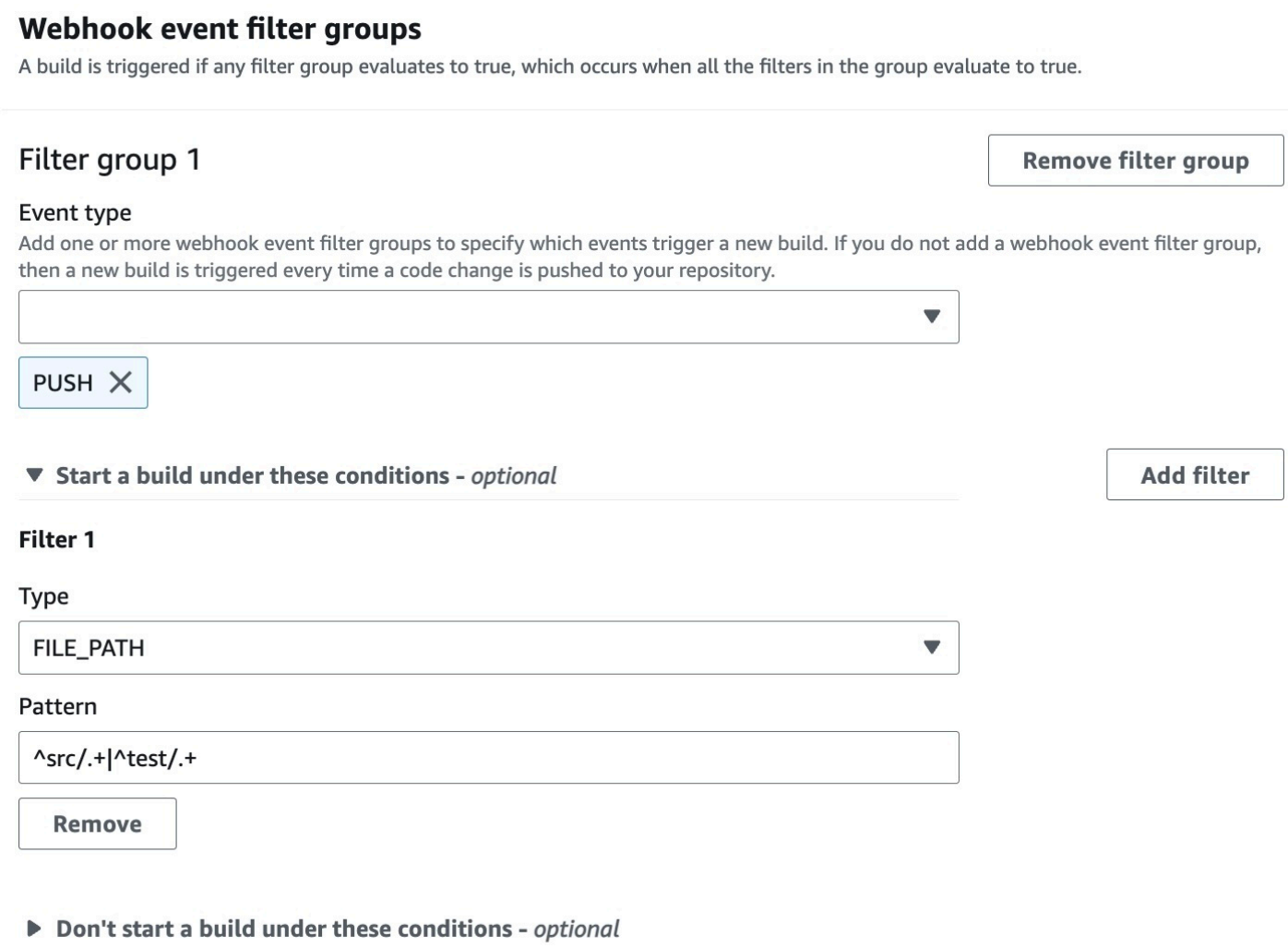
在這個範例中，Webhook 篩選群組針對所有請求 (標籤事件除外) 觸發組建。



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。

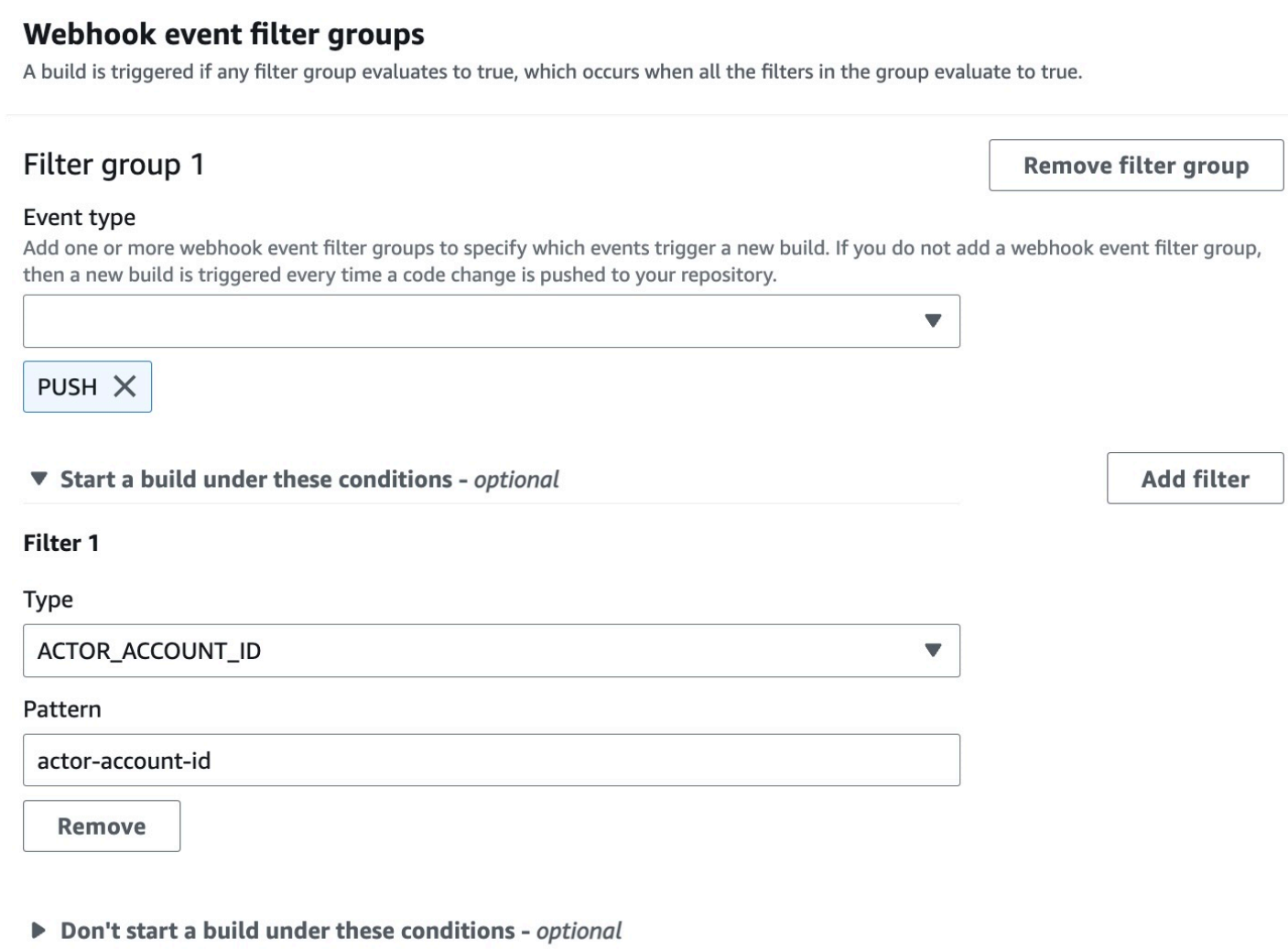


在此範例中，Webhook 篩選器群組只會在`src`或`test`資料夾中變更檔案時觸發組建。

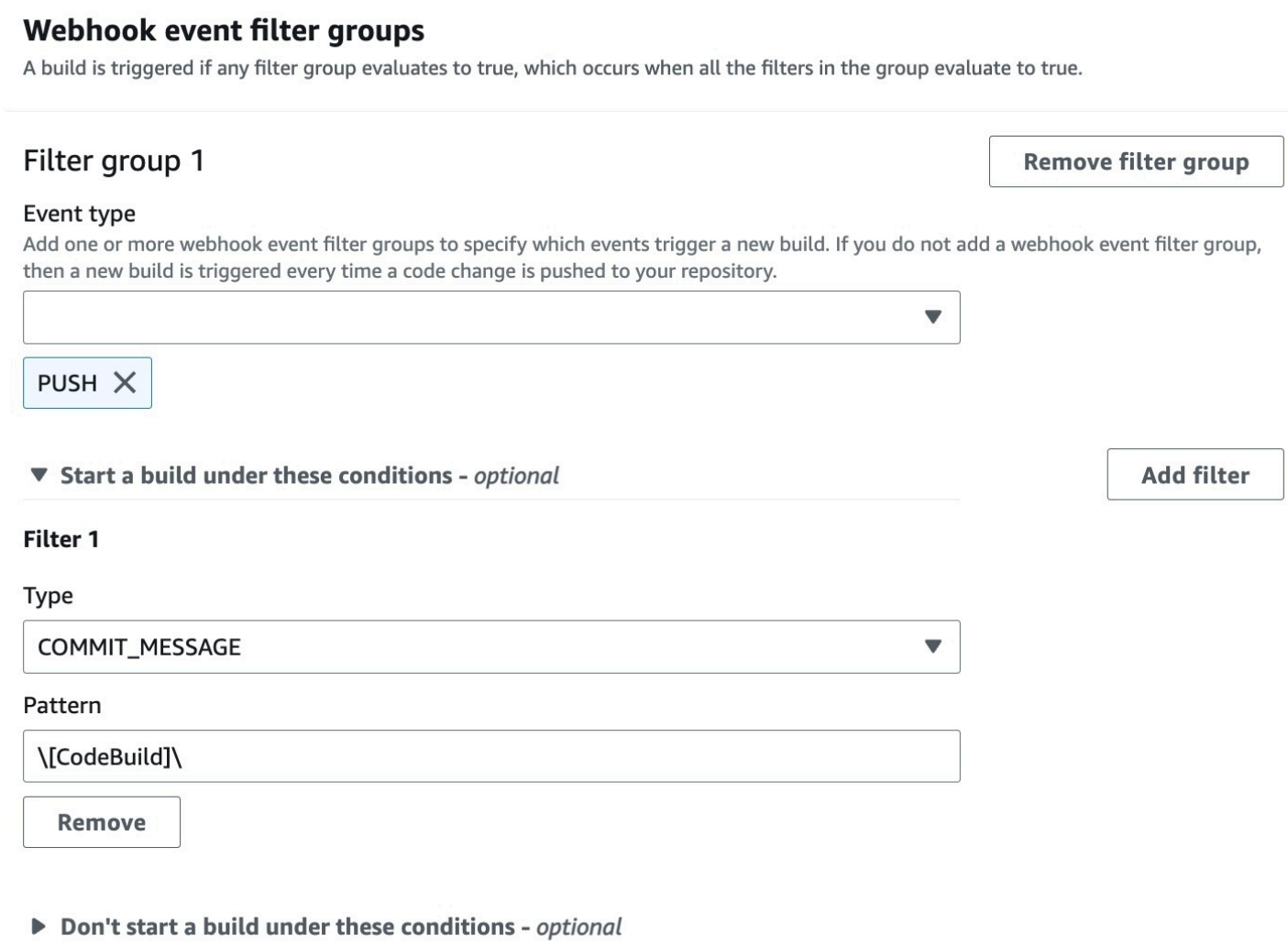


在此範例中，只有在沒有符合規則運`actor-account-id`算式之帳戶 ID 的 GitLab 使用者進行變更時，Webhook 篩選器群組才會觸發組建。

> **Note**
>
> 如需如何尋找 GitLab 帳戶 ID 的詳細資訊，請參閱 https://api.github.com/users/ *#####*，其中使用*###*是您的 GitLab 使用者名稱。



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。



篩選 GitLab 網路掛鉤事件 (SDK)

若要使用 AWS CodeBuild SDK 來篩選網路掛接事件，請在`CreateWebhook`或 `UpdateWebhook` API 方法的要求語法中使用`filterGroups`欄位。如需詳細資訊，請參閱 CodeBuild API 參考[WebhookFilter](#)中的。

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
    }
  ]
]
```

您可以建立篩選器，只有在具有帳戶 ID 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 的詳細資訊，請參閱 https://api.github.com/users/ *#####*，其中使用*###*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，篩選器群組會指定只有在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 網路掛鉤事件 ()AWS CloudFormation

若要使用 AWS CloudFormation 範本來篩選 webhook 事件，請使用 AWS CodeBuild 專案的`FilterGroups`屬性。在 AWS 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：

- 第一個過濾器組指定在具有 Git 引 GitLab 用名稱匹配正則表達式的分支上創建或更新提取請求，`^refs/heads/main$`由沒有帳戶 ID 的用戶`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。

```
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: \[CodeBuild\]
```

在 AWS CodeBuild 中變更建置專案的設定

您可以使用 AWS CodeBuild 主控台、AWS CLI 或 AWS 軟體開發套件來變更組建專案的設定。

如果您將測試報告新增至建置專案，請確定您的 IAM 角色有使用測試報告許可。

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 開發套件)

變更建置專案的設定 (主控台)

若要變更組建專案的設定，請執行下列步驟：

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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

- 項目配置
- 來源

- 環境
- 建置規格
- Batch 設定
- 成品
- 日誌

項目配置

在 [專案組態] 區段中，選擇 [編輯]。變更完成後，請選擇 [更新組態] 以儲存新的組態。

您可以修改下列屬性。

Description

輸入建置專案的選用描述，以協助其他使用者瞭解此專案的用途。

建立徽章

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，則不適用建立徽章。

啟用並行建置限制

如果您要限制此專案的並行建構數目，請執行下列步驟：

1. 選取「限制此專案可啟動的並行建構數目」。
2. 在並行建構限制中，輸入此專案允許的並行建構數目上限。此限制不得大於針對帳戶設定的並行建置限制。如果您嘗試輸入的數字大於帳號限制，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。

啟用公用組建存取權

若要讓公眾使用專案的建置結果，包括無法存取 AWS 帳戶的使用者，請選取 [啟用公用組建存取權限]，並確認您要公開組建結果。以下屬性用於公共構建項目：

公用建置服務角色

如果您想要為您 CodeBuild 建立新的服務角色，請選取 [新增服務角色]；如果您想要使用現有的服務角色，請選取 [現有的服務角色]。

公開建置服務角色可 CodeBuild 讀取 CloudWatch 日誌並下載專案組建的 Amazon S3 成品。這是為了使公眾可用項目的構建日誌和工件是必需的。

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 [啟用公用組建存取權]。

如需詳細資訊，請參閱 中的公共組建專案AWS CodeBuild。

> **Warning**
>
> 將項目的構建結果公開時，應牢記以下幾點：
>
> - 專案的所有組建結果、記錄檔和成品，包括專案為私有時執行的組建，都可供公開使用。
> - 所有構建日誌和成品都可供公眾使用。環境變量，源代碼和其他敏感信息可能已輸出到構建日誌和成品。您必須小心將哪些信息輸出到構建日誌。一些最佳做法是：
>   - 不要在環境變量中存儲敏感值，特別是 AWS 訪問密鑰 ID 和秘密訪問密鑰。我們建議您使用 Amazon EC2 Systems Manager 參數存放區或 AWS Secrets Manager 存放機密值。
>   - 請遵循使用網路掛鉤的最佳做法以限制哪些實體可以觸發構建，並且不要將 buildspec 存儲在項目本身中，以確保您的 webhook 盡可能安全。
> - 惡意使用者可使用公開組建來散佈惡意成品。我們建議專案管理員檢閱所有提取要求，以確認提取要求是否為合法變更。我們也建議您使用其總和檢查碼來驗證任何人工因素，以確保下載正確的成品。

其他資訊

在「標籤」中，輸入您希望支援 AWS 服務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] 以儲存新的組態。

您可以修改下列屬性：

來源提供者

選擇源代碼提供者類型。使用下列清單來選取適合您的來源提供者的選項：

> **Note**
>
> CodeBuild 不支援比特桶伺服器。

Amazon S3

儲存貯體

選擇包含原始程式碼之輸入值區的名稱。

S3 物件金鑰或 S3 資料夾

輸入 ZIP 檔案的名稱或包含原始程式碼之資料夾的路徑。輸入正斜線 (/) 以下載 S3 儲存貯體中的所有項目。

來源版本

輸入代表輸入檔案組建之物件的版本 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

CodeCommit

儲存庫

選擇您要使用的存放庫。

參考類型

選擇「分支」、「Git 標籤」或「提交 ID」來指定原始程式碼的版本。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

選擇此選項可建立歷程記錄截斷為指定數目的簡易複製。如果您想要完整複製，請選擇 Full (完整)。

Git 子模塊

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

儲存庫

選擇 [使用 OAuth Connec t] 或 [使用 Bitbucket 應用程式密碼連線]，然後依照指示進行連線 (或重新連線) 至 Bitbucket。

在您的帳戶中選擇一個公共存儲庫或存儲庫。

來源版本

輸入分支、提交 ID、標籤或參照，以及提交 ID。如需更多資訊，請參閱原始碼版本範例 AWS CodeBuild

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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 子模塊

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

針對狀態內容，請在 Bitbucket 認可狀態中輸入要用於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」，請在 Bitbucket 認可狀態中輸入要用於url參數的值。如需詳細資訊，請參閱 Bitbucket API 文件中的建置。

由 webhook 觸發的構建狀態始終報告給源提供程序。若要讓從主控台啟動的組建狀態，或向來源提供者報告 API 呼叫，您必須選取此設定。

如果您的項目的構建是由 webhook 觸發的，則必須將新的提交推送到存儲庫，以使對此設置的更改生效。

如果您想要在每次將程式碼變更推送至此儲存庫時建立原始程式碼，請在 [主 CodeBuild 要來源 webhook 事件] 中選取 [每次程式碼變更推送至此儲存庫時重建]。如需有關 Webhook 和篩選器群組的詳細資訊，請參閱比特桶網絡鉤事件。

GitHub

儲存庫

選擇 [使用 OAuth Connect] 或 [使用個 GitHub 人存取權杖連線]，然後依照指示進行連線 (或重新連線) GitHub 並授權存取 AWS CodeBuild權。

在您的帳戶中選擇一個公共存儲庫或存儲庫。

來源版本

輸入分支、提交 ID、標籤或參照，以及提交 ID。如需更多資訊，請參閱原始碼版本範例 AWS CodeBuild

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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 子模塊

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。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 確認狀態中`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提交狀態。

在「目標 URL」中，輸入 GitHub 確認狀態中要用於`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提交狀態。

由 webhook 觸發的構建狀態始終報告給源提供程序。若要讓從主控台啟動的組建狀態，或向來源提供者報告 API 呼叫，您必須選取此設定。

如果您的項目的構建是由 webhook 觸發的，則必須將新的提交推送到存儲庫，以使對此設置的更改生效。

如果您想要在每次將程式碼變更推送至此儲存庫時建立原始程式碼，請在 [主 CodeBuild 要來源 webhook 事件] 中選取 [每次程式碼變更推送至此儲存庫時重建]。如需有關 Webhook 和篩選器群組的詳細資訊，請參閱GitHub 網絡掛鉤事件。

GitHub Enterprise Server

GitHub 企業個人存取權杖

如需有GitHub 企業伺服器範例關如何將個人存取權杖複製到剪貼簿的資訊，請參閱。將字符貼入文字欄位，接著選擇 Save Token (儲存字符)。

> **Note**
>
> 您只需要輸入及儲存個人存取字符一次。 CodeBuild 在 future 的所有項目中使用此令牌。

來源版本

輸入提取請求、分支、提交 ID、標籤或參照，以及提交 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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 子模塊

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。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 確認狀態中`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提交狀態。

在「目標 URL」中，輸入 GitHub 確認狀態中要用於`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提交狀態。

由 webhook 觸發的構建狀態始終報告給源提供程序。若要讓從主控台啟動的組建狀態，或向來源提供者報告 API 呼叫，您必須選取此設定。

如果您的項目的構建是由 webhook 觸發的，則必須將新的提交推送到存儲庫，以使對此設置的更改生效。

不安全的 SSL

選取 [啟用不安全的 SSL] 以在連線至您的 GitHub 企業專案存放庫時忽略 SSL 警告。

如果您想要在每次將程式碼變更推送至此儲存庫時建立原始程式碼，請在 [主 CodeBuild 要來源 webhook 事件] 中選取 [每次程式碼變更推送至此儲存庫時重建]。如需有關 Webhook 和篩選器群組的詳細資訊，請參閱GitHub 網絡掛鉤事件。

GitLab

Connection (連線)

使用 Connect 連接您的 GitLab 帳戶 AWS CodeConnections，並使用連接將第三方存儲庫關聯為構建項目的源。

選擇預設連線或自訂連線。

預設連線會在所有專案中套用預設 GitLab 連線。自訂連線會套用覆寫帳戶預設設定的自訂 GitLab 連線。

預設連線

與您的帳戶相關聯的預設連線名稱。

如果您尚未建立與提供者的連線，請參閱以建立連線至 GitLab (主控台)取得指示。

自訂連線

選擇您要使用的自訂連線名稱。

如果您尚未建立與提供者的連線，請參閱以建立連線至 GitLab (主控台)取得指示。

儲存庫

選擇您要使用的存放庫。

來源版本

輸入提取請求 ID、分支、提交 ID、標籤或參照，以及提交 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

GitLab Self Managed

Connection (連線)

使用 Connect 連接您的 GitLab 帳戶 AWS CodeConnections，並使用連接將第三方存儲庫關聯為構建項目的源。

選擇預設連線或自訂連線。

預設連線會在所有專案中套用預設的「 GitLab 自我管理」連線。自訂連線會套用 GitLab 自訂「自我管理」連線，該連線會覆寫您帳戶的預設設定。

預設連線

與您的帳戶相關聯的預設連線名稱。

如果您尚未建立與提供者的連線，請參閱開發人員工具主控台使用者指南中的建立與 GitLab 自我管理的連線以取得指示。

自訂連線

選擇您要使用的自訂連線名稱。

如果您尚未建立與提供者的連線，請參閱開發人員工具主控台使用者指南中的建立與 GitLab 自我管理的連線以取得指示。

儲存庫

選擇您要使用的存放庫。

來源版本

輸入提取請求 ID、分支、提交 ID、標籤或參照，以及提交 ID。如需詳細資訊，請參閱 原始碼版本範例 AWS CodeBuild。

> **Note**
>
> 我們建議您選擇看起來不像提交 ID 的 Git 分支名稱，例如`811dd1ba1aba14473856cee38308caed7190c0d`或`5392f7`. 這有助於避免與實際提交的 Git 結帳衝突。

Git 克隆深度

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

如果您想要將組建的開始和完成狀態報告給來源提供者，請選取 [在組建開始和完成時向來源提供者報告組建狀態]。

若要能夠向來源提供者報告組建狀態，與來源提供者關聯的使用者必須具有存放庫的寫入存取權。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

環境

在「環境」段落中，選擇編輯。變更完成後，請選擇 [更新組態] 以儲存新的組態。

您可以修改下列屬性：

佈建模式

若要變更佈建模式，請選擇 [變更佈建模式]，然後執行下列其中一個動作：

- 若要使用由管理的隨選叢集 AWS CodeBuild，請選擇 [隨選]。透過隨選叢集，為您的組建 CodeBuild 提供運算。當構建完成時，機器將被銷毀。隨需叢集是全受管的，並包含自動擴充功能，可處理尖峰的需求。
- 若要使用由管理的保留容量叢集 AWS CodeBuild，請選擇保留容量，然後選取叢集名稱。使用預留容量叢集，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，可立即處理構建或測試，並減少構建持續時間。使用保留容量叢集時，您的機器會一直在執行，而且只要佈建機器就會繼續產生成本。

如需相關資訊，請參閱使用保留容量 AWS CodeBuild。

環境影像

若要變更組建映像檔，請選擇 [覆寫影像]，然後執行下列其中一個動作：

- 若要使用由管理的 Docker 映像檔 AWS CodeBuild，請選擇 [受管理的映像檔]，然後從 [作業系統]、[執行階段]、[映像] 和 [映像] 版本中進行選取。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)。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。如果您選擇 [其他登錄]，對於 [外部登錄 URL]，請使用格式在 Docker Hub 中輸入 Docker 映像的名稱和標記。*docker repository*/*docker image name*如果您選擇 Amazon ECR，請使用 Amazon ECR 存儲庫和 Amazon ECR 映像在您的帳戶中選擇碼頭映像。 AWS
- 若要使用私人 Docker 映像檔，請選擇 [自訂映像檔]。在「環境類型」中，選擇 ARM、Linux 、GPU 或視窗。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。認證必須由 Secrets Manager 建立。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

> **Note**
>
> CodeBuild 會覆寫自`ENTRYPOINT`訂泊塢視窗影像的。

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 [新增服務角色]。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**
>
> 使用主控台建立組建專案時，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 8 小時之間的值，如果建置未完成，則 CodeBuild 會停止建置。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 映像檔，或希望組建取得提升的權限，請選取 [啟用此旗標]。 僅當您打算使用此構建項目來構建 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。執行這項操作的一種方式是執行下列建置命令，以在建置規格的 `install` 階段中初始化 Docker 協助程式。如果您選擇由 CodeBuild Docker 支援提供的建置環境映像檔，請勿執行這些命令。

> **Note**
>
> 依預設，非 VPC 組建會啟用 Docker 精靈。如果您想使用 Docker 容器進行 VPC 構建，請參閱 Docker 文檔網站上的運行時特權和 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

如果您想 CodeBuild 要使用 VPC：

- 對於 VPC，請選擇使 CodeBuild 用的 VPC 識別碼。
- 對於 VPC 子網路，請選擇包含使用之資源的子網路。 CodeBuild
- 對於 VPC 安全性群組，請選擇 CodeBuild 用來允許存取 VPC 中資源的安全性群組。

如需詳細資訊，請參閱 搭 AWS CodeBuild 配 Amazon Virtual Private Cloud 使用。

運算

選擇其中一個可用選項。

環境變數

輸入名稱和值，然後選擇要使用之組建之每個環境變數的類型。

> **Note**
>
> CodeBuild 自動設定「 AWS 區域」的環境變數。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

控制台和 AWS CLI 用戶可以看到環境變量。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)。

我們建議您將具有敏感值的環境變數 (例如 AWS 存取金鑰 ID、 AWS 秘密存取金鑰或密碼) 存放在 Amazon EC2 Systems Manager 參數存放區或 AWS Secrets Manager。

如果您使用 Amazon EC2 Systems Manager 參數存放區，請選擇「參數」做為「類型」。在「名稱」中，輸入 CodeBuild 要參考的識別碼。在值中，輸入存放在 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)。針對名稱，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，您可以指定帳戶中金 AWS KMS 鑰的 ARN。 Amazon EC2 Systems Manager 使用此金鑰在儲存期間加密參數的值，並在擷取期間將其解密。) 如果您使用 CodeBuild 主控台建立參數，則主控台會在儲存參數名稱時以其開始。`/CodeBuild/`如需詳細資訊，請參閱 Amazon EC2 Systems Manager 使用指南中的 Systems Manager 參數存放區和系統管理員參數存放主控台逐步解說。

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，則建置專案的服務角色必須允許該`ssm:GetParameters`動作。如果您先前選擇 [新增服務角色]，請將此動作 CodeBuild 包含在組建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照存放在 Amazon EC2 Systems Manager 參數存放區中，參數名稱不是開頭為的參數`/CodeBuild/`，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以的參數名稱`/CodeBuild/`。這是因為該服務角色僅允許存取開頭為 `/CodeBuild/` 的參數名稱。

如果選擇 [新增服務角色]，服務角色會包含解密 Amazon EC2 Systems Manager 參數存放區中`/CodeBuild/`命名空間下所有參數的權限。

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，請選擇 Secrets Manager 做為類型。在「名稱」中，輸入 CodeBuild 要參考的識別碼。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」，建議您以 /CodeBuild/ (例如/CodeBuild/dockerLoginPassword) 開頭的名稱來儲存密碼。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。
> 如果您的組建專案參照儲存在 Secrets Manager 中的密碼，則組建專案的服務角色必須允許該secretsmanager:GetSecretValue動作。如果您先前選擇 [新增服務角色]，請將此動作 CodeBuild 包含在組建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。
> 如果您的組建專案參照儲存在 Secrets Manager 中的密碼/CodeBuild/，而且您選擇了新服務角色，則必須更新服務角色，以允許存取開頭不是以的密碼名稱/CodeBuild/。這是因為服務角色只允許存取以開頭的密碼名稱/CodeBuild/。
> 如果您選擇 [新增服務角色]，服務角色會包含解密 Secrets Manager 中/CodeBuild/命名空間下所有密碼的權限。

建置規格

在「建置規格」區段中，選擇「編輯」。變更完成後，請選擇 [更新組態] 以儲存新的組態。

您可以修改下列屬性：

建置規格

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)。依預設，會 CodeBuild尋找在原始程式碼根目錄buildspec.yml中名為的檔案。如果您的 buildspec 檔案使用不同的名稱或位置，請在 Buildspec 名稱 (例如，或. buildspec-two.yml configuration/buildspec.yml 如果 buildspec 檔案位於 S3 儲存貯體中，則該檔案必須與您的建置專案位於相同的 AWS 區域。 使用其 ARN 指定構建規格文件 (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，或者您有一長串的build階段指令，請將buildspec.yml檔案新增至原始程式碼根目錄，將命令新增至檔案，然後在原始程式碼根目錄中選擇 [使用 buildspec.yml]。

如需更多資訊，請參閱Buildspec 參考。

Batch 設定

在「Batch 設定」區段中，選擇 「編輯」。變更完成後，請選擇 [更新組態] 以儲存新的組態。如需詳細資訊，請參閱 Batch 建置 AWS CodeBuild。

您可以修改下列屬性：

Batch 服務角色

提供批次組建的服務角色。

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。

Batch 組建會在批次設定中引入新的資訊安全角色。此新角色是必要的，因為 CodeBuild 必須能夠代表您呼叫StartBuildStopBuild、和RetryBuild動作，才能在批次中執行組建。客戶應該使用新角色，而不是在組建中使用的角色，原因有兩個：

- 賦予構建角色 StartBuildStopBuild，和RetryBuild權限將允許單個構建通過 buildspec 啟動更多構建。
- CodeBuild 批次組建會提供限制，限制可用於批次中組建的組建數量和運算類型。如果組建角色具有這些權限，則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。選取所有適用的項目。

批次允許的最大組建

輸入批次中允許的最大建構數目。如果批次超過此限制，批次就會失敗。

Batch 逾時

輸入批次建置完成的時間上限。

合併成品

選取「將批次中的所有人工因素合併至單一位置」，以將批次中的所有人工因素合併至單一位置。

Batch 報告模式

為批次組建選取所需的建置狀態報告模式。

> **Note**
>
> 只有當專案來源為 Bitbucket 或 GitHub企業 GitHub，並且在 [來源] 下選取 [組建開始和結束] 時，才能使用此欄位。

彙總組建

選取此選項可將批次中所有建構的狀態合併為單一狀態報表。

個別組建

選取此選項可分別報告批次中所有組建的建構狀態。

成品

在「成品」區段中，選擇 「編輯」。變更完成後，請選擇 [更新組態] 以儲存新的組態。

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，或者想要將 Docker 映像推送到 Amazon ECR 存放庫，則可能需要執行此操作。
- 若要將組建輸出存放在 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

執行以下任意一項：

- 若要使用帳戶中的 AWS 受管金鑰 Amazon S3 加密建置輸出成品，請將加密金鑰保留空白。此為預設值。
- 若要使用客戶受管金鑰來加密組建輸出成品，請在加密金鑰中，輸入客戶管理金鑰的 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。
  - (選擇性) 對於快取路徑前置詞，請輸入 Amazon S3 路徑前置詞。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「在 AWS CodeBuild 中建立快取」。

日誌

在「記錄檔」區段中，選擇 「編輯」。變更完成後，請選擇 [更新組態] 以儲存新的組態。

您可以修改下列屬性：

選擇您要建立的記錄檔。您可以建立 Amazon CloudWatch 日誌、Amazon S3 日誌，或兩者兼而有之。

CloudWatch

如果你想要 Amazon CloudWatch 日誌日誌：

CloudWatch 日誌

選取CloudWatch 記錄檔。

Group name (群組名稱)

輸入您的 Amazon CloudWatch 日誌日誌群組的名稱。

串流名稱

輸入您的 Amazon CloudWatch 日誌日誌流名稱。

S3

如果你想要 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。

路徑前綴

輸入記錄檔的前置字元。

停用 S3 日誌加密

如果您不想要加密 S3 日誌，請選取此選項。

變更建置專案的設定 (AWS CLI)

如需搭配使用 AWS CLI 與 AWS CodeBuild 的資訊，請參閱命令列參考。

若要更新CodeBuild項目與AWS CLI時，您可以使用更新的屬性建立 JSON 檔案，並將該檔案傳遞至update-project指令。更新檔案中未包含的任何性質均保持不變。

在更新 JSON 檔案中，只有name屬性和修改的屬性是必需的。該name性質可識別要修改的專案。對於任何修改的結構，也必須包括這些結構的必要參數。例如，若要修改專案的環境，environment/type和environment/computeType屬性是必需的。以下是更新環境影像的範例：

```
{
  "name": "<project-name>",
  "environment": {
    "type": "LINUX_CONTAINER",
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:4.0"
  }
}
```

如果您需要取得專案的目前性質值，請使用batch-get-projects指令以取得您正在修改之專案的目前性質，並將輸出寫入檔案。

```
aws codebuild batch-get-projects --names "<project-name>" > project-info.json
```

該####文件包含一個項目數組，因此不能直接用於更新項目。但是，您可以從中複製要修改的性質#####檔案並將其貼到您的更新檔案中，做為您要修改之性質的基準線。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)，然後儲存您的結果。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，以識別遺失的參數。例如，這是顯示的錯誤訊息environment/type缺少參數：

```
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 開發套件)

如需搭配使用 AWS CodeBuild 與 AWS 開發套件的資訊，請參閱 AWS 開發套件和工具參考。

在 AWS CodeBuild 中刪除建置專案

您可以使用 CodeBuild 主控台、AWS CLI, 或AWS開發套件，在 CodeBuild 中刪除組建專案。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)
- 刪除建置專案 (AWS 開發套件)

刪除建置專案 (主控台)

1. 開啟AWS CodeBuild主控台，位於https://console.aws.amazon.com/codesuite/codebuild/home。
2. 在導覽窗格中，選擇 Build projects (建置專案)。

3. 執行下列任一步驟：

   - 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：

   - *##*：必要的字串。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。

如需搭配使用 AWS CLI 與 AWS CodeBuild 的詳細資訊，請參閱命令列參考。

刪除建置專案 (AWS 開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

使用共用專案

專案共享功能可讓專案擁有者共用其AWS CodeBuild項目與其他AWS帳户或用户。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。

內容

- 相關 服務
- 共用專案
- 取消共用已共用的專案
- 識別共用的專案
- 共用的專案許可

共用專案的先決條件

若要共用專案，您的 AWS 帳戶必須擁有該專案。您無法將已分享給您的專案再分享出去。

存取與您共用之專案的先決條件

若要存取共用專案，消費者的 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 使用以身分識別為基礎的原則 AWS CodeBuild。

相關 服務

專案共用功能與 AWS Resource Access Manager (AWS RAM) 服務整合。該服務可讓您將您的 AWS 資源分享給任何 AWS 帳戶或透過 AWS Organizations 來共用。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。消費者可以是個別的 AWS 帳戶，或 AWS Organizations 中的組織單位或 AWS Organizations 中的整個組織。

如需詳細資訊，請參閱《AWS RAM 使用者指南》。

共用專案

消費者可以使用AWS CLI和AWS CodeBuild主控台來檢視您已共用的專案和建置。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 來共用自己擁有的專案。

共用您擁有的專案 (CodeBuild 主控台)

1. 開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
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「」建立資源共享中的AWS RAM使用者指南。

共用您擁有的專案 (AWS RAM 主控台)

請參閱建立資源共享中的AWS RAM使用者指南。

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。

```
{
```

```
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Principal":{
      "AWS":"<consumer-aws-account-id-or-user>"
    },
    "Action":[
      "codebuild:BatchGetProjects",
      "codebuild:BatchGetBuilds",
      "codebuild:ListBuildsForProject"],
    "Resource":"<arn-of-project-to-share>"
  }]
}
```

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以存取AWS帳戶已經確定。

```
{
  "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-紮實命令。

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. 獲取AWS RAM資源共享 ARN。

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

這會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>"
    }
  ]
}
```

從響應中，複製*<resource-share-arn>*值以便在下一個步驟使用。

5. 執行AWS RAM 促銷資源共享創建自策略命令。

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-share-arn>
```

取消共用已共用的專案

只有其擁有者可以存取已取消共享的專案 (包括其建置)。如果您取消共享專案，您之前與之共用的任何 AWS 帳戶或使用者無法存取專案或其建置。

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 執行此作業。

取消共享您擁有的共用專案 (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。

取消共享您擁有的專案 (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
```

識別共用的專案

擁有者和消費者可以使用 AWS CLI 識別已共用的專案。

識別與 AWS 帳戶或使用者 (AWS CLI) 共用的專案

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。

在 AWS CodeBuild 中標記專案

標籤是一種自訂屬性標籤，可由您或 AWS 指派給 AWS 資源。每個 AWS 標籤都有兩個部分：

- 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 或 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，請參閱標籤。

標籤可協助您識別和整理 AWS 資源。許多 AWS 服務支援標記，因此您可以對來自不同服務的資源指派相同的標籤，指示資源是相關的。例如，您可以將指派給 S3 儲存貯體的相同標籤，指派給 CodeBuild 專案。如需有關在用標籤的詳細資，請參閱。

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 API 或 AWS 軟體開發套件來新增、管理和移除專案的標籤。此外，若要使用標籤來識別、整理和追蹤您的專案，您可以在 IAM 政策中使用標籤，以協助控制誰可以檢視您的專案並與其互動。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 AWS CodeBuild 資源的存取。

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

新增標籤到專案可協助您識別和整理您的 AWS 資源和管理存取權。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，請參閱標籤。擁有標籤後，您可以建立 IAM 政策，以根據這些標籤管理專案的存取權限。您可以使用 CodeBuild 主控台或 AWS CLI，將標籤新增至專案。

> **Important**
>
> 在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 (例如建置專案) 的 IAM 政策。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 AWS CodeBuild 資源的存取。

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)。

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到 CodeBuild 專案。

1. 開啟主CodeBuild控台，網址為 https://console.aws.amazon.com/codebuild/。
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 Settings (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。

如果成功，此命令不會傳回任何內容。

檢視專案的標籤

標籤可協助您辨識和整理您的 AWS 資源和管理存取權。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，請參閱使用標籤來控制對 AWS CodeBuild 資源的存取。

檢視專案的標籤 (主控台)

您可以使用 CodeBuild 主控台以檢視與 CodeBuild 專案相關聯的標籤。

1. 開啟主CodeBuild控台，網址為 https://console.aws.amazon.com/codebuild/。
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 Settings (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。您也可以變更金鑰名稱，等於移除目前的標籤，並新增一個不同的新的名稱和相同的值作為其他金鑰。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 (金鑰) 或值之前，請務必檢閱任何可能使用標籤金鑰或值的 IAM 政策來控制資源 (例如建置專案) 的存取權。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 AWS CodeBuild 資源的存取。

編輯專案的標籤 (主控台)

您可以使用 CodeBuild 主控台以編輯與 CodeBuild 專案相關聯的標籤。

1. 開啟主CodeBuild控台，網址為 https://console.aws.amazon.com/codebuild/。
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 Settings (設定)。選擇 Build project tags (組建專案標籤)。
4. 選擇 編輯 。
5. 執行下列任意一項：
   - 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。移除標籤不會從其他 AWS 資源刪除與該標籤相關聯的標籤。

> **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 政策來控制對資源 (例如建置專案) 的存取。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 AWS CodeBuild 資源的存取。

從專案移除標籤 (主控台)

您可以使用 CodeBuild 主控台移除標籤和 CodeBuild 專案之間的關聯。

1. 開啟主CodeBuild控台，網址為 https://console.aws.amazon.com/codebuild/。
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 Settings (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。

Batch 建置 AWS CodeBuild

您可以使 AWS CodeBuild 用批次建置來執行專案的並行與協調建置。

主題

安全性角色

Batch 組建會在批次設定中引入新的資訊安全角色。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，才能在批次中執行組建。客戶應該使用新角色，而不是在組建中使用相同的角色，原因有兩個：

- 賦予構建角色 `StartBuild`、`StopBuild`，和`RetryBuild`權限將允許單個構建通過 buildspec 啟動更多構建。
- CodeBuild 批次組建會提供限制，限制可用於批次中組建的組建數量和運算類型。如果組建角色具有這些權限，則組建本身可能會略過這些限制。

Batch 建置類型

CodeBuild 支援下列批次建置類型：

Batch 建置類型

構建圖

組建圖形會定義一組工作，這些工作與批次中的其他工作有相依性。

下列範例會定義建立相依性鏈的建置圖形。

```
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
```

在此範例中：

- build1首先運行，因為它沒有依賴關係。
- build2具有依賴關係build1，因此在build1完成後build2運行。
- build3具有依賴關係build2，因此在build2完成後build3運行。

如需建置圖形 Buildspec 語法的詳細資訊，請參閱。batch/build-graph

組建清單

組建清單會定義一些並行執 parallel 的工作。

下列範例會定義組建清單。build1和build2組建將 parallel 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
```

如需組建清單 Buildspec 語法的詳細資訊，請參閱。[batch/build-list]

建立矩陣

構建矩陣定義具有並行運 parallel 的不同配置的任務。 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組建矩陣。

```
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，建 CodeBuild 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

每個構建將具有以下設置：

- `ignore-failure`設定為 `false`
- `env/type`設定為 `LINUX_CONTAINER`
- `env/image`設定為 `aws/codebuild/amazonlinux2-x86_64-standard:4.0`
- `env/privileged-mode`設定為 `true`

這些組建並行執 parallel。

如需建置矩陣 Buildspec 語法的詳細資訊，請參閱。`batch/build-matrix`

Batch 報告模式

如果您專案的來源提供者是 Bitbucket 或 GitHub Enterprise GitHub，且您的專案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給來源提供者的方式。您可以選擇將狀態作為批次的單一彙總狀態報表傳送，或者分別報告批次中每個組建的狀態。

如需詳細資訊，請參閱下列主題：

- Batch 設定 (建立)
- Batch 設定 (更新)

其他資訊

如需詳細資訊，請參閱下列主題：

- Batch 量生成構建規範參考
- Batch 設定
- 執行批次建置 (AWS CLI)
- 在中停止批次組建AWS CodeBuild

GitHub 動作亞軍 AWS CodeBuild

「GitHub 動作」 是專為與工作 GitHub 流程搭配使用而開發的動作。如需有關 GitHub 動作的詳細資訊，請參閱GitHub 動作文件。

有兩種方法可搭配使用 GitHub 「動作」 CodeBuild：

- 您可以將專案設定為在 CodeBuild 容器中設定自我託管的 GitHub Actions 執行程式，以處理動作工 GitHub 作流程作業。
- 您可以使用 CodeBuild管理的動作執行器在其中 CodeBuild執行 GitHub 動作。

您可以選擇在 CodeBuild中設定自託管的 GitHub 動作執行器。這涉及使用您的 CodeBuild 項目設置 webhook，並更新 GitHub 操作工作流程 YAML 以使用託管在機器上的自託管運行器。 CodeBuild 這可讓您的 GitHub 「動作」工作流程作業取得原生整合 AWS。

您也可以選擇使用 CodeBuild-managed 動作執行器在其中 CodeBuild執行 GitHub 動作。這涉及`steps`到使用 GitHub 操作語法添加到您的 buildspec，該語法在命令的單獨階段運行。 CodeBuild 這可讓您的 GitHub 動作與 CodeBuild 功能整合，例如相依性快取和批次建置。

主題

- 在中設定自託管的 GitHub 動作跑步者 AWS CodeBuild
- 在構建 GitHub 規範中使用操作語法 AWS CodeBuild

在中設定自託管的 GitHub 動作跑步者 AWS CodeBuild

您可以將專案設定為在 CodeBuild 容器中設定自我託管的 GitHub Actions 執行程式，以處理動作工 GitHub 作流程作業。這可以通過使用您的 CodeBuild 項目設置一個 webhook，並更新 GitHub 操作工作流 YAML 以使用託管在機器上的自託管運行器來完成。 CodeBuild 若要取得更多資訊，請參閱〈關於自託管料道〉

將 CodeBuild 專案設定為執行 GitHub 動作工作的高階步驟如下：

1. 如果您尚未這樣做，請創建個人訪問令牌或與 OAuth 應用程序連接以將項目連接到 GitHub。
2. 導航到 CodeBuild 控制台並使用 webhook 創建一個 CodeBuild 項目並設置您的 webhook 過濾器。
3. 在中更新 YAML 中的 GitHub 動作工作流程， GitHub 以設定您的組建環境。

如需更詳細的程序，請參閱自學課程：設定 CodeBuild 自我託管的 GitHub 動作亞軍。

此功能可讓您的 GitHub 動作工作流程任務與原生整合 AWS，透過 IAM、 AWS Secrets Manager 整合和 Amazon VPC 等功能提供安全性和便利性。 AWS CloudTrail您可以存取最新的執行個體類型，包括 ARM 型執行個體。

主題

- 自學課程：設定 CodeBuild 自我託管的 GitHub 動作亞軍

- 關於 CodeBuild主體 GitHub 動作流道

自學課程：設定 CodeBuild 自我託管的 GitHub 動作亞軍

本教學課程說明如何將 CodeBuild 專案設定為執行 GitHub 動作工作。

必要條件

若要完成此教學課程，您必須先：

- 與 OAuth 應用程序 Connect 或創建個人訪問令牌。如果您想與 OAuth 應用程序連接，則必須使用 CodeBuild 控制台來執行此操作。如果您想創建個人訪問令牌，則可以使用 CodeBuild 控制台或使用 ImportSourceCredentials API。如需更多指示，請參閱GitHub 和 GitHub 企業服務器訪問令牌。
- Connect CodeBuild 到您的 GitHub 帳戶。若要這麼做，您可以執行下列其中一項作業：
  - 您可以在主控台中新增 GitHub 為來源提供者。您可以與 OAuth 應用程序或個人訪問令牌連接。如需說明，請參閱Connect GitHub 訪問令牌（控制台）。
  - 您可以通過 ImportSourceCredentials API 導入 GitHub 憑據。這只能使用個人訪問令牌來完成。如果您使用 OAuth 應用程式進行連線，則必須改為使用主控台進行連線。如需說明，請參閱 Connect GitHub 用存取權杖 (CLI) 連線。

> **Note**
>
> 只有在您尚未連接到您的帳戶時，才需要執 GitHub 行此操作。

第 1 步：使用網絡掛鉤創建 CodeBuild 項目

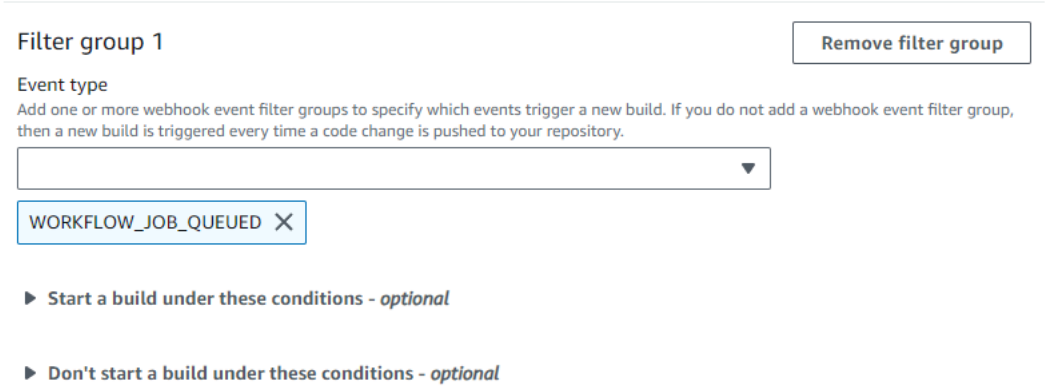
在此步驟中，您將使用 webhook 創建一個 CodeBuild 項目，並在 GitHub 控制台中查看它。

若要使用網路掛接建立 CodeBuild 專案

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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：
     - 對於來源提供者，請選擇GitHub。
     - 對於存儲庫，選擇我的 GitHub 帳戶中的存儲庫。

- 針對 Repository URL (儲存庫 URL)，輸入 **https://github.com/*user-name*/*repository-name***。
- 在主要來源網路掛鉤事件中：
  - 對於 Webhook-選擇性，請在每次將程式碼變更推送至此儲存庫時選取重建。
  - 對於事件類型，選取工作流程 _ 工作佇列。啟用此選項後，組建只會由 GitHub 「動作」工作流程作業事件觸發。

> **Note**
>
> CodeBuild 只有在網路勾點具有包含 WORKFLOW_JO B_QUEUE 事件篩選器的篩選器群組時，才會處理 GitHub 「動作」工作流程作業事件。



- 在 Environment (環境) 中：
  - 選擇支援的環境映像檔並運算。請注意，您可以選擇在 GitHub 動作工作流程 YAML 中使用標籤覆寫映像檔和執行個體設定。如需更多資訊，請參閱步驟 2：更新 GitHub 動作工作流程 YAML
- 在 Buildspec (建置規格) 中：
  - 請注意，您的構建規格將被忽略。而是 CodeBuild 將取代它，以使用將設置自我主體料道的指令。此專案的主要職責是在 CodeBuild 中設定自我託管的執行者，以執行 GitHub 「動作」工作流程工作流程作業。

3. 繼續使用預設值，然後選擇 [建立組建專案]。
4. 在開啟 GitHub 主控台`https://github.com/`*`user-name`*`/`*`repository-name`*`/settings/hooks`以確認 Webhook 已建立，且已啟用可傳遞工作流程作業事件。

步驟 2：更新 GitHub 動作工作流程 YAML

在此步驟中，您將更新中的 GitHub 動作工作流程 YAML 檔案，GitHub以設定您的組建環境，並在中使用 GitHub 動作自託管執行程式。 CodeBuild若要取得更多資訊，請參閱〈將標示與自主體料道一起使

更新您的 GitHub 動作工作流程 YAML

瀏覽GitHub並更新 GitHub 動作工作流程 YAML 中的runs-on設定，以設定您的組建環境。若要這麼做，您可以執行下列其中一項作業：

- 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來執行運算、映像檔、映像檔版本和執行個體大小。需要專案名稱，才能將 GitHub 「動作」工作的 AWS相關設定連結至特定 CodeBuild 專案。藉由在 YAML 中加入專案名稱， CodeBuild 就可以叫用具有正確專案設定的工作。藉由提供執行 ID， CodeBuild 將您的組建對應至特定的工作流程執行，並在取消工作流程執行時停止建置。如需詳細資訊，請參閱github前後關聯。

  ```
  runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
  ```

  > **Note**
  >
  > 請確定您<project-name>的名稱與您在上一個步驟中建立的專案名稱相符。如果它不匹配，CodeBuild 將不會處理 webhook，並且 GitHub 操作工作流程可能會掛起。

  下列是 YAML GitHub 動作工作流程的範例：

  ```
  name: Hello World
  on: [push]
  jobs:
    Hello-World-Job:
      runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      steps:
        - run: echo "Hello World!"
  ```

- 您也可以覆寫標籤中的映像檔和運算類型。這將覆蓋項目上的環境設置。若要覆寫 Amazon EC2 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 }}-<image>-<image-version>-<instance-size>
  ```

若要覆寫 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>-<runtime-version>-<instance-size>
```

下列是 YAML GitHub 動作工作流程的範例：

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}-
arm-3.0-small
    steps:
      - run: echo "Hello World!"
```

> **Note**
>
> 如果在 CodeBuild 環境中無法使用 GitHub-hosted runner 提供的相依性，您可以在工作流程執行中使用 GitHub 動作來安裝相依性。例如，您可以使用setup-python動作為建置環境安裝 Python。

支援的運算映像

在標籤中，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 運算映像檔：

映像	影像版本	執行個體大小	平台	映像識別符	定義
linux	4.0	small medium large xlarge 2xlarge	Amazon Linux 2	aws/codebuild/amazonlinux2-x86_64-standard:4.0	AL2/標準/4.0

映像	影像版本	執行個體大小	平台	映像識別符	定義
linux	5.0	gpu_small gpu_large	Amazon Linux 2023	aws/codebuild/amazonlinux2-x86_64-standard:5.0	AL2/標準/5.0
arm	2.0	small large	Amazon Linux 2	aws/codebuild/amazonlinux2-aarch64-standard:2.0	AL2/aarch64/標準/2.0
arm	3.0		Amazon Linux 2023	aws/codebuild/amazonlinux2-aarch64-standard:3.0	AL2/aarch64/標準/3.0
ubuntu	5.0	small medium	Ubuntu 20.04	aws/codebuild/standard:5.0	支持/標準/5.0
ubuntu	6.0	large xlarge 2xlarge gpu_small gpu_large	Ubuntu	aws/codebuild/standard:6.0	支持/標準/6.0
ubuntu	7.0		Ubuntu	aws/codebuild/standard:7.0	支持/標準 /7.0

映像	影像版本	執行個體大小	平台	映像識別符	定義
windows	1.0	medium large	視窗伺服器核心	aws/codebuild/windows-base:2019-1.0	N/A
windows	2.0		視窗伺服器核心	aws/codebuild/windows-base:2019-2.0	N/A
windows	3.0		視窗伺服器核心	aws/codebuild/windows-base:2019-3.0	N/A

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，請參閱使用[在中使用 AWS Lambda 計算 AWS CodeBuild](). CodeBuild 支援下列 Lambda 運算映像檔：

環境類型	執行時間版本	執行個體大小			
linux-lambda	dotnet6 go1.21 corretto11 corretto17 corretto21 nodejs18 nodejs20	1GB 2GB 4GB 8GB 10GB			
arm-lambda					

環境類型	執行時間版本	執行個體大小			
	python3.11 python3.12 ruby3.2				

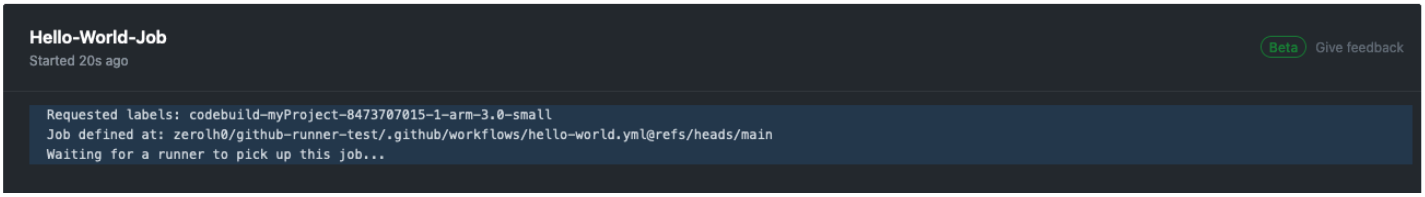
如需詳細資訊，請參閱 建置環境運算模式和類型 及 碼頭圖片提供 CodeBuild。

步驟 3：查看結果

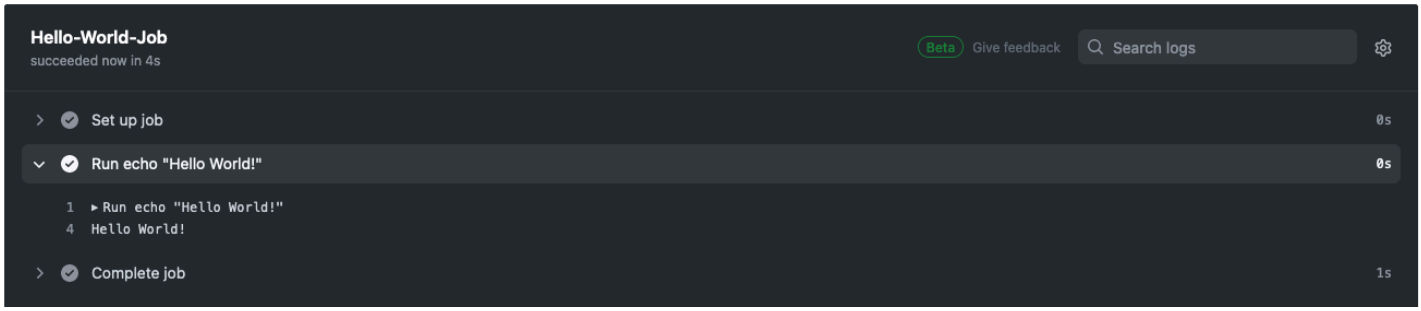
每當 GitHub 「動作」工作流程執行發生時，都 CodeBuild 會透過 webhook 接收工作流程工作事件。對於工作流程中的每個工作， CodeBuild 啟動組建以執行暫時的動作執行程序 GitHub 。流道負責執行單一工作流程工作。工作完成後，流道和相關聯的構建過程將立即終止。

若要檢視您的 Job 流程工作記錄，請瀏覽至中 GitHub的存放庫，選擇 [動作]，選擇您想要的工作流程，然後選擇要檢閱記錄的特定工作。

當工作正在等待自託管的跑道中拾取時，您可以在 CodeBuild日誌中查看請求的標籤。



作業完成後，您將能夠檢視工作的記錄。



關於 CodeBuild主體 GitHub 動作流道

什麼時候應該在標籤中包含映像和實例覆蓋？

您可以在標籤中包含映像檔和執行個體覆寫，以便為每個 GitHub Actions 工作流程工作指定不同的建置環境。這可以在不需要創建多個 CodeBuild 項目或 Webhook 的情況下完成。例如，當您需要為工作流程工作使用矩陣時，這很有用。

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}-
${{ matrix.os }}
    strategy:
      matrix:
        os: [arm-3.0-small, al2-5.0-large]
    steps:
      - run: echo "Hello World!"
```

> **Note**
>
> 如果runs-on有多個包含 GitHub 「動作」前後關聯的標籤，則可能需要引號。

我可以使 AWS CloudFormation 用此功能嗎？

是的，您可以在 AWS CloudFormation 模板中包含一個過濾器組，該過濾器組指定了項目 webhook 中的 GitHub 操作工作流程作業事件過濾器。

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
```

如需詳細資訊，請參閱 篩選 GitHub 網路掛鉤事件 ()AWS CloudFormation。

如果您需要在 AWS CloudFormation 範本中設定專案認證的說明，請參閱AWS CloudFormation 使用者指南AWS::CodeBuild::SourceCredential中的以取得更多資訊。

哪些區域支援使用 CodeBuild託管的 GitHub 動作亞軍？

CodeBuild所有 CodeBuild 地區都支持託管的 GitHub 操作運行器。如需有關可用 AWS 區域 位置 CodeBuild 的詳細資訊，請參閱按地區分類的AWS 服務。

哪些平台支持使用 CodeBuild託管的 GitHub 操作運行器？

CodeBuild-託管的 GitHub 動作執行器在 Amazon EC2 和AWS Lambda運算上都受到支援。您可以使用以下平台：Amazon Linux 2，Amazon Linux 2023，Ubuntu 和視窗服務器核心 2019 年。如需詳細資訊，請參閱 EC2 運算映像檔 及 Lambda 運算映像。

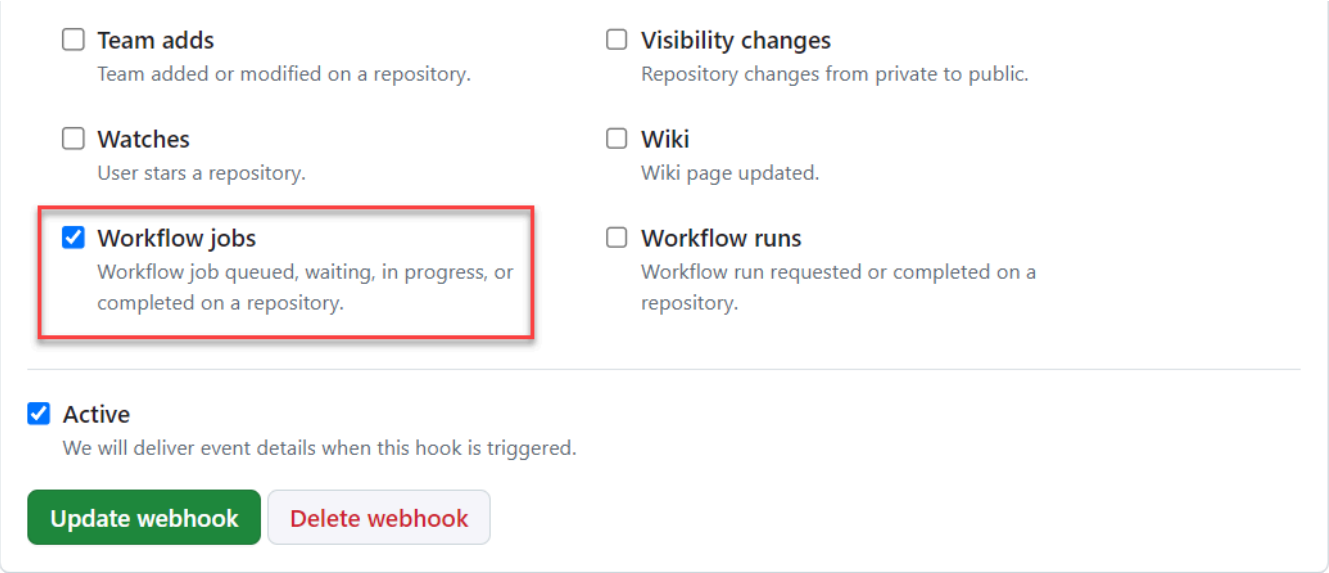
疑難排解：如果 webhook 無法正常工作，該如何進行疑難排解？

問題：您的 webhook 無法正常工作，或者您的工作流程工作掛起。 GitHub

可能的原因：您的 webhook 工作流程作業事件可能無法觸發構建。檢閱回應記錄以檢視回應或錯誤訊息。

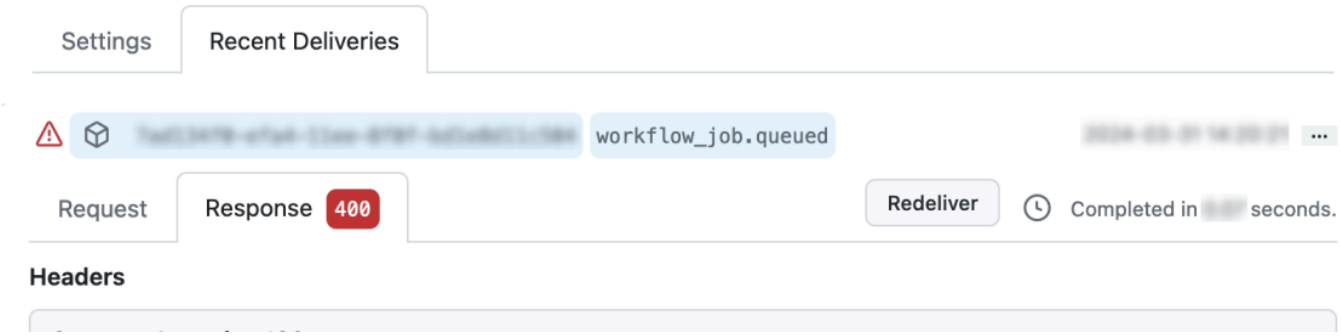
建議的解決方案：若要偵錯此錯誤，請使用下列指示。

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，您將看到為您的存儲庫創建的 webhook。
2. 選擇編輯並確定 Webhook 已啟用以傳遞 Work 工作事件。



3. 切換作業選項至「最近的交貨」頁標，搜尋對應的`workflow_job.queued`事件，然後展開事件。
4. 檢閱裝載中的標籤欄位，並確定其符合預期。

5. 最后，請查看「響應」標簽，因為它包含返回的響應或錯誤信息 CodeBuild。



在構建 GitHub 規範中使用操作語法 AWS CodeBuild

您可以使用 CodeBuild管理的動作執行器在其中 CodeBuild執行 GitHub 動作。這可以通過添加`steps`到 buildspec 文件中的任何階段來完成。

CodeBuild buildspec 支持與命令分開階段運行的順序 GitHub 操作步驟列表。 CodeBuild 這些 GitHub 動作與現 CodeBuild有功能整合，包括相依性快取、批次建置 AWS Secrets Manager、存取等等。

主題

- 我如何開始在我的構建規格中使用 GitHub 操作？
- 我可以在我的構建規格中使用哪些 GitHub 操作？
- 除了在構建規格中使用 GitHub 操作以外，我可以 GitHub 使用源提供程序嗎？
- 為什麼我需要連接到作 GitHub 為源提供程序才能在我的 buildspec 中使用 GitHub 操作？
- 在我的構建規格中使用 GitHub 操作需要多少費用？
- 哪些地區支持在我的構建規格中使用 GitHub 操作？
- 在構建規格中使用 GitHub 操作的最佳實踐
- 在構建規格中使用 GitHub 操作的限制 CodeBuild
- GitHub 動作亞軍構建規格參考
- GitHub 動作語法範例 AWS CodeBuild

我如何開始在我的構建規格中使用 GitHub 操作？

在構建規格中使用 GitHub 操作的高級步驟如下：

1. 如果您尚未這樣做，請將您的項目連接到 GitHub.

若要這麼做，您可以執行下列其中一項作業：

- 您可以在主控台中新增 GitHub 為來源提供者。如需詳細資訊，請參閱 Connect GitHub 訪問令牌（控制台）。
- 您可以通過 CodeBuild API 導入 GitHub 憑據。如需詳細資訊，請參閱 Connect GitHub 用存取權杖 (CLI) 連線。

> **Note**
>
> 只有 GitHub 在您尚未連接到另一個項目時才需要完成此操作。

2. 在你的項目的 buildspec 中，你可以添加`steps`，每個都引用一個 GitHub 動作。這可以在 CodeBuild 控制台或源存儲庫中進行編輯。每個構建階段都支持命令列表或步驟列表，但兩者都不能在同一個階段使用。如需詳細資訊，請參閱 在構建 GitHub 規範中使用操作語法 AWS CodeBuild。

我可以在我的構建規格中使用哪些 GitHub 操作？

您可以使用任何不與這些限制衝突的 GitHub Marketplace 中可用的動作。

除了在構建規格中使用 GitHub 操作以外，我可以 GitHub 使用源提供程序嗎？

是，但仍需要 GitHub 連線到才能驗證 GitHub和存取 GitHub 動作。如需詳細資訊，請參閱 GitHub 和 GitHub 企業服務器訪問令牌。

為什麼我需要連接到作 GitHub 為源提供程序才能在我的 buildspec 中使用 GitHub 操作？

為了在構建規格中使用 GitHub 操作，必須在構建計算上下載源代碼。匿名下載將受到速率限制，因此通過連接到 GitHub，它可以幫助確保一致的訪問。

在我的構建規格中使用 GitHub 操作需要多少費用？

支持在構建規格中使用 GitHub 操作，無需額外費用。

哪些地區支持在我的構建規格中使用 GitHub 操作？

在所 CodeBuild 有地區都支持在構建規格中使用 GitHub 操作。如需有關可用 AWS 區域 位置 CodeBuild 的詳細資訊，請參閱按地區分類的AWS 服務。

在構建規格中使用 GitHub 操作的最佳實踐

GitHub 行動是開源的，由社區構建和維護。我們遵循共同的責任模型，並將 GitHub Actions 源代碼視為您負責的客戶數據。 GitHub 您可以將動作授與密碼、儲存庫權杖、原始程式碼和帳戶連結的存取權。確保您對計劃 GitHub 執行動作的可信度和安全性充滿信心。

更具體的 GitHub 動作指引和安全性最佳做法：

- 安全性強化
- 防止 pwn 請求
- 不受信任的輸入
- 如何信任您的建構區塊

在構建規格中使用 GitHub 操作的限制 CodeBuild

- GitHub 在中不支援組建規格中內部依賴`github`前後關聯或參考 GitHub特定資源的動作，例如提取要求和問題。 CodeBuild例如，下列動作不適用於 CodeBuild：
  - GitHub 嘗試新增、變更或更新 GitHub資源的動作，例如更新提取請求或在中建立問題的動作 GitHub。

  > **Note**
  >
  > https://github.com/actions 中列出的大多數官方 GitHub 操作都依賴於上`github`下文。而是使用 GitHub Marketplace 中可用的動作。

- GitHub 構建規格中的操作是 D ocker 容器操作將起作用，但是您的構建項目必須啟用特權模式，並由默認的 Docker 用戶（root）運行。
  - 動作必須以 root 使用者身分執行。如需詳細資訊，請參閱 Docker 檔案動作 GitHub 支援中的使用者主題。
- GitHub 在配置為在 Windows 上運行的 CodeBuild 項目中不支持構建規格中的操作。
- GitHub 不支援組建規格中的動作工 GitHub 作 (步驟群組) 和動作工作屬性。
- GitHub 在配置為由公共 Git 存儲庫的 webhook 觸發的 CodeBuild 項目中，不支持構建規格中的操作。如需詳細資訊，請參閱git-credential-helper。
- 沒有公共互聯網訪問的 VPC 構建無法在您的構建規格中運行 GitHub 操作。
- 每個構建階段都支持命令列表或步驟列表，但兩者都不能在同一個階段使用。例如，在以下示例中，步驟用於構建前階段列出 GitHub 操作，而在構建階段使用命 CodeBuild 令列出命令。

```
version: 0.2
phases:
  pre-build:
    steps:
      - name: Lint Code Base
        uses: github/super-linter@v4
        env:
          VALIDATE_ALL_CODEBASE: 'true'
          DEFAULT_BRANCH: main
  build:
    commands:
      - echo "Building..."
      - npm run build
```

GitHub 動作亞軍構建規格參考

本主題包含動作執行器屬性的 buildspec GitHub 參考。

steps

選用的序列。步驟用於執行中的命令和動作 CodeBuild。如需詳細資訊，請參閱 在構建 GitHub 規範中使用操作語法 AWS CodeBuild。

> **Note**
>
> 每個構建階段都支持列表`commands`或列表`steps`，但兩者都不能在同一個階段中使用。

每個建置步驟都包含下列屬性。

id

選用。可用來參照其他前後關聯中步驟的步驟識別元。

if

選用。除非符合條件，否則可用來防止步驟執行的條件陳述式。這個陳述式可以使用任何支援的內容，例如從中參考環境變數 CodeBuild，以及運算式。

name

選用。步驟的名稱。如果未指定名稱，名稱將預設為指`run`令中指定的文字。

用途

針對步驟執行的動作。某些動作會要求您使用來設定輸入`with`。請參考動作的 README 檔案，以判斷需要哪些輸入。如需詳細資訊，請參閱 我可以在我的構建規格中使用哪些 GitHub 操作？。

如果`uses`在構建階段中指定，則不能與`run`.

> **Note**
>
> 建議您加入您正在使用的動作版本。這可以通過指定 Git 參考，SHA 或碼頭標籤來完成。如需詳細資訊，請參閱 steps.use 語法。

運行

執行命令列程式的命令。這些可以是單行命令或多行命令。依預設，這些命令會使用非登入殼層執行。若要選擇不同的外殼，請使用`shell`。

如果`run`在構建階段中指定，則不能與`uses`.

殼

選用。為此序列指定的外殼。有關支持的外殼參數，請參閱步驟 shell。如果未指定，則使用的外殼是 bash。如果 bash 不可用，則使用 sh。

與

選用。由動作定義的輸入參數對映。每個參數都是一個鍵/值對。

與. 參數

選用。定義 Docker 容器輸入的字串。

與. 入口點

選用。為碼頭檔案指定的碼頭工人入口點。

環境

選用。為要在環境中使用的步驟指定的變數。

continue-on-error

選用。Boolean 值，指出是否可以忽略此步驟序列失敗。

`false`

預設值。如果此步驟順序失敗，則構建將失敗。

`true`

如果此步驟序列失敗，組建仍然可以成功。

超時-分鐘

選用。在終止之前，步驟可以執行的最大分鐘數。默認情況下，沒有超時。如果步驟逾時超過構建超時，則到達構建超時時，步驟將停止。

以下是使用超林特動 GitHub 作的範例：

```
version: 0.2
phases:
  build:
    steps:
      - name: Lint Code Base
        uses: github/super-linter@v5
        env:
          VALIDATE_ALL_CODEBASE: true
          USE_FIND_ALGORITHM: true
          FILTER_REGEX_INCLUDE: '/github/workspace/buildspec.yml'
```

GitHub 動作語法範例 AWS CodeBuild

這些樣本組可用於在中嘗試構建規格中的 GitHub 操作。 CodeBuild

主題

- 超棉絨動作 GitHub 示例
- Batch 構建圖示例
- Amazon 評論 CodeGuru 者示例
- AWS Secrets Manager 樣本
- 環境變數範例
- 匯出的環境變數範例

超棉絨動作 GitHub 示例

此範例示範如何將超級林特 GitHub 動作加入至專案。 CodeBuild Super-Linter 動作會檢查程式碼、尋找程式碼有錯誤、格式化問題和可疑結構的區域，然後將結果輸出至主控台。 CodeBuild

您可以通過更新 buildspec 文件的階段部分將超級林特 GitHub 操作添加到您的 CodeBuild 項目中。

```
version: 0.2
phases:
  build:
    steps:
      - name: Lint Code Base
        uses: github/super-linter@v5
        env:
          VALIDATE_ALL_CODEBASE: true
```

超林特記錄檔看起來會類似下列內容：

```
/github/workspace/hello-world/app.js:3:13: Extra semicolon.
/github/workspace/hello-world/app.js:9:92: Trailing spaces not allowed.
/github/workspace/hello-world/app.js:21:7: Unnecessarily quoted property 'body' found.
/github/workspace/hello-world/app.js:31:1: Expected indentation of 2 spaces but found
 4.
/github/workspace/hello-world/app.js:32:2: Newline required at end of file but not
 found.
```

Batch 構建圖示例

下列範例會定義建立相依性鏈並使用執行指令的建置圖形`steps`。在此範例中，先`build1`執行，因為它沒有相依性。由於`build2`具有依賴關係`build1`，因此在 build1 完成後`build2`運行。如需相關資訊，請參閱構建圖。

```
version: 0.2
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1
```

```
phases:
  build:
    steps:
      - run: echo $BUILD_ID
```

Amazon 評論 CodeGuru 者示例

Amazon CodeGuru 審核者會在您的 Java 和 Python 程式碼中找到問題，並建議如何修復這些問題。下列範例使用 CodeGuru Reviewer 來提供完整的儲存庫分析程式碼檢閱。這些代碼審查掃描指定分支中的所有代碼。如需詳細資訊，請參閱 Amazon CodeGuru 審核者使用者指南中的使用 GitHub 動作建立程式碼檢閱。

```
version: 0.2
phases:
  build:
    steps:
        - name: Amazon CodeGuru Reviewer Scanner
          if: ${{ always() }}
          uses: aws-actions/codeguru-reviewer@v1.1
          with:
            s3_bucket: codeguru-reviewer-user

artifacts:
  files:
    - codeguru-results.sarif.json
```

> **Note**
>
> 您的 Amazon S3 儲存貯體必須以`codeguru-reviewer-`前綴開頭。

記錄檔看起來會類似下列內容：

```
INFO CodeReview created with arn=arn:aws:codeguru-reviewer:region:account-
id:association:id:code-review:RepositoryAnalysis-job for job=job
INFO SARIF persisted to /github/workspace/codeguru-results.sarif.json
INFO Amazon CodeGuru Reviewer job execution completed
```

Amazon CodeGuru 審核者任務完成後，會產生一份 sarif 報告作為 CodeBuild 成品。如需詳細資訊，請參閱 Amazon CodeGuru 審核者使用者指南中的完整儲存庫分析。

AWS Secrets Manager 樣本

AWS Secrets Manager 協助您在整個生命週期中管理、擷取和輪換資料庫認證、應用程式登入資料、OAuth 權杖、API 金鑰和其他機密。下列範例會使用 Secret 管理員定義密碼，並使用來執行命令steps。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。

```
version: 0.2
env:
  secrets-manager:
    SECRET_VALUE: "arn:aws:secretsmanager:us-east-1:xxxx:secret:/secret-
l3IJg9:my_super_secret_key"
phases:
  build:
    steps:
      - run: echo $SECRET_VALUE
```

記錄檔看起來會類似下列內容：

```
echo $SECRET_VALUE
env:
  SECRET_VALUE: ***
***
```

環境變數範例

下列範例會定義env序列下的環境變數。*S3_BUCKET* <bucket-name>變數會在建置規格中定義，並指派為其值。此變數會在 if 條件中參考，如同一般環境變數，方法是使用美元符號 ($) 來存取 GitHub Action env 內容。如需詳細資訊，請參閱env序列。

```
version: 0.2
env:
  variables:
    S3_BUCKET: "<bucket-name>"
phases:
  build:
    steps:
      - if: ${{ env.S3_BUCKET == '<bucket-name>' }}
        run: echo "S3 bucket is $S3_BUCKET"
```

記錄檔看起來會類似下列內容：

```
echo "S3 bucket is $S3_BUCKET"
env:
  S3_BUCKET: my-s3-bucket
S3 bucket is my-s3-bucket
```

匯出的環境變數範例

匯出的環境變數可與搭配使用， CodePipeline 將環境變數從目前的建置階段匯出至管線中的後續階段。下列範例會在名為 *MY_VARIABLE #env################ GITHUB_EN V #####*

```
version: 0.2
env:
  exported-variables:
    - MY_VARIABLE
phases:
  build:
    steps:
      - run: echo "MY_VARIABLE=my-value" >> $GITHUB_ENV
```

如需詳細資訊，請參閱 AWS CodeBuild API 參考 ExportedEnvironmentVariable中的。

中的公共組建專案AWS CodeBuild

AWS CodeBuild允許您將生成項目的構建結果、日誌和工件提供給公眾。這允許源存儲庫的貢獻者查看結果並下載構建的工件，而無需他們訪問AWS帳户。

當您將項目的內部版本提供給公眾時，項目的所有生成結果、日誌和工件（包括在項目為私有時運行的構建）都可供公眾使用。同樣，當您將公共構建項目設置為私有時，該項目的生成結果將不再向公眾提供。

如需如何變更專案組建結果的公開可見性，請參閱啟用公用組建存取權。

CodeBuild 為專案提供專案專屬的公共組建 URL。若要獲取組建專案的公共 URL，請執行下列程序：

1. 開啟AWS CodeBuild主控台位於https://console.aws.amazon.com/codesuite/codebuild/home。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 選擇您希望獲取其公共 URL 的組建專案連結。
4. 公共 URL 會顯示在公共項目 URL欄位組態區段。您可以選擇鏈接以打開 URL，或使用複製按鈕複製 URL。

> **Warning**
>
> 在公開專案建置結果時，應牢記以下幾點：
>
> - 項目的所有構建結果、日誌和工件（包括在項目為私有時運行的構建）都可供公眾使用。
> - 所有構建日誌和工件都可供公眾使用。環境變量、源代碼和其他敏感信息可能已輸出到構建日誌和工件。您必須小心輸出到構建日誌的信息。一些最佳實踐包括：
>   - 請勿存放機密值，尤其是AWS存取金鑰 ID 和私密存取金鑰。我們建議您使用 Amazon EC2 Systems Manager 參數存放區或AWS Secrets Manager來存儲敏感值。
>   - 跟隨使用網路掛鉤的最佳做法來限制哪些實體可以觸發構建，並且不在項目本身中存儲 buildspec，以確保您的 webhooks 儘可能安全。
> - 惡意用户可以使用公共構建來分發惡意工件。我們建議項目管理員查看所有拉取請求，以驗證拉取請求是否合法更改。我們還建議您使用校驗和驗證任何對象，以確保下載正確的工件。

使用 AWS CodeBuild 中的建置

組建代表的是 AWS CodeBuild 執行的一組動作，用來根據一組輸入成品 (例如 Java 類別檔案的集合) 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 AWS CodeBuild 的配額。
- 如果生成項目具有併發生成限制集，則如果正在運行的生成數量達到項目的併發生版本限制，生成將返回錯誤。如需詳細資訊，請參閱「」啟用併發生成限制。
- 如果生成項目沒有併發生版本限制集，則如果正在運行的生成數量達到平台和計算類型的併發構建限制，則構建將排隊。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 AWS CodeBuild 的配額。

  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 在 AWS CodeBuild 中執行建置。

  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

在 AWS CodeBuild 中執行建置

您可以使用AWS CodeBuild主控台、AWS CLI, 或AWS開發套件，以在代碼組建中執行組建。

主題

執行建置 (主控台)

使用AWS CodePipeline若要使用 CodeBuild 執行組建，請跳過這些步驟，並遵循 CodePipeline 搭配使用 CodeBuild。

1. 開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 在組建專案清單中，選擇組建專案。
4. 您可以使用默認的生成項目設置運行構建，或僅覆蓋此版本的生成設置。

   a. 若要使用默認組建專案設定執行組建，請選擇啟動組建專案。組建會即刻開始。

   b. 若要覆寫默認組建專案設定，請選擇啟動組建專案。在 中啟動組建專案頁面上，您可以覆寫下列項目：

      - 建置組態
      - Source (來源)
      - 環境變數覆寫

      如果需要選擇更高級的覆蓋，請選擇進階組建覆寫。在此頁面上，您可以覆寫下列項目：

      - 建置組態
      - Source (來源)
      - 環境
      - BuildSpec
      - 成品
      - Logs (日誌)

      完成覆寫選擇後，請選擇啟動組建專案。

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 以利用 AWS CodeBuild 執行組建，請跳過這些步驟，並遵循建立使用 CodeBuild 的管道 (AWS CLI)中的說明。
> 如需搭配使用 AWS CLI 與 CodeBuild 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. 如果您執行start-build使用指令--project-name選項，取代*<project-name>*使用建置專案的名稱，然後跳至此程序的步驟 6。若要取得組建專案清單，請參閱檢視建置專案名稱清單。

3. 如果您執行start-build使用指令--idempotency-tokenoption (區分大小寫的唯一識別碼或權杖)包含在start-build請求。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來重複 start-build 請求，但變更參數，則 CodeBuild 會傳回參數不符錯誤。

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。將資料複製至本機電腦或執行個體上 AWS CLI 安裝位置中的檔案 (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，對應於您要建立之輸入 ZIP 檔案版本的版本識別碼。如果未指定 *sourceVersion*，則會使用最新版本。
  - 針對 CodeCommit，為與您想要建置之來源碼版本對應的遞交 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)。
  - 對於GitHub、提交 ID、提取要求 ID、分支名稱或標籤名稱，對應於您要建置的原始程式碼版本。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*：可選。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-##*：可選。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##*：可選。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*：可選。您想要針對此建置覆寫組建專案中所定義的環境變數值。
- *gitCloneDepth##*：可選。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。該令牌在開始構建請求後的 5 分鐘內有效。如果您使用相同的令牌重複啟動構建請求，但更改參數，CodeBuild返回參數不匹配錯誤。
- *insecureSslOverride*: 選用的布林值，指定是否覆寫建置專案中指定的不安全 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。此覆蓋僅在構建的源代碼為時適用GitHub企業伺服器。
- *privilegedModeOverride*: 可選布林值。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*: 選擇性整數，指定組建在逾時前允許排入佇列的分鐘數。其最小值為五分鐘，其最大值為 480 分鐘 (八小時)。
- *reportBuildStatus##*: 選用的布林值，指定是否要傳送組建開始和完成狀態的來源提供者。如果您使用非來源提供者進行設定GitHub,GitHub企業服務器，或比特桶，invalidInputException被拋出。
- *sourceAuthOverride*: 可選字串。此建置的授權類型，可覆寫組建專案中所定義的授權類型。此覆蓋僅適用於構建項目的源代碼是 Bitbucket 或GitHub。
- *sourceLocationOverride*: 可選字串。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*: 可選字串。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*: 可選字串。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`。
- *timeoutInMinutes##*：可選數字。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。

我們建議您儲存具有敏感值的環境變數，例如AWS存取金鑰識別碼，一個AWS秘密存取金鑰或密碼做為 Amazon EC2 系統管理員參數存放區中的參數。CodeBuild只有在參數名稱開頭為時，才能使用存放在 Amazon EC2 系統管理員參數存放區中的參數`/CodeBuild/`（例如，`/CodeBuild/dockerLoginPassword`). 您可以使用CodeBuild在亞馬遜 EC2 系統管理器中創建參數的控制台。選擇 Create a parameter (建立參數)，然後遵循指示進行。(在該對話方塊中，用於KMS 金鑰，您可以選擇性地指定的 ARNAWS KMS輸入您的帳戶。 Amazon EC2 系統管理員使用此金鑰在儲存期間加密參數的值，並在擷取期間進行解密。) 如果您使用 CodeBuild 主控台建立參數，則主控台會使用存放的 `/CodeBuild/` 來啟動參數。但是，如果您使用 Amazon EC2 系

統管理員參數存放區主控台建立參數，則必須以/CodeBuild/，且您必須設定类型至安全字串。如需詳細資訊，請參閱〈AWS Systems Manager參數存放區和逐步解說：建立和測試 String 參數 (主控台)在亞馬遜 EC2 系統管理器用戶指南。

如果您的建置專案參考存放在 Amazon EC2 系統管理員參數存放區中的參數，則建置專案的服務角色必須允許ssm:GetParameters動作。如果您稍早選擇 Create a new service role in your account (在帳戶中建立新的服務角色)，則 CodeBuild 會將此動作自動包含在您建置專案的預設服務角色中。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，請參閱檢視建置專案的詳細資訊。

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 程序中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. 如果您搭配執行start-build-batch命令與`--project-name`選項，替換*<project-name>*與組建專案的名稱，然後跳到本程序的步驟 6。若要取得組建專案清單，請參閱檢視建置專案名稱清單。
3. 如果您搭配執行start-build-batch命令與`--idempotency-token`選項，則會隨附唯一區分大小寫識別符或字符。`start-build-batch`請求。此字符在 請求之後的 5 分鐘內有效。如果重複`start-build-batch`請求，但變更參數，則 CodeBuild 會返回參數不符錯誤。
4. 如果您搭配執行start-build-batch命令與`--generate-cli-skeleton`選項時，JSON 格式的數據將輸出到*<json-file>*file. 此檔案與start-build命令，並添加以下對象。如需公共物件的詳細資訊，請參執行建置 (AWS CLI)。

修改此檔案以添加任何建置覆蓋，並儲存您的結果。

```
"buildBatchConfigOverride": {
  "combineArtifacts": combineArtifacts,
  "restrictions": {
    "computeTypesAllowed": [
      allowedComputeTypes
    ],
    "maximumBuildsAllowed": maximumBuildsAllowed
  },
  "serviceRole": "batchServiceRole",
  "timeoutInMins": batchTimeout
}
```

所以此`buildBatchConfigOverride`物件是ProjectBuildBatchConfig結構，該結構包含此版本的批處理構建配置覆蓋。

####

布爾值，指定批次建置的建置成品是否應合併成單一成品位置。

########

字串陣列，指定批次建置允許的運算類型。請參閱建置環境運算類型對於這些值。

######

指定允許的建置數目上限。

####

指定批次建置專案的服務角色 ARN。

#####

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，則構建批處理物件會顯示於主控台輸出。請參。啟動建置批次回應語法查看此數據的示例。

自動開始執行建置 (AWS CLI)

如果您的來源碼存放在 GitHub 或 GitHub Enterprise Server 儲存庫中，則您可以使用 GitHub Webhook 讓 AWS CodeBuild 只要將程式碼變更推送至儲存庫時就重建來源碼。

執行 create-webhook 命令，如下所示：

```
aws codebuild create-webhook --project-name <project-name>
```

*<project-name>*是包含要重建之來源碼的組建專案名稱。

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-ZFyY02rs4JSE7OmLW3w-gh-ryoVB8OSSSC1aAtBtuPkHw
YuncCCmdogCVCfniQ7ukYX2_xM--nlDma5EngIg_Bi_N465yi33zyTUNPoQlxCpLO-BwghcVa9lAurwR77-uY7i-_XCJFahwMx1f4ubOgBBSmMT2A16apqjq
QJoKSb61XVKyZy1Giuy4nliAXfv9WNn76CaCsndb3fVIE78fpygfo41xYxSQ6vpo6LRTKtPzbyeTHbVXGda1PJvnkBlnKmJDo0RTgI1m2oYr17dWziQlrrvo
CoNgy1SOO_7LKfA-nNXFc_f1SiFy0AqeMB43-dOOcdkzybHncE81QTRwEUCFfmX-AJCwmLXVOkg0G67T92Sjbpz0fRlkh5pwIFl93_bB_jOHDinK6iOiPpf2
dIDAIZgGMagqZeWb-axDeTAbopoU8J6gFI1yKo5aq9ql51zC1PERUsMgJFtJr_a-Z-L_kylr-4hSSxasSJNuJ43_XOBRWqT51xqvH-A69bVO7KbVT_Kc6wxk
SHyYCEMoa_Pfa7ZQgyfY6B00ogMNj31yFbjthORNL1cDo6-3J-McDLoyrRtSEOV9QnxvsG5zu1N5-z20rkJtg_MOfNwocfUutFXb7vrGTduHlR1dzXLRusHu
xOVVuDUWm9vhWMr-hUkeGo_1kDKyk4E2QFvZXpjYw0vFfV-dwxFRR_mifzxW1wyfmt2iFtLkp_YZj_4WeFAckGefr-ilNaYvsZpzXj78Ae1adVoLf48AmDdN
2pWsWJjatU9zt942gLisFFmKakcvJuy5yxXHaxxbhUyC8NHYiESUWPfcfnqrMsr8op3P4AUCHlpiZCYYuiwI_cac-pIUBO0Xaur_lu_fyFghgOJc7cfTnA36
rv5X5DnFDM8P3HNBeLjaF9QZ6AijegPEwTHIkJON3AUDwpkz_hwTXyUoAU8MdZfPTXbBoT6N5Z5THBHsYxR",
        "payloadUrl": "https://codebuild.us-east-2.amazonaws.com/webhooks?t=eyJlbmNyeXB0ZWREYXRhIjoiUmFqMmJERGRQbGhwLzNT
Nld3R0VGRjZZOTNwLzlZVG1NZlpIR1E0RUsxdzhGeWhnVFFqWTR0WEFwT2dJRnNmRHc3S3RNc0xYMENncXFTakg1cE1nSy9zPSIsIml2UGFyYW1ldGVyU3Bl
YyI6IndSQ1Qrc2VPQjBCZzhPeVYiLCJtYXRlcmlhbFNldFNlcmlhbCI6MX0%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，選擇存放您專案的儲存庫。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，則您可以設定 GitHub Webhook，讓 AWS CodeBuild 在每當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>
```

- 哪裡*<project-name>*是包含要重建之來源碼的組建專案名稱。

如果此命令成功，則輸入中不會顯示任何資訊和錯誤。

> **Note**
>
> 這只會刪除專案 CodeBuild Webhook。您也應該刪除 GitHub 或 GitHub Enterprise Server 儲存庫中的 Webhook。

執行建置 (AWS 開發套件)

若要使 CodePipeline 運行AWS CodeBuild，請跳過這些步驟，並遵循搭配 AWS CodeBuild 使用 AWS CodePipeline 測試程式碼及執行建置INSTEAD OP

如需將 CodeBuild 與AWS開發套件，請參AWS 開發套件和工具參考。

在 AWS CodeBuild 中檢視建置的詳細資訊

您可以使用AWS CodeBuild主控台、AWS CLI, 或AWS開發套件，以查看 CodeBuild 管理的組建詳細資訊。

主題

檢視建置的詳細資訊 (主控台)

1. 開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 執行下列任一步驟：
   - 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)

如需搭配使用 AWS CLI 與 AWS CodeBuild 的詳細資訊，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：

- *ID*：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
```

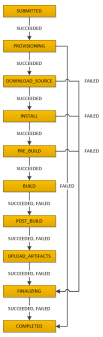
如果命令成功，輸出中會顯示類似於 檢視摘要建置資訊 所述的資料。

檢視建置的詳細資訊 (AWS 開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

建置階段轉換

在 AWS CodeBuild 中的組建會分階段進行：



> **Important**
>
> 即使 BUILD 階段失敗，也一定會嘗試 UPLOAD_ARTIFACTS 階段。

在 AWS CodeBuild 中檢視建置 ID 清單

您可以使用AWS CodeBuild主控台、AWS CLI, 或AWS開發套件：檢視 CodeBuild 管理的組建 ID 清單。

主題

檢視建置 ID 清單 (主控台)

1. 開啟AWS CodeBuild主控台：https://console.aws.amazon.com/codesuite/codebuild/home。
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)

如需使用AWS CLI與 CodeBuild，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：

- *####*：用於表示如何列出組建 ID 的選用字串。有效值包括 ASCENDING 與 DESCENDING。
- *#####*：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"
  ]
}
```

檢視批處理組建 ID 清單 (AWS CLI)

如需使用AWS CLI與 CodeBuild，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：

- *####*：用於表示如何列出批處理組建 ID 的選用字串。有效值包括 ASCENDING 與 DESCENDING。
- *#####*：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...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-build-batches --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"
  ]
}
```

檢視建置 ID 清單 (AWS 開發套件)

如需使用 CodeBuild 的詳細資訊，請參AWS開發套件，請參AWS 開發套件和工具參考。

在 AWS CodeBuild 中檢視建置專案的建置 ID 清單

您可以使用AWS CodeBuild主控台、AWS CLI, 或AWSSDK，用於在 CodeBuild 中檢視建置專案的建置 ID 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)
- 檢視建置專案的批處理建置 ID 清單 (AWS CLI)
- 檢視建置專案的建置 ID 清單 (AWS 開發套件)

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)

如需搭配使用 AWS CLI 與 AWS CodeBuild 的詳細資訊，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：

- *####*：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)。
- *####*：用於表示如何列出組建 ID 的選用字串。有效值包括 ASCENDING 與 DESCENDING。
- *#####*：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，如果您執行與下列類似的這個命令：

```
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"
  ]
}
```

檢視建置專案的批處理建置 ID 清單 (AWS CLI)

如需搭配使用 AWS CLI 與 AWS CodeBuild 的詳細資訊，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：

- *####*：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)。
- *####*：用於表示如何列出組建 ID 的選用字串。有效值包括 ASCENDING 與 DESCENDING。
- *#####*：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，如果您執行與下列類似的這個命令：

```
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"
  ]
}
```

檢視建置專案的建置 ID 清單 (AWS 開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

在 AWS CodeBuild 中停止建置

您可以使用 AWS CodeBuild 主控台、AWS CLI 或 AWS 開發套件，以停止 AWS CodeBuild 中的組建。

主題

停止建置 (主控台)

1. 開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 執行下列任一步驟：
   - 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 無法成功停止組建 (例如組建程序已完成)，則 Stop (停止) 按鈕會停用，也可能不會出現。

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 成功停止組建，輸出中 `build` 物件中的 `buildStatus` 值將會是 `STOPPED`。

如果 CodeBuild 無法成功停止組建 (例如組建已完成建置)，則`buildStatus`值`build`對象是最終構建狀態（ 例如，`SUCCEEDED`。

停止建置 (AWS 開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

在中停止批次組建AWS CodeBuild

您可以使用AWS CodeBuild主控台、AWS CLI，或AWS在中停止批次組建的開發套件AWS CodeBuild。

主題

- 停止批次組建 (主控台)
- 停止批次組建 (AWS CLI)
- 停止批次組建 (AWS開發套件)

停止批次組建 (主控台)

1. 開啟AWS CodeBuild在的主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 執行下列任一步驟：
   - 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如果無法成功停止批次組建 (例如組建程序已完成)，則停止建置按鈕處於禁用狀態。

停止批次組建 (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>*：必要的字串。要停止的批次組建標識符。若要取得批次組建標識符的清單，請參下列主題：
    - 檢視批處理組建 ID 清單 (AWS CLI)
    - 檢視建置專案的批處理建置 ID 清單 (AWS CLI)

  如果AWS CodeBuild成功停止批處理構建，buildBatchStatus中的值buildBatch對象是STOPPED。

  如果 CodeBuild 無法成功停止批次組建 (例如批次組建已完成)，則buildBatchStatus中的值buildBatch對象是最終構建狀態（例如SUCCEEDED。

停止批次組建 (AWS開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

重試組建AWS CodeBuild

您可以使用AWS CodeBuild主控台、AWS CLI，或AWSSDK 重試單個構建或批量構建AWS CodeBuild。

主題

- 重試組建 (主控台)
- 重試組建 (AWS CLI)
- 重試組建 (AWS開發套件)

重試組建 (主控台)

1. 開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 執行下列任一步驟：
   - 如果******：*** ID**頁面，選擇重試組建。
   - 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。

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>*：必要的字串。要重試的組建或批處理建置的 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)
  - 檢視批處理組建 ID 清單 (AWS CLI)
  - 檢視建置專案的建置 ID 清單 (AWS CLI)

- 檢視建置專案的批處理建置 ID 清單 (AWS CLI)
- `--idempotency-token`：選用。如果您執行retry-build命令搭配選項，則會隨附唯一區分大小寫識別符或字符。`retry-build`請求。此字符在 請求之後的 5 分鐘內有效。如果重複`retry-build`請求，但更改參數，則 CodeBuild 會返回參數不符錯誤。

重試組建 (AWS開發套件)

如需使用 AWS CodeBuild 與 AWS 開發套件的詳細資訊，請參閱AWS 開發套件和工具參考。

在工作階段管理員中檢視正在執行

在中 AWS CodeBuild，您可以暫停執行中的組建，然後使用 AWS Systems Manager 工作階段管理員連線至組建容器並檢視容器的狀態。

> **Note**
>
> 此功能在 Windows 環境中無法使用。

主題

- 必要條件
- 暫停組建
- 開始構建
- Connect 到構建容器
- 繼續組建

必要條件

要允許會話管理器與構建會話一起使用，您必須為構建啟用會話連接。有兩個先決條件：

- CodeBuild Linux 標準組織映像已安裝 SSM 代理程式，且已啟用 SSM 代理程式 ContainerMode 。

  如果您要為組建使用自訂映像檔，請執行下列動作：

  1. 安裝 SSM Agent。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的在適用於 Linux 的 EC2 執行個體上手動安裝 SSM 代理程式。SSM 代理程式版本必須是 3.0.1295.0 或更新版本。

2. 將檔案複製到您aws-codebuild-docker-images的映像檔中的目錄中amazon-ssm-agent。/etc/amazon/ssm/ https://github.com/aws/ 這會在 SSM 代理程式中啟用容器模式。

> **Note**
>
> 自訂映像檔需要最新的 SSM 代理程式，此功能才能如預期般運作。

- CodeBuild 服務角色必須具有下列 SSM 原則：

```
{
  "Effect": "Allow",
  "Action": [
    "ssmmessages:CreateControlChannel",
    "ssmmessages:CreateDataChannel",
    "ssmmessages:OpenControlChannel",
    "ssmmessages:OpenDataChannel"
  ],
  "Resource": "*"
}
```

當您啟動組建時，您可以讓 CodeBuild 主控台自動將此原則附加至您的服務角色。或者，您可以手動將此原則附加至服務角色。

- 如果您在 Systems Manager 偏好設定中啟用稽核和記錄工作階段作業活動，則 CodeBuild 服務角色也必須具有其他權限。權限會有所不同，具體取決於記錄檔的儲存位置。

  CloudWatch 日誌

  如果使用 CloudWatch Logs 儲存記錄檔，請將下列權限新增至 CodeBuild 服務角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:DescribeLogGroups",
      "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:<log-group-
name>:*"
    }
  ]
}
```

Amazon S3

如果使用 Amazon S3 存放日誌，請將以下權限新增至 CodeBuild 服務角色：

```
{
  "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 使用指南中的稽核和記錄工作階段活動。

暫停組建

要暫停構建，請在 buildspec 文件中的任何構建階段中插入codebuild-breakpoint命令。此時組建將暫停，這可讓您連接到組建容器，並以目前狀態檢視容器。

例如，將以下內容添加到 buildspec 文件中的構建階段中。

```
phases:
  pre_build:
    commands:
```

```
    - echo Entered the pre_build phase...
    - echo "Hello World" > /tmp/hello-world
    - codebuild-breakpoint
```

此代碼創建/tmp/hello-world文件，然後在此時暫停構建。

開始構建

若要允許工作階段管理員與建置工作階段搭配使用，您必須啟用組建的工作階段連線。若要這麼做，請在開始建置時執行下列步驟：

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 在導覽窗格中，選擇 Build projects (建置專案)。選擇組建專案，然後選擇 [使用覆寫項目開始建置]。
3. 選擇 Advanced build overrides (進階組建覆寫)。
4. 在「環境」段落中，選擇啟用階段作業連線選項。如果未選取此選項，則會忽略所有codebuild-breakpoint和codebuild-resume指令。
5. 進行任何其他所需的變更，然後選擇 [開始建置]。
6. 在主控台中監視組建狀態。當工作階段可用時，[工AWS 作階段管理員] 連結會顯示在 [建置狀態] 區段中。

Connect 到構建容器

您可以使用以下兩種方式之一連接到構建容器：

CodeBuild 控制台

在 Web 瀏覽器中，開啟 [AWS 工作階段管理員] 連結以連線到組建容器。終端機工作階段隨即開啟，可讓您瀏覽及控制組建容器。

AWS CLI

> **Note**
>
> 您的本機電腦必須安裝工作階段管理員外掛程式才能執行此程序。如需詳細資訊，請參閱《 AWS Systems Manager 使用者指南》中的 < 安裝 AWS CLI 的工作階段管理員外掛程式 > 。

1. 使用構建 ID 調用 batch-get-builds api 以獲取有關構建的信息，包括會話目標標識符。會話目標標識符屬性名稱取決於aws命令的輸出類型。這就是為什--output json麼添加到命令中。

```
aws codebuild batch-get-builds --ids <buildID> --region <region> --output json
```

2. 複製sessionTarget屬性值。sessionTarget屬性名稱可能會根據aws命令的輸出類型而有所不同。這就--output json是為什麼在上一個步驟中添加到命令中的原因。
3. 使用以下命令連接到構建容器。

```
aws ssm start-session --target <sessionTarget> --region <region>
```

在此範例中，請確認/tmp/hello-world檔案是否存在且包含文字Hello World。

繼續組建

完成檢查組建容器之後，從容器殼層發出codebuild-resume命令。

```
$ codebuild-resume
```

在 AWS CodeBuild 中刪除建置

您可以使用 AWS CLI 或 AWS 開發套件，在 AWS CodeBuild 中刪除建置。

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：

- *ID*：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"
  ]
}
```

刪除建置 (AWS 開發套件)

如需搭配使用 AWS CodeBuild 與 AWS 開發套件的資訊，請參閱 AWS 開發套件和工具參考。

在中使用 AWS Lambda 計算 AWS CodeBuild

AWS Lambda 運算可為您的組建提供最佳化的啟動速度。 AWS Lambda 由於較低的啟動延遲，因此支援更快的組建。 AWS Lambda 也會自動擴展，因此構建不會在隊列中等待運行。但是，有些用例 AWS Lambda 不支持，如果它們影響您，請使用 EC2 計算。如需詳細資訊，請參閱 AWS Lambda 計算的限制。

主題

哪些工具和運行時將包含在運行的運行時環境 docker 映像中？ AWS Lambda

AWS Lambda 支持以下工具： AWS CLI V2， AWS SAM CLI，混帳，圍棋，Java，Node.js，Python，點子，紅寶石和.

如果策劃的圖像不包含我需要的工具怎麼辦？

如果策劃的映像檔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 映像檔。

請注意，您需要下列 Amazon ECR 許可，才能將自訂映像檔用於 Lambda 運算：

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

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "arn:aws:ecr:image-region:image-account-id:repository/image-repo"
    }
  ]
}
```

另請注意，curl或wget必須安裝才能使用自訂映像檔。

哪些地區支援 AWS Lambda 運算 CodeBuild？

在中 CodeBuild，支援 AWS Lambda compute 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)。如需有關可用 AWS 區域 位置 CodeBuild 的詳細資訊，請參閱按地區分類的AWS 服務。

AWS Lambda 計算的限制

有些用例 AWS Lambda 不支持，如果它們影響您，請使用 EC2 計算：

- AWS Lambda 不支援需要 root 權限的工具。對於yum或之類的工具rpm，請使用 EC2 運算類型或其他不需要 root 許可的工具。
- AWS Lambda 不支持 Docker 構建或運行。您可以使用不需要 root 權限的替代方法，例如 Podman。
- AWS Lambda 不支持寫入外部文件/tmp。依預設，隨附的套件管理員會設定為使用/tmp目錄來下載和參考套件。
- AWS Lambda 不支援的環境類型，LINUX_GPU_CONTAINER並不支援 Windows 伺服器核心 2019.
- AWS Lambda 不支持緩存，批處理構建，自定義構建超時，隊列超時，構建徽章，特權模式，自定義運行時環境或超過 15 分鐘的運行時。
- AWS Lambda 不支援 VPC 連線、固定範圍的 CodeBuild 來源 IP 位址、EFS、語意版本控制、安裝憑證或透過工作階段管理員進行 SSH 存取。

AWS Lambda 運算範例 AWS CodeBuild

這些範例群組可用於在中進行 AWS Lambda 計算實驗 CodeBuild。

主題

使用 Lambda Java 來部署 AWS SAM L CodeBuild ambda 函數

AWS Serverless Application Model(AWS SAM) 是用來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（詳見）的AWS Serverless Application Model存放庫 GitHub。下面的 Java 示例使用搖籃來構建和測試AWS Lambda函數。之後，AWS SAMCLI 將用於部署AWS CloudFormation範本和部署服務包。透過使用 CodeBuild Lambda，所有建置、測試和部署步驟都會自動處理，讓基礎設施快速更新，無需在單一組建中進行手動介入。

設定您的儲AWS SAM存庫

使用 AWS SAM CLI 建立AWS SAM`Hello World`專案。

若要建立您的AWS SAM專案

1. 請遵循在本機電腦上安裝 AWS SAM CLI 的AWS Serverless Application Model開發人員指南中的指示。
2. 運行`sam init`並選擇以下項目配置。

```
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。

創建一個 CodeBuild Lambda Java 項目

建立 AWS CodeBuild Lambda Java 專案，並設定組建所需的身分與存取權管理權限。

若要建立您 CodeBuild Lambda Java 專案

1. 請在以下位置開啟AWS CodeBuild主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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 帳戶中都必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
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：
   - 對於運算，選取 Lambda。
   - 對於執行階段，請選取 Java。
   - 對於圖像，請選擇 AWS/ 代碼生成器/亞馬遜鏈-x86_64-羔羊標準：確定 21。
   - 對於服務角色，請保持選取 [新增服務角色]。記下「角色」名稱。當您稍後在本範例中更新專案的 IAM 許可時，將需要執行此操作。
6. 選擇 Create build project (建立建置專案)。
7. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
8. 在導覽窗格中，選擇 [角色]，然後選取與專案相關聯的服務角色。您可以在 CodeBuild 中找到您的專案角色，方法是選取您的組建專案，然後選擇 [編輯]、[環境]，然後選取 [服務
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基礎結構。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。

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

```
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "iam:*",
                "apigateway:*",
                "s3:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

設置項目構建規格

為了構建，測試和部署 Lambda 函數，請從構建規格 CodeBuild 讀取和執行構建命令。

若要設定您的專案建置規格

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，選擇插入構建命令，然後選擇切換到編輯器。
3. 刪除預先填充的構建命令並粘貼到以下 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)。

部署您的基 AWS SAM Lambda 架構

使用 CodeBuild Lambda 自動部署您的基礎架構

部署您的 Lambda 基礎架構

1. 選擇 Start build (開始組建)。這將自動構建，測試和部署您的AWS SAM應用程序AWS Lambda使用AWS CloudFormation.
2. 建置完成後，瀏覽至AWS Lambda主控台並在AWS SAM專案名稱下搜尋新的 Lambda 函數。
3. 選取函數概觀下的 API Gateway，然後按一下 API 端點網址，以測試您的 Lambda 函數。您應該看到一個打開的頁面，其中包含該消息`"message": "hello world"`。

清理您的基礎架構

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AWS SAM範本和所建立的資源CodeBuild。

若要清理您的基礎架構

1. 導覽至主AWS CloudFormation控台，然後選取`aws-sam-cli-managed-default`。
2. 在資源中，清空部署值區`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AWS CloudFormation堆疊。此堆疊應該與您的AWS SAM專案具有相同的名稱。
5. 導覽至 CloudWatch 主控台，然後刪除與 CodeBuild 專案相關聯的 CloudWatch 記錄群組。
6. 導航到 CodeBuild 控制台並通過選擇刪除構建 CodeBuild 項目刪除項目。

使用 CodeBuild Lambda Node.js 創建一個單一頁面的應用程序

創建反應應用程序是一種創建單頁 React 應用程序的方法。下面的 Node.js 示例使用 Node.js 從創建反應應用程序構建源成品，並返回構建工件。

設定來源儲存庫和成品值區

使用紗線創建一個源代碼庫，並創建反應應用程序為您的項目。

若要設定來源儲存庫和成品值區

1. 在您的本機電腦上執行`yarn create react-app <app-name>`以建立一個簡單的 React 應用程式。
2. 將 React 應用程式專案資料夾上傳到支援的來源儲存庫。如需支援的來源類型清單，請參閱ProjectSource。

創建一個 CodeBuild Lambda Node.js 項目

創建一個 AWS CodeBuild Lambda Node.js 項目。

若要建立您 CodeBuild Lambda Node.js 專案

1. 請在以下位置開啟AWS CodeBuild主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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 帳戶中都必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
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：
   - 對於運算，選取 Lambda。
   - 對於執行階段，請選取 Node.js。
   - 對於圖像，請選擇 AWS/ 代碼生成器/亞馬遜鏈接-x86_64 標準：節點 20。
6. 在 Artifacts (成品) 中：
   - 對於類型，選取 Amazon S3。
   - 對於值區名稱，請選取您先前建立的專案成品值區。
   - 對於成品封裝，選取 Zip。
7. 選擇 Create build project (建立建置專案)。

設置項目構建規格

為了建立你的 React 應用程序， CodeBuild 讀取並從 buildspec 文件執行構建命令。

若要設定您的專案建置規格

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，選擇插入構建命令，然後選擇切換到編輯器。
3. 刪除預先填充的構建命令並粘貼到以下 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)。

建立並執行您的 React 應用程式

在 CodeBuild Lambda 上構建 React 應用程序，下載構建成品，然後在本地運行 React 應用程序。

若要建立並執行您的 React 應用程式

1. 選擇 Start build (開始組建)。
2. 建置完成後，瀏覽至 Amazon S3 專案成品儲存貯體並下載 React 應用程式成品。
3. 解壓縮 React 構建工件並`run npm install -g serve && serve -s build`在項目文件夾中。

4. 該`serve`命令將為本地端口上的靜態站點提供服務，並將輸出打印到終端機。您可以訪問終端輸出`Local:`中的本地主機 URL 來查看您的 React 應用程序。

要了解有關如何處理基於 React 的服務器的部署的更多信息，請參閱創建 React 應用程序部署。

清理您的基礎架構

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。

若要清理您的基礎架構

1. 刪除您的項目工件 Amazon S3 存儲桶
2. 導覽至 CloudWatch 主控台，然後刪除與 CodeBuild 專案相關聯的 CloudWatch 記錄群組。
3. 導航到 CodeBuild 控制台並通過選擇刪除構建 CodeBuild 項目刪除項目。

使用 Python 更新 Lamb CodeBuild da 函數配置

下列 Python 範例使用 Boto3 和 CodeBuild Lambda Python 來更新 Lambda 函數的組態。您可以擴充此範例，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。

必要條件

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，並將用 CodeBuild 來更新 Lambda 函數的環境變數。如需透過設定 Lambda 函數的詳細資訊 CodeBuild，請參閱使用 Lambda Java 來部署 AWS SAM L CodeBuild ambda 函數範例或造訪AWS Lambda。

設定您的來源儲存庫

創建一個源代碼存儲庫來存儲您的 Boto3 蟒蛇腳本。

若要設定來源儲存庫

1. 將以下 python 腳本複製到一個名為的新文件中`update_lambda_environment_variables.py`。

```
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 文件上傳到支持的源存儲庫。如需支援的來源類型清單，請參閱ProjectSource。

創建一個 CodeBuild Lambda 蛇項目

創建一 CodeBuild Lambda Python 項目。

若要建立您 CodeBuild Lambda Java 專案

1. 請在以下位置開啟AWS CodeBuild主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
2. 如果顯示 CodeBuild 資訊頁，請選擇 [建立組建專案]。否則，在瀏覽窗格中，展開 [組建]，選擇 [建置專案]，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 帳戶中都必須是唯一的。您還可以包括構建項目的可選描述，以幫助其他用戶了解該項目的用途。
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：

- 對於運算，選取 Lambda。
- 對於執行階段，請選取 Python。
- 對於圖像，選擇 AWS/ 代碼生成器/亞馬遜鏈-x86_64-羔羊標準：蟒蛇 3.12。
- 對於服務角色，請保持選取 [新增服務角色]。記下「角色」名稱。當您稍後在本範例中更新專案的 IAM 許可時，將需要執行此操作。

6. 選擇 Create build project (建立建置專案)。
7. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
8. 在導覽窗格中，選擇 [角色]，然後選取與專案相關聯的服務角色。您可以在 CodeBuild 中找到您的專案角色，方法是選取您的組建專案，然後選擇 [編輯]、[環境]，然後選取 [服務
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基礎結構。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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "UpdateLambdaPermissions",
            "Effect": "Allow",
            "Action": [
                "lambda:UpdateFunctionConfiguration"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

設置項目構建規格

為了更新 Lambda 函數，指令碼會從建置規格讀取環境變數，以尋找 Lambda 函數的名稱、環境變數名稱和環境變數值。

若要設定您的專案建置規格

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，選擇插入構建命令，然後選擇切換到編輯器。
3. 刪除預先填充的構建命令並粘貼到以下 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 Python 自動更新您的 Lambda 函數的組態。

若要更新 Lambda 函數的組態

1. 選擇 Start build (開始組建)。
2. 建置完成後，瀏覽至您的 Lambda 函數。
3. 選取組態，然後選取環境變數。您應該會看到包含索引鍵FEATURE_ENABLED和值的新環境變數true。

清理您的基礎架構

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。

若要清理您的基礎架構

1. 導覽至 CloudWatch 主控台，然後刪除與 CodeBuild 專案相關聯的 CloudWatch 記錄群組。
2. 導航到 CodeBuild 控制台並通過選擇刪除構建 CodeBuild 項目刪除項目。
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。

擴充

如果您想要擴充此範例，以便使用 AWS CodeBuild Python 管理其他AWS資源：

- 更新 Python 腳本以使用 Boto3 修改新的資源。
- 更新與 CodeBuild 專案關聯的 IAM 角色，以取得新資源的許可。
- 將與新資源關聯的任何新環境變量添加到構建規格中。

使用保留容量 AWS CodeBuild

CodeBuild 提供下列運算叢集：

- 按需艦隊
- 預留容量叢集

透過隨選叢集，為您的組建 CodeBuild 提供運算。當構建完成時，機器將被銷毀。隨需叢集是全受管的，並包含自動擴充功能，可處理尖峰的需求。

> **Note**
>
> 依需求不支援視窗伺服器 2022。

CodeBuild 也提供預留容量叢集，其中包含由維護的 Amazon EC2 提供支援的執行個體 CodeBuild。使用預留容量叢集，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，可立即處理構建或測試，並減少構建持續時間。使用保留容量叢集時，您的機器會一直在執行，而且只要佈建機器就會繼續產生成本。

> **Important**
>
> 無論執行執行個體的時間長短，預留容量叢集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

如何開始使用預留容量叢集？

建立保留容量叢集

1. 請登入 AWS Management Console 並開啟 AWS CodeBuild 主控台，網址為 https://console.aws.amazon.com/codesuite/codebuild/home。
2. 在瀏覽窗格中，選擇 [計算叢集]，然後選擇 [建立運算叢集]。
3. 在 [計算叢集名稱] 文字欄位中，輸入叢集的名稱。
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. 從運算下拉式功能表中，選擇機器的運算機器類型。
7. 在容量文字欄位中，輸入叢集中的執行個體數目下限。
8. 在「溢位行為」欄位中，選擇需求超過叢集容量時的行為。如需關於這些選項的詳細資訊，請參閱保留容量叢集屬性。
9. 選擇 [建立運算叢集]。
10. 建立運算叢集之後，建立新 CodeBuild 專案或編輯現有專案。從環境中，選擇佈建模式下的保留容量，然後在叢集名稱下選擇指定的叢集。

最佳實務

使用保留容量叢集時，建議您遵循這些最佳做法。

- 我們建議使用源緩存模式，以幫助通過緩存源來幫助提高構建性能。
- 我們建議使用 Docker 層緩存，以幫助通過緩存現有的 Docker 層來幫助提高構建性能。

是否可以跨多個 CodeBuild 專案共用預留容量叢集？

是的，您可以跨多個專案使用叢集的容量，將其利用率最大化。

哪些區域支援預留容量叢集？

以下支援預留容量叢集 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)。如需有關可用 AWS 區域 位置 CodeBuild 的詳細資訊，請參閱[按地區分類的AWS 服務]。

保留容量叢集屬性

保留容量叢集包含下列內容：

作業系統

作業系統。以下是可用的作業系統：

- Amazon Linux
- Windows Server 2019
- Windows Server 2022

架構

處理器架構。以下是可用的架構：

- x86_64
- 阿姆斯特丹

運算

每個執行個體的運算機器類型。以下是可用的機器類型：

運算類型	環境 computeType 值	環境類型值	記憶體	vCPU	磁碟空間
小臂	BUILD_GENERAL1_SMALL	ARM_CONTAINER	4 GB	2	50 GB
手臂大	BUILD_GENERAL1_LARGE	ARM_CONTAINER	16 GB	8	50 GB
小型	BUILD_GENERAL1_SMALL	LINUX_CONTAINER	3 GB	2	64 GB

運算類型	環境 computeType 值	環境類型值	記憶體	vCPU	磁碟空間
中型	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER	7 GB	4	128 GB
大型	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	15 GB	8	128 GB
大型	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	70 英鎊	36	256 GB
大型	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	145 GB	72	824 GB (SSD)
小型 GPU	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GB	4	220 GB
大型 GPU	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GB	32	50 GB
視窗中	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER	7 GB	4	128 GB
視窗中	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2022_CONTAINER	7 GB	4	128 GB

運算類型	環境 computeType 值	環境類型值	記憶體	vCPU	磁碟空間
大窗戶	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER	15 GB	8	128 GB
大窗戶	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2022_CONTAINER	15 GB	8	128 GB

容量

分配給叢集的初始機器數目，定義可以 parallel 執行的組建數目。

溢位行為

定義組建數目超過叢集容量時的行為。

按需求

溢出構建按 CodeBuild 需運行。

> **Important**
> 如果您選擇將溢位行為設定為隨需，請注意，溢位組建會分開計費，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。這會限制額外的成本，因為沒有配置額外的機器。

保留容量範例 AWS CodeBuild

這些範例可用於在 CodeBuild中試驗保留容量叢集。

主題

使用保留容量範例快取

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保留容量在建置專案中啟用快取。如需詳細資訊，請參閱 在 AWS CodeBuild 中建立快取。

您可以通過在項目設置中指定一個或多個緩存模式開始：

```
Cache:
        Type: LOCAL
        Modes:
          - LOCAL_CUSTOM_CACHE
          - LOCAL_DOCKER_LAYER_CACHE
          - LOCAL_SOURCE_CACHE
```

> **Note**
>
> 確保啟用特權模式以使用 Docker 層緩存。

您的項目構建規格設置應如下所示：

```
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'
```

您可以通過運行具有新項目的構建以種子緩存開始。完成後，您應該使用覆蓋的 buildspec 啟動另一個構建，類似於以下內容：

```
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'
```

保留容量叢集的限制

有些使用案例不支援保留容量叢集，如果這些使用案例對您造成影響，請改用隨需叢集：

- 保留容量叢集不支援批次建置、建置使用率指標或語意版本控制。
- 保留容量叢集不支援 VPC 連線。

如需限制和配額的詳細資訊，請參閱運算叢集。

使用測試報告 AWS CodeBuild

您可以在中建立報告 CodeBuild ，其中包含組建期間執行之測試的詳細資料。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：

- 黃瓜
- 六月 XML（的 .xml）
- 單位 XML（.xml）
- nUnit3 XML (.xml)
- 。TestNG。。XML。
- 視覺工作室 TRX (.trx)
- 可視化工作室 TRX XML（.xml）

> **Note**
>
> 最新支援的版本`cucumber-js`為 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。執行測試之前，無須建立報告群組。如果您指定報表群組名稱，則 CodeBuild 會在執行報表時為您建立報表群組。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，可以將測試結果的原始資料檔案匯出到 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 服務角色用於上傳至 S3 儲存貯體的許可。

主題

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 AWS CodeBuild 中建立建置專案。
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:`區段並指定現有報表群組的 ARN，或報表群組的名稱。

      如果您指定 ARN，則 CodeBuild 會使用該報表群組。

      如果您指定名稱，請使用專案名稱及您指定的名稱，格式為您 CodeBuild 建立報表群組 <project-name>*`report-group-name#< >`*。如果具名報表群組已存在，就 CodeBuild 會使用該報表群組。

   b. 在報告群組下，指定包含測試結果之檔案的位置。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 。

以下是構建規格reports部分的示例：

```
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 在 AWS CodeBuild 中執行建置。
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。

使用報告群組

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
- 更新報告群組
- 指定測試檔案
- 指定測試命令
- 報告群組命名
- 在 AWS CodeBuild 中標記報告群組
- 使用共用報表群組

建立報告群組

您可以使用 CodeBuild 主控台 AWS CLI、或 buildspec 檔案來建立報表群組。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 使用測試報告許可。

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 (AWS 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. 選擇與帳戶中報表群組沒有關聯的報表群組名 AWS 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'
```

報告組名稱也可以通過在 buildspec 中使用環境變量來指定：

```
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)

建立測試報告

1. 開啟AWS CodeBuild主控台位於https://console.aws.amazon.com/codesuite/codebuild/home。
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) 針對 Tags (標籤)，輸入您要支援 AWS 服務使用的任何標籤名稱和值。使用 Add row (新增資料列) 來新增標籤。您最多可新增 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 儲存儲體：
   a. 選擇匯出至 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. （可選）S3 儲存儲體擁有者中，輸入AWS擁有 S3 儲存儲體的賬户標識符。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：

- 預設值AWS受管金鑰來使用AWS 受管金鑰適用於 Amazon S3。如需詳細資訊，請參閱「」客戶受管 CMK中的AWS Key Management Service使用者指南。這是預設加密選項。
- 選擇自訂金鑰以使用您建立和配置的客户託管金鑰。適用於AWS KMS加密金鑰中，輸入您加密金鑰的 ARN。格式為 `arn:aws:kms:<region-id>:<aws-account-id>:key/<key-id>` 。如需詳細資訊，請參閱「」創建 KMS 金鑰中的AWS Key Management Service使用者指南。
- 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**
>
> 在項目中指定的 CodeBuild 服務角色用於上傳至 S3 儲存儲體的許可。

7. 選擇 Create report group (建立報告群組)。

建立報告群組 (CLI)

建立報告羣組

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`：

- 若要讓您的測試報告羣組將原始測試結果檔案匯出至 Amazon S3 儲存儲存儲存儲存儲體，請使用下列 JSON。

```
{
  "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"
      }
    ]
  }
}
```

- Replace*<bucket-name>*取代為您的 Amazon S3 儲存儲存儲體名稱，*<path>*取代為您要匯出檔案的儲存儲存儲體的路徑。
- 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`是可選的，僅當 Amazon S3 儲存儲存儲存儲體由運行版本賬戶以外的賬戶所擁有，才是必需的。
- 使用 `encryptionDisabled` 指定是否要將匯出的檔案加密。如果您將匯出的檔案加密，請輸入您的客戶託管密鑰。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 服務角色用於上傳至 S3 儲存儲存儲存儲存儲體的許可。

3. 執行以下 命令：

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

建立報告群組 (AWS CloudFormation)

使用 AWS CloudFormation 範本建立測試報告

您可以使用 AWS CloudFormation 範本檔案來建立和佈建報表群組。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南。

下列 AWS CloudFormation YAML 範本會建立不匯出原始測試結果檔案的報表群組。

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: NO_EXPORT
```

下列 AWS CloudFormation YAML 範本會建立將原始測試結果檔案匯出至 Amazon S3 儲存貯體的報告群組。

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: S3
```

```
        S3Destination:
          Bucket: my-s3-bucket-name
          Path: path-to-folder-for-exported-files
          Packaging: ZIP
          EncryptionKey: my-KMS-encryption-key
          EncryptionDisabled: false
```

> **Note**
>
> 專案中指定的 CodeBuild 服務角色用於上傳至 S3 儲存貯體的許可。

更新報告群組

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：
  - Amazon S3 AWS 受管金鑰 的一個。
  - 您建立和設定的客戶管理金鑰。

如需詳細資訊，請參閱 資料加密。

如果您使用 AWS CLI 更新報表群組，也可以更新或新增標記。如需詳細資訊，請參閱 在 AWS CodeBuild 中標記報告群組。

> **Note**
>
> 專案中指定的 CodeBuild 服務角色用於上傳至 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)

更新報告群組

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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. 選擇或清除 Backup 到 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 AWS 受管金鑰 於 Amazon S3 的預設 AWS 受管金鑰。如需詳細資訊，請參閱AWS Key Management Service 使用指南中的客戶管理 CMK。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管理金鑰。對於AWS KMS 加密金鑰，請輸入加密金鑰的 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)

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":"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，請輸入您的客戶管理金鑰。
   - 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
```

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。可選discard-paths部分指定是否捨棄上傳至 Amazon S3 儲存貯體的測試結果檔案路徑。

```
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。

報告群組命名

當您使用 AWS CLI 或主 AWS CodeBuild 控台建立報表群組時，您可以指定報表群組的名稱。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 CodeBuild 想建立新的報表群組，請在組建專案的 buildspec 檔案中指定報表群組的 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，如果您的 AWS 帳戶中沒有名稱的報表群組`new-report-group`，並且在名為的組建專案中指定它`test-project`，則組建執行會以該名稱建立新的報表群組`test-project-new-report-group`。

在 AWS CodeBuild 中標記報告群組

標籤是一種自訂屬性標籤，可由您或 AWS 指派給 AWS 資源。每個 AWS 標籤都有兩個部分：

- 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 或 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。

這些合稱為金鑰值組。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，請參閱標籤。

標籤可協助您識別和整理 AWS 資源。許多 AWS 服務支援標記，因此您可以對來自不同服務的資源指派相同的標籤，指出資源是相關的。例如，您可以將指派給 Amazon S3 儲存儲存儲體的相同標籤，指派給 CodeBuild 報告 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，主要資源是報告羣組和專案。您可以使用 CodeBuild 主控台、AWS CLI、CodeBuild API 或AWS軟體開發套件來新增、管理和移除報告羣組的標籤。除了使用標籤來識別、整理和追蹤您的報告羣組，您也可以使用 IAM 政策中的標籤，以協助控制誰可以檢視並與您的報告羣組互動。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 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
- 編輯報表群組的標籤
- 從報告群組移除標籤

將標籤新增至報告群組

新增標籤到報告群組可協助您識別和整理您的 AWS 資源，並管理這些資源的存取權。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 標籤。當您擁有標籤後，可以根據這些標籤建立 IAM 政策，以管理報告 您可以使用 CodeBuild 主控台或AWS CLI將標籤新增至報告

> **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。將標籤新增到報告羣組之前，務必檢任何可能會使用標籤控制存取資源 (例如報告 group) 的 IAM 政策。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 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 控台，將一或多個標籤新增到 CodeBuild 報告

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 Settings (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

標籤可協助您辨識和整理您的 AWS 資源和管理存取權。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)

您可以使用代碼建立主控台以檢視與 CodeBuild 報告

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 Settings (設定)。

檢視報告群組的標籤 (AWS CLI)

依照下列步驟使用 AWS CLI 以檢視報告群組的 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，找出報告群組的 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。您也可以變更金鑰名稱，等於移除目前的標籤，並新增一個不同的新的名稱和相同的值作為其他金鑰。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 標籤。

> **Important**
>
> 編輯報告群組的謙可能會影響該報告群組的存取權。編輯報告羣組的名稱 (金鑰) 或標籤值之前，務必檢任何可能會使用標籤金鑰或值來控制存取資源的 IAM 政策，例如報告 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)

您可以使用代碼建立主控台以編輯與 CodeBuild 報告

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 Settings (設定)。
4. 選擇 Edit (編輯)。
5. 執行下列任一步驟：
   - 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。移除標籤不會從其他 AWS 資源刪除與該標籤相關聯的標籤。

> **Important**
>
> 移除報表群組的標籤可能會影響該報表群組的存取權。從報告羣組移除標籤之前，務必檢任何可能會使用標籤金鑰或值來控制存取資源 (例如報告 group) 的 IAM 政策。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 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 主控台移除標籤和 CodeBuild 報告

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 Settings (設定)。
4. 選擇 Edit (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)

請依照以下步驟，使用AWS CLI從 CodeBuild 報告 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 group，所有標籤關聯會從已刪除的報告 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: []"
```

使用共用報表群組

報告群組共用功能可讓多個 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。

內容

共用報告群組的先決條件

若要共用報告群組，您的 AWS 帳戶必須擁有該報告群組。您無法將已共用給您的報告群組再共用出去。

存取與您共用之報告群組的先決條件

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 使用以身分識別為基礎的原則 AWS CodeBuild。

相關 服務

報告群組共用功能與 AWS Resource Access Manager (AWS RAM) 服務整合。該服務可讓您將您的 AWS 資源分享給任何 AWS 帳戶或透過 AWS Organizations 來共用。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。消費者可以是個別的 AWS 帳戶，或 AWS Organizations 中的組織單位或 AWS Organizations 中的整個組織。

如需詳細資訊，請參閱《AWS RAM 使用者指南》。

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。消費者可以使用 AWS CLI 檢視報告群組、其報告，以及各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羣組或其報告。
- 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 主控台將報告羣組新增至現有的資源共用。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 主控台AWS RAM主控台或AWS CLI共用您擁有的報告羣組。

共用您擁有的報告羣組 (CodeBuild 主控台)

1. 開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。

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「」建立資源共享中的AWS RAM使用者指南。

共用您擁有的報告群組 (AWS RAM 主控台)

請參閱建立資源共享中的AWS RAM使用者指南。

共用您擁有的報告群組 (AWS RAM 命令)

使用 create-resource-share 命令。

共用您擁有的報告羣組 (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。

```
{
   "Version":"2012-10-17",
   "Statement":[{
     "Effect":"Allow",
     "Principal":{
       "AWS":"consumer-aws-account-id-or-user"
     },
     "Action":[
       "codebuild:BatchGetReportGroups",
       "codebuild:BatchGetReports",
       "codebuild:ListReportsForReportGroup",
       "codebuild:DescribeTestCases"],
     "Resource":"arn-of-report-group-to-share"
   }]
 }
```

2. 使用報告群組 ARN 和識別符更新 policy.json，以與之共用。下列範例會將唯讀存取權授予以使用 ARN 的報告arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group添加到愛麗絲和根用户AWS帳户已被確認為是否已經確定。

```
{
   "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
```

取消共享已共用的報告存取

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。如果您取消共享報告群組，您之前與之共享的任何 AWS 帳戶或使用者都無法存取該報告群組、其報告或報告中的測試案例結果。

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 執行此作業。

取消共享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。

取消共享您擁有的共用報告群組 (AWS RAM 命令)

使用 disassociate-resource-share 命令。

取消共享您擁有的共用報告羣組 (CodeBuild)

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

擁有者和消費者可以使用 AWS CLI 識別已共用的報告群組。

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。

使用報告

報告包含為一個報告群組指定之測試案例的結果。測試報告會在執行建置專案期間建立。您可以指定報告群組、測試案例檔案和命令，以便在其 buildspec 檔案中執行測試案例。每次測試案例執行時，就會在報告群組中建立新的測試報告。

測試報告會在建立 30 天後過期。雖然無法檢視過期的測試報告，但您可以將測試結果匯出至 S3 儲存貯體中的原始測試結果檔案。匯出的原始測試檔案不會過期。如需詳細資訊，請參閱 更新報告群組。

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 使用測試報告許可。
  - 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。如果運行超過 500 個測試用例，則 CodeBuild 優先考慮具有狀態的測試FAILED並截斷測試用例結果。

使用測試報告許可

本主題說明與測試報告相關的重要許可資訊。

主題

建立測試報告的角色

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。這些權限包含在預先定義的 AWS 受管理策略中。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權限。

> **Note**
>
> BatchPutTestCases、CreateReportUpdateReport、和不BatchPutCodeCoverages是公開權限。您無法針對這些權限呼叫對應的 AWS CLI 命令或 SDK 方法。

為確保您擁有這些許可，您可以將以下政策附加到 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 服務角色用於上傳至 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 作業的權限：

- BatchGetReportGroups
- BatchGetReports
- CreateReportGroup
- DeleteReportGroup
- DeleteReport
- DescribeTestCases
- ListReportGroups
- ListReports

- ListReportsForReportGroup
- UpdateReportGroup

如需詳細資訊，請參閱 AWS CodeBuild 權限參考。

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

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。您可以檢視依組建執行、報表群組或您的 AWS 帳戶分組的測試報告。選擇主控台中的測試報告，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。您無法在中檢視過期的報告 CodeBuild。

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. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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

在您的 AWS 帳戶中檢視測試報告

1. 請在以下位置開啟 AWS CodeBuild 主控台。 https://console.aws.amazon.com/codesuite/codebuild/home
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。

使用測試框架測試報告

本節中的主題示範如何 AWS CodeBuild 針對各種測試架構設定中的測試報告。

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

下列程序示範如何設置AWS CodeBuild使用JasmineBDD 試框架。

此程序需要下列先決條件：

- 您具備現有的 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。所以此`test`腳本確保 Jasmine 在npm test執行。

```
{
  "scripts": {
    "test": "npx jasmine"
  }
}
```

CodeBuild 支援以下 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

下面的過程演示了如何設置AWS CodeBuild與Jest 測試框架。

此程序需要下列先決條件：

- 您具備現有的 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 使用此軟件包，在JunitXml格式。

```
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。所以此test指令碼確保時，會呼叫 Jest。npm test請執行。

```
{
  "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

下面的過程演示了如何設置 AWS CodeBuild 與 pytest 測試框架測試報告。

此程序需要下列先決條件：

- 您具備現 CodeBuild 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

下面的過程演示了如何設置AWS CodeBuild與RSpec 測試框架。

此程序需要下列先決條件：

- 您具備現有的 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
```

代碼覆蓋率報告

CodeBuild 允許您為測試生成代碼覆蓋率報告。提供下列程式碼涵蓋範圍報告：

線路覆蓋

線路覆蓋率測量您的測試涵蓋的陳述數量。語句是一個單一的指令，不包括註釋或條件。

line coverage = (total lines covered)/(total number of lines)

分公司覆蓋

分支覆蓋度量測您的測試涵蓋了控制結構的每個可能分支（例如if或case語句）中的分支的數量。

branch coverage = (total branches covered)/(total number of branches)

支援下列程式碼涵蓋範圍報表檔案格式：

- JaCoCo XML
- SimpleCov JSON[1]
- 三叶草
- XML 科技
- 科夫資訊

[1] CodeBuild 接受簡單生成的 JSON 代碼覆蓋率報告，而不是簡單的 JSON 代碼覆蓋率。

建立程式碼涵蓋範圍報告

若要建立程式碼涵蓋範圍報告，您可以執行組建專案，該專案在 buildspec 檔案中設定至少有一個程式碼涵蓋範圍報表群組。 CodeBuild 將解釋代碼覆蓋結果，並為運行提供代碼覆蓋率報告。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 AWS CodeBuild 中建立建置專案。
2. 使用測試報告信息配置項目的 buildspec 文件：

   a. 新增`reports:`區段並指定報表群組的名稱。 CodeBuild 使用您的專案名稱和您以格式`project-name`指定的名稱，為您建立報表群組`report-group-name-in-buildspec`。如果您已有要使用的報告群組，請指定其 ARN。如果您使用名稱而不是 ARN，則 CodeBuild 會建立新的報表群組。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之檔案的位置。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。每次執行組建專案時，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 結果文件生成代碼覆蓋率報告的示例`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 在 AWS CodeBuild 中執行建置。
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。如需更多詳細資訊，請參閱 檢視建置的測試報告。

報告自動探索

透過自動探索功能，可在建置階段完成後 CodeBuild 搜尋所有組建檔案、搜尋任何支援的報表檔案類型，以及自動建立新的測試和程式碼涵蓋範圍報告群組和報告。針對任何探索到的報表類型，CodeBuild 建立具有下列模式的新報告群組：

```
<project-name>-<report-file-format>-AutoDiscovered
```

> **Note**
>
> 如果探索到的報告檔案具有相同的格式類型，則會將它們置於相同的報告群組或報告中。

報表自動探索是由您的專案環境變數所設定：

CODEBUILD_CONFIG_AUTO_DISCOVER

此變數會決定是否在建置期間停用報表自動探索。根據預設，所有組建都會啟用報表自動探索功能。若要停用此功能，請CODEBUILD_CONFIG_AUTO_DISCOVER將設定為false。

CODEBUILD_CONFIG_AUTO_DISCOVER_DIR

(選擇性) 此變數決定 CodeBuild 搜尋潛在報表檔案的位置。請注意，依預設，會在中 CodeBuild **/*搜尋。

這些環境變量可以在構建階段進行修改。例如，如果您只想為 main git 分支上的構建啟用報表自動發現，則可以在構建過程中檢查 git 分支，如果構建不在main分支上，則CODEBUILD_CONFIG_AUTO_DISCOVER將其設置為 false。您可以使用主控台或使用專案環境變數來停用報表自動探索。

主題

使用主控台設定報告自動探索

使用主控台設定報表自動探索

1. 建立組建專案或選擇要編輯的建置專案。如需詳細資訊，請參閱 在 AWS CodeBuild 中建立建置專案 或 在 AWS CodeBuild 中變更建置專案的設定。
2. 在環境中，選取其他組態。
3. 若要停用報表自動探索，請在 [報表自動探索] 中選取 [停用報表自動探索]。
4. (選擇性) 在自動探索目錄-選用中，輸入目錄模式 CodeBuild 以搜尋支援的報表格式檔案。請注意， CodeBuild **/*依預設會搜尋。

使用專案環境變數設定報表自動探索

若要使用專案環境變數設定報表自動探索

1. 建立組建專案或選擇要編輯的建置專案。如需詳細資訊，請參閱 在 AWS CodeBuild 中建立建置專案 或 在 AWS CodeBuild 中變更建置專案的設定。
2. 在環境變數中，執行下列動作：

    a. 若要停用報表自動探索，請在名稱中輸入**CODEBUILD_CONFIG_AUTO_DISCOVER**和做為值。**false**這會停用報表自動探索。

    b. (選擇性) 在名稱中輸入**CODEBUILD_CONFIG_AUTO_DISCOVER_DIR**和做為值，輸入 CodeBuild 應在其中搜尋支援報表格式檔案的目錄。例如，`output/*xml`搜尋`output`目錄中的`.xml`檔案

AWS CodeBuild 中的記錄和監控

監控是維護 AWS CodeBuild 及您 AWS 解決方案可靠性、可用性和效能的重要部分。您應該收集監控資料來自AWS解決方案，以便在發生多點故障的情況下，更輕鬆地偵錯。AWS提供以下工具用於監控 CodeBuild 資源和構建及回應潛在事件：

主題

- 使用 AWS CloudTrail 記錄 AWS CodeBuild API 呼叫
- 監控 AWS CodeBuild

使用 AWS CloudTrail 記錄 AWS CodeBuild API 呼叫

AWS CodeBuild與整合AWS CloudTrail，這項服務可提供由使用者、角色或中AWS服務所採取之動作的記錄 CodeBuild。 CloudTrail 擷取 CodeBuild 為事件的所有 API 呼叫，包括來自 CodeBuild 主控台的呼叫以及來自對 API 發出的程 CodeBuild 式碼呼叫。如果建立線索，則可將事件 (包括的 CloudTrail 事件) 持續交付到 S3 儲存貯體 CodeBuild 如果您不設定追蹤記錄，仍然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新的事件。使用由收集的資訊 CloudTrail，您就可以判斷傳送至的請求 CodeBuild、提出請求的 IP 地址、提出請求的對象、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail用者指南。

AWS CodeBuild中的資訊 CloudTrail

CloudTrail 當您建立AWS帳戶時，系統會在您的帳戶中啟用。當中發生活動時 CodeBuild，系統便會將該活動記錄到事件中，其他AWS服務 CloudTrail 事件則記錄於 Event history (Event Trail)。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱AWS CloudTrail使用指南中的檢視具有 CloudTrail 事件歷程記錄的事件。

如需您AWS帳戶中正在進行事件的記錄 (包含的事件) CodeBuild，請建立線索。線索可讓 CloudTrail 將日誌檔案傳遞到 S3 儲存貯體。根據預設，當您在主控台建立權杖時，權杖會套用到所有區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 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 支援的服務與整合
- 設定的 Amazon SNS 通知 CloudTrail
- 接收多個區域的 CloudTrail 日誌檔案及接收多個帳戶的 CloudTrail 日誌檔案

所有 CodeBuild 動作均由「API 參考」記錄 CloudTrail 並記錄在「CodeBuild API 參考」中。例如，呼叫`CreateProject` (在AWS CLI、中`create-project`)、`StartBuild` (在中AWS CLI) 和`UpdateProject` (在中`start-project`update-project) 動作會在AWS CLI CloudTrail 記錄檔中產生項目。

每一筆事件或日誌項目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。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱《AWS CloudTrail使用者指南》中的「使用者CloudTrail userIdentity」元素。

了解 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，記 CodeBuild 錄檔中會隱藏下列項目：
>
> - AWS 存取金鑰 ID。如需詳細資訊，請參閱使用AWS Identity and Access Management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 使用指南中的 Systems Manager 參數存放區和系統管理員參數存放主控台逐步解說。
> - 使用指定的字串AWS Secrets Manager。如需詳細資訊，請參閱金鑰管理。

以下範例顯示的 CloudTrail 日誌項目會示範在中建立組建專案 CodeBuild。

```
{
  "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"
}
```

監控 AWS CodeBuild

您可以使用 Amazon CloudWatch 來監看組建，在發現錯誤時回報，並適時自動採取動作。您可以在兩個層級監控組建：

項目級別

這些指標用於指定專案中的所有組建。若要查看專案的指標，請指定ProjectName中的CloudWatch。

AWS帳户級別

這些指標用於帳户中的所有組建。若要查看AWS帳户級別，請勿在 CloudWatch 中輸入維度。構建資源利用率衡量指標在AWS帳户級別。

CloudWatch 指標會顯示組建隨著時間的行為。例如，您可以監控：

- 隨著時間在組建專案或 AWS 帳戶中嘗試了多少個組建。
- 隨著時間在組建專案或 AWS 帳戶中成功了多少個組建。
- 隨著時間在組建專案或 AWS 帳戶中失敗了多少個組建。
- CodeBuild 用於運行組建耗費了多少時間。AWS隨著時間的帳號。
- 構建生成或整個構建項目的資源利用率。構建資源使用率指標包含 CPU、記憶體和存儲使用率這類指標。

如需詳細資訊，請參閱 監控 CodeBuild 指標。

CodeBuild 指標

以下指標可通過AWS帳户或組建專案。

構建

測量組建的 BUILD 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

建置數

測量已觸發的組建數目。

個單位: 計數

有效的 CloudWatch 統計資訊：總和

下載源持續時間

測量組建的 DOWNLOAD_SOURCE 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

Duration

測量隨著時間所有組建的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

FailedBuilds

測量因為客户端錯誤或超時而失敗的組建數量。

個單位: 計數

有效的 CloudWatch 統計資訊：總和

最終確定持續時間

測量組建的 FINALIZING 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

安裝持續時間

測量組建的 INSTALL 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

構建後

測量組建的 POST_BUILD 階段的持續時間

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

預構建

測量組建的 PRE_BUILD 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

配置持續時間

測量組建的 PROVISIONING 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

隊列持續時間

測量組建的 QUEUED 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

提交

測量組建的 SUBMITTED 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

SucceededBuilds

測量成功的組建數目。

個單位: 計數

有效的 CloudWatch 統計資訊：總和

上載工件持續時間

測量組建的 UPLOAD_ARTIFACTS 階段的持續時間。

個單位: 秒鐘

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

CodeBuild CloudWatch 資源利用率指標

> **Note**
>
> 下列區域才能取得 CodeBuild 資源使用率指標：
>
> - 亞太區域 (東京) 區域
> - Asia Pacific (Seoul) Region
> - Asia Pacific (Mumbai) Region
> - Asia Pacific (Singapore) Region
> - Asia Pacific (Sydney) Region
> - Canada (Central) Region
> - 歐洲區域 (法蘭克福)
> - Europe (Ireland) Region

- Europe (London) Region
- 歐洲 (巴黎) 區域
- South America (São Paulo) Region
- US East (N. Virginia) Region
- US East (Ohio) Region
- US West (N. California) Region
- 美國西部區域 (奧勒岡)

可以跟蹤以下資源利用率指標。

CPU 使用

組建容器使用的已配置處理單位數。

個單位: CPU 單位

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

CPU 使用率百分比

構建容器使用的已分配處理的百分比。

個單位: 百分比

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

MemoryUtilized

組建容器使用的記憶體數。

個單位: MB

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

記憶佔用百分比

組建容器使用的已配置記憶體百分比。

個單位: 百分比

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

StorageReadBytes

構建容器使用的存儲讀取速度。

個單位: 位位/秒

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

StorageWriteBytes

構建容器使用的存儲寫入速度。

個單位: 位位/秒

有效的 CloudWatch 統計資訊：平均值 (建議)、最大值

CodeBuild 維度

CodeBuild 提供了以下 CloudWatch 指標維度。如果未指定這些指標，則指標用於當前AWS帳户。

BuildId、內部編號、ProjectName

為構建標識符、內部版本號和項目名稱提供了指標。

ProjectName

針對專案名稱提供了指標。

CodeBuild 警示

您可以使用 CloudWatch 控台來根據 CodeBuild 指標建立警示，使得您可以在組建發生錯誤時加以應對。對警示來說，最有用的兩個指標是：

- `FailedBuild`。 您可以建立警示，此警示會在預定秒數內偵測到特定數目的失敗組建時觸發。在 CloudWatch 中，您可以指定將觸發警示的秒數和失敗組建的數目。
- `Duration`。 您可以建立當組建耗費的時間較預期長時觸發的警示。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 指標建立警示的詳細資訊，請參閱使用 CloudWatch 警示監控組建。如需警示的詳細資訊，請參閱建立 Amazon CloudWatch 警示中的Amazon CloudWatch 使用者指南。

監控 CodeBuild 指標

AWS CodeBuild代表您自動監視功能，並透過 Amazon CloudWatch 回報指標。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 主控台來監視 CodeBuild 指標。下列程序顯示如何存取指標。

主題

存取建置指標 (CodeBuild 主控台)

> **Note**
>
> CodeBuild 主控台中，您無法自訂指標或用來顯示它們的圖形。如果您想要自訂顯示，請使用 Amazon CloudWatch 主控台來檢視您的構建指標。

帳户層級指標

若要存取AWS帳户層級指標

1. 登入AWS Management Console，然後開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 在導覽窗格中，選擇 Account metrics (帳戶指標)。

項目層級指標

存取專案層級指標

1. 登入AWS Management Console，然後開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
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 Metrics (指標) 標籤。

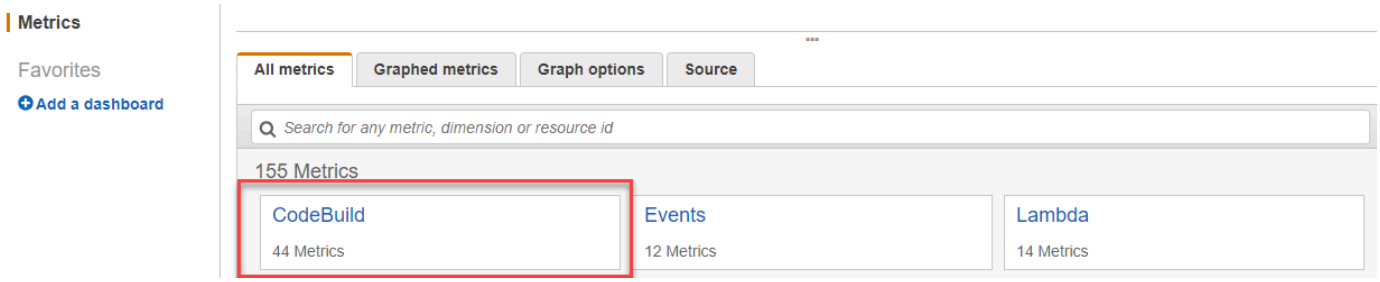
存取建置指標 (Amazon CloudWatch 主控台)

使用 CloudWatch 主控台來自訂指標以及用來顯示它們的圖形。

帳戶層級指標

存取帳戶層級指標

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (指標)。
3. 在 All metrics (所有指標) 標籤上，選擇 CodeBuild。

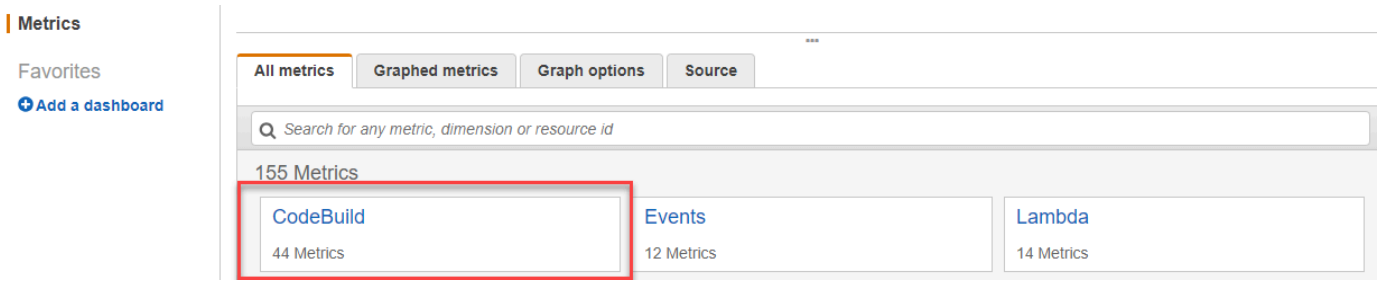


4. 選擇 Account Metrics (帳戶指標)。
5. 選擇一或多個專案和指標。針對每個專案，您可以選擇 SucceededBuilds、FailedBuilds、Builds 和 Duration 指標。所有選取的專案和指標組合都會顯示在頁面的圖形中。

項目層級指標

存取專案層級指標

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (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Statistic欄中，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「」圖形指標和檢視可用的指標中的Amazon CloudWatch 使用者指南。

監視 CodeBuild 資源利用率指標

AWS CodeBuild代您監控組建置資源使用率，並透過 Amazon CloudWatch 報告指標。這些指標包括 CPU、記憶體和存儲使用率。

> **Note**
>
> CodeBuild 資源利用率度量僅記錄運行一分鐘以上的版本。

您可以使用 CodeBuild 控制台或 CloudWatch 控制台來監視 CodeBuild 的資源利用率指標。

> **Note**
>
> 下列區域才能取得 CodeBuild 資源使用率指標：
>
> - 亞太區域 (東京) 區域
> - Asia Pacific (Seoul) Region
> - Asia Pacific (Mumbai) Region
> - Asia Pacific (Singapore) Region
> - Asia Pacific (Sydney) Region
> - Canada (Central) Region
> - 歐洲區域 (法蘭克福)
> - Europe (Ireland) Region
> - Europe (London) Region
> - 歐洲 (巴黎) 區域
> - South America (São Paulo) Region

- US East (N. Virginia) Region
- US East (Ohio) Region
- US West (N. California) Region
- 美國西部區域 (奧勒岡)

下列程序顯示如何存取資源使用率指標。

主題

- 訪問資源利用率指標（CodeBuild 控制台）
- 訪問資源利用率指標（Amazon CloudWatch 控制台）

訪問資源利用率指標（CodeBuild 控制台）

> **Note**
>
> 您無法自訂指標或用來在 CodeBuild 主控台中顯示它們的圖形。如果想要自訂顯示，請使用 Amazon CloudWatch 主控台來檢視組建指標。

專案層級的資源使用率指標

存取專案層級的資源使用率指標

1. 登入AWS Management Console並開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 在導覽窗格中，選擇 Build projects (建置專案)。
3. 在組建專案清單中，於名稱欄中，選擇您要檢視使用率指標的專案。
4. 選擇 Metrics (指標) 標籤。資源利用率度量顯示在資源使用率指標區段。
5. 若要在 CloudWatch 主控台中檢視專案層級的資源使用率指標，請選擇在 CloudWatch 中檢視中的資源使用率指標區段。

組建層級的資源使用率指標

存取組建層級的資源使用率指標

1. 登入AWS Management Console並開啟AWS CodeBuild主控台https://console.aws.amazon.com/codesuite/codebuild/home。
2. 在導覽窗格中，選擇 Build history (組建歷史記錄)。
3. 在構建列表中，在建置運行欄中，選擇您要檢視其使用率指標的組建。
4. 選擇資源使用率標籤。
5. 若要在 CloudWatch 主控台中檢視組建置層級的資源使用率指標，請選擇在 CloudWatch 中檢視中的資源使用率指標區段。

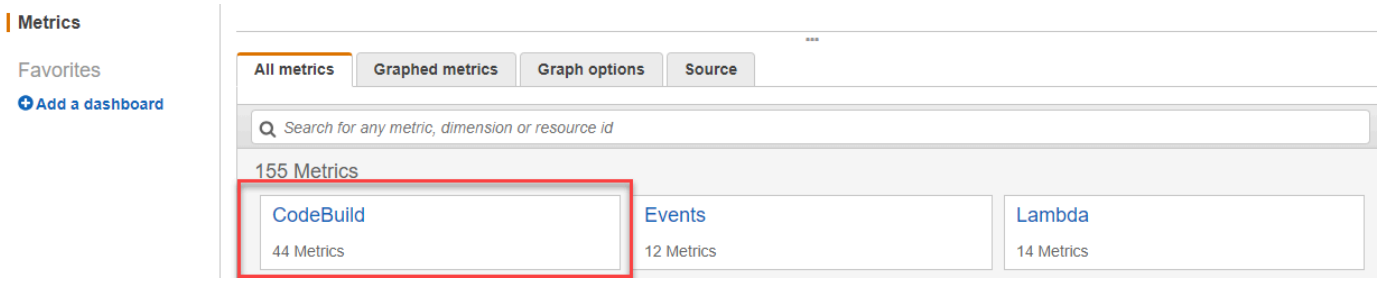
訪問資源利用率指標（Amazon CloudWatch 控制台）

Amazon CloudWatch 控制台可用於訪問 CodeBuild 資源利用率指標。

專案層級的資源使用率指標

存取專案層級的資源使用率指標

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (指標)。
3. 在 All metrics (所有指標) 標籤上，選擇 CodeBuild。



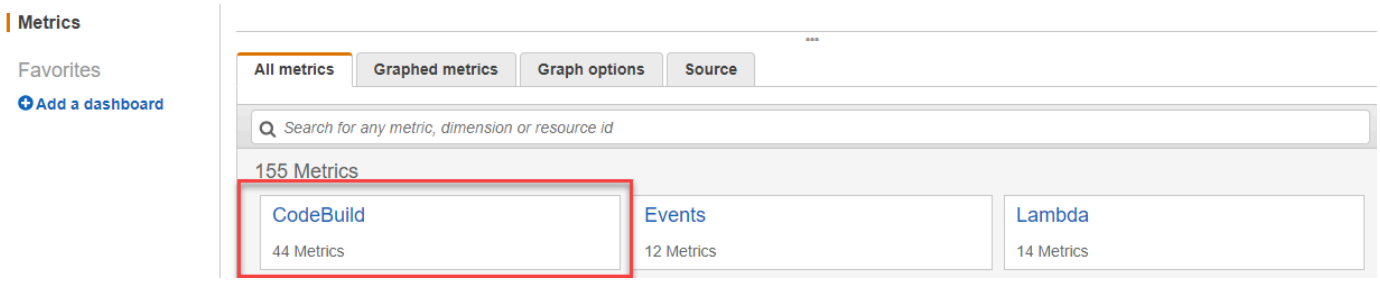
4. 選擇 By Project (依專案)。
5. 選擇要添加至圖形的一或多個專案和指標組合。所有選取的專案和指標組合都會顯示在頁面的圖形中。
6. （選用）您可以從建立指標圖表標籤。例如，從下拉式清單中Statistic欄中，您可以選擇要顯示的不同統計資料。或者，從下拉式功能表中，於 Period (期間) 欄中，您可以選擇用來監控指標的不同時間期間。

如需詳細資訊，請參閱「」建立指標圖表和檢視可用的指標中的Amazon CloudWatch 使用者指南。

組建層級的資源使用率指標

存取組建層級的資源使用率指標

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (指標)。
3. 在 All metrics (所有指標) 標籤上，選擇 CodeBuild。



4. 選擇BuildId、內部編號、ProjectName。
5. 選擇要添加至圖形的一或多個組合。所有選取的建置和指標組合都會顯示在頁面的圖形中。
6. （選用）您可以從建立指標圖表標籤。例如，從下拉式清單中Statistic欄中，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「」建立指標圖表和檢視可用的指標中的Amazon CloudWatch 使用者指南。

使用 CloudWatch 警示監控組建

您可以建立 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。您可以使用 CloudWatch 警示功能，指定 CloudWatch 支援的任何動作。例如，您可以指定當帳戶中在 15 分鐘內有超過三個組建失敗時傳送 Amazon SNS 通知。

建立 CodeBuild 指標的 CloudWatch 警示

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。

2. 在導覽窗格中，選擇 Alarms (警示)。
3. 選擇 Create Alarm (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。如需觸發警示時的 Amazon SNS 通知的設定詳細資訊，請參設定 Amazon SNS 通知中的Amazon SNS 開發人員指南。
8. 選擇 Create Alarm (建立警示)。

中的安全性 AWS CodeBuild

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全性和合規性是您 AWS 與您之間的共同責任。這個共用模型有助於減輕 AWS 營運負擔：操作、管理和控制從主機作業系統和虛擬化層到服務設施的實體安全性等各種元件。您需承擔相關責任並管理訪客作業系統 (包括更新與安全性修補程式) 和其他相關應用程式軟體。您也必須負責 AWS 所提供安全群組防火牆的設定。您的責任取決於您使用的服務、這些服務與 IT 環境的整合，以及適用的法律和法規。因此，您應該仔細考慮您的組織使用的服務。如需詳細資訊，請參閱共同責任模式。

若要瞭解如何保護 CodeBuild 資源的安全，請參閱下列主題。

主題

- 資料保護 AWS CodeBuild
- 身分識別與存取管理 AWS CodeBuild
- 符合性驗證 AWS CodeBuild
- 韌性 AWS CodeBuild
- 基礎架構安全性 AWS CodeBuild
- 存取您的來源供應商 CodeBuild
- 預防跨服務混淆代理人

資料保護 AWS CodeBuild

AWS 共用責任模型適用於中的資料保護 AWS CodeBuild。如此模型中所述， AWS 負責保護執行所有 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。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。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 資源進行通訊。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。

- 使用 AWS 加密解決方案以及其中的所有默認安全控制 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。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，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) 140-2 概觀。

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。這包括當您使用主控台、API CodeBuild 或 AWS SDK 時 AWS 服務 使用或其他使用時。 AWS CLI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，記 CodeBuild 錄檔中會隱藏下列項目：

- 在 CodeBuild 專案環境變數或 buildspec `env/parameter-store` 區段中使用參數存放區指定的字串。有關詳情，請參閱 Amazon EC2 Systems Manager 使用者指南中的 Systems Manager 參數存放區和系統管理員參數存放區主控台逐步解說。
- AWS Secrets Manager 在 CodeBuild 項目環境變量或 buildspec `env/secrets-manager` 部分中使用指定的字符串。如需詳細資訊，請參閱 金鑰管理。

如需有關資料保護的詳細資訊，請參閱AWS 安全部落格上的AWS 共同責任模型和 GDPR 部落格文章。

主題

資料加密

加密是安 CodeBuild 全性的重要組成部分。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-預設情況下，建置成品 (例如快取、記錄、匯出的原始測試報告資料檔案和組建結果) 都會使 AWS 受管金鑰用。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。如需使用者指南中建立 KMS 金AWS 鑰和金鑰管理服務概念的詳細AWS Key Management Service 資訊。

- 您可以將 CodeBuild 用來加密組建輸出成品之 AWS KMS 金鑰的識別碼儲存在CODEBUILD_KMS_KEY_ID環境變數中。如需更多資訊，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管理的金鑰。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，建置叢集的 Amazon 彈性區塊存放區磁碟區會使用加密 AWS 受管金鑰。

- 傳輸中的資料加密-客戶之間以及 CodeBuild CodeBuild 及其下游相依性之間的所有通訊都會使用使用「簽名版本 4」簽署程序簽署的 TLS 連線來保護。所有 CodeBuild 端點都使用由管理的 SHA-256 憑證 AWS Private Certificate Authority。如需詳細資訊，請參閱簽章版本 4 簽署程序和什麼是 ACM PCA。
- 組建成品加密-與建置專案相關聯的 CodeBuild 服務角色需要存取 KMS 金鑰，才能加密其組建輸出成品。根據預設， CodeBuild 會在您的 AWS 帳戶中使用適用 AWS 受管金鑰 於 Amazon S3 的。如果您不想使用此功能 AWS 受管金鑰，則必須建立並設定客戶管理的金鑰。如需詳細資訊，請參閱建立客戶受管金鑰開AWS KMS 發人員指南中的和建立金鑰。

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。將您的加密金鑰儲存在中 AWS Secrets Manager，然後授與組建專案相關聯的 CodeBuild 服務角色權限，以便從 Secrets Manager 帳戶取得加密金鑰。如需詳細資訊，請參閱建立並設定客戶受管金鑰 CodeBuild、在 AWS CodeBuild 中建立建置專案、在 AWS CodeBuild 中執行建置和教學：儲存並擷取秘密。

在構建命令中使用CODEBUILD_KMS_KEY_ID環境變量來獲取 AWS KMS 密鑰標識符。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 將認證保護到私人登錄，該登錄會儲存用於執行階段環境的 Docker 映像檔。如需詳細資訊，請參閱 帶有 AWS Secrets Manager 示例的私人註冊表 CodeBuild。

流量隱私權

您可以透過設定 CodeBuild 為使用介面 VPC 端點來改善組建的安全性。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。它也不需要配置 PrivateLink，雖然這是建議的。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存取 AWS 服務。

身分識別與存取管理 AWS CodeBuild

存取需 AWS CodeBuild 要認證。這些登入資料必須具有存取 AWS 資源的許可，例如在 S3 儲存貯體中存放和擷取建置成品，以及檢視用於組建的 Amazon CloudWatch 日誌。以下各節說明如何使用 AWS Identity and Access Management(IAM) 以及協 CodeBuild 助確保資源存取安全：

管理資 AWS CodeBuild 源存取權限概觀

每個 AWS 資源都由一個 AWS 帳號擁有，建立或存取資源的權限由權限原則控制。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 資源與營運
- 了解資源所有權
- 管理資源存取
- 指定政策元素：動作、效果和委託人

AWS CodeBuild 資源與營運

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 資源名稱 (ARN) 和 AWS 服務命名空間。Amazon Web Services 一般參考

資源類型	ARN 格式
建置專案	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:project/` *project-name*

資源類型	ARN 格式
組建	arn:aws:codebuild: *region-ID* :*account-ID* :build/*build-ID*
報告群組	arn:aws:codebuild: *region-ID* :*account-ID* :report-group/ *report-group-name*
報告	arn:aws:codebuild: *region-ID* :*account-ID* :report/*report-ID*
所有 CodeBuild 資源	arn:aws:codebuild:*
指定 AWS 區域中指定帳號擁有的所有 CodeBuild 資源	arn:aws:codebuild: *region-ID* :*account-ID* :*

> **Note**
>
> 大多數 AWS 服務會將冒號 (:) 或正斜線 (/) 視為 ARN 中的相同字元。但是，在資源模式和規則中 CodeBuild 使用完全匹配。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，您可以使用其 ARN 在陳述式中指示特定的建置專案 (*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": "*"
```

某些 CodeBuild API 動作可接受多個資源 (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 提供了一組操作來處理資 CodeBuild 源。如需清單，請參閱AWS CodeBuild 權限參考。

了解資源所有權

AWS 帳號擁有在帳號中建立的資源，無論是誰建立資源。具體而言，資源擁有者是驗證資源建立請求的主體實體 (即根帳戶、使用者或 IAM 角色) 的帳戶。 AWS 下列範例說明其如何運作：

- 如果您使用帳號的根帳 AWS 號認證來建立規則，您的 AWS 帳號就是資 CodeBuild 源的擁有者。
- 如果您在 AWS 帳戶中建立使用者，並將建立 CodeBuild 資源的權限授與該使用者，則該使用者可以建立 CodeBuild 資源。但是，使用者所屬的 AWS 帳戶擁有資 CodeBuild源。
- 如果您在具有建立 CodeBuild 資源權限的 AWS 帳戶中建立 IAM 角色，則任何可以擔任該角色的人都可以建立 CodeBuild資源。您的 AWS 帳戶 (角色所屬) 擁有資 CodeBuild源。

管理資源存取

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)。附加至資源的策略稱為以資源為基礎的策略。CodeBuild 支援以身分識別為基礎的政策，以及針對跨帳號資源共用的特定唯讀 API 的資源型政策。

安全存取 S3 儲存貯體

強烈建議您在 IAM 角色中包含以下許可，以驗證與您的 CodeBuild 專案關聯的 S3 儲存貯體是您或您信任的人所擁有。這些權限不包括在 AWS 受管理的策略和角色中。您必須自行新增這些許可。

- `s3:GetBucketAcl`
- `s3:GetBucketLocation`

如果專案使用的 S3 儲存貯體的擁有者發生變更，則必須確認您仍擁有該儲存貯體，並更新 IAM 角色中的許可 (如果沒有)。如需詳細資訊，請參閱 將 CodeBuild 存取權限新增至 IAM 群組或使用者 及 建立 CodeBuild 服務角色。

指定政策元素：動作、效果和委託人

服務會針對每個 AWS CodeBuild 資源定義一組 API 作業。若要授與這些 API 作業的權限，請 CodeBuild 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 資源與營運 及 AWS CodeBuild 權限參考。

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作業。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，該政策附加的使用者為隱含主體。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 動作及其套用之資源的表格，請參閱AWS CodeBuild 權限參考.

使用以身分識別為基礎的原則 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**
>
> 我們建議您先檢閱介紹性主題，其中說明可用於管理 CodeBuild 資源存取權的基本概念和選項。如需詳細資訊，請參閱 管理資 AWS CodeBuild 源存取權限概觀。

主題

- 使用 AWS CodeBuild 主控台所需的許可

- 連接 AWS CodeBuild 到 Amazon 彈性容器註冊表所需的許可
- AWS CodeBuild 主控台連線至來源提供者所需的權限
- AWS 的管理 (預先定義) 策略 AWS CodeBuild
- CodeBuild 受管理的策略和通知
- CodeBuild AWS 受管理策略的更新
- 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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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權限，以允許使用者描述 AWS 帳號的其他 AWS 資源。您必須具備下列服務的許可：

- AWS CodeBuild
- Amazon CloudWatch
- CodeCommit （如果您將源代碼存儲在存儲 AWS CodeCommit 庫中）
- Amazon Elastic Container Registry (Amazon ECR)（如果您使用的是依賴 Amazon ECR 存儲庫中的 Docker 映像的構建環境）

> **Note**
>
> 自 2022 年 7 月 26 日起，預設的身分與存取權與存取權管理政策已更新。如需詳細資訊，請參閱 連接 AWS CodeBuild 到 Amazon 彈性容器註冊表所需的許可。

- Amazon Elastic Container Service (Amazon ECS) (如果您使用的是依賴 Amazon ECR 存儲庫中的 Docker 映像的構建環境)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

如果您建立的 IAM 政策比所需的最低權限更嚴格，則主控台將無法如預期般運作。

連接 AWS CodeBuild 到 Amazon 彈性容器註冊表所需的許可

自 2022 年 7 月 26 日起， AWS CodeBuild 已更新其針對 Amazon ECR 許可的默認 IAM 政策。下列權限已從預設原則中移除：

```
"ecr:PutImage",
"ecr:InitiateLayerUpload",
"ecr:UploadLayerPart",
"ecr:CompleteLayerUpload"
```

對於在 2022 年 7 月 26 日之前建立的 CodeBuild 專案，我們建議您使用下列 Amazon ECR 政策更新政策：

```
"Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
]
```

如需更新政策的詳細資訊，請參閱將 CodeBuild 存取權限新增至 IAM 群組或使用者。

AWS CodeBuild 主控台連線至來源提供者所需的權限

主 AWS CodeBuild 控台使用下列 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 控制台將源提供程序（例如 GitHub 存儲庫）與構建項目相關聯。若要這麼做，您必須先將上述 API 動作新增至與您用來存取 AWS CodeBuild 主控台的使用者相關聯的 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 動作不會包含在 AWS CLI 和 AWS SDK 中。

AWS 的管理 (預先定義) 策略 AWS CodeBuild

AWS 透過提供由建立和管理的獨立 IAM 政策來解決許多常見使用案例 AWS。這些 AWS 受管理的政策會為常見使用案例授與必要的權限，因此您可以避免調查需要哪些權限。的受管政策 CodeBuild 還提供許可以在其他服務 (例如 IAM、Amazon EC2、Amazon ECR AWS CodeCommit、Amazon ECR、Amazon SNS 和 Amazon CloudWatch 事件) 中執行操作，這些許可對已授予有關政策的使用者負責所需的責任。例如，該`AWSCodeBuildAdminAccess`政策是管理層級的使用者政策，允許具有此政策的使用者建立和管理專案組建的 CloudWatch事件規則，以及針對專案相關事件 (名稱前綴為主題的主題`arn:aws:codebuild:`) 建立和管理 Amazon SNS 主題，以及管理中的專案和報表群組。CodeBuild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 受管理的策略 (您可以附加至帳戶中的使用者) 是特定的 AWS CodeBuild。

AWSCodeBuildAdminAccess

提供完整存取權， CodeBuild 包括管理 CodeBuild 組建專案的權限。

AWSCodeBuildDeveloperAccess

提供存取權， CodeBuild 但不允許組建專案管理。

AWSCodeBuildReadOnlyAccess

提供的唯讀存取權 CodeBuild。

若要存取建 CodeBuild 立的組建輸出成品，您也必須附加名為的 AWS 受管理原則`AmazonS3ReadOnlyAccess`。

若要建立和管理 CodeBuild 服務角色，您還必須附加名為的 AWS 受管理策略`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

AWSCodeBuildAdminAccess原則提供對於的完整存取權 CodeBuild，包括管理 CodeBuild 組建專案的權限。只將此原則套用至系統管理層級的使用者，以授與他們完全控制您 AWS 帳戶中的 CodeBuild 專案、報表群組和相關資源，包括刪除專案和報表群組的功能。

AWSCodeBuildAdminAccess 政策包含以下政策陳述式：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:*",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "codecommit:ListBranches",
        "codecommit:ListRepositories",
        "cloudwatch:GetMetricStatistics",
        "ec2:DescribeVpcs",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "elasticfilesystem:DescribeFileSystems",
        "events:DeleteRule",
        "events:DescribeRule",
        "events:DisableRule",
        "events:EnableRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "logs:GetLogEvents",
        "s3:GetBucketLocation",
        "s3:ListAllMyBuckets"
```

```
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CWLDeleteLogGroupAccess",
      "Action": [
        "logs:DeleteLogGroup"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:log-group:/aws/codebuild/*:log-stream:*"
    },
    {
      "Sid": "SSMParameterWriteAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/CodeBuild/*"
    },
    {
      "Sid": "SSMStartSessionAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession"
      ],
      "Resource": "arn:aws:ecs:*:*:task/*/*"
    },
    {
      "Sid": "CodeStarConnectionsReadWriteAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:CreateConnection",
        "codestar-connections:DeleteConnection",
        "codestar-connections:UpdateConnectionInstallation",
        "codestar-connections:TagResource",
        "codestar-connections:UntagResource",
        "codestar-connections:ListConnections",
        "codestar-connections:ListInstallationTargets",
        "codestar-connections:ListTagsForResource",
        "codestar-connections:GetConnection",
        "codestar-connections:GetIndividualAccessToken",
        "codestar-connections:GetInstallationUrl",
        "codestar-connections:PassConnection",
```

```
        "codestar-connections:StartOAuthHandshake",
        "codestar-connections:UseConnection"
      ],
      "Resource": [
        "arn:aws:codestar-connections:*:*:connection/*",
        "arn:aws:codeconnections:*:*:connection/*"
      ]
    },
    {
      "Sid": "CodeStarNotificationsReadWriteAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:CreateNotificationRule",
        "codestar-notifications:DescribeNotificationRule",
        "codestar-notifications:UpdateNotificationRule",
        "codestar-notifications:DeleteNotificationRule",
        "codestar-notifications:Subscribe",
        "codestar-notifications:Unsubscribe"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets",
        "codestar-notifications:ListTagsforResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns:SetTopicAttributes"
      ],
```

```
      "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
      "Sid": "SNSTopicListAccess",
      "Effect": "Allow",
      "Action": [
        "sns:ListTopics",
        "sns:GetTopicAttributes"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsChatbotAccess",
      "Effect": "Allow",
      "Action": [
        "chatbot:DescribeSlackChannelConfigurations",
        "chatbot:ListMicrosoftTeamsChannelConfigurations"
      ],
      "Resource": "*"
    }
  ]
}
```

AWSCodeBuildDeveloperAccess

此AWSCodeBuildDeveloperAccess原則可讓您存取與專案 CodeBuild 和報表群組相關資源的所有功能。此原則不允許使用者刪除 CodeBuild 專案或報告群組，或其他 AWS 服務 (例如 CloudWatch 事件) 中的相關資源。建議將此政策套用到大多數使用者。

AWSCodeBuildDeveloperAccess 政策包含以下政策陳述式：

```
{
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:StartBuildBatch",
        "codebuild:StopBuildBatch",
        "codebuild:RetryBuild",
        "codebuild:RetryBuildBatch",
        "codebuild:BatchGet*",
```

```
        "codebuild:GetResourcePolicy",
        "codebuild:DescribeTestCases",
        "codebuild:DescribeCodeCoverages",
        "codebuild:List*",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "codecommit:ListBranches",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "logs:GetLogEvents",
        "s3:GetBucketLocation",
        "s3:ListAllMyBuckets"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "SSMParameterWriteAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/CodeBuild/*"
    },
    {
      "Sid": "SSMStartSessionAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession"
      ],
      "Resource": "arn:aws:ecs:*:*:task/*/*"
    },
    {
      "Sid": "CodeStarConnectionsUserAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:ListConnections",
        "codestar-connections:GetConnection"
      ],
      "Resource": [
        "arn:aws:codestar-connections:*:*:connection/*",
```

```
        "arn:aws:codeconnections:*:*:connection/*"
      ]
    },
    {
      "Sid": "CodeStarNotificationsReadWriteAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:CreateNotificationRule",
        "codestar-notifications:DescribeNotificationRule",
        "codestar-notifications:UpdateNotificationRule",
        "codestar-notifications:Subscribe",
        "codestar-notifications:Unsubscribe"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets",
        "codestar-notifications:ListTagsforResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "SNSTopicListAccess",
      "Effect": "Allow",
      "Action": [
        "sns:ListTopics",
        "sns:GetTopicAttributes"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsChatbotAccess",
      "Effect": "Allow",
      "Action": [
```

```
        "chatbot:DescribeSlackChannelConfigurations",
        "chatbot:ListMicrosoftTeamsChannelConfigurations"
      ],
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

AWSCodeBuildReadOnlyAccess

此原AWSCodeBuildReadOnlyAccess則會授與其他 AWS 服務中相關資源的唯讀存取權限。CodeBuild 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。

AWSCodeBuildReadOnlyAccess 政策包含以下政策陳述式：

```
{
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:BatchGet*",
        "codebuild:GetResourcePolicy",
        "codebuild:List*",
        "codebuild:DescribeTestCases",
        "codebuild:DescribeCodeCoverages",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "logs:GetLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CodeStarConnectionsUserAccess",
      "Effect": "Allow",
      "Action": [
```

```
        "codestar-connections:ListConnections",
        "codestar-connections:GetConnection"
      ],
      "Resource": [
        "arn:aws:codestar-connections:*:*:connection/*",
        "arn:aws:codeconnections:*:*:connection/*"
      ]
    },
    {
      "Sid": "CodeStarNotificationsPowerUserAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:DescribeNotificationRule"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets"
      ],
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

CodeBuild 受管理的策略和通知

CodeBuild 支持通知，可以通知用戶構建項目的重要更改。 CodeBuild 包含通知功能的政策聲明的受管理策略。如需詳細資訊，請參閱什麼是通知？。

完整存取受管政策中的通知相關許可

AWSCodeBuildFullAccess 受管政策包含下列陳述式，允許對通知的完整存取權限。套用此受管政策的使用者也可以針對通知建立和管理 Amazon SNS 主題、訂閱和取消訂閱使用者主題、列出要選擇作為通知規則目標的主題，以及列出針對 Slack 設定的用 AWS Chatbot 戶端。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
```

```
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

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" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild:*"}
        }
    },
    {
        "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" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild*"}
        }
    },
    {
        "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[Identity and Access Management閱AWS CodeStar通知](#)。

CodeBuild AWS 受管理策略的更新

檢視 CodeBuild 自此服務開始追蹤這些變更以來的 AWS 受管理策略更新詳細資料。如需有關此頁面變更的自動警示，請在訂閱 RSS 摘要[AWS CodeBuild 使用者指南文件記錄](#)。

變更	描述	日期
AWSCodeBuildAdminAccess AWSCodeBuildDeveloperAccess 、和 AWSCodeBuildReadOnlyAccess — 現有策略的更新	CodeBuild 在這些策略中加入了資源以支援 AWS CodeConnections 品牌重塑。 AWSCodeBuildAdminAccess AWSCodeBuildDeveloperAccess 、和AWSCodeBuildReadOnlyAccess 策略已變更為新增資源、arn:aws:codeconnections:*:*:connection/* 。	2024年4月18日
AWSCodeBuildAdminAccess 和 AWSCodeBuildDeveloperAccess — 更新現有政策	CodeBuild 為這些策略添加了權限，以支持使用的其他通知類型 AWS Chatbot。 AWSCodeBuildAdminAccess 和AWSCodeBuildDeveloperAccess	2023 年 5 月 16 日

變更	描述	日期
	原則已變更為新增權限`chatbot:ListMicrosoftTeamsChannelConfigurations` 。	
CodeBuild 開始追蹤變更	CodeBuild 開始追蹤其 AWS 受管理策略的變更。	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 SDK 或 AWS CLI時，這些原則會運作。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 IAM 政策範例來限制使用者和角色的 CodeBuild 存取權限。

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
- 允許使用者建立報告群組
- 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
- 允許使用者取得由管理的 Docker 映像檔的相關資訊 CodeBuild
- 允許 CodeBuild 存取建立虛擬私人 VPC 網路介面所需的 AWS 服務
- 使用 deny 陳述式來防止 AWS CodeBuild 中斷來源提供者的連線

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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReportGroups",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReports",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

允許使用者建立建置專案

下列範例原則陳述式可讓使用者以任何名稱建立建置專案，但僅限於 [us-east-2區域] (Region for account)，123456789012且只能使用指定的 CodeBuild 服務角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole"
    }
  ]
}
```

下列範例原則陳述式可讓使用者使用任何名稱建立建置專案，但僅限於 [區域] (us-east-2Region for) 帳戶中，而123456789012且只能使用指定的 CodeBuild 服務角色。它還強制使用者只能搭配任何其他服務使用指定的服務角色， AWS CodeBuild 而不能使用任何其他 AWS 服務。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole",
      "Condition": {
          "StringEquals": {"iam:PassedToService": "codebuild.amazonaws.com"}
      }
    }
  ]
}}
```

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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReport",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

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

```
      "Action": "codebuild:DeleteProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

```
{
  "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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole"
    }
  ]
}
```

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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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 中名稱為 my-build-project 和 my-other-build-project 的組建專案：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetBuilds",
      "Resource": [
        "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
        "arn:aws:codebuild:us-east-2:123456789012: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 中名稱為 my-build-project 和 my-other-build-project 的組建專案：

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListBuildsForProject",
      "Resource": [
        "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
        "arn:aws:codebuild:us-east-2:123456789012: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：

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

```
{
  "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：

```
{
  "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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReportsForReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

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

```
      "Action": "codebuild:DescribeTestCases",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StopBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012: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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchDeleteBuilds",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

允許使用者取得由管理的 Docker 映像檔的相關資訊 CodeBuild

下列範例原則陳述式可讓使用者取得由 CodeBuild管理之所有 Docker 映像檔的相關資訊：

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

允許 CodeBuild 存取建立虛擬私人 VPC 網路介面所需的 AWS 服務

下列範例原則陳述式 AWS CodeBuild 授與在具有兩個子網路的 VPC 中建立網路介面的權限：

```
{
  "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:region:account-id:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:AuthorizedService": "codebuild.amazonaws.com"
        },
        "ArnEquals": {
          "ec2:Subnet": [
            "arn:aws:ec2:region:account-id:subnet/subnet-id-1",
            "arn:aws:ec2:region:account-id:subnet/subnet-id-2"
          ]
        }
      }
    }
  ]
}
```

使用 deny 陳述式來防止 AWS CodeBuild 中斷來源提供者的連線

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權限]。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "codebuild:DeleteOAuthToken",
      "Resource": "*"
    }
  ]
}
```

AWS CodeBuild 權限參考

您可以在 AWS CodeBuild 原則中使用 AWS寬條件金鑰來表示條件。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，codebuild:*指定所有 CodeBuild 動作，並codebuild:Batch*指定以該字開頭的所有 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 操作和動作所需的權限

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 專案所需。

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

需要刪除一組物件，這些SourceCredentialsInfo物件包含有關 GitHub、 GitHub 企業伺服器或 Bitbucket 儲存庫認證的資訊。

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

需要匯入一組SourceCredentialsInfo物件，其中包含有關 GitHub、 GitHub 企業伺服器或 Bitbucket 儲存庫認證的資訊。

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

需要取得組建批次 ID 清單。

資源：*

ListBuildBatchesForProject

動作：codebuild:ListBuildBatchesForProject

需要取得特定專案的建置批次 ID 清單。

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

需要重試組建。

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

需要變更專案組建的公開可見度。

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。

使用標籤來控制對 AWS CodeBuild 資源的存取

IAM 政策陳述式中的條件是語法的一部分，您可以用來指定 CodeBuild 專案型動作的權限。您可以建立政策，根據與這些專案相關聯的標記允許或拒絕專案執行動作，然後將這些政策套用到為管理

使用者設定的 IAM 群組。如需使用主控台將標籤套用至專案的詳細資訊 AWS CLI，請參閱在 AWS CodeBuild 中建立建置專案。如需使用 CodeBuild SDK 套用標籤的詳細資訊，請參閱 CodeBuildAPI 參考中的CreateProject 和標籤。如需使用標籤來控制 AWS 資源存取權的詳細資訊，請參閱 IAM 使用 AWS 者指南中的使用資源標籤控制資源存取。

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之外，使用者的管理員還必須將此 IAM 政策附加至未經授權的使用者。aws:ResourceTag 條件索引鍵用於依據其標籤來控制資源的存取。

```
{
  "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 政策附加到未授權執行這些動作的使用者。aws:RequestTag條件金鑰用來控制哪些標籤可以在 IAM 要求中傳遞

```
{
  "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:StringEquals": {
          "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關聯的 AWS 標記建立政策，允許或拒絕對 CodeBuild 資源 (專案和報表群組) 執行動作，然後將這些政策套用至為管理使用者設定的 IAM 群組。*########################## AWS ####Status######## CodeBuild ##Secret################## (####) ### IAM ###*然後，您必須確定處理這些標記報表群組的開發人員不是該一般*###*員群組的成員，而是屬於未套用限制性政策的其他 IAM 群組 (SecretDevelopers)。

下列範例會拒絕 CodeBuild 針對以索引鍵Status和索引鍵值標記之報表群組的Secret所有動作：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : [
        "codebuild:BatchGetReportGroups,"
```

```
        "codebuild:CreateReportGroup",
        "codebuild:DeleteReportGroup",
        "codebuild:ListReportGroups",
        "codebuild:ListReportsForReportGroup",
        "codebuild:UpdateReportGroup"
       ]
      "Resource" : "*",
      "Condition" : {
         "StringEquals" : "aws:ResourceTag/Status": "Secret"
        }
    }
  ]
}
```

Example 範例 4： AWSCodeBuildDeveloperAccess 根據資源標籤將 CodeBuild 動作限制為

您可以建立政策，允許 CodeBuild 對未使用特定標記標記的所有報表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：

```
{
   "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 控制台需要有`ListRepositories`權限才能在您登錄的 AWS 地區中顯示您 AWS 帳戶的存儲庫列表。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。這項搜尋會在您登入所在 AWS 地區的 AWS 帳戶中執行。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，但如果有明確的 `Deny` 而無法檢視部分資源，則不會傳回這些資源。

符合性驗證 AWS CodeBuild

協力廠商稽核人員會評估其安全性與合規性， AWS CodeBuild 做為多個 AWS 合規計畫的一部分。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

如需特定法規遵循計劃範圍內的 AWS 服AWS 務清單，請參閱合規計劃範圍內的服務。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，請參閱在 AWS Artifact 中下載報表。

您在使用時的合規責任取決 CodeBuild 於資料的敏感性、公司的合規目標以及適用的法律和法規。如果您的 CodeBuild 使用必須遵守 HIPAA、PCI 或 FedRAMP 等標準，可 AWS 提供資源以協助：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供在上部署以安全性和法規遵循為重點的基準環境的步驟。 AWS
- 建構 HIPAA 安全性與合規性白皮書 — 本白皮書說明公司如何使用建立符合 HIPAA 標準的應用 AWS 程式。
- AWS 合規性資源 — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS Config— 此 AWS 服務評估您的資源配置是否符合內部實踐，行業準則和法規。
- AWS Security Hub— 監視您的使用， AWS CodeBuild 因為它涉及到安全最佳實踐通過使用 AWS Security Hub. Security Hub 會透過安全控制來評估資源組態和安全標準，協助您遵守各種合規架構。如需有關使用 Security Hub 評估 CodeBuild 資源的詳細資訊，請參閱使用 AWS Security Hub 者指南中的AWS CodeBuild 控制項。

韌性 AWS CodeBuild

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。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。

如需區域和可用區域的詳 AWS 細資訊，請參閱AWS 全域基礎結構。

基礎架構安全性 AWS CodeBuild

作為託管服務， AWS CodeBuild 受到 AWS 全球網絡安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫透 CodeBuild 過網路進行存取。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。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

存取您的來源供應商 CodeBuild

對於 GitHub 或 GitHub 企業服務器，您可以使用個人訪問令牌或 OAuth 應用程序來訪問源提供程序。對於 Bitbucket，您可以使用訪問令牌，應用程序密碼或 OAuth 應用程序來訪問源提供程序。

> **Note**
>
> GitLab 和自我管理 GitLab 源提供程序不是直接訪問， CodeBuild 而是通過 AWS CodeConnections。

主題

GitHub 和 GitHub 企業服務器訪問令牌

存取字符先決條件

在開始之前，您必須將適當的權限範圍添加到 GitHub 訪問令牌中。

對於 GitHub，您的個人訪問令牌必須具有以下範圍。

- repo：授予私有儲存庫的完全控制。
- rep：status：授予公共和私有存儲庫提交狀態的讀/寫訪問權限。
- 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。

有關詳情，請參閱了解 GitHub 網站上 OAuth 應用程式的範圍。

如果您使用的是細粒度的個人訪問令牌，具體取決於您的用例，您的個人訪問令牌可能需要以下權限：

- 內容：唯讀：授與私人儲存區域的存取權。如果您使用私有存放庫作為來源，則需要此權限。
- 確認狀態：讀取和寫入：授與建立提交狀態的權限。如果您的項目已設置 webhook，或者您已啟用報告構建狀態功能，則需要此權限。
- 網絡掛鉤：讀取和寫入：授予管理網絡掛鉤的權限。如果您的項目設置了 webhook，則需要此權限。
- 提取要求：唯讀：授與存取提取要求的權限。如果您的 webhook 對提取請求事件具有`FILE_PATH`過濾器，則需要此權限。
- 管理：讀取和寫入：如果搭 CodeBuild配使用自我託管的 GitHub 動作執行器功能，則需要此權限。有關更多詳細信息，請參閱為存儲庫和創建註冊令牌在中設定自託管的 GitHub 動作跑步者 AWS CodeBuild。

> **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 網站上精細個人存取權杖所需的權限。

Connect GitHub 訪問令牌（控制台）

要使用控制台 GitHub 使用訪問令牌將項目連接到，請在創建項目時執行以下操作。如需相關資訊，請參閱建立組建專案 (主控台)。

1. 對於來源提供者，請選擇GitHub。
2. 對於存放庫，選擇使用個 GitHub 人存取權杖 Connect。
3. 在GitHub 個人訪問令牌中，輸入您的 GitHub個人訪問令牌。
4. 選擇 Save token (儲存字符)。

Connect GitHub 用存取權杖 (CLI) 連線

請按照以下步驟使用 AWS CLI 將您的項目連接到 GitHub 使用訪問令牌。若要取得有關使用 AWS CLI 與的資訊 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。將資料複製到本機電腦或執行個體上安裝的位置中的檔案 (例如，*import-source-credentials.json*)。 AWS CLI 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 或企業。
- *auth-type*：必要值。用來連線至 GitHub 或 GitHub 企業伺服器儲存區域的驗證類型。有效值包括 PERSONAL_ACCESS_TOKEN 和 BASIC_AUTH。您無法使用 CodeBuild API 來建立 OAUTH 連線。您必須改用 CodeBuild 主控台。
- *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 企業服務器，這是個人訪問令牌。
- *username*：選用值。 GitHub 和「 GitHub 企業伺服器」來源提供者會忽略此參數。

2. 若要使用存取字符連接您的帳戶，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符。

在您的帳戶與訪問令牌連接後，您可以使用`create-project`來創建 CodeBuild 項目。如需詳細資訊，請參閱 建立建置專案 (AWS CLI)。

3. 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。此值可以為 `OAUTH`、`BASIC_AUTH` 或 `PERSONAL_ACCESS_TOKEN`。
- `serverType` 是來源供應商的類型。此值可以為 `GITHUB`、`GITHUB_ENTERPRISE` 或 `BITBUCKET`。
- `arn` 為字符的 ARN。

4. 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

GitHub 使用 OAuth（控制台）進行 Connect

要使用控制台將項目連接到 GitHub 使用 OAuth 應用程序，請在創建項目時執行以下操作。如需相關資訊，請參閱建立組建專案 (主控台)。

1. 對於來源提供者，請選擇GitHub。
2. 對於存放庫，選擇使用 OAuth Connect。
3. 選擇 Connect 到 GitHub，登錄並授權您的帳戶。
4. 選擇「確認」以連接 CodeBuild 到您的 GitHub 帳戶。
5. 在GitHub 儲存庫中，輸入您的 GitHub存放庫連結。

若要檢閱已授權的 OAuth 應用程式，請瀏覽至開啟的應用程式 GitHub，並確認是否列出 aws-codebuild AWS CodeBuild (*region*) 擁有的應用程式。

比特桶應用程序密碼或訪問令牌

必要條件

在開始之前，您必須將適當的權限範圍添加到您的 Bitbucket 應用程序密碼或訪問令牌。

對於 Bitbucket，您的應用程序密碼或訪問令牌必須具有以下範圍。

- 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。如果您的項目具有 Bitbucket webhook，那麼您的應用程序密碼或訪問令牌必須具有此範圍。
- webhook：授予 Webhook 的存取權。如果您的項目具有 webhook 操作，則您的應用程序密碼或訪問令牌必須具有此範圍。

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。

使用應用程序密碼（控制台）Connect Bitbucket

要使用控制台使用應用程序密碼將項目連接到 Bitbucket，請在創建項目時執行以下操作。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。

> **Note**
>
> CodeBuild 不支援比特桶伺服器。

2. 對於 Repository (儲存庫)，選擇 Connect with a Bitbucket app password (使用 Bitbucket 應用程式密碼連接)。
3. 在 Bitbucket username (Bitbucket 使用者名稱) 中，輸入您的 Bitbucket 使用者名稱。
4. 在 Bitbucket app password (Bitbucket 應用程式密碼) 中，輸入您的 Bitbucket 應用程式密碼。
5. 選擇 Save Bitbucket credentials (儲存 Bitbucket 登入資料)。

使用訪問令牌（控制台）Connect Bitbucket

要使用控制台使用訪問令牌將項目連接到 Bitbucket，請在創建項目時執行以下操作。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。

> **Note**
>
> CodeBuild 不支援比特桶伺服器。

2. 對於存儲庫，選擇使用 Bitbucket 訪問令牌 Connect。
3. 在 Bitbucket 存取權杖中，輸入您的 Bitbucket 存取權杖。
4. 選擇 Save token (儲存字符)。

使用應用程序密碼或訪問令牌（CLI）Connect Bitbucket

請依照下列步驟使用應 AWS CLI 用程式密碼或存取權杖將專案連線至 Bitbucket。若要取得有關使用 AWS CLI 與的資訊 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。將資料複製到本機電腦或執行個體上安裝的位置中的檔案 (例如，*import-source-credentials.json*)。 AWS CLI 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：

- *auth-type*：必要值。用來連線至 Bitbucket 儲存庫的驗證類型。有效值包括 PERSONAL_ACCESS_TOKEN 和 BASIC_AUTH。您無法使用 CodeBuild API 來建立 OAUTH 連線。您必須改用 CodeBuild 主控台。
- *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 Bitbucket，這是訪問令牌或應用程序密碼。
- *username*：選用值。當authType為基本 _AUTH 時，位桶使用者名稱。將對其他類型的來源供應商或連線忽略此參數。

2. 若要將您的帳戶與應用程式密碼或存取權杖連線，請切換至包含您在步驟 1 中儲存之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。

使用應用程式密碼連結帳戶後，您就可以用`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)。

3. 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分驗證類型。此值可以為 `OAUTH`、`BASIC_AUTH` 或 `PERSONAL_ACCESS_TOKEN`。
- `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"
}
```

比特桶 OAuth 應用程序

使用 OAuth（控制台）Connect 比特桶

要使用控制台使用 OAuth 應用程序將項目連接到 Bitbucket，請在創建項目時執行以下操作。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. 對於存放庫，選擇使用 OAuth Connect。
3. 選擇 Connect 到 Bitbucket，登錄並授權您的帳戶。
4. 選擇「確認」以連接 CodeBuild 到您的 Bitbucket 帳戶。
5. 在 Bitbucket 儲存庫中，輸入您的 Bitbucket 儲存庫連結。

要查看您授權的 OAuth 應用程序，請導航到 Bitbucket 上的應用程序授權，並驗證是否列出了名為AWS CodeBuild (*region*)的應用程序。

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。在中 AWS，跨服務模擬可能會導致混淆的副問題。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。

我們建議在資源策略中使用aws:SourceArn和aws:SourceAccount全域條件前後關聯索引鍵，以限制將其他服務 AWS CodeBuild 提供給資源的權限。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。

下列範例顯示如何在中使用aws:SourceArn和aws:SourceAccount全域條件前後關聯鍵字 CodeBuild 來避免混淆的副問題。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:codebuild:region-ID:account-
ID:project/project-name"
                }
            }
        }
    ]
}
```

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

- 進階設定
- AWS CodeBuild 的命令列參考
- 適用於 AWS CodeBuild 的 AWS 開發套件和工具參考
- 指定 AWS CodeBuild 端點
- 搭配 AWS CodeBuild 使用 AWS CodePipeline 測試程式碼及執行建置
- 使用 AWS CodeBuild 搭配 Jenkins
- 搭配 Codecov 使用 AWS CodeBuild
- 使用AWS CodeBuild與無伺服器應用程式

進階設定

如果您遵循開始使用主控台中的步驟來首次存取 AWS CodeBuild，您大概不需要本主題中的資訊。不過，當您繼續使用時 CodeBuild，您可能會想要執行諸如授與組織中的 IAM 群組和使用者存取權限 CodeBuild、修改 IAM 中現有的服務角色或存AWS KMS keys取 CodeBuild，或是設定組織工作站中的 AWS CLI各個工作站進行存取 CodeBuild。本主題描述如何完成相關的設定步驟。

我們假設您已經有 AWS 帳戶。不過，如果您還沒有，請前往 http://aws.amazon.com，選擇「登入主機」，然後依照線上指示操作。

主題

- 將 CodeBuild 存取權限新增至 IAM 群組或使用者
- 建立 CodeBuild 服務角色
- 建立並設定客戶受管金鑰 CodeBuild
- 安裝及設定 AWS CLI

將 CodeBuild 存取權限新增至 IAM 群組或使用者

若要AWS CodeBuild透過 IAM 群組或使用者存取，您必須新增存取權限。本節說明如何使用 IAM 主控台或AWS CLI.

如果您將使 CodeBuild 用AWS root 帳戶（不建議）或帳戶中的管理員用AWS戶進行訪問，則不需要按照這些說明進行操作。

如需有關AWS root 帳號和管理員使用者的資訊，請參閱《使用指南》中的「AWS 帳戶AWS 帳戶root 使用者」和「建立您的第一個 root 使用者和群組」。

將 CodeBuild 存取權限新增至 IAM 群組或使用者 (主控台)

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

   您應已使用下列其中一項登入 AWS Management Console：

   - 您的 AWS 根帳戶。此為不建議的選項。如需詳細資訊，請參閱《使用指南》中的AWS 帳戶 root 使用者。
   - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的〈建立您的第一個AWS 帳戶 root 使用者和群組〉。
   - 您AWS帳戶中有權執行下列最少動作集的使用者：

   ```
   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. 如要將一組自訂的AWS CodeBuild存取權限新增至 IAM 群組或 IAM 使用者，請直接跳到此程序中的步驟 4。

   若要將一組預設 CodeBuild 存取權限新增至 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：

   - 若要新增完整存取權限 CodeBuild，請選取名為的方塊 AWSCodeBuildAdminAccess，選擇 [原則動作]，然後選擇 [附加]。選取目標 IAM 群組或使用者旁邊的核取方塊，然後選擇 [附加政策]。對名為亞馬遜 S3ReadOnlyAccess 和 IAM 的政策重複此操作FullAccess。

- 若要將存取權限新增至除組建專案管理以外的所有項目，請選取名 CodeBuild 為的方塊 AWSCodeBuildDeveloperAccess，選擇 [原則動作]，然後選擇 [附加]。選取目標 IAM 群組或使用者旁邊的核取方塊，然後選擇 [附加政策]。對名為亞馬遜 S3 的策略重複此操作 ReadOnlyAccess。
- 若要將唯讀存取權限新增至 CodeBuild，請選取名為的方塊AWSCodeBuildReadOnlyAccess。選取目標 IAM 群組或使用者旁邊的核取方塊，然後選擇 [附加政策]。對名為亞馬遜 S3 的策略重複此操作ReadOnlyAccess。

您現在已將一組預設的 CodeBuild 存取權限新增至 IAM 群組或使用者。略過此程序的其餘步驟。

4. 選擇 建立政策 。
5. 在 [建立原則] 頁面上，選擇 [建立您自己的原則] 旁邊的 [選取]。
6. 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。
7. 針對 Policy Document (政策文件)，輸入下列項目，然後選擇 Create Policy (建立政策)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::account-ID: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**
>
> 此原則允許存取所有 CodeBuild 動作，以及可能存取大量AWS資源。若要將權限限制為特定 CodeBuild動作，請變更 CodeBuild 政策陳述式`codebuild:*`中的值。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。
>
> 該`CodeBuildRolePolicy`語句是允許創建或修改構建項目所必需的。

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，選擇要新增 CodeBuild 存取權限的 IAM 群組或 IAM 使用者的名稱。
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，請在 [附加原則] 頁面上選取 CodeBuildAccessPolicy，然後選擇 [附加原則]。

    對於使用者，請在 [新增權限] 頁面上選擇 [直接附加現有策略]。選取 CodeBuildAccessPolicy，選擇 [下一步：檢閱]，然後選擇 [新增權限]。

將 CodeBuild 存取權限新增至 IAM 群組或使用者 (AWS CLI)

1. 請確定您已設定AWS CLI與其中一個 IAM 實體對應的存取金鑰和AWS秘密存取金鑰，如上述程序所述。AWS若要取得更多資訊，請參閱《AWS Command Line Interface使用者指南》AWS Command Line Interface中的〈進行設定〉。
2. 若要將自訂AWS CodeBuild存取權限集新增至 IAM 群組或 IAM 使用者，請跳至本程序中的步驟 3。

   若要將一組預設 CodeBuild 存取權限新增至 IAM 群組或 IAM 使用者，請執行以下操作：

   視您要將權限新增至 IAM 群組或使用者而定，執行下列其中一個命令：

   ```
   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，將####或#####取代為 IAM 群組名稱或使用者名稱，並針對下列每個政策 Amazon 資源名稱 (*ARN*) 取代政策 arn 一次：

   - 若要將完整存取權限新增至 CodeBuild，請使用下列原則 ARN：
     - arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess
     - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
     - arn:aws:iam::aws:policy/IAMFullAccess
   - 若要將存取權限新增至 CodeBuild 除組建專案管理以外的所有項目，請使用下列原則 ARN：
     - arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess
     - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
   - 若要將唯讀存取權限新增至 CodeBuild，請使用下列原則 ARN：
     - arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess

- arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess

您現在已將一組預設的 CodeBuild 存取權限新增至 IAM 群組或使用者。略過此程序的其餘步驟。

3. 在已安裝 AWS CLI 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 put-user-policy.json 的檔案。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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::account-ID: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**
>
> 此原則允許存取所有 CodeBuild 動作，以及可能存取大量AWS資源。若要將權限限制為特定 CodeBuild動作，請變更 CodeBuild 政策陳述式`codebuild:*`中的值。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。該`CodeBuildRolePolicy`語句是允許創建或修改構建項目所必需的。

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
   ```

   如果是 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，將群###或#####取代為目標 IAM 群組或使用者的名稱。

建立 CodeBuild 服務角色

您需要AWS CodeBuild服務角色，才 CodeBuild 能代表您與相依AWS服務互動。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 的管道 (CodePipeline 主控台)
- 將 CodeBuild 建置動作新增至管道 (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AWS CLI. CodeBuild

> **Important**
>
> CodeBuild 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。
> 此頁面描述的服務角色包含一個政策，此政策授予使用 CodeBuild 所需的最低許可。您可能需要新增其他權限，視您的使用案例而定。

建立 CodeBuild 服務角色 (主控台)

1. 前往網址 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《使用指南》中的AWS 帳戶 root 使用者。
   - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的〈建立您的第一個AWS 帳戶 root 使用者和群組〉。
   - 您AWS帳戶中有權執行下列最少動作集的使用者：

   ```
   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)：

```
{
  "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**
>
> 此政策包含的陳述式允許存取可能許多的 AWS 資源。若要限制 AWS CodeBuild 對特定 AWS 資源的存取，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。

6. 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。

7. 在導覽窗格中，選擇 Roles (角色)。
8. 選擇 Create Role (建立角色)。
9. 在 [建立角色] 頁面上，選擇 [AWS服務] CodeBuild，然後選擇 [下一步:權限]。
10. 在 [附加權限原則] 頁面上，選取 CodeBuildServiceRolePolicy，然後選擇 [下一步：檢閱]。
11. 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)。

若要建立 CodeBuild 服務角色 (AWS CLI)

1. 請確定您已設定AWS CLI與其中一個 IAM 實體對應的存取金鑰和AWS秘密存取金鑰，如上述程序所述。AWS若要取得更多資訊，請參閱《AWS Command Line Interface使用者指南》AWS Command Line Interface中的〈進行設定〉。
2. 在已安裝 AWS CLI 的本機工作站或執行個體上的空目錄中，建立名為 `create-role.json` 和 `put-role-policy.json` 這兩個檔案。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`:

```
{
  "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，而且aws:SourceArn是 CodeBuild 專案 ARN。

如果您想將服務角色限制為AWS帳戶，create-role.json可能看起來像這樣：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "account-ID"
                    ]
                }
            }
        }
    ]
}
```

如果您想將服務角色限制為特定 CodeBuild 專案，create-role.json可能會類似下列內容：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:codebuild:region-ID:account-
ID:project/project-name"
                }
            }
        }
    ]
}
```

> **Note**
>
> 如果您不清楚或尚未決定 CodeBuild 專案的名稱，並希望對某個 ARN 模式有信任政策限制，您可以使用萬用字元 (*) 取代該部分的 ARN。建立專案之後，您就可以更新信任原則。

`put-role-policy.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**
>
> 此政策包含的陳述式允許存取可能許多的 AWS 資源。若要限制 AWS CodeBuild 對特定 AWS 資源的存取，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
```

建立並設定客戶受管金鑰 CodeBuild

為了AWS CodeBuild加密其構建輸出成品，它需要訪問 KMS 密鑰。根據預設， CodeBuild 會在您的AWS帳戶中使用適用AWS 受管金鑰於 Amazon S3 的。

如果您不想使用AWS 受管金鑰，您必須自行建立和設定客戶受管金鑰。本節說明如何使用 IAM 主控台執行此動作。

如需有關客戶受管金鑰的詳細資訊，請參閱AWS KMS開發人員指南中的AWS Key Management Service概念和建立金鑰。

若要設定客戶管理的金鑰以供使用 CodeBuild，請依照AWS KMS開發人員指南中修改金鑰原則中 < 如何修改金鑰原則 > 一節中的指示進行。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，也不要在金鑰政策中輸入這些省略符號。

```
{
  "Version": "2012-10-17",
  "Id": "...",
  "Statement": [
    ### BEGIN ADDING STATEMENTS HERE ###
    {
      "Sid": "Allow access through Amazon S3 for all principals in the account that are
 authorized to use Amazon S3",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
```

```
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.region-ID.amazonaws.com",
          "kms:CallerAccount": "account-ID"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-ID:role/CodeBuild-service-role"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    ### END ADDING STATEMENTS HERE ###
    {
      "Sid": "Enable IAM User Permissions",
      ...
    },
    {
      "Sid": "Allow access for Key Administrators",
      ...
    },
    {
      "Sid": "Allow use of the key",
      ...
    },
    {
      "Sid": "Allow attachment of persistent resources",
      ...
```

```
    }
  ]
}
```

- *## ID* 代表與之相關聯之 Amazon S3 儲存貯體所在AWS區域 CodeBuild 的識別碼 (例如，us-east-1)。
- *## ID* 代表擁有客戶受管金鑰之AWS帳戶的 ID。
- *CodeBuild-service role* 代表您在本主題稍早建立或識別之 CodeBuild 服務角色的名稱。

> **Note**
>
> 若要透過 IAM 主控台建立或設定客戶管理金鑰，您必須先使AWS Management Console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號根使用者。
> - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的〈建立您的第一個AWS 帳戶 root 使用者和群組〉。
> - 您AWS帳戶中有權建立或修改客戶管理金鑰的使用者。如需詳細資訊，請參閱AWS KMS開發人員指南中的使用AWS KMS主控台所需的權限。

安裝及設定 AWS CLI

若要存取AWS CodeBuild，您可以使用AWS CLI與 (或取代) CodeBuild 主 CodePipeline 控台、主控台或AWS SDK。若要安裝和規劃AWS CLI，請參閱《使用者指南》AWS Command Line Interface中的〈AWS Command Line Interface使用〉進行設定。

1. 執行下列命令，以確認您是否安裝AWS CLI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。若要取得更多資訊，請參閱《使用指南》AWS Command Line Interface中的〈解除安裝AWS CLI和安裝AWS Command Line Interface〉。

AWS CodeBuild 的命令列參考

AWS CLI 提供用於自動化 AWS CodeBuild 的命令。使用本主題中的資訊作為AWS Command Line Interface使用者指南與AWS CLI適用於 的 參考AWS CodeBuild。

不是您想找的內容嗎？ 如果您希望重新使用AWSSDK 調用 CodeBuild，請參閱AWS 開發套件和工具參考。

若要使用本主題中的資訊，您應該已安裝AWS CLI並將其配置為與 CodeBuild 一起使用，如安裝及設定 AWS CLI。

若要使用 AWS for WordPressAWS CLI若要指定代碼組建的端點，請參指定 AWS CodeBuild 端點 (AWS CLI)。

執行此命令，以取得代碼組建命令的清單。

```
aws codebuild help
```

執行此命令，以取得與 CodeBuild 命令相關的資訊，*###*是命令的名稱。

```
aws codebuild command-name help
```

CodeBuild 命令包括：

- `batch-delete-builds`：刪除代碼組建中的一或多個組建。如需詳細資訊，請參閱 刪除建置 (AWS CLI)。
- `batch-get-builds`：取得代碼組建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`：列出代碼組建的 Amazon Resource Names (ARN)。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`：取得由 CodeBuild 管理、您可以用於組建的 Docker 影像的清單。如需詳細資訊，請參閱 碼頭圖片提供 CodeBuild。
- `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)。

適用於 AWS CodeBuild 的 AWS 開發套件和工具參考

若要使用其中一個 AWS 開發套件或工具來自動化 AWS CodeBuild，請參閱下列資源。

如果您想要使用AWS CLI運行 CodeBuild，請參閱命令列參考。

支援 AWS CodeBuild 的 AWS 開發套件和工具

如下所示AWS開發套件和工具支援 CodeBuild：

- 適用於 C++ 的 AWS 開發套件。如需詳細資訊，請參閱 。AWS:: CodeBuild命名空間部分AWSSDK for C++ 開發套件 API 參考。
- 適用於 Go 的 AWS 開發套件。如需詳細資訊，請參閱 。codebuild的 區段AWSSDK for Go API 參考。
- 適用於 Java 的 AWS 開發套件。如需詳細資訊，請參閱 。`com.amazonaws.services.codebuild`和`com.amazonaws.services.codebuild.model`章節位置AWS適用 SDK for Java 開發套件 API 參考。
- 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.CodeBuild的 區段AWS適用於 JavaScript 的開發套件 API 參考。

- 適用於 .NET 的 AWS 開發套件。如需詳細資訊，請參閱 。卓越亞馬遜代碼構建和卓越亞馬遜代碼構建型號命名空間部分AWS適用 SDK for .NET 開發套件 API 參考。
- 適用於 PHP 的 AWS 開發套件。如需詳細資訊，請參閱 。命名空間 AWS\ CodeBuild的 區段AWS適用 SDK for PHP 開發套件 API 參考。
- 適用於 Python 的 AWS 開發套件 (Boto3)。如需詳細資訊，請參閱 。CodeBuild的 區段博託 3 文件。
- 適用於 Ruby 的 AWS 開發套件。如需詳細資訊，請參閱 。模組：AWS:: CodeBuild的 區段AWS適用 SDK for Ruby 開發套件 API 參考。
- 適用於 PowerShell 的 AWS 工具。如需詳細資訊，請參閱 。AWS CodeBuild的 區段AWSTools for PowerShell Cmdlet 參考。

指定 AWS CodeBuild 端點

您可以使用 AWS Command Line Interface (AWS CLI) 或其中一個 AWS 軟體開發套件來指定 AWS CodeBuild 使用的端點。CodeBuild 可供使用的每個區域中會有一個端點。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。如果您沒有明確告知要使用CodeBuild哪個端點，則服務會使用與您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。如果您要使用 FIPS 端點，您必須使用以下其中一個方法，將 CodeBuild 與它產生關聯。

> **Note**
>
> 您可以使用別名或區域名稱，以使用 AWS 軟體開發套件指定端點。如果您使用 AWS CLI，則必須使用完整端點名稱。

如需可以與 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 開發套件)

指定 AWS CodeBuild 端點 (AWS CLI)

您可以使用 AWS CLI 來指定端點，透過該端點可在任何 CodeBuild 命令中使用 `--endpoint-url` 引數存取 AWS CodeBuild。例如，執行下列命令以取得美國東部 (維吉尼亞北部) 區域的聯邦資訊處理標準 (FIPS) 區域的專案組建名稱清單：

```
aws codebuild list-projects --endpoint-url https://codebuild-fips.us-
east-1.amazonaws.com
```

在端點的開頭包含 `https://`。

`--endpoint-url` AWS CLI 引數可供所有 AWS 服務使用。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 開發套件)

您可以使用 AWS 軟體開發套件來指定端點，透過該端點存取 AWS CodeBuild。雖然此範例使用 AWS SDK for Java，但您可以使用其他 AWS 軟體開發套件來指定端點。

構建AWSCodeBuild客戶端時使用該`withEndpointConfiguration`方法。這是要使用的格式：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",
 "region")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

如需相關資訊`AWSCodeBuildClientBuilder`，請參閱類別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。

請勿在端點的開頭包含 `https://`。

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	端點	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 FIPS 端點在美國西部 (奧勒岡) 區域的使用：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-
fips", "us-west-2")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

若要在美國東部 (維吉尼亞北部) 區域指定非 FIP 端點的使用：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1",
 "us-east-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

若要在亞太區域 (孟買) 區域的非 FIP 端點的使用方式：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1",
 "ap-south-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

搭配 AWS CodeBuild 使用 AWS CodePipeline 測試程式碼及執行建置

您可以使用 AWS CodePipeline 自動化您的發行程序來測試您的程式碼，並使用 AWS CodeBuild 執行您的建置。

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
建立持續傳遞 (CD) 管道 CodePipeline ，並自動化建置 CodeBuild	• CodePipeline 控制台 • AWS CLI • AWS SDK	• 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 SDK 一節中適用於您程式設計語言的`create-pipeline` 動作文件，或參閱 AWS CodePipelineAPI 參考*`CreatePipeline`* 中的。
將測試和構建自動化添加 CodeBuild 到中的現有管道 CodePipeline	• CodePipeline 控制台 • AWS CLI • AWS SDK	• 使用主 CodePipeline 控台新增組建自動化 • 使用 CodePipeline 控制台添加測試自動化 • 對於AWS CLI，您可以調整本主題中的資訊，以建立包含 CodeBuild 建置動作或測試動作的管道。如需詳細資訊，請參閱《AWS CodePipeline使用指南》中的編輯 CodePipeline 配管 (AWS CLI) 和配管結構參照。

任務	可用方法	本主題中說明的方法
		• 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 Amazon Web Services 工具的 SDK 一節，針對您的程式設計語言的update-pipeline 動作文件，或參閱 AWS CodePipelineAPI 參考*UpdatePipeline* 中的說明文件。

必要條件

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的受管政策附加到使用者 (或使用者所屬的 IAM 群組)。不建議使用 AWS 根帳戶。此原則授予使用者在 CodePipeline 中建立管道的許可。如需詳細資訊，請參閱附加使用指南中的受管理策略。

   > **Note**
   >
   > 將政策附加到使用者 (或使用者所屬的 IAM 群組) 的 IAM 實體必須具有 IAM 中的權限才能附加政策。如需詳細資訊，請參閱使用者指南中的委派許可以管理 IAM 使用者、群組和登入資料。

3. 如果您的AWS帳戶中還沒有可用的 CodePipeline 服務角色，請建立服務角色。 CodePipeline 使用此服務角色與其他AWS服務互動AWS CodeBuild，包括代表您。例如，若要使用AWS CLI建立 CodePipeline 服務角色，請執行 IAM `create-role` 命令：

   若為 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 實體必須具有 IAM 中的許可，才能建立服務角色。

4. 建立 CodePipeline 服務角色或識別現有角色之後，您必須將預設 CodePipeline 服務角色原則新增至服務角色原則，如AWS CodePipeline使用者指南中的檢閱預設服 CodePipeline 務角色原則 (如果該角色尚未包含該角色原則的一部分)，則必須將預設服務角色原則新增至服務角色原則。

> **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. 建立原始程式碼並將其上傳至 CodeBuild 和支援的儲存庫類型 CodePipeline CodeCommit，例如 Amazon S3、Bitbucket 或 GitHub. 來源碼應該包含 Buildspec 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 Buildspec 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。
>
> - 對於AWS OpsWorks，請參閱《使用指南》AWS OpsWorks中的應 CodePipeline 用程式來源和搭配AWS OpsWorks使用。

主題

- 建立使用 CodeBuild 的管道 (CodePipeline 主控台)
- 建立使用 CodeBuild 的管道 (AWS CLI)
- 將 CodeBuild 建置動作新增至管道 (CodePipeline 主控台)
- 將 CodeBuild 測試動作新增至管道 (CodePipeline 主控台)

建立使用 CodeBuild 的管道 (CodePipeline 主控台)

請遵循以下程序來建立使用 CodeBuild 的管道，以建立及部署您的來源碼。

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線。
- 使用本主題中其他程序中的其中一項來建立管線，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線。

使用 CodePipeline 中的建立管線精靈建立使用 CodeBuild 的管線

1. 使用下列方法登入 AWS Management Console：
   - 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號根使用者。
   - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的「建立您的第一個AWS 帳戶 root 使用者和群組」。
   - 您AWS帳戶中有權使用下列最少動作集的使用者：

```
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. 請在以下位置開啟AWS CodePipeline主控台。 https://console.aws.amazon.com/codesuite/codepipeline/home
3. 在 [地AWS區] 選取器中，選擇建置專案AWS資源所在的AWS區域。這必須CodeBuild是受支援的AWS地區。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。若顯示 CodePipeline 資訊頁面，選擇 Create pipeline (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 Existing service role (現有服務角色)，然後選擇您在本主題之先決條件中建立或找到的 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：

   - 選擇預設位置以針對您為管道選取的區域中的管道使用預設成品存放AWS區，例如指定為預設值的 S3 成品儲存貯體。
   - 如果您已經在與管道位於相同區域中建立的現有成品存放AWS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。每個管道都需要有個別成品存放AWS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：

- 如需存放在 S3 儲存貯體的原始程式碼，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 object key (S3 物件金鑰)，輸入包含來源碼之檔案的名稱 (例如，*file-name*.zip)。選擇 下一步。
- 若您的來源碼存放於 AWS CodeCommit 儲存庫，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。
- 如果您的原始程式碼儲存在儲存GitHub庫中，請選擇GitHub。選擇 [Connect 至]GitHub，然後依照指示進行驗證GitHub。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 Step 3: Add build stage (步驟 3：新增建置階段) 頁面，針對 Build provider (建置提供者)，選擇 CodeBuild。
11. 如果您已經有要使用的組建專案，請選擇建置專案的名稱做為 [專案名稱]，然後跳到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。

    若您選擇現有的建置專案，則該專案必須具備已定義的建置輸出成品設定 (即使 CodePipeline 會覆寫它們)。如需詳細資訊，請參閱變更建置專案的設定 (主控台)。

    > **Important**
    >
    > 若您啟用 CodeBuild 專案的 Webhook，且專案是做為 CodePipeline 中的組建步驟使用，則會為每一次的遞交建立兩個完全相同的組建。其中一個組建是透過 Webhook 觸發，另一個則是透過 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，若您使用 CodePipeline，我們建議您在 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。當管道在 CodePipeline 主控台中顯示時，在 Build (建置) 動作內，選擇工具提示。記下「輸出」加工品的值 (例如，MyAppBuild)。

    > **Note**
    >
    > 您也可以在CodeBuild主控台的 [組建詳細資料] 頁面上選擇 [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)。

15. 請在 https://console.aws.amazon.com/s3/ 開啟 Amazon Simple Storage Service (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AWS CLI用執行命CodePipelineget-pipeline令來取得值區的名稱，其中*`my-pipeline-name`*是管線的顯示名稱：

    ```
    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. 若您指示 CodePipeline 部署組建輸出成品，請使用部署提供者的說明以取得部署目標上的部署輸出成品。

建立使用 CodeBuild 的管道 (AWS CLI)

請遵循以下程序來建立使用 CodeBuild 的管道，來組建您的來源碼。

若要使用建立管道來部署您的建置原始程式碼或僅測試您的原始程式碼，您可以調整使用AWS CodePipeline者指南中編輯管道 (AWS CLI) 和CodePipeline管線結構參考中的指示。AWS CLI

1. 建立或找出 CodeBuild 中的組建專案。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 會覆寫它們)。如需詳細資訊，請參閱建立建置專案 (AWS CLI)中的 `artifacts` 描述。

2. 請確定您已使AWS CLI用與本主題中描述的其中一個 IAM 實體相對應的存取金鑰和AWS秘密存取金鑰進行設定。AWS若要取得更多資訊，請參閱《AWS Command Line Interface使用者指南》AWS Command Line Interface中的〈進行設定〉。

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 格式結構會使用參考 S3 輸入儲存貯體的來源動作，以及使用 CodeBuild 的建置動作建立管線：

```
{
  "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：

- `roleArn` 的值必須符合您在事前準備中建立或找到之 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。如需其他原始程式碼儲存庫類型的設定，請參閱《AWS CodePipeline使用指南》中的CodePipeline管線結構參考。
- `ProjectName` 的值為您在此程序中先前建立的 CodeBuild 建置專案名稱。
- `location` 的值為此管線使用的 S3 儲存貯體名稱。如需詳細資訊，請參閱使用AWS CodePipeline者指南CodePipeline中的建立 S3 儲存貯體用作成品存放區的政策。
- `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，您可以新增與測試、部署建置輸出成品、呼叫 AWS Lambda 函數等有關的活動動作。如需詳細資訊，請參閱《AWS CodePipeline使用指南》中的AWS CodePipeline配管結構參照。

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 地區中建立管道。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 AWS CodeBuild。

JSON 格式資料會顯示在輸出中，並且 CodePipeline 會建立管線。

5. 若要取得管道狀態的資訊，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。請在 https://console.aws.amazon.com/s3/ 開啟 Amazon Simple Storage Service (Amazon S3) 主控台。

> **Note**
>
> 您也可以在CodeBuild主控台的相關組建詳細資料頁面上選擇 [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，或是執行 CodePipeline get-pipeline 命令取得儲存貯體名稱。

```
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 建置動作新增至管道 (CodePipeline 主控台)

1. 使用下列方法登入 AWS Management Console：

   - 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號根使用者。
   - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的「建立您的第一個AWS 帳戶 root 使用者和群組」。
   - 您AWS帳戶中有權執行下列最少動作集的使用者：

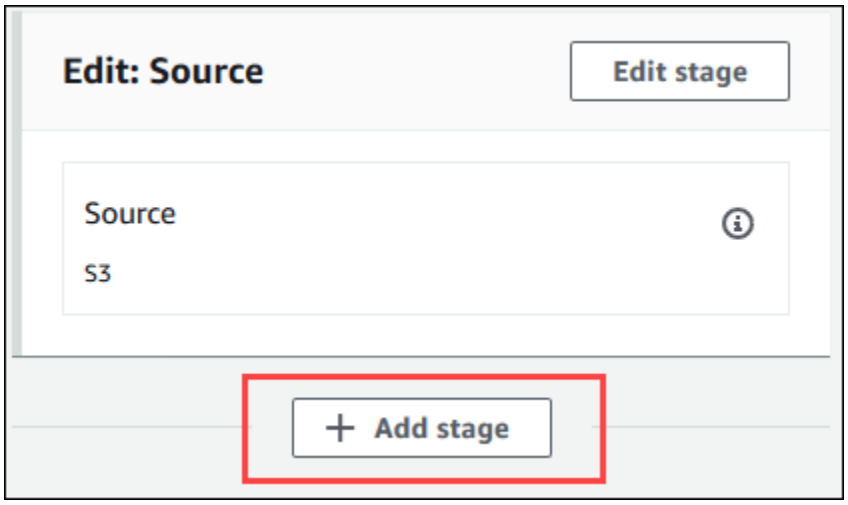
   ```
   codepipeline:*
   iam:ListRoles
   iam:PassRole
   s3:CreateBucket
   s3:GetBucketPolicy
   s3:GetObject
   s3:ListAllMyBuckets
   s3:ListBucket
   s3:PutBucketPolicy
   codecommit:ListBranches
   codecommit:ListRepositories
   codedeploy:GetApplication
   codedeploy:GetDeploymentGroup
   codedeploy:ListApplications
   codedeploy:ListDeploymentGroups
   elasticbeanstalk:DescribeApplications
   elasticbeanstalk:DescribeEnvironments
   lambda:GetFunctionConfiguration
   lambda:ListFunctions
   opsworks:DescribeStacks
   opsworks:DescribeApps
   opsworks:DescribeLayers
   ```

2. 請在以下位置開啟CodePipeline主控台。 https://console.aws.amazon.com/codesuite/codepipeline/home
3. 在AWS區域選取器中，選擇管線所在的AWS區域。這必須是支援 CodeBuild 的區域。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 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「輸出」加工品的值 (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 Action provider (動作提供者)，選擇 CodeBuild。
12. 如果您已經有要使用的組建專案，請選擇建置專案的名稱做為 [專案名稱]，然後跳到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。

    若您選擇現有的建置專案，則該專案必須具備已定義的建置輸出成品設定 (即使 CodePipeline 會覆寫它們)。如需詳細資訊，請參閱建立組建專案 (主控台)或中的人工因素說明變更建置專案的設定 (主控台)。

    > **Important**
    >
    > 若您啟用 CodeBuild 專案的 Webhook，且專案是做為 CodePipeline 中的組建步驟使用，則會為每一次的遞交建立兩個完全相同的組建。其中一個建置是透過 Webhook 觸發，另一個則是透過 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，若您使用 CodePipeline，我們建議您在 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 Add action (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。當管道在 CodePipeline 主控台中顯示時，在 Build (建置) 動作內，選擇工具提示。記下「輸出」加工品的值 (例如，MyAppBuild)。

> **Note**
>
> 您也可以在CodeBuild主控台的 [組建詳細資料] 頁面上選擇 [組建成品] 連結，以取得組建輸出成品。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. 請在 https://console.aws.amazon.com/s3/ 開啟 Amazon Simple Storage Service (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，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. 若您指示 CodePipeline 部署組建輸出成品，請使用部署提供者的說明以取得部署目標上的部署輸出成品。

將 CodeBuild 測試動作新增至管道 (CodePipeline 主控台)

1. 使用下列方法登入 AWS Management Console：
    - 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號根使用者。
    - 您AWS帳戶中的系統管理員使用者。如需詳細資訊，請參閱《使用指南》中的「建立您的第一個AWS 帳戶 root 使用者和群組」。
    - 您AWS帳戶中有權執行下列最少動作集的使用者：

    ```
    codepipeline:*
    iam:ListRoles
    ```

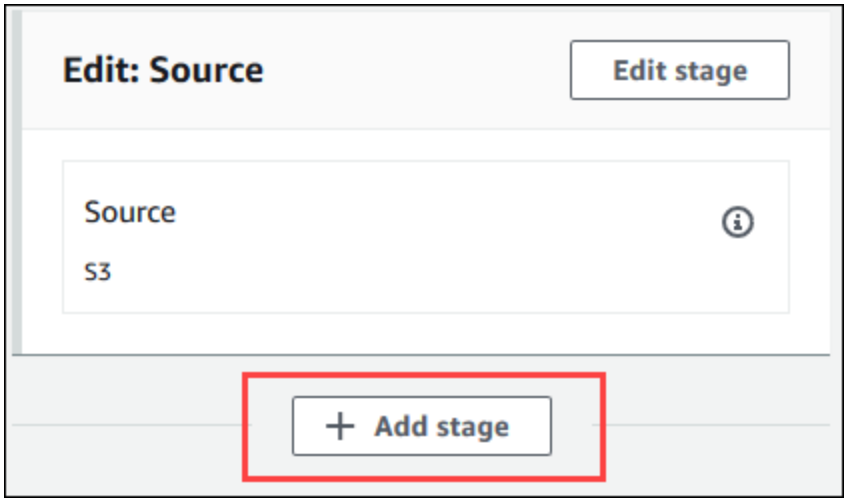
```
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. 請在以下位置開啟CodePipeline主控台。 https://console.aws.amazon.com/codesuite/codepipeline/home
3. 在AWS區域選取器中，選擇管線所在的AWS區域。這必須CodeBuild是受支援的AWS地區。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。記下輸出加工品的值 (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. 針對 Action provider (動作提供者)，於 Test (測試) 下選擇 CodeBuild。
12. 如果您已經有要使用的組建專案，請選擇建置專案的名稱做為 [專案名稱]，然後跳到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**
>
> 若您啟用 CodeBuild 專案的 Webhook，且專案是做為 CodePipeline 中的組建步驟使用，則會為每一次的遞交建立兩個完全相同的組建。其中一個建置是透過 Webhook 觸發，另一個則是透過 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，若您使用 CodePipeline，我們建議您在 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 Buildspec 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 Test (測試) 階段內，選擇 CodeBuild 超連結，在 CodeBuild 主控台中開啟相關建置專案頁面。
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 [檢視整個記錄超連結]，在 AmazonCloudWatch 主控台中開啟建置記錄。
20. 捲動組建日誌，檢視測試結果。

使用 AWS CodeBuild 搭配 Jenkins

您可以使用詹金斯插件 CodeBuild 與您的詹金斯構建工作集成。AWS CodeBuild不需將組建工作傳送至 Jenkins 組建節點，而是可以使用外掛程式來將組建工作傳送至 CodeBuild。透過此方式，您便不需佈建、設定和管理 Jenkins 組建節點。

設定 Jenkins

如需使用外掛程式設定 Jenkins，以及下載AWS CodeBuild外掛程式原始程式碼的相關資訊，請參閱 https://github.com/awslabs/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

使用 AWS CodeBuild 搭配來自 VPC 外部的來源

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)。
   - 選擇您要執行組建的多個AWS區域。
   - (選擇性) 設定 Amazon VPC 組態以允許 CodeBuild 建置容器存取 VPC 中的資源。
   - 寫下專案的名稱。您在步驟 3 會用到。
   - (選擇性) 如果原生不支援您的來源儲存庫 CodeBuild，您可以將 Amazon S3 設定為專案的輸入來源類型。
2. 在 IamConsole 中，創建一個由詹金斯插件使用的用戶。
   - 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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:logs:{{region}}:{{awsAccountId}}: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:{{region}}:{{awsAccountId}}: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 [執行建置於] 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 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 服務器上安裝插件（或源存儲庫提供程序的 Jenkins 插件）。 GitHub

   - 在 [設定] 頁面上，選擇 [新增建置步驟]，然後選擇 [執行建置於]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。

使用 AWS CodeBuild 外掛程式搭配 Jenkins 管道外掛程式

- 在您的 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，該指令碼會新增 CodeBuild 為管線中的一個步驟。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'
```

搭配 Codecov 使用 AWS CodeBuild

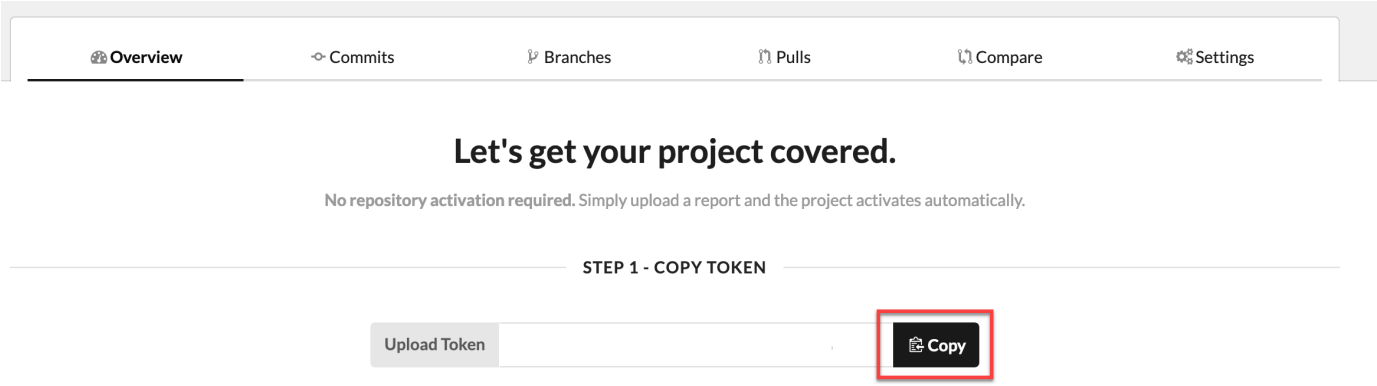
Codecov 是一種用於測量程式碼測試涵蓋範圍的工具。Codecov 可識別程式碼中的哪些方法和陳述式尚未經過測試。使用結果來決定要將測試寫入哪個位置，提升程式碼的品質。Codecov 適用於 Codecov 支援的三個來源儲存庫：GitHub、GitHub Enterprise Server 和 Bitbucket。如果您的建置專案使用 GitHub Enterprise Server，則必須使用 Codecov Enterprise。

當您執行與 Codecov 整合的 Codecov 建置時，Codecov 會報告儲存庫中的分析程式碼已上傳至 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

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，請參閱在 AWS CodeBuild 中執行建置和使用 AWS CloudTrail 記錄 AWS CodeBuild API 呼叫。建置日誌中的 Codecov 資訊看起來類似下列內容：

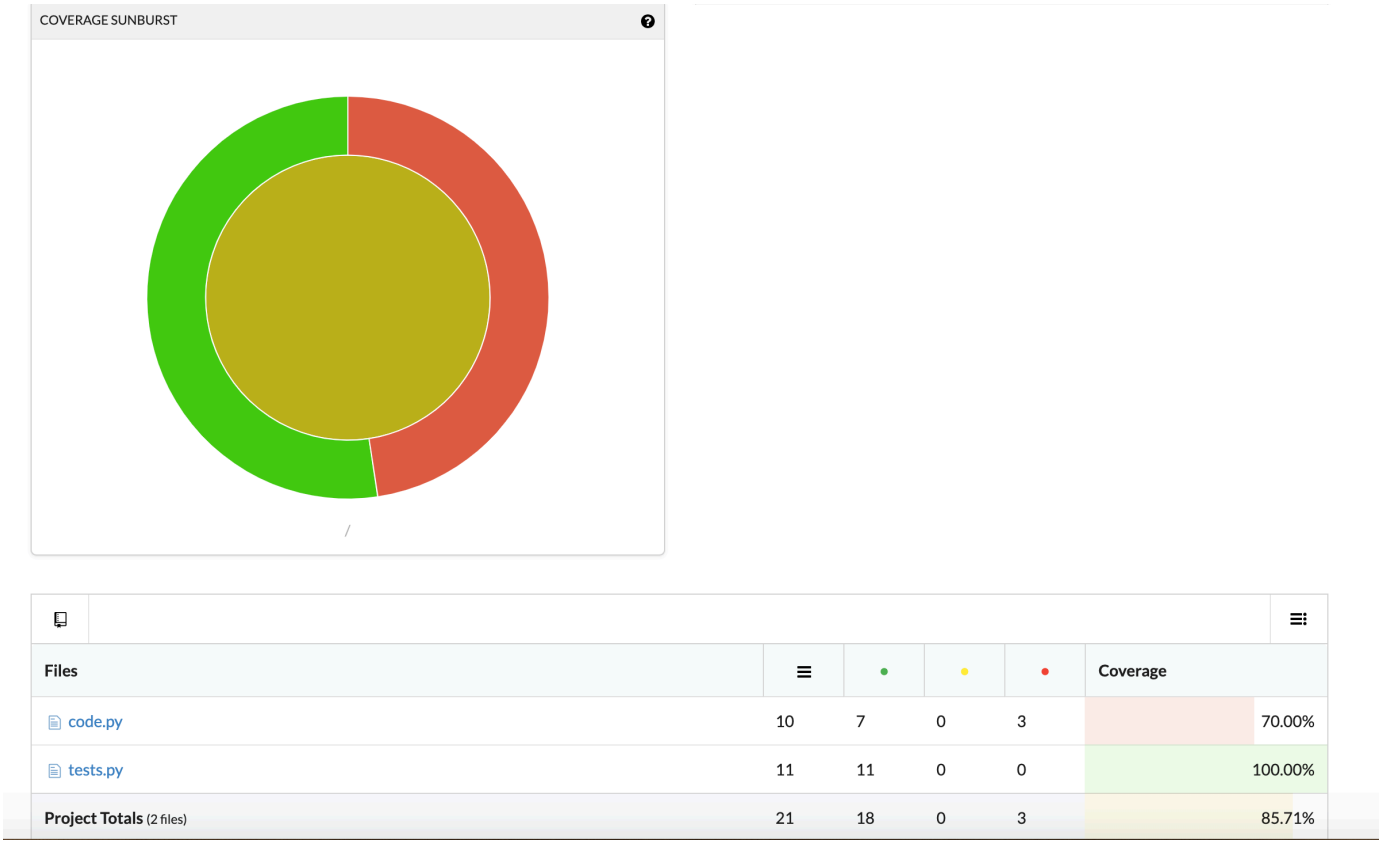
```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh


  _____          _
 / ____|        | |
| |     ___   __| | ___  ___ _____   __
| |    / _ \ / _` |/ _ \/ __/ _ \ \ / /
| |___| (_) | (_| |  __/ (_| (_) \ V /
 \_____\___/ \__,_|\___|\___\___/ \_/
                                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：



使用AWS CodeBuild與無伺服器應用程式

所以此AWS Serverless Application Model(AWS SAM）是一項開放原始碼架構，可用於建置無伺服器應用程式。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AWS SAM標準。針對部署步驟，CodeBuild 可以使用AWS CloudFormation。若要使用 CodeBuild 和AWS CloudFormation，您可以使用AWS CodePipeline。

如需詳細資訊，請參閱「」部署無伺服器應用程式中的AWS Serverless Application Model開發人員指南。

相關資源

- 如需 AWS CodeBuild 入門的相關資訊，請參閱使用主控台開始使用 AWS CodeBuild。
- 如需故障診斷 CodeBuild 中的問題的資訊，請參疑難排 AWS CodeBuild。
- 如需有關 CodeBuild 中的配額的詳細資訊，請參AWS CodeBuild 的配額。

疑難排 AWS CodeBuild

使用本主題的資訊來協助您查明、診斷和解決問題。若要瞭解如何記錄和監視 CodeBuild 組建以進行問題疑難排解，請參閱記錄和監控。

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。 英文字符
- 從 Amazon EC2 參數存放區取得參數時，組建可能會失敗
- 無法在 CodeBuild 主控台存取分支篩選條件
- 無法檢視組建成功或失敗
- 未報告給來源提供者的組建狀態
- 無法找到並選擇 Windows 服務器核心 2019 平台的基本映像
- 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：「構建容器在完成構建之前發現已死機。構建容器因內存不足而死亡，或者不支持 Docker 映像。 ErrorCode:
- 執行組建時發生錯誤：「無法連接到 Docker 協助程式」
- 建立或更新建置專案時，錯誤：AssumeRole"CodeBuild 未授權執行:sts:"
- 錯誤：「調用錯誤 GetBucketAcl：存儲桶所有者已更改或服務角色不再具有調用 s3 的權限：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：「無法下載緩存： RequestError：發送請求失敗，原因是：x509：無法加載系統根目錄並且沒有提供根目錄」

- 錯誤：「無法從 S3 下載憑證。 AccessDenied」
- 錯誤：「找不到登入資料」
- RequestError 在代理服務器 CodeBuild 中運行時出現超時錯誤
- 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 主控台時出現「無法驗證 JobWorker 身分」
- 建置無法啟動
- 存取本機快取組建中的中 GitHub 繼資
- AccessDenied：報表群組的儲存貯體擁有者不符合 S3 儲存貯體的擁有者...

Apache Maven 建置參考錯誤儲存庫中的成品

問題：當您在 AWS CodeBuild提供的 Java 構建環境中使用 Maven 時，Maven 會從 https://repo1.maven.org/maven2 的安全中央 Maven 存儲庫中提取構建和插件依賴關係。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。

可能的原因： CodeBuild-提供的 Java 構建環境包含一個名為的文件，`settings.xml`該文件已預先安裝在構建環境的`/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. 在構建項目的`install`階段，指示將`settings.xml`文件複製 CodeBuild 到構建環境的`/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

問題：以 root 使用者身分 AWS CodeBuild 執行您的建置命令。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：默認情況下，以 root 用戶身份 CodeBuild 運行所有構建命令。

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。 英文字符

問題：當您執行組建時，使用檔案名稱包含非 U.S. 英文字元 (例如，中文字元)，建置失敗。

可能的原因：所提供的建置環境 AWS CodeBuild 的預設語言環境設定為POSIX。 POSIX當地語系化設定 CodeBuild 與包含非美國的檔案名稱較不相容 英文字元，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。這些指令會使建置環境使用美式英文 UTF-8 作為其本地化設定，此設定與包含非美式的檔案名稱 CodeBuild 和檔案名稱更相容。 英文字符。

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 locales
```

對於基於 Amazon Linux 的構建環境：

```
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 Amazon EC2 參數存放區中存放的一或多個參數值時，組建會在DOWNLOAD_SOURCE階段失敗並顯示錯誤Parameter does not exist。

可能的原因：組建專案所依賴的服務角色沒有呼叫ssm:GetParameters動作的權限，或者組建專案使用由動作產生 AWS CodeBuild 且允許呼叫ssm:GetParameters動作的服務角色，但參數的名稱不是以開頭/CodeBuild/。

建議解決方案：

- 如果服務角色不是由所產生 CodeBuild，請更新其定義 CodeBuild 以允許呼叫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：

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

```
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/CodeBuild/*"
    }
  ]
}
```

- 如果服務角色是由產生的 CodeBuild，請更新其定義， CodeBuild 以允許在 Amazon EC2 參數存放區中存取名稱以外的參數/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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/PARAMETER_NAME"
    }
  ]
}
```

無法在 CodeBuild 主控台存取分支篩選條件

問題：當您建立或更新專案時，主控台中無法使用分支篩選器選 AWS CodeBuild 項。

可能的原因：分支篩選選項已棄用。它已由 Webhook 篩選群組取代，可更精確控制哪些 Webhook 事件會在 CodeBuild 中觸發新的建置。

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 比特桶網絡鉤事件 及 過濾 GitHub 網絡掛鉤事件（控制台）。

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：在建立或更新 CodeBuild 專案時啟用報表組建狀態。此選項告知 CodeBuild 在您觸發建置時回報狀態。如需詳細資訊，請參閱 AWS CodeBuild API 參考中的 reportBuildStatus。

未報告給來源提供者的組建狀態

問題：允許向來源提供者 (例如 GitHub 或 Bitbucket) 報告建置狀態後，不會更新組建狀態。

可能的原因：與來源提供者關聯的使用者沒有存放庫的寫入權限。

建議的解決方案：若要能夠向來源提供者報告組建狀態，與來源提供者相關聯的使用者必須具有存放庫的寫入權限。如果使用者沒有寫入權限，則無法更新組建狀態。如需詳細資訊，請參閱 來源提供者存取。

無法找到並選擇 Windows 服務器核心 2019 平台的基本映像

問題：您無法找到或選取 Windows 伺服器核心 2019 平台的基本映像。

可能的原因：您使用的 AWS 地區不支援此影像。

建議的解決方案：使用下列其中一個 AWS 區域，其中支援 Windows 伺服器核心 2019 平台的基本映像：

- 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所有其他命令隔離執行。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：建議您使用建置規格 0.2 版，即可解決此問題。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。
- 所使用的服務角色 CodeBuild 沒有`s3:GetObject`持有快取之 S3 儲存貯體的`s3:PutObject`權限。

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。

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`。

可能的原因：組建映像檔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，或減少自訂組建映像檔的大小。

可能的原因： AWS CodeBuild 沒有權限從 Amazon Elastic Container Registry (Amazon ECR) 提取構建映像。

建議的解決方案：更新 Amazon ECR 儲存庫中的許可， CodeBuild 以便將自訂組建映像提取到建置環境中。如需更多資訊，請參閱 Amazon ECR 樣本。

可能的原因：您要求的 Amazon ECR 映像無法在您的 AWS 帳戶使用的 AWS 地區使用。

建議的解決方案：使用與您 AWS 帳戶使用的區域位於相同 AWS 區域的 Amazon ECR 映像檔。

可能的原因：您在沒有公共互聯網訪問權限的 VPC 中使用私有註冊表。 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 帶有 AWS Secrets Manager 示例的私人註冊表 CodeBuild。

推薦的解決方案：如果您在 VPC 中使用私有註冊表，請確保 VPC 具有公共 Internet 訪問權限。

可能的原因：如果錯誤消息包含 toomanyrequests「「，並且映像是從 Docker Hub 獲取，則此錯誤表示已達到 Docker Hub 提取限制。

建議的解決方案：使用泊塢集線器私有註冊表，或從 Amazon ECR 獲取映像。如需使用私人登錄的詳細資訊，請參閱 帶有 AWS Secrets Manager 示例的私人註冊表 CodeBuild。如需使用 Amazon ECR 的詳細資訊，請參閱Amazon ECR 樣品 CodeBuild 。

錯誤：「構建容器在完成構建之前發現已死機。構建容器因內存不足而死亡，或者不支持 Docker 映像。 ErrorCode:

問題：當您嘗試在中使用 Microsoft 視窗或 Linux 容器時 AWS CodeBuild，會在佈建階段發生此錯誤。

可能原因：

- 不支援容器作業系統版本 CodeBuild。
- 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：要修復此錯誤，您必須啟用特權模式並使用以下說明更新 buildspec。

若要以特殊權限模式執行組建，請依照下列步驟執行：

1. 請在以下位置開啟 CodeBuild 主控台。 https://console.aws.amazon.com/codebuild/
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 映像或希望組建取得提升的權限，請從「授權」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。

您還需要在容器內啟動 Docker 守護進程。你的構建規格的`install`階段可能看起來類似於這個。

```
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"
> ```

若要深入瞭解如何使用建置和執行 Docker 映像檔 AWS CodeBuild，請參閱自訂影像範例中的泊塢視窗 CodeBuild 。

建立或更新建置專案時，錯誤：AssumeRole"CodeBuild 未授權執行:sts:"

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的 AWS 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，或者沒有足夠的權限可以信任 CodeBuild。

建議解決方案：

- 請確 AWS STS 定您嘗試建立或更新建置專案的 AWS 區域已啟用。如需詳細資訊，請參閱 IAM 使用者指南AWS STS 中的在 AWS 區域中啟用和停用。
- 確定您的 AWS 帳戶中存在目標 CodeBuild 服務角色。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)。
- 請確定目標 CodeBuild 服務角色具有足夠的權限可以信任 CodeBuild。如需詳細資訊，請參閱建立 CodeBuild 服務角色中的信任關係政策陳述式。

錯誤：「調用錯誤 GetBucketAcl：存儲桶所有者已更改或服務角色不再具有調用 s3 的權限：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。

可能的原因：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，然後再次向 IAM 角色新增許可。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。

可能的原因：您的 S3 輸出儲存貯體 (AWS CodeBuild 儲存組建輸出的儲存貯體) 位於不同於 CodeBuild 建置專案的 AWS 區域。

建議的解決方案：更新組建專案的設定，以指向與建置專案位於相同 AWS 區域的輸出值區。

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「不安全的 SSL」可在連線至您的 GitHub 企業伺服器專案存放庫時忽略 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。

可能的原因：您預先建置的原始程式碼存放在 S3 儲存貯體中，而該儲存貯體位於與 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。請確定值區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。

推薦的解決方案：如果您使用`aws/codebuild/standard:2.0` CodeBuild 託管映像，則必須在 buildspec 文件的`runtime-versions`部分中指定運行時版本。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 標準映像 2.0 或更新版本以外的映像檔，或 Amazon Linux 2 (AL2) 標準映像 1.0 或更新版本，則組建會發出警告「`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 VPC 路由器保留。
- `10.0.0.2`：由保留 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`：保留 AWS 供 future 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：「無法下載緩存： RequestError：發送請求失敗，原因是：x509：無法加載系統根目錄並且沒有提供根目錄」

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。

推薦的解決方案：更新項目中正在使用的 AWS CodeBuild Docker 映像。如需詳細資訊，請參閱 碼頭圖片提供 CodeBuild。

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 或元件的版本與不相容 AWS CodeBuild。
- 您正在使用 Docker 的組建環境中執行 Docker 容器，且容器預設無法存取 AWS 認證。

建議解決方案：

- 確保您的構建環境具有以下版本或更高版本的 AWS CLI、 AWS SDK 或組件。
  - AWS CLI: 1.10.47
  - AWS SDK for C++：0.2.19
  - AWS SDK for Go：1.2.5
  - AWS SDK for Java：1.11.16
  - AWS 適用於 JavaScript：2.4.7 的開發套件
  - AWS SDK for PHP：3.18.28
  - AWS SDK for Python (Boto3): 1.4.0
  - AWS SDK for Ruby：2.3.22
  - Botocore: 1.4.37
  - CoreCLR: 3.2.6-beta
  - Node.js: 2.4.7

- 如果您需要在組建環境中執行 Docker 容器，而容器需要 AWS 認證，則必須將認證從組建環境傳遞至容器。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`此選項會透過容器存取 AWS 認證所需的環境變數。

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag aws s3 ls
```

- 如果您正在建置 Docker 映像，且組建需要 AWS 登入資料 (例如，從 Amazon S3 下載檔案)，則必須依下列方式將登入資料從建置環境傳遞至 Docker 建置程序。

  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。這些選`--build-arg`項會設定 Docker 建置程序存取 AWS 認證所需的環境變數。

```
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 .
```

RequestError 在代理服務器 CodeBuild 中運行時出現超時錯誤

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`從 CloudWatch 日誌。
- `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。
- 請依照下列步驟使用 Amazon S3 和 CloudWatch 日誌的私有端點：

  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 日誌端點，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. 確認已選取在您的 Amazon VPC 中啟用私人 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：您使用的是不是由提供的組建映像 AWS CodeBuild，而且您的組建失敗並顯示訊息`Build container found dead before completing the build`。

可能的原因：Bourne shell（`sh`）不包含在構建映像中。 CodeBuild 需`sh`要運行構建命令和腳本。

推薦的解決方案：如果構建映像`sh`中不存在，請確保在啟動任何其他使用映像的構建之前包含它。（CodeBuild 已包含`sh`在其構建映像中。）

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」

問題：執行建置時，建置日誌包含此警告。

可能的原因：您的組建未使用版本 1.0 或更新版本的 Amazon Linux 2 (AL2) 標準映像檔，或是 2.0 版或更新版本的 Ubuntu 標準映像檔，並且在組建規格檔案的一`runtime-versions`節中指定了執行階段。

建議的解決方案：確保 buildspec 檔案不包含 `runtime-versions` 區段。僅當您使用 Amazon Linux 2（AL2）標準映像或更高版本或 Ubuntu 標準映像版本 2.0 或更高版本時，才需要此`runtime-versions`部分。

錯誤：開啟 CodeBuild 主控台時出現「無法驗證 JobWorker 身分」

問題：開啟 CodeBuild 主控台時，會顯示「無法驗證 JobWorker 身分識別」錯誤訊息。

可能的原因：用於主控台存取的 IAM 角色有一個標籤`jobId`做為金鑰。此標籤鍵保留給 CodeBuild ，如果存在，則會導致此錯誤。

建議的解決方案：變更任何具有不同金鑰`jobId`的自訂 IAM 角色標籤，例如`jobIdentifier`.

建置無法啟動

問題：啟動組建時，您會收到組建無法啟動錯誤訊息。

可能的原因：已達到並行組建的數目。

推薦的解決方案：等到其他構建完成，或增加項目的同時構建限制，然後重新啟動構建。如需詳細資訊，請參閱 項目配置。

存取本機快取組建中的中 GitHub 繼資

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。

可能的原因：為構建啟用本地源緩存時，為該`.git`目錄 CodeBuild 創建一個 gitlink。這意味著該`.git`目錄實際上是包含目錄路徑的文本文件。

建議的解決方案：在所有情況下，都可以使用下列命令來取得 Git 中繼資料目錄。無論以下格式如何，此命令都可以工作`.git`：

```
git rev-parse --git-dir
```

AccessDenied：報表群組的儲存貯體擁有者不符合 S3 儲存貯體的擁有者...

問題：將測試資料上傳到 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：

- 為報表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符。
- 服務角色沒有值區的寫入存取權。

建議解決方案：

- 變更報表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 儲存貯體進行寫入存取。

AWS CodeBuild 的配額

下表列出 AWS CodeBuild 中的目前配額。除非另有指示，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	描述
每個專案的關聯標籤	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 (分鐘)	每個支持地區：480	否	建置逾時上限 (分鐘)
並發請求有關構建的信息	每個受支援的區域：100	否	您可以使用 AWS CLI 或 AWS SDK 一次要求相關資訊的組建數目上限。
並行請求建置專案的資訊	每個受支援的區域：100	否	您可以使用 AWS CLI 或 AWS SDK 一次要求相關資訊的建置專案數目上限。
同時運行為ARM Lamb達/10GB 環境的構建	每個受支援的區域：1	是	適用於 ARM Lambda /10GB 環境的同時執行組建的最大數量
同時運行為ARM Lambda /1GB 環境的構建	每個受支援的區域：1	是	適用於 ARM Lambda /1GB 環境的同時執行組建的最大數量

名稱	預設	可調整	描述
同時運行為ARM Lamb達/2GB 環境的構建	每個受支援的區域：1	是	適用於 ARM Lambda /2GB 環境的同時執行組建的最大數量
同時運行為ARM Lamb達/4GB 環境的構建	每個受支援的區域：1	是	適用於 ARM Lambda /4GB 環境的同時執行組建的最大數量
同時運行為ARM Lamb達/8GB 環境的構建	每個受支援的區域：1	是	適用於 ARM Lambda /8GB 環境的同時執行組建的最大數量
同時運行為ARM /大型環境的構建	每個受支援的區域：1	是	ARM /大型環境同時執行組建的最大數量
同時運行為ARM /小型環境的構建	每個受支援的區域：1	是	針對 ARM /小型環境同時執行的組建最大數量
同時執行適用於 Linux GPU 大型環境的組建	每個支持地區：0	是	Linux GPU /大型環境同時執行組建的最大數量
同時執行適用於 Linux GPU 小型環境的組建	每個支持地區：0	是	針對 Linux GPU /小型環境同時執行的組建最大數量
同時執行適用於 Linux Lambda /10GB 環境的組建	每個受支援的區域：1	是	針對 Linux Lambda /10GB 環境的同時執行組建的最大數量
同時執行適用於 Linux Lambda /1GB 環境的組建	每個受支援的區域：1	是	針對 Linux Lambda /1GB 環境的同時執行組建的最大數量

名稱	預設	可調整	描述
同時運行為linux Lamb達/2GB 環境的構建	每個受支援的區域：1	是	針對 Linux Lambda /2GB 環境的同時執行組建的最大數量
同時執行適用於 Linux Lambda /4GB 環境的組建	每個受支援的區域：1	是	對於 Linux Lambda /4GB 環境的同時運行構建的最大數量
同時執行適用於 Linux Lambda /8GB 環境的組建	每個受支援的區域：1	是	對於 Linux Lambda /8GB 環境的同時運行構建的最大數量
同時針對 LinuX/2XL 環境執行組建	每個支持地區：0	是	在 LinuX/2XL 環境中同時執行的組建數目上限
同時運行為Linu/大型環境的構建	每個受支援的區域：1	是	Linu/大型環境同時執行組建的最大數量
同時針對 Linux/中型環境執行構建	每個受支援的區域：1	是	Linux/中型環境同時執行組建的最大數量
同時針對 Linux/小型環境執行構建	每個受支援的區域：1	是	LinuX/小型環境同時執行組建的最大數量
同時為 LinuX/XL 環境執行組建	每個受支援的區域：1	是	在 LinuX/XLARGE 環境中同時執行的組建數目上限
同時運行的構建視窗服務器2019/大型環境	每個受支援的區域：1	是	視窗服務器 2019 /大型環境同時運行的構建的最大數量
同時運行為視窗服務器2019/中型環境的構建	每個受支援的區域：1	是	視窗伺服器 2019 /中型環境同時執行的組建的最大數量

名稱	預設	可調整	描述
為Windows/大型環境同時運行構建	每個受支援的區域：1	是	Windows/大型環境同時執行的組建數上限
同時執行適用於視窗/中型環境的組建	每個受支援的區域：1	是	Windows/中環境同時執行的組建數目上限
建置逾時的最短期間 (分鐘)	每個受支援的區域：5	否	建置逾時下限 (分鐘)
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	可用於 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因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。若要要求較高的並行建置配額，或出現「帳戶的作用中組建不能超過 X 個」錯誤訊息，請使用上面的連結提出要求。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

資源	預設
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 255 個字元 (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 480 (8 小時)

建置數

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 480 (8 小時)

運算叢集

資源	預設
並行數量的運算叢集	10
同時執行ARM /小型環境群組的執行個體	1
同時執行ARM /大型環境群組的執行個體	1
同時執行 Linux/小型環境群組的執行個體	1
同時執行 Linux/中型環境群組的執行個體	1
同時執行 Linux/大型環境群組的執行個體	1

資源	預設
同時執行 LinuX/XLARGE 環境叢集的執行個體	1
同時執行 Linux/2XL 環境叢集的執行個體	0
同時執行 Linux GPU 的執行個體/小型環境叢集	0
同時執行 Linux GPU 的執行個體/大型環境叢集	0
視窗伺服器 2019 /中型環境機集同時執行的執行個體	1
視窗伺服器 2019 /大型環境機集同時執行的執行個體	1
適用於視窗伺服器 2022 /中型環境群組同時執行的執行個體	1
視窗伺服器 2022 /大型環境群組同時執行的執行個體	1

報告

資源	預設
測試報告建立後可用的最大期間	30 天
測試用例消息的最大長度	五千個字元
測試用例名稱的最大長度	千字元
每個 AWS 帳戶的報告群組上限數量	5,000
每個報告的測試案例上限數量	500

標籤

標記限制適用於組 CodeBuild 建專案和 CodeBuild 報表群組資源的標記。

資源	預設
資源標籤金鑰名稱	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 + - = . _ : / @ 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： • 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 + - = . _ : / @ 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： ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

第三方視窗 AWS CodeBuild 通知

當您使 CodeBuild 用 Windows 組建時，您可以選擇使用某些協力廠商套件和模組，讓您的建置應用程式能夠在 Microsoft Windows 作業系統上執行，並與某些協力廠商產品互通。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) 基本碼頭映像 — 視窗伺服器核心

(授權條款可在以下網址取得：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/useterms) 的約束。

2）基於窗口的碼頭圖像-巧克力

(授權條款可在以下位置取得:https://github.com/chocolatey/choco/blob/master/LICENSE)

版權所有 2011-目前的 RealDimensions 軟件有限公司

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）基於窗口的碼頭圖像-git 版本 2.16.2

(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）基於窗口的碼頭圖像-版本 15.0.26320.2 microsoft-build-tools

(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。軟體可能包含公用程式清單上的某些項目，網址為 https://docs.microsoft.com/en-us/visualstudio/productinfo/2015-redistribution-vs。 您可以將這些項目 (如果隨附於軟體) 複製並安裝到您或其他協力廠商電腦上，以偵錯和部署您使用該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。該軟件可能包括從 Microsoft 視窗組件; Microsoft 視窗服務器; Microsoft SQL 服務器; Microsoft 交換; Microsoft 辦公室; 和 Microsoft SharePoint。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。本軟體可能包含第三方元件，並附有個別法律聲明，或受其他合約管理，如軟體隨附的 ThirdPartyNotices 檔案中所述。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。這些授權的副本 (如果適用) 會包含在 ThirdPartyNotices 檔案中。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 的匯票或支援寄到此地址：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 二零一五年視覺工作室；
      - 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，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擔保免責聲明。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。在貴用戶當地法律允許的範圍內，MICROSOFT 排除適銷性、特定用途適用性及非侵權之默示擔保。

11損害賠償的限制和排除。您得向 MICROSOFT 及其供應商僅索取直接損害賠償最多 5.00 USD。您不能補償任何其他損害，包括衍生性、利潤損失、特殊、間接或附帶損害。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。

使用者授權合約識別碼：2015_ 更新 3 _ ShellsRedist <ENU>

5) 基於窗口的碼頭圖像-刪除。命令行-版本 4.5.1

(授權條款可在以下網址NuGet取得) https://github.com/ /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）基於窗口的碼頭圖像-netfx-4.6.2 開發包

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。

使用本補充，即表示您接受這些條款。如果你不接受他們, 不要使用本補充.

若您遵守這些授權條款，則您享有下列權利。

1. 可分發的代碼。補充是由可分發的代碼. 「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. 配送需求。對於您散佈的任何可發佈程式碼，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. 發佈限制。你可能不
      - 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）基於窗口的碼頭圖像-可視化工具，v 4.0

(授權條款可在以下位置取得: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。

9）基於窗口的碼頭圖像--4.6 netfx-pcl-reference-assemblies

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。

使用本軟體即表示您接受這些條款。如果您不接受它們，請勿使用該軟件。

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。該軟件是「可分發的代碼」。
         - 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. 配送需求。對於您散佈的任何可發佈程式碼，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. 發佈限制。你可能不
         - 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. 擔保免責聲明。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。在貴用戶當地法律允許的範圍內，MICROSOFT 排除適銷性、特定用途適用性及非侵權之默示擔保。

對於澳大利亞-根據澳大利亞消費者法，您擁有法定保證，並且這些條款中的任何內容都不會影響這些權利。

12補救和損害賠償的限制和排除。您得向 MICROSOFT 及其供應商僅索取直接損害賠償最多 5.00 USD。您不能補償任何其他損害，包括衍生性、利潤損失、特殊、間接或附帶損害。

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）基於窗口的碼頭圖像-可視化構建工具 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。該軟件可能包含 BuildServer .TXT 文件中列出的某些構建服務器組件，和/或位於本 Microsoft 軟件許可條款列 BuildeServer 表中列出的任何文件。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。該軟件可能包括從 Microsoft 視窗組件; Microsoft 視窗服務器; Microsoft SQL 服務器; Microsoft 交換; Microsoft 辦公室; 和 Microsoft SharePoint。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。本軟體可能包含第三方元件，並附有個別法律聲明，或受其他合約管理，如軟體隨附的 ThirdPartyNotices 檔案中所述。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. 擔保免責聲明。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。在貴用戶當地法律允許的範圍內，MICROSOFT 排除適銷性、特定用途適用性及非侵權之默示擔保。

12. 損害賠償的限制和排除。您得向 MICROSOFT 及其供應商僅索取直接損害賠償最多 5.00 USD。您不能補償任何其他損害，包括衍生性、利潤損失、特殊、間接或附帶損害。

    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）基於窗口的碼頭圖像-3-ondemand-package.cab microsoft-windows-netfx

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。

使用本補充，即表示您接受這些條款。如果你不接受他們, 不要使用本補充.

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）基於窗口的碼頭圖像-網絡 SDK

(可於以下網址取得) https://github.com/dotnet/core/blob/main/LICENSE.TXT

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。

AWS CodeBuild 使用者指南文件記錄

下表說明自上次發行版本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。

- 最新應用程式介面版本:

變更	描述	日期
更新內容：Lambda 運算映像	新增對 .NET 8 (`al-lambda/aarch64/dotnet8` 和`al-lambda/x86_64/dotnet8`) 的 Lambda 支援	2024年5月8日
更新的內容： AWS 受管理 (預先定義) 的政策 AWS CodeBuild	AWSCodeBuildAdminAccess AWSCodeBuildDeveloperAccess、和 AWSCodeBuildReadOnlyAccess 政策已更新，以反映 AWS CodeConnections 品牌重塑。	2024 年 4 月 30 日
新內容：比特桶應用程序密碼或訪問令牌	添加對比特桶訪問令牌的支持。	2024年4月11日
新內容：報告自動探索	CodeBuild 現在支援報表自動探索。	2024年4月4日
新內容：設置自託管的 GitHub 操作跑步者	為自託管的 GitHub 動作跑步者添加新內容	2024年4月2日
新內容： GitLab 連接	添加對 GitLab 和 GitHub 自我管理連接的支持。	2024年3月25日
新內容：添加新的 Webhook 事件和過濾器類型	添加對新的 webhook 事件（`RELEASED`和`PRERELEASED` ）和過濾器類型	2024年3月15日

	（TAG_NAME和RELEASE_NAME ）的支持。	
新內容：添加一個新的網絡掛鉤事件：PULL_REQUEST_CLOSED	添加對新的網絡掛鉤事件的支持：PULL_REQUEST_CLOSED 。	2024年2月20日
更新內容：碼頭圖片提供 CodeBuild	添加對視窗服務器核心 2019 的支持 (windows-base:2019-3.0)	2024年2月7日
更新內容：碼頭圖片提供 CodeBuild	添加對 Amazon Linux 2023 的新運行時的支持 () al2/aarch64/standard/3.0	2024年1月29日
新內容：預留容量	CodeBuild 現在支援中 CodeBuild的預留容量叢集。	2024年1月18日
新的運算類型	CodeBuild 現在支援 Linux 大型運算類型。如需詳細資訊，請參閱建置環境運算類型。	2024 年 1 月 8 日
更新內容：碼頭圖片提供 CodeBuild	添加對 Amazon Linux 2 (al2/standard/5.0) 和 Ubuntu (ubuntu/standard/7.0) 的新運行時的支持	2023 年 12 月 14 日
更新內容：碼頭圖片提供 CodeBuild	新增對 Lambda 運算映像檔的支援	2023 年 12 月 8 日
新內容： AWS Lambda 運算	為 AWS Lambda 運算新增內容	2023 年 11 月 6 日
新內容：在構建規格中使用 GitHub 操作語法	為在構建規格中使用 GitHub 操作語法添加新內容	2023 年 7 月 6 日
更新內容：碼頭圖片提供 CodeBuild	添加對 Amazon Linux 2 的支持 (al2/standard/5.0)	2023 年 5 月 17 日

下列項目的受管理策略變 CodeBuild	現已提供 CodeBuild 有關受 AWS 管理政策更新的詳細資料。如需詳細資訊，請參閱AWS 受管理策略的更CodeBuild 新。	2023 年 5 月 16 日
更新內容：碼頭圖片提供 CodeBuild	刪除對 Amazon Linux 2 的支持 (`al2/standard/3.0`) 並添加對 Amazon Linux 2 的支持 (`al2/standard/corretto8`) 和 Amazon Linux 2 (`al2/standard/corretto11`)	2023 年 5 月 9 日
更新內容：碼頭圖片提供 CodeBuild	添加對 Ubuntu 的支持 () `ubuntu/standard/7.0`	2023 年 4 月 13 日
更新內容：碼頭圖片提供 CodeBuild	刪除對 Ubuntu 的支持 18.04 (`ubuntu/standard/4.0`) 和 Amazon Linux 2 (`al2/aarch64/standard/1.0`)	2023 年 3 月 31 日
更新內容：移除 VPC 限制	移除下列限制：如果您設定 CodeBuild 為使用 VPC，則不支援本機快取。從 22 年 2 月 28 日開始，您的 VPC 組建將花費更長的時間，因為每個組建都會使用新的 Amazon EC2 執行個體。	2023 年 3 月 1 日
更新內容：碼頭圖片提供 CodeBuild	刪除對 Ubuntu 的支持 18.04 (`ubuntu/standard/3.0`) 和 Amazon Linux 2 (`al2/standard/2.0`)	2022 年 6 月 30 日

Amazon ECR 示例：限制圖像訪問	使用 CodeBuild 登入資料提取 Amazon ECR 映像時，您可以限制對特定 CodeBuild 專案的影像存取。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	下列其他區域現在支援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	如果您設定 CodeBuild 為使用 VPC，則不支援本機快取。從 22 年 2 月 28 日開始，您的 VPC 組建將花費更長的時間，因為每個組建都會使用新的 Amazon EC2 執行個體。	2022 年 2 月 25 日
Batch 報告模式	CodeBuild 現在可讓您選取如何將批次建置狀態傳送至專案的來源提供者。如需詳細資訊，請參閱 Batch 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
公開建置專案	CodeBuild 現在可讓您將建置專案的建置結果提供給公眾使用，而不需要存取 AWS 帳戶。如需詳細資訊，請參閱公用建置專案。	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。有關更多信息，請參閱構建圖和構建列表。	2021 年 3 月 3 日
專案層次並行建置限制	CodeBuild 現在允許您限制構建項目的並發構建數量。如需詳細資訊，請參閱專案組態和concurrentBuildLimit。	2021 年 2 月 16 日
新的構建規格屬性：s3 前綴	CodeBuild 現在提供構件的 `s3-prefix` Buildspec 屬性，可讓您為上傳到 Amazon S3 的成品指定路徑前置詞。如需詳細資訊，請參閱 S3 前置詞。	2021 年 2 月 9 日
新的構建規格屬性：失敗	CodeBuild 現在為構建階段提供 `on-failure` buildspec 屬性，允許您確定構建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
新的構建規格屬性：排除路徑	CodeBuild 現在提供構件的 `exclude-paths` buildspec 屬性，可讓您從組建成品中排除路徑。如需詳細資訊，請參閱排除路徑。	2021 年 2 月 9 日
新的構建規格屬性：啟用符號鏈接	CodeBuild 現在提供構件的 `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，請參閱啟用符號連結。	2021 年 2 月 9 日

建構規格人工因素名稱增強	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
代碼覆蓋率報告	CodeBuild 現在提供程式碼涵蓋率報告。如需詳細資訊，請參閱程式碼涵蓋率報告。	2020 年 7 月 30 日
Batch 建置	CodeBuild 現在支持運行項目的並發和協調構建。如需詳細資訊，請參閱 Batch 建置於 CodeBuild.	2020 年 7 月 30 日
視窗服務器 2019 圖片	CodeBuild 現在提供了一個視窗服務器核心 2019 構建映像。如需詳細資訊，請參閱由 CodeBuild提供的 Docker 映像檔。	2020 年 7 月 20 日
會話管理器	CodeBuild 現在可讓您暫停執行中的組建，然後使用 AWS Systems Manager 工作階段管理員連線至組建容器，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 現在支持在 buildspec 文件中指定要在其構建環境中使用的 shell。如需詳細資訊，請參閱組建規格參考。	2020 年 6 月 25 日
測試報告與測試框架	已新增幾個主題，說明如何使用多個 CodeBuild 測試架構產生測試報告。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

更新的主題	CodeBuild 現在支援將標籤新增至報表群組。如需詳細資訊，請參閱ReportGroup。	2020 年 5 月 21 日
Support 測試報告	CodeBuild 測試報告的支援現已正式推出。	2020 年 5 月 21 日
更新的主題	CodeBuild 現在支持為 Github 和 Bitbucket 創建創建 webhook 過濾器，該過濾器僅在頭提交消息與指定的表達式匹配時才觸發構建。有關更多信息，請參閱提GitHub 取請求和 Webhook 過濾器樣本和 Bitbucket 提取請求和 Webhook 過濾器示例。	2020 年 5 月 6 日
新主題	CodeBuild 現在支援共用建置專案和報表群組資源。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
新主題和更新的主題	CodeBuild 現在支持在構建項目運行期間的測試報告。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

更新的主題	CodeBuild 現在支援所有組建上的組建編號、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
更新的主題	CodeBuild 現在支持安卓版本 29 和圍棋版本 1.13 運行時。如需詳細資訊，請參閱由 Docker 映像檔 CodeBuild和 Build spec 語法提供。	2019 年 9 月 10 日
更新的主題	建立專案時，您現在可以選擇 Amazon Linux 2 (AL2) 受管的映像。有關更多信息，請參閱構建規格文件示例中提供的 Docker 映像 CodeBuild和運行時版本。 CodeBuild	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，請參閱中的建立組建專案 CodeBuild。	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，請參閱中的組建快取 CodeBuild。	2019 年 2 月 21 日
新主題	CodeBuild 現在支援 webhook 篩選器群組，以指定觸發組建的事件。如需詳細資訊，請參閱篩選 GitHub 網路掛接事件和篩選位桶網路掛接事件。	2019 年 2 月 8 日
新主題	用 CodeBuild 戶指南現在介紹了如何 CodeBuild 與代理服務器一起使用。如需詳細資訊，請參閱搭 CodeBuild 配 Proxy 伺服器使用。	2019 年 2 月 4 日
更新的主題	CodeBuild 現在支援使用另一個 AWS 帳戶中的 Amazon ECR 映像檔。已更新數個主題以反映此變更，包括用於的 Amazon ECR 範例 CodeBuild、建立組建專案和建立 CodeBuild服務角色。	2019 年 1 月 24 日
Support 私人泊塢視窗登錄	CodeBuild 現在支援使用儲存在私人登錄中的 Docker 映像檔做為您的執行階段環境。如需詳細資訊，請參閱含 AWS Secrets Manager 範例的私人登錄。	2019 年 1 月 24 日

更新主題	CodeBuild 現在支持使用訪問令牌連接 GitHub（使用個人訪問令牌）和 Bitbucket（使用應用程序密碼）存儲庫。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 現在支援新的組建指標，以測量組建中每個階段的持續時間。如需詳細資訊，請參閱 CodeBuild CloudWatch 量度。	2018 年 11 月 15 日
VPC 端點策略主題	CodeBuild 現在支援政策的 Amazon VPC 端點。如需詳細資訊，請參閱建立的 CodeBuild VPC 端點原則。	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 樣本	CodeBuild 可以使用專案建置規格檔案中的命令，在建置期間掛載 Amazon EFS 檔案系統。如需詳細資訊 Amazon EFS 參閱 CodeBuild.	2018 年 10 月 26 日
比特桶鉤	CodeBuild 當您將 Bitbucket 用於存儲庫時，現在支持網絡掛鉤。如需詳細資訊，請參閱. CodeBuild	2018 年 10 月 2 日
S3 日誌	CodeBuild 現在支援 S3 儲存貯體中的建置日誌。以前，您只能使用記錄檔建置 CloudWatch 記錄檔。如需詳細資訊，請參閱建立專案。	2018 年 9 月 17 日

多個輸出來源和多個輸出人工因素	CodeBuild 現在支援使用多個輸入來源並發佈多組成品的專案。如需詳細資訊，請參閱多個輸入來源和輸入成品範例以 CodeBuild 及CodePipeline與多個輸入來源和輸出成品範例整合。	2018 年 8 月 30 日
語意版本化範例	使用 CodeBuild 者指南現在有一個以使用案例為基礎的範例，示範如何在建置時使用語意版本控制來建立成品名稱。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
新的靜態網站樣本	使用 CodeBuild 者指南現在有一個以使用案例為基礎的範例，示範如何在 S3 儲存貯體中託管組建輸出。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
Support 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 用來命名組建成品的格式。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

Support 未加密的建置成品	CodeBuild 現在支援含有未加密組建成品的組建。如需詳細資訊，請參閱建立組建專案 (主控台)。	2018 年 7 月 26 日
Support 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 主控台來監視專案和帳戶層級的組建。如需詳細資訊，請參閱監控建置。	2018 年 7 月 19 日
Support 報告組建狀態	CodeBuild 現在可以向來源提供者報告組建的開始和完成狀態。如需詳細資訊，請參閱中的建立組建專案 CodeBuild。	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
Support 構建規格文件中的`finally`塊	CodeBuild 文檔已更新，其中包含 buildspec 文件中可選`finally`塊的詳細信息。finally 區塊中的指令一律會在其對應指令區塊中的指令之後執行。如需詳細資訊，請參閱 Buildspect 語法。	2018 年 6 月 20 日

CodeBuild 用戶端更新通知	CodeBuild 文件已更新，其中包含有關如何使用 Amazon SNS 的詳細資訊，以便在新版 CodeBuild 代理程式發佈時收到通知。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 視窗伺服器平台的組建，包括 Windows 上的 .NET 核心 2.0 的預先封裝的建置環境。如需詳細資訊，請參閱Microsoft 視窗樣本 CodeBuild。	2018 年 5 月 25 日
支援組建冪等	使用 AWS Command Line Interface (AWS CLI) 執行 `start-build` 命令時，您可以指定建置為冪等。如需詳細資訊，請參閱執行建置 (AWS CLI)。	2018 年 5 月 15 日
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，請參閱在 AWS CodeBuild 中執行建置。	2018 年 5 月 15 日

變更	描述	日期
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。	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 SDK 告訴 CodeBuild 使用四個聯邦資訊處理標準 (FIPS) 端點之一。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 現已在亞太區域 (孟買)、歐洲 (巴黎) 和南美洲 (聖保羅) 區域推出。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 企業伺服器支援	CodeBuild 現在可以從存儲在 GitHub 企業服務器存儲庫中的源代碼進行構建。如需詳細資訊，請參閱GitHub 企業伺服器範例。	2018 年 1 月 25 日
Git 複製深度支援	CodeBuild 現在支援建立淺層複製，並將歷史記錄截斷為指定數量的提交。如需詳細資訊，請參閱建立組建專案。	2018 年 1 月 25 日

變更	描述	日期
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 現在支持依賴緩存。這允許 CodeBuild 將某些可重複使用的構建環境保存在緩存中，並在構建中使用它。	2017 年 11 月 27 日
組建識別證支援	CodeBuild 現在支持使用構建徽章，它提供了可嵌入的動態生成的圖像（徽章），該圖像（徽章）顯示項目的最新版本的狀態。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 現在支持 CodeBuild 作為 AWS 資源，這意味著該服務可以跟踪您的 CodeBuild 項目。如需有關的更多資訊 AWS Config，請參閱AWS Config 樣本。	2017 年 10 月 20 日
在 GitHub儲存庫中自動重建更新的原始碼	如果您的原始程式碼儲存在儲存 GitHub 庫中，您可 AWS CodeBuild 以在將程式碼變更推送至儲存庫時重建原始程式碼。如需詳細資訊，請參閱GitHub 提取請求和網絡掛鉤過濾器樣本。	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	您現在可以使用主 AWS CodeBuild 控台或擷 AWS CLI 取存放在 Amazon EC2 Systems Manager 參數存放區中的敏感或大型環境變數。您現在也可以使用 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 Buildspec t 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。如需詳細資訊建立組建專案，請參閱變更建置專案的設定、和執行建置。	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
更新了使用構建規格檢索存儲在 Amazon EC2 Systems Manager 參數存放區中的敏感或大型環境變量的方法	AWS CodeBuild 現在可以更輕鬆地使用組建規格擷取存放在 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，請參閱中的`parameter-store`對應Buildspec 語法。	2017 年 8 月 10 日

變更	描述	日期
AWS CodeBuild 支持比特桶	CodeBuild 現在可以從存儲在 Bitbucket 存儲庫中的源代碼進行構建。如需詳細資訊，請參閱建立組建專案和執行建置。	2017 年 8 月 10 日
AWS CodeBuild 適用於美國西部 (加利佛尼亞北部)、歐洲 (倫敦) 和加拿大 (中部)	AWS CodeBuild 目前已在美國西部 (加利佛尼亞北部)、歐洲 (倫敦) 和加拿大 (中部) 區域推出。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
更新的組建通知範例	CodeBuild 現在透過 Amazon CloudWatch 活動和亞馬遜簡單通知服務 (Amazon SNS) 為建置通知提供內建支援。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	已添加示例，顯示如何使用以 CodeBuild 及自定義 Docker 構建映像以構建和運行 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

變更	描述	日期
擷取提取要求的原始程 GitHub式碼	當您執行依賴儲存在存 GitHub放庫中 CodeBuild 之原始程式碼的組建時，您現在可以指定要建置的提 GitHub 取要求識別碼。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)或中的`sourceVersion` 值執行建置 (AWS CLI)。	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 版解決了在默認 shell 的單獨實例中 CodeBuild 運行每個構建命令的問題。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，請參閱建立的規格參考 CodeBuild。	2017 年 5 月 9 日
碼頭文件可用於構建圖像 GitHub	中提供的許多組建映像檔的定義 AWS CodeBuild 都以 Docker 檔案的形式提供。GitHub如需詳細資訊，請參閱中表格的「定義」欄碼頭圖片提供 CodeBuild。	2017 年 5 月 2 日

變更	描述	日期
AWS CodeBuild 適用於歐洲 (法蘭克福)、亞太區域 (新加坡)、亞太區域 (雪梨) 和亞太區域 (東京)	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) 等地區推出。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
CodePipeline 測試動作支援 CodeBuild	您現在可以在使用的測試動 CodePipeline 作中新增至管線 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的頂層目錄，您可以指示 CodeBuild 其內容包含在建置輸出成品中。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
內建的環境變數	AWS CodeBuild 提供額外的內建環境變數供您的組建使用。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

變更	描述	日期
AWS CodeBuild 適用於美國東部 (俄亥俄州)	AWS CodeBuild 現已在美國東部 (俄亥俄) 區域推出。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 運行您在構建環境的默認 shell 的單獨實例中指定的每個命令。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 提供數個環境變數，您可以在建置命令中使用這些變數。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，請參閱疑難排 AWS CodeBuild。	2016 年 12 月 5 日
Jenkins 外掛程式初始版本	這是 CodeBuild 詹金斯插件的初始版本。如需詳細資訊，請參閱使用 AWS CodeBuild 搭配 Jenkins。	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

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

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。