사용자 가이드

AWS CodeBuild

API 버전 2016-10-06

AWS CodeBuild: 사용자 가이드

Table of Contents

AWS CodeBuild이란?

AWS CodeBuild클라우드의 완전 관리형 빌드 서비스입니다. CodeBuild 소스 코드를 컴파일하고, 단위 테스트를 실행하고, 배포할 준비가 된 아티팩트를 생성합니다. CodeBuild 자체 빌드 서버를 프로비저닝, 관리, 확장할 필요가 없습니다. 이 서비스는 Apache Maven, Gradle 등과 같은 널리 사용되는 프로그래밍 언어 및 빌드 도구에 맞게 사전 패키지된 빌드 환경을 제공합니다. 자체 빌드 도구를 CodeBuild 사용하도록 빌드 환경을 사용자 지정할 수도 있습니다. CodeBuild 최대 빌드 요청에 맞춰 자동으로 확장됩니다.

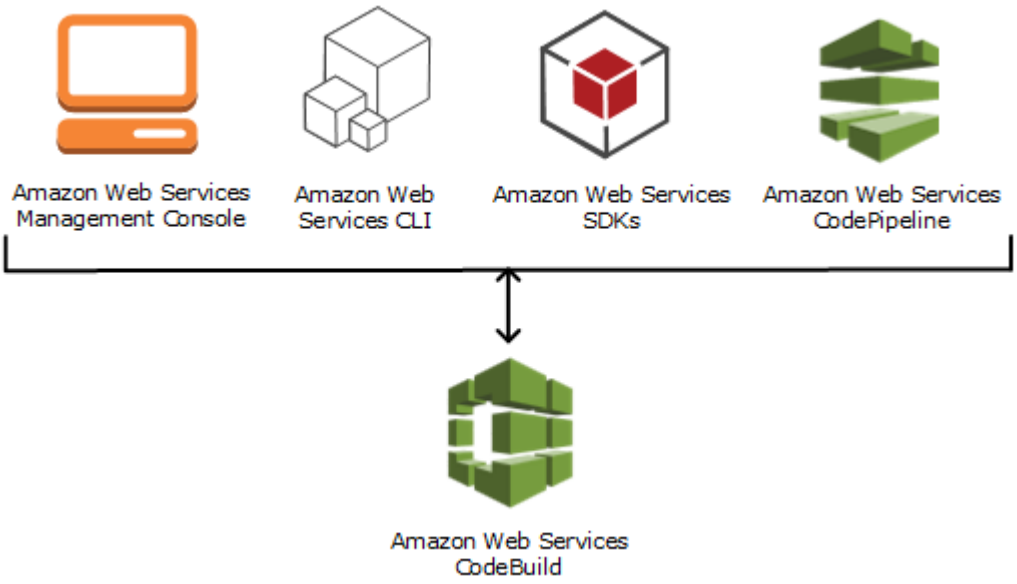
CodeBuild 다음과 같은 이점을 제공합니다.

- 완전 관리형 — CodeBuild 자체 빌드 서버를 설정, 패치, 업데이트, 관리할 필요가 없습니다.
- 온디맨드 — 필요에 따라 빌드 요구 사항에 맞게 CodeBuild 확장할 수 있습니다. 사용한 빌드 시간만큼만 요금을 지불합니다.
- 기본 CodeBuild 제공 — 가장 많이 사용되는 프로그래밍 언어를 위한 사전 구성된 빌드 환경을 제공합니다. 빌드 스크립트를 선택하여 시작하기만 하면 됩니다.

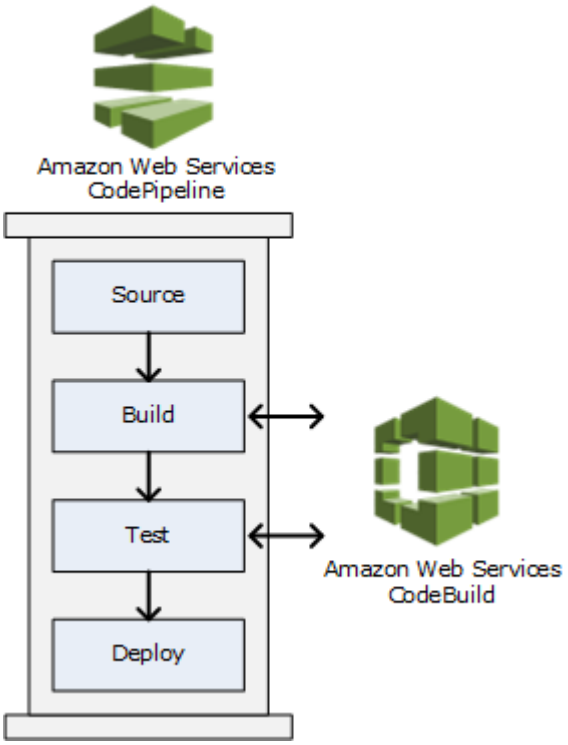
자세한 설명은 AWS CodeBuild 섹션을 참조하세요.

실행 방법 CodeBuild

AWS CodeBuild 또는 AWS CodePipeline 콘솔을 사용하여 CodeBuild를 실행할 수 있습니다. AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 CodeBuild 사용하여 실행을 자동화할 수도 있습니다.



다음 다이어그램에서 볼 수 있듯이 에서 AWS CodePipeline 파이프라인의 빌드 또는 테스트 단계에 빌드 또는 테스트 작업으로 추가할 CodeBuild 수 있습니다. AWS CodePipeline코드를 릴리스하는 데 필요한 단계를 모델링, 시각화 및 자동화하는 데 사용할 수 있는 지속적 전달 서비스입니다. 여기에는 코드 빌드도 포함됩니다. 파이프라인은 코드 변경 사항이 릴리스 프로세스를 통과하는 방식을 설명하는 워크플로우 구성입니다.



파이프라인을 만든 다음 CodeBuild 빌드 또는 테스트 작업을 추가하는 데 사용하려면 CodePipeline을 참조하십시오 CodePipeline 함께 사용 CodeBuild. 에 대한 CodePipeline 자세한 내용은 AWS CodePipeline사용 설명서를 참조하십시오.

또한 CodeBuild 콘솔에서는 리포지토리, 빌드 프로젝트, 배포 애플리케이션, 파이프라인과 같은 리소스를 빠르게 검색할 수 있는 방법을 제공합니다. 리소스로 이동을 선택하거나 / 키를 누른 후 리소스 이름을 입력합니다. 목록에 일치 항목이 나타납니다. 검색은 대/소문자를 구분하지 않습니다. 보기 권한이 있는 리소스만 표시됩니다. 자세한 설명은 콘솔에서 리소스 보기 섹션을 참조하세요.

CodeBuild 요금

자세한 내용은 가격을 참조하십시오. CodeBuild

시작하려면 어떻게 해야 하나요 CodeBuild?

다음 단계를 수행하는 것이 좋습니다.

1. 에 CodeBuild 있는 정보를 읽고 자세히 알아보십시오개념.
2. CodeBuild 의 지침에 따라 예제 시나리오를 실험해 보십시오콘솔을 사용하여 시작하기.
3. CodeBuild 의 지침을 따라 자신의 시나리오에 맞게 사용하세요빌드 계획.

AWS CodeBuild 개념

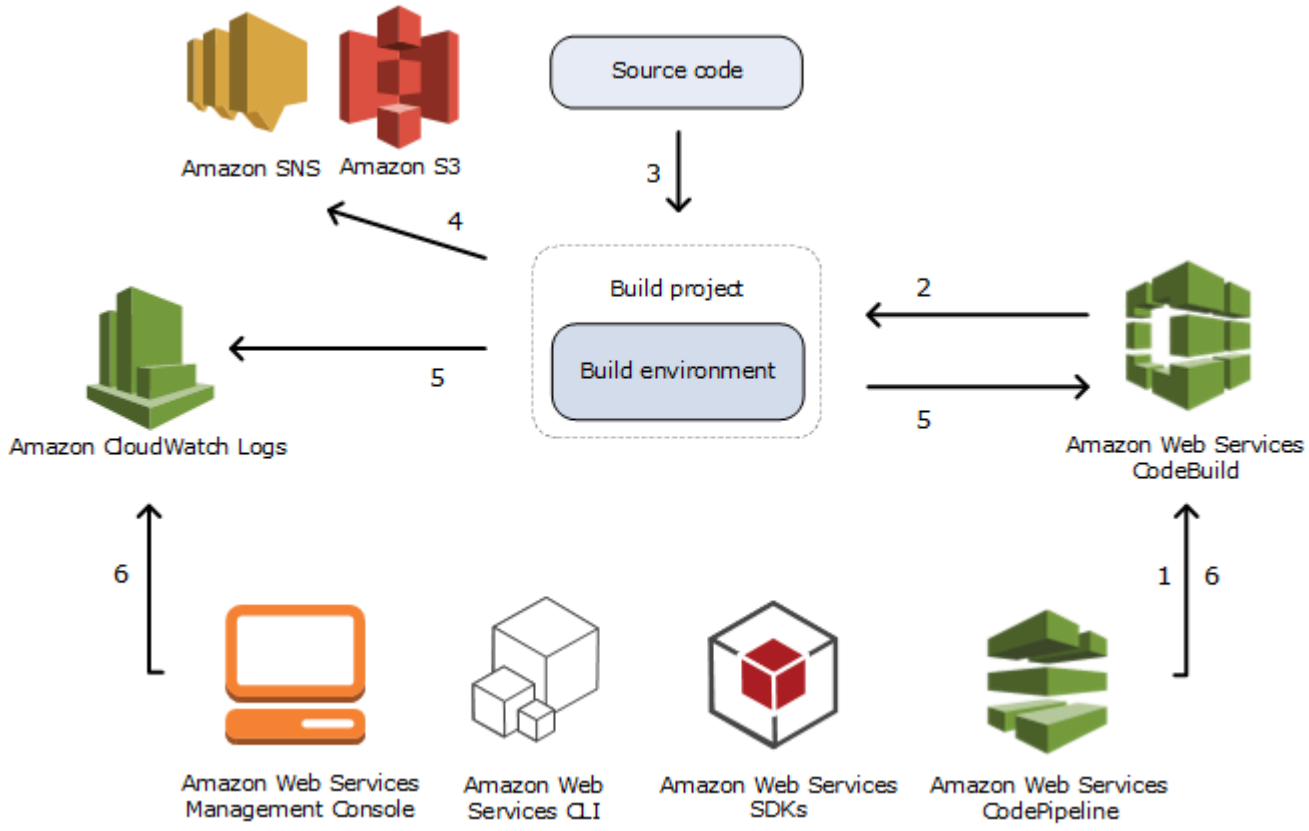
다음은 CodeBuild 작동 방식을 이해하는 데 필요한 중요한 개념입니다.

주제

- CodeBuild 작동 방식
- 다음 단계

CodeBuild 작동 방식

다음 다이어그램은 CodeBuild를 사용하여 빌드를 실행할 때 나타나는 현상을 보여 줍니다.



1. 입력으로 빌드 프로젝트와 함께 CodeBuild를 제공해야 합니다. 이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다. 빌드 환경은 CodeBuild가 빌드를 실행하는 데 사용하는 운영 체제, 프로그래밍 언어 런타임 및 도구의 조합을 나타냅니다. 자세한 내용은 다음을 참조하세요.

   - 빌드 프로젝트 생성
   - 빌드 환경 참조

2. CodeBuild가 빌드 프로젝트를 사용하여 빌드 환경을 생성합니다.
3. CodeBuild가 빌드 환경에 소스 코드를 다운로드한 다음, 빌드 프로젝트에 정의되거나 소스 코드에 직접 포함된 빌드 사양(buildspec)을 사용합니다. buildspec은 CodeBuild가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다. 자세한 내용은 buildspec 참조 부분을 참조하세요.
4. 빌드 출력이 있으면 빌드 환경에서 출력을 S3 버킷에 업로드합니다. 빌드 환경에서 사용자가 buildspec에 지정한 작업을 수행할 수도 있습니다(예: Amazon SNS 주제에 빌드 알림 전송). 예시는 빌드 알림 샘플에서 확인하세요.

5. 빌드가 실행되는 동안 빌드 환경이 정보를 CodeBuild 및 Amazon CloudWatch Logs에 전송합니다.
6. 빌드가 실행되는 동안 AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서는 요약된 빌드 정보를 가져오고 Amazon CloudWatch Logs에서는 자세한 빌드 정보를 가져올 수 있습니다. AWS CodePipeline을 사용하여 빌드를 실행하는 경우에는 CodePipeline에서 제한된 빌드 정보를 가져올 수 있습니다.

다음 단계

이제 AWS CodeBuild에 대해 자세히 알고 있으므로 다음 단계를 권장합니다.

1. 콘솔을 사용하여 시작하기의 지침에 따라 예제 시나리오에서 CodeBuild를 실험해 보세요.
2. 빌드 계획의 지침에 따라 자체 시나리오에서 CodeBuild를 사용하세요.

CodeBuild 시작하기

다음 자습서에서는 AWS CodeBuild를 사용하여 샘플 소스 코드 입력 파일 집합을 배포 가능한 소스 코드 버전으로 빌드합니다.

두 자습서의 입력과 결과는 동일하지만 한 자습서는 AWS CodeBuild 콘솔을 사용하고 다른 자습서는 AWS CLI를 사용합니다.

> **Important**
>
> AWS 루트 계정을 사용하여 이 자습서를 완료하지 않는 것이 좋습니다.

콘솔을 사용하여 AWS CodeBuild 시작하기

이 자습서에서는 AWS CodeBuild를 사용하여 샘플 소스 코드 입력 파일 모음(빌드 입력 결과물 또는 빌드 입력)을 소스 코드의 배포 가능한 버전(빌드 출력 결과물 또는 빌드 출력)으로 빌드합니다. 특히, 일반적인 빌드 도구인 Apache Maven을 사용하여 Java 클래스 파일 세트를 JAR 아카이브 (JAR) 파일로 빌드하도록 CodeBuild 지시합니다. 이 자습서는 Apache Maven이나 Java에 익숙하지 않아도 완료할 수 있습니다.

CodeBuild 콘솔AWS CodePipeline, A 또는 CodeBuild SDK를 통해 작업할 수 있습니다. AWS CLI AWS 이 가이드에서는 콘솔 사용 방법을 보여줍니다. CodeBuild CodePipeline 사용에 대한 자세한 내용은 CodePipeline 함께 사용 CodeBuild을 참조하십시오.

> **Important**
>
> 이 자습서의 단계에서는 AWS 계정에 요금이 부과될 수 있는 리소스(예: S3 버킷)를 생성해야 합니다. 여기에는 Amazon S3AWS KMS, CloudWatch 로그와 관련된 AWS 리소스 및 작업에 대한 가능한 요금이 포함됩니다. CodeBuild 자세한 내용은 AWS CodeBuild요금, Amazon S3 요금, AWS Key Management Service요금 및 Amazon CloudWatch 요금을 참조하십시오.

단계

- 1단계: 소스 코드 생성
- 2단계: buildspec 파일 생성
- 3단계: 두 개의 S3 버킷 생성

1단계: 소스 코드 생성

(콘솔을 사용하여 AWS CodeBuild 시작하기의 일부)

이 단계에서는 출력 CodeBuild 버킷에 빌드하려는 소스 코드를 생성합니다. 이 소스 코드는 두 개의 Java 클래스 파일 및 Apache Maven Project Object Model(POM) 파일로 구성됩니다.

1. 로컬 컴퓨터나 인스턴스의 빈 디렉터리에 다음 디렉터리 구조를 생성합니다.

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

2. 원하는 텍스트 편집기를 사용하여 다음 파일을 생성하고 이름을 `MessageUtil.java`로 지정한 다음 이를 `src/main/java` 디렉터리에 저장합니다.

```
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;
  }
}
```

이 클래스 파일은 자신에게 전달되는 문자열을 출력으로 생성합니다. `MessageUtil` 생성자는 문자열을 설정합니다. `printMessage` 메서드는 출력을 생성합니다. `salutationMessage` 메서드는 `Hi!` 다음에 문자열을 출력합니다.

3. 다음 파일을 생성하고 이름을 `TestMessageUtil.java`로 지정한 다음 이를 `/src/test/java` 디렉터리에 저장합니다.

```
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());
  }
}
```

이 클래스 파일은 `MessageUtil` 클래스의 `message` 변수를 `Robert`로 설정합니다. 그런 다음 테스트를 수행하여 `Robert` 및 `Hi!Robert` 문자열이 출력에 나타나는지 여부를 확인하여 `message` 변수가 성공적으로 설정되었는지를 확인합니다.

4. 다음 파일을 생성하고 이름을 `pom.xml`로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
<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 Maven은 이 파일의 지침을 따라 `MessageUtil.java` 및 `TestMessageUtil.java` 파일을 `messageUtil-1.0.jar`이라는 파일로 변환한 다음 지정된 테스트를 실행합니다.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

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

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

다음 단계

[2단계: buildspec 파일 생성]

2단계: buildspec 파일 생성

(이전 단계: 1단계: 소스 코드 생성)

이 단계에서는 빌드 사양 파일을 생성합니다. 빌드스펙은 빌드를 실행하는 데 사용되는 빌드 명령 및 관련 설정 (YAML 형식) 의 모음입니다. CodeBuild 빌드 사양이 없으면 빌드 입력을 빌드 출력으로 성공적으로 변환하거나 빌드 환경에서 빌드 출력 아티팩트를 찾아 출력 버킷에 업로드할 수 CodeBuild 없습니다.

다음 파일을 생성하고 이름을 `buildspec.yml`로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
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**
>
> 빌드 사양 선언은 올바른 YAML이어야 하므로 빌드 사양 선언의 공백이 중요합니다. 빌드 사양 선언의 공백 수가 YAML과 맞지 않으면 빌드가 즉시 실패할 수 있습니다. YAML 검사기를 사용하여 빌드 사양 선언이 올바른 YAML인지 여부를 테스트할 수 있습니다.

> **Note**
>
> 소스 코드에 빌드 사양 파일을 포함하는 대신, 빌드 프로젝트를 생성할 때 빌드 명령을 별도로 선언할 수 있습니다. 이 방법은 매번 소스 코드 리포지토리를 업데이트하지 않아도 되도록 여러 개의 빌드 명령이 있는 소스 코드를 빌드하려는 경우 유용합니다. 자세한 설명은 buildspec 구문 섹션을 참조하세요.

이 빌드 사양 선언에서:

- `version`은 사용 중인 빌드 사양 표준의 버전을 나타냅니다. 이 빌드 사양 선언은 최신 버전인 `0.2`을 사용합니다.
- `phases`는 CodeBuild 가 명령을 실행하도록 지시할 수 있는 빌드 단계를 나타냅니다. 여기에서는 이 빌드 단계가 `install`, `pre_build`, `build` 및 `post_build`로 나열되어 있습니다. 이 빌드 단계 이름의 철자는 변경할 수 없으며 추가로 빌드 단계 이름을 생성할 수도 없습니다.

  이 예시에서는 `build` 단계 중에 명령을 CodeBuild 실행합니다. `mvn install` 이 명령은 Apache Maven이 Java 클래스 파일을 컴파일 및 테스트하고 컴파일된 Java 클래스 파일을 빌드 출력 결과물에 패키지하도록 지시합니다. 그리고 몇 가지 `echo` 명령을 각 빌드 단계에 추가하여 이 연습을 마치게 됩니다. 이 자습서의 뒷부분에서 자세한 빌드 정보를 확인할 때 이러한 `echo` 명령의 출력을 확인하면 CodeBuild가 명령을 실행하는 방법 및 명령 실행 순서를 이해하는 데 많은 도움이 됩니다. (이 예에는 모든 빌드 단계가 포함되어 있지만, 해당 단계에서 아무 명령도 실행하지 않으려면 빌드 단계를 포함하지 않아도 됩니다.) 각 빌드 단계에서 지정된 각 명령을 나열된 순서대로 한 번에 하나씩 처음부터 끝까지 CodeBuild 실행합니다.
- `artifacts`출력 버킷에 CodeBuild 업로드되는 빌드 출력 아티팩트 세트를 나타냅니다. `files`빌드 출력에 포함할 파일을 나타냅니다. CodeBuild 빌드 환경의 `target` 상대 디렉터리에 있는 단일 `messageUtil-1.0.jar` 파일을 업로드합니다. 파일 이름 `messageUtil-1.0.jar` 및 디렉터리 이름 `target`은 Apache Maven이 이 예제에서만 빌드 출력 결과물을 생성 및 저장하는 방식에 따라 달라집니다. 사용자 자체 빌드에서는 이러한 파일 이름과 디렉터리가 다릅니다.

자세한 내용은 buildspec 참조 섹션을 참조하십시오.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

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

다음 단계

3단계: 두 개의 S3 버킷 생성

3단계: 두 개의 S3 버킷 생성

(이전 단계: 2단계: buildspec 파일 생성)

한 개의 버킷을 사용하여 이 실습을 수행할 수도 있지만, 두 개의 버킷을 사용해야 빌드 입력이 들어오는 위치 및 빌드 출력이 나가는 위치를 쉽게 확인할 수 있습니다.

- 이러한 버킷 중 하나(입력 버킷)는 빌드 입력을 저장합니다. 이 자습서에서 이 입력 버킷의 이름은 `codebuild-region-ID-account-ID-input-bucket`입니다. 여기서 *region-ID*는 버킷의 AWS 리전이고 *account-ID*는 AWS 계정 ID입니다.
- 다른 버킷(출력 버킷)은 빌드 출력을 저장합니다. 이 자습서에서 이 출력 버킷의 이름은 `codebuild-region-ID-account-ID-output-bucket`입니다.

이러한 버킷에 대해 다른 이름을 선택한 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

이 두 버킷은 빌드와 같은 AWS 리전에 있어야 합니다. 예를 들어 미국 동부 (오하이오) 지역에서 빌드를 CodeBuild 실행하도록 지시하는 경우 이러한 버킷은 미국 동부 (오하이오) 지역에도 있어야 합니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 버킷 생성을 참조하세요.

> **Note**
>
> CodeCommit GitHub, 및 Bitbucket 리포지토리에 저장된 빌드 CodeBuild 입력도 지원하지만 이 자습서에서는 사용 방법을 설명하지 않습니다. 자세한 설명은 빌드 계획 섹션을 참조하세요.

다음 단계

4단계: 소스 코드 및 buildspec 파일 업로드

4단계: 소스 코드 및 buildspec 파일 업로드

(이전 단계: 3단계: 두 개의 S3 버킷 생성)

이 단계에서는 소스 코드 및 빌드 사양 파일을 입력 버킷에 추가합니다.

사용하는 운영 체제의 zip 유틸리티를 사용하여 `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml` 및 `buildspec.yml`을 포함하는 `MessageUtil.zip`이라는 파일을 생성합니다.

`MessageUtil.zip` 파일의 디렉터리 구조가 다음과 같이 나타나야 합니다.

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

> **Important**
>
> *(root directory name)* 디렉터리는 포함하지 말고, *(root directory name)* 디렉터리 안에 있는 디렉터리 및 파일만 포함하십시오.

`MessageUtil.zip` 파일을 `codebuild-`*region-ID*`-`*account-ID*`-input-bucket`이라는 입력 버킷에 업로드합니다.

> **Important**
>
> 및 Bitbucket 리포지토리의 경우 일반적으로 각 리포지토리의 루트 (최상위 수준) `buildspec.yml` 에 이름이 지정된 빌드 사양 파일을 저장하거나 빌드 프로젝트 정의의 일부로 빌드 사양 선언을 포함해야 합니다. CodeCommit GitHub 리포지토리의 소스 코드 및 빌드 사양 파일을 포함하는 ZIP 파일은 생성하지 마십시오.
> S3 버킷에만 저장된 빌드 입력의 경우, 일반적으로 루트(최상위)에 소스 코드 및 `buildspec.yml`이라는 빌드 사양 파일을 포함하는 ZIP 파일을 생성하거나 빌드 사양 선언을 빌드 프로젝트 정의의 일부로 포함해야 합니다.
> 빌드 사양 파일에 다른 이름을 사용하거나 루트가 아닌 위치에서 빌드 사양을 참조하려면 빌드 사양 재정의를 빌드 프로젝트 정의의 일부로 지정할 수 있습니다. 자세한 설명은 buildspec 파일 이름 및 스토리지 위치 섹션을 참조하세요.

다음 단계

5단계: 빌드 프로젝트 생성

5단계: 빌드 프로젝트 생성

(이전 단계: 4단계: 소스 코드 및 buildspec 파일 업로드)

이 단계에서는 AWS CodeBuild가 빌드를 실행하는 데 사용할 빌드 프로젝트를 생성합니다. 이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다. 빌드 환경은 운영 체제, 프로그래밍 언어 런타임 및 빌드를 실행하는 데 CodeBuild 사용되는 도구의 조합을 나타냅니다. 빌드 환경은 도커 이미지로 표현됩니다. 자세한 정보는 Docker Docs 웹 사이트에서 Docker 개요 주제를 참조하십시오.

이 빌드 환경에서는 JDK (자바 개발 키트) 버전과 Apache Maven이 포함된 Docker 이미지를 사용하도록 CodeBuild 지시합니다.

빌드 프로젝트를 생성하려면

1. AWS Management Console로그인하고 https://console.aws.amazon.com/codesuite/codebuild/home 에서 콘솔을 엽니다. AWS CodeBuild
2. AWS지역 선택기를 사용하여 지원되는 AWS 지역을 CodeBuild 선택합니다. 자세한 설명은 Amazon Web Services 일반 참조의 AWS CodeBuild 엔드포인트 및 할당량을 참조하세요.
3. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.

4. 빌드 프로젝트 만들기 페이지의 프로젝트 구성에서 프로젝트 이름에 이 빌드 프로젝트의 이름(이 예에서는 `codebuild-demo-project`)을 입력합니다. 각 AWS 계정에서 빌드 프로젝트 이름은 고유해야 합니다. 다른 이름을 사용하는 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

   > **Note**
   >
   > 빌드 프로젝트 생성 페이지에 You are not authorized to perform this operation.와 같은 오류 메시지가 표시될 수 있습니다. 빌드 프로젝트를 생성할 권한이 없는 사용자로 AWS Management Console에 로그인했기 때문일 가능성이 큽니다. 이 문제를 해결하려면 AWS Management Console에서 로그아웃한 후, 다음 IAM 엔터티 중 하나에 속한 보안 인증을 사용하여 다시 로그인합니다.
   >
   > - AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   > - 해당 사용자 또는 사용자가 속한 IAM 그룹에 연결된 `AWSCodeBuildAdminAccess`, `AmazonS3ReadOnlyAccess` 및 `IAMFullAccess` 관리형 정책을 사용하는 AWS 계정의 사용자. AWS 계정에 이러한 권한이 있는 사용자 또는 그룹이 없으며, 이러한 권한을 사용자나 그룹에 추가할 수 없는 경우, AWS 계정 관리자에게 지원을 요청하세요. 자세한 설명은 AWS 에 대한 관리형 (사전 정의된) 정책 AWS CodeBuild 섹션을 참조하세요.
   >
   > 두 옵션 모두 이 자습서를 완료할 수 있도록 빌드 프로젝트를 생성할 수 있는 관리자 권한이 포함되어 있습니다. 작업을 완료하는 데 필요한 최소 권한을 항상 사용하는 것이 좋습니다. 자세한 설명은 AWS CodeBuild 권한 참조 섹션을 참조하세요.

5. 소스의 소스 공급자에서 Amazon S3를 선택합니다.
6. 버킷에서 codebuild-***region-ID*-*account-ID***-input-bucket을 선택합니다.
7. S3 object key(S3 객체 키)에 **`MessageUtil.zip`**을 입력합니다.
8. 환경의 환경 이미지에서 관리형 이미지를 선택된 상태로 둡니다.
9. 운영 체제에서 Amazon Linux 2를 선택합니다.
10. 런타임에서 표준을 선택합니다.
11. 이미지의 경우 aws/codebuild/amazonlinux2-x86_64-standard:4.0을 선택합니다.
12. 서비스 역할에서 새 서비스 역할을 선택된 상태로 두고, 역할 이름도 변경하지 않고 그대로 둡니다.
13. Buildspec(빌드 사양)에서 Use a buildspec file(빌드 사양 파일 사용)을 선택된 상태로 둡니다.
14. 아티팩트에서 유형으로 Amazon S3를 선택합니다.

15. 버킷 이름에서 codebuild-***region-ID***-***account-ID***-output-bucket을 선택합니다.
16. 이름 및 경로는 비워 둡니다.
17. 빌드 프로젝트 생성을 선택합니다.

다음 단계

6단계: 빌드 실행

6단계: 빌드 실행

(이전 단계: 5단계: 빌드 프로젝트 생성)

이 단계에서는 빌드 프로젝트의 설정으로 빌드를 실행하도록 AWS CodeBuild에 지시를 내립니다.

빌드를 실행하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 빌드 프로젝트 목록에서 을 선택한 codebuild-demo-project다음 빌드 시작을 선택합니다. 빌드가 즉시 시작됩니다.

다음 단계

7단계: 요약된 빌드 정보 보기

7단계: 요약된 빌드 정보 보기

(이전 단계: 6단계: 빌드 실행)

이 단계에서는 빌드 상태에 대한 요약 정보를 확인합니다.

요약된 빌드 정보를 보려면

1. codebuild-demo-project: <build-ID>페이지가 표시되지 않는 경우 내비게이션 바에서 빌드 기록을 선택합니다. 그런 다음 빌드 프로젝트 목록에서 프로젝트에 대해 빌드 실행 대상 링크를 선택합니다 codebuild-demo-project. 일치하는 링크가 하나만 있어야 합니다. (이전에 이 자습서를 완료한 경우 완료됨 열에서 가장 최근 값이 포함된 링크를 선택합니다.)
2. 빌드 상태 페이지의 단계 세부 정보에는 다음과 같은 빌드 단계가 표시되고 상태 열에 성공이 표시되어야 합니다.

- SUBMITTED
- 대기됨
- PROVISIONING
- DOWNLOAD_SOURCE
- INSTALL
- PRE_BUILD
- BUILD
- POST_BUILD
- UPLOAD_ARTIFACTS
- FINALIZING
- COMPLETED

빌드 상태에, 성공이 표시되어야 합니다.

대신 진행 중이 표시되면 새로 고침 단추를 선택합니다.

3. 각 빌드 단계 옆에 있는 기간 값은 빌드 단계가 지속되는 기간을 나타냅니다. 종료 시간 값은 빌드 단계가 종료되었음을 나타냅니다.

다음 단계

8단계: 자세한 빌드 정보 보기

8단계: 자세한 빌드 정보 보기

(이전 단계: 7단계: 요약된 빌드 정보 보기)

이 단계에서는 CloudWatch 로그에서 빌드에 대한 자세한 정보를 볼 수 있습니다.

> **Note**
>
> 민감한 정보를 보호하기 위해 CodeBuild 로그에는 다음이 숨겨져 있습니다.
>
> - AWS액세스 키 ID: 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.

- 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
- AWS Secrets Manager을 사용하여 지정한 문자열입니다. 자세한 설명은 키 관리 섹션을 참조하세요.

자세한 빌드 정보를 보려면

1. 이전 단계의 빌드 세부 정보 페이지가 계속 표시된 상태에서 [Build logs]에 빌드 로그의 마지막 10,000행이 표시되어 있습니다. 로그에서 전체 빌드 CloudWatch 로그를 보려면 전체 로그 보기 링크를 선택합니다.
2. 로그 CloudWatch 로그 스트림에서 로그 이벤트를 찾아볼 수 있습니다. 기본적으로 가장 최근의 로그 이벤트 세트만 표시됩니다. 이전의 로그 이벤트를 보려면 목록의 처음으로 스크롤합니다.
3. 이 자습서에서는 대부분의 로그 이벤트에 CodeBuild 가 빌드 종속성 파일을 빌드 환경에 다운로드 및 설치하는 작업에 대한 자세한 정보가 들어 있는데, 대부분의 사용자에게 필요하지 않은 정보입니다. [Filter events]를 사용하면 표시되는 정보를 줄일 수 있습니다. 예를 들어, 필터 이벤트에 `"[INFO]"`를 입력하면 `[INFO]`를 포함하는 이벤트만 표시됩니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 필터 및 패턴 구문을 참조하십시오.

다음 단계

9단계: 빌드 출력 아티팩트 가져오기

9단계: 빌드 출력 아티팩트 가져오기

(이전 단계: 8단계: 자세한 빌드 정보 보기)

이 단계에서는 출력 버킷에 CodeBuild 빌드되고 업로드된 `messageUtil-1.0.jar` 파일을 가져옵니다.

CodeBuild 콘솔 또는 Amazon S3 콘솔을 사용하여 이 단계를 완료할 수 있습니다.

빌드 출력 결과물을 가져오려면(AWS CodeBuild 콘솔)

1. CodeBuild 콘솔이 여전히 열려 있고 이전 단계의 빌드 세부 정보 페이지가 계속 표시된 상태에서 빌드 세부 정보 탭을 선택하고 Artifacts 섹션으로 스크롤합니다.

> **Note**
>
> 빌드 세부 정보 페이지가 표시되지 않으면, 탐색 모음에서 빌드 이력을 선택한 다음, 빌드 실행 링크를 선택합니다.

2. Amazon S3 폴더로 연결되는 링크는 아티팩트 업로드 위치 아래에 있습니다. 이 링크를 클릭하면 `messageUtil-1.0.jar` 빌드 출력 아티팩트 파일을 찾는 Amazon S3의 폴더가 열립니다.

빌드 출력 아티팩트를 가져오려면(Amazon S3 콘솔)

1. https://console.aws.amazon.com/s3/ 에서 Amazon S3 콘솔을 엽니다.
2. `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`를 엽니다.
3. `codebuild-demo-project` 폴더를 엽니다.
4. `target` 폴더를 엽니다. 이 폴더에서 `messageUtil-1.0.jar` 빌드 출력 결과물 파일을 찾을 수 있습니다.

다음 단계

10단계: S3 버킷 삭제

10단계: S3 버킷 삭제

(이전 단계: 9단계: 빌드 출력 아티팩트 가져오기)

AWS 계정에 계속하여 요금이 부과되지 않도록 이 자습서에서 사용한 입력 및 출력 버킷을 삭제할 수 있습니다. 지침을 보려면 Amazon Simple Storage Service 사용 설명서에서 버킷 삭제 또는 비우기를 참조하세요.

IAM 사용자 또는 관리자 IAM 사용자를 통해 이러한 버킷을 삭제하는 경우 사용자에게 추가 액세스 권한이 있어야 합니다. 사용자의 기존 액세스 정책에 마커 사이의 다음 명령문(*`### BEGIN ADDING STATEMENT HERE ###`* 및 *`### END ADDING STATEMENTS HERE ###`*)을 추가합니다.

이 명령문의 줄임표(...)는 간결하게 나타내기 위해 사용됩니다. 기존 액세스 정책의 어떤 명령문도 제거하지 마십시오. 이러한 줄임표는 정책에 입력하지 않아야 합니다.

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

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

다음 단계

마무리

마무리

이 자습서에서는 AWS CodeBuild를 사용하여 Java 클래스 파일 세트를 JAR 파일에 빌드했습니다. 그리고 빌드 결과를 확인했습니다.

이제 자체 CodeBuild 시나리오에서 사용해 볼 수 있습니다. 빌드 계획의 지침을 따르세요. 아직 시도해 볼 준비가 되지 않은 것 같으면 샘플 몇 가지를 더 빌드해 볼 수 있습니다. 자세한 내용은 샘플을(를) 참조하세요.

AWS CLI를 사용하여 AWS CodeBuild 시작하기

이 자습서에서는 AWS CodeBuild를 사용하여 샘플 소스 코드 입력 파일 모음(빌드 입력 결과물 또는 빌드 입력이라고 함)을 소스 코드의 배포 가능한 버전(빌드 출력 결과물 또는 빌드 출력이라고 함)으로 빌드합니다. 특히, 일반적인 빌드 도구인 Apache Maven을 사용하여 Java 클래스 파일 세트를 JAR 아카이브 (JAR) 파일로 빌드하도록 CodeBuild 지시합니다. 이 자습서는 Apache Maven이나 Java에 익숙하지 않아도 완료할 수 있습니다.

CodeBuild 콘솔AWS CodePipeline, A 또는 CodeBuild SDK를 통해 작업할 수 있습니다. AWS CLI AWS 이 자습서에서는 CodeBuild 와 함께 사용하는 방법을 보여줍니다. AWS CLI 사용에 대한 자세한 내용은 CodePipeline 을 참조하십시오 CodePipeline 함께 사용 CodeBuild.

> **Important**
>
> 이 자습서의 단계에서는 AWS 계정에 요금이 부과될 수 있는 리소스(예: S3 버킷)를 생성해야 합니다. 여기에는 Amazon S3AWS KMS, CloudWatch 로그와 관련된 AWS 리소스 및 작업에 대한 가능한 요금이 포함됩니다. CodeBuild 자세한 내용은 CodeBuild요금, Amazon S3 요금, AWS Key Management Service요금 및 Amazon CloudWatch 요금을 참조하십시오.

단계

1단계: 소스 코드 생성

(AWS CLI를 사용하여 AWS CodeBuild 시작하기의 일부)

이 단계에서는 출력 CodeBuild 버킷에 빌드하려는 소스 코드를 생성합니다. 이 소스 코드는 두 개의 Java 클래스 파일 및 Apache Maven Project Object Model(POM) 파일로 구성됩니다.

1. 로컬 컴퓨터나 인스턴스의 빈 디렉터리에 다음 디렉터리 구조를 생성합니다.

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

```
                `-- java
```

2. 원하는 텍스트 편집기를 사용하여 다음 파일을 생성하고 이름을 MessageUtil.java로 지정한 다음 이를 src/main/java 디렉터리에 저장합니다.

```
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;
  }
}
```

이 클래스 파일은 자신에게 전달되는 문자열을 출력으로 생성합니다. MessageUtil 생성자는 문자열을 설정합니다. printMessage 메서드는 출력을 생성합니다. salutationMessage 메서드는 Hi! 다음에 문자열을 출력합니다.

3. 다음 파일을 생성하고 이름을 TestMessageUtil.java로 지정한 다음 이를 /src/test/java 디렉터리에 저장합니다.

```
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());
  }
}
```

이 클래스 파일은 MessageUtil 클래스의 message 변수를 Robert로 설정합니다. 그런 다음 테스트를 수행하여 Robert 및 Hi!Robert 문자열이 출력에 나타나는지 여부를 확인하여 message 변수가 성공적으로 설정되었는지를 확인합니다.

4. 다음 파일을 생성하고 이름을 pom.xml로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
<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 Maven은 이 파일의 지침을 따라 `MessageUtil.java` 및 `TestMessageUtil.java` 파일을 `messageUtil-1.0.jar`이라는 파일로 변환한 다음 지정된 테스트를 실행합니다.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

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

다음 단계

2단계: buildspec 파일 생성

2단계: buildspec 파일 생성

(이전 단계: 1단계: 소스 코드 생성)

이 단계에서는 빌드 사양 파일을 생성합니다. 빌드스펙은 빌드를 실행하는 데 사용되는 빌드 명령 및 관련 설정 (YAML 형식) 의 모음입니다. CodeBuild 빌드 사양이 없으면 빌드 입력을 빌드 출력으로 성공적으로 변환하거나 빌드 환경에서 빌드 출력 아티팩트를 찾아 출력 버킷에 업로드할 수 CodeBuild 없습니다.

다음 파일을 생성하고 이름을 `buildspec.yml`로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
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

빌드 사양 선언은 올바른 YAML이어야 하므로 빌드 사양 선언의 공백이 중요합니다. 빌드 사양 선언의 공백 수가 YAML과 맞지 않으면 빌드가 즉시 실패할 수 있습니다. YAML 검사기를 사용하여 빌드 사양 선언이 올바른 YAML인지 여부를 테스트할 수 있습니다.

Note

소스 코드에 빌드 사양 파일을 포함하는 대신, 빌드 프로젝트를 생성할 때 빌드 명령을 별도로 선언할 수 있습니다. 이 방법은 매번 소스 코드 리포지토리를 업데이트하지 않아도 되도록 여러 개의 빌드 명령이 있는 소스 코드를 빌드하려는 경우 유용합니다. 자세한 설명은 buildspec 구문 섹션을 참조하세요.

이 빌드 사양 선언에서:

- `version`은 사용 중인 빌드 사양 표준의 버전을 나타냅니다. 이 빌드 사양 선언은 최신 버전인 `0.2`을 사용합니다.
- `phases`는 CodeBuild 가 명령을 실행하도록 지시할 수 있는 빌드 단계를 나타냅니다. 여기에서는 이 빌드 단계가 `install`, `pre_build`, `build` 및 `post_build`로 나열되어 있습니다. 이 빌드 단계 이름의 철자는 변경할 수 없으며 추가로 빌드 단계 이름을 생성할 수도 없습니다.

  이 예시에서는 `build` 단계 중에 명령을 CodeBuild 실행합니다. `mvn install` 이 명령은 Apache Maven이 Java 클래스 파일을 컴파일 및 테스트하고 컴파일된 Java 클래스 파일을 빌드 출력 결과물에 패키지하도록 지시합니다. 그리고 몇 가지 `echo` 명령을 각 빌드 단계에 추가하여 이 연습을 마치

게 됩니다. 이 자습서의 뒷부분에서 자세한 빌드 정보를 확인할 때 이러한 echo 명령의 출력을 확인하면 CodeBuild가 명령을 실행하는 방법 및 명령 실행 순서를 이해하는 데 많은 도움이 됩니다. (이 예에는 모든 빌드 단계가 포함되어 있지만, 해당 단계에서 아무 명령도 실행하지 않으려면 빌드 단계를 포함하지 않아도 됩니다.) 각 빌드 단계에서 지정된 각 명령을 나열된 순서대로 한 번에 하나씩 처음부터 끝까지 CodeBuild 실행합니다.

- `artifacts`출력 버킷에 CodeBuild 업로드되는 빌드 출력 아티팩트 세트를 나타냅니다. `files`빌드 출력에 포함할 파일을 나타냅니다. CodeBuild 빌드 환경의 `target` 상대 디렉터리에 있는 단일 `messageUtil-1.0.jar` 파일을 업로드합니다. 파일 이름 `messageUtil-1.0.jar` 및 디렉터리 이름 `target`은 Apache Maven이 이 예제에서만 빌드 출력 결과물을 생성 및 저장하는 방식에 따라 달라집니다. 사용자 자체 빌드에서는 이러한 파일 이름과 디렉터리가 다릅니다.

자세한 내용은 buildspec 참조 섹션을 참조하십시오.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

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

다음 단계

3단계: 두 개의 S3 버킷 생성

3단계: 두 개의 S3 버킷 생성

(이전 단계: 2단계: buildspec 파일 생성)

한 개의 버킷을 사용하여 이 실습을 수행할 수도 있지만, 두 개의 버킷을 사용해야 빌드 입력이 들어오는 위치 및 빌드 출력이 나가는 위치를 쉽게 확인할 수 있습니다.

- 이러한 버킷 중 하나(입력 버킷)는 빌드 입력을 저장합니다. 이 자습서에서 이 입력 버킷의 이름은 `codebuild-`*`region-ID`*`-`*`account-ID`*`-input-bucket`입니다. 여기서 *`region-ID`*는 버킷의 AWS 리전이고 *`account-ID`*는 AWS 계정 ID입니다.

- 다른 버킷(출력 버킷)은 빌드 출력을 저장합니다. 이 자습서에서 이 출력 버킷의 이름은 `codebuild-region-ID-account-ID-output-bucket`입니다.

이러한 버킷에 대해 다른 이름을 선택한 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

이 두 버킷은 빌드와 같은 AWS 리전에 있어야 합니다. 예를 들어 미국 동부 (오하이오) 지역에서 빌드를 CodeBuild 실행하도록 지시하는 경우 이러한 버킷은 미국 동부 (오하이오) 지역에도 있어야 합니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 버킷 생성을 참조하세요.

> **Note**
>
> CodeCommit GitHub, 및 Bitbucket 리포지토리에 저장된 빌드 CodeBuild 입력도 지원하지만 이 자습서에서는 사용 방법을 설명하지 않습니다. 자세한 설명은 빌드 계획 섹션을 참조하세요.

다음 단계

4단계: 소스 코드 및 buildspec 파일 업로드

4단계: 소스 코드 및 buildspec 파일 업로드

(이전 단계: 3단계: 두 개의 S3 버킷 생성)

이 단계에서는 소스 코드 및 빌드 사양 파일을 입력 버킷에 추가합니다.

사용하는 운영 체제의 zip 유틸리티를 사용하여 `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml` 및 `buildspec.yml`을 포함하는 `MessageUtil.zip`이라는 파일을 생성합니다.

`MessageUtil.zip` 파일의 디렉터리 구조가 다음과 같이 나타나야 합니다.

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

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

> **Important**
>
> *(root directory name)* 디렉터리는 포함하지 말고, *(root directory name)* 디렉터리 안에 있는 디렉터리 및 파일만 포함하십시오.

`MessageUtil.zip` 파일을 `codebuild-`*region-ID*`-`*account-ID*`-input-bucket`이라는 입력 버킷에 업로드합니다.

> **Important**
>
> 및 Bitbucket 리포지토리의 경우 일반적으로 각 리포지토리의 루트 (최상위 수준) `buildspec.yml` 에 이름이 지정된 빌드 사양 파일을 저장하거나 빌드 프로젝트 정의의 일부로 빌드 사양 선언을 포함해야 합니다. CodeCommit GitHub 리포지토리의 소스 코드 및 빌드 사양 파일을 포함하는 ZIP 파일은 생성하지 마십시오.
> S3 버킷에만 저장된 빌드 입력의 경우, 일반적으로 루트(최상위)에 소스 코드 및 `buildspec.yml`이라는 빌드 사양 파일을 포함하는 ZIP 파일을 생성하거나 빌드 사양 선언을 빌드 프로젝트 정의의 일부로 포함해야 합니다.
> 빌드 사양 파일에 다른 이름을 사용하거나 루트가 아닌 위치에서 빌드 사양을 참조하려면 빌드 사양 재정의를 빌드 프로젝트 정의의 일부로 지정할 수 있습니다. 자세한 설명은 buildspec 파일 이름 및 스토리지 위치 섹션을 참조하세요.

다음 단계

5단계: 빌드 프로젝트 생성

5단계: 빌드 프로젝트 생성

(이전 단계: 4단계: 소스 코드 및 buildspec 파일 업로드)

이 단계에서는 AWS CodeBuild가 빌드를 실행하는 데 사용할 빌드 프로젝트를 생성합니다. 이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다. 빌드 환경은 운영 체제, 프로그래밍 언어 런타임 및 빌드를 실행하는 데 CodeBuild 사용되는 도구의 조합을 나타냅니다. 빌드 환경은 도커 이미지로 표현됩니다. 자세한 정보는 Docker Docs 웹 사이트에서 Docker 개요 주제를 참조하십시오.

이 빌드 환경에서는 JDK (자바 개발 키트) 버전과 Apache Maven이 포함된 Docker 이미지를 사용하도록 CodeBuild 지시합니다.

빌드 프로젝트를 생성하려면

1. AWS CLI를 사용하여 다음 create-project 명령을 실행합니다.

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

JSON 형식 데이터가 출력에 표시됩니다. AWS CLI가 설치된 로컬 컴퓨터 또는 인스턴스의 위치에 있는 `create-project.json`이라는 파일에 데이터를 복사합니다. 다른 파일 이름을 사용하기로 선택하는 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

복사된 데이터를 다음 형식으로 수정한 다음 결과를 저장합니다.

```
{
  "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"
}
```

ServiceIamrole# 서비스 역할 (예:) CodeBuild 의 Amazon 리소스 이름 (ARN) 으로 대체합니다. `arn:aws:iam::`*account-ID*`:role/`*role-name* 파일을 만들려면 CodeBuild 서비스 역할 생성 섹션을 참조하세요.

이 데이터에서:

- `name`은 이 빌드 프로젝트에 대한 필수 식별자를 나타냅니다(이 예에서는 `codebuild-demo-project`). 빌드 프로젝트 이름은 계정에 있는 모든 빌드 프로젝트에서 고유해야 합니다.

- `source`의 `type`은 소스 코드의 리포지토리 유형을 나타내는 필수 값입니다(이 예에서는 Amazon S3 버킷의 경우 `S3`).
- `source`의 `location`은 소스 코드의 경로를 나타냅니다(이 예에서는 입력 버킷 이름과 그 뒤의 ZIP 파일 이름).
- `artifacts`의 `type`은 빌드 출력 아티팩트의 리포지토리 유형을 나타내는 필수 값입니다(이 예에서는 Amazon S3 버킷의 경우 `S3`).
- `artifacts`의 `location`은 앞에서 생성하거나 지정한 출력 버킷의 이름을 나타냅니다(이 예에서는 `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`).
- `environment`의 `type`은 빌드 환경의 유형을 나타내는 필수 값입니다(이 예에서는 `LINUX_CONTAINER`).
- For `image` 는 Docker 이미지 리포지토리 유형별로 지정된 대로 이 빌드 프로젝트에서 사용하는 Docker 이미지 이름 및 태그 조합을 나타내는 필수 값입니다 (이 예에서는 Docker 이미지 리포지토리의 Docker `aws/codebuild/standard:5.0` 이미지용). `environment` CodeBuild `aws/codebuild/standard`Docker 이미지의 이름입니다. `5.0`도커 이미지의 태그입니다.

  자신의 시나리오에서 사용할 수 있는 더 많은 도커 이미지를 찾으려면 빌드 환경 참조 단원을 참조하십시오.
- `environment`For `computeType` 는 CodeBuild 사용하는 컴퓨팅 리소스를 나타내는 필수 값입니다 (이 예에서는`BUILD_GENERAL1_SMALL`).

> **Note**
>
> 원래의 JSON 형식 데이터에서 사용 가능한 다른 값으로, `description`, `buildspec`, `auth` (`type` 및 `resource` 포함), `path`, `namespaceType`, `name`(`artifacts`의 경우), `packaging`, `environmentVariables`(`name` 및 `value` 포함), `timeoutInMinutes`, `encryptionKey` 및 `tags`(`key` 및 `value` 포함) 등이 있으며 선택 사항입니다. 이러한 값은 이 자습서에서 사용되지 않으므로 여기서는 다루지 않습니다. 자세한 설명은 빌드 프로젝트 생성(AWS CLI) 섹션을 참조하세요.

2. 방금 저장한 파일이 들어 있는 디렉터리로 전환한 다음, create-project 명령을 다시 실행합니다.

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

이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다.

```
{
  "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`는 이 빌드 프로젝트에 대한 정보를 나타냅니다.
  - `tags`는 선언된 태그를 나타냅니다.
  - `packaging`은 빌드 출력 결과물이 출력 버킷에 저장되는 방식을 나타냅니다. `NONE`은 폴더가 출력 버킷 내부에 생성됨을 의미합니다. 빌드 출력 결과물이 해당 폴더에 저장됩니다.
  - `lastModified`는 빌드 프로젝트에 대한 정보가 마지막으로 변경된 시간을 Unix 시간 형식으로 나타냅니다.
  - `timeoutInMinutes`빌드가 완료되지 않은 경우 빌드가 CodeBuild 중지되기까지 걸리는 시간 (분) 을 나타냅니다. (기본값은 60분입니다.)
  - `created`는 빌드 프로젝트가 생성된 시간을 Unix 시간 형식으로 나타냅니다.

- `environmentVariables`선언되어 빌드 중에 사용할 CodeBuild 수 있는 모든 환경 변수를 나타냅니다.
- `encryptionKey`빌드 출력 아티팩트를 암호화하는 CodeBuild 데 사용된 고객 관리 키의 ARN을 나타냅니다.
- `arn`은 빌드 프로젝트의 ARN을 나타냅니다.

> **Note**
>
> create-project명령을 실행한 후 다음과 유사한 오류 메시지가 출력될 수 있습니다. 사용자: ***User-ARN#*** 수행할 권한이 없음: codebuild:. CreateProject 이는 빌드 프로젝트를 만드는 데 사용할 CodeBuild 수 있는 충분한 권한이 없는 사용자의 자격 AWS CLI 증명으로 를 구성했기 때문일 가능성이 큽니다. 이 문제를 해결하려면 IAM 엔터티 중 하나에 속한 보안 인증을 사용하여 AWS CLI를 구성하세요.
>
> - AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
> - 해당 사용자 또는 사용자가 속한 IAM 그룹에 연결된 `AWSCodeBuildAdminAccess`, `AmazonS3ReadOnlyAccess` 및 `IAMFullAccess` 관리형 정책을 사용하는 AWS 계정의 사용자. AWS 계정에 이러한 권한이 있는 사용자 또는 그룹이 없으며, 이러한 권한을 사용자나 그룹에 추가할 수 없는 경우, AWS 계정 관리자에게 지원을 요청하세요. 자세한 설명은 AWS에 대한 관리형 (사전 정의된) 정책 AWS CodeBuild 섹션을 참조하세요.

다음 단계

6단계: 빌드 실행

6단계: 빌드 실행

(이전 단계: 5단계: 빌드 프로젝트 생성)

이 단계에서는 빌드 프로젝트의 설정으로 빌드를 실행하도록 AWS CodeBuild에 지시를 내립니다.

빌드를 실행하려면

1. AWS CLI를 사용하여 다음 start-build 명령을 실행합니다.

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

*project-name*을 이전 단계의 빌드 프로젝트 이름으로 바꿉니다(예: codebuild-demo-project).

2. 이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다.

```
{
  "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"
  }
}
```

- build는 이 빌드에 대한 정보를 나타냅니다.
  - buildComplete는 빌드 완료 여부를 나타냅니다(true). 그렇지 않을 경우 false입니다.
  - initiator는 빌드를 시작한 엔터티를 나타냅니다.
  - artifacts는 빌드 출력에 대한 정보를 나타냅니다(위치 포함).

- `projectName`은 빌드 프로젝트의 이름을 나타냅니다.
- `buildStatus`는 start-build 명령이 실행되었을 당시의 빌드 상태를 나타냅니다.
- `currentPhase`는 start-build 명령이 실행되었을 당시의 빌드 단계를 나타냅니다.
- `startTime`은 빌드 프로세스가 시작된 시간을 Unix 시간 형식으로 나타냅니다.
- `id`는 빌드의 ID를 나타냅니다.
- `arn`은 빌드의 ARN을 나타냅니다.

[id] 값을 기록해 둡니다. 이 정보는 다음 단계에서 필요합니다.

다음 단계

7단계: 요약된 빌드 정보 보기

7단계: 요약된 빌드 정보 보기

(이전 단계: 6단계: 빌드 실행)

이 단계에서는 빌드 상태에 대한 요약 정보를 확인합니다.

요약된 빌드 정보를 보려면

AWS CLI를 사용하여 batch-get-builds 명령을 실행합니다.

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

*id*를 이전 단계의 출력에 표시된 `id` 값으로 바꿉니다.

이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다.

```
{
  "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`는 정보를 찾을 수 없는 빌드의 빌드 ID를 나타냅니다. 이 예에서는 비어 있어야 합니다.
- `builds`는 정보를 찾을 수 있는 각 빌드에 대한 정보를 나타냅니다. 이 예에서는 한 빌드에 대한 정보만 출력에 표시됩니다.
  - `phases`빌드 프로세스 중에 CodeBuild 실행되는 빌드 단계 세트를 나타냅니다. 각 빌드 단계에 대한 정보는 `startTime`, `endTime` 및 `durationInSeconds`(빌드 단계가 시작 및 종료된 시간은 Unix 형식으로, 지속된 기간은 초로 표시)뿐만 아니라 `phaseType`(`SUBMITTED`, `PROVISIONING`, `DOWNLOAD_SOURCE`, `INSTALL`, `PRE_BUILD`, `BUILD`, `POST_BUILD`, `UPLOAD_ARTIFACTS`, `FINALIZING` 또는 `COMPLETED` 등) 및 `phaseStatus`(`SUCCEEDED`, `FAILED`, `FAULT`, `TIMED_OUT`, `IN_PROGRESS` 또는 `STOPPED`)가 개별적으로 나열됩니다. batch-get-builds 명령을 처음으로 실행하면 단계가 많이 표시되지 않거나 아예 하나도 표시되지 않을 수 있습니다. 동일한 빌드 ID로 batch-get-builds 명령을 계속하여 실행하면 더 많은 빌드 단계가 출력에 표시됩니다.
  - `logs`빌드 CloudWatch 로그에 대한 Amazon Logs의 정보를 나타냅니다.
  - `md5sum` 및 `sha256sum`은 빌드 출력 결과물의 MD5 및 SHA-256 해시를 나타냅니다. 이러한 값은 빌드 프로젝트의 `packaging` 값이 `ZIP`으로 설정되어 있는 경우에만 출력에 표시됩니다. (이 자습서에서는 이 값을 설정하지 않음) 이러한 해시를 체크섬 도구와 함께 사용하면 파일 무결성 및 신뢰성을 확인할 수 있습니다.

    > **Note**
    >
    > Amazon S3 콘솔을 사용해서도 이러한 해시를 확인할 수 있습니다. 빌드 출력 결과물 옆의 확인란을 선택하고 작업, 속성을 차례로 선택합니다. 속성 창에서 메타데이터를 확장하고 x-amz-meta-codebuild-content-md5 및 -content-sha256의 값을 확인합니다. x-amz-meta-codebuild (Amazon S3 콘솔에서 빌드 출력 아티팩트의 ETag 값을 MD5 또는 SHA-256 해시로 해석하지 않아야 합니다.)
    > AWS SDK를 사용하여 이러한 해시를 확인하려면 `codebuild-content-md5` 및 `codebuild-content-sha256`이라는 값을 확인하십시오.

  - `endTime`은 빌드 프로세스가 종료된 시간을 Unix 시간 형식으로 나타냅니다.

> **Note**
>
> Amazon S3 메타데이터에는 Amazon S3에 아티팩트를 게시하는 `buildArn` CodeBuild 빌드의 이름이 `x-amz-meta-codebuild-buildarn` 포함된 CodeBuild 헤더가 있습니다. 알림에 대한 소스 추적을 허용하고 아티팩트가 생성된 빌드를 참조할 수 있도록 하기 위해 `buildArn`이 추가되었습니다.

다음 단계

8단계: 자세한 빌드 정보 보기

8단계: 자세한 빌드 정보 보기

(이전 단계: 7단계: 요약된 빌드 정보 보기)

이 단계에서는 로그에서 CloudWatch 빌드에 대한 세부 정보를 확인합니다.

> **Note**
>
> 민감한 정보를 보호하기 위해 CodeBuild 로그에는 다음이 숨겨져 있습니다.
>
> - AWS액세스 키 ID: 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.
> - 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
> - AWS Secrets Manager을 사용하여 지정한 문자열입니다. 자세한 설명은 키 관리 섹션을 참조하세요.

자세한 빌드 정보를 보려면

1. 웹 브라우저를 사용하여 이전 단계의 출력에 표시된 `deepLink` 위치로 이동합니다(예: `https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE`).
2. 로그 CloudWatch 로그 스트림에서 로그 이벤트를 찾아볼 수 있습니다. 기본적으로 가장 최근의 로그 이벤트 세트만 표시됩니다. 이전의 로그 이벤트를 보려면 목록의 처음으로 스크롤합니다.

3. 이 자습서에서는 대부분의 로그 이벤트에 CodeBuild 가 빌드 종속성 파일을 빌드 환경에 다운로드 및 설치하는 작업에 대한 자세한 정보가 들어 있는데, 대부분의 사용자에게 필요하지 않은 정보입니다. [Filter events]를 사용하면 표시되는 정보를 줄일 수 있습니다. 예를 들어, 필터 이벤트에 "[INFO]"를 입력하면 [INFO]를 포함하는 이벤트만 표시됩니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 필터 및 패턴 구문을 참조하십시오.

로그 CloudWatch 로그 스트림의 다음 부분은 이 자습서와 관련이 있습니다.

```
...
[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
```

이 예제에서는 사전 빌드, 빌드 및 빌드 후 빌드 단계를 CodeBuild 성공적으로 완료했습니다. 또한 단위 테스트를 실행하고 `messageUtil-1.0.jar` 파일을 성공적으로 빌드했습니다.

다음 단계

9단계: 빌드 출력 아티팩트 가져오기

9단계: 빌드 출력 아티팩트 가져오기

(이전 단계: 8단계: 자세한 빌드 정보 보기)

이 단계에서는 CodeBuild 빌드된 `messageUtil-1.0.jar` 파일을 가져와 출력 버킷에 업로드합니다.

CodeBuild 콘솔 또는 Amazon S3 콘솔을 사용하여 이 단계를 완료할 수 있습니다.

빌드 출력 결과물을 가져오려면(AWS CodeBuild 콘솔)

1. CodeBuild 콘솔이 여전히 열려 있고 이전 단계의 빌드 세부 정보 페이지가 계속 표시된 상태에서 빌드 세부 정보 탭을 선택하고 Artifacts 섹션으로 스크롤합니다.

> **Note**
>
> 빌드 세부 정보 페이지가 표시되지 않으면, 탐색 모음에서 빌드 이력을 선택한 다음, 빌드 실행 링크를 선택합니다.

2. Amazon S3 폴더로 연결되는 링크는 아티팩트 업로드 위치 아래에 있습니다. 이 링크를 클릭하면 `messageUtil-1.0.jar` 빌드 출력 아티팩트 파일을 찾는 Amazon S3의 폴더가 열립니다.

빌드 출력 아티팩트를 가져오려면(Amazon S3 콘솔)

1. https://console.aws.amazon.com/s3/ 에서 Amazon S3 콘솔을 엽니다.
2. `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`를 엽니다.
3. `codebuild-demo-project` 폴더를 엽니다.
4. `target` 폴더를 엽니다. 이 폴더에서 `messageUtil-1.0.jar` 빌드 출력 결과물 파일을 찾을 수 있습니다.

다음 단계

10단계: S3 버킷 삭제

10단계: S3 버킷 삭제

(이전 단계: 9단계: 빌드 출력 아티팩트 가져오기)

AWS 계정에 계속하여 요금이 부과되지 않도록 이 자습서에서 사용한 입력 및 출력 버킷을 삭제할 수 있습니다. 지침을 보려면 Amazon Simple Storage Service 사용 설명서에서 버킷 삭제 또는 비우기를 참조하세요.

IAM 사용자 또는 관리자 IAM 사용자를 통해 이러한 버킷을 삭제하는 경우 사용자에게 추가 액세스 권한이 있어야 합니다. 사용자의 기존 액세스 정책에 마커 사이의 다음 명령문(*`### BEGIN ADDING STATEMENT HERE ###`* 및 *`### END ADDING STATEMENTS HERE ###`*)을 추가합니다.

이 명령문의 줄임표(...)는 간결하게 나타내기 위해 사용됩니다. 기존 액세스 정책의 어떤 명령문도 제거하지 마십시오. 이러한 줄임표는 정책에 입력하지 않아야 합니다.

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

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

다음 단계

마무리

마무리

이 자습서에서는 AWS CodeBuild를 사용하여 Java 클래스 파일 세트를 JAR 파일에 빌드했습니다. 그리고 빌드 결과를 확인했습니다.

이제 자체 CodeBuild 시나리오에서 사용해 볼 수 있습니다. 빌드 계획의 지침을 따르세요. 아직 시도해 볼 준비가 되지 않은 것 같으면 샘플 몇 가지를 더 빌드해 볼 수 있습니다. 자세한 내용은 샘플을(를) 참조하세요.

CodeBuild 샘플

다음 샘플 그룹을 사용하여 AWS CodeBuild다음을 실험할 수 있습니다.

주제

- 사례 기반 샘플을 사용하여 다음을 수행하십시오. CodeBuild
- 마이크로소프트 윈도우용 샘플 CodeBuild

사례 기반 샘플을 사용하여 다음을 수행하십시오. CodeBuild

다음과 같은 사용 사례 기반 샘플을 사용하여 다음을 실험할 수 있습니다. AWS CodeBuild

크로스 서비스 샘플

실험해 볼 수 있는 크로스 서비스 샘플 목록. AWS CodeBuild

빌드 배지 샘플

빌드 배지를 CodeBuild 사용하여 설정하는 방법을 보여 줍니다.

AWS CLI 샘플을 사용하여 테스트 보고서 작성

AWS CLI 를 사용하여 테스트 보고서를 만들고 실행하고 결과를 확인합니다.

도커 샘플: CodeBuild

사용자 지정 Docker 이미지를 사용하고, Amazon ECR의 리포지토리에 Docker 이미지를 게시하고, 프라이빗 레지스트리에서 Docker 이미지를 사용하는 방법을 보여 줍니다.

빌드 출력을 S3 버킷에서 호스팅

S3 버킷에서 암호화되지 않은 빌드 아티팩트를 사용해 정적 웹사이트를 생성하는 방법을 보여줍니다.

다중 입력 소스 및 출력 아티팩트 샘플

빌드 프로젝트에서 다중 입력 소스와 다중 출력 아티팩트를 사용하는 방법을 보여 줍니다.

buildspec 파일 샘플의 런타임 버전

buildspec 파일에서 런타임과 그 버전을 지정하는 방법을 보여줍니다.

소스 버전 샘플

빌드 프로젝트에서 특정 버전의 소스를 사용하는 방법을 보여 줍니다. CodeBuild

[에 대한 타사 소스 리포지토리 샘플 CodeBuild]

웹 후크를 사용하여 GitHub CodeBuild Enterprise Server 및 GitHub 풀 요청을 생성하는 BitBucket 방법을 보여 줍니다.

[의미 체계 버전 관리를 사용하여 빌드 아티팩트 샘플 이름 지정]

빌드 시 의미 체계 버전 관리를 사용해 아티팩트 이름을 생성하는 방법을 보여 줍니다.

에 대한 크로스 서비스 샘플 CodeBuild

이러한 크로스 서비스 샘플을 사용하여 다음을 실험할 수 있습니다. AWS CodeBuild

[Amazon ECR 샘플]

Amazon ECR 리포지토리의 도커 이미지를 사용하여 Apache Maven을 사용하여 단일 JAR 파일을 생성합니다.

[Amazon EFS 샘플]

CodeBuild 프로젝트가 Amazon EFS 파일 시스템에 마운트되고 빌드되도록 buildspec 파일을 구성하는 방법을 보여 줍니다.

[AWS CodePipeline 샘플]

를 AWS CodePipeline 사용하여 배치 빌드뿐 아니라 여러 입력 소스 및 여러 출력 아티팩트가 포함된 빌드를 생성하는 방법을 보여 줍니다.

[AWS Config 샘플]

설정 AWS Config방법을 보여 줍니다. 추적되는 CodeBuild 리소스를 나열하고 CodeBuild 프로젝트를 조회하는 방법을 설명합니다. AWS Config

[빌드 알림 샘플]

Apache Maven을 사용하여 단일 JAR 파일을 생성합니다. Amazon SNS 주제 구독자에게 빌드 알림을 보냅니다.

에 대한 아마존 ECR 샘플 CodeBuild

이 샘플에서는 Amazon Elastic Container Registry(Amazon ECR) 이미지 리포지토리에 있는 도커 이미지를 사용하여 샘플 Go 프로젝트를 빌드합니다.

> **Important**
>
> 이 샘플을 실행하면 AWS 계정에 요금이 청구될 수 있습니다. 여기에는 Amazon S3 AWS KMS, CloudWatch 로그 및 Amazon ECR과 관련된 AWS 리소스 및 작업에 대한 가능한 요금이 포함됩니다. AWS CodeBuild 자세한 내용은 요금, Amazon S3 CodeBuild 요금, 요금, Amazon AWS Key Management Service 요금 및 Amazon CloudWatch Elastic 컨테이너 레지스트리 요금을 참조하십시오.

샘플 실행

이 샘플을 실행하려면

1. 도커 이미지를 생성하고 Amazon ECR의 이미지 리포지토리에 푸시하려면, Amazon ECR 이미지 리포지토리에 Docker 이미지 게시 샘플의 "샘플 실행" 섹션에 있는 단계를 수행하세요.
2. Go 프로젝트 만들기:

   a. 이 주제의 Go 프로젝트 구조 및 Go 프로젝트 파일 섹션에 설명된 대로 파일을 만든 다음 S3 입력 버킷 또는 AWS CodeCommit GitHub, 또는 Bitbucket 리포지토리에 업로드하십시오.

   > **Important**
   >
   > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
   > S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

   b. 빌드 프로젝트를 만들고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.

   를 사용하여 빌드 프로젝트를 AWS CLI 만드는 경우 JSON 형식의 `create-project` 명령 입력이 이와 비슷해 보일 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "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. 빌드 출력 아티팩트를 가져오려면 S3 출력 버킷을 엽니다.

d. *GoOutputArtifact*.zip 파일을 로컬 컴퓨터나 인스턴스에 다운로드한 다음 파일의 내용을 추출합니다. 추출한 내용에서 hello 파일을 가져옵니다.

3. 다음 중 하나에 해당하는 경우 Docker 이미지를 빌드 환경으로 가져올 AWS CodeBuild 수 있도록 Amazon ECR의 이미지 리포지토리에 권한을 추가해야 합니다.

- 프로젝트는 CodeBuild 자격 증명을 사용하여 Amazon ECR 이미지를 가져옵니다. 이는 ProjectEnvironment의 imagePullCredentialsType 속성에 CODEBUILD 값으로 표시됩니다.
- 프로젝트에서는 교차 계정 Amazon ECR 이미지를 사용합니다. 이 경우에는 프로젝트에서 서비스 역할을 사용하여 Amazon ECR 이미지를 끌어와야 합니다. 이 동작을 활성화하려면 ProjectEnvironment의 imagePullCredentialsType 속성을 SERVICE_ROLE로 설정합니다.

1. Amazon ECR 콘솔(https://console.aws.amazon.com/ecr/)을 엽니다.
2. 리포지토리 이름 목록에서 생성했거나 선택한 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 권한을 선택하고, 편집을 선택한 다음 설명문 추가를 선택합니다.
4. Statement name(설명문 이름)에 식별자(예: **CodeBuildAccess**)를 입력합니다.
5. 효과에 대해 허용이 선택된 채로 둡니다. 이는 다른 AWS 계정으로의 액세스를 허용하겠다는 의미입니다.
6. 보안 주체에 대해 다음 중 하나를 실시합니다.

- 프로젝트에서 CodeBuild 자격 증명을 사용하여 Amazon ECR 이미지를 가져오는 경우 서비스 주체에 다음을 입력합니다**codebuild.amazonaws.com**.
- 프로젝트에서 교차 계정 Amazon ECR 이미지를 사용할 경우에는 AWS 계정 ID에 액세스 권한을 부여할 AWS 계정의 ID를 입력합니다.

7. All IAM entities(모든 IAM 엔터티) 목록을 건너뜁니다.
8. [액션] 에서 풀 전용 작업 (ecr:, ecr:GetDownloadUrlForLayer, ecr: 및 ecr:BatchGetImage) 을 선택합니다. BatchCheckLayerAvailability
9. 조건에 다음을 추가합니다.

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

10저장을 선택합니다.

이 정책이 Permissions(권한)에 표시됩니다. 보안 주체는 이 절차의 3단계에서 보안 주체에 입력한 내용입니다.

- 프로젝트에서 CodeBuild 자격 증명을 사용하여 Amazon ECR 이미지를 가져오는 경우 서비스 주체 아래에 "codebuild.amazonaws.com" 표시됩니다.
- 프로젝트에서 교차 계정 Amazon ECR 이미지를 사용하는 경우 액세스 권한을 부여하려는 계정의 ID가 AWS 계정 ID 아래에AWS 표시됩니다.

다음 샘플 정책은 CodeBuild 자격 증명과 계정 간 Amazon ECR 이미지를 모두 사용합니다.

```
{
   "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"
         ]
      }
   ]
}
```

- 프로젝트에서 CodeBuild 자격 증명을 사용하고 프로젝트에서 Amazon ECR 리포지토리에 대한 공개 액세스 권한을 갖도록 하려면 Condition 키를 생략하고 다음 샘플 정책을 추가할 수 있습니다. CodeBuild

```
{
  "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. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 빌드 정보를 확인합니다.

   를 사용하여 빌드 프로젝트를 AWS CLI 만드는 경우 JSON 형식의 `create-project` 명령 입력이 다음과 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "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. 빌드 출력 아티팩트를 가져오려면 S3 출력 버킷을 엽니다.
6. *GoOutputArtifact*.zip 파일을 로컬 컴퓨터나 인스턴스에 다운로드한 다음 *GoOutputArtifact*.zip 파일의 내용을 추출합니다. 추출한 내용에서 hello 파일을 가져옵니다.

Go 프로젝트 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

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

Go 프로젝트 파일

이 샘플은 다음 파일을 사용합니다.

buildspec.yml(*(root directory name)*에 있음)

```
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(*(root directory name)*에 있음)

```
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)
}
```

관련 리소스

- 시작하는 방법에 대한 자세한 내용은 을 참조하십시오. AWS CodeBuild콘솔을 사용하여 AWS CodeBuild 시작하기
- 의 문제 해결에 대한 자세한 내용은 CodeBuild 을 참조하십시오문제 해결 AWS CodeBuild.
- 할당량에 대한 자세한 내용은 CodeBuild 을 참조하십시오. AWS CodeBuild에 대한 할당량

에 대한 Amazon Elastic File System 샘플 AWS CodeBuild

Amazon EC2 인스턴스를 위한 확장 가능한 공유 파일 서비스인 Amazon Elastic File System에서 AWS CodeBuild 빌드를 생성할 수도 있습니다. Amazon EFS를 통한 스토리지 용량은 탄력적이므로 파일이 추가 및 제거될 때 확장되거나 축소됩니다. 또한 파일 시스템을 만들고 구성할 수 있는 간편한 웹 서비스 인터페이스를 제공합니다. 이 서비스는 모든 파일 스토리지 인프라를 관리하므로, 파일 시스템 구성을 배포하거나 패치를 적용하거나 유지 보수하는 데 신경을 쓸 필요가 없습니다. 자세한 내용은 Amazon Elastic File System 사용 설명서에서 Amazon Elastic File System이란?을 참조하세요.

이 샘플은 Java 애플리케이션을 Amazon EFS 파일 시스템에 마운트하고 빌드하도록 CodeBuild 프로젝트를 구성하는 방법을 보여줍니다. 시작하기 전에 S3 입력 버킷 또는, AWS CodeCommit GitHub, GitHub 엔터프라이즈 서버 또는 Bitbucket 리포지토리에 업로드된 Java 애플리케이션을 빌드할 준비가 되어 있어야 합니다.

파일 시스템에 전송되는 데이터는 암호화됩니다. 다른 이미지를 사용하여 전송 중인 데이터를 암호화하려면 전송 중 데이터 암호화를 참조하십시오.

상위 수준 단계

이 샘플은 다음과 함께 AWS CodeBuild Amazon EFS를 사용하는 데 필요한 세 가지 상위 단계를 다룹니다.

1. 계정에 가상 사설 클라우드 (VPC) 를 AWS 생성합니다.
2. 이 VPC를 사용하는 파일 시스템을 생성합니다.
3. VPC를 사용하는 CodeBuild 프로젝트를 만들고 빌드합니다. CodeBuild 프로젝트는 다음을 사용하여 파일 시스템을 식별합니다.
   - 고유한 파일 시스템 식별자. 식별자는 빌드 프로젝트에서 파일 시스템을 지정할 때 선택합니다.
   - 파일 시스템 ID. Amazon EFS 콘솔에서 파일 시스템을 볼 때 ID가 표시됩니다.
   - 탑재 지점. 파일 시스템을 탑재하는 Docker 컨테이너에 있는 디렉터리입니다.
   - 탑재 옵션. 여기에는 파일 시스템을 탑재하는 방법에 대한 세부 정보가 포함됩니다.

> **Note**
>
> Amazon EFS에서 생성한 파일 시스템은 Linux 플랫폼에서만 지원됩니다.

를 사용하여 VPC 생성 AWS CloudFormation

템플릿으로 AWS CloudFormation VPC를 생성합니다.

1. AWS CloudFormation VPC 템플릿에 나와 있는 지침에 따라 VPC를 생성하십시오. AWS CloudFormation

   > **Note**
   >
   > 이 AWS CloudFormation 템플릿으로 만든 VPC에는 프라이빗 서브넷 2개와 퍼블릭 서브넷 2개가 있습니다. Amazon EFS에서 생성한 파일 시스템을 AWS CodeBuild 마운트하는 데 사용할 때는 프라이빗 서브넷만 사용해야 합니다. 퍼블릭 서브넷 중 하나를 사용할 경우 빌드가 실패합니다.

2. AWS Management Console 로그인하고 https://console.aws.amazon.com/vpc/ 에서 Amazon VPC 콘솔을 엽니다.
3. 생성할 때 사용한 VPC를 선택합니다. AWS CloudFormation
4. 설명 탭에 표시된 VPC의 이름과 ID를 기록해 둡니다. 이 샘플의 뒷부분에서 AWS CodeBuild 프로젝트를 생성할 때는 둘 다 필요합니다.

VPC에 Amazon Elastic File System 파일 시스템 생성

이 예제를 위해 앞에서 만든 VPC를 사용하여 간단한 Amazon EFS 파일 시스템을 생성합니다.

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/efs/ 에서 Amazon EFS 콘솔을 엽니다.
2. 파일 시스템 생성을 선택합니다.
3. 이 예제 앞부분에서 기록해 둔 VPC 이름을 VPC에서 선택합니다.
4. 가용 영역은 선택한 서브넷과 연결된 채로 둡니다.
5. 다음 단계를 선택합니다.
6. 태그 추가에서 기본 이름 키의 값에 Amazon EFS 파일 시스템의 이름을 입력합니다.
7. 기본 성능 및 처리량 모드로 선택한 Bursting(버스팅) 및 범용 모드를 그대로 두고 다음 단계를 선택합니다.
8. 클라이언트 액세스 구성에서 다음 단계를 선택합니다.
9. 파일 시스템 생성을 선택합니다.
10. (선택 사항) 전송 중 데이터를 암호화하는 정책을 Amazon EFS 파일 시스템에 추가하는 것이 좋습니다. Amazon EFS 콘솔에서 파일 시스템 정책을 선택하고 편집을 선택한 다음, 모든 클라이언트에 전송 중 암호화 적용이라는 레이블이 붙은 상자를 선택하고 저장을 선택합니다.

Amazon EFS와 함께 사용할 CodeBuild 프로젝트 생성

이 샘플의 앞부분에서 만든 VPC를 사용하는 AWS CodeBuild 프로젝트를 생성합니다. 빌드가 실행되면 이전에 생성된 Amazon EFS 파일 시스템을 탑재합니다. 그런 다음 Java 애플리케이션에서 생성된 .jar 파일을 파일 시스템의 탑재 지점 디렉터리에 저장합니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 빌드 프로젝트를 선택한 후 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 프로젝트의 이름을 입력합니다.
4. 소스 공급자에서 빌드하려는 Java 애플리케이션을 포함하는 리포지토리를 선택합니다.
5. 애플리케이션을 찾는 데 CodeBuild 사용되는 정보 (예: 저장소 URL) 를 입력합니다. 옵션은 소스 공급자마다 다릅니다. 자세한 정보는 Choose source provider을 참조하세요.
6. 환경 이미지에서 관리형 이미지를 선택합니다.
7. 운영 체제에서 Amazon Linux 2를 선택합니다.

8. 런타임에서 표준을 선택합니다.
9. 이미지에서 aws/codebuild/amazonlinux2-x86_64-standard:4.0을 선택합니다.
10. 환경 유형에서 Linux를 선택합니다.
11. 서비스 역할에서 새 서비스 역할을 선택합니다. 역할 이름에 사용자를 위해 CodeBuild 생성한 역할의 이름을 입력합니다.
12. 추가 구성을 확장합니다.
13. Docker 이미지를 빌드하거나 빌드에서 승격된 권한을 얻으려는 경우 이 플래그 활성화를 선택합니다.

> **Note**
>
> 기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.

14. VPC에서 VPC ID를 선택합니다.
15. 서브넷에서 VPC에 연결된 프라이빗 서브넷을 한 개 이상 선택합니다. Amazon EFS 파일 시스템을 마운트하는 빌드에 프라이빗 서브넷을 사용해야 합니다. 퍼블릭 서브넷을 사용할 경우 빌드가 실패합니다.
16. 보안 그룹에서 기본 보안 그룹을 선택합니다.
17. 파일 시스템에 다음 정보를 입력합니다.

    - 식별자에 고유한 파일 시스템 식별자를 입력합니다. 길이는 129자 미만이어야 하고 영숫자 문자와 밑줄만 사용할 수 있습니다. CodeBuild 이 식별자를 사용하여 엘라스틱 파일 시스템을 식별하는 환경 변수를 생성합니다. 환경 변수 형식은 `CODEBUILD_<file_system_identifier>`(대문자)입니다. 예를 들어 `my_efs`을 입력하면 환경 변수는 `CODEBUILD_MY_EFS`입니다.
    - ID에서 파일 시스템 ID를 선택합니다.
    - (선택 사항) 파일 시스템의 디렉토리를 입력합니다. CodeBuild 이 디렉토리를 마운트합니다. 디렉토리 경로를 비워 두면 전체 파일 CodeBuild 시스템을 마운트합니다. 경로는 파일 시스템의 루트와 관련이 있습니다.
    - 탑재 지점에 파일 시스템을 탑재하는 빌드 컨테이너의 디렉터리 절대 경로를 입력합니다. 이 디렉토리가 없는 경우 빌드 중에 디렉토리를 CodeBuild 생성합니다.
    - (선택 사항) 탑재 옵션을 입력합니다. 마운트 옵션을 비워 두면 기본 마운트 옵션을 CodeBuild 사용합니다.

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

자세한 내용은 Amazon Elastic File System 사용 설명서의 권장되는 NFS 탑재 옵션을 참조하세요.

18. 빌드 사양에서 빌드 명령 삽입을 선택한 후 편집기로 전환을 선택합니다.
19. 편집기에 다음 buildspec 명령을 입력합니다. 17단계에서 입력한 식별자로 *<file_system_identifier>*를 바꿉니다. 대문자(예: CODEBUILD_MY_EFS)를 사용합니다.

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

20. 기타 모든 설정에 대해 기본값을 사용하고 빌드 프로젝트 생성을 선택합니다. 빌드가 완료되면 프로젝트의 콘솔 페이지가 표시됩니다.
21. 빌드 시작를 선택합니다.

CodeBuild 및 Amazon EFS 샘플 요약

AWS CodeBuild 프로젝트가 빌드된 후:

- Amazon EFS 파일 시스템에 빌드된 Java 애플리케이션에서 생성된 .jar 파일을 탑재 지점 디렉터리에 저장합니다.
- 파일 시스템을 식별하는 환경 변수는 프로젝트를 만들 때 입력한 파일 시스템 식별자를 사용하여 생성됩니다.

자세한 내용은 Amazon Elastic File System 사용 설명서에서 파일 시스템 탑재를 참조하세요.

문제 해결

Amazon EFS를 설정할 때 발생할 수 있는 오류는 다음과 같습니다 CodeBuild.

주제

- CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 권한이 거부되었습니다.
- CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 피어에서 연결을 재설정했습니다.
- VPC_CLIENT_ERROR: 예상치 못한 EC2 오류: UnauthorizedOperation

CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 권한이 거부되었습니다.

Amazon EFS를 사용하여 CodeBuild 마운트하는 경우에는 IAM 인증이 지원되지 않습니다. 사용자 지정 Amazon EFS 파일 시스템 정책을 사용하는 경우 모든 IAM 보안 주체에 읽기 및 쓰기 액세스 권한을 부여해야 합니다. 예:

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

CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 피어에서 연결을 재설정했습니다.

이 오류의 가능한 두 가지 원인은 다음과 같습니다.

- CodeBuild VPC 서브넷은 Amazon EFS 탑재 대상과 다른 가용 영역에 있습니다. Amazon EFS 탑재 대상과 동일한 가용 영역에 VPC 서브넷을 추가하여 이 문제를 해결할 수 있습니다.
- 보안 그룹에는 Amazon EFS와 통신할 권한이 없습니다. VPC(VPC의 기본 CIDR 블록 추가) 또는 보안 그룹 자체에서 들어오는 모든 트래픽을 허용하는 인바운드 규칙을 추가하여 이 문제를 해결할 수 있습니다.

VPC_CLIENT_ERROR: 예상치 못한 EC2 오류: UnauthorizedOperation

이 오류는 프로젝트의 VPC 구성에 있는 모든 서브넷이 퍼블릭 CodeBuild 서브넷일 때 발생합니다. 네트워크 연결을 보장하려면 VPC에 프라이빗 서브넷이 하나 이상 있어야 합니다.

CodePipeline 샘플: CodeBuild

주제

- AWS CodePipeline 통합 CodeBuild 및 일괄 빌드

- AWS CodePipeline 여러 입력 소스 CodeBuild 및 출력 아티팩트와의 통합 샘플

AWS CodePipeline 통합 CodeBuild 및 일괄 빌드

AWS CodeBuild 이제 일괄 빌드를 지원합니다. 이 샘플은 배치 빌드를 사용하는 빌드 프로젝트를 만드는 AWS CodePipeline 데 사용하는 방법을 보여줍니다.

파이프라인의 구조를 정의하는 JSON 형식의 파일을 사용한 다음, 와 함께 사용하여 파이프라인을 생성할 수 있습니다. AWS CLI 자세한 내용은AWS CodePipeline 사용 설명서의 AWS CodePipeline 파이프라인 구조 참조를 참조하세요.

개별 아티팩트를 사용한 배치 빌드

다음 JSON 파일을 별도의 아티팩트가 있는 배치 빌드를 생성하는 파이프라인 구조의 예로 사용하세요. 일괄 빌드를 활성화하려면 객체의 `BatchEnabled` 파라미터를 로 설정합니다. CodePipeline `configuration` `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
  }
}
```

다음은 이 파이프라인 구성과 함께 사용할 수 있는 CodeBuild buildspec 파일의 예제입니다.

```
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
```

파이프라인의 JSON 파일에서 지정하는 출력 아티팩트의 이름은 buildspec 파일에서 정의하는 빌드 및 아티팩트의 식별자와 일치해야 합니다. 구문은 기본 아티팩트의 경우 *buildIdentifier*이고 보조 아티팩트의 경우 *buildIdentifier_artifactIdentifier*입니다.

예를 들어 출력 아티팩트 이름의 build1 CodeBuild 경우 의 기본 아티팩트를 의 build1 위치에 업로드합니다. build1 출력 이름의 build1_artifact1 CodeBuild 경우 보조 build1 아티팩트를 artifact1 위치 build1_artifact1 등에 업로드합니다. 출력 위치를 하나만 지정하는 경우 이름은 *buildIdentifier*입니다.

JSON 파일을 생성하였으면 이제 파이프라인을 만들 수 있습니다. AWS CLI 를 사용하여 create-pipeline 명령을 실행하고 파일을 파라미터에 전달합니다. --cli-input-json 자세한 내용은AWS CodePipeline 사용 설명서의 파이프라인 생성(CLI)을 참조하세요.

결합된 아티팩트를 사용한 배치 빌드

다음 JSON 파일을 결합된 아티팩트가 있는 배치 빌드를 생성하는 파이프라인 구조의 예로 사용하세요. 일괄 빌드를 활성화하려면 CodePipeline 개체의 BatchEnabled configuration 매개변수를 로 설정합니다. true 빌드 아티팩트를 같은 위치에 결합하려면 configuration 객체의 CombineArtifacts 파라미터를 true로 설정합니다.

```
{
 "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
```

```
 }
}
```

다음은 이 파이프라인 구성과 함께 사용할 수 있는 CodeBuild buildspec 파일의 예제입니다.

```
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
```

결합 아티팩트가 배치 빌드에 활성화된 경우 출력은 하나만 허용됩니다. CodeBuild 모든 빌드의 기본 아티팩트를 하나의 ZIP 파일로 결합합니다.

JSON 파일을 생성하였으면 이제 파이프라인을 만들 수 있습니다. AWS CLI 를 사용하여 create-pipeline 명령을 실행하고 파일을 매개변수에 전달합니다. `--cli-input-json` 자세한 내용은AWS CodePipeline 사용 설명서의 파이프라인 생성(CLI)을 참조하세요.

AWS CodePipeline 여러 입력 소스 CodeBuild 및 출력 아티팩트와의 통합 샘플

AWS CodeBuild 프로젝트는 둘 이상의 입력 소스를 사용할 수 있습니다. 이에 따라 출력 아티팩트도 다수를 생성할 수 있습니다. 이 샘플은 여러 입력 소스를 AWS CodePipeline 사용하여 여러 출력 아티팩트를 만드는 빌드 프로젝트를 만드는 데 사용하는 방법을 보여줍니다. 자세한 정보는 다중 입력 소스 및 출력 아티팩트 샘플을 참조하세요.

파이프라인의 구조를 정의하는 JSON 형식의 파일을 사용한 다음, 와 함께 사용하여 파이프라인을 생성할 수 있습니다. AWS CLI 다음 JSON 파일을 입력 소스 1개 이상과 출력 아티팩트 1개 이상을 사용해 빌드를 생성하는 파이프라인 구조의 예로 사용합니다. 이번 샘플 후반에서 이 파일이 다중 입력 및

출력을 어떻게 지정하는지 확인할 수 있습니다. 자세한 내용은 사용 설명서의 CodePipeline 파이프라인 구조 참조를 참조하십시오.AWS CodePipeline

```
{
 "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
 }
}
```

위 JSON 파일에서,

- 입력 소스 중 하나는 `PrimarySource`로 지정되어야 합니다. 이 소스는 CodeBuild buildspec 파일을 찾아 실행하는 디렉터리입니다. `PrimarySource`키워드는 JSON 파일의 CodeBuild 스테이지 `configuration` 섹션에 기본 소스를 지정하는 데 사용됩니다.
- 각 입력 소스는 자체 디렉터리에 설치됩니다. 이 디렉터리는 기본 소스의 경우 기본 제공 환경 변수 `$CODEBUILD_SRC_DIR`에, 기타 모든 소스의 경우 `$CODEBUILD_SRC_DIR_yourInputArtifactName`에 저장됩니다. 위 샘플의 파이프라인에서는 2개의 입력 소스 디렉터리가 `$CODEBUILD_SRC_DIR`과 `$CODEBUILD_SRC_DIR_source2`입니다. 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.
- 파이프라인의 JSON 파일에서 지정하는 출력 아티팩트의 이름은 buildspec 파일에서 정의하는 보조 아티팩트의 이름과 일치해야 합니다. 이 파이프라인에서 사용하는 buildspec 파일은 다음과 같습니다. 자세한 정보는 buildspec 구문을 참조하세요.

```
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
```

JSON 파일을 생성하였으면 이제 파이프라인을 만들 수 있습니다. AWS CLI 를 사용하여 create-pipeline 명령을 실행하고 파일을 파라미터에 전달합니다. `--cli-input-json` 자세한 내용은AWS CodePipeline 사용 설명서의 파이프라인 생성(CLI)을 참조하세요.

샘플과 함께 사용 AWS Config CodeBuild

AWS Config 리소스 인벤토리와 해당 AWS 리소스의 구성 변경 기록을 제공합니다. AWS Config 이제 AWS CodeBuild AWS 리소스로 지원되므로 서비스에서 CodeBuild 프로젝트를 추적할 수 있습니다. 에 대한 자세한 내용은 AWS ConfigWhat is AWS Config? 를 참조하십시오. AWS Config 개발자 안내서에서

AWS Config 콘솔의 리소스 인벤토리 페이지에서 CodeBuild 리소스에 대한 다음 정보를 확인할 수 있습니다.

- CodeBuild 구성 변경 일정.
- 각 CodeBuild 프로젝트의 구성 세부 정보.
- 다른 AWS 리소스와의 관계.
- CodeBuild 프로젝트 변경 목록.

이 항목의 절차는 CodeBuild 프로젝트를 설정하고 AWS Config 조회하고 보는 방법을 보여줍니다.

주제

- 필수 조건
- 설정 AWS Config
- AWS CodeBuild 프로젝트 찾아보기
- 콘솔에서 AWS CodeBuildAWS Config 구성 세부 정보 보기

필수 조건

AWS CodeBuild 프로젝트를 생성하세요. 지침은 빌드 프로젝트 생성을 참조하세요.

설정 AWS Config

- AWS Config 설정(콘솔)
- 설정 AWS Config (AWS CLI)

> **Note**
>
> 설정을 완료한 후 AWS Config 콘솔에서 AWS CodeBuild 프로젝트를 볼 수 있을 때까지 최대 10분이 걸릴 수 있습니다.

AWS CodeBuild 프로젝트 찾아보기

1. AWS 관리 콘솔에 로그인하고 https://console.aws.amazon.com/config 에서 AWS Config 콘솔을 엽니다.
2. 리소스 인벤토리 페이지의 리소스 유형에서AWS CodeBuild 프로젝트를 선택합니다. 아래로 스크롤하여 CodeBuild 프로젝트 확인란을 선택합니다.
3. Look up(조회)을 선택합니다.
4. CodeBuild 프로젝트 목록을 추가한 후 Config 타임라인 열에서 CodeBuild 프로젝트 이름 링크를 선택합니다.

콘솔에서 AWS CodeBuildAWS Config 구성 세부 정보 보기

리소스 인벤토리 페이지에서 리소스를 조회할 때 AWS Config 타임라인을 선택하여 CodeBuild 프로젝트에 대한 세부 정보를 볼 수 있습니다. 리소스에 대한 세부 정보 페이지에서는 구성, 관계, 리소스 변경 사항 수에 대한 정보를 확인할 수 있습니다.

페이지 상단에 있는 블록을 타임라인이라고 합니다. 타임라인은 기록이 생성된 날짜와 시간을 표시합니다.

자세한 내용은 AWS Config 개발자 안내서의 AWS Config 콘솔에서 구성 세부 정보 보기를 참조하십시오.

빌드 알림 샘플은 다음과 같습니다. CodeBuild

Amazon CloudWatch Events에는 에 대한 지원이 내장되어 AWS CodeBuild있습니다. CloudWatch 이벤트는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트 스트림입니다. CloudWatch 이벤트를 사

용하면 선언적 규칙을 작성하여 관심 있는 이벤트를 취해야 할 자동화된 작업과 연결합니다. 이 샘플은 Amazon CloudWatch Events와 Amazon Simple Notification Service (Amazon SNS) 를 사용하여 빌드가 성공하거나, 실패하거나, 한 빌드 단계에서 다른 빌드 단계로 진행하거나, 이러한 이벤트가 조합될 때마다 구독자에게 빌드 알림을 보냅니다.

> **Important**
>
> 이 샘플을 실행하면 AWS 계정에 요금이 청구될 수 있습니다. 여기에는 Amazon CodeBuild 및 Amazon SNS와 관련된 AWS 리소스 및 작업에 대한 비용 CloudWatch 및 비용이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon CloudWatch 요금 및 Amazon SNS 요금을 참조하십시오.

샘플 실행

이 샘플을 실행하려면

1. Amazon SNS에서 이 샘플에 사용할 주제를 설정하고 구독한 경우 4단계로 건너뜁니다. 그렇지 않고 AWS 루트 계정 또는 관리자 사용자 대신 IAM 사용자를 사용하여 Amazon SNS를 사용하는 경우 사용자 (또는 사용자가 연결된 IAM 그룹) 에 다음 명령문 (*### ### ### ## ## ##### ## #### ## ## ## ###* 사이) 을 추가하십시오. 루트 계정을 사용하는 것은 권장되지 않습니다. AWS 이 명령문을 사용하면 Amazon SNS의 주제로의 알림 전송을 보고, 생성하고, 구독하고, 테스트할 수 있습니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 기존 정책에 입력하지 않아야 합니다.

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "sns:CreateTopic",
        "sns:GetTopicAttributes",
        "sns:List*",
        "sns:Publish",
        "sns:SetTopicAttributes",
        "sns:Subscribe"
      ],
      "Resource": "*",
      "Effect": "Allow"
```

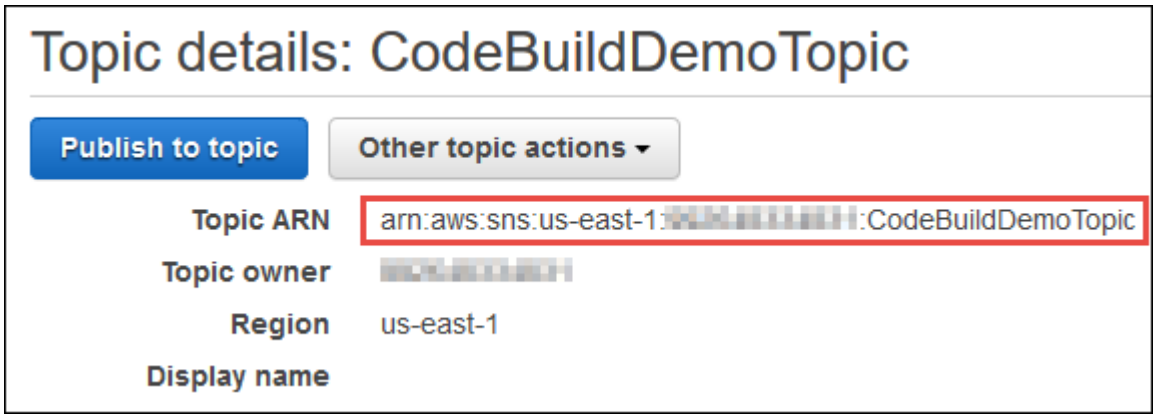
```
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다. 자세한 내용은 고객 관리형 정책 편집 또는 IAM 사용 설명서의 인라인 정책 작업(콘솔)에서 “그룹, 사용자 또는 역할에 대한 인라인 정책을 편집 또는 삭제하려면” 섹션을 참조하세요.

2. Amazon SNS에서 주제를 만들거나 식별하십시오. AWS CodeBuild CloudWatch 이벤트를 사용하여 Amazon SNS를 통해 이 주제에 대한 빌드 알림을 보냅니다.

   주제를 생성하려면 다음과 같이 합니다.

   1. https://console.aws.amazon.com/sns 에서 아마존 SNS 콘솔을 엽니다.
   2. 주제 생성을 선택합니다.
   3. 새로운 주제 생성의 주제 이름에 주제 이름(예: **CodeBuildDemoTopic**)을 입력합니다. (다른 이름을 선택하는 경우 이 샘플 전체에서 해당 이름으로 바꿉니다.)
   4. 주제 생성을 선택합니다.
   5. 주제 세부 정보: CodeBuildDemoTopic 페이지에서 주제 ARN 값을 복사합니다. 다음 단계에서 이 값을 사용합니다.

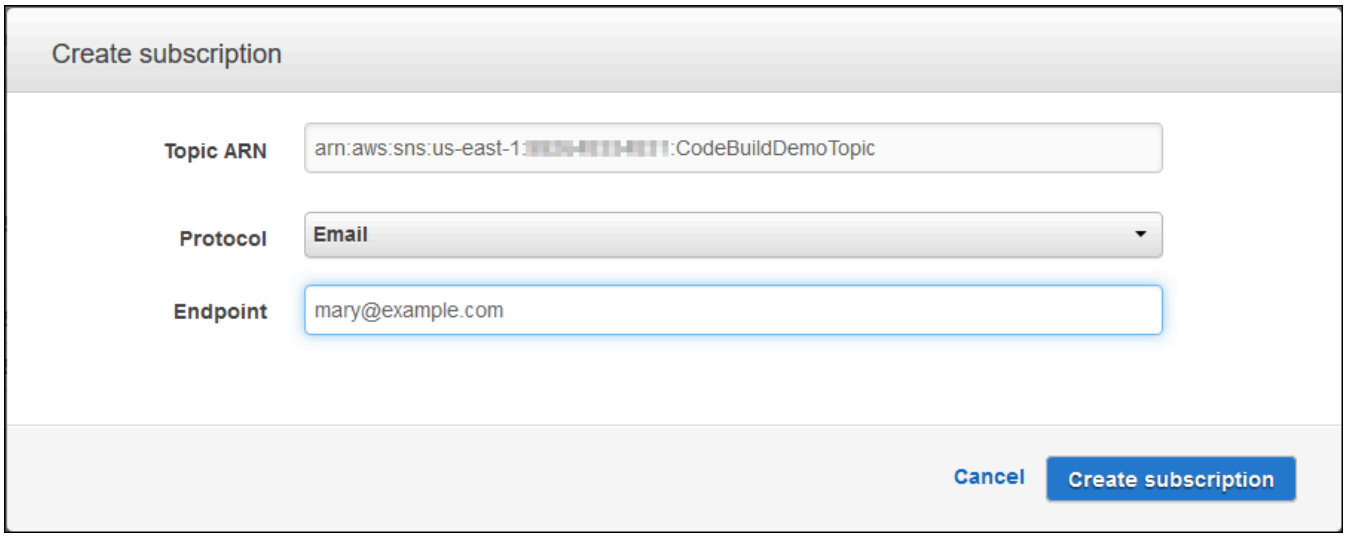


자세한 내용은 Amazon SNS 개발자 안내서의 주제 생성을 참조하세요.

3. 한 명 이상의 수신자가 주제를 구독하여 이메일 알림을 수신하게 합니다.

   수신자가 주제를 구독하게 하려면 다음과 같이 합니다.

   1. 이전 단계에서 Amazon SNS 콘솔을 연 상태에서 탐색 창에서 구독을 선택한 다음, 구독 생성을 선택합니다.
   2. 구독 생성의 주제 ARN에 이전 단계에서 복사한 주제 ARN을 붙여 넣습니다.
   3. 프로토콜에서 이메일을 선택합니다.
   4. 엔드포인트에 수신자의 전체 이메일 주소를 입력합니다.



   5. 구독 생성을 선택합니다.
   6. Amazon SNS가 수신자에게 구독 확인 이메일을 보냅니다. 수신자는 알림을 수신하려면 구독 확인 이메일에서 구독 확인을 선택해야 합니다. 수신자가 링크를 클릭한 후 구독에 성공하면 Amazon SNS가 해당 수신자의 웹 브라우저에 확인 메시지를 표시합니다.

   자세한 내용은 Amazon SNS 개발자 가이드의 주제 구독을 참조하세요.

4. AWS 루트 계정이나 관리자 사용자 대신 사용자를 사용하여 CloudWatch 이벤트를 처리하는 경우 사용자 (또는 사용자가 연결된 IAM 그룹) 에 다음 명령문 (*### ### ### ## ## ##### ## ### # ### ## ## ###* 사이) 을 추가하십시오. 루트 계정을 사용하는 것은 AWS 권장되지 않습니다. 이 명령문은 사용자가 CloudWatch Events를 사용할 수 있도록 허용하는 데 사용됩니다. 간결하게

나타내고 명령문 추가 위치를 알 수 있도록 줄임표(...)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 기존 정책에 입력하지 않아야 합니다.

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

> **Note**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다. 자세한 내용은 고객 관리형 정책 편집 또는 IAM 사용 설명서의 인라인 정책 작업(콘솔)에서 “그룹, 사용자 또는 역할에 대한 인라인 정책을 편집 또는 삭제하려면” 섹션을 참조하세요.

5. CloudWatch 이벤트에서 규칙을 생성하십시오. 이렇게 하려면 https://console.aws.amazon.com/cloudwatch 에서 CloudWatch 콘솔을 여십시오.
6. 탐색 창의 이벤트에서 규칙을 선택한 다음 규칙 생성을 선택합니다.
7. 1단계: 규칙 생성 페이지에서 이벤트 패턴 및 서비스별 이벤트와 일치시킬 이벤트 패턴 빌드가 선택된 상태여야 합니다.
8. 서비스 이름에서 CodeBuild를 선택합니다. 이벤트 유형에서 모든 이벤트가 이미 선택된 상태여야 합니다.
9. 이벤트 패턴 미리 보기에 다음 코드가 표시되어야 합니다.

```
{
  "source": [
    "aws.codebuild"
```

```
  ]
}
```

10. 편집을 선택하여 이벤트 패턴 미리 보기를 다음 두 가지 규칙 패턴 중 하나로 교체합니다.

    이 첫 번째 규칙 패턴은 빌드가 시작되거나 완료될 때 AWS CodeBuild에 지정된 빌드 프로젝트에 대해 이벤트를 트리거합니다.

```
{
  "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"
    ]
  }
}
```

    위의 규칙에서 다음과 같이 코드를 변경하세요.

    - 빌드가 시작되거나 완료될 때 이벤트를 트리거하려면 `build-status` 어레이에 표시된 모든 값을 그대로 두거나 `build-status` 어레이를 모두 제거합니다.
    - 빌드가 완료될 때만 이벤트를 트리거하려면 `build-status` 배열에서 `IN_PROGRESS`를 제거합니다.
    - 빌드가 시작될 때만 이벤트를 트리거하려면 `build-status` 배열에서 `IN_PROGRESS`를 제외한 모든 값을 제거합니다.
    - 모든 빌드 프로젝트에 대해 이벤트를 트리거하려면 `project-name` 배열을 모두 제거합니다.
    - 개별 빌드 프로젝트에 대해서만 이벤트를 트리거하려면 `project-name` 배열에 각 빌드 프로젝트의 이름을 지정합니다.

이 두 번째 규칙 패턴은 AWS CodeBuild에 지정된 빌드 프로젝트에 대해 빌드가 한 빌드 단계에서 다른 빌드 단계로 이동할 때마다 이벤트를 트리거합니다.

```
{
  "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"
    ]
  }
}
```

위의 규칙에서 다음과 같이 코드를 변경하세요.

- 모든 빌드 단계 변경(각 빌드에 대해 최대 9개의 알림을 보낼 수 있음)에 대해 이벤트를 트리거하려면 `completed-phase` 어레이에 표시된 모든 값을 그대로 두거나 `completed-phase` 어레이를 모두 제거합니다.
- 개별 빌드 단계 변경에 대해서만 이벤트를 트리거하려면 이벤트를 트리거하지 않으려는 `completed-phase` 배열의 각 빌드 단계 이름을 제거합니다.
- 모든 빌드 단계 상태 변경에 대한 이벤트를 트리거하려면 `completed-phase-status` 배열에 표시된 모든 값을 그대로 두거나 `completed-phase-status` 배열을 모두 제거합니다.
- 개별 빌드 단계 상태 변경에 대해서만 이벤트를 트리거하려면 이벤트를 트리거하지 않으려는 `completed-phase-status` 배열의 각 빌드 단계 이름을 제거합니다.
- 모든 빌드 프로젝트에 대한 이벤트를 트리거하려면 `project-name` 배열을 제거합니다.
- 개별 빌드 프로젝트에 대한 이벤트만 트리거하려면 `project-name` 배열에 각 빌드 프로젝트의 이름을 지정합니다.

이벤트 패턴에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 이벤트 패턴을 참조하십시오.

이벤트 패턴을 사용한 필터링에 대한 자세한 내용은 Amazon EventBridge User Guide의 이벤트 패턴을 사용한 콘텐츠 기반 필터링을 참조하십시오.

> **Note**
>
> 빌드 상태 변경 및 빌드 단계 변경에 대한 이벤트를 트리거하려는 경우, 빌드 상태 변경에 대한 규칙 및 빌드 단계 변경을 위한 규칙이라는 두 가지 별도의 규칙을 생성해야 합니다. 두 가지 규칙을 단일 규칙으로 결합하려고 하면 결합된 해당 규칙으로 인해 예기치 않은 결과가 발생하거나 작업이 모두 중단될 수 있습니다.

코드 교체를 완료하면 저장을 선택합니다.

11. 대상(Targets)에서 대상 추가(Add target)를 선택합니다.
12. 대상 목록에서 SNS 주제를 선택합니다.
13. 주제에서 이전에 식별했거나 생성한 주제를 선택합니다.
14. 입력 구성을 확장한 후 입력 변환기를 선택합니다.
15. 입력 경로 상자에 다음 입력 경로 중 하나를 입력합니다.

CodeBuild Build State Change의 detail-type 값을 사용하는 규칙에 대해 다음을 입력합니다.

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

CodeBuild Build Phase Change의 detail-type 값을 사용하는 규칙에 대해 다음을 입력합니다.

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

다른 유형의 정보를 보려면 빌드 알림 입력 형식 참조 섹션을 참조하세요.

16. 입력 템플릿 상자에 다음 입력 템플릿 중 하나를 입력합니다.

    CodeBuild Build State Change의 detail-type 값을 사용하는 규칙에 대해 다음을 입력합니다.

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

    CodeBuild Build Phase Change의 detail-type 값을 사용하는 규칙에 대해 다음을 입력합니다.

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

17. 세부 정보 구성을 선택합니다.
18. 2단계: 규칙 세부 정보 구성 페이지에 이름 및 선택적인 설명을 입력합니다. 상태에 대해 사용을 선택한 상태로 둡니다.
19. Create rule을 선택합니다.
20. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 빌드 정보를 확인하십시오.
21. 이제 빌드 알림이 성공적으로 CodeBuild 전송되고 있는지 확인하세요. 예를 들어 빌드 알림 이메일이 현재 받은 편지함에 있는지 확인합니다.

규칙의 동작을 변경하려면 CloudWatch 콘솔에서 변경하려는 규칙을 선택하고 작업을 선택한 다음 편집을 선택합니다. 규칙을 변경하고 구성 세부 정보를 선택한 후 규칙 업데이트를 선택합니다.

규칙을 사용하여 빌드 알림을 보내는 것을 중단하려면 CloudWatch 콘솔에서 사용을 중지하려는 규칙을 선택하고 작업을 선택한 다음 비활성화를 선택합니다.

규칙을 완전히 삭제하려면 CloudWatch 콘솔에서 삭제하려는 규칙을 선택하고 작업을 선택한 다음 삭제를 선택합니다.

관련 리소스

- 시작하는 방법에 대한 자세한 내용은 AWS CodeBuild을 참조하십시오콘솔을 사용하여 AWS CodeBuild 시작하기.
- 의 문제 해결에 대한 자세한 내용은 CodeBuild 을 참조하십시오문제 해결 AWS CodeBuild.
- 할당량에 대한 자세한 내용은 CodeBuild 을 참조하십시오. AWS CodeBuild에 대한 할당량

빌드 알림 입력 형식 참조

CloudWatch 알림을 JSON 형식으로 전달합니다.

빌드 상태 변경 알림은 다음 형식을 사용합니다.

```
{
  "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"
  }
}
```

빌드 단계 변경 알림은 다음 형식을 사용합니다.

```
{
  "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"
  }
}
```

다음을 사용하여 배지 샘플을 빌드하십시오. CodeBuild

AWS CodeBuild 이제 프로젝트의 최신 빌드 상태를 표시하는 내장 가능하고 동적으로 생성되는 이미지 (배지) 를 제공하는 빌드 배지를 사용할 수 있습니다. 이 이미지는 프로젝트용으로 생성된 공개적으로 사용 가능한 URL을 통해 액세스할 수 있습니다. CodeBuild 이렇게 하면 누구나 CodeBuild 프로젝트 상태를 볼 수 있습니다. 빌드 배지에는 보안 정보가 포함되어 있지 않으므로 인증이 필요하지 않습니다.

활성화된 빌드 배지를 사용하여 빌드 프로젝트 생성(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.

2. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 만들기를 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 AWS 계정별로 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스의 소스 공급자에서, 소스 코드 공급자 유형을 선택한 다음, 다음 중 하나를 수행합니다.

> **Note**
>
> CodeBuild Amazon S3 소스 공급자를 통한 빌드 배지는 지원하지 않습니다. 아티팩트 전송에 Amazon S3를 AWS CodePipeline 사용하기 때문에 에서 생성된 파이프라인의 일부인 빌드 프로젝트에는 빌드 배지가 지원되지 않습니다. CodePipeline

- 선택한 CodeCommit경우 리포지토리에서 리포지토리 이름을 선택합니다. 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다.
- 선택한 GitHub경우 지침에 따라 연결 (또는 재연결) 하십시오 GitHub. 응용 프로그램 GitHub 승인 페이지에서 조직 액세스에 대해 AWS CodeBuild 액세스하려는 각 저장소 옆에 있는 액세스 요청을 선택합니다. Authorize application(애플리케이션 권한 부여)을 선택한 후 AWS CodeBuild 콘솔로 돌아가서 리포지토리에서 소스 코드를 포함하는 리포지토리의 이름을 선택합니다. 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다.
- [Bitbucket]을 선택했다면, Bitbucket과 연결(다시 연결)하는 지침을 따르십시오. [Confirm access to your account] 페이지의 [Organization access]에서 [Grant access]를 선택합니다. 액세스 권한 부여를 선택한 후 AWS CodeBuild 콘솔로 돌아가서 리포지토리에 대해 소스 코드가 들어 있는 리포지토리의 이름을 선택합니다. 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다.

> **Important**
>
> 프로젝트 소스를 업데이트하면 프로젝트 빌드 배지의 정확성에 영향을 미칠 수 있습니다.

5. 환경에서 다음과 같이 합니다.

   [Environment image]에서 다음 중 하나를 수행합니다.

- 관리되는 AWS CodeBuild Docker 이미지를 사용하려면 관리 이미지를 선택한 다음 운영 체제, 런타임, 이미지 및 이미지 버전 중에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 계정의 Docker 이미지를 선택하십시오. AWS
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

6. 서비스 역할에서 다음 중 하나를 수행합니다.

- 서비스 역할이 없는 경우 새 CodeBuild 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 만들거나 업데이트할 때 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

7. Buildspec에서 다음 중 하나를 수행합니다.

- buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
- 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

자세한 내용은 buildspec 참조을 참조하세요.

8. 결과물의 유형에서 다음 중 하나를 수행합니다.

   - 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
   - S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
     - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
     - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
     - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.
9. 추가 구성을 확장하고 적절한 옵션을 선택합니다.
10. 빌드 프로젝트 생성을 선택합니다. 검토 페이지에서 빌드 시작을 선택하여 빌드를 실행합니다.

활성화된 빌드 배지를 사용하여 빌드 프로젝트 생성(CLI)

빌드 프로젝트 생성에 대한 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 섹션을 참조하십시오. AWS CodeBuild 프로젝트에 빌드 배지를 포함하려면 *`badgeEnabled`* 값을 `true`로 지정해야 합니다.

AWS CodeBuild 빌드 배지에 액세스하세요.

AWS CodeBuild 콘솔 또는 를 사용하여 빌드 AWS CLI 배지에 액세스할 수 있습니다.

- CodeBuild 콘솔의 빌드 프로젝트 목록에 있는 이름 열에서 빌드 프로젝트에 해당하는 링크를 선택합니다. 빌드 프로젝트: ***`project-name`*** 페이지의 구성에서 배지 URL 복사를 선택합니다. 자세한 정보는 빌드 프로젝트 세부 정보 보기(콘솔)을 참조하세요.
- AWS CLI에서 `batch-get-projects` 명령을 실행합니다. 빌드 배지 URL은 출력의 프로젝트 환경 세부 정보 섹션에 포함됩니다. 자세한 정보는 빌드 프로젝트 세부 정보 보기(AWS CLI)을 참조하세요.

빌드 배지 요청 URL은 일반적인 기본 분기로 생성되지만, 빌드를 실행하는 데 사용한 소스 리포지토리의 어떤 분기도 지정할 수 있습니다. 예:

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

branch 파라미터를 배지 URL의 tag 파라미터를 대체하여 소스 리포지토리의 태그를 지정할 수도 있습니다. 예:

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

CodeBuild 빌드 배지를 게시하세요.

마크다운 이미지의 빌드 배지 URL을 사용하여 마크다운 파일에 최신 빌드의 상태를 표시할 수 있습니다. 이는 소스 리포지토리의 readme.md 파일에 최신 빌드의 상태를 표시하는 데 유용합니다 (예: 또는). GitHub CodeCommit 예:

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

CodeBuild 배지 상태

- PASSING 특정 분기의 최신 빌드가 전달되었습니다.
- FAILING 특정 분기의 최신 빌드가 시간 초과, 실패, 오류 또는 중지되었습니다.
- IN_PROGRESS 특정 분기의 최신 빌드가 진행 중입니다.
- UNKNOWN 프로젝트가 아직 특정 분기 또는 전부에 대한 빌드를 실행하지 않았습니다. 또한 배지 빌드 기능이 비활성화되었을 수 있습니다.

샘플을 CodeBuild 사용하여 테스트 보고서 만들기 AWS CLI

buildspec 파일에 지정한 테스트는 빌드 중에 실행됩니다. 이 샘플은 를 사용하여 테스트를 빌드에 AWS CLI 통합하는 방법을 보여줍니다 CodeBuild. JUnit을 사용하여 단위 테스트를 만들거나, 다른 도구를 사용하여 구성 테스트를 만들 수 있습니다. 그런 다음 테스트 결과를 평가하여 문제를 해결하거나 애플리케이션을 최적화할 수 있습니다.

CodeBuild API 또는 AWS CodeBuild 콘솔을 사용하여 테스트 결과에 액세스할 수 있습니다. 이 샘플에서는 테스트 결과를 S3 버킷으로 내보내도록 보고서를 구성하는 방법을 보여줍니다.

주제

- 보고서 실행 및 결과 보기

필수 조건

- 테스트 케이스를 만듭니다. 이 샘플은 샘플 테스트 보고서에 포함할 테스트 케이스가 있다는 것을 전제로 작성된 것입니다. buildspec 파일에서 테스트 파일의 위치를 지정합니다.

  지원되는 테스트 보고서 파일 형식은 다음과 같습니다.

  - Cucumber JSON(.json)
  - JUnit XML(.xml)
  - NUnit XML(.xml)
  - NUnit3 XML(.xml)
  - TestNG XML(.xml)
  - Visual Studio TRX(.trx)
  - 비주얼 스튜디오 TRX XML (.xml)

  이러한 형식 중 하나로 보고서 파일을 만들 수 있는 테스트 프레임워크로 테스트 케이스를 만듭니다(예: Surefire JUnit plugin, TestNG, Cucumber).
- S3 버킷을 만들고 이름을 기록해 둡니다. 자세한 내용을 알아보려면 Amazon S3 사용 설명서의 S3 버킷을 생성하는 방법을 참조하세요.
- IAM 역할을 생성하고 해당 ARN을 기록해 둡니다. 빌드 프로젝트를 만들 때 ARN이 필요합니다.
- 역할에 다음 권한이 없는 경우 권한을 추가합니다.

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

  자세한 정보는 테스트 보고 작업에 대한 권한을 참조하세요.

보고서 그룹 만들기

1. `CreateReportGroupInput.json`이라는 이름의 파일을 만듭니다.
2. S3 버킷에 테스트 결과를 내보낼 폴더를 만듭니다.
3. 다음을 `CreateReportGroupInput.json`에 복사합니다. *<bucket-name>*는 S3 버킷의 이름을 사용합니다. *<path-to-folder>*은 S3 버킷의 폴더 경로를 입력합니다.

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

4. `CreateReportGroupInput.json`을 포함하는 디렉터리에서 다음 명령을 실행합니다.

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

출력은 다음과 같습니다. `reportGroup`의 ARN을 기록해 둡니다. 이 보고서 그룹을 사용하는 프로젝트를 만들 때 사용합니다.

```
{
  "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
  }
}
```

보고서 그룹을 사용하여 프로젝트 구성

보고서를 실행하려면 먼저 보고서 그룹으로 구성된 CodeBuild 빌드 프로젝트를 만들어야 합니다. 보고서 그룹에 지정된 테스트 케이스는 빌드를 실행할 때 실행됩니다.

1. 이름이 `buildspec.yml`인 buildspec 파일을 만듭니다
2. 다음 YAML을 `buildspec.yml` 파일의 템플릿으로 사용합니다. 테스트를 실행하는 명령이 포함되어야 합니다. `reports` 섹션에서는 테스트 케이스 결과가 포함된 파일을 지정합니다. 이 파일에는 액세스할 수 있는 테스트 결과가 저장됩니다 CodeBuild. 작성되고 30일 후에 만료됩니다. 이러한 파일은 S3 버킷으로 내보내는 원시 테스트 케이스 결과 파일과 다릅니다.

```
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**
>
> 기존 보고서 그룹의 ARN 대신, 생성되지 않은 보고서 그룹의 이름을 지정할 수도 있습니다. ARN 대신 이름을 지정하는 경우 빌드를 실행할 때 보고서 그룹이 CodeBuild 생성됩니다. 이름에는 프로젝트 이름과 buildspec 파일에 지정한 이름이 `project-name-`

`report-group-name` 형식으로 포함되어 있습니다. 자세한 내용은 테스트 보고서 작성 및 보고서 그룹 이름 지정 섹션을 참조하세요.

3. `project.json`이라는 이름의 파일을 만듭니다. 이 파일에는 create-project 명령에 대한 입력이 들어 있습니다.
4. 다음 JSON을 `project.json`에 복사합니다. `source`는 소스 파일이 들어있는 저장소의 유형과 위치를 입력합니다. `serviceRole`은 사용 중인 역할의 ARN을 지정합니다.

```
{
  "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. `project.json`을 포함하는 디렉터리에서 다음 명령을 실행합니다. 이렇게 하면 이름이 `test-project`인 프로젝트가 생성됩니다.

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

보고서 실행 및 결과 보기

이 섹션에서는 이전에 만든 프로젝트의 빌드를 실행합니다. 빌드 프로세스 중에 테스트 사례의 결과가 포함된 보고서를 CodeBuild 생성합니다. 보고서는 지정한 보고서 그룹에 포함되어 있습니다.

1. 빌드를 시작하려면 다음 명령을 실행합니다. `test-report-project`는 위에서 만든 빌드 프로젝트의 이름입니다. 출력에 나타나는 빌드 ID를 기록해 둡니다.

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

2. 다음 명령을 실행하여 보고서의 ARN을 포함하여 빌드에 대한 정보를 가져옵니다. *<build-id>*은 빌드 ID를 지정합니다. 출력의 `reportArns` 속성에 보고서 ARN을 기록해 둡니다.

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

3. 보고서에 대한 세부 정보를 가져오려면 다음 명령을 실행합니다. *<report-arn>*은 보고서 ARN을 지정합니다.

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

출력은 다음과 같습니다. 이 샘플 출력은 성공했거나, 실패했거나, 건너뛰었거나, 오류가 발생했거나, 알 수 없는 상태를 반환하는 테스트 수를 보여 줍니다.

```
{
  "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. 보고서의 테스트 케이스에 대한 정보를 나열하려면 다음 명령을 실행합니다. *<report-arn>*은 보고서의 ARN을 지정합니다. 선택 사항인 `--filter` 매개변수의 경우 하나의 상태 결과(`SUCCEEDED`, `FAILED`, `SKIPPED`, `ERROR` 또는 `UNKNOWN`)를 지정할 수 있습니다.

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

출력은 다음과 같습니다.

```
{
  "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>"
    }
  ]
}
```

Docker 샘플에 대한 CodeBuild

주제

- Docker의 사용자 지정 이미지 샘플은 다음과 같습니다. CodeBuild
- Docker 이미지를 Amazon Elastic 컨테이너 레지스트리 이미지 리포지토리 샘플에 게시하십시오. CodeBuild
- 샘플이 포함된 AWS Secrets Manager 사설 레지스트리 CodeBuild

Docker의 사용자 지정 이미지 샘플은 다음과 같습니다. CodeBuild

이 샘플은 사용자 지정 Docker 빌드 이미지 (Docker `docker:dind` Hub에서) 를 사용하여 Docker 이미지를 AWS CodeBuild 빌드하고 실행합니다.

Docker 지원에서 제공하는 CodeBuild 빌드 이미지를 대신 사용하여 Docker 이미지를 빌드하는 방법을 알아보려면 당사를 참조하십시오. Amazon ECR 이미지 리포지토리에 Docker 이미지 게시 샘플

> **Important**
>
> 이 샘플을 실행하면 계정에 요금이 청구될 수 AWS 있습니다. 여기에는 Amazon S3 AWS KMS, CloudWatch 로그와 관련된 AWS 리소스 및 작업에 대한 가능한 요금이 포함됩니다. CodeBuild 자세한 내용은 CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금 및 Amazon CloudWatch 요금을 참조하십시오.

주제

- 샘플 실행
- 디렉터리 구조
- 파일

- 관련 리소스

샘플 실행

이 샘플을 실행하려면

1. 이 항목의 “디렉터리 구조” 및 “파일” 섹션에 설명된 대로 파일을 만든 다음 S3 입력 버킷 또는 AWS CodeCommit GitHub, 또는 Bitbucket 리포지토리에 업로드합니다.

   > **Important**
   >
   > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
   > S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

2. 빌드 프로젝트를 만들고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.

   를 사용하여 빌드 프로젝트를 AWS CLI 만드는 경우 JSON 형식의 `create-project` 명령 입력이 이와 비슷해 보일 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "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**
>
> 기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.

3. 빌드 결과를 확인하려면 빌드 로그에서 `Hello, World!` 문자열에 대해 찾아보십시오. 자세한 정보는 빌드 세부 정보 보기을 참조하세요.

디렉터리 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

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

파일

이 샘플에 사용되는 운영 체제의 기본 이미지는 Ubuntu입니다. 샘플은 이러한 파일을 사용합니다.

`buildspec.yml`(*(root directory name)*에 있음)

```
version: 0.2

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

`Dockerfile`(*(root directory name)*에 있음)

```
FROM maven:3.3.9-jdk-8
```

```
RUN echo "Hello World"
```

관련 리소스

- 시작하는 방법에 대한 자세한 내용은 을 참조하십시오. AWS CodeBuild콘솔을 사용하여 AWS CodeBuild 시작하기
- 의 문제 해결에 대한 자세한 내용은 CodeBuild 을 참조하십시오문제 해결 AWS CodeBuild.
- 할당량에 대한 자세한 내용은 CodeBuild 을 참조하십시오. AWS CodeBuild에 대한 할당량

Docker 이미지를 Amazon Elastic 컨테이너 레지스트리 이미지 리포지토리 샘플에 게시하십시오. CodeBuild

이 샘플은 빌드 출력으로 도커 이미지를 생산한 다음 도커 이미지를 Amazon Elastic Container Registry(Amazon ECR) 이미지 리포지토리에 푸시합니다. 이 샘플을 응용하여 도커 이미지를 Docker Hub에 푸시할 수도 있습니다. 자세한 정보는 샘플을 응용하여 이미지를 도커 허브에 푸시을 참조하세요.

사용자 지정 도커 빌드 이미지(도커 허브의 `docker:dind`)를 사용하여 도커 이미지를 빌드하는 방법을 알아보려면 도커 사용자 지정 이미지 샘플 단원을 참조하십시오.

이 샘플은 `golang:1.12`를 참조하여 테스트됩니다.

이 샘플에서는 도커 이미지를 빌드 출력으로 생성하는 새로운 다단계 Docker 빌드 기능을 사용합니다. 그런 다음, 도커 이미지를 Amazon ECR 이미지 리포지토리로 푸시합니다. 다단계 도커 이미지 빌드는 최종 도커 이미지의 크기를 줄이는 데 도움이 됩니다. 자세한 내용은 Use multi-stage builds with Docker를 참조하십시오.

> **Important**
>
> 이 샘플을 실행하면 계정에 요금이 청구될 수 AWS 있습니다. 여기에는 Amazon S3 AWS KMS, CloudWatch 로그 및 Amazon ECR과 관련된 AWS 리소스 및 작업에 대한 가능한 요금이 포함됩니다. AWS CodeBuild 자세한 내용은 요금, Amazon S3 CodeBuild 요금, 요금, Amazon AWS Key Management Service 요금 및 Amazon CloudWatch Elastic 컨테이너 레지스트리 요금을 참조하십시오.

주제

- 샘플 실행
- 디렉터리 구조
- 파일
- 샘플을 응용하여 이미지를 도커 허브에 푸시
- 관련 리소스

샘플 실행

이 샘플을 실행하려면

1. Amazon ECR에 사용할 이미지 리포지토리가 이미 있으면 3단계로 이동하세요. 그렇지 않으면 Amazon ECR을 사용할 때 AWS 루트 계정이나 관리자 사용자 대신 사용자를 사용하는 경우 이 명령문 (### 여기에 명령문 추가 *## #### ## #### ### ## ## ###* 사이) 을 사용자 (또는 사용자와 관련된 IAM 그룹) 에 추가하십시오. AWS 루트 계정을 사용하는 것은 권장되지 않습니다. 이 명령문을 사용하면 Docker 이미지를 저장하기 위한 Amazon ECR 리포지토리를 생성할 수 있습니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 정책에 입력하지 않아야 합니다. 자세한 내용은 사용 설명서의 AWS Management Console을 사용한 인라인 정책 작업을 참조하세요.

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

> **Note**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다.

2. Amazon ECR에서 이미지 리포지토리를 생성합니다. 빌드 환경을 생성하고 빌드를 실행하는 AWS 지역과 동일한 지역에 리포지토리를 생성해야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 리포지토리 생성을 참조하세요. 이 리포지토리의 이름은 이 절차의 뒷부분에서 지정하는 리포지토리 이름(IMAGE_REPO_NAME 환경 변수에 의해 표시됨)과 일치해야 합니다. Amazon ECR 리포지토리 정책이 CodeBuild 서비스 IAM 역할에 이미지 푸시 액세스를 허용하는지 확인하십시오.

3. 서비스 역할에 연결한 정책에 이 명령문 (*### ### ### #### #### ### ### ### #### ## ### ###* 사이) 을 추가합니다. AWS CodeBuild 이 명령문을 사용하면 Docker 이미지를 Amazon ECR 리포지토리에 CodeBuild 업로드할 수 있습니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 정책에 입력하지 않아야 합니다.

```
{
  "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**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다.

4. 이 항목의 “디렉터리 구조” 및 “파일” 섹션에 설명된 대로 파일을 만든 다음 S3 입력 버킷 또는, 또는 Bitbucket 리포지토리에 업로드합니다. AWS CodeCommit GitHub 자세한 내용은AWS CodePipeline 사용 설명서의 이미지 정의 파일 참조를 참조하세요.

> **Important**
>
> *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
> S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

5. 빌드 프로젝트를 만들고, 빌드를 실행하고, 빌드 정보를 확인합니다.

   콘솔을 사용하여 프로젝트를 생성할 경우:

   a. [Operating system]에서 [Ubuntu]를 선택합니다.

   b. 실행 시간에서 표준을 선택합니다.

   c. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.

   d. 다음 환경 변수를 추가합니다.

      - *region-ID* 값이 있는 AWS_DEFAULT_REGION
      - *account-ID* 값이 있는 AWS_ACCOUNT_ID
      - 최신 값이 있는 IMAGE_TAG
      - *Amazon-ECR-repo-name* 값이 있는 IMAGE_REPO_NAME

   를 사용하여 빌드 프로젝트를 AWS CLI 만드는 경우 JSON 형식의 `create-project` 명령 입력이 다음과 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "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. Docker 이미지를 리포지토리로 CodeBuild 성공적으로 푸시했는지 확인하세요.

   1. Amazon ECR 콘솔(https://console.aws.amazon.com/ecr/)을 엽니다.
   2. 리포지토리 이름을 선택합니다. Image tag(이미지 태그) 열에 이미지가 있어야 합니다.

디렉터리 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

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

파일

이 샘플은 다음 파일을 사용합니다.

`buildspec.yml`(*(root directory name)*에 있음)

```
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(*(root directory name)*에 있음)

```
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 사용자 지정 Docker 이미지의 ENTRYPOINT 경우 를 재정의합니다.

샘플을 응용하여 이미지를 도커 허브에 푸시

도커 이미지를 Amazon ECR이 아닌 Docker Hub에 푸시하려면 이 샘플 코드를 편집합니다.

> **Note**
>
> 17.06 이전의 도커 버전을 사용하는 경우 `--no-include-email` 옵션을 제거합니다.

1. `buildspec.yml` 파일 내의 다음 Amazon ECR 특정 코드 행을 바꿉니다.

```
...
  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
...
```

   다음 도커 허브 관련 코드 행으로 바꿉니다.

```
...
  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. 편집한 코드를 S3 입력 버킷 AWS CodeCommit, GitHub, 또는 Bitbucket 리포지토리에 업로드합니다.

> **Important**
>
> *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
> S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

3. `create-project` 명령에 대한 JSON 형식 입력에서 다음 코드 행을 바꿉니다.

```
...
    "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"
      }
```

```
    ]
...
```

다음 코드 행으로 바꿉니다.

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

4. 빌드 환경을 만들고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.
5. Docker 이미지를 리포지토리로 AWS CodeBuild 성공적으로 푸시했는지 확인하세요. Docker Hub에 로그인하고, 리포지토리로 이동한 다음 [Tags] 탭을 선택합니다. `latest` 태그에 가장 최근의 [Last Updated] 값이 포함되어 있어야 합니다.

관련 리소스

- 시작하는 방법에 대한 자세한 내용은 AWS CodeBuild을 참조하십시오콘솔을 사용하여 AWS CodeBuild 시작하기.
- 의 문제 해결에 대한 자세한 내용은 CodeBuild 을 참조하십시오문제 해결 AWS CodeBuild.
- 할당량에 대한 자세한 내용은 CodeBuild 을 참조하십시오. AWS CodeBuild에 대한 할당량

샘플이 포함된 AWS Secrets Manager 사설 레지스트리 CodeBuild

이 샘플은 프라이빗 레지스트리에 저장된 Docker 이미지를 AWS CodeBuild 런타임 환경으로 사용하는 방법을 보여줍니다. 프라이빗 레지스트리의 보안 인증은 AWS Secrets Manager에 저장됩니다. 모든 사설 레지스트리가 함께 작동합니다. CodeBuild 이 샘플은 Docker Hub를 사용합니다.

> **Note**
>
> 보안 암호는 작업에 표시되며 파일에 기록될 때 가려지지 않습니다.

프라이빗 레지스트리 샘플 요구 사항

에서 사설 레지스트리를 AWS CodeBuild사용하려면 다음이 있어야 합니다.

- Docker Hub 보안 인증을 저장하는 Secrets Manager 보안 암호. 보안 인증은 프라이빗 리포지토리 액세스에 사용합니다.

  > **Note**
  >
  > 생성한 보안 암호에 대해 요금이 청구됩니다.

- 프라이빗 리포지토리 또는 계정
- Secrets Manager 시크릿에 대한 액세스 권한을 부여하는 CodeBuild 서비스 역할 IAM 정책입니다.

다음 단계에 따라 이러한 리소스를 생성한 다음 프라이빗 레지스트리에 저장된 Docker 이미지를 사용하여 CodeBuild 빌드 프로젝트를 생성하세요.

사설 레지스트리를 사용하여 CodeBuild 프로젝트를 생성하세요.

1. 프리 프라이빗 리포지토리 생성 방법에 대한 자세한 내용은 Docker Hub의 리포지토리를 참조하십시오. 또한 터미널에서 다음 명령을 실행하여 이미지를 가져오고, 이미지의 ID를 확보하고, 새 리포지토리로 푸시할 수 있습니다.

   ```
   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. AWS Secrets Manager 사용 설명서의 AWS Secrets Manager 시크릿 만들기의 단계를 따르세요.

   a. 3단계의 보안 암호 유형 선택에서 다른 유형의 보안 암호를 선택합니다.

   b. 키/값 페어에서 Docker Hub 사용자 이름에 대한 키-값 페어와 Docker Hub 암호에 대한 키-값 페어를 하나씩 생성합니다.

c. 계속해서 AWS Secrets Manager 시크릿 만들기의 단계를 따르세요.

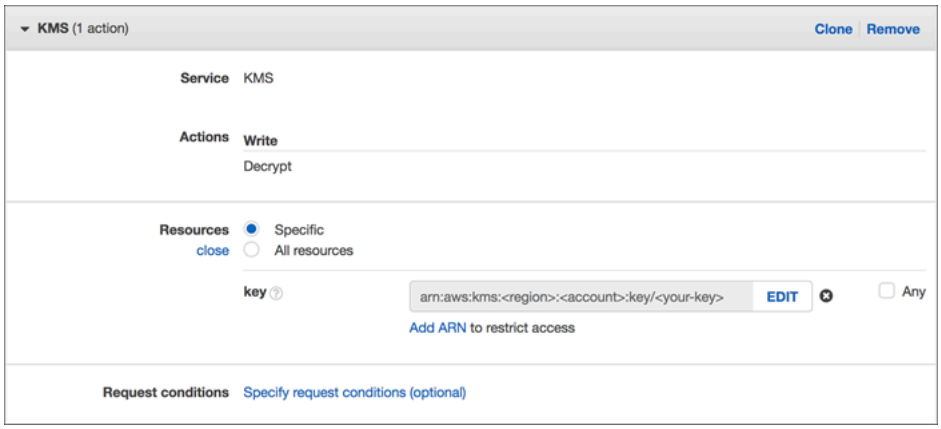
d. 키는 Docker Hub 보안 인증에 해당하므로 5단계의 자동 교체 구성 페이지에서 이 옵션을 끕니다.

e. AWS Secrets Manager 시크릿 만들기의 단계를 완료하세요.

자세한 내용은 AWS Secrets Manager란 무엇입니까?를 참조하십시오.

3. 콘솔에서 AWS CodeBuild 프로젝트를 생성할 때 필요한 권한을 CodeBuild 첨부합니다. 이외의 AWS KMS `DefaultEncryptionKey` 키를 사용하는 경우 해당 키를 서비스 역할에 추가해야 합니다. 자세한 내용은 IAM 사용 설명서의 역할 수정(콘솔)을 참조하세요.

   Secrets Manager에서 서비스 역할이 작동하려면 최소한 `secretsmanager:GetSecretValue` 권한이 있어야 합니다.



4. 콘솔을 사용하여 환경이 프라이빗 레지스트리에 저장되는 프로젝트를 생성하려면 프로젝트를 생성하면서 다음을 수행합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔)을 참조하세요.

> **Note**
>
> VPC에 프라이빗 레지스트리가 있는 경우 퍼블릭 인터넷 액세스가 가능해야 합니다. CodeBuild VPC의 프라이빗 IP 주소에서는 이미지를 가져올 수 없습니다.

a. 환경 이미지에서 사용자 지정 이미지를 선택합니다.

b. 환경 유형에서 Linux 또는 Windows를 선택합니다.

c. 이미지 레지스트리의 경우 다른 레지스트리를 선택합니다.

d. 외부 레지스트리 URL에 이미지 위치를 입력하고 레지스트리 보안 인증 - 선택 사항에 Secrets Manager 보안 인증의 ARN 또는 이름을 입력합니다.

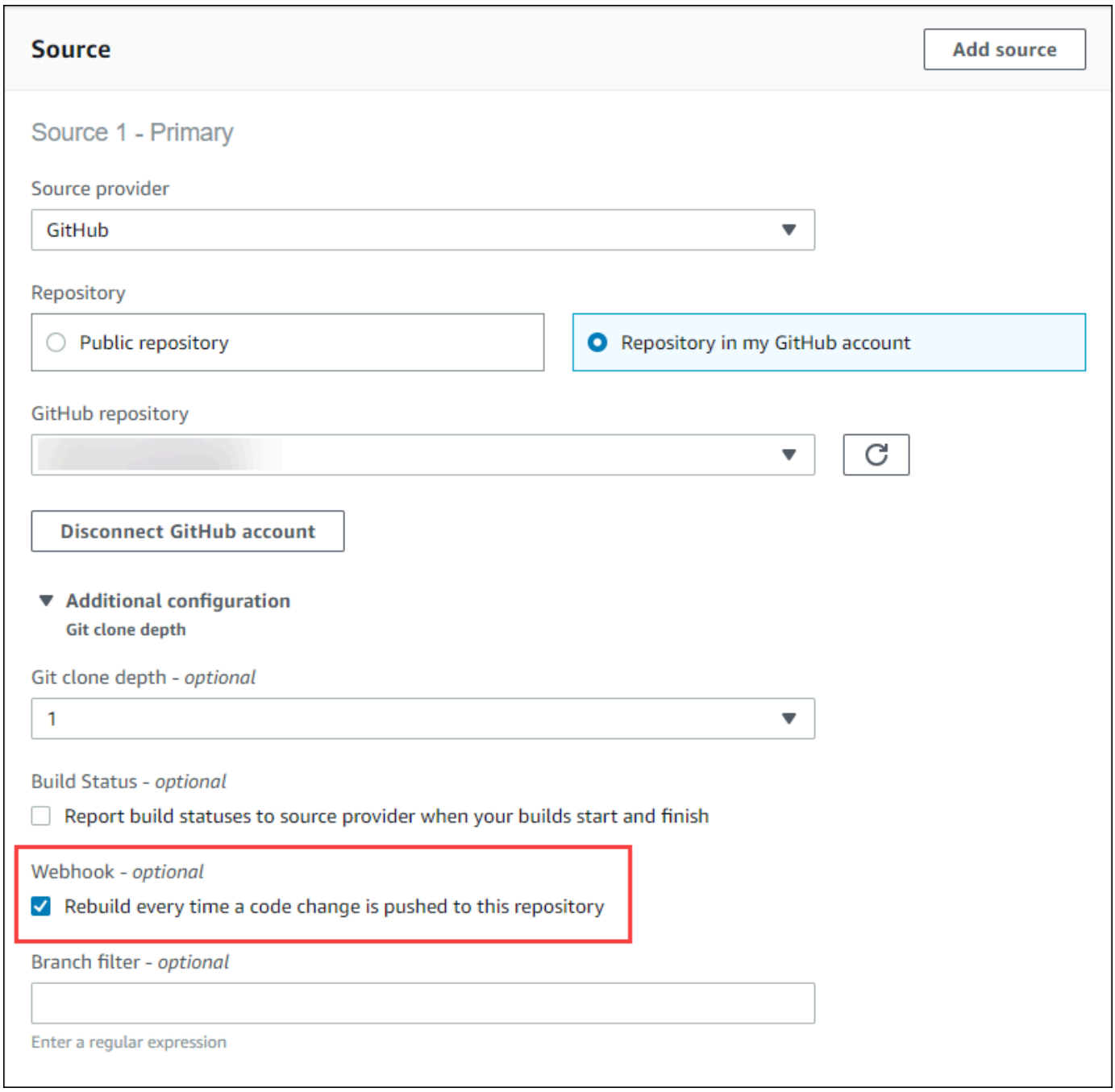
> **Note**
>
> 현재 리전에 보안 인증이 없을 경우 ARN을 사용해야 합니다. 다른 리전에 있는 보안 인증의 이름은 사용할 수 없습니다.

빌드 출력을 S3 버킷에서 호스팅하여 정적 웹사이트 생성

빌드의 아티팩트 암호화를 비활성화할 수 있습니다. 이렇게 하면 웹 사이트를 호스팅하도록 구성된 위치에 아티팩트를 게시할 수 있습니다. (암호화된 아티팩트는 게시할 수 없습니다.) 이 샘플에서는 Webhook를 사용해 빌드를 트리거한 후 웹사이트를 구성하는 S3 버킷에 아티팩트를 게시하는 방법에 대해서 설명합니다.

1. 정적 웹사이트 설정의 지침을 따라 웹사이트처럼 작동하도록 S3 버킷을 구성합니다.
2. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
3. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 만들기를 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
4. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 AWS 계정별로 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
5. [소스] 에서 [소스 공급자] 를 선택합니다 GitHub. 지침에 따라 연결 (또는 재연결) 한 다음 Authorize (권한 부여) 를 선택합니다. GitHub

   Webhook에서 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다. 이 확인란은 내 GitHub 계정의 리포지토리를 선택한 경우에만 선택할 수 있습니다.



6. 환경에서 다음과 같이 합니다.

   [Environment image]에서 다음 중 하나를 수행합니다.

   - 관리되는 AWS CodeBuild Docker 이미지를 사용하려면 관리 이미지를 선택한 다음 운영 체제, 런타임, 이미지 및 이미지 버전 중에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.

- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 계정의 Docker 이미지를 선택하십시오. AWS
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

7. 서비스 역할에서 다음 중 하나를 수행합니다.

   - 서비스 역할이 없는 경우 새 CodeBuild 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
   - CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

   > **Note**
   >
   > 콘솔을 사용하여 빌드 프로젝트를 만들거나 업데이트할 때 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

8. Buildspec에서 다음 중 하나를 수행합니다.

   - buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
   - 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

   자세한 내용은 buildspec 참조을 참조하세요.

9. 아티팩트의 유형에서 Amazon S3를 선택하여 빌드 출력을 S3 버킷에 저장합니다.
10. 버킷 이름에서 1단계에서 웹사이트처럼 작동하도록 구성한 S3 버킷의 이름을 선택합니다.

11. 환경에서 빌드 명령 삽입을 선택한 경우 출력 파일에 대해 출력 버킷에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 두 개 이상인 경우 쉼표를 사용하여 각 위치를 구분합니다(예: **`appspec.yml, target/my-app.jar`**). 자세한 정보는 Artifacts reference-key in the buildspec file을 참조하세요.
12. 아티팩트 암호화 비활성화를 선택합니다.
13. 추가 구성을 확장하고 적절한 옵션을 선택합니다.
14. 빌드 프로젝트 생성을 선택합니다. 빌드 프로젝트 페이지의 빌드 기록에서 빌드 시작을 선택하여 빌드를 실행합니다.
15. (선택 사항) Amazon S3 개발자 안내서의 예제: Amazon과 함께 웹 사이트 속도 CloudFront 향상에 나와 있는 지침을 따르십시오.

다중 입력 소스 및 출력 아티팩트 샘플

둘 이상의 입력 소스와 둘 이상의 출력 아티팩트 세트를 사용하여 AWS CodeBuild 빌드 프로젝트를 생성할 수 있습니다. 이번 샘플은 아래와 같은 빌드 프로젝트를 설정하는 방법에 대한 내용입니다.

- 유형에 따라 여러 가지 소스와 리포지토리를 사용합니다.
- 단일 빌드에서 다수의 S3 버킷에 빌드 아티팩트를 게시합니다.

이번 샘플에서는 빌드 프로젝트를 생성하여 빌드를 실행하는 데 사용합니다. 또한 빌드 프로젝트의 buildspec 파일을 사용해 소스 1개 이상을 포함시키고, 아티팩트 세트 1개 이상을 생성하는 방법에 대해서 설명합니다.

1. 소스를 하나 이상의 S3 버킷,, CodeCommit GitHub, GitHub 엔터프라이즈 서버 또는 Bitbucket 리포지토리에 업로드합니다.
2. 기본 소스로 사용할 소스를 선택합니다. 이 소스는 buildspec 파일을 CodeBuild 찾아 실행하는 소스입니다.
3. 빌드 프로젝트를 생성합니다. 자세한 정보는 AWS CodeBuild에서 빌드 프로젝트 생성을 참조하세요.
4. 빌드 프로젝트를 만들고, 빌드를 실행하고, 빌드에 대한 정보를 얻으세요.
5. 를 사용하여 빌드 프로젝트를 만드는 경우 JSON 형식의 `create-project` 명령 입력은 다음과 비슷할 수 있습니다. AWS CLI

```
{
  "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"
}
```

기본 소스는 source 속성에서 정의됩니다. 그 밖에 다른 소스는 보조 소스라고 불리며, secondarySources에 표시됩니다. 보조 소스는 모두 자체 디렉터리에 설치됩니다. 이 디렉터리는 내장 환경 변수인 CODEBUILD_SRC_DIR_*sourceIdentifer*에 저장됩니다. 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.

`secondaryArtifacts` 속성에는 아티팩트 정의 목록이 포함됩니다. 이러한 아티팩트는 `secondary-artifacts` 블록 내에 중첩되는 buildspec 파일의 `artifacts` 블록을 사용합니다.

buildspec 파일의 보조 아티팩트는 아티팩트와 동일한 구조를 가지고 있지만 아티팩트 식별자로 구분됩니다.

> **Note**
>
> CodeBuild API에서 보조 아티팩트의 `artifactIdentifier` on은 및 의 필수 속성입니다. `CreateProject` `UpdateProject` 보조 아티팩트를 참조할 때 사용해야 합니다.

앞서 얘기한 JSON 형식의 입력을 사용하면 프로젝트의 buildspec 파일은 다음과 같은 모습이 될 수 있습니다.

```
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
```

기본 속성의 버전은 API를 사용해 sourceVersion의 StartBuild 속성에서 재정의할 수 있습니다. 보조 소스 버전을 1개 이상 재정의할 때는 secondarySourceVersionOverride 속성을 사용하십시오.

JSON 형식의 start-build 명령 입력은 다음과 같을 수 있습니다. AWS CLI

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

소스 샘플이 없는 프로젝트

소스를 구성할 때 **NO_SOURCE** 소스 유형을 선택하여 CodeBuild 프로젝트를 구성할 수 있습니다. 소스 유형이 **NO_SOURCE**일 경우, 프로젝트에 소스가 없으므로 buildspec 파일을 지정할 수 없습니다. 대신에 create-project CLI 명령에 대한 JSON 형식 입력의 buildspec 속성 내 YAML 형식 buildspec 문자열 지정이 필요합니다. 값이 다음과 같을 것입니다.

```
{
  "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"
}
```

자세한 정보는 빌드 프로젝트 생성(AWS CLI)을 참조하세요.

여러 소스 입력을 사용하여 여러 출력 아티팩트를 만드는 파이프라인을 만드는 방법을 알아보려면 CodeBuild 을 참조하십시오AWS CodePipeline 여러 입력 소스 CodeBuild 및 출력 아티팩트와의 통합 샘플 .

buildspec 파일 샘플의 런타임 버전: CodeBuild

Amazon Linux 2(AL2) 표준 이미지 버전 1.0 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하는 경우 buildspec 파일의 `runtime-versions` 섹션에서 하나 이상의 런타임을 지정할 수 있습니다. 이 샘플은 프로젝트 런타임 변경, 둘 이상의 런타임 지정, 다른 런타임에 종속되는 런타임 지정 방법을 보여줍니다. 지원되는 런타임에 대한 자세한 내용은 Docker 이미지 제공: CodeBuild 단원을 참조하십시오.

> **Note**
>
> 빌드 컨테이너에서 도커를 사용할 경우에는 권한이 있는 모드에서 빌드가 실행되어야 합니다. 자세한 내용은 AWS CodeBuild에서 빌드 실행 및 AWS CodeBuild에서 빌드 프로젝트 생성 섹션을 참조하세요.

런타임 버전 업데이트

buildspec 파일의 `runtime-versions` 섹션을 업데이트하여 프로젝트에서 사용되는 런타임을 새 버전으로 수정할 수 있습니다. 다음 예제는 Java 버전 8 및 11을 지정하는 방법을 보여 줍니다.

- Java 버전 8을 지정하는 `runtime-versions` 부분:

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

- Java 버전 11을 지정하는 `runtime-versions` 부분:

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

다음 예제는 Ubuntu 표준 이미지 5.0 또는 Amazon Linux 2 표준 이미지 3.0을 사용하여 Python의 다양한 버전을 지정하는 방법을 보여 줍니다.

- Python 버전 3.7을 지정하는 `runtime-versions` 섹션:

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

- Python 버전 3.8을 지정하는 `runtime-versions` 섹션:

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

이 샘플은 Java 버전 8 런타임으로 시작해서 이후 Java 버전 10 런타임으로 업데이트되는 프로젝트를 보여줍니다.

1. Maven을 다운로드하고 설치합니다. 자세한 정보는 Apache Maven 웹 사이트의 Downloading Apache Maven 및 Installing Apache Maven 단원을 참조하십시오.
2. 로컬 컴퓨터나 인스턴스의 빈 디렉터리로 전환한 다음, 아래 Maven 명령을 실행합니다.

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

   성공하면 다음 디렉터리 구조 및 파일이 생성됩니다.

   ```
   .
   ### ROOT
       ### pom.xml
       ### src
           ### main
               ### resources
               ### webapp
                   ### WEB-INF
                   #   ### web.xml
                   ### index.jsp
   ```

3. 다음 콘텐츠를 가진 buildspec.yml이라는 파일을 생성합니다: 파일을 *(root directory name)*/my-web-app 디렉터리에 저장합니다.

```
version: 0.2

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

buildspec 파일에서

- runtime-versions 부분은 해당 프로젝트에서 Java 런타임 버전 8을 사용하도록 지정합니다.
- - java -version 명령은 빌드할 때 프로젝트에서 사용하는 Java 버전을 표시합니다.

파일 구조가 아래와 같이 나타날 것입니다.

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

4. my-web-app디렉터리의 콘텐츠를 S3 입력 버킷 CodeCommit GitHub, 또는 Bitbucket 리포지토리에 업로드합니다.

> **Important**
>
> *(root directory name)* 또는 *(root directory name)*/my-web-app은 업로드하지 말고, *(root directory name)*/my-web-app 안에 있는 디렉터리 및 파일만 업로드하십시오.
> S3 입력 버킷을 사용하고 있는 경우, 디렉터리 구조 및 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드합니다. *(root directory name)* 또는 *(root directory name)*/my-web-app을 ZIP 파일에 추가하지 말고, *(root directory name)*/my-web-app 안에 있는 디렉터리 및 파일만 추가하십시오.

5. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
6. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요. 다음 설정을 제외하고 모든 설정을 기본값 그대로 둡니다.
   - 환경:
     - 환경 이미지에서 이미지 관리를 선택합니다.
     - 운영 체제에서 Amazon Linux 2를 선택합니다.
     - 런타임에서 표준을 선택합니다.
     - 이미지의 경우 aws/codebuild/amazonlinux2-x86_64-standard:4.0을 선택합니다.
7. 빌드 시작를 선택합니다.
8. Build configuration(빌드 구성)에서 기본값을 적용한 다음 빌드 시작을 선택합니다.
9. 빌드가 완료되면 빌드 로그 탭에서 빌드 출력을 확인합니다. 다음과 유사한 출력 화면이 표시되어야 합니다.

```
[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. Java 버전 11의 `runtime-versions` 부분 업데이트:

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

11. 변경을 저장한 후 빌드를 다시 실행하고 빌드 출력을 확인합니다. Java 설치 버전이 11인지 확인해야 합니다. 다음과 유사한 출력 화면이 표시되어야 합니다.

```
[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/*;
```

런타임 2개 지정

동일한 CodeBuild 빌드 프로젝트에서 둘 이상의 런타임을 지정할 수 있습니다. 이 샘플은 두 개의 소스 파일을 사용하는데 하나는 Go 런타임을 사용하고, 다른 하나는 Node.js 런타임을 사용합니다.

1. `my-source`이라는 디렉터리를 생성합니다.
2. `my-source` 디렉터리 안에 이름이 `golang-app`인 디렉터리를 생성합니다.

3. 다음 콘텐츠를 가진 hello.go이라는 파일을 생성합니다: 파일을 golang-app 디렉터리에 저장합니다.

```
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. my-source 디렉터리 안에 이름이 nodejs-app인 디렉터리를 생성합니다. golang-app 디렉터리와 레벨이 같아야 합니다.

5. 다음 콘텐츠를 가진 index.js이라는 파일을 생성합니다: 파일을 nodejs-app 디렉터리에 저장합니다.

```
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. 다음 콘텐츠를 가진 package.json이라는 파일을 생성합니다: 파일을 nodejs-app 디렉터리에 저장합니다.

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

```
}
```

7. 다음 콘텐츠를 가진 buildspec.yml이라는 파일을 생성합니다: my-source 디렉터리에, nodejs-app 및 golang-app 디렉터리와 같은 레벨에 파일을 저장합니다. runtime-versions 섹션은 Node.js 버전 12 및 Go 버전 1.13 런타임을 지정합니다.

```
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. 파일 구조가 아래와 같이 나타날 것입니다.

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

9. `my-source`디렉터리의 콘텐츠를 S3 입력 버킷 또는 CodeCommit GitHub, 또는 Bitbucket 리포지토리에 업로드합니다.

   > **Important**
   >
   > S3 입력 버킷을 사용하고 있는 경우, 디렉터리 구조 및 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드합니다. `my-source`를 ZIP 파일에 추가하지 말고, `my-source`에 있는 디렉터리와 파일만 추가하십시오.

10. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
11. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요. 다음 설정을 제외하고 모든 설정을 기본값 그대로 둡니다.

    - 환경:
      - 환경 이미지에서 이미지 관리를 선택합니다.
      - 운영 체제에서 Amazon Linux 2를 선택합니다.
      - 런타임에서 표준을 선택합니다.
      - 이미지의 경우 aws/codebuild/amazonlinux2-x86_64-standard:4.0을 선택합니다.
12. 빌드 프로젝트 생성을 선택합니다.
13. 빌드 시작를 선택합니다.
14. Build configuration(빌드 구성)에서 기본값을 적용한 다음 빌드 시작을 선택합니다.
15. 빌드가 완료되면 빌드 로그 탭에서 빌드 출력을 확인합니다. 다음과 유사한 출력 화면이 표시되어야 합니다. Go 및 Node.js 런타임의 출력을 보여줍니다. Go 및 Node.js 애플리케이션의 출력도 보여줍니다.

```
[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
```

소스 버전 샘플은 다음과 같습니다. AWS CodeBuild

이 샘플은 커밋 ID 외의 형식(커밋 SHA라고 함)을 사용하여 소스 버전을 지정하는 방법을 입증합니다. 다음 방법으로 소스 버전을 지정할 수 있습니다.

- Amazon S3 소스 공급자의 경우 빌드 입력 ZIP 파일을 나타내는 객체의 버전 ID를 사용합니다.
- CodeCommit, Bitbucket GitHub, GitHub 엔터프라이즈 서버의 경우 다음 중 하나를 사용하십시오.

  - 풀 요청 참조로서 풀 요청(예: `refs/pull/1/head`).
  - 브랜치 이름으로서 브랜치.
  - 커밋 ID.
  - 태그.
  - 참조 및 커밋 ID. 참조는 다음 중 하나일 수 있습니다.
    - 태그(예: `refs/tags/mytagv1.0^{full-commit-SHA}`).
    - 브랜치(예: `refs/heads/mydevbranch^{full-commit-SHA}`).
    - 풀 요청(예: `refs/pull/1/head^{full-commit-SHA}`).
- GitLab 자체 관리형의 경우 다음 중 하나를 사용하십시오. GitLab
  - 브랜치 이름으로서 브랜치.
  - 커밋 ID.
  - 태그.

> **Note**
>
> 리포지토리가 GitHub 또는 GitHub Enterprise Server인 경우에만 풀 요청 소스의 버전을 지정할 수 있습니다.

참조 및 커밋 ID를 사용하여 버전을 지정하는 경우 빌드의 `DOWNLOAD_SOURCE` 단계는 버전만을 제공하는 경우보다 더 빠릅니다. 참조를 추가할 때 커밋을 찾기 위해 전체 리포지토리를 다운로드할 필요가 CodeBuild 없기 때문입니다.

- 커밋 ID(예: `1234567890123456789012346789012345678 9`)만을 사용하여 소스 버전을 지정할 수 있습니다. 이렇게 하면 버전을 찾으려면 전체 저장소를 CodeBuild 다운로드해야 합니다.
- 다음 형식으로 참조 및 커밋 ID를 사용하여 소스 버전을 지정할 수 있습니다. *`refs/heads/branchname^{full-commit-SHA}`*(예: `refs/heads/main^{1234567890123456789012346789012345678 9}`). 이렇게 하면 지정된 브랜치만 CodeBuild 다운로드하여 버전을 찾을 수 있습니다.

> **Note**
>
> 빌드 DOWNLOAD_SOURCE 단계를 가속화하기 위해 Git 클론 깊이를 낮은 수로 설정할 수도 있습니다. CodeBuild 더 적은 수의 리포지토리 버전을 다운로드합니다.

커밋 ID로 GitHub 리포지토리 버전을 지정하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요. 다음 설정을 제외하고 모든 설정을 기본값 그대로 둡니다.
   - 소스에서 다음과 같이 합니다.
     - 소스 제공자의 경우 선택하십시오 GitHub. 연결되지 않은 GitHub 경우 지침에 따라 연결하세요.
     - 리포지토리에서 퍼블릭 리포지토리를 선택합니다.
     - 리포지토리 URL에 **https://github.com/aws/aws-sdk-ruby.git**을 입력합니다.
   - 환경에서 다음과 같이 합니다.
     - 환경 이미지에서 이미지 관리를 선택합니다.
     - 운영 체제에서 Amazon Linux 2를 선택합니다.
     - 런타임에서 표준을 선택합니다.
     - 이미지의 경우 aws/codebuild/amazonlinux2-x86_64-standard:4.0을 선택합니다.
3. 빌드 사양에서 빌드 명령 삽입을 선택한 후 편집기로 전환을 선택합니다.
4. 빌드 명령에서 자리표시자 텍스트를 다음으로 바꿉니다.

```
version: 0.2

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

Ubuntu 표준 이미지 2.0 사용 시 `runtime-versions` 섹션이 필요합니다. 여기서 Ruby 버전 2.6 런타임이 지정되지만, 모든 런타임을 사용할 수 있습니다. `echo` 명령은 `CODEBUILD_RESOLVED_SOURCE_VERSION` 환경 변수에 저장된 소스 코드 버전을 표시합니다.

5. Build configuration(빌드 구성)에서 기본값을 적용한 다음 빌드 시작을 선택합니다.
6. 소스 버전에 **`046e8b67481d53bdc86c3f6affdd5d1afae6d369`**를 입력합니다. 이는 `https://github.com/aws/aws-sdk-ruby.git` 리포지토리의 커밋 SHA입니다.
7. 빌드 시작를 선택합니다.
8. 빌드 완료 시 다음과 같은 모양이어야 합니다.

   - 빌드 로그 탭에서 프로젝트의 어떤 버전이 사용되었습니다. 의 예는 다음과 같습니다.

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

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

   - 환경 변수 탭에서 Resolved source version(해결된 소스 버전)이 빌드 생성에 사용된 커밋 ID와 일치합니다.
   - 단계 세부 정보 탭에서 `DOWNLOAD_SOURCE` 단계의 기간입니다.

이들 단계는 소스의 동일한 버전을 사용하여 빌드를 생성하는 방법을 보여줍니다. 이 시점에 소스의 버전은 커밋 ID와 함께 참조를 사용하여 지정됩니다.

커밋 ID 및 참조를 사용하여 GitHub 리포지토리 버전을 지정하려면

1. 왼쪽 탐색 창에서 빌드 프로젝트를 선택한 후 이전에 생성된 프로젝트를 선택합니다.
2. 빌드 시작를 선택합니다.
3. 소스 버전에 **`refs/heads/main^{046e8b67481d53bdc86c3f6affdd5d1afae6d369}`**를 입력합니다. 이는 *`refs/heads/branchname`*`^{`*`full-commit-SHA`*`}` 형식의 브랜치 및 커밋 ID와 동일합니다.
4. 빌드 시작를 선택합니다.
5. 빌드 완료 시 다음과 같은 모양이어야 합니다.

   - 빌드 로그 탭에서 프로젝트의 어떤 버전이 사용되었습니다. 의 예는 다음과 같습니다.

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

```
046e8b67481d53bdc86c3f6affdd5d1afae6d369

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

- 환경 변수 탭에서 Resolved source version(해결된 소스 버전)이 빌드 생성에 사용된 커밋 ID와 일치합니다.
- 단계 세부 정보 탭에서 DOWNLOAD_SOURCE 단계의 기간은 커밋 ID만을 사용하여 소스 버전을 지정했을 때의 기간보다 짧아야 합니다.

에 대한 타사 소스 리포지토리 샘플 CodeBuild

주제

- 에 대한 Bitbucket 풀 리퀘스트 및 웹후크 필터 샘플 CodeBuild
- GitHub 에 대한 엔터프라이즈 서버 샘플 CodeBuild
- GitHub 풀 리퀘스트 및 웹훅 필터 샘플: CodeBuild

에 대한 Bitbucket 풀 리퀘스트 및 웹후크 필터 샘플 CodeBuild

AWS CodeBuild 소스 리포지토리가 Bitbucket인 경우 웹후크를 지원합니다. 즉, 소스 코드가 Bitbucket 리포지토리에 저장되어 있는 CodeBuild 빌드 프로젝트의 경우 코드 변경 사항이 리포지토리로 푸시될 때마다 웹후크를 사용하여 소스 코드를 다시 빌드할 수 있습니다. 자세한 정보는 Bitbucket Webhook 이벤트을 참조하세요.

이 샘플에서는 Bitbucket 리포지토리를 사용하여 풀 요청을 생성하는 방법을 보여줍니다. 또한 Bitbucket 웹후크를 CodeBuild 트리거하여 프로젝트 빌드를 만드는 방법도 보여줍니다.

> **Note**
>
> Webhook를 사용할 때 사용자가 예상치 못한 빌드를 트리거할 수 있습니다. 이 위험을 줄이려면 webhook 사용 모범 사례 섹션을 참조하세요.

주제

- 필수 조건
- Bitbucket을 소스 리포지토리로 사용하여 빌드 프로젝트 생성 및 Webhook 활성화
- Bitbucket Webhook를 사용하여 빌드 트리거

필수 조건

이 샘플을 실행하려면 AWS CodeBuild 프로젝트를 Bitbucket 계정에 연결해야 합니다.

> **Note**
>
> CodeBuild Bitbucket으로 권한을 업데이트했습니다. 이전에 프로젝트를 Bitbucket에 연결했는데 이제 Bitbucket 연결 오류가 발생하는 경우 웹훅을 관리할 CodeBuild 권한을 부여하려면 다시 연결해야 합니다.

Bitbucket을 소스 리포지토리로 사용하여 빌드 프로젝트 생성 및 Webhook 활성화

다음 단계는 Bitbucket을 소스 리포지토리로 사용하여 AWS CodeBuild 프로젝트를 만들고 웹후크를 활성화하는 방법을 설명합니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 빌드 프로젝트 생성을 선택합니다.
4. 프로젝트 구성에서 다음과 같이 합니다.

   프로젝트 이름

   이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 AWS 계정별로 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
5. 소스에서 다음과 같이 합니다.

   소스 공급자

   Bitbucket을 선택합니다. Bitbucket과 연결(다시 연결)하는 지침을 따르고 승인을 선택합니다.

   리포지토리

   내 Bitbucket 계정의 리포지토리를 선택합니다.

   이전에 Bitbucket 계정에 연결한 적이 없으면 Bitbucket 사용자 이름과 앱 암호를 입력하고 Bitbucket 보안 인증 저장을 선택합니다.

Bitbucket 리포지토리

Bitbucket 리포지토리의 URL을 입력합니다.

6. 기본 소스 webhook 이벤트에서 다음을 선택합니다.

> **Note**
>
> 기본 소스 webhook 이벤트 섹션은 이전 단계에서 Bitbucket 계정의 리포지토리를 선택한 경우에만 표시됩니다.

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. 필요한 경우 필터 그룹 추가를 선택하여 다른 필터 그룹을 추가합니다.

Bitbucket webhook 이벤트 유형 및 필터에 대한 자세한 내용은 Bitbucket Webhook 이벤트 섹션을 참조하세요.

7. 환경에서 다음과 같이 합니다.

환경 이미지

다음 중 하나를 선택합니다.

관리하는 AWS CodeBuild Docker 이미지를 사용하려면:

관리형 이미지를 선택한 후 운영 체제, 런타임, 이미지 및 이미지 버전에서 옵션을 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.

다른 도커 이미지를 사용하려면:

사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경

우 Amazon ECR 리포지토리 및 Amazon ECR 이미지를 사용하여 AWS 계정의 도커 이미지를 선택합니다.

프라이빗 도커 이미지를 사용하려면:

사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager무엇입니까를 참조하십시오. AWS Secrets Manager 사용 설명서에서.

서비스 역할

다음 중 하나를 선택합니다.

- CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 만들거나 업데이트할 때 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

8. Buildspec에서 다음 중 하나를 수행합니다.

   - buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
   - 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

   자세한 내용은 buildspec 참조을 참조하세요.

9. 결과물에서 다음과 같이 합니다.

   Type

   다음 중 하나를 선택합니다.

- 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
- S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
  - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
  - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
  - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.

추가 구성

Additional configuration(추가 구성)을 확장하고 옵션을 적절하게 설정합니다.

10. 빌드 프로젝트 생성을 선택합니다. 검토 페이지에서 빌드 시작을 선택하여 빌드를 실행합니다.

Bitbucket Webhook를 사용하여 빌드 트리거

Bitbucket 웹후크를 사용하는 프로젝트의 경우 Bitbucket 리포지토리가 소스 코드의 변경을 감지하면 빌드를 AWS CodeBuild 만듭니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 콘솔을 엽니다. AWS CodeBuild
2. 탐색 창에서 빌드 프로젝트를 선택한 다음 webhook에서 Bitbucket 리포지토리와 연결된 프로젝트를 선택합니다. Bitbucket webhook 프로젝트 생성에 대한 자세한 내용은 the section called “Bitbucket을 소스 리포지토리로 사용하여 빌드 프로젝트 생성 및 Webhook 활성화” 섹션을 참조하세요.
3. 프로젝트의 Bitbucket 리포지토리에서 코드를 변경합니다.
4. Bitbucket 리포지토리에서 풀 요청을 생성합니다. 자세한 내용은 API 요청 생성을 참조하십시오.
5. Bitbucket Webhook 페이지에서 View request(요청 보기)를 선택하여 최신 이벤트 목록을 봅니다.
6. 에서 반환한 응답에 대한 세부 정보를 보려면 세부 정보 보기를 선택합니다 CodeBuild. 값이 다음과 같을 것입니다.

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

7. Bitbucket 풀 요청 페이지로 이동하여 빌드 상태를 확인합니다.

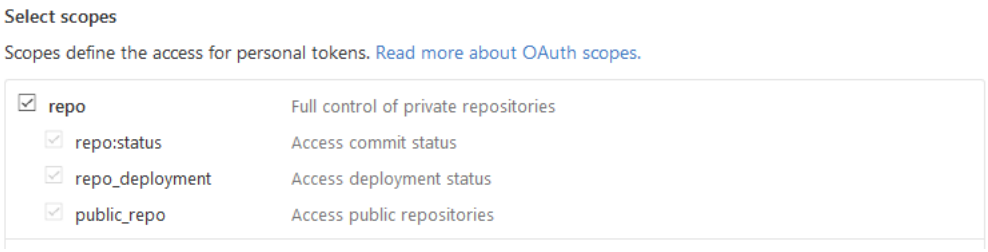
GitHub 에 대한 엔터프라이즈 서버 샘플 CodeBuild

AWS CodeBuild GitHub 엔터프라이즈 서버를 소스 리포지토리로 지원합니다. 이 샘플은 GitHub Enterprise Server 저장소에 인증서가 설치된 경우 CodeBuild 프로젝트를 설정하는 방법을 보여줍니다. 또한 코드 변경 사항이 GitHub Enterprise Server 리포지토리에 푸시될 때마다 소스 코드를 CodeBuild 다시 빌드하도록 웹후크를 활성화하는 방법도 보여줍니다.

필수 조건

1. 프로젝트를 위한 개인용 액세스 토큰을 생성하십시오. CodeBuild GitHub Enterprise 사용자를 생성하고 이 사용자에 대한 개인 액세스 토큰을 생성하는 것이 좋습니다. 프로젝트를 만들 때 사용할 수 있도록 클립보드에 복사하십시오. CodeBuild 자세한 내용은 GitHub 도움말 웹 사이트의 명령줄용 개인용 액세스 토큰 만들기를 참조하십시오.

   개인 액세스 토큰을 생성할 때 정의에 리포지토리 범위를 포함시킵니다.



2. GitHub 엔터프라이즈 서버에서 인증서를 다운로드하십시오. CodeBuild 인증서를 사용하여 저장소에 신뢰할 수 있는 SSL 연결을 설정합니다.

   Linux/macOS 클라이언트:

   터미널 창에서 다음 명령을 실행합니다.

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

   명령에서 자리 표시자를 다음 값으로 바꿉니다.

   HOST. GitHub 엔터프라이즈 서버 리포지토리의 IP 주소.

   PORTNUMBER. 연결에 사용하는 포트 번호입니다(예: 443).

   folder. 인증서를 다운로드한 폴더입니다.

filename. 인증서 파일의 파일 이름입니다.

> **Important**
>
> 인증서를 .pem 파일로 저장합니다.

Windows 클라이언트:

브라우저를 사용하여 GitHub 엔터프라이즈 서버에서 인증서를 다운로드합니다. 사이트의 인증서 세부 정보를 보려면 자물쇠 아이콘을 선택합니다. 인증서를 내보내는 방법에 대한 자세한 내용은 브라우저 설명서를 참조하십시오.

> **Important**
>
> 인증서를 .pem 파일로 저장합니다.

3. S3 버킷으로 인증서 파일을 업로드합니다. S3 버킷을 생성하는 방법에 대한 자세한 내용은 S3 버킷을 생성하려면 어떻게 해야 합니까?를 참조하십시오. S3 버킷으로 객체를 업로드하는 방법에 대한 자세한 내용은 버킷에 파일 및 폴더를 업로드하려면 어떻게 해야 합니까?를 참조하십시오.

> **Note**
>
> 이 버킷은 빌드와 같은 AWS 지역에 있어야 합니다. 예를 들어 미국 동부 (오하이오) 지역에서 빌드를 CodeBuild 실행하도록 지시하는 경우 버킷은 미국 동부 (오하이오) 지역에 있어야 합니다.

GitHubEnterprise Server를 원본 리포지토리로 사용하여 빌드 프로젝트를 만들고 웹훅을 활성화합니다 (콘솔).

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 AWS 계정별로 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.

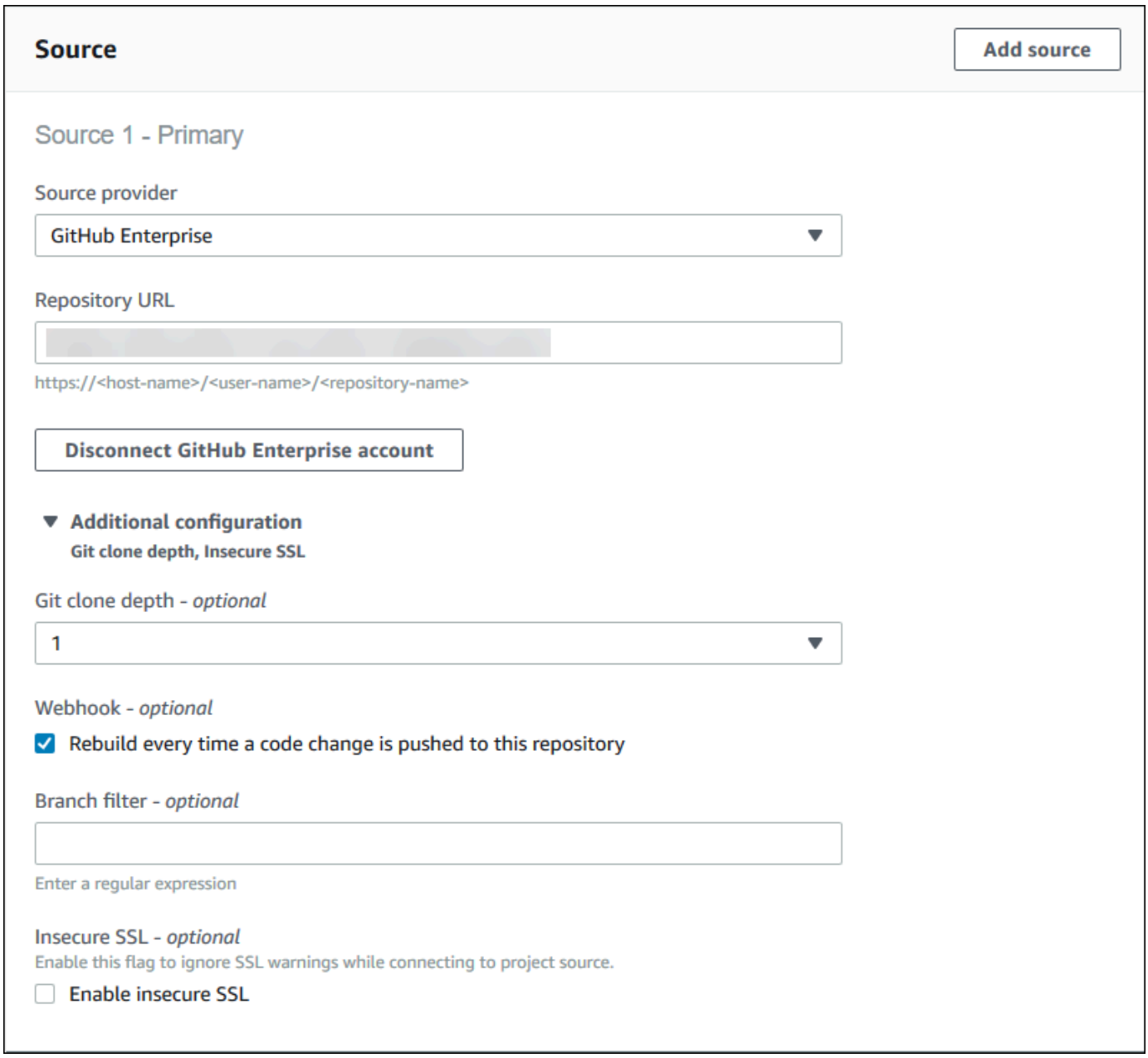
4. 소스의 소스 제공자에서 GitHub Enterprise를 선택합니다.

   - [Personal Access Token]에서 클리보드에 복사해 놓은 토큰을 붙여 넣고 [Save Token]을 선택합니다. 리포지토리 URL에 GitHub 엔터프라이즈 서버 리포지토리의 URL을 입력합니다.

     > **Note**
     >
     > 개인 액세스 토큰은 한 번만 입력하고 저장하면 됩니다. 향후 모든 AWS CodeBuild 프로젝트에서 이 토큰을 사용합니다.

   - 리포지토리 URL에 리포지토리 이름을 포함하여 리포지토리에 대한 경로를 입력합니다.
   - 추가 구성을 확장합니다.
   - 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드하려면 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
   - GitHub Enterprise Server 프로젝트 리포지토리에 연결하는 동안 SSL 경고를 무시하려면 [비보안 SSL 활성화] 를 선택합니다.

     > **Note**
     >
     > Enable insecure SSL(안전하지 않은 SSL 활성화)는 테스트 용도로만 사용하는 것이 좋습니다. 프로덕션 환경에 사용하면 안 됩니다.



5. 환경에서 다음과 같이 합니다.

   [Environment image]에서 다음 중 하나를 수행합니다.

   - 관리되는 AWS CodeBuild Docker 이미지를 사용하려면 관리 이미지를 선택한 다음 운영 체제, 런타임, 이미지 및 이미지 버전 중에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
   - 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우

External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 계정의 Docker 이미지를 선택하십시오. AWS

- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

6. 서비스 역할에서 다음 중 하나를 수행합니다.

   - 서비스 역할이 없는 경우 새 CodeBuild 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
   - CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

   > **Note**
   >
   > 콘솔을 사용하여 빌드 프로젝트를 만들거나 업데이트할 때 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

7. 추가 구성을 확장합니다.

   VPC로 CodeBuild 작업하려는 경우:

   - VPC의 경우 사용할 VPC ID를 선택합니다. CodeBuild
   - VPC 서브넷의 경우 사용하는 리소스가 포함된 서브넷을 선택합니다. CodeBuild
   - VPC 보안 그룹의 경우 VPC의 리소스에 대한 액세스를 허용하는 데 CodeBuild 사용하는 보안 그룹을 선택합니다.

   자세한 정보는 Amazon Virtual Private 클라우드와 AWS CodeBuild 함께 사용을 참조하세요.

8. Buildspec에서 다음 중 하나를 수행합니다.

- buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
- 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

자세한 내용은 buildspec 참조을 참조하세요.

9. 결과물의 유형에서 다음 중 하나를 수행합니다.

- 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
- S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
  - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
  - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
  - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.

10. Cache type(캐시 유형)에서 다음 중 하나를 선택합니다.

- 캐시를 사용하지 않으려면 [No cache]를 선택합니다.
- Amazon S3 캐시를 사용하려면 Amazon S3를 선택하고 다음을 수행합니다.
  - 버킷에서 캐시가 저장된 S3 버킷의 이름을 선택합니다.
  - (선택 사항) 캐시 경로 접두사에 Amazon S3 경로 접두사를 입력합니다. Cache path prefix(캐시 경로 접두사) 값은 디렉터리 이름과 비슷합니다. 따라서 캐시를 버킷의 동일한 디렉터리에 저장할 수 있습니다.

> **Important**
>
> 경로 접두사 끝에 후행 슬래시(/)를 추가하지 마십시오.

- 로컬 캐시를 사용하려면 로컬을 선택한 다음 하나 이상의 로컬 캐시 모드를 선택해야 합니다.

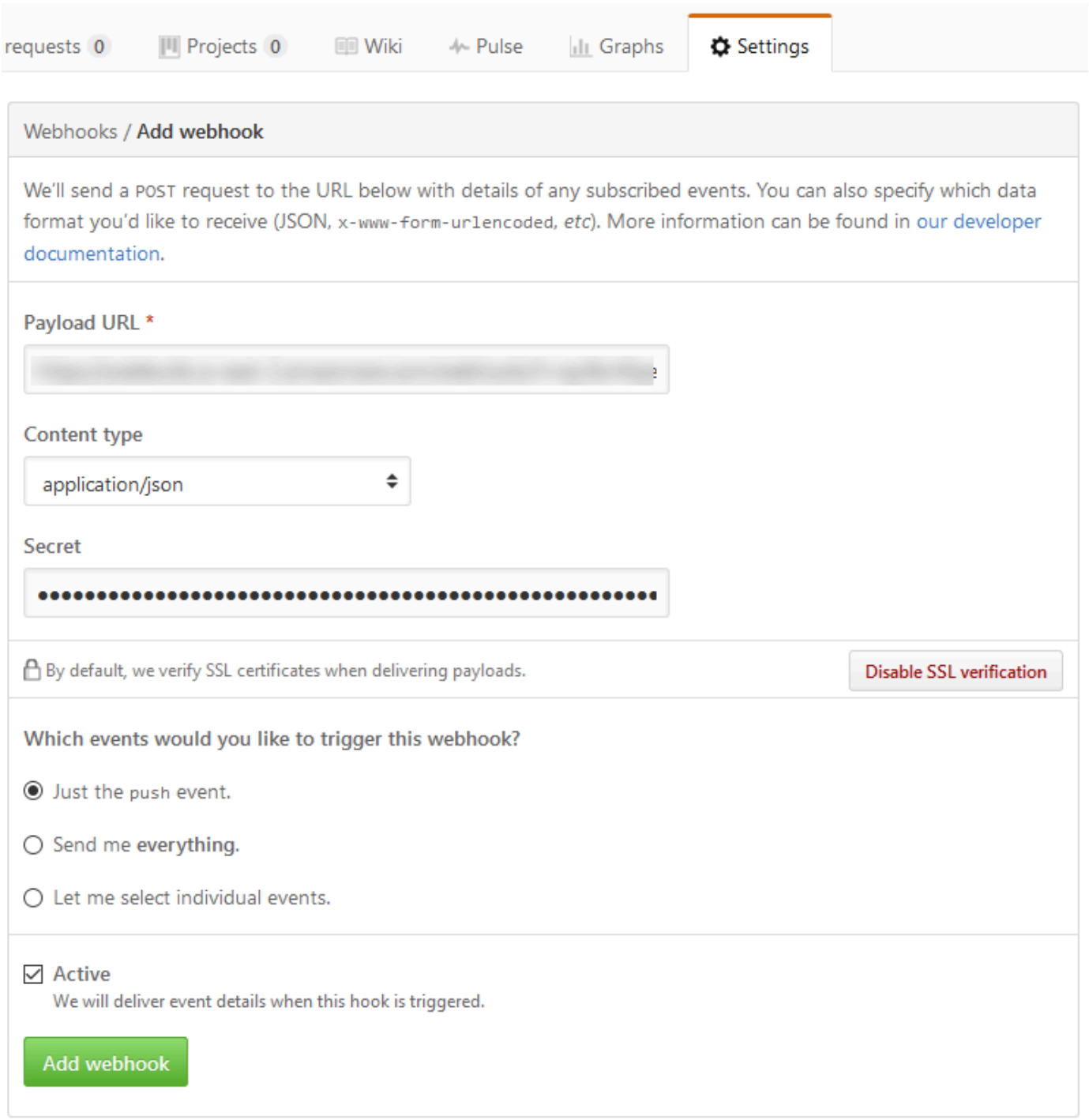
> **Note**
>
> Docker 계층 캐시 모드는 Linux에서만 사용할 수 있습니다. 이 모드를 선택할 경우 프로젝트를 권한이 있는 모드에서 실행해야 합니다.

캐시를 사용하면 빌드 환경의 재사용 가능한 특정 부분이 캐시에 저장되고 빌드 전반에서 사용되기 때문에 상당한 빌드 시간을 절약할 수 있습니다. buildspec 파일에 캐시를 지정하는 것에 대한 자세한 정보는 buildspec 구문 단원을 참조하십시오. 캐싱에 대한 자세한 정보는 AWS CodeBuild의 빌드 캐싱을 참조하십시오.

11. 빌드 프로젝트 생성을 선택합니다. 빌드 프로젝트 페이지에서 빌드 시작을 선택합니다.
12. 소스에서 webhook를 활성화한 경우 페이로드 URL 및 보안 암호 값이 포함된 webhook 생성 대화 상자가 표시됩니다.

> **Important**
>
> [Create webhook] 대화 상자는 한 번만 나타납니다. 페이로드 URL 및 보안 키를 복사합니다. 엔터프라이즈 서버에서 웹후크를 추가할 때 필요합니다. GitHub
> 페이로드 URL과 비밀 키를 다시 생성해야 하는 경우 먼저 GitHub 엔터프라이즈 서버 저장소에서 웹후크를 삭제해야 합니다. CodeBuild 프로젝트에서 Webhook 확인란의 선택을 취소한 다음 [Save] 를 선택합니다. 그런 다음 Webhook 확인란을 선택하여 CodeBuild 프로젝트를 만들거나 업데이트할 수 있습니다. [Create webhook] 대화 상자가 다시 나타납니다.

13. GitHub Enterprise Server에서 CodeBuild 프로젝트가 저장되어 있는 저장소를 선택합니다.
14. 설정, Hooks & services(후크 및 서비스), Add webhook(webhook 추가)를 차례로 선택합니다.
15. 페이로드 URL 및 보안 키를 입력하고 그 외 필드에 대해서는 기본값을 수락한 다음 [Add webhook]를 선택합니다.



16. CodeBuild 프로젝트로 돌아가십시오. [Create webhook] 대화 상자를 선택하고 [Start build]를 선택합니다.

GitHub 풀 리퀘스트 및 웹훅 필터 샘플: CodeBuild

AWS CodeBuild 소스 리포지토리가 다음과 GitHub 같은 경우 웹후크를 지원합니다. 즉, GitHub 리포지토리에 소스 코드가 저장된 CodeBuild 빌드 프로젝트의 경우 코드 변경 사항이 리포지토리에 푸시될 때마다 웹후크를 사용하여 소스 코드를 다시 빌드할 수 있습니다. CodeBuild 샘플은 샘플을 참조하십시오AWS CodeBuild .

> **Note**
>
> Webhook를 사용할 때 사용자가 예상치 못한 빌드를 트리거할 수 있습니다. 이 위험을 줄이려면 webhook 사용 모범 사례 섹션을 참조하세요.

소스 리포지토리로 사용하여 GitHub 빌드 프로젝트를 만들고 웹훅을 활성화합니다 (콘솔).

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 빌드 프로젝트 생성을 선택합니다.
4. 프로젝트 구성에서 다음과 같이 합니다.

   프로젝트 이름

   이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 AWS 계정별로 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
5. 소스에서 다음과 같이 합니다.

   소스 공급자

   선택하세요 GitHub. 지침에 따라 연결 (또는 재연결) 한 GitHub 다음 승인을 선택합니다.

   리포지토리

   내 GitHub 계정에서 리포지토리를 선택합니다.

   GitHub 리포지토리

   GitHub 리포지토리의 URL을 입력합니다.
6. 기본 소스 webhook 이벤트에서 다음을 선택합니다.

> **Note**
>
> 기본 소스 웹후크 이벤트 섹션은 이전 단계에서 내 GitHub 계정의 리포지토리를 선택한 경우에만 표시됩니다.

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. 필요한 경우 필터 그룹 추가를 선택하여 다른 필터 그룹을 추가합니다.

GitHub 웹후크 이벤트 유형 및 필터에 대한 자세한 내용은 을 참조하십시오. GitHub 웹후크 이벤트

7. 환경에서 다음과 같이 합니다.

   환경 이미지

   다음 중 하나를 선택합니다.

   에서 관리하는 Docker 이미지를 사용하려면: AWS CodeBuild

   관리형 이미지를 선택한 후 운영 체제, 런타임, 이미지 및 이미지 버전에서 옵션을 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.

   다른 도커 이미지를 사용하려면:

   사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리 및 Amazon ECR 이미지를 사용하여 AWS 계정의 도커 이미지를 선택합니다.

프라이빗 도커 이미지를 사용하려면:

사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager무엇입니까를 참조하십시오. AWS Secrets Manager 사용 설명서에서.

서비스 역할

다음 중 하나를 선택합니다.

- CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 만들거나 업데이트할 때 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

8. Buildspec에서 다음 중 하나를 수행합니다.

- buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
- 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

자세한 내용은 buildspec 참조을 참조하세요.

9. 결과물에서 다음과 같이 합니다.

Type

다음 중 하나를 선택합니다.

- 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
- S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.

- 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
- [Bucket name]에서 출력 버킷의 이름을 선택합니다.
- 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.

추가 구성

Additional configuration(추가 구성)을 확장하고 옵션을 적절하게 설정합니다.

10. 빌드 프로젝트 생성을 선택합니다. 검토 페이지에서 빌드 시작을 선택하여 빌드를 실행합니다.

확인 검사

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 다음 중 하나를 수행하십시오.
   - webhook를 검증하려는 빌드 프로젝트의 링크를 선택한 후 빌드 세부 정보를 선택합니다.
   - webhook를 검증하려는 빌드 프로젝트 옆에 있는 라디오 버튼을 선택하고 세부 정보 보기를 선택한 후 빌드 세부 정보를 선택합니다.
4. 기본 소스 webhook 이벤트에서 Webhook URL 링크를 선택합니다.
5. GitHub 리포지토리의 설정 페이지의 웹훅에서 풀 리퀘스트 및 푸시가 선택되어 있는지 확인합니다.
6. GitHub 프로필 설정의 개인 설정, 애플리케이션, 승인된 OAuth 앱에서 애플리케이션이 선택한 지역에 액세스할 수 있는 권한을 부여받았음을 확인할 수 있습니다. AWS

의미 체계 버전 관리를 사용하여 빌드 아티팩트 샘플 이름 지정

이번 샘플에는 빌드 시 생성되는 아티팩트 이름을 지정하는 방법에 대한 buildspec 파일 예제가 포함되어 있습니다. buildspec 파일에서 지정하는 이름에 Shell 명령과 환경 변수를 포함시켜 고유성을 유지할 수 있습니다. 또한 buildspec 파일에서 지정하는 이름은 프로젝트 생성 시 콘솔에 입력하는 이름을 재정의합니다.

여러 차례 빌드하는 경우 buildspec 파일에서 지정한 아티팩트 이름을 사용하면 출력 아티팩트 파일 이름의 고유성을 유지할 수 있습니다. 예를 들어 빌드 시 날짜와 타임스탬프를 사용해 아티팩트 이름에 삽입할 수 있습니다.

콘솔에서 입력한 아티팩트 이름을 buildspec 파일의 이름으로 재정의하고 싶다면 다음과 같이 실행하십시오.

1. 아티팩트 이름을 buildspec 파일의 이름으로 재정의하도록 빌드 프로젝트를 설정합니다.
   - 콘솔을 사용하여 빌드 프로젝트를 생성하는 경우 의미 체계 버전 관리 사용을 선택합니다. 자세한 정보는 빌드 프로젝트 만들기(콘솔)을 참조하세요.
   - 를 사용하는 경우 전달된 AWS CLI JSON `overrideArtifactName` 형식 파일에서 를 true로 설정하십시오. `create-project` 자세한 정보는 빌드 프로젝트 생성(AWS CLI)을 참조하세요.
   - AWS CodeBuild API를 사용하는 경우 프로젝트가 생성 또는 업데이트되거나 `ProjectArtifacts` 빌드가 시작될 때 객체에 `overrideArtifactName` 플래그를 설정하세요.
2. buildspec 파일에서 이름을 지정합니다. 아래 buildspec 파일 샘플을 참고하십시오.

아래 Linux 예제는 빌드가 생성된 날짜를 포함시켜 아티팩트 이름을 지정하는 방법을 보여 줍니다.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

이 Linux 예제는 CodeBuild 환경 변수를 사용하는 아티팩트 이름을 지정하는 방법을 보여줍니다. 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
```

```
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

아래 Windows 예제는 빌드가 생성된 날짜와 시간을 포함시켜 아티팩트 이름을 지정하는 방법에 대한 설명입니다.

```
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")
```

이 Windows 예제에서는 buildspec 파일에 선언된 변수와 환경 변수를 사용하여 아티팩트 이름을 지정하는 방법을 보여줍니다. CodeBuild 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.

```
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
```

자세한 정보는 에 대한 빌드 사양 참조 CodeBuild을 참조하세요.

마이크로소프트 윈도우용 샘플 CodeBuild

이 샘플은 Microsoft Windows Server 2019, .NET 프레임워크 및 .NET Core SDK를 실행하는 AWS CodeBuild 빌드 환경을 사용하여 F #과 Visual Basic으로 작성된 코드로 런타임 파일을 빌드합니다.

> **Important**
>
> 이러한 샘플을 실행하면 계정에 요금이 청구될 수 있습니다. AWS 여기에는 Amazon S3 AWS KMS, CloudWatch 로그와 관련된 AWS 리소스 및 작업에 대한 가능한 요금이 포함됩니다. CodeBuild 자세한 내용은 CodeBuild요금, Amazon S3 요금, AWS Key Management Service 요금 및 Amazon CloudWatch 요금을 참조하십시오.

샘플 실행

이러한 샘플을 실행하려면

1. 이 항목의 “디렉터리 구조” 및 “파일” 섹션에 설명된 대로 파일을 생성한 다음 S3 입력 버킷 CodeCommit 또는 GitHub 리포지토리에 업로드하십시오.

   > **Important**
   >
   > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
   > S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

2. 빌드 프로젝트를 생성합니다. 빌드 프로젝트는 `mcr.microsoft.com/dotnet/framework/sdk:4.8` 이미지를 사용하여 .NET Framework 프로젝트를 빌드해야 합니다.

   를 사용하여 빌드 프로젝트를 생성하는 경우 JSON 형식의 `create-project` 명령 입력이 이와 비슷해 보일 수 있습니다. AWS CLI (자리 표시자는 사용자의 값으로 바꾸십시오.)

   ```
   {
     "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. 빌드를 실행하고 빌드 실행의 단계를 따르세요.
4. 빌드 출력 아티팩트를 얻으려면 S3 출력 버킷에서 *windows-build-output-artifact*.zip 파일을 로컬 컴퓨터나 인스턴스에 다운로드합니다. 콘텐츠를 추출하여 런타임 및 기타 파일로 이동합니다.

   - NET Framework, FSharpHelloWorld.exe를 사용하는 F# 샘플의 런타임 파일은 FSharpHelloWorld\bin\Debug 디렉터리에 있습니다.
   - .NET Framework, VBHelloWorld.exe를 사용하는 Visual Basic 샘플의 런타임 파일은 VBHelloWorld\bin\Debug 디렉터리에 있습니다.

디렉터리 구조

이 샘플은 다음과 같은 디렉터리 구조를 가정합니다.

F# 및 .NET Framework

```
(root directory name)
### buildspec.yml
### FSharpHelloWorld.sln
### FSharpHelloWorld
    ### App.config
    ### AssemblyInfo.fs
    ### FSharpHelloWorld.fsproj
```

```
    ### Program.fs
```

Visual Basic 및 .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
```

파일

이러한 샘플은 다음 파일을 사용합니다.

F# 및 .NET Framework

buildspec.yml(*(root directory name)*에 있음):

```
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(*(root directory name)*에 있음):

```
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(*(root directory name)*\FSharpHelloWorld에 있음):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

AssemblyInfo.fs(*(root directory name)*\FSharpHelloWorld에 있음):

```
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(*root directory name*\FSharpHelloWorld에 있음):

```
<?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(*(root directory name)*\FSharpHelloWorld에 있음):

```
// 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 및 .NET Framework

buildspec.yml(*(root directory name)*에 있음):

```
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(*(root directory name)*에 있음):

```
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(*(root directory name)*\VBHelloWorld에 있음):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

HelloWorld.vb(*(root directory name)*\VBHelloWorld에 있음):

```
Module HelloWorld

  Sub Main()
    MsgBox("Hello World")
  End Sub

End Module
```

VBHelloWorld.vbproj(*(root directory name)*\VBHelloWorld에 있음):

```
<?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(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
'------------------------------------------------------------------------------
' <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(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
<?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(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
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(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
'------------------------------------------------------------------------------
' <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(*root directory name*)\VBHelloWorld\My Project에 있음):

```
<?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(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
'------------------------------------------------------------------------------
' <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(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
<?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>
```

AWS CodeBuild에서 빌드 계획

AWS CodeBuild를 사용하기 전에 다음 질문에 답해야 합니다.

1. 소스 코드는 어디에 저장되나요? CodeBuild 현재 다음 소스 코드 리포지토리 공급자를 통한 빌드를 지원합니다. 소스 코드에는 빌드 사양(buildspec) 파일이 포함되어 있어야 합니다. buildspec은 빌드를 실행하는 데 사용되는 빌드 명령 및 관련 설정 (YAML 형식) 의 모음입니다. CodeBuild 빌드 프로젝트 정의에서 buildspec을 선언할 수 있습니다.

리포지토리 공급자	필수	설명서
CodeCommit	리포지토리 이름. (선택 사항) 소스 코드와 연결된 커밋 ID.	AWS CodeCommit 사용 설명서에서 다음 주제를 참조하십시오. 리포지토리 만들기 CodeCommit 에서 커밋 생성 CodeCommit
Amazon S3	입력 버킷 이름. 소스 코드가 포함된 빌드 입력 ZIP 파일에 해당하는 객체 이름. (선택 사항) 빌드 입력 ZIP 파일에 연결된 버전 ID.	Amazon S3 시작 안내서에서 다음 주제를 참조하세요. 버킷 생성 버킷에 객체 추가
GitHub	리포지토리 이름. (선택 사항) 소스 코드와 연결된 커밋 ID.	GitHub 도움말 웹 사이트에서 다음 항목을 참조하십시오. 리포지토리 생성

리포지토리 공급자	필수	설명서
Bitbucket	리포지토리 이름. (선택 사항) 소스 코드와 연결된 커밋 ID.	Bitbucket Cloud 설명서 웹 사이트에서 다음 주제를 참조하십시오. 리포지토리 생성

2. 어떤 빌드 명령을 실행해야 하며 어떤 순서로 실행해야 합니까? 기본적으로 는 지정한 공급자로부터 빌드 입력을 CodeBuild 다운로드하고 지정한 버킷에 빌드 출력을 업로드합니다. 빌드 사양을 사용하면 다운로드된 빌드 입력을 원하는 빌드 출력으로 전환하는 방법을 지시할 수 있습니다. 자세한 내용은 buildspec 참조 섹션을 참조하십시오.
3. 빌드를 실행하는 데 어떤 런타임 및 도구가 필요합니까? 예를 들어 Java, Ruby, Python 또는 Node.js 중 어떤 용도로 빌드하고 있습니까? 빌드에 Maven이나 Ant 또는 Java, Ruby, Python용 컴파일러가 필요합니까? 빌드에 Git, AWS CLI 또는 다른 도구가 필요합니까?

   CodeBuild Docker 이미지를 사용하는 빌드 환경에서 빌드를 실행합니다. 이러한 Docker 이미지는 CodeBuild가 지원하는 리포지토리 유형에 저장되어 있어야 합니다. 여기에는 CodeBuild 도커 이미지 리포지토리, 도커 허브, 아마존 Elastic Container 레지스트리 (Amazon ECR) 가 포함됩니다. CodeBuild Docker 이미지 리포지토리에 대한 자세한 내용은 을 참조하십시오. Docker 이미지 제공: CodeBuild
4. 에서 자동으로 제공되지 않는 AWS 리소스가 필요하신가요? CodeBuild 그렇다면 해당 리소스에는 어떤 보안 정책이 필요합니까? 예를 들어 해당 리소스를 사용할 수 있도록 CodeBuild 서비스 역할을 수정해야 CodeBuild 할 수 있습니다.
5. VPC로 작업하고 CodeBuild 싶으신가요? 그렇다면 VPC 구성에 대한 VPC ID, 서브넷 ID 및 보안 그룹 ID가 필요합니다. 자세한 설명은 Amazon Virtual Private 클라우드와 AWS CodeBuild 함께 사용 섹션을 참조하세요.

위의 질문에 답을 했다면 빌드를 성공적으로 실행하는 데 필요한 설정 및 리소스가 확인되었을 것입니다. 빌드를 실행하려면 다음을 수행하면 됩니다.

- AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용합니다. 자세한 설명은 빌드 실행 섹션을 참조하세요.

- 에서 AWS CodePipeline 파이프라인을 만들거나 식별한 다음 코드를 자동으로 테스트하거나 빌드를 실행하거나 둘 다 CodeBuild 실행하도록 지시하는 빌드 또는 테스트 작업을 추가하세요. 자세한 내용은 CodePipeline 함께 사용 CodeBuild을(를) 참조하세요.

에 대한 빌드 사양 참조 CodeBuild

이 주제에서는 빌드 사양(buildspec) 파일에 대한 중요한 참조 정보를 제공합니다. 빌드스펙은 빌드를 실행하는 데 사용되는 빌드 명령 및 관련 설정 (YAML 형식) 의 모음입니다. CodeBuild 소스 코드의 일부로 buildspec을 포함하거나, 빌드 프로젝트를 생성할 때 buildspec을 정의할 수 있습니다. 빌드 사양의 작동 방식에 대한 자세한 정보는 CodeBuild 작동 방식 단원을 참조하십시오.

주제

- buildspec 파일 이름 및 스토리지 위치
- buildspec 구문
- buildspec 예제
- buildspec 버전
- 배치 빌드 buildspec 참조

buildspec 파일 이름 및 스토리지 위치

buildspec을 소스 코드의 일부로 포함하는 경우, 기본적으로 buildspec 파일에 `buildspec.yml`이라는 이름을 지정해야 하며 이 파일을 소스 디렉터리의 루트에 배치해야 합니다.

기본 buildspec 파일 이름과 위치를 재정의할 수 있습니다. 예를 들어, 다음을 수행할 수 있습니다.

- 동일한 리포지토리의 다른 빌드에 대해 `buildspec_debug.yml` 및 `buildspec_release.yml`과 같은 다른 buildspec 파일을 사용합니다.
- buildspec 파일을 `config/buildspec.yml`과 같은 소스 디렉터리의 루트가 아닌 다른 위치 또는 S3 버킷에 저장합니다. S3 버킷은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).

buildspec 파일의 이름과 상관없이, 한 빌드 프로젝트에 대해 하나의 buildspec만 지정할 수 있습니다.

기본 buildspec 파일 이름이나 위치 또는 둘 다를 재정의하려면 다음 중 하나를 수행합니다.

- AWS CLI `create-project`or `update-project` 명령을 실행하여 기본 제공 환경 변수 `buildspec` 값을 기준으로 대체 buildspec 파일의 경로로 값을 설정합니다. `CODEBUILD_SRC_DIR` SDK에서의 `create project` 작업과 동일한 작업을 수행할 수도 있습니다. AWS 자세한 내용은 빌드 프로젝트 생성 또는 빌드 프로젝트 설정 변경을 참조하세요.
- AWS CLI `start-build`명령을 실행하여 기본 제공 환경 변수 `buildspecOverride` 값을 기준으로 대체 buildspec 파일의 경로로 값을 설정합니다. `CODEBUILD_SRC_DIR` SDK에서의 `start build` 작업과 동일한 작업을 수행할 수도 있습니다. AWS 자세한 정보는 빌드 실행을 참조하세요.
- AWS CloudFormation 템플릿에서 특정 유형의 `AWS::CodeBuild::Project` 리소스의 `BuildSpec` 속성을 빌트인 환경 변수 값을 기준으로 대체 buildspec 파일의 경로로 설정합니다. `Source` `CODEBUILD_SRC_DIR` 자세한 내용은 사용 설명서의 AWS CodeBuild 프로젝트 소스의 BuildSpec 속성을 참조하십시오.AWS CloudFormation

buildspec 구문

buildspec 파일은 YAML 형식으로 표시해야 합니다.

명령에 YAML에서 지원하지 않는 문자 또는 문자열이 포함된 경우 명령을 따옴표("")로 묶어야 합니다. YAML에서는 콜론(:) 뒤에 공백이 올 수 없으므로 다음 명령을 따옴표로 묶습니다. 명령의 따옴표는 이스케이프(\")됩니다.

```
"export PACKAGE_NAME=$(cat package.json | grep name | head -1 | awk -F: '{ print $2 }'
 | sed 's/[\",]//g')"
```

buildspec 구문은 다음과 같습니다.

```
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
```

buildspec에는 다음이 포함됩니다.

version

필수 매핑. buildspec 버전을 나타냅니다. `0.2`을 사용할 것을 권장합니다.

> **Note**
>
> 버전 0.1도 계속 지원되지만 가능하면 버전 0.2를 사용할 것을 권장합니다. 자세한 정보는 buildspec 버전을 참조하세요.

run-as

선택적 시퀀스. Linux 사용자만 사용할 수 있습니다. 이 buildspec 파일의 명령을 실행하는 Linux 사용자를 지정합니다. `run-as`는 지정한 사용자에게 읽기 및 실행 권한을 부여합니다. buildspec 파일 처음에 `run-as`를 지정할 경우 모든 명령에 전역적으로 적용됩니다. 모든 buildspec 파일 명령에 대한 사용자를 지정하지 않으려는 경우 `phases` 블록 중 하나에 `run-as`를 사용하여 단계에 명령에 대한 사용자를 지정할 수 있습니다. `run-as`를 지정하지 않으면 모든 명령이 루트 사용자로 실행됩니다.

env

선택적 시퀀스. 하나 이상의 사용자 지정 환경 변수에 대한 정보를 나타냅니다.

> **Note**
>
> 민감한 정보를 보호하기 위해 CodeBuild 로그에는 다음과 같은 내용이 숨겨져 있습니다.
>
> - AWS 액세스 키 ID. 자세한 내용은AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.
> - 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.

- 를 사용하여 지정된 문자열 AWS Secrets Manager. 자세한 정보는 키 관리을 참조하세요.

env/shell

선택적 시퀀스. Linux 또는 Windows 운영 체제에서 지원되는 쉘을 지정합니다.

Linux 운영 체제의 경우 지원되는 쉘 태그는 다음과 같습니다.

- `bash`
- `/bin/sh`

Windows 운영 체제의 경우 지원되는 쉘 태그는 다음과 같습니다.

- `powershell.exe`
- `cmd.exe`

env/variables

env가 지정된 경우 및 사용자 지정 환경 변수를 일반 텍스트로 정의하려고 할 때 필요합니다. *key*/*value* 스칼라 매핑을 포함하며, 각 매핑은 일반 텍스트의 단일 사용자 지정 환경 변수를 나타냅니다. *key*는 사용자 지정 환경 변수의 이름이고, *value*는 이 변수의 값입니다.

> **Important**
>
> 중요한 값을 환경 변수에 저장하지 마세요. CodeBuild 콘솔 및 와 같은 도구를 사용하여 환경 변수를 일반 텍스트로 표시할 수 AWS CLI있습니다. 이 단원의 뒷부분에서 설명하는 것처럼 중요한 값의 경우 `parameter-store` 또는 `secrets-manager` 매핑을 사용하는 것이 좋습니다.
>
> 사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.
>
> `CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
>
> 여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다. 빌드를 생성할 때 환경 변수를 추가 또는 재정의할 수 있습니다. 자세한 정보는 AWS CodeBuild에서 빌드 실행을 참조하세요.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다. 프로젝트를 생성 또는 편집할 때 프로젝트 수준에서 환경 변수를 추가할 수 있습니다. 자세한 내용은 AWS CodeBuild에서 빌드 프로젝트 생성 및 AWS CodeBuild에서 빌드 프로젝트 설정 변경 섹션을 참조하세요.
- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

env/parameter-store

env가 지정되었으며 Amazon EC2 Systems Manager Parameter Store에 저장된 사용자 지정 환경 변수를 검색하려고 할 때 필요합니다. *key*/*value* 스칼라 매핑을 포함하며, 각 매핑은 Amazon EC2 Systems Manager Parameter Store에 저장된 하나의 사용자 지정 환경 변수를 나타냅니다. *key*는 이 사용자 지정 환경 변수를 참조하기 위해 나중에 빌드 명령에서 사용할 이름이고, *value*는 Amazon EC2 Systems Manager Parameter Store에 저장되는 사용자 지정 환경 변수의 이름입니다. 중요한 값을 저장하려면 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 안내: 문자열 파라미터 생성 및 테스트(콘솔)를 참조하세요.

> **Important**
>
> Amazon EC2 Systems Manager 파라미터 스토어에 저장된 사용자 지정 환경 변수를 CodeBuild 검색하려면 서비스 역할에 작업을 `ssm:GetParameters` 추가해야 합니다. CodeBuild 자세한 정보는 CodeBuild 서비스 역할 생성을 참조하세요.
> Amazon EC2 Systems Manager Parameter Store에서 검색하는 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `MY_VAR`인 `my_value`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 검색하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `PATH`인 `/usr/local/sbin:/usr/local/bin`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 검색하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.
> `CODEBUILD_`로 시작하는 이름으로 환경 변수를 저장하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
> 여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다. 빌드를 생성할 때 환경 변수를 추가 또는 재정의할 수 있습니다. 자세한 정보는 AWS CodeBuild에서 빌드 실행을 참조하세요.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다. 프로젝트를 생성 또는 편집할 때 프로젝트 수준에서 환경 변수를 추가할 수 있습니다. 자세한 내용은 AWS CodeBuild에서 빌드 프로젝트 생성 및 AWS CodeBuild에서 빌드 프로젝트 설정 변경 섹션을 참조하세요.
- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

env/secrets-manager

에 저장된 사용자 지정 환경 변수를 검색하려는 경우 필요합니다. AWS Secrets Manager다음 패턴을 사용하여 Secrets Manager `reference-key`를 지정합니다.

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

<key>

(필수) 로컬 환경 변수 이름입니다. 이 이름을 사용하면 빌드 중에 변수에 액세스할 수 있습니다.

<secret-id>

(필수) 보안 암호에 대한 고유 식별자 역할을 하는 이름 또는 Amazon 리소스 이름(ARN)입니다. AWS 계정에 있는 암호에 액세스하려면 암호 이름을 지정합니다. 다른 AWS 계정의 비밀에 액세스하려면 비밀 ARN을 지정하십시오.

<json-key>

(선택 사항) 해당 값을 검색하려는 Secrets Manager 키-값 페어의 키 이름을 지정합니다. `json-keya`를 지정하지 않으면 전체 비밀 CodeBuild 텍스트를 검색합니다.

<version-stage>

(선택 사항) 버전에 연결된 레이블을 스테이징하여 검색하려는 보안 암호 버전을 지정합니다. 스테이징 레이블은 교체 프로세스 도중 다른 버전을 추적하는 데 사용됩니다. `version-stage`를 사용하는 경우 `version-id`를 지정하지 마십시오. 버전 스테이지 또는 버전 ID를 지정하지 않으면 기본값은 `AWSCURRENT`의 버전 스테이지 값으로 버전을 검색하는 것입니다.

<version-id>

(선택 사항) 사용하려는 암호의 버전에 대한 고유 식별자를 지정합니다. `version-id`을 지정할 경우 `version-stage`을 지정하지 마십시오. 버전 스테이지 또는 버전 ID를 지정하지 않으면 기본값은 `AWSCURRENT`의 버전 스테이지 값으로 버전을 검색하는 것입니다.

다음 예제에서 `TestSecret`은 Secrets Manager에 저장되는 키-값 페어의 이름입니다. `TestSecret`의 키는 `MY_SECRET_VAR`입니다. 빌드 중에 `LOCAL_SECRET_VAR` 이름을 사용하여 변수에 액세스합니다.

```
env:
  secrets-manager:
    LOCAL_SECRET_VAR: "TestSecret:MY_SECRET_VAR"
```

자세한 내용은AWS Secrets Manager 사용 설명서에서 AWS Secrets Manager란 무엇입니까? 단원을 참조하세요.

env/exported-variables

선택적 매핑. 내보낼 환경 변수를 나열하는 데 사용됩니다. `exported-variables`에서 별도의 행에 내보낼 각 변수의 이름을 지정합니다. 내보낼 변수는 빌드 중에 컨테이너에서 사용할 수 있어야 합니다. 내보내는 변수는 환경 변수일 수 있습니다.

내보낸 환경 변수는 와 함께 AWS CodePipeline 사용하여 현재 빌드 단계에서 파이프라인의 후속 단계로 환경 변수를 내보내는 데 사용됩니다. 자세한 내용을 알아보려면AWS CodePipeline 사용 설명서의 변수 작업을 참조하세요.

빌드하는 동안 변수의 값은 `install` 단계부터 사용할 수 있습니다. 이 값은 `install` 단계 시작과 `post_build` 단계 끝 사이에서 업데이트할 수 있습니다. `post_build` 단계가 끝나면 내보낸 변수의 값을 변경할 수 없습니다.

> **Note**
>
> 다음은 내보낼 수 없습니다.
>
> - 빌드 프로젝트에서 지정된 Amazon EC2 Systems Manager Parameter Store 보안 암호
> - 빌드 프로젝트에서 지정된 Secrets Manager 보안 암호
> - `AWS_`로 시작하는 환경 변수

env/ git-credential-helper

선택적 매핑. Git 자격 증명 도우미를 CodeBuild 사용하여 Git 자격 증명을 제공하는지 여부를 나타내는 데 사용됩니다. `yes`사용하는 경우 그렇지 않은 경우 `no`이거나 값이 지정되지 않은 것입니다. 자세한 내용은 Git 웹사이트의 gitcredentials를 참조하십시오.

> **Note**
>
> 퍼블릭 Git 리포지토리에 대해 webhook에서 트리거한 빌드의 경우에는 `git-credential-helper`가 지원되지 않습니다.

proxy

선택적 시퀀스. 명시적 프록시 서버에서 빌드를 실행할 경우 설정을 나타내는 데 사용됩니다. 자세한 정보는 명시적 프록시 서버에서 CodeBuild 실행을 참조하세요.

proxy/upload-artifacts

선택적 매핑. 명시적 프록시 서버에서 빌드가 아티팩트를 업로드하도록 하려면 `yes`로 설정합니다. 기본값은 `no`입니다.

proxy/logs

선택적 매핑. 명시적 프록시 서버에서 빌드하여 CloudWatch 로그를 생성하도록 `yes` 설정합니다. 기본값은 `no`입니다.

단계

필수 시퀀스. 빌드의 각 단계에서 CodeBuild 실행되는 명령을 나타냅니다.

> **Note**
>
> buildspec 버전 0.1에서는 빌드 환경에 있는 기본 셸의 개별 인스턴스에서 각 명령을 CodeBuild 실행합니다. 즉, 각 명령이 다른 모든 명령과 독립적으로 실행됩니다. 따라서 기본적으로 이전 명령의 상태에 따라 실행되는 단일 명령을 실행할 수 없습니다(예: 디렉터리 변경 또는 환경 변수 설정). 이 제한 사항을 해결하려면 이 문제를 해결하는 버전 0.2를 사용하는 것이 좋습니다. buildspec 버전 0.1을 사용해야 하는 경우 빌드 환경의 쉘 및 명령 단원의 접근 방식을 따르는 것이 좋습니다.

phases/*/run-as

선택적 시퀀스. 해당 명령을 실행하는 Linux 사용자를 지정하려면 빌드 단계에 사용합니다. buildspec 파일의 상단에 모든 명령에 대해 전역적으로 `run-as`도 지정할 경우 단계 수준 사용자가 우선 적용됩니다. 예를 들어 전체적으로 `run-as`에서 User-1을 지정하고 해당 `install` 단계에 대해서만 `run-as` 명령문이 User-2를 지정하는 경우, 해당 `install` 단계의 명령이 User-2로 실행되는 것을 제외하고 buildspec 파일의 모든 명령은 User-1으로 실행됩니다.

phases/*/on-failure

선택적 시퀀스. 단계 중에 오류가 발생할 경우 수행할 작업을 지정합니다. 다음 값 중 하나일 수 있습니다:

- `ABORT` - 빌드를 중단합니다.
- `CONTINUE` - 다음 단계를 계속 진행합니다.

이 속성을 지정하지 않으면 오류 프로세스는 빌드 단계 진행에 표시된 전환 단계를 따릅니다.

phases/*/finally

선택적 블록. `finally` 블록에 지정된 명령은 `commands` 블록의 명령 후에 실행됩니다. `finally` 블록의 명령은 `commands` 블록의 명령이 실패할 경우에도 실행됩니다. 예를 들어 `commands` 블록에 세 개의 명령이 포함되어 있는데 첫 번째 명령이 실패하면 나머지 두 명령을 CodeBuild 건너뛰고 블록에서 명령을 실행합니다. `finally commands` 및 `finally` 블록의 모든 명령이 성공적으로 실행되면 단계가 성공합니다. 어느 단계의 명령이 하나라도 실패하면 그 단계는 실패합니다.

허용되는 빌드 단계 이름은 다음과 같습니다.

phases/install

선택적 시퀀스. 설치 중에 CodeBuild 실행되는 명령 (있는 경우) 을 나타냅니다. 빌드 환경에서 패키지를 설치하는 경우에만 `install` 단계를 사용하는 것이 좋습니다. 예를 들어, Mocha나 RSpec 같은 코드 테스팅 프레임워크를 설치하기 위해 이 단계를 사용할 수 있습니다.

phases/install/runtime-versions

선택적 시퀀스. 런타임 버전은 Ubuntu 표준 이미지 5.0 이상 및 Amazon Linux 2 표준 이미지 4.0 이상에서 지원됩니다. 지정된 경우 적어도 하나의 실행 시간이 이 섹션에 포함되어야 합니다. 특정 버전을 사용하여 런타임을 지정하고, 메이저 버전 다음에 메이저 버전을 최신 마이너 버전으로 사용하도록 지정하거나`latest`, 최신 메이저 버전과 마이너 버전 (예: `ruby: 3.2nodejs: 18.x`, 또는`java: latest`) 을 사용하도록 지정합니다. `.x` CodeBuild 숫자나 환경 변수를 사용하여 실행 시간을 지정할 수 있습니다. 예를 들어 Amazon Linux 2 표준 이미지

4.0을 사용하는 경우 다음은 Java 버전 17, Python 버전 3의 최신 부 버전, Ruby 환경 변수에 포함된 버전이 설치되도록 지정합니다. 자세한 정보는 Docker 이미지 제공: CodeBuild을 참조하세요.

```
phases:
  install:
    runtime-versions:
      java: corretto8
      python: 3.x
      ruby: "$MY_RUBY_VAR"
```

빌드 사양 파일의 `runtime-versions` 섹션에서 하나 이상의 런타임을 지정할 수 있습니다. 런타임이 다른 런타임에 종속되는 경우 빌드 사양 파일에서 종속 런타임을 지정할 수도 있습니다. buildspec 파일에 런타임을 지정하지 않는 경우 사용하는 이미지에서 사용할 수 있는 기본 런타임을 CodeBuild 선택합니다. 하나 이상의 런타임을 지정하는 경우 는 해당 런타임만 사용합니다. CodeBuild 종속 런타임이 지정되지 않은 경우 종속 런타임을 자동으로 선택해 봅니다. CodeBuild

지정된 두 실행 시간이 충돌하면 빌드가 실패합니다. 예를 들어 `android: 29`와 `java: openjdk11`이 충돌하므로 둘 다 지정하면 빌드가 실패합니다.

사용 가능한 런타임에 대한 자세한 내용은 사용 가능한 런타임 섹션을 참조하세요.

> **Note**
>
> `runtime-versions` 섹션을 지정하고 Ubuntu 표준 이미지 2.0 이상 또는 Amazon Linux 2(AL2) 표준 이미지 1.0 이상 외의 이미지를 사용하는 경우, 빌드에서 “`Skipping install of runtimes. Runtime version selection is not supported by this build image`”라는 경고가 발생합니다.

phases/install/commands

선택적 시퀀스. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 설치 중에 CodeBuild 실행되는 단일 명령을 나타냅니다. CodeBuild 나열된 순서대로 각 명령을 한 번에 하나씩 처음부터 끝까지 실행합니다.

phases/pre_build

선택적 시퀀스. 빌드 전에 CodeBuild 실행되는 명령 (있는 경우) 을 나타냅니다. 예를 들어, Amazon ECR에 로그인하기 위해 이 단계를 사용할 수 있습니다. 또는 npm 종속성을 설치할 수도 있습니다.

phases/pre_build/commands

`pre_build`를 지정한 경우 필수 시퀀스입니다. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 빌드 전에 CodeBuild 실행되는 단일 명령을 나타냅니다. CodeBuild나열된 순서대로 각 명령을 한 번에 하나씩 처음부터 끝까지 실행합니다.

phases/build

선택적 시퀀스. 빌드 중에 CodeBuild 실행되는 명령 (있는 경우) 을 나타냅니다. 예를 들어, Mocha, RSpec 또는 sbt를 실행하기 위해 이 단계를 사용할 수 있습니다.

phases/build/commands

`build`를 지정한 경우 필수입니다. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 빌드 중에 CodeBuild 실행되는 단일 명령을 나타냅니다. CodeBuild 나열된 순서대로 처음부터 끝까지 각 명령을 한 번에 하나씩 실행합니다.

phases/post_build

선택적 시퀀스. 빌드 후에 CodeBuild 실행되는 명령 (있는 경우) 을 나타냅니다. 예를 들어, Maven을 사용하여 빌드 아티팩트를 JAR 또는 WAR 파일에 패키지할 수 있으며, Amazon ECR에 도커 이미지를 푸시할 수도 있습니다. 그런 다음, Amazon SNS를 통해 빌드 알림을 전송할 수도 있습니다.

phases/post_build/commands

`post_build`를 지정한 경우 필수입니다. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 빌드 후에 CodeBuild 실행되는 단일 명령을 나타냅니다. CodeBuild 나열된 순서대로 처음부터 끝까지 각 명령을 한 번에 하나씩 실행합니다.

보고서

report-group-name-or-arn

선택적 시퀀스. 보고서를 전송할 보고서 그룹을 지정합니다. 프로젝트에는 최대 5개의 보고서 그룹이 포함될 수 있습니다. 기존 보고서 그룹의 ARN 또는 새 보고서 그룹의 이름을 지정합니다. 이름을 지정하는 경우 프로젝트 이름과 형식에 지정된 이름을 사용하여 보고서 그룹을 CodeBuild 생성합니다. `<project-name>-<report-group-name>` 보고서 그룹 이름은 다음과 같은 buildspec의 환경 변수를 사용하여 설정할 수도 있습니다. `$REPORT_GROUP_NAME` 자세한 정보는 보고서 그룹 이름 지정을 참조하세요.

reports/<report-group>/files

필수 시퀀스. 보고서에 의해 생성된 테스트 결과의 원시 데이터가 포함된 위치를 나타냅니다. 스칼라 시퀀스를 포함하며, 각 스칼라는 원래 빌드 위치 또는 설정된 경우 를 기준으로 테스트 파일을 찾을 CodeBuild 수 있는 별도의 위치를 나타냅니다. base-directory 위치에는 다음이 포함될 수 있습니다.

- 단일 파일(예: my-test-report-file.json).
- 하위 디렉터리의 단일 파일입니다(예: *my-subdirectory*/my-test-report-file.json 또는 *my-parent-subdirectory*/*my-subdirectory*/my-test-report-file.json).
- '**/*'는 모든 파일을 재귀적으로 나타냅니다.
- *my-subdirectory*/*는 *my-subdirectory*라는 하위 디렉터리에 있는 모든 파일을 나타냅니다.
- *my-subdirectory*/**/*는 *my-subdirectory*라는 하위 디렉터리에서 시작하는 모든 파일을 재귀적으로 나타냅니다.

reports/<report-group>/file-format

선택적 매핑. 보고서 파일 형식을 나타냅니다. 지정하지 않으면 JUNITXML가 사용됩니다. 이 값은 대소문자를 구분하지 않습니다. 가능한 값은 다음과 같습니다.

테스트 보고서

CUCUMBERJSON

Cucumber JSON

JUNITXML

JUnit XML

NUNITXML

NUnit XML

NUNIT3XML

NUnit 3 XML

TESTNGXML

TestNG XML

VISUALSTUDIOTRX

Visual Studio TRX

코드 범위 보고서

`CLOVERXML`

Clover XML

`COBERTURAXML`

Cobertura XML

`JACOCOXML`

JaCoCo XML

`SIMPLECOV`

SimpleCov JSON

> **Note**
>
> CodeBuild 심플코브-json이 아닌 simplecov에서 생성된 JSON 코드 커버리지 보고서를 수락합니다.

보고서/<report-group>/기본 디렉터리

선택적 매핑. 원본 빌드 위치를 기준으로 원시 테스트 파일을 찾을 위치를 결정하는 데 사용하는 하나 이상의 최상위 디렉터리를 나타냅니다. CodeBuild

reports/<report-group>/discard-paths

선택 사항입니다. 보고서 파일 디렉터리가 출력에서 평면화되는지 여부를 지정합니다. 이 값이 지정되지 않거나 no를 포함하는 경우 보고서 파일은 디렉터리 구조가 손상되지 않은 상태로 출력됩니다. yes가 포함된 경우 모든 테스트 파일이 동일한 출력 디렉터리에 배치됩니다. 예를 들어, 테스트 결과에 대한 경로가 `com/myapp/mytests/TestResult.xml`인 경우 yes를 지정하면 이 파일이 `/TestResult.xml`에 배치됩니다.

artifacts

선택적 시퀀스. 빌드 출력을 찾을 CodeBuild 수 있는 위치와 S3 출력 버킷에 업로드하기 위해 빌드 출력을 CodeBuild 준비하는 방법에 대한 정보를 나타냅니다. 도커 이미지를 빌드하여 Amazon ECR에 푸시하는 경우 또는 소스 코드에 단위 테스트만 실행하고 빌드는 하지 않는 경우 등에는 이 시퀀스가 필요하지 않습니다.

> **Note**
>
> Amazon S3 메타데이터에는 Amazon S3에 아티팩트를 게시하는 buildArn CodeBuild 빌드의 이름이 x-amz-meta-codebuild-buildarn 포함된 CodeBuild 헤더가 있습니다. 알림에 대한 소스 추적을 허용하고 아티팩트가 생성된 빌드를 참조할 수 있도록 하기 위해 buildArn이 추가되었습니다.

artifacts/files

필수 시퀀스. 빌드 환경의 빌드 출력 결과물을 포함하는 위치를 나타냅니다. 스칼라 시퀀스를 포함하며, 각 스칼라는 원래 빌드 위치 또는 기본 디렉토리 (설정된 경우) 를 기준으로 빌드 출력 아티팩트를 찾을 CodeBuild 수 있는 별도의 위치를 나타냅니다. 위치에는 다음이 포함될 수 있습니다.

- 단일 파일(예: my-file.jar).
- 하위 디렉터리의 단일 파일입니다(예: *my-subdirectory*/my-file.jar 또는 *my-parent-subdirectory*/*my-subdirectory*/my-file.jar).
- '**/*'는 모든 파일을 재귀적으로 나타냅니다.
- *my-subdirectory*/*는 *my-subdirectory*라는 하위 디렉터리에 있는 모든 파일을 나타냅니다.
- *my-subdirectory*/**/*는 *my-subdirectory*라는 하위 디렉터리에서 시작하는 모든 파일을 재귀적으로 나타냅니다.

빌드 출력 아티팩트 위치를 지정하면 빌드 환경에서 원래 빌드 위치를 찾을 CodeBuild 수 있습니다. 빌드 아티팩트 출력 위치 앞에 원래 빌드 위치 경로를 추가하거나 ./ 같은 것을 지정하지 않아도 됩니다. 이 위치에 대한 경로를 알고 싶으면 빌드 중에 echo $CODEBUILD_SRC_DIR과 같은 명령을 실행하면 됩니다. 각 빌드 환경의 위치는 약간씩 다를 수 있습니다.

artifacts/name

선택적 이름. 빌드 아티팩트의 이름을 지정합니다. 이 이름은 다음 중 하나에 해당될 때 사용됩니다.

- CodeBuild API를 사용하여 빌드를 생성하면 프로젝트가 업데이트되거나, 프로젝트가 생성되거나, 빌드가 시작될 때 ProjectArtifacts 객체에 overrideArtifactName 플래그가 설정됩니다.
- CodeBuild 콘솔을 사용하여 빌드를 만들고, buildspec 파일에 이름을 지정하고, 프로젝트를 만들거나 업데이트할 때 시맨틱 버전 관리 활성화를 선택합니다. 자세한 정보는 빌드 프로젝트 만들기(콘솔)을 참조하세요.

빌드할 때 계산되는 buildspec 파일에서 이름을 지정할 수 있습니다. buildspec 파일에 지정된 이름은 Shell 명령 언어를 사용합니다. 예를 들어 결과물 이름이 항상 고유하도록 날짜와 시간을 결과물 이름에 추가할 수 있습니다. 고유한 결과물 이름을 사용하면 결과물을 덮어쓰지 않을 수 있습니다. 자세한 정보는 Shell 명령 언어를 참조하십시오.

- 다음은 아티팩트가 생성된 날짜가 추가된 아티팩트 이름의 예입니다

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

- 다음은 환경 변수를 사용하는 아티팩트 이름의 예입니다. CodeBuild 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

- 다음은 아티팩트의 작성 날짜가 추가된 CodeBuild 환경 변수를 사용하는 아티팩트 이름의 예입니다.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: $AWS_REGION-$(date +%Y-%m-%d)
```

이름에 경로 정보를 추가하여 이름의 경로를 기준으로 명명된 아티팩트가 디렉터리에 배치되도록 할 수 있습니다. 이 예제에서는 빌드 아티팩트가 출력의 builds/<build number>/my-artifacts 아래에 배치됩니다.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: builds/$CODEBUILD_BUILD_NUMBER/my-artifacts
```

artifacts/discard-paths

선택 사항입니다. 빌드 아티팩트 디렉터리가 출력에서 평면화되는지 여부를 지정합니다. 이 값이 지정되지 않거나 no를 포함하는 경우 빌드 아티팩트는 디렉터리 구조가 손상되지 않은 상태로 출력됩니다. yes가 포함된 경우 모든 빌드 아티팩트가 동일한 출력 디렉터리에 배치됩니다. 예를 들어, 빌드 출력 아티팩트의 파일 경로가 com/mycompany/app/HelloWorld.java인 경우 yes를 지정하면 이 파일이 /HelloWorld.java에 배치됩니다.

artifacts/base-directory

선택적 매핑. 원본 빌드 위치를 기준으로 빌드 출력 아티팩트에 포함할 파일 및 하위 디렉터리를 결정하는 데 CodeBuild 사용하는 하나 이상의 최상위 디렉터리를 나타냅니다. 유효한 값으로는 다음이 포함됩니다.

- 단일 최상위 디렉터리입니다(예: my-directory).
- 'my-directory*'는 이름이 my-directory로 시작하는 모든 최상위 디렉터리를 나타냅니다.

빌드 출력 결과물에는 이 최상위 디렉터리가 포함되지 않으며, 파일 및 하위 디렉터리만 포함됩니다.

포함할 파일 및 하위 디렉터리를 보다 더 제한하려면 files 및 discard-paths를 사용하면 됩니다. 예를 들어 다음 디렉터리구조에서

```
.
### my-build-1
#   ### my-file-1.txt
### my-build-2
    ### my-file-2.txt
```

```
    ### my-subdirectory
        ### my-file-3.txt
```

다음 artifacts 시퀀스에 대해:

```
artifacts:
  files:
    - '*/my-file-3.txt'
  base-directory: my-build-2
```

다음 하위 디렉터리 및 파일이 빌드 출력 결과물에 포함됩니다.

```
.
### my-subdirectory
    ### my-file-3.txt
```

다음 artifacts 시퀀스 동안

```
artifacts:
  files:
    - '**/*'
  base-directory: 'my-build*'
  discard-paths: yes
```

다음 파일이 빌드 출력 결과물에 포함됩니다.

```
.
### my-file-1.txt
### my-file-2.txt
### my-file-3.txt
```

artifacts/exclude-paths

선택적 매핑. 빌드 아티팩트에서 CodeBuild 제외할 base-directory 상대 경로를 하나 이상 나타냅니다. 별표(*) 문자는 폴더의 경계를 넘지 않고 0개 이상의 이름 구성 요소 문자와 해당합니다. 이중 별표(**)는 모든 디렉터리에서 이름 구성 요소의 0개 이상 문자와 일치합니다.

exclude-paths의 예는 다음과 같습니다.

- 모든 디렉터리에서 파일을 제외하려면: "**/*file-name*/**/*"
- 모든 dot 폴더를 제외하려면: "**/.*/**/*"

- 모든 dot 파일을 제외하려면: `"**/.*"`

artifacts/enable-symlinks

선택 사항입니다. 출력 유형이 `ZIP`인 경우 내부 심볼 링크를 ZIP 파일에 보존할지 여부를 지정합니다. 이 파일에 `yes`가 포함된 경우 소스의 모든 내부 심볼 링크가 아티팩트 ZIP 파일에 보존됩니다.

artifacts/s3-prefix

선택 사항입니다. 아티팩트가 Amazon S3 버킷으로 출력되고 네임스페이스 유형이 `BUILD_ID`일 때 사용되는 접두사를 지정합니다. 사용될 경우 버킷의 출력 경로는 `<s3-prefix>/<build-id>/<name>.zip`입니다.

artifacts/secondary-artifacts

선택적 시퀀스. 1개 이상의 아티팩트 정의를 아티팩트 식별자와 아티팩트 정의를 연결하는 매핑으로 나타냅니다. 이 블록에서는 각 아티팩트가 프로젝트의 `secondaryArtifacts` 속성에서 정의하는 아티팩트와 일치해야 합니다. 각 정의는 위의 `artifacts` 블록과 동일한 구문을 갖습니다.

> **Note**
>
> 2차 아티팩트만 정의된 경우에도 artifacts/files 시퀀스는 항상 필요합니다.

예를 들어 프로젝트가 다음과 같은 구조라고 가정할 경우,

```
{
  "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"
    }
  ]
}
```

buildpec 파일은 다음과 비슷합니다.

```
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
```

cache

선택적 시퀀스. 캐시를 S3 캐시 버킷에 업로드하기 위해 파일을 CodeBuild 준비할 수 있는 위치에 대한 정보를 나타냅니다. 프로젝트의 캐시 유형이 `No Cache`인 경우 이 시퀀스는 필수가 아닙니다.

cache/paths

필수 시퀀스. 캐시의 위치를 나타냅니다. 스칼라 시퀀스를 포함하며, 각 스칼라는 원래 빌드 위치 또는 기본 디렉터리 (설정된 경우) 를 기준으로 빌드 출력 아티팩트를 찾을 CodeBuild 수 있는 별도의 위치를 나타냅니다. 위치에는 다음이 포함될 수 있습니다.

- 단일 파일(예: `my-file.jar`).
- 하위 디렉터리의 단일 파일입니다(예: *`my-subdirectory`*`/my-file.jar` 또는 *`my-parent-subdirectory`*`/`*`my-subdirectory`*`/my-file.jar`).
- `'**/*'`는 모든 파일을 재귀적으로 나타냅니다.
- *`my-subdirectory`*`/*`는 *`my-subdirectory`*라는 하위 디렉터리에 있는 모든 파일을 나타냅니다.
- *`my-subdirectory`*`/**/*`는 *`my-subdirectory`*라는 하위 디렉터리에서 시작하는 모든 파일을 재귀적으로 나타냅니다.

> **Important**
>
> buildspec 선언은 올바른 YAML이어야 하므로 buildspec 선언의 공백 설정이 중요합니다. buildspec 선언의 공백 수가 잘못되면 빌드가 즉시 실패할 수 있습니다. YAML 유효성 검사기를 사용하여 buildspec 선언이 올바른 YAML인지 여부를 테스트할 수 있습니다.
> 빌드 프로젝트를 만들거나 업데이트할 때 또는 AWS SDK를 사용하여 buildspec을 선언하는 경우 buildspec은 필수 공백 및 줄 바꿈 이스케이프 문자와 함께 YAML 형식으로 표현되는 단일 문자열이어야 합니다. AWS CLI다음 섹션에 예가 나와 있습니다.
> buildspec.yml AWS CodePipeline 파일 대신 CodeBuild 또는 콘솔을 사용하는 경우 해당 단계에 대한 명령만 삽입할 수 있습니다. `build` 앞에 나온 구문을 사용하는 대신, 빌드 단계 중에 실행하려는 모든 명령을 하나의 행에 나열합니다. 명령이 여러 개인 경우 각 명령을 &&로 구분합니다(예: `mvn test && mvn package`).
> buildspec.yml 파일 대신 CodeBuild or CodePipeline 콘솔을 사용하여 빌드 환경에서 빌드 출력 아티팩트의 위치를 지정할 수 있습니다. 앞에 나온 구문을 사용하는 대신, 모든 위치를 하나의 행에 나열합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `buildspec.yml, target/my-app.jar`).

buildspec 예제

다음은 buildspec.yml 파일의 예입니다.

```
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/**/*'
```

다음은, 또는 SDK와 함께 사용할 수 있는 이전 빌드스펙의 예시입니다. 이 예시는 단일 문자열로 표현됩니다. AWS CLI 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/**/*'"
```

다음은 또는 콘솔과 함께 사용할 수 있는 build 단계의 명령 예시입니다. CodeBuild CodePipeline

```
echo Build started on `date` && mvn install
```

아래 예에서

- JAVA_HOME의 키 및 /usr/lib/jvm/java-8-openjdk-amd64의 값이 있는 일반 텍스트의 사용자 지정 환경 변수가 설정됩니다.
- Amazon EC2 Systems Manager Parameter Store에 저장된 dockerLoginPassword라는 사용자 지정 환경 변수는 나중에 LOGIN_PASSWORD 키를 사용하여 빌드 명령에서 참조됩니다.
- 이러한 빌드 단계 이름은 변경할 수 없습니다. 이 예에서 실행될 명령은 apt-get update -y 및 apt-get install -y maven(Apache Maven을 설치하는 데 사용), mvn install(소스 코드를 빌드 출력 아티팩트로 컴파일, 테스트 및 패키징하고 빌드 출력 아티팩트를 내부 리포지토리에 설치하는 데 사용), docker login(Amazon EC2 Systems Manager Parameter Store에 설정한 사용자 지정 환경 변수 dockerLoginPassword의 값에 해당하는 암호로 Docker에 로그인하는 데 사용) 및 여러 echo 명령입니다. echo명령 CodeBuild 실행 방법 및 실행 순서를 보여 주는 명령이 여기에 포함되어 있습니다.

- `files`는 빌드 출력 위치에 업로드할 파일을 나타냅니다. 이 예시에서는 단일 파일을 CodeBuild `messageUtil-1.0.jar` 업로드합니다. `messageUtil-1.0.jar` 파일은 빌드 환경의 `target`이라는 상대적 디렉터리에서 찾을 수 있습니다. `discard-paths: yes`가 지정되어 있으므로, `messageUtil-1.0.jar`가 바로 업로드됩니다(중간의 `target` 디렉터리를 거치지 않음). 파일 이름 `messageUtil-1.0.jar` 및 상대적 디렉터리 이름 `target`은 Apache Maven이 이 예제에서만 빌드 출력 결과물을 생성 및 저장하는 방식에 따라 달라집니다. 사용자 자체 시나리오에서는 이러한 파일 이름과 디렉터리가 다릅니다.
- `reports`는 빌드 중에 보고서를 생성하는 두 개의 보고서 그룹을 나타냅니다.
  - `arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1`는 보고서 그룹의 ARN을 지정합니다. 테스트 프레임워크에 의해 생성된 테스트 결과는 `target/tests/reports` 디렉터리에 있습니다. 파일 형식은 `JunitXml`이고 테스트 결과가 포함된 파일에서 경로가 제거되지 않습니다.
  - `reportGroupCucumberJson`는 새 보고서 그룹을 지정합니다. 프로젝트 이름이 `my-project`인 경우 빌드가 실행될 때 이름이 `my-project-reportGroupCucumberJson`인 보고서 그룹이 생성됩니다. 테스트 프레임워크에 의해 생성된 테스트 결과가 `cucumber/target/cucumber-tests.xml`에 있습니다. 테스트 파일 형식은 `CucumberJson`이고 테스트 결과가 포함된 파일에서 경로가 제거됩니다.

buildspec 버전

다음 표에는 buildspec 버전과 버전 간의 변경 사항이 나열되어 있습니다.

버전	변경
0.2	• `environment_variables` 의 이름이 `env`로 다시 지정되었습니다. • `plaintext` 의 이름이 `variables` 로 다시 지정되었습니다. • `artifacts` 의 `type` 속성이 사용 중단되었습니다. • 버전 0.1에서는 빌드 환경에 있는 기본 셸의 개별 인스턴스에서 각 빌드 명령을 AWS CodeBuild 실행합니다. 버전 0.2에서는 빌드 환경의 기본 셸의 동일한 인스턴스에서 모든 빌드 명령을 CodeBuild 실행합니다.

버전	변경
0.1	이는 빌드 사양 형식의 초기 정의입니다.

배치 빌드 buildspec 참조

이 주제에는 배치 빌드 속성에 대한 buildspec 참조가 포함되어 있습니다.

배치

선택적 매핑. 프로젝트에 대한 배치 빌드 설정입니다.

배치/빠른 실패

선택 사항. 하나 이상의 빌드 태스크가 실패할 경우 배치 빌드의 동작을 지정합니다.

`false`

기본값입니다. 실행 중인 모든 빌드가 완료됩니다.

`true`

빌드 태스크 중 하나가 실패하면 실행 중인 모든 빌드가 중지됩니다.

기본적으로 모든 배치 빌드 태스크는 buildspec 파일에 지정된 env 및 phases와 같은 빌드 설정으로 실행됩니다. batch/*<batch-type>*/buildspec 파라미터에 다른 env 값이나 다른 buildspec 파일을 지정하여 기본 빌드 설정을 재정의할 수 있습니다.

batch 속성의 내용은 지정된 배치 빌드 유형에 따라 달라집니다. 가능한 배치 빌드 유형은 다음과 같습니다.

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

batch/build-graph

빌드 그래프를 정의합니다. 빌드 그래프는 일괄 처리의 다른 태스크에 종속되는 일련의 태스크를 정의합니다. 자세한 내용은 빌드 그래프 섹션을 참조하세요.

이 요소에는 빌드 태스크의 배열이 포함되어 있습니다. 각 빌드 태스크는 다음 속성을 포함합니다.

identifier

필수. 태스크의 식별자입니다.

buildspec

선택 사항. 태스크에 사용할 buildspec 파일의 경로 및 파일 이름입니다. 이 파라미터를 지정하지 않으면 현재 buildspec인 파일이 사용됩니다.

debug-session

선택 사항. 이 배치 빌드에 세션 디버깅을 활성화할지를 여부를 나타내는 부울 값입니다. 세션 디버깅에 대한 자세한 내용은 Session Manager에서 실행 중인 빌드 보기 섹션을 참조하세요.

`false`

세션 디버깅이 비활성화되었습니다.

`true`

세션 디버깅이 활성화되었습니다.

depend-on

선택 사항. 이 태스크가 의존하는 태스크 식별자의 배열입니다. 이 태스크는 이러한 태스크가 완료될 때까지 실행되지 않습니다.

env

선택 사항. 태스크에 대한 빌드 환경 재정의입니다. 여기에는 다음 속성이 포함됩니다.

compute-type

태스크에 사용할 컴퓨팅 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”의 computeType을 참조하세요.

image

태스크에 사용할 이미지의 식별자입니다. 가능한 값은 the section called “Docker 이미지 제공: CodeBuild”의 이미지 식별자를 참조하세요.

privileged-mode

Docker 컨테이너 내부에서 Docker 대몬(daemon)을 실행할지 여부를 나타내는 부울 값입니다. 빌드 프로젝트가 도커 이미지를 빌드하는 데 사용되는 경우에만 `true`로 설정합니다. 그렇지 않으면 도커 데몬과 상호 작용을 시도하는 빌드가 실패합니다. 기본 설정은 `false`입니다.

type

태스크에 사용할 환경 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”에서 환경 유형을 참조하세요.

variables

빌드 환경에 표시될 환경 변수입니다. 자세한 내용은 env/variables 섹션을 참조하세요.

ignore-failure

선택 사항. 이 빌드 태스크의 실패를 무시할 수 있는지 여부를 나타내는 부울 값입니다.

`false`

기본값입니다. 이 빌드 태스크가 실패하면 배치 빌드가 실패합니다.

`true`

이 빌드 태스크가 실패하더라도 배치 빌드는 여전히 성공할 수 있습니다.

다음은 빌드 그래프 buildSpec 항목의 예제입니다.

```
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

빌드 목록을 정의합니다. 빌드 목록은 병렬로 실행되는 여러 태스크를 정의하는 데 사용됩니다. 자세한 내용은 빌드 목록 섹션을 참조하세요.

이 요소에는 빌드 태스크의 배열이 포함되어 있습니다. 각 빌드 태스크는 다음 속성을 포함합니다.

identifier

필수. 태스크의 식별자입니다.

buildspec

선택 사항. 태스크에 사용할 buildspec 파일의 경로 및 파일 이름입니다. 이 파라미터를 지정하지 않으면 현재 buildspec인 파일이 사용됩니다.

debug-session

선택 사항. 이 배치 빌드에 세션 디버깅을 활성화할지를 여부를 나타내는 부울 값입니다. 세션 디버깅에 대한 자세한 내용은 Session Manager에서 실행 중인 빌드 보기 섹션을 참조하세요.

`false`

세션 디버깅이 비활성화되었습니다.

`true`

세션 디버깅이 활성화되었습니다.

env

선택 사항. 태스크에 대한 빌드 환경 재정의입니다. 여기에는 다음 속성이 포함됩니다.

compute-type

태스크에 사용할 컴퓨팅 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”의 computeType을 참조하세요.

image

태스크에 사용할 이미지의 식별자입니다. 가능한 값은 the section called “Docker 이미지 제공: CodeBuild”의 이미지 식별자를 참조하세요.

privileged-mode

Docker 컨테이너 내부에서 Docker 대몬(daemon)을 실행할지 여부를 나타내는 부울 값입니다. 빌드 프로젝트가 도커 이미지를 빌드하는 데 사용되는 경우에만 `true`로 설정합니다. 그렇지 않으면 도커 데몬과 상호 작용을 시도하는 빌드가 실패합니다. 기본 설정은 `false`입니다.

유형

태스크에 사용할 환경 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”에서 환경 유형을 참조하세요.

variables

빌드 환경에 표시될 환경 변수입니다. 자세한 내용은 env/variables 섹션을 참조하세요.

ignore-failure

선택 사항. 이 빌드 태스크의 실패를 무시할 수 있는지 여부를 나타내는 부울 값입니다.

`false`

기본값입니다. 이 빌드 태스크가 실패하면 배치 빌드가 실패합니다.

`true`

이 빌드 태스크가 실패하더라도 배치 빌드는 여전히 성공할 수 있습니다.

다음은 빌드 목록 buildSpec 항목의 예제입니다.

```
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

빌드 매트릭스를 정의합니다. 빌드 매트릭스는 병렬로 실행되는 다양한 구성의 태스크를 정의합니다. CodeBuild는 가능한 각 구성 조합에 대해 별도의 빌드를 생성합니다. 자세한 내용은 빌드 매트릭스 섹션을 참조하세요.

static

정적 속성은 모든 빌드 태스크에 적용됩니다.

ignore-failure

선택 사항. 이 빌드 태스크의 실패를 무시할 수 있는지 여부를 나타내는 부울 값입니다.

`false`

기본값입니다. 이 빌드 태스크가 실패하면 배치 빌드가 실패합니다.

`true`

이 빌드 태스크가 실패하더라도 배치 빌드는 여전히 성공할 수 있습니다.

env

선택 사항. 모든 태스크에 대한 빌드 환경 재정의입니다.

privileged-mode

Docker 컨테이너 내부에서 Docker 대몬(daemon)을 실행할지 여부를 나타내는 부울 값입니다. 빌드 프로젝트가 도커 이미지를 빌드하는 데 사용되는 경우에만 `true`로 설정합니다. 그렇지 않으면 도커 데몬과 상호 작용을 시도하는 빌드가 실패합니다. 기본 설정은 `false`입니다.

type

태스크에 사용할 환경 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”에서 환경 유형을 참조하세요.

dynamic

동적 속성은 빌드 매트릭스를 정의합니다.

buildspec

선택 사항. 이러한 태스크에 사용할 buildspec 파일의 경로와 파일 이름을 포함하는 배열입니다. 이 파라미터를 지정하지 않으면 현재 buildspec인 파일이 사용됩니다.

env

선택 사항. 이러한 태스크에 대해 빌드 환경이 재정의됩니다.

compute-type

이러한 태스크에 사용할 컴퓨팅 유형의 식별자가 포함된 배열입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”의 computeType을 참조하세요.

image

이러한 태스크에 사용할 이미지의 식별자가 포함된 배열입니다. 가능한 값은 the section called “Docker 이미지 제공: CodeBuild”의 이미지 식별자를 참조하세요.

variables

이러한 태스크에 대한 빌드 환경에 표시될 환경 변수를 포함하는 배열입니다. 자세한 내용은 env/variables 섹션을 참조하세요.

다음은 빌드 매트릭스 buildSpec 항목의 예제입니다.

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
            - VALUE3
```

자세한 내용은 빌드 매트릭스 섹션을 참조하세요.

AWS CodeBuild의 빌드 환경 참조

AWS CodeBuild를 호출하여 빌드를 실행할 때 빌드 환경에 대한 정보를 제공해야 합니다. 빌드 환경은 CodeBuild가 빌드를 실행하는 데 사용하는 운영 체제, 프로그래밍 언어 런타임 및 도구의 조합을 나타냅니다. 빌드 환경의 작동 방식에 대한 자세한 내용은 CodeBuild 작동 방식 섹션을 참조하세요.

빌드 환경에는 도커 이미지가 들어 있습니다. 자세한 내용은 Docker Docs 웹 사이트에서 Docker Glossary를 참조하십시오.

빌드 환경에 대한 정보를 CodeBuild에 제공할 때는 지원되는 리포지토리 유형의 도커 이미지 식별자를 지정해야 합니다. 여기에는 CodeBuild 도커 이미지 리포지토리, Docker Hub에서 공개적으로 사용 가

능한 이미지, AWS 계정에 액세스 권한이 주어진 Amazon Elastic Container Registry(Amazon ECR) 리포지토리가 포함됩니다.

- CodeBuild 도커 이미지 리포지토리에 저장된 도커 이미지는 해당 서비스에 사용하도록 최적화되어 있으므로 이를 사용하는 것이 좋습니다. 자세한 내용은 Docker 이미지 제공: CodeBuild 섹션을 참조하세요.
- Docker Hub에 저장되어 있는 공개적으로 사용 가능한 도커 이미지 식별자를 가져오려면 Docker Docs 웹 사이트의 리포지토리 검색을 참조하십시오.
- AWS 계정의 Amazon ECR 리포지토리에 저장되어 있는 도커 이미지로 작업하는 방법을 알아보려면 Amazon ECR 샘플 섹션을 참조하세요.

도커 이미지 식별자 외에도 빌드 환경에서 사용할 컴퓨팅 리소스 세트를 지정할 수 있습니다. 자세한 내용은 빌드 환경 컴퓨팅 모드 및 유형 섹션을 참조하세요.

주제

Docker 이미지 제공: CodeBuild

지원되는 이미지는 에서 사용할 수 있는 이미지의 최신 메이저 버전이며 마이너 CodeBuild 및 패치 버전 업데이트로 업데이트됩니다. CodeBuild 지원되는 이미지를 머신의 Amazon 머신 이미지 (AMI)에 캐싱하여 빌드의 프로비저닝 기간을 최적화합니다. 캐싱의 이점을 활용하고 빌드의 프로비저닝 기간을 최소화하려면 CodeBuild 콘솔의 이미지 버전 섹션에서 이 런타임 버전에 대해 더 세분화된 버전 (예: 항상 최신 이미지 사용) 을 선택합니다. `aws/codebuild/amazonlinux2-x86_64-standard:4.0-1.0.0`

CodeBuild Docker 이미지 목록을 자주 업데이트하여 최신 이미지를 추가하고 이전 이미지는 더 이상 사용하지 않습니다. 최신 목록을 가져오려면 다음 중 하나를 수행합니다.

- CodeBuild 콘솔의 빌드 프로젝트 생성 마법사 또는 빌드 프로젝트 편집 페이지에서 환경 이미지에 대해 관리 이미지를 선택합니다. 운영 체제, 런타임 및 런타임 버전 드롭다운 목록에서 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 또는 빌드 프로젝트 설정 변경(콘솔)을 참조하세요.

- 의 AWS CLI경우 다음 `list-curated-environment-images` 명령을 실행합니다.

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

- AWS SDK의 경우 대상 프로그래밍 언어에 맞게 `ListCuratedEnvironmentImages` 작업을 호출하십시오. 자세한 내용은 AWS SDK 및 도구 참조을 참조하세요.

Windows Server Core 2019 플랫폼의 기본 이미지는 다음 리전에서만 사용할 수 있습니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(오레곤)
- 유럽(아일랜드)

EC2 컴퓨팅 이미지

AWS CodeBuild EC2 컴퓨팅에 사용할 수 있는 다음과 같은 Docker 이미지를 지원합니다. CodeBuild

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:4.0`	al2/standard/4.0
Amazon Linux 2023	`aws/codebuild/amazonlinux2-x86_64-standard:5.0`	al2/standard/5.0
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:corretto8`	al2/standard/corretto8
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:corretto11`	al2/standard/corretto11

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux2-aarch64-standard:2.0`	al2/aarch64/standard/2.0
Amazon Linux 2023	`aws/codebuild/amazonlinux2-aarch64-standard:3.0`	al2/aarch64/standard/3.0
Ubuntu 20.04	`aws/codebuild/standard:5.0`	ubuntu/standard/5.0
Ubuntu 22.04	`aws/codebuild/standard:6.0`	ubuntu/standard/6.0
Ubuntu 22.04	`aws/codebuild/standard:7.0`	ubuntu/standard/7.0
Windows Server Core 2019	`aws/codebuild/windows-base:2019-1.0`	N/A
Windows Server Core 2019	`aws/codebuild/windows-base:2019-2.0`	N/A
Windows Server Core 2019	`aws/codebuild/windows-base:2019-3.0`	N/A
윈도우 서버 코어 2022	`aws/codebuild/windows-base:2022-1.0`	N/A

Lambda 컴퓨팅 이미지

AWS CodeBuild 컴퓨팅에 사용할 수 있는 다음과 같은 Docker 이미지를 지원합니다. AWS Lambda CodeBuild

aarch64 아키텍처

플랫폼	이미지 식별자	정의
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet6	al-lambda/aarch64/dotnet6
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet8	al-lambda/aarch64/dotnet8
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.21	al-lambda/aarch64/go1.21
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto11	al-lambda/aarch64/corretto11
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto17	al-lambda/aarch64/corretto17
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto21	al-lambda/aarch64/corretto21
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs18	al-lambda/aarch64/nodejs18

플랫폼	이미지 식별자	정의
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs20`	al-lambda/aarch64/nodejs20
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.11`	al-lambda/aarch64/python3.11
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.12`	al-lambda/aarch64/python3.12
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.2`	al-lambda/aarch64/ruby3.2

x86_64 아키텍처

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet6`	al-lambda/x86_64/dotnet6
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet8`	알 람다/x86_64/dotnet8

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.21`	al-lambda/x86_64/go1.21
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto11`	al-lambda/x86_64/corretto11
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto17`	al-lambda/x86_64/corretto17
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21`	al-lambda/x86_64/corretto21
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs18`	al-lambda/x86_64/nodejs18
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20`	al-lambda/x86_64/nodejs20
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.11`	al-lambda/x86_64/python3.11

플랫폼	이미지 식별자	정의
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12`	al-lambda/x86_64/python3.12
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.2`	al-lambda/x86_64/ruby3.2

더 이상 사용되지 않는 이미지

더 이상 사용되지 않는 이미지는 에 의해 더 이상 캐시되거나 업데이트되지 않는 이미지입니다. CodeBuild 더 이상 사용되지 않는 이미지는 더 이상 부 버전 업데이트나 패치 버전 업데이트를 받지 않으며, 더 이상 업데이트되지 않으므로 이미지를 사용하는 것이 안전하지 않을 수 있습니다. CodeBuild 프로젝트가 이전 이미지 버전을 사용하도록 구성된 경우 프로비저닝 프로세스는 이 docker 이미지를 다운로드하고 이를 사용하여 컨테이너화된 런타임 환경을 만들어 프로비저닝 기간과 전체 빌드 기간을 늘릴 수 있습니다.

CodeBuild 다음 Docker 이미지는 더 이상 사용되지 않습니다. 이러한 이미지는 계속 사용할 수 있지만 빌드 호스트에 캐시되지 않으므로 프로비저닝 시간이 길어집니다.

플랫폼	이미지 식별자	정의	사용 중단 날짜
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:3.0`	al2/standard/3.0	2023년 5월 9일
Ubuntu 18.04	`aws/codebuild/standard:4.0`	ubuntu/standard/4.0	2023년 3월 31일
Amazon Linux 2	`aws/codebuild/amazonlinux2-`	al2/aarch64/standard/1.0	2023년 3월 31일

플랫폼	이미지 식별자	정의	사용 중단 날짜
	aarch64-standard:1.0		
Ubuntu 18.04	aws/codebuild/standard:3.0	ubuntu/standard/3.0	2022년 6월 30일
Amazon Linux 2	aws/codebuild/amazonlinux2-x86_64-standard:2.0	al2/standard/2.0	2022년 6월 30일

주제

- 사용 가능한 런타임
- 실행 시간 버전

사용 가능한 런타임

빌드 사양 파일의 `runtime-versions` 섹션에서 하나 이상의 런타임을 지정할 수 있습니다. 런타임이 다른 런타임에 종속되는 경우 빌드 사양 파일에서 종속 런타임을 지정할 수도 있습니다. buildspec 파일에 런타임을 지정하지 않은 경우 사용하는 이미지에서 사용할 수 있는 기본 런타임을 CodeBuild 선택합니다. 하나 이상의 런타임을 지정하는 경우 는 해당 런타임만 사용합니다. CodeBuild 종속 런타임이 지정되지 않은 경우 종속 런타임을 자동으로 선택해 봅니다. CodeBuild 자세한 정보는 Specify runtime versions in the buildspec file을 참조하세요.

주제

- Linux 이미지 런타임
- Windows 이미지 런타임

Linux 이미지 런타임

다음 표에는 사용 가능한 런타임과 이를 지원하는 표준 Linux 이미지가 나와 있습니다.

Ubuntu 및 Amazon Linux 플랫폼 런타임

실행 시간 이름	버전	이미지
dotnet	3.1	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0
	5.0	Ubuntu 표준:5.0
	6.0	Amazon Linux 2 x86_64 Lambda 표준:dotnet6 Amazon Linux 2 AArch64 Lambda 표준:dotnet6 Amazon Linux 2 x86_64 표준:4.0 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:6.0 Ubuntu 표준:7.0
	8.0	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
golang	1.12	Amazon Linux 2 AArch64 표준:2.0

실행 시간 이름	버전	이미지
	1.13	Amazon Linux 2 AArch64 표준:2.0
	1.14	Amazon Linux 2 AArch64 표준:2.0
	1.15	Ubuntu 표준:5.0
	1.16	Ubuntu 표준:5.0
	1.18	Amazon Linux 2 x86_64 표준:4.0 Ubuntu 표준:6.0
	1.20	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
	1.21	Amazon Linux 2 x86_64 Lambda 표준:go1.21 Amazon Linux 2 AArch64 Lambda 표준:go1.21 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0

실행 시간 이름	버전	이미지
	1.22	Amazon Linux 2023 x86_64 표준:5.0 Ubuntu 표준:7.0
java	corretto8	Amazon Linux 2 x86_64 표준:corretto8 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0 Ubuntu 표준:7.0
	corretto11	Amazon Linux 2 x86_64 표준:corretto11 Amazon Linux 2 x86_64 Lambda 표준:corretto11 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2 AArch64 Lambda 표준:corretto11 Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0 Ubuntu 표준:7.0

실행 시간 이름	버전	이미지
	corretto17	Amazon Linux 2 x86_64 Lambda 표준:corretto17 Amazon Linux 2 AArch64 Lambda 표준:corretto17 Amazon Linux 2 x86_64 표준:4.0 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:6.0 Ubuntu 표준:7.0
	corretto21	Amazon Linux 2 x86_64 Lambda 표준:corretto21 Amazon Linux 2 AArch64 Lambda 표준:corretto21 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
nodejs	10	Amazon Linux 2 AArch64 표준:2.0

실행 시간 이름	버전	이미지
	12	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0
	14	Ubuntu 표준:5.0
	16	Amazon Linux 2 x86_64 표준:4.0 Ubuntu 표준:6.0
	18	Amazon Linux 2 x86_64 Lambda 표준:nodejs18 Amazon Linux 2 AArch64 Lambda 표준:nodejs18 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
	20	Amazon Linux 2 x86_64 Lambda 표준:nodejs20 Amazon Linux 2 AArch64 Lambda 표준:nodejs20 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0

실행 시간 이름	버전	이미지
php	7.3	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0
	7.4	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0
	8.0	Ubuntu 표준:5.0
	8.1	Amazon Linux 2 x86_64 표준:4.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:6.0
	8.2	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
	8.3	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0

실행 시간 이름	버전	이미지
python	3.7	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0
	3.8	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0
	3.9	Amazon Linux 2 x86_64 표준:4.0 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0 Ubuntu 표준:7.0
	3.10	Amazon Linux 2023 x86_64 표준:5.0 Ubuntu 표준:6.0 Ubuntu 표준:7.0

실행 시간 이름	버전	이미지
	3.11	Amazon Linux 2 x86_64 Lambda 표준:python3.11 Amazon Linux 2 AArch64 Lambda 표준:python3.11 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
	3.12	Amazon Linux 2 x86_64 Lambda 표준:python3.12 Amazon Linux 2 AArch64 Lambda 표준:python3.12 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
ruby	2.6	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0
	2.7	Amazon Linux 2 AArch64 표준:2.0 Ubuntu 표준:5.0

실행 시간 이름	버전	이미지
	3.1	Amazon Linux 2 x86_64 표준:4.0 Amazon Linux 2023 x86_64 표준:5.0 Ubuntu 표준:6.0 Ubuntu 표준:7.0
	3.2	Amazon Linux 2 x86_64 Lambda 표준:ruby3.2 Amazon Linux 2 AArch64 Lambda 표준:ruby3.2 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 표준:7.0
	3.3	Amazon Linux 2023 x86_64 표준:5.0 Ubuntu 표준:7.0

Windows 이미지 런타임

Windows Server Core 2019의 기본 이미지에는 다음 실행 시간이 포함됩니다.

Windows 플랫폼 런타임

실행 시간 이름	윈도우 서버 코어 2019 표준:1.0 버전	윈도우 서버 코어 2019 표준:2.0 버전	윈도우 서버 코어 2019 표준:3.0 버전
dotnet	3.1	3.1	6.0

실행 시간 이름	윈도우 서버 코어 2019 표준:1.0 버전	윈도우 서버 코어 2019 표준:2.0 버전	윈도우 서버 코어 2019 표준:3.0 버전
	5.0	6.0 7.0	7.0 8.0
닷넷 SDK	3.1 5.0	3.1 6.0 7.0	8.0
golang	1.14	1.18	1.21
그라들	6.7	7.6	8.5
java	코레토 11	11로 수정했습니다. 17로 수정	21로 수정했습니다.
maven	3.6	3.8	3.9
nodejs	14.15	16.19	20.11
php	7.4	8.1	8.3
powershell	7.1	7.2	7.4
python	3.8	3.10	3.12
ruby	2.7	3.1	3.3

실행 시간 버전

buildspec 파일의 `runtime-versions` 섹션에서 런타임을 지정할 때 특정 버전, 특정 메이저 버전 및 최신 마이너 버전 또는 최신 버전을 지정할 수 있습니다. 다음 표에는 사용 가능한 런타임과 이를 지정하는 방법이 나와 있습니다. 모든 이미지에서 모든 런타임 버전을 사용할 수 있는 것은 아닙니다. 사용자 지정 이미지에는 런타임 버전 선택도 지원되지 않습니다. 자세한 정보는 사용 가능한 런타임을 참조하세요. 사전 설치된 런타임 버전 대신 사용자 지정 런타임 버전을 설치하여 사용하려면 을 참조하십시오사용자 지정 런타임 버전.

Ubuntu 및 Amazon Linux 2 플랫폼 런타임 버전

실행 시간 이름	버전	특정 버전	특정 메이저 버전 및 최신 마이너 버전	최신 버전
android	28	`android: 28`	`android: 28.x`	`android: latest`
	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: corretto11`	`java: corretto11.x`	

실행 시간 이름	버전	특정 버전	특정 메이저 버전 및 최신 마이너 버전	최신 버전
	corretto17	`java: corretto 7`	`java: corretto 7.x`	
	corretto21	`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`		

실행 시간 이름	버전	특정 버전	특정 메이저 버전 및 최신 마이너 버전	최신 버전
	3.11	`python: 3.11`		
	3.12	`python: 3.12`		
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`		

빌드 사양을 사용하여 빌드 단계에서 다른 구성 요소 (예: Apache Maven AWS CLI, Apache Ant, Mocha, RSpec 등) 를 설치할 수 있습니다. `install` 자세한 정보는 buildspec 예제을 참조하세요.

사용자 지정 런타임 버전

CodeBuild-managed 이미지에 사전 설치된 런타임 버전을 사용하는 대신 원하는 사용자 지정 버전을 설치하여 사용할 수 있습니다. 다음 표에는 사용 가능한 사용자 지정 런타임과 이를 지정하는 방법이 나와 있습니다.

> **Note**
>
> 사용자 지정 런타임 버전 선택은 Ubuntu 및 Amazon Linux 이미지에서만 지원됩니다.

사용자 지정 런타임 버전

실행 시간 이름	구문	예
dotnet	*<major>*.*<minor>*.*<patch>*	`5.0.408`
golang	*<major>*.*<minor>*	`1.19`

실행 시간 이름	구문	예
	<major>.*<minor>*.*<patch>*	1.19.1
java	corretto*<major>*	corretto15
nodejs	*<major>*	14
	<major>.*<minor>*	14.21
	<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

커스텀 런타임 빌드/스펙 예제

다음은 커스텀 런타임 버전을 지정하는 빌드스펙의 예시입니다.

```
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
```

빌드 환경 컴퓨팅 모드 및 유형

CodeBuild에서는 빌드를 실행하는 데 CodeBuild 사용하는 컴퓨팅 및 런타임 환경 이미지를 지정할 수 있습니다. 컴퓨트는 관리 및 유지 관리되는 컴퓨팅 엔진 (CPU, 메모리, 운영 체제) 을 의미합니다 CodeBuild. 런타임 환경 이미지는 선택한 컴퓨팅 플랫폼에서 실행되는 컨테이너 이미지이며, 빌드에 필요할 수 있는 추가 도구(예: AWS CLI)가 포함되어 있습니다.

주제

컴퓨팅 모드 정보

CodeBuild 다음과 같은 컴퓨팅 모드를 제공합니다.

- EC2
- AWS Lambda

EC2는 구축 중에 최적화된 유연성을 AWS Lambda 제공하고 최적화된 시작 속도를 제공합니다. AWS Lambda 시작 지연 시간이 짧아 더 빠른 빌드를 지원합니다. AWS Lambda 또한 자동으로 크기가 조정되므로 빌드가 실행될 때까지 대기하지 않아도 됩니다. 자세한 설명은 AWS Lambda 컴퓨트 인에서 작업하기 AWS CodeBuild 섹션을 참조하세요.

환경 유형 정보

AWS CodeBuild 는 EC2 컴퓨팅 모드에 사용할 수 있는 다음과 같은 사용 가능한 메모리, vCPU 및 디스크 공간을 갖춘 빌드 환경을 제공합니다.

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	vCPU	디스크 공간
ARM Small	`BUILD_GENERAL1_SMALL`	`ARM_CONTAINER`	4GB	2	50GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	vCPU	디스크 공간
ARM Large	BUILD_GENERAL1_LARGE	ARM_CONTAINER	16 GB	8	50GB
Linux Small [1]	BUILD_GENERAL1_SMALL	LINUX_CONTAINER	3GB	2	64GB
Linux Medium [1]	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER	7GB	4	128GB
Linux Large [1]	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	15GB	8	128GB
Linux XLarge	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	70GB	36	256GB
Linux 2XLarge	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	145 GB	72	824 GB(SSD)
Linux GPU Sma	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GB	4	220GB
Linux GPU Larg	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GB	32	50GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	vCPU	디스크 공간
Windows Mediu	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER	7GB	4	128GB
Windows Large	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER	15GB	8	128GB

[1] 이 이미지 유형의 최신 버전이 캐시됩니다. 보다 구체적인 버전을 지정하는 경우 캐시된 버전 대신 해당 버전을 CodeBuild 프로비저닝합니다. 이로 인해 빌드 시간이 길어질 수 있습니다. 예를 들어 캐싱을 사용하려면 `aws/codebuild/amazonlinux2-x86_64-standard:5.0-1.0.0`과 같이 보다 세분화된 버전 대신 `aws/codebuild/amazonlinux2-x86_64-standard:5.0`을 지정합니다.

AWS CodeBuild AWS Lambda 컴퓨팅 모드에 사용할 수 있는 다음과 같은 메모리 및 디스크 공간이 있는 빌드 환경을 제공합니다.

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	디스크 공간
ARM Lambda 1GB	BUILD_LAMBDA_1GB	ARM_LAMBDA_CONTAINER	1GB	10GB
ARM Lambda 2GB	BUILD_LAMBDA_2GB	ARM_LAMBDA_CONTAINER	2GB	10GB
ARM Lambda 4GB	BUILD_LAMBDA_4GB	ARM_LAMBDA_CONTAINER	4GB	10GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	디스크 공간
ARM 람다 8기가바이트	BUILD_LAMBDA_8GB	ARM_LAMBDA_CONTAINER	8GB	10GB
ARM Lambda 10GI	BUILD_LAMBDA_10GB	ARM_LAMBDA_CONTAINER	10GB	10GB
Linux Lambda 1GB	BUILD_LAMBDA_1GB	LINUX_LAMBDA_CONTAINER	1GB	10GB
Linux Lambda 2GB	BUILD_LAMBDA_2GB	LINUX_LAMBDA_CONTAINER	2GB	10GB
Linux Lambda 4GB	BUILD_LAMBDA_4GB	LINUX_LAMBDA_CONTAINER	4GB	10GB
Linux Lambda 8GB	BUILD_LAMBDA_8GB	LINUX_LAMBDA_CONTAINER	8GB	10GB
Linux Lambda 10G	BUILD_LAMBDA_10GB	LINUX_LAMBDA_CONTAINER	10GB	10GB

다른 환경 유형을 사용할 때는 캐시된 이미지를 사용하여 빌드 시간을 줄이는 것이 좋습니다.

각 빌드 환경에 대해 나열된 디스크 공간은 CODEBUILD_SRC_DIR 환경 변수로 지정된 디렉터리에서만 사용할 수 있습니다.

컴퓨팅 유형을 선택하려면:

- CodeBuild 콘솔의 빌드 프로젝트 생성 마법사 또는 빌드 프로젝트 편집 페이지의 환경에서 추가 구성을 확장한 다음 컴퓨팅 유형에서 옵션 중 하나를 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 또는 빌드 프로젝트 설정 변경(콘솔)를 참조하세요.
- 의 AWS CLI경우 `create-project` 또는 `update-project` 명령을 실행하여 `environment` 개체 `computeType` 값을 지정합니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 또는 빌드 프로젝트 설정 변경(AWS CLI)를 참조하세요.
- AWS SDK의 경우 대상 프로그래밍 언어에 해당하는 `CreateProject` or `UpdateProject` 연산을 호출하고 `environment` 객체의 등가 `computeType` 값을 지정합니다. 자세한 내용은 AWS SDK 및 도구 참조 섹션을 참조하십시오.

일부 환경 및 컴퓨팅 유형에는 다음과 같은 리전 가용성 제한이 있습니다.

- 컴퓨팅 유형 Linux GPU Small(`LINUX_GPU_CONTAINER`)은 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(버지니아 북부)
  - 미국 서부(오레곤)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 유럽(프랑크푸르트)
  - 유럽(아일랜드)
  - 유럽(런던)
- 컴퓨팅 유형 Linux GPU Large(`LINUX_GPU_CONTAINER`)는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(오리건)
  - 아시아 태평양(서울)
  - 아시아 태평양(싱가포르)
  - 아시아 태평양(시드니)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 중국(베이징)
  - 중국(닝샤)
  - 유럽(프랑크푸르트)

- 유럽(아일랜드)
- 유럽(런던)

- 환경 유형 ARM_CONTAINER는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(캘리포니아 북부)
  - 미국 서부(오리건)
  - 아시아 태평양(홍콩)
  - 아시아 태평양(자카르타)
  - 아시아 태평양(하이데라바드)
  - 아시아 태평양(뭄바이)
  - 아시아 태평양(오사카)
  - 아시아 태평양(서울)
  - 아시아 태평양(싱가포르)
  - 아시아 태평양(시드니)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 중국(베이징)
  - 중국(닝샤)
  - 유럽(프랑크푸르트)
  - 유럽(아일랜드)
  - 유럽(런던)
  - 유럽(밀라노)
  - 유럽(파리)
  - 유럽(스페인)
  - 유럽(스톡홀름)
  - 이스라엘(텔아비브)
  - 중동(바레인)
  - 중동(UAE)
  - 남아메리카(상파울루)

- 컴퓨팅 유형 BUILD_GENERAL1_2XLARGE는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(캘리포니아 북부)
  - 미국 서부(오리건)
  - 아시아 태평양(하이데라바드)
  - 아시아 태평양(홍콩)
  - 아시아 태평양(자카르타)
  - 아시아 태평양(멜버른)
  - 아시아 태평양(뭄바이)
  - 아시아 태평양(서울)
  - 아시아 태평양(싱가포르)
  - 아시아 태평양(시드니)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 중국(베이징)
  - 중국(닝샤)
  - 유럽(프랑크푸르트)
  - 유럽(아일랜드)
  - 유럽(런던)
  - 유럽(파리)
  - 유럽(스페인)
  - 유럽(스톡홀름)
  - 유럽(취리히)
  - 이스라엘(텔아비브)
  - 중동(바레인)
  - 중동(UAE)
  - 남아메리카(상파울루)
- 컴퓨팅 모드 AWS Lambda (ARM_LAMBDA_CONTAINER및LINUX_LAMBDA_CONTAINER) 는 다음 지역에서만 사용할 수 있습니다.
  - 미국 동부(버지니아 북부)

- 미국 동부(오하이오)
- 미국 서부(오레곤)
- 아시아 태평양(뭄바이)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 남아메리카(상파울루)

컴퓨팅 유형 `BUILD_GENERAL1_2XLARGE`은 압축되지 않은 최대 100GB의 도커 이미지가 지원됩니다.

> **Note**
>
> 사용자 지정 빌드 환경 이미지의 경우 컴퓨팅 유형에 관계없이 Linux 및 Windows에서 압축되지 않은 최대 50GB의 Docker 이미지를 CodeBuild 지원합니다. 빌드 이미지의 크기를 확인하려면 Docker를 사용하여 `docker images` *`REPOSITORY`*`:`*`TAG`* 명령을 실행합니다.

Amazon EFS를 사용하여 빌드 컨테이너의 더 많은 공간에 액세스할 수 있습니다. 자세한 설명은 에 대한 Amazon Elastic File System 샘플 AWS CodeBuild 섹션을 참조하세요. 빌드 중 컨테이너 디스크 공간을 조정하려면 권한을 가진 모드에서 빌드를 실행해야 합니다.

> **Note**
>
> 기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.

빌드 환경의 쉘 및 명령

빌드 수명 주기 동안 사용자는 AWS CodeBuild용 명령 세트를 제공하여 빌드 환경에서 실행합니다(예: 빌드 종속성 설치 및 소스 코드 테스트와 컴파일). 이러한 명령을 지정하는 데에는 다음과 같은 몇 가지 방법이 있습니다.

- 빌드 사양 파일을 만들어 소스 코드에 포함합니다. 이 파일에서 빌드 수명 주기의 각 단계에서 실행할 명령을 지정합니다. 자세한 내용은 에 대한 빌드 사양 참조 CodeBuild 부분을 참조하세요.
- CodeBuild 콘솔을 사용하여 빌드 프로젝트를 생성합니다. 빌드 명령 삽입에서 빌드 명령에 `build` 단계에서 실행하려는 명령을 입력합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.
- CodeBuild 콘솔을 사용하여 빌드 프로젝트 설정을 변경합니다. 빌드 명령 삽입에서 빌드 명령에 `build` 단계에서 실행하려는 명령을 입력합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.
- AWS CLI 또는 AWS SDK를 사용하여 빌드 프로젝트를 생성하거나 빌드 프로젝트 설정을 변경합니다. 명령을 사용하여 빌드 사양 파일이 들어 있는 소스 코드를 참조하거나, 빌드 사양 파일에 해당하는 파일의 내용이 들어 있는 단일 문자열을 지정합니다. 자세한 내용은 빌드 프로젝트 생성 또는 빌드 프로젝트 설정 변경 섹션을 참조하세요.
- AWS CLI 또는 AWS SDK를 사용하여, 빌드 사양 파일 또는 빌드 사양 파일에 해당하는 파일 내용이 들어 있는 단일 문자열을 지정하여 빌드를 시작합니다. 자세한 정보는 빌드 실행의 `buildspecOverride` 값에 대한 설명을 참조하십시오.

Shell 명령 언어(sh) 명령을 지정할 수 있습니다. 빌드 사양 버전 0.1에서 CodeBuild는 빌드 환경에 있는 서로 다른 인스턴스에서 각 Shell 명령을 실행합니다. 즉, 각 명령이 다른 모든 명령과 독립적으로 실행됩니다. 따라서 기본적으로 이전 명령의 상태에 따라 실행되는 단일 명령을 실행할 수 없습니다(예: 디렉터리 변경 또는 환경 변수 설정). 이 제한 사항을 해결하려면 이 문제를 해결하는 버전 0.2를 사용하는 것이 좋습니다. 버전 0.1을 사용해야 하는 경우 다음 접근 방식을 따르는 것이 좋습니다.

- 기본 쉘의 단일 인스턴스에서 실행하려는 명령이 들어 있는 쉘 스크립트를 소스 코드에 포함합니다. 예를 들어, `my-script.sh`와 같은 명령이 들어 있는 `cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd;`라는 파일을 소스 코드에 포함할 수 있습니다. 그런 다음 빌드 사양 파일에서 `./my-script.sh` 명령을 지정합니다.
- buildspec 파일 또는 해당 `build` 단계의 빌드 명령 설정에서만 기본 쉘의 단일 인스턴스에서 실행하려는 모든 명령이 포함된 단일 명령을 입력합니다(예: `cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd`).

CodeBuild에서 오류가 발생하는 경우 기본 쉘의 자체 인스턴스에서 단일 명령을 실행하는 것보다 오류를 해결하기가 더 어려울 수 있습니다.

Windows Server Core 이미지에서 실행되는 명령은 Powershell 쉘을 사용합니다.

빌드 환경의 환경 변수

AWS CodeBuild는 빌드 명령에서 사용할 수 있는 다양한 환경 변수를 제공합니다.

AWS_DEFAULT_REGION

빌드가 실행되고 있는 AWS 리전입니다(예: `us-east-1`). 이 환경 변수는 AWS CLI에 의해 주로 사용됩니다.

AWS_REGION

빌드가 실행되고 있는 AWS 리전입니다(예: `us-east-1`). 이 환경 변수는 AWS SDK에 의해 주로 사용됩니다.

코드빌드_배치_빌드_식별자

배치 빌드의 빌드 식별자입니다. 이는 배치 buildspec에 지정되어 있습니다. 자세한 내용은 the section called “배치 buildspec 참조” 섹션을 참조하세요.

CODEBUILD_BUILD_ARN

빌드의 Amazon 리소스 이름(ARN)입니다(예: `arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).

CODEBUILD_BUILD_ID

빌드의 CodeBuild ID입니다(예: `codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).

CODEBUILD_BUILD_IMAGE

CodeBuild 빌드 이미지 식별자입니다(예: `aws/codebuild/standard:2.0`).

CODEBUILD_BUILD_NUMBER

프로젝트의 현재 빌드 번호입니다.

CODEBUILD_BUILD_SUCCEEDING

현재 빌드가 성공적으로 진행되는지 여부입니다. 빌드가 실패하는 경우 `0`으로 설정하고, 성공하는 경우 `1`로 설정합니다.

CODEBUILD_INITIATOR

빌드를 시작한 엔터티입니다. CodePipeline이 빌드를 시작한 경우 파이프라인의 이름입니다(예: `codepipeline/my-demo-pipeline`). 사용자가 빌드를 시작했으면 사용자의 이름입니다

(예: `MyUserName`). CodeBuild용 Jenkins 플러그인이 빌드를 시작했으면 문자열 `CodeBuild-Jenkins-Plugin`입니다.

CODEBUILD_KMS_KEY_ID

CodeBuild가 빌드 출력 아티팩트를 암호화하는 데 사용하고 있는 `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* 키 식별자입니다(예: AWS KMS 또는 `alias/`*`key-alias`*).

CODEBUILD_LOG_PATH

해당 빌드에 대한 CloudWatch Logs의 로그 스트림 이름입니다.

CODEBUILD_PUBLIC_BUILD_URL

퍼블릭 빌드 웹사이트에 있는 이 빌드의 빌드 결과 URL입니다. 이 변수는 빌드 프로젝트에 퍼블릭 빌드가 활성화된 경우에만 설정됩니다. 자세한 내용은 AWS CodeBuild의 퍼블릭 빌드 프로젝트 섹션을 참조하세요.

CODEBUILD_RESOLVED_SOURCE_VERSION

빌드 소스 코드의 버전 식별자입니다. 내용은 다음 소스 코드 리포지토리에 따라 달라집니다.

CodeCommit, GitHub, GitHub Enterprise Server 및 Bitbucket

이 변수에는 커밋 ID가 포함됩니다.

CodePipeline

이 변수에는 CodePipeline에서 제공하는 소스 수정 버전이 포함되어 있습니다.

CodePipeline이 소스 수정 사항을 확인할 수 없는 경우(예: 소스가 버전 관리가 활성화되지 않은 Amazon S3 버킷인 경우), 이 환경 변수는 설정되지 않습니다.

Amazon S3

이 변수는 설정되지 않습니다.

해당하는 경우 `CODEBUILD_RESOLVED_SOURCE_VERSION` 변수는 `DOWNLOAD_SOURCE` 단계 이후에만 사용할 수 있습니다.

CODEBUILD_SOURCE_REPO_URL

입력 아티팩트 또는 소스 코드 리포지토리에 대한 URL입니다. Amazon S3의 경우 `s3://` 뒤에 버킷 이름과 입력 아티팩트에 대한 경로가 옵니다. CodeCommit과 GitHub의 경우 리포지토리의 복제 URL입니다. CodePipeline에서 빌드를 시작한 경우 이 환경 변수는 비어 있을 수 있습니다.

보조 소스의 경우 보조 소스 리포지토리 URL의 환경 변수는 `CODEBUILD_SOURCE_REPO_URL_<sourceIdentifier>`입니다. 여기서 `<sourceIdentifier>`는 사용자가 생성한 소스 식별자입니다.

CODEBUILD_SOURCE_VERSION

값의 형식은 소스 코드 리포지토리에 따라 다릅니다.

- Amazon S3의 경우 입력 아티팩트에 연결된 버전 ID입니다.
- CodeCommit의 경우, 커밋 ID 또는 빌드할 소스 코드 버전과 연관된 분기 이름입니다.
- GitHub, GitHub Enterprise Server, Bitbucket의 경우, 커밋 ID, 분기 이름 또는 빌드할 소스 코드 버전과 연관된 태그 이름입니다.

> **Note**
>
> Webhook pull 요청 이벤트에서 트리거하는 GitHub 또는 GitHub Enterprise Server 빌드의 경우 `pr/pull-request-number`입니다.

보조 소스의 경우 보조 소스 버전의 환경 변수는 `CODEBUILD_SOURCE_VERSION_<sourceIdentifier>`입니다. 여기서 `<sourceIdentifier>`는 사용자가 생성한 소스 식별자입니다. 자세한 내용은 다중 입력 소스 및 출력 아티팩트 샘플 섹션을 참조하세요.

CODEBUILD_SRC_DIR

CodeBuild가 빌드에 사용하는 디렉터리 경로입니다(예: `/tmp/src123456789/src`).

보조 소스를 사용하는 경우 보조 소스 디렉터리 경로의 환경 변수는 `CODEBUILD_SRC_DIR_<sourceIdentifier>`입니다. 여기서 `<sourceIdentifier>`는 사용자가 생성한 소스 식별자입니다. 자세한 내용은 다중 입력 소스 및 출력 아티팩트 샘플 섹션을 참조하세요.

CODEBUILD_START_TIME

밀리초 단위의 Unix 타임스탬프로 지정된 빌드의 시작 시간입니다.

CODEBUILD_WEBHOOK_ACTOR_ACCOUNT_ID

Webhook 이벤트를 트리거한 사용자의 계정 ID입니다.

CODEBUILD_WEBHOOK_BASE_REF

현재 빌드를 트리거하는 Webhook 이벤트의 기본 참조 이름입니다. pull 요청의 경우 이를 브랜치 참조라고 합니다.

CODEBUILD_WEBHOOK_EVENT

현재 빌드를 트리거하는 Webhook 이벤트입니다.

CODEBUILD_WEBHOOK_MERGE_COMMIT

빌드에 사용된 병합 커밋의 식별자입니다. 이 변수는 Bitbucket 풀 요청이 스쿼시 전략과 병합되고 pull 요청 분기가 닫힐 때 설정됩니다. 이 경우 원래의 풀 요청 커밋은 더 이상 존재하지 않으므로 이 환경 변수에는 스쿼시된 병합 커밋의 식별자가 포함됩니다.

CODEBUILD_WEBHOOK_PREV_COMMIT

현재 빌드를 트리거하는 webhook 푸시 이벤트 전 최신 커밋의 ID입니다.

CODEBUILD_WEBHOOK_HEAD_REF

현재 빌드를 트리거하는 Webhook 이벤트의 헤드 참조 이름입니다. 브랜치 참조 또는 태그 참조일 수 있습니다.

CODEBUILD_WEBHOOK_TRIGGER

빌드를 트리거하는 Webhook 이벤트를 표시합니다. 이 변수는 Webhook가 트리거하는 빌드에만 사용할 수 있습니다. 이 값은 GitHub, GitHub Enterprise Server 또는 Bitbucket이 CodeBuild로 전송하는 페이로드에서 구문 분석됩니다. 값의 형식은 빌드를 트리거한 이벤트 유형에 따라 다릅니다.

- pull 요청이 트리거한 빌드의 경우 `pr/`*`pull-request-number`*입니다.
- 새 브랜치를 생성하거나 브랜치로 커밋을 푸시하여 트리거된 빌드의 경우 `branch/`*`branch-name`*입니다.
- 리포지토리로 태그를 푸시하여 트리거된 빌드의 경우 `tag/`*`tag-name`*입니다.

HOME

이 환경 변수는 항상 `/root`로 설정되어 있습니다.

자체 환경 변수를 사용하여 빌드 환경을 제공할 수도 있습니다. 자세한 정보는 다음 주제를 참조하세요.

- CodePipeline 함께 사용 CodeBuild
- 빌드 프로젝트 생성
- 빌드 프로젝트 설정 변경
- 빌드 실행
- buildspec 참조

빌드 환경에서 사용 가능한 모든 환경 변수를 나열하려면 빌드 중에 `printenv` (Linux 기반 빌드 환경의 경우) 또는 `"Get-ChildItem Env:"` (Windows 기반 빌드 환경의 경우) 명령을 실행하면 됩니다. 앞에서 나열한 환경 변수를 제외하고, `CODEBUILD_`로 시작하는 환경 변수는 CodeBuild에서 내부적으로 사용합니다. 이러한 환경 변수는 빌드 명령에서 사용하면 안 됩니다.

> **Important**
>
> 환경 변수를 사용하여 중요한 값(특히 AWS 액세스 키 ID)을 저장하는 것은 가급적 피해야 합니다. 환경 변수는 CodeBuild 콘솔 및 AWS CLI와 같은 도구를 사용하여 일반 텍스트로 표시할 수 있습니다.
>
> 중요한 값을 Amazon EC2 Systems Manager Parameter Store에 저장한 후에 buildspec에서 검색하는 것이 좋습니다. 중요한 값을 저장하려면 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 안내: 문자열 파라미터 생성 및 테스트(콘솔)를 참조하세요. 검색하려면 buildspec 구문의 `parameter-store` 매핑을 참조하십시오.

빌드 환경의 배경 작업

빌드 환경에서 배경 작업을 실행할 수 있습니다. 이렇게 하려면 buildspec에서 빌드 프로세스가 쉘을 종료하더라도 nohup 명령을 사용하여 명령을 배경 작업으로서 실행합니다. disown 명령을 사용하여 실행 중인 배경 작업을 강제로 중단합니다.

예제:

- 배경 프로세스를 시작하고 나중에 완료될 때까지 기다립니다.

```
|
nohup sleep 30 & echo $! > pidfile
…
wait $(cat pidfile)
```

- 배경 프로세스를 시작하고 완료될 때까지 기다리지 않습니다.

```
|
nohup sleep 30 & disown $!
```

- 배경 프로세스를 시작하고 나중에 종료합니다.

```
|
```

```
nohup sleep 30 & echo $! > pidfile
…
kill $(cat pidfile)
```

AWS CodeBuild 에이전트를 사용하여 로컬에서 빌드 실행

로컬 시스템에서 AWS CodeBuild 에이전트를 사용하여 CodeBuild 빌드를 실행할 수 있습니다. x86_64 및 ARM 플랫폼에 사용할 수 있는 에이전트가 있습니다.

새 버전의 에이전트가 게시되는 시기를 알 수 있도록 알림을 구독할 수도 있습니다.

필수 조건

시작하려면 다음을 수행해야 합니다.

- 로컬 시스템에 Git를 설치합니다.
- 로컬 컴퓨터에 Docker를 설치하고 설정합니다.

빌드 이미지 설정

에이전트를 처음 실행할 때 또는 이미지가 변경된 경우에만 빌드 이미지를 설정해야 합니다.

빌드 이미지를 설정하려면

1. 큐레이트된 Amazon Linux 2 이미지를 사용하려면 다음 명령을 사용하여 CodeBuild 퍼블릭 Amazon ECR 리포지토리 https://gallery.ecr.aws/codebuild/amazonlinux2-x86_64-standard에서 이미지를 끌어올 수 있습니다.

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

   다른 Linux 이미지를 사용하려면 다음 단계를 수행합니다.

   a. CodeBuild 이미지 리포지토리를 복제합니다.

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

   b. image 디렉터리로 변경합니다. 이 예에서는 `aws/codebuild/standard:5.0` 이미지를 사용합니다.

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

c. 이미지를 빌드합니다. 이 작업은 몇 분 정도 걸릴 수 있습니다.

```
$ docker build -t aws/codebuild/standard:5.0 .
```

2. CodeBuild 에이전트를 다운로드합니다.

   x86_64 버전의 에이전트를 다운로드하려면 다음 명령을 실행합니다.

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

   ARM 버전의 에이전트를 다운로드하려면 다음 명령을 실행합니다.

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

3. CodeBuild 에이전트는 https://gallery.ecr.aws/codebuild/local-builds에서 사용할 수 있습니다.

   에이전트의 x86_64 버전에 대한 보안 해시 알고리즘(SHA) 서명은 다음과 같습니다.

```
sha256:fac17c6d6c3cb500f6e7975887de1e41d29a9e70a86d6f49f76a2beacfcf967e
```

   에이전트의 ARM 버전에 대한 SHA 서명은 다음과 같습니다.

```
sha256:57a5dfda63be50edce13dea16dcd5e73e8d8559029658ba08b793c9a7adc68c7
```

   SHA를 사용하여 에이전트 버전을 확인할 수 있습니다. 에이전트의 SHA 서명을 보려면 다음 명령을 실행하고 `RepoDigests`에서 SHA를 찾습니다.

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

CodeBuild 에이전트 실행

CodeBuild 에이전트를 실행하려면

1. 빌드 프로젝트 소스가 들어 있는 디렉터리로 변경합니다.
2. codebuild_build.sh 스크립트를 다운로드합니다.

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

3. codebuild_build.sh 스크립트를 실행하고 컨테이너 이미지와 출력 디렉터리를 지정합니다.

   x86_64 빌드를 실행하려면 다음 명령을 실행합니다.

   ```
   $ ./codebuild_build.sh -i <container-image> -a <output directory>
   ```

   ARM 빌드를 실행하려면 다음 명령을 실행합니다.

   ```
   $ ./codebuild_build.sh -i <container-image> -a <output directory> -l
    public.ecr.aws/codebuild/local-builds:aarch64
   ```

   *<container-image>*를 컨테이너 이미지의 이름(예: aws/codebuild/standard:5.0 또는 public.ecr.aws/codebuild/amazonlinux2-x86_64-standard:4.0)으로 바꿉니다.

   스크립트는 빌드 이미지를 시작하고 현재 디렉터리의 프로젝트에서 빌드를 실행합니다. 빌드 프로젝트의 위치를 지정하려면 스크립트 명령에 -s *<build project directory>* 옵션을 추가합니다.

새 CodeBuild 에이전트 버전에 대한 알림 받기

새 버전의 AWS CodeBuild 에이전트가 릴리스되는 시기를 알 수 있도록 Amazon SNS 알림을 구독할 수 있습니다.

CodeBuild 에이전트 알림을 구독하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 모음에서 아직 선택되지 않은 경우 AWS 리전을 미국 동부(버지니아 북부)로 변경합니다. 구독을 신청하는 Amazon SNS 알림이 AWS 리전에 생성되기 때문에 이 리전을 선택해야 합니다.
3. 탐색 창에서 구독을 선택합니다.
4. Create subscription을 선택합니다.
5. 구독 생성에서 다음을 수행합니다.

   a. 주제 ARN에 다음 Amazon 리소스 이름(ARN)을 사용합니다.

```
arn:aws:sns:us-east-1:850632864840:AWS-CodeBuild-Local-Agent-Updates
```

b. 프로토콜의 경우 이메일 또는 SMS를 선택합니다.

c. 엔드포인트의 경우 알림을 수신할 위치(이메일 또는 SMS)를 선택합니다. 지역 번호를 포함한 이메일 주소, 우편 주소 또는 전화번호를 입력합니다.

d. 구독 생성을 선택합니다.

e. 구독 사실을 확인하는 이메일을 받으려면 이메일을 선택합니다. 이메일의 지침에 따라 구독을 완료합니다.

이런 알림을 더 이상 받지 않기를 원하는 경우, 다음 절차를 수행해서 구독을 해제하세요.

CodeBuild 에이전트 알림을 구독 해제하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독을 선택한 후 작업에서 구독 삭제를 선택합니다. 확인 메시지가 표시되면 [Delete]를 선택합니다.

Amazon Virtual Private 클라우드와 AWS CodeBuild 함께 사용

일반적으로 VPC의 리소스에 액세스할 AWS CodeBuild 수 없습니다. 액세스를 활성화하려면 프로젝트 구성에 VPC별 구성 정보를 추가로 제공해야 합니다. CodeBuild 여기에는 VPC ID, VPC 서브넷 ID 및 VPC 보안 그룹 ID가 포함됩니다. 그러면 VPC 활성화 빌드가 VPC 내부의 리소스에 액세스할 수 있습니다. Amazon VPC의 VPC 설정에 대한 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

주제

사용 사례

AWS CodeBuild 빌드의 VPC 연결을 통해 다음을 수행할 수 있습니다.

- 빌드에서 사설 서브넷에 격리된 RDS 데이터베이스의 데이터에 대해 통합 테스트를 실행합니다.
- 테스트에서 직접 Amazon ElastiCache 클러스터의 데이터를 쿼리합니다.
- Amazon EC2, Amazon ECS에서 호스팅되는 내부 웹 서비스 또는 내부 Elastic Load Balancing을 사용하는 서비스와 상호 작용합니다.
- Python용 PyPI, Java용 Maven 및 Node.js용 npm과 같은 자체 호스팅된 내부 결과물 리포지토리의 종속성을 검색합니다.
- Amazon VPC 엔드포인트를 통해서만 액세스할 수 있도록 구성된 S3 버킷의 객체에 액세스합니다.
- 서브넷과 연결된 NAT 게이트웨이 또는 NAT 인스턴스의 탄력적 IP 주소를 통해 고정 IP 주소가 필요한 외부 웹 서비스를 쿼리합니다.

빌드는 VPC에서 호스팅되는 모든 리소스에 액세스할 수 있습니다.

프로젝트에서 Amazon VPC 액세스 허용 CodeBuild

VPC 구성에 다음 설정을 포함합니다.

- VPC ID의 경우 사용하는 VPC ID를 선택합니다. CodeBuild
- 서브넷의 경우 에서 사용하는 리소스를 포함하거나 해당 리소스에 대한 경로가 있는 NAT 변환이 있는 프라이빗 서브넷을 선택하십시오. CodeBuild
- 보안 그룹의 경우 VPC의 리소스에 대한 액세스를 허용하는 데 CodeBuild 사용하는 보안 그룹을 선택합니다.

콘솔을 사용하여 빌드 프로젝트를 생성하려면 빌드 프로젝트 만들기(콘솔) 단원을 참조하십시오. CodeBuild 프로젝트를 만들거나 변경할 때 VPC에서 VPC ID, 서브넷, 보안 그룹을 선택합니다.

를 사용하여 빌드 프로젝트를 AWS CLI 만들려면 을 참조하십시오. 빌드 프로젝트 생성(AWS CLI) 를 사용하는 경우 IAM 사용자를 대신하여 서비스와 상호 작용하는 CodeBuild 데 사용하는 서비스 역할에 정책이 연결되어 있어야 합니다. AWS CLI CodeBuild 자세한 내용은 VPC 네트워크 인터페이스 생성에 필요한 AWS 서비스에 CodeBuild 대한 액세스 허용을 참조하세요.

VPCConfig #### vPCID, # #### ##### ###. securityGroupIds

- *vpcId*: 필수 항목입니다. 사용하는 VPC ID입니다. CodeBuild 리전의 모든 Amazon VPC ID 목록을 가져오려면 다음 명령을 실행합니다.

```
aws ec2 describe-vpcs
```

- *subnets*: 필수 항목입니다. 에서 사용하는 리소스가 포함된 서브넷 ID. CodeBuild 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

> **Note**
>
> us-east-1을 해당 리전으로 바꿉니다.

- *securityGroupIds*: 필수. 에서 VPC의 리소스에 대한 액세스를 허용하는 CodeBuild 데 사용하는 보안 그룹 ID. 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

> **Note**
>
> us-east-1을 해당 리전으로 바꿉니다.

VPC 모범 사례

작업할 VPC를 설정할 때 이 체크리스트를 사용하십시오. CodeBuild

- 퍼블릭 및 프라이빗 서브넷과 NAT 게이트웨이가 있는 VPC를 설정합니다. NAT 게이트웨이는 퍼블릭 서브넷에 상주해야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 퍼블릭 및 프라이빗 서브넷이 있는 VPC(NAT)를 참조하세요.

  > **Important**
  >
  > 퍼블릭 엔드포인트에 도달할 CodeBuild 수 있도록 CodeBuild VPC와 함께 사용할 NAT 게이트웨이 또는 NAT 인스턴스가 필요합니다 (예: 빌드 실행 시 CLI 명령 실행). 인터넷 게이트웨이는 생성하는 네트워크 인터페이스에 엘라스틱 IP 주소를 할당하는 것을 CodeBuild 지원하지 않기 때문에 NAT 게이트웨이 또는 NAT 인스턴스 대신 인터넷 게이트웨이를 사용할 수 없습니다. Amazon EC2는 Amazon EC2 인스턴스 시작 외부에서 생성된 네트워크 인터페이스에 대해 퍼블릭 IP 주소 자동 할당을 지원하지 않기 때문입니다.

- VPC에 여러 가용 영역을 포함합니다.
- 보안 그룹에 빌드에 허용되는 인바운드 (인그레스) 트래픽이 없는지 확인하십시오. CodeBuild 아웃바운드 트래픽에 대한 특정 요구 사항은 없지만 Amazon S3와 같이 GitHub 빌드에 필요한 모든 인터넷 리소스에 대한 액세스를 허용해야 합니다.

  자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹 규칙을 참조하세요.
- 빌드에 대해 별도의 서브넷을 설정합니다.
- VPC에 액세스하도록 CodeBuild 프로젝트를 설정할 때는 프라이빗 서브넷만 선택하십시오.

Amazon VPC의 VPC 설정에 대한 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

VPC 기능을 사용하도록 VPC를 구성하는 AWS CloudFormation 데 사용하는 방법에 대한 자세한 내용은 를 참조하십시오. CodeBuild AWS CloudFormation VPC 템플릿

VPC 설정 문제 해결

오류 메시지에 표시되는 정보를 사용하면 해당 문제를 식별, 진단 및 해결하는 데 도움이 됩니다.

다음은 일반적인 CodeBuild VPC 오류를 해결할 때 도움이 되는 몇 가지 지침입니다. `Build does not have internet connectivity. Please check subnet network configuration`

1. 인터넷 게이트웨이가 VPC에 연결되어 있는지 확인합니다.
2. 퍼블릭 서브넷의 라우팅 테이블이 인터넷 게이트웨이를 가리키는지 확인합니다.
3. 네트워크 ACL이 트래픽의 흐름을 허용하는지 확인합니다.
4. 보안 그룹이 트래픽의 흐름을 허용하는지 확인합니다.
5. NAT 게이트웨이 문제를 해결합니다.
6. 프라이빗 서브넷의 라우팅 테이블이 NAT 게이트웨이를 가리키는지 확인합니다.
7. IAM 사용자를 대신하여 서비스와 상호 작용하는 CodeBuild 데 사용하는 서비스 역할에 이 정책의 권한이 있는지 확인하십시오. 자세한 정보는 CodeBuild 서비스 역할 생성을 참조하세요.

   권한이 CodeBuild 누락된 경우 다음과 같은 오류 메시지가 표시될 수 있습니다. `Unexpected EC2 error: UnauthorizedOperation` VPC를 사용하는 데 필요한 Amazon EC2 권한이 없는 경우 CodeBuild 이 오류가 발생할 수 있습니다.

VPC의 제한 사항

- Windows에서는 VPC 연결이 지원되지 않습니다. CodeBuild
- 공유 VPC의 VPC CodeBuild 연결은 지원되지 않습니다.

VPC 엔드포인트 사용

인터페이스 VPC 엔드포인트를 사용하도록 AWS CodeBuild를 구성하여 빌드의 보안을 향상할 수 있습니다. 인터페이스 엔드포인트는 프라이빗 IP 주소를 사용하여 Amazon EC2 및 CodeBuild에 비공개로 액세스할 수 있는 기술인 PrivateLink로 구동됩니다. PrivateLink는 관리형 인스턴스, CodeBuild 및 Amazon EC2 간의 모든 네트워크 트래픽을 Amazon 네트워크로 제한합니다. (관리형 인스턴스는 인터넷에 액세스할 수 없음) 또한 인터넷 게이트웨이, NAT 디바이스 또는 가상 프라이빗 게이트웨이가

필요 없습니다. PrivateLink를 구성하는 것이 필수는 아니지만 구성하는 것이 좋습니다. PrivateLink 및 VPC 엔드포인트에 대한 자세한 내용은 AWS PrivateLink란?을 참조하세요.

VPC 엔드포인트를 생성하기 전에

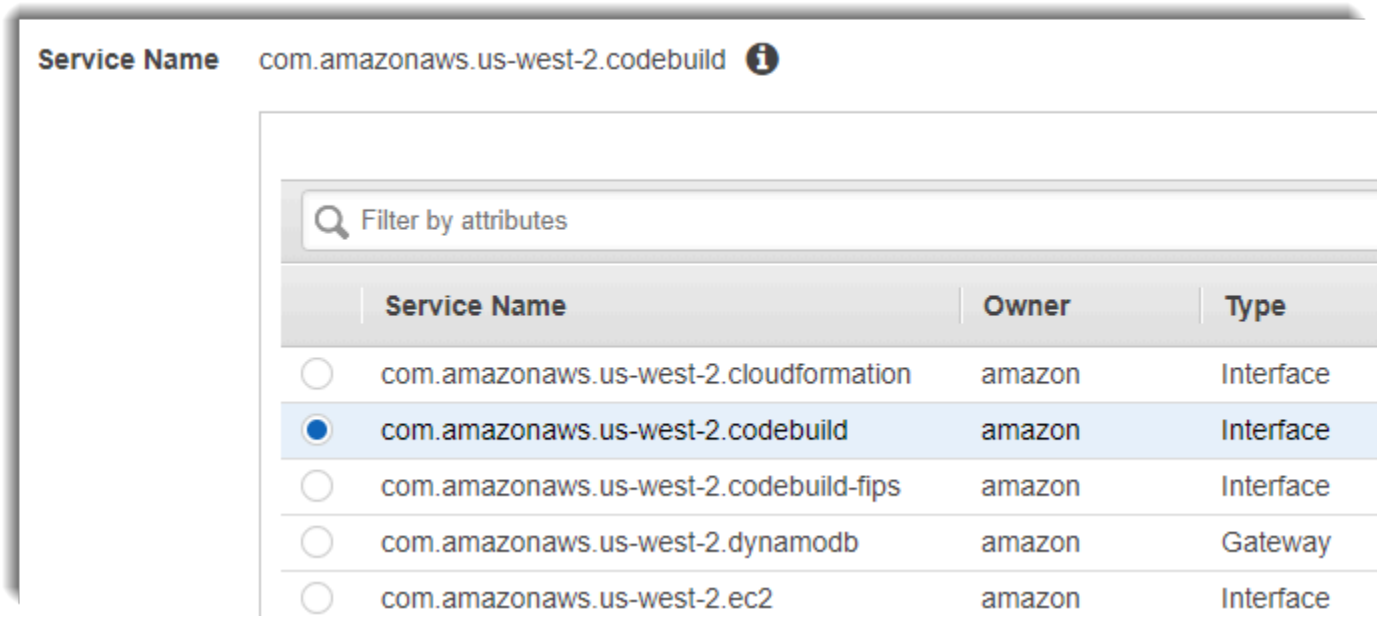
AWS CodeBuild에 대해 VPC 엔드포인트를 구성하기 전에 다음 제한 사항에 유의하십시오.

> **Note**
>
> Amazon VPC PrivateLink 연결을 지원하지 않는 AWS 서비스와 함께 CodeBuild를 사용하려면 NAT 게이트웨이를 사용합니다.

- VPC 엔드포인트는 Amazon Route 53을 통해 Amazon이 제공하는 DNS만 지원합니다. 자신의 DNS를 사용하는 경우에는 조건적인 DNS 전송을 사용할 수 있습니다. 자세한 정보는 Amazon VPC 사용 설명서의 DHCP 옵션 세트를 참조하세요.
- VPC 엔드포인트는 교차 리전 요청을 현재 지원하지 않습니다. 빌드 입력 및 출력을 저장하는 S3 버킷과 같은 AWS 리전에서 엔드포인트를 생성해야 합니다. Amazon S3 콘솔 또는 get-bucket-location 명령을 사용하여 버킷 위치를 찾을 수 있습니다. 리전별 Amazon S3 엔드포인트를 사용하여 버킷에 액세스하세요(예: `<bucket-name>.s3-us-west-2.amazonaws.com`). Amazon S3의 리전별 엔드포인트에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon Simple Storage Service를 참조하세요. AWS CLI를 사용하여 Amazon S3에 요청할 경우, 기본 리전을 버킷이 생성된 리전과 동일한 리전으로 설정하거나, 요청에 `--region` 파라미터를 사용하세요.

CodeBuild용 VPC 엔드포인트 생성

인터페이스 엔드포인트 생성의 지침에 따라 엔드포인트 `com.amazonaws.region.codebuild`를 만듭니다. AWS CodeBuild에 대한 VPC 엔드포인트입니다.



##은 미국 동부(오하이오) 리전의 `us-east-2` 같이 CodeBuild가 지원하는 AWS 리전의 리전 식별자를 나타냅니다. 지원되는 AWS 리전 목록은 AWS 일반 참조에서 CodeBuild를 참조하세요. 엔드포인트는 AWS에 로그인할 때 지정한 리전으로 사전 입력됩니다. 리전을 변경하면 그에 따라 VPC 엔드포인트가 업데이트됩니다.

CodeBuild에 대한 VPC 엔드포인트 정책 생성

AWS CodeBuild에 대한 Amazon VPC 엔드포인트의 정책을 생성할 수 있습니다. 이 정책에서 다음을 지정할 수 있습니다.

- 태스크를 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업입니다.
- 수행되는 작업을 가질 수 있는 리소스입니다.

다음 예제 정책은 모든 보안 주체가 `project-name` 프로젝트에 대한 빌드를 시작하고 볼 수만 있도록 지정합니다.

```
{
    "Statement": [
        {
            "Action": [
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuilds"
```

```
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:codebuild:region-ID:account-ID:project/project-name",
            "Principal": "*"
        }
    ]
}
```

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

AWS CloudFormation VPC 템플릿

AWS CloudFormation을 사용하면, 템플릿 파일을 사용하여 리소스 모음을 단일 유닛(스택)으로 함께 생성하고 삭제함으로써 AWS 인프라 배포를 예측 가능하고 반복적으로 생성하고 프로비저닝할 수 있습니다. 자세한 정보는 AWS CloudFormation 사용 설명서를 참조하세요.

다음은 VPC를 구성하여 AWS CodeBuild를 사용하기 위한 AWS CloudFormation YAML 템플릿입니다. 이 파일은 samples.zip에서도 사용할 수 있습니다.

```
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
```

프록시 서버에 AWS CodeBuild 사용

프록시 서버에 AWS CodeBuild를 사용하여 인터넷과의 HTTP 및 HTTPS 트래픽을 제어할 수 있습니다. 프록시 서버에서 CodeBuild를 실행하려면 VPC에서 퍼블릭 서브넷에 프록시 서버를 설치하고 가상 서브넷에 CodeBuild를 설치합니다.

프록시 서버에서 CodeBuild를 실행하는 주요 사용 사례는 두 가지입니다.

- VPC에서 NAT 게이트웨이 또는 NAT 인스턴스를 사용할 필요가 없어집니다.
- 프록시 서버 내 인스턴스가 액세스할 수 있는 URL을 지정하고 프록시 서버가 액세스를 거부하는 URL을 지정할 수 있습니다.

CodeBuild를 두 가지 유형의 프록시 서버에 사용할 수 있습니다. 두 유형 모두, 프록시 서버는 퍼블릭 서브넷에서 실행되고 CodeBuild는 프라이빗 서브넷에서 실행됩니다.

- 명시적 프록시: 명시적 프록시 서버를 사용하는 경우 프로젝트 수준에서 CodeBuild에 NO_PROXY, HTTP_PROXY 및 HTTPS_PROXY 환경 변수를 구성해야 합니다. 자세한 정보는 AWS CodeBuild에서 빌드 프로젝트 설정 변경 및 AWS CodeBuild에서 빌드 프로젝트 생성 섹션을 참조하세요.
- 투명 프록시: 투명 프록시 서버를 사용하는 경우 특별한 구성이 필요하지 않습니다.

주제

- 프록시 서버에서 CodeBuild를 실행하기 위해 필요한 구성 요소
- 명시적 프록시 서버에서 CodeBuild 실행
- 투명 프록시 서버에서 CodeBuild 실행
- 프록시 서버에서 패키지 관리자 및 기타 도구 실행

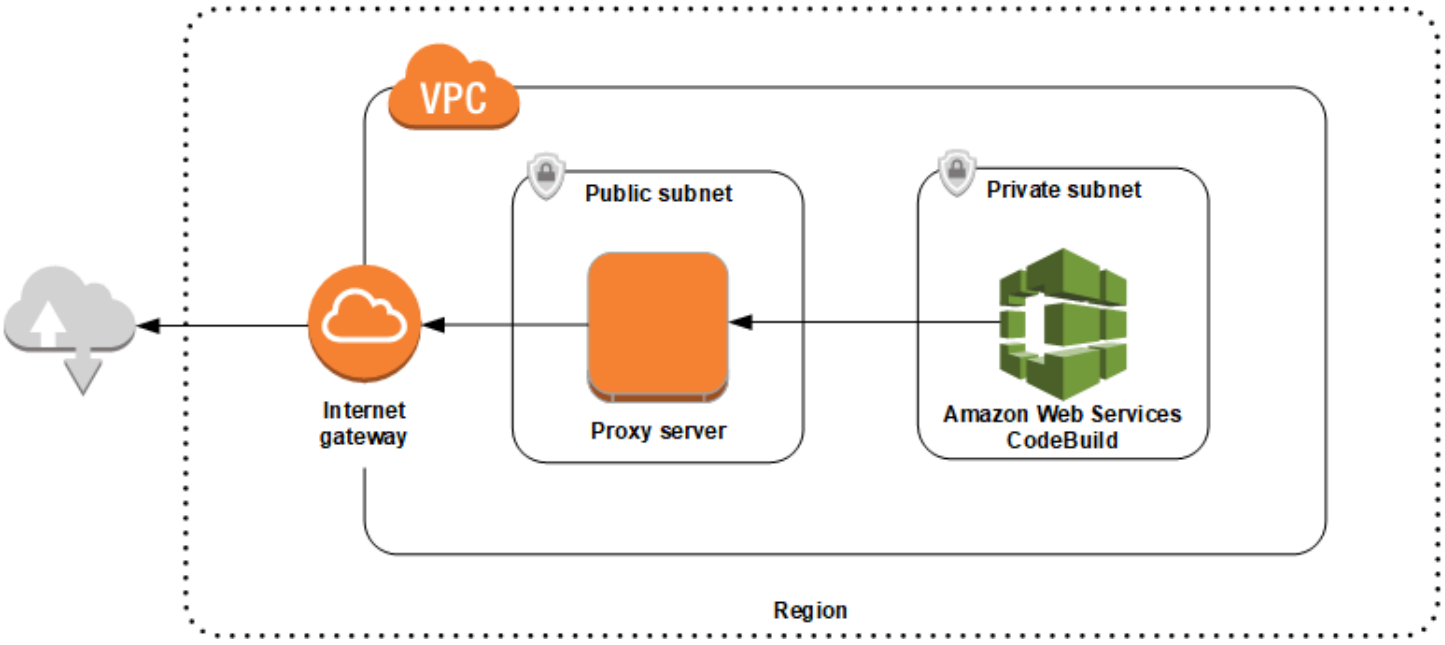
프록시 서버에서 CodeBuild를 실행하기 위해 필요한 구성 요소

투명 또는 명시적 프록시 서버에서 AWS CodeBuild를 실행하려면 다음 구성 요소가 필요합니다.

- VPC
- 프록시 서버용 VPC에 퍼블릭 서브넷 1개.
- CodeBuild용 VPC에 프라이빗 서브넷 1개.

- VPC와 인터넷 간 통신을 허용하는 인터넷 게이트웨이.

다음 다이어그램은 구성 요소가 상호 작용하는 방식을 보여 줍니다.



VPC, 서브넷 및 네트워크 게이트웨이 설정

다음은 투명 또는 명시적 프록시 서버에서 AWS CodeBuild를 실행하는 데 필요한 단계입니다.

1. VPC를 생성합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 생성을 참조하세요.
2. VPC에서 서브넷 2개를 만듭니다. 하나는 프록시 서버가 실행되는 `Public Subnet`이라는 퍼블릭 서브넷입니다. 다른 하나는 CodeBuild가 실행되는 `Private Subnet`이라는 프라이빗 서브넷입니다.

   자세한 내용은 VPC에서 서브넷 만들기 단원을 참조하십시오.
3. 인터넷 게이트웨이를 생성하여 VPC에 연결합니다. 자세한 내용은 인터넷 게이트웨이 생성 및 연결을 참조하십시오.
4. 기본 라우팅 테이블에 VPC(0.0.0.0/0)에서 인터넷 게이트웨이로 가는 트래픽을 라우팅하는 규칙을 추가합니다. 자세한 내용은 라우팅 테이블에 경로 추가 및 라우팅 테이블에서 경로 제거 단원을 참조하십시오.
5. VPC의 기본 보안 그룹에 VPC(0.0.0.0/0)로부터 들어오는 SSH 트래픽(TCP 22)을 허용하는 규칙을 추가합니다.
6. Amazon EC2 인스턴스를 시작하려면 Amazon EC2 사용 설명서에서 인스턴스 시작 마법사를 사용하여 인스턴스 시작의 지침을 따르세요. 마법사를 실행할 때 다음 옵션을 선택합니다.

- 인스턴스 유형 선택에서 Amazon Linux Amazon Machine Image(AMI)를 선택합니다.
- 서브넷에서 이 주제의 앞부분에서 만든 퍼블릭 서브넷을 선택합니다. 제안된 이름을 사용했다면 퍼블릭 서브넷입니다.
- [Auto-assign Public IP]에서 [Enable]을 선택합니다.
- 보안 그룹 구성 페이지의 보안 그룹 할당에서 Select an existing security group(기존 보안 그룹 선택)을 선택합니다. 그런 다음 기본 보안 그룹을 선택합니다.
- 시작을 선택한 후 기존 키 페어를 하나 선택하거나 생성합니다.

다른 옵션은 모두 기본 설정을 선택합니다.

7. EC2 인스턴스가 실행되면 원본/대상 확인을 비활성화합니다. 자세한 내용은 Amazon VPC 사용 설명서의 소스/대상 확인 비활성화를 참조하세요.
8. VPC에서 라우팅 테이블을 만듭니다. 라우팅 테이블에 인터넷으로 향하는 트래픽을 프록시 서버로 라우팅하는 규칙을 추가합니다. 이 라우팅 테이블을 프라이빗 서브넷과 연결합니다. 이 단계는 CodeBuild가 실행되는 프라이빗 서브넷 내 인스턴스의 아웃바운드 요청이 항상 프록시 서버를 통해 라우팅되도록 하는 데 필요합니다.

프록시 서버 설치 및 구성

선택할 수 있는 프록시 서버는 많습니다. 여기서는 오픈 소스 프록시 서버 Squid를 사용하여 프록시 서버에서 AWS CodeBuild를 실행하는 방법을 시연합니다. 동일한 개념을 다른 프록시 서버에도 적용할 수 있습니다.

Squid를 설치하려면 다음 명령을 실행하여 yum repo를 사용합니다.

```
sudo yum update -y
sudo yum install -y squid
```

Squid를 설치한 후 이 주제의 뒷부분에 나오는 지침에 따라 `squid.conf` 파일을 편집합니다.

HTTPS 트래픽에 대해 Squid 구성

HTTPS의 경우, TLS(전송 계층 보안) 연결에서 HTTP 트래픽이 캡슐화된 것입니다. Squid는 SslPeekAndSplice라는 기능을 사용하여 TLS 초기화에서 요청된 인터넷 호스트를 포함하는 SNI(서버 이름 표시)를 검색합니다. 이는 Squid가 HTTPS 트래픽을 해독할 필요가 없도록 하기 위해 필요합니다. SslPeekAndSplice를 활성화하려면 Squid가 인증서를 요구합니다. OpenSSL을 사용하여 이 인증서를 만듭니다.

```
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**
>
> HTTP의 경우, Squid를 구성할 필요가 없습니다. 모든 HTTP/1.1 요청 메시지에서 요청되는 인터넷 호스를 지정하는 호스트 헤더 필드를 검색할 수 있습니다.

명시적 프록시 서버에서 CodeBuild 실행

주제

- Squid를 명시적 프록시 서버로 구성
- CodeBuild 프로젝트 생성
- 명시적 프록시 서버 샘플 squid.conf 파일

명시적 프록시 서버에서 AWS CodeBuild를 실행하려면 프록시 서버가 외부 사이트와의 트래픽을 허용 또는 거부하도록 구성한 다음, HTTP_PROXY 및 HTTPS_PROXY 환경 변수를 구성해야 합니다.

Squid를 명시적 프록시 서버로 구성

Squid를 명시적 프록시 서버로 구성하려면 다음과 같이 `/etc/squid/squid.conf` 파일을 수정해야 합니다.

- 다음 기본 ACL(액세스 제어 목록) 규칙을 제거합니다.

```
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
```

제거한 기본 ACL 규칙 대신 다음 규칙을 추가합니다. 첫 번째 줄은 VPC의 요청을 허용합니다. 다음 두 줄은 프록시 서버에 AWS CodeBuild가 사용할 수 있는 대상 URL에 대한 액세스 권한을 부여합니다. 마지막 줄의 정규식을 편집하여 AWS 리전의 S3 버킷 또는 CodeCommit 리포지토리를 지정합니다. 예:

- 소스가 Amazon S3일 경우 acl download_src dstdom_regex .*s3\.us-west-1\.amazonaws\.com 명령을 사용하여 `us-west-1` 리전 내 S3 버킷에 대한 액세스 권한을 부여합니다.
- 소스가 AWS CodeCommit인 경우 `git-codecommit.<your-region>.amazonaws.com`을 사용하여 허용 목록에 AWS 리전을 추가합니다.

```
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
```

- `http_access allow localnet`을 다음으로 바꿉니다.

```
http_access allow localnet allowed_sites
http_access allow localnet download_src
```

- 빌드가 로그 및 아티팩트를 업로드하도록 하려면 다음 중 하나를 수행하십시오.

  1. `http_access deny all` 문 앞에 다음 문을 삽입합니다. 이를 통해 CodeBuild는 CloudWatch와 Amazon S3에 액세스할 수 있습니다. CodeBuild에서 CloudWatch Logs를 생성하려면 CloudWatch에 대한 액세스 권한이 필요합니다. Amazon S3에 대한 액세스는 아티팩트 및 Amazon S3 캐싱을 업로드하기 위해 필요합니다.

     -
     ```
     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
     ```
     - `squid.conf`를 저장한 후 다음 명령을 실행합니다.

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service squid restart
```

2. buildspec 파일에 `proxy`를 추가합니다. 자세한 내용은 buildspec 구문 섹션을 참조하세요.

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
      - command
```

> **Note**
>
> RequestError 시간 초과 오류가 발생할 경우 RequestError 프록시 서버에서 실행할 CodeBuild 때 시간 초과 오류가 발생했습니다. 단원을 참조하십시오.

자세한 내용은 이 주제의 후반부에서 명시적 프록시 서버 샘플 `squid.conf` 파일 섹션을 참조하세요.

CodeBuild 프로젝트 생성

명시적 프록시 서버에서 AWS CodeBuild를 실행하려면 프로젝트 수준에서 HTTP_PROXY 및 HTTPS_PROXY 환경 변수를 프록시 서버용으로 생성한 EC2 인스턴스의 프라이빗 IP 주소와 포트 3128로 설정합니다. 프라이빗 IP 주소는 `http://`*`your-ec2-private-ip-address`*`:3128`과 비슷합니다. 자세한 정보는 AWS CodeBuild에서 빌드 프로젝트 생성 및 AWS CodeBuild에서 빌드 프로젝트 설정 변경 섹션을 참조하세요.

다음 명령을 사용하여 Squid 프록시 액세스 로그를 확인합니다.

```
sudo tail -f /var/log/squid/access.log
```

명시적 프록시 서버 샘플 **`squid.conf`** 파일

다음은 명시적 프록시 서버용으로 구성된 `squid.conf` 파일의 예입니다.

```
  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
```

투명 프록시 서버에서 CodeBuild 실행

투명 프록시 서버에서 AWS CodeBuild를 실행하려면 프록시 서버가 상호 작용하는 웹 사이트 및 도메인에 액세스할 수 있도록 구성해야 합니다.

Squid를 투명 프록시 서버로 구성

프록시 서버를 투명으로 구성하려면 액세스하려는 도메인 및 웹 사이트에 대한 액세스 권한을 부여해야 합니다. 투명 프록시 서버에서 AWS CodeBuild를 실행하려면 `amazonaws.com`에 대한 액세스 권한을 부여해야 합니다. 또한 CodeBuild가 사용하는 다른 웹 사이트에 대해서도 액세스 권한을 부여해야 합니다. 이러한 웹 사이트는 CodeBuild 프로젝트를 생성하는 방식에 따라 달라집니다. 예를 들어 GitHub, Bitbucket, Yum, Maven 같은 리포지토리용 웹 사이트입니다. Squid에 특정 도메인 및 웹 사이트에 대한 액세스 권한을 부여하려면 다음과 같은 명령을 사용하여 `squid.conf` 파일을 업데이트합

니다. 이 샘플 명령은 amazonaws.com, github.com 및 bitbucket.com에 대한 액세스 권한을 부여합니다. 이 샘플을 편집하여 다른 웹 사이트에 대한 액세스 권한을 부여할 수 있습니다.

```
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
```

프라이빗 서브넷에 있는 인스턴스에서 오는 수신 요청은 Squid 포트로 리디렉션해야 합니다. Squid는 HTTP 트래픽의 경우 포트 3129(80이 아님)에서 수신 대기하고, HTTPS 트래픽의 경우 포트 3130(443이 아님)에서 수신 대기합니다. iptables 명령을 사용하여 트래픽을 라우팅합니다.

```
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
```

CodeBuild 프로젝트 생성

프록시 서버를 구성하면 더 이상 구성할 필요 없이 프라이빗 서브넷에서 AWS CodeBuild를 사용할 수 있습니다. 모든 HTTP 및 HTTPS 요청이 퍼블릭 프록시 서버를 통과합니다. 다음 명령을 사용하여 Squid 프록시 액세스 로그를 확인합니다.

```
sudo tail -f /var/log/squid/access.log
```

프록시 서버에서 패키지 관리자 및 기타 도구 실행

프록시 서버에서 패키지 관리자와 같은 도구를 실행하려면

1. `squid.conf` 파일에 문을 추가하여 프록시 서버의 허용 목록에 도구를 추가합니다.
2. 프록시 서버의 프라이빗 엔드포인트를 가리키는 줄을 buildspec 파일에 추가합니다.

다음 예제는 `apt-get`, `curl` 및 `maven`에서 이렇게 하는 방법을 보여줍니다. 다른 도구를 사용하는 경우에도 동일한 원칙이 적용됩니다. CodeBuild가 프록시 서버의 엔드포인트를 인식하도록 `squid.conf` 파일의 허용 목록에 추가하고 buildspec 파일에 명령을 추가합니다.

프록시 서버에서 **`apt-get`**을 실행하려면

1. `squid.conf` 파일에 다음 문을 추가하여 프록시 서버의 허용 목록에 `apt-get`을 추가합니다. 처음 세 줄은 빌드 환경에서 `apt-get`을 실행하도록 허용합니다.

   ```
   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. `apt-get` 명령이 `/etc/apt/apt.conf.d/00proxy`에서 프록시 구성을 검색하도록 buildspec 파일에 다음 문을 추가합니다.

   ```
   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
```

프록시 서버에서 **curl**을 실행하려면

1. squid.conf 파일에 다음을 추가하여 빌드 환경의 허용 목록에 curl을 추가합니다.

```
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. curl이 프라이빗 프록시 서버를 사용하여 squid.conf에 추가한 웹 사이트에 액세스하도록 buildspec 파일에 다음 문을 추가합니다. 이 예제에서 웹 사이트는 google.com입니다.

```
curl -x <private-ip-of-proxy-server>:3128 https://www.google.com
```

프록시 서버에서 **maven**을 실행하려면

1. squid.conf 파일에 다음을 추가하여 빌드 환경의 허용 목록에 maven을 추가합니다.

```
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. buildspec 파일에 다음 문을 추가합니다.

```
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -
DproxyPort=3128
```

AWS CodeBuild의 빌드 프로젝트 및 빌드 작업

시작하려면 빌드 프로젝트 생성의 단계를 수행한 다음, 빌드 실행의 단계를 따르세요. 빌드 프로젝트 및 빌드에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 빌드 프로젝트 작업
- AWS CodeBuild의 빌드 작업

빌드 프로젝트 작업

이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다.

빌드 프로젝트 작업을 수행할 때 이러한 태스크를 수행할 수 있습니다.

주제

- AWS CodeBuild에서 빌드 프로젝트 생성
- 알림 규칙 생성
- AWS CodeBuild에서 빌드 프로젝트 이름 목록 보기
- AWS CodeBuild에서 빌드 프로젝트 세부 정보 보기
- AWS CodeBuild의 빌드 캐싱
- 에서 트리거 생성 AWS CodeBuild
- GitLab 연결
- 다음과 함께 웹후크 사용 AWS CodeBuild
- AWS CodeBuild에서 빌드 프로젝트 설정 변경
- AWS CodeBuild에서 빌드 프로젝트 삭제
- 공유 프로젝트 작업
- AWS CodeBuild에서 프로젝트 태그 지정
- Batch 빌드 인 AWS CodeBuild
- GitHub 액션 러너 인 AWS CodeBuild

- AWS CodeBuild의 퍼블릭 빌드 프로젝트

AWS CodeBuild에서 빌드 프로젝트 생성

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 빌드 프로젝트를 생성할 수 있습니다.

필수 조건

빌드 프로젝트를 생성하기 전에 빌드 계획의 질문에 답변하세요.

주제

- 빌드 프로젝트 만들기(콘솔)
- 빌드 프로젝트 생성(AWS CLI)
- 빌드 프로젝트 생성(AWS SDK)
- 빌드 프로젝트 생성(AWS CloudFormation)

빌드 프로젝트 만들기(콘솔)

https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.

CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.

빌드 프로젝트 생성을 선택합니다.

다음 섹션을 채웁니다. 완료되면 페이지 하단에서 빌드 프로젝트 생성을 선택합니다.

섹션:

- 프로젝트 구성
- 소스
- 환경
- Buildspec
- 배치 구성
- 아티팩트
- 로그

프로젝트 구성

프로젝트 이름

이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 AWS 계정별로 고유해야 합니다.

설명

선택적으로 빌드 프로젝트에 대한 설명을 입력하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.

빌드 배지

(선택 사항) 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 빌드 배치 활성화를 선택합니다. 자세한 정보는 빌드 배지 샘플을 참조하세요.

> **Note**
>
> 소스 공급자가 Amazon S3인 경우 빌드 배지가 적용되지 않습니다.

동시 빌드 제한 활성화

(선택 사항) 이 프로젝트의 동시 빌드 수를 제한하려면 다음 단계를 수행합니다.

1. 이 프로젝트에서 시작할 수 있는 동시 빌드 수 제한을 선택합니다.
2. 동시 빌드 제한에 이 프로젝트에 허용되는 최대 동시 빌드 수를 입력합니다. 이 제한은 계정에 설정된 동시 빌드 제한보다 클 수 없습니다. 계정 제한보다 큰 숫자를 입력하려고 하면 오류 메시지가 표시됩니다.

현재 빌드 수가 이 한도 이하인 경우에만 새 빌드가 시작됩니다. 현재 빌드 수가 이 한도에 도달하면 새 빌드가 제한되고 실행되지 않습니다.

추가 정보

(선택 사항) 태그에는 지원 AWS 서비스에서 사용할 태그의 이름과 값을 입력합니다. [Add row]를 사용하여 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다.

소스

소스 공급자

소스 코드 공급자 유형을 선택합니다. 다음 목록을 사용하여 소스 공급자에 알맞은 유형을 선택합니다.

> **Note**
>
> CodeBuild Bitbucket 서버는 지원하지 않습니다.

Amazon S3

버킷

소스 코드가 포함된 입력 버킷의 이름을 선택합니다.

S3 객체 키 또는 S3 폴더

소스 코드가 포함된 ZIP 파일의 이름 또는 폴더 경로를 입력합니다. S3 버킷의 모든 항목을 다운로드하려면 슬래시(/)를 입력합니다.

소스 버전

입력 파일의 빌드를 나타내는 객체의 버전 ID를 입력합니다. 자세한 내용은 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild 섹션을 참조하세요.

CodeCommit

리포지토리

사용할 리포지토리를 선택합니다.

참조 유형

분기, Git 태그 또는 커밋 ID를 선택하여 소스 코드 버전을 지정합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

이력이 지정된 커밋 수로 잘린 부분 복제를 생성하려면 선택합니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

Bitbucket

리포지토리

OAuth를 사용하여 연결 또는 Bitbucket 앱 암호를 사용하여 연결을 선택하고 지침에 따라 Bitbucket에 연결(또는 다시 연결)합니다.

퍼블릭 리포지토리 또는 계정의 리포지토리를 선택합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild단원을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

상태 컨텍스트의 경우 Bitbucket 커밋 상태의 `name` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

대상 URL에 Bitbucket 커밋 상태의 `url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 소스 코드를 빌드하려면 기본 소스 웹후크 이벤트에서 코드 변경 사항이 이 리포지토리에 푸시될 때마다 재빌드를 선택합니다. CodeBuild webhook 및 필터 그룹에 대한 자세한 내용은 Bitbucket Webhook 이벤트 섹션을 참조하세요.

GitHub

리포지토리

OAuth를 사용한 연결 또는 GitHub 개인용 액세스 토큰으로 연결을 선택하고 지침에 따라 연결(또는 재연결) GitHub 하고 액세스를 승인합니다. AWS CodeBuild

퍼블릭 리포지토리 또는 계정의 리포지토리를 선택합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild단원을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

상태 컨텍스트의 경우 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. GitHub 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

타겟 URL의 경우 GitHub 커밋 상태에 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 소스 코드를 빌드하려면 기본 소스 웹후크 이벤트에서 코드 변경 사항이 이 리포지토리에 푸시될 때마다 재빌드를 선택합니다. CodeBuild webhook 및 필터 그룹에 대한 자세한 내용은 GitHub 웹후크 이벤트 섹션을 참조하세요.

GitHub Enterprise Server

GitHub 엔터프라이즈 개인용 액세스 토큰

GitHub 엔터프라이즈 서버 샘플 섹션에서 개인용 액세스 토큰을 클립보드에 복사하는 방법을 참조하세요. 토큰을 텍스트 필드에 붙여넣고, 토큰 저장을 선택합니다.

> **Note**
>
> 개인 액세스 토큰은 한 번만 입력하고 저장하면 됩니다. CodeBuild 향후 모든 프로젝트에서 이 토큰을 사용합니다.

소스 버전

풀 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d 또는 5392f7과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

상태 컨텍스트의 경우 GitHub 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

타겟 URL의 경우 GitHub 커밋 상태에 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

비보안 SSL

GitHub Enterprise 프로젝트 리포지토리에 연결하는 동안 SSL 경고를 무시하려면 비보안 SSL 활성화를 선택합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 소스 코드를 빌드하려면 기본 소스 웹후크 이벤트에서 코드 변경 사항이 이 리포지토리에 푸시될 때마다 재빌드를 선택합니다. CodeBuild webhook 및 필터 그룹에 대한 자세한 내용은 GitHub 웹후크 이벤트 섹션을 참조하세요.

GitLab

Connection

를 사용하여 GitLab AWS CodeConnections계정을 연결하고 연결을 사용하여 타사 리포지토리를 빌드 프로젝트의 소스로 연결합니다.

기본 연결 또는 사용자 지정 연결을 선택합니다.

기본 연결은 모든 프로젝트에 기본 GitLab 연결을 적용합니다. 사용자 지정 연결은 계정의 기본 설정을 재정의하는 사용자 지정 GitLab 연결을 적용합니다.

기본 연결

계정에 연결된 기본 연결의 이름.

공급자와의 연결을 아직 만들지 않은 경우 지침을 GitLab(콘솔) 에 대한 연결 만들기 참조하십시오.

사용자 지정 연결

사용하려는 사용자 지정 연결의 이름을 선택합니다.

공급자에 GitLab(콘솔) 에 대한 연결 만들기 대한 연결을 아직 생성하지 않은 경우 지침을 참조하십시오.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

풀 리퀘스트 ID, 브랜치, 커밋 ID, 태그 또는 레퍼런스와 커밋 ID를 입력합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

GitLab Self Managed

Connection

를 사용하여 GitLab AWS CodeConnections계정을 연결하고 연결을 사용하여 타사 리포지토리를 빌드 프로젝트의 소스로 연결합니다.

기본 연결 또는 사용자 지정 연결을 선택합니다.

기본 연결은 모든 프로젝트에 기본 GitLab 자체 관리형 연결을 적용합니다. 사용자 지정 연결은 계정의 기본 설정을 재정의하는 사용자 지정 GitLab 자체 관리형 연결을 적용합니다.

기본 연결

계정에 연결된 기본 연결의 이름.

공급자에 대한 연결을 아직 만들지 않았다면 개발자 도구 콘솔 사용 설명서의 GitLab 자체 관리형 연결 만들기의 지침을 참조하십시오.

사용자 지정 연결

사용하려는 사용자 지정 연결의 이름을 선택합니다.

공급자에 대한 연결을 아직 생성하지 않은 경우 개발자 도구 콘솔 사용 설명서에서 GitLab 자체 관리형 연결 만들기를 참조하여 지침을 확인하십시오.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

풀 리퀘스트 ID, 브랜치, 커밋 ID, 태그 또는 레퍼런스와 커밋 ID를 입력합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

환경

프로비저닝 모델

다음 중 하나를 수행하십시오.

- 에서 관리하는 온디맨드 플릿을 사용하려면 온디맨드를 AWS CodeBuild선택합니다. 온디맨드 플릿을 사용하면 빌드에 필요한 컴퓨팅 CodeBuild 기능을 제공합니다. 빌드가 완료되면 머신이 파괴됩니다. 온디맨드 플릿은 완전 관리형이며, 수요 급증을 처리할 수 있는 자동 규모 조정 기능이 포함되어 있습니다.
- 에서 관리하는 예약 용량 플릿을 사용하려면 예약 용량을 선택한 다음 플릿 이름을 선택합니다. AWS CodeBuild 예약 용량 플릿을 사용하면 빌드 환경을 위한 전용 인스턴스 세트를 구성할 수 있습니다. 이러한 머신은 유휴 상태로 유지되므로 빌드 또는 테스트를 즉시 처리하고 빌드 기간을 단축할 수 있습니다. 예약 용량 플릿을 사용하면 머신이 상시 가동되므로 프로비저닝하는 한 계속해서 비용이 발생합니다.

자세한 내용은 에서 예약된 용량으로 작업하기 AWS CodeBuild을 참조하세요.

환경 이미지

다음 중 하나를 수행하십시오.

- 관리되는 AWS CodeBuild Docker 이미지를 사용하려면 관리 이미지를 선택한 다음 운영 체제, 런타임, 이미지 및 이미지 버전 중에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 계정의 Docker 이미지를 선택하십시오. AWS
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

> **Note**
>
> CodeBuild 사용자 지정 Docker 이미지의 경우 를 재정의합니다. `ENTRYPOINT`

컴퓨팅

다음 중 하나를 수행하십시오.

- EC2 컴퓨팅을 사용하려면 EC2를 선택합니다. EC2 컴퓨팅은 작업 실행 중에 최적화된 유연성을 제공합니다.
- Lambda 컴퓨팅을 사용하려면 Lambda를 선택하십시오. Lambda 컴퓨팅은 빌드에 최적화된 시작 속도를 제공합니다. Lambda는 시작 지연 시간이 짧아 더 빠른 빌드를 지원합니다. Lambda는 또한 자동으로 확장되므로 빌드가 실행될 때까지 대기하지 않아도 됩니다. 자세한 내용은 AWS Lambda 컴퓨트 인에서 작업하기 AWS CodeBuild을 참조하세요.

서비스 역할

다음 중 하나를 수행하십시오.

- 서비스 역할이 없는 경우 새 CodeBuild 서비스 역할을 선택하십시오. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 만들 때 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

추가 구성

제한 시간

5분에서 8시간 사이의 값을 지정합니다. 이 값을 지정한 후 완료되지 않으면 빌드가 CodeBuild 중지됩니다. [hours] 및 [minutes]가 비어 있는 경우 기본값인 60분이 사용됩니다.

권한 있음

(선택 사항) 이 빌드 프로젝트를 사용하여 Docker 이미지를 빌드하려는 경우에만 Docker 이미지를 빌드하거나 빌드에 높은 권한을 부여하려는 경우 이 플래그 활성화를 선택합니다. 그렇지 않으면 Docker 데몬과 상호 작용을 시도하는 모든 연결된 빌드가 실패합니다. 또한 빌드가 상호 작용할 수 있도록 Docker 데몬을 시작해야 합니다. 이를 수행하는 한 가지 방법은 다음 빌드 명령을 실행하여 빌드 사양의 `install` 단계에서 Docker 데몬을 초기화하는 것입니다. Docker 지원에서 제공하는 CodeBuild 빌드 환경 이미지를 선택한 경우에는 이 명령을 실행하지 마세요.

> **Note**
>
> 기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.

```
- 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

VPC로 CodeBuild 작업하려는 경우:

- VPC의 경우 사용할 VPC ID를 선택합니다. CodeBuild
- VPC 서브넷의 경우 사용하는 리소스가 포함된 서브넷을 선택합니다. CodeBuild
- VPC 보안 그룹의 경우 VPC의 리소스에 대한 액세스를 허용하는 데 CodeBuild 사용하는 보안 그룹을 선택합니다.

자세한 정보는 Amazon Virtual Private 클라우드와 AWS CodeBuild 함께 사용을 참조하세요.

컴퓨팅

사용 가능한 옵션 중 하나를 선택합니다.

환경 변수

사용할 빌드의 각 환경 변수에 대해 이름 및 값을 입력하고 유형을 선택합니다.

> **Note**
>
> CodeBuild 해당 AWS 지역의 환경 변수를 자동으로 설정합니다. buildspec.yml에 추가하지 않은 경우 다음 환경 변수를 설정해야 합니다.
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME
> - IMAGE_TAG

콘솔 및 AWS CLI 사용자는 환경 변수를 볼 수 있습니다. 환경 변수의 가시성에 대한 문제가 없다면 [Name] 및 [Value] 필드를 설정한 다음 [Type]을 [Plaintext]로 설정합니다.

AWS 액세스 키 ID, AWS 보안 액세스 키 또는 암호와 같은 민감한 값이 포함된 환경 변수를 Amazon EC2 Systems Manager 파라미터 스토어 AWS Secrets Manager또는 파라미터로 저장하는 것이 좋습니다.

Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 유형에서 파라미터를 선택합니다. [Name] 에는 CodeBuild 참조할 식별자를 입력합니다. 값에 Amazon EC2 Systems Manager Parameter Store에 저장되는 파라미터의 이름을 입력합니다. 예를 들어 `/CodeBuild/dockerLoginPassword`라는 이름의 파라미터를 사용하여 유형에서 파라미터를 선택합니다. 이름에 `LOGIN_PASSWORD`를 입력합니다. 값에 `/CodeBuild/dockerLoginPassword`를 입력합니다.

> **Important**
>
> Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 `/CodeBuild/`로 시작하는 파라미터 이름(예: `/CodeBuild/dockerLoginPassword`)으로 파라미터를 저장하는 것이 좋습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택하고 대화 상자에 표시되는 지시에 따릅니다. (이 대화 상자에서 KMS 키의 경우 계정 내 키의 AWS KMS ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는 이 키를 사용하여 저장 중에 파라미터 값을 암호화하고 검색 중에 복호화합니다.) 콘솔을 사용하여 파라미터를 생성하는 경우, CodeBuild 콘솔은 파라미터가 저장되는 `/CodeBuild/` 대로 파라미터 이름을 시작합니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.

빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `ssm:GetParameters` 작업을 허용해야 합니다. 이전에 새 서비스 역할을 선택한 경우 빌드 프로젝트의 기본 서비스 역할에 이 작업을 CodeBuild 포함하세요. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 파라미터 이름으로 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 파라미터 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 파라미터 이름에만 액세스할 수 있기 때문입니다.
새 서비스 역할을 선택하면 Amazon EC2 Systems Manager Parameter Store에 있는 `/CodeBuild/` 네임스페이스의 모든 파라미터를 해독할 권한이 서비스 역할에 포함됩니다.
사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.
`CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.
- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

Secrets Manager를 사용하는 경우 유형으로 Secrets Manager로 선택합니다. 이름에는 CodeBuild 참조할 식별자를 입력합니다. 값에 *`secret-id`*`:`*`json-key`*`:`*`version-stage`*`:`*`version-id`* 패턴을 사용하여 `reference-key`를 입력합니다. 자세한 내용은 Secrets Manager reference-key in the buildspec file을 참조하세요.

> **Important**
>
> Secrets Manager를 사용하는 경우 이름이 `/CodeBuild/`로 시작하는 암호를 저장하는 것이 좋습니다(예: `/CodeBuild/dockerLoginPassword`). 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.
> 빌드 프로젝트가 Secrets Manager에 저장된 암호를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `secretsmanager:GetSecretValue` 작업을 허용해야 합니다. 이전에 새 서비스 역할을 선택한 경우 빌드 프로젝트의 기본 서비스 역할에 이 작업을 CodeBuild 포함시키십시오. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
> 빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 보안 암호 이름으로 Secrets Manager에 저장된 암호를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 보안 암호 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 암호 이름에만 액세스할 수 있기 때문입니다.
> 새 서비스 역할을 선택하면 에 있는 `/CodeBuild/` 네임스페이스의 모든 암호를 해독할 권한이 서비스 역할에 포함됩니다.

Buildspec

빌드 사양

다음 중 하나를 수행하십시오.

- 소스 코드에 buildspec 파일이 있는 경우 Use a buildspec file(빌드 사양 파일 사용) 을 선택합니다. 기본적으로 소스 코드 루트 `buildspec.yml` 디렉터리에서 이름이 지정된 파일을 CodeBuild 찾습니다. buildspec 파일이 다른 이름이나 위치를 사용하는 경우 Buildspec 이름에 소스 루트의 경로를 입력합니다(예: `buildspec-two.yml` 또는 `configuration/buildspec.yml`). buildspec 파일이 S3 버킷에 있는 경우 해당 파일은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).
- 소스 코드에 buildspec 파일이 포함되어 있지 않거나, 소스 코드의 루트 디렉터리에 있는 `buildspec.yml` 파일의 `build` 단계에 지정된 것과 다른 빌드 명령 세트를 실행하려는 경우 빌드 명령 삽입을 선택합니다. 빌드 명령의 `build` 단계에서 실행하려는 명령을 입력합니다. 명령이 여러 개인 경우 각 명령을 `&&`로 구분합니다(예: `mvn test && mvn package`). 다른 구문에서 명령을 실행하려는 경우 또는 `build` 단계에 대해 특히 긴 명령 목록이 있는 경우에는 소스 코

드 루트 디렉터리에 `buildspec.yml` 파일을 추가하고, 이 파일에 명령을 추가한 다음, 소스 코드 루트 디렉터리에서 buildspec.yml 사용을 선택합니다.

자세한 내용은 buildspec 참조을 참조하세요.

배치 구성

빌드 그룹을 단일 작업으로 실행할 수 있습니다. 자세한 정보는 Batch 빌드 인 AWS CodeBuild을 참조하세요.

배치 구성 정의

이 프로젝트에서 배치 빌드를 허용하려면 선택합니다.

배치 서비스 역할

배치 빌드에 대한 서비스 역할을 제공합니다.

다음 중 하나를 선택합니다.

- 배치 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 서비스 역할에 새 역할의 이름을 입력합니다.
- 배치 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 서비스 역할에서 서비스 역할을 선택합니다.

배치 빌드는 배치 구성에 새로운 보안 역할을 도입합니다. 일괄 처리의 일부로 빌드를 실행하려면 사용자 대신 `StartBuild`,`StopBuild`, `RetryBuild` 작업을 호출할 수 CodeBuild 있어야 하므로 이 새 역할이 필요합니다. 고객은 다음과 같은 두 가지 이유로 빌드에 사용하는 것과 동일한 역할이 아닌 새 역할을 사용해야 합니다.

- 빌드 역할 `StartBuild`, `StopBuild` 및 `RetryBuild` 권한을 부여하면 단일 빌드에서 buildspec을 통해 더 많은 빌드를 시작할 수 있습니다.
- CodeBuild 배치 빌드는 일괄 빌드에 사용할 수 있는 빌드 및 컴퓨팅 유형의 수를 제한하는 제한을 제공합니다. 빌드 역할에 이러한 권한이 있는 경우 빌드 자체가 이러한 제한을 우회할 수 있습니다.

배치에 허용되는 컴퓨팅 유형

배치에 허용되는 컴퓨팅 유형을 선택합니다. 해당하는 항목을 모두 선택합니다.

배치에 허용되는 최대 빌드 수

배치에 허용되는 최대 빌드 수를 입력합니다. 이 제한을 초과하는 배치는 실패합니다.

배치 제한 시간

배치 빌드가 완료되는 최대 시간을 입력합니다.

아티팩트 결합

배치의 모든 아티팩트를 단일 위치로 결합을 선택하면 배치의 모든 아티팩트가 단일 위치로 결합됩니다.

배치 보고서 모드

배치 빌드에 대해 원하는 빌드 상태 보고서 모드를 선택합니다.

> **Note**
>
> 이 필드는 프로젝트 소스가 Bitbucket 또는 GitHub Enterprise인 경우에만 사용할 수 있으며, 소스 아래에서 빌드 시작 및 종료를 선택하면 소스 공급자에게 빌드 상태를 보고할 수 있습니다. GitHub

빌드 집계

배치의 모든 빌드 상태를 단일 상태 보고서로 통합하려면 선택합니다.

개별 빌드

배치에 있는 모든 빌드의 빌드 상태를 별도로 보고하려면 선택합니다.

아티팩트

Type

다음 중 하나를 수행하십시오.

- 빌드 출력 결과물을 생성하지 않으려면 [No artifacts]를 선택합니다. 빌드 테스트만 실행하고 있는 경우 또는 Amazon ECR 리포지토리에 도커 이미지를 푸시하려는 경우에 이 작업을 원할 수 있습니다.
- S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
  - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. (ZIP 파일을 출력하고 ZIP 파일에 파일 확장명을 넣으려는 경우, ZIP 파일 이름 뒤에 이를 포함하십시오.)

- buildspec 파일에 지정된 이름으로 콘솔에서 지정한 이름을 재정의하려는 경우 의미 체계 버전 관리 사용을 선택합니다. buildspec 파일의 이름은 빌드 시 계산되며 Shell 명령 언어를 사용합니다. 예를 들어 결과물 이름이 항상 고유하도록 날짜와 시간을 결과물 이름에 추가할 수 있습니다. 고유한 결과물 이름을 사용하면 결과물을 덮어쓰지 않을 수 있습니다. 자세한 정보는 buildspec 구문을 참조하세요.
- [Bucket name]에서 출력 버킷의 이름을 선택합니다.
- 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.
- 빌드 아티팩트를 암호화하지 않으려면 Remove artifacts encryption(결과물 암호화 제거)을 선택합니다.

각각 원하는 보조 아티팩트 세트마다 다음과 같이 실행합니다.

1. Atrifact identifier(아티팩트 식별자)에서 128자 미만으로 영숫자와 밑줄만 포함된 값을 입력합니다.
2. Add artifact(아티팩트 추가)를 선택합니다.
3. 이전 단계에 따라 보조 결과물을 구성합니다.
4. Save artifact(아티팩트 저장)를 선택합니다.

추가 구성

암호화 키

(선택 사항) 다음 중 하나를 수행하십시오.

- 계정에서 Amazon S3에 대한 AWS 관리형 키 를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키를 비워 둡니다. 이 값이 기본값입니다.
- 고객 관리형 키를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키에 KMS 키의 ARN을 입력합니다. `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* 형식을 사용합니다.

캐시 유형

Cache type(캐시 유형)에서 다음 중 하나를 선택합니다.

- 캐시를 사용하지 않으려면 [No cache]를 선택합니다.
- Amazon S3 캐시를 사용하려면 Amazon S3를 선택하고 다음을 수행합니다.

- 버킷에서 캐시가 저장된 S3 버킷의 이름을 선택합니다.
- (선택 사항) 캐시 경로 접두사에 Amazon S3 경로 접두사를 입력합니다. Cache path prefix(캐시 경로 접두사) 값은 디렉터리 이름과 비슷합니다. 따라서 캐시를 버킷의 동일한 디렉터리에 저장할 수 있습니다.

> **Important**
>
> 경로 접두사 끝에 후행 슬래시(/)를 추가하지 마십시오.

- 로컬 캐시를 사용하려면 로컬을 선택한 다음 하나 이상의 로컬 캐시 모드를 선택해야 합니다.

> **Note**
>
> Docker 계층 캐시 모드는 Linux에서만 사용할 수 있습니다. 이 모드를 선택할 경우 프로젝트를 권한이 있는 모드에서 실행해야 합니다.

캐시를 사용하면 빌드 환경의 재사용 가능한 특정 부분이 캐시에 저장되고 빌드 전반에서 사용되기 때문에 상당한 빌드 시간을 절약할 수 있습니다. buildspec 파일에 캐시를 지정하는 것에 대한 자세한 정보는 buildspec 구문 단원을 참조하십시오. 캐싱에 대한 자세한 정보는 AWS CodeBuild의 빌드 캐싱을 참조하십시오.

로그

생성하려는 로그를 선택합니다. Amazon CloudWatch 로그, Amazon S3 로그 또는 둘 다를 생성할 수 있습니다.

CloudWatch

Amazon CloudWatch 로그 로그를 원하는 경우:

CloudWatch 로그

CloudWatch 로그를 선택합니다.

그룹 이름

Amazon CloudWatch Logs 로그 그룹 이름을 입력합니다.

스트림 이름

Amazon CloudWatch Logs 로그 스트림 이름을 입력합니다.

S3

Amazon S3 로그를 원할 경우:

S3 로그

S3 로그를 선택합니다.

버킷

로그에 대한 S3 버킷 이름을 선택합니다.

경로 접두사

로그의 접두사를 입력합니다.

S3 로그 암호화 비활성화

S3 로그를 암호화하지 않으려면 선택합니다.

빌드 프로젝트 생성(AWS CLI)

with 사용에 대한 자세한 내용은 CodeBuild 를 참조하십시오명령줄 참조. AWS CLI

를 사용하여 CodeBuild 빌드 프로젝트를 만들려면 JSON 형식의 프로젝트 구조를 만들고 구조를 채운 create-project다음 명령을 호출하여 프로젝트를 생성합니다. AWS CLI

JSON 파일 생성

`--generate-cli-skeleton` 옵션을 사용하여 `create-project` 명령으로 스켈레톤 JSON 파일을 생성합니다.

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

이렇게 하면 *<json-file>*로 지정된 경로와 파일 이름을 가진 JSON 파일이 생성됩니다.

JSON 파일 채우기

JSON 데이터를 다음과 같이 수정하고 결과를 저장합니다.

```
{
  "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>
}
```

다음을 바꿉니다.

이름

필수 사항입니다. 이 빌드 프로젝트의 이름입니다. 이 이름은 계정의 모든 빌드 프로젝트에서 고유해야 합니다. AWS

description

선택 사항입니다. 이 빌드 프로젝트의 설명입니다.

source

필수 사항입니다. 이 빌드 프로젝트의 소스 코드 설정에 대한 정보가 들어 있는 ProjectSource객체입니다. source 객체를 추가한 후에는 를 사용해 소스를 최대 12개까지 더 추가할 수 있습니다. 이러한 설정에는 다음이 포함됩니다.

source/type

필수 사항입니다. 빌드할 소스 코드가 포함된 리포지토리의 유형입니다. 유효한 값으로는 다음이 포함됩니다.

- CODECOMMIT
- CODEPIPELINE
- GITHUB
- GITHUB_ENTERPRISE
- GITLAB

- GITLAB_SELF_MANAGED
- BITBUCKET
- S3
- NO_SOURCE

NO_SOURCE를 사용할 경우 프로젝트에 소스가 없으므로 buildspec은 파일이 될 수 없습니다. 대신에 buildspec 속성을 사용하여 buildspec에 대해 YAML 형식이 지정된 문자열을 지정해야 합니다. 자세한 정보는 소스 샘플이 없는 프로젝트을 참조하세요.

source/location

*<source-type>*을 CODEPIPELINE으로 설정하지 않은 경우 필수입니다. 지정한 리포지토리 유형의 소스 코드 위치입니다.

- 소스 코드와 buildspec 파일이 들어 있는 저장소의 HTTPS 복제 URL (예:) 의 경우 CodeCommit `https://git-codecommit.`*<region-id>*`.amazonaws.com/v1/repos/`*<repo-name>*
- Amazon S3의 경우 빌드 입력 버킷 이름 다음에 소스 코드 및 buildspec이 포함된 ZIP 파일의 경로와 이름이 옵니다. 예:
  - 입력 버킷의 루트에 있는 ZIP 파일의 경우: *<bucket-name>*/*<object-name>*.zip
  - 입력 버킷의 하위 폴더에 있는 ZIP 파일의 경우: *<bucket-name>*/*<subfoler-path>*/*<object-name>*.zip
- 소스 코드와 빌드스펙 파일이 들어 있는 저장소의 HTTPS 복제 URL의 경우 GitHub URL에 github.com이 포함되어야 합니다. AWS 계정을 계정에 연결해야 합니다. GitHub 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 만드세요.
  - [Authorize application]을 선택합니다. (GitHub 계정에 연결한 후에는 빌드 프로젝트 생성을 완료할 필요가 없습니다. CodeBuild 콘솔을 닫을 수 있습니다.
- GitHub 엔터프라이즈 서버의 경우 소스 코드와 buildspec 파일이 들어 있는 저장소의 HTTP 또는 HTTPS 복제 URL입니다. 또한 계정을 엔터프라이즈 서버 AWS 계정에 연결해야 합니다 GitHub. 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 만드십시오.
  1. GitHub 엔터프라이즈 서버에서 개인용 액세스 토큰을 생성합니다.
  2. 이 토큰을 클립보드에 복사하면 프로젝트를 만들 때 사용할 수 있습니다 CodeBuild . 자세한 내용은 GitHub 도움말 웹 사이트의 명령줄용 개인용 액세스 토큰 만들기를 참조하십시오.
  3. 콘솔을 사용하여 CodeBuild 프로젝트를 만들 때는 소스에서 소스 공급자로 GitHub Enterprise를 선택합니다.
  4. [Personal Access Token]에서 클리보드에 복사한 토큰을 붙여 넣습니다. 토큰 저장을 선택합니다. 이제 CodeBuild 계정이 GitHub 엔터프라이즈 서버 계정에 연결되었습니다.

- 소스 코드와 buildspec 파일이 들어 있는 저장소의 HTTPS 복제 URL은 GitLab 자체 관리됩니다. GitLab 를 사용하는 경우 URL에 gitlab.com이 포함되어야 GitLab 한다는 점에 유의하세요. GitLab 자체 관리형을 사용하는 경우 URL에 gitlab.com이 포함되지 않아도 됩니다. AWS 계정을 본인 계정 또는 자체 관리 계정에 연결해야 합니다 GitLab. GitLab 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 만드세요.
  - 개발자 도구 탐색 창에서 설정, 연결, 연결 생성을 선택합니다. 이 페이지에서 연결 GitLab 또는 GitLab 자체 관리형 연결을 만든 다음 Connect to GitLab (연결 대상) 를 선택합니다.
- Bitbucket의 경우 HTTPS가 소스 코드 및 buildspec 파일을 포함하는 리포지토리에 URL을 복제합니다. URL에 bitbucket.org가 포함되어야 합니다. 또한 계정을 Bitbucket AWS 계정에 연결해야 합니다. 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 만드세요.
  1. 콘솔을 사용하여 Bitbucket에 연결(또는 재연결)하면 Bitbucket [Confirm access to your account] 페이지에서 [Grant access]를 선택합니다. (Bitbucket 계정에 연결한 후에는 빌드 프로젝트 생성을 완료할 필요가 없습니다. CodeBuild 콘솔을 닫을 수 있습니다.
- 의 AWS CodePipeline경우 `location` 값을 지정하지 마십시오`source`. CodePipeline 에서 CodePipeline 파이프라인을 생성할 때 파이프라인의 소스 단계에서 소스 코드 위치를 지정하므로 이 값을 무시합니다.

소스/ gitCloneDepth

선택 사항입니다. 다운로드할 이력의 수준입니다. 최소값은 0입니다. 이 값이 0이거나, 25를 초과하거나, 지정되지 않은 경우 각 빌드 프로젝트에서 전체 이력이 다운로드됩니다. 소스 유형이 Amazon S3일 경우 이 값이 지원되지 않습니다.

source/buildspec

선택 사항입니다. 사용할 빌드 사양 정의 또는 파일입니다. 이 값을 제공하지 않거나 빈 문자열로 설정하는 경우 소스 코드에 루트 디렉터리의 `buildspec.yml` 파일이 포함되어 있어야 합니다. 이 값이 설정된 경우 인라인 buildspec 정의, 기본 소스의 루트 디렉터리에 상대적인 대체 buildspec 파일의 경로 또는 S3 버킷의 경로가 될 수 있습니다. 버킷은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`). 자세한 정보는 buildspec 파일 이름 및 스토리지 위치을 참조하세요.

source/auth

사용하지 않습니다. 이 객체는 CodeBuild 콘솔에서만 사용됩니다.

소스/ reportBuildStatus

빌드의 시작 및 완료 상태를 소스 공급자에게 보낼지 여부를 지정합니다. GitHub 엔터프라이즈 서버 또는 Bitbucket이 아닌 GitHub 다른 소스 공급자로 이 값을 설정하면 `invalidInputException` an이 발생합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

소스/ buildStatusConfig

CodeBuild 빌드 프로젝트가 빌드 상태를 소스 공급자에게 보고하는 방법을 정의하는 정보가 들어 있습니다. 이 옵션은 소스 유형이 `GITHUB`, `GITHUB_ENTERPRISE` 또는 `BITBUCKET`인 경우에만 사용됩니다.

소스//컨텍스트 buildStatusConfig

Bitbucket 소스의 경우 이 파라미터는 Bitbucket 커밋 상태의 `name` 파라미터에 사용됩니다. GitHub 소스의 경우 이 매개변수는 GitHub 커밋 상태의 `context` 매개변수에 사용됩니다.

예를 들어, CodeBuild 환경 변수를 사용하여 빌드 번호와 웹후크 트리거를 `context` 포함할 수 있습니다.

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

그 결과 webhook 풀 요청 이벤트에 의해 트리거된 빌드 #24에 대해 다음과 같은 컨텍스트가 나타납니다.

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

소스//타겟 URL buildStatusConfig

Bitbucket 소스의 경우 이 파라미터는 Bitbucket 커밋 상태의 `url` 파라미터에 사용됩니다. GitHub 소스의 경우 이 매개변수는 커밋 상태의 `target_url` 매개변수에 사용됩니다. GitHub

예를 들어 `targetUrl`를 `https://aws.amazon.com/codebuild/`*`<path to build>`*로 설정하면 커밋 상태가 이 URL에 연결됩니다.

URL에 추가 정보를 `targetUrl` 추가하기 위해 에 CodeBuild 환경 변수를 포함할 수도 있습니다. 예를 들어 URL에 빌드 리전을 영역을 추가하려면 다음과 같이 `targetUrl`을 설정합니다.

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

빌드 리전이 us-east-2이면 다음과 같이 확장됩니다.

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

소스/ gitSubmodulesConfig

선택 사항입니다. Git 하위 모듈 구성에 대한 정보입니다. CodeCommit, GitHub, GitHub 엔터프라이즈 서버 및 비트버킷에만 사용됩니다.

소스/ gitSubmodulesConfig 페치 서브 모듈

리포지토리에 Git 하위 모듈을 포함하려면 fetchSubmodules를 true로 설정합니다. 포함된 Git 하위 모듈은 HTTPS로 구성해야 합니다.

소스/ InsecureSsl

선택 사항입니다. GitHub 엔터프라이즈 서버에만 사용됩니다. GitHub Enterprise Server 프로젝트 리포지토리에 연결하는 동안 TLS 경고를 true 무시하려면 이 값을 로 설정하십시오. 기본값은 false입니다. InsecureSsl은 테스트 용도로만 사용해야 합니다. 프로덕션 환경에 사용하면 안 됩니다.

source/sourceIdentifier

프로젝트 소스의 사용자 정의 식별자입니다. 기본 소스의 경우 선택 사항입니다. 보조 소스에 필요합니다.

secondarySources

선택 사항입니다. 빌드 프로젝트의 보조 소스에 대한 정보가 포함된 ProjectSource개체 배열입니다. 보조 소스는 12개까지 추가할 수 있습니다. secondarySources 객체는 객체에서 사용하는 것과 동일한 속성을 사용합니다. 보조 소스 객체에는 sourceIdentifier가 필요합니다.

secondarySourceVersions

선택 사항입니다. ProjectSourceVersion 객체 어레이. 빌드 레벨에서 secondarySourceVersions이 지정되어 있으면 그 버전이 이 버전보다 우선합니다.

sourceVersion

선택 사항입니다. 이 프로젝트에 대해 빌드할 빌드 입력의 버전입니다. 지정하지 않으면 최신 버전이 사용됩니다. 지정하면 다음 중 하나여야 합니다.

- For CodeCommit, 사용할 커밋 ID, 브랜치 또는 Git 태그
- F의 경우 GitHub, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 풀 리퀘스트 ID, 브랜치 이름 또는 태그 이름. 풀 요청 ID가 지정된 경우 `pr/pull-request-ID` 형식을 사용해야 합니다(예: `pr/25`). 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. 지정되지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
- 커밋 ID GitLab, 풀 리퀘스트 ID, 브랜치 이름, 태그 이름 또는 참조 및 커밋 ID의 경우 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.
- Bitbucket의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 분기 이름 또는 태그 이름입니다. 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. 지정되지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
- Amazon S3의 경우 사용할 빌드 입력 ZIP 파일을 나타내는 객체의 버전 ID입니다.

빌드 수준에서 `sourceVersion`이 지정되어 있으면 (프로젝트 수준에서) 그 버전이 이 `sourceVersion`보다 우선합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

artifacts

필수 사항입니다. 이 빌드 프로젝트의 출력 아티팩트 설정에 대한 정보가 들어 있는 ProjectArtifacts객체입니다. `artifacts` 객체를 추가한 후에는 를 사용해 아티팩트를 최대 12개까지 더 추가할 수 있습니다. 이러한 설정에는 다음이 포함됩니다.

artifacts/type

필수 사항입니다. 빌드 출력 결과물의 유형입니다. 유효한 값은 다음과 같습니다.

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

artifacts/location

`S3` 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

사전 요구 사항에서 생성했거나 식별한 출력 버킷의 이름입니다.

artifacts/path

S3 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

ZIP 파일 또는 폴더를 배치할 출력 버킷의 경로입니다. 값을 지정하지 않은 경우path, and를 CodeBuild 사용하여 빌드 출력 ZIP 파일 또는 폴더의 경로와 이름을 결정합니다 namespaceType (지정된 경우). name 예를 들어 path에 대해 MyPath, name에 대해 MyArtifact.zip을 를 지정하면 경로와 이름은 MyPath/MyArtifact.zip이 됩니다.

artifacts/namespaceType

S3 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

빌드 출력 ZIP 파일 또는 폴더의 네임스페이스입니다. 유효한 값에는 BUILD_ID 및 NONE(이)가 있습니다. 빌드 출력 ZIP 파일 또는 폴더의 경로에 빌드 ID를 삽입하려면 BUILD_ID를 사용하고 그렇지 않은 경우 NONE을 사용합니다. 값을 지정하지 않을 경우 빌드 출력 ZIP 파일 또는 폴더의 경로와 이름을 path (지정된 경우) 와 CodeBuild 사용하여 결정합니다. namespaceType name 예를 들어 path에 대해 MyPath, namespaceType에 대해 BUILD_ID, name에 대해 MyArtifact.zip을 지정하면 경로와 이름은 MyPath/*build-ID*/MyArtifact.zip이 됩니다.

artifacts/name

S3 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

location 안에 있는 빌드 출력 ZIP 파일 또는 폴더의 경로 및 이름입니다. 예를 들어 path에 대해 MyPath, name에 대해 MyArtifact.zip을 를 지정하면 경로와 이름은 MyPath/MyArtifact.zip이 됩니다.

아티팩트/ overrideArtifactName

S3 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

선택 사항입니다. true로 설정할 경우 buildspec 파일의 artifacts 블록에서 지정한 이름이 name을 재정의합니다. 자세한 정보는 에 대한 빌드 사양 참조 CodeBuild을 참조하세요.

artifacts/packaging

S3 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

선택 사항입니다. 아티팩트를 패키징하는 방법을 지정합니다. 허용되는 값:

NONE

빌드 아티팩트가 포함된 폴더를 생성합니다. 이것이 기본값입니다.

ZIP

빌드 아티팩트가 포함된 ZIP 파일을 생성합니다.

secondaryArtifacts

선택 사항입니다. 빌드 프로젝트의 보조 아티팩트 설정에 대한 정보가 포함된 ProjectArtifacts객체 배열입니다. 보조 아티팩트는 최대 12개까지 추가할 수 있습니다. `secondaryArtifacts`는 객체에서 사용하는 것과 동일한 수의 설정을 사용합니다.

cache

필수 사항입니다. 이 빌드 프로젝트의 캐시 설정에 대한 정보가 들어 있는 ProjectCache객체입니다. 자세한 정보는 빌드 캐싱을 참조하세요.

환경

필수 사항입니다. 이 프로젝트의 빌드 환경 설정에 대한 정보가 들어 있는 ProjectEnvironment객체입니다. 이러한 설정은 다음과 같습니다.

environment/type

필수 사항입니다. 빌드 환경의 유형입니다. 자세한 내용은 CodeBuild API 레퍼런스 입력을 참조하십시오.

environment/image

필수 사항입니다. 이 빌드 환경에서 사용하는 도커 이미지 식별자입니다. 일반적으로 이 식별자는 *`image-name`*:*`tag`*로 표현됩니다. 예를 들어 Docker 이미지를 관리하는 데 CodeBuild 사용하는 Docker 저장소에서는 다음과 같을 수 있습니다. `aws/codebuild/standard:5.0` Docker Hub에서는 `maven:3.3.9-jdk-8`입니다. Amazon ECR에서는 *`account-id`*`.dkr.ecr.`*`region-id`*`.amazonaws.com/`*`your-Amazon-ECR-repo-name`*`:`*`tag`*입니다. 자세한 정보는 Docker 이미지 제공: CodeBuild을 참조하세요.

environment/computeType

필수 사항입니다. 이 빌드 환경에서 사용하는 컴퓨팅 리소스를 지정합니다. 자세한 내용은 API 레퍼런스의 CodeBuild ComputeType을 참조하십시오.

environment/certificate

선택 사항입니다. PEM 인코딩된 인증서를 포함하는 Amazon S3 버킷, 경로 접두사 및 객체 키의 ARN입니다. 객체 키는 단지 .pem 파일일 수도 있고 PEM 인코딩된 인증서를 포함하는 .zip 파일일

수도 있습니다. 예를 들어 Amazon S3 버킷 이름이 *<my-bucket>*이고 경로 접두사는 *<cert>*이고 객체 키 이름이 *<certificate.pem>*인 경우 certificate에 허용되는 형식은 *<my-bucket/cert/certificate.pem>* 또는 arn:aws:s3:::*<my-bucket/cert/certificate.pem>*입니다.

environment/environmentVariables

선택 사항입니다. 이 빌드 환경에 지정하려는 환경 변수를 포함하는 EnvironmentVariable객체 배열입니다. 각 환경 변수는 name, value, type의 name, value, type을 포함하는 객체로 표현됩니다.

콘솔 및 AWS CLI 사용자는 모든 환경 변수를 볼 수 있습니다. 환경 변수의 가시성에 대한 문제가 없으면 name 및 value를 설정하고 type을 PLAINTEXT로 설정합니다.

AWS 액세스 키 ID, AWS 보안 액세스 키 또는 암호와 같은 민감한 값이 있는 환경 변수를 Amazon EC2 Systems Manager 파라미터 스토어 AWS Secrets Manager또는 의 파라미터로 저장하는 것이 좋습니다. 저장된 파라미터의 name 경우 CodeBuild 참조할 식별자를 설정하십시오.

Amazon EC2 Systems Manager Parameter Store를 사용하는 경우, value에 대해 파라미터 이름을 Parameter Store에 저장된 것으로 설정합니다. type를 PARAMETER_STORE으로 설정합니다. 이름이 /CodeBuild/dockerLoginPassword인 파라미터를 예제로 사용하여 name을 LOGIN_PASSWORD로 설정합니다. value을 /CodeBuild/dockerLoginPassword으로 설정합니다. type를 PARAMETER_STORE로 설정합니다.

> **Important**
>
> Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 /CodeBuild/로 시작하는 파라미터 이름(예: /CodeBuild/dockerLoginPassword)으로 파라미터를 저장하는 것이 좋습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택하고 대화 상자에 표시되는 지시에 따릅니다. (이 대화 상자에서 KMS 키의 경우 계정 내 키의 AWS KMS ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는 이 키를 사용하여 저장 중에 파라미터 값을 암호화하고 검색 중에 이를 복호화합니다.) 콘솔을 사용하여 파라미터를 생성하는 경우, CodeBuild 콘솔은 파라미터가 저장되는 /CodeBuild/ 대로 파라미터 이름을 시작합니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
> 빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 ssm:GetParameters 작업을 허용해야 합니다. 이전에 새 서비스 역할을 선택한 경우 빌드 프로젝트의 기본 서비스 역할에 이

작업을 CodeBuild 포함하세요. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
빌드 프로젝트가 /CodeBuild/로 시작되지 않는 파라미터 이름으로 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 새 서비스 역할을 선택하면 /CodeBuild/로 시작하지 않는 파라미터 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 /CodeBuild/로 시작하는 파라미터 이름에만 액세스할 수 있기 때문입니다.
새 서비스 역할을 선택하면 Amazon EC2 Systems Manager Parameter Store에 있는 /CodeBuild/ 네임스페이스의 모든 파라미터를 해독할 권한이 서비스 역할에 포함됩니다.
사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 my_value인 MY_VAR이라는 환경 변수가 이미 포함되어 있는데, 사용자가 MY_VAR 환경 변수의 값을 other_value로 설정하면, my_value가 other_value로 바뀝니다. 마찬가지로, 도커 이미지에 값이 /usr/local/sbin:/usr/local/bin인 PATH라는 환경 변수가 이미 포함되어 있는데, 사용자가 PATH 환경 변수의 값을 $PATH:/usr/share/ant/bin으로 설정하면, /usr/local/sbin:/usr/local/bin이 $PATH:/usr/share/ant/bin 리터럴 값으로 바뀝니다.
CODEBUILD_로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.
- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

Secrets Manager를 value 사용하는 경우 파라미터 이름을 Secrets Manager에 저장된 항목으로 설정합니다. type를 SECRETS_MANAGER으로 설정합니다. 이름이 /CodeBuild/dockerLoginPassword인 보안 암호를 예제로 사용하여 name을 LOGIN_PASSWORD로 설정합니다. value을 /CodeBuild/dockerLoginPassword으로 설정합니다. type를 SECRETS_MANAGER로 설정합니다.

Important

Secrets Manager를 사용하는 경우 이름이 /CodeBuild/로 시작하는 암호를 저장하는 것이 좋습니다(예: /CodeBuild/dockerLoginPassword). 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

빌드 프로젝트가 Secrets Manager에 저장된 암호를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `secretsmanager:GetSecretValue` 작업을 허용해야 합니다. 이전에 새 서비스 역할을 선택한 경우 빌드 프로젝트의 기본 서비스 역할에 이 작업을 CodeBuild 포함하세요. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 보안 암호 이름으로 Secrets Manager에 저장된 암호를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 보안 암호 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 암호 이름에만 액세스할 수 있기 때문입니다.
새 서비스 역할을 선택하면 에 있는 `/CodeBuild/` 네임스페이스의 모든 암호를 해독할 권한이 서비스 역할에 포함됩니다.

environment/registryCredential

선택 사항입니다. 프라이빗 Docker 레지스트리에 대한 액세스를 제공하는 자격 증명을 지정하는 RegistryCredential객체입니다.

environment/registryCredential/credential

AWS Managed Services를 사용하여 생성한 보안 인증의 ARN 또는 이름을 지정합니다. 현재 리전에 있는 자격 증명의 이름만 사용할 수 있습니다.

environment/registryCredential/credentialProvider

유일한 유효 값은 `SECRETS_MANAGER`입니다.

이를 설정할 경우 다음과 같이 해야 합니다.

- `imagePullCredentials`를 `SERVICE_ROLE`로 설정해야 합니다.
- 이 이미지는 큐레이트된 이미지 또는 Amazon ECR 이미지일 수 없습니다.

환경/ 유형 imagePullCredentials

선택 사항입니다. 빌드에서 이미지를 가져오는 데 CodeBuild 사용하는 자격 증명의 유형입니다. 두 가지 값을 사용할 수 있습니다.

CODEBUILD

`CODEBUILD`자체 자격 증명을 CodeBuild 사용하도록 지정합니다. Amazon ECR 리포지토리 정책을 편집하여 CodeBuild 서비스 보안 주체를 신뢰해야 합니다.

SERVICE_ROLE

빌드 프로젝트의 서비스 역할을 CodeBuild 사용하도록 지정합니다.

교차 계정 또는 프라이빗 레지스트리 이미지를 사용할 경우 SERVICE_ROLE 자격 증명을 사용해야 합니다. CodeBuild 큐레이션된 이미지를 사용할 때는 CODEBUILD 자격 증명을 사용해야 합니다.

environment/privilegedMode

이 빌드 프로젝트를 사용하여 도커 이미지를 빌드하려는 경우에만 true로 설정합니다. 그렇지 않으면 Docker 데몬과 상호 작용을 시도하는 모든 연결된 빌드가 실패합니다. 또한 빌드가 상호 작용할 수 있도록 Docker 데몬을 시작해야 합니다. 이를 수행하는 한 가지 방법은 다음 빌드 명령을 실행하여 buildspec 파일의 install 단계에서 Docker 데몬을 초기화하는 것입니다. Docker 지원에서 제공하는 빌드 환경 이미지를 지정한 경우에는 이 명령을 실행하지 마세요. CodeBuild

> **Note**
>
> 기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.

```
- 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

필수 사항입니다. 서비스 역할의 ARN은 사용자를 대신하여 서비스와 상호 작용하는 데 CodeBuild 사용합니다 (예:). arn:aws:iam::*account-id*:role/*role-name*

timeoutInMinutes

선택 사항입니다. 5분에서 480분 (8시간) 사이이며, 이 시간이 지나면 빌드가 완료되지 않으면 빌드가 CodeBuild 중지됩니다. 지정하지 않을 경우 기본값인 60을 사용합니다. 타임아웃으로 인해 빌드가 CodeBuild 중지되었는지 여부와 중단된 시기를 확인하려면 명령을 실행하세요. batch-get-builds 빌드가 중지되었는지 확인하려면 출력에서 FAILED의 buildStatus 값을 살펴봅니다. 빌드가 시간 초과된 시간을 확인하려면 출력에서 TIMED_OUT의 phaseStatus와 연결된 endTime 값을 살펴봅니다.

queuedTimeoutIn분

선택 사항입니다. 5분에서 480분 (8시간) 사이이며, 이 시간이 지나면 빌드가 아직 대기 중인 경우 빌드가 CodeBuild 중지됩니다. 지정하지 않을 경우 기본값인 60을 사용합니다.

encryptionKey

선택 사항입니다. 에서 빌드 출력을 암호화하는 데 사용하는 CodeBuild 별칭 또는 ARN입니다. AWS KMS key 별칭을 지정하는 경우 `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* 형식을 사용하고, 별칭이 있는 경우 `alias/`*`key-alias`* 형식을 사용합니다. 지정하지 않으면 Amazon S3의 AWS관리형 KMS 키가 사용됩니다.

tags

선택 사항입니다. 이 빌드 프로젝트와 연결할 태그를 제공하는 Tag 객체 배열입니다. 최대 50개의 태그를 지정할 수 있습니다. CodeBuild 빌드 프로젝트 태그를 지원하는 모든 AWS 서비스에서 이러한 태그를 사용할 수 있습니다. 각 태그는 `key`와 `value`가 있는 객체로 표현됩니다.

vpcConfig

선택 사항입니다. 프로젝트의 VPC 구성에 대한 정보가 들어 있는 VpcConfig객체입니다. 자세한 정보는 Amazon Virtual Private 클라우드와 AWS CodeBuild 함께 사용을 참조하세요.

이러한 속성은 다음과 같습니다.

vpcId

필수 사항입니다. 사용하는 VPC ID입니다. CodeBuild 리전의 모든 VPC ID 목록을 가져오려면 다음 명령을 실행합니다.

```
aws ec2 describe-vpcs --region <region-ID>
```

subnets

필수 사항입니다. 에서 사용하는 리소스를 포함하는 서브넷 ID의 배열. CodeBuild 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region <region-ID>
```

securityGroupIds

필수 사항입니다. 에서 VPC의 리소스에 대한 액세스를 허용하는 CodeBuild 데 사용하는 보안 그룹 ID의 배열입니다. 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --<region-ID>
```

badgeEnabled

선택 사항입니다. 프로젝트에 빌드 배지를 포함할지 여부를 지정합니다. CodeBuild 빌드 배지를 활성화하려면 `true`로 설정하고, 그렇지 않으면 `false`로 설정합니다. 자세한 정보는 다음을 사용하여 배지 샘플을 빌드하십시오. CodeBuild 을 참조하세요.

logsConfig

이 빌드의 로그 위치에 대한 정보가 들어 있는 LogsConfig객체입니다.

LogsConfig/ cloudWatchLogs

로그를 Log로 푸시하는 방법에 대한 정보가 들어 있는 CloudWatchLogsConfig객체입니다. CloudWatch

logsConfig/s3Logs

Amazon S3로 로그를 푸시하는 방법에 대한 정보가 들어 있는 S3 LogsConfig 객체입니다.

fileSystemLocations

선택 사항입니다. Amazon EFS 구성에 대한 정보가 들어 있는 ProjectFileSystemsLocation객체 배열입니다.

buildBatchConfig

선택 사항입니다. `buildBatchConfig`객체는 프로젝트의 배치 빌드 구성 정보를 포함하는 ProjectBuildBatchConfig구조입니다.

buildBatchConfig/서비스 역할

배치 빌드 프로젝트에 대한 서비스 역할 ARN입니다.

buildBatchConfig/아티팩트 결합

배치 빌드의 빌드 아티팩트를 단일 아티팩트 위치로 결합할지 여부를 지정하는 부울 값입니다.

buildBatchConfig/제한 사항/ maximumBuildsAllowed

허용되는 최대 빌드 수입니다.

buildBatchConfig/제한 사항/ computeTypesAllowed

배치 빌드에 허용되는 컴퓨팅 유형을 지정하는 문자열 배열입니다. 이러한 값은 빌드 환경 컴퓨팅 유형를 참조하세요.

buildBatchConfig/timeoutInMinutes

배치 빌드를 완료해야 하는 최대 시간(분)입니다.

buildBatchConfig/batchReportMode

배치 빌드를 위해 빌드 상태 보고서를 소스 제공자에게 보내는 방법을 지정합니다. 유효한 값으로는 다음이 포함됩니다.

`REPORT_AGGREGATED_BATCH`

(기본값) 모든 빌드 상태를 단일 상태 보고서로 집계합니다.

`REPORT_INDIVIDUAL_BUILDS`

각 개별 빌드에 대해 별도의 상태 보고서를 보냅니다.

concurrentBuildLimit

이 프로젝트에 허용된 최대 동시 실행 빌드 수입니다.

현재 빌드 수가 이 한도 이하인 경우에만 새 빌드가 시작됩니다. 현재 빌드 수가 이 한도에 도달하면 새 빌드가 제한되고 실행되지 않습니다.

프로젝트 생성

프로젝트를 생성하려면 JSON 파일을 전달하여 **`create-project`** 명령을 다시 실행합니다.

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

성공하면 Project 객체의 JSON 표현이 콘솔 출력에 나타납니다. 이 데이터의 예는 CreateProject 응답 구문을 참조하십시오.

빌드 프로젝트 이름을 제외한 모든 빌드 프로젝트 설정은 나중에 변경할 수 있습니다. 자세한 정보는 빌드 프로젝트 설정 변경(AWS CLI)을 참조하세요.

빌드 실행을 시작하려면 빌드 실행(AWS CLI) 단원을 참조하십시오.

소스 코드가 리포지토리에 저장되어 있고 코드 변경 사항이 GitHub 리포지토리에 푸시될 때마다 소스 코드를 다시 CodeBuild 빌드하려는 경우 을 참조하십시오빌드 실행 자동 시작(AWS CLI).

빌드 프로젝트 생성(AWS SDK)

AWS CodeBuild를 AWS SDK와 함께 사용하는 방법에 대한 자세한 정보는 AWS SDK 및 도구 참조 단원을 참조하십시오.

빌드 프로젝트 생성(AWS CloudFormation)

AWS CloudFormation에서 AWS CodeBuild를 사용하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 CodeBuild용 AWS CloudFormation 템플릿을 참조하세요.

알림 규칙 생성

알림 규칙을 사용하여 빌드 성공 및 실패와 같은 중요한 변경 사항이 발생할 경우 사용자에게 알릴 수 있습니다. 알림 규칙은 알림을 보내는 데 사용되는 이벤트와 Amazon SNS 주제를 모두 지정합니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

콘솔 또는 AWS CLI를 사용하여 AWS CodeBuild에 대한 알림 규칙을 생성할 수 있습니다.

알림 규칙을 생성하려면(콘솔)

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드, 빌드 프로젝트를 차례로 선택한 다음 알림을 추가할 빌드 프로젝트를 선택합니다.
3. 빌드 프로젝트 페이지에서 알림을 선택하고 Create notification rule(알림 규칙 생성)을 선택합니다. 빌드 프로젝트의 설정 페이지로 이동하여 Create notification rule(알림 규칙 생성)를 선택할 수도 있습니다.
4. 알림 이름에 규칙에 대한 이름을 입력합니다.
5. 알림에 포함된 Amazon EventBridge에 제공된 정보만 원하는 경우 세부 정보 유형에서 기본을 선택합니다. Amazon EventBridge에 제공된 정보와 CodeBuild 또는 알림 관리자에서 제공할 수 있는 정보를 포함하려는 경우 전체를 선택합니다.

   자세한 내용은 알림 내용 및 보안 이해를 참조하십시오.

6. Events that trigger notifications(알림을 트리거하는 이벤트)에서 알림을 보내고자 하는 이벤트를 선택합니다. 자세한 내용은 빌드 프로젝트의 알림 규칙 이벤트를 참조하십시오.
7. 대상에서 다음 중 하나를 수행합니다.
   - 알림과 함께 사용할 리소스를 이미 구성한 경우 Choose target type(대상 유형 선택)에서 AWS Chatbot(Slack) 또는 SNS 주제를 선택합니다. 대상 선택에서 클라이언트의 이름(AWS Chatbot에 구성된 Slack 클라이언트의 경우) 또는 Amazon SNS 주제의 Amazon 리소스 이름(ARN)(알림에 필요한 정책으로 이미 구성된 Amazon SNS 주제의 경우)을 선택합니다.
   - 알림과 함께 사용할 리소스를 구성하지 않은 경우 Create target(대상 생성)을 선택한 다음 SNS 주제를 선택합니다. codestar-notifications- 뒤에 주제 이름을 입력한 다음 생성을 선택합니다.

   > **Note**
   >
   > - 알림 규칙을 만드는 과정에서 Amazon SNS 주제를 생성하면 알림 기능이 주제에 이벤트를 게시할 수 있도록 허용하는 정책이 적용됩니다. 알림 규칙에 대해 생성된 주제를 사용하면 이 리소스에 대한 알림을 받기를 원하는 사용자만 구독할 수 있습니다.
   > - 알림 규칙 생성 중에는 AWS Chatbot 클라이언트를 생성할 수 없습니다. AWS Chatbot(Slack)을 선택하면 AWS Chatbot에서 클라이언트를 구성하도록 지시하는 버튼이 표시됩니다. 이 옵션을 선택하면 AWS Chatbot 콘솔이 열립니다. 자세한 내용은 알림과 AWS Chatbot 간의 통합 구성을 참조하십시오.
   > - 기존 Amazon SNS 주제를 대상으로 사용하려면 해당 주제에 대해 존재할 수 있는 다른 정책 외에 AWS CodeStar Notifications에 필요한 정책을 추가해야 합니다. 자세한 내용은 알림에 대한 Amazon SNS 주제 구성과 알림 내용 및 보안 이해를 참조하십시오.
8. 규칙 생성을 완료하려면 제출을 선택합니다.
9. 사용자가 알림을 받으려면 먼저 규칙에 대한 Amazon SNS 주제를 사용자가 구독하도록 해야 합니다. 자세한 내용은 대상인 Amazon SNS 주제에 사용자 구독을 참조하세요. Amazon Chime 채팅룸에 알림을 보내도록 알림과 AWS Chatbot 간의 통합을 설정할 수도 있습니다. 자세한 내용은 알림과 AWS Chatbot 간의 통합 구성을 참조하십시오.

알림 규칙을 생성하려면(AWS CLI)

1. 터미널 또는 명령 프롬프트에서 create-notification rule 명령을 실행하여 JSON 스켈레톤을 생성합니다.

```
aws codestarnotifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

원하는 대로 파일 이름을 지정할 수 있습니다. 이 예에서는 *rule.json*으로 파일 이름을 지정합니다.

2. 일반 텍스트 편집기에서 JSON 파일을 열고 규칙에 대해 원하는 리소스, 이벤트 유형 및 대상을 포함하도록 편집합니다. 다음 예는 ID가 *123456789012*인 AWS 계정의 *MyBuildProject*라는 리포지토리에 대한 **MyNotificationRule**이라는 알림 규칙을 보여 줍니다. 빌드가 성공적일 때 *codestar-notifications-MyNotificationTopic*이라는 Amazon SNS 주제에 전체 세부 정보 유형과 함께 알림이 전송됩니다.

```
{
    "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"
}
```

파일을 저장합니다.

3. 터미널 또는 명령줄에서 create-notification-rule 명령을 다시 실행하여 조금 전 편집한 파일을 사용해 알림 규칙을 생성합니다.

```
aws codestarnotifications create-notification-rule --cli-input-json
 file://rule.json
```

4. 성공한 경우 명령에서 다음과 유사한 알림 규칙의 ARN을 반환합니다.

```
{
```

```
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

AWS CodeBuild에서 빌드 프로젝트 이름 목록 보기

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서 빌드 프로젝트 목록을 볼 수 있습니다.

주제

- 빌드 프로젝트 이름 목록 보기(콘솔)
- 빌드 프로젝트 이름 목록 보기(AWS CLI)
- 빌드 프로젝트 이름 목록 보기(AWS SDK)

빌드 프로젝트 이름 목록 보기(콘솔)

콘솔에서 AWS 리전의 빌드 프로젝트 목록을 볼 수 있습니다. 정보에는 이름, 소스 공급자, 리포지토리, 최신 빌드 상태 및 설명(있는 경우)이 포함됩니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 프로젝트 10개만 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

빌드 프로젝트 이름 목록 보기(AWS CLI)

list-projects 명령을 실행합니다.

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-
token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *sort-by*: 빌드 프로젝트 이름을 나열하는 데 사용할 기준을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값으로는 다음이 포함됩니다.
  - CREATED_TIME: 각 빌드 프로젝트가 생성된 시기를 기준으로 빌드 프로젝트 이름을 나열합니다.
  - LAST_MODIFIED_TIME: 각 빌드 프로젝트에 대한 정보가 마지막으로 변경된 시기를 기준으로 빌드 프로젝트 이름을 나열합니다.
  - NAME: 각 빌드 프로젝트의 이름을 기준으로 빌드 프로젝트 이름을 나열합니다.
- *sort-order*: *sort-by*에 따라 빌드 프로젝트를 나열하는 순서를 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING이 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "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"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-token
 Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "projects": [
```

```
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
  ]
}
```

빌드 프로젝트 이름 목록 보기(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

AWS CodeBuild에서 빌드 프로젝트 세부 정보 보기

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서 빌드 프로젝트의 세부 정보를 볼 수 있습니다.

주제

- 빌드 프로젝트 세부 정보 보기(콘솔)
- 빌드 프로젝트 세부 정보 보기(AWS CLI)
- 빌드 프로젝트 세부 정보 보기(AWS SDK)

빌드 프로젝트 세부 정보 보기(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다.

   > **Note**
   >
   > 기본적으로 가장 최근의 빌드 프로젝트 10개만 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

3. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트에 대한 링크를 선택합니다.
4. 프로젝트 빌드: ***project-name*** 페이지에서 빌드 세부 정보를 선택합니다.

빌드 프로젝트 세부 정보 보기(AWS CLI)

batch-get-projects 명령을 실행합니다.

```
aws codebuild batch-get-projects --names names
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *names*: 세부 정보를 볼 수 있는 하나 이상의 빌드 프로젝트 이름을 나타내는 데 사용되는 필수 문자열입니다. 둘 이상의 빌드 프로젝트를 지정하려면 각 빌드 프로젝트 이름을 공백을 사용하여 구분해야 합니다. 최대 100개의 빌드 프로젝트 이름을 지정할 수 있습니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-project2
 my-other-demo-project
```

다음과 비슷한 결과가 출력에 나타납니다. 줄임표(...)는 간결성을 위해 생략된 데이터를 나타내는 데 사용됩니다.

```
{
  "projectsNotFound": [
    "my-other-demo-project"
  ],
  "projects": [
    {
      ...
      "name": codebuild-demo-project,
      ...
    },
    {
      ...
      "name": codebuild-demo-project2",
      ...
    }
  ]
}
```

위 출력에서 projectsNotFound 배열에는 지정되었지만 찾을 수 없는 모든 빌드 프로젝트 이름이 나열됩니다. projects 배열에는 정보가 발견된 각 빌드 프로젝트의 세부 정보가 나열됩니다. 간결성

을 위해 이전 출력에서 빌드 프로젝트 세부 정보가 생략되었습니다. 자세한 내용은 빌드 프로젝트 생성 (AWS CLI)의 출력을 참조하세요.

batch-get-projects 명령은 특정 속성 값에 대한 필터링을 지원하지 않지만 프로젝트의 속성을 열거하는 스크립트를 작성할 수 있습니다. 예를 들어, 다음 Linux 쉘 스크립트는 현재 계정의 현재 리전에 있는 프로젝트를 열거하고 각 프로젝트에서 사용되는 이미지를 인쇄합니다.

```
#!/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
```

AWS CLI와 AWS CodeBuild를 함께 사용하는 방법에 대한 자세한 정보는 명령줄 참조 단원을 참조하십시오.

빌드 프로젝트 세부 정보 보기(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

AWS CodeBuild의 빌드 캐싱

프로젝트가 빌드될 때 캐시를 사용하여 시간을 절약할 수 있습니다. 캐시는 빌드 환경에서 재사용할 수 있는 정보를 저장하여 여러 빌드에 사용할 수 있습니다. 빌드 프로젝트는 Amazon S3 또는 로컬의 두 캐싱 유형 중 하나를 사용할 수 있습니다. 로컬 캐시를 사용하는 경우 다음 세 캐시 모드에서 하나 이상을 선택해야 합니다. 소스 캐시, Docker 계층 캐시 및 사용자 지정 캐시.

> **Note**
>
> Docker 계층 캐시 모드는 Linux 환경에서만 사용할 수 있습니다. 이 모드를 선택하는 경우 빌드를 권한 모드에서 실행해야 합니다. CodeBuild 권한 모드가 부여된 프로젝트는 컨테이너에 모든 기기에 대한 액세스 권한을 부여합니다. 자세한 내용은 Docker 문서 웹 사이트의 런타임 권한 및 Linux 기능을 참조하십시오.

주제

- Amazon S3 캐싱
- 로컬 캐싱

Amazon S3 캐싱

Amazon S3 캐싱은 여러 빌드 호스트에 사용 가능한 캐시를 Amazon S3 버킷에 저장합니다. 이는 다운로드보다 빌드가 더 비용이 많이 드는 소형-중형 빌드 아티팩트에 적합한 옵션입니다. 대용량 빌드 아티팩트의 경우에는 네트워크를 통해 전송하는 데 오랜 시간이 걸릴 수 있고 따라서 빌드 성능에 영향을 미칠 수 있으므로 최선의 옵션은 아닙니다. Docker 계층을 사용하는 경우에는 최선의 옵션이 아닙니다.

로컬 캐싱

로컬 캐싱은 해당 빌드 호스트에만 사용할 수 있는 캐시를 빌드에 로컬로 저장합니다. 빌드 호스트에서 즉각적으로 캐시를 사용할 수 있으므로 중형-대형 빌드 아티팩트에 적합한 옵션입니다. 빌드가 드문 경우에는 최선의 옵션이 아닙니다. 이는 빌드 성능이 네트워크 전송 시간의 영향을 받지 않는다는 의미입니다.

로컬 캐싱을 선택할 경우 다음 캐시 모드 중 하나 이상을 선택해야 합니다.

- 소스 캐시 모드는 기본 및 보조 소스를 위해 Git 메타데이터를 캐싱합니다. 캐시가 생성되면 이후의 빌드는 커밋 사이의 변경 사항만 끌어옵니다. 이 모드는 클린 작업 디렉터리를 사용하고 소스가 대규모 Git 리포지토리인 프로젝트에 적합한 선택입니다. 프로젝트에서 Git 리포지토리 (AWS CodeCommit, GitHub, GitHub 엔터프라이즈 서버 또는 Bitbucket) 를 사용하지 않는 경우 이 옵션은 무시됩니다.
- Docker 계층 캐시 모드는 기존 Docker 계층을 캐싱합니다. 이 모드는 대용량 도커 이미지를 빌드하거나 끌어오는 프로젝트에 적합한 선택입니다. 네트워크에서 대용량 도커 이미지를 끌어올 때 발생하는 성능 문제를 방지할 수 있습니다.

  > **Note**
  >
  > - Docker 계층 캐시는 Linux 환경에서만 사용할 수 있습니다.
  > - 프로젝트가 필요한 Docker 권한을 가지도록 `privileged` 플래그를 설정해야 합니다.
  >
  >   기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.
  > - Docker 계층 캐시를 사용하기 전에 보안에 미치는 영향을 고려해야 합니다.

- 사용자 지정 캐시 모드는 buildspec 파일에 지정한 디렉터리를 캐싱합니다. 이 모드는 다른 두 로컬 캐시 모드에 적합하지 않은 빌드 시나리오에 적합한 선택입니다. 사용자 지정 캐시를 사용할 경우
  - 캐싱을 위해 디렉터리만 지정할 수 있습니다. 개별 파일은 지정할 수 없습니다.
  - Symlink를 사용하여 캐싱된 디렉터리를 참조합니다.
  - 캐싱된 디렉터리는 프로젝트 소스를 다운로드하기 전에 빌드에 연결됩니다. 캐시된 항목은 이름이 같은 경우 소스 항목을 재정의합니다. 디렉터리는 buildspec 파일에서 경로를 사용하여 지정합니다. 자세한 설명은 buildspec 구문 섹션을 참조하세요.

- 소스와 캐시에서 동일한 디렉터리 이름은 사용하지 마십시오. 로컬로 캐시된 디렉터리는 소스 리포지토리에서 이름이 같은 디렉터리의 내용을 재정의하거나 삭제할 수 있습니다.

> **Note**
>
> `LINUX_GPU_CONTAINER` 환경 유형 및 `BUILD_GENERAL1_2XLARGE` 컴퓨팅 유형에서는 로컬 캐싱이 지원되지 않습니다. 자세한 설명은 빌드 환경 컴퓨팅 모드 및 유형 섹션을 참조하세요.

> **Note**
>
> VPC와 함께 CodeBuild 작동하도록 구성할 때는 로컬 캐싱이 지원되지 않습니다. 에서 VPC를 사용하는 방법에 대한 자세한 내용은 을 CodeBuild 참조하십시오. Amazon Virtual Private 클라우드와 AWS CodeBuild 함께 사용

주제

AWS CLI, 콘솔, SDK 또는 AWS CloudFormation을 사용하여 로컬 캐시를 지정할 수 있습니다.

로컬 캐싱 지정(CLI)

AWS CLI에서 `--cache` 파라미터를 사용하여 세 가지 로컬 캐시 유형을 각각 지정할 수 있습니다.

- 소스 캐시를 지정하려면

```
--cache type=LOCAL,mode=[LOCAL_SOURCE_CACHE]
```

- Docker 계층 캐시를 지정하려면

```
--cache type=LOCAL,mode=[LOCAL_DOCKER_LAYER_CACHE]
```

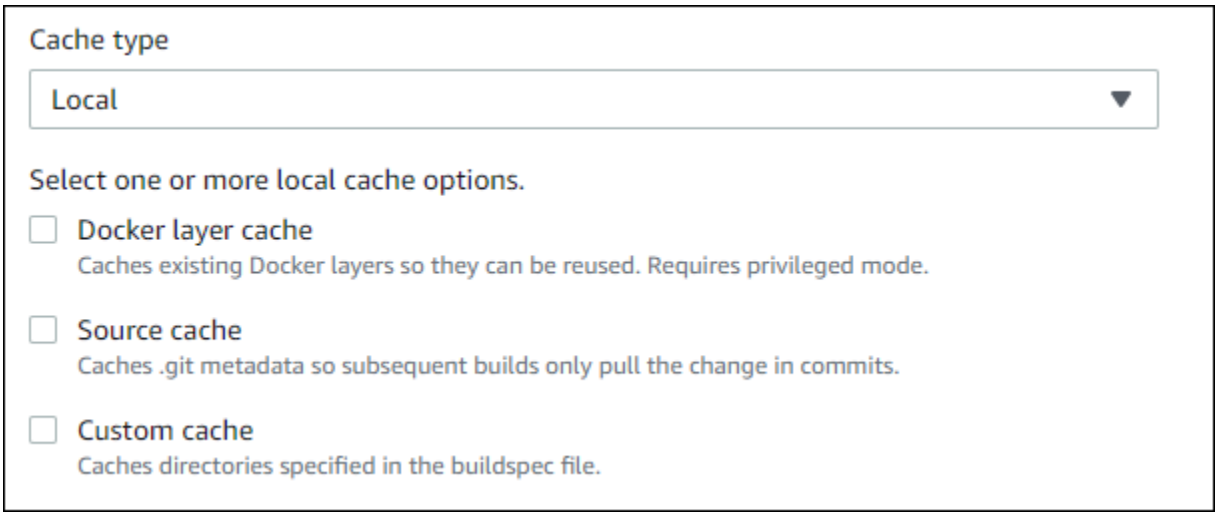
- 사용자 지정 캐시를 지정하려면

```
--cache type=LOCAL,mode=[LOCAL_CUSTOM_CACHE]
```

자세한 설명은 빌드 프로젝트 생성(AWS CLI) 섹션을 참조하세요.

로컬 캐싱 지정(콘솔)

콘솔의 결과물 섹션에서 캐시를 지정합니다. 캐시 유형은 Amazon S3 또는 로컬을 선택합니다. 로컬을 선택한 경우 세 로컬 캐시 옵션 중 하나 이상을 선택합니다.



자세한 설명은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.

로컬 캐싱 지정(AWS CloudFormation)

AWS CloudFormation을 사용하여 로컬 캐시를 지정할 경우 Cache 속성에서 Type은 LOCAL을 지정합니다. 다음 샘플 YAML 형식 AWS CloudFormation 코드는 세 가지 로컬 캐시 유형을 모두 지정합니다. 각 유형을 임의로 조합하여 지정할 수 있습니다. Docker 계층 캐시를 사용하는 경우 Environment에서 PrivilegedMode를 true로 설정하고 Type을 LINUX_CONTAINER로 설정해야 합니다.

```
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**
>
> 기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.

자세한 내용은 빌드 프로젝트 생성(AWS CloudFormation)을(를) 참조하세요.

에서 트리거 생성 AWS CodeBuild

주제

- AWS CodeBuild 트리거 생성
- AWS CodeBuild 트리거 편집

AWS CodeBuild 트리거 생성

AWS CodeBuild 트리거 생성(콘솔)

프로젝트에 트리거를 생성하여 1시간, 1일 또는 일주일에 한 번씩 빌드를 예약할 수 있습니다. Amazon CloudWatch cron 표현식이 포함된 사용자 지정 규칙을 사용하여 트리거를 생성할 수도 있습니다. 예를 들어 cron 표현식을 사용하여 매주 평일 특정 시간에 빌드를 예약할 수 있습니다.

> **Note**
>
> 빌드 트리거, Amazon EventBridge 이벤트 또는 AWS Step Functions 작업에서 배치 빌드를 시작하는 것은 불가능합니다.

트리거를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 트리거를 추가하려는 빌드 프로젝트의 링크를 선택한 후 빌드 트리거 탭을 선택합니다.

   > **Note**
   >
   > 기본적으로 가장 최근의 빌드 프로젝트 100개가 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

4. 트리거 생성을 선택합니다.
5. 트리거 이름에 이름을 입력합니다.
6. 빈도 드롭다운 목록에서 트리거의 빈도를 선택합니다. cron 표현식으로 빈도를 만들고 싶다면, 사용자 지정을 선택하십시오.
7. 트리거의 빈도에 대한 파라미터를 지정합니다. 텍스트 상자에 선택 항목의 처음 몇 자를 입력하면 드롭다운 메뉴 항목을 필터링할 수 있습니다.

   > **Note**
   >
   > 시작 시간과 분은 0 기준입니다. 시작 분은 0에서 59 사이의 숫자입니다. 시작 시간은 0에서 23 사이의 숫자입니다. 예를 들어, 매일 오후 12시 15분에 시작하는 일일 트리거는 시작 시간이 12이고 시작 분이 15입니다. 매일 자정에 시작하는 일일 트리거는 시작 시간이 0이

고 시작 분이 0입니다. 매일 오후 11시 59분에 시작하는 일일 트리거는 시작 시간이 23이고 시작 분이 59입니다.

빈도	필요한 파라미터	Details
시간당	시작 분	Start minute(시작 분) 드롭다운 메뉴를 사용합니다.
일별	시작 분 시작 시간	Start minute(시작 분) 드롭다운 메뉴를 사용합니다. Start hour(시작 시간) 드롭다운 메뉴를 사용합니다.
주별	시작 분 시작 시간 시작 일	Start minute(시작 분) 드롭다운 메뉴를 사용합니다. Start hour(시작 시간) 드롭다운 메뉴를 사용합니다. Start day(시작 일) 드롭다운 메뉴를 사용합니다.

빈도	필요한 파라미터	Details
사용자 지정(Custom)	cron 표현식	cron 표현식에 cron 표현식을 입력합니다. cron 표현식에는 각각 공백으로 구분되는 필수 필드 6개가 있습니다. 필드는 분, 시간, 일, 월, 요일 및 연도의 시작 값을 지정합니다. 와일드카드를 사용하여 범위, 추가 값 등을 지정할 수 있습니다. 예를 들어 cron 표현식은 매주 평일 오전 9시에 빌드를 **`0 9 ? * MON-FRI *`** 예약합니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서의 Cron 표현식을 참조하십시오.

8. Enable this trigger(이 트리거 사용)를 선택합니다.
9. (선택 사항) 고급 섹션을 확장합니다. 소스 버전에서 소스의 버전을 입력합니다.
   - Amazon S3의 경우 빌드하려는 입력 아티팩트의 버전에 해당하는 버전 ID를 입력합니다. 소스 버전을 비워 두면 최신 버전이 사용됩니다.
   - AWS CodeCommit의 경우, 커밋 ID를 입력하십시오. 소스 버전을 비워 두면 기본 브랜치의 HEAD 커밋 ID가 사용됩니다.
   - GitHub 또는 GitHub 엔터프라이즈의 경우 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 풀 요청 ID, 브랜치 이름 또는 태그 이름을 입력합니다. 풀 요청 ID를 지정하는 경우 `pr/`*`pull-request-ID`* 형식을 사용해야 합니다(예: `pr/25`). 분기 이름을 지정할 경우 분기의 HEAD 커밋 ID가 사용됩니다. [Source version]이 비어 있으면 기본 분기의 HEAD 커밋 ID가 사용됩니다.
   - Bitbucket의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 분기 이름 또는 태그 이름을 입력하십시오. 분기 이름을 지정할 경우 분기의 HEAD 커밋 ID가 사용됩니다. [Source version]이 비어 있으면 기본 분기의 HEAD 커밋 ID가 사용됩니다.
10. (선택 사항) 제한 시간을 5분에서 480분(8시간) 사이로 지정합니다. 이 값은 AWS CodeBuild가 중지되기 전에 빌드를 시도하는 시간을 지정합니다. 시간과 분을 비워 두면 프로젝트에 지정된 기본 제한 시간 값이 사용됩니다.
11. 트리거 생성을 선택합니다.

프로그래밍 방식으로 AWS CodeBuild 트리거 생성

CodeBuild 빌드 트리거에 Amazon EventBridge 규칙을 사용합니다. EventBridge API를 사용하여 프로젝트를 위한 빌드 트리거를 프로그래밍 방식으로 생성할 수 있습니다. CodeBuild 자세한 내용은 Amazon EventBridge API 레퍼런스를 참조하십시오.

AWS CodeBuild 트리거 편집

AWS CodeBuild 트리거 편집 (콘솔)

프로젝트에서 트리거를 편집하여 시간, 일 또는 주에 한 번씩 빌드를 예약할 수 있습니다. Amazon CloudWatch cron 표현식과 함께 사용자 지정 규칙을 사용하도록 트리거를 편집할 수도 있습니다. 예를 들어 cron 표현식을 사용하여 매주 평일 특정 시간에 빌드를 예약할 수 있습니다. 트리거 생성에 대한 자세한 내용은 AWS CodeBuild 트리거 생성 섹션을 참조하세요.

트리거를 편집하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 변경하려는 빌드 프로젝트의 링크를 선택한 다음, 빌드 트리거 탭을 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 프로젝트 100개가 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

4. 변경할 트리거 옆에 있는 라디오 버튼을 선택한 다음, 편집을 선택합니다.
5. 빈도 드롭다운 목록에서 트리거의 빈도를 선택합니다. cron 표현식으로 빈도를 만들고 싶다면, 사용자 지정을 선택하십시오.
6. 트리거의 빈도에 대한 파라미터를 지정합니다. 텍스트 상자에 선택 항목의 처음 몇 자를 입력하면 드롭다운 메뉴 항목을 필터링할 수 있습니다.

> **Note**
>
> 시작 시간과 분은 0 기준입니다. 시작 분은 0에서 59 사이의 숫자입니다. 시작 시간은 0에서 23 사이의 숫자입니다. 예를 들어, 매일 오후 12시 15분에 시작하는 일일 트리거는 시작 시간이 12이고 시작 분이 15입니다. 매일 자정에 시작하는 일일 트리거는 시작 시간이 0이

고 시작 분이 0입니다. 매일 오후 11시 59분에 시작하는 일일 트리거는 시작 시간이 23이고 시작 분이 59입니다.

빈도	필요한 파라미터	Details
시간당	시작 분	Start minute(시작 분) 드롭다운 메뉴를 사용합니다.
일별	시작 분 시작 시간	Start minute(시작 분) 드롭다운 메뉴를 사용합니다. Start hour(시작 시간) 드롭다운 메뉴를 사용합니다.
주별	시작 분 시작 시간 시작 일	Start minute(시작 분) 드롭다운 메뉴를 사용합니다. Start hour(시작 시간) 드롭다운 메뉴를 사용합니다. Start day(시작 일) 드롭다운 메뉴를 사용합니다.

빈도	필요한 파라미터	Details
사용자 지정(Custom)	cron 표현식	cron 표현식에 cron 표현식을 입력합니다. cron 표현식에는 각각 공백으로 구분되는 필수 필드 6개가 있습니다. 필드는 분, 시간, 일, 월, 요일 및 연도의 시작 값을 지정합니다. 와일드카드를 사용하여 범위, 추가 값 등을 지정할 수 있습니다. 예를 들어 cron 표현식은 매주 평일 오전 9시에 빌드를 **`0 9 ? * MON-FRI *`** 예약합니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서의 Cron 표현식을 참조하십시오.

7. Enable this trigger(이 트리거 사용)를 선택합니다.

> **Note**
>
> https://console.aws.amazon.com/cloudwatch/ 의 Amazon CloudWatch 콘솔을 사용하여 소스 버전, 타임아웃 및 에서 AWS CodeBuild 사용할 수 없는 기타 옵션을 편집할 수 있습니다.

프로그래밍 방식으로 AWS CodeBuild 트리거 편집

CodeBuild 빌드 트리거에 Amazon EventBridge 규칙을 사용합니다. EventBridge API를 사용하여 프로젝트의 빌드 트리거를 프로그래밍 방식으로 편집할 수 있습니다. CodeBuild 자세한 내용은 Amazon EventBridge API 레퍼런스를 참조하십시오.

GitLab 연결

연결을 통해 타사 공급자를 를 사용하는 AWS AWS CodeConnections리소스와 연결하는 구성을 승인하고 설정할 수 있습니다. 타사 리포지토리를 빌드 프로젝트의 소스로 연결하려면 연결을 사용합니다.

소스 제공자 GitLab 또는 GitLab 자체 관리형 소스 공급자를 추가하려면 다음 중 하나를 선택할 수 있습니다. CodeBuild

- CodeBuild 콘솔의 빌드 프로젝트 생성 마법사 또는 소스 편집 페이지를 사용하여 GitLab또는 GitLab 자체 관리형 제공자 옵션을 선택합니다. 소스 제공자를 GitLab(콘솔) 에 대한 연결 만들기 추가하려면 을 참조하십시오. 콘솔을 사용하면 연결 리소스를 만들 수 있습니다.
- CLI를 사용하여 연결 리소스를 생성합니다. CLI로 연결 리소스를 GitLab (CLI) 에 대한 연결 생성 생성하려면 을 참조하십시오.

> **Note**
>
> 설정의 개발자 도구 콘솔을 사용하여 연결을 생성할 수도 있습니다. 연결 생성을 참조하세요.

> **Note**
>
> 에서 GitLab 이 연결 설치를 승인하면 서비스에 계정에 액세스하여 데이터를 처리할 수 있는 권한을 부여하게 되며, 애플리케이션을 제거하여 언제든지 권한을 취소할 수 있습니다.

시작하기 전:

- 에 이미 계정을 생성했어야 합니다. GitLab

  > **Note**
  >
  > 연결은 연결을 만들고 권한을 부여하는 데 사용된 계정이 소유한 리포지토리에 대한 액세스 권한만 제공합니다.

  > **Note**
  >
  > 소유자 역할이 있는 저장소에 대한 연결을 만든 다음 다음과 같은 리소스가 있는 저장소와 연결을 사용할 수 CodeBuild 있습니다. GitLab 그룹 내 리포지토리의 경우 그룹 소유자가 아니어도 됩니다.

- 빌드 프로젝트의 소스를 지정하려면 에 이미 저장소를 생성해야 합니다 GitLab.

주제

- GitLab(콘솔) 에 대한 연결 만들기
- GitLab (CLI) 에 대한 연결 생성

GitLab(콘솔) 에 대한 연결 만들기

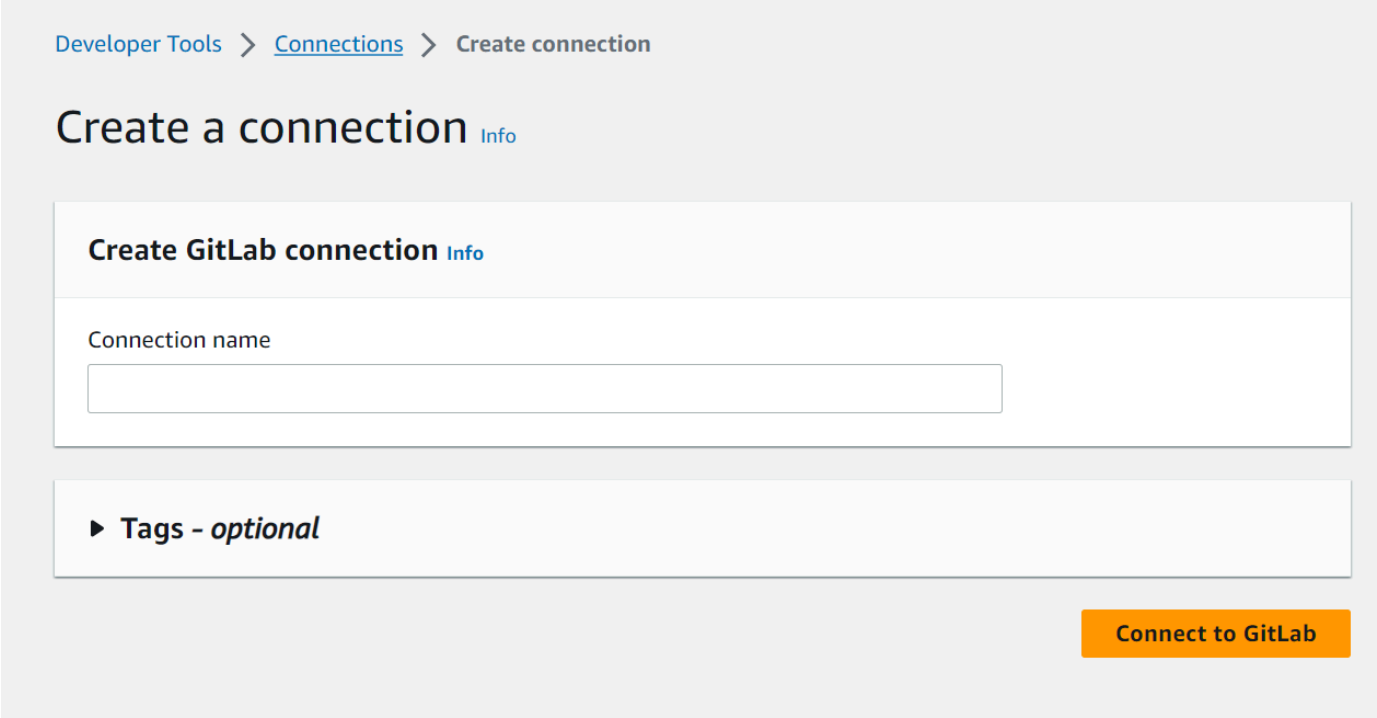
다음 단계를 사용하여 CodeBuild 콘솔을 사용하여 프로젝트 (리포지토리) 에 대한 연결을 추가할 수 GitLab 있습니다.

빌드 프로젝트를 만들거나 편집하려면

1. CodeBuild 콘솔에 로그인합니다.
2. 다음 중 하나를 선택합니다.
   - 빌드 프로젝트 생성을 선택합니다. 단계에 따라 첫 번째 화면을 완성하고 소스 섹션의 소스 제공자에서 선택하세요 GitLab. 빌드 프로젝트 만들기(콘솔)
   - 기존 빌드 프로젝트를 편집하도록 선택합니다. 편집을 선택한 다음 소스를 선택합니다. 소스 편집 페이지의 소스 제공자에서 을 선택합니다 GitLab.
3. 다음 중 하나를 선택합니다.
   - 연결에서 기본 연결을 선택합니다. 기본 연결은 모든 프로젝트에 기본 GitLab 연결을 적용합니다.
   - 연결에서 사용자 지정 연결을 선택합니다. 사용자 지정 연결은 사용자 계정의 기본 설정을 재정의하는 사용자 지정 GitLab 연결을 적용합니다.
4. 다음 중 하나를 수행하십시오.
   - 기본 연결 또는 사용자 지정 연결에서 공급자와의 연결을 아직 만들지 않은 경우 새 GitLab 연결 만들기를 선택합니다. 5단계로 진행하여 연결을 생성합니다.
   - 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 10단계로 이동합니다.

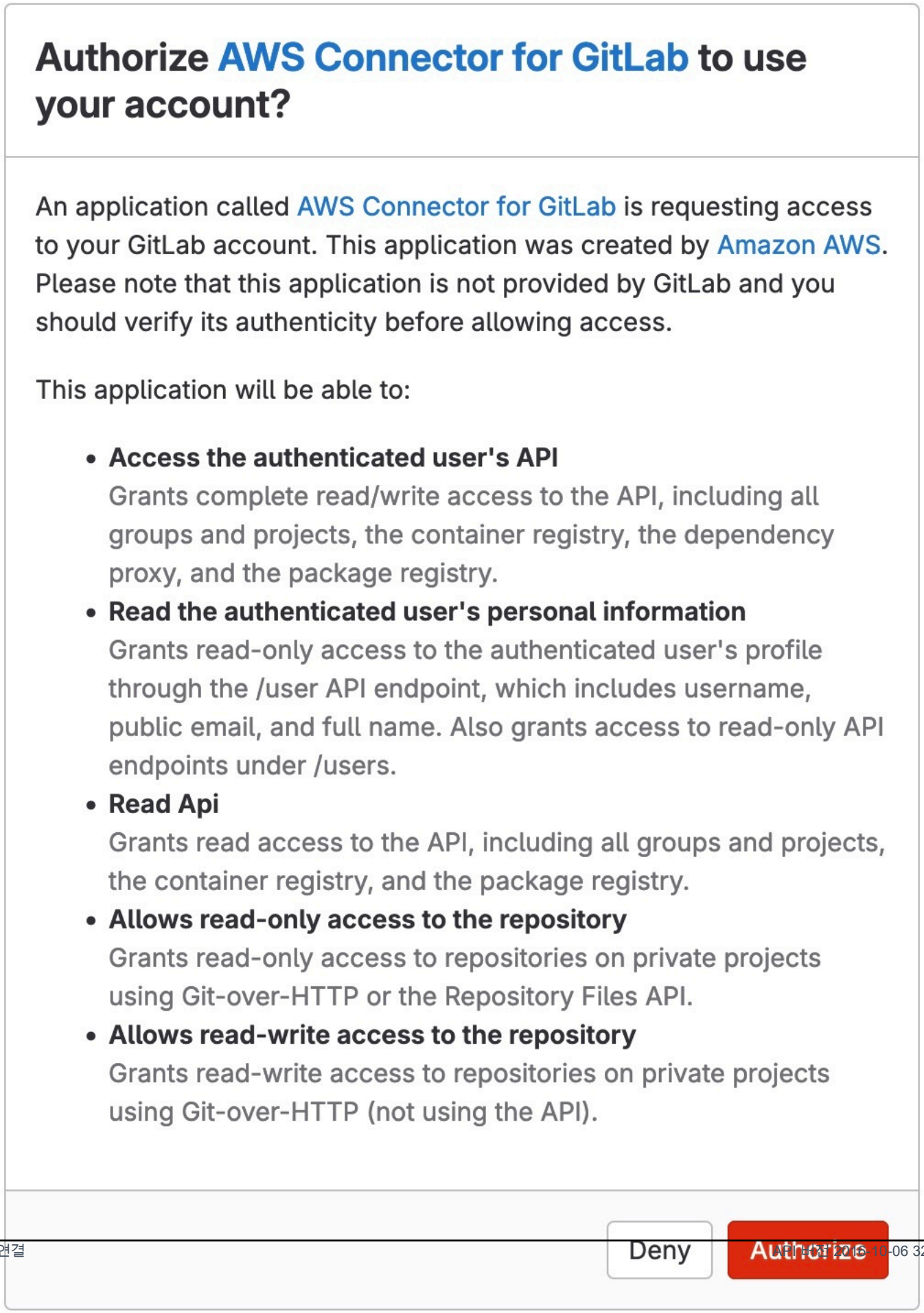
   > **Note**
   >
   > GitLab 연결이 생성되기 전에 팝업 창을 닫으면 페이지를 새로 고쳐야 합니다.

5. GitLab 리포지토리에 대한 연결을 만들려면 공급자 선택에서 을 선택합니다 GitLab. [연결 이름 (Connection name)]에 생성하려는 연결의 이름을 입력합니다. 연결 대상을 선택합니다 GitLab.



6. 의 로그인 페이지가 GitLab 표시되면 자격 증명으로 로그인한 다음 로그인을 선택합니다.
7. 연결을 처음으로 승인하는 경우 계정에 GitLab 액세스할 수 있도록 연결을 승인하라는 메시지가 포함된 승인 페이지가 표시됩니다.

   Authorize를 선택합니다.


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. 브라우저가 연결 콘솔 페이지로 돌아갑니다. GitLab연결 설정에서 새 연결이 연결 이름에 표시됩니다.
9. 연결을 선택합니다.

   GitLab 연결이 성공적으로 생성되면 상단에 성공 배너가 표시됩니다.
10. 빌드 프로젝트 만들기 페이지의 기본 연결 또는 사용자 지정 연결 드롭다운 목록에서 연결 ARN이 나열되어 있는지 확인합니다. 그렇지 않은 경우 새로 고침 버튼을 선택하여 표시되도록 하세요.
11. 리포지토리에서 네임스페이스로 프로젝트 경로를 GitLab 지정하여 프로젝트 이름을 선택합니다. 예를 들어 그룹 수준 리포지토리의 경우 리포지토리 이름을 `group-name/repository-name` 형식으로 입력합니다. 경로와 네임스페이스에 대한 자세한 내용은 https://docs.gitlab.com/ee/api/projects.html #의 `path_with_namespace` 필드를 참조하십시오. get-single-project 의 네임스페이스에 GitLab 대한 자세한 내용은 https://docs.gitlab.com/ee/user/namespace/ 을 참조하십시오.

   > **Note**
   >
   > 그룹의 GitLab 경우 네임스페이스를 사용하여 프로젝트 경로를 수동으로 지정해야 합니다. 예를 들어 그룹 `mygroup`의 `myrepo`라는 리포지토리의 경우 `mygroup/myrepo` 형식으로 입력합니다. URL에서 네임스페이스가 있는 프로젝트 경로를 찾을 수 있습니다. GitLab

12. 소스 버전 - 선택 사항에서 풀 리퀘스트 ID, 브랜치, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

   > **Note**
   >
   > `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

13. Git clone depth - 선택 사항에서 지정된 커밋 수만큼 기록이 잘린 얕은 클론을 만들 수 있습니다. 전체 복제가 필요할 경우 전체를 선택합니다.
14. 빌드 상태 - 선택 사항에서 빌드의 시작 및 완료 상태를 소스 제공자에게 보고하려면 빌드 시작 및 완료 시 소스 제공자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

GitLab (CLI) 에 대한 연결 생성

AWS Command Line Interface (AWS CLI) 를 사용하여 연결을 생성할 수 있습니다.

이렇게 하려면 create-connection 명령을 사용합니다.

> **Important**
>
> OR를 통해 생성된 AWS CloudFormation 연결은 기본적으로 PENDING 상태입니다. AWS CLI CLI를 사용하여 연결을 생성한 후 콘솔을 사용하여 연결을 편집하여 상태를 설정합니다. AWS CloudFormationAVAILABLE

연결 생성

- GitLab (CLI) 에 대한 연결 생성에 대한 개발자 도구 콘솔 사용 설명서의 지침을 따르십시오.

다음과 함께 웹후크 사용 AWS CodeBuild

AWS CodeBuild GitHub 엔터프라이즈 서버 GitHub GitLab, GitLab 자체 관리형 및 Bitbucket과의 웹후크 통합을 지원합니다.

주제

- AWS CodeBuild의 webhook 사용 모범 사례
- Bitbucket Webhook 이벤트
- GitHub 웹후크 이벤트
- GitLab 웹후크 이벤트

AWS CodeBuild의 webhook 사용 모범 사례

퍼블릭 리포지토리를 사용하여 webhook를 설정하는 프로젝트의 경우 다음 옵션을 권장합니다.

ACTOR_ACCOUNT_ID 필터 설정

프로젝트의 webhook 필터 그룹에 ACTOR_ACCOUNT_ID 필터를 추가하여 빌드를 트리거할 수 있는 사용자를 지정합니다. 에 전달되는 모든 웹후크 이벤트는 행위자의 식별자를 지정하는 발신자 정보와 함께 CodeBuild 제공됩니다. CodeBuild 필터에 제공된 정규 표현식 패턴을 기반으로 웹후크를 필터링합니다. 이 필터를 사용하여 빌드를 트리거할 수 있는 특정 사용자를 지정할 수 있습니다. 자세한 내용은 GitHub 웹후크 이벤트 및 Bitbucket Webhook 이벤트 섹션을 참조하세요.

FILE_PATH 필터 설정

프로젝트의 webhook 필터 그룹에 FILE_PATH 필터를 추가하여 변경 시 빌드를 트리거할 수 있는 파일을 포함하거나 제외합니다. 예를 들어 excludeMatchedPattern 속성과 함께 ^buildspec.yml$과 같은 정규 표현식 패턴을 사용하여 buildspec.yml 파일 변경에 대한 빌드 요청을 거부할 수 있습니다. 자세한 내용은 GitHub 웹후크 이벤트 및 Bitbucket Webhook 이벤트 섹션을 참조하세요.

빌드 IAM 역할의 권한 범위 좁히기

webhook로 트리거되는 빌드는 프로젝트에 지정된 IAM 서비스 역할을 사용합니다. 서비스 역할의 권한을 빌드를 실행하는 데 필요한 최소 권한 세트로 설정하는 것이 좋습니다. 예를 들어 테스트 및 배포 시나리오에서는 테스트용 프로젝트 하나와 배포용 프로젝트 하나를 생성합니다. 테스트 프로젝트는 리포지토리의 webhook 빌드를 수락하지만 리소스에 대한 쓰기 권한은 제공하지 않습니다. 배포 프로젝트는 리소스에 대한 쓰기 권한을 제공하고 webhook 필터는 신뢰할 수 있는 사용자만 빌드를 트리거할 수 있도록 구성됩니다.

인라인 또는 Amazon S3 저장 buildspec 사용

프로젝트 내에서 buildspec 인라인을 정의하거나 Amazon S3 버킷에 buildspec 파일을 저장하는 경우, buildspec 파일은 프로젝트 소유자만 볼 수 있습니다. 이렇게 하면 풀 요청이 buildspec 파일의 코드를 변경하여 원치 않는 빌드를 트리거하는 것을 방지할 수 있습니다. 자세한 내용은 API 레퍼런스의 ProjectSource.buildspec을 참조하십시오. CodeBuild

Bitbucket Webhook 이벤트

Webhook 필터 그룹을 사용하여 어느 Bitbucket Webhook 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 예를 들어 특정 분기가 변경된 경우에만 빌드가 트리거되도록 지정할 수 있습니다.

하나 이상의 Webhook 필터 그룹을 생성하여 어느 Webhook 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 필터 그룹이 true로 평가(그룹 내 모든 필터가 true로 평가)되면 빌드가 트리거됩니다. 필터 그룹을 생성할 때 다음을 지정합니다.

이벤트

Bitbucket의 경우 다음 이벤트 중 하나 이상을 선택할 수 있습니다.

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED
- PULL_REQUEST_CLOSED

webhook의 이벤트 유형은 X-Event-Key 필드의 헤더에 있습니다. 다음 표에서는 X-Event-Key 헤더 값이 이벤트 유형에 매핑되는 방법을 보여 줍니다.

> **Note**
>
> PULL_REQUEST_MERGED 이벤트 유형을 사용하는 webhook 필터 그룹을 생성할 경우 Bitbucket webhook 설정의 merged 이벤트를 활성화해야 합니다. 또한 declined 이벤트 유형을 사용하는 웹후크 필터 그룹을 만드는 경우 Bitbucket 웹후크 설정에서 이벤트를 활성화해야 합니다PULL_REQUEST_CLOSED.

X-Event-Key 헤더 값	이벤트 유형
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED
pullrequest:rejected	PULL_REQUEST_CLOSED

PULL_REQUEST_MERGED의 경우 풀 요청이 스쿼시 전략에 병합되고 풀 요청 분기가 닫히면 원래의 풀 요청 커밋은 더 이상 존재하지 않게 됩니다. 이 경우 CODEBUILD_WEBHOOK_MERGE_COMMIT 환경 변수에는 스쿼시된 병합 커밋의 식별자가 포함됩니다.

하나 이상의 선택적 필터

정규식을 사용하여 필터를 지정합니다. 이벤트가 빌드를 트리거하려면 연결된 그룹 내의 필터가 모두 true로 평가되어야 합니다.

`ACTOR_ACCOUNT_ID`(콘솔의 `ACTOR_ID`)

Bitbucket 계정 ID가 정규식 패턴과 일치하면 Webhook 이벤트가 빌드를 트리거합니다. Webhook 필터 페이로드에 있는 `actor` 객체의 `account_id` 속성에 이 값이 표시됩니다.

`HEAD_REF`

헤드 참조가 정규식 패턴(예: `refs/heads/branch-name` 및 `refs/tags/tag-name`)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. `HEAD_REF` 필터가 브랜치나 태그의 Git 참조 이름을 평가합니다. Webhook 페이로드의 `push` 객체에 있는 `new` 객체의 `name` 필드에 브랜치 또는 태그 이름이 표시됩니다. pull 요청 이벤트의 경우 webhook 페이로드의 `source` 객체에 있는 `branch` 객체의 `name` 필드에 브랜치 이름이 표시됩니다.

`BASE_REF`

베이스 참조가 정규식 패턴과 일치하면 webhook 이벤트가 빌드를 트리거합니다. `BASE_REF` 필터는 풀 요청 이벤트에서만 작동합니다(예: `refs/heads/branch-name`). `BASE_REF` 필터가 브랜치의 Git 참조 이름을 평가합니다. Webhook 페이로드의 `destination` 객체에 있는 `branch` 객체의 `name` 필드에 브랜치 이름이 표시됩니다.

`FILE_PATH`

변경된 파일의 경로가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

`COMMIT_MESSAGE`

헤드 커밋 메시지가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

`WORKFLOW_NAME`

웹후크는 워크플로 이름이 정규 표현식 패턴과 일치할 때 빌드를 트리거합니다.

> **Note**
>
> Bitbucket 리포지토리의 webhook 설정에서 webhook 페이로드를 찾을 수 있습니다.

주제

- Bitbucket Webhook 이벤트 필터링(콘솔)
- Bitbucket Webhook 이벤트 필터링(SDK)
- Bitbucket Webhook 이벤트 필터링(AWS CloudFormation)

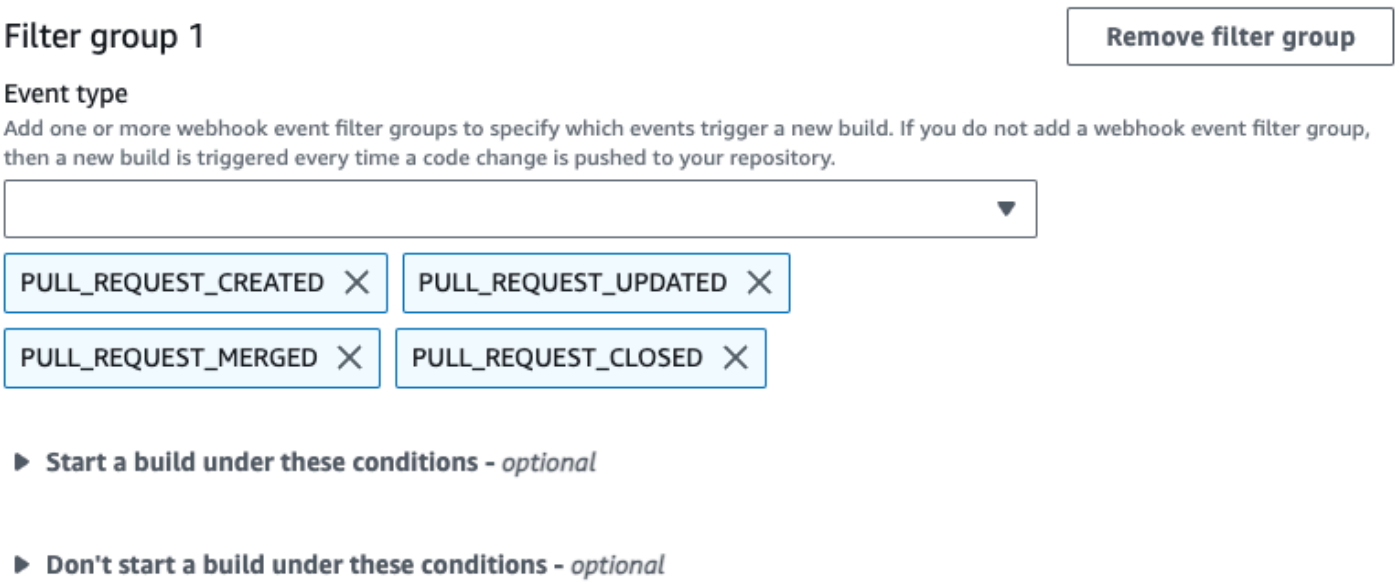
Bitbucket Webhook 이벤트 필터링(콘솔)

를 사용하여 웹후크 AWS Management Console 이벤트를 필터링하려면:

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. Add filter group(필터 그룹 추가)를 선택하여 다른 필터 그룹을 추가합니다.

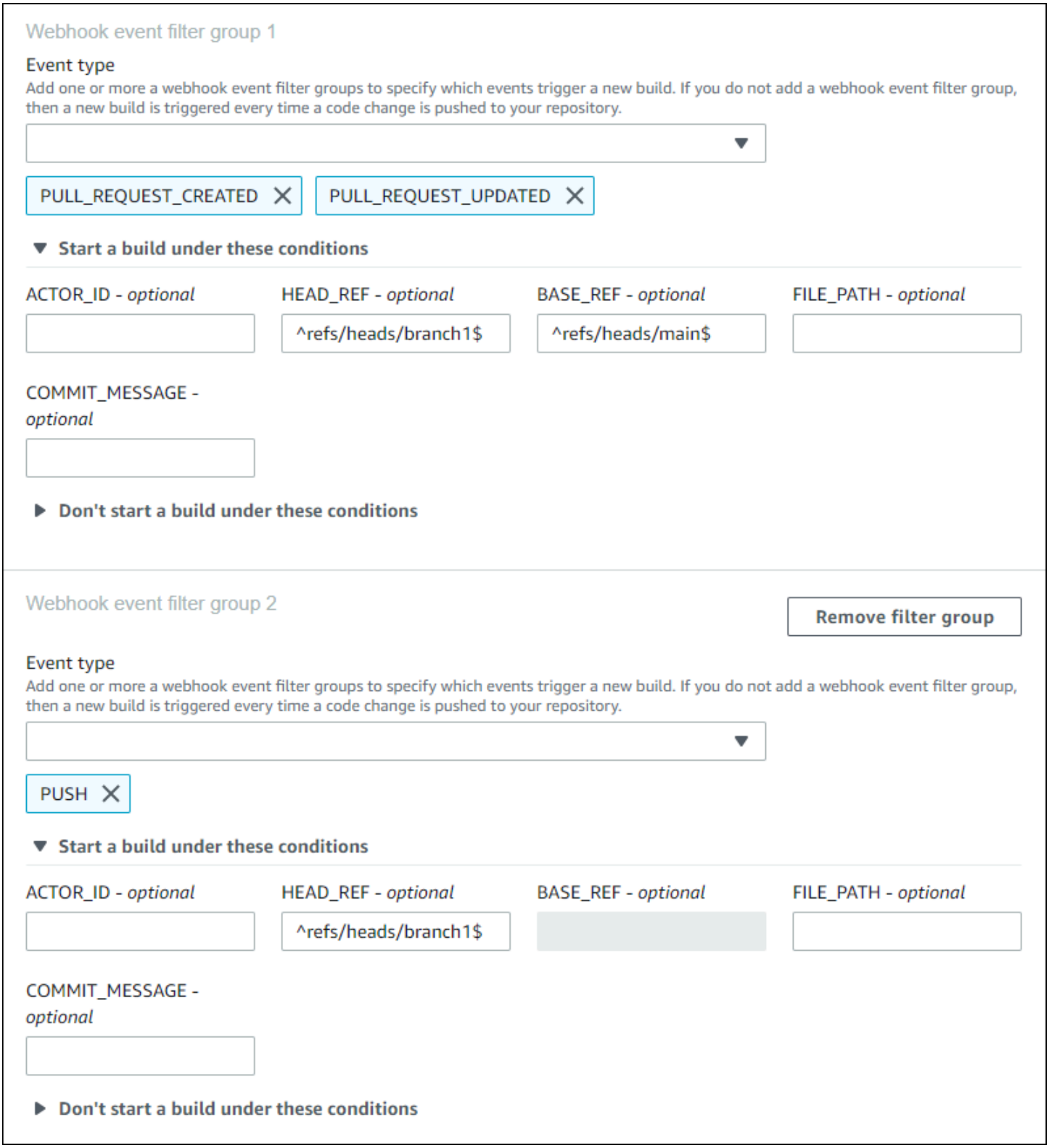
자세한 내용은 AWS CodeBuild API WebhookFilter참조의 빌드 프로젝트 만들기(콘솔) 및 를 참조하십시오.

이 예제에서는 Webhook 필터 그룹이 pull 요청에 대해서만 빌드를 트리거합니다.

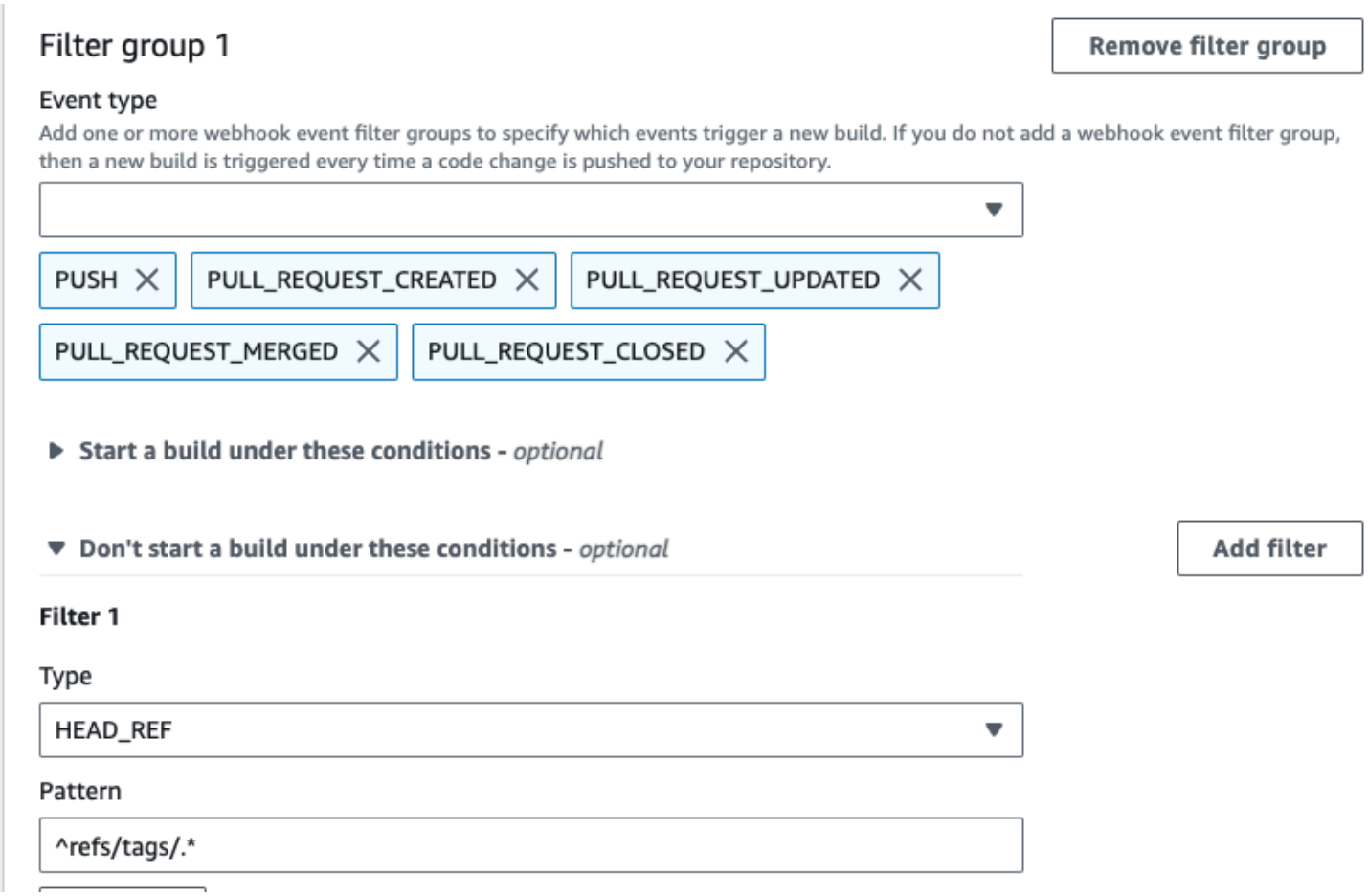


두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

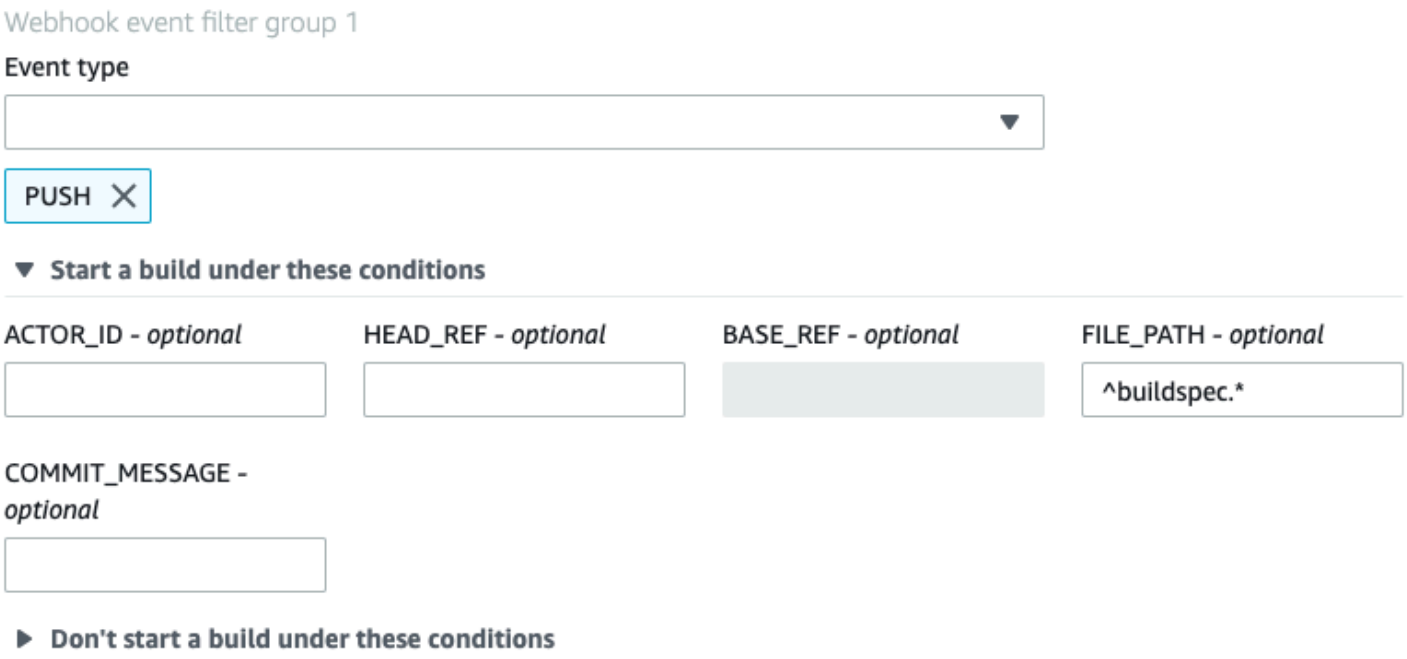
- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/branch1!`와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/branch1$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.



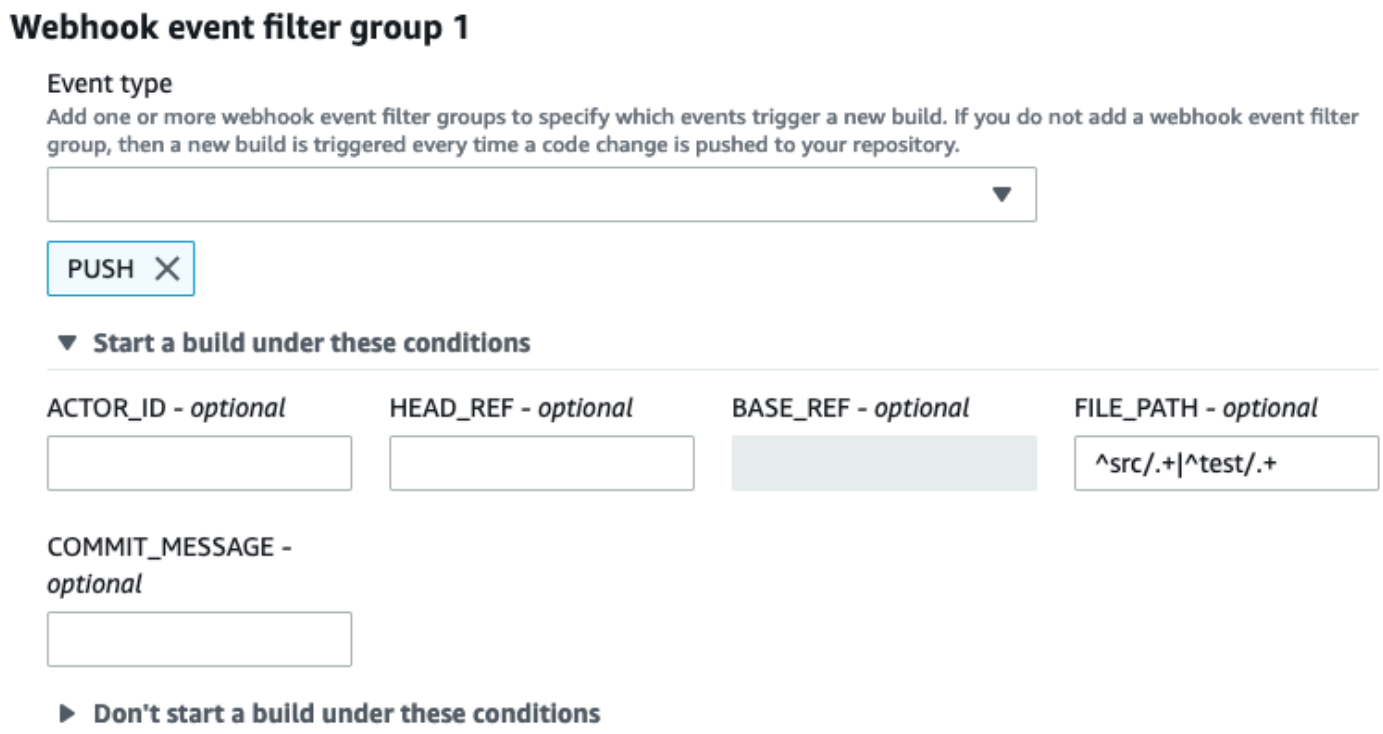
이 예제에서는 Webhook 필터 그룹이 태그 이벤트를 제외한 모든 요청에 대해 빌드를 트리거합니다.



이 예제에서는 Webhook 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드를 트리거합니다.

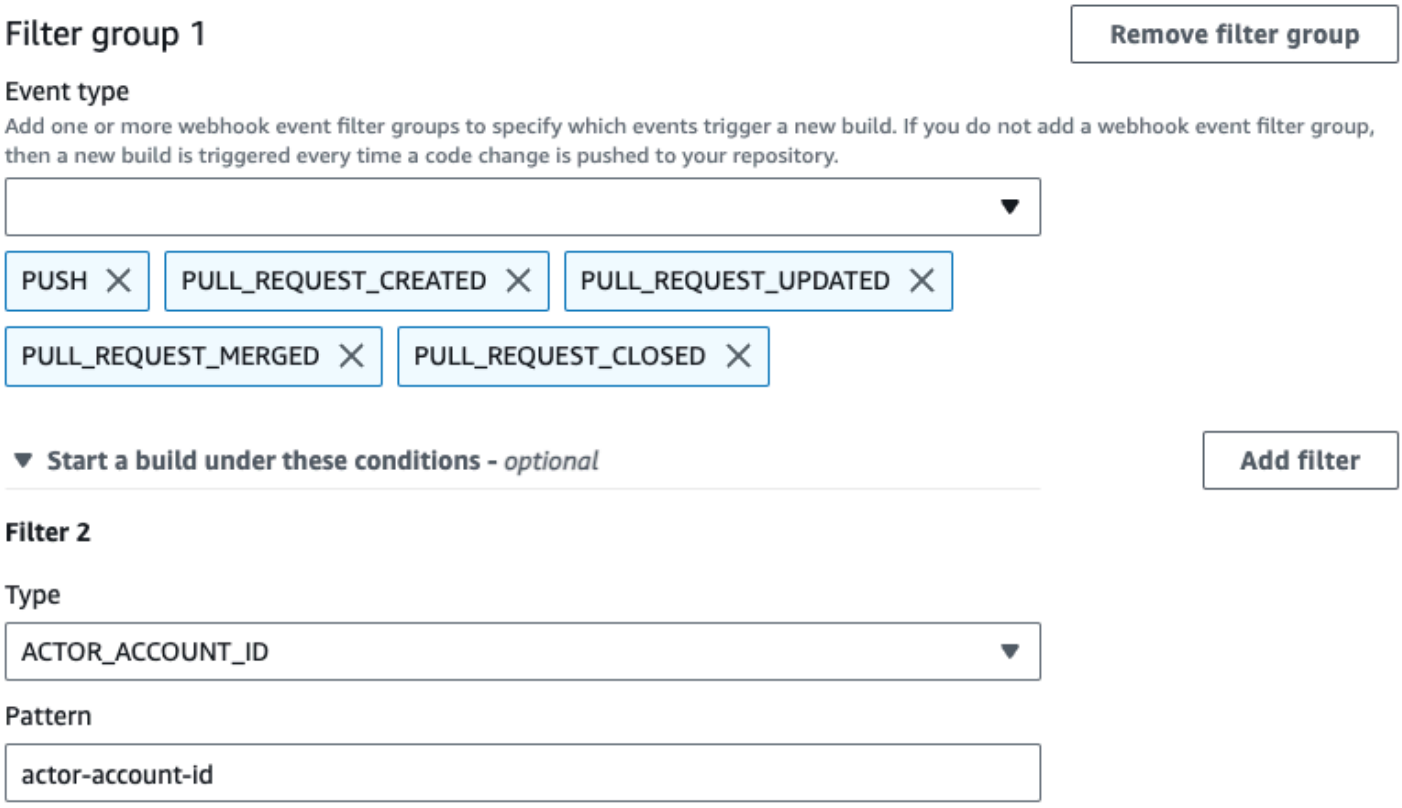


이 예제에서 Webhook 필터 그룹은 파일이 `src` 또는 `test` 폴더에서 변경된 경우에만 빌드를 트리거합니다.

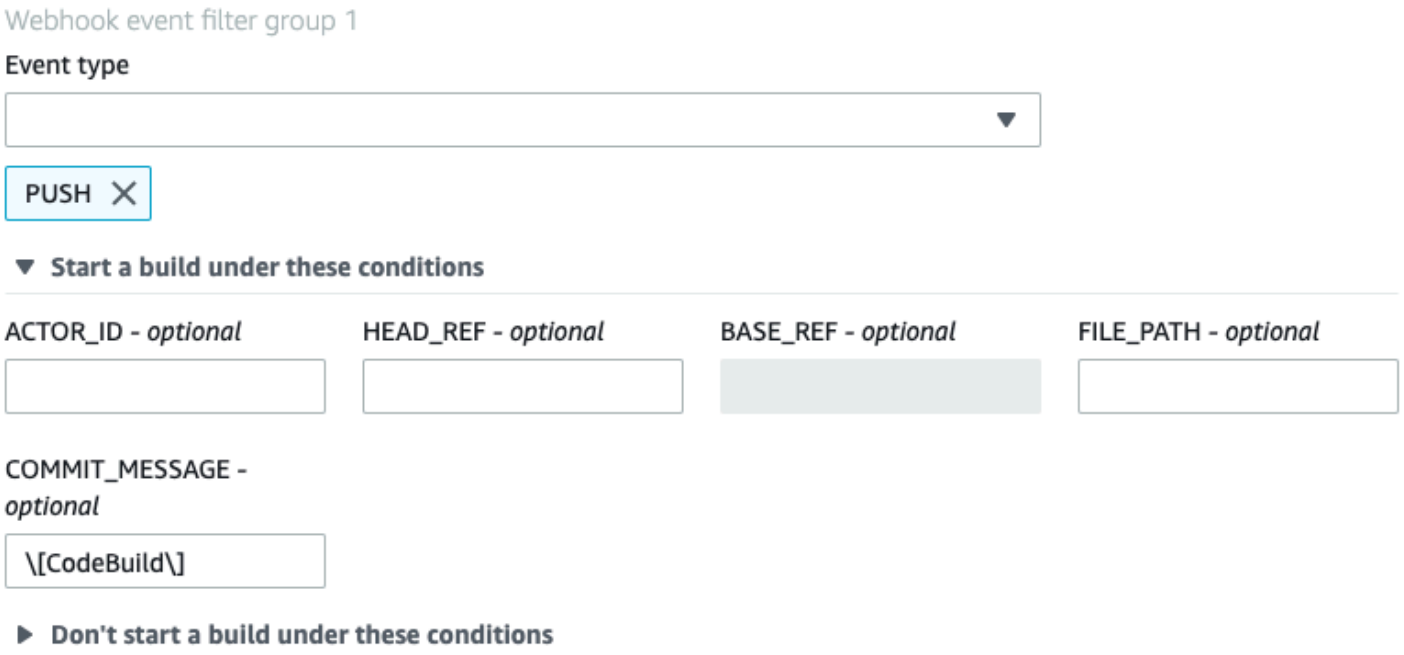


이 예제에서는 Webhook 필터 그룹이 계정 ID가 정규식 `actor-account-id`와 일치하지 않는 Bitbucket 사용자가 변경을 수행한 경우에만 빌드를 트리거합니다.

> **Note**
>
> Bitbucket 계정 ID를 확인하는 자세한 방법은 https://api.bitbucket.org/2.0/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 Bitbucket 사용자 이름입니다.



이 예제에서 webhook 필터 그룹은 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때 푸시 이벤트에 대한 빌드를 트리거합니다.



Bitbucket Webhook 이벤트 필터링(SDK)

AWS CodeBuild SDK를 사용하여 웹후크 이벤트를 필터링하려면 `CreateWebhook` 또는 `UpdateWebhook` API 메서드의 요청 구문에 있는 `filterGroups` 필드를 사용하십시오. 자세한 내용은 CodeBuild API WebhookFilter참조를 참조하십시오.

pull 요청에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 요청 구문에 다음을 삽입합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED,
 PULL_REQUEST_CLOSED"
    }
  ]
]
```

지정된 브랜치에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 `pattern` 파라미터를 사용하여 브랜치 이름을 필터링하는 정규식을 지정합니다. 두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 ^refs/heads/main$와 일치하는 Git 참조 이름과 ^refs/heads/myBranch$와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 ^refs/heads/myBranch$와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

```
"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$"
    }
  ]
]
```

excludeMatchedPattern 파라미터를 사용하여 빌드를 트리거하지 않을 이벤트를 지정할 수 있습니다. 이 예제에서는 태그 이벤트를 제외한 모든 요청에 대해 빌드가 트리거됩니다.

```
"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
    }
  ]
]
```

계정 ID가 `actor-account-id`인 Bitbucket 사용자가 변경을 수행한 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다.

> **Note**
>
> Bitbucket 계정 ID를 확인하는 자세한 방법은 https://api.bitbucket.org/2.0/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 Bitbucket 사용자 이름입니다.

```
"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"
    }
  ]
]
```

`pattern` 인수의 정규식과 일치하는 이름을 갖는 파일이 변경되는 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서는 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
  [
    {
```

```
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

이 예제에서 필터 그룹은 파일이 src 또는 test 폴더에서 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

헤드 커밋 메시지가 패턴 인수의 정규식과 일치할 때만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서 필터 그룹은 푸시 이벤트의 헤드 커밋 메시지가 정규식 \[CodeBuild\]와 일치할 때만 빌드가 트리거되도록 지정합니다.

```
  "filterGroups": [
    [
      {
        "type": "EVENT",
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
      }
    ]
  ]
```

Bitbucket Webhook 이벤트 필터링(AWS CloudFormation)

AWS CloudFormation 템플릿을 사용하여 웹후크 이벤트를 필터링하려면 AWS CodeBuild 프로젝트의 `FilterGroups` 속성을 사용하십시오. 다음 AWS CloudFormation 템플릿의 YAML 형식 부분은 두 개의 필터 그룹을 생성합니다. 이들은 하나 또는 둘 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 `12345`와 일치하지 않는 Bitbucket 사용자가 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성 또는 업데이트한 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/.*`와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 `\[CodeBuild\]`와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.

```
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 웹후크 이벤트

웹후크 필터 그룹을 사용하여 빌드를 트리거하는 GitHub 웹후크 이벤트를 지정할 수 있습니다. 예를 들어 특정 분기가 변경된 경우에만 빌드가 트리거되도록 지정할 수 있습니다.

하나 이상의 Webhook 필터 그룹을 생성하여 어느 Webhook 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 필터 그룹이 true로 평가(그룹 내 모든 필터가 true로 평가)되면 빌드가 트리거됩니다. 필터 그룹을 생성할 때 다음을 지정합니다.

이벤트

의 경우 GitHub,,,,PUSH,, PULL_REQUEST_CREATED PULL_REQUEST_UPDATED PULL_REQUEST_REOPENED PULL_REQUEST_MERGED PULL_REQUEST_CLOSED RELEASEDPRERELEASED, 및 이벤트 중 하나 이상을 선택할 수 있습니다. WORKFLOW_JOB_QUEUED webhook 이벤트 유형은 webhook 페이로드의 `X-GitHub-Event` 헤더에 있습니다. `X-GitHub-Event` 헤더에서 `pull_request` 또는 `push`를 볼 수 있습니다. 풀 요청 이벤트의 경우 유형은 webhook 이벤트 페이로드의 `action` 필드에 있습니다. 다음 표에서는 `X-GitHub-Event` 헤더 값과 webhook 풀 요청 페이로드 `action` 필드 값이 사용 가능한 이벤트 유형에 매핑되는 방법을 보여 줍니다.

`X-GitHub-Event` 헤더 값	Webhook 이벤트 페이로드 **`action`** 값	이벤트 유형
`pull_request`	`opened`	`PULL_REQUEST_CREATED`
`pull_request`	`reopened`	`PULL_REQUEST_REOPENED`
`pull_request`	`synchronize`	`PULL_REQUEST_UPDATED`

X-GitHub-Event 헤더 값	Webhook 이벤트 페이로드 **action** 값	이벤트 유형
pull_request	closed 및 merged 필드는 true임	PULL_REQUEST_MERGED
pull_request	closed 및 merged 필드는 false임	PULL_REQUEST_CLOSED
push	해당 사항 없음	PUSH
release	공개	RELEASED
release	프리릴리즈	PRERELEASED
workflow_job	queued	WORKFLOW_JOB_QUEUED

> **Note**
>
> PULL_REQUEST_REOPENED이벤트 유형은 GitHub 엔터프라이즈 서버에서만 사용할 수 있습니다. GitHub RELEASEDPRERELEASED, 및 WORKFLOW_JOB_QUEUED 이벤트 유형은 GitHub 함께만 사용할 수 있습니다. WORKFLOW_JOB_QUEUED에 대한 자세한 내용은 자습서: CodeBuild 자체 호스팅 Actions 러너 GitHub 구성 단원을 참조하세요.

하나 이상의 선택적 필터

정규식을 사용하여 필터를 지정합니다. 이벤트가 빌드를 트리거하려면 연결된 그룹 내의 필터가 모두 true로 평가되어야 합니다.

ACTOR_ACCOUNT_ID(콘솔의 ACTOR_ID)

웹후크 이벤트는 GitHub 또는 GitHub Enterprise Server 계정 ID가 정규 표현식 패턴과 일치할 때 빌드를 트리거합니다. webhook 페이로드에 있는 sender 객체의 id 속성에서 이 값을 찾을 수 있습니다.

HEAD_REF

헤드 참조가 정규식 패턴(예: refs/heads/branch-name 또는 refs/tags/tag-name)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. 푸시 이벤트의 경우 webhook 페이로드의

ref 속성에서 참조 이름을 찾을 수 있습니다. 풀 요청 이벤트의 경우 webhook 페이로드에 있는 head 객체의 ref 속성에서 브랜치 이름을 찾을 수 있습니다.

BASE_REF

기본 참조가 정규식 패턴(예: refs/heads/branch-name)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. 풀 요청 이벤트에서만 BASE_REF 필터를 사용할 수 있습니다. webhook 페이로드에 있는 base 객체의 ref 속성에서 브랜치 이름을 찾을 수 있습니다.

FILE_PATH

변경된 파일의 경로가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.
FILE_PATH필터는 GitHub 푸시 및 풀 요청 이벤트 및 GitHub 엔터프라이즈 서버 푸시 이벤트와 함께 사용할 수 있습니다. GitHub엔터프라이즈 서버 풀 리퀘스트 이벤트와 함께 사용할 수 없습니다.

COMMIT_MESSAGE

헤드 커밋 메시지가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.
COMMIT_MESSAGE필터는 GitHub 푸시 및 풀 요청 이벤트 및 GitHub 엔터프라이즈 서버 푸시 이벤트와 함께 사용할 수 있습니다. GitHub엔터프라이즈 서버 풀 리퀘스트 이벤트와 함께 사용할 수 없습니다.

TAG_NAME

웹후크는 릴리스의 태그 이름이 정규 표현식 패턴과 일치할 때 빌드를 트리거합니다.
TAG_NAME필터는 릴리즈된 요청 이벤트 및 GitHub 프리릴리즈된 요청 이벤트와 함께 사용할 수 있습니다.

RELEASE_NAME

웹후크는 릴리스 이름이 정규 표현식 패턴과 일치할 때 빌드를 트리거합니다.
RELEASE_NAME필터는 릴리즈된 요청 이벤트 및 GitHub 프리릴리즈된 요청 이벤트와 함께 사용할 수 있습니다.

WORKFLOW_NAME

웹후크는 워크플로 이름이 정규 표현식 패턴과 일치할 때 빌드를 트리거합니다.
WORKFLOW_NAME필터는 GitHub Actions 워크플로 작업 대기 요청 이벤트와 함께 사용할 수 있습니다.

> **Note**
>
> 웹후크 페이로드는 리포지토리의 웹후크 설정에서 찾을 수 있습니다. GitHub

주제

- GitHub 웹후크 이벤트 필터링 (콘솔)
- GitHub 웹후크 이벤트 (SDK) 를 필터링합니다.
- GitHub 웹후크 이벤트 필터링 ()AWS CloudFormation

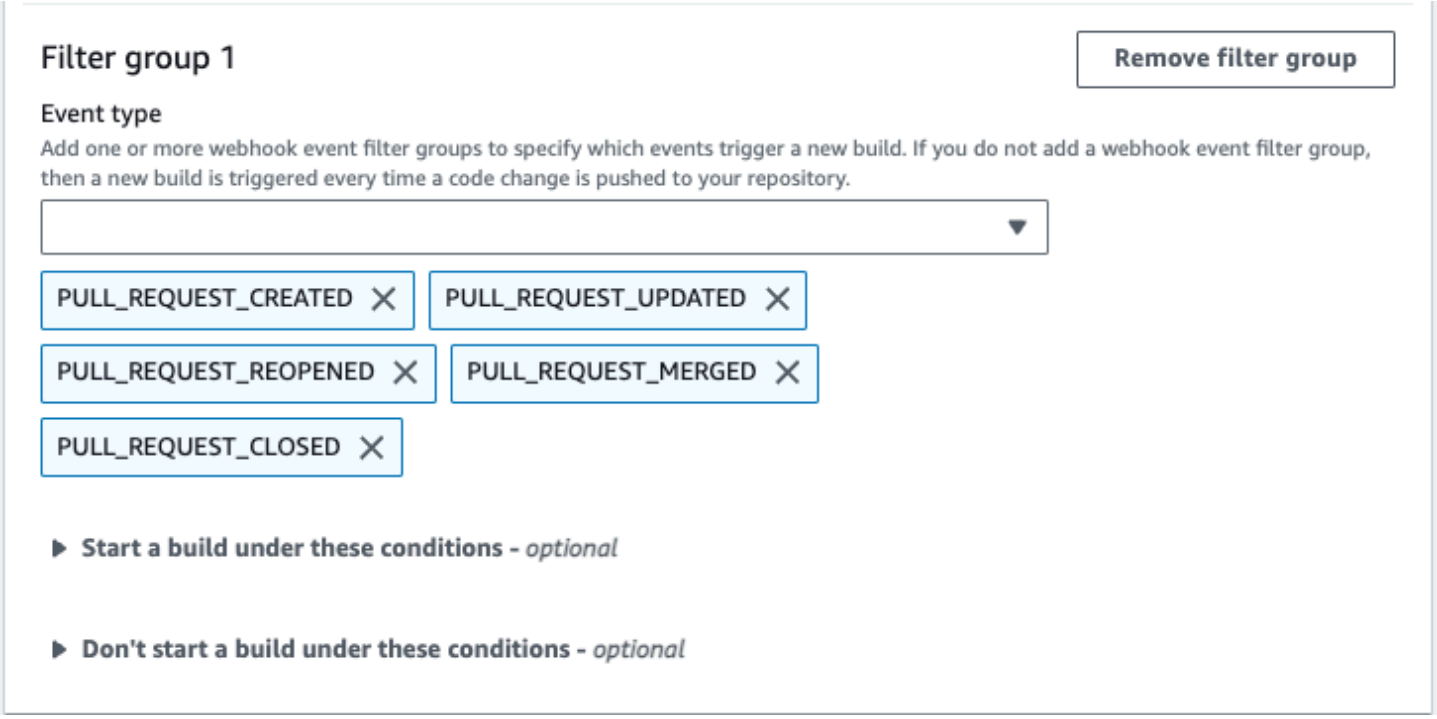
GitHub 웹후크 이벤트 필터링 (콘솔)

기본 소스 webhook 이벤트에서 다음을 선택합니다. 이 섹션은 내 GitHub 계정에서 소스 리포지토리로 리포지토리를 선택한 경우에만 사용할 수 있습니다.

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. 필요한 경우 필터 그룹 추가를 선택하여 다른 필터 그룹을 추가합니다.

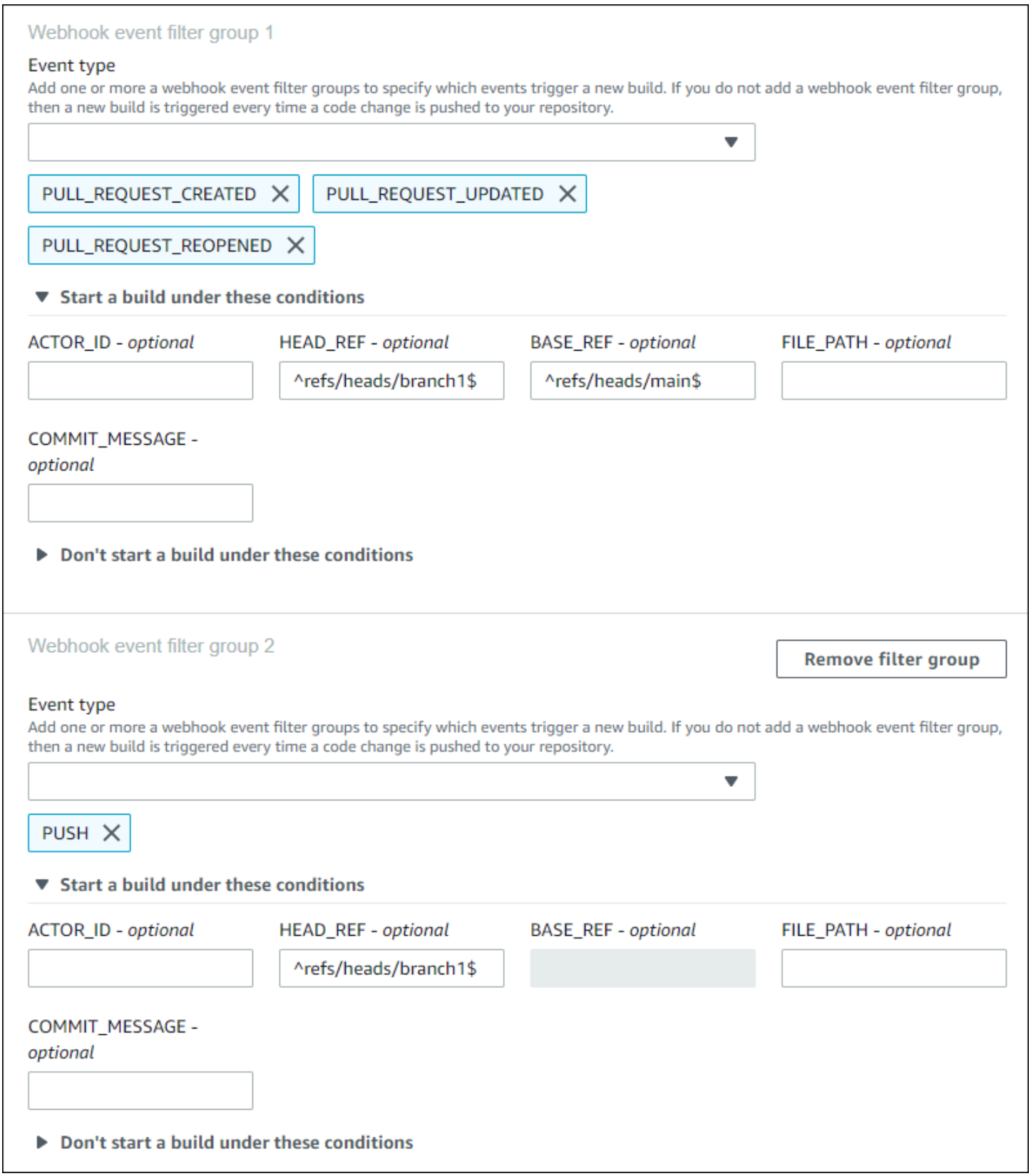
자세한 내용은 AWS CodeBuild API WebhookFilter참조의 빌드 프로젝트 만들기(콘솔) 및 를 참조하십시오.

이 예제에서는 Webhook 필터 그룹이 pull 요청에 대해서만 빌드를 트리거합니다.

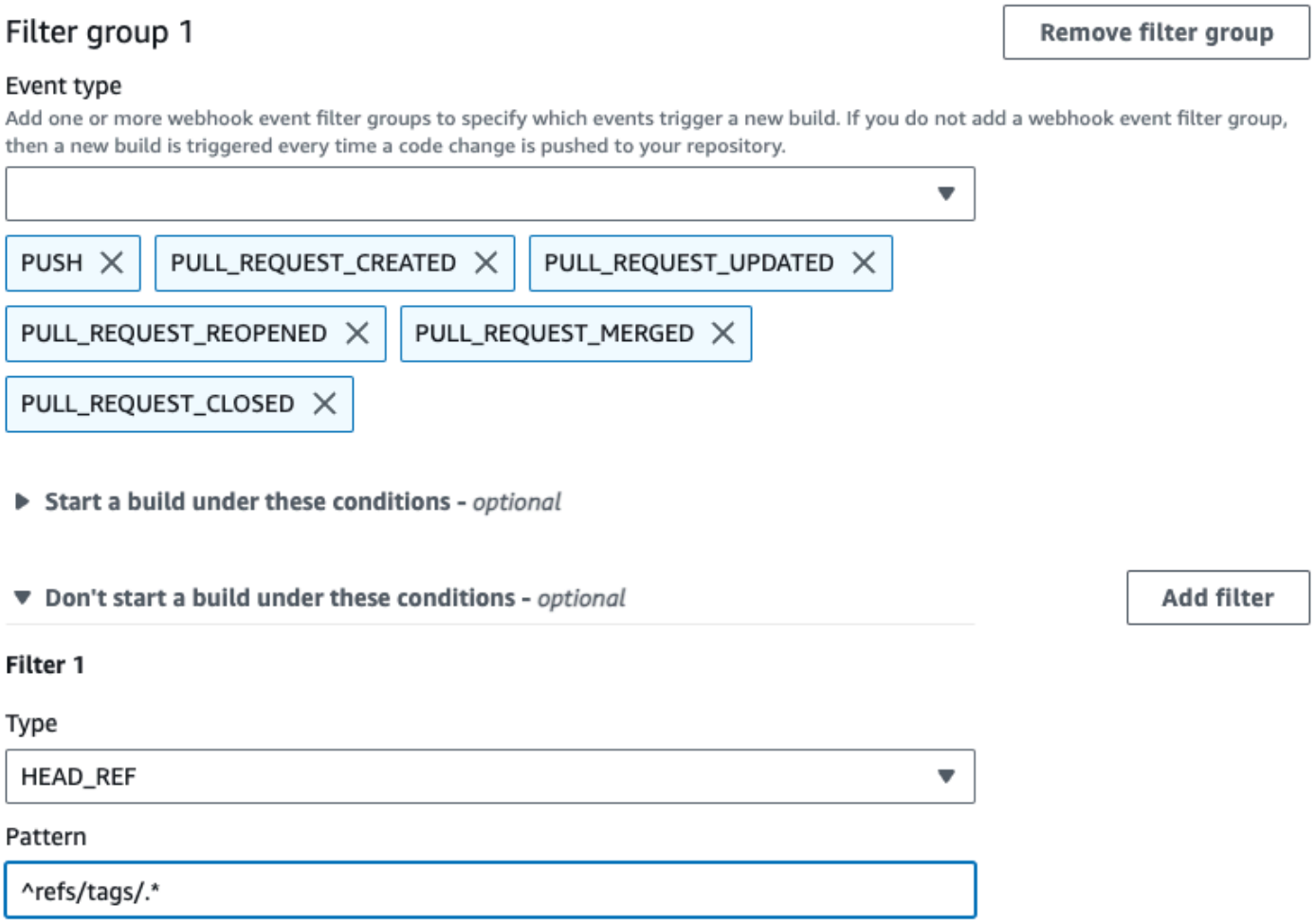


두 Webhook 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

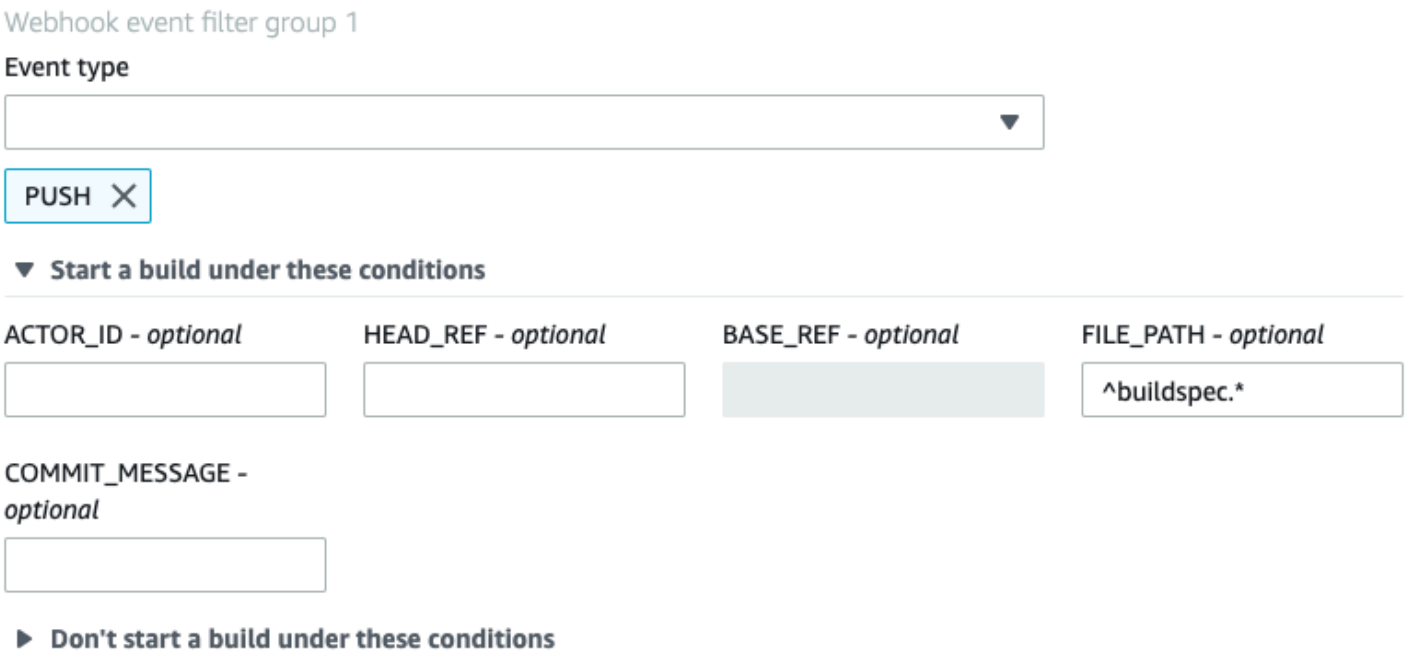
- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/branch1$`와 일치하는 헤드 참조를 갖는 브랜치에서 생성되거나 업데이트되거나 다시 열린 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/branch1$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.



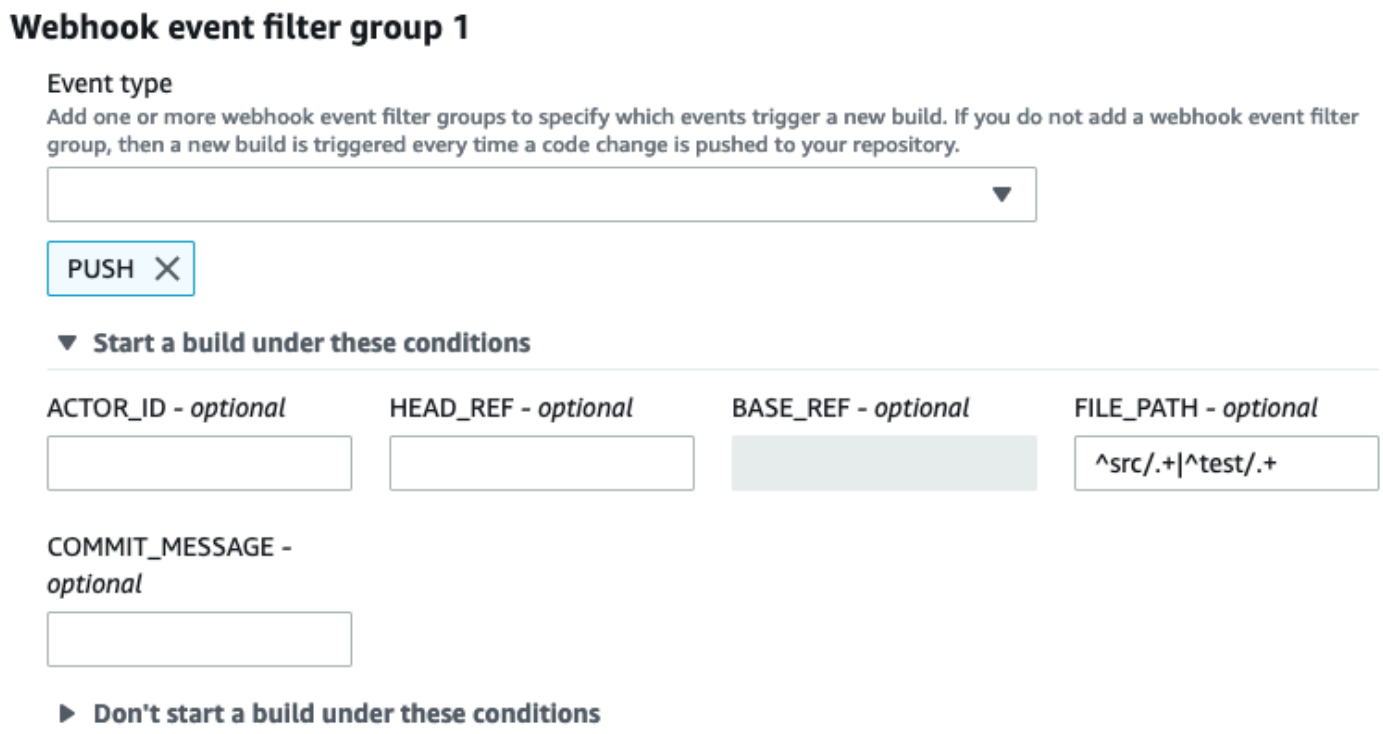
이 예제에서는 Webhook 필터 그룹이 태그 이벤트를 제외한 모든 요청에 대해 빌드를 트리거합니다.



이 예제에서는 Webhook 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드를 트리거합니다.

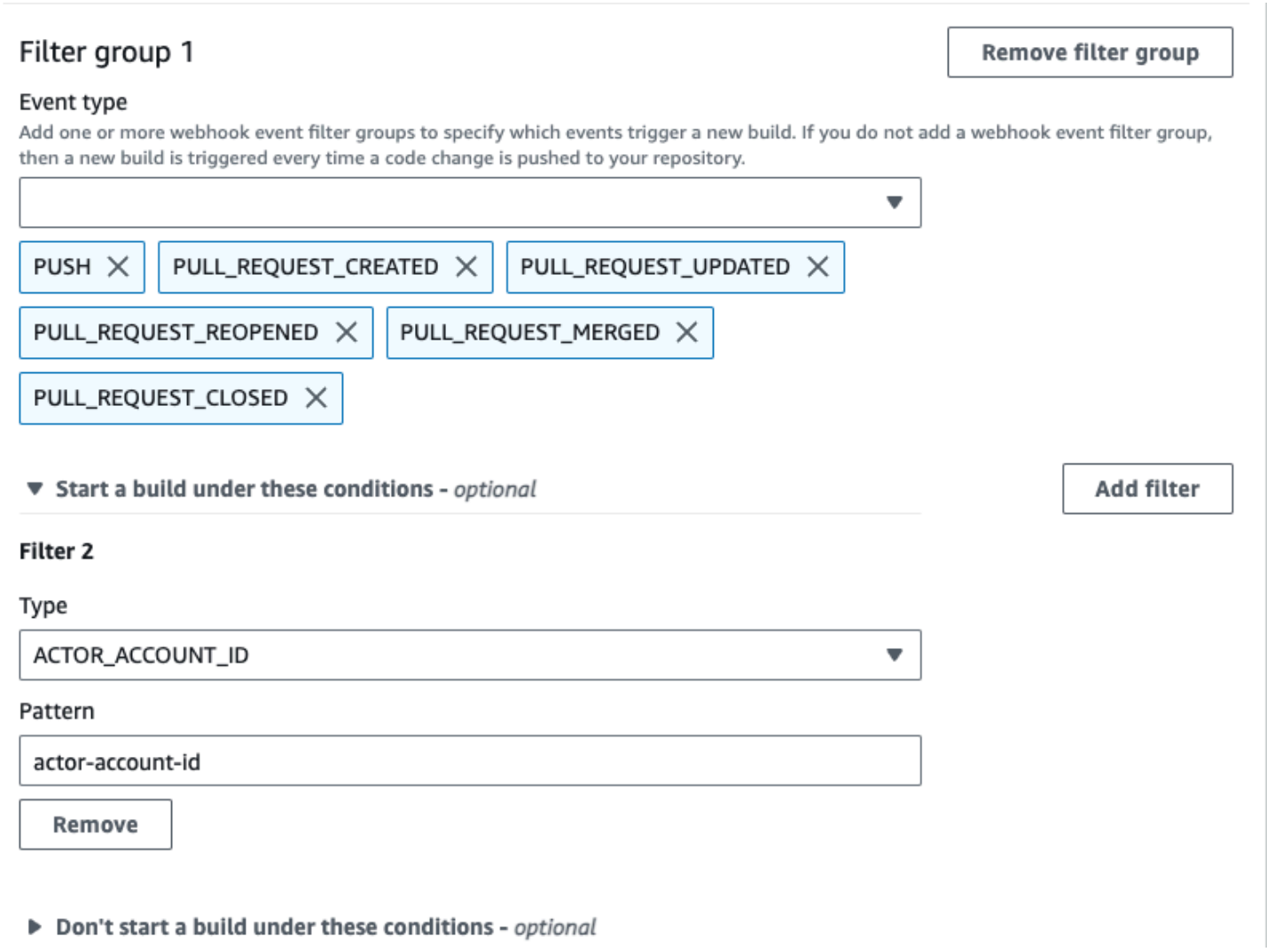


이 예제에서 Webhook 필터 그룹은 파일이 `src` 또는 `test` 폴더에서 변경된 경우에만 빌드를 트리거합니다.

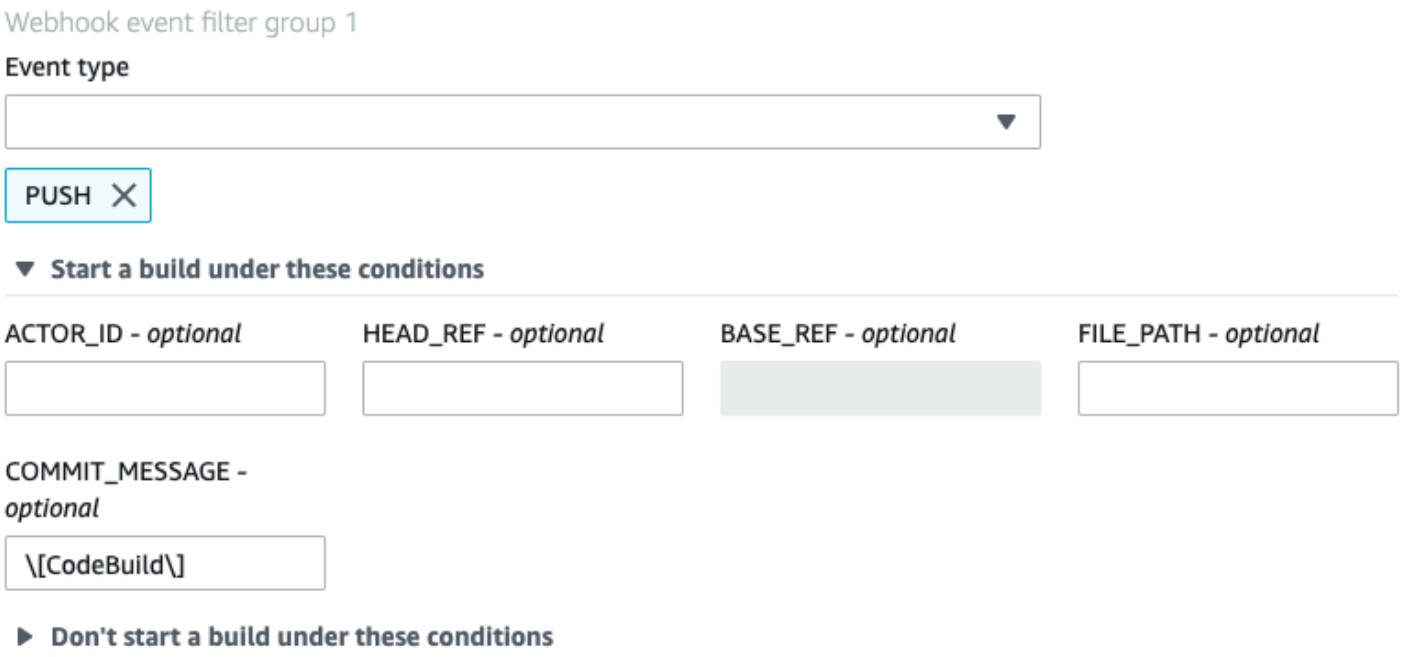


이 예제에서 웹후크 필터 그룹은 정규식과 일치하는 계정 ID를 가진 지정된 사용자 GitHub 또는 GitHub Enterprise Server 사용자가 변경한 경우에만 빌드를 트리거합니다. `actor-account-id`

> **Note**
>
> GitHub 계정 ID를 찾는 방법에 대한 자세한 내용은 https://api.github.com/users/ ### ### ## ####. ### ### ### ### ### #####. GitHub

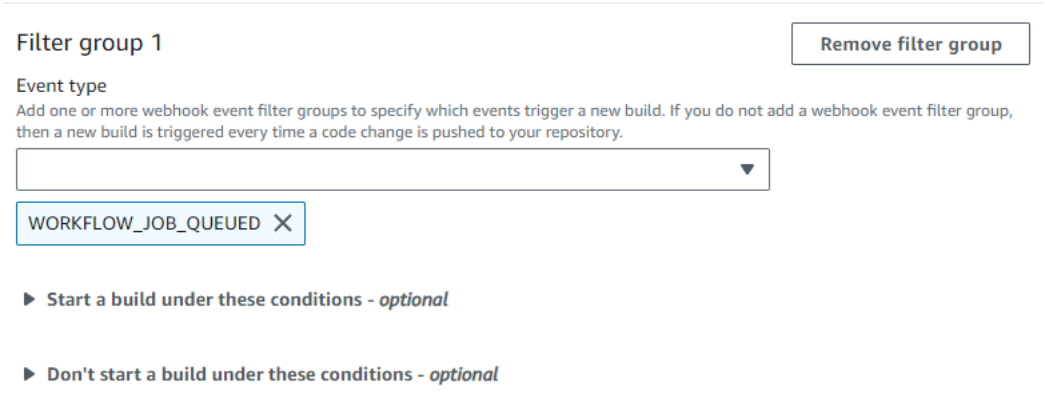


이 예제에서 webhook 필터 그룹은 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때 푸시 이벤트에 대한 빌드를 트리거합니다.

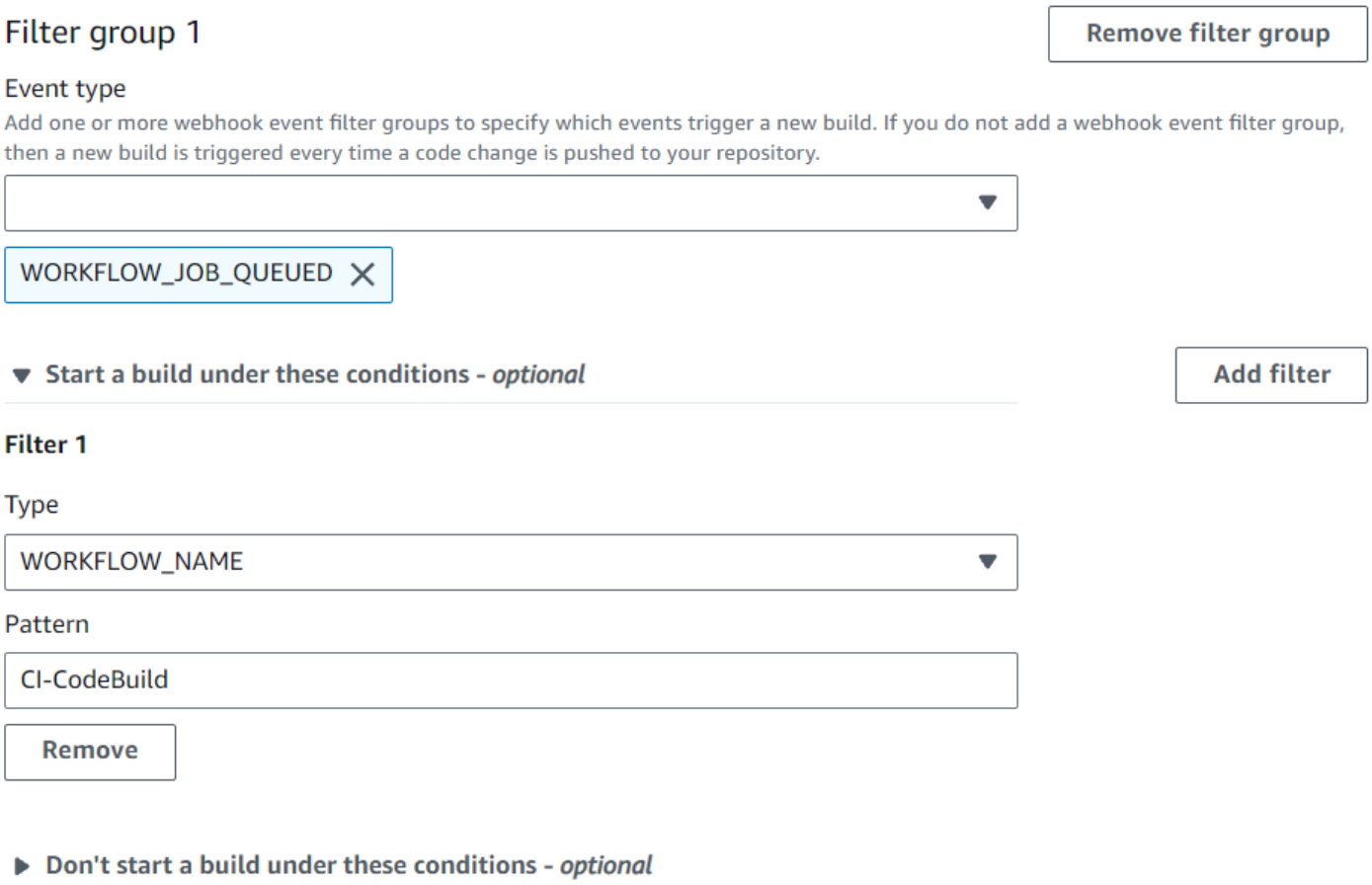


이 예제에서 웹후크 필터 그룹은 GitHub Actions 빌드용 워크플로 작업 이벤트만 트리거합니다.

> **Note**
>
> CodeBuild 웹후크에 WORKFLOW_JOB_QUEUED 이벤트 필터가 포함된 필터 그룹이 있는 경우에만 작업 워크플로 GitHub 작업을 처리합니다.



이 예제에서 웹후크 필터 그룹은 정규 표현식과 일치하는 워크플로 이름에 대한 빌드를 트리거합니다. `CI-CodeBuild`



GitHub 웹후크 이벤트 (SDK) 를 필터링합니다.

AWS CodeBuild SDK를 사용하여 웹후크 이벤트를 필터링하려면 또는 API 메서드의 요청 구문에 있는 `filterGroups` 필드를 사용하십시오. `CreateWebhook` `UpdateWebhook` 자세한 내용은 CodeBuild API WebhookFilter참조를 참조하십시오.

pull 요청에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 요청 구문에 다음을 삽입합니다.

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        }
    ]
]
```

지정된 브랜치에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 pattern 파라미터를 사용하여 브랜치 이름을 필터링하는 정규식을 지정합니다. 두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 ^refs/heads/main$와 일치하는 Git 참조 이름과 ^refs/heads/myBranch$와 일치하는 헤드 참조를 갖는 브랜치에서 생성되거나 업데이트되거나 다시 열린 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 ^refs/heads/myBranch$와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

```
"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$"
        }
    ]
]
```

excludeMatchedPattern 파라미터를 사용하여 빌드를 트리거하지 않을 이벤트를 지정할 수 있습니다. 예를 들어 이 예제에서는 태그 이벤트를 제외한 모든 요청에 대해 빌드가 트리거됩니다.

```
"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
        }
    ]
]
```

pattern 인수의 정규식과 일치하는 이름을 갖는 파일이 변경되는 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서는 필터 그룹이 정규식 ^buildspec.*와 일치하는 이름을 갖는 파일이 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^buildspec.*"
        }
    ]
]
```

이 예제에서 필터 그룹은 파일이 src 또는 test 폴더에서 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
```

```
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

계정 ID를 `actor-account-id` 가진 지정된 사용자 GitHub 또는 GitHub Enterprise Server 사용자가 변경한 경우에만 빌드를 트리거하는 필터를 만들 수 있습니다.

> **Note**
>
> GitHub 계정 ID를 찾는 방법에 대한 자세한 내용은 https://api.github.com/users/ *username#* 참조하십시오. 여기서 *### ### ### GitHub #####*.

```
"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"
        }
    ]
]
```

헤드 커밋 메시지가 패턴 인수의 정규식과 일치할 때만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서 필터 그룹은 푸시 이벤트의 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "COMMIT_MESSAGE",
```

```
            "pattern": "\[CodeBuild\]"
        }
    ]
]
```

Actions용 빌드 워크플로 GitHub 작업만 트리거하는 웹후크 필터를 만들려면 요청 구문에 다음을 삽입하세요.

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

GitHub 웹후크 이벤트 필터링 ()AWS CloudFormation

AWS CloudFormation 템플릿을 사용하여 웹후크 이벤트를 필터링하려면 AWS CodeBuild 프로젝트의 `FilterGroups` 속성을 사용하십시오. 다음 AWS CloudFormation 템플릿의 YAML 형식 부분은 두 개의 필터 그룹을 생성합니다. 이들은 하나 또는 둘 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 없는 GitHub 사용자가 정규 표현식과 `^refs/heads/main$` 일치하는 Git 참조 이름을 가진 브랜치에서 pull 요청을 만들거나 업데이트하도록 지정합니다. 12345
- 두 번째 필터 그룹은 정규식 `^refs/heads/.*`와 일치하는 Git 참조 이름을 갖는 브랜치에서 정규식 `READ_ME`와 일치하는 이름을 갖는 파일에 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 `\[CodeBuild\]`와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.
- 네 번째 필터 그룹은 정규 표현식과 `\[CI-CodeBuild\]` 일치하는 워크플로 이름을 사용하여 GitHub Actions 워크플로 작업 요청을 지정합니다.

```
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 웹후크 이벤트

웹후크 필터 그룹을 사용하여 빌드를 트리거하는 GitLab 웹후크 이벤트를 지정할 수 있습니다. 예를 들어 특정 분기가 변경된 경우에만 빌드가 트리거되도록 지정할 수 있습니다.

하나 이상의 Webhook 필터 그룹을 생성하여 어느 Webhook 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 필터 그룹이 true로 평가(그룹 내 모든 필터가 true로 평가)되면 빌드가 트리거됩니다. 필터 그룹을 생성할 때 다음을 지정합니다.

이벤트

의 GitLab 경우 다음 이벤트 중 하나 이상을 선택할 수 있습니다.

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED

webhook의 이벤트 유형은 X-Event-Key 필드의 헤더에 있습니다. 다음 표에서는 X-Event-Key 헤더 값이 이벤트 유형에 매핑되는 방법을 보여 줍니다.

> **Note**
>
> merged이벤트 유형을 사용하는 GitLab 웹후크 필터 그룹을 만드는 경우 웹후크 설정에서 PULL_REQUEST_MERGED 이벤트를 활성화해야 합니다.

X-Event-Key 헤더 값	이벤트 유형
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED

PULL_REQUEST_MERGED의 경우 풀 요청이 스쿼시 전략에 병합되고 풀 요청 분기가 닫히면 원래의 풀 요청 커밋은 더 이상 존재하지 않게 됩니다. 이 경우 CODEBUILD_WEBHOOK_MERGE_COMMIT 환경 변수에는 스쿼시된 병합 커밋의 식별자가 포함됩니다.

하나 이상의 선택적 필터

정규식을 사용하여 필터를 지정합니다. 이벤트가 빌드를 트리거하려면 연결된 그룹 내의 필터가 모두 true로 평가되어야 합니다.

ACTOR_ACCOUNT_ID(콘솔의 ACTOR_ID)

웹후크 이벤트는 GitLab 계정 ID가 정규 표현식 패턴과 일치할 때 빌드를 트리거합니다. Webhook 필터 페이로드에 있는 actor 객체의 account_id 속성에 이 값이 표시됩니다.

HEAD_REF

헤드 참조가 정규식 패턴(예: refs/heads/branch-name 및 refs/tags/tag-name)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. HEAD_REF 필터가 브랜치나 태그의 Git 참조 이름을 평가합니다. Webhook 페이로드의 push 객체에 있는 new 객체의 name 필드에 브랜치 또는 태그 이름이 표시됩니다. pull 요청 이벤트의 경우 webhook 페이로드의 source 객체에 있는 branch 객체의 name 필드에 브랜치 이름이 표시됩니다.

BASE_REF

베이스 참조가 정규식 패턴과 일치하면 webhook 이벤트가 빌드를 트리거합니다. BASE_REF 필터는 풀 요청 이벤트에서만 작동합니다(예: refs/heads/branch-name). BASE_REF 필터가 브랜치의 Git 참조 이름을 평가합니다. Webhook 페이로드의 destination 객체에 있는 branch 객체의 name 필드에 브랜치 이름이 표시됩니다.

FILE_PATH

변경된 파일의 경로가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

COMMIT_MESSAGE

헤드 커밋 메시지가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

> **Note**
>
> 웹후크 페이로드는 리포지토리의 웹후크 설정에서 찾을 수 있습니다. GitLab

주제

- GitLab 웹후크 이벤트 필터링 (콘솔)
- GitLab 웹후크 이벤트 (SDK) 필터링
- GitLab 웹후크 이벤트 필터링 ()AWS CloudFormation

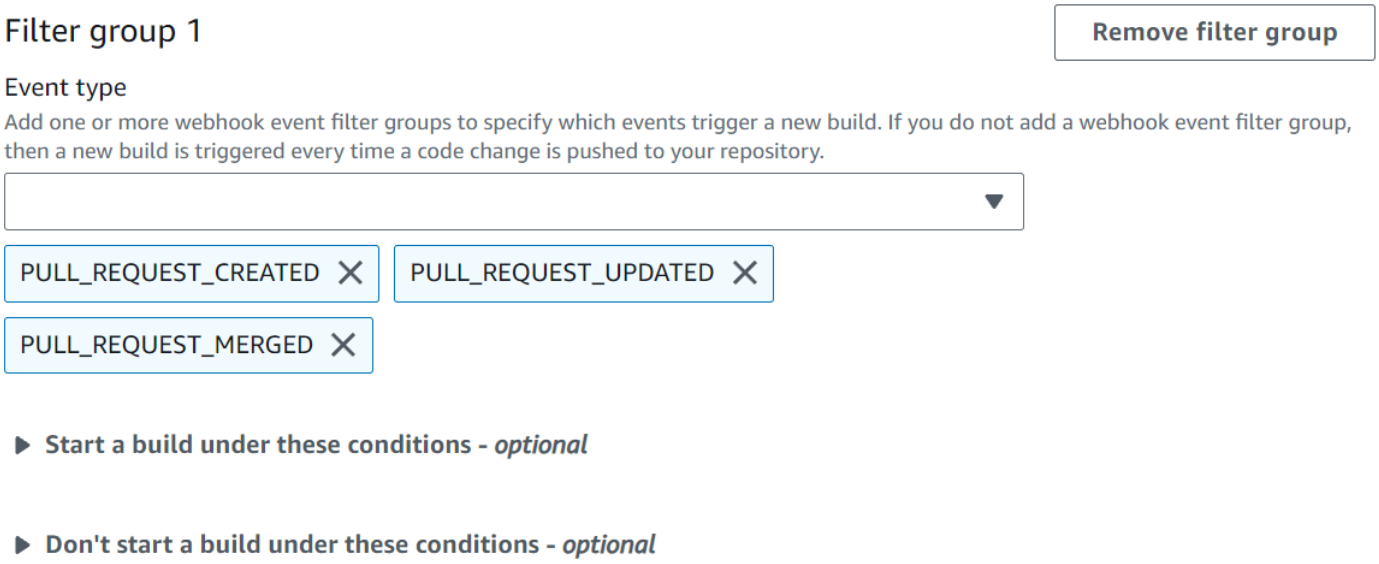
GitLab 웹후크 이벤트 필터링 (콘솔)

를 사용하여 웹후크 AWS Management Console 이벤트를 필터링하려면:

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. Add filter group(필터 그룹 추가)를 선택하여 다른 필터 그룹을 추가합니다.

자세한 내용은 AWS CodeBuild API WebhookFilter참조의 빌드 프로젝트 만들기(콘솔) 및 를 참조하십시오.

이 예제에서는 Webhook 필터 그룹이 pull 요청에 대해서만 빌드를 트리거합니다.



두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/branch1!`와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/branch1$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

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.

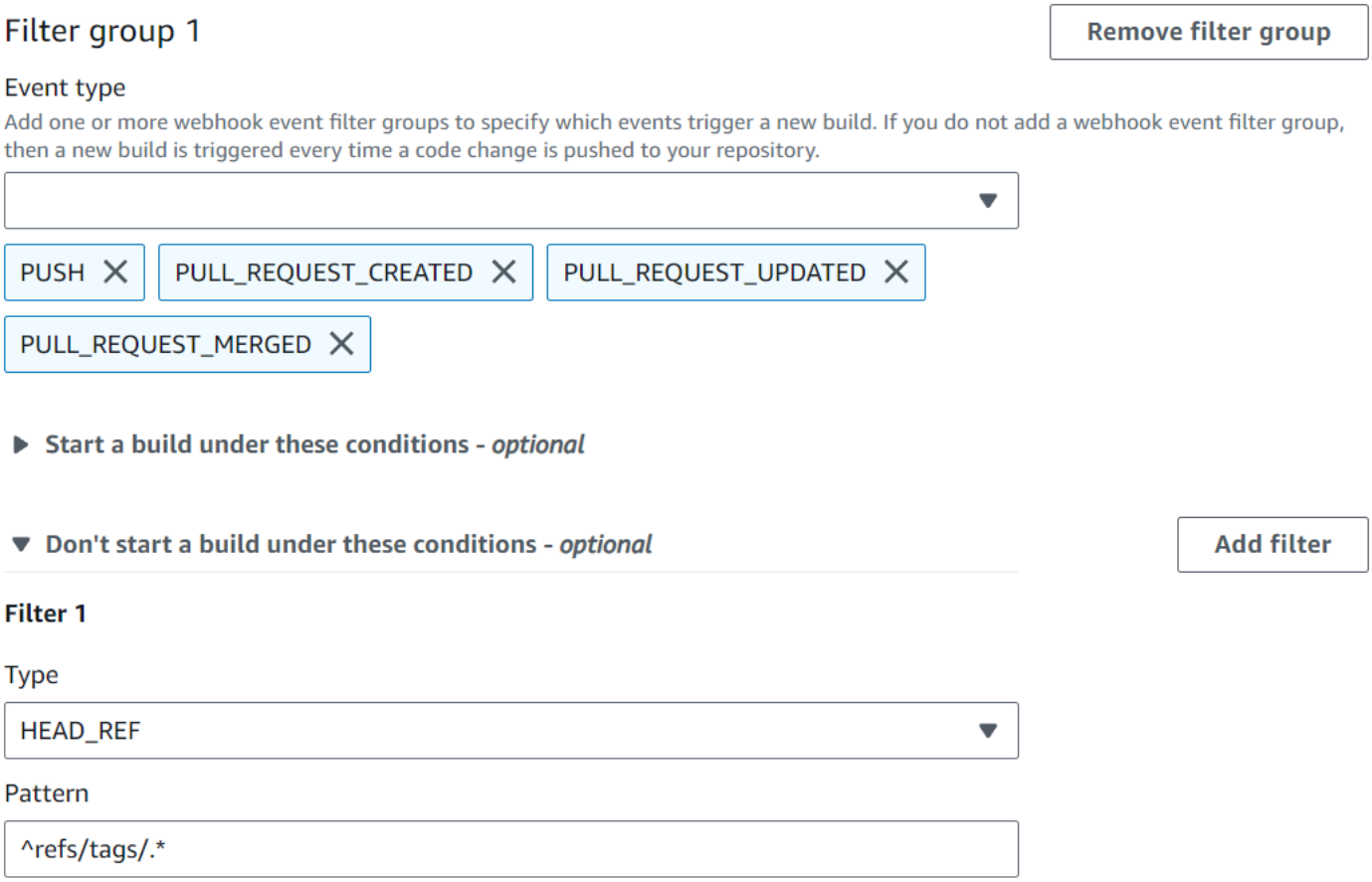
PUSH ✕

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

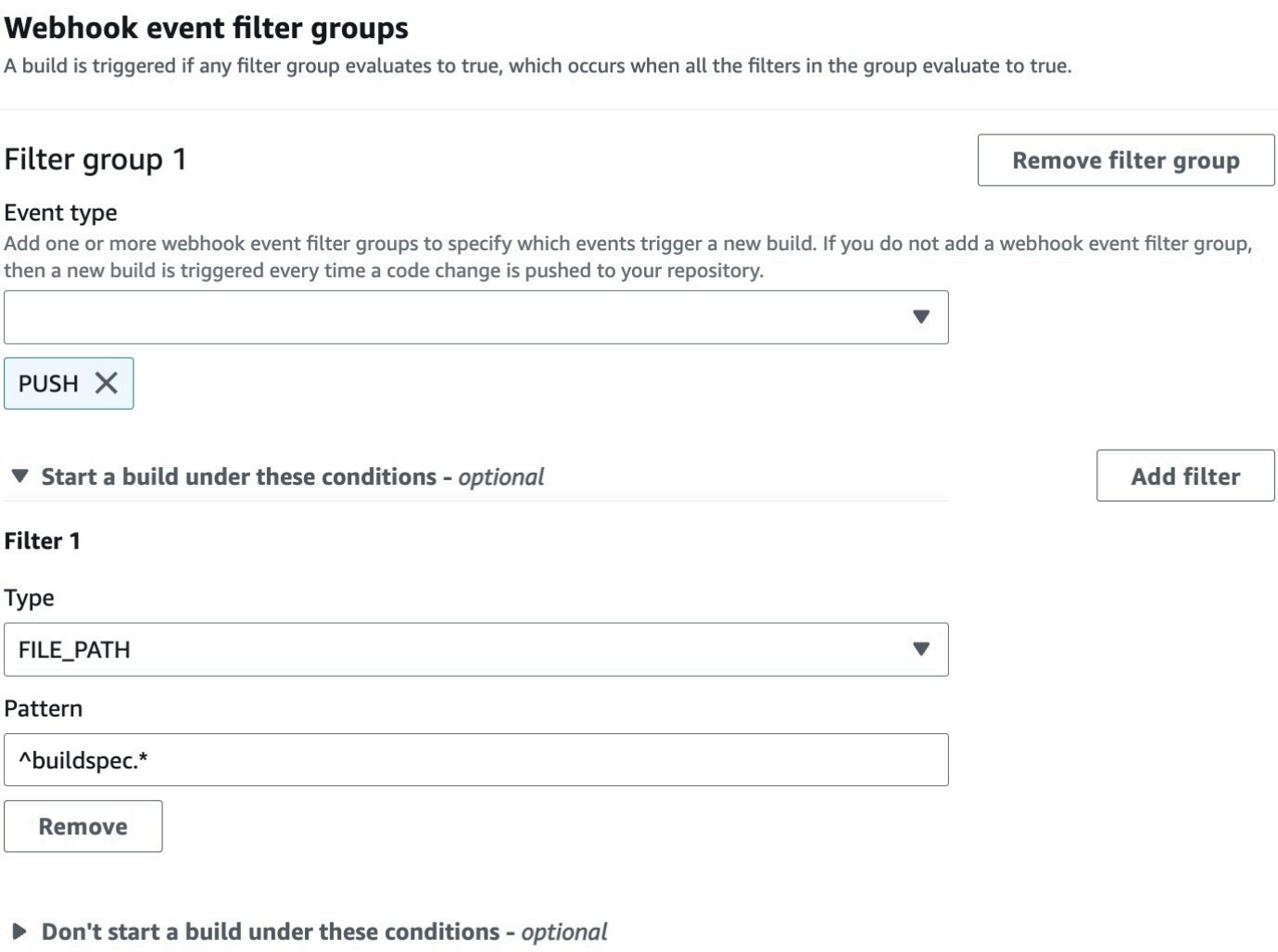
Add filter

Filter 1

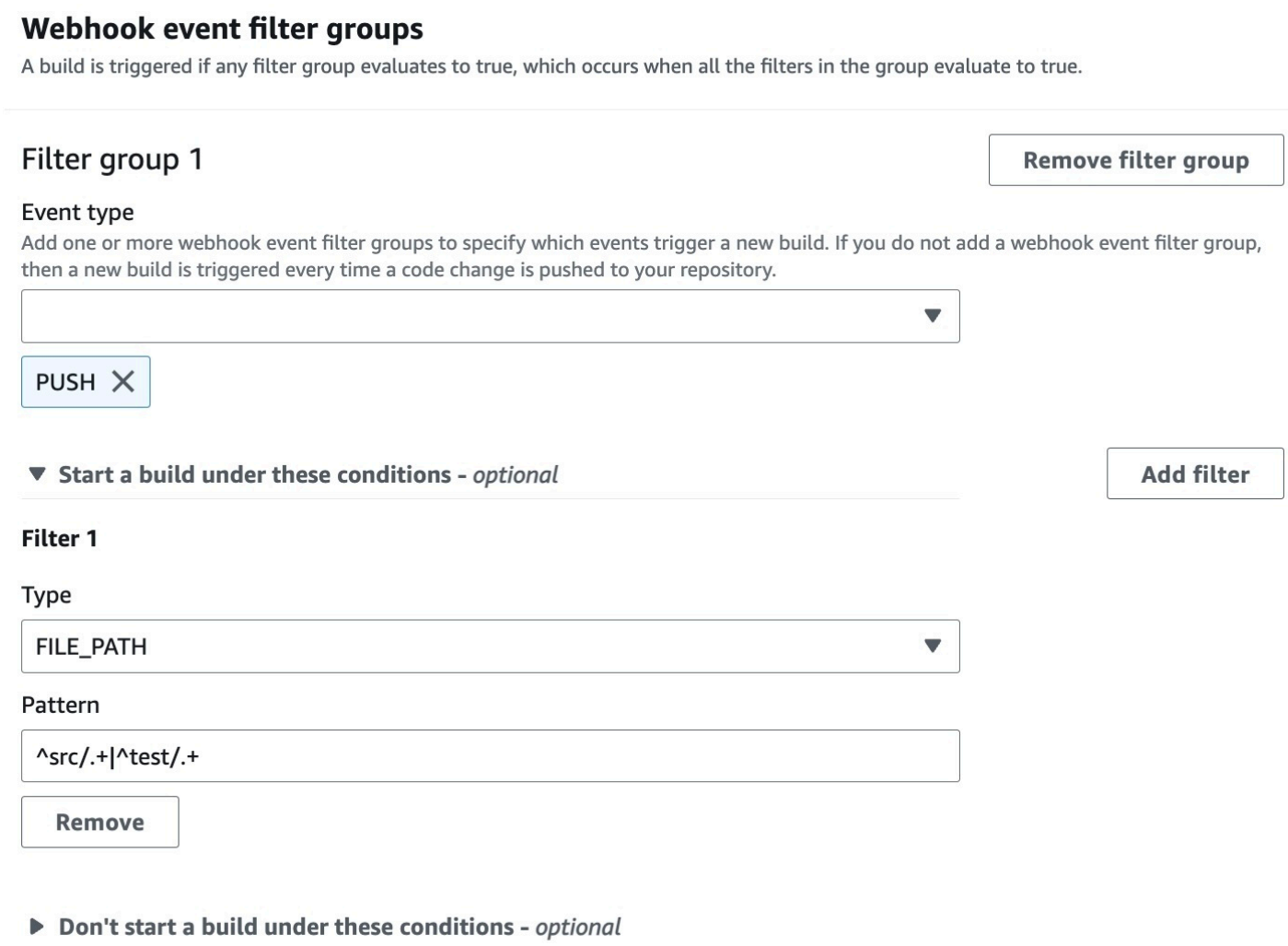
이 예제에서는 Webhook 필터 그룹이 태그 이벤트를 제외한 모든 요청에 대해 빌드를 트리거합니다.



이 예제에서는 Webhook 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드를 트리거합니다.

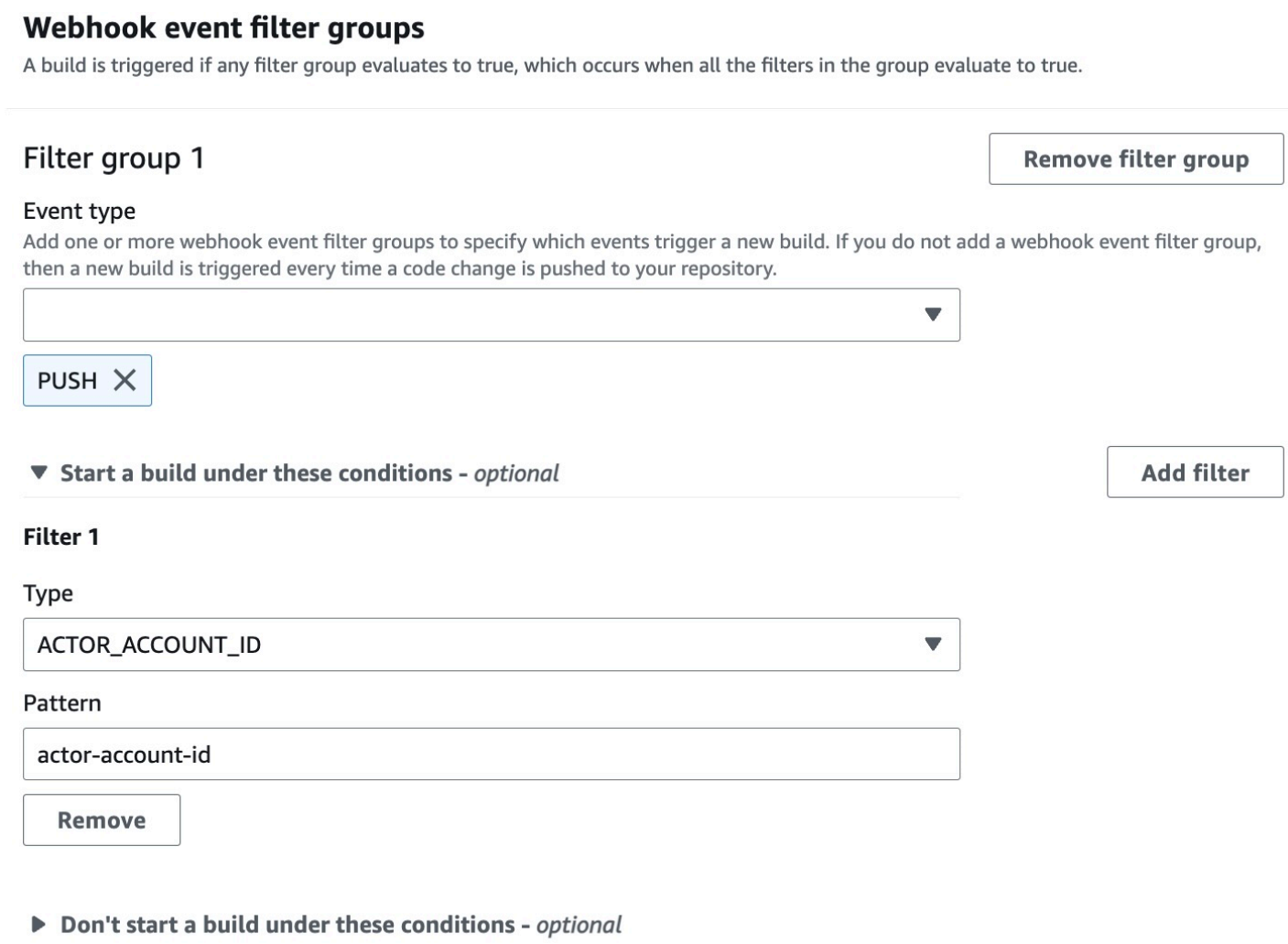


이 예제에서 Webhook 필터 그룹은 파일이 `src` 또는 `test` 폴더에서 변경된 경우에만 빌드를 트리거합니다.

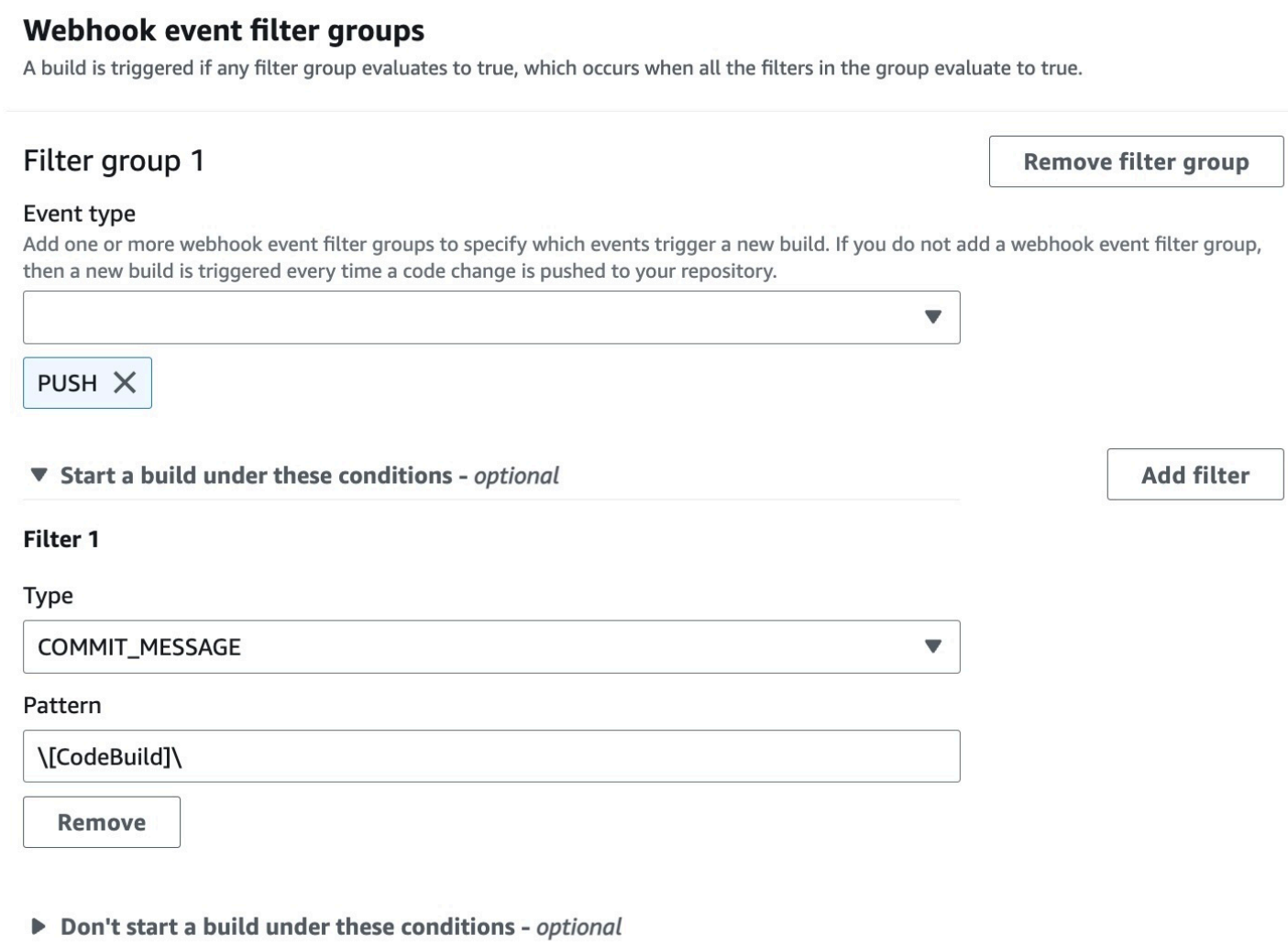


이 예제에서 웹후크 필터 그룹은 정규 표현식과 일치하는 계정 ID가 없는 GitLab 사용자가 변경한 경우에만 빌드를 트리거합니다. `actor-account-id`

> **Note**
>
> GitLab 계정 ID를 찾는 방법에 대한 자세한 내용은 https://api.github.com/users/ `### ### ## ####. ### ### ### ### #####`. GitLab



이 예제에서 webhook 필터 그룹은 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때 푸시 이벤트에 대한 빌드를 트리거합니다.



GitLab 웹후크 이벤트 (SDK) 필터링

AWS CodeBuild SDK를 사용하여 웹후크 이벤트를 필터링하려면 또는 API 메서드의 요청 구문에 있는 `filterGroups` 필드를 사용하십시오. `CreateWebhook` `UpdateWebhook` 자세한 내용은 CodeBuild API WebhookFilter참조를 참조하십시오.

pull 요청에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 요청 구문에 다음을 삽입합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
    }
```

```
  ]
]
```

지정된 브랜치에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 `pattern` 파라미터를 사용하여 브랜치 이름을 필터링하는 정규식을 지정합니다. 두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/myBranch$`와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/myBranch$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

```
"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$"
    }
  ]
]
```

`excludeMatchedPattern` 파라미터를 사용하여 빌드를 트리거하지 않을 이벤트를 지정할 수 있습니다. 이 예제에서는 태그 이벤트를 제외한 모든 요청에 대해 빌드가 트리거됩니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

계정 ID를 `actor-account-id` 가진 GitLab 사용자가 변경한 경우에만 빌드를 트리거하는 필터를 만들 수 있습니다.

> **Note**
>
> GitLab 계정 ID를 찾는 방법에 대한 자세한 내용은 https://api.github.com/users/ *### ###* 참조하십시오. 여기서 *### ### ### GitLab #####*.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
]
```

pattern 인수의 정규식과 일치하는 이름을 갖는 파일이 변경되는 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서는 필터 그룹이 정규식 ^buildspec.*와 일치하는 이름을 갖는 파일이 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

이 예제에서 필터 그룹은 파일이 src 또는 test 폴더에서 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

헤드 커밋 메시지가 패턴 인수의 정규식과 일치할 때만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서 필터 그룹은 푸시 이벤트의 헤드 커밋 메시지가 정규식 \[CodeBuild\]와 일치할 때만 빌드가 트리거되도록 지정합니다.

```
  "filterGroups": [
    [
      {
        "type": "EVENT",
```

```
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
      }
    ]
  ]
```

GitLab 웹후크 이벤트 필터링 ()AWS CloudFormation

AWS CloudFormation 템플릿을 사용하여 웹후크 이벤트를 필터링하려면 AWS CodeBuild 프로젝트의 `FilterGroups` 속성을 사용하십시오. 다음 AWS CloudFormation 템플릿의 YAML 형식 부분은 두 개의 필터 그룹을 생성합니다. 이들은 하나 또는 둘 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 없는 GitLab 사용자가 정규 표현식과 `^refs/heads/main$` 일치하는 Git 참조 이름을 가진 브랜치에서 pull 요청을 만들거나 업데이트하도록 지정합니다. 12345
- 두 번째 필터 그룹은 정규식 `^refs/heads/.*`와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 `\[CodeBuild\]`와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.

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

AWS CodeBuild에서 빌드 프로젝트 설정 변경

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 빌드 프로젝트의 설정을 변경할 수 있습니다.

빌드 프로젝트에 테스트 보고를 추가하는 경우 IAM 역할에 테스트 보고서 권한 작업에서 설명하는 사용 권한이 있는지 확인합니다.

주제

- 빌드 프로젝트 설정 변경(콘솔)
- 빌드 프로젝트 설정 변경(AWS CLI)
- 빌드 프로젝트 설정 변경(AWS SDK)

빌드 프로젝트 설정 변경(콘솔)

빌드 프로젝트의 설정을 변경하려면 다음 절차에 수행하세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 다음 중 하나를 수행하십시오.
   - 변경하려는 빌드 프로젝트의 링크를 선택한 다음 Edit project(프로젝트 편집)를 선택합니다.
   - 변경하려는 빌드 프로젝트 옆에 있는 라디오 버튼을 선택하고 세부 정보 보기를 선택한 후 빌드 세부 정보를 선택합니다.

다음 섹션을 수정할 수 있습니다.

Sections

- 프로젝트 구성
- 소스
- 환경
- Buildspec
- 배치 구성
- 아티팩트
- 로그

프로젝트 구성

프로젝트 구성 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

설명

선택적으로 빌드 프로젝트에 대한 설명을 입력하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.

빌드 배지

프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다. 자세한 정보는 빌드 배지 샘플을 참조하세요.

> **Note**
>
> 소스 공급자가 Amazon S3인 경우 빌드 배지가 적용되지 않습니다.

동시 빌드 제한 활성화

이 프로젝트의 동시 빌드 수를 제한하려면 다음 단계를 수행합니다.

1. 이 프로젝트에서 시작할 수 있는 동시 빌드 수 제한을 선택합니다.

2. 동시 빌드 제한에 이 프로젝트에 허용되는 최대 동시 빌드 수를 입력합니다. 이 제한은 계정에 설정된 동시 빌드 제한보다 클 수 없습니다. 계정 제한보다 큰 숫자를 입력하려고 하면 오류 메시지가 표시됩니다.

현재 빌드 수가 이 한도 이하인 경우에만 새 빌드가 시작됩니다. 현재 빌드 수가 이 한도에 도달하면 새 빌드가 제한되고 실행되지 않습니다.

퍼블릭 빌드 액세스 활성화

AWS 계정에 액세스할 수 없는 사용자를 포함하여 모든 사람이 프로젝트의 빌드 결과를 사용할 수 있게 하려면 공개 빌드 액세스 활성화를 선택하고 빌드 결과를 공개할 것인지 확인합니다. 퍼블릭 빌드 프로젝트에는 다음과 같은 속성이 사용됩니다.

퍼블릭 빌드 서비스 역할

새 서비스 역할을 직접 CodeBuild 만들려면 새 서비스 역할을 선택하고, 기존 서비스 역할을 사용하려면 기존 서비스 역할을 선택합니다.

공개 빌드 서비스 역할을 사용하면 CodeBuild CloudWatch 로그를 읽고 프로젝트 빌드의 Amazon S3 아티팩트를 다운로드할 수 있습니다. 이 역할은 프로젝트의 빌드 로그와 아티팩트를 퍼블릭으로 지정하는 데 필요합니다.

서비스 역할

새 서비스 역할 또는 기존 서비스 역할의 이름을 입력합니다.

프로젝트의 빌드 결과를 프라이빗으로 설정하려면 퍼블릭 빌드 액세스 활성화를 선택 취소합니다.

자세한 정보는 AWS CodeBuild의 퍼블릭 빌드 프로젝트을 참조하세요.

> **Warning**
>
> 프로젝트의 빌드 결과를 퍼블릭으로 유지할 때는 다음에 유의해야 합니다.
>
> - 프로젝트가 프라이빗일 때 실행된 빌드를 포함하여 프로젝트의 모든 빌드 결과, 로그, 아티팩트는 누구나 이용할 수 있습니다.
> - 모든 빌드 로그와 아티팩트는 누구나 이용할 수 있습니다. 환경 변수, 소스 코드 및 기타 중요한 정보가 빌드 로그와 아티팩트에 출력되었을 수 있습니다. 빌드 로그에 출력되는 정보에 주의해야 합니다. 몇 가지 모범 사례는 다음과 같습니다.
>   - 민감한 값, 특히 AWS 액세스 키 ID와 보안 액세스 키를 환경 변수에 저장하지 마십시오. Amazon EC2 Systems Manager 파라미터 스토어를 AWS Secrets Manager 사용하거나 민감한 값을 저장하는 것이 좋습니다.

- webhook를 최대한 안전하게 보호하려면 webhook 사용 모범 사례에 따라 빌드를 트리거할 수 있는 엔터티를 제한하고, buildspec을 프로젝트 자체에 저장하지 마세요.
- 악의적인 사용자는 퍼블릭 빌드를 사용하여 악성 아티팩트를 배포할 수 있습니다. 프로젝트 관리자는 모든 풀 요청을 검토하여 풀 요청이 합법적인 변경인지 확인하는 것이 좋습니다. 또한 체크섬으로 모든 아티팩트의 유효성을 검사하여 올바른 아티팩트가 다운로드되고 있는지 확인하는 것이 좋습니다.

추가 정보

태그에는 지원 AWS 서비스에서 사용할 태그의 이름과 값을 입력합니다. [Add row]를 사용하여 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다.

소스

소스 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

소스 공급자

소스 코드 공급자 유형을 선택합니다. 다음 목록을 사용하여 소스 공급자에 알맞은 유형을 선택합니다.

> **Note**
>
> CodeBuild Bitbucket 서버는 지원하지 않습니다.

Amazon S3

버킷

소스 코드가 포함된 입력 버킷의 이름을 선택합니다.

S3 객체 키 또는 S3 폴더

소스 코드가 포함된 ZIP 파일의 이름 또는 폴더 경로를 입력합니다. S3 버킷의 모든 항목을 다운로드하려면 슬래시(/)를 입력합니다.

소스 버전

입력 파일의 빌드를 나타내는 객체의 버전 ID를 입력합니다. 자세한 내용은 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild 섹션을 참조하세요.

CodeCommit

리포지토리

사용할 리포지토리를 선택합니다.

참조 유형

분기, Git 태그 또는 커밋 ID를 선택하여 소스 코드 버전을 지정합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d 또는 5392f7과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

이력이 지정된 커밋 수로 잘린 부분 복제를 생성하려면 선택합니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

Bitbucket

리포지토리

OAuth를 사용하여 연결 또는 Bitbucket 앱 암호를 사용하여 연결을 선택하고 지침에 따라 Bitbucket에 연결(또는 다시 연결)합니다.

퍼블릭 리포지토리 또는 계정의 리포지토리를 선택합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild단원을 참조하세요.

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d 또는 5392f7과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

상태 컨텍스트의 경우 Bitbucket 커밋 상태의 name 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

대상 URL에 Bitbucket 커밋 상태의 url 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 소스 코드를 빌드하려면 기본 소스 웹후크 이벤트에서 코드 변경 사항이 이 리포지토리에 푸시될 때마다 재빌드를 선택합니다. CodeBuild webhook 및 필터 그룹에 대한 자세한 내용은 Bitbucket Webhook 이벤트 섹션을 참조하세요.

GitHub

리포지토리

OAuth를 사용한 연결 또는 GitHub 개인용 액세스 토큰으로 연결을 선택하고 지침에 따라 연결(또는 재연결) GitHub 하고 액세스를 승인합니다. AWS CodeBuild

퍼블릭 리포지토리 또는 계정의 리포지토리를 선택합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild단원을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

상태 컨텍스트의 경우 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. GitHub 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

타겟 URL의 경우 GitHub 커밋 상태에 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 소스 코드를 빌드하려면 기본 소스 웹후크 이벤트에서 코드 변경 사항이 이 리포지토리에 푸시될 때마다 재빌드를 선택합니다. CodeBuild webhook 및 필터 그룹에 대한 자세한 내용은 GitHub 웹후크 이벤트 섹션을 참조하세요.

GitHub Enterprise Server

GitHub 엔터프라이즈 개인용 액세스 토큰

GitHub 엔터프라이즈 서버 샘플 섹션에서 개인용 액세스 토큰을 클립보드에 복사하는 방법을 참조하세요. 토큰을 텍스트 필드에 붙여넣고, 토큰 저장을 선택합니다.

> **Note**
>
> 개인 액세스 토큰은 한 번만 입력하고 저장하면 됩니다. CodeBuild 향후 모든 프로젝트에서 이 토큰을 사용합니다.

소스 버전

풀 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

상태 컨텍스트의 경우 GitHub 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

타겟 URL의 경우 GitHub 커밋 상태에 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 가이드의 커밋 상태 만들기를 참조하십시오.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

비보안 SSL

GitHub Enterprise 프로젝트 리포지토리에 연결하는 동안 SSL 경고를 무시하려면 비보안 SSL 활성화를 선택합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 소스 코드를 빌드하려면 기본 소스 웹후크 이벤트에서 코드 변경 사항이 이 리포지토리에 푸시될 때마다 재빌드를 선택합니다. CodeBuild webhook 및 필터 그룹에 대한 자세한 내용은 GitHub 웹후크 이벤트 섹션을 참조하세요.

GitLab

Connection

를 사용하여 GitLab 계정을 연결하고 AWS CodeConnections, 연결을 사용하여 타사 리포지토리를 빌드 프로젝트의 소스로 연결합니다.

기본 연결 또는 사용자 지정 연결을 선택합니다.

기본 연결은 모든 프로젝트에 기본 GitLab 연결을 적용합니다. 사용자 지정 연결은 계정의 기본 설정을 재정의하는 사용자 지정 GitLab 연결을 적용합니다.

기본 연결

계정에 연결된 기본 연결의 이름.

공급자와의 연결을 아직 만들지 않은 경우 지침을 GitLab(콘솔) 에 대한 연결 만들기 참조하십시오.

사용자 지정 연결

사용하려는 사용자 지정 연결의 이름을 선택합니다.

공급자에 GitLab(콘솔) 에 대한 연결 만들기 대한 연결을 아직 생성하지 않은 경우 지침을 참조하십시오.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

풀 리퀘스트 ID, 브랜치, 커밋 ID, 태그 또는 레퍼런스와 커밋 ID를 입력합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

GitLab Self Managed

Connection

를 사용하여 GitLab 계정을 연결하고 AWS CodeConnections, 연결을 사용하여 타사 리포지토리를 빌드 프로젝트의 소스로 연결합니다.

기본 연결 또는 사용자 지정 연결을 선택합니다.

기본 연결은 모든 프로젝트에 기본 GitLab 자체 관리형 연결을 적용합니다. 사용자 지정 연결은 계정의 기본 설정을 재정의하는 사용자 지정 GitLab 자체 관리형 연결을 적용합니다.

기본 연결

계정에 연결된 기본 연결의 이름.

공급자에 대한 연결을 아직 생성하지 않은 경우 개발자 도구 콘솔 사용 설명서에서 GitLab 자체 관리형 연결 만들기를 참조하여 지침을 확인하십시오.

사용자 지정 연결

사용하려는 사용자 지정 연결의 이름을 선택합니다.

공급자에 대한 연결을 아직 생성하지 않은 경우 개발자 도구 콘솔 사용 설명서에서 GitLab 자체 관리형 연결 만들기를 참조하여 지침을 확인하십시오.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

풀 리퀘스트 ID, 브랜치, 커밋 ID, 태그 또는 레퍼런스와 커밋 ID를 입력합니다. 자세한 정보는 소스 버전 샘플은 다음과 같습니다. AWS CodeBuild을 참조하세요.

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d 또는 5392f7과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 정보는 소스 공급자 액세스을 참조하세요.

환경

환경 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

프로비저닝 모델

프로비저닝 모델을 변경하려면 프로비저닝 모델 변경을 선택하고 다음 중 하나를 수행하십시오.

- 에서 관리하는 온디맨드 플릿을 사용하려면 온디맨드를 선택합니다. AWS CodeBuild 온디맨드 플릿을 사용하면 빌드에 필요한 컴퓨팅 CodeBuild 기능을 제공합니다. 빌드가 완료되면 머신이 파괴됩니다. 온디맨드 플릿은 완전 관리형이며, 수요 급증을 처리할 수 있는 자동 규모 조정 기능이 포함되어 있습니다.

- 에서 관리하는 예약 용량 플릿을 사용하려면 예약 용량을 선택한 다음 플릿 이름을 선택합니다. AWS CodeBuild 예약 용량 플릿을 사용하면 빌드 환경을 위한 전용 인스턴스 세트를 구성할 수 있습니다. 이러한 머신은 유휴 상태로 유지되므로 빌드 또는 테스트를 즉시 처리하고 빌드 기간을 단축할 수 있습니다. 예약 용량 플릿을 사용하면 머신이 상시 가동되므로 프로비저닝하는 한 계속해서 비용이 발생합니다.

자세한 내용은 에서 예약된 용량으로 작업하기 AWS CodeBuild을 참조하세요.

환경 이미지

빌드 이미지를 변경하려면 이미지 재정의를 선택하고 다음 중 하나를 수행합니다.

- 관리되는 AWS CodeBuild Docker 이미지를 사용하려면 관리 이미지를 선택한 다음 운영 체제, 런타임, 이미지 및 이미지 버전 중에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 계정의 Docker 이미지를 선택하십시오. AWS
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

> **Note**
>
> CodeBuild 사용자 지정 Docker 이미지의 경우 를 재정의합니다. `ENTRYPOINT`

서비스 역할

다음 중 하나를 수행하십시오.

- 서비스 역할이 없는 경우 새 CodeBuild 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 만들 때 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

추가 구성

제한 시간

5분에서 8시간 사이의 값을 지정합니다. 이 값을 지정한 후 완료되지 않으면 빌드가 CodeBuild 중지됩니다. [hours] 및 [minutes]가 비어 있는 경우 기본값인 60분이 사용됩니다.

권한 있음

Docker 이미지를 빌드하거나 빌드에 높은 권한을 부여하려면 이 플래그 활성화를 선택합니다. 이 빌드 프로젝트를 사용하여 Docker 이미지를 빌드하려는 경우에만 가능합니다. 그렇지 않으면 Docker 데몬과 상호 작용을 시도하는 모든 연결된 빌드가 실패합니다. 또한 빌드가 상호 작용할 수 있도록 Docker 데몬을 시작해야 합니다. 이를 수행하는 한 가지 방법은 다음 빌드 명령을 실행하여 빌드 사양의 `install` 단계에서 Docker 데몬을 초기화하는 것입니다. Docker 지원에서 제공하는 CodeBuild 빌드 환경 이미지를 선택한 경우에는 이 명령을 실행하지 마세요.

> **Note**
>
> 기본적으로 Docker 데몬은 VPC가 아닌 빌드에는 활성화되어 있습니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 모드를 활성화하세요. 또한 Windows는 권한 모드를 지원하지 않습니다.

```
- 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

VPC로 CodeBuild 작업하려는 경우:

- VPC의 경우 사용할 VPC ID를 선택합니다. CodeBuild

- VPC 서브넷의 경우 사용하는 리소스가 포함된 서브넷을 선택합니다. CodeBuild
- VPC 보안 그룹의 경우 VPC의 리소스에 대한 액세스를 허용하는 데 CodeBuild 사용하는 보안 그룹을 선택합니다.

자세한 정보는 Amazon Virtual Private 클라우드와 AWS CodeBuild 함께 사용을 참조하세요.

컴퓨팅

사용 가능한 옵션 중 하나를 선택합니다.

환경 변수

사용할 빌드의 각 환경 변수에 대해 이름 및 값을 입력하고 유형을 선택합니다.

> **Note**
>
> CodeBuild 해당 AWS 지역의 환경 변수를 자동으로 설정합니다. buildspec.yml에 추가하지 않은 경우 다음 환경 변수를 설정해야 합니다.
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME
> - IMAGE_TAG

콘솔 및 AWS CLI 사용자는 환경 변수를 볼 수 있습니다. 환경 변수의 가시성에 대한 문제가 없다면 [Name] 및 [Value] 필드를 설정한 다음 [Type]을 [Plaintext]로 설정합니다.

AWS 액세스 키 ID, AWS 보안 액세스 키 또는 암호와 같은 민감한 값이 포함된 환경 변수를 Amazon EC2 Systems Manager 파라미터 스토어 AWS Secrets Manager또는 파라미터로 저장하는 것이 좋습니다.

Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 유형에서 파라미터를 선택합니다. [Name] 에는 CodeBuild 참조할 식별자를 입력합니다. 값에 Amazon EC2 Systems Manager Parameter Store에 저장되는 파라미터의 이름을 입력합니다. 예를 들어 `/CodeBuild/dockerLoginPassword`라는 이름의 파라미터를 사용하여 유형에서 파라미터를 선택합니다. 이름에 `LOGIN_PASSWORD`를 입력합니다. 값에 `/CodeBuild/dockerLoginPassword`를 입력합니다.

> **Important**
>
> Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 `/CodeBuild/`로 시작하는 파라미터 이름(예: `/CodeBuild/dockerLoginPassword`)으로 파라미터를

저장하는 것이 좋습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택하고 대화 상자에 표시되는 지시에 따릅니다. (이 대화 상자에서 KMS 키의 경우 계정 내 키의 AWS KMS ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는 이 키를 사용하여 저장 중에 파라미터 값을 암호화하고 검색 중에 이를 복호화합니다.) 콘솔을 사용하여 파라미터를 생성하는 경우, CodeBuild 콘솔은 파라미터가 저장되는 `/CodeBuild/` 대로 파라미터 이름을 시작합니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.

빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `ssm:GetParameters` 작업을 허용해야 합니다. 이전에 새 서비스 역할을 선택한 경우 빌드 프로젝트의 기본 서비스 역할에 이 작업을 CodeBuild 포함하세요. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.

빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 파라미터 이름으로 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 파라미터 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 파라미터 이름에만 액세스할 수 있기 때문입니다.

새 서비스 역할을 선택하면 Amazon EC2 Systems Manager Parameter Store에 있는 `/CodeBuild/` 네임스페이스의 모든 파라미터를 해독할 권한이 서비스 역할에 포함됩니다.

사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.

`CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.

여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.

- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

Secrets Manager를 사용하는 경우 유형으로 Secrets Manager로 선택합니다. 이름에는 CodeBuild 참조할 식별자를 입력합니다. 값에 *secret-id*:*json-key*:*version-stage*:*version-id* 패턴을 사용하여 reference-key를 입력합니다. 자세한 내용은 Secrets Manager reference-key in the buildspec file을 참조하세요.

> **Important**
>
> Secrets Manager를 사용하는 경우 이름이 /CodeBuild/로 시작하는 암호를 저장하는 것이 좋습니다(예: /CodeBuild/dockerLoginPassword). 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.
> 빌드 프로젝트가 Secrets Manager에 저장된 암호를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 secretsmanager:GetSecretValue 작업을 허용해야 합니다. 이전에 새 서비스 역할을 선택한 경우 빌드 프로젝트의 기본 서비스 역할에 이 작업을 CodeBuild 포함시키십시오. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
> 빌드 프로젝트가 /CodeBuild/로 시작되지 않는 보안 암호 이름으로 Secrets Manager에 저장된 암호를 참조하는 경우 새 서비스 역할을 선택하면 /CodeBuild/로 시작하지 않는 보안 암호 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 /CodeBuild/로 시작하는 암호 이름에만 액세스할 수 있기 때문입니다.
> 새 서비스 역할을 선택하면 에 있는 /CodeBuild/ 네임스페이스의 모든 암호를 해독할 권한이 서비스 역할에 포함됩니다.

Buildspec

Buildspec 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

빌드 사양

다음 중 하나를 수행하십시오.

- 소스 코드에 buildspec 파일이 있는 경우 Use a buildspec file(빌드 사양 파일 사용) 을 선택합니다. 기본적으로 소스 코드 루트 `buildspec.yml` 디렉터리에서 이름이 지정된 파일을 CodeBuild 찾습니다. buildspec 파일이 다른 이름이나 위치를 사용하는 경우 Buildspec 이름에 소스 루트의 경로를 입력합니다(예: `buildspec-two.yml` 또는 `configuration/buildspec.yml`). buildspec 파일이 S3 버킷에 있는 경우 해당 파일은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).
- 소스 코드에 buildspec 파일이 포함되어 있지 않거나, 소스 코드의 루트 디렉터리에 있는 `buildspec.yml` 파일의 `build` 단계에 지정된 것과 다른 빌드 명령 세트를 실행하려는 경우 빌드 명령 삽입을 선택합니다. 빌드 명령의 `build` 단계에서 실행하려는 명령을 입력합니다. 명령이 여러 개인 경우 각 명령을 `&&`로 구분합니다(예: `mvn test && mvn package`). 다른 구문에서 명령을 실행하려는 경우 또는 `build` 단계에 대해 특히 긴 명령 목록이 있는 경우에는 소스 코드 루트 디렉터리에 `buildspec.yml` 파일을 추가하고, 이 파일에 명령을 추가한 다음, 소스 코드 루트 디렉터리에서 buildspec.yml 사용을 선택합니다.

자세한 내용은 buildspec 참조을 참조하세요.

배치 구성

배치 구성 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다. 자세한 정보는 Batch 빌드 인 AWS CodeBuild을 참조하세요.

다음 속성을 수정할 수 있습니다.

배치 서비스 역할

배치 빌드에 대한 서비스 역할을 제공합니다.

다음 중 하나를 선택합니다.

- 배치 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 서비스 역할에 새 역할의 이름을 입력합니다.
- 배치 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 서비스 역할에서 서비스 역할을 선택합니다.

배치 빌드는 배치 구성에 새로운 보안 역할을 도입합니다. 일괄 처리의 일부로 빌드를 실행하려면 사용자 대신 `StartBuild`StopBuild,, `RetryBuild` 작업을 호출할 수 CodeBuild 있어야 하므로 이 새 역할이 필요합니다. 고객은 다음과 같은 두 가지 이유로 빌드에 사용하는 것과 동일한 역할이 아닌 새 역할을 사용해야 합니다.

- 빌드 역할 `StartBuild`, `StopBuild` 및 `RetryBuild` 권한을 부여하면 단일 빌드에서 buildspec을 통해 더 많은 빌드를 시작할 수 있습니다.
- CodeBuild 배치 빌드는 일괄 빌드에 사용할 수 있는 빌드 및 컴퓨팅 유형의 수를 제한하는 제한을 제공합니다. 빌드 역할에 이러한 권한이 있는 경우 빌드 자체가 이러한 제한을 우회할 수 있습니다.

배치에 허용되는 컴퓨팅 유형

배치에 허용되는 컴퓨팅 유형을 선택합니다. 해당하는 항목을 모두 선택합니다.

배치에 허용되는 최대 빌드 수

배치에 허용되는 최대 빌드 수를 입력합니다. 이 제한을 초과하는 배치는 실패합니다.

배치 제한 시간

배치 빌드가 완료되는 최대 시간을 입력합니다.

아티팩트 결합

배치의 모든 아티팩트를 단일 위치로 결합을 선택하면 배치의 모든 아티팩트가 단일 위치로 결합됩니다.

배치 보고서 모드

배치 빌드에 대해 원하는 빌드 상태 보고서 모드를 선택합니다.

> **Note**
>
> 이 필드는 프로젝트 소스가 Bitbucket 또는 GitHub Enterprise인 경우에만 사용할 수 있으며, 소스 아래에서 빌드 시작 및 종료를 선택하면 소스 공급자에게 빌드 상태를 보고할 수 있습니다. GitHub

빌드 집계

배치의 모든 빌드 상태를 단일 상태 보고서로 통합하려면 선택합니다.

개별 빌드

배치에 있는 모든 빌드의 빌드 상태를 별도로 보고하려면 선택합니다.

아티팩트

아티팩트 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

Type

다음 중 하나를 수행하십시오.

- 빌드 출력 결과물을 생성하지 않으려면 [No artifacts]를 선택합니다. 빌드 테스트만 실행하고 있는 경우 또는 Amazon ECR 리포지토리에 도커 이미지를 푸시하려는 경우에 이 작업을 원할 수 있습니다.
- S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
  - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. (ZIP 파일을 출력하고 ZIP 파일에 파일 확장명을 넣으려는 경우, ZIP 파일 이름 뒤에 이를 포함하십시오.)
  - buildspec 파일에 지정된 이름으로 콘솔에서 지정한 이름을 재정의하려는 경우 의미 체계 버전 관리 사용을 선택합니다. buildspec 파일의 이름은 빌드 시 계산되며 Shell 명령 언어를 사용합니다. 예를 들어 결과물 이름이 항상 고유하도록 날짜와 시간을 결과물 이름에 추가할 수 있습니다. 고유한 결과물 이름을 사용하면 결과물을 덮어쓰지 않을 수 있습니다. 자세한 정보는 buildspec 구문을 참조하세요.
  - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
  - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.
  - 빌드 아티팩트를 암호화하지 않으려면 Remove artifacts encryption(결과물 암호화 제거)을 선택합니다.

각각 원하는 보조 아티팩트 세트마다 다음과 같이 실행합니다.

1. Atrifact identifier(아티팩트 식별자)에서 128자 미만으로 영숫자와 밑줄만 포함된 값을 입력합니다.
2. Add artifact(아티팩트 추가)를 선택합니다.
3. 이전 단계에 따라 보조 결과물을 구성합니다.
4. Save artifact(아티팩트 저장)를 선택합니다.

추가 구성

암호화 키

다음 중 하나를 수행하십시오.

- 계정의 AWS 관리형 키 Amazon S3를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키를 비워 두십시오. 이 값이 기본값입니다.
- 고객 관리형 키를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키에 고객 관리형 키의 ARN을 입력합니다. `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* 형식을 사용합니다.

캐시 유형

Cache type(캐시 유형)에서 다음 중 하나를 선택합니다.

- 캐시를 사용하지 않으려면 [No cache]를 선택합니다.
- Amazon S3 캐시를 사용하려면 Amazon S3를 선택하고 다음을 수행합니다.
  - 버킷에서 캐시가 저장된 S3 버킷의 이름을 선택합니다.
  - (선택 사항) 캐시 경로 접두사에 Amazon S3 경로 접두사를 입력합니다. Cache path prefix(캐시 경로 접두사) 값은 디렉터리 이름과 비슷합니다. 따라서 캐시를 버킷의 동일한 디렉터리에 저장할 수 있습니다.

    > **Important**
    >
    > 경로 접두사 끝에 후행 슬래시(/)를 추가하지 마십시오.

- 로컬 캐시를 사용하려면 로컬을 선택한 다음 하나 이상의 로컬 캐시 모드를 선택해야 합니다.

  > **Note**
  >
  > Docker 계층 캐시 모드는 Linux에서만 사용할 수 있습니다. 이 모드를 선택할 경우 프로젝트를 권한이 있는 모드에서 실행해야 합니다.

캐시를 사용하면 빌드 환경의 재사용 가능한 특정 부분이 캐시에 저장되고 빌드 전반에서 사용되기 때문에 상당한 빌드 시간을 절약할 수 있습니다. buildspec 파일에 캐시를 지정하는 것에 대한 자세한 정보는 buildspec 구문 단원을 참조하십시오. 캐싱에 대한 자세한 정보는 AWS CodeBuild의 빌드 캐싱을 참조하십시오.

로그

로그 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

생성하려는 로그를 선택합니다. Amazon CloudWatch 로그, Amazon S3 로그 또는 둘 다를 생성할 수 있습니다.

CloudWatch

Amazon CloudWatch 로그 로그를 원하는 경우:

CloudWatch 로그

CloudWatch 로그를 선택합니다.

그룹 이름

Amazon CloudWatch Logs 로그 그룹 이름을 입력합니다.

스트림 이름

Amazon CloudWatch Logs 로그 스트림 이름을 입력합니다.

S3

Amazon S3 로그를 원할 경우:

S3 로그

S3 로그를 선택합니다.

버킷

로그에 대한 S3 버킷 이름을 선택합니다.

경로 접두사

로그의 접두사를 입력합니다.

S3 로그 암호화 비활성화

S3 로그를 암호화하지 않으려면 선택합니다.

빌드 프로젝트 설정 변경(AWS CLI)

AWS CLI와 AWS CodeBuild를 함께 사용하는 방법에 대한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI를 사용하여 CodeBuild 프로젝트를 업데이트하려면 업데이트된 속성으로 JSON 파일을 만들고 이 파일을 update-project 명령에 전달합니다. 업데이트 파일에 포함되지 않은 속성은 변경되지 않습니다.

업데이트 JSON 파일에는 name 속성과 수정된 속성만 필요합니다. name 속성은 수정할 프로젝트를 식별합니다. 수정된 구조의 경우 해당 구조의 필수 파라미터도 포함되어야 합니다. 예를 들어, 프로젝트에 대한 환경을 수정하려면 environment/type 및 environment/computeType 속성이 필요합니다. 다음은 환경 이미지를 업데이트하는 예입니다.

```
{
  "name": "<project-name>",
  "environment": {
    "type": "LINUX_CONTAINER",
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:4.0"
  }
}
```

프로젝트의 현재 속성 값을 가져와야 하는 경우 batch-get-projects 명령을 사용하여 수정하려는 프로젝트의 현재 속성을 가져오고 결과를 파일에 기록합니다.

```
aws codebuild batch-get-projects --names "<project-name>" > project-info.json
```

project-info.json 파일에는 프로젝트 배열이 포함되어 있으므로 프로젝트를 업데이트하는 데 직접 사용할 수는 없습니다. 하지만 *project-info.json* 파일에서 수정하려는 속성을 복사한 다음, 수정하려는 속성의 기준으로 업데이트 파일에 붙여넣을 수 있습니다. 자세한 내용은 빌드 프로젝트 세부 정보 보기(AWS CLI) 섹션을 참조하세요.

빌드 프로젝트 생성(AWS CLI)에 설명된 대로 업데이트 JSON 파일을 수정하고 결과를 저장합니다. 업데이트 JSON 파일의 수정이 끝나면 update-project 명령을 실행하여 업데이트 JSON 파일을 전달합니다.

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

성공하면 업데이트된 프로젝트 JSON이 출력에 나타납니다. 필수 파라미터가 누락된 경우 누락된 파라미터를 식별하는 오류 메시지가 출력에 표시됩니다. 예를 들어, environment/type 파라미터가 누락된 경우 표시되는 오류 메시지는 다음과 같습니다.

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

```
Parameter validation failed:
Missing required parameter in environment: "type"
```

빌드 프로젝트 설정 변경(AWS SDK)

AWS CodeBuild를 AWS SDK와 함께 사용하는 방법에 대한 자세한 정보는 AWS SDK 및 도구 참조 단원을 참조하십시오.

AWS CodeBuild에서 빌드 프로젝트 삭제

CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서 빌드 프로젝트를 삭제할 수 있습니다. 프로젝트를 삭제하면 해당 빌드가 삭제되지 않습니다.

> **Warning**
>
> 빌드 및 리소스 정책이 있는 프로젝트는 삭제할 수 없습니다. 리소스 정책 및 빌드가 있는 프로젝트를 삭제하려면 먼저 리소스 정책을 제거하고 빌드를 삭제합니다.

주제

- 빌드 프로젝트 삭제(콘솔)
- 빌드 프로젝트 삭제(AWS CLI)
- 빌드 프로젝트 삭제(AWS SDK)

빌드 프로젝트 삭제(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다.
3. 다음 중 하나를 수행하세요.
   - 삭제하려는 빌드 프로젝트 옆에 있는 라디오 버튼을 선택한 후 삭제를 선택합니다.
   - 삭제하려는 빌드 프로젝트의 링크를 선택한 다음, [Delete]를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 10개의 빌드 프로젝트가 표시됩니다. 더 많은 빌드 프로젝트를 보려면 Projects per page(페이지당 프로젝트)에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택하여 프로젝트를 봅니다.

빌드 프로젝트 삭제(AWS CLI)

1. `delete-project` 명령을 실행합니다.

```
aws codebuild delete-project --name name
```

   다음과 같이 자리 표시자를 바꿉니다.

   - *name*: 필수 문자열입니다. 삭제할 빌드 프로젝트의 이름입니다. 사용 가능한 빌드 프로젝트 목록을 보려면 `list-projects` 명령을 실행합니다. 자세한 내용은 빌드 프로젝트 이름 목록 보기(AWS CLI) 섹션을 참조하세요.
2. 이 명령이 제대로 실행되면 출력에 데이터나 오류가 표시되지 않습니다.

AWS CLI와 AWS CodeBuild를 함께 사용하는 방법에 대한 자세한 정보는 명령줄 참조 단원을 참조하십시오.

빌드 프로젝트 삭제(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

공유 프로젝트 작업

프로젝트 공유를 통해 프로젝트 소유자는 다른 AWS 계정 또는 사용자와 AWS CodeBuild 프로젝트를 공유할 수 있습니다. 이 모델에서는 프로젝트를 소유한 계정(소유자)이 다른 계정(소비자)과 프로젝트를 공유합니다. 소비자는 프로젝트를 편집하거나 실행할 수 없습니다.

목차

프로젝트 공유를 위한 전제 조건

프로젝트를 공유하려면 AWS 계정이 해당 프로젝트를 소유해야 합니다. 사용자와 공유된 프로젝트를 공유할 수 없습니다.

사용자와 공유된 공유 프로젝트에 액세스하기 위한 전제 조건

공유 프로젝트에 액세스하려면 소비자의 IAM 역할에 `BatchGetProjects` 권한이 필요합니다. 다음 정책을 해당 IAM 역할에 연결할 수 있습니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetProjects"
    ]
}
```

자세한 내용은 ID 기반 정책 사용: AWS CodeBuild 섹션을 참조하세요.

관련 서비스

프로젝트를 공유하면 AWS Organizations를 통해 또는 AWS 계정과 AWS 리소스를 공유할 수 있게 해 주는 서비스인 AWS Resource Access Manager(AWS RAM)와 통합됩니다. AWS RAM에서는 리소스와 리소스를 공유할 소비자를 지정하는 리소스 공유를 생성하여 리소스를 공유합니다. 소비자는 개인 AWS 계정, AWS Organizations의 조직 단위 또는 AWS Organizations의 전체 조직일 수 있습니다.

자세한 정보는 AWS RAM 사용 설명서를 참조하세요.

프로젝트 공유

소비자는 AWS CLI 및 AWS CodeBuild 콘솔 둘 다를 사용해서 공유한 프로젝트와 빌드를 볼 수 있습니다. 소비자는 프로젝트를 편집하거나 실행할 수 없습니다.

기존 리소스 공유에 프로젝트를 추가하거나 AWS RAM콘솔에서 프로젝트를 만들 수 있습니다.

> **Note**
>
> 리소스 공유에 추가된 빌드가 있는 프로젝트는 삭제할 수 없습니다.

프로젝트를 조직 단위 또는 전체 조직과 공유하려면 AWS Organizations와의 공유를 활성화해야 합니다. 자세한 내용은 AWS RAM 사용 설명서에서 AWS Organizations를 사용하여 공유 사용을 참조하세요.

AWS CodeBuild 콘솔, AWS RAM 콘솔 또는 AWS CLI를 사용하여 소유한 프로젝트를 공유할 수 있습니다.

소유한 프로젝트를 공유하려면(CodeBuild 콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다.

   > **Note**
   >
   > 기본적으로 가장 최근의 빌드 프로젝트 10개만 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

3. 공유할 프로젝트를 선택한 다음 공유를 선택합니다. 자세한 내용은 AWS RAM 사용 설명서의 리소스 공유 생성을 참조하세요.

소유한 프로젝트를 공유하려면(AWS RAM콘솔)

AWS RAM 사용 설명서에서 리소스 공유 생성을 참조하세요.

소유한 프로젝트를 공유하려면(AWS RAM 명령)

create-resource-share 명령을 사용합니다.

소유한 프로젝트를 공유하려면(CodeBuild 명령)

put-resource-policy 명령을 사용합니다.

1. 이름이 `policy.json`인 파일을 만들고 다음으로 복사합니다.

```
{
  "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. 프로젝트 ARN 및 식별자로 `policy.json`을 업데이트하여 공유합니다. 다음 예제에서는 123456789012로 식별된 AWS 계정의 루트 사용자에게 읽기 전용 액세스 권한을 부여합니다.

```
{
  "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. put-resource-policy 명령을 실행합니다.

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. AWS RAM 리소스 공유 ARN을 가져옵니다.

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

이렇게 하면 다음과 비슷한 응답이 반환됩니다.

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

응답에서 다음 단계에서 사용할 *<resource-share-arn>* 값을 복사합니다.

5. AWS RAMpromote-resource-share-created-from-policy 명령을 실행합니다.

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-
share-arn>
```

공유 프로젝트의 공유 해제

빌드를 포함하여 공유가 해제된 프로젝트는 소유자만 액세스할 수 있습니다. 프로젝트 공유를 해제하면 이전에 공유한 AWS 계정이나 사용자가 프로젝트 또는 빌드에 액세스할 수 없습니다.

소유하고 있는 공유 프로젝트의 공유를 해제하려면 리소스 공유에서 제거해야 합니다. AWS CodeBuild 콘솔, AWS RAM 콘솔 또는 AWS CLI를 사용하여 이 작업을 수행할 수 있습니다.

소유한 공유 프로젝트의 공유를 해제하려면(AWS RAM 콘솔)

AWS RAM 사용 설명서에서 리소스 공유 업데이트를 참조하세요.

소유한 공유 프로젝트의 공유를 해제하려면(AWS CLI)

disassociate-resource-share 명령을 사용합니다.

소유한 프로젝트의 공유를 해제하려면(CodeBuild 명령)

delete-resource-policy 명령을 실행하고 공유를 해제할 프로젝트의 ARN을 지정합니다.

```
aws codebuild delete-resource-policy --resource-arn project-arn
```

공유 프로젝트 식별

소유자와 소비자는 AWS CLI를 사용하여 공유 프로젝트를 식별할 수 있습니다.

AWS 계정 또는 사용자와 공유된 프로젝트를 식별하려면(AWS CLI)

list-shared-projects 명령을 사용하여 공유된 프로젝트를 반환합니다.

공유 프로젝트 권한

소유자에 대한 권한

프로젝트 소유자는 프로젝트를 편집하고 빌드를 실행하는 데 사용할 수 있습니다.

소비자에 대한 권한

프로젝트 소비자는 프로젝트와 해당 빌드를 볼 수 있지만 프로젝트를 편집하거나 빌드를 실행하는 데 사용할 수는 없습니다.

AWS CodeBuild에서 프로젝트 태그 지정

태그는 사용자 또는 AWS에서 AWS 리소스에 할당하는 사용자 지정 속성 레이블입니다. 각 AWS 태그는 두 부분으로 구성됩니다.

- 태그 키(예: `CostCenter`, `Environment`, `Project` 또는 `Secret`). 태그 키는 대/소문자를 구별합니다.
- 태그 값(예: `111122223333`, `Production` 또는 팀 이름)으로 알려진 선택적 필드. 태그 값을 생략하는 것은 빈 문자열을 사용하는 것과 같습니다. 태그 키처럼 태그 값은 대/소문자를 구별합니다.

태그 키와 태그 값을 합해서 키 값 페어라고 합니다. 프로젝트에서 포함할 수 있는 태그 수와 태그 키 및 값 제한에 대한 자세한 내용은 Tags 단원을 참조하십시오..

태그를 사용하면 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당하여 해당 리소스의 관련 여부를 나타낼 수 있습니다. 예를 들어 S3 버킷에 할당한 것과 동일한 태그를 CodeBuild 프로젝트에 할당할 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례를 참조하세요.

CodeBuild에서 기본 리소스는 프로젝트와 보고서 그룹입니다. CodeBuild 콘솔, AWS CLI, CodeBuild API 또는 AWS SDK를 사용하여 프로젝트의 태그를 추가, 관리 및 제거할 수 있습니다. 태그로 프로젝트를 식별, 구성 및 추적하는 것 외에도 IAM 정책의 태그를 사용하여 프로젝트를 보고, 상호 작용할 수 있는 사용자를 제어할 수 있습니다. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

주제

- 프로젝트에 태그 추가
- 프로젝트의 태그 보기
- 프로젝트의 태그 편집
- 프로젝트에서 태그 제거

프로젝트에 태그 추가

프로젝트에 태그를 추가하면 AWS 리소스를 식별 및 구성하고 해당 리소스에 대한 액세스를 관리할 수 있습니다. 먼저 프로젝트에 하나 이상의 태그(키-값 페어)를 추가합니다. 프로젝트에 태그 수에 대한 제한이 있음을 알아두십시오. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있습니다. 자세한 내용은 Tags 섹션을 참조하세요. 태그가 생성된 후 해당 태그를 기준으로 프로젝트에 대한 액세스를 관리하는 IAM 정책을 생성할 수 있습니다. CodeBuild 콘솔 또는 AWS CLI를 사용하여 프로젝트에 태그를 추가할 수 있습니다.

> **Important**
>
> 프로젝트에 태그를 추가하기 전에 프로젝트와 같은 리소스에 대한 액세스를 제어하는 태그를 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

프로젝트를 생성할 때 프로젝트에 태그를 추가하는 방법에 대한 자세한 내용은 프로젝트에 태그 추가(콘솔) 단원을 참조하십시오.

주제

- 프로젝트에 태그 추가(콘솔)
- 프로젝트에 태그 추가(AWS CLI)

프로젝트에 태그 추가(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 프로젝트에 하나 이상의 태그를 추가할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트에서 태그를 추가할 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다. 빌드 프로젝트 태그를 선택합니다.
4. 프로젝트에 태그가 추가되지 않은 경우 태그 추가를 선택합니다. 또는 편집을 선택한 다음 태그 추가를 선택합니다.
5. 키에 태그 이름을 입력합니다. 값(Value)에 태그의 선택적 값을 추가할 수 있습니다.
6. (선택 사항) 다른 태그를 추가하려면 다시 태그 추가를 선택합니다.
7. 태그 추가를 마쳤으면 제출을 선택합니다.

프로젝트에 태그 추가(AWS CLI)

프로젝트를 생성할 때 프로젝트에 태그를 추가하려면 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오. `create-project.json`에서 태그를 추가합니다.

이 단계에서는 사용자가 이미 최신 버전의 AWS CLI를 설치했거나 현재 버전으로 업데이트했다고 가정합니다. 자세한 정보는 AWS Command Line Interface 설치 섹션을 참조하세요.

성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

프로젝트의 태그 보기

태그를 사용하면 AWS 리소스를 식별 및 구성하고 해당 리소스에 대한 액세스를 관리할 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례 백서를 참조하십시오. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

프로젝트의 태그 보기(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 프로젝트와 연결된 태그를 볼 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.

2. 빌드 프로젝트에서 태그를 보려는 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다. 빌드 프로젝트 태그를 선택합니다.

프로젝트의 태그 보기(AWS CLI)

빌드 프로젝트의 태그를 보려면 다음 명령을 실행합니다. `--names` 파라미터에 대해 프로젝트 이름을 사용합니다.

```
aws codebuild batch-get-projects --names your-project-name
```

성공하면, 이 명령은 다음과 같은 내용을 포함하는 빌드 프로젝트에 대한 JSON 형식의 정보를 반환합니다.

```
{
    "tags": {
        "Status": "Secret",
        "Team": "JanesProject"
    }
}
```

프로젝트에 태그가 없는 경우에는 `tags` 섹션이 비어 있습니다.

```
"tags": []
```

프로젝트의 태그 편집

프로젝트와 연결된 태그에 대한 값을 변경할 수 있습니다. 또한 키 이름을 변경할 수 있습니다. 이는 현재 태그를 제거하고 새 이름 및 다른 키와 동일한 값을 가진 다른 태그를 추가하는 것과 동일합니다. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있음을 알아두십시오. 자세한 내용은 Tags 섹션을 참조하세요.

> **Important**
>
> 프로젝트의 태그를 편집하면 해당 프로젝트에 대한 액세스에 영향을 줄 수 있습니다. 프로젝트에 대한 태그의 이름(키) 또는 값을 편집하기 전에 프로젝트와 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

프로젝트의 태그 편집(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 프로젝트와 연결된 태그를 편집할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트에서 태그를 편집할 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다. 빌드 프로젝트 태그를 선택합니다.
4. 편집(Edit)을 선택합니다.
5. 다음 중 하나를 수행하세요.
   - 태그를 변경하려면 키에 새 이름을 입력합니다. 태그 이름을 변경하는 것은 태그를 제거하고 새 키 이름의 새 태그를 추가하는 것과 동일합니다.
   - 태그 값을 변경하려면 새 값을 입력합니다. 값을 어떤 것으로도 변경하지 않으려면 현재 값을 삭제하고 필드를 비워둡니다.
6. 태그 편집을 마쳤으면 제출을 선택합니다.

프로젝트의 태그 편집(AWS CLI)

빌드 프로젝트에서 태그를 추가, 변경 또는 삭제하려면 빌드 프로젝트 설정 변경(AWS CLI) 단원을 참조하십시오. 프로젝트를 업데이트하는 데 사용하는 JSON 형식 데이터의 `tags` 섹션을 업데이트합니다.

프로젝트에서 태그 제거

프로젝트와 연결된 태그를 하나 이상 제거할 수 있습니다. 태그를 제거할 때 해당 태그와 연결된 다른 AWS 리소스에서 해당 태그가 삭제되지는 않습니다.

> **Important**
>
> 프로젝트의 태그를 제거하면 해당 프로젝트에 대한 액세스에 영향을 줄 수 있습니다. 프로젝트에서 태그를 제거하기 전에 프로젝트와 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

프로젝트에서 태그 제거(콘솔)

CodeBuild 콘솔을 사용하면 태그와 CodeBuild 프로젝트 간의 연결을 제거할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트에서 태그를 제거할 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다. 빌드 프로젝트 태그를 선택합니다.
4. 편집(Edit)을 선택합니다.
5. 제거할 태그를 찾은 다음 태그 제거를 선택합니다.
6. 태그 제거를 마쳤으면 제출을 선택합니다.

프로젝트에서 태그 제거(AWS CLI)

빌드 프로젝트에서 하나 이상의 태그를 삭제하려면 빌드 프로젝트 설정 변경(AWS CLI) 단원을 참조하십시오. 삭제하려는 태그가 포함되지 않은 업데이트된 태그 목록으로 JSON 형식 데이터의 `tags` 섹션을 업데이트합니다. 모든 태그를 삭제하려면 `tags` 섹션을 다음과 같이 업데이트하십시오.

```
"tags: []"
```

> **Note**
>
> CodeBuild 빌드 프로젝트를 삭제하면 삭제된 빌드 프로젝트에서 모든 태그 연결이 제거됩니다. 프로젝트를 삭제하기 전에 태그를 제거할 필요가 없습니다.

Batch 빌드 인 AWS CodeBuild

를 사용하여 일괄 빌드를 AWS CodeBuild 사용하여 프로젝트의 동시 및 조정된 빌드를 실행할 수 있습니다.

주제

- 보안 역할
- 배치 빌드 유형
- 배치 보고서 모드
- 추가 정보

보안 역할

배치 빌드는 배치 구성에 새로운 보안 역할을 도입합니다. 일괄 처리의 일부로 빌드를 실행하려면 사용자 대신`StartBuild`,`StopBuild`, `RetryBuild` 작업을 호출할 수 CodeBuild 있어야 하므로 이 새 역

할이 필요합니다. 고객은 다음과 같은 두 가지 이유로 빌드에 사용하는 것과 동일한 역할이 아닌 새 역할을 사용해야 합니다.

- 빌드 역할 `StartBuild`, `StopBuild` 및 `RetryBuild` 권한을 부여하면 단일 빌드에서 buildspec을 통해 더 많은 빌드를 시작할 수 있습니다.
- CodeBuild 배치 빌드는 일괄 빌드에 사용할 수 있는 빌드 및 컴퓨팅 유형의 수를 제한하는 제한을 제공합니다. 빌드 역할에 이러한 권한이 있는 경우 빌드 자체가 이러한 제한을 우회할 수 있습니다.

배치 빌드 유형

CodeBuild 다음과 같은 배치 빌드 유형을 지원합니다.

배치 빌드 유형

- 빌드 그래프
- 빌드 목록
- 빌드 매트릭스

빌드 그래프

빌드 그래프는 일괄 처리의 다른 태스크에 종속되는 일련의 태스크를 정의합니다.

다음 예제는 종속성 체인을 생성하는 빌드 그래프를 정의합니다.

```
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
```

이 예제에서는 다음이 적용됩니다.

- 종속성이 없으므로 build1이 먼저 실행됩니다.
- build2는 build1에 종속되어 있으므로 build2는 build1 완료 후에 실행됩니다.
- build3는 build2에 종속되어 있으므로 build3는 build2 완료 후에 실행됩니다.

빌드 그래프 buildspec 구문에 대한 자세한 내용은 batch/build-graph 섹션을 참조하세요.

빌드 목록

빌드 목록은 병렬로 실행되는 여러 태스크를 정의합니다.

다음 예제에서는 빌드 목록을 정의합니다. build1 및 build2 빌드는 병렬로 실행됩니다.

```
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
```

빌드 목록 buildspec 구문에 대한 자세한 내용은 batch/build-list 섹션을 참조하세요.

빌드 매트릭스

빌드 매트릭스는 병렬로 실행되는 다양한 구성의 태스크를 정의합니다. CodeBuild 가능한 각 구성 조합에 대해 별도의 빌드를 만듭니다.

다음 예제는 buildspec 파일 2개와 환경 변수 값 3개가 포함된 빌드 매트릭스를 보여 줍니다.

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
            - VALUE3
```

이 예제에서는 6개의 빌드를 CodeBuild 만듭니다.

- `$MY_VAR=VALUE1`가 있는 `matrix1.yml`
- `$MY_VAR=VALUE2`가 있는 `matrix1.yml`
- `$MY_VAR=VALUE3`가 있는 `matrix1.yml`
- `$MY_VAR=VALUE1`가 있는 `matrix2.yml`
- `$MY_VAR=VALUE2`가 있는 `matrix2.yml`
- `$MY_VAR=VALUE3`가 있는 `matrix2.yml`

각 빌드에는 다음과 같은 설정이 있습니다.

- `ignore-failure`가 `false`로 설정됨
- `env/type`이 `LINUX_CONTAINER`로 설정됨
- `env/image`이 `aws/codebuild/amazonlinux2-x86_64-standard:4.0`로 설정됨
- `env/privileged-mode`가 `true`로 설정됨

이러한 빌드는 병렬로 실행됩니다.

매트릭스 buildspec 구문에 대한 자세한 내용은 batch/build-matrix 섹션을 참조하세요.

배치 보고서 모드

프로젝트의 소스 제공자가 Bitbucket 또는 GitHub Enterprise이고 프로젝트가 빌드 상태를 소스 제공자에게 보고하도록 구성된 경우 배치 빌드 상태를 소스 제공자에게 보내는 방법을 선택할 수 있습니다. GitHub 상태를 배치에 대한 단일 집계 상태 보고서로 전송하거나, 배치에 있는 각 빌드의 상태를 개별적으로 보고하도록 선택할 수 있습니다.

자세한 정보는 다음 주제를 참조하세요.

- 빌드 구성(생성)
- 빌드 구성(업데이트)

추가 정보

자세한 정보는 다음 주제를 참조하세요.

- 배치 빌드 buildspec 참조
- 배치 구성
- 배치 빌드 실행(AWS CLI)
- AWS CodeBuild에서 배치 빌드 중지

GitHub 액션 러너 인 AWS CodeBuild

GitHub 액션은 GitHub 워크플로우와 함께 사용하도록 특별히 개발된 액션입니다. GitHub 액션에 대한 자세한 내용은 GitHub 액션 설명서를 참조하십시오.

GitHub 액션을 CodeBuild 다음과 함께 사용하는 두 가지 방법이 있습니다.

- CodeBuild 컨테이너에 자체 호스팅된 GitHub 액션 러너를 설정하여 액션 워크플로 GitHub 작업을 처리하도록 프로젝트를 구성할 수 있습니다.
- CodeBuild-managed 액션 러너를 사용하여 내에서 액션을 실행할 수 있습니다. GitHub CodeBuild

에서 자체 호스팅 GitHub 액션 러너를 설정하도록 선택할 수 있습니다. CodeBuild 여기에는 CodeBuild 프로젝트를 사용하여 웹후크를 설정하고, 머신에서 호스팅되는 자체 호스팅 러너를 사용하도록 GitHub Actions 워크플로 YAML을 업데이트하는 작업이 포함됩니다. CodeBuild 이를 통해 GitHub Actions 워크플로 작업을 기본적으로 통합할 수 있습니다. AWS

또한 CodeBuild -managed 액션 러너를 사용하여 내에서 액션을 GitHub 실행하도록 선택할 수도 있습니다. CodeBuild 여기에는 명령과 별도의 단계에서 실행되는 GitHub Actions 구문을 사용하여 빌드스펙에 `steps` 추가하는 작업이 포함됩니다. CodeBuild 이렇게 하면 GitHub 액션을 종속성 캐싱 CodeBuild 및 배치 빌드와 같은 기능과 통합할 수 있습니다.

주제

- 에서 자체 호스팅 GitHub 액션 러너 설정 AWS CodeBuild
- 빌드 사양에서 GitHub 액션 구문 사용 AWS CodeBuild

에서 자체 호스팅 GitHub 액션 러너 설정 AWS CodeBuild

CodeBuild 컨테이너에 자체 호스팅된 GitHub 액션 러너를 설정하여 액션 워크플로 작업을 처리하도록 프로젝트를 구성할 수 있습니다 GitHub . CodeBuild 프로젝트를 사용하여 웹후크를 설정하고 머신에서 호스팅되는 자체 호스팅 러너를 사용하도록 GitHub Actions 워크플로 YAML을 업데이트하면 됩니다. CodeBuild 자세한 내용은 자체 호스팅 러너 정보를 참조하십시오.

GitHub Actions 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 상위 단계는 다음과 같습니다.

1. 아직 만들지 않았다면 개인용 액세스 토큰을 만들거나 OAuth 앱으로 연결하여 프로젝트를 연결하세요. GitHub
2. CodeBuild 콘솔로 이동하여 웹후크가 포함된 CodeBuild 프로젝트를 만들고 웹후크 필터를 설정합니다.
3. 에서 GitHub 액션 워크플로 YAML을 GitHub 업데이트하여 빌드 환경을 구성하세요.

자세한 절차는 을 참조하십시오자습서: CodeBuild 자체 호스팅 Actions 러너 GitHub 구성.

이 기능을 사용하면 GitHub Actions 워크플로 작업이 기본적으로 통합되어 IAM AWS, 통합, AWS Secrets Manager Amazon VPC와 같은 기능을 통해 보안 및 편의성을 제공할 수 있습니다. AWS CloudTrail ARM 기반 인스턴스를 비롯한 최신 인스턴스 유형에 액세스할 수 있습니다.

주제

- 자습서: CodeBuild 자체 호스팅 Actions 러너 GitHub 구성
- CodeBuild-호스트 GitHub 액션 러너에 대한 정보

자습서: CodeBuild 자체 호스팅 Actions 러너 GitHub 구성

이 가이드에서는 GitHub Actions 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 방법을 보여줍니다.

필수 조건

이 자습서를 완료하려면 먼저 다음을 수행해야 합니다.

- OAuth 앱으로 연결하거나 개인용 액세스 토큰을 만드세요. OAuth 앱에 연결하려면 CodeBuild 콘솔을 사용해야 합니다. 개인용 액세스 토큰을 만들려면 CodeBuild 콘솔을 사용하거나 API를 사용할 수 있습니다. ImportSourceCredentials 자세한 지침은 을 참조하십시오GitHub 및 GitHub 엔터프라이즈 서버 액세스 토큰.
- GitHub 계정에 CodeBuild 연결하세요. 이렇게 하려면 다음 중 하나를 수행할 수 있습니다.
  - 콘솔에서 소스 GitHub 공급자로 추가할 수 있습니다. OAuth 앱 또는 개인용 액세스 토큰으로 연결할 수 있습니다. 지침은 액세스 GitHub 토큰으로 연결 (콘솔) 을 참조하세요.
  - API를 통해 GitHub 자격 증명을 가져올 수 있습니다. ImportSourceCredentials 개인용 액세스 토큰으로만 이 작업을 수행할 수 있습니다. OAuth 앱을 사용하여 연결하는 경우 대신 콘솔을 사용하여 연결해야 합니다. 지침은 액세스 토큰 (CLI) GitHub 으로 연결 을 참조하세요.

> **Note**
>
> 계정에 연결하지 않은 경우에만 이 작업을 수행하면 됩니다. GitHub

1단계: CodeBuild 웹후크를 사용하여 프로젝트 만들기

이 단계에서는 웹후크가 포함된 CodeBuild 프로젝트를 만들고 콘솔에서 검토합니다 GitHub .

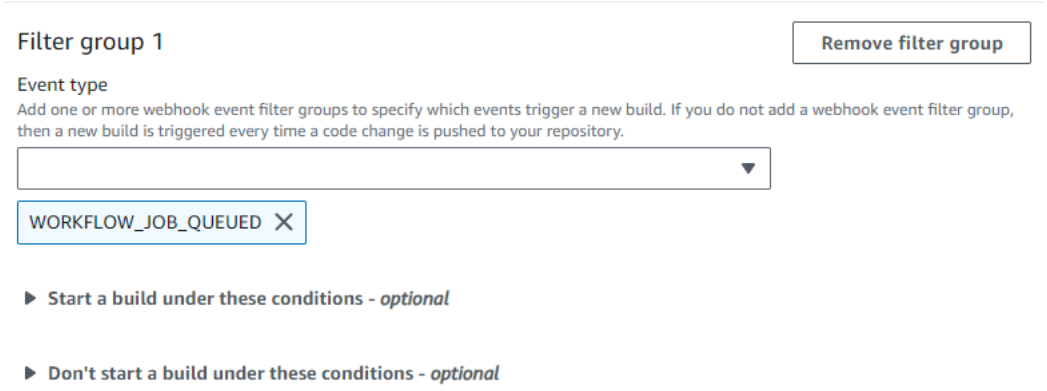
웹후크가 있는 CodeBuild 프로젝트를 만들려면

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
   - 소스에서 다음과 같이 합니다.
     - 소스 제공자의 경우 선택하십시오 GitHub.
     - 리포지토리의 경우 내 GitHub 계정의 리포지토리를 선택합니다.

- 리포지토리 URL에 **https://github.com/*user-name*/*repository-name***을 입력합니다.

- 기본 소스 웹후크 이벤트에서:
  - Webhook - 선택 사항의 경우 코드 변경 사항이 이 리포지토리로 푸시될 때마다 재구축을 선택합니다.
  - 이벤트 유형으로는 WORKFLOW_JOB_QUEUED을 선택합니다. 이 기능을 활성화하면 작업 워크플로 작업 이벤트에 의해서만 빌드가 트리거됩니다. GitHub

> **Note**
>
> CodeBuild 웹후크에 WORKFLOW_JOB_QUEUED 이벤트 필터가 포함된 필터 그룹이 있는 경우에만 작업 워크플로 작업 이벤트를 처리합니다 GitHub .



- 환경에서 다음과 같이 합니다.
  - 지원되는 환경 이미지를 선택하고 컴퓨팅하십시오. 참고로 GitHub Actions 워크플로 YAML에는 라벨을 사용하여 이미지 및 인스턴스 설정을 재정의할 수 있는 옵션이 있습니다. 자세한 내용은 2단계: GitHub 액션 워크플로 YAML 업데이트단원을 참조하세요.
- Buildspec에서 다음과 같이 합니다.
  - 참고로 빌드스펙은 무시됩니다. 대신 자체 CodeBuild 호스팅 러너를 설정하는 명령을 사용하도록 오버라이드합니다. 이 프로젝트의 주요 책임은 Actions 워크플로 작업을 실행하도록 자체 호스팅 러너를 설정하는 CodeBuild 것입니다. GitHub

3. 기본값을 계속 사용하고 빌드 프로젝트 만들기를 선택합니다.
4. 에서 GitHub `https://github.com/`*`user-name`*`/`*`repository-name`*`/settings/hooks` 콘솔을 열어 웹후크가 생성되었고 Workflow 작업 이벤트를 전달할 수 있도록 활성화되었는지 확인합니다.

2단계: GitHub 액션 워크플로 YAML 업데이트

이 단계에서는 GitHub Actions 워크플로 YAML 파일을 GitHub업데이트하여 빌드 환경을 구성하고 에서 GitHub Actions 자체 호스팅 러너를 사용합니다. CodeBuild 자세한 내용은 자체 호스팅 러너에서 레이블 사용을 참조하세요.

GitHub 액션 워크플로 YAML을 업데이트하세요.

GitHub Actions 워크플로 YAML에서 runs-on설정을 GitHub탐색하고 업데이트하여 빌드 환경을 구성하세요. 이렇게 하려면 다음 중 하나를 수행할 수 있습니다.

- 프로젝트 이름과 실행 ID를 지정할 수 있습니다. 이 경우 빌드는 컴퓨팅, 이미지, 이미지 버전, 인스턴스 크기에 기존 프로젝트 구성을 사용합니다. GitHub Actions 작업의 AWS관련 설정을 특정 CodeBuild 프로젝트에 연결하려면 프로젝트 이름이 필요합니다. YAML에 프로젝트 이름을 포함하면 올바른 프로젝트 설정으로 작업을 호출할 수 있습니다. CodeBuild 실행 ID를 CodeBuild 제공하면 빌드를 특정 워크플로 실행에 매핑하고 워크플로 실행이 취소되면 빌드를 중지합니다. 자세한 내용은 github컨텍스트를 참조하십시오.

  ```
  runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
  ```

  > **Note**
  >
  > 가 <project-name>이전 단계에서 만든 프로젝트의 이름과 일치하는지 확인하세요. 일치하지 않으면 CodeBuild 웹후크를 처리하지 않고 GitHub 액션 워크플로가 중단될 수 있습니다.

  다음은 GitHub 액션 워크플로 YAML의 예입니다.

  ```
  name: Hello World
  on: [push]
  jobs:
    Hello-World-Job:
      runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      steps:
        - run: echo "Hello World!"
  ```

- 라벨의 이미지와 컴퓨팅 유형을 재정의할 수도 있습니다. 이렇게 하면 프로젝트의 환경 설정이 재정의됩니다. Amazon EC2 컴퓨팅 빌드의 환경 설정을 재정의하려면 다음 구문을 사용하십시오.

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<image>-<image-version>-<instance-size>
```

Lambda 컴퓨팅 빌드의 환경 설정을 재정의하려면 다음 구문을 사용하십시오.

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<environment-type>-<runtime-version>-<instance-size>
```

다음은 GitHub 액션 워크플로 YAML의 예시입니다.

```
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**
>
> CodeBuild 환경에서 GitHub -hosted runners에서 제공하는 종속성을 사용할 수 없는 경우 워크플로 실행의 Actions를 사용하여 GitHub 종속성을 설치할 수 있습니다. 예를 들어 setup-python작업을 사용하여 빌드 환경에 Python을 설치할 수 있습니다.

지원되는 컴퓨팅 이미지

레이블에서 처음 세 열의 값을 사용하여 Amazon EC2 환경 설정을 재정의할 수 있습니다. CodeBuild는 다음과 같은 Amazon EC2 컴퓨팅 이미지를 제공합니다.

이미지	이미지 버전	인스턴스 크기	플랫폼	이미지 식별자	정의
linux	4.0	small medium	Amazon Linux 2	aws/codebuild/amazonlinux2-	al2/standard/4.0

이미지	이미지 버전	인스턴스 크기	플랫폼	이미지 식별자	정의
		large xlarge		x86_64-standard:4.0	
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2023	aws/codebuild/amazonlinux2-x86_64-standard:5.0	al2/standard/5.0
arm	2.0	small large	Amazon Linux 2	aws/codebuild/amazonlinux2-aarch64-standard:2.0	al2/aarch64/standard/2.0
arm	3.0		Amazon Linux 2023	aws/codebuild/amazonlinux2-aarch64-standard:3.0	al2/aarch64/standard/3.0

이미지	이미지 버전	인스턴스 크기	플랫폼	이미지 식별자	정의
ubuntu	5.0	small medium large xlarge 2xlarge gpu_small gpu_large	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0		Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0		Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	N/A
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A

또한 다음 값을 사용하여 Lambda 환경 설정을 재정의할 수 있습니다. CodeBuild Lambda 컴퓨팅에 대한 자세한 내용은 사용을 참조하십시오. AWS Lambda 컴퓨트 인에서 작업하기 AWS CodeBuild CodeBuild 다음과 같은 Lambda 컴퓨팅 이미지를 지원합니다.

환경 유형	실행 시간 버전	인스턴스 크기			
linux-lambda arm-lambda	dotnet6 go1.21 corretto11 corretto17 corretto21 nodejs18 nodejs20 python3.11 python3.12 ruby3.2	1GB 2GB 4GB 8GB 10GB			

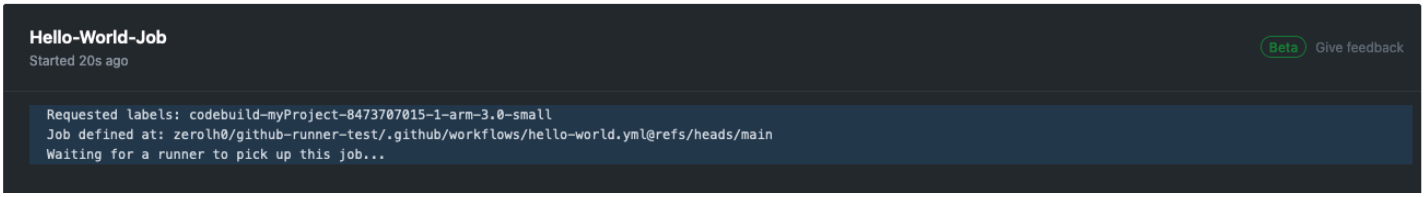
자세한 내용은 빌드 환경 컴퓨팅 모드 및 유형 및 Docker 이미지 제공: CodeBuild 섹션을 참조하세요.

3단계: 결과 검토

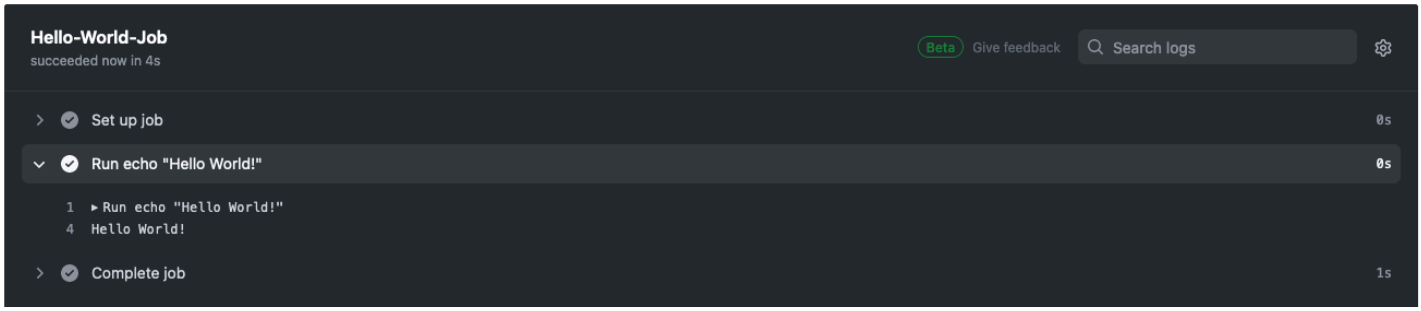
GitHub Actions 워크플로가 CodeBuild 실행될 때마다 웹후크를 통해 워크플로 작업 이벤트를 수신합니다. 워크플로의 각 작업에 대해 임시 GitHub 액션 러너를 실행하기 위한 빌드를 CodeBuild 시작합니다. 러너는 단일 워크플로 작업 실행을 담당합니다. 작업이 완료되면 러너 및 관련 빌드 프로세스가 즉시 종료됩니다.

워크플로 작업 로그를 보려면 의 GitHub 저장소로 이동하여 작업을 선택하고 원하는 워크플로를 선택한 다음 로그를 검토하려는 특정 작업을 선택합니다.

자체 호스팅 실행자가 작업을 선택하기를 기다리는 동안 로그에서 요청된 레이블을 검토할 수 있습니다. CodeBuild



작업이 완료되면 작업 로그를 볼 수 있습니다.



CodeBuild-호스트 GitHub 액션 러너에 대한 정보

라벨에 이미지와 인스턴스 오버라이드를 언제 포함해야 하나요?

각 Actions 워크플로 작업에 대해 서로 다른 빌드 환경을 지정하기 위해 라벨에 이미지 및 인스턴스 오버라이드를 포함할 수 있습니다 GitHub . 여러 CodeBuild 프로젝트나 웹훅을 만들 필요 없이 이 작업을 수행할 수 있습니다. 예를 들어 워크플로 작업에 매트릭스를 사용해야 할 때 유용합니다.

```
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**
>
> GitHub 작업 `runs-on` 컨텍스트가 포함된 레이블이 여러 개 있는 경우 따옴표가 필요할 수 있습니다.

이 기능에 AWS CloudFormation 사용할 수 있나요?

예. 프로젝트 웹후크의 GitHub Actions 워크플로 작업 이벤트 필터를 지정하는 필터 그룹을 AWS CloudFormation 템플릿에 포함할 수 있습니다.

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
```

자세한 정보는 GitHub 웹후크 이벤트 필터링 ()AWS CloudFormation을 참조하세요.

AWS CloudFormation 템플릿에서 프로젝트 자격 증명을 설정하는 데 도움이 필요한 경우 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하십시오 AWS::CodeBuild::SourceCredential.

CodeBuild-hosted GitHub Actions 러너 사용을 지원하는 지역은 어디입니까?

CodeBuild-호스트 GitHub 액션 러너는 모든 지역에서 지원됩니다. CodeBuild 사용 가능한 AWS 리전 위치에 CodeBuild 대한 자세한 내용은 지역별AWS 서비스를 참조하십시오.

CodeBuild-호스트 GitHub 액션 러너 사용을 지원하는 플랫폼은 무엇입니까?

CodeBuild-호스팅 GitHub 액션 러너는 Amazon AWS LambdaEC2와 컴퓨팅 모두에서 지원됩니다. 다음 플랫폼을 사용할 수 있습니다: 아마존 리눅스 2, 아마존 리눅스 2023, 우분투, 윈도우 서버 코어 2019. 자세한 내용은 EC2 컴퓨팅 이미지 및 Lambda 컴퓨팅 이미지 섹션을 참조하세요.

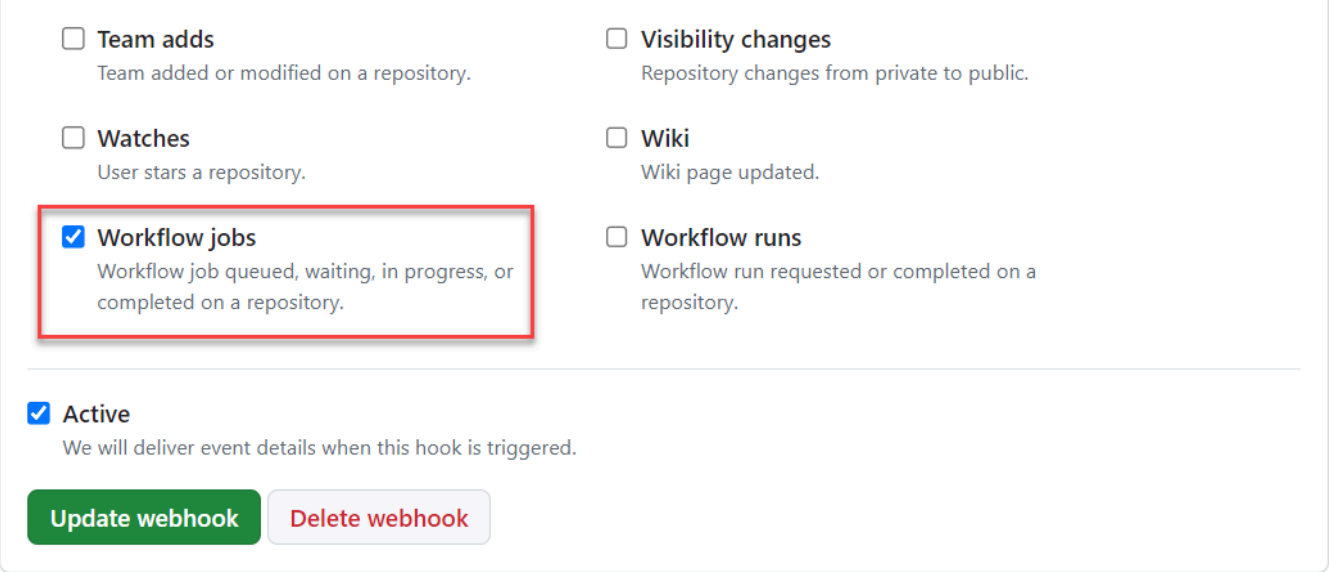
문제 해결: 웹훅이 작동하지 않는 경우 문제를 해결하려면 어떻게 해야 합니까?

문제: 웹후크가 작동하지 않거나 워크플로 작업이 중단되고 있습니다. GitHub

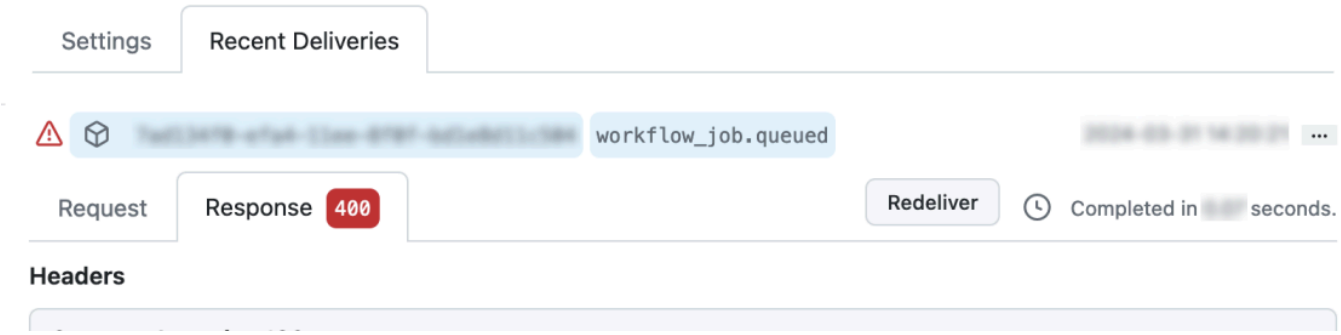
가능한 원인: 웹후크 워크플로 작업 이벤트가 빌드를 트리거하지 못할 수 있습니다. 응답 로그를 검토하여 응답 또는 오류 메시지를 확인하세요.

권장 해결 방법: 이 오류를 디버깅하려면 다음 지침을 사용하십시오.

1. 에서 GitHub `https://github.com/user-name/repository-name/settings/hooks` 콘솔을 열어 리포지토리의 웹후크 설정을 확인하십시오. 이 페이지에서는 리포지토리용으로 생성된 웹후크를 볼 수 있습니다.
2. 편집을 선택하고 웹후크가 Workflow jobs 이벤트를 전송할 수 있도록 활성화되었는지 확인합니다.



3. 최근 배송 탭으로 이동하여 해당 `workflow_job.queued` 이벤트를 찾고 이벤트를 확장하십시오.
4. 페이로드의 레이블 필드를 검토하여 예상한 대로인지 확인하십시오.
5. 마지막으로 Response 탭을 검토하세요. Response 탭에는 에서 CodeBuild 반환된 응답 또는 오류 메시지가 포함되어 있습니다.



빌드 사양에서 GitHub 액션 구문 사용 AWS CodeBuild

CodeBuild-managed 액션 러너를 사용하여 내에서 액션을 실행할 수 있습니다. GitHub CodeBuild 이 작업은 buildspec 파일의 아무 단계에나 `steps`를 추가하여 수행할 수 있습니다.

CodeBuild buildspec은 명령과는 별도의 단계에서 실행되는 순차적 GitHub 액션 단계 목록을 지원합니다. CodeBuild 이러한 GitHub 액션은 종속성 캐싱, 배치 빌드, 액세스 등을 포함하는 CodeBuild 의 기존 기능과 통합됩니다. AWS Secrets Manager

주제

- 빌드 사양에서 GitHub Action을 사용하려면 어떻게 해야 하나요?
- 빌드 사양에서 어떤 GitHub 액션을 사용할 수 있나요?
- 빌드스펙에서 GitHub 액션을 사용할 GitHub 때 말고 다른 소스 공급자를 사용할 수 있나요?
- 빌드 사양에서 GitHub 액션을 사용하려면 소스 GitHub 공급자로 연결해야 하는 이유는 무엇입니까?
- 빌드 사양에서 GitHub 액션을 사용하는 데 비용이 얼마나 드나요?
- 내 빌드스펙에서 GitHub 액션을 사용할 수 있는 지역은 어디입니까?
- 빌드 사양에서 GitHub 액션을 사용하는 베스트 프랙티스
- 빌드 사양에서 GitHub 액션을 사용할 때의 제한 사항 CodeBuild
- GitHub 액션 러너 빌드스펙 참조
- GitHub 다음과 같은 액션 구문 샘플 AWS CodeBuild

빌드 사양에서 GitHub Action을 사용하려면 어떻게 해야 하나요?

빌드 사양에서 GitHub 액션을 사용하는 상위 단계는 다음과 같습니다.

1. 아직 연결하지 않았다면 프로젝트를 에 연결하세요. GitHub

   이렇게 하려면 다음 중 하나를 수행할 수 있습니다.

   - 콘솔에서 소스 GitHub 공급자로 추가할 수 있습니다. 자세한 정보는 액세스 GitHub 토큰으로 연결 (콘솔) 을 참조하세요.
   - CodeBuild API를 통해 GitHub 자격 증명을 가져올 수 있습니다. 자세한 정보는 액세스 토큰 (CLI) GitHub 으로 연결 을 참조하세요.

   > **Note**
   >
   > 다른 프로젝트에 연결하지 않은 경우에만 이 작업을 수행하면 됩니다. GitHub

2. 프로젝트의 빌드스펙에서 각각 액션을 참조하는 것을 추가할 `steps` 수 있습니다. GitHub CodeBuild 콘솔이나 소스 리포지토리에서 편집할 수 있습니다. 각 빌드 단계는 명령 목록이나 단계 목록을 지원하지만 둘 다 같은 단계에서 사용할 수는 없습니다. 자세한 정보는 빌드 사양에서 GitHub 액션 구문 사용 AWS CodeBuild을 참조하세요.

빌드 사양에서 어떤 GitHub 액션을 사용할 수 있나요?

GitHub Marketplace에서 사용할 수 있는 작업 중 이러한 제한과 충돌하지 않는 모든 작업을 사용할 수 있습니다.

빌드스펙에서 GitHub 액션을 사용할 GitHub 때 말고 다른 소스 공급자를 사용할 수 있나요?

예. 하지만 Action을 사용하여 GitHub 인증하고 액세스하려면 여전히 GitHub 연결이 필요합니다. GitHub 자세한 정보는 GitHub 및 GitHub 엔터프라이즈 서버 액세스 토큰을 참조하세요.

빌드 사양에서 GitHub 액션을 사용하려면 소스 GitHub 공급자로 연결해야 하는 이유는 무엇입니까?

빌드 사양에서 GitHub Action을 사용하려면 소스를 빌드 컴퓨트에 다운로드해야 합니다. 익명 다운로드는 속도가 제한되므로 에 연결하면 일관된 액세스를 보장하는 데 도움이 될 수 있습니다. GitHub

빌드 사양에서 GitHub 액션을 사용하는 데 비용이 얼마나 드나요?

빌드 사양에서 GitHub 액션을 사용하는 것은 추가 비용 없이 지원됩니다.

내 빌드스펙에서 GitHub 액션을 사용할 수 있는 지역은 어디입니까?

빌드 사양에서 GitHub 액션을 사용하는 것은 모든 지역에서 지원됩니다. CodeBuild 사용 가능한 AWS 리전 위치에 CodeBuild 대한 자세한 내용은 지역별 AWS 서비스를 참조하십시오.

빌드 사양에서 GitHub 액션을 사용하는 베스트 프랙티스

GitHub 액션은 오픈소스이며 커뮤니티에서 빌드하고 유지 관리합니다. 우리는 공동 책임 모델을 따르며 GitHub Actions 소스 코드를 귀하가 책임져야 하는 고객 데이터로 간주합니다. GitHub 액션에는 비밀, 리포지토리 토큰, 소스 코드, 계정 링크에 대한 액세스 권한을 부여할 수 있습니다. 실행하려는 GitHub 작업의 신뢰성과 보안에 확신이 있는지 확인하세요.

작업에 대한 보다 구체적인 지침 및 보안 모범 사례: GitHub

- 보안 강화
- pwn 요청 방지
- 신뢰할 수 없는 입력
- 구성 요소를 신뢰하는 방법

빌드 사양에서 GitHub 액션을 사용할 때의 제한 사항 CodeBuild

- GitHub 내부적으로 `github`컨텍스트를 사용하거나 pull 요청 및 issue와 같은 GitHub 특정 리소스를 참조하는 빌드 사양 내 액션은 에서 지원되지 않습니다. CodeBuild 예를 들어, 다음 액션은 다음에서는 작동하지 않습니다. CodeBuild
  - GitHub GitHub리소스를 추가, 변경 또는 업데이트하려는 작업 (예: pull 요청을 업데이트하거나 에서 GitHub 이슈를 생성하는 작업)

> **Note**
>
> https://github.com/actions 에 나열된 대부분의 공식 GitHub 조치는 `github` 상황에 따라 달라집니다. 대신 GitHub Marketplace에서 사용할 수 있는 작업을 사용하세요.

- GitHub Docker 컨테이너 작업인 빌드 사양의 액션은 작동하지만 빌드 프로젝트는 권한 모드를 활성화하고 기본 Docker 사용자 (루트) 가 실행해야 합니다.
  - Actions는 루트 사용자로 실행해야 합니다. 자세한 내용은 액션에 대한 Dockerfile 지원의 USER 주제를 참조하세요. GitHub
- GitHub 빌드 사양의 액션은 Windows에서 실행되도록 구성된 CodeBuild 프로젝트에서는 지원되지 않습니다.
- GitHub 빌드스펙의 액션 작업 (단계 그룹) 및 GitHub 액션 작업 속성은 지원되지 않습니다.
- GitHub 빌드 스펙의 액션은 공개 Git 리포지토리의 웹후크에 의해 트리거되도록 구성된 CodeBuild 프로젝트에서는 지원되지 않습니다. 자세한 내용은 을 참조하십시오. git-credential-helper
- 퍼블릭 인터넷 액세스가 없는 VPC 빌드는 빌드 사양에서 GitHub 액션을 실행할 수 없습니다.
- 각 빌드 단계는 명령 목록이나 단계 목록을 지원하지만 둘 다 같은 단계에서 사용할 수는 없습니다. 예를 들어 다음 샘플에서는 사전 빌드 단계에서는 단계를 사용하여 GitHub 작업을 나열하고 빌드 단계에서는 명령을 사용하여 명령을 나열합니다. CodeBuild

```
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 액션 러너 빌드스펙 참조

이 항목에는 액션 러너 속성에 대한 buildspec 참조가 포함되어 있습니다. GitHub

단계(steps)

선택적 시퀀스. 단계는 에서 명령과 액션을 실행하는 데 사용됩니다. CodeBuild 자세한 정보는 빌드 사양에서 GitHub 액션 구문 사용 AWS CodeBuild을 참조하세요.

> **Note**
>
> 각 빌드 단계는 `commands` 목록이나 `steps` 목록을 지원하지만 둘 다 같은 단계에서 사용할 수는 없습니다.

각 빌드 단계는 다음 속성을 포함합니다.

id

선택 사항입니다. 다른 컨텍스트의 단계를 참조하는 데 사용할 수 있는 단계의 식별자입니다.

if

선택 사항입니다. 조건이 충족되지 않으면 단계가 실행되지 않도록 하는 데 사용할 수 있는 조건문입니다. 이 명령문은 표현식뿐 아니라 에서 CodeBuild 환경 변수를 참조하는 등 지원되는 모든 컨텍스트를 사용할 수 있습니다.

이름

선택 사항입니다. 단계의 이름입니다. 이름을 지정하지 않으면 기본 이름은 `run` 명령에 지정된 텍스트입니다.

uses

단계에 대해 실행되는 작업입니다. 일부 작업에서는 `with`를 사용하여 입력을 설정해야 합니다. 작업의 README를 참조하여 필요한 입력을 확인합니다. 자세한 정보는 빌드 사양에서 어떤 GitHub 액션을 사용할 수 있나요?을 참조하세요.

빌드 단계에서 `uses`를 지정한 경우 `run`과 함께 사용할 수 없습니다.

> **Note**
>
> 사용 중인 작업의 버전을 포함하는 것이 좋습니다. 이 작업은 Git ref, SHA 또는 Docker 태그를 지정하여 수행할 수 있습니다. 자세한 내용은 steps.uses 구문을 참조하세요.

run

명령줄 프로그램을 실행하는 명령입니다. 이것은 한 줄 명령일 수도 있고 여러 줄 명령일 수도 있습니다. 기본적으로 이러한 명령은 비로그인 쉘을 사용하여 실행됩니다. 다른 쉘을 선택하려면 `shell`을 사용합니다.

빌드 단계에서 `run`를 지정한 경우 `uses`과 함께 사용할 수 없습니다.

shell

선택 사항입니다. 이 시퀀스에 지정된 쉘입니다. 지원되는 쉘 파라미터는 steps.shell을 참조하세요. 지정하지 않은 경우 사용되는 쉘은 bash입니다. bash를 사용할 수 없는 경우 sh가 사용됩니다.

with

선택 사항입니다. 작업에 의해 정의된 입력 파라미터의 맵입니다. 각 파라미터는 키/값 페어로 구성됩니다.

with.args

선택 사항입니다. Docker 컨테이너의 입력을 정의하는 문자열입니다.

with.entrypoint

선택 사항입니다. Dockerfile에 지정된 Docker 진입점입니다.

env

선택 사항입니다. 환경에서 사용할 단계에 대해 지정된 변수입니다.

continue-on-error

선택 사항입니다. 이 단계 시퀀스의 실패를 무시할 수 있는지 여부를 나타내는 부울 값입니다.

`false`

기본값입니다. 이 단계 시퀀스가 실패하면 빌드가 실패합니다.

`true`

이 단계 시퀀스가 실패하더라도 빌드는 여전히 성공할 수 있습니다.

timeout-minutes

선택 사항입니다. 단계가 종료되기 전에 실행할 수 있는 최대 시간(분)입니다. 기본값은 제한 시간 없음입니다. 단계 제한 시간이 빌드 제한 시간을 초과하는 경우 빌드 제한 시간에 도달하면 단계가 중지됩니다.

다음은 슈퍼 GitHub 린터 액션을 사용한 예제입니다.

```
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 다음과 같은 액션 구문 샘플 AWS CodeBuild

이 샘플 그룹을 사용하여 빌드스펙의 GitHub 액션을 실험해 볼 수 있습니다. CodeBuild

주제

- 슈퍼 린터 액션 샘플 GitHub
- 배치 빌드 그래프 샘플
- 아마존 CodeGuru 리뷰어 샘플
- AWS Secrets Manager 샘플
- 환경 변수 샘플
- 내보낸 환경 변수 샘플

슈퍼 린터 액션 샘플 GitHub

이 샘플은 프로젝트에 수퍼 GitHub 린터 액션을 추가하는 방법을 보여줍니다. CodeBuild Super-Linter 액션은 코드를 검사하고 코드에 오류가 있는 영역, 형식 지정 문제, 의심스러운 구문을 찾아 결과를 콘솔에 출력합니다. CodeBuild

buildspec 파일의 단계 섹션을 업데이트하여 CodeBuild 프로젝트에 슈퍼 린터 GitHub 액션을 추가할 수 있습니다.

```
version: 0.2
phases:
  build:
    steps:
      - name: Lint Code Base
        uses: github/super-linter@v5
        env:
          VALIDATE_ALL_CODEBASE: true
```

다음과 비슷한 Super-Linter 로그가 표시됩니다.

```
/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.
```

배치 빌드 그래프 샘플

다음 예제는 종속성 체인을 생성하고 steps를 사용하여 명령을 실행하는 빌드 그래프를 정의합니다. 이 예제에서는 종속성이 없으므로 build1이 먼저 실행됩니다. build2에는 build1에 대한 종속 관계가 있으므로 build1이 완료된 후에 build2가 실행됩니다. 자세한 내용은 빌드 그래프을 참조하세요.

```
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
```

아마존 CodeGuru 리뷰어 샘플

Amazon CodeGuru Reviewer는 Java 및 Python 코드에서 문제를 발견하고 해결 방법을 권장합니다. 다음 예제는 CodeGuru Reviewer를 사용하여 전체 리포지토리 분석 코드 검토를 제공합니다. 이러한 코드 검토는 지정된 분기의 모든 코드를 스캔합니다. 자세한 내용은 Amazon CodeGuru Reviewer 사용 설명서의 GitHub 작업을 사용하여 코드 리뷰 생성을 참조하십시오.

```
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**
>
> Amazon S3 버킷은 `codeguru-reviewer-` 접두사로 시작해야 합니다.

다음과 비슷한 로그가 표시됩니다.

```
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 Reviewer 작업이 완료되면 사리프 보고서가 아티팩트로 CodeBuild 생성됩니다. 자세한 내용은 Amazon CodeGuru Reviewer 사용 설명서의 전체 리포지토리 분석을 참조하십시오.

AWS Secrets Manager 샘플

AWS Secrets Manager 수명 주기 전반에 걸쳐 데이터베이스 자격 증명, 애플리케이션 자격 증명, OAuth 토큰, API 키 및 기타 비밀을 관리, 검색 및 교체하는 데 도움이 됩니다. 다음 예제에서는 Secrets Manager를 사용하여 비밀을 정의하고 `steps`를 사용하여 명령을 실행합니다. 자세한 내용은 무엇입니까를 참조하십시오. AWS Secrets Manager AWS Secrets Manager 사용 설명서에서.

```
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
```

다음과 비슷한 로그가 표시됩니다.

```
echo $SECRET_VALUE
env:
  SECRET_VALUE: ***
***
```

환경 변수 샘플

다음 예제에서는 env 시퀀스에 따라 환경 변수를 정의합니다. *S3_BUCKET* 변수는 buildspec에서 정의되고 해당 값으로 *<bucket-name>*이 할당됩니다. 이 변수는 if 조건부에서 일반 환경 변수처럼 달러 기호 ($) 를 사용하여 GitHub Action env 컨텍스트에 액세스하는 방식으로 참조됩니다. 자세한 내용은 env 시퀀스를 참조하세요.

```
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"
```

다음과 비슷한 로그가 표시됩니다.

```
echo "S3 bucket is $S3_BUCKET"
env:
  S3_BUCKET: my-s3-bucket
S3 bucket is my-s3-bucket
```

내보낸 환경 변수 샘플

내보낸 환경 변수는 과 함께 CodePipeline 사용하여 현재 빌드 단계에서 파이프라인의 후속 단계로 환경 변수를 내보내는 데 사용됩니다. 다음 예제에서는 내보낸 환경 변수를 *MY_VARIABLE*이라는 env 시퀀스에 따라 정의하고 *GITHUB_ENV* 환경 파일에 씁니다.

```
version: 0.2
env:
  exported-variables:
    - MY_VARIABLE
phases:
  build:
    steps:
      - run: echo "MY_VARIABLE=my-value" >> $GITHUB_ENV
```

자세한 내용은 AWS CodeBuild API ExportedEnvironmentVariable참조를 참조하십시오.

AWS CodeBuild의 퍼블릭 빌드 프로젝트

AWS CodeBuild에서 빌드 프로젝트의 빌드 결과, 로그, 아티팩트를 일반 사용자에게 공개할 수 있습니다. 이렇게 하면 소스 리포지토리의 기여자가 AWS 계정에 액세스할 필요 없이 빌드의 결과를 보고 아티팩트를 다운로드할 수 있습니다.

프로젝트의 빌드를 공개하면 프로젝트가 프라이빗일 때 실행된 빌드를 포함하여 프로젝트의 모든 빌드 결과, 로그, 아티팩트는 누구나 사용할 수 있습니다. 마찬가지로 퍼블릭 빌드 프로젝트를 프라이빗으로 설정하면 해당 프로젝트의 빌드 결과를 더 이상 공개할 수 없습니다.

프로젝트 빌드 결과의 퍼블릭 가시성을 변경하는 방법에 대한 자세한 내용은 퍼블릭 빌드 액세스 활성화 섹션을 참조하세요.

CodeBuild는 프로젝트에 고유한 퍼블릭 빌드의 URL을 제공합니다. 빌드 프로젝트의 퍼블릭 URL을 얻으려면 다음 절차를 수행하세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.

2. 탐색 창에서 [Build projects]를 선택합니다.
3. 퍼블릭 URL을 가져오려는 빌드 프로젝트의 링크를 선택합니다.
4. 퍼블릭 URL은 구성 섹션의 퍼블릭 프로젝트 URL 필드에 표시됩니다. 링크를 선택하여 URL을 열거나 복사 버튼을 사용하여 URL을 복사할 수 있습니다.

> **Warning**
>
> 프로젝트의 빌드 결과를 공개할 때는 다음 사항을 염두에 두어야 합니다.
>
> - 프로젝트가 프라이빗일 때 실행된 빌드를 포함하여 프로젝트의 모든 빌드 결과, 로그, 아티팩트는 누구나 이용할 수 있습니다.
> - 모든 빌드 로그와 아티팩트는 누구나 이용할 수 있습니다. 환경 변수, 소스 코드 및 기타 중요한 정보가 빌드 로그와 아티팩트에 출력되었을 수 있습니다. 빌드 로그에 출력되는 정보는 주의해야 합니다. 몇 가지 모범 사례는 다음과 같습니다.
>   - 중요한 값(특히 AWS 액세스 키 ID 및 비밀 액세스 키)은 환경 변수에 저장하지 마세요. Amazon EC2 Systems Manager Parameter Store 또는 AWS Secrets Manager을 사용하여 중요한 값을 저장하는 것이 좋습니다.
>   - webhook를 최대한 안전하게 보호하려면 webhook 사용 모범 사례에 따라 빌드를 트리거할 수 있는 엔터티를 제한하고, buildspec을 프로젝트 자체에 저장하지 마세요.
> - 악의적인 사용자는 퍼블릭 빌드를 사용하여 악성 아티팩트를 배포할 수 있습니다. 프로젝트 관리자는 모든 풀 요청을 검토하여 풀 요청이 합법적인 변경인지 확인하는 것이 좋습니다. 또한 체크섬으로 모든 아티팩트의 유효성을 검사하여 올바른 아티팩트가 다운로드되고 있는지 확인하는 것이 좋습니다.

AWS CodeBuild의 빌드 작업

빌드는 AWS CodeBuild가 수행하는 일련의 작업 세트로, 입력 결과물(예: Java 클래스 파일의 모음)를 토대로 출력 결과물(예: JAR 파일)를 생성합니다.

빌드를 여러 개 실행할 경우 다음 규칙이 적용됩니다.

- 가능한 경우 동시에 빌드가 실행됩니다. 동시에 실행할 수 있는 최대 빌드 수는 다를 수 있습니다. 자세한 내용은 AWS CodeBuild에 대한 할당량 섹션을 참조하세요.

- 빌드 프로젝트에 동시 빌드 제한이 설정되어 있는 경우 실행 중인 빌드 수가 프로젝트의 동시 빌드 제한에 도달하면 빌드에서 오류가 반환됩니다. 자세한 내용은 동시 빌드 제한 활성화를 참조하세요.
- 빌드 프로젝트에 동시 빌드 제한이 설정되어 있지 않은 경우 실행 중인 빌드 수가 플랫폼 및 컴퓨팅 유형의 동시 빌드 제한에 도달하면 빌드가 대기열에 추가됩니다. 대기열의 최대 빌드 수는 동시 빌드 수 한도의 다섯 배입니다. 자세한 내용은 AWS CodeBuild에 대한 할당량 섹션을 참조하세요.

  제한 시간 값에 지정한 시간(분)이 지난 후 시작되지 않은 대기열의 빌드는 대기열에서 삭제됩니다. 기본 제한 시간 값은 8분입니다. 빌드를 실행할 때 빌드 대기열 제한 시간을 5분에서 8시간까지 원하는 대로 지정할 수 있습니다. 자세한 내용은 AWS CodeBuild에서 빌드 실행 섹션을 참조하세요.

  대기열에 추가된 빌드가 시작되는 순서는 예측할 수 없습니다.

> **Note**
>
> 1년치 빌드의 기록에 액세스할 수 있습니다.

빌드 작업을 수행할 때 이 작업을 수행할 수 있습니다.

주제

- AWS CodeBuild에서 빌드 실행
- AWS CodeBuild의 빌드 세부 정보 보기
- AWS CodeBuild에서 빌드 ID 목록 보기
- AWS CodeBuild에서 빌드 프로젝트의 빌드 ID 목록 보기
- AWS CodeBuild에서 빌드 중지
- AWS CodeBuild에서 배치 빌드 중지
- AWS CodeBuild에서 빌드 재시도
- Session Manager에서 실행 중인 빌드 보기
- AWS CodeBuild에서 빌드 삭제

AWS CodeBuild에서 빌드 실행

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서 빌드를 실행할 수 있습니다.

주제

- 빌드 실행(콘솔)
- 빌드 실행(AWS CLI)
- 배치 빌드 실행(AWS CLI)
- 빌드 실행 자동 시작(AWS CLI)
- 빌드 실행 자동 중지(AWS CLI)
- 빌드 실행(AWS SDK)

빌드 실행(콘솔)

AWS CodePipeline을 사용하여 CodeBuild에서 빌드를 실행하려면 다음 단계를 건너뛰고 CodePipeline 함께 사용 CodeBuild 섹션의 지침을 따르세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다.
3. 빌드 프로젝트 목록에서 빌드 프로젝트를 선택합니다.
4. 기본 빌드 프로젝트 설정으로 빌드를 실행하거나 이 빌드의 빌드 설정만 재정의할 수 있습니다.

   a. 기본 빌드 프로젝트 설정으로 빌드를 실행하려면 빌드 시작을 선택합니다. 빌드가 즉시 시작됩니다.

   b. 기본 빌드 프로젝트 설정을 재정의하려면 재정의로 빌드 시작을 선택합니다. 빌드 시작 페이지에서 다음을 재정의할 수 있습니다.

      - 빌드 구성
      - 소스
      - 환경 변수 재정의

      고급 재정의를 더 선택해야 하는 경우 고급 빌드 재정의를 선택합니다. 이 페이지에서 다음을 재정의할 수 있습니다.

      - 빌드 구성
      - 소스
      - Environment
      - Buildspec

- 아티팩트
- 로그

재정의를 선택했으면 빌드 시작을 선택합니다.

이 빌드에 대한 자세한 정보는 빌드 세부 정보 보기(콘솔) 단원을 참조하십시오.

빌드 실행(AWS CLI)

> **Note**
>
> CodePipeline을 사용하여 AWS CodeBuild에서 빌드를 실행하려면 다음 단계를 건너뛰고 CodeBuild를 사용하는 파이프라인 생성(AWS CLI) 섹션의 지침을 따르세요.
> AWS CLI와 CodeBuild를 함께 사용하는 방법에 대한 자세한 내용은 명령줄 참조 섹션을 참조하세요.

1. 다음 중 한 방법으로 `start-build` 명령을 실행합니다.

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

   빌드 입력 결과물의 최신 버전과 빌드 프로젝트의 기존 설정을 사용하는 빌드를 실행하려면 이 방법을 사용합니다.

   ```
   aws codebuild start-build --generate-cli-skeleton
   ```

   이전 버전의 빌드 입력 결과물을 사용하여 빌드를 실행하려는 경우 또는 빌드 출력 결과물, 환경 변수, buildspec 또는 기본 빌드 제한 시간의 설정을 재정의하려는 경우 이 방법을 사용합니다.

2. start-build 명령을 `--project-name` 옵션과 함께 실행하는 경우 *<project-name>*을 빌드 프로젝트의 이름으로 바꾼 다음 이 절차의 6단계로 이동합니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기 단원을 참조하십시오.
3. start-build 명령을 `--idempotency-token` 옵션과 함께 실행하면 고유의 대소문자 구분 식별자인 토큰이 `start-build` 요청에 포함됩니다. 토큰은 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 `start-build` 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.

4. start-build 명령을 `--generate-cli-skeleton` 옵션과 함께 실행하는 경우 JSON 형식 데이터가 출력에 표시됩니다. AWS CLI가 설치된 로컬 컴퓨터 또는 인스턴스의 위치에 있는 파일(예: *start-build.json*)에 데이터를 복사합니다. 복사된 데이터를 다음 형식으로 수정한 다음 결과를 저장합니다.

```
{
  "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"
}
```

다음과 같이 자리 표시자를 바꿉니다.

- *projectName*: 필수 문자열입니다. 이 빌드에 사용할 빌드 프로젝트의 이름입니다.
- *sourceVersion*: 선택적 문자열입니다. 빌드할 소스 코드의 버전으로, 다음과 같습니다.
  - Amazon S3의 경우, 빌드하려는 입력 ZIP 파일의 버전에 해당하는 버전 ID입니다. *sourceVersion*을 지정하지 않은 경우 최신 버전이 사용됩니다.
  - CodeCommit의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID입니다. *sourceVersion*을 지정하지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다. (*sourceVersion*의 태그 이름은 지정할 수 없지만, 태그의 커밋 ID는 지정할 수 있습니다.)
  - GitHub의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 풀 요청 ID, 분기 이름 또는 태그 이름입니다. 풀 요청 ID가 지정된 경우 `pr/`*pull-request-ID* 형식을 사용해야 합니다(예: `pr/25`). 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. *sourceVersion*을 지정하지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
  - Bitbucket의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 분기 이름 또는 태그 이름입니다. 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. *sourceVersion*을 지정하지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
- 다음 자리 표시자는 `artifactsOverride`용입니다.
  - *type*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 유형이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 유형입니다.
  - *location*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 위치가 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 위치입니다.
  - *path*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 경로가 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 경로입니다.
  - *namespaceType*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 유형이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 유형입니다.
  - *name*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 이름이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 이름입니다.
  - *packaging*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 패키징이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 패키징입니다.
- *buildspecOverride*: 선택 사항입니다. 빌드 프로젝트에서 정의된 buildspec 선언이 아닌 이 빌드에서만 사용되는 buildspec 선언입니다. 이 값이 설정된 경우 인라인 buildspec 정의, 내장된 `CODEBUILD_SRC_DIR` 환경 변수의 값에 상대적인 대체 buildspec 파일 또는 S3 버킷의 경로가 될 수 있습니다. S3 버킷은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을

사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`). 이 값을 제공하지 않거나 빈 문자열로 설정하는 경우 소스 코드에 루트 디렉터리의 `buildspec.yml` 파일이 포함되어 있어야 합니다. 자세한 내용은 buildspec 파일 이름 및 스토리지 위치 섹션을 참조하세요.

- 다음 자리 표시자는 `cacheOverride`용입니다.
  - *`cacheOverride-location`*: 선택 사항입니다. 빌드 프로젝트에서 지정된 `ProjectCache` 객체를 재정의하는 이 빌드에 대한 `ProjectCache` 객체의 위치입니다. `cacheOverride`는 선택 사항이며 `ProjectCache` 객체를 가져옵니다. `location`는 `ProjectCache` 객체에서 필수입니다.
  - *`cacheOverride-type`*: 선택 사항입니다. 이 빌드에 대한 `ProjectCache` 객체의 유형으로, 빌드 프로젝트에서 지정된 `ProjectCache` 객체를 재정의합니다. `cacheOverride`는 선택 사항이며 `ProjectCache` 객체를 가져옵니다. `type`는 `ProjectCache` 객체에서 필수입니다.
- *`certificateOverride`*: 선택 사항입니다. 빌드 프로젝트에서 지정된 인증서를 재정의하는 이 빌드에 대한 인증서의 이름입니다.
- *`environmentTypeOverride`*: 선택 사항입니다. 빌드 프로젝트에서 지정된 컨테이너를 재정의하는 이 빌드의 컨테이너 유형입니다. 현재 유효한 문자열은 `LINUX_CONTAINER`입니다.
- 다음 자리 표시자는 `environmentVariablesOverride`용입니다.
  - *`environmentVariablesOverride-name`*: 선택 사항입니다. 이 빌드에서 값을 재정의하려는 빌드 프로젝트의 환경 변수 이름입니다.
  - *`environmentVariablesOverride-type`*: 선택 사항입니다. 이 빌드에서 값을 재정의하려는 빌드 프로젝트의 환경 변수 유형입니다.
  - *`environmentVariablesValue`*: 선택 사항입니다. 이 빌드에서 재정의하려는 빌드 프로젝트에 정의된 환경 변수 값입니다.
- *`gitCloneDepthOverride`*: 선택 사항입니다. 이 빌드에서 값을 재정의하려는 빌드 프로젝트의 Git clone depth 값입니다. 소스 유형이 Amazon S3일 경우 이 값이 지원되지 않습니다.
- *`imageOverride`*: 선택 사항입니다. 빌드 프로젝트에서 지정된 이미지를 재정의하는 이 빌드에 대한 이미지의 이름입니다.
- *`idempotencyToken`*: 선택 사항입니다. 빌드 요청이 idempotent임을 지정하기 위해 토큰으로 제공되는 문자열입니다. 64자 이하의 모든 문자열을 선택할 수 있습니다. 토큰은 start-build 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 start-build 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.
- *`insecureSslOverride`*: 빌드 프로젝트에서 지정된 안전하지 않은 TLS 설정을 재정의할지 여부를 지정하는 선택적 부울입니다. 안전하지 않은 TLS 설정은 프로젝트 소스 코드에 연결하

는 동안 TLS 경고를 무시할지 여부를 결정합니다. 이 설정은 빌드의 소스가 GitHub Enterprise Server인 경우에만 적용됩니다.

- *privilegedModeOverride*: 선택적 부울입니다. true로 설정하면 빌드는 빌드 프로젝트에서 권한이 있는 모드를 재정의합니다.
- *queuedTimeoutInMinutesOverride*: 빌드 대기 시간이 얼마나 지나야 시간 초과로 처리되는지를 지정하는 정수(분)입니다(선택 사항). 최솟값은 5분이며, 최댓값은 480분(8시간)입니다.
- *reportBuildStatusOverride*: 빌드의 시작 및 완료 상태를 소스 공급자에게 보낼지 여부를 지정하는 선택적 부울입니다. GitHub, GitHub Enterprise Server 또는 Bitbucket이 아닌 소스 공급자로 설정하는 경우, invalidInputException이 발생합니다.
- *sourceAuthOverride*: 선택적 문자열입니다. 빌드 프로젝트에서 정의된 권한 부여를 재정의하는 이 빌드에 대한 권한 부여 유형입니다. 이 재정의는 빌드 프로젝트의 소스가 Bitbucket 또는 GitHub인 경우에만 적용됩니다.
- *sourceLocationOverride*: 선택적 문자열입니다. 빌드 프로젝트에서 정의된 위치의 소스 위치를 이 빌드에 대해 재정의하는 위치입니다.
- *serviceRoleOverride*: 선택 사항 문자열. 빌드 프로젝트에서 지정된 서비스 역할을 재정의하는 이 빌드에 대한 서비스 역할의 이름입니다.
- *sourceTypeOverride*: 선택적 문자열입니다. 빌드 프로젝트에서 정의된 소스 입력을 재정의하는 이 빌드에 대한 소스 입력 유형입니다. 유효한 문자열은 `NO_SOURCE`, `CODECOMMIT`, `CODEPIPELINE`, `GITHUB`, `S3`, `BITBUCKET` 및 `GITHUB_ENTERPRISE`입니다.
- *timeoutInMinutesOverride*: 선택적 숫자입니다. 빌드 프로젝트에서 정의된 빌드 제한 시간(분)이 아닌 이 빌드에서만 사용되는 빌드 제한 시간(분)입니다.

AWS 액세스 키 ID, AWS 비밀 액세스 키 또는 암호와 같은 중요한 값을 가진 환경 변수는 Amazon EC2 Systems Manager Parameter Store 또는 에 파라미터로 저장하는 것이 좋습니다. CodeBuild는 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터 이름이 `/CodeBuild/`(예: `/CodeBuild/dockerLoginPassword`)로 시작하는 경우에만 해당 파라미터를 사용할 수 있습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택한 후 지침에 따릅니다. 경우에 따라 대화 상자의 KMS 키에 해당 계정의 AWS KMS 키에 대한 ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는 이 키를 사용하여 저장 시 파라미터의 값을 암호화하고 검색 시 암호를 해독합니다. CodeBuild 콘솔을 사용하여 파라미터를 생성하는 경우 콘솔은 `/CodeBuild/`로 파라미터를 시작합니다. 하지만 Amazon EC2 Systems Manager Parameter Store 콘솔을 사용하여 파라미터를 생성하는 경우, `/CodeBuild/`로 파라미터 이름을 시작해야 하고 유형을 보안 문자열로 설정해야 합

니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 AWS Systems Manager 파라미터 스토어 및 연습: 문자열 파라미터 생성 및 테스트(콘솔)을 참조하세요.

빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `ssm:GetParameters` 작업을 허용해야 합니다. 앞에서 해당 계정에 새로운 서비스 역할 생성을 선택한 경우 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 자동으로 포함합니다. 하지만 [Choose an existing service role from your account]를 선택한 경우 이 작업을 서비스 역할에 별도로 포함해야 합니다.

사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.

`CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.

여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 환경 변수 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.
- buildspec 파일 선언의 값이 가장 낮은 우선 순위를 갖습니다.

이 자리 표시자의 유효한 값에 대한 자세한 정보는 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오. 빌드 프로젝트의 최신 설정 목록은 빌드 프로젝트 세부 정보 보기 단원을 참조하십시오.

5. 방금 저장한 파일이 들어 있는 디렉터리로 전환한 다음, `start-build` 명령을 다시 실행합니다.

```
aws codebuild start-build --cli-input-json file://start-build.json
```

6. 이 명령이 제대로 실행되면 빌드를 실행하려면 절차에 설명된 것과 유사한 데이터가 출력에 표시됩니다.

이 빌드에 대한 자세한 정보를 보려면 출력에서 `id` 값을 적어 둔 다음 빌드 세부 정보 보기(AWS CLI) 단원을 참조하십시오.

배치 빌드 실행(AWS CLI)

1. 다음 중 한 방법으로 `start-build-batch` 명령을 실행합니다.

   ```
   aws codebuild start-build-batch --project-name <project-name>
   ```

   빌드 입력 결과물의 최신 버전과 빌드 프로젝트의 기존 설정을 사용하는 빌드를 실행하려면 이 방법을 사용합니다.

   ```
   aws codebuild start-build-batch --generate-cli-skeleton > <json-file>
   ```

   이전 버전의 빌드 입력 결과물을 사용하여 빌드를 실행하려는 경우 또는 빌드 출력 결과물, 환경 변수, buildspec 또는 기본 빌드 제한 시간의 설정을 재정의하려는 경우 이 방법을 사용합니다.

2. start-build-batch 명령을 `--project-name` 옵션과 함께 실행하는 경우 *<project-name>*을 빌드 프로젝트의 이름으로 바꾼 다음 이 절차의 6단계로 이동합니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기 단원을 참조하십시오.

3. start-build-batch 명령을 `--idempotency-token` 옵션과 함께 실행하면 고유의 대소문자 구분 식별자 또는 토큰이 `start-build-batch` 요청에 포함됩니다. 토큰은 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 `start-build-batch` 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.

4. start-build-batch 명령을 `--generate-cli-skeleton` 옵션과 함께 실행하는 경우 JSON 형식 데이터가 *<json-file>* 파일에 출력됩니다. 이 파일은 start-build 명령으로 생성된 스켈레톤과 비슷하지만 다음 객체가 추가되었습니다. 일반 객체에 대한 자세한 내용은 빌드 실행(AWS CLI) 섹션을 참조하세요.

   이 파일을 수정하여 빌드 재정의를 추가하고 결과를 저장하세요.

   ```
   "buildBatchConfigOverride": {
     "combineArtifacts": combineArtifacts,
     "restrictions": {
       "computeTypesAllowed": [
         allowedComputeTypes
       ],
       "maximumBuildsAllowed": maximumBuildsAllowed
     },
     "serviceRole": "batchServiceRole",
     "timeoutInMins": batchTimeout
   }
   ```

`buildBatchConfigOverride` 객체는 이 빌드에 대한 배치 빌드 구성 재정의를 포함하는 ProjectBuildBatchConfig 구조입니다.

combineArtifacts

배치 빌드의 빌드 아티팩트를 단일 아티팩트 위치로 결합할지 여부를 지정하는 부울입니다.

allowedComputeTypes

배치 빌드에 허용되는 컴퓨팅 유형을 지정하는 문자열 배열입니다. 이러한 값은 빌드 환경 컴퓨팅 유형를 참조하세요.

maximumBuildsAllowed

허용되는 최대 빌드 수를 지정합니다.

batchServiceRole

배치 빌드 프로젝트에 대한 서비스 역할 ARN을 지정합니다.

batchTimeout

배치 빌드를 완료해야 하는 최대 시간(분)을 지정합니다.

5. 방금 저장한 파일이 들어 있는 디렉터리로 전환한 다음, `start-build-batch` 명령을 다시 실행합니다.

```
aws codebuild start-build-batch --cli-input-json file://start-build.json
```

6. 성공하면 BuildBatch 객체의 JSON 표현이 콘솔 출력에 나타납니다. 이 데이터의 예는 StartBuildBatch 응답 구문을 참조하세요.

빌드 실행 자동 시작(AWS CLI)

소스 코드가 GitHub 또는 GitHub Enterprise Server 리포지토리에 저장되는 경우, GitHub Webhook을 사용하면 코드 변경이 리포지토리로 푸시될 때마다 AWS CodeBuild가 소스 코드를 다시 빌드하도록 할 수 있습니다.

다음과 같이 create-webhook 명령을 실행합니다.

```
aws codebuild create-webhook --project-name <project-name>
```

*<project-name>*은 다시 빌드할 소스 코드가 포함되어 있는 빌드 프로젝트의 이름입니다.

GitHub의 경우, 다음과 비슷한 정보가 결과에 나타납니다.

```
{
  "webhook": {
    "url": "<url>"
  }
}
```

*<url>*은 GitHub Webhook에 대한 URL입니다.

GitHub Enterprise Server의 경우, 다음과 비슷한 정보가 결과에 나타납니다.

1. 출력에서 보안 키 및 페이로드 URL을 복사합니다. 이들 값은 GitHub Enterprise Server에 Webhook를 추가할 때 필요합니다.
2. GitHub Enterprise Server에서 CodeBuild 프로젝트가 저장된 리포지토리를 선택합니다. 설정, Hooks & services(후크 및 서비스), Add webhook(webhook 추가)를 차례로 선택합니다.
3. 페이로드 URL 및 보안 키를 입력하고 그 외 필드에 대해서는 기본값을 수락한 다음 [Add webhook]를 선택합니다.

빌드 실행 자동 중지(AWS CLI)

소스 코드가 GitHub 또는 GitHub Enterprise Server 리포지토리에 저장되는 경우, GitHub Webhook을 설정하여 코드 변경이 리포지토리로 푸시될 때마다 AWS CodeBuild가 소스 코드를 다시 빌드하도록 할 수 있습니다. 자세한 내용은 빌드 실행 자동 시작(AWS CLI) 섹션을 참조하세요.

이 동작을 활성화한 경우 다음과 같이 `delete-webhook` 명령을 실행하여 해제할 수 있습니다.

```
aws codebuild delete-webhook --project-name <project-name>
```

- *<project-name>*은 다시 빌드할 소스 코드가 포함되어 있는 빌드 프로젝트의 이름입니다.

이 명령이 제대로 실행되면 출력에 정보나 오류가 표시되지 않습니다.

> **Note**
>
> 이렇게 하면 Webhook가 CodeBuild 프로젝트에서만 삭제됩니다. Webhook를 GitHub 또는 GitHub Enterprise Server 리포지토리에서도 삭제해야 합니다.

빌드 실행(AWS SDK)

CodePipeline을 사용하여 AWS CodeBuild에서 빌드를 실행하려면 다음 단계를 건너뛰고 AWS CodePipeline를 AWS CodeBuild와 함께 사용하여 코드 테스트 및 빌드 실행 섹션의 지침을 따르세요.

AWS SDK에서 CodeBuild를 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 섹션을 참조하세요.

AWS CodeBuild의 빌드 세부 정보 보기

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서 관리하는 빌드에 대한 세부 정보를 볼 수 있습니다.

주제

- 빌드 세부 정보 보기(콘솔)
- 빌드 세부 정보 보기(AWS CLI)
- 빌드 세부 정보 보기(AWS SDK)
- 빌드 단계 진행

빌드 세부 정보 보기(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 다음 중 하나를 수행하세요.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록의 빌드 실행 열에서 빌드에 대한 링크를 선택합니다.
   - 탐색 창에서 [Build projects]를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 해당하는 링크를 선택합니다. 그런 다음 빌드 목록의 빌드 실행 열에서 빌드에 대한 링크를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 10개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.

빌드 세부 정보 보기(AWS CLI)

AWS CLI와 AWS CodeBuild를 함께 사용하는 방법에 대한 자세한 정보는 명령줄 참조 단원을 참조하십시오.

batch-get-builds 명령을 실행합니다.

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

다음과 같이 자리 표시자를 바꿉니다.

- *ids*: 필수 문자열입니다. 세부 정보를 보려는 하나 이상의 빌드 ID입니다. 둘 이상의 빌드 ID를 지정하려면 각 빌드 ID를 공백을 사용하여 구분해야 합니다. 최대 100개의 빌드 ID를 지정할 수 있습니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

예를 들면 다음 명령을 실행하는 경우

```
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
```

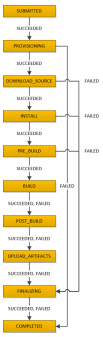
명령이 제대로 실행되면 요약된 빌드 정보를 보려면 단원에 설명된 것과 유사한 데이터가 출력에 표시됩니다.

빌드 세부 정보 보기(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

빌드 단계 진행

AWS CodeBuild의 빌드는 다음 단계로 진행됩니다.



> **Important**
>
> `BUILD` 단계가 실패하더라도 `UPLOAD_ARTIFACTS` 단계는 항상 시도됩니다.

AWS CodeBuild에서 빌드 ID 목록 보기

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서 관리하는 빌드의 빌드 ID 목록을 볼 수 있습니다.

주제

- 빌드 ID 목록 보기(콘솔)
- 빌드 ID 목록 보기(AWS CLI)
- 배치 빌드 ID 목록 보기(AWS CLI)
- 빌드 ID 목록 보기(AWS SDK)

빌드 ID 목록 보기(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build history]를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 10개만 표시됩니다. 더 많은 빌드를 보려면 기어 아이콘을 선택하고 페이지당 빌드 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

빌드 ID 목록 보기(AWS CLI)

AWS CLI와 CodeBuild를 함께 사용하는 방법에 대한 자세한 내용은 명령줄 참조 섹션을 참조하세요.

- list-builds 명령을 실행합니다.

```
aws codebuild list-builds --sort-order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *sort-order*: 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING이 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild list-builds --sort-order ASCENDING
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "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"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full
 token has been omitted for brevity...MzY2OA==
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "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"
  ]
}
```

배치 빌드 ID 목록 보기(AWS CLI)

AWS CLI와 CodeBuild를 함께 사용하는 방법에 대한 자세한 내용은 명령줄 참조 섹션을 참조하세요.

- list-build-batches 명령을 실행합니다.

```
aws codebuild list-build-batches --sort-order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *sort-order*: 배치 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING이 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild list-build-batches --sort-order ASCENDING
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "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"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-build-batches --sort-order ASCENDING --next-token 4AEA6u7J...The
 full token has been omitted for brevity...MzY2OA==
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "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"
  ]
}
```

빌드 ID 목록 보기(AWS SDK)

AWS SDK에서 CodeBuild를 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 섹션을 참조하세요.

AWS CodeBuild에서 빌드 프로젝트의 빌드 ID 목록 보기

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 CodeBuild에서 관리하는 빌드 프로젝트의 빌드 ID 목록을 볼 수 있습니다.

주제

빌드 프로젝트의 빌드 ID 목록 보기(콘솔)

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.

빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

AWS CLI와 AWS CodeBuild를 함께 사용하는 방법에 대한 자세한 정보는 명령줄 참조 단원을 참조하십시오.

다음과 같이 list-builds-for-project 명령을 실행합니다.

```
aws codebuild list-builds-for-project --project-name project-name --sort-order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *project-name*: 빌드 ID를 나열할 빌드 프로젝트의 이름을 나타내는 데 사용되는 필수 문자열입니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.
- *sort-order*: 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING이 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 반환되는 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음과 비슷한 명령을 실행하는 경우

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-
order ASCENDING
```

다음과 같은 결과가 출력에 나타날 수 있습니다.

```
{
  "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"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
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==
```

출력에서 다음과 같은 결과를 볼 수 있습니다.

```
{
  "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"
  ]
}
```

빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI)

AWS CLI와 AWS CodeBuild를 함께 사용하는 방법에 대한 자세한 정보는 명령줄 참조 단원을 참조하십시오.

다음과 같이 list-build-batches-for-project 명령을 실행합니다.

```
aws codebuild list-build-batches-for-project --project-name project-name --sort-
order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *project-name*: 빌드 ID를 나열할 빌드 프로젝트의 이름을 나타내는 데 사용되는 필수 문자열입니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.
- *sort-order*: 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING이 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 반환되는 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음과 비슷한 명령을 실행하는 경우

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project --
sort-order ASCENDING
```

다음과 같은 결과가 출력에 나타날 수 있습니다.

```
{
  "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"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
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==
```

출력에서 다음과 같은 결과를 볼 수 있습니다.

```
{
  "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"
  ]
}
```

빌드 프로젝트의 빌드 ID 목록 보기(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

AWS CodeBuild에서 빌드 중지

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 AWS CodeBuild에서 빌드를 중지할 수 있습니다.

주제

- 빌드 중지(콘솔)
- 빌드 중지(AWS CLI)
- 빌드 중지(AWS SDK)

빌드 중지(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.

2. 다음 중 하나를 수행하세요.

   - ***build-project-name*:*build-ID*** 페이지가 표시되면 빌드 중지를 선택합니다.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.
   - 탐색 창에서 [Build projects]를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 대한 링크를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.
> AWS CodeBuild에서 빌드를 성공적으로 중지할 수 없는 경우(예: 빌드 프로세스가 이미 완료된 경우) 중지 버튼이 비활성화되거나 표시되지 않을 수 있습니다.

빌드 중지(AWS CLI)

- stop-build 명령을 실행합니다.

```
aws codebuild stop-build --id id
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *id*: 필수 문자열입니다. 중지할 빌드의 ID입니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

AWS CodeBuild에서 빌드를 성공적으로 중지한 경우 출력에서 `build` 객체의 `buildStatus` 값은 `STOPPED`입니다.

CodeBuild가 빌드를 성공적으로 중지할 수 없는 경우(예: 빌드가 이미 완료된 경우) 출력에서 `build` 객체의 `buildStatus` 값이 최종 빌드 상태(예: `SUCCEEDED`)입니다.

빌드 중지(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

AWS CodeBuild에서 배치 빌드 중지

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 AWS CodeBuild에서 배치 빌드를 중지할 수 있습니다.

주제

- 배치 빌드 중지(콘솔)
- 배치 빌드 중지(AWS CLI)
- 배치 빌드 중지(AWS SDK)

배치 빌드 중지(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 다음 중 하나를 수행하세요.
   - ***build-project-name*:*build-ID*** 페이지가 표시되면 빌드 중지를 선택합니다.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.
   - 탐색 창에서 [Build projects]를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 대한 링크를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.
> AWS CodeBuild에서 배치 빌드를 성공적으로 중지할 수 없는 경우(예: 빌드 프로세스가 이미 완료된 경우) 빌드 중지 버튼이 비활성화됩니다.

배치 빌드 중지(AWS CLI)

- stop-build-batch 명령을 실행합니다.

```
aws codebuild stop-build-batch --id <batch-build-id>
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *<batch-build-id>*: 필수 문자열입니다. 중지할 배치 빌드의 식별자입니다. 배치 빌드 식별자 목록을 가져오려면 다음 주제를 참조하세요.
  - 배치 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI)

AWS CodeBuild에서 배치 빌드를 성공적으로 중지한 경우 출력에서 `buildBatch` 객체의 `buildBatchStatus` 값은 `STOPPED`입니다.

CodeBuild가 배치 빌드를 성공적으로 중지할 수 없는 경우(예: 배치 빌드가 이미 완료된 경우) 출력에서 `buildBatch` 객체의 `buildBatchStatus` 값이 최종 빌드 상태(예: `SUCCEEDED`)입니다.

배치 빌드 중지(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

AWS CodeBuild에서 빌드 재시도

AWS CodeBuild 콘솔, AWS CLI 또는 AWS SDK를 사용하여 AWS CodeBuild에서 단일 빌드나 배치 빌드를 재시도할 수 있습니다.

주제

- 빌드 다시 시도(콘솔)
- 빌드 재시도(AWS CLI)
- 빌드 재시도(AWS SDK)

빌드 다시 시도(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.

2. 다음 중 하나를 수행하세요.

   - ***build-project-name*:*build-ID*** 페이지가 표시되면 빌드 재시도를 선택합니다.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 재시도를 선택합니다.
   - 탐색 창에서 [Build projects]를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 대한 링크를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 재시도를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.

빌드 재시도(AWS CLI)

- retry-build 명령을 실행합니다.

```
aws codebuild retry-build --id <build-id> --idempotency-token <idempotencyToken>
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *<build-id>*: 필수 문자열입니다. 재시도할 빌드 또는 배치 빌드의 ID입니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 배치 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI)
- `--idempotency-token`: 선택 사항. retry-build 명령을 옵션과 함께 실행하면 고유의 대소문자 구분 식별자 또는 토큰이 `retry-build` 요청에 포함됩니다. 토큰은 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 `retry-build` 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.

빌드 재시도(AWS SDK)

AWS CodeBuild와 AWS SDK를 함께 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 단원을 참조하십시오.

Session Manager에서 실행 중인 빌드 보기

AWS CodeBuild에서는 실행 중인 빌드를 일시 중지한 다음 AWS Systems Manager 세션 관리자를 사용하여 빌드 컨테이너에 연결하고 컨테이너의 상태를 볼 수 있습니다.

> **Note**
>
> 이 기능은 Windows 환경에서는 제공되지 않습니다.

주제

필수 조건

Session Manager를 빌드 세션과 함께 사용할 수 있게 하려면 빌드의 세션 연결을 활성화해야 합니다. 다음과 같은 두 가지 사전 요구 사항이 있습니다.

- CodeBuild Linux 표준 큐레이션된 이미지에는 이미 SSM 에이전트가 설치되어 있고 SSM 에이전트가 활성화되어 있습니다. ContainerMode

  빌드에 사용자 지정 이미지를 사용하는 경우 다음을 수행하세요.

  1. SSM Agent 설치 자세한 내용은 AWS Systems Manager 사용 설명서의 Linux용 EC2 인스턴스에 수동으로 SSM Agent 설치를 참조하세요. SSM Agent는 3.0.1295.0 이상 버전이어야 합니다.
  2. https://github.com/aws/ aws-codebuild-docker-images /blob/master/ubuntu/standard/5.0/ .json 파일을 이미지의 amazon-ssm-agent 디렉터리에 복사합니다. `/etc/amazon/ssm/` 이렇게 하면 SSM 에이전트에서 컨테이너 모드가 활성화됩니다.

> **Note**
>
> 이 기능이 예상대로 작동하려면 사용자 지정 이미지에 최근 업데이트한 SSM 에이전트가 필요합니다.

- 서비스 CodeBuild 역할에는 다음과 같은 SSM 정책이 있어야 합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "ssmmessages:CreateControlChannel",
    "ssmmessages:CreateDataChannel",
    "ssmmessages:OpenControlChannel",
    "ssmmessages:OpenDataChannel"
  ],
  "Resource": "*"
}
```

  빌드를 시작할 때 CodeBuild 콘솔이 이 정책을 서비스 역할에 자동으로 연결하도록 할 수 있습니다. 또는 이 정책을 서비스 역할에 수동으로 연결할 수도 있습니다.

- Systems Manager 기본 설정에서 감사 및 로깅 세션 활동을 활성화한 경우 CodeBuild 서비스 역할에도 추가 권한이 있어야 합니다. 권한은 로그가 저장되는 위치에 따라 다릅니다.

  CloudWatch 로그

  CloudWatch 로그를 사용하여 로그를 저장하는 경우 CodeBuild 서비스 역할에 다음 권한을 추가하십시오.

```
{
  "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

Amazon S3를 사용하여 로그를 저장하는 경우 CodeBuild 서비스 역할에 다음 권한을 추가하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetEncryptionConfiguration",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket-name>",
        "arn:aws:s3:::<bucket-name>/*"
      ]
    }
  ]
}
```

자세한 내용은 AWS Systems Manager 사용 설명서의 세션 활동 감사 및 로깅을 참조하세요.

빌드 일시 중지

빌드를 일시 중지하려면 buildspec 파일의 모든 빌드 단계에 codebuild-breakpoint 명령을 삽입합니다. 이 시점에서 빌드가 일시 중지되므로 빌드 컨테이너에 연결하여 컨테이너를 현재 상태로 볼 수 있습니다.

예를 들어, buildspec 파일의 빌드 단계에 다음을 추가합니다.

```
phases:
```

```
pre_build:
  commands:
    - echo Entered the pre_build phase...
    - echo "Hello World" > /tmp/hello-world
    - codebuild-breakpoint
```

이 코드는 /tmp/hello-world 파일을 생성한 다음, 이 시점에서 빌드를 일시 중지합니다.

빌드를 시작합니다.

Session Manager를 빌드 세션과 함께 사용할 수 있게 하려면 빌드의 세션 연결을 활성화해야 합니다. 이렇게 하려면 빌드를 시작할 때 다음 절차를 따르세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트를 선택한 다음, 재정의로 빌드 시작을 선택합니다.
3. Advanced build overrides(고급 빌드 재정의)를 선택합니다.
4. 환경 섹션에서 세션 연결 활성화 옵션을 선택합니다. 이 옵션을 선택하지 않으면 codebuild-breakpoint 및 codebuild-resume 명령이 모두 무시됩니다.
5. 다른 사항을 원하는 대로 변경하고 빌드 시작을 선택합니다.
6. 콘솔에서 빌드 상태를 모니터링합니다. 세션을 사용할 수 있게 되면 빌드 상태 섹션에 AWS Session Manager 링크가 나타납니다.

빌드 컨테이너에 연결

다음 두 방법 중 하나로 빌드 컨테이너에 연결할 수 있습니다.

CodeBuild 콘솔

웹 브라우저에서 AWS Session Manager 링크를 열어 빌드 컨테이너에 연결합니다. 빌드 컨테이너를 탐색하고 제어할 수 있는 터미널 세션이 열립니다.

AWS CLI

> **Note**
>
> 이 절차를 수행하려면 로컬 컴퓨터에 Session Manager 플러그인이 설치되어 있어야 합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS CLI용 세션 관리자 플러그인 설치를 참조하십시오.

1. 빌드 ID로 batch-get-builds API를 호출하여 세션 대상 식별자를 비롯한 빌드 관련 정보를 가져옵니다. 세션 대상 식별자 속성 이름은 aws 명령의 출력 유형에 따라 달라집니다. 바로 이때문에 `--output json`이 명령에 추가된 것입니다.

   ```
   aws codebuild batch-get-builds --ids <buildID> --region <region> --output json
   ```

2. `sessionTarget` 속성 값을 복사합니다. `sessionTarget` 속성 이름은 aws 명령의 출력 유형에 따라 달라질 수 있습니다. 바로 이때문에 `--output json`이 이전 단계의 명령에 추가된 것입니다.
3. 다음 명령을 사용하여 빌드 컨테이너에 연결합니다.

   ```
   aws ssm start-session --target <sessionTarget> --region <region>
   ```

이 예제에서는 `/tmp/hello-world` 파일이 존재하고 텍스트 `Hello World`가 포함되어 있는지 확인하세요.

빌드 재개

빌드 컨테이너 검사를 마친 후 컨테이너 쉘에서 codebuild-resume 명령을 실행합니다.

```
$ codebuild-resume
```

AWS CodeBuild에서 빌드 삭제

AWS CLI 또는 AWS SDK를 사용하여 AWS CodeBuild에서 빌드를 삭제할 수 있습니다.

빌드 삭제(AWS CLI)

`batch-delete-builds` 명령을 실행합니다.

```
aws codebuild batch-delete-builds --ids ids
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *ids*: 필수 문자열입니다. 삭제할 빌드의 ID입니다. 여러 개의 빌드를 지정하려면 각 빌드 ID를 공백으로 구분합니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

성공하면 성공적으로 삭제된 각 빌드의 Amazon 리소스 이름(ARN)이 포함된 `buildsDeleted` 배열이 출력에 나타납니다. 성공적으로 삭제되지 않은 빌드에 대한 정보는 `buildsNotDeleted` 배열 내 출력에 표시됩니다.

예를 들면 다음 명령을 실행하는 경우

```
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
```

다음과 비슷한 정보가 출력에 나타납니다.

```
{
  "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"
  ]
}
```

빌드 삭제(AWS SDK)

AWS CodeBuild를 AWS SDK와 함께 사용하는 방법에 대한 자세한 정보는 AWS SDK 및 도구 참조 단원을 참조하십시오.

AWS Lambda 컴퓨트 인에서 작업하기 AWS CodeBuild

AWS Lambda 컴퓨트는 빌드에 최적화된 시작 속도를 제공합니다. AWS Lambda 시작 지연 시간이 짧아 더 빠른 빌드를 지원합니다. AWS Lambda 또한 자동으로 크기가 조정되므로 빌드가 실행될 때까지 대기하지 않아도 됩니다. 하지만 AWS Lambda 지원하지 않는 일부 사용 사례가 있으며, 영향을 받는 경우 EC2 컴퓨팅을 사용하세요. 자세한 정보는 AWS Lambda 컴퓨팅의 한계을 참조하세요.

주제

AWS Lambda에서 실행되는 큐레이팅된 런타임 환경 도커 이미지에는 어떤 도구와 런타임이 포함되나요?

AWS Lambda AWS CLI v2, AWS SAM CLI, git, go, 자바, Node.js, Python, pip, Ruby, .NET과 같은 도구를 지원합니다.

큐레이션된 이미지에 필요한 도구가 포함되어 있지 않으면 어떻게 되나요?

큐레이션된 이미지에 필요한 도구가 포함되어 있지 않은 경우 필요한 도구가 포함된 사용자 지정 환경 Docker 이미지를 제공할 수 있습니다.

Lambda 컴퓨팅용 사용자 지정 이미지를 사용하려면 다음과 같은 Amazon ECR 권한이 필요합니다.

```
{
  "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"
    }
  ]
}
```

또한 사용자 지정 `curl` 이미지를 `wget` 사용하려면 OR를 설치해야 합니다.

AWS Lambda 컴퓨팅이 지원되는 지역은 CodeBuild 어디입니까?

에서 CodeBuild AWS Lambda 컴퓨팅은 미국 동부 (버지니아 북부), 미국 동부 (오하이오), 미국 서부 (오레곤), 아시아 태평양 (뭄바이), 아시아 태평양 (싱가포르), 아시아 태평양 (시드니), 아시아 태평양 (도쿄), 유럽 (프랑크푸르트), 유럽 (아일랜드), 남아메리카 (상파울루) 에서 지원됩니다. AWS 리전[이용 가능 AWS 리전 지역에 CodeBuild 대한 자세한 내용은 지역별 서비스를 참조하십시오.AWS]

AWS Lambda 컴퓨팅의 한계

AWS Lambda 지원하지 않는 몇 가지 사용 사례가 있으며, 문제가 발생할 경우 EC2 컴퓨팅을 사용하십시오.

- AWS Lambda 루트 권한이 필요한 도구는 지원하지 않습니다. `yum` 또는 `rpm` 등의 도구에는 EC2 컴퓨팅 유형이나 루트 권한이 필요하지 않은 기타 도구를 사용하세요.
- AWS Lambda Docker 빌드 또는 실행을 지원하지 않습니다. Podman과 같이 루트 권한이 필요하지 않은 대안을 사용할 수 있습니다.
- AWS Lambda 외부 `/tmp` 파일에 쓰는 것은 지원하지 않습니다. 포함된 패키지 관리자는 기본적으로 이 `/tmp` 디렉터리를 사용하여 패키지를 다운로드하고 참조하도록 구성되어 있습니다.
- AWS Lambda 환경 유형을 `LINUX_GPU_CONTAINER` 지원하지 않으며 Windows Server Core 2019 에서는 지원되지 않습니다.
- AWS Lambda 캐싱, 일괄 빌드, 사용자 지정 빌드 시간 제한, 대기열 제한 시간, 빌드 배지, 권한 모드, 사용자 지정 런타임 환경 또는 15분 이상의 런타임을 지원하지 않습니다.

- AWS Lambda VPC 연결, 고정된 범위의 CodeBuild 소스 IP 주소, EFS, 시맨틱 버전 관리, 인증서 설치 또는 세션 관리자를 통한 SSH 액세스를 지원하지 않습니다.

AWS Lambda 컴퓨팅 샘플은 다음과 같습니다. AWS CodeBuild

이러한 샘플 그룹을 사용하여 AWS Lambda 컴퓨팅을 실험할 수 CodeBuild 있습니다.

주제

- Lambda 자바와 함께 사용하여 AWS SAM Lambda 함수를 배포하십시오. CodeBuild
- CodeBuild Lambda Node.js 를 사용하여 단일 페이지 리액트 앱 생성
- Lambda Python으로 Lambda 함수 구성 업데이트 CodeBuild

Lambda 자바와 함께 사용하여 AWS SAM Lambda 함수를 배포하십시오. CodeBuild

AWS Serverless Application Model(AWS SAM)은 서버리스 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. 자세한 내용은 의 리포지토리를 참조하십시오. AWS Serverless Application Model GitHub 다음 Java 샘플은 Gradle을 사용하여 AWS Lambda 함수를 빌드하고 테스트합니다. 그런 다음 AWS SAM CLI를 사용하여 AWS CloudFormation 템플릿과 배포 번들을 배포합니다. CodeBuild Lambda를 사용하면 빌드, 테스트 및 배포 단계가 모두 자동으로 처리되므로 단일 빌드에서 수동 개입 없이 인프라를 빠르게 업데이트할 수 있습니다.

리포지토리 설정 AWS SAM

AWS SAMCLI를 사용하여 AWS SAM `Hello World` 프로젝트를 생성합니다.

프로젝트를 AWS SAM 만들려면

1. 로컬 시스템에 AWS SAMCLI를 설치하기 위한 AWS Serverless Application Model개발자 안내서의 지침을 따르십시오.
2. 다음 프로젝트 구성을 `sam init` 실행하고 선택합니다.

```
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. 지원되는 소스 저장소에 AWS SAM 프로젝트 폴더를 업로드합니다. 지원되는 소스 유형 목록은 을 참조하십시오 ProjectSource.

CodeBuild Lambda 자바 프로젝트 생성

AWS CodeBuildLambda Java 프로젝트를 생성하고 빌드에 필요한 IAM 권한을 설정합니다.

CodeBuild Lambda 자바 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 각 AWS 계정에서 빌드 프로젝트 이름은 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스에서 AWS SAM 프로젝트가 위치한 소스 저장소를 선택합니다.
5. 환경에서 다음과 같이 합니다.
   - 컴퓨팅에서 Lambda를 선택합니다.
   - 런타임에서 Java를 선택합니다.
   - 이미지에서 AWS/codebuild/amazonlinux-x86_64-람다-스탠다드:corretto21을 선택합니다.
   - 서비스 역할의 경우 새 서비스 역할을 선택된 상태로 두십시오. 역할 이름을 기록해 둡니다. 이 샘플의 뒷부분에서 프로젝트의 IAM 권한을 업데이트할 때 필요합니다.
6. 빌드 프로젝트 생성을 선택합니다.
7. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
8. 탐색 창에서 역할을 선택하고 프로젝트와 관련된 서비스 역할을 선택합니다. 에서 빌드 프로젝트를 선택하고 편집, 환경, 서비스 역할을 차례로 CodeBuild 선택하여 프로젝트 역할을 찾을 수 있습니다.
9. 신뢰 관계(Trust relationships) 탭을 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.

10. 다음 인라인 정책을 IAM 역할에 추가합니다. 이는 나중에 AWS SAM 인프라를 배포하는 데 사용됩니다. 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "iam:*",
                "apigateway:*",
                "s3:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

프로젝트 빌드 사양을 설정합니다.

Lambda 함수를 CodeBuild 빌드, 테스트 및 배포하기 위해 buildspec에서 빌드 명령을 읽고 실행합니다.

프로젝트 빌드 사양을 설정하려면

1. CodeBuild 콘솔에서 빌드 프로젝트를 선택한 다음 편집 및 Buildspec을 선택합니다.
2. Buildspec에서 빌드 명령 삽입을 선택한 다음 편집기로 전환을 선택합니다.
3. 미리 입력된 빌드 명령을 삭제하고 다음 빌드 사양에 붙여넣습니다.

```
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. Update buildspec(buildspec 업데이트)을 선택합니다.

AWS SAMLambda 인프라를 배포하세요

CodeBuild Lambda를 사용하여 Lambda 인프라를 자동으로 배포하십시오.

Lambda 인프라를 배포하려면

1. 빌드 시작를 선택합니다. 그러면 AWS SAM 애플리케이션이 자동으로 빌드, 테스트 및 using에 AWS Lambda 배포됩니다. AWS CloudFormation
2. 빌드가 완료되면 AWS Lambda 콘솔로 이동하여 프로젝트 이름 아래에서 AWS SAM 새 Lambda 함수를 검색합니다.
3. 함수 개요에서 API Gateway를 선택한 다음 API 엔드포인트 URL을 클릭하여 Lambda 함수를 테스트합니다. 메시지가 포함된 페이지가 열려 있는 것이 보일 것입니다. `"message": "hello world"`

인프라를 정리하세요.

이 자습서에서 사용한 리소스에 대한 추가 요금이 부과되지 않도록 하려면 AWS SAM 템플릿으로 생성한 리소스를 삭제하고 CodeBuild

인프라를 정리하려면

1. AWS CloudFormation콘솔로 이동하여 를 선택합니다`aws-sam-cli-managed-default`.
2. 리소스에서 배포 버킷을 비웁니다`SamCliSourceBucket`.
3. `aws-sam-cli-managed-default` 스택을 삭제합니다.
4. AWS SAM프로젝트와 관련된 AWS CloudFormation 스택을 삭제합니다. 이 스택은 AWS SAM 프로젝트와 이름이 같아야 합니다.

5. CloudWatch 콘솔로 이동하여 CodeBuild 프로젝트와 관련된 CloudWatch 로그 그룹을 삭제합니다.
6. CodeBuild 콘솔로 이동한 다음 빌드 CodeBuild 프로젝트 삭제를 선택하여 프로젝트를 삭제합니다.

CodeBuild Lambda Node.js 를 사용하여 단일 페이지 리액트 앱 생성

React 앱 생성은 단일 페이지 React 애플리케이션을 생성하는 방법입니다. 다음 Node.js 샘플은 Node.js 를 사용하여 Create React App에서 소스 아티팩트를 빌드하고 빌드 아티팩트를 반환합니다.

소스 리포지토리 및 아티팩트 버킷을 설정합니다.

yarn을 사용하여 프로젝트의 소스 리포지토리를 만들고 React 앱을 생성하세요.

소스 리포지토리 및 아티팩트 버킷을 설정하려면

1. 로컬 컴퓨터에서 를 `yarn create react-app <app-name>` 실행하여 간단한 React 앱을 생성합니다.
2. React 앱 프로젝트 폴더를 지원되는 소스 리포지토리에 업로드합니다. 지원되는 소스 유형 목록은 을 참조하십시오 ProjectSource.

CodeBuild Lambda Node.js 프로젝트 생성

AWS CodeBuildLambda Node.js 프로젝트를 생성합니다.

CodeBuild Lambda Node.js 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 각 AWS 계정에서 빌드 프로젝트 이름은 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스에서 AWS SAM 프로젝트가 위치한 소스 저장소를 선택합니다.
5. 환경에서 다음과 같이 합니다.
   - 컴퓨팅에서 Lambda를 선택합니다.

- 런타임의 경우 Node.js 를 선택합니다.
- 이미지의 경우 AWS/codebuild/amazonlinux-x86_64-람다-스탠다드:nodejs20을 선택합니다.

6. 결과물에서 다음과 같이 합니다.

- 유형에서 Amazon S3를 선택합니다.
- 버킷 이름에서 이전에 생성한 프로젝트 아티팩트 버킷을 선택합니다.
- 아티팩트 패키징의 경우 Zip을 선택합니다.

7. 빌드 프로젝트 생성을 선택합니다.

프로젝트 빌드 사양을 설정합니다.

React 앱을 빌드하려면 buildspec 파일에서 빌드 명령을 CodeBuild 읽고 실행합니다.

프로젝트 빌드 사양을 설정하려면

1. CodeBuild 콘솔에서 빌드 프로젝트를 선택한 다음 편집 및 Buildspec을 선택합니다.
2. Buildspec에서 빌드 명령 삽입을 선택한 다음 편집기로 전환을 선택합니다.
3. 미리 입력된 빌드 명령을 삭제하고 다음 빌드 사양에 붙여넣습니다.

```
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. Update buildspec(buildspec 업데이트)을 선택합니다.

React 앱을 빌드하고 실행합니다.

CodeBuild Lambda에서 React 앱을 빌드하고, 빌드 아티팩트를 다운로드하고, 로컬에서 React 앱을 실행합니다.

React 앱을 빌드하고 실행하려면

1. 빌드 시작를 선택합니다.
2. 빌드가 완료되면 Amazon S3 프로젝트 아티팩트 버킷으로 이동하여 React 앱 아티팩트를 다운로드합니다.
3. React 빌드 아티팩트의 압축을 풀고 프로젝트 폴더에 저장합니다`run npm install -g serve && serve -s build`.
4. 이 `serve` 명령은 로컬 포트의 정적 사이트를 제공하고 터미널에 출력을 출력합니다. 터미널 `Local:` 출력에서 아래에 있는 localhost URL을 방문하여 React 앱을 볼 수 있습니다.

React 기반 서버의 배포를 처리하는 방법에 대해 자세히 알아보려면 React 앱 배포 만들기를 참조하십시오.

인프라를 정리하세요.

이 자습서에서 사용한 리소스에 대한 추가 요금이 부과되지 않도록 하려면 CodeBuild 프로젝트용으로 만든 리소스를 삭제하세요.

인프라를 정리하려면

1. 프로젝트 아티팩트 Amazon S3 버킷을 삭제합니다.
2. CloudWatch 콘솔로 이동하여 CodeBuild 프로젝트와 관련된 CloudWatch 로그 그룹을 삭제합니다.
3. CodeBuild 콘솔로 이동한 다음 빌드 CodeBuild 프로젝트 삭제를 선택하여 프로젝트를 삭제합니다.

Lambda Python으로 Lambda 함수 구성 업데이트 CodeBuild

다음 파이썬 샘플은 Boto3와 Lambda Python을 사용하여 CodeBuild Lambda 함수의 구성을 업데이트합니다. 이 샘플을 확장하여 프로그래밍 방식으로 다른 리소스를 관리할 수 있습니다. AWS 자세한 내용은 Boto3 설명서를 참조하십시오.

필수 조건

계정에서 Lambda 함수를 생성하거나 검색하십시오.

이 샘플은 계정에 Lambda 함수를 이미 생성했으며 Lambda 함수의 환경 변수를 CodeBuild 업데이트하는 데 사용할 것으로 가정합니다. Lambda 함수를 설정하는 방법에 대한 자세한 내용은 샘플 또는 CodeBuild Lambda 자바와 함께 사용하여 AWS SAM Lambda 함수를 배포하십시오. CodeBuild 을 참조하십시오. AWS Lambda

소스 리포지토리를 설정합니다.

Boto3 python 스크립트를 저장할 소스 리포지토리를 만드세요.

소스 리포지토리를 설정하려면

1. 라는 새 파일에 다음 python 스크립트를 `update_lambda_environment_variables.py` 복사합니다.

```
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. 지원되는 소스 리포지토리에 python 파일을 업로드합니다. 지원되는 소스 유형 목록은 을 참조하십시오 ProjectSource.

CodeBuild Lambda Python 프로젝트 생성

CodeBuild Lambda Python 프로젝트를 생성합니다.

CodeBuild Lambda 자바 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 각 AWS 계정에서 빌드 프로젝트 이름은 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스에서 AWS SAM 프로젝트가 위치한 소스 저장소를 선택합니다.
5. 환경에서 다음과 같이 합니다.
   - 컴퓨팅에서 Lambda를 선택합니다.
   - 런타임에서 Python을 선택합니다.
   - 이미지의 경우 AWS/codebuild/amazonlinux-x86_64-람다-스탠다드:python3.12를 선택합니다.
   - 서비스 역할의 경우 새 서비스 역할을 선택된 상태로 둡니다. 역할 이름을 기록해 둡니다. 이 샘플의 뒷부분에서 프로젝트의 IAM 권한을 업데이트할 때 필요합니다.
6. 빌드 프로젝트 생성을 선택합니다.
7. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
8. 탐색 창에서 역할을 선택하고 프로젝트와 관련된 서비스 역할을 선택합니다. 에서 빌드 프로젝트를 선택하고 편집, 환경, 서비스 역할을 차례로 CodeBuild 선택하여 프로젝트 역할을 찾을 수 있습니다.
9. 신뢰 관계(Trust relationships) 탭을 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.

10. 다음 인라인 정책을 IAM 역할에 추가합니다. 이는 나중에 AWS SAM 인프라를 배포하는 데 사용됩니다. 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "UpdateLambdaPermissions",
            "Effect": "Allow",
            "Action": [
                "lambda:UpdateFunctionConfiguration"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

프로젝트 빌드 사양을 설정합니다.

Lambda 함수를 업데이트하기 위해 스크립트는 buildspec에서 환경 변수를 읽어 Lambda 함수 이름, 환경 변수 이름 및 환경 변수 값을 찾습니다.

프로젝트를 설정하려면: buildspec을 빌드하십시오.

1. CodeBuild 콘솔에서 빌드 프로젝트를 선택한 다음 편집 및 Buildspec을 선택합니다.
2. Buildspec에서 빌드 명령 삽입을 선택한 다음 편집기로 전환을 선택합니다.
3. 미리 입력된 빌드 명령을 삭제하고 다음 빌드 사양에 붙여넣습니다.

```
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. Update buildspec(buildspec 업데이트)을 선택합니다.

Lambda 구성 업데이트

CodeBuild Lambda Python을 사용하여 Lambda 함수의 구성을 자동으로 업데이트하십시오.

Lambda 함수의 구성을 업데이트하려면

1. 빌드 시작를 선택합니다.
2. 빌드가 완료되면 Lambda 함수로 이동합니다.
3. 구성을 선택한 다음 환경 변수를 선택합니다. FEATURE_ENABLED키와 값이 있는 새 환경 변수가 표시되어야 `true` 합니다.

인프라를 정리하세요.

이 자습서에서 사용한 리소스에 대한 추가 요금이 부과되지 않도록 하려면 CodeBuild 프로젝트용으로 만든 리소스를 삭제하세요.

인프라를 정리하려면

1. CloudWatch 콘솔로 이동하여 CodeBuild 프로젝트와 관련된 CloudWatch 로그 그룹을 삭제합니다.
2. CodeBuild 콘솔로 이동한 다음 빌드 CodeBuild 프로젝트 삭제를 선택하여 프로젝트를 삭제합니다.
3. 이 샘플의 목적으로 Lambda 함수를 생성한 경우, 작업 및 삭제 함수를 선택하여 Lambda 함수를 정리하십시오.

확장 프로그램

AWS CodeBuildLambda Python을 사용하여 이 샘플을 확장하여 다른 AWS 리소스를 관리하려는 경우:

- Boto3를 사용하여 Python 스크립트를 업데이트하여 새 리소스를 수정합니다.
- 새 리소스에 대한 권한을 갖도록 CodeBuild 프로젝트와 관련된 IAM 역할을 업데이트하세요.

- 새 리소스와 관련된 모든 새 환경 변수를 buildspec에 추가합니다.

에서 예약된 용량으로 작업하기 AWS CodeBuild

CodeBuild 다음과 같은 컴퓨팅 플릿을 제공합니다.

- 온디맨드 플릿
- 예약 용량 플릿

온디맨드 플릿을 사용하면 빌드를 위한 컴퓨팅을 CodeBuild 제공합니다. 빌드가 완료되면 머신이 파괴됩니다. 온디맨드 플릿은 완전 관리형이며, 수요 급증을 처리할 수 있는 자동 규모 조정 기능이 포함되어 있습니다.

> **Note**
>
> 온디맨드 플릿은 Windows Server 2022를 지원하지 않습니다.

CodeBuild 에서 유지 관리하는 Amazon EC2 기반 인스턴스를 포함하는 예약 용량 플릿도 제공합니다. CodeBuild 예약 용량 플릿을 사용하면 빌드 환경을 위한 전용 인스턴스 세트를 구성할 수 있습니다. 이러한 머신은 유휴 상태로 유지되므로 빌드 또는 테스트를 즉시 처리하고 빌드 기간을 단축할 수 있습니다. 예약 용량 플릿을 사용하면 머신이 상시 가동되므로 프로비저닝하는 한 계속해서 비용이 발생합니다.

> **Important**
>
> 인스턴스 실행 기간에 관계없이 예약 용량 플릿은 인스턴스당 초기 요금이 발생하며, 그 이후에는 추가 관련 비용이 발생할 수 있습니다. 자세한 설명은 https://aws.amazon.com/codebuild/pricing/ 섹션을 참조하세요.

주제

- 예약 용량 샘플은 다음과 같습니다. AWS CodeBuild
- 예약 용량 플릿의 제한

예약 용량 플릿을 시작하려면 어떻게 해야 합니까?

예약 용량 플릿 생성

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 컴퓨팅 플릿을 선택한 다음, 컴퓨팅 플릿 생성을 선택합니다.
3. 컴퓨팅 플릿 이름 텍스트 필드에 플릿의 이름을 입력합니다.
4. 운영 체제 드롭다운 메뉴에서 운영 체제를 선택합니다.
5. 아키텍처 드롭다운 메뉴에서 아키텍처를 선택합니다.
6. 컴퓨팅 드롭다운 메뉴에서 해당 머신의 컴퓨팅 머신 유형을 선택합니다.
7. 용량 텍스트 필드에 플릿의 최소 인스턴스 수를 입력합니다.
8. 오버플로 동작 필드에서 수요가 플릿 용량을 초과할 때의 동작을 선택합니다. 이러한 옵션에 대한 자세한 내용은 예약 용량 플릿 속성 섹션을 참조하세요.
9. 컴퓨팅 플릿 생성을 선택합니다.
10. 컴퓨팅 플릿이 생성된 후 새 CodeBuild 프로젝트를 만들거나 기존 프로젝트를 편집하십시오. 환경에서 프로비저닝 모델의 예약 용량을 선택한 다음 플릿 이름에서 지정된 플릿을 선택합니다.

모범 사례

예약 용량 플릿을 사용할 때는 다음 모범 사례를 따르는 것이 좋습니다.

- 소스를 캐싱하여 빌드 성능을 향상시키려면 소스 캐시 모드를 사용하는 것이 좋습니다.
- 기존 Docker 계층을 캐싱하여 빌드 성능을 개선하려면 Docker 계층 캐싱을 사용하는 것이 좋습니다.

예약 용량 플릿을 여러 CodeBuild 프로젝트에서 공유할 수 있나요?

예. 여러 프로젝트에서 플릿을 사용하여 플릿의 용량 활용도를 극대화할 수 있습니다.

예약 용량 플릿을 지원하는 리전은 어디입니까?

예약 용량 플릿은 미국 동부 (버지니아 북부), 미국 동부 (오하이오), 미국 서부 (오레곤), 아시아 태평양 (뭄바이), 아시아 태평양 (싱가포르), 아시아 태평양 (시드니), 아시아 태평양 (도쿄), 유럽 (프랑크푸르트), 유럽 (아일랜드), 남아메리카 (상파울루) 에서 지원됩니다. AWS 리전이용 가능 AWS 리전 지역에 대한 자세한 내용은 지역별 서비스를 참조하십시오 CodeBuild .AWS

예약 용량 플릿 속성

예약 용량 플릿에는 다음 속성이 포함됩니다.

운영 체제

운영 체제입니다. 사용할 수 있는 운영 체제는 다음과 같습니다.

- Amazon Linux
- Windows Server 2019
- Windows Server 2022

아키텍처

프로세서 아키텍처. 사용할 수 있는 아키텍처는 다음과 같습니다.

- x86_64
- Arm64

컴퓨팅

각 인스턴스의 컴퓨팅 머신 유형. 사용할 수 있는 유형의 로그는 다음과 같습니다.

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	vCPU	디스크 공간
ARM Small	`BUILD_GENERAL1_SMALL`	`ARM_CONTAINER`	4GB	2	50GB
ARM Large	`BUILD_GENERAL1_LARGE`	`ARM_CONTAINER`	16 GB	8	50GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	vCPU	디스크 공간
Linux Small [1]	BUILD_GENERAL1_SMALL	LINUX_CONTAINER	3GB	2	64GB
Linux Medium [1]	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER	7GB	4	128GB
Linux Large [1]	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	15GB	8	128GB
Linux XLarge	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	70GB	36	256GB
Linux 2XLarge	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	145 GB	72	824 GB(SSD)
Linux GPU Sm	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GB	4	220GB
Linux GPU Lar	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GB	32	50GB
Windows Medi	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER	7GB	4	128GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	메모리	vCPU	디스크 공간
Windows Medi	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2022_CONTAINER	7GB	4	128GB
Windows Large	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER	15GB	8	128GB
Windows Large	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2022_CONTAINER	15GB	8	128GB

Capacity

플릿에 할당된 초기 머신 수로, 병렬로 실행할 수 있는 빌드 수를 정의합니다.

오버플로 동작

빌드 수가 플릿 용량을 초과할 때의 동작을 정의합니다.

온디맨드

오버플로 빌드는 온디맨드로 실행됩니다. CodeBuild

> **Important**
>
> 오버플로 동작을 온디맨드로 설정하는 경우 오버플로 빌드는 온디맨드 Amazon EC2와 마찬가지로 별도로 요금이 청구된다는 점에 유의하십시오. 자세한 설명은 https://aws.amazon.com/codebuild/pricing/ 섹션을 참조하세요.

대기열

머신을 사용할 수 있을 때까지 빌드 실행이 대기열에 배치됩니다. 이렇게 하면 추가 머신이 할당되지 않으므로 추가 비용이 제한됩니다.

예약 용량 샘플은 다음과 같습니다. AWS CodeBuild

이 샘플을 사용하여 예약된 용량 플릿을 실험해 볼 수 있습니다. CodeBuild

주제

예약 용량 샘플을 사용한 캐싱

캐시는 빌드 환경에서 재사용할 수 있는 정보를 저장하여 여러 빌드에 사용할 수 있습니다. 이 샘플은 예약 용량을 사용하여 빌드 프로젝트 내에서 캐싱을 활성화하는 방법을 보여주었습니다. 자세한 설명은 AWS CodeBuild의 빌드 캐싱 섹션을 참조하세요.

프로젝트 설정에서 하나 이상의 캐시 모드를 지정하여 시작할 수 있습니다.

```
Cache:
        Type: LOCAL
        Modes:
          - LOCAL_CUSTOM_CACHE
          - LOCAL_DOCKER_LAYER_CACHE
          - LOCAL_SOURCE_CACHE
```

> **Note**
>
> Docker 계층 캐시를 사용하려면 권한 모드를 활성화해야 합니다.

프로젝트 buildspec 설정이 다음과 같아야 합니다.

```
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'
```

캐시를 시드하는 새 프로젝트로 빌드를 실행하여 시작할 수 있습니다. 작업이 완료되면 다음과 같이 buildspec을 재정의하는 다른 빌드를 시작해야 합니다.

```
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'
```

예약 용량 플릿의 제한

예약 용량 플릿이 지원하지 않는 몇 가지 사용 사례가 있으며, 이로 인해 영향을 받는 경우에는 온디맨드 플릿을 대신 사용하십시오.

- 예약 용량 플릿은 배치 빌드, 빌드 사용률 지표 또는 의미 체계 버전 관리를 지원하지 않습니다.
- 예약 용량 플릿은 VPC 연결을 지원하지 않습니다.

제한과 할당량에 대한 자세한 내용은 컴퓨팅 플릿 섹션을 참조하십시오.

에서 테스트 리포팅 사용하기 AWS CodeBuild

빌드 중에 실행되는 테스트에 대한 세부 정보가 CodeBuild 포함된 보고서를 생성할 수 있습니다. 단위 테스트, 구성 테스트 및 기능 테스트와 같은 테스트를 만들 수 있습니다.

지원되는 테스트 보고서 파일 형식은 다음과 같습니다.

- Cucumber JSON(.json)
- JUnit XML(.xml)
- NUnit XML(.xml)
- NUnit3 XML(.xml)
- TestNG XML(.xml)
- Visual Studio TRX(.trx)
- 비주얼 스튜디오 TRX XML (.xml)

> **Note**
>
> `cucumber-js`의 지원되는 최신 버전은 7.3.2입니다.

이러한 형식 중 하나로 보고서 파일을 만들 수 있는 테스트 프레임워크로 테스트 케이스를 만듭니다 (예: Surefire JUnit plugin, TestNG, Cucumber).

테스트 보고서를 만들려면 테스트 케이스에 관한 정보와 함께 빌드 프로젝트의 buildspec 파일에 보고서 그룹 이름을 추가합니다. 빌드 프로젝트를 실행하면 테스트 케이스가 실행되고 테스트 보고서가 만들어집니다. 테스트를 실행하기 전에 보고서 그룹을 만들 필요가 없습니다. 보고서 그룹 이름을 지정하면 보고서를 실행할 때 보고서 그룹이 자동으로 CodeBuild 만들어집니다. 이미 존재하는 보고서 그룹을 사용하려면 buildspec 파일에서 해당 ARN을 지정합니다.

테스트 보고서를 사용하여 빌드 실행 중에 문제를 해결할 수 있습니다. 빌드 프로젝트의 여러 빌드에 많은 테스트 보고서가 있는 경우 테스트 보고서를 사용하여 추세와 테스트 및 실패율을 보고 빌드를 쉽게 최적화할 수 있습니다.

보고서는 생성되고 30일 후에 만료됩니다. 만료된 테스트 보고서는 볼 수 없습니다. 테스트 보고서를 30일 이상 보관하려면 테스트 결과의 원시 데이터 파일을 Amazon S3 버킷으로 내보내면 됩니다. 내보낸 테스트 파일은 만료되지 않습니다. S3 버킷에 대한 정보는 보고서 그룹을 생성할 때 지정됩니다.

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드할 수 있는 권한에 사용됩니다.

주제

테스트 보고서 작성

테스트 보고서를 만들려면 buildspec 파일에서 1~5개의 보고서 그룹으로 구성된 빌드 프로젝트를 실행합니다. 실행 중에 테스트 보고서가 작성됩니다. 보고서 그룹에 지정된 테스트 케이스의 결과가 포함되어 있습니다. 동일한 buildspec 파일을 사용하는 각 후속 빌드에 대해 새 테스트 보고서가 생성됩니다.

테스트 보고서를 작성하려면

1. 빌드 프로젝트를 생성합니다. 자세한 내용은 AWS CodeBuild에서 빌드 프로젝트 생성을 참조하세요.
2. 테스트 보고서 정보를 사용하여 프로젝트의 buildspec 파일을 구성합니다.

   a. `reports:` 섹션을 추가하고 기존 보고서 그룹의 ARN 또는 보고서 그룹의 이름을 지정합니다.

      ARN을 지정하는 경우 이 보고서 CodeBuild 그룹을 사용합니다.

      이름을 지정하면 프로젝트 이름과 지정된 이름을 사용하여 <project-name>-*< report-group-name >* 형식으로 보고서 그룹이 자동으로 CodeBuild 생성됩니다. 지정된 보고서 그룹이 이미 있는 경우, 해당 보고서 그룹을 CodeBuild 사용합니다.

b. 보고서 그룹에서 테스트 결과를 포함하는 파일의 위치를 지정합니다. 둘 이상의 보고서 그룹을 사용하는 경우 각각에 대해 테스트 결과 파일 위치를 지정합니다. 빌드 프로젝트가 실행될 때마다 새 테스트 보고서가 만들어집니다. 자세한 정보는 테스트 파일 지정을 참조하세요.

c. `build` 또는 `post_build` 시퀀스의 `commands` 섹션에서 보고서 그룹에 대해 지정한 테스트 케이스를 실행하는 명령을 지정합니다. 자세한 정보는 테스트 명령 지정 을 참조하세요.

다음은 buildspec `reports` 섹션의 예입니다.

```
reports:
  php-reports:
    files:
      - "reports/php/*.xml"
    file-format: "JUNITXML"
  nunit-reports:
    files:
      - "reports/nunit/*.xml"
    file-format: "NUNITXML"
```

3. 빌드 프로젝트의 빌드를 실행합니다. 자세한 정보는 AWS CodeBuild에서 빌드 실행을 참조하세요.
4. 빌드가 완료되면 프로젝트 페이지의 빌드 기록에서 새 빌드 실행을 선택하십시오. 보고서를 선택하여 테스트 보고서를 봅니다. 자세한 정보는 빌드에 대한 테스트 보고서 보기을 참조하세요.

보고서 그룹 작업

보고서 그룹은 그 안에 테스트 보고서가 포함되어 있으며 공유 설정을 지정합니다. buildspec 파일을 사용하여 실행할 테스트 케이스와 빌드할 때 실행할 명령을 지정합니다. 빌드 프로젝트에 구성된 각 보고서 그룹에 대해 빌드 프로젝트를 실행하면 테스트 보고서가 만들어집니다. 보고서 그룹으로 구성된 빌드 프로젝트를 여러 번 실행하면 해당 보고서 그룹에 여러 테스트 보고서가 작성되며, 각각 해당 보고서 그룹에 대해 지정된 동일한 테스트 케이스의 결과가 표시됩니다.

테스트 케이스는 빌드 프로젝트의 buildspec 파일에 있는 보고서 그룹에 대해 지정됩니다. 하나의 빌드 프로젝트에서 최대 5개의 보고서 그룹을 지정할 수 있습니다. 빌드를 실행하면 모든 테스트 케이스가 실행됩니다. 보고서 그룹에 지정된 각 테스트 케이스의 결과와 함께 새 테스트 보고서가 작성됩니다. 새 빌드를 실행할 때마다 테스트 케이스가 실행되고 새 테스트 보고서가 새 테스트 결과와 함께 만들어집니다.

보고서 그룹은 둘 이상의 빌드 프로젝트에서 사용할 수 있습니다. 하나의 보고서 그룹으로 작성된 모든 테스트 보고서는 테스트 보고서가 다른 빌드 프로젝트를 사용하여 작성된 경우에도 내보내기 옵션 및 사용 권한과 같은 동일한 구성을 공유합니다. 여러 빌드 프로젝트에서 하나의 보고서 그룹으로 만든 테스트 보고서에는 서로 다른 테스트 케이스 세트(각 빌드 프로젝트마다 테스트 케이스 세트 하나씩)를 실행한 결과가 포함될 수 있습니다. 이는 각 프로젝트의 buildspec 파일에서 보고서 그룹에 대해 서로 다른 테스트 케이스 파일을 지정할 수 있기 때문입니다. 그 buildspec 파일을 편집하여 빌드 프로젝트의 보고서 그룹에 대한 테스트 케이스 파일을 변경할 수도 있습니다. 후속 빌드를 실행하면 업데이트된 buildspec에서 테스트 케이스 파일의 결과를 포함하는 새 테스트 보고서가 생성됩니다.

주제

보고서 그룹 만들기

CodeBuild 콘솔 AWS CLI, 또는 buildspec 파일을 사용하여 보고서 그룹을 생성할 수 있습니다. IAM 역할에는 보고서 그룹을 생성하는 데 필요한 권한이 있어야 합니다. 자세한 정보는 테스트 보고서 권한 작업을 참조하세요.

주제

보고서 그룹 생성(buildspec)

buildspec을 사용하여 만든 보고서 그룹은 원시 테스트 결과 파일을 내보내지 않습니다. 보고서 그룹을 보고 내보내기 설정을 지정할 수 있습니다. 자세한 정보는 보고서 그룹 업데이트을 참조하세요.

buildspec 파일을 사용하여 보고서 그룹을 생성하려면

1. 계정의 보고서 그룹과 연결되지 않은 보고서 그룹 이름을 선택합니다. AWS
2. 이 이름으로 buildspec 파일의 `reports` 섹션을 구성합니다. 이 예에서는 보고서 그룹 이름이 `new-report-group`이고, 사용 테스트 케이스는 JUnit 프레임워크로 생성됩니다.

```
reports:
 new-report-group: #surefire junit reports
   files:
     - '**/*'
   base-directory: 'surefire/target/surefire-reports'
```

   buildspec의 환경 변수를 사용하여 보고서 그룹 이름을 지정할 수도 있습니다.

```
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'
```

   자세한 내용은 테스트 파일 지정 및 Reports syntax in the buildspec file 섹션을 참조하세요.
3. `commands` 섹션에서, 테스트를 실행할 명령을 지정합니다. 자세한 정보는 테스트 명령 지정 을 참조하세요.
4. 빌드를 실행합니다. 빌드가 완료되면 `project-name-report-group-name` 형식을 사용하는 이름으로 새 보고서 그룹이 만들어집니다. 자세한 정보는 보고서 그룹 이름 지정을 참조하세요.

보고서 그룹 만들기(콘솔)

테스트 보고서를 작성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.
3. Create report(보고서 그룹 생성)를 선택합니다.
4. 보고서 그룹 이름은 보고서 그룹의 이름을 입력합니다.
5. (선택 사항) 태그의 경우 지원되는 AWS 서비스에서 사용할 태그의 이름과 값을 입력합니다. [Add row]를 사용하여 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다.
6. 테스트 보고서 결과의 원시 데이터를 Amazon S3 버킷에 업로드하려면

   a. Amazon S3로 내보내기를 선택합니다.

   b. S3 버킷 이름은 S3 버킷의 이름을 입력합니다.

   c. (선택 사항) S3 버킷 소유자의 경우 S3 버킷을 소유한 계정의 AWS 계정 식별자를 입력합니다. 이 속성을 사용하여 빌드를 실행하는 계정이 아닌 다른 계정이 소유한 Amazon S3 버킷으로 보고서 데이터를 내보낼 수 있습니다.

   d. 경로 접두사는 테스트 결과를 업로드할 S3 버킷의 경로를 입력합니다.

   e. 원시 테스트 결과 데이터 파일을 압축하려면 Compress test result data in a zip file(테스트 결과 데이터를 zip 파일로 압축)을 선택합니다.

   f. 추가 구성을 확장하여 암호화 옵션을 표시합니다. 다음 중 하나를 선택합니다.

      - Amazon S3용 AWS 관리형 키를 사용하기 위한 기본 AWS 관리형 키. 자세한 내용은 AWS Key Management Service 사용 설명서의 고객 관리형 CMK를 참조하세요. 이것은 기본 암호화 옵션입니다.
      - 생성하여 구성하는 고객 관리형 키를 사용할 사용자 지정 키를 선택합니다. AWS KMS 암호화 키는 암호화 키의 ARN을 입력합니다. 형식은 `arn:aws:kms:<region-id>:<aws-account-id>:key/<key-id>` 입니다. 자세한 내용을 알아보려면 AWS Key Management Service 사용 설명서의 KMS 키 생성을 참조하세요.
      - 아티팩트 암호화를 비활성화하여 암호화를 비활성화합니다. 테스트 결과를 공유하거나 정적 웹사이트에 게시할 경우에 이를 선택할 수 있습니다. (동적 웹사이트에서 테스트 결과를 해독하는 코드를 실행할 수 있습니다.)

      유휴 데이터 암호화에 대한 자세한 내용은 데이터 암호화 단원을 참조하십시오.

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

7. Create report(보고서 그룹 생성)를 선택합니다.

보고서 그룹 생성(CLI)

보고서 그룹을 생성하려면

1. `CreateReportGroup.json`이라는 이름의 파일을 만듭니다.
2. 요구 사항에 따라 다음 JSON 코드 조각 중 하나를 `CreateReportGroup.json`에 복사합니다.

   - 다음 JSON을 사용하여 테스트 보고서 그룹이 원시 테스트 결과 파일을 Amazon S3 버킷으로 내보내도록 지정합니다.

```
{
  "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"
      }
    ]
  }
}
```

- *<bucket-name>*은 Amazon S3 버킷 이름으로 바꾸고, *<path>*는 파일을 내보낼 버킷의 경로로 바꿉니다.
- 내보낸 파일을 압축하려면 packaging을 ZIP로 지정합니다. 아닌 경우에는 NONE로 지정합니다.
- bucketOwner는 선택 사항으로 빌드를 실행하는 계정이 아닌 다른 계정이 Amazon S3 버킷을 소유한 경우에만 필요합니다.
- 내보낸 파일을 암호화할지 여부를 지정할 때 encryptionDisabled을 사용합니다. 내보낸 파일을 암호화할 경우에는 고객 관리형 키를 입력합니다. 자세한 내용은 보고서 그룹 업데이트 섹션을 참조하세요.

- 다음 JSON을 사용하여 테스트 보고서가 원시 테스트 파일을 내보내지 않도록 지정합니다.

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "NO_EXPORT"
  }
}
```

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

3. 다음 명령을 실행합니다.

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

보고서 그룹 생성(AWS CloudFormation)

AWS CloudFormation 템플릿을 사용하여 테스트 보고서를 작성하려면

AWS CloudFormation 템플릿 파일을 사용하여 보고서 그룹을 만들고 제공할 수 있습니다. 자세한 내용은 사용 설명서AWS CloudFormation를 참조하세요.

다음 AWS CloudFormation YAML 템플릿은 원시 테스트 결과 파일을 내보내지 않는 보고서 그룹을 만듭니다.

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: NO_EXPORT
```

다음 AWS CloudFormation YAML 템플릿은 원시 테스트 결과 파일을 Amazon S3 버킷으로 내보내는 보고서 그룹을 생성합니다.

```
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**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드할 수 있는 권한에 사용됩니다.

보고서 그룹 업데이트

보고서 그룹을 업데이트할 때 원시 테스트 결과 데이터를 Amazon S3 버킷의 파일로 내보낼지 여부에 대한 정보를 지정할 수 있습니다. S3 버킷으로 내보내도록 선택하는 경우에는 보고서 그룹에 대해 다음을 지정할 수 있습니다.

- 원시 테스트 결과 파일이 ZIP 파일로 압축되는지 여부.
- 원시 테스트 결과 파일이 암호화되는지 여부. 다음 중 하나를 사용하여 암호화를 지정할 수 있습니다.
  - 아마존 AWS 관리형 키 S3용입니다.
  - 직접 생성하고 구성한 고객 관리형 키.

자세한 정보는 데이터 암호화을 참조하세요.

를 사용하여 보고서 그룹을 AWS CLI 업데이트하는 경우 태그를 업데이트하거나 추가할 수도 있습니다. 자세한 정보는 AWS CodeBuild에서 보고서 그룹 태그 지정을 참조하세요.

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드할 수 있는 권한에 사용됩니다.

주제

- 보고서 그룹 업데이트(콘솔)
- 보고서 그룹 업데이트(CLI)

보고서 그룹 업데이트(콘솔)

보고서 그룹을 업데이트하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.
3. 업데이트할 보고서 그룹을 선택합니다.
4. 편집을 선택합니다.
5. Amazon S3로 백업을 선택하거나 선택을 취소합니다. 이 옵션을 선택한 경우 다음과 같은 내보내기 설정을 지정합니다.
   a. S3 버킷 이름은 S3 버킷의 이름을 입력합니다.
   b. 경로 접두사는 테스트 결과를 업로드할 S3 버킷의 경로를 입력합니다.
   c. 원시 테스트 결과 데이터 파일을 압축하려면 Compress test result data in a zip file(테스트 결과 데이터를 zip 파일로 압축)을 선택합니다.

d. 추가 구성을 확장하여 암호화 옵션을 표시합니다. 다음 중 하나를 선택합니다.

- Amazon S3에서 사용하기 AWS 관리형 키 위한 기본 AWS 관리 키입니다. 자세한 내용은 AWS Key Management Service 사용 설명서의 고객 관리형 CMK를 참조하세요. 이것은 기본 암호화 옵션입니다.
- 생성하여 구성하는 고객 관리형 키를 사용할 사용자 지정 키를 선택합니다. AWS KMS 암호화 키는 암호화 키의 ARN을 입력합니다. 형식은 `arn:aws:kms:<region-id>:<aws-account-id>:key/<key-id>` 입니다. 자세한 내용을 알아보려면AWS Key Management Service 사용 설명서의 KMS 키 생성을 참조하세요.
- 아티팩트 암호화를 비활성화하여 암호화를 비활성화합니다. 테스트 결과를 공유하거나 정적 웹사이트에 게시할 경우에 이를 선택할 수 있습니다. (동적 웹사이트에서 테스트 결과를 해독하는 코드를 실행할 수 있습니다.)

보고서 그룹 업데이트(CLI)

보고서 그룹을 업데이트하려면

1. `UpdateReportGroupInput.json`이라는 이름의 파일을 만듭니다.
2. 다음을 `UpdateReportGroupInput.json`에 복사합니다.

```
{
    "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. arn 줄에 보고서 그룹의 ARN을 입력합니다(예: "arn":"arn:aws:codebuild:*region*:*123456789012*:report-group/*report-group-1*")).
4. 보고서 그룹에 적용할 업데이트 내용으로 UpdateReportGroupInput.json을 업데이트합니다.

   - 원시 테스트 결과 파일을 S3 버킷으로 내보내도록 보고서 그룹을 업데이트하려면 exportConfig 섹션을 업데이트합니다. bucket-name은 S3 버킷 이름으로 바꾸고, path는 파일을 내보낼 S3 버킷의 경로로 바꿉니다. 내보낸 파일을 압축하려면 packaging을 ZIP로 지정합니다. 아닌 경우에는 NONE로 지정합니다. 내보낸 파일을 암호화할지 여부를 지정할 때 encryptionDisabled을 사용합니다. 내보낸 파일을 암호화할 경우에는 고객 관리형 키를 입력합니다.
   - 보고서 그룹을 업데이트하여 원시 테스트 결과 파일을 S3 버킷으로 내보내지 않으려면 exportConfig 섹션을 다음 JSON으로 업데이트합니다.

     ```
     {
       "exportConfig": {
           "exportConfigType": "NO_EXPORT"
       }
     }
     ```

   - 보고서 그룹의 태그를 업데이트하려면 tags 섹션을 업데이트합니다. 태그를 변경, 추가 또는 제거할 수 있습니다. 모든 태그를 제거하려면 다음 JSON으로 업데이트하십시오.

     ```
     "tags": []
     ```

5. 다음 명령을 실행합니다:

   ```
   aws codebuild update-report-group \
   --cli-input-json file://UpdateReportGroupInput.json
   ```

테스트 파일 지정

빌드 프로젝트의 buildspec 파일에서 reports 섹션에 있는 각 보고서 그룹에 대한 테스트 결과 파일 및 해당 위치를 지정합니다. 자세한 정보는 Reports syntax in the buildspec file을 참조하세요.

다음은 빌드 프로젝트에 대해 두 개의 보고서 그룹을 지정하는 샘플 reports 섹션입니다. 하나는 ARN으로 지정되고 다른 하나는 이름으로 지정됩니다. 이 files 섹션에서는 테스트 케이스 결과를 포

함하는 파일을 지정합니다. 선택 사항인 base-directory 섹션에서는 테스트 케이스 파일이 있는 디렉터리를 지정합니다. 선택 사항인 discard-paths 섹션에서는 Amazon S3 버킷에 업로드된 테스트 결과 파일의 경로를 무시할지 여부를 지정합니다.

```
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
```

테스트 명령 지정

buildspec 파일의 commands 섹션에서 테스트 케이스를 실행하는 명령을 지정합니다. 이 명령은 buildspec 파일의 reports 섹션에서 보고서 그룹에 대해 지정된 테스트 케이스를 실행합니다. 다음은 테스트 파일에서 테스트를 실행하는 명령이 포함된 샘플 commands 섹션입니다.

```
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
```

자세한 정보는 buildspec 구문을 참조하세요.

보고서 그룹 이름 지정

AWS CLI 또는 AWS CodeBuild 콘솔을 사용하여 보고서 그룹을 생성할 때는 보고서 그룹의 이름을 지정합니다. buildspec을 사용하여 새 보고서 그룹을 생성하는 경우 *project-name*-*report-group-name-specified-in-buildspec* 형식을 사용하여 이름이 지정됩니다. 해당 빌드 프로젝트의 빌드를 실행하여 만든 모든 보고서는 새 이름을 가진 새 보고서 그룹에 속합니다.

새 보고서 그룹을 만들지 CodeBuild 않으려면 빌드 프로젝트의 buildspec 파일에 보고서 그룹의 ARN을 지정하세요. 여러 빌드 프로젝트에서 보고서 그룹의 ARN을 지정할 수 있습니다. 각 빌드 프로젝트가 실행되면 보고서 그룹에는 각 빌드 프로젝트에서 만든 테스트 보고서가 포함됩니다.

예를 들어, 이름 `my-report-group`로 하나의 보고서 그룹을 만든 다음 이름이 `my-project-1`와 `my-project-2`인 서로 다른 두 개의 빌드 프로젝트에서 해당 이름을 사용하고 두 프로젝트의 빌드를 작성하는 경우 두 개의 새 보고서 그룹이 만들어집니다. 그 결과 다음과 같은 이름의 세 개의 보고서 그룹이 만들어집니다.

- `my-report-group`: 테스트 보고서가 없습니다.
- `my-project-1-my-report-group`: 이름이 `my-project-1`인 빌드 프로젝트에서 실행한 테스트 결과가 있는 보고서가 포함되어 있습니다.
- `my-project-2-my-report-group`: 이름이 `my-project-2`인 빌드 프로젝트에서 실행한 테스트 결과가 있는 보고서가 포함되어 있습니다.

두 프로젝트에서 이름이 `my-report-group`로 지정된 보고서 그룹의 ARN을 사용한 후 각 프로젝트의 빌드를 실행해도 보고서 그룹(`my-report-group`)은 한 개입니다. 이 보고서 그룹에는 두 빌드 프로젝트에서 실행한 테스트 결과가 있는 테스트 보고서가 포함되어 있습니다.

AWS 계정의 보고서 그룹에 속하지 않는 보고서 그룹 이름을 선택한 다음 buildspec 파일의 보고서 그룹에 해당 이름을 사용하고 빌드 프로젝트의 빌드를 실행하면 새 보고서 그룹이 만들어집니다. 새 보고서 그룹의 이름 형식은 *`project-name`*`-`*`new-group-name`*입니다. 예를 들어 AWS 계정에 이름이 `test-project` 같은 보고서 그룹이 없고 이 그룹을 빌드 프로젝트에 지정하면 빌드를 실행하면 이 이름을 `new-report-group` 가진 새 보고서 그룹이 만들어집니다. `test-project-new-report-group`

AWS CodeBuild에서 보고서 그룹 태그 지정

태그는 사용자 또는 AWS에서 AWS 리소스에 할당하는 사용자 지정 속성 레이블입니다. 각 AWS 태그는 두 부분으로 구성됩니다.

- 태그 키(예: `CostCenter`, `Environment`, `Project` 또는 `Secret`). 태그 키는 대/소문자를 구별합니다.
- 태그 값(예: `111122223333`, `Production` 또는 팀 이름)으로 알려진 선택적 필드. 태그 값을 생략하는 것은 빈 문자열을 사용하는 것과 같습니다. 태그 키처럼 태그 값은 대/소문자를 구별합니다.

태그 키와 태그 값을 합해서 키 값 페어라고 합니다. 보고서 그룹에 포함할 수 있는 태그 수 제한 및 태그 키 및 값에 대한 제한은 Tags 단원을 참조하십시오.

태그를 사용하면 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당하여 해당 리소스의 관련 여부를 나타낼 수 있습니다. 예를 들어 Amazon S3 버킷에 할당한 것과 동일한 태그를 CodeBuild 보고서 그룹에 할당할 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례 백서를 참조하십시오.

CodeBuild에서 기본 리소스는 보고서 그룹 및 프로젝트입니다. CodeBuild 콘솔, AWS CLI, CodeBuild API 또는 AWS SDK를 사용하여 보고서 그룹의 태그를 추가, 관리 및 제거할 수 있습니다. 태그로 보고서 그룹을 식별, 구성 및 추적하는 것 외에도 IAM 정책의 태그를 사용하여 보고서 그룹을 보고 상호 작용할 수 있는 사용자를 제어할 수 있습니다. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

주제

보고서 그룹에 태그 추가

보고서 그룹에 태그를 추가하면 AWS 리소스를 식별 및 구성하고 해당 리소스에 대한 액세스를 관리할 수 있습니다. 먼저 보고서 그룹에 하나 이상의 태그(키-값 페어)를 추가합니다. 보고서 그룹에 태그 수에 대한 제한이 있음을 알아두십시오. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있습니다. 자세한 내용은 Tags 섹션을 참조하세요. 태그가 생성된 후 해당 태그를 기준으로 보고서 그룹에 대한 액세스를 관리하는 IAM 정책을 생성할 수 있습니다. CodeBuild 콘솔 또는 AWS CLI를 사용하여 보고서 그룹에 태그를 추가할 수 있습니다.

> **Important**
>
> 보고서 그룹에 태그를 추가하면 해당 보고서 그룹에 대한 액세스에 영향을 줄 수 있습니다. 보고서 그룹에 태그를 추가하기 전에 보고서 그룹과 같은 리소스에 대한 액세스를 제어하는 태그를 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

보고서 그룹을 생성할 때 보고서 그룹에 태그를 추가하는 방법에 대한 자세한 내용은 보고서 그룹 만들기(콘솔) 단원을 참조하십시오.

주제

- 보고서 그룹에 태그 추가(콘솔)
- 보고서 그룹에 태그 추가(AWS CLI)

보고서 그룹에 태그 추가(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 보고서 그룹에 하나 이상의 태그를 추가할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 추가할 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다.
4. 보고서 그룹에 추가된 태그가 없는 경우 태그 추가를 선택합니다. 편집을 선택한 다음 태그 추가를 선택할 수도 있습니다.
5. 키에 태그 이름을 입력합니다. 값(Value)에 태그의 선택적 값을 추가할 수 있습니다.
6. (선택 사항) 다른 태그를 추가하려면 다시 태그 추가를 선택합니다.
7. 태그 추가를 마쳤으면 제출을 선택합니다.

보고서 그룹에 태그 추가(AWS CLI)

보고서 그룹을 생성할 때 보고서 그룹에 태그를 추가하려면 보고서 그룹 생성(CLI) 단원을 참조하십시오. `CreateReportGroup.json`에서 태그를 추가합니다.

기존 보고서 그룹에 태그를 추가하려면 보고서 그룹 업데이트(CLI) 단원을 참조하고 `UpdateReportGroupInput.json`에서 태그를 추가하십시오.

이 단계에서는 사용자가 이미 최신 버전의 AWS CLI를 설치했거나 현재 버전으로 업데이트했다고 가정합니다. 자세한 정보는 AWS Command Line Interface 설치 섹션을 참조하세요.

보고서 그룹의 태그 보기

태그를 사용하면 AWS 리소스를 식별 및 구성하고 해당 리소스에 대한 액세스를 관리할 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례 백서를 참조하십시오. 태그 기반 액세스 정책의 예는 Deny or allow actions on report groups based on resource tags 단원을 참조하십시오.

보고서 그룹의 태그 보기(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 보고서 그룹과 연결된 태그를 볼 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 보려는 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다.

보고서 그룹의 태그 보기(AWS CLI)

AWS CLI를 사용하여 보고서 그룹에 대한 AWS 태그를 보려면 다음 단계를 수행하십시오. 태그가 추가되지 않은 경우 반환된 목록은 비어 있습니다.

1. 콘솔 또는 AWS CLI를 사용하여 보고서 그룹의 ARN을 찾습니다. 해당 ARN을 기록해 둡니다.

   AWS CLI

   다음 명령을 실행합니다.

   ```
   aws list-report-groups
   ```

   이 명령은 다음과 유사한 JSON 형식의 정보를 반환합니다.

   ```
   {
       "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"
       ]
   }
   ```

   보고서 그룹 ARN은 보고서 그룹의 ARN을 식별하는 데 사용할 수 있는 이름으로 끝납니다.

   Console

   1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
   2. 보고서 그룹에서 보려는 태그가 있는 보고서 그룹의 이름을 선택합니다.
   3. 구성에서 보고서 그룹의 ARN을 찾습니다.
2. 다음 명령을 실행합니다. `--report-group-arns` 파라미터에 대해 기록한 ARN을 사용합니다.

```
aws codebuild batch-get-report-groups --report-group-arns
 arn:aws:codebuild:region:123456789012:report-group/report-group-name
```

성공하면 이 명령은 다음과 유사한 tags 섹션이 포함된 JSON 형식의 정보를 반환합니다.

```
{
    ...
    "tags": {
        "Status": "Secret",
        "Project": "TestBuild"
    }
    ...
}
```

보고서 그룹의 태그 편집

보고서 그룹과 연결된 태그에 대한 값을 변경할 수 있습니다. 또한 키 이름을 변경할 수 있습니다. 이는 현재 태그를 제거하고 새 이름 및 다른 키와 동일한 값을 가진 다른 태그를 추가하는 것과 동일합니다. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있음을 알아두십시오. 자세한 내용은 Tags 섹션을 참조하세요.

> **Important**
>
> 보고서 그룹의 태그를 편집하면 해당 보고서 그룹에 대한 액세스에 영향을 줄 수 있습니다. 보고서 그룹에 대한 태그의 이름(키) 또는 값을 편집하기 전에 보고서 그룹과 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 Deny or allow actions on report groups based on resource tags 단원을 참조하십시오.

보고서 그룹의 태그 편집(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 보고서 그룹과 연결된 태그를 편집할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 편집할 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다.

4. 편집(Edit)을 선택합니다.
5. 다음 중 하나를 수행하세요.
   - 태그를 변경하려면 키에 새 이름을 입력합니다. 태그 이름을 변경하는 것은 태그를 제거하고 새 키 이름의 새 태그를 추가하는 것과 동일합니다.
   - 태그 값을 변경하려면 새 값을 입력합니다. 값을 어떤 것으로도 변경하지 않으려면 현재 값을 삭제하고 필드를 비워둡니다.
6. 태그 편집을 마쳤으면 제출을 선택합니다.

보고서 그룹의 태그 편집(AWS CLI)

보고서 그룹에서 태그를 추가, 변경 또는 삭제하려면 보고서 그룹 업데이트(CLI) 단원을 참조하십시오. `UpdateReportGroupInput.json`에서 태그를 업데이트합니다.

보고서 그룹에서 태그 제거

보고서 그룹과 연결된 태그를 하나 이상 제거할 수 있습니다. 태그를 제거할 때 해당 태그와 연결된 다른 AWS 리소스에서 해당 태그가 삭제되지는 않습니다.

> **Important**
>
> 보고서 그룹의 태그를 제거하면 해당 보고서 그룹에 대한 액세스에 영향을 줄 수 있습니다. 보고서 그룹에서 태그를 제거하기 전에 보고서 그룹과 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제 단원을 참조하십시오.

보고서 그룹에서 태그 제거(콘솔)

CodeBuild 콘솔을 사용하면 태그와 CodeBuild 보고서 그룹 간의 연결을 제거할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 제거할 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정(Settings)을 선택합니다.
4. 편집(Edit)을 선택합니다.
5. 제거할 태그를 찾은 다음 태그 제거를 선택합니다.
6. 태그 제거를 마쳤으면 제출을 선택합니다.

보고서 그룹에서 태그 제거(AWS CLI)

AWS CLI를 사용하여 CodeBuild 보고서 그룹에 대한 태그를 제거하려면 다음 단계를 수행하세요. 태그를 제거하면 태그는 삭제되지 않고 태그와 보고서 그룹 간의 연결만 제거됩니다.

> **Note**
>
> CodeBuild 보고서 그룹을 삭제하면 삭제된 보고서 그룹에서 모든 태그 연결이 제거됩니다. 보고서 그룹을 삭제하기 전에 태그를 제거할 필요가 없습니다.

보고서 그룹에서 하나 이상의 태그를 삭제하려면 보고서 그룹의 태그 편집(AWS CLI) 단원을 참조하십시오. 삭제하려는 태그가 포함되지 않은 업데이트된 태그 목록으로 JSON 형식 데이터의 `tags` 섹션을 업데이트합니다. 모든 태그를 삭제하려면 `tags` 섹션을 다음과 같이 업데이트하십시오.

```
"tags: []"
```

공유 보고서 그룹 작업

보고서 그룹 공유를 사용하여 여러 AWS 계정 또는 사용자가 보고서 그룹, 만료되지 않은 보고서 및 보고서의 테스트 결과를 볼 수 있습니다. 이 모델에서는 보고서 그룹을 소유하는 계정(소유자)은 다른 계정(소비자)과 보고서 그룹을 공유합니다. 소비자는 보고서 그룹을 편집할 수 없습니다. 보고서는 생성되고 30일 후에 만료됩니다.

목차

보고서 그룹 공유를 위한 전제 조건

보고서 그룹을 공유하려면 AWS 계정에서 보고서 그룹을 소유해야 합니다. 사용자와 공유된 보고서 그룹은 공유할 수 없습니다.

사용자와 공유된 보고서 그룹에 액세스하기 위한 전제 조건

공유 보고서 그룹에 액세스하려면 소비자의 IAM 역할에 `BatchGetReportGroups` 권한이 필요합니다. 다음 정책을 해당 IAM 역할에 연결할 수 있습니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetReportGroups"
    ]
}
```

자세한 내용은 ID 기반 정책 사용: AWS CodeBuild 섹션을 참조하세요.

관련 서비스

보고서 그룹을 공유하면 AWS Organizations를 통해 또는 AWS 계정과 AWS 리소스를 공유할 수 있게 해주는 서비스인 AWS Resource Access Manager(AWS RAM)와 통합됩니다. AWS RAM에서는 리소스와 리소스를 공유할 소비자를 지정하는 리소스 공유를 생성하여 소유한 리소스를 공유합니다. 소비자는 개인 AWS 계정, AWS Organizations의 조직 단위 또는 AWS Organizations의 전체 조직일 수 있습니다.

자세한 정보는 AWS RAM 사용 설명서를 참조하세요.

보고서 그룹 공유

보고서 그룹을 공유하면 소비자에게 보고서 그룹 및 해당 보고서에 대한 읽기 전용 액세스 권한이 부여됩니다. 소비자는 AWS CLI를 사용하여 보고서 그룹, 그 보고서 및 각 보고서의 테스트 케이스 결과를 볼 수 있습니다. 소비자는 다음을 수행할 수 없습니다.

- CodeBuild 콘솔에서 공유 보고서 그룹 또는 해당 보고서 보기.
- 공유 보고서 그룹 편집하기.
- 프로젝트에서 공유 보고서 그룹의 ARN을 사용하여 보고서 실행하기. 공유 보고서 그룹을 지정하는 프로젝트 빌드가 실패합니다.

CodeBuild 콘솔을 사용하여 기존 리소스 공유에 보고서 그룹을 추가할 수 있습니다. 보고서 그룹을 새 리소스 공유에 추가하려면 먼저 AWS RAM콘솔에서 보고서 그룹을 만들어야 합니다.

보고서 그룹을 조직 단위 또는 전체 조직과 공유하려면 AWS Organizations와의 공유를 활성화해야 합니다. 자세한 내용은 AWS RAM 사용 설명서에서 AWS Organizations를 사용하여 공유 사용을 참조하세요.

CodeBuild 콘솔, AWS RAM 콘솔 또는 AWS CLI를 사용하여 소유한 보고서 그룹을 공유할 수 있습니다.

소유한 보고서 그룹을 공유하려면(CodeBuild 콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.
3. 공유할 프로젝트를 선택한 다음 공유를 선택합니다. 자세한 내용은 AWS RAM 사용 설명서의 리소스 공유 생성을 참조하세요.

소유한 보고서 그룹을 공유하려면(AWS RAM 콘솔)

AWS RAM 사용 설명서의 리소스 공유 생성을 참조하세요.

소유한 보고서 그룹을 공유하려면(AWS RAM 명령)

create-resource-share 명령을 사용합니다.

소유한 보고서 그룹을 공유하려면(CodeBuild 명령)

put-resource-policy 명령을 사용합니다.

1. 이름이 `policy.json`인 파일을 만들고 다음으로 복사합니다.

```
{
   "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. 보고서 그룹 ARN 및 식별자로 policy.json을 업데이트하여 공유합니다. 다음 예제에서는 ARN arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group이 있는 보고서 그룹에 대한 읽기 전용 액세스 권한을 Alice와 123456789012로 식별된 AWS 계정의 루트 사용자에게 부여합니다.

```
{
   "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. 다음 명령을 실행합니다.

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://
policy.json
```

공유 보고서 그룹 공유 해제

보고서 및 테스트 케이스 결과를 포함해 공유되지 않은 보고서 그룹은 소유자만 액세스할 수 있습니다. 보고서 그룹 공유를 해제하면 이전에 공유한 AWS 계정이나 사용자가 보고서 그룹, 보고서 또는 보고서의 테스트 케이스 결과에 액세스할 수 없습니다.

소유하고 있는 공유된 보고서 그룹의 공유를 해제하려면 리소스 공유에서 제거해야 합니다. AWS RAM 콘솔 또는 AWS CLI를 사용하여 이 작업을 수행할 수 있습니다.

소유한 공유 보고서 그룹의 공유를 해제하려면 (AWS RAM 콘솔)

AWS RAM 사용 설명서에서 리소스 공유 업데이트를 참조하세요.

소유한 공유 보고서 그룹의 공유를 해제하려면(AWS RAM 명령)

disassociate-resource-share 명령을 사용합니다.

소유한 보고서 그룹 공유를 해제하려면(CodeBuild 명령)

delete-resource-policy 명령을 실행하고 공유를 해제할 보고서 그룹의 ARN을 지정합니다.

```
aws codebuild delete-resource-policy --resource-arn report-group-arn
```

공유 보고서 그룹 식별

소유자와 소비자는 AWS CLI를 사용하여 공유 보고서 그룹을 식별할 수 있습니다.

공유 보고서 그룹 및 해당 보고서에 대한 정보를 식별하고 가져오려면 다음 명령을 사용합니다.

- 공유된 보고서 그룹의 ARN을 보려면 list-shared-report-groups을 실행합니다.

  ```
  aws codebuild list-shared-report-groups
  ```

- 보고서 그룹에서 보고서의 ARN을 보려면 보고서 그룹 ARN을 사용하여 list-reports-for-report-group을 실행합니다.

  ```
  aws codebuild list-reports-for-report-group --report-group-arn report-group-arn
  ```

- 보고서의 테스트 케이스에 대한 정보를 보려면 보고서 ARN을 사용하여 describe-test-cases을 실행합니다.

  ```
  aws codebuild describe-test-cases --report-arn report-arn
  ```

  출력은 다음과 같습니다.

  ```
  {
      "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"
        }
    ]
}
```

공유 보고서 그룹 권한

소유자에 대한 권한

보고서 그룹 소유자는 보고서 그룹을 편집하고 프로젝트에서 보고서를 실행할 수 있도록 지정할 수 있습니다.

소비자에 대한 권한

보고서 그룹 소비자는 보고서 그룹, 보고서 및 보고서에 대한 테스트 케이스 결과를 볼 수 있습니다. 소비자는 보고서 그룹이나 해당 보고서를 편집할 수 없으며 이를 사용하여 보고서를 만들 수 없습니다.

보고서 작업

보고서 그룹에 지정된 테스트 케이스의 결과가 보고서에 포함되어 있습니다. 테스트 보고서는 빌드 프로젝트를 실행하는 동안 작성됩니다. 보고서 그룹, 테스트 케이스 파일 및 명령을 지정하여 buildspec 파일에서 테스트 케이스를 실행합니다. 테스트 케이스가 실행될 때마다 새로운 테스트 보고서가 보고서 그룹에 생성됩니다.

테스트 보고서는 생성되고 30일 후에 만료됩니다. 만료된 테스트 보고서는 볼 수 없지만 S3 버킷의 원시 테스트 결과 파일로 테스트 결과를 내보낼 수 있습니다. 내보낸 원시 테스트 파일은 만료되지 않습니다. 자세한 정보는 보고서 그룹 업데이트을 참조하세요.

테스트 보고서의 상태는 다음 중 하나일 수 있습니다.

- GENERATING: 테스트 케이스의 실행이 아직 진행 중입니다.
- DELETING: 테스트 보고서가 삭제 중입니다. 테스트 보고서가 삭제되면 그 테스트 케이스도 삭제됩니다. S3 버킷으로 내보낸 원시 테스트 결과 데이터 파일은 삭제되지 않습니다.
- INCOMPLETE: 테스트 보고서가 완료되지 않았습니다. 이 상태는 다음 이유 중 하나 때문에 반환될 수 있습니다.
  - 이 보고서의 테스트 케이스를 지정하는 보고서 그룹의 구성에 문제가 있는 경우. 예를 들어, buildspec 파일의 보고서 그룹 아래의 테스트 케이스 경로가 올바르지 않을 수 있습니다.
  - 빌드를 실행한 IAM 사용자에게 테스트를 실행할 권한이 없는 경우. 자세한 정보는 테스트 보고서 권한 작업을 참조하세요.
  - 테스트와 관련이 없는 오류로 인해 빌드가 완료되지 않은 경우.
- SUCCEEDED: 모든 테스트 케이스가 성공했습니다.
- FAILED: 일부 테스트 케이스는 성공하지 못했습니다.

각 테스트 케이스에서 상태를 반환합니다. 테스트 케이스의 상태는 다음 중 하나일 수 있습니다.

- SUCCEEDED: 테스트 케이스가 통과되었습니다.
- FAILED: 테스트 케이스가 실패했습니다.
- ERROR: 테스트 케이스로 인해 예기치 않은 오류가 발생했습니다.
- SKIPPED: 테스트 케이스가 실행되지 않았습니다.
- UNKNOWN: 테스트 케이스가 SUCCEEDED, FAILED, ERROR 또는 SKIPPED 이외의 상태를 반환했습니다.

테스트 보고서에는 최대 500개의 테스트 케이스 결과가 있을 수 있습니다. 500개가 넘는 테스트 사례를 실행하면 상태를 FAILED 기준으로 테스트의 CodeBuild 우선 순위를 지정하고 테스트 사례 결과를 잘라냅니다.

테스트 보고서 권한 작업

이 주제에서는 테스트 보고와 관련된 사용 권한에 대한 중요한 정보를 설명합니다.

주제

- 테스트 보고서의 역할 생성

- 테스트 보고 작업에 대한 권한
- 테스트 보고 권한 예제

테스트 보고서의 역할 생성

테스트 보고서를 실행하고 테스트 보고서를 포함하도록 프로젝트를 업데이트하려면 IAM 역할에 다음 권한이 필요합니다. 이러한 권한은 사전 정의된 관리형 정책에 포함됩니다. AWS 기존 빌드 프로젝트에 테스트 보고를 추가하려면 이러한 권한을 직접 추가해야 합니다.

- `CreateReportGroup`
- `CreateReport`
- `UpdateReport`
- `BatchPutTestCases`

코드 범위 보고서를 실행하려면 IAM 역할에 `BatchPutCodeCoverages` 권한도 포함되어야 합니다.

> **Note**
>
> `BatchPutTestCases`, `CreateReport`, `UpdateReport` 및 `BatchPutCodeCoverages`는 퍼블릭 권한이 아닙니다. 이러한 권한에 대해서는 해당 AWS CLI 명령 또는 SDK 메서드를 호출할 수 없습니다.

이러한 권한을 가지려면 다음 정책을 IAM 역할에 연결하면 됩니다.

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

이 정책은 사용해야 하는 보고서 그룹으로만 제한하는 것이 좋습니다. 다음은 정책에서 ARN이 두 개인 보고서 그룹으로만 권한을 제한합니다.

```
{
    "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"
    ]
}
```

다음은 이름이 `my-project`로 지정된 프로젝트의 빌드를 실행하여 만든 보고서 그룹으로만 권한을 제한합니다.

```
{
    "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**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드할 수 있는 권한에 사용됩니다.

테스트 보고 작업에 대한 권한

다음과 같은 테스트 보고 CodeBuild API 작업에 대한 권한을 지정할 수 있습니다.

- `BatchGetReportGroups`
- `BatchGetReports`
- `CreateReportGroup`
- `DeleteReportGroup`
- `DeleteReport`
- `DescribeTestCases`
- `ListReportGroups`
- `ListReports`
- `ListReportsForReportGroup`
- `UpdateReportGroup`

자세한 정보는 AWS CodeBuild 권한 참조을 참조하세요.

테스트 보고 권한 예제

테스트 보고와 관련된 샘플 정책에 대한 내용은 다음을 참조하십시오.

- 사용자가 보고서 그룹을 변경하도록 허용
- 사용자가 보고서 그룹을 생성하도록 허용
- 사용자가 보고서를 삭제하도록 허용
- 사용자가 보고서 그룹을 삭제하도록 허용
- 사용자가 보고서 그룹에 대한 정보를 가져오도록 허용
- 사용자가 보고서에 대한 정보를 가져오도록 허용
- 사용자가 보고서 그룹 목록을 가져오도록 허용
- 사용자가 보고서 목록을 가져오도록 허용
- 사용자가 보고서 그룹에 대한 보고서 목록을 가져오도록 허용
- 사용자가 보고서에 대한 테스트 케이스 목록을 가져오도록 허용

테스트 보고서 보기

테스트 케이스에 대한 정보, 통화 및 실패 횟수, 보고서 실행에 걸린 시간 등 테스트 보고서에 대한 세부 정보를 볼 수 있습니다. 빌드 실행, 보고서 그룹 또는 AWS 계정별로 그룹화된 테스트 보고서를 볼 수 있습니다. 콘솔에서 테스트 보고서를 선택하여 세부 정보 및 테스트 케이스의 결과를 확인합니다.

만료되지 않은 테스트 보고서를 볼 수 있습니다. 테스트 보고서는 작성되고 30일 후에 만료됩니다. 에서는 만료된 보고서를 볼 수 없습니다. CodeBuild

주제

빌드에 대한 테스트 보고서 보기

빌드에 대한 테스트 보고서를 보려면

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 보려는 빌드를 찾습니다. 테스트 보고서를 만든 빌드를 실행한 프로젝트를 알고 있는 경우:

   1. 탐색 창에서 프로젝트 빌드를 선택한 다음, 보고자 하는 테스트 보고서를 실행한 빌드가 있는 프로젝트를 선택합니다.
   2. 빌드 기록을 선택한 다음, 보고자 하는 보고서를 만들어 실행한 빌드를 선택합니다.

   AWS 계정의 빌드 기록에서도 빌드를 찾을 수 있습니다.

   1. 탐색 창에서 빌드 기록을 선택한 다음, 보고자 하는 테스트 보고서를 만든 빌드를 선택합니다.
3. 빌드 페이지에서 보고서를 선택한 다음 테스트 보고서를 선택하여 세부 정보를 확인합니다.

보고서 그룹에 대한 테스트 보고서 보기

보고서 그룹에서 테스트 보고서를 보려면

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.

3. 보려는 테스트 보고서가 포함된 보고서 그룹을 선택합니다.
4. 테스트 보고서를 선택하여 세부 정보를 확인합니다.

AWS 계정에서 테스트 보고서 보기

AWS 계정에서 테스트 보고서를 보려면

1. https://console.aws.amazon.com/codesuite/codebuild/home 에서 AWS CodeBuild 콘솔을 여십시오.
2. 탐색 창에서 보고서 기록을 선택합니다.
3. 테스트 보고서를 선택하여 세부 정보를 확인합니다.

테스트 프레임워크를 사용하여 테스트 보고

이 섹션의 항목에서는 다양한 테스트 프레임워크에 대한 테스트 보고를 설정하는 방법을 보여줍니다. AWS CodeBuild

주제

- Jasmine을 사용하여 테스트 보고 설정
- Jest를 사용하여 테스트 보고 설정
- pytest를 사용하여 테스트 보고 설정
- RSpec을 사용하여 테스트 보고 설정

Jasmine을 사용하여 테스트 보고 설정

다음 절차는 JasmineBDD 테스트 프레임워크의 AWS CodeBuild에서 테스트 보고를 설정하는 방법을 보여 줍니다.

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 Jasmine 테스트 프레임워크를 사용하도록 설정된 Node.js 프로젝트입니다.

jasmine-reporters 패키지를 프로젝트의 package.json 파일의 devDependencies 섹션에 추가합니다. 이 패키지에는 Jasmine과 함께 사용할 수 있는 JavaScript 리포터 클래스의 컬렉션이 있습니다.

```
npm install --save-dev jasmine-reporters
```

아직 없으면 프로젝트의 package.json 파일에 test 스크립트를 추가합니다. test 스크립트는 npm test가 실행될 때 Jasmine이 호출되도록 합니다.

```
{
  "scripts": {
    "test": "npx jasmine"
  }
}
```

CodeBuild는 다음과 같은 Jasmine 테스트 리포터를 지원합니다.

JUnitXmlReporter

JunitXml 형식으로 보고서를 생성하는 데 사용됩니다.

NUnitXmlReporter

NunitXml 형식으로 보고서를 생성하는 데 사용됩니다.

Jasmine과 함께 사용할 수 있는 Node.js 프로젝트에는 기본적으로 Jasmine 구성 및 테스트 스크립트를 포함하는 spec 하위 디렉터리가 있습니다.

JunitXML 형식으로 보고서를 생성하는 Jasmine을 구성하려면 테스트에 다음 코드를 추가하여 JUnitXmlReporter 리포터를 인스턴스화합니다.

```
var reporters = require('jasmine-reporters');

var junitReporter = new reporters.JUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(junitReporter);
```

NunitXML 형식으로 보고서를 생성하는 Jasmine을 구성하려면 테스트에 다음 코드를 추가하여 NUnitXmlReporter 리포터를 인스턴스화합니다.

```
var reporters = require('jasmine-reporters');

var nunitReporter = new reporters.NUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(nunitReporter)
```

테스트 보고서는 *<test report directory>*/*<report filename>*으로 지정된 파일로 내보내집니다.

buildspec.yml 파일에서 다음 섹션을 추가/업데이트합니다.

```
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>
```

NunitXml 보고서 형식을 사용하는 경우 file-format 값을 다음과 같이 변경합니다.

```
    file-format: NUNITXML
```

Jest를 사용하여 테스트 보고 설정

다음 절차는 Jest 테스트 프레임워크의 AWS CodeBuild에서 테스트 보고를 설정하는 방법을 보여 줍니다.

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 Jest 테스트 프레임워크를 사용하도록 설정된 Node.js 프로젝트입니다.

`jest-junit` 패키지를 프로젝트의 `package.json` 파일의 `devDependencies` 섹션에 추가합니다. CodeBuild는 이 패키지를 사용하여 `JunitXml` 형식으로 보고서를 생성합니다.

```
npm install --save-dev jest-junit
```

아직 없으면 프로젝트의 `package.json` 파일에 `test` 스크립트를 추가합니다. `test` 스크립트는 npm test가 실행될 때 Jest가 호출되도록 합니다.

```
{
  "scripts": {
    "test": "jest"
  }
}
```

Jest 구성 파일에 다음을 추가하여 `JunitXml` 리포터를 사용하도록 Jest를 구성하세요. 프로젝트에 Jest 구성 파일이 없는 경우, 프로젝트 루트에 `jest.config.js`라는 파일을 생성하고 다음을 추가하세요. 테스트 보고서는 *<test report directory>*/*<report filename>*으로 지정된 파일로 내보내집니다.

```
module.exports = {
  reporters: [
    'default',
    [ 'jest-junit', {
      outputDirectory: <test report directory>,
      outputName: <report filename>,
    } ]
  ]
};
```

buildspec.yml 파일에서 다음 섹션을 추가/업데이트합니다.

```
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>
```

pytest를 사용하여 테스트 보고 설정

다음 절차는 pytest 테스트 프레임워크의 AWS CodeBuild에서 테스트 보고를 설정하는 방법을 보여 줍니다.

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 pytest 테스트 프레임워크를 사용하도록 설정된 Python 프로젝트입니다.

buildspec.yml 파일의 build 또는 post_build 단계에 다음 항목을 추가합니다. 이 코드는 현재 디렉터리에서 테스트를 자동으로 검색하고 *<test report directory>*/*<report filename>*으로 지정된 파일로 테스트 보고서를 내보냅니다. 보고서는 JunitXml 형식을 사용합니다.

```
      - python -m pytest --junitxml=<test report directory>/<report filename>
```

buildspec.yml 파일에서 다음 섹션을 추가/업데이트합니다.

```
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
```

RSpec을 사용하여 테스트 보고 설정

다음 절차는 RSpec 테스트 프레임워크의 AWS CodeBuild에서 테스트 보고를 설정하는 방법을 보여 줍니다.

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 RSpec 테스트 프레임워크를 사용하도록 설정된 Node.js 프로젝트입니다.

`buildspec.yml` 파일에서 다음을 추가/업데이트합니다. 이 코드는 *<test source directory>* 디렉터리에서 테스트를 실행하고 *<test report directory>*/*<report filename>*으로 지정된 파일로 테스트 보고서를 내보냅니다. 보고서는 `JunitXml` 형식을 사용합니다.

```
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
```

코드 범위 보고서

CodeBuild 테스트용 코드 커버리지 보고서를 생성할 수 있습니다. 다음과 같은 코드 범위 보고서가 제공됩니다.

행 범위

행 범위는 테스트에서 다루는 명령문 수를 측정합니다. 명령문은 주석이나 조건문을 포함하지 않는 단일 명령입니다.

`line coverage = (total lines covered)/(total number of lines)`

분기 범위

분기 범위는 제어 구조의 가능한 모든 분기(예: `if` 또는 `case` 문) 중에서 테스트에 포함되는 분기 수를 측정합니다.

`branch coverage = (total branches covered)/(total number of branches)`

지원되는 코드 범위 보고서 파일 형식은 다음과 같습니다.

- JaCoCo XML
- SimpleCov JSON[1]
- Clover XML
- Cobertura XML
- LCOV 정보

CodeBuild [1]는 심플코브-json이 아닌 simplecov에서 생성한 JSON 코드 커버리지 보고서를 수락합니다.

코드 범위 보고서 생성

코드 커버리지 보고서를 만들려면 buildspec 파일에 하나 이상의 코드 커버리지 보고서 그룹이 구성되어 있는 빌드 프로젝트를 실행해야 합니다. CodeBuild 코드 커버리지 결과를 해석하고 실행에 대한 코드 커버리지 보고서를 제공합니다. 동일한 buildspec 파일을 사용하는 각 후속 빌드에 대해 새 테스트 보고서가 생성됩니다.

테스트 보고서를 작성하려면

1. 빌드 프로젝트를 생성합니다. 자세한 내용은 AWS CodeBuild에서 빌드 프로젝트 생성을 참조하세요.
2. 테스트 보고서 정보를 사용하여 프로젝트의 buildspec 파일을 구성합니다.

   a. `reports:`섹션을 추가하고 보고서 그룹의 이름을 지정하세요. CodeBuild 프로젝트 이름과 `project-name` - 형식으로 지정한 이름을 사용하여 보고서 그룹을 생성합니다`report-group-name-in-buildspec`. 사용할 보고서 그룹이 이미 있는 경우 해당 ARN을 지정합니다. ARN 대신 이름을 사용하는 경우 새 보고서 CodeBuild 그룹이 생성됩니다. 자세한 정보는 Reports syntax in the buildspec file을 참조하세요.

   b. 보고서 그룹에서 코드 범위 결과를 포함하는 파일의 위치를 지정합니다. 둘 이상의 보고서 그룹을 사용하는 경우 각 보고서 그룹에 대해 결과 파일 위치를 지정합니다. 빌드 프로젝트가 실행될 때마다 새 코드 범위 보고서가 생성됩니다. 자세한 정보는 테스트 파일 지정을 참조하세요.

   다음은 `results/jacoco-coverage-report.xml` test-에 있는 JaCoCo XML 결과 파일에 대한 코드 커버리지 보고서를 생성하는 예제입니다.

   ```
   reports:
     jacoco-report:
       files:
         - 'test-results/jacoco-coverage-report.xml'
       file-format: 'JACOCOXML'
   ```

   c. `build` 또는 `post_build` 시퀀스의 `commands` 섹션에서 코드 범위 분석을 실행하는 명령을 지정합니다. 자세한 정보는 테스트 명령 지정 을 참조하세요.
3. 빌드 프로젝트의 빌드를 실행합니다. 자세한 정보는 AWS CodeBuild에서 빌드 실행을 참조하세요.

4. 빌드가 완료되면 프로젝트 페이지의 빌드 기록에서 새 빌드 실행을 선택하십시오. 보고서를 선택하여 코드 범위 보고서를 확인합니다. 자세한 내용은 빌드에 대한 테스트 보고서 보기을(를) 참조하세요.

보고서 자동 검색

자동 검색 기능을 사용하면 빌드 단계가 완료된 후 모든 빌드 파일을 검색하고, 지원되는 모든 보고서 파일 유형을 검색하고, 새 테스트 및 코드 적용 범위 보고서 그룹과 보고서를 자동으로 생성합니다. CodeBuild 검색된 모든 보고서 유형에 대해 다음 패턴으로 새 보고서 그룹을 CodeBuild 생성합니다.

```
<project-name>-<report-file-format>-AutoDiscovered
```

> **Note**
>
> 검색된 보고서 파일의 형식 유형이 동일한 경우 검색된 보고서 파일은 동일한 보고서 그룹 또는 보고서에 배치됩니다.

보고서 자동 검색은 프로젝트 환경 변수로 구성됩니다.

`CODEBUILD_CONFIG_AUTO_DISCOVER`

이 변수는 빌드 중에 보고서 자동 검색을 비활성화할지 여부를 결정합니다. 기본적으로 보고서 자동 검색은 모든 빌드에 활성화되어 있습니다. 이 기능을 비활성화하려면 `CODEBUILD_CONFIG_AUTO_DISCOVER` 로 설정합니다. `false`

`CODEBUILD_CONFIG_AUTO_DISCOVER_DIR`

(선택 사항) 이 변수는 잠재적 보고서 파일을 CodeBuild 검색할 위치를 결정합니다. 기본적으로 CodeBuild 검색은 기본적으로 한다는 점에 유의하십시오. `**/*`

이러한 환경 변수는 빌드 단계에서 수정할 수 있습니다. 예를 들어 `main` git 브랜치의 빌드에 대한 보고서 자동 검색만 활성화하려는 경우 빌드 프로세스 중에 git 브랜치를 확인하고 빌드가 브랜치에 없으면 `CODEBUILD_CONFIG_AUTO_DISCOVER` false로 설정할 수 있습니다. `main` 콘솔이나 프로젝트 환경 변수를 사용하여 보고서 자동 검색을 비활성화할 수 있습니다.

주제

- 콘솔을 사용하여 보고서 자동 검색 구성

- 프로젝트 환경 변수를 사용하여 보고서 자동 검색 구성

콘솔을 사용하여 보고서 자동 검색 구성

콘솔을 사용하여 보고서 자동 검색을 구성하려면

1. 빌드 프로젝트를 만들거나 편집할 빌드 프로젝트를 선택합니다. 자세한 내용은 AWS CodeBuild에서 빌드 프로젝트 생성 또는 AWS CodeBuild에서 빌드 프로젝트 설정 변경 섹션을 참조하세요.
2. 환경에서 추가 구성을 선택합니다.
3. 보고서 자동 검색을 비활성화하려면 보고서 자동 검색에서 보고서 자동 검색 비활성화를 선택합니다.
4. (선택 사항) 디렉터리 자동 검색 - 선택 사항에서 지원되는 보고서 형식 파일을 검색할 디렉터리 패턴을 입력합니다. CodeBuild `**/*`기본적으로 CodeBuild 검색한다는 점에 유의하십시오.

프로젝트 환경 변수를 사용하여 보고서 자동 검색 구성

프로젝트 환경 변수를 사용하여 보고서 자동 검색을 구성하려면

1. 빌드 프로젝트를 만들거나 편집할 빌드 프로젝트를 선택합니다. 자세한 내용은 AWS CodeBuild에서 빌드 프로젝트 생성 또는 AWS CodeBuild에서 빌드 프로젝트 설정 변경 섹션을 참조하세요.
2. 환경 변수에서 다음을 수행하세요.

   a. 보고서 자동 검색을 비활성화하려면 이름에 을 **`CODEBUILD_CONFIG_AUTO_DISCOVER`** 입력하고 값에 를 입력합니다**`false`**. 이렇게 하면 보고서 자동 검색이 비활성화됩니다.

   b. (선택 사항) [**`CODEBUILD_CONFIG_AUTO_DISCOVER_DIR`**Name] 에 [Value] 를 입력하고 [Value] 에는 지원되는 보고서 형식 파일을 검색할 CodeBuild 디렉토리를 입력합니다. 예를 들어, `output` 디렉토리에서 `.xml` 파일을 `output/*xml` 검색합니다.

AWS CodeBuild의 로깅 및 모니터링

모니터링은 AWS CodeBuild와 사용자 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 중요한 역할을 합니다. 다중 지점 실패가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다. AWS는 CodeBuild 리소스 및 빌드를 모니터링하고 잠재적 인시던트에 대응하기 위한 다음 도구를 제공합니다.

주제

AWS CloudTrail을 사용하여 AWS CodeBuild API 직접 호출 로깅

AWS CodeBuild는 CodeBuild에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 CodeBuild 콘솔의 호출 및 CodeBuild API에 대한 코드 호출을 포함하여 CodeBuild에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 추적을 생성하면 CodeBuild 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 CodeBuild에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 AWS CodeBuild 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. CodeBuild에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서에서 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

CodeBuild에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. 추적은 CloudTrail이 S3 버킷으로 로그 파일을 전송할 수 있도록 합니다. 콘솔에서 추적을 생성하면 기본적으로 모든 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 S3 버킷으로 로그 파일을 전송합니다. CloudTrail 로그에 수집된 이벤트 데이터를 좀 더 분석하고 작업하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 추적 생성 개요

- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 지역에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 CodeBuild 작업은 CloudTrail에서 로깅되고 CodeBuild API 참조에 기록됩니다. 예를 들어 `CreateProject`(AWS CLI, `create-project`) 작업, `StartBuild`(AWS CLI, `start-project`) 작업, `UpdateProject`(AWS CLI, `update-project`) 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 사용자 보안 인증으로 했는지 여부
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail userIdentity 요소를 참조하세요.

AWS CodeBuild 로그 파일 항목 이해

추적이란 지정한 S3 버킷에 이벤트를 로그 파일로 입력할 수 있도록 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

> **Note**
>
> 중요한 정보를 보호하기 위해 CodeBuild 로그에 다음 항목이 숨겨져 있습니다.
>
> - AWS액세스 키 ID: 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.
> - 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.

- AWS Secrets Manager을 사용하여 지정한 문자열입니다. 자세한 내용은 키 관리 섹션을 참조하세요.

다음은 CodeBuild에서 빌드 프로젝트 만들기 작업을 보여 주는 CloudTrail 로그 항목의 예입니다.

```
{
  "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"
}
```

AWS CodeBuild 모니터링

Amazon CloudWatch를 사용하여 빌드를 관찰하고, 문제 발생 시 보고하고, 적절한 경우 자동 조치를 취할 수 있습니다. 두 수준에서 빌드를 모니터링할 수 있습니다.

프로젝트 수준

이러한 지표는 지정된 프로젝트의 모든 빌드에 대한 것입니다. 프로젝트의 지표를 보려면 CloudWatch의 차원에 `ProjectName`을 지정합니다.

AWS 계정 수준

이러한 지표는 한 계정의 모든 빌드에 대한 것입니다. AWS 계정 수준에서 지표를 보려면 CloudWatch에 차원을 입력하지 마세요. 빌드 리소스 사용률 지표는 AWS 계정 수준에서 사용할 수 없습니다.

CloudWatch 지표는 일정 기간 동안의 빌드의 양상을 보여 줍니다. 예를 들면, 다음을 모니터링할 수 있습니다.

- 빌드 프로젝트 또는 AWS 계정에서 일정 기간 동안 시도된 빌드 수.
- 빌드 프로젝트 또는 AWS 계정에서 일정 기간 동안 성공한 빌드 수.
- 빌드 프로젝트 또는 AWS 계정에서 일정 기간 동안 실패한 빌드 수.
- CodeBuild가 일정 기간 동안 빌드 프로젝트 또는 AWS 계정에서 빌드를 실행하는 데 소요된 시간.
- 빌드 또는 전체 빌드 프로젝트의 빌드 리소스 사용률입니다. 빌드 리소스 사용률 지표에는 CPU, 메모리, 스토리지 사용률과 같은 지표가 포함됩니다.

자세한 내용은 CodeBuild 지표 모니터링 섹션을 참조하세요.

CodeBuild CloudWatch 지표

AWS 계정 또는 빌드 프로젝트를 기준으로 다음 지표를 추적할 수 있습니다.

BuildDuration

빌드의 `BUILD` 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

빌드

트리거된 빌드 수를 측정합니다.

단위: 개수

유효한 CloudWatch 통계: 합계

DownloadSourceDuration

빌드의 `DOWNLOAD_SOURCE` 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

Duration

일정 기간 동안 모든 빌드 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

FailedBuilds

클라이언트 오류 또는 시간 초과로 인해 실패한 빌드 수를 측정합니다.

단위: 개수

유효한 CloudWatch 통계: 합계

FinalizingDuration

빌드의 `FINALIZING` 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

InstallDuration

빌드의 `INSTALL` 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

PostBuildDuration

빌드의 `POST_BUILD` 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

PreBuildDuration

빌드의 `PRE_BUILD` 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

ProvisioningDuration

빌드의 PROVISIONING 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

QueuedDuration

빌드의 QUEUED 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

SubmittedDuration

빌드의 SUBMITTED 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

SucceededBuilds

성공한 빌드 수를 측정합니다.

단위: 개수

유효한 CloudWatch 통계: 합계

UploadArtifactsDuration

빌드의 UPLOAD_ARTIFACTS 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

CodeBuild CloudWatch 리소스 사용률 지표

> **Note**
>
> CodeBuild 리소스 사용률 지표는 다음 리전에서만 사용할 수 있습니다.

- Asia Pacific (Tokyo) Region
- 아시아 태평양(서울) 리전
- 아시아 태평양(뭄바이) 리전
- Asia Pacific (Singapore) Region
- Asia Pacific (Sydney) Region
- 캐나다(중부) 리전
- 유럽(프랑크푸르트) 리전
- Europe (Ireland) Region
- 유럽(런던) 리전
- 유럽(파리) 리전
- South America (São Paulo) Region
- US East (N. Virginia) Region
- 미국 동부(오하이오) 리전
- 미국 서부(캘리포니아 북부) 리전
- 미국 서부(오레곤) 리전

다음 리소스 사용률 지표를 추적할 수 있습니다.

CpuUtilized

빌드 컨테이너에서 사용하는 할당된 처리의 CPU 단위 수입니다.

단위: CPU 단위

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

CPUUtilizedPercent

빌드 컨테이너에서 사용된 할당된 처리의 비율입니다.

단위: 백분율

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

MemoryUtilized

빌드 컨테이너가 사용된 메모리 크기(메가바이트)입니다.

단위: 메가바이트

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

MemoryUtilizedPercent

빌드 컨테이너에서 사용된 할당된 메모리의 비율입니다.

단위: 백분율

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

StorageReadBytes

빌드 컨테이너에서 사용된 스토리지 읽기 속도입니다.

단위: 바이트/초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

StorageWriteBytes

빌드 컨테이너에서 사용된 스토리지 쓰기 속도입니다.

단위: 바이트/초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

CodeBuild CloudWatch 차원

CodeBuild는 다음과 같은 CloudWatch 지표 차원을 제공합니다. 어떤 지표도 지정하지 않으면 현재 AWS 계정에 대한 지표가 측정됩니다.

BuildId, BuildNumber, ProjectName

빌드 식별자, 빌드 번호, 프로젝트 이름에 대한 지표가 제공됩니다.

ProjectName

프로젝트 이름에 대한 지표가 제공됩니다.

CodeBuild CloudWatch 경보

CloudWatch 콘솔을 사용하여 CodeBuild 지표를 기준으로 경보를 생성함으로써 빌드에 문제가 생길 경우 조치를 취할 수 있습니다. 다음은 경보에 가장 유용한 두 가지 지표입니다.

- `FailedBuild`. 지정한 시간(초) 내에 특정 수의 실패한 빌드 수가 감지될 경우 트리거되는 경보를 만들 수 있습니다. CloudWatch에서 경보를 발생시킬 실패 빌드 수와 시간(초)을 지정합니다.
- `Duration`. 빌드가 예상한 것보다 오래 걸릴 경우 트리거되는 경보를 만들 수 있습니다. 빌드가 시작된 후 완료되기까지 걸리는 시간(초)을 지정하여 이 시간을 초과할 경우 경보를 생성하도록 지정합니다.

CodeBuild 지표에 대한 경보를 만드는 방법은 CloudWatch 경보를 사용한 빌드 모니터링 섹션을 참조하세요. 경보에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 생성을 참조하세요.

CodeBuild 지표 모니터링

AWS CodeBuild는 사용자를 대신해 함수를 모니터링하고 Amazon CloudWatch를 통해 지표를 보고합니다. 이러한 지표에는 총 빌드, 실패한 빌드, 성공한 빌드, 빌드 기간 등이 포함됩니다.

CodeBuild 콘솔 또는 CloudWatch 콘솔을 사용하여 CodeBuild에 대한 지표를 모니터링할 수 있습니다. 다음 절차에서는 지표에 액세스하는 방법을 보여 줍니다.

주제

- 빌드 지표 액세스(CodeBuild 콘솔)
- 빌드 지표 액세스(Amazon CloudWatch 콘솔)

빌드 지표 액세스(CodeBuild 콘솔)

> **Note**
>
> CodeBuild 콘솔에서는 지표를 표시하는 데 사용되는 지표나 그래프를 사용자 지정할 수 없습니다. 디스플레이를 사용자 지정하려면 Amazon CloudWatch 콘솔을 사용하여 빌드 지표를 확인합니다.

계정 수준 지표

AWS 계정 수준 지표에 액세스하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.

2. 탐색 창에서 계정 지표를 선택합니다.

프로젝트 수준 지표

프로젝트 수준 지표에 액세스하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다.
3. 빌드 프로젝트 목록의 이름 열에서 지표를 보려는 프로젝트를 선택합니다.
4. 지표 탭을 선택합니다.

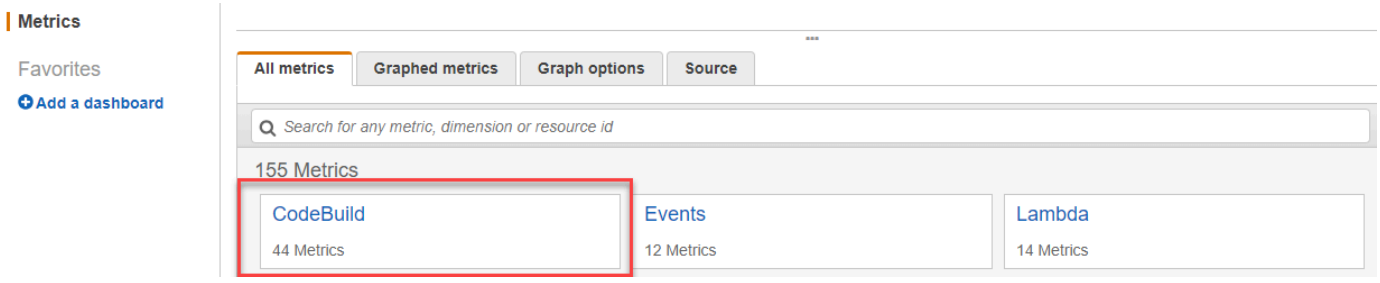
빌드 지표 액세스(Amazon CloudWatch 콘솔)

CloudWatch 콘솔에서 지표 및 지표를 표시하는 데 사용되는 그래프를 사용자 지정할 수 있습니다.

계정 수준 지표

계정 수준 지표에 액세스하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [지표(Metrics)]를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.

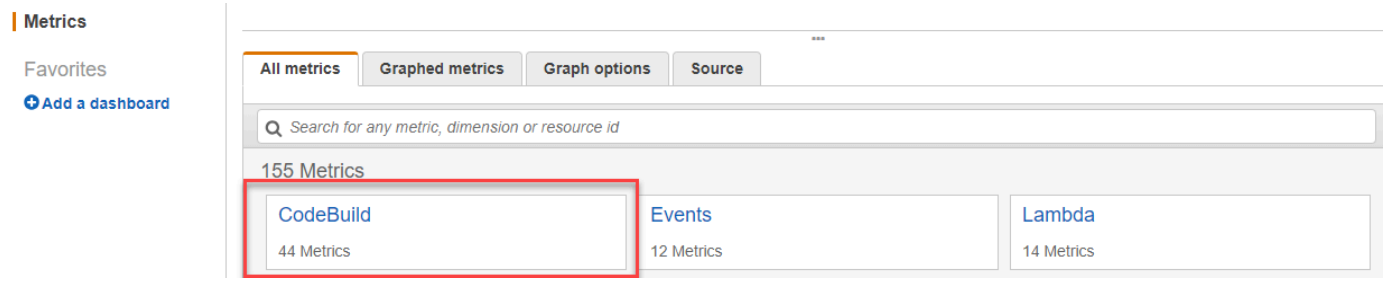


4. 계정 지표를 선택합니다.
5. 하나 이상의 프로젝트 및 지표를 선택합니다. 각 프로젝트에 대해 SucceededBuilds, FailedBuilds, Builds 및 Duration 지표를 선택할 수 있습니다. 선택한 모든 프로젝트 및 지표 조합이 페이지의 그래프에 표시됩니다.

프로젝트 수준 지표

프로젝트 수준 지표에 액세스하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [지표(Metrics)]를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.



4. 프로젝트별을 선택합니다.
5. 하나 이상의 프로젝트 및 지표 조합을 선택합니다. 각 프로젝트에 대해 SucceededBuilds, FailedBuilds, Builds 및 Duration 지표를 선택할 수 있습니다. 선택한 모든 프로젝트 및 지표 조합이 이 페이지의 그래프에 표시됩니다.
6. (선택 사항) 지표 및 그래프를 사용자 지정할 수 있습니다. 예를 들어 통계 열의 드롭다운 목록에서 표시할 다른 통계를 선택할 수 있습니다. 또는 기간 열의 드롭다운 메뉴에서 지표를 모니터링하는 데 사용할 다른 기간을 선택할 수 있습니다.

   자세한 내용은 Amazon CloudWatch 사용 설명서의 그래프 지표 및 사용 가능한 지표 보기를 참조하세요.

CodeBuild 리소스 사용률 지표 모니터링

AWS CodeBuild는 사용자를 대신해 빌드 리소스 사용률을 모니터링하고 Amazon CloudWatch를 통해 지표를 보고합니다. 여기에는 CPU, 메모리, 스토리지 사용률과 같은 지표가 포함됩니다.

> **Note**
>
> CodeBuild 리소스 사용률 지표는 1분 넘게 실행되는 빌드에 대해서만 기록됩니다.

CodeBuild 콘솔 또는 CloudWatch 콘솔을 사용하여 CodeBuild에 대한 리소스 사용률 지표를 모니터링할 수 있습니다.

> **Note**
>
> CodeBuild 리소스 사용률 지표는 다음 리전에서만 사용할 수 있습니다.
>
> - Asia Pacific (Tokyo) Region
> - 아시아 태평양(서울) 리전
> - 아시아 태평양(뭄바이) 리전
> - Asia Pacific (Singapore) Region
> - Asia Pacific (Sydney) Region
> - 캐나다(중부) 리전
> - 유럽(프랑크푸르트) 리전
> - Europe (Ireland) Region
> - 유럽(런던) 리전
> - 유럽(파리) 리전
> - South America (São Paulo) Region
> - US East (N. Virginia) Region
> - 미국 동부(오하이오) 리전
> - 미국 서부(캘리포니아 북부) 리전
> - 미국 서부(오레곤) 리전

다음 절차에서는 리소스 사용률 지표에 액세스하는 방법을 보여 줍니다.

주제

- 리소스 사용률 지표 액세스(CodeBuild 콘솔)
- 리소스 사용률 지표 액세스(Amazon CloudWatch 콘솔)

리소스 사용률 지표 액세스(CodeBuild 콘솔)

> **Note**
>
> CodeBuild 콘솔에서는 지표를 표시하는 데 사용되는 지표나 그래프를 사용자 지정할 수 없습니다. 디스플레이를 사용자 지정하려면 Amazon CloudWatch 콘솔을 사용하여 빌드 지표를 확인합니다.

프로젝트 수준 리소스 사용률 지표

프로젝트 수준 리소스 사용률 지표에 액세스하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build projects]를 선택합니다.
3. 빌드 프로젝트 목록의 이름 열에서 사용률 지표를 보려는 프로젝트를 선택합니다.
4. 지표 탭을 선택합니다. 리소스 사용률 지표는 리소스 사용률 지표 섹션에 표시됩니다.
5. CloudWatch 콘솔에서 프로젝트 수준 리소스 사용률 지표를 보려면 리소스 사용률 지표 섹션에서 CloudWatch에서 보기를 선택합니다.

빌드 수준의 리소스 사용률 지표

빌드 수준 리소스 사용률 지표에 액세스하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/codesuite/codebuild/home에서 AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build history]를 선택합니다.
3. 빌드 목록의 빌드 실행 열에서 사용률 지표를 보려는 빌드를 선택합니다.
4. 리소스 사용률 탭을 선택합니다.
5. CloudWatch 콘솔에서 빌드 수준 리소스 사용률 지표를 보려면 리소스 사용률 지표 섹션에서 CloudWatch에서 보기를 선택합니다.

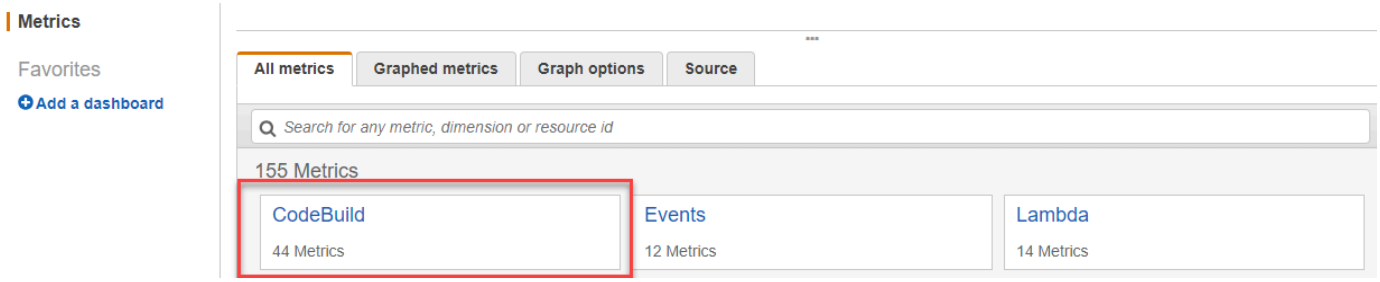
리소스 사용률 지표 액세스(Amazon CloudWatch 콘솔)

Amazon CloudWatch 콘솔을 사용하여 CodeBuild 리소스 사용률 지표에 액세스할 수 있습니다.

프로젝트 수준 리소스 사용률 지표

프로젝트 수준 리소스 사용률 지표에 액세스하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [지표(Metrics)]를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.



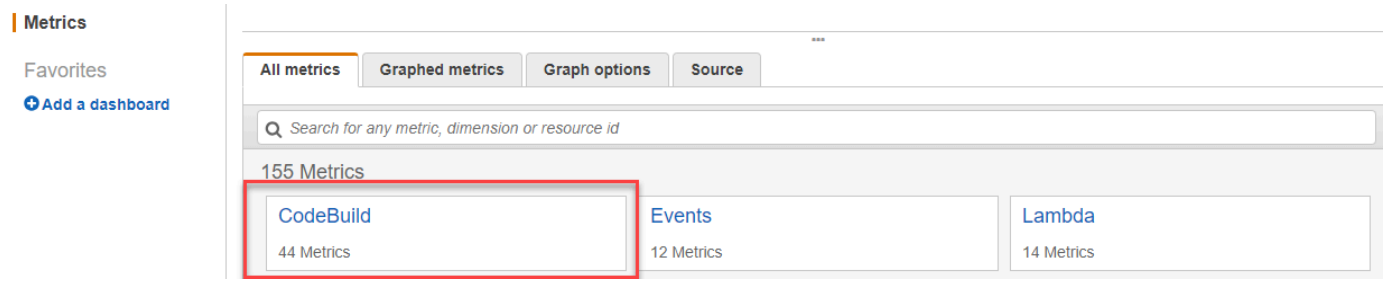
4. 프로젝트별을 선택합니다.
5. 그래프에 추가할 하나 이상의 프로젝트와 지표 조합을 선택합니다. 선택한 모든 프로젝트 및 지표 조합이 페이지의 그래프에 표시됩니다.
6. (선택 사항) 그래프로 표시된 지표 탭에서 지표와 그래프를 사용자 지정할 수 있습니다. 예를 들어 통계 열의 드롭다운 목록에서 표시할 다른 통계를 선택할 수 있습니다. 또는 기간 열의 드롭다운 메뉴에서 지표를 모니터링하는 데 사용할 다른 기간을 선택할 수 있습니다.

   자세한 내용은 Amazon CloudWatch 사용 설명서의 그래프 지표 및 사용 가능한 지표 보기를 참조하세요.

빌드 수준의 리소스 사용률 지표

빌드 수준 리소스 사용률 지표에 액세스하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [지표(Metrics)]를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.



4. BuildId, BuildNumber, ProjectName을 선택합니다.
5. 그래프에 추가할 하나 이상의 빌드와 지표 조합을 선택합니다. 선택한 모든 빌드 및 지표 조합이 페이지의 그래프에 표시됩니다.
6. (선택 사항) 그래프로 표시된 지표 탭에서 지표와 그래프를 사용자 지정할 수 있습니다. 예를 들어 통계 열의 드롭다운 목록에서 표시할 다른 통계를 선택할 수 있습니다. 또는 기간 열의 드롭다운 메뉴에서 지표를 모니터링하는 데 사용할 다른 기간을 선택할 수 있습니다.

   자세한 내용은 Amazon CloudWatch 사용 설명서의 그래프 지표 및 사용 가능한 지표 보기를 참조하세요.

CloudWatch 경보를 사용한 빌드 모니터링

빌드에 대한 CloudWatch 경보를 생성할 수 있습니다. 경보는 지정한 기간에 단일 메트릭을 감시하고 여러 기간에 지정된 임계값에 대한 메트릭 값을 기준으로 작업을 하나 이상 수행합니다. 기본 CloudWatch 경보 기능을 사용하면 임계값 초과 시 CloudWatch에서 지원하는 모든 작업을 지정할 수 있습니다. 예를 들어, 15분 이내에 계정의 빌드가 4개 이상 실패할 경우 Amazon SNS 알림이 전송되도록 지정할 수 있습니다.

CodeBuild 지표에 대한 CloudWatch 경보를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택하세요.
3. 경보 생성을 선택합니다.
4. 범주별 CloudWatch 지표에서 CodeBuild 지표를 선택합니다. 프로젝트 수준 지표만 사용하려는 경우 프로젝트별을 선택합니다. 계정 수준 지표만 사용하려는 경우 계정 지표를 선택합니다.
5. 경보 생성에서 아직 선택하지 않은 경우 지표 선택을 선택합니다.
6. 경보를 생성하려는 지표를 선택합니다. 옵션은 프로젝트별 또는 계정 지표입니다.

7. 다음 또는 경보 정의를 선택한 다음, 경보를 생성합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 생성을 참조하세요. 경보가 트리거될 때 Amazon SNS 알림을 설정하는 방법에 대한 자세한 내용은 Amazon SNS 개발자 안내서의 Amazon SNS 알림 설정을 참조하세요.
8. 경보 생성을 선택합니다.

보안은 AWS CodeBuild

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안 및 규정 준수는 기업과 귀사 간의 AWS 공동 책임입니다. 이 공유 모델은 호스트 운영 체제 및 가상화 계층에서 서비스 시설의 물리적 보안에 이르기까지 구성 요소를 운영, 관리 및 제어하는 등 운영 부담을 줄이는 데 도움이 될 수 있습니다. AWS 고객은 게스트 운영 체제(업데이트 및 보안 패치 포함) 및 기타 관련 애플리케이션 소프트웨어에 대한 책임과 관리를 담당합니다. 또한 AWS 제공된 보안 그룹 방화벽의 구성도 사용자 책임입니다. 고객의 책임은 사용하는 서비스, 이러한 서비스를 고객 IT 환경에 통합, 적용되는 법규 및 규정에 따라 달라집니다. 따라서 고객의 조직이 사용하는 서비스를 신중하게 고려해야 합니다. 자세한 내용은 공동 책임 모델을 참조하십시오.

CodeBuild 리소스를 보호하는 방법을 알아보려면 다음 항목을 참조하십시오.

주제

- 데이터 보호: AWS CodeBuild
- 의 ID 및 액세스 관리 AWS CodeBuild
- 규정 준수 검증: AWS CodeBuild
- 의 레질리언스 AWS CodeBuild
- 의 인프라 보안 AWS CodeBuild
- 에서 소스 제공자에 액세스하세요. CodeBuild
- 교차 서비스 혼동된 대리인 방지

데이터 보호: AWS CodeBuild

AWS 공동 책임 모델 의 데이터 보호에 적용됩니다 AWS CodeBuild. 이 모델에 설명된 대로 AWS 는 모든 데이터를 실행하는 글로벌 인프라를 보호하는 역할을 AWS 클라우드합니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은AWS 보안 블로그에서 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사

용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API CodeBuild 또는 AWS 서비스 SDK를 사용하거나 다른 방법으로 작업하는 경우가 포함됩니다. AWS CLI AWS 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

민감한 정보를 보호하기 위해 다음과 같은 내용이 CodeBuild 로그에 숨겨져 있습니다.

- CodeBuild 프로젝트 환경 변수 또는 `env/parameter-store` buildspec 섹션의 파라미터 저장소를 사용하여 지정된 문자열. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
- CodeBuild 프로젝트 환경 변수 또는 AWS Secrets Manager buildspec 섹션에서 사용하여 지정된 문자열입니다. `env/secrets-manager` 자세한 정보는 키 관리을 참조하세요.

데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

주제

- 데이터 암호화
- 키 관리
- 트래픽 개인 정보 보호

데이터 암호화

암호화는 보안의 중요한 부분입니다. CodeBuild 일부 암호화(예: 전송 중인 데이터 암호화)는 기본으로 제공되며 어떠한 것도 필요하지 않습니다. 기타 암호화(예: 유휴 상태의 데이터 암호화)는 프로젝트나 빌드 생성 시 구성할 수 있습니다.

- 저장 데이터 암호화 - 캐시, 로그, 내보낸 원시 테스트 보고서 데이터 파일, 빌드 결과 등의 빌드 아티팩트는 기본적으로 를 사용하여 암호화됩니다. AWS 관리형 키이러한 KMS 키를 사용하지 않으려면 고객 관리형 키를 생성하고 구성해야 합니다. KMS 키 생성 및 AWS 키 관리 서비스 개념에 대한 자세한 내용은AWS Key Management Service 사용 설명서를 참조하세요.
  - 빌드 출력 아티팩트를 암호화하는 데 CodeBuild 사용하는 AWS KMS 키의 식별자를 환경 변수에 저장할 수 있습니다. `CODEBUILD_KMS_KEY_ID` 자세한 내용은 빌드 환경의 환경 변수단원을 참조하세요.
  - 빌드 프로젝트 생성 시 고객 관리형 키를 지정할 수 있습니다. 자세한 내용은 Set the Encryption Key Using the Console 및 CLI를 사용하여 암호화 키 설정을 참조하세요.

  빌드 플릿의 Amazon Elastic Block Store 볼륨은 기본적으로 를 사용하여 암호화됩니다 AWS 관리형 키.
- 전송 중인 데이터의 암호화 - 고객 간 및 고객 간 통신 CodeBuild 및 다운스트림 종속성은 서명 버전 4 서명 프로세스를 사용하여 서명된 TLS 연결을 사용하여 보호됩니다. CodeBuild 모든 CodeBuild 엔드포인트는 에서 관리하는 SHA-256 인증서를 사용합니다. AWS Private Certificate Authority자세한 내용은 서명 버전 4 서명 프로세스 및 ACM PCA란 무엇입니까를 참조하십시오.
- 빌드 아티팩트 암호화 - 빌드 프로젝트와 관련된 CodeBuild 서비스 역할을 수행하려면 빌드 출력 아티팩트를 암호화하기 위해 KMS 키에 액세스해야 합니다. 기본적으로 AWS 계정에서 Amazon AWS 관리형 키 S3용으로 를 CodeBuild 사용합니다. 이 AWS 관리형 키를 사용하지 않으려면 고객 관리형 키를 생성 및 구성해야 합니다. 자세한 내용은 AWS KMS 개발자 안내서의 고객 관리형 키 생성 및 키 생성을 참조하세요.

키 관리

암호화를 통해 콘텐츠의 무단 사용을 방지할 수 있습니다. 암호화 키를 저장한 다음 빌드 프로젝트와 관련된 CodeBuild 서비스 역할에 Secrets Manager 계정에서 암호화 키를 받을 수 있는 권한을 부여합니다. AWS Secrets Manager자세한 내용은 CodeBuild용 고객 관리형 키를 생성하고 구성합니다., AWS CodeBuild에서 빌드 프로젝트 생성, AWS CodeBuild에서 빌드 실행 및 자습서: 보안 암호 저장 및 검색을 참조하십시오.

빌드 명령에서 CODEBUILD_KMS_KEY_ID 환경 변수를 사용하여 AWS KMS 키 식별자를 확보하세요. 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.

Secrets Manager를 사용하여 런타임 환경에 사용된 도커 이미지를 저장하는 프라이빗 레지스트리의 보안 인증을 보호할 수 있습니다. 자세한 정보는 샘플이 포함된 AWS Secrets Manager 사설 레지스트리 CodeBuild을 참조하세요.

트래픽 개인 정보 보호

인터페이스 VPC 엔드포인트를 사용하도록 CodeBuild 구성하여 빌드의 보안을 개선할 수 있습니다. 이를 위해 인터넷 게이트웨이, NAT 디바이스 또는 가상 프라이빗 게이트웨이가 필요 없습니다. 또한 구성하지 않아도 PrivateLink 되지만 권장되는 방법도 있습니다. 자세한 정보는 VPC 엔드포인트 사용을 참조하세요. VPC 엔드포인트에 대한 자세한 내용은 PrivateLink 및 VPC 엔드포인트를 통한 서비스 액세스를 AWS 참조하십시오 AWS PrivateLink. PrivateLink

의 ID 및 액세스 관리 AWS CodeBuild

에 액세스하려면 자격 증명이 AWS CodeBuild 필요합니다. 이러한 자격 증명에는 S3 버킷에 빌드 아티팩트를 저장 및 검색하고 빌드용 Amazon CloudWatch Logs 보기와 같은 AWS 리소스에 액세스할 수 있는 권한이 있어야 합니다. 다음 섹션에서는 AWS Identity and Access Management(IAM) 사용 방법과 리소스에 대한 보안 액세스를 지원하는 CodeBuild 방법을 설명합니다.

리소스에 대한 액세스 권한 관리 개요 AWS CodeBuild

모든 AWS 리소스는 AWS 계정이 소유하며 리소스를 만들거나 액세스할 수 있는 권한은 권한 정책에 의해 관리됩니다. 계정 관리자는 IAM 자격 증명(사용자, 그룹 및 역할)에 권한 정책을 연결할 수 있습니다.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하십시오.

권한을 부여할 때는 권한을 부여 받을 사용자, 사용자가 액세스할 수 있는 리소스 및 해당 리소스에 수행할 수 있는 작업을 결정합니다.

주제

- AWS CodeBuild 리소스 및 운영
- 리소스 소유권 이해
- 리소스 액세스 관리
- 정책 요소 지정: 작업, 효과, 보안 주체

AWS CodeBuild 리소스 및 운영

에서 AWS CodeBuild기본 리소스는 빌드 프로젝트입니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다. 빌드도 리소스이며 빌드에는 이와 연결된 ARN이 들어 있습니다. 자세한 내용은 의 Amazon 리소스 이름 (ARN) 및 AWS 서비스 네임스페이스를 참조하십시오. Amazon Web Services 일반 참조

리소스 유형	ARN 형식
빌드 프로젝트	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:project/` *project-name*
빌드	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:build/` *build-ID*
보고서 그룹	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:report-group/` *report-group-name*
보고서	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:report/` *report-ID*
모든 리소스 CodeBuild	`arn:aws:codebuild:*`
지정된 AWS 지역의 지정된 계정이 소유한 모든 CodeBuild 리소스	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:*`

> **Note**
>
> 대부분의 AWS 서비스는 ARN에서 콜론 (:) 또는 전방향 슬래시 (/) 를 동일한 문자로 취급합니다. 하지만 에서는 리소스 패턴 및 규칙이 정확히 일치하는 항목을 CodeBuild 사용합니다. 따

라서 이벤트 패턴을 만들 때 리소스에서 ARN 구문이 일치하도록 정확한 문자를 사용해야 합니다.

예를 들어 다음과 같이 ARN을 사용하여 명령문에 특정 빌드 프로젝트 (*myBuildProject*) 를 표시할 수 있습니다.

```
"Resource": "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject"
```

모든 리소스를 지정해야 하거나, API 작업이 ARN을 지원하지 않는 경우 다음과 같이 Resource 요소에 와일드카드 문자(*)를 사용합니다.

```
"Resource": "*"
```

일부 CodeBuild API 작업은 여러 리소스를 허용합니다 (예:BatchGetProjects). 명령문 하나에 여러 리소스를 지정하려면 다음과 같이 각 ARN을 쉼표로 구분합니다.

```
"Resource": [
  "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject",
  "arn:aws:codebuild:us-east-2:123456789012:project/myOtherBuildProject"
]
```

CodeBuild CodeBuild 리소스로 작업하기 위한 일련의 작업을 제공합니다. 목록을 보려면 AWS CodeBuild 권한 참조 섹션을 참조하세요.

리소스 소유권 이해

리소스를 만든 사람이 누구인지와 상관없이 계정에서 생성된 리소스는 계정을 소유합니다. AWS 구체적으로, 리소스 소유자는 리소스 생성 요청을 인증하는 보안 주체 (즉, 루트 계정, 사용자 또는 IAM 역할) 의 계정입니다. AWS 다음 예에서는 이러한 작동 방식을 설명합니다.

- 계정의 루트 계정 자격 증명을 사용하여 규칙을 생성하는 경우 해당 AWS 계정이 리소스의 CodeBuild 소유자가 됩니다. AWS
- AWS 계정에서 사용자를 생성하고 해당 사용자에게 CodeBuild 리소스를 생성할 권한을 부여하면 사용자가 CodeBuild 리소스를 생성할 수 있습니다. 하지만 사용자가 속한 AWS 계정이 CodeBuild 리소스를 소유합니다.
- AWS 계정에서 리소스를 생성할 권한이 있는 IAM 역할을 생성하는 경우, 역할을 수임할 수 있는 사람은 누구나 CodeBuild 리소스를 생성할 CodeBuild 수 있습니다. 역할이 속한 AWS 계정이 리소스를 소유합니다. CodeBuild

리소스 액세스 관리

권한 정책은 누가 어떤 리소스에 액세스 할 수 있는지를 나타냅니다.

> **Note**
>
> 이 섹션에서는 AWS CodeBuild에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. IAM 설명서 전체 내용은 IAM 사용 설명서의 IAM이란 무엇입니까? 섹션을 참조하세요. IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

IAM 보안 인증에 연결된 정책을 보안 인증 기반 정책(IAM 정책)이라고 합니다. 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다. CodeBuild 계정 간 리소스 공유를 위해 ID 기반 정책과 특정 읽기 전용 API에 대한 리소스 기반 정책을 지원합니다.

S3 버킷에 대한 보안 액세스

CodeBuild 프로젝트와 관련된 S3 버킷이 본인 또는 신뢰할 수 있는 사람의 소유인지 확인하려면 IAM 역할에 다음 권한을 포함하는 것이 좋습니다. 이러한 권한은 AWS 관리형 정책 및 역할에 포함되지 않습니다. 사용자가 직접 추가해야 합니다.

- `s3:GetBucketAcl`
- `s3:GetBucketLocation`

프로젝트에서 사용되는 S3 버킷 소유자가 변경될 경우에는 사용자가 해당 버킷을 아직도 소유하고 있는지 확인하고 아닐 경우에는 사용자의 IAM 역할에 권한을 업데이트해야 합니다. 자세한 내용은 IAM 그룹 또는 사용자에게 CodeBuild 액세스 권한 추가 및 CodeBuild 서비스 역할 생성 섹션을 참조하세요.

정책 요소 지정: 작업, 효과, 보안 주체

서비스는 각 AWS CodeBuild 리소스에 대해 API 작업 세트를 정의합니다. 이러한 API 작업에 대한 권한을 부여하려면 정책에서 지정할 수 있는 작업 세트를 CodeBuild 정의하십시오. 일부 API 작업에서는 API 작업을 수행하기 위해 복수의 작업에 대한 권한이 필요할 수 있습니다. 자세한 내용은 AWS CodeBuild 리소스 및 운영 및 AWS CodeBuild 권한 참조 섹션을 참조하세요.

다음은 기본 정책 요소입니다.

- 리소스 – Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다.
- 작업 – 작업 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어, `codebuild:CreateProject` 권한은 사용자에게 `CreateProject` 작업 수행 권한을 제공합니다.
- 효과 – 사용자가 작업을 요청하는 경우 효과(허용 또는 거부)를 지정합니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 리소스에 대한 액세스를 명시적으로 거부할 수도 있습니다. 다른 정책에서 액세스 권한을 부여하더라도 사용자가 해당 리소스에 액세스할 수 없도록 하려고 할 때 이러한 작업을 수행할 수 있습니다.
- 보안 주체 – 자격 증명 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우 사용자, 계정, 서비스 또는 권한의 수신자인 기타 엔터티를 지정합니다.

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조를 참조하십시오.

모든 CodeBuild API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 를 참조하십시오AWS CodeBuild 권한 참조.

ID 기반 정책 사용: AWS CodeBuild

이 주제에서는 자격 증명 기반 정책의 예를 통해 계정 관리자가 IAM 자격 증명(사용자, 그룹, 역할)에 권한 정책을 연결해 AWS CodeBuild 리소스에 대한 작업 수행 권한을 부여하는 방법을 보여줍니다.

> **Important**
>
> 먼저 리소스에 대한 액세스를 관리하는 데 사용할 수 있는 기본 개념과 옵션을 설명하는 소개 항목을 검토하는 것이 좋습니다. CodeBuild 자세한 정보는 리소스에 대한 액세스 권한 관리 개요 AWS CodeBuild을 참조하세요.

주제

- AWS CodeBuild 콘솔 사용에 필요한 권한
- Amazon Elastic 컨테이너 레지스트리에 연결하는 AWS CodeBuild 데 필요한 권한
- AWS CodeBuild 콘솔을 소스 공급자에 연결하는 데 필요한 권한
- AWS 에 대한 관리형 (사전 정의된) 정책 AWS CodeBuild
- CodeBuild 관리형 정책 및 알림
- CodeBuild AWS 관리형 정책 업데이트
- 고객 관리형 정책 예제

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드 프로젝트에 대해서만 정보를 가져오도록 허용하는 권한 정책의 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

AWS CodeBuild 콘솔 사용에 필요한 권한

AWS CodeBuild 콘솔을 사용하는 사용자에게는 AWS 계정의 다른 AWS 리소스를 설명할 수 있는 최소 권한 집합이 있어야 합니다. 사용자에게는 다음 서비스에 대한 권한이 있어야 합니다.

- AWS CodeBuild
- 아마존 CloudWatch
- CodeCommit (AWS CodeCommit 리포지토리에 소스 코드를 저장하는 경우)
- Amazon Elastic Container Registry(Amazon ECR)(Amazon ECR 리포지토리의 도커 이미지를 사용하는 빌드 환경을 사용하는 경우)

> **Note**
>
> 2022년 7월 26일부로 기본 IAM 정책이 업데이트되었습니다. 자세한 정보는 Amazon Elastic 컨테이너 레지스트리에 연결하는 AWS CodeBuild 데 필요한 권한을 참조하세요.

- Amazon Elastic Container Service(Amazon ECS)(Amazon ECR 리포지토리의 도커 이미지를 사용하는 빌드 환경을 사용하는 경우)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service(S3)

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔이 의도대로 작동하지 않습니다.

Amazon Elastic 컨테이너 레지스트리에 연결하는 AWS CodeBuild 데 필요한 권한

2022년 7월 26일부터 Amazon ECR 권한에 대한 기본 IAM 정책이 AWS CodeBuild 업데이트되었습니다. 다음 권한은 기본 정책에서 제거되었습니다.

```
"ecr:PutImage",
"ecr:InitiateLayerUpload",
"ecr:UploadLayerPart",
"ecr:CompleteLayerUpload"
```

2022년 7월 26일 이전에 생성된 CodeBuild 프로젝트의 경우 다음 Amazon ECR 정책으로 정책을 업데이트하는 것이 좋습니다.

```
"Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
]
```

정책 업데이트에 대한 자세한 내용은 IAM 그룹 또는 사용자에게 CodeBuild 액세스 권한 추가 섹션을 참조하세요.

AWS CodeBuild 콘솔을 소스 공급자에 연결하는 데 필요한 권한

AWS CodeBuild 콘솔은 다음 API 작업을 사용하여 소스 제공자 (예: GitHub 리포지토리) 에 연결합니다.

- `codebuild:ListConnectedOAuthAccounts`
- `codebuild:ListRepositories`
- `codebuild:PersistOAuthToken`
- `codebuild:ImportSourceCredentials`

콘솔을 사용하여 소스 제공자 (예: GitHub 리포지토리) 를 빌드 프로젝트에 연결할 수 있습니다. AWS CodeBuild 이렇게 하려면 먼저 콘솔에 액세스하는 데 사용하는 사용자와 관련된 IAM 액세스 정책에 위의 API 작업을 추가해야 합니다. AWS CodeBuild

`ListConnectedOAuthAccounts`, `ListRepositories` 및 `PersistOAuthToken` API 작업은 코드로 호출되는 것이 아닙니다. 따라서 이러한 API 작업은 AWS CLI 및 AWS SDK에 포함되지 않습니다.

AWS 에 대한 관리형 (사전 정의된) 정책 AWS CodeBuild

AWS 에서 생성하고 관리하는 독립형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 해결합니다. AWS이러한 AWS 관리형 정책은 일반적인 사용 사례에 필요한 권한을 부여하므로 필요한 권한을 조사하지 않아도 됩니다. 의 관리형 정책은 해당 정책을 부여받은 사용자의 책임에 따라 IAM, AWS CodeCommit Amazon EC2, Amazon ECR, Amazon SNS, CloudWatch Amazon Events와 같은 다른 서비스에서 작업을 수행할 수 있는 CodeBuild 권한도 제공합니다. 예를 들어 이 정책은 관리자 수준의 사용자 `AWSCodeBuildAdminAccess` 정책으로, 이 정책을 사용하는 사용자는 프로젝트 빌드에 대한 CloudWatch 이벤트 규칙과 프로젝트 관련 이벤트 (이름 접두사가 붙은 주제`arn:aws:codebuild:`) 에 대한 알림에 대한 Amazon SNS 주제를 생성 및 관리하고 프로젝트를 관리하고 에서 프로젝트를 관리하고 그룹을 보고할 수 있습니다. CodeBuild 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

계정의 사용자에게 연결할 수 있는 다음과 같은 AWS 관리형 정책은 해당 정책에만 적용됩니다. AWS CodeBuild

AWSCodeBuildAdminAccess

CodeBuild 빌드 프로젝트를 관리할 수 있는 권한을 CodeBuild 포함하여 모든 권한을 제공합니다.

AWSCodeBuildDeveloperAccess

빌드 프로젝트 관리에 대한 액세스를 CodeBuild 제공하지만 허용하지는 않습니다.

AWSCodeBuildReadOnlyAccess

에 대한 읽기 전용 액세스를 CodeBuild 제공합니다.

CodeBuild 생성하는 빌드 출력 아티팩트에 액세스하려면 라는 AWS 관리형 정책도 첨부해야 합니다. `AmazonS3ReadOnlyAccess`

CodeBuild 서비스 역할을 만들고 관리하려면 라는 AWS `IAMFullAccess` 관리형 정책도 첨부해야 합니다.

사용자 지정 IAM 정책을 생성하여 CodeBuild 작업 및 리소스에 대한 권한을 허용할 수도 있습니다. 해당 권한이 필요한 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

주제

- AWSCodeBuildAdminAccess
- AWSCodeBuildDeveloperAccess
- AWSCodeBuildReadOnlyAccess

AWSCodeBuildAdminAccess

AWSCodeBuildAdminAccess정책은 CodeBuild 빌드 프로젝트 관리 권한을 CodeBuild 포함하여 에 대한 모든 액세스 권한을 제공합니다. 관리자 수준의 사용자에게만 이 정책을 적용하여 프로젝트 및 보고서 그룹 삭제 기능을 포함하여 AWS 계정의 CodeBuild 프로젝트, 보고서 그룹 및 관련 리소스에 대한 모든 권한을 부여할 수 있습니다.

AWSCodeBuildAdminAccess 정책에는 다음 정책 설명이 포함되어 있습니다.

```
{
  "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

이 AWSCodeBuildDeveloperAccess 정책은 프로젝트 CodeBuild 및 보고서 그룹 관련 리소스의 모든 기능에 대한 액세스를 허용합니다. 이 정책은 사용자가 다른 AWS 서비스 (예: 이벤트) 의 CodeBuild 프로젝트, 보고서 그룹 또는 관련 리소스를 삭제하는 것을 허용하지 않습니다. CloudWatch 이 정책은 대부분의 사용자에게 적용하는 것이 좋습니다.

AWSCodeBuildDeveloperAccess 정책에는 다음 정책 설명이 포함되어 있습니다.

```
{
  "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

이 AWSCodeBuildReadOnlyAccess 정책은 다른 AWS 서비스의 관련 리소스 CodeBuild 및 관련 리소스에 대한 읽기 전용 액세스 권한을 부여합니다. 빌드를 보고 실행하고 프로젝트를 보고 보고서 그룹을 볼 수 있지만 변경할 수 없는 사용자에게 이 정책을 적용하십시오.

AWSCodeBuildReadOnlyAccess 정책에는 다음 정책 설명이 포함되어 있습니다.

```
{
  "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 관리형 정책 및 알림

CodeBuild 알림을 지원하여 빌드 프로젝트의 중요한 변경 사항을 사용자에게 알릴 수 있습니다. 의 관리형 CodeBuild 정책에는 알림 기능을 위한 정책 설명이 포함됩니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

전체 액세스 관리형 정책의 알림과 관련된 권한

`AWSCodeBuildFullAccess` 관리형 정책에는 알림에 대한 전체 액세스를 허용하는 다음 설명이 포함되어 있습니다. 또한 이러한 관리형 정책이 적용된 사용자는 알림에 대한 Amazon SNS 주제를 생성 및

관리하고, 주제에 대해 사용자를 구독 및 구독 취소하고, 알림 규칙의 대상으로 선택할 주제를 나열하고, Slack에 대해 구성된 AWS Chatbot 클라이언트를 나열할 수 있습니다.

```
    {
        "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": "*"
    }
```

읽기 전용 관리형 정책의 알림과 관련된 권한

AWSCodeBuildReadOnlyAccess 관리형 정책에는 알림에 대한 읽기 전용 액세스를 허용하는 다음 설명이 포함되어 있습니다. 이 관리형 정책이 적용된 사용자는 리소스에 대한 알림을 볼 수 있지만 리소스를 생성, 관리 또는 구독할 수는 없습니다.

```
  {
        "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": "*"
    }
```

다른 관리형 정책의 알림과 관련된 권한

AWSCodeBuildDeveloperAccess 관리형 정책에는 사용자가 알림을 생성, 편집 및 구독할 수 있도록 허용하는 다음 설명이 포함되어 있습니다. 사용자는 알림 규칙을 삭제하거나 리소스에 대한 태그를 관리할 수는 없습니다.

```
    {
        "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": "*"
    }
```

IAM 및 알림에 대한 자세한 내용은 AWS CodeStar 알림에 대한 Identity and Access Management를 참조하세요.

CodeBuild AWS 관리형 정책 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 CodeBuild 이후의 AWS 관리형 정책 업데이트에 대한 세부 정보를 볼 수 있습니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 AWS CodeBuild 사용자 가이드 문서 기록 에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
`AWSCodeBuildAdminAccess` ,`AWSCodeBuildDeveloperAccess` , `AWSCodeBuildReadOnlyAccess` — 기존 정책 업데이트	CodeBuild AWS CodeConnections 브랜드 변경을 지원하는 리소스를 이러한 정책에 추가했습니다. `AWSCodeBuildAdminAccess` `AWSCodeBuildDeveloperAccess` , 및 `AWSCodeBuildReadOnlyAccess` 정책이 리소스, 를 추가하도록 변경되었습니다. `arn:aws:codeconnections:*:*:connection/*`	2024년 4월 18일
`AWSCodeBuildAdminAccess` 및 `AWSCodeBuildDeveloperAccess` - 기존 정책 업데이트	CodeBuild 을 (를) 사용하는 AWS Chatbot추가 알림 유형을 지원하는 권한을 이러한 정책에 추가했습니다.	2023년 5월 16일

변경 사항	설명	날짜
	`AWSCodeBuildAdminAccess` 및 `AWSCodeBuildDeveloperAccess` 정책이 권한, `chatbot:ListMicrosoftTeamsChannelConfigurations` 를 추가하도록 변경되었습니다.	
CodeBuild 변경 내용 추적 시작	CodeBuild AWS 관리형 정책의 변경 사항 추적을 시작했습니다.	2021년 5월 16일

고객 관리형 정책 예제

이 섹션에서는 AWS CodeBuild 작업에 대한 권한을 부여하는 사용자 정책의 예를 제공합니다. 이러한 정책은 CodeBuild API, AWS SDK 또는 를 사용할 때 작동합니다. AWS CLI콘솔을 사용하는 경우 콘솔별 추가 권한을 부여해야 합니다. 자세한 내용은 AWS CodeBuild 콘솔 사용에 필요한 권한을 참조하세요.

다음 샘플 IAM 정책을 사용하여 사용자 및 역할의 CodeBuild 액세스를 제한할 수 있습니다.

주제

- 사용자가 빌드 프로젝트에 대한 정보를 가져오도록 허용
- 사용자가 보고서 그룹에 대한 정보를 가져오도록 허용
- 사용자가 보고서에 대한 정보를 가져오도록 허용
- 사용자가 빌드 프로젝트를 생성하도록 허용
- 사용자가 보고서 그룹을 생성하도록 허용
- 사용자가 보고서 그룹을 삭제하도록 허용
- 사용자가 보고서를 삭제하도록 허용
- 사용자가 빌드 프로젝트를 삭제하도록 허용
- 사용자가 빌드 프로젝트 이름 목록을 가져오도록 허용
- 사용자가 빌드 프로젝트에 대한 정보를 변경하도록 허용

- 사용자가 보고서 그룹을 변경하도록 허용
- 사용자가 빌드에 대한 정보를 가져오도록 허용
- 사용자가 빌드 프로젝트의 빌드 ID 목록을 가져오도록 허용
- 사용자가 빌드 ID 목록을 가져오도록 허용
- 사용자가 보고서 그룹 목록을 가져오도록 허용
- 사용자가 보고서 목록을 가져오도록 허용
- 사용자가 보고서 그룹에 대한 보고서 목록을 가져오도록 허용
- 사용자가 보고서에 대한 테스트 케이스 목록을 가져오도록 허용
- 사용자가 빌드 실행을 시작하도록 허용
- 사용자가 빌드 중지를 시도하도록 허용
- 사용자가 빌드 삭제를 시도하도록 허용
- 에서 관리하는 Docker 이미지에 대한 정보를 사용자가 얻을 수 있도록 허용 CodeBuild
- VPC 네트워크 인터페이스 생성에 필요한 AWS 서비스에 CodeBuild 대한 액세스 허용
- 거부 명령문을 사용하면 소스 공급자와의 연결이 끊기지 않도록 AWS CodeBuild 방지할 수 있습니다.

사용자가 빌드 프로젝트에 대한 정보를 가져오도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드 프로젝트에 대한 정보를 가져오도록 허용하는 정책 설명의 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

사용자가 보고서 그룹에 대한 정보를 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹에 대한 정보를 가져올 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReportGroups",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

사용자가 보고서에 대한 정보를 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서에 대한 정보를 가져올 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReports",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

사용자가 빌드 프로젝트를 생성하도록 허용

다음 예제 정책 설명을 사용하면 사용자가 특정 CodeBuild 서비스 역할만 사용하여 us-east-2 Region for 123456789012 account에만 어떤 이름으로든 빌드 프로젝트를 생성할 수 있습니다.

```
{
  "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"
    }
  ]
}
```

다음 예제 정책 설명을 사용하면 사용자가 지정된 CodeBuild 서비스 역할만 사용하여 us-east-2 Region for 123456789012 account에서만 어떤 이름으로든 빌드 프로젝트를 만들 수 있습니다. 또한 사용자가 지정된 서비스 역할만 다른 서비스에는 사용할 수 AWS CodeBuild 없고 다른 AWS 서비스에는 사용할 수 없도록 합니다.

```
{
  "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"}
      }
    }
  ]
}}
```

사용자가 보고서 그룹을 생성하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹을 생성할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
```

```
    }
  ]
}
```

사용자가 보고서 그룹을 삭제하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹을 삭제할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

사용자가 보고서를 삭제하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서를 삭제할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReport",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

사용자가 빌드 프로젝트를 삭제하도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드 프로젝트를 삭제하도록 허용하는 정책 설명의 예입니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

사용자가 빌드 프로젝트 이름 목록을 가져오도록 허용

다음은 사용자가 동일한 계정의 빌드 프로젝트 이름 목록을 가져오도록 허용하는 정책 설명의 예입니다.

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

사용자가 빌드 프로젝트에 대한 정보를 변경하도록 허용

다음은 사용자에게 모든 이름의 빌드 프로젝트에 대한 정보를 변경하도록 허용하지만, 123456789012 계정의 us-east-2 리전에만 있어야 하며, 지정된 AWS CodeBuild 서비스 역할만 사용해야 하는 정책 설명의 예입니다.

```
{
  "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"
    }
  ]
}
```

사용자가 보고서 그룹을 변경하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹을 변경할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

사용자가 빌드에 대한 정보를 가져오도록 허용

다음은 us-east-2 및 123456789012라는 이름의 빌드 프로젝트에 대해 사용자가 my-build-project 계정의 my-other-build-project 리전에 있는 빌드에 대한 정보를 가져오도록 허용하는 정책 설명의 예입니다.

```
{
  "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"
      ]
    }
  ]
}
```

사용자가 빌드 프로젝트의 빌드 ID 목록을 가져오도록 허용

다음은 us-east-2 및 123456789012라는 이름의 빌드 프로젝트에 대해 사용자가 my-build-project 계정의 my-other-build-project 리전에 있는 빌드 ID 목록을 가져오도록 허용하는 정책 설명의 예입니다.

```
{
  "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"
      ]
    }
  ]
}
```

사용자가 빌드 ID 목록을 가져오도록 허용

다음은 사용자가 동일한 계정의 모든 빌드 ID 목록을 가져오도록 허용하는 정책 설명의 예입니다.

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

사용자가 보고서 그룹 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹 목록을 가져올 수 있습니다.

```
{
```

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

사용자가 보고서 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 목록을 가져올 수 있습니다.

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

사용자가 보고서 그룹에 대한 보고서 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹에 대한 보고서 목록을 가져올 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReportsForReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

사용자가 보고서에 대한 테스트 케이스 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서에 대한 테스트 케이스 목록을 가져올 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DescribeTestCases",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

사용자가 빌드 실행을 시작하도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드를 실행하도록 허용하는 정책 설명의 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

사용자가 빌드 중지를 시도하도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드만 실행 중지를 시도하도록 허용하는 정책 설명의 예입니다.

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

```
      "Action": "codebuild:StopBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

사용자가 빌드 삭제를 시도하도록 허용

다음은 이름이 my로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 us-east-2 리전에 있는 빌드만 삭제하려고 시도하는 것을 허용하는 정책 설명의 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchDeleteBuilds",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

에서 관리하는 Docker 이미지에 대한 정보를 사용자가 얻을 수 있도록 허용 CodeBuild

다음 예제 정책 설명을 통해 사용자는 에서 관리하는 모든 Docker 이미지에 대한 정보를 얻을 수 있습니다. CodeBuild

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

VPC 네트워크 인터페이스 생성에 필요한 AWS 서비스에 CodeBuild 대한 액세스 허용

다음 예제 정책 설명은 두 개의 서브넷이 있는 VPC에서 네트워크 인터페이스를 생성할 수 있는 AWS CodeBuild 권한을 부여합니다.

```
{
  "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"
          ]
        }
      }
    }
  ]
}
```

거부 명령문을 사용하면 소스 공급자와의 연결이 끊기지 않도록 AWS CodeBuild 방지할 수 있습니다.

다음 예제 정책 명령문은 거부문을 사용하여 AWS CodeBuild 가 소스 공급자 연결을 해제하지 않도록 합니다. `codebuild:PersistOAuthToken` 및 `codebuild:ImportSourceCredentials`의

역인 codebuild:DeleteOAuthToken을 사용하여 소스 공급자와 연결합니다. 자세한 정보는 AWS CodeBuild 콘솔을 소스 공급자에 연결하는 데 필요한 권한을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "codebuild:DeleteOAuthToken",
      "Resource": "*"
    }
  ]
}
```

AWS CodeBuild 권한 참조

AWS CodeBuild 정책에서 AWS-wide 조건 키를 사용하여 조건을 표현할 수 있습니다. 목록은 IAM 사용 설명서에서 사용 가능한 키를 참조하세요.

정책의 Action 필드에 작업을 지정합니다. 작업을 지정하려면 codebuild: 접두사 다음에 API 작업 이름을 사용합니다(예: codebuild:CreateProject 및 codebuild:StartBuild). 문장 하나에 여러 작업을 지정하려면 쉼표로 구분합니다(예: "Action": ["codebuild:CreateProject", "codebuild:StartBuild"]).

와일드카드 문자 사용

정책의 Resource 필드에 리소스 값으로 와일드카드 문자(*)를 사용하거나 사용하지 않고 ARN을 지정합니다. 와일드카드를 사용하여 여러 작업 또는 리소스를 지정할 수 있습니다. 예를 들어, codebuild:* 는 모든 CodeBuild 작업을 지정하고 해당 단어로 Batch 시작하는 모든 CodeBuild 작업을 codebuild:Batch* 지정합니다. 다음 정책은 이름이 my로 시작되는 모든 빌드 프로젝트에 대한 액세스 권한을 부여합니다.

```
arn:aws:codebuild:us-east-2:123456789012:project/my*
```

CodeBuild API 작업 및 작업에 필요한 권한

BatchDeleteBuilds

작업: codebuild:BatchDeleteBuilds

빌드를 삭제하는 데 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetBuilds

작업: codebuild:BatchGetBuilds

빌드에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetProjects

작업: codebuild:BatchGetProjects

빌드 프로젝트에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetReportGroups

작업: codebuild:BatchGetReportGroups

보고서 그룹에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

BatchGetReports

작업: codebuild:BatchGetReports

보고서에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

BatchPutTestCases[1]

작업: codebuild:BatchPutTestCases

테스트 보고서를 생성하거나 업데이트하는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateProject

작업: `codebuild:CreateProject`, `iam:PassRole`

빌드 프로젝트를 생성하는 권한이 필요합니다.

리소스:

- `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*
- `arn:aws:iam::`*`account-ID`*`:role/`*`role-name`*

CreateReport[1]

작업: `codebuild:CreateReport`

테스트 보고서를 생성하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

CreateReportGroup

작업: `codebuild:CreateReportGroup`

보고서 그룹을 만드는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

CreateWebhook

작업: `codebuild:CreateWebhook`

Webhook를 생성하는 데 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

DeleteProject

작업: `codebuild:DeleteProject`

CodeBuild 프로젝트를 삭제하는 데 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

DeleteReport

작업: `codebuild:DeleteReport`

규칙을 삭제하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

DeleteReportGroup

작업: `codebuild:DeleteReportGroup`

보고서 그룹을 삭제하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

DeleteSourceCredentials

작업: `codebuild:DeleteSourceCredentials`

GitHub 엔터프라이즈 서버 또는 Bitbucket 리포지토리의 자격 증명에 대한 정보가 들어 있는 `SourceCredentialsInfo` 개체 세트를 삭제하는 데 필요합니다. GitHub

리소스: *

DeleteWebhook

작업: `codebuild:DeleteWebhook`

Webhook를 생성하는 데 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

DescribeTestCases

작업: `codebuild:DescribeTestCases`

테스트 케이스의 페이지 매김 목록을 반환하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

ImportSourceCredentials

작업: `codebuild:ImportSourceCredentials`

GitHub 엔터프라이즈 서버 또는 Bitbucket 리포지토리의 자격 증명에 대한 정보가 포함된 `SourceCredentialsInfo` 객체 세트를 가져오는 데 필요합니다. GitHub

리소스: *

InvalidateProjectCache

작업: `codebuild:InvalidateProjectCache`

프로젝트용 캐시를 재설정하는 데 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

ListBuildBatches

작업: `codebuild:ListBuildBatches`

빌드 배치 ID 목록을 가져오는 권한이 필요합니다.

리소스: *

ListBuildBatchesForProject

작업: `codebuild:ListBuildBatchesForProject`

특정 프로젝트의 빌드 배치 ID 목록을 가져오는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

ListBuilds

작업: `codebuild:ListBuilds`

빌드 ID 목록을 가져오는 권한이 필요합니다.

리소스: *

ListBuildsForProject

작업: `codebuild:ListBuildsForProject`

빌드 프로젝트의 빌드 ID 목록을 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

ListCuratedEnvironmentImages

작업: codebuild:ListCuratedEnvironmentImages

AWS CodeBuild가 관리하는 모든 도커 이미지에 대한 정보를 가져오는 권한이 필요합니다.

리소스: *(필요, 단, 주소 지정 가능한 AWS 리소스는 제외)

ListProjects

작업: codebuild:ListProjects

빌드 프로젝트 이름 목록을 가져오는 권한이 필요합니다.

리소스: *

ListReportGroups

작업: codebuild:ListReportGroups

보고서 그룹 목록을 가져오는 권한이 필요합니다.

리소스: *

ListReports

작업: codebuild:ListReports

보고서 목록을 가져오는 권한이 필요합니다.

리소스: *

ListReportsForReportGroup

작업: codebuild:ListReportsForReportGroup

보고서 그룹에 대한 보고서 목록을 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

RetryBuild

작업: codebuild:RetryBuild

빌드를 재시도하는 데 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

StartBuild

작업: codebuild:StartBuild

빌드 실행을 시작하는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

StopBuild

작업: codebuild:StopBuild

빌드 실행 중지를 시도할 수 있는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

UpdateProject

작업: codebuild:UpdateProject, iam:PassRole

빌드에 대한 정보를 변경하는 권한이 필요합니다.

리소스:

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

UpdateProjectVisibility

작업: codebuild:UpdateProjectVisibility, iam:PassRole

프로젝트 빌드의 퍼블릭 가시성을 변경하는 데 필요합니다.

리소스:

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

UpdateReport[1]

작업: codebuild:UpdateReport

테스트 보고서를 생성하거나 업데이트하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

UpdateReportGroup

작업: `codebuild:UpdateReportGroup`

보고서 그룹을 업데이트하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

UpdateWebhook

작업: `codebuild:UpdateWebhook`

Webhook를 업데이트하는 데 필요합니다.

리소스: `arn:aws:codebuild:region-ID:account-ID:project/project-name`

[1] 권한에만 사용됩니다. 이 작업에 대한 API가 없습니다.

태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 통제

IAM 정책 명령문의 조건은 CodeBuild 프로젝트 기반 작업에 대한 권한을 지정하는 데 사용할 수 있는 구문의 일부입니다. 해당 프로젝트와 연결된 태그를 기준으로 프로젝트에 대한 작업을 허용하거나 거부하는 정책을 생성한 다음, 사용자를 관리하기 위해 구성하는 IAM 그룹에 해당 정책을 적용할 수 있습니다. 콘솔을 사용하여 프로젝트에 태그를 적용하는 방법에 대한 자세한 내용은 또는 AWS CLI을 참조하십시오. AWS CodeBuild에서 빌드 프로젝트 생성 CodeBuild SDK를 사용하여 태그를 적용하는 방법에 대한 자세한 내용은 CodeBuildAPI 참조의 태그를 참조하십시오 CreateProject . 태그를 사용하여 AWS 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 리소스 태그를 사용한 리소스 액세스 제어를 참조하십시오. AWS

Example 예 1: 리소스 태그를 기반으로 CodeBuild 프로젝트 작업 제한

다음 예제에서는 키 `Environment` 및 키 값 `Production`으로 태그 지정된 프로젝트에 대한 모든 `BatchGetProjects` 작업을 거부합니다. 고객의 관리자는 권한 없는 사용자에게 관리형 사용자 정책 외에도 IAM 정책을 연결해야 합니다. `aws:ResourceTag` 조건 키는 태그를 기반으로 리소스에 대한 액세스를 제어하는 데 사용됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetProjects"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:ResourceTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Example 예 2: 요청 태그에 따른 CodeBuild 프로젝트 작업 제한

다음 정책은 요청에 키 Environment 및 키 값 Production을 사용하는 태그가 포함된 경우 CreateProject 작업에 대한 사용자 권한을 거부합니다. 또한 이 정책은 요청에 키 Environment를 사용하는 태그가 포함된 경우 UpdateProject를 허용하지 않도록 이러한 권한 없는 사용자가 aws:TagKeys 조건 키를 사용하여 프로젝트를 수정하는 것을 방지합니다. 관리자는 이러한 작업을 수행할 권한이 없는 사용자에게 관리형 사용자 정책 외에도 이 IAM 정책을 연결해야 합니다. aws:RequestTag 조건 키는 IAM 요청에서 전달할 수 있는 태그를 제어하는 데 사용됩니다.

```
{
  "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 예 3: 리소스 태그를 기반으로 보고서 그룹에 대한 작업 거부 또는 허용

CodeBuild 리소스와 관련된 AWS 태그를 기반으로 리소스 (프로젝트 및 보고서 그룹) 에 대한 작업을 허용하거나 거부하는 정책을 만든 다음, 사용자 관리를 위해 구성한 IAM 그룹에 해당 정책을 적용할 수 있습니다. *## ##, AWS ## Status ## # ## # ### ### ## CodeBuild ### #### ### # # ## ## ### (Developer) ### ## IAM ### ## ### ### # ####. Secret* 그런 다음, 태그가 지정된 보고서 그룹에 대해 작업하는 개발자가 일반 *###* 그룹에 속하지 않고, 대신에 제한 정책이 적용되지 않는 다른 IAM 그룹(SecretDevelopers)에 속하게 해야 합니다.

다음 예시에서는 Status 키와 키 값이 다음과 같은 태그가 지정된 보고서 그룹의 모든 CodeBuild 작업을 거부합니다. Secret

```
{
  "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 예 4: 리소스 태그를 AWSCodeBuildDeveloperAccess 기반으로 CodeBuild 작업을 제한하십시오.

특정 태그가 지정되지 않은 모든 보고서 그룹 및 프로젝트에 대한 CodeBuild 작업을 허용하는 정책을 만들 수 있습니다. 예를 들어 다음 정책은 특정 태그로 태그 지정된 보고서 그룹 및 프로젝트를 제외한 모든 보고서 그룹 및 프로젝트에 대해 AWSCodeBuildDeveloperAccess 권한과 동등한 권한을 허용합니다.

```
{
   "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"
```

```
            }
         }
      }
   ]
}
```

콘솔에서 리소스 보기

AWS CodeBuild 콘솔에는 로그인한 AWS 지역의 사용자 AWS 계정에 대한 리포지토리 목록을 표시할 수 있는 `ListRepositories` 권한이 필요합니다. 또한 콘솔에는 리소스의 대/소문자를 구분하지 않고 빠르게 검색할 수 있는 Go to resource(리소스로 이동) 기능이 포함되어 있습니다. 이 검색은 로그인한 AWS 지역의 사용자 AWS 계정에서 수행됩니다. 다음 서비스에 표시되는 리소스는 다음과 같습니다.

- AWS CodeBuild: 빌드 프로젝트
- AWS CodeCommit: 리포지토리
- AWS CodeDeploy: 애플리케이션
- AWS CodePipeline: 파이프라인

모든 서비스의 리소스에서 이 검색을 수행하려면 다음 권한이 있어야 합니다.

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

해당 서비스에 대한 권한이 없는 경우 서비스의 리소스에 대한 결과가 반환되지 않습니다. 리소스를 볼 수 있는 권한이 있더라도 해당 리소스 보기에 대한 명시적 `Deny`가 있으면 일부 리소스가 반환되지 않습니다.

규정 준수 검증: AWS CodeBuild

제3자 감사자는 여러 규정 AWS 준수 프로그램의 AWS CodeBuild 일환으로 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램 범위 내 AWS 서비스 목록은 규정 준수 프로그램별 범위 내AWS 서비스를 참조하십시오. 일반 정보는 AWS 규정 준수 프로그램을 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 AWS Artifact있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하십시오.

사용 시 규정 준수 CodeBuild 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. 를 사용하는 데 HIPAA, PCI 또는 FedRAMP와 같은 표준을 준수해야 하는 경우 다음과 같은 도움이 되는 리소스가 제공됩니다. CodeBuild AWS

- 보안 및 규정 준수 퀵 스타트 가이드 — 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다. AWS
- HIPAA 보안 및 규정 준수를 위한 설계 백서 — 이 백서는 기업이 HIPAA 준수 애플리케이션을 개발하는 데 사용할 수 있는 방법을 설명합니다. AWS
- AWS 규정 준수 리소스 — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS Config— 이 AWS 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 를 사용하여 AWS Security Hub보안 모범 AWS CodeBuild 사례와 관련된 사용량을 모니터링하십시오. Security Hub는 보안 제어를 사용하여 리소스 구성 및 보안 표준을 평가하여 다양한 규정 준수 프레임워크를 준수할 수 있도록 지원합니다. Security Hub를 사용하여 CodeBuild 리소스를 평가하는 방법에 대한 자세한 내용은 사용 AWS Security Hub 설명서의 AWS CodeBuild 컨트롤을 참조하십시오.

의 레질리언스 AWS CodeBuild

AWS 글로벌 인프라는 AWS 지역 및 가용 영역을 중심으로 구축됩니다. AWS 지역은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며, 이러한 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 지역 및 가용 영역에 대한 자세한 내용은 글로벌 인프라를 참조하십시오AWS .

의 인프라 보안 AWS CodeBuild

관리형 서비스로서 AWS 글로벌 네트워크 보안으로 AWS CodeBuild 보호됩니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 액세스할 CodeBuild 수 있습니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

에서 소스 제공자에 액세스하세요. CodeBuild

GitHub 엔터프라이즈 서버의 경우 개인 액세스 토큰 또는 OAuth 앱을 사용하여 소스 공급자에 액세스합니다. GitHub Bitbucket의 경우 액세스 토큰, 앱 암호 또는 OAuth 앱을 사용하여 소스 공급자에 액세스합니다.

> **Note**
>
> GitLab 또한 자체 관리형 GitLab 소스 공급자는 직접 액세스하는 것이 CodeBuild 아니라 이를 통해 액세스할 수 있습니다. AWS CodeConnections

주제

- GitHub 및 GitHub 엔터프라이즈 서버 액세스 토큰
- GitHub OAuth 앱
- Bitbucket 앱 암호 또는 액세스 토큰
- 비트버킷 OAuth 앱

GitHub 및 GitHub 엔터프라이즈 서버 액세스 토큰

액세스 토큰 사전 조건

시작하기 전에 GitHub 액세스 토큰에 적절한 권한 범위를 추가해야 합니다.

의 GitHub 경우 개인용 액세스 토큰의 범위는 다음과 같아야 합니다.

- repo: 프라이빗 리포지토리의 전체 제어를 부여합니다.
- repo:status: 퍼블릭 및 프라이빗 리포지토리 커밋 상태에 대한 읽기/쓰기 권한을 부여합니다.
- admin:repo_hook: 리포지토리 후크의 전체 제어를 부여합니다. 토큰에 `repo` 범위가 있을 경우 이 범위가 필요하지 않습니다.

자세한 내용은 웹 사이트의 OAuth 앱 범위 이해를 참조하십시오. GitHub

세분화된 개인용 액세스 토큰을 사용하는 경우 사용 사례에 따라 개인용 액세스 토큰에 다음 권한이 필요할 수 있습니다.

- 내용: 읽기 전용: 개인 리포지토리에 대한 액세스 권한을 부여합니다. 개인 리포지토리를 원본으로 사용하는 경우 이 권한이 필요합니다.
- 커밋 상태: 읽기 및 쓰기: 커밋 상태를 만들 수 있는 권한을 부여합니다. 이 권한은 프로젝트에 웹훅이 설정되어 있거나 보고서 작성 상태 기능이 활성화된 경우 필요합니다.
- 웹후크: 읽기 및 쓰기: 웹훅을 관리할 권한을 부여합니다. 프로젝트에 웹후크가 설정된 경우 이 권한이 필요합니다.
- 풀 리퀘스트: 읽기 전용: 풀 리퀘스트에 액세스할 수 있는 권한을 부여합니다. 웹후크에 풀 리퀘스트 이벤트에 대한 `FILE_PATH` 필터가 있는 경우 이 권한이 필요합니다.
- 관리: 읽기 및 쓰기: 이 권한은 자체 호스팅 GitHub 액션 러너 기능을 에서 사용하는 경우 필요합니다. CodeBuild 자세한 내용은 리포지토리의 등록 토큰 만들기 및 을 참조하십시오. 에서 자체 호스팅 GitHub 액션 러너 설정 AWS CodeBuild

> **Note**
>
> 조직 리포지토리에 액세스하려면 조직을 액세스 토큰의 리소스 소유자로 지정해야 합니다.

자세한 내용은 웹 사이트의 세분화된 개인용 액세스 토큰에 필요한 권한을 참조하십시오. GitHub

액세스 GitHub 토큰으로 연결 (콘솔)

콘솔을 GitHub 사용하여 프로젝트를 액세스 토큰 사용에 연결하려면 프로젝트를 만들 때 다음을 수행하십시오. 자세한 내용은 빌드 프로젝트 만들기(콘솔)을 참조하세요.

1. 소스 제공자의 경우 선택합니다 GitHub.
2. 리포지토리의 경우 GitHub 개인용 액세스 토큰으로 연결을 선택합니다.

3. GitHub 개인용 액세스 토큰에 GitHub 개인용 액세스 토큰을 입력합니다.
4. 토큰 저장을 선택합니다.

액세스 토큰 (CLI) GitHub 으로 연결

다음 단계에 따라 를 GitHub 사용하여 프로젝트를 액세스 토큰 사용에 연결합니다. AWS CLI with 사용에 대한 자세한 내용은 AWS CodeBuild를 참조하십시오명령줄 참조. AWS CLI

1. import-source-credentials 명령 실행:

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON 형식 데이터가 출력에 표시됩니다. 설치된 로컬 컴퓨터 또는 인스턴스의 위치에 있는 파일 (예:*import-source-credentials.json*) 에 데이터를 복사합니다. AWS CLI 복사된 데이터를 다음과 같이 수정하고 결과를 저장합니다.

```
{
  "serverType": "server-type",
  "authType": "auth-type",
  "shouldOverwrite": "should-overwrite",
  "token": "token",
  "username": "username"
}
```

다음을 바꿉니다.

- *server-type*: 필수 값. 이 자격 증명에 사용되는 소스 공급자. 유효한 값은 GITHUB 또는 GITHUB_ENTERPRISE입니다.
- *auth-type*: 필수 값. GitHub 또는 GitHub Enterprise Server 리포지토리에 연결하는 데 사용되는 인증 유형입니다. 유효한 값은 PERSONAL_ACCESS_TOKEN, BASIC_AUTH입니다. CodeBuild API를 사용하여 OAUTH 연결을 생성할 수는 없습니다. 대신 CodeBuild 콘솔을 사용해야 합니다.
- *should-overwrite*: 선택적 값입니다. 리포지토리 소스 자격 증명을 덮어쓰지 않도록 하려면 `false`로 설정합니다. 리포지토리 소스 자격 증명을 덮어쓰려면 `true`로 설정합니다. 기본 값은 `true`입니다.
- *token*: 필수 값. GitHub 엔터프라이즈 서버의 경우 이 토큰은 개인용 액세스 토큰입니다. GitHub

- *username*: 선택 사항 값. GitHub 및 GitHub Enterprise Server 소스 공급자에 대해서는 이 매개 변수가 무시됩니다.

2. 액세스 토큰으로 계정에 연결하려면 1단계에서 저장한 `import-source-credentials.json` 파일이 있는 디렉터리로 이동한 후 import-source-credentials 명령을 다시 실행합니다.

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 형식 데이터와 Amazon 리소스 이름(ARN)이 출력에 표시됩니다.

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

> **Note**
>
> import-source-credentials 명령을 동일한 서버 유형과 인증 유형으로 다시 실행하면 저장된 액세스 토큰이 업데이트됩니다.

계정이 액세스 토큰으로 연결되면 CodeBuild 프로젝트를 만드는 `create-project` 데 사용할 수 있습니다. 자세한 정보는 빌드 프로젝트 생성(AWS CLI)을 참조하세요.

3. 연결된 액세스 토큰을 보려면 list-source-credentials 명령을 실행합니다.

```
aws codebuild list-source-credentials
```

JSON 형식의 `sourceCredentialsInfos` 객체가 출력에 표시됩니다.

```
{
    "sourceCredentialsInfos": [
        {
            "authType": "auth-type",
            "serverType": "server-type",
            "arn": "arn"
        }
    ]
}
```

sourceCredentialsObject에는 연결된 소스 자격 증명 정보 목록이 포함되어 있습니다.

- authType은 자격 증명에서 사용하는 인증 유형입니다. OAUTH, BASIC_AUTH 또는 PERSONAL_ACCESS_TOKEN 유형을 지정할 수 있습니다.
- serverType은 소스 공급자의 유형입니다. GITHUB, GITHUB_ENTERPRISE 또는 BITBUCKET 유형을 지정할 수 있습니다.
- arn은 토큰의 ARN입니다.

4. 소스 공급자와의 연결을 해제하고 액세스 토큰을 삭제하려면 해당 ARN을 지정하여 delete-source-credentials 명령을 실행합니다.

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON 형식의 데이터가 반환되고 삭제된 자격 증명의 ARN이 표시됩니다.

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

GitHub OAuth 앱

GitHub OAuth를 사용하여 연결 (콘솔)

콘솔을 GitHub 사용하여 프로젝트를 OAuth 앱 사용에 연결하려면 프로젝트를 생성할 때 다음을 수행하세요. 자세한 내용은 빌드 프로젝트 만들기(콘솔)을 참조하세요.

1. 소스 제공자의 경우 선택하세요. GitHub
2. 리포지토리의 경우 OAuth를 사용한 연결을 선택합니다.
3. Connect to GitHub (연결 대상) 를 선택하고 로그인한 다음 계정을 승인합니다.
4. 확인을 CodeBuild 선택하여 GitHub 계정에 연결합니다.
5. GitHub 리포지토리에 GitHub 리포지토리 링크를 입력합니다.

승인된 OAuth 앱을 검토하려면 애플리케이션 온으로 이동하여 GitHub aws-codesuite가 AWS CodeBuild (*region*) 소유한 애플리케이션이 목록에 있는지 확인하십시오.

Bitbucket 앱 암호 또는 액세스 토큰

필수 조건

시작하기 전에 Bitbucket 앱 암호 또는 액세스 토큰에 적절한 권한 범위를 추가해야 합니다.

Bitbucket의 경우 앱 암호 또는 액세스 토큰의 범위가 다음과 같아야 합니다.

- repository:read: 권한 있는 사용자가 액세스할 수 있는 모든 리포지토리에 대한 읽기 액세스 권한을 부여합니다.
- pullrequest:read: pull 요청에 대한 읽기 액세스를 부여합니다. 프로젝트에 Bitbucket 웹후크가 있는 경우 앱 암호 또는 액세스 토큰에 이 범위가 있어야 합니다.
- webhook: Webhook에 대한 액세스를 부여합니다. 프로젝트에 웹후크 작업이 있는 경우 앱 암호 또는 액세스 토큰에 이 범위가 있어야 합니다.

자세한 내용은 Bitbucket 웹사이트의 Bitbucket 클라우드 REST API에 대한 범위 및 Bitbucket 클라우드의 OAuth를 참조하십시오.

앱 암호로 Bitbucket에 연결(콘솔)

콘솔에서 앱 암호를 사용하여 프로젝트를 Bitbucket에 연결하려면 프로젝트를 생성할 때 다음과 같이 합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔)을 참조하세요.

1. 소스 공급자에서 Bitbucket을 선택합니다.

   > **Note**
   >
   > CodeBuild Bitbucket 서버는 지원하지 않습니다.

2. 리포지토리에서 Bitbucket 앱 암호로 연결을 선택합니다.
3. Bitbucket 사용자 이름에 Bitbucket 사용자 이름을 입력합니다.
4. Bitbucket 앱 암호에 Bitbucket 앱 암호를 입력합니다.
5. Bitbucket 자격증 저장을 선택합니다.

액세스 토큰을 사용하여 Bitbucket을 연결 (콘솔)

콘솔을 사용하여 액세스 토큰을 사용하여 프로젝트를 Bitbucket에 연결하려면 프로젝트를 만들 때 다음과 같이 하십시오. 자세한 내용은 빌드 프로젝트 만들기(콘솔)을 참조하세요.

1. 소스 공급자에서 Bitbucket을 선택합니다.

> **Note**
>
> CodeBuild 비트버킷 서버를 지원하지 않습니다.

2. 리포지토리의 경우 Bitbucket 액세스 토큰을 사용한 연결을 선택합니다.
3. 비트버킷 액세스 토큰에 비트버킷 액세스 토큰을 입력합니다.
4. 토큰 저장을 선택합니다.

앱 암호 또는 액세스 토큰 (CLI) 을 사용하여 Bitbucket을 연결합니다.

다음 단계에 따라 를 사용하여 앱 암호 또는 액세스 토큰을 사용하여 프로젝트를 Bitbucket에 연결합니다. AWS CLI with 사용에 대한 자세한 내용은 AWS CodeBuild를 참조하십시오. AWS CLI 명령줄 참조

1. import-source-credentials 명령 실행:

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON 형식 데이터가 출력에 표시됩니다. 설치된 로컬 컴퓨터 또는 인스턴스의 위치에 있는 파일 (예:*import-source-credentials.json*) 에 데이터를 복사합니다. AWS CLI 복사된 데이터를 다음과 같이 수정하고 결과를 저장합니다.

```
{
  "serverType": "BITBUCKET",
  "authType": "auth-type",
  "shouldOverwrite": "should-overwrite",
  "token": "token",
  "username": "username"
}
```

다음을 바꿉니다.

- *auth-type*: 필수 값. Bitbucket 리포지토리에 연결하는 데 사용되는 인증 유형입니다. 유효한 값은 PERSONAL_ACCESS_TOKEN, BASIC_AUTH입니다. CodeBuild API를 사용하여 OAUTH 연결을 생성할 수는 없습니다. 대신 CodeBuild 콘솔을 사용해야 합니다.

- *should-overwrite*: 선택적 값입니다. 리포지토리 소스 자격 증명을 덮어쓰지 않도록 하려면 false로 설정합니다. 리포지토리 소스 자격 증명을 덮어쓰려면 true로 설정합니다. 기본 값은 true입니다.
- *token*: 필수 값. Bitbucket의 경우 이는 액세스 토큰 또는 앱 비밀번호입니다.
- *username*: 선택 사항 값. authType이 BASIC_AUTH일 경우 Bitbucket 사용자 이름입니다. 이 파라미터는 다른 유형의 소스 공급자 또는 연결에서는 무시됩니다.

2. 계정을 앱 암호 또는 액세스 토큰에 연결하려면 1단계에서 저장한 import-source-credentials.json 파일이 들어 있는 디렉터리로 전환하고 import-source-credentials 명령을 다시 실행합니다.

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 형식 데이터와 Amazon 리소스 이름(ARN)이 출력에 표시됩니다.

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

> **Note**
>
> import-source-credentials 명령을 동일한 서버 유형과 인증 유형으로 다시 실행하면 저장된 액세스 토큰이 업데이트됩니다.

계정이 앱 비밀번호로 연결되면 를 create-project 사용하여 CodeBuild 프로젝트를 만들 수 있습니다. 자세한 정보는 빌드 프로젝트 생성(AWS CLI)을 참조하세요.

3. 연결된 앱 암호 또는 액세스 토큰을 보려면 list-source-credentials 명령어를 실행합니다.

```
aws codebuild list-source-credentials
```

JSON 형식의 sourceCredentialsInfos 객체가 출력에 표시됩니다.

```
{
    "sourceCredentialsInfos": [
        {
            "authType": "auth-type",
```

```
            "serverType": "BITBUCKET",
            "arn": "arn"
        }
    ]
}
```

sourceCredentialsObject에는 연결된 소스 자격 증명 정보 목록이 포함되어 있습니다.

- authType은 자격 증명에서 사용하는 인증 유형입니다. OAUTH, BASIC_AUTH 또는 PERSONAL_ACCESS_TOKEN 유형을 지정할 수 있습니다.
- arn은 토큰의 ARN입니다.

4. 소스 제공자와의 연결을 끊고 해당 앱 비밀번호 또는 액세스 토큰을 제거하려면 해당 ARN과 함께 delete-source-credentials 명령을 실행합니다.

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON 형식의 데이터가 반환되고 삭제된 자격 증명의 ARN이 표시됩니다.

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

비트버킷 OAuth 앱

OAuth를 사용하여 비트버킷 연결 (콘솔)

콘솔을 사용하여 OAuth 앱을 사용하여 프로젝트를 Bitbucket에 연결하려면 프로젝트를 만들 때 다음과 같이 하십시오. 자세한 내용은 빌드 프로젝트 만들기(콘솔)을 참조하세요.

1. 소스 공급자에서 Bitbucket을 선택합니다.
2. 리포지토리의 경우 OAuth를 사용한 연결을 선택합니다.
3. Bitbucket에 연결을 선택하고 로그인한 다음 계정을 승인합니다.
4. 확인을 선택하여 Bitbucket CodeBuild 계정에 연결합니다.
5. 비트버킷 리포지토리에 비트버킷 리포지토리 링크를 입력합니다.

승인된 OAuth 앱을 검토하려면 Bitbucket의 애플리케이션 인증으로 이동하여 이름이 지정된 애플리케이션이 나열되어 있는지 확인하십시오. AWS CodeBuild (*region*)

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 크로스 서비스 사칭으로 인해 AWS대리인 문제가 발생할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에 aws:SourceArn및 aws:SourceAccount글로벌 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스에 AWS CodeBuild 부여하는 권한을 제한하는 것이 좋습니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 aws:SourceArn을 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount을 사용하세요.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자(*)를 포함한 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용합니다. 예를 들어 arn:aws:codebuild:*:*123456789012*:*입니다.

만약 aws:SourceArn 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 글로벌 조건 컨텍스트 키를 모두 사용해야 합니다.

의 값은 CodeBuild 프로젝트 aws:SourceArn ARN이어야 합니다.

다음 예제는 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하여 혼동되는 대리자 문제를 방지하는 CodeBuild 방법을 보여줍니다.

```
{
    "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"
                }
            }
        }
    ]
}
```

고급 주제

이 단원에서는 더욱 숙련된 AWS CodeBuild 사용자에게 유용한 몇 가지 고급 주제를 다룹니다.

주제

고급 설정

콘솔을 사용하여 시작하기의 단계를 따라 처음으로 AWS CodeBuild에 액세스하는 경우 이 주제의 정보가 거의 필요 없습니다. 그러나, 계속하여 CodeBuild를 사용하게 되면 조직의 IAM 그룹 및 사용자에게 CodeBuild에 대한 액세스 권한 부여, CodeBuild에 액세스할 수 있도록 IAM 또는 AWS KMS keys의 기존 서비스 역할 수정, CodeBuild에 액세스할 수 있도록 조직 워크스테이션 전반의 AWS CLI 설정 등과 같은 작업이 필요할 수 있습니다. 이 주제에서는 관련 설치 단계를 완료하는 방법을 설명합니다.

이미 AWS 계정을 가지고 있다고 가정합니다. 단, 아직 계정이 없으면 http://aws.amazon.com으로 이동하고, 콘솔에 로그인을 선택하고, 온라인 지침을 따릅니다.

주제

IAM 그룹 또는 사용자에게 CodeBuild 액세스 권한 추가

IAM 그룹 또는 사용자로 AWS CodeBuild에 액세스하려면 액세스 권한을 추가해야 합니다. 이 섹션에서는 IAM 콘솔이나 AWS CLI를 사용하여 이를 수행하는 방법을 설명합니다.

AWS 루트 계정(권장 안 함)을 사용하여 또는 AWS 계정의 관리자 사용자로 CodeBuild에 액세스하는 경우 다음 지침을 따르지 않아도 됩니다.

AWS 계정 루트 사용자 및 관리자 사용자에 대한 자세한 내용은 사용 설명서에서 AWS 계정 루트 사용자 및 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.

IAM 그룹 또는 사용자에게 CodeBuild 액세스 권한을 추가하려면(콘솔)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   다음 중 하나를 사용하여 AWS Management Console에 이미 로그인되어 있어야 합니다.

   - 사용자의 AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 AWS 계정 루트 사용자를 참조하세요.
   - AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   - 최소한 다음 작업 세트를 수행할 수 있는 권한이 있는 AWS 계정의 사용자.

     ```
     iam:AttachGroupPolicy
     iam:AttachUserPolicy
     iam:CreatePolicy
     iam:ListAttachedGroupPolicies
     iam:ListAttachedUserPolicies
     iam:ListGroups
     iam:ListPolicies
     iam:ListUsers
     ```

     자세한 내용은 사용 설명서의 IAM 정책 개요를 참조하세요.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. IAM 그룹이나 IAM 사용자에게 사용자 지정 AWS CodeBuild 액세스 권한 세트를 추가하려면 이 절차의 4단계로 이동합니다.

   IAM 그룹이나 IAM 사용자에게 기본 CodeBuild 액세스 권한 세트를 추가하려면 정책 유형, AWS 관리형을 선택한 후 다음을 수행합니다.

- CodeBuild에 대한 전체 액세스 권한을 추가하려면 AWSCodeBuildAdminAccess 확인란을 선택하고 정책 작업을 선택한 다음, 연결을 선택합니다. 대상 IAM그룹 및 사용자 옆의 확인란을 선택하고 정책 연결을 선택합니다. [AmazonS3ReadOnlyAccess] 및 [IAMFullAccess]라는 정책에 대해서도 이 절차를 반복합니다.
- 빌드 프로젝트 관리를 제외한 모든 항목에 CodeBuild에 대한 액세스 권한을 추가하려면 AWSCodeBuildDeveloperAccess 확인란을 선택하고 정책 작업을 선택한 다음, 연결을 선택합니다. 대상 IAM그룹 및 사용자 옆의 확인란을 선택하고 정책 연결을 선택합니다. [AmazonS3ReadOnlyAccess] 정책에 대해서도 이 절차를 반복합니다.
- CodeBuild에 대해 읽기 전용 액세스 권한을 추가하려면 AWSCodeBuildReadOnlyAccess 확인란을 선택합니다. 대상 IAM그룹 및 사용자 옆의 확인란을 선택하고 정책 연결을 선택합니다. [AmazonS3ReadOnlyAccess] 정책에 대해서도 이 절차를 반복합니다.

이제 IAM 그룹 또는 사용자에게 기본 CodeBuild 액세스 권한 세트가 추가되었습니다. 이 절차의 나머지 단계는 건너뜁니다.

4. 정책 생성(Create Policy)을 선택합니다.
5. [Create Policy] 페이지에서 [Create Your Own Policy] 옆의 [Select]를 선택합니다.
6. 정책 검토 페이지의 정책 이름에 정책 이름을 입력합니다(예: **CodeBuildAccessPolicy**). 다른 이름을 사용하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.
7. 정책 문서에 다음을 입력한 다음 정책 생성을 선택합니다.

```
{
  "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**
>
> 다음 정책은 모든 CodeBuild 작업 및 대다수의 AWS 리소스에 대한 액세스를 허용합니다. 특정 CodeBuild 작업으로 권한을 제한하려면 CodeBuild 정책 명령에서 `codebuild:*`의 값을 변경합니다. 자세한 내용은 자격 증명 및 액세스 관리 섹션을 참조하세요. 특정 AWS 리소스로 액세스를 제한하려면 `Resource` 객체 값을 변경하십시오. 자세한 내용은 자격 증명 및 액세스 관리 섹션을 참조하세요.

이 CodeBuildRolePolicy 문은 빌드 프로젝트를 생성하거나 수정할 수 있도록 하는 데 필요합니다.

8. 탐색 창에서 [Groups] 또는 [Users]를 선택합니다.
9. 그룹 또는 사용자 목록에서 CodeBuild 액세스 권한을 추가하려는 IAM 그룹 또는 IAM 사용자의 이름을 선택합니다.
10. 그룹의 경우 그룹 설정 페이지의 권한 탭에서 관리형 정책을 확장한 다음 정책 연결을 선택합니다.

    사용자의 경우 사용자 설정 페이지의 [Permissions] 탭에서 [Add permissions]를 선택합니다.
11. 그룹의 경우 정책 연결 페이지에서 CodeBuildAccessPolicy를 선택한 후 정책 연결을 선택합니다.

    사용자의 경우 권한 추가 페이지에서 기존 정책 직접 연결을 선택합니다. CodeBuildAccessPolicy, 다음: 검토를 차례로 선택한 후 권한 추가를 선택합니다.

CodeBuild 액세스 권한을 IAM 그룹 또는 사용자에게 추가하려면(AWS CLI)

1. 이전 절차에서 설명한 대로 IAM 엔터티 중 하나에 해당하는 AWS 액세스 키 및 AWS 비밀 액세스 키를 사용하여 AWS CLI를 구성했는지 확인합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 설정을 참조하세요.
2. IAM 그룹이나 IAM 사용자에게 사용자 지정 AWS CodeBuild 액세스 권한 세트를 추가하려면 이 절차의 3단계로 이동합니다.

   IAM 그룹 또는 IAM 사용자에게 기본 CodeBuild 액세스 권한 세트를 추가하려면 다음을 수행합니다.

   IAM 그룹 또는 사용자에게 권한을 추가할 것인지에 따라 다음 명령 중 하나를 실행합니다.

```
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
```

   group-name 또는 *user-name*을 IAM 그룹 이름 또는 사용자 이름으로 변경하고 다음 정책 Amazon 리소스 이름(ARN)마다 한 번씩 *policy-arn*을 변경하는 등 명령을 3번 실행해야 합니다.

   - CodeBuild에 전체 액세스 권한을 추가하려면 다음 정책 ARN을 사용합니다.
     - arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess

  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
  - `arn:aws:iam::aws:policy/IAMFullAccess`
- CodeBuild에 빌드 프로젝트 관리를 제외한 전체 액세스 권한을 추가하려면 다음 정책 ARN을 사용합니다.
  - `arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess`
  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
- CodeBuild에 읽기 전용 액세스 권한을 추가하려면 다음 정책 ARN을 사용합니다.
  - `arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess`
  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`

이제 IAM 그룹 또는 사용자에게 기본 CodeBuild 액세스 권한 세트가 추가되었습니다. 이 절차의 나머지 단계는 건너뜁니다.

3. AWS CLI가 설치된 로컬 워크스테이션 또는 인스턴스의 빈 디렉터리에 `put-group-policy.json` 또는 `put-user-policy.json`이라는 파일을 생성합니다. 다른 파일 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.

```
{
  "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**
>
> 다음 정책은 모든 CodeBuild 작업 및 대다수의 AWS 리소스에 대한 액세스를 허용합니다. 특정 CodeBuild 작업으로 권한을 제한하려면 CodeBuild 정책 명령에서 `codebuild:*`의 값을 변경합니다. 자세한 내용은 자격 증명 및 액세스 관리 섹션을 참조하세요. 특정 AWS 리소스로 액세스를 제한하려면 관련된 `Resource` 객체 값을 변경하십시오. 자세한 정보는 자격 증명 및 액세스 관리 또는 특정 AWS 서비스의 보안 설명서를 참조하십시오.
> 이 `CodeBuildRolePolicy` 문은 빌드 프로젝트를 생성하거나 수정할 수 있도록 하는 데 필요합니다.

4. 파일을 저장한 디렉터리로 전환한 다음, 다음 명령 중 하나를 실행합니다. `CodeBuildGroupAccessPolicy` 및 `CodeBuildUserAccessPolicy`에 다른 값을 사용할 수도 있습니다. 다른 값을 사용하는 경우 여기에서 사용해야 합니다.

   IAM 그룹의 경우:

   ```
   aws iam put-group-policy --group-name group-name --policy-name
    CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
   ```

   사용자의 경우:

   ```
   aws iam put-user-policy --user-name user-name --policy-name
    CodeBuildUserAccessPolicy --policy-document file://put-user-policy.json
   ```

   앞의 명령에서 *group-name* 또는 *user-name*을 대상 IAM 그룹 또는 사용자의 이름으로 변경합니다.

CodeBuild 서비스 역할 생성

CodeBuild가 사용자를 대신하여 종속 AWS 서비스와 상호 작용할 수 있으려면 AWS CodeBuild 서비스 역할이 있어야 합니다. CodeBuild 또는 AWS CodePipeline 콘솔을 사용하면 CodeBuild 서비스 역할을 생성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 빌드 프로젝트 만들기(콘솔)
- CodeBuild를 사용하는 파이프라인 생성(CodePipeline 콘솔)
- CodeBuild 빌드 작업을 파이프라인에 추가(CodePipeline 콘솔)
- 빌드 프로젝트 설정 변경(콘솔)

이러한 콘솔을 사용하지 않을 사용자를 위해 이 섹션에서는 IAM 콘솔 또는 AWS CLI를 사용하여 CodeBuild 서비스 역할을 생성하는 방법을 설명합니다.

> **Important**
>
> CodeBuild에서는 사용자를 대신하여 수행되는 모든 작업에 대해 서비스 역할을 사용합니다. 역할에 사용자에게 불필요한 권한이 포함되어 있는 경우 사용자의 권한을 실수로 에스컬레이션할 수 있습니다. 이 역할은 최소한의 권한을 부여해야 합니다.

이 페이지에서 설명하는 서비스 역할에는 CodeBuild를 사용하기 위해 필요한 최소한의 권한을 부여하는 정책이 포함되어 있습니다. 사용 사례에 따라 다른 권한을 추가해야 할 수 있습니다.

CodeBuild 서비스 역할을 생성하려면(콘솔)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   다음 중 하나를 사용하여 콘솔에 이미 로그인되어 있어야 합니다.

   - 사용자의 AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 AWS 계정 루트 사용자를 참조하세요.
   - AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   - 최소한 다음 작업 세트를 수행할 수 있는 권한이 있는 AWS 계정의 사용자.

     ```
     iam:AddRoleToInstanceProfile
     iam:AttachRolePolicy
     iam:CreateInstanceProfile
     iam:CreatePolicy
     iam:CreateRole
     iam:GetRole
     iam:ListAttachedRolePolicies
     iam:ListPolicies
     iam:ListRoles
     iam:PassRole
     iam:PutRolePolicy
     iam:UpdateAssumeRolePolicy
     ```

     자세한 내용은 사용 설명서의 IAM 정책 개요를 참조하세요.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 생성(Create Policy)을 선택합니다.
4. [Create Policy] 페이지에서 [JSON]을 선택합니다.
5. JSON 정책에 대해 다음을 입력한 다음 정책 검토를 선택합니다.

   ```
   {
     "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**
>
> 다음 정책에는 대다수의 AWS 리소스에 대한 액세스를 허용하는 명령문이 포함되어 있습니다. 특정 AWS 리소스로 AWS CodeBuild 액세스를 제한하려면 `Resource` 어레이 값을 변경하십시오. 자세한 정보는 AWS 서비스의 보안 설명서를 참조하십시오.

6. 정책 검토 페이지의 정책 이름에 정책의 이름(예: **`CodeBuildServiceRolePolicy`**)을 입력한 후 정책 생성을 선택합니다.

> **Note**
>
> 다른 이름을 사용하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.

7. 탐색 창에서 역할을 선택합니다.
8. Create role(역할 생성)을 선택합니다.
9. 역할 생성 페이지에서 AWS 서비스가 이미 선택되어 있으면 CodeBuild를 선택하고 다음: 권한을 선택합니다.
10. 권한 정책 연결 페이지에서 CodeBuildServiceRolePolicy를 선택한 후 다음: 검토를 선택합니다.

11. 역할 생성 및 검토 페이지의 역할 이름에 역할의 이름(예: **CodeBuildServiceRole**)을 입력한 후 역할 생성을 선택합니다.

CodeBuild 서비스 역할을 생성하려면(AWS CLI)

1. 이전 절차에서 설명한 대로 IAM 엔터티 중 하나에 해당하는 AWS 액세스 키 및 AWS 비밀 액세스 키를 사용하여 AWS CLI를 구성했는지 확인합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 설정을 참조하세요.
2. AWS CLI가 설치된 로컬 워크스테이션 또는 인스턴스의 빈 디렉터리에 `create-role.json` 및 `put-role-policy.json`이라는 두 개의 파일을 생성합니다. 다른 파일 이름을 선택하는 경우 이 절차 전체에서 해당 이름으로 바꿉니다.

   `create-role.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

> **Note**
>
> 혼동된 대리자 문제로부터 자신을 보호하기 위하여 `aws:SourceAccount` 및 `aws:SourceArn` 조건 키를 사용할 것을 권장합니다. 예를 들어 다음 조건 블록을 사용해서 이전 신뢰 정책을 편집할 수 있습니다. `aws:SourceAccount`는 CodeBuild 프로젝트의 소유자이고 `aws:SourceArn`은 CodeBuild 프로젝트 ARN입니다.

   서비스 역할을 AWS 계정으로 제한하려는 경우 `create-role.json`이 다음과 비슷할 수 있습니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "account-ID"
                    ]
                }
            }
        }
    ]
}
```

서비스 역할을 특정 CodeBuild 프로젝트로 제한하려는 경우 `create-role.json`이 다음과 비슷할 수 있습니다.

```
{
    "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**
>
> CodeBuild 프로젝트의 이름을 모르거나 아직 결정하지 않은 상태에서 특정 ARN 패턴에 대한 신뢰 정책 제한을 적용하려면 ARN의 해당 부분을 와일드카드(*)로 바꿀 수 있습니다. 프로젝트를 생성한 후 신뢰 정책을 업데이트할 수 있습니다.

`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**
>
> 다음 정책에는 대다수의 AWS 리소스에 대한 액세스를 허용하는 명령문이 포함되어 있습니다. 특정 AWS 리소스로 AWS CodeBuild 액세스를 제한하려면 `Resource` 어레이 값을 변경하십시오. 자세한 정보는 AWS 서비스의 보안 설명서를 참조하십시오.

3. 앞의 파일을 저장한 디렉터리로 전환한 다음, 다음 두 가지 명령을 다음 순서로 한 번에 하나씩 실행합니다. `CodeBuildServiceRole`과 `CodeBuildServiceRolePolicy`에 다른 값을 사용하려면 여기에서 사용해야 합니다.

```
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
```

CodeBuild용 고객 관리형 키를 생성하고 구성합니다.

AWS CodeBuild에서 빌드 출력 아티팩트를 암호화하려면 KMS 키에 액세스할 수 있어야 합니다. 기본적으로 CodeBuild는 AWS 계정의 Amazon S3에 AWS 관리형 키를 사용합니다.

AWS 관리형 키를 사용하지 않으려면 고객 관리형 키를 사용자가 직접 생성 및 구성해야 합니다. 이 섹션에서는 IAM 콘솔을 사용하여 이를 수행하는 방법을 설명합니다.

고객 관리형 키에 대한 자세한 내용은 AWS KMS 개발자 가이드의 AWS Key Management Service 개념 및 키 생성을 참조하세요.

CodeBuild에서 사용할 고객 관리형 키를 구성하려면 AWS KMS 개발자 안내서의 키 정책 수정의 "키 정책 수정 방법" 섹션에 있는 지침을 따르세요. 그런 다음 키 정책에 다음 명령문(*### BEGIN ADDING STATEMENTS HERE ###*과 *### END ADDING STATEMENTS HERE ###* 사이)을 추가합니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 키 정책에 입력하지 않아야 합니다.

```
{
  "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",
      ...
    }
  ]
}
```

- *region-ID*는 CodeBuild가 연결된 Amazon S3 버킷이 있는 AWS 리전의 ID를 나타냅니다(예: `us-east-1`).
- *account-ID*는 고객 관리형 키를 소유하는 AWS 계정의 ID를 나타냅니다.
- *CodeBuild-service-role*은 이 주제의 앞부분에서 생성하거나 식별한 CodeBuild 서비스 역할 이름을 나타냅니다.

> **Note**
>
> IAM 콘솔을 통해 고객 관리형 키를 생성 또는 구성하려면 먼저 다음 중 하나를 사용하여 AWS Management Console에 로그인해야 합니다.

- 사용자의 AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.
- AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
- 고객 관리형 키를 생성 또는 수정할 수 있는 권한이 있는 AWS 계정의 사용자. 자세한 내용은 AWS KMS 개발자 안내서의 AWS KMS 콘솔 사용에 필요한 권한을 참조하세요.

AWS CLI 설치 및 구성

AWS CodeBuild에 액세스하기 위해 CodeBuild 콘솔, CodePipeline 콘솔 또는 AWS SDK와 함께(이러한 기능 대신) AWS CLI를 사용할 수 있습니다. AWS CLI를 아직 설치 및 구성하지 않은 경우 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 설정을 참조하세요.

1. 다음 명령을 실행하여 CodeBuild에 AWS CLI를 설치할 수 있는지 여부를 확인합니다.

   ```
   aws codebuild list-builds
   ```

   이 명령이 제대로 실행되면 다음과 비슷한 정보가 출력에 표시됩니다.

   ```
   {
     "ids": []
   }
   ```

   빈 대괄호는 실행한 빌드가 아직 없다는 의미입니다.
2. 오류가 출력되면 현재 AWS CLI 버전을 제거한 다음 최신 버전을 설치해야 합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 제거 및 AWS Command Line Interface 설치를 참조하세요.

AWS CodeBuild용 명령줄 참조

AWS CLI는 AWS CodeBuild 자동화를 위한 명령을 제공합니다. 이 주제의 정보를 AWS Command Line Interface 사용 설명서 및 AWS CodeBuild에 대한 AWS CLI 참조의 추가 자료로 사용하세요.

찾는 항목이 보이지 않습니까? AWS SDK를 사용하여 CodeBuild를 호출하려면 AWS SDK 및 도구 참조 섹션을 참조하세요.

이 주제의 정보를 사용하려면 AWS CLI를 설치하고 AWS CLI 설치 및 구성에 설명된 대로 CodeBuild와 함께 사용할 수 있도록 구성했어야 합니다.

AWS CLI를 사용하여 CodeBuild의 엔드포인트를 지정하려면 AWS CodeBuild 엔드포인트 지정(AWS CLI) 섹션을 참조하세요.

이 명령을 실행하여 CodeBuild 명령 목록을 가져옵니다.

```
aws codebuild help
```

이 명령을 실행하여 CodeBuild 명령에 대한 정보를 가져옵니다. 여기서 *command-name*은 명령의 이름입니다.

```
aws codebuild command-name help
```

CodeBuild 명령에는 다음이 포함됩니다.

- `batch-delete-builds`: CodeBuild에서 하나 이상의 빌드를 삭제합니다. 자세한 내용은 빌드 삭제(AWS CLI) 섹션을 참조하세요.
- `batch-get-builds`: CodeBuild의 여러 빌드에 대한 정보를 가져옵니다. 자세한 내용은 빌드 세부 정보 보기(AWS CLI) 섹션을 참조하세요.
- `batch-get-projects`: 하나 이상의 지정된 빌드 프로젝트에 대한 정보를 가져옵니다. 자세한 내용은 빌드 프로젝트 세부 정보 보기(AWS CLI) 섹션을 참조하세요.
- `create-project`: 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 섹션을 참조하세요.
- `delete-project`: 빌드 프로젝트를 삭제합니다. 자세한 내용은 빌드 프로젝트 삭제(AWS CLI) 섹션을 참조하세요.
- `list-builds`: CodeBuild의 빌드를 위한 Amazon 리소스 이름(ARN)을 표시합니다. 자세한 내용은 빌드 ID 목록 보기(AWS CLI) 섹션을 참조하세요.
- `list-builds-for-project`: 지정된 빌드 프로젝트에 연결된 빌드 ID 목록을 가져옵니다. 자세한 내용은 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI) 섹션을 참조하세요.
- `list-curated-environment-images`: 빌드에 사용할 수 있는 CodeBuild 관리형 도커 이미지 목록을 가져옵니다. 자세한 내용은 Docker 이미지 제공: CodeBuild 섹션을 참조하세요.
- `list-projects`: 빌드 프로젝트 이름 목록을 가져옵니다. 자세한 내용은 빌드 프로젝트 이름 목록 보기(AWS CLI) 섹션을 참조하세요.

- `start-build`: 빌드 실행을 시작합니다. 자세한 내용은 빌드 실행(AWS CLI) 섹션을 참조하세요.
- `stop-build`: 지정된 빌드의 실행을 중지하려고 시도합니다. 자세한 내용은 빌드 중지(AWS CLI) 섹션을 참조하세요.
- `update-project`: 지정된 빌드 프로젝트에 대한 정보를 변경합니다. 자세한 내용은 빌드 프로젝트 설정 변경(AWS CLI) 섹션을 참조하세요.

AWS SDK 및 AWS CodeBuild용 도구 참조

하나의 AWS SDK 또는 도구를 사용하여 AWS CodeBuild를 자동화하려면 다음 리소스를 참조하십시오.

AWS CLI를 사용하여 CodeBuild를 실행하려면 명령줄 참조 섹션을 참조하세요.

지원되는 AWS SDK 및 AWS CodeBuild용 도구

다음 AWS SDK 및 도구는 CodeBuild를 지원합니다.

- C++용 AWS SDK. 자세한 내용은 C++용 AWS SDK API 참조의 Aws::CodeBuild 네임스페이스 섹션을 참조하세요.
- Go용 AWS SDK. 자세한 내용은 Go용 AWS SDK API 참조의 codebuild 섹션을 참조하세요.
- Java용 AWS SDK. 자세한 내용은 Java용 AWS SDK API 참조의 `com.amazonaws.services.codebuild` 및 `com.amazonaws.services.codebuild.model` 섹션을 참조하세요.
- 브라우저에서 JavaScript용 AWS SDK 사용 및 Node.js에서 JavaScript용 AWS SDK 사용. 자세한 내용은 JavaScript용 AWS SDK API 참조의 클래스: AWS.CodeBuild 섹션을 참조하세요.
- .NET용 AWS SDK. 자세한 내용은 .NET용 AWS SDK API 참조의 Amazon.CodeBuild 및 Amazon.CodeBuild.Model 네임스페이스 섹션을 참조하세요.
- PHP용 AWS SDK. 자세한 내용은 PHP용 AWS SDK API 참조의 네임스페이스 Aws\CodeBuild 섹션을 참조하세요.
- Python용 AWS SDK(Boto3). 자세한 내용은 Boto 3 설명서의 CodeBuild 섹션을 참조하세요.
- Ruby용 AWS SDK. 자세한 내용은 Ruby용 AWS SDK API 참조의 모듈: Aws::CodeBuild 섹션을 참조하세요.
- PowerShell용 AWS 도구. 자세한 내용은 PowerShell Cmdlet용 AWS 도구 참조의 AWS CodeBuild 섹션을 참조하세요.

AWS CodeBuild 엔드포인트 지정

AWS Command Line Interface(AWS CLI) 또는 AWS SDK 중 하나를 사용하여 AWS CodeBuild에서 사용하는 엔드포인트를 지정할 수 있습니다. CodeBuild를 사용할 수 있는 각 리전에 대한 엔드포인트가 있습니다. 4개 리전에는 1개의 리전 엔드포인트 외에 Federal Information Processing Standards(FIPS) 엔드포인트도 있습니다. FIPS 엔드포인트에 대한 자세한 내용은 FIPS 140-2 개요를 참조하세요.

엔드포인트 지정은 선택 사항입니다. CodeBuild에 사용할 엔드포인트를 명시적으로 지정하지 않으면 이 서비스는 AWS 계정이 사용하는 리전과 연결된 엔드포인트를 사용합니다. CodeBuild의 기본 엔드포인트는 FIPS 엔드포인트가 아닙니다. FIPS 엔드포인트를 사용하려면 다음 방법 중 하나를 사용하여 CodeBuild를 이 엔드포인트와 연결해야 합니다.

> **Note**
>
> 별칭 또는 리전 이름을 사용하여 AWS SDK에서 엔드포인트를 지정할 수 있습니다. AWS CLI를 사용할 경우 전체 엔드포인트 이름을 사용해야 합니다.

CodeBuild에 사용할 수 있는 엔드포인트에 대해서는 CodeBuild 리전 및 엔드포인트를 참조하세요.

주제

- AWS CodeBuild 엔드포인트 지정(AWS CLI)
- AWS CodeBuild 엔드포인트 지정(AWS SDK)

AWS CodeBuild 엔드포인트 지정(AWS CLI)

AWS CLI에서 모든 CodeBuild 명령의 `--endpoint-url` 인수를 사용하여 AWS CodeBuild에 액세스할 때 사용되는 엔드포인트를 지정할 수 있습니다. 예를 들어 다음 명령을 실행하여 미국 동부(버지니아 북부) 리전에서 Federal Information Processing Standards(FIPS) 엔드포인트를 통해 프로젝트 빌드 이름 목록을 가져옵니다.

```
aws codebuild list-projects --endpoint-url https://codebuild-fips.us-
east-1.amazonaws.com
```

엔드포인트 시작 부분에 `https://`를 포함합니다.

`--endpoint-url` AWS CLI 인수는 모든 AWS 서비스에서 사용할 수 있습니다. 이 인수 및 다른 AWS CLI 인수에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하세요.

AWS CodeBuild 엔드포인트 지정(AWS SDK)

AWS SDK를 사용하여 AWS CodeBuild에 액세스할 때 사용되는 엔드포인트를 지정할 수 있습니다. 이 예제에서는 Java용 AWS SDK를 사용하지만 다른 AWS SDK를 사용하여 엔드포인트를 지정할 수 있습니다.

AWSCodeBuild 클라이언트를 구성할 때는 이 `withEndpointConfiguration` 메서드를 사용합니다. 사용할 형식은 다음과 같습니다.

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",
 "region")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

`AWSCodeBuildClientBuilder`에 대한 자세한 내용은 클래스 AWSCodeBuildClientBuilder를 참조하세요.

`withCredentials`에서 사용되는 보안 인증은 `AWSCredentialsProvider` 형식이어야 합니다. 자세한 내용은 AWS 자격 증명 작업을 참조하십시오.

엔드포인트 시작 부분에 `https://`를 포함하지 마세요.

비 FIPS 엔드포인트를 지정하려는 경우 실제 엔드포인트 대신 리전을 사용할 수 있습니다. 예를 들어 미국 동부(버지니아 북부) 리전에서 엔드포인트를 지정하려면 전체 엔드포인트 이름 `codebuild.us-east-1.amazonaws.com` 대신 `us-east-1`을 사용할 수 있습니다.

FIPS 엔드포인트를 지정하려는 경우 별칭을 사용하여 코드를 단순화할 수 있습니다. FIPS 엔드포인트에만 별칭이 있습니다. 다른 엔드포인트는 해당 리전 또는 전체 이름을 사용하여 지정해야 합니다.

다음 표에는 사용 가능한 FIPS 엔드포인트 4개 각각의 별칭이 나열되어 있습니다.

리전 이름	Region	Endpoint	별칭
미국 동부(버지니아 북부)	us-east-1	codebuild-fips.us-east-1.amazonaws.com	us-east-1-fips

리전 이름	Region	Endpoint	별칭
미국 동부(오하이오)	us-east-2	codebuild-fips.us-east-2.amazonaws.com	us-east-2-fips
미국 서부(캘리포니아 북부)	us-west-1	codebuild-fips.us-west-1.amazonaws.com	us-west-1-fips
미국 서부(오레곤)	us-west-2	codebuild-fips.us-west-2.amazonaws.com	us-west-2-fips

별칭을 사용하여 미국 서부(오레곤) 리전에서 FIPS 엔드포인트 사용을 지정하려면:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-
fips", "us-west-2")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

미국 동부(버지니아 북부) 리전에서 비 FIPS 엔드포인트 사용을 지정하려면:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1",
 "us-east-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

아시아 태평양(뭄바이) 리전에서 비 FIPS 엔드포인트 사용을 지정하려면:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1",
 "ap-south-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

AWS CodePipeline를 AWS CodeBuild와 함께 사용하여 코드 테스트 및 빌드 실행

AWS CodePipeline를 사용하여 릴리스 프로세스를 자동화하면 AWS CodeBuild를 사용하여 코드를 테스트하고 빌드를 실행할 수 있습니다.

다음 표에는 태스크 및 태스크 수행에 사용할 수 있는 방법이 나와 있습니다. AWS SDK를 사용하여 이러한 태스크를 수행하는 내용은 본 주제에서 다루지 않습니다.

작업	사용 가능한 접근 방식	이 주제에 설명된 접근 방식
다음과 같이 빌드를 CodePipeline 자동화하는 지속적 전달 (CD) 파이프라인을 만드세요. CodeBuild	• CodePipeline 콘솔 • AWS CLI • AWS SDK	• CodePipeline 콘솔 사용 • AWS CLI 사용 • 이 주제의 정보를 AWS SDK를 사용하도록 조정할 수 있습니다. 자세한 내용은 Amazon Web Services용 도구의 SDK 섹션에서 프로그래밍 언어에 대한 `create-pipeline` 작업 설명서를 참조하거나 AWS CodePipeline API 참조에서 `CreatePipeline` 을 참조하세요.
테스트 및 빌드 CodeBuild 자동화를 기존 파이프라인에 추가 CodePipeline	• CodePipeline 콘솔 • AWS CLI • AWS SDK	• CodePipeline 콘솔을 사용하여 빌드 자동화를 추가하세요. • CodePipeline 콘솔을 사용하여 테스트 자동화를 추가합니다. • 의 AWS CLI 경우 이 항목의 정보를 수정하여 CodeBuild 빌드 작업 또는 테스트 작업이 포함된 파이프라인을 만들 수 있습니다. 자세한 내용은 AWS CodePipeline사용자 가이드의 파이프라인 편집 (AWS CLI) 및 CodePipeline 파이프라인 구조 참조를 참조하십시오. • 이 주제의 정보를 AWS SDK를 사용하도록 조정할 수 있습니다. 자세한 내용은 Amazon Web Services용 도구의 SDK 섹션을 통해 프로그래밍 언어에 대한 `update-pi`

작업	사용 가능한 접근 방식	이 주제에 설명된 접근 방식
		peline 작업 설명서를 참조하거나 AWS CodePipeline API 참조에서 UpdatePipeline 을 참조하세요.

필수 조건

1. 빌드 계획 섹션의 질문에 답하십시오.
2. AWS루트 계정이나 관리자 사용자 CodePipeline 대신 사용자를 사용하여 액세스하는 경우 사용자 (또는 사용자가 속한 IAM 그룹) AWSCodePipelineFullAccess 에게 이름이 지정된 관리형 정책을 연결하십시오. AWS 루트 계정을 사용하는 것은 권장되지 않습니다. 이 정책은 사용자에게 CodePipeline에서 파이프라인을 생성할 권한을 부여합니다. 자세한 내용은 사용 설명서의 관리형 정책 연결을 참조하세요.

   > **Note**
   >
   > 정책을 사용자(또는 사용자가 속한 IAM 그룹)에 연결하는 IAM 엔터티는 IAM에서 정책을 연결할 수 있는 권한이 있어야 합니다. 자세한 내용은 사용 설명서의 IAM 사용자, 그룹 및 보안 인증 관리 권한 위임을 참조하세요.

3. AWS계정에 아직 CodePipeline 서비스 역할이 없는 경우 서비스 역할을 생성하세요. CodePipeline 이 서비스 역할을 사용하여 다른 AWS 서비스 (예: 사용자 대신) 와 상호 작용합니다. AWS CodeBuild 예를 들어 를 사용하여 CodePipeline 서비스 역할을 AWS CLI 생성하려면 IAM create-role 명령을 실행합니다.

   Linux, macOS, Unix의 경우:

   ```
   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"}}'
   ```

   Windows의 경우:

   ```
   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**
>
> 이 CodePipeline 서비스 역할을 생성하는 IAM 개체는 IAM에서 서비스 역할을 생성할 수 있는 권한을 가지고 있어야 합니다.

4. 서비스 역할을 생성하거나 기존 CodePipeline 서비스 역할을 식별한 후에는 해당 역할 정책에 아직 포함되지 않은 경우 AWS CodePipeline사용 설명서의 기본 CodePipeline 서비스 역할 정책 검토에 설명된 대로 기본 서비스 역할 정책을 서비스 역할에 추가해야 합니다. CodePipeline

> **Note**
>
> 이 CodePipeline 서비스 역할 정책을 추가하는 IAM 개체는 IAM에서 서비스 역할 정책을 서비스 역할에 추가할 수 있는 권한이 있어야 합니다.

5. 소스 코드를 생성하여 CodeBuild 및 CodePipeline 에서 지원하는 리포지토리 유형 (예: Amazon S3 CodeCommit, Bitbucket 또는 GitHub) 에 업로드합니다. 소스 코드에 빌드 사양 파일을 포함해야 하지만, 이 주제의 후반부에서 빌드 프로젝트를 정의할 때 빌드 사양 파일을 선언할 수 있습니다. 자세한 내용은 buildspec 참조 섹션을 참조하십시오.

> **Important**
>
> 파이프라인을 사용하여 빌드 소스 코드를 배포하려는 경우, 빌드 출력 아티팩트와 사용할 배포 시스템이 호환 가능해야 합니다.
>
> - 자세한 AWS OpsWorks 내용은 사용 설명서의 AWS OpsWorks 애플리케이션 소스 및 CodePipeline 함께 AWS OpsWorks 사용을 참조하십시오.

주제

- CodeBuild를 사용하는 파이프라인 생성(CodePipeline 콘솔)
- CodeBuild를 사용하는 파이프라인 생성(AWS CLI)
- CodeBuild 빌드 작업을 파이프라인에 추가(CodePipeline 콘솔)
- CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔)

CodeBuild를 사용하는 파이프라인 생성(CodePipeline 콘솔)

다음 절차에 따라 CodeBuild를 사용하여 소스 코드를 빌드하고 배포하는 파이프라인을 생성하세요.

소스 코드만 테스트하는 파이프라인을 생성하려면:

- 다음 절차를 사용하여 파이프라인을 생성한 다음, 파이프라인에서 빌드 및 베타 단계를 삭제합니다. 그런 다음, 이 주제의 CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔) 절차를 사용하여 CodeBuild를 사용하는 테스트 작업을 파이프라인에 추가합니다.
- 이 주제의 다른 절차 중 하나를 사용하여 파이프라인을 생성한 다음, 이 주제의 CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔) 절차를 사용하여 CodeBuild를 사용하는 테스트 작업을 파이프라인에 추가합니다.

CodePipeline의 파이프라인 생성 마법사를 사용하여 CodeBuild를 사용하는 파이프라인을 생성하려면

1. 다음을 사용하여 AWS Management Console에 로그인합니다.

   - 사용자의 AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.
   - AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   - 최소한 다음 작업 세트를 수행할 수 있는 권한이 있는 AWS 계정의 사용자.

```
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. https://console.aws.amazon.com/codesuite/codepipeline/home에서 AWS CodePipeline 콘솔을 엽니다.
3. AWS 리전 선택기에서 빌드 프로젝트 AWS 리소스가 있는 AWS 리전을 선택합니다. 이 리전은 CodeBuild가 지원되는 AWS 리전이어야 합니다. 자세한 내용은 AWS CodeBuild의 Amazon Web Services 일반 참조 섹션을 참조하세요.
4. 파이프라인을 생성합니다. CodePipeline 정보 페이지가 표시되면 파이프라인 생성을 선택합니다. 파이프라인 페이지가 표시되면 파이프라인 생성을 선택합니다.
5. 1단계: 파이프라인 설정 선택 페이지의 파이프라인 이름에 파이프라인 이름(예: **CodeBuildDemoPipeline**)을 입력합니다. 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.
6. 역할 이름의 경우 다음 중 하나를 수행합니다.

   새 서비스 역할을 선택하고 역할 이름에 새 서비스 역할의 이름을 입력합니다.

   기존 서비스 역할을 선택한 다음, 이 주제의 필수 조건의 일부로 생성하거나 식별한 CodePipeline 서비스 역할을 선택합니다.
7. 아티팩트 스토어에서 다음 중 하나를 수행합니다.

   - 파이프라인에 대해 선택한 AWS 리전의 파이프라인에 대해 기본값으로 지정된 S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용하려면 기본 위치를 선택합니다.
   - 파이프라인과 동일한 AWS 리전에 S3 아티팩트 버킷과 같이 이미 생성한 기존 아티팩트 스토어가 있는 경우 사용자 지정 위치를 선택합니다.

   > **Note**
   >
   > 이는 파이프라인 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 파이프라인과 동일한 AWS 리전에 있는 각 파이프라인에 필요합니다.
8. 다음(Next)을 선택합니다.
9. 2단계: 소스 단계 추가 페이지에서 소스 공급자에 대해 다음 중 하나를 수행하세요.

- 소스 코드가 S3 버킷에 저장되어 있는 경우 Amazon S3를 선택합니다. 버킷에서 소스 코드를 포함하는 S3 버킷을 선택합니다. S3 객체 키의 경우 소스 코드가 들어 있는 파일의 이름(예: *file-name*.zip)을 입력합니다. 다음(Next)을 선택합니다.
- 소스 코드가 AWS CodeCommit 리포지토리에 저장되어 있는 경우 CodeCommit을 선택합니다. 리포지토리 이름의 경우 소스 코드가 포함된 리포지토리의 이름을 선택합니다. [Branch name]에서 빌드하려는 소스 코드의 버전이 포함된 브랜치 이름을 선택합니다. 다음(Next)을 선택합니다.
- 소스 코드가 GitHub 리포지토리에 저장되어 있는 경우 GitHub를 선택합니다. GitHub에 연결을 선택하고 지침에 따라 GitHub에서 인증을 받습니다. 리포지토리의 경우 소스 코드가 포함된 리포지토리의 이름을 선택합니다. [Branch]에서 빌드하려는 소스 코드의 버전이 포함된 브랜치 이름을 선택합니다.

다음(Next)을 선택합니다.

10. 3단계: 빌드 단계 추가 페이지에서 빌드 공급자에 대해 CodeBuild를 선택합니다.
11. 사용하려는 빌드 프로젝트가 이미 있는 경우 프로젝트 이름에서 빌드 프로젝트의 이름을 선택하고 이 절차의 다음 단계로 건너뜁니다.

    새 CodeBuild 빌드 프로젝트를 생성해야 하는 경우 빌드 프로젝트 만들기(콘솔)의 지침을 따르고 이 절차로 돌아갑니다.

    기존 빌드 프로젝트를 선택하는 경우 CodePipeline에서 재정의되더라도 빌드 출력 아티팩트 설정이 이미 정의되어 있어야 합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.

    > **Important**
    >
    > CodeBuild 프로젝트에 대해 webhook를 활성화하고 해당 프로젝트가 CodePipeline의 빌드 단계로 사용되는 경우 각 커밋에 대해 두 개의 동일한 빌드가 생성됩니다. 하나의 빌드는 webhook를 통해 트리거되고 다른 하나는 CodePipeline을 통해 트리거됩니다. 빌드 기준으로 요금이 청구되므로 두 빌드 모두에 대해 요금이 청구됩니다. 따라서 CodePipeline을 사용하는 경우 CodeBuild에서 webhook를 비활성화하는 것이 좋습니다. AWS CodeBuild 콘솔에서 Webhook 상자를 해제합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.

12. 4단계: 배포 단계 추가 페이지에서 다음 중 하나를 수행합니다.

- 빌드 출력 아티팩트를 배포하지 않으려면 건너뛰기를 선택하고 메시지가 표시되면 이 옵션을 확인합니다.
- 빌드 출력 아티팩트를 배포하려는 경우 배포 공급자에 대해 배포 공급자를 선택한 다음, 메시지가 표시되면 설정을 지정합니다.

다음(Next)을 선택합니다.

13. 검토 페이지에서 선택 사항을 검토한 다음, 파이프라인 생성을 선택합니다.
14. 파이프라인이 성공적으로 실행되면 빌드 출력 아티팩트를 가져올 수 있습니다. CodePipeline 콘솔에 파이프라인이 표시된 상태에서 빌드 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyAppBuild)의 값을 적어 놓습니다.

> **Note**
>
> CodeBuild 콘솔의 빌드 세부 정보 페이지에서 빌드 아티팩트 링크를 선택하여 빌드 출력 아티팩트를 가져올 수도 있습니다. 이 페이지로 이동하려면 이 절차의 나머지 단계를 건너뛰고 빌드 세부 정보 보기(콘솔) 섹션을 참조하세요.

15. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
16. 버킷 목록에서 파이프라인에서 사용하는 버킷을 엽니다. 버킷의 이름은 `codepipeline-`*`region-ID`*`-`*`random-number`* 형식을 따릅니다. AWS CLI에서 CodePipeline get-pipeline 명령을 실행하여 버킷 이름을 가져올 수 있습니다. 여기서 *`my-pipeline-name`*은 파이프라인의 표시 이름입니다.

```
aws codepipeline get-pipeline --name my-pipeline-name
```

출력에서 `pipeline` 객체는 `artifactStore` 객체를 포함하며, 이 객체에는 버킷 이름의 `location` 값이 들어 있습니다.

17. 파이프라인의 이름과 일치하는 폴더를 열고(파이프라인의 이름 길이에 따라 폴더 이름이 잘릴 수 있음) 앞에서 적어 둔 출력 아티팩트 값과 일치하는 폴더를 엽니다.
18. 파일 내용의 압축을 풉니다. 해당 폴더에 파일이 여러 개 있는 경우 가장 최근의 마지막 수정 시간 타임스탬프를 사용하여 파일의 내용을 추출합니다. (시스템의 ZIP 유틸리티에서 작업할 수 있도록 파일에 `.zip` 확장자를 지정해야 할 수도 있습니다.) 빌드 출력 아티팩트는 파일의 추출된 내용에 있습니다.
19. CodePipeline에 빌드 출력 아티팩트를 배포하도록 지시한 경우 배포 공급자의 지침을 사용하여 배포 대상의 빌드 출력 아티팩트로 이동합니다.

CodeBuild를 사용하는 파이프라인 생성(AWS CLI)

다음 절차에 따라 CodeBuild를 사용하여 소스 코드를 빌드하는 파이프라인을 생성하세요.

AWS CLI를 사용하여 빌드된 소스 코드를 배포하거나 소스 코드만 테스트하는 파이프라인을 생성하려면 AWS CodePipeline 사용 설명서의 파이프라인 편집(AWS CLI) 및 CodePipeline 파이프라인 구조 참조의 지침에 따라 작업할 수 있습니다.

1. CodeBuild에서 빌드 프로젝트를 생성하거나 식별합니다. 자세한 내용은 빌드 프로젝트 생성 섹션을 참조하세요.

   > **Important**
   >
   > CodePipeline에서 재정의되더라도 빌드 프로젝트는 빌드 출력 아티팩트 설정을 정의해야 합니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI)의 `artifacts` 설명을 참조하십시오.

2. 이 주제에서 설명한 대로 IAM 엔터티 중 하나에 해당하는 AWS 액세스 키 및 AWS 비밀 액세스 키를 사용하여 AWS CLI를 구성했는지 확인합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 설정을 참조하세요.

3. 파이프라인 구조를 나타내는 JSON 형식의 파일을 생성합니다. 파일 이름을 `create-pipeline.json` 또는 비슷한 이름으로 지정합니다. 예를 들어, 다음 JSON 형식 구조는 CodeBuild를 사용하는 빌드 작업 및 S3 입력 버킷을 참조하는 소스 작업으로 파이프라인을 생성합니다.

```
{
  "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
  }
}
```

이 JSON 형식의 데이터에서는 다음이 적용됩니다.

- `roleArn`의 값은 사용자가 생성했거나 사전 요구 사항의 일부로 식별한 CodePipeline 서비스 역할의 ARN과 일치해야 합니다.
- `configuration`의 `S3Bucket` 및 `S3ObjectKey` 값은 소스 코드가 S3 버킷에 저장되어 있다고 가정합니다. 다른 소스 코드 리포지토리 유형에 대한 설정은 사용 AWS CodePipeline 사용 설명서의 CodePipeline 파이프라인 구조 참조를 참조하세요.
- `ProjectName`의 값은 이 절차의 앞부분에서 생성한 CodeBuild 빌드 프로젝트의 이름입니다.
- `location`의 값은 이 파이프라인에서 사용하는 S3 버킷의 이름입니다. 자세한 내용은 AWS CodePipeline 사용 설명서의 CodePipeline의 아티팩트 스토어로 사용할 S3 버킷에 대한 정책 생성을 참조하세요.
- `name`의 값은 이 파이프라인의 이름입니다. 모든 파이프라인 이름은 계정에서 고유해야 합니다.

이 데이터는 소스 작업과 빌드 작업만 설명하지만 테스트, 빌드 출력 아티팩트 배포, AWS Lambda 함수 호출 등과 관련된 활동에 대한 작업을 추가할 수 있습니다. 자세한 내용은 AWS CodePipeline 사용 설명서의 AWS CodePipeline 파이프라인 구조 참조를 참조하세요.

4. JSON 파일이 들어 있는 폴더로 전환한 다음, 파일 이름을 지정하여 create-pipeline CodePipeline 명령을 실행합니다.

```
aws codepipeline create-pipeline --cli-input-json file://create-pipeline.json
```

> **Note**
>
> CodeBuild가 지원되는 AWS 리전에 파이프라인을 생성해야 합니다. 자세한 내용은 AWS CodeBuild의 Amazon Web Services 일반 참조 섹션을 참조하세요.

JSON 형식의 데이터가 출력에 나타나고 CodePipeline이 파이프라인을 생성합니다.

5. 파이프라인 상태에 대한 정보를 가져오려면 파이프라인 이름을 지정하여 CodePipeline get-pipeline-state 명령을 실행합니다.

```
aws codepipeline get-pipeline-state --name <my-pipeline-name>
```

출력에서 빌드가 성공했음을 확인하는 정보를 찾습니다. 간결하게 나타내기 위해 생략된 데이터를 표시하는 데 줄임표(...)가 사용됩니다.

```
{
  ...
  "stageStates": [
    ...
    {
      "actionStates": [
        {
          "actionName": "CodeBuild",
          "latestExecution": {
            "status": "SUCCEEDED",
            ...
          },
          ...
        }
      ]
    }
  ]
}
```

이 명령을 너무 일찍 실행하면 빌드 작업에 대해 어떤 정보도 표시되지 않을 수 있습니다. 파이프라인에서 빌드 작업 실행이 완료될 때까지 이 명령을 여러 번 실행해야 할 수도 있습니다.

6. 빌드에 성공하면 다음 지침에 따라 빌드 출력 아티팩트를 가져오세요. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.

> **Note**
>
> CodeBuild 콘솔의 관련 빌드 세부 정보 페이지에서 빌드 아티팩트 링크를 선택하여 빌드 출력 아티팩트를 가져올 수도 있습니다. 이 페이지로 이동하려면 이 절차의 나머지 단계를 건너뛰고 빌드 세부 정보 보기(콘솔) 섹션을 참조하세요.

7. 버킷 목록에서 파이프라인에서 사용하는 버킷을 엽니다. 버킷의 이름은 codepipeline-*<region-ID>*-*<random-number>* 형식을 따릅니다. create-

pipeline.json 파일에서 버킷 이름을 가져오거나 CodePipeline get-pipeline 명령을 실행하여 버킷 이름을 가져올 수 있습니다.

```
aws codepipeline get-pipeline --name <pipeline-name>
```

출력에서 pipeline 객체는 artifactStore 객체를 포함하며, 이 객체에는 버킷 이름의 location 값이 들어 있습니다.

8. 파이프라인 이름(예: *<pipeline-name>*)과 일치하는 폴더를 엽니다.
9. 해당 폴더에서 이름이 default인 폴더를 엽니다.
10. 파일 내용의 압축을 풉니다. 해당 폴더에 파일이 여러 개 있는 경우 가장 최근의 마지막 수정 시간 타임스탬프를 사용하여 파일의 내용을 추출합니다. (시스템의 ZIP 유틸리티에서 작업할 수 있도록 파일에 .zip 확장자를 지정해야 할 수도 있습니다.) 빌드 출력 아티팩트는 파일의 추출된 내용에 있습니다.

CodeBuild 빌드 작업을 파이프라인에 추가(CodePipeline 콘솔)

1. 다음을 사용하여 AWS Management Console에 로그인합니다.
   - 사용자의 AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.
   - AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   - 최소한 다음 작업 세트를 수행할 수 있는 권한이 있는 AWS 계정의 사용자.

```
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
```

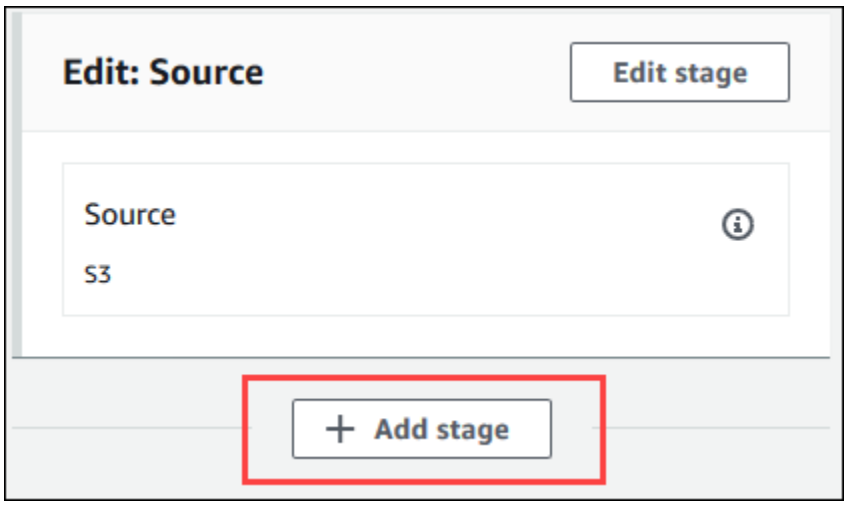
```
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. https://console.aws.amazon.com/codesuite/codepipeline/home에서 CodePipeline 콘솔을 엽니다.
3. AWS 리전 선택기에서 파이프라인이 위치하는 AWS 리전을 선택합니다. 이는 CodeBuild가 지원되는 리전이어야 합니다. 자세한 내용은 Amazon Web Services 일반 참조의 CodeBuild를 참조하세요.
4. 파이프라인 페이지에서 파이프라인의 이름을 선택합니다.
5. 파이프라인 세부 정보 페이지의 소스 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyApp)의 값을 적어 놓습니다.

> **Note**
>
> 이 절차에서는 소스와 베타 단계 사이의 빌드 단계에 빌드 작업을 추가하는 방법을 보여 줍니다. 빌드 작업을 다른 위치에 추가하려면 빌드 작업을 추가할 위치 바로 앞에 있는 작업의 도구 설명을 선택하고 출력 아티팩트 값을 기록해 둡니다.

6. 편집(Edit)을 선택합니다.
7. 소스 단계와 베타 단계 사이에서 단계 추가를 선택합니다.

> **Note**
>
> 이 절차에서는 소스와 베타 단계 사이의 빌드 단계를 파이프라인에 추가하는 방법을 보여줍니다. 기존 단계에 빌드 작업을 추가하려면 단계에서 단계 편집을 선택한 다음, 이 절차의 8단계로 건너뜁니다. 빌드 단계를 다른 위치에 추가하려면 원하는 위치에서 단계 추가를 선택합니다.



8. 단계 이름에 빌드 단계 이름(예:**Build**)을 입력합니다. 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
9. 선택한 단계 내에서 작업 추가를 선택합니다.

> **Note**
>
> 이 절차에서는 빌드 단계 내에 빌드 작업을 추가하는 방법을 보여줍니다. 빌드 작업을 다른 위치에 추가하려면 원하는 위치에서 작업 추가를 선택합니다. 먼저 빌드 작업을 추가하려는 기존 단계에서 단계 편집을 선택해야 할 수 있습니다.

10. 작업 편집의 작업 이름에 작업 이름을 입력합니다(예: **CodeBuild**). 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
11. 작업 공급자의 경우 CodeBuild를 선택합니다.
12. 사용하려는 빌드 프로젝트가 이미 있는 경우 프로젝트 이름에서 빌드 프로젝트의 이름을 선택하고 이 절차의 다음 단계로 건너뜁니다.

    새 CodeBuild 빌드 프로젝트를 생성해야 하는 경우 빌드 프로젝트 만들기(콘솔)의 지침을 따르고 이 절차로 돌아갑니다.

    기존 빌드 프로젝트를 선택하는 경우 CodePipeline에서 재정의되더라도 빌드 출력 아티팩트 설정이 이미 정의되어 있어야 합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 또는 빌드 프로젝트 설정 변경(콘솔)의 아티팩트 설명을 참조하세요.

> **Important**
>
> CodeBuild 프로젝트에 대해 webhook를 활성화하고 해당 프로젝트가 CodePipeline의 빌드 단계로 사용되는 경우 각 커밋에 대해 두 개의 동일한 빌드가 생성됩니다. 하나의 빌드는 webhook를 통해 트리거되고 다른 하나는 CodePipeline을 통해 트리거됩니다. 빌드 기준으로 요금이 청구되므로 두 빌드 모두에 대해 요금이 청구됩니다. 따라서 CodePipeline을 사용하는 경우 CodeBuild에서 webhook를 비활성화하는 것이 좋습니다. CodeBuild 콘솔에서 Webhook 상자를 선택 취소합니다. 자세한 정보는 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.

13. 입력 아티팩트에서 이 절차의 앞에서 적어 둔 출력 아티팩트를 선택합니다.
14. 출력 아티팩트의 경우 출력 아티팩트의 이름(예: **MyAppBuild**)을 입력합니다.
15. 작업 추가를 선택합니다.
16. 저장을 선택한 다음, 저장을 선택하여 변경 사항을 파이프라인에 저장합니다.
17. 변경 사항 릴리스를 선택합니다.
18. 파이프라인이 성공적으로 실행되면 빌드 출력 아티팩트를 가져올 수 있습니다. CodePipeline 콘솔에 파이프라인이 표시된 상태에서 빌드 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyAppBuild)의 값을 적어 놓습니다.

> **Note**
>
> CodeBuild 콘솔의 빌드 세부 정보 페이지에서 빌드 아티팩트 링크를 선택하여 빌드 출력 아티팩트를 가져올 수도 있습니다. 이 페이지로 이동하려면 빌드 세부 정보 보기(콘솔)을 참조하고 이 절차의 31단계로 건너뛰세요.

19. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
20. 버킷 목록에서 파이프라인에서 사용하는 버킷을 엽니다. 버킷의 이름은 `codepipeline-`*`region-ID`*`-`*`random-number`* 형식을 따릅니다. AWS CLI에서 CodePipeline get-pipeline 명령을 실행하여 버킷의 이름을 가져옵니다.

```
aws codepipeline get-pipeline --name my-pipeline-name
```

출력에서 `pipeline` 객체는 `artifactStore` 객체를 포함하며, 이 객체에는 버킷 이름의 `location` 값이 들어 있습니다.

21. 파이프라인의 이름과 일치하는 폴더를 열고(파이프라인의 이름 길이에 따라 폴더 이름이 잘릴 수 있음) 이 절차의 앞에서 적어 둔 출력 아티팩트 값과 일치하는 폴더를 엽니다.
22. 파일 내용의 압축을 풉니다. 해당 폴더에 파일이 여러 개 있는 경우 가장 최근의 마지막 수정 시간 타임스탬프를 사용하여 파일의 내용을 추출합니다. (시스템의 ZIP 유틸리티에서 작업할 수 있도록 파일에 `.zip` 확장자를 지정해야 할 수도 있습니다.) 빌드 출력 아티팩트는 파일의 추출된 내용에 있습니다.
23. CodePipeline에 빌드 출력 아티팩트를 배포하도록 지시한 경우 배포 공급자의 지침을 사용하여 배포 대상의 빌드 출력 아티팩트로 이동합니다.

CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔)

1. 다음을 사용하여 AWS Management Console에 로그인합니다.
   - 사용자의 AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.
   - AWS 계정의 관리자 사용자. 자세한 내용은 사용 설명서에서 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   - 최소한 다음 작업 세트를 수행할 수 있는 권한이 있는 AWS 계정의 사용자.

```
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
```

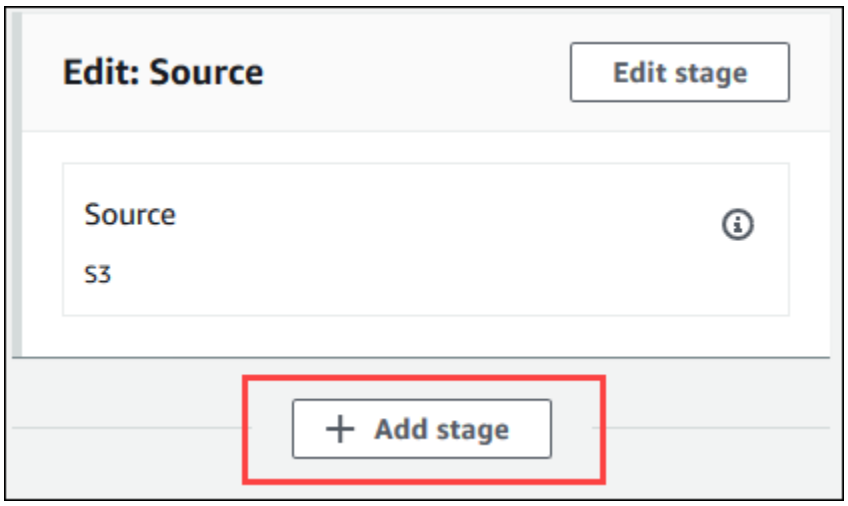
```
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. https://console.aws.amazon.com/codesuite/codepipeline/home에서 CodePipeline 콘솔을 엽니다.
3. AWS 리전 선택기에서 파이프라인이 위치하는 AWS 리전을 선택합니다. 이 리전은 CodeBuild가 지원되는 AWS 리전이어야 합니다. 자세한 내용은 AWS CodeBuild의 Amazon Web Services 일반 참조 섹션을 참조하세요.
4. 파이프라인 페이지에서 파이프라인의 이름을 선택합니다.
5. 파이프라인 세부 정보 페이지의 소스 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyApp)의 값을 적어 놓습니다.

> **Note**
>
> 이 절차에서는 소스와 베타 단계 사이의 테스트 단계에 테스트 작업을 추가하는 방법을 보여 줍니다. 테스트 작업을 다른 위치에 추가하려면 마우스 포인터를 바로 앞에 있는 작업에 놓고 출력 아티팩트의 값을 기록해 둡니다.

6. 편집(Edit)을 선택합니다.
7. 소스 단계 바로 다음에 단계 추가를 선택합니다.

> **Note**
>
> 또한 이 절차에서는 소스 단계 바로 다음의 테스트 단계를 파이프라인에 추가하는 방법을 보여줍니다. 기존 단계에 테스트 작업을 추가하려면 단계에서 단계 편집을 선택한 다음, 이 절차의 8단계로 건너뜁니다. 테스트 단계를 다른 위치에 추가하려면 원하는 위치에서 단계 추가를 선택합니다.



8. 단계 이름에 테스트 단계 이름(예:**Test**)을 입력합니다. 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
9. 선택한 단계에서 작업 추가를 선택합니다.

> **Note**
>
> 이 절차에서는 테스트 단계에 테스트 작업을 추가하는 방법을 보여 줍니다. 테스트 작업을 다른 위치에 추가하려면 원하는 위치에서 작업 추가를 선택합니다. 테스트 작업을 추가하려는 기존 단계에서 먼저 편집을 선택해야 할 수도 있습니다.

10. 작업 편집의 작업 이름에 작업 이름을 입력합니다(예: **Test**). 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
11. 작업 제공자의 경우 테스트에서 CodeBuild를 선택합니다.
12. 사용하려는 빌드 프로젝트가 이미 있는 경우 프로젝트 이름에서 빌드 프로젝트의 이름을 선택하고 이 절차의 다음 단계로 건너뜁니다.

    새 CodeBuild 빌드 프로젝트를 생성해야 하는 경우 빌드 프로젝트 만들기(콘솔)의 지침을 따르고 이 절차로 돌아갑니다.

> **Important**
>
> CodeBuild 프로젝트에 대해 webhook를 활성화하고 해당 프로젝트가 CodePipeline의 빌드 단계로 사용되는 경우 각 커밋에 대해 두 개의 동일한 빌드가 생성됩니다. 하나의 빌드는 webhook를 통해 트리거되고 다른 하나는 CodePipeline을 통해 트리거됩니다. 빌드 기

준으로 요금이 청구되므로 두 빌드 모두에 대해 요금이 청구됩니다. 따라서 CodePipeline을 사용하는 경우 CodeBuild에서 webhook를 비활성화하는 것이 좋습니다. CodeBuild 콘솔에서 Webhook 상자를 선택 취소합니다. 자세한 정보는 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.

13. 입력 아티팩트에 이 절차의 앞에서 적어 둔 출력 아티팩트 값을 입력합니다.
14. (선택 사항) 테스트 작업에서 출력 결과물을 생성하도록 하려고 하며 이에 맞게 빌드 사양을 설정했다면, 출력 아티팩트에 출력 결과물에 지정하려는 값을 입력합니다.
15. Save를 선택합니다.
16. 변경 사항 릴리스를 선택합니다.
17. 파이프라인이 성공적으로 실행되면 테스트 결과를 얻을 수 있습니다. 파이프라인의 테스트 단계에서 CodeBuild 하이퍼링크를 선택하여 CodeBuild 콘솔에서 관련 빌드 프로젝트 페이지를 엽니다.
18. 빌드 프로젝트 페이지의 빌드 이력에서 빌드 실행 하이퍼링크를 선택합니다.
19. 빌드 실행 페이지의 빌드 로그에서 전체 로그 보기 하이퍼링크를 선택하여 Amazon CloudWatch 콘솔에서 빌드 로그를 엽니다.
20. 빌드 로그를 스크롤하여 테스트 결과를 확인합니다.

Jenkins에서 AWS CodeBuild 사용

AWS CodeBuild용 Jenkins 플러그인을 사용하면 Jenkins 빌드 작업과 CodeBuild를 통합할 수 있습니다. 빌드 작업을 Jenkins 빌드 노드로 보내는 대신, 이 플러그인을 사용하여 CodeBuild로 빌드 작업을 전송합니다. 따라서 Jenkins 빌드 노드를 프로비저닝, 구성 및 관리할 필요가 없습니다.

Jenkins 설정

AWS CodeBuild 플러그인으로 Jenkins를 설정하는 방법과 플러그인 소스 코드를 다운로드하는 방법에 대한 자세한 내용은 https://github.com/awslabs/aws-codebuild-jenkins-plugin을 참조하세요.

플러그인 설치

Jenkins 서버를 이미 설정했고 AWS CodeBuild 플러그인만 설치하려는 경우 Jenkins 인스턴스의 Plugin Manager에서 **`CodeBuild Plugin for Jenkins`**를 검색합니다.

플러그인 사용

VPC 외부의 소스에서 AWS CodeBuild를 사용하려면

1. CodeBuild 콘솔에서 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.
   - 빌드를 실행하려는 AWS 리전을 선택합니다.
   - (선택 사항) CodeBuild 빌드 컨테이너가 VPC의 리소스에 액세스할 수 있도록 Amazon VPC 구성을 설정합니다.
   - 프로젝트 이름을 적어 둡니다. 3단계에서 이 이름이 필요합니다.
   - (선택 사항) CodeBuild에서 소스 리포지토리를 기본적으로 지원하지 않는 경우 Amazon S3를 프로젝트의 입력 소스 유형으로 설정할 수 있습니다.
2. IAM 콘솔에서 Jenkins 플러그인에서 사용할 사용자를 생성합니다.
   - 사용자에 대한 보안 인증을 생성할 때는 프로그래밍 방식 액세스를 선택합니다.
   - 다음과 유사한 정책을 만든 다음, 정책을 사용자에게 연결합니다.

```
{
  "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. Jenkins에서 자유형 프로젝트를 생성합니다.

   - 구성 페이지에서 빌드 단계 추가를 선택한 다음, CodeBuild에서 빌드 실행을 선택합니다.
   - 빌드 단계를 구성합니다.
     - 리전, 보안 인증, 프로젝트 이름에 값을 입력합니다.
     - 프로젝트 소스 사용을 선택합니다.
     - 구성을 저장하고 Jenkins에서 빌드를 실행합니다.
4. 소스 코드 관리에서 원하는 소스 검색 방법을 선택합니다. Jenkins 서버에 GitHub 플러그인(또는 소스 리포지토리 공급자용 Jenkins 플러그인)을 설치해야 할 수 있습니다.

   - 구성 페이지에서 빌드 단계 추가를 선택한 다음, AWS CodeBuild에서 빌드 실행을 선택합니다.
   - 빌드 단계를 구성합니다.
     - 리전, 보안 인증, 프로젝트 이름에 값을 입력합니다.
     - Jenkins 소스 사용을 선택합니다.
     - 구성을 저장하고 Jenkins에서 빌드를 실행합니다.

Jenkins 파이프라인 플러그인과 함께 AWS CodeBuild 플러그인을 사용하려면

- Jenkins 파이프라인 프로젝트 페이지에서 스니펫 생성기를 사용하여 CodeBuild를 파이프라인의 단계로 추가하는 파이프라인 스크립트를 생성합니다. 다음과 유사한 스크립트가 생성됩니다.

```
awsCodeBuild projectName: 'project', credentialsType: 'keys', region: 'us-west-2',
 sourceControlType: 'jenkins'
```

Codecov와 함께 AWS CodeBuild 사용

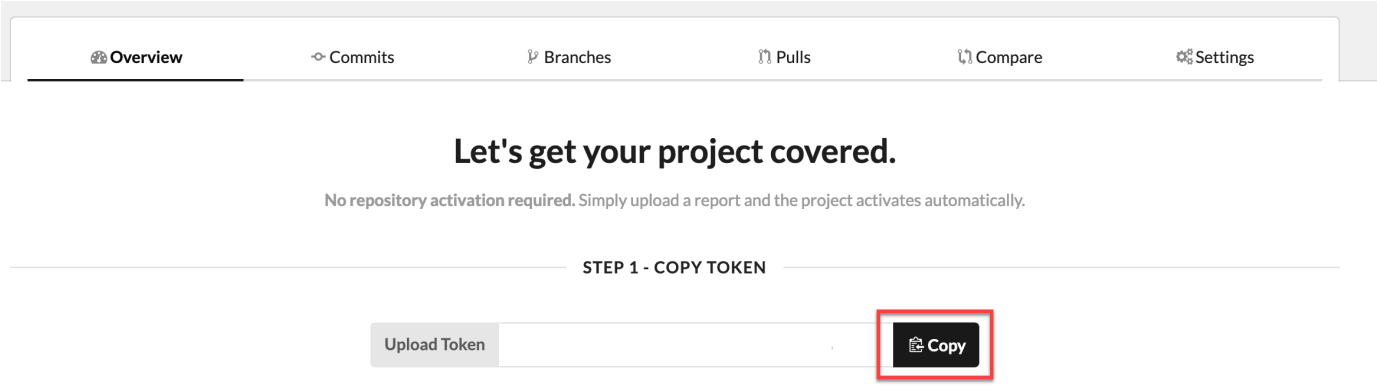
Codecov는 코드의 테스트 범위를 측정하는 도구입니다. Codecov는 코드에서 테스트되지 않은 메서드와 문을 식별합니다. 결과를 사용하여 코드의 품질을 향상시키기 위해 테스트를 작성할 위치를 결정합니다. Codecov는 CodeBuild에서 지원되는 3개 소스 리포지토리인 GitHub, GitHub Enterprise Server 및 Bitbucket을 사용할 수 있습니다. 빌드 프로젝트가 GitHub Enterprise Server를 사용하는 경우 Codecov Enterprise를 사용해야 합니다.

Codecov와 통합된 CodeBuild 프로젝트의 빌드를 실행하면 리포지토리의 코드를 분석하는 Codecov 보고서가 Codecov에 업로드됩니다. 빌드 로그에는 보고서로 연결되는 링크가 있습니다. 이 샘플은 Python과 Java 빌드 프로젝트를 Codecov와 통합하는 방법을 보여줍니다. Codecov에서 지원하는 언어 목록은 Codecov 웹사이트의 Codecov 지원 언어를 참조하세요.

Codecov를 빌드 프로젝트와 통합

Codecov를 빌드 프로젝트에 통합

1. https://codecov.io/signup으로 이동하여 GitHub 또는 Bitbucket 소스 리포지토리에 등록합니다. GitHub Enterprise를 사용하는 경우 Codecov 웹사이트의 Codecov Enterprise를 참조하세요.
2. Codecov에서 적용 범위를 원하는 리포지토리를 추가합니다.
3. 토큰 정보가 표시되면 복사(Copy)를 선택합니다.



4. 복사된 토큰을 빌드 프로젝트에 이름이 `CODECOV_TOKEN`인 환경 변수로 추가합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.
5. 리포지토리에서 `my_script.sh`(이)라는 텍스트 파일을 생성합니다. 다음을 파일에 입력합니다.

```
#/bin/bash
bash <(curl -s https://codecov.io/bash) -t $CODECOV_TOKEN
```

6. 빌드 프로젝트 사용에 적합한 Python 또는 Java 탭을 선택하고 다음 단계를 수행합니다.

Java

1. 다음 JaCoCo 플러그인을 리포지토리의 pom.xml에 추가합니다.

```
<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. buildspec 파일에 다음 명령을 입력합니다. 자세한 내용은 buildspec 구문 섹션을 참조하세요.

```
build:
  - mvn test -f pom.xml -fn
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

Python

buildspec 파일에 다음 명령을 입력합니다. 자세한 내용은 buildspec 구문 섹션을 참조하세요.

```
build:
  - pip install coverage
  - coverage run -m unittest discover
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

7. 빌드 프로젝트의 빌드를 실행합니다. 프로젝트에 대해 생성된 Codecov 보고서에 대한 링크가 빌드 로그에 나타납니다. Codecov 보고서를 보려면 링크를 사용합니다. 자세한 정보는 AWS CodeBuild에서 빌드 실행 및 AWS CloudTrail을 사용하여 AWS CodeBuild API 직접 호출 로깅 섹션을 참조하세요. 빌드 로그의 Codecov 정보는 다음과 같습니다.

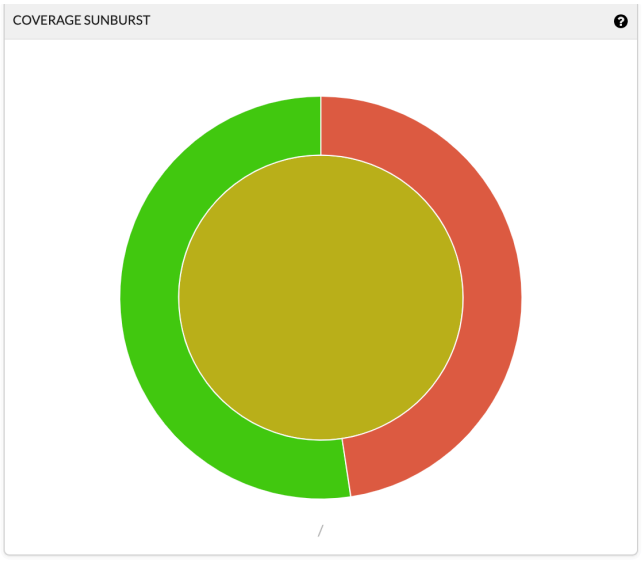
```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh


  _____          _
 / ____|        | |
| |     ___   __| | ___  ___ _____   __
| |    / _ \ / _` |/ _ \/ __/ _ \ \ / /
| |___| (_) | (_| |  __/ (_| (_) \ 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
```

보고서는 다음과 같습니다.



Files				Coverage
code.py	10	7	0	3	70.00%
tests.py	11	11	0	0	100.00%
Project Totals (2 files)	21	18	0	3	85.71%

서버리스 애플리케이션에서 AWS CodeBuild 사용

AWS Serverless Application Model(AWS SAM)은 서버리스 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. 자세한 내용은 GitHub에서 AWS Serverless Application Model 리포지토리를 참조하십시오.

AWS SAM 표준을 따르는 서버리스 애플리케이션을 패키징하고 배포하는 데 AWS CodeBuild를 사용할 수 있습니다. 배포 단계에는 CodeBuild에서 AWS CloudFormation을 사용할 수 있습니다. CodeBuild 및 AWS CloudFormation을 사용하여 서버리스 애플리케이션의 빌드 및 배포를 자동화하려면 AWS CodePipeline을 사용할 수 있습니다.

자세한 내용을 알아보려면 AWS Serverless Application Model 개발자 안내서의 서버리스 애플리케이션 배포를 참조하세요.

관련 리소스

- AWS CodeBuild 시작하기에 대한 자세한 내용은 콘솔을 사용하여 AWS CodeBuild 시작하기 단원을 참조하십시오.

- CodeBuild의 문제 해결에 대한 자세한 내용은 문제 해결 AWS CodeBuild 섹션을 참조하세요.
- CodeBuild의 할당량에 대한 자세한 내용은 AWS CodeBuild에 대한 할당량 섹션을 참조하세요.

문제 해결 AWS CodeBuild

이 주제의 정보를 활용하면 문제를 식별, 진단 및 해결하는 데 도움이 됩니다. CodeBuild 빌드를 기록하고 모니터링하여 문제를 해결하는 방법을 알아보려면 을 참조하십시오로깅 및 모니터링.

주제

- 오류: 빌드 대기열의 빌드가 실패할 때 "QUEUED: INSUFFICIENT_SUBNET"이라는 메시지가 표시됨
- 오류: “캐시를 다운로드할 수 없음: RequestError: 전송 요청 실패 원인: x509: 시스템 루트를 로드하지 못했고 루트가 제공되지 않았습니다.”
- 오류: “S3에서 인증서를 다운로드할 수 없습니다. AccessDenied
- 오류: "Unable to locate credentials"
- RequestError 프록시 서버에서 실행할 CodeBuild 때 시간 초과 오류가 발생했습니다.
- 빌드 이미지에 있어야 하는 Bourne 쉘(sh)
- 경고: 빌드 실행 시 “런타임 설치를 건너뜁니다. 런타임 버전 선택은 이 빌드 이미지에서 지원되지 않습니다.”가 표시됨
- 오류: CodeBuild 콘솔을 열 때 “ JobWorker ID를 확인할 수 없습니다.”
- 빌드를 시작하지 못함
- 로컬에 캐시된 빌드의 GitHub 메타데이터에 액세스
- AccessDenied: 보고서 그룹의 버킷 소유자가 S3 버킷 소유자와 일치하지 않습니다...

Apache Maven 빌드가 잘못된 리포지토리의 아티팩트를 참조함

문제: Maven을 AWS CodeBuild제공된 Java 빌드 환경과 함께 사용하는 경우 Maven은 https://repo1.maven.org/maven2 의 안전한 중앙 Maven 리포지토리에서 빌드 및 플러그인 종속성을 가져옵니다. 빌드 프로젝트의 `pom.xml` 파일에 대신 사용할 다른 위치가 명시적으로 선언되어 있는 경우에도 이러한 문제가 발생합니다.

가능한 원인: CodeBuild -제공된 Java 빌드 환경에 빌드 환경 디렉터리에 사전 설치된 이름이 지정된 `settings.xml` 파일이 포함되어 있습니다. `/root/.m2` 이 `settings.xml` 파일에는 다음 선언이 포함되어 있는데, 이는 Maven이 항상 https://repo1.maven.org/maven2에 있는 보안된 중앙 Maven 리포지토리로부터 빌드 및 플러그인 종속성을 가져오도록 지시합니다.

```
<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>
```

권장 솔루션: 다음을 실행합니다.

1. 소스 코드에 `settings.xml` 파일을 추가합니다.
2. 이 `settings.xml` 파일에서, 이전 `settings.xml` 형식을 참고하여 Maven이 빌드 및 플러그인 종속성을 가져오게 하려는 다른 리포지토리를 선언합니다.
3. 빌드 프로젝트 `install` 단계에서 `settings.xml` 파일을 빌드 환경의 디렉터리에 CodeBuild 복사하도록 지시하세요. `/root/.m2` 예를 들어 이러한 작업을 수행하는 `buildspec.yml` 파일의 다음 코드 조각을 고려해 보십시오.

```
version 0.2

phases:
  install:
    commands:
      - cp ./settings.xml /root/.m2/settings.xml
```

기본적으로 루트로 실행되는 빌드 명령

문제: 루트 사용자로 빌드 명령을 AWS CodeBuild 실행합니다. 관련 빌드 이미지의 Dockerfile이 USER 명령을 다른 사용자에게 설정하는 경우에도 이러한 문제가 발생합니다.

원인: 기본적으로 모든 빌드 명령을 루트 사용자로 CodeBuild 실행합니다.

권장 솔루션: 없음.

파일 이름에 미국 영어 이외의 문자가 있으면 빌드가 실패할 수 있음

문제: 영어 이외의 문자(예: 중국어 문자)가 포함된 파일 이름을 사용하는 빌드를 실행하면 빌드가 실패합니다.

가능한 원인: 에서 AWS CodeBuild 제공하는 빌드 환경의 기본 로케일이 로 `POSIX` 설정되어 있습니다. `POSIX`현지화 설정은 미국 이외의 국가를 포함하는 파일 CodeBuild 이름과 호환되지 않습니다. 영문자이므로 관련 빌드가 실패할 수 있습니다.

권장 솔루션: 다음 명령을 buildspec 파일의 `pre_build` 섹션에 추가합니다. 이러한 명령을 사용하면 빌드 환경에서 현지화 설정에 미국 영어 UTF-8 형식을 사용하므로 미국 외 언어를 포함하는 파일 CodeBuild 이름과 호환성이 더 높습니다. 영어 문자.

Ubuntu 기반의 빌드 환경:

```
pre_build:
  commands:
    - export LC_ALL="en_US.UTF-8"
    - locale-gen en_US en_US.UTF-8
    - dpkg-reconfigure locales
```

Amazon Linux 기반의 빌드 환경:

```
pre_build:
  commands:
    - export LC_ALL="en_US.utf8"
```

Amazon EC2 Parameter Store에서 파라미터를 가져올 때 빌드가 실패할 수 있음

문제: 빌드가 Amazon EC2 Parameter Store에 저장된 하나 이상의 파라미터 값을 가져오려고 하면 `Parameter does not exist`라는 오류가 표시되면서 `DOWNLOAD_SOURCE` 단계에서 빌드가 실패합니다.

가능한 원인: 빌드 프로젝트가 사용하는 서비스 역할에 작업을 호출할 권한이 없거나, 빌드 프로젝트가 `ssm:GetParameters` 작업을 생성하여 AWS CodeBuild 호출을 허용하는 서비스 역할을 사용하지만 매개 변수의 이름이 로 `/CodeBuild/` 시작되지 않습니다. `ssm:GetParameters`

권장 솔루션:

- 에서 서비스 역할을 생성하지 않은 경우 `ssm:GetParameters` 작업을 호출할 수 CodeBuild 있도록 해당 정의를 업데이트하세요. CodeBuild 예를 들어 다음 정책 설명은 `ssm:GetParameters` 작업 호출을 허용하여 `/CodeBuild/`로 시작하는 이름을 가진 파라미터를 가져옵니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/CodeBuild/*"
    }
  ]
}
```

- 에서 서비스 역할을 생성한 경우 로 CodeBuild 시작하는 이름이 아닌 다른 이름으로 Amazon EC2 Parameter Store의 파라미터에 액세스할 수 있도록 CodeBuild 해당 정의를 업데이트하십시오. `/CodeBuild/` 예를 들어 다음 정책 설명은 `ssm:GetParameters` 작업 호출을 허용하여 지정된 이름을 가진 파라미터를 가져옵니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/PARAMETER_NAME"
    }
  ]
}
```

CodeBuild 콘솔에서 브랜치 필터에 액세스할 수 없음

문제: AWS CodeBuild 프로젝트를 생성하거나 업데이트할 때 콘솔에서 브랜치 필터 옵션을 사용할 수 없습니다.

가능한 원인: 브랜치 필터 옵션이 사용되지 않습니다. 이 옵션은 CodeBuild에서 새 빌드를 트리거하는 Webhook 이벤트에 대해 더 다양한 제어를 제공하는 Webhook 필터 그룹으로 대체되었습니다.

권장 솔루션: Webhook 필터 도입 이전에 생성된 브랜치 필터를 마이그레이션하려면 정규식 `^refs/heads/`*`branchName`*`$`를 사용하여 `HEAD_REF` 필터를 포함하는 Webhook 필터 그룹을 생성합니다. 예를 들어 브랜치 필터 정규식이 `^branchName$`이었다면 `HEAD_REF` 필터에 삽입하는 업데이트된 정규식은 `^refs/heads/branchName$`입니다. 자세한 내용은 Bitbucket Webhook 이벤트 및 GitHub 웹후크 이벤트 필터링 (콘솔) 섹션을 참조하세요.

빌드 성공 또는 실패 여부를 볼 수 없음

문제: 재시도한 빌드의 성공 또는 실패 여부를 볼 수 없습니다.

가능한 원인: 빌드 상태를 보고하는 옵션이 활성화되어 있지 않습니다.

권장 해결 방법: CodeBuild 프로젝트를 만들거나 업데이트할 때 보고서 빌드 상태를 활성화하세요. 이 옵션은 CodeBuild에게 빌드가 트리거되었을 때 상태를 다시 보고하도록 지시합니다. 자세한 내용을 알아보려면 AWS CodeBuild API 참조의 reportBuildStatus(을)를 참조하세요.

빌드 상태가 소스 공급자에게 보고되지 않음

문제: GitHub 또는 Bitbucket과 같은 소스 공급자에게 빌드 상태 보고를 허용한 후 빌드 상태가 업데이트되지 않습니다.

가능한 원인: 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 없습니다.

권장 솔루션: 소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 설명은 소스 공급자 액세스 섹션을 참조하세요.

Windows Server Core 2019 플랫폼의 기본 이미지를 찾고 선택할 수 없습니다.

문제: Windows Server Core 2019 플랫폼의 기본 이미지를 찾거나 선택할 수 없습니다.

가능한 원인: 이 이미지를 지원하지 않는 AWS 지역을 사용하고 있습니다.

권장 솔루션: Windows Server Core 2019 플랫폼의 기본 이미지가 지원되는 다음 AWS 리전 중 하나를 사용합니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(오레곤)
- 유럽(아일랜드)

buildspec 파일의 앞에 있는 명령을 나중에 있는 명령에서 인식하지 못함

문제: buildspec 파일에 있는 하나 이상의 명령 결과를 동일한 buildspec 파일의 나중에 있는 명령에서 인식하지 못합니다. 예를 들어, 명령에서 로컬 환경 변수를 설정했지만 나중에 명령이 실행될 때 해당 로컬 환경 변수 값을 가져오지 못할 수 있습니다.

가능한 원인: buildspec 파일 버전 0.1에서 AWS CodeBuild 는 빌드 환경에 있는 기본 쉘의 서로 다른 인스턴스에서 각 명령을 실행합니다. 즉, 각 명령이 다른 모든 명령과 독립적으로 실행됩니다. 그러면 기본적으로 이전 명령의 상태에 따라 실행 여부가 결정되는 단일 명령을 실행할 수 없습니다.

권장 솔루션: 이 문제를 해결하는 빌드 사양 버전 0.2를 사용하는 것이 좋습니다. buildspec 버전 0.1을 사용해야 하는 경우, 쉘 명령 결합 연산자(예: Linux의 &&)를 사용하여 여러 명령을 하나의 명령으로 결합하는 것이 좋습니다. 또는 여러 명령을 포함하는 쉘 스크립트를 소스 코드에 포함한 다음, buildspec 파일에서 단일 명령으로 해당 쉘 스크립트를 호출합니다. 자세한 내용은 빌드 환경의 쉘 및 명령 및 빌드 환경의 환경 변수 섹션을 참조하세요.

오류: 캐시를 다운로드하려고 할 때 “Access denied”라는 메시지가 표시됨

문제: 캐시가 활성화된 빌드 프로젝트에서 캐시를 다운로드하려고 하면 `Access denied` 오류가 표시됩니다.

가능한 원인:

- 방금 빌드 프로젝트의 일부로 캐싱을 구성했습니다.

- 최근 `InvalidateProjectCache` API를 통해 캐시가 무효화되었습니다.
- 에서 사용하는 서비스 역할에는 캐시를 보관하는 S3 버킷에 `s3:PutObject` 대한 CodeBuild 권한이 `s3:GetObject` 없습니다.

권장 솔루션: 처음으로 사용하는 경우 일반적으로 캐시 구성을 업데이트한 직후에 확인할 수 있습니다. 이러한 오류가 계속되는 경우 서비스 역할에 캐시를 보유하고 있는 S3 버킷에 대한 `s3:GetObject` 및 `s3:PutObject` 권한이 있는지 확인해야 합니다. 자세한 내용을 알아보려면 Amazon S3 개발자 안내서의 S3 권한 지정을 참조하세요.

오류: 사용자 지정 빌드 이미지를 사용할 때 "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE"라는 메시지가 표시됨

문제: 사용자 지정 빌드 이미지를 사용하는 빌드를 실행하려고 할 때 `BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE`라는 오류가 표시되며 빌드가 실패합니다.

가능한 원인: 빌드 이미지의 압축되지 않은 전체 크기는 빌드 환경 컴퓨팅 유형의 사용 가능한 디스크 공간보다 큽니다. 빌드 이미지의 크기를 확인하려면 Docker를 사용하여 *`docker images REPOSITORY:TAG`* 명령을 실행합니다. 컴퓨팅 유형별로 사용 가능한 디스크 공간 목록은 빌드 환경 컴퓨팅 모드 및 유형 단원을 참조하십시오.

> 권장 솔루션: 사용 가능한 디스크 공간이 많은 더 큰 컴퓨팅 유형을 사용하거나 사용자 지정 빌드 이미지의 크기를 줄입니다.

가능한 원인: Amazon Elastic 컨테이너 레지스트리 (Amazon ECR) 에서 빌드 이미지를 가져올 AWS CodeBuild 권한이 없습니다.

> 권장 솔루션: 사용자 지정 빌드 이미지를 빌드 환경으로 가져올 CodeBuild 수 있도록 Amazon ECR의 리포지토리 권한을 업데이트하십시오. 자세한 내용은 Amazon ECR 샘플 섹션을 참조하십시오.

가능한 원인: 요청하신 Amazon ECR 이미지는 AWS 계정에서 사용 중인 AWS 지역에서 사용할 수 없습니다.

> 권장 해결 방법: AWS 계정에서 사용하는 것과 동일한 AWS 지역에 있는 Amazon ECR 이미지를 사용하십시오.

가능한 원인: 퍼블릭 인터넷 액세스가 불가능한 VPC에서 프라이빗 레지스트리를 사용하고 있습니다. CodeBuild VPC의 프라이빗 IP 주소에서는 이미지를 가져올 수 없습니다. 자세한 설명은 샘플이 포함된 AWS Secrets Manager 사설 레지스트리 CodeBuild 섹션을 참조하세요.

권장 솔루션: VPC에서 프라이빗 레지스트리를 사용하는 경우 VPC가 퍼블릭 인터넷에 액세스할 수 있는지 확인합니다.

가능한 원인: 오류 메시지에 "toomanyrequests“이 포함되어 있고 Docker Hub에서 이미지를 가져온 경우 이 오류는 Docker Hub 풀 제한에 도달했음을 의미합니다.

권장 솔루션: Docker Hub 프라이빗 레지스트리를 사용하거나 Amazon ECR에서 이미지를 가져옵니다. 프라이빗 레지스트리 사용에 대한 자세한 내용은 샘플이 포함된 AWS Secrets Manager 사설 레지스트리 CodeBuild 섹션을 참조하세요. Amazon S3 사용에 대한 자세한 내용은 에 대한 아마존 ECR 샘플 CodeBuild 섹션을 참조하세요.

오류: “빌드를 완료하기 전에 빌드 컨테이너가 고장난 것으로 확인되었습니다. 빌드 컨테이너가 메모리가 부족하거나 Docker 이미지가 지원되지 않아 종료되었습니다. ErrorCode: 500"

문제: 에서 AWS CodeBuild Microsoft Windows 또는 Linux 컨테이너를 사용하려고 하면 프로비저닝 단계에서 이 오류가 발생합니다.

가능한 원인:

- 에서는 컨테이너 OS 버전을 지원하지 CodeBuild 않습니다.
- HTTP_PROXY, HTTPS_PROXY 또는 둘 다 컨테이너에서 지정됩니다.

권장 솔루션:

- Microsoft Windows의 경우, microsoft/windowsservercore:10.0.x 버전의 컨테이너 OS를 설치한 Windows 컨테이너를 사용합니다(예: microsoft/windowsservercore:10.0.14393.2125).
- Linux의 경우, 도커 이미지에서 HTTP_PROXY 및 HTTPS_PROXY 설정을 지우거나 빌드 프로젝트에서 VPC 구성을 지정합니다.

오류: 빌드 실행 시 "도커 데몬에 연결할 수 없음"

문제: 빌드에 실패하여 빌드 로그에 `Cannot connect to the Docker daemon at unix:/var/run/docker.sock. Is the docker daemon running?`과 유사한 오류가 표시됩니다.

가능한 원인: 권한이 있는 모드에서 빌드를 실행하지 않았습니다.

권장 해결 방법: 이 오류를 해결하려면 다음 지침에 따라 권한 모드를 활성화하고 buildspec을 업데이트해야 합니다.

빌드를 권한 모드에서 실행하려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/codebuild/ 에서 CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택한 다음 빌드 프로젝트를 선택합니다.
3. Edit(편집)에서 Environment(환경)을 선택합니다.
4. 추가 구성을 선택합니다.
5. Docker 이미지를 빌드하거나 빌드에 높은 권한을 부여하려면 Privileged에서 이 플래그 활성화를 선택합니다. .
6. Update environment(환경 업데이트)를 선택합니다.
7. Start build(빌드 시작)을 선택하여 빌드를 다시 시도합니다.

또한 컨테이너 내에서 Docker 데몬을 시작해야 합니다. 빌드스펙의 `install` 단계는 다음과 비슷할 수 있습니다.

```
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"
```

buildspec 파일에 참조된 OverlayFS 스토리지 드라이버에 대한 자세한 내용은 도커 웹사이트의 OverlayFS 스토리지 드라이버 사용을 참조하십시오.

> **Note**
>
> 기본 운영 체제가 Alpine Linux인 경우 `buildspec.yml`에서 `-t` 인수를 `timeout`에 추가합니다.

```
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

를 사용하여 Docker 이미지를 빌드하고 실행하는 방법에 대한 자세한 내용은 을 참조하십시오. AWS CodeBuildDocker의 사용자 지정 이미지 샘플은 다음과 같습니다. CodeBuild

오류: 빌드 CodeBuild 프로젝트를 만들거나 업데이트할 때 AssumeRole “sts:"를 수행할 권한이 없습니다.

문제: 빌드 프로젝트를 생성하거나 업데이트하려고 하면 `Code:InvalidInputException, Message:CodeBuild is not authorized to perform: sts:AssumeRole on arn:aws:iam::account-ID:role/service-role-name` 오류가 표시됩니다.

가능한 원인:

- 빌드 프로젝트를 만들거나 업데이트하려는 AWS 지역에서 AWS Security Token Service (AWS STS) 가 비활성화되었습니다.
- 빌드 프로젝트와 관련된 AWS CodeBuild 서비스 역할이 없거나 신뢰할 수 있는 충분한 권한이 없습니다. CodeBuild

권장 솔루션:

- 빌드 프로젝트를 만들거나 업데이트하려는 AWS 지역에서 AWS STS 이 활성화되어 있는지 확인하세요. 자세한 내용은 IAM 사용 설명서의 AWS 리전 활성화 및 비활성화를 AWS STS 참조하십시오.
- 대상 CodeBuild 서비스 역할이 계정에 존재하는지 확인하십시오. AWS 콘솔을 사용하고 있지 않은 경우, 빌드 프로젝트를 생성하거나 업데이트할 때 서비스 역할의 Amazon 리소스 이름(ARN)을 잘못 입력하지 않았는지 확인합니다.
- 대상 CodeBuild 서비스 역할에 신뢰할 수 있는 충분한 권한이 있는지 확인하세요 CodeBuild. 자세한 내용은 CodeBuild 서비스 역할 생성 섹션의 신뢰 관계 정책 설명을 참조하십시오.

오류: “호출 오류 GetBucketAcl: 버킷 소유자가 변경되었거나 서비스 역할에 더 이상 s3를 호출할 권한이 없습니다.GetBucketAcl”

문제: 빌드를 실행하면 S3 버킷 및 `GetBucketAcl` 권한 소유자 변동에 관한 오류가 표시됩니다.

가능한 원인: `s3:GetBucketAcl` 및 `s3:GetBucketLocation` 권한을 IAM 역할에 추가했습니다. 이러한 권한은 프로젝트의 S3 버킷을 보호하여 사용자만 액세스할 수 있게 합니다. 이러한 권한을 추가한 후에 S3 버킷 소유자가 변경되었습니다.

권장 솔루션: 사용자가 S3 버킷 소유자인지 확인한 후에 IAM 역할에 다시 권한을 추가합니다. 자세한 설명은 S3 버킷에 대한 보안 액세스 섹션을 참조하세요.

오류: 빌드를 실행할 때 "Failed to upload artifacts: Invalid arn"이라는 메시지가 표시됨

문제: 빌드를 실행하면 `Failed to upload artifacts: Invalid arn` 오류가 표시되면서 `UPLOAD_ARTIFACTS` 빌드 단계가 실패합니다.

가능한 원인: S3 출력 버킷 (빌드의 출력을 AWS CodeBuild 저장하는 버킷) 이 CodeBuild 빌드 프로젝트와 다른 AWS 지역에 있습니다.

권장 해결 방법: 빌드 프로젝트와 동일한 AWS 리전에 있는 출력 버킷을 가리키도록 빌드 프로젝트 설정을 업데이트하세요.

오류: "Git Clone Failed: unable to access **`'your-repository-URL'`**: SSL certificate problem: self signed certificate"

문제: 빌드 프로젝트를 실행하려고 하면 이 오류가 표시되면서 빌드가 실패합니다.

가능한 원인: 소스 리포지토리에 자체 서명된 인증서가 있지만 빌드 프로젝트의 일부로 S3 버킷에서 인증서를 설치하도록 선택하지 않은 것입니다.

권장 솔루션:

- 프로젝트를 편집합니다. 인증서에서 Install certificate from S3(S3에서 인증서 설치)를 선택합니다. Bucket of certificate(인증서 버킷)에 SSL 인증서가 저장된 S3 버킷을 선택합니다. 인증서의 객체 키에 S3 객체 키의 이름을 입력합니다.
- 프로젝트를 편집합니다. GitHub Enterprise Server 프로젝트 리포지토리에 연결하는 동안 SSL 경고를 무시하려면 비보안 SSL을 선택합니다.

> **Note**
>
> [Insecure SSL]은 테스트 용도로만 사용하는 것이 좋습니다. 프로덕션 환경에 사용하면 안 됩니다.

오류: 빌드를 실행할 때 "The bucket you are attempting to access must be addressed using the specified endpoint..."라는 메시지가 표시됨

문제: 빌드를 실행하면 `The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint` 오류가 표시되면서 `DOWNLOAD_SOURCE` 빌드 단계가 실패합니다.

가능한 원인: 사전 빌드된 소스 코드가 S3 버킷에 저장되어 있고 해당 버킷이 AWS CodeBuild 빌드 프로젝트와 다른 AWS 지역에 있습니다.

권장 솔루션: 사전 작성된 소스 코드가 포함되어 있는 버킷을 가리키도록 빌드 프로젝트 설정을 업데이트합니다. 버킷이 빌드 프로젝트와 같은 AWS 리전에 있는지 확인하세요.

오류: "이 빌드 이미지는 하나 이상의 런타임 버전을 선택해야 합니다."

문제: 빌드를 실행하면 `YAML_FILE_ERROR: This build image requires selecting at least one runtime version` 오류가 표시되면서 `DOWNLOAD_SOURCE` 빌드 단계가 실패합니다.

가능한 원인: 빌드에서 Amazon Linux 2(AL2) 표준 이미지 버전 1.0 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하고, buildspec 파일에 런타임이 지정되어 있지 않습니다.

권장 해결 방법: `aws/codebuild/standard:2.0` CodeBuild 관리 이미지를 사용하는 경우 buildspec 파일의 `runtime-versions` 섹션에서 런타임 버전을 지정해야 합니다. 예를 들어, PHP를 사용하는 프로젝트에는 다음 buildspec 파일을 사용해야 합니다.

```
version: 0.2

phases:
  install:
```

```
    runtime-versions:
        php: 7.3
  build:
    commands:
      - php --version
artifacts:
  files:
    -  README.md
```

> **Note**
>
> `runtime-versions` 섹션을 지정하고 Ubuntu 표준 이미지 2.0 이상 또는 Amazon Linux 2(AL2) 표준 이미지 1.0 이상 외의 이미지를 사용하는 경우, 빌드에서 “`Skipping install of runtimes. Runtime version selection is not supported by this build image`” 경고가 발생합니다.

자세한 설명은 Specify runtime versions in the buildspec file 섹션을 참조하세요.

오류: 빌드 대기열의 빌드가 실패할 때 "QUEUED: INSUFFICIENT_SUBNET"이라는 메시지가 표시됨

문제: `QUEUED: INSUFFICIENT_SUBNET`과 유사한 오류로 빌드 대기열의 빌드가 실패합니다.

가능한 원인: VPC에 지정된 IPv4 CIDR 블록이 예약된 IP 주소를 사용합니다. 각 서브넷 CIDR 블록에서 처음 4개의 IP 주소와 마지막 IP 주소는 사용자가 사용할 수 없으므로 인스턴스에 할당할 수 없습니다. 예를 들어 `10.0.0.0/24` CIDR 블록의 서브넷에서는 다음 5개 IP 주소가 예약되어 있습니다.

- `10.0.0.0:`: 네트워크 주소
- `10.0.0.1`: VPC AWS 라우터용으로 예약합니다.
- `10.0.0.2`: 예약자. AWS DNS 서버의 IP 주소는 항상 기본 VPC 네트워크 범위에 2를 더한 주소입니다. 그러나 각 서브넷 범위에 2를 더한 주소도 예약합니다. CIDR 블록이 여러 개인 VPC의 경우, DNS 서버의 IP 주소가 기본 CIDR에 위치합니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon DNS 서버를 참조하세요.
- `10.0.0.3`: 나중에 AWS 사용할 수 있도록 예약했습니다.
- `10.0.0.255`: 네트워크 브로드캐스트 주소. VPC에서는 브로드캐스트를 지원하지 않습니다. 이 주소는 예약되어 있습니다.

권장 솔루션: VPC가 예약된 IP 주소를 사용하는지 확인합니다. 예약된 IP 주소를 예약되지 않은 IP 주소로 바꿉니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷 크기 조정을 참조하세요.

오류: “캐시를 다운로드할 수 없음: RequestError: 전송 요청 실패 원인: x509: 시스템 루트를 로드하지 못했고 루트가 제공되지 않았습니다.”

문제: 빌드 프로젝트를 실행하려고 하면 이 오류가 표시되면서 빌드가 실패합니다.

가능한 원인: 캐시를 빌드 프로젝트의 일부로 구성했으며 만료된 루트 인증서가 포함된 기존 도커 이미지를 사용하고 있습니다.

권장 해결 방법: 프로젝트에서 사용 중인 Docker 이미지를 업데이트하세요. AWS CodeBuild 자세한 설명은 Docker 이미지 제공: CodeBuild 섹션을 참조하세요.

오류: “S3에서 인증서를 다운로드할 수 없습니다. AccessDenied

문제: 빌드 프로젝트를 실행하려고 하면 이 오류가 표시되면서 빌드가 실패합니다.

가능한 원인:

- 인증서에 대해 잘못된 S3 버킷을 선택한 것입니다.
- 인증서에 대해 잘못된 객체 키를 입력한 것입니다.

권장 솔루션:

- 프로젝트를 편집합니다. Bucket of certificate(인증서 버킷)에 SSL 인증서가 저장된 S3 버킷을 선택합니다.
- 프로젝트를 편집합니다. 인증서의 객체 키에 S3 객체 키의 이름을 입력합니다.

오류: "Unable to locate credentials"

문제: 를 실행하거나 AWS SDK를 사용하거나 빌드의 일부로 다른 유사한 구성 요소를 호출하려고 하면 AWS CLI, AWS SDK 또는 구성 요소와 직접 관련된 빌드 오류가 발생합니다. AWS CLI예를 들어 `Unable to locate credentials`와 같은 빌드 오류가 발생할 수 있습니다.

가능한 원인:

- 빌드 환경의 AWS CLI, AWS SDK 또는 구성 요소 버전이 호환되지 않습니다. AWS CodeBuild
- Docker를 사용하는 빌드 환경 내에서 Docker 컨테이너를 실행하고 있는데 컨테이너는 기본적으로 자격 증명에 액세스할 수 없습니다. AWS

권장 솔루션:

- 빌드 환경에 다음 버전 이상의 AWS SDK 또는 구성 요소가 AWS CLI있는지 확인하세요.
  - AWS CLI: 1.10.47
  - AWS C++용 SDK: 0.2.19
  - AWS Go용 SDK: 1.2.5
  - AWS 자바용 SDK: 1.11.16
  - AWS SDK 버전: 2.4.7 JavaScript
  - AWS PHP용 SDK: 3.18.28
  - AWS 파이썬용 SDK (Boto3): 1.4.0
  - AWS 루비용 SDK: 2.3.22
  - Botocore: 1.4.37
  - CoreCLR: 3.2.6-beta
  - Node.js: 2.4.7
- 빌드 환경에서 Docker 컨테이너를 실행해야 하고 컨테이너에 AWS 자격 증명이 필요한 경우 빌드 환경의 자격 증명을 컨테이너로 전달해야 합니다. buildspec 파일에서 다음과 같은 도커 `run` 명령을 포함합니다. 이 예에서는 `aws s3 ls` 명령을 사용하여 사용 가능한 S3 버킷을 나열합니다. `-e`옵션은 컨테이너가 AWS 자격 증명에 액세스하는 데 필요한 환경 변수를 통과합니다.

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag aws s3 ls
```

- Docker 이미지를 빌드하고 있고 빌드에 AWS 자격 증명이 필요한 경우 (예: Amazon S3에서 파일 다운로드), 다음과 같이 빌드 환경의 자격 증명을 Docker 빌드 프로세스로 전달해야 합니다.

  1. 도커 이미지용 소스 코드의 Dockerfile에서 다음 `ARG` 지침을 지정합니다.

```
ARG AWS_DEFAULT_REGION
ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

  2. buildspec 파일에서 다음과 같은 도커 `build` 명령을 포함합니다. `--build-arg`옵션은 Docker 빌드 프로세스가 자격 증명에 액세스하는 데 필요한 환경 변수를 설정합니다. AWS

```
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 프록시 서버에서 실행할 CodeBuild 때 시간 초과 오류가 발생했습니다.

문제: 다음 중 하나와 비슷한 `RequestError` 오류가 발생합니다.

- `RequestError: send request failed caused by: Post https://logs.<your-region>.amazonaws.com/: dial tcp 52.46.158.105:443: i/o timeout` CloudWatch 로그에서.
- Amazon S3의 `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`

가능한 원인:

- `ssl-bump`가 제대로 구성되지 않았습니다.
- 조직의 보안 정책이 사용자가 `ssl_bump`를 사용하는 것을 허용하지 않습니다.
- buildspec 파일에 `proxy` 요소를 사용하여 지정한 프록시 설정이 없습니다.

권장 솔루션:

- `ssl-bump`가 올바로 구성되었는지 확인하십시오. Squid를 프록시 서버로 사용하는 경우 Squid를 명시적 프록시 서버로 구성 단원을 참조하십시오.
- Amazon S3 및 CloudWatch Logs에 프라이빗 엔드포인트를 사용하려면 다음 단계를 따르십시오.

  1. 프라이빗 서브넷 라우팅 테이블에서 이전에 추가한 인터넷으로 향하는 트래픽을 프록시 서버로 라우팅하는 규칙을 제거합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC에서 서브넷 만들기를 참조하세요.
  2. 프라이빗 Amazon S3 엔드포인트와 CloudWatch 로그 엔드포인트를 생성하고 이를 Amazon VPC의 프라이빗 서브넷과 연결합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트 서비스를 참조하세요.

3. Amazon VPC에서 프라이빗 DNS 이름 활성화가 선택되었는지 확인합니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하십시오.

- 명시적 프록시 서버에 ssl-bump를 사용하지 않을 경우 proxy 요소를 사용하여 buildspec 파일에 프록시 구성을 추가하십시오. 자세한 내용은 명시적 프록시 서버에서 CodeBuild 실행 및 buildspec 구문 섹션을 참조하세요.

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
```

빌드 이미지에 있어야 하는 Bourne 쉘(sh)

문제: 에서 제공하지 않은 빌드 이미지를 사용하고 있는데 이 메시지와 함께 빌드가 실패합니다. AWS CodeBuildBuild container found dead before completing the build

가능한 원인: Bourne shell (sh) 이 빌드 이미지에 포함되어 있지 않습니다. CodeBuild 빌드 명령과 스크립트를 sh 실행해야 합니다.

권장 솔루션: sh가 빌드 이미지에 없는 경우, 이미지를 사용하는 더 많은 빌드를 시작하기 전에 먼저 포함해야 합니다. (CodeBuild 이미 빌드 이미지에 포함되어 sh 있습니다.)

경고: 빌드 실행 시 “런타임 설치를 건너뜁니다. 런타임 버전 선택은 이 빌드 이미지에서 지원되지 않습니다.”가 표시됨

문제: 빌드를 실행하면 빌드 로그에 이 경고가 포함되어 있습니다.

가능한 원인: 빌드에서 Amazon Linux 2(AL2) 표준 이미지 버전 1.0 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하지 않고, buildspec 파일의 runtime-versions 섹션에 런타임이 지정되어 있습니다.

권장 솔루션: buildspec 파일에 runtime-versions 섹션이 포함되지 않게 하십시오. runtime-versions 섹션은 Amazon Linux 2(AL2) 표준 이미지 버전 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하는 경우에만 필요합니다.

오류: CodeBuild 콘솔을 열 때 " JobWorker ID를 확인할 수 없습니다."

문제: CodeBuild 콘솔을 열면 " JobWorker ID를 확인할 수 없음" 오류 메시지가 표시됩니다.

가능한 원인: 콘솔 액세스에 사용되는 IAM 역할에 `jobId` 키가 있는 태그가 있습니다. 이 태그 키는 CodeBuild 예약용이며 있는 경우 이 오류가 발생합니다.

권장 솔루션: `jobId` 키가 있는 사용자 지정 IAM 역할 태그를 다른 키를 사용하도록 변경합니다(예: `jobIdentifier`).

빌드를 시작하지 못함

문제: 빌드를 시작할 때 빌드를 시작하지 못함 오류 메시지가 나타납니다.

가능한 원인: 동시 빌드 수에 도달했습니다.

권장 솔루션: 다른 빌드가 완료될 때까지 기다리거나 프로젝트의 동시 빌드 제한을 늘린 다음, 빌드를 다시 시작하세요. 자세한 설명은 프로젝트 구성 섹션을 참조하세요.

로컬에 캐시된 빌드의 GitHub 메타데이터에 액세스

문제: 캐시된 빌드의 .git 디렉터리가 디렉터리가 아닌 텍스트 파일인 경우가 있습니다.

가능한 원인: 빌드에 로컬 소스 캐싱이 활성화되면 디렉터리에 gitlink가 CodeBuild 생성됩니다. `.git` 즉, `.git` 디렉터리는 실제로 디렉터리 경로가 포함된 텍스트 파일입니다.

권장 솔루션: 모든 경우에 다음 명령을 사용하여 Git 메타데이터 디렉터리를 가져오세요. 이 명령은 다음과 같은 `.git` 형식에 관계없이 작동합니다.

```
git rev-parse --git-dir
```

AccessDenied: 보고서 그룹의 버킷 소유자가 S3 버킷 소유자와 일치하지 않습니다...

문제: Amazon S3 버킷에 테스트 데이터를 업로드할 때 테스트 데이터를 버킷에 쓸 수 없습니다. CodeBuild

가능한 원인:

- 보고서 그룹 버킷 소유자로 지정된 계정이 Amazon S3 버킷 소유자와 일치하지 않습니다.
- 서비스 역할에는 버킷에 대한 쓰기 권한이 없습니다.

권장 솔루션:

- Amazon S3 버킷 소유자와 일치하도록 보고서 그룹 버킷 소유자를 변경합니다.
- Amazon S3 버킷에 대한 쓰기 액세스 권한을 허용하도록 서비스 역할을 수정합니다.

AWS CodeBuild에 대한 할당량

다음 표에 AWS CodeBuild와 관련된 현재 할당량이 나와 있습니다. 이 할당량은 달리 지정되지 않는 한, 각 AWS 계정의 지원되는 AWS 리전 하나에 대해 적용됩니다.

서비스 할당량

다음 표에는 AWS CodeBuild 서비스에 대한 기본 할당량이 나와 있습니다.

명칭	기본값	조정 가능	Description
프로젝트별 관련 태그	지원되는 각 리전: 50	아니요	빌드 프로젝트와 연결할 수 있는 최대 태그 수
빌드 프로젝트	지원되는 각 리전: 5,000	예	최대 빌드 프로젝트 수
빌드 제한 시간(분)	지원되는 각 리전: 480	아니요	최대 빌드 제한 시간(분)
빌드 정보에 대한 동시 요청	지원되는 각 리전: 100	아니요	AWS CLI 또는 AWS SDK를 사용하여 한 번에 정보를 요청할 수 있는 빌드의 최대 수
빌드 프로젝트 정보에 대한 동시 요청	지원되는 각 리전: 100	아니요	AWS CLI 또는 AWS SDK를 사용하여 한 번에 정보를 요청할 수 있는 빌드 프로젝트의 최대 수

명칭	기본값	조정 가능	Description
ARM Lambda/10GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/10GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/1GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/1GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/2GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/2GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/4GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/4GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/8GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/8GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM/Large 환경에서 동시에 실행되는 빌드의 최대 수
ARM/Small 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM/Small 환경에서 동시에 실행되는 빌드의 최대 수
Linux GPU Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 0	예	Linux GPU/Large 환경에서 동시에 실행되는 빌드의 최대 수

명칭	기본값	조정 가능	Description
Linux GPU Small 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 0	예	Linux GPU/Small 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/10GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/10GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/1GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/1GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/2GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/2GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/4GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/4GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/8GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/8GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux/2XLarge 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 0	예	Linux/2XLarge 환경에서 동시에 실행되는 빌드의 최대 수
Linux/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux/Large 환경에서 동시에 실행되는 빌드의 최대 수

명칭	기본값	조정 가능	Description
Linux/Medium 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux/Medium 환경에서 동시에 실행되는 빌드의 최대 수
Linux/Small 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux/Small 환경에서 동시에 실행되는 빌드의 최대 수
Linux/XLarge 환경 플릿에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux/XLarge 환경에서 동시에 실행되는 빌드의 최대 수
Windows Server 2019/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows Server 2019/Large 환경에서 동시에 실행되는 빌드의 최대 수
Windows Server 2019/Medium 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows Server 2019/Medium 환경에서 동시에 실행되는 빌드의 최대 수
Windows/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows/Large 환경에서 동시에 실행되는 빌드의 최대 수
Windows/Medium 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows/Medium 환경에서 동시에 실행되는 빌드의 최대 수
빌드 제한 시간의 최소 기간(분)	지원되는 각 리전: 5	아니요	최소 빌드 제한 시간(분)

명칭	기본값	조정 가능	Description
VPC 구성의 보안 그룹	지원되는 각 리전: 5	아니요	VPC 구성에 사용할 수 있는 보안 그룹
VPC 구성의 서브넷	지원되는 각 리전: 16	아니요	VPC 구성에 사용할 수 있는 서브넷

> **Note**
>
> 내부 지표에 따라 동시 실행 빌드의 기본 할당량이 결정됩니다.

최대 동시 실행 빌드 수 할당량은 컴퓨팅 유형에 따라 다릅니다. 일부 플랫폼 및 컴퓨팅 유형의 경우 기본값은 20입니다. 더 높은 동시 빌드 할당량을 요청하거나 "계정에 X개 이상의 활성 빌드를 가질 수 없음" 오류가 발생하는 경우 위 링크를 사용하여 요청을 수행하세요. 요금에 대한 자세한 내용은 AWS CodeBuild 요금을 참조하세요.

기타 제한

빌드 프로젝트

Resource	기본값
빌드 프로젝트 설명에 허용되는 문자	모두
빌드 프로젝트 이름에 허용되는 문자	글자(`A-Z` 및 `a-z`), 숫자(`0-9`) 및 특수 문자(`-` 및 `_`)
빌드 프로젝트 이름의 길이	2~255자(경계값 포함)
빌드 프로젝트 설명의 최대 길이	255자

Resource	기본값
프로젝트에 추가할 수 있는 최대 보고서 수	5
관련된 모든 빌드의 빌드 제한 시간으로 빌드 프로젝트에 지정할 수 있는 시간(분)	5~480(8시간)

빌드

Resource	기본값
빌드 기록이 유지되는 최대 시간	1년
단일 빌드의 빌드 제한 시간으로 지정할 수 있는 시간(분)	5~480(8시간)

컴퓨팅 플릿

Resource	기본값
동시 컴퓨팅 플릿 수	10
ARM/Small 환경 플릿에서 동시에 실행되는 인스턴스	1
ARM/Large 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/Small 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/Medium 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/Large 환경 플릿에서 동시에 실행되는 인스턴스	1

Resource	기본값
Linux/XLarge 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/2XLarge 환경 플릿에서 동시에 실행되는 인스턴스	0
Linux GPU/Small 환경 플릿에서 동시에 실행되는 인스턴스	0
Linux GPU/Large 환경 플릿에서 동시에 실행되는 인스턴스	0
Windows Server 2019/Medium 환경 플릿에서 동시에 실행되는 인스턴스	1
Windows Server 2019/Large 환경 플릿에서 동시에 실행되는 인스턴스	1
Windows Server 2022/Medium 환경 플릿에서 동시에 실행되는 인스턴스	1
Windows Server 2022/Large 환경 플릿에서 동시에 실행되는 인스턴스	1

보고서

Resource	기본값
테스트 보고서를 만든 후 사용할 수 있는 최대 기간	30일
테스트 사례 메시지의 최대 길이	5,000자
테스트 사례 이름의 최대 길이	1,000자
AWS 계정당 최대 보고서 그룹 수	5,000

Resource	기본값
보고서당 최대 테스트 케이스 수	500

Tags

태그 제한은 CodeBuild 빌드 프로젝트 및 CodeBuild 보고서 그룹 리소스의 태그에 적용됩니다.

Resource	기본값
리소스 태그 키 이름	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허용되는 문자 조합(1 - 127 문자). 허용되는 문자는 `+ - = . _ : / @`입니다. 태그 키 이름은 고유해야 하며 각 키는 하나의 값만 가질 수 있습니다. 태그 키 이름에는 다음 사항이 적용됩니다. • `aws:`로 시작할 수 없음 • 공백으로만 이루어짐 • 공백으로 끝남 • 이모티콘 또는 다음 문자를 포함할 수 없음: `? ^ * [ \ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;`
리소스 태그 값	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허용되는 문자 조합(0 - 255 문자). 허용되는 문자는 `+ - = . _ : / @`입니다. 키는 하나의 값만 가질 수 있지만 많은 키가 동일한 값을 가질 수 있습니다. 태그 키 값에는 이모티콘 또는 다음 문자를 포함할 수 없습니다. `? ^ * [ \ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;`

윈도우에 AWS CodeBuild 대한 타사 알림

Windows CodeBuild 빌드에 사용하는 경우 일부 타사 패키지 및 모듈을 사용하여 빌드된 응용 프로그램을 Microsoft Windows 운영 체제에서 실행하고 일부 타사 제품과 상호 운용할 수 있도록 하는 옵션이 있습니다. 다음 목록에는 지정된 타사 패키지 및 모듈의 사용에 적용되는 타사 법률 약관이 포함되어 있습니다.

주제

1) 기본 도커 이미지—windowsservercore

(라이선스 약관은 https://hub.docker.com/_/ microsoft-windows-servercore 에서 확인할 수 있습니다.)

라이선스: 이 Windows 컨테이너용 컨테이너 OS 이미지를 요청하고 사용할 경우 다음 추가 프로그램 라이선스 약관을 인정하고 이해하며 동의하는 것으로 간주됩니다.

MICROSOFT 소프트웨어 추가 프로그램 라이선스 약관

컨테이너 OS 이미지

Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)(“우리”, “당사” 또는 “Microsoft”로 지칭)에서 이 컨테이너 OS 이미지 프로그램의 사용을 허가합니다. 귀하는 호스트 소프트웨어의 컨테이너 기능 실행을 지원하는 용도로만 이 추가 프로그램을 기본 호스트 운영 체제 소프트웨어(“호스트 소프트웨어”)와 함께 사용할 수 있습니다. 호스트 소프트웨어 라이선스 약관이 이 추가 프로

그램의 사용에도 적용됩니다. 호스트 소프트웨어에 대한 라이선스가 없는 경우 이 소프트웨어를 사용할 수 없습니다. 유효한 라이선스가 부여된 각 호스트 소프트웨어 사본과 함께 이 추가 프로그램을 사용할 수 있습니다.

추가 라이선스 요구 사항 및/또는 사용 권한

이전 단락에 명시된 대로 추가 프로그램을 사용하면 특정 추가 프로그램 구성 요소를 포함하는 컨테이너 이미지가 생성되거나 수정될 수 있습니다. 명확히 설명하자면 컨테이너 이미지는 가상 머신 또는 가상 어플라이언스 이미지와는 별개로 구분됩니다. 본 라이선스 조건에 따라 당사는 다음 조건에서 해당 추가 프로그램 구성 요소를 재배포할 수 있는 제한된 권리를 사용자에게 부여합니다.

(i) 추가 프로그램 구성 요소는 컨테이너 이미지에 사용된 용도로만 사용할 수 있으며 컨테이너 이미지의 일부로만 사용할 수 있습니다.

(ii) 추가 프로그램과 실질적으로 분리되고 별개인 중요한 주요 기능이 컨테이너 이미지에 있는 한, 컨테이너 이미지에 이러한 추가 프로그램 구성 요소를 사용할 수 있습니다.

(iii) 귀하는 최종 사용자가 추가 프로그램 구성 요소를 사용할 수 있도록 하기 위해 컨테이너 이미지에 이러한 라이선스 약관(또는 당사 또는 호스팅 업체가 요구하는 유사한 조건)을 포함시키는 데 동의합니다.

당사는 여기에서 명시적으로 부여되지 않은 기타 모든 권리를 보유합니다.

이 추가 프로그램을 사용하면 이 약관에 동의하는 것으로 간주됩니다. 동의하지 않는 경우 이 추가 프로그램을 사용하지 마세요.

이 Windows 컨테이너용 컨테이너 OS 이미지에 대한 추가 라이선스 약관의 일부로 https://www.microsoft.com/en-us/useterms에 제공되는 기본 Windows Server 호스트 소프트웨어 라이선스 약관도 적용됩니다.

2) Windows 기반 도커 이미지—choco

(라이선스 조건은 https://github.com/chocolatey/choco/blob/master/LICENSE 에서 확인할 수 있습니다.)

저작권 2011 - 프레젠트 RealDimensions 소프트웨어, LLC

Apache 라이선스, 버전 2.0(이하 “라이선스”)에 따라 라이선스가 부여됩니다.이 라이선스를 준수하는 경우를 제외하고는 이러한 파일을 사용할 수 없습니다. 라이선스 사본은 다음 사이트에서 구할 수 있습니다.

http://www.apache.org/licenses/LICENSE-2.0

관련 법률에서 요구하거나 서면으로 합의한 경우를 제외하고, 라이선스에 따라 배포된 소프트웨어는 명시적이든 묵시적이든 어떠한 종류의 보증이나 조건 없이 “있는 그대로” 배포됩니다. 라이선스에 따른 구체적인 언어 관리 권한과 제한 사항은 라이선스를 참조하십시오.

3) Windows 기반 도커 이미지—git --버전 2.16.2

(라이선스 약관 참조: https://chocolatey.org/packages/git/2.16.2)

GNU 일반 공개 라이선스, 버전 2에 따라 사용이 허가되어 https://www.gnu.org/licenses/old-licenses/gpl-2.0.html에서 참조할 수 있습니다.

4) 윈도우 기반 도커 이미지— —버전 15.0.26320.2 microsoft-build-tools

(라이선스 약관 참조: https://www.visualstudio.com/license-terms/mt171552/)

MICROSOFT VISUAL STUDIO 2015 EXTENSIONS, VISUAL STUDIO SHELLS 및 C++ 재배포 가능 패키지

본 라이선스 약관은 Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)과 귀하 사이의 계약입니다. 이 조건은 위에 명시된 소프트웨어에 적용됩니다. 본 조건은 추가 조건이 있는 경우를 제외하고 소프트웨어의 모든 Microsoft 서비스 또는 업데이트에도 적용됩니다.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 설치 및 사용 권한. 원하는 수만큼 소프트웨어 사본을 설치하고 사용할 수 있습니다.
2. 특정 구성 요소에 대한 조건.
   a. 유틸리티. 소프트웨어에는 유틸리티 목록(https://docs.microsoft.com/en-us/visualstudio/productinfo/2015-redistribution-vs)의 일부 항목이 포함될 수 있습니다. 소프트웨어에 포함된 경우 해당 항목을 사용자 또는 타사 컴퓨터에 복사하고 설치하여 소프트웨어로 개발한 애플리케이션 및 데이터베이스를 디버그 및 배포할 수 있습니다. 유틸리티는 임시 사용을 위해 설계되었으므로 Microsoft가 나머지 소프트웨어와는 별도로 유틸리티를 패치하거나 업데이트하지 못할 수 있으며 일부 유틸리티는 그 특성상 다른 사용자가 해당 유틸리티가 설치된 컴퓨터에 액세스할 수 있다는 점에 유의하세요. 따라서 애플리케이션 및 데이터베이스의 디버깅 또는 배포를 마친 후에

는 설치한 유틸리티를 모두 삭제해야 합니다. Microsoft는 사용자가 컴퓨터에 설치한 유틸리티의 제3자 사용 또는 액세스에 대해 책임을 지지 않습니다.

b. Microsoft 플랫폼. 소프트웨어에는 마이크로소프트 윈도우, 마이크로소프트 윈도우 서버, 마이크로소프트 SQL 서버, 마이크로소프트 익스체인지, 마이크로소프트 오피스 및 마이크로소프트의 구성 요소가 포함될 수 SharePoint 있습니다. 이러한 구성 요소에는 해당 구성 요소의 설치 디렉터리 또는 소프트웨어와 함께 제공되는 “Licenses” 폴더에 있는 라이선스 약관에 설명된 대로 별도의 계약 및 자체 제품 지원 정책이 적용됩니다.

c. 타사 구성 요소. 소프트웨어에는 소프트웨어와 함께 제공되는 ThirdPartyNotices 파일에 설명된 대로 별도의 법적 고지가 있거나 다른 계약이 적용되는 타사 구성 요소가 포함될 수 있습니다. 이러한 구성 요소가 다른 계약의 적용을 받는 경우에도 아래의 고지 사항, 손해 배상 제한 및 제외 조항이 함께 적용됩니다. 소프트웨어에는 소스 코드 가용성 의무가 있는 오픈 소스 라이선스에 따라 라이선스가 부여된 구성 요소도 포함될 수 있습니다. 해당하는 경우 해당 라이센스의 사본이 파일에 포함됩니다. ThirdPartyNotices 관련 오픈 소스 라이선스에 따라 필요한 경우, Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052의 Source Code Compliance Team으로 $5.00의 우편환 또는 수표를 보내면 Microsoft에서 이 소스 코드를 받을 수 있습니다. 아래 나열된 구성 요소 중 하나 이상의 소스 코드를 결제 메모란에 적어주세요.

- Visual Studio 2015용 원격 도구
- Visual Studio 2015용 독립 실행형 프로파일러
- IntelliTraceCollector 비주얼 스튜디오 2015의 경우,
- Microsoft VC++ 재배포 가능 2015
- Visual Studio 2015용 멀티바이트 MFC 라이브러리
- Microsoft Build Tools 2015
- Feedback Client
- Visual Studio 2015 통합 쉘 또는
- Visual Studio 2015 격리 쉘.

소스 코드의 사본이 http://thirdpartysource.microsoft.com에 제공될 수도 있습니다.

3. 데이터. 소프트웨어에서 사용자 및 사용자의 소프트웨어 사용에 대한 정보를 수집하여 Microsoft에 보낼 수 있습니다. Microsoft는 이 정보를 사용하여 서비스를 제공하고 제품 및 서비스를 개선할 수 있습니다. 제품 설명서에 설명된 대로 이러한 시나리오를 대부분 옵트아웃할 수 있지만 전부는 아닙니다. 또한 소프트웨어에는 애플리케이션 사용자로부터 데이터를 수집할 수 있는 몇 가지 기능이 있습니다. 이러한 기능을 사용하여 애플리케이션에서 데이터를 수집할 수 있게 하는 경우 애플리케이션 사용자에게 적절한 고지를 제공하는 것을 포함하여 관련 법률을 준수해야 합니다. 데이터 수집 및 사용에 대한 자세한 내용은 도움말 설명서 및 https://privacy.microsoft.com/en-us/

privacystatement의 개인 정보 보호 정책에서 확인할 수 있습니다. 소프트웨어를 사용하면 이러한 정책에 동의하는 것으로 간주됩니다.

4. 라이선스 범위. 본 소프트웨어는 판매되는 것이 아니라 그 사용이 허여되는 것입니다. 본 계약은 소프트웨어를 사용할 수 있는 일부 권리만 제공합니다. Microsoft는 기타 모든 권리를 보유합니다. 이러한 제한과 관계없이 관련 법률에서 귀하에게 더 많은 권한을 부여하지 않는 한, 이 계약에서 명시적으로 허용된 대로만 소프트웨어를 사용할 수 있습니다. 이 과정에서 귀하는 특정 방식으로만 사용할 수 있도록 하는 소프트웨어의 모든 기술적 제한 사항을 준수해야 합니다. 다음과 같은 행위는 허용되지 않습니다.
   - 소프트웨어의 모든 기술적 제한 사항을 해결하는 행위
   - 소프트웨어를 리버스 엔지니어링, 디컴파일 또는 디스어셈블하거나 그러한 시도를 하는 행위(소프트웨어에 포함될 수 있는 특정 오픈 소스 구성 요소의 사용을 규제하는 제3자 라이선스 약관에서 요구하는 경우 제외)
   - 소프트웨어에서 Microsoft 또는 해당 공급자의 알림을 제거, 최소화, 차단 또는 수정하는 행위
   - 법률에 위배되는 방식으로 소프트웨어를 사용하는 행위 또는
   - 소프트웨어를 공유, 게시, 대여 또는 임대하거나 다른 사람이 사용할 수 있도록 소프트웨어를 독립 실행형으로 호스팅된 솔루션으로 제공하는 행위
5. 수출 제한. 목적지, 최종 사용자 및 최종 사용에 대한 제한 사항을 포함하여 소프트웨어에 적용되는 모든 국내 및 국제 수출 법률 및 규정을 준수해야 합니다. 수출 제한에 대한 자세한 내용은 aka.ms/export를 참조하세요.
6. 지원 서비스. 이 소프트웨어는 “있는 그대로” 제공되므로 본 소프트웨어에 대한 지원 서비스가 제공되지 않을 수 있습니다.
7. 완전 합의. 본 계약 및 귀하가 이용하는 추가 구성 요소, 업데이트, 인터넷 기반 서비스 및 지원 서비스에 대한 조항은 소프트웨어 및 지원 서비스에 대한 완전 합의입니다.
8. 관련 법률. 미국에서 본 소프트웨어를 구입한 경우 워싱턴주 법이 본 계약의 위반에 대한 해석 및 클레임에 적용되며, 귀하가 거주하는 지역의 법률이 다른 모든 클레임에 적용됩니다. 다른 국가에서 본 소프트웨어를 구입한 경우 해당 국가의 법률이 적용됩니다.
9. 소비자 권리, 지역 또는 국가별 다양성. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 거주하는 국가 또는 시/도의 법률에 따라 소비자 권리를 비롯한 기타 권리를 보유할 수 있습니다. Microsoft와 맺은 관계와 별도로, 귀하는 소프트웨어를 판매한 당사자와 관련해서 권리를 보유할 수도 있습니다. 귀하가 거주하고 있는 국가 또는 시/도의 법에서 권리 변경을 허용하지 않는 경우 본 계약은 이러한 기타 권리를 변경하지 않습니다. 예를 들어 아래 지역 중 하나에서 소프트웨어를 구입했거나 필수 국가 법률이 적용되는 경우 다음 조항이 적용됩니다.
   a. 오스트레일리아. 귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.

b. 캐나다. 캐나다에서 본 소프트웨어를 구입한 경우 자동 업데이트 기능을 끄거나, 인터넷에서 디바이스 연결을 끊거나(단, 인터넷에 다시 연결하면 소프트웨어가 업데이트 확인 및 설치를 다시 시작함), 소프트웨어를 제거하여 업데이트 수신을 중지할 수 있습니다. 제품 설명서(있는 경우)에서 특정 디바이스 또는 소프트웨어의 업데이트를 끄는 방법을 지정할 수도 있습니다.

c. 독일 및 오스트리아.

i. 보증. 적절하게 사용이 허가된 소프트웨어는 대체로 본 소프트웨어와 함께 제공되는 Microsoft 자료에 설명된 대로 작동합니다. 그러나 Microsoft는 사용이 허가된 소프트웨어와 관련하여 어떠한 계약 보장도 제공하지 않습니다.

ii. 책임의 제한. 의도적인 행위, 중과실, 제조물 책임법에 따른 청구 및 개인 또는 신체 부상이나 사망의 경우 Microsoft는 성문법에 따라 책임을 집니다. Microsoft는 앞 조항 (ii)에 따라 이와 같은 중요한 계약 의무를 위반하는 경우에만 약간의 과실에 대해 책임집니다. 해당 의무를 다하면 본 계약의 적절한 이행을 촉진하고, 위반하면 본 계약의 목적이 위태롭게 되며, 준수하면 당사자가 “가장 중요한 의무”라는 의무를 지속해서 신뢰할 수 있습니다. 이 외의 경우에는 약간의 과실에 대해 책임지지 않습니다.

10보증의 부인. 이 소프트웨어는 “있는 그대로” 라이선스가 허가됩니다. 해당 사용으로 인해 발생하는 모든 위험은 귀하의 책임입니다. Microsoft는 어떠한 명시적 보증, 보장 또는 조건도 제시하지 않습니다. 귀하가 거주하는 지역의 법규가 허용하는 범위 내에서 MICROSOFT는 상업성, 특정 목적에의 적합성 및 비침해성과 관련된 묵시적 보증을 배제합니다.

11.손해의 제한 및 배제. MICROSOFT와 공급업체에서 최대 $5.00의 직접적인 손해에 대해서만 보상받을 수 있습니다. 결과적 손해, 이익 손실, 특별, 간접 또는 부수적 손해를 포함한 기타 모든 손해에 대해서는 보상을 받을 수 없습니다. 이러한 제한은 (a) 소프트웨어, 서비스, 타사 인터넷 사이트의 콘텐츠(코드 포함) 또는 타사 애플리케이션 (b) 계약, 보증, 보장 또는 조건의 불이행, 무과실 책임, 과실 또는 불법 행위로 인한 청구나 관련 법률에서 허용하는 범위 내의 기타 사례로 청구에 적용됩니다.

또한 본 제한 사항은 Microsoft가 손해 가능성을 알고 있었거나 알았어야 하는 경우에도 적용됩니다. 귀하가 거주하고 있는 국가나 지역에서 부수적, 파생적 또는 기타 손해의 배제나 제한을 허용하지 않는 경우에는 위의 제한이나 배제가 적용되지 않을 수 있습니다.

EULA ID: ShellsRedist VS2015_업데이트3_ _ <ENU>

5) Windows 기반 도커 이미지—nuget.commandline --버전 4.5.1

(라이선스 약관은 https://github.com/ /Home/blob/dev/LICENSE.txt 에서 확인할 수 있습니다.) NuGet

Apache 라이선스, 버전 2.0(이하 “라이선스”)에 따라 라이선스가 부여됩니다.이 라이선스를 준수하는 경우를 제외하고는 이러한 파일을 사용할 수 없습니다. 라이선스 사본은 다음 사이트에서 구할 수 있습니다.

http://www.apache.org/licenses/LICENSE-2.0

관련 법률에서 요구하거나 서면으로 합의한 경우를 제외하고, 라이선스에 따라 배포된 소프트웨어는 명시적이든 묵시적이든 어떠한 종류의 보증이나 조건 없이 “있는 그대로” 배포됩니다. 라이선스에 따른 구체적인 언어 관리 권한과 제한 사항은 라이선스를 참조하십시오.

7) Windows 기반 도커 이미지—netfx-4.6.2-devpack

MICROSOFT 소프트웨어 추가 프로그램 라이선스 약관

MICROSOFT WINDOWS 운영 체제용 .NET FRAMEWORK 및 관련 언어 팩

Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)에서 이 추가 프로그램의 사용을 허가합니다. Microsoft Windows 운영 체제 소프트웨어(이하 “소프트웨어”)를 사용할 수 있는 라이선스를 받은 경우 이 추가 프로그램을 사용할 수 있습니다. 소프트웨어에 대한 라이선스가 없는 경우 이 소프트웨어를 사용할 수 없습니다. 유효한 라이선스가 부여된 각 소프트웨어 사본과 함께 이 추가 프로그램을 사용할 수 있습니다.

다음 라이선스 약관은 이 추가 프로그램의 추가 사용 조건을 기술합니다. 본 조건 및 소프트웨어 라이선스 약관은 추가 프로그램의 사용에 적용됩니다. 상충되는 부분이 있는 경우 이러한 추가 프로그램의 라이선스 약관이 적용됩니다.

이 추가 프로그램을 사용하면 이 약관에 동의하는 것으로 간주됩니다. 동의하지 않는 경우 이 추가 프로그램을 사용하지 마세요.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 배포 가능 코드. 이 추가 프로그램은 배포 가능 코드로 구성되어 있습니다. “배포 가능 코드”는 아래 조건을 준수하는 경우 개발 중인 프로그램에 배포할 수 있는 코드입니다.
   a. 사용 및 배포 권한.
      - 이 추가 프로그램의 개체 코드 양식을 복사하여 배포할 수 있습니다.
      - 제3자 배포. 프로그램 배포자가 해당 프로그램의 일부로 배포 가능 코드를 복사하고 배포하도록 허용할 수 있습니다.

b. 배포 요구 사항. 배포하는 모든 배포 가능 코드에 대해 다음을 수행해야 합니다.

- 프로그램에 중요한 기본 기능을 추가합니다.
- 파일 확장명이 .lib인 배포 가능한 코드의 경우, 프로그램과 함께 링커를 통해 해당 배포 가능한 코드를 실행한 결과만 배포합니다.
- 설치 프로그램에 포함된 배포 가능 코드를 수정 없이 설치 프로그램의 일부로만 배포합니다.
- 배포자와 외부 최종 사용자에게 최소한 본 계약만큼 해당 코드를 보호하는 조항에 동의하도록 요구합니다.
- 프로그램에 유효한 저작권 고지를 표시합니다. 그리고
- 프로그램의 배포나 사용과 관련하여 변호사 비용을 포함한 모든 청구에 대해 Microsoft를 면책하고 해를 입히지 않으며 방어해야 합니다.

c. 배포 제한 사항. 다음과 같은 행위는 허용되지 않습니다.

- 배포 가능 코드의 저작권, 상표 또는 특허 고지를 변경하는 행위
- 귀하의 애플리케이션이 Microsoft에서 제공했거나 승인했다고 제안하는 방식으로 Microsoft의 상표를 귀하의 애플리케이션 이름에 사용하는 행위
- Windows 플랫폼 이외의 플랫폼에서 실행되도록 배포 가능 코드를 배포하는 행위
- 악성, 기만성 또는 불법 프로그램에 배포 가능 코드를 포함시키는 행위 또는
- 배포 가능 코드의 일부분에 예외적 라이선스가 적용되도록 배포 가능 코드의 소스 코드를 수정하거나 배포하는 행위 예외적 라이선스는 사용, 수정 또는 배포의 조건으로서,
  - 코드가 소스 코드 형식으로 공개 또는 배포되어야 하거나
  - 다른 사람에게 해당 소프트웨어를 수정할 수 있는 권리가 있어야 하는 라이선스입니다.

2. 추가 프로그램 지원 서비스. Microsoft는 www.support.microsoft.com/common/international.aspx에 설명된 대로 이 소프트웨어에 대한 지원 서비스를 제공합니다.

8) Windows 기반 도커 이미지—visualfsharptools, v 4.0

(라이선스 약관 참조: https://github.com/dotnet/fsharp/blob/main/License.txt)

Copyright (c) Microsoft Corporation All rights reserved.

Apache 라이선스, 버전 2.0(이하 “라이선스”)에 따라 라이선스가 부여됩니다.이 라이선스를 준수하는 경우를 제외하고는 이러한 파일을 사용할 수 없습니다. 라이선스 사본은 다음 사이트에서 구할 수 있습니다.

http://www.apache.org/licenses/LICENSE-2.0

관련 법률에서 요구하거나 서면으로 합의한 경우를 제외하고, 라이선스에 따라 배포된 소프트웨어는 명시적이든 묵시적이든 어떠한 종류의 보증이나 조건 없이 “있는 그대로” 배포됩니다. 라이선스에 따른 구체적인 언어 관리 권한과 제한 사항은 라이선스를 참조하십시오.

9) 윈도우 기반 도커 이미지 — -4.6 netfx-pcl-reference-assemblies

MICROSOFT 소프트웨어 라이선스 약관

MICROSOFT .NET 이식 가능한 클래스 라이브러리 참조 어셈블리– 4.6

본 라이선스 약관은 Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)과 귀하 사이의 계약입니다. 읽어보세요. 이 조건은 위에 명시된 소프트웨어에 적용됩니다. 또한 다른 조건이 있는 경우를 제외하고는 이 소프트웨어에 대한 Microsoft

- 업데이트,
- 추가 프로그램,
- 인터넷 기반 서비스 및
- 지원 서비스

등에도 적용됩니다. 다른 조건이 있는 경우 해당 조건이 적용됩니다.

소프트웨어를 사용하면 본 약관에 동의하는 것으로 간주됩니다. 이 조건에 동의하지 않는 경우에는 소프트웨어를 사용하지 마세요.

본 라이선스 약관을 준수하는 경우 아래와 같은 영구적 권한을 갖습니다.

1. 설치 및 사용 권한. 프로그램을 설계, 개발 및 테스트하기 위해 이 소프트웨어의 사본을 원하는 수만큼 설치하고 사용할 수 있습니다.
2. 추가 라이선스 요구 사항 및/또는 사용 권한.
   a. 배포 가능 코드. 아래 조건을 준수하는 경우 개발하는 개발자 도구 프로그램에 소프트웨어를 배포하여 프로그램 고객이 모든 디바이스 또는 운영 체제에서 사용할 수 있는 이식 가능한 라이브러리를 개발하도록 할 수 있습니다.
      i. 사용 및 배포 권한. 소프트웨어는 “배포 가능 코드”입니다.
         - 배포 가능 코드. 이 소프트웨어의 개체 코드 양식을 복사하여 배포할 수 있습니다.

- 제3자 배포. 프로그램 배포자가 해당 프로그램의 일부로 배포 가능 코드를 복사하고 배포하도록 허용할 수 있습니다.

ii. 배포 요구 사항. 배포하는 모든 배포 가능 코드에 대해 다음을 수행해야 합니다.

- 프로그램에 중요한 기본 기능을 추가합니다.
- 배포자와 고객에게 최소한 본 계약만큼 해당 코드를 보호하는 조항에 동의하도록 요구합니다.
- 프로그램에 유효한 저작권 고지를 표시합니다. 그리고
- 프로그램의 배포나 사용과 관련하여 변호사 비용을 포함한 모든 청구에 대해 Microsoft를 면책하고 해를 입히지 않으며 방어해야 합니다.

iii. 배포 제한 사항. 다음과 같은 행위는 허용되지 않습니다.

- 배포 가능 코드의 저작권, 상표 또는 특허 고지를 변경하는 행위
- 귀하의 애플리케이션이 Microsoft에서 제공했거나 승인했다고 제안하는 방식으로 Microsoft의 상표를 귀하의 애플리케이션 이름에 사용하는 행위
- 악성, 기만성 또는 불법 프로그램에 배포 가능 코드를 포함시키는 행위 또는
- 배포 가능 코드의 일부분에 예외적 라이선스가 적용되도록 배포 가능 코드를 수정하거나 배포하는 행위 예외적 라이선스는 사용, 수정 또는 배포의 조건으로서,
  - 코드가 소스 코드 형식으로 공개 또는 배포되어야 하거나
  - 다른 사람에게 해당 소프트웨어를 수정할 수 있는 권리가 있어야 하는 라이선스입니다.

3. 라이선스 범위. 본 소프트웨어는 판매되는 것이 아니라 그 사용이 허여되는 것입니다. 본 계약은 소프트웨어를 사용할 수 있는 일부 권리만 제공합니다. Microsoft는 기타 모든 권리를 보유합니다. 이러한 제한과 관계없이 관련 법률에서 귀하에게 더 많은 권한을 부여하지 않는 한, 이 계약에서 명시적으로 허용된 대로만 소프트웨어를 사용할 수 있습니다. 이 과정에서 귀하는 특정 방식으로만 사용할 수 있도록 하는 소프트웨어의 모든 기술적 제한 사항을 준수해야 합니다. 다음과 같은 행위는 허용되지 않습니다.
   - 소프트웨어의 모든 기술적 제한 사항을 해결하는 행위
   - 이러한 제한에도 불구하고 관련 법률에서 명시적으로 허용하는 경우를 제외한 소프트웨어의 리버스 엔지니어링, 디컴파일 또는 디스어셈블 작업을 수행하는 행위
   - 다른 사람이 복사할 수 있도록 소프트웨어를 게시하는 행위
   - 소프트웨어를 임대, 대여 또는 대부하는 행위
4. 피드백. 소프트웨어에 대한 피드백을 제공할 수 있습니다. Microsoft에 소프트웨어에 대한 피드백을 제공하는 경우 귀하는 피드백을 어떠한 방식과 목적으로든 체험으로 사용, 공유 및 상품화할 수 있는 권리를 Microsoft에 제공하는 것입니다. 또한 피드백이 포함된 Microsoft 소프트웨어 또는 서비스의 일부를 사용하거나 상호 작용하기 위해 제품, 기술 및 서비스에 필요한 특허권을 제3자에게 무료

로 제공합니다. Microsoft가 귀하의 피드백을 포함함으로써 제3자에게 소프트웨어 또는 문서의 사용을 허가해야 하는 라이선스가 적용되는 피드백을 제공하지 않습니다. 이러한 권리는 이 계약에서 효력을 유지합니다.

5. 제3자에게 양도. 소프트웨어의 최초 사용자는 소프트웨어 및 본 계약을 제3자에게 직접 양도할 수 있습니다. 양도하기 전에 해당 당사자는 본 계약이 소프트웨어의 전송 및 사용에 적용된다는 데 동의해야 합니다. 최초 사용자는 소프트웨어를 디바이스와 별도로 양도하기 전에 먼저 소프트웨어를 제거해야 합니다. 최초 사용자에게 소프트웨어 사본이 없을 수도 있습니다.
6. 수출 제한. 소프트웨어는 미국 수출 법률 및 규정의 적용을 받습니다. 귀하는 소프트웨어에 적용되는 모든 국내 및 국제 수출법과 규정을 준수해야 합니다. 이러한 법규에는 목적지, 최종 사용자 및 최종 용도에 대한 제한이 포함됩니다. 자세한 내용은 www.microsoft.com/exporting을 참조하세요.
7. 지원 서비스. 이 소프트웨어는 “있는 그대로” 제공되므로 본 소프트웨어에 대한 지원 서비스가 제공되지 않을 수 있습니다.
8. 완전 합의. 본 계약 및 귀하가 이용하는 추가 구성 요소, 업데이트, 인터넷 기반 서비스 및 Microsoft에서 제공하는 지원 서비스에 대한 조항은 소프트웨어 및 지원 서비스에 대한 완전 합의입니다.
9. 관련 법률.
   a. 미국. 소프트웨어를 미국에서 구입한 경우, 국제사법 원칙과 관계없이 본 계약의 해석은 워싱턴 주법을 따르며 계약 위반에 대한 청구 발생 시에도 워싱턴 주법이 적용됩니다. 소비자 보호법, 불공정거래법 및 기타 불법 행위 관련 법규의 적용을 받는 청구가 발생한 경우 귀하가 거주하고 있는 주의 주법이 적용됩니다.
   b. 미국 외 지역. 다른 국가에서 소프트웨어를 구입한 경우 해당 국가의 법률이 적용됩니다.
10. 법적 효력. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 귀하가 거주하고 있는 국가의 법규가 보장하는 다른 권리를 보유할 수 있습니다. 또한 귀하가 소프트웨어를 구입한 당사자와 관련된 권리를 보유할 수도 있습니다. 귀하가 거주하고 있는 국가의 법에서 권리 변경을 허용하지 않는 경우 본 계약은 해당 권리를 변경하지 않습니다.
11. 보증의 부인. 이 소프트웨어는 “있는 그대로” 라이선스가 허가됩니다. 해당 사용으로 인해 발생하는 모든 위험은 귀하의 책임입니다. Microsoft는 어떠한 명시적 보증, 보장 또는 조건도 제시하지 않습니다. 고객은 현지 법에 따라 본 라이선스로 변경될 수 없는 추가적인 소비자 권리를 보유할 수 있습니다. 귀하가 거주하는 지역의 법규가 허용하는 범위 내에서 MICROSOFT는 상업성, 특정 목적에의 적합성 및 비침해성과 관련된 묵시적 보증을 배제합니다.

    오스트레일리아의 경우—귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.
12. 손해 및 보상의 제한 및 배제. MICROSOFT와 공급업체에서 최대 $5.00의 직접적인 손해에 대해서만 보상받을 수 있습니다. 결과적 손해, 이익 손실, 특별, 간접 또는 부수적 손해를 포함한 기타 모든 손해에 대해서는 보상을 받을 수 없습니다.

이러한 제한은

- 소프트웨어, 서비스, 타사 인터넷 사이트의 콘텐츠(코드 포함) 또는 타사 애플리케이션
- 계약, 보증, 보장 또는 조건의 불이행, 무과실 책임, 과실 또는 불법 행위로 인한 청구나 관련 법률에서 허용하는 범위 내의 기타 사례에 대한 청구에 적용됩니다.

또한 본 제한 사항은 Microsoft가 손해 가능성을 알고 있었거나 알았어야 하는 경우에도 적용됩니다. 귀하가 거주하고 있는 국가나 지역에서 부수적, 파생적 또는 기타 손해의 배제나 제한을 허용하지 않는 경우에는 위의 제한이나 배제가 적용되지 않을 수 있습니다.

10) Windows 기반 도커 이미지—visualcppbuildtools v 14.0.25420.1

(라이선스 약관 참조: https://www.visualstudio.com/license-terms/mt644918/)

MICROSOFT VISUAL C++ 빌드 도구

MICROSOFT 소프트웨어 라이선스 약관

MICROSOFT VISUAL C++ 빌드 도구

본 라이선스 약관은 Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)과 귀하 사이의 계약입니다. 이 조건은 위에 명시된 소프트웨어에 적용됩니다. 본 조건은 다른 조건이 있는 경우를 제외하고 소프트웨어의 모든 Microsoft 서비스 또는 업데이트에도 적용됩니다.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 설치 및 사용 권한.

   a. 한 명의 사용자가 소프트웨어 사본을 사용하여 애플리케이션을 개발하고 테스트할 수 있습니다.

2. 데이터. 소프트웨어에서 사용자 및 사용자의 소프트웨어 사용에 대한 정보를 수집하여 Microsoft에 보낼 수 있습니다. Microsoft는 이 정보를 사용하여 서비스를 제공하고 제품 및 서비스를 개선할 수 있습니다. 제품 설명서에 설명된 대로 이러한 시나리오를 대부분 옵트아웃할 수 있지만 전부는 아닙니다. 또한 소프트웨어에는 애플리케이션 사용자로부터 데이터를 수집할 수 있는 몇 가지 기능이 있습니다. 이러한 기능을 사용하여 애플리케이션에서 데이터를 수집할 수 있게 하는 경우 애플리케이션 사용자에게 적절한 고지를 제공하는 것을 포함하여 관련 법률을 준수해야 합니다. 데이터 수집

및 사용에 대한 자세한 내용은 도움말 설명서 및 http://go.microsoft.com/fwlink/?LinkID=528096의 개인 정보 보호 정책에서 확인할 수 있습니다. 소프트웨어를 사용하면 이러한 정책에 동의하는 것으로 간주됩니다.

3. 특정 구성 요소에 대한 조건.

   a. 빌드 서버. BuildServer소프트웨어에는.TXT 파일에 나열된 일부 빌드 서버 구성 요소 및/또는 이 Microsoft 소프트웨어 사용 약관에 따라 BuildeServer 목록에 나열된 모든 파일이 포함될 수 있습니다. 소프트웨어에 포함된 경우 해당 항목을 빌드 컴퓨터에 복사하여 설치할 수 있습니다. 사용자 및 조직의 다른 사용자는 애플리케이션을 컴파일, 빌드, 확인 및 보관하거나 빌드 프로세스의 일부로 품질 또는 성능 테스트를 실행할 목적으로만 빌드 시스템에서 이러한 항목을 사용할 수 있습니다.

   b. Microsoft 플랫폼. 소프트웨어에는 마이크로소프트 윈도우, 마이크로소프트 윈도우 서버, 마이크로소프트 SQL 서버, 마이크로소프트 익스체인지, 마이크로소프트 오피스 및 마이크로소프트의 구성 요소가 포함될 수 SharePoint 있습니다. 이러한 구성 요소에는 해당 구성 요소의 설치 디렉터리 또는 소프트웨어와 함께 제공되는 “Licenses” 폴더에 있는 라이선스 약관에 설명된 대로 별도의 계약 및 자체 제품 지원 정책이 적용됩니다.

   c. 타사 구성 요소. 소프트웨어에는 소프트웨어와 함께 제공되는 ThirdPartyNotices 파일에 설명된 대로 별도의 법적 고지가 있거나 다른 계약이 적용되는 타사 구성 요소가 포함될 수 있습니다. 이러한 구성 요소가 다른 계약의 적용을 받는 경우에도 아래의 고지 사항, 손해 배상 제한 및 제외 조항이 함께 적용됩니다.

   d. 패키지 관리자. 소프트웨어에는 애플리케이션과 함께 사용할 다른 Microsoft 및 타사 소프트웨어 패키지를 다운로드할 수 있는 옵션을 제공하는 Nuget과 같은 패키지 관리자가 포함될 수 있습니다. 이러한 패키지에는 자체 라이선스가 적용되며 본 계약이 적용되지 않습니다. Microsoft는 타사 패키지를 배포하거나 라이선스를 부여하거나 보증을 제공하지 않습니다.

4. 라이선스 범위. 본 소프트웨어는 판매되는 것이 아니라 그 사용이 허여되는 것입니다. 본 계약은 소프트웨어를 사용할 수 있는 일부 권리만 제공합니다. Microsoft는 기타 모든 권리를 보유합니다. 이러한 제한과 관계없이 관련 법률에서 귀하에게 더 많은 권한을 부여하지 않는 한, 이 계약에서 명시적으로 허용된 대로만 소프트웨어를 사용할 수 있습니다. 이 과정에서 귀하는 특정 방식으로만 사용할 수 있도록 하는 소프트웨어의 모든 기술적 제한 사항을 준수해야 합니다. 자세한 내용은 https://docs.microsoft.com/en-us/legal/information-protection/ #1 -를 참조하십시오. software-license-terms installation-and-use-rights 다음과 같은 행위는 허용되지 않습니다.

   - 소프트웨어의 모든 기술적 제한 사항을 해결하는 행위
   - 소프트웨어를 리버스 엔지니어링, 디컴파일 또는 디스어셈블하거나 그러한 시도를 하는 행위(소프트웨어에 포함될 수 있는 특정 오픈 소스 구성 요소의 사용을 규제하는 제3자 라이선스 약관에서 요구하는 경우 제외)
   - Microsoft 또는 해당 공급자의 알림을 제거, 최소화, 차단 또는 수정하는 행위

- 법률에 위배되는 방식으로 소프트웨어를 사용하는 행위 또는
- 소프트웨어를 공유, 게시, 대여 또는 임대하거나 다른 사람이 사용할 수 있도록 소프트웨어를 독립 실행형으로 호스팅된 솔루션으로 제공하는 행위

5. 수출 제한. 목적지, 최종 사용자 및 최종 사용에 대한 제한 사항을 포함하여 소프트웨어에 적용되는 모든 국내 및 국제 수출 법률 및 규정을 준수해야 합니다. 수출 제한에 대한 자세한 내용은 aka.ms/export를 참조하세요.
6. 지원 서비스. 이 소프트웨어는 "있는 그대로" 제공되므로 본 소프트웨어에 대한 지원 서비스가 제공되지 않을 수 있습니다.
7. 완전 합의. 본 계약 및 귀하가 이용하는 추가 구성 요소, 업데이트, 인터넷 기반 서비스 및 지원 서비스에 대한 조항은 소프트웨어 및 지원 서비스에 대한 완전 합의입니다.
8. 관련 법률. 미국에서 본 소프트웨어를 구입한 경우 워싱턴주 법이 본 계약의 위반에 대한 해석 및 클레임에 적용되며, 귀하가 거주하는 지역의 법률이 다른 모든 클레임에 적용됩니다. 다른 국가에서 본 소프트웨어를 구입한 경우 해당 국가의 법률이 적용됩니다.
9. 소비자 권리, 지역 또는 국가별 다양성. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 거주하는 국가 또는 시/도의 법률에 따라 소비자 권리를 비롯한 기타 권리를 보유할 수 있습니다. Microsoft와 맺은 관계와 별도로, 귀하는 소프트웨어를 판매한 당사자와 관련해서 권리를 보유할 수도 있습니다. 귀하가 거주하고 있는 국가 또는 시/도의 법에서 권리 변경을 허용하지 않는 경우 본 계약은 이러한 기타 권리를 변경하지 않습니다. 예를 들어 아래 지역 중 하나에서 소프트웨어를 구입했거나 필수 국가 법률이 적용되는 경우 다음 조항이 적용됩니다.
   - 오스트레일리아. 귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.
   - 캐나다. 캐나다에서 본 소프트웨어를 구입한 경우 자동 업데이트 기능을 끄거나, 인터넷에서 디바이스 연결을 끊거나(단, 인터넷에 다시 연결하면 소프트웨어가 업데이트 확인 및 설치를 다시 시작함), 소프트웨어를 제거하여 업데이트 수신을 중지할 수 있습니다. 제품 설명서(있는 경우)에서 특정 디바이스 또는 소프트웨어의 업데이트를 끄는 방법을 지정할 수도 있습니다.
   - 독일 및 오스트리아.
     - 보증. 적절하게 사용이 허가된 소프트웨어는 대체로 본 소프트웨어와 함께 제공되는 Microsoft 자료에 설명된 대로 작동합니다. 그러나 Microsoft는 사용이 허가된 소프트웨어와 관련하여 어떠한 계약 보장도 제공하지 않습니다.
     - 책임의 제한. 의도적인 행위, 심각한 부주의, 제조물 책임법에 따른 클레임 및 개인 또는 신체 부상이나 사망의 경우 Microsoft는 성문법에 따라 책임을 집니다.

       앞 조항 (ii)에 따라 Microsoft는 Microsoft가 이러한 중요한 계약 의무를 위반하거나 본 계약의 적절한 수행을 가능하게 하는 사항을 이행하지 않거나 이행을 위반하거나 본 계약의 목적 및 당

사자가 지속적으로 신뢰할 수 있도록 하는 규정 준수 사항을 위반하는 경우에만 약간의 부주의에 대해 책임집니다. 이 외의 경우에는 약간의 과실에 대해 책임지지 않습니다.

10. 법적 효력. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 귀하가 거주하고 있는 지역 또는 국가의 법규가 보장하는 다른 권리를 보유할 수 있습니다. 귀하가 거주하고 있는 지역 또는 국가의 법에서 권리 변경을 허용하지 않는 경우 본 계약은 해당 권리를 변경하지 않습니다. 이전 조항에 대한 제한 없이, 오스트레일리아의 경우 귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.
11. 보증의 부인. 이 소프트웨어는 “있는 그대로” 라이선스가 허가됩니다. 해당 사용으로 인해 발생하는 모든 위험은 귀하의 책임입니다. Microsoft는 어떠한 명시적 보증, 보장 또는 조건도 제시하지 않습니다. 귀하가 거주하는 지역의 법규가 허용하는 범위 내에서 MICROSOFT는 상업성, 특정 목적에의 적합성 및 비침해성과 관련된 묵시적 보증을 배제합니다.
12. 손해의 제한 및 배제. MICROSOFT와 공급업체에서 최대 $5.00의 직접적인 손해에 대해서만 보상받을 수 있습니다. 결과적 손해, 이익 손실, 특별, 간접 또는 부수적 손해를 포함한 기타 모든 손해에 대해서는 보상을 받을 수 없습니다.

    이러한 제한은 (a) 소프트웨어, 서비스, 타사 인터넷 사이트의 콘텐츠(코드 포함) 또는 타사 애플리케이션 (b) 계약, 보증, 보장 또는 조건의 불이행, 무과실 책임, 과실 또는 불법 행위로 인한 청구나 관련 법률에서 허용하는 범위 내의 기타 사례로 청구에 적용됩니다.

    또한 본 제한 사항은 Microsoft가 손해 가능성을 알고 있었거나 알았어야 하는 경우에도 적용됩니다. 귀하가 거주하고 있는 국가나 지역에서 부수적, 파생적 또는 기타 손해의 배제나 제한을 허용하지 않는 경우에는 위의 제한이나 배제가 적용되지 않을 수 있습니다.

11) 윈도우 기반 도커 이미지— 3-ondemand-package.cab microsoft-windows-netfx

MICROSOFT 소프트웨어 추가 프로그램 라이선스 약관

MICROSOFT WINDOWS 운영 체제용 MICROSOFT .NET FRAMEWORK 3.5 SP1

Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)에서 이 추가 프로그램의 사용을 허가합니다. Microsoft Windows 운영 체제 소프트웨어(본 추가 프로그램이 적용되는)(이하 “소프트웨어”)를 사용할 수 있는 라이선스를 받은 경우 이 추가 프로그램을 사용할 수 있습니다. 소프트웨어에 대한 라이선스가 없는 경우 이 소프트웨어를 사용할 수 없습니다. 유효한 라이선스가 부여된 각 소프트웨어 사본과 함께 이 추가 프로그램 사본을 사용할 수 있습니다.

다음 라이선스 약관은 이 추가 프로그램의 추가 사용 조건을 기술합니다. 본 조건 및 소프트웨어 라이선스 약관은 추가 프로그램의 사용에 적용됩니다. 상충되는 부분이 있는 경우 이러한 추가 프로그램의 라이선스 약관이 적용됩니다.

이 추가 프로그램을 사용하면 이 약관에 동의하는 것으로 간주됩니다. 동의하지 않는 경우 이 추가 프로그램을 사용하지 마세요.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 추가 프로그램 지원 서비스. Microsoft는 www.support.microsoft.com/common/international.aspx에 설명된 대로 이 소프트웨어에 대한 지원 서비스를 제공합니다.
2. MICROSOFT .NET 벤치마크 테스트. 소프트웨어에는 Windows 운영 체제의 .NET Framework, Windows Communication Foundation, Windows Presentation Foundation 및 Windows Workflow Foundation 구성 요소(.NET 구성 요소)가 포함됩니다. .NET 구성 요소의 내부 벤치마크 테스트를 수행할 수 있습니다. http://go.microsoft.com/fwlink/?LinkID=66406에 명시된 조건을 준수하는 경우 .NET 구성 요소의 모든 벤치마크 테스트 결과를 공개할 수 있습니다.

   Microsoft와 체결한 다른 계약에도 불구하고 그러한 벤치마크 테스트 결과를 공개하면 Microsoft는 http://go.microsoft.com/fwlink/?LinkID=66406에 명시된 것과 동일한 조건을 준수하는 경우 해당 .NET 구성 요소와 경쟁하는 제품에 대해 수행한 벤치마크 테스트 결과를 공개할 권리가 있습니다.

12) Windows 기반 도커 이미지—dotnet-sdk

(https://github.com/dotnet/core/blob/main/LICENSE.TXT에서 사용 가능)

MIT 라이선스(MIT)

Copyright (c) Microsoft Corporation

이 소프트웨어 및 관련 문서 파일("소프트웨어") 사본을 획득하는 사람에게 다음 조건에 따라 소프트웨어의 사본을 사용, 복사, 수정, 병합, 게시, 배포, 재사용 허가 및/또는 판매하고, 소프트웨어를 제공 받은 사람이 이렇게 할 수 있도록 허용하기 위한 권한을 제한 없이 포함하여 소프트웨어를 자유롭게 거래할 수 있는 권한이 무료로 허여됩니다.

위의 저작권 알림 및 이 권한은 소프트웨어의 모든 사본 또는 중요한 부분에 포함되어야 합니다.

이 소프트웨어는 상품성, 특정한 목적을 위한 적합성 및 비침해에 관한 보증을 포함하되 이에 제한하지 않고 명시적 또는 묵시적인 어떠한 보증도 없이 “그대로” 제공됩니다. 어떤 경우에도 작성자 또는 저작권 소유자는 계약상 행위, 불법 또는 그 밖의 문제로 인한 것인지와 관계없이, 본 소프트웨어나 소프트웨어의 사용 또는 기타 거래로 인해 발생하는 손해 배상, 손해 또는 기타 문제에 대한 책임을 지지 않습니다.

AWS CodeBuild 사용자 가이드 문서 기록

다음 표에는 의 마지막 릴리스 이후 설명서에서 변경된 주요 내용이 설명되어 AWS CodeBuild있습니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- 최신 API 버전: 2016-10-06

변경 사항	설명	날짜
업데이트 내용: Lambda 컴퓨팅 이미지	.NET 8에 대한 Lambda 지원 추가 (및) `al-lambda/aarch64/dotnet8` `al-lambda/x86_64/dotnet8`	2024년 5월 8일
업데이트된 콘텐츠: AWS 관리 (사전 정의된) 정책 AWS CodeBuild	브랜드 변경을 반영하도록 AWSCodeBuildAdminAccess AWSCodeBuildDeveloperAccess, 및 AWSCodeBuildReadOnlyAccess 정책이 업데이트되었습니다. AWS CodeConnections	2024년 4월 30일
새 콘텐츠: Bitbucket 앱 암호 또는 액세스 토큰	Bitbucket 액세스 토큰에 대한 지원을 추가합니다.	2024년 4월 11일
새 콘텐츠: 보고서 자동 검색	CodeBuild 이제 보고서 자동 검색을 지원합니다.	2024년 4월 4일
새 콘텐츠: 자체 호스팅 액션 러너 GitHub 설정	셀프 GitHub 호스팅 액션 러너를 위한 새 콘텐츠 추가	2024년 4월 2일
새 콘텐츠: 연결 GitLab	지원 GitLab 및 GitHub 자체 관리형 연결을 추가합니다.	2024년 3월 25일
새 콘텐츠: 새 웹후크 이벤트 및 필터 유형 추가	새 웹후크 이벤트 (`RELEASED`및`PRERELEAS`	2024년 3월 15일

	ED) 및 필터 유형 (TAG_NAME및RELEASE_NAME) 에 대한 지원을 추가합니다.	
새 콘텐츠: 새 웹후크 이벤트 추가: PULL_REQUEST_CLOSED	새 웹후크 이벤트에 대한 지원 추가:. PULL_REQUEST_CLOSED	2024년 2월 20일
업데이트 내용: Docker 이미지 제공: CodeBuild	윈도우 서버 코어 2019 지원 추가 () windows-base:2019-3.0	2024년 2월 7일
업데이트 내용: Docker 이미지 제공: CodeBuild	아마존 리눅스 2023의 새 런타임에 대한 지원 추가 () al2/aarch64/standard/3.0	2024년 1월 29일
새 콘텐츠: 예약 용량	CodeBuild 이제 예약 용량 플릿을 지원합니다. CodeBuild	2024년 1월 18일
새 컴퓨팅 유형	CodeBuild 이제 Linux XLarge 컴퓨팅 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2024년 1월 8일
업데이트 내용: Docker 이미지 제공: CodeBuild	Amazon Linux 2(al2/standard/5.0) 및 Ubuntu(ubuntu/standard/7.0)의 새 런타임에 대한 지원 추가	2023년 12월 14일
업데이트 내용: Docker 이미지 제공: CodeBuild	새로운 Lambda 컴퓨팅 이미지에 대한 지원 추가	2023년 12월 8일
새 콘텐츠: 컴퓨팅 AWS Lambda	AWS Lambda 컴퓨팅에 새 콘텐츠 추가	2023년 11월 6일

새 콘텐츠: 빌드스펙에서 GitHub Actions 구문 사용	빌드 사양에 using GitHub Actions 구문을 위한 새 콘텐츠 추가	2023년 7월 6일
업데이트 내용: Docker 이미지 제공: CodeBuild	Amazon Linux 2(`al2/standard/5.0`)에 대한 지원 추가	2023년 5월 17일
에 대한 관리형 정책 변경 CodeBuild	이제 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인할 수 CodeBuild 있습니다. 자세한 내용은 AWS 관리형 정책 CodeBuild 업데이트를 참조하십시오.	2023년 5월 16일
업데이트 내용: Docker 이미지 제공: CodeBuild	Amazon Linux 2(`al2/standard/3.0`)에 대한 지원 제거, Amazon Linux 2(`al2/standard/corretto8`) 및 Amazon Linux 2(`al2/standard/corretto11`)에 대한 지원 추가	2023년 5월 9일
업데이트 내용: Docker 이미지 제공: CodeBuild	Ubuntu 22.04(`ubuntu/standard/7.0`)에 대한 지원 추가	2023년 4월 13일
업데이트 내용: Docker 이미지 제공: CodeBuild	Ubuntu 18.04(`ubuntu/standard/4.0`) 및 Amazon Linux 2(`al2/aarch64/standard/1.0`)에 대한 지원 제거	2023년 3월 31일

콘텐츠 업데이트: VPC 제한 제거	다음 제한 제거: VPC와 함께 CodeBuild 작동하도록 구성한 경우 로컬 캐싱은 지원되지 않습니다. 2022년 2월 28일부터 각 빌드에 새 Amazon EC2 인스턴스가 사용되므로 VPC 빌드에 시간이 더 오래 걸립니다.	2023년 3월 1일
업데이트 내용: Docker 이미지 제공: CodeBuild	Ubuntu 18.04(`ubuntu/standard/3.0`) 및 Amazon Linux 2(`al2/standard/2.0`)에 대한 지원 제거	2022년 6월 30일
Amazon ECR 샘플: 이미지 액세스 제한	CodeBuild 자격 증명을 사용하여 Amazon ECR 이미지를 가져오는 경우 이미지 액세스를 특정 CodeBuild 프로젝트로 제한할 수 있습니다. 자세한 내용은 Amazon ECR 샘플을 참조하세요.	2022년 3월 10일
리전 지원 추가	이제 아시아 태평양(서울), 캐나다(중부), 유럽(런던), 유럽(파리) 리전에서 `ARM_CONTAINER` 컴퓨팅 유형이 지원됩니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2022년 3월 10일
새 VPC 제한 사항	VPC와 함께 CodeBuild 작동하도록 구성한 경우 로컬 캐싱은 지원되지 않습니다. 2022년 2월 28일부터 각 빌드에 새 Amazon EC2 인스턴스가 사용되므로 VPC 빌드에 시간이 더 오래 걸립니다.	2022년 2월 25일

배치 보고서 모드	CodeBuild 이제 배치 빌드 상태를 프로젝트의 소스 공급자에게 보내는 방법을 선택할 수 있습니다. 자세한 내용은 배치 보고서 모드를 참조하세요.	2021년 10월 4일
새 컴퓨팅 유형	CodeBuild 이제 소형 ARM 컴퓨팅 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2021년 9월 13일
퍼블릭 빌드 프로젝트	CodeBuild 이제 AWS 계정에 액세스할 필요 없이 빌드 프로젝트의 빌드 결과를 대중에게 공개할 수 있습니다. 자세한 내용은 퍼블릭 빌드 프로젝트를 참조하세요.	2021년 8월 11일
배치 빌드를 위한 세션 디버깅	CodeBuild 이제 배치 빌드를 위한 세션 디버깅이 지원됩니다. 자세한 내용은 build-graph 및 build-list를 참조하세요.	2021년 3월 3일
프로젝트 수준 동시 빌드 제한	CodeBuild 이제 빌드 프로젝트의 동시 빌드 수를 제한할 수 있습니다. 자세한 내용은 프로젝트 구성 및 concurrentBuildLimit을 참조하십시오.	2021년 2월 16일
새 buildspec 속성: s3-prefix	CodeBuild 이제 Amazon S3에 업로드되는 아티팩트의 경로 접두사를 지정할 수 있는 아티팩트에 대한 `s3-prefix` buildspec 속성을 제공합니다. 자세한 내용은 s3-prefix를 참조하세요.	2021년 2월 9일

새 buildspec 속성: on-failure	CodeBuild 이제 빌드 단계에 실패할 경우 어떤 일이 발생하는지 확인할 수 있는 빌드 단계를 위한 `on-failure` buildspec 속성을 제공합니다. 자세한 내용은 on-failure를 참조하세요.	2021년 2월 9일
새 buildspec 속성: exclude-paths	CodeBuild 이제 빌드 아티팩트에서 경로를 제외할 수 있는 아티팩트에 대한 `exclude-paths` buildspec 속성을 제공합니다. 자세한 내용은 exclude-paths를 참조하세요.	2021년 2월 9일
새 buildspec 속성: enable-symlinks	CodeBuild 이제 ZIP 아티팩트에 심볼릭 링크를 보존할 수 있는 아티팩트에 대한 `enable-symlinks` buildspec 속성을 제공합니다. 자세한 내용은 enable-symlinks를 참조하세요.	2021년 2월 9일
Buildspec 아티팩트 이름 개선	CodeBuild 이제 속성에 경로 정보가 포함될 수 있습니다. `artifacts/name` 자세한 내용은 name을 참조하세요.	2021년 2월 9일
코드 범위 보고	CodeBuild 이제 코드 커버리지 보고서를 제공합니다. 자세한 내용은 코드 범위 보고서를 참조하세요.	2020년 7월 30일
배치 빌드	CodeBuild 이제 프로젝트의 동시 빌드 및 조정 빌드 실행을 지원합니다. 자세한 내용은 Batch 빌드를 참조하십시오 CodeBuild.	2020년 7월 30일

Windows Server 2019 이미지	CodeBuild 이제 Windows Server Core 2019 빌드 이미지를 제공합니다. 자세한 내용은 에서 제공하는 CodeBuild Docker 이미지를 참조하십시오.	2020년 7월 20일
세션 관리자	CodeBuild 이제 실행 중인 빌드를 일시 중지한 다음 AWS Systems Manager 세션 관리자를 사용하여 빌드 컨테이너에 연결하고 컨테이너 상태를 볼 수 있습니다. 자세한 내용은 세션 관리자를 참조하세요.	2020년 7월 20일
업데이트된 주제	CodeBuild 이제 buildspec 파일에서 빌드 환경에서 사용할 셸을 지정할 수 있습니다. 자세한 내용은 빌드 사양 참조를 참조하세요.	2020년 6월 25일
테스트 프레임워크를 사용하여 테스트 보고	여러 테스트 프레임워크를 사용하여 CodeBuild 테스트 보고서를 생성하는 방법을 설명하는 몇 가지 항목이 추가되었습니다. 자세한 내용은 테스트 프레임워크를 사용하여 테스트 보고를 참조하십시오.	2020년 5월 29일
업데이트된 주제	CodeBuild 이제 보고서 그룹에 태그를 추가할 수 있습니다. 자세한 내용은 을 참조하십시오 ReportGroup.	2020년 5월 21일
테스트 보고 지원	CodeBuild 이제 테스트 보고에 대한 지원이 일반적으로 제공됩니다.	2020년 5월 21일

업데이트된 주제	CodeBuild 이제 헤드 커밋 메시지가 지정된 표현식과 일치하는 경우에만 빌드를 트리거하는 Github 및 Bitbucket용 웹후크 필터 생성을 지원합니다. 자세한 내용은 GitHub 풀 요청 및 웹후크 필터 샘플과 Bitbucket 풀 요청 및 웹후크 필터 샘플을 참조하십시오.	2020년 5월 6일
새로운 주제	CodeBuild 이제 빌드 프로젝트 및 보고서 그룹 리소스 공유를 지원합니다. 자세한 내용은 공유 프로젝트 작업 및 공유 보고서 그룹 작업을 참조하십시오.	2019년 12월 13일
신규 및 업데이트된 주제	CodeBuild 이제 빌드 프로젝트 실행 중 테스트 보고를 지원합니다. 자세한 내용은 테스트 보고 사용하기, 테스트 보고서 만들기 및 AWS CLI 샘플을 사용하여 테스트 보고서 만들기를 참조하십시오.	2019년 11월 25일
업데이트된 주제	CodeBuild 이제 Linux GPU 및 Arm 환경 유형과 `2xlarge` 컴퓨팅 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2019년 11월 19일
업데이트된 주제	CodeBuild 이제 모든 빌드에서 빌드 번호, 환경 변수 내보내기 및 AWS Secrets Manager 통합을 지원합니다. 자세한 내용은 내보낸 변수 및 Buildspec 구문의 Secrets Manager를 참조하세요.	2019년 11월 6일

새 주제	CodeBuild 이제 알림 규칙을 지원합니다. 알림 규칙을 사용하여 사용자에게 빌드 프로젝트의 중요한 변경 사항을 알릴 수 있습니다. 자세한 내용은 알림 규칙 생성을 참조하세요.	2019년 11월 5일
업데이트된 주제	CodeBuild 이제 안드로이드 버전 29와 Go 버전 1.13 런타임을 지원합니다. 자세한 내용은 에서 제공하는 Docker 이미지 CodeBuild 및 Buildspec 구문을 참조하세요.	2019년 9월 10일
업데이트된 주제	이제 프로젝트를 생성할 때 Amazon Linux 2(AL2) 관리형 이미지를 선택할 수 있습니다. 자세한 내용은 에서 제공하는 Docker 이미지 CodeBuild 및 buildspec 파일 샘플의 런타임 버전을 참조하십시오. CodeBuild	2019년 8월 16일
업데이트된 주제	이제 프로젝트를 생성할 때 S3 로그 암호화를 비활성화할 수 있고, Git 기반 소스 리포지토리를 사용할 경우 Git 하위 모듈을 포함할 수 있습니다. 자세한 내용은 에서 빌드 프로젝트 만들기를 참조하십시오. CodeBuild	2019년 3월 8일
새 주제	CodeBuild 이제 로컬 캐싱을 지원합니다. 빌드를 생성할 때 로컬 캐싱을 네 가지 모드 중 하나 이상 지정할 수 있습니다. 자세한 내용은 캐싱 빌드를 참조하십시오. CodeBuild	2019년 2월 21일

새로운 주제	CodeBuild 이제 빌드를 트리거하는 이벤트를 지정하는 웹후크 필터 그룹을 지원합니다. 자세한 내용은 웹후크 이벤트 필터링 및 Bitbucket GitHub 웹후크 이벤트 필터링을 참조하십시오.	2019년 2월 8일
새 주제	이제 사용 CodeBuild 설명서에 프록시 CodeBuild 서버와 함께 사용하는 방법이 나와 있습니다. 자세한 내용은 프록시 CodeBuild 서버와 함께 사용을 참조하십시오.	2019년 2월 4일
업데이트된 주제	CodeBuild 이제 다른 AWS 계정에 있는 Amazon ECR 이미지 사용을 지원합니다. 이 변경 사항을 반영하기 위해 Amazon ECR 샘플, 빌드 프로젝트 생성 CodeBuild, CodeBuild서비스 역할 생성을 비롯한 여러 주제가 업데이트되었습니다.	2019년 1월 24일
프라이빗 Docker 레지스트리 지원	CodeBuild 이제 프라이빗 레지스트리에 저장된 Docker 이미지를 런타임 환경으로 사용할 수 있습니다. 자세한 내용은 AWS Secrets Manager 샘플이 포함된 사설 레지스트리를 참조하십시오.	2019년 1월 24일

업데이트된 주제	CodeBuild 이제 액세스 토큰을 사용하여 GitHub (개인 액세스 토큰 사용) 리포지토리에 연결하고 Bitbucket (앱 비밀번호 사용) 리포지토리에 연결할 수 있습니다. 자세한 내용은 빌드 프로젝트 생성(콘솔) 및 소소 공급자에 액세스 토큰 사용을 참조하십시오.	2018년 12월 6일
업데이트된 주제	CodeBuild 이제 빌드의 각 단계 기간을 측정하는 새 빌드 메트릭을 지원합니다. 자세한 내용은 CodeBuild CloudWatch 메트릭을 참조하십시오.	2018년 11월 15일
VPC 엔드포인트 정책 주제	Amazon VPC 엔드포인트는 CodeBuild 현재 정책을 지원합니다. 자세한 내용은 VPC 엔드포인트 정책 만들기를 참조하십시오. CodeBuild	2018년 11월 9일
업데이트 내용	새 콘솔 환경을 반영하도록 주제가 업데이트되었습니다.	2018년 10월 30일
Amazon EFS 샘플	CodeBuild 프로젝트의 buildspec 파일에 있는 명령을 사용하여 빌드 중에 Amazon EFS 파일 시스템을 마운트할 수 있습니다. 자세한 내용은 Amazon EFS 샘플을 참조하십시오 CodeBuild.	2018년 10월 26일

Bitbucket webhook	CodeBuild 이제 Bitbucket을 리포지토리로 사용할 때 웹후크를 지원합니다. 자세한 내용은 Bitbucket 풀 리퀘스트 샘플을 참조하십시오. CodeBuild	2018년 10월 2일
S3 로그	CodeBuild 이제 S3 버킷의 빌드 로그를 지원합니다. 이전에는 로그를 CloudWatch 사용해서만 로그를 작성할 수 있었습니다. 자세한 내용은 프로젝트 생성을 참조하십시오.	2018년 9월 17일
다중 입력 소스 및 다중 출력 아티팩트	CodeBuild 이제 둘 이상의 입력 소스를 사용하고 둘 이상의 아티팩트 세트를 게시하는 프로젝트가 지원됩니다. 자세한 내용은 다중 입력 소스 및 입력 아티팩트 샘플 및 CodePipeline 통합 및 다중 입력 소스 CodeBuild 및 출력 아티팩트 샘플을 참조하십시오.	2018년 8월 30일
의미 체계 버전 관리 샘플	이제 사용 CodeBuild 설명서에 시맨틱 버전 관리를 사용하여 빌드 시 아티팩트 이름을 만드는 방법을 보여주는 사용 사례 기반 샘플이 있습니다. 자세한 내용은 의미 체계 버전 관리를 사용해 빌드 아티팩트 샘플 이름 지정을 참조하십시오.	2018년 8월 14일

새로운 정적 웹 사이트 샘플	이제 사용 CodeBuild 설명서에는 S3 버킷에서 빌드 출력을 호스팅하는 방법을 보여주는 사용 사례 기반 샘플이 있습니다. 샘플은 암호화되지 않은 빌드 아티팩트를 지원하는 최근 옵션을 이용합니다. 자세한 내용은 빌드 출력을 S3 버킷에서 호스팅하여 정적 웹사이트 생성을 참조하십시오.	2018년 8월 14일
의미 체계 버전 관리를 사용해 아티팩트 이름을 재정의할 수 있도록 지원하는 옵션	이제 시맨틱 버전 관리를 사용하여 빌드 아티팩트의 이름을 지정하는 데 사용하는 형식을 지정할 수 있습니다. CodeBuild 하드 코딩된 이름을 가진 빌드 아티팩트는 동일한 하드 코딩 이름을 사용하는 이전 빌드 아티팩트를 덮어쓰기 때문에 이 기능은 유용합니다. 예를 들어 하루에 여러 번 빌드를 트리거하는 경우 결과물 이름에 타임스탬프를 추가할 수 있습니다. 각 빌드 아티팩트 이름은 고유하므로 이전 빌드의 아티팩트를 덮어쓰지 않습니다.	2018년 8월 7일
암호화되지 않은 빌드 아티팩트 지원	CodeBuild 이제 암호화되지 않은 빌드 아티팩트가 포함된 빌드가 지원됩니다. 자세한 내용은 빌드 프로젝트 생성(콘솔)을 참조하세요.	2018년 7월 26일

Amazon CloudWatch 지표 및 경보에 대한 지원	CodeBuild 이제 CloudWatch 지표 및 경보와의 통합을 제공합니다. CodeBuild 또는 CloudWatch 콘솔을 사용하여 프로젝트 및 계정 수준에서 빌드를 모니터링할 수 있습니다. 자세한 내용은 빌드 모니터링을 참조하십시오.	2018년 7월 19일
빌드 상태 보고 지원	CodeBuild 이제 빌드의 시작 및 완료 상태를 소스 제공자에게 보고할 수 있습니다. 자세한 내용은 에서 빌드 프로젝트 만들기를 참조하십시오 CodeBuild.	2018년 7월 10일
CodeBuild 문서에 추가된 환경 변수	빌드 환경 내의 환경 변수 페이지가 CODEBUILD_BUILD_ID, CODEBUILD_LOG_PATH 및 CODEBUILD_START_TIME 환경 변수와 함께 업데이트되었습니다.	2018년 7월 9일
buildspec 파일에서 `finally` 블록 지원	buildspec 파일의 선택적 `finally` 블록에 대한 세부 정보로 CodeBuild 설명서가 업데이트되었습니다. 마지막 블록에서의 명령은 항상 해당 명령 블록에서의 명령 후에 실행됩니다. 자세한 내용은 Buildspec 구문을 참조하십시오.	2018년 6월 20일

CodeBuild 에이전트 업데이트 알림	CodeBuild 설명서는 CodeBuild 에이전트의 새 버전이 출시될 때 Amazon SNS를 사용하여 알림을 받는 방법에 대한 세부 정보로 업데이트되었습니다. 자세한 내용은 새 AWS CodeBuild 에이전트 버전에 대한 알림 수신을 참조하십시오.	2018년 15월 6일

이전 업데이트

다음 표에서는 2018년 6월 이전 AWS CodeBuild 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	날짜
Windows 빌드 지원	CodeBuild 이제 Windows용 .NET Core 2.0용 사전 패키징된 빌드 환경을 포함하여 Microsoft Windows Server 플랫폼용 빌드를 지원합니다. 자세한 정보는 마이크로소프트 윈도우용 샘플 CodeBuild을 참조하세요.	2018년 5월 25일
빌드 멱등성 지원	AWS CLI(AWS Command Line Interface)로 `start-build` 명령을 실행할 때 빌드가 멱등적(idempotent)임을 지정할 수 있습니다. 자세한 정보는 빌드 실행(AWS CLI)을 참조하세요.	2018년 5월 15일
더 많은 빌드 프로젝트 설정 재정의 지원	빌드를 생성할 때 이제 더 많은 빌드 프로젝트 설정을 재정의	2018년 5월 15일

변경 사항	설명	날짜
	할 수 있습니다. 재정의는 해당 빌드에만 적용됩니다. 자세한 정보는 AWS CodeBuild에서 빌드 실행을 참조하세요.	
VPC 엔드포인트 지원	이제 VPC 종단점을 사용하여 빌드의 보안을 향상시킬 수 있습니다. 자세한 정보는 VPC 엔드포인트 사용을 참조하세요.	2018년 3월 18일
트리거의 지원	이제 정기적으로 빌드를 예약하도록 트리거를 생성할 수 있습니다. 자세한 정보는 AWS CodeBuild 트리거 생성을 참조하세요.	2018년 3월 28일
FIPS 엔드포인트 설명서	이제 AWS Command Line Interface (AWS CLI) 또는 AWS SDK를 사용하여 네 가지 연방 정보 처리 표준 (FIPS) 엔드포인트 중 하나를 사용하도록 CodeBuild 지시하는 방법에 대해 알아볼 수 있습니다. 자세한 정보는 AWS CodeBuild 엔드포인트 지정을 참조하세요.	2018년 3월 28일
AWS CodeBuild 아시아 태평양 (뭄바이), 유럽 (파리), 남아메리카 (상파울루) 에서 사용 가능	AWS CodeBuild 이제 아시아 태평양 (뭄바이), 유럽 (파리), 남아메리카 (상파울루) 지역에서 사용할 수 있습니다. 자세한 내용은 AWS CodeBuild의 Amazon Web Services 일반 참조 섹션을 참조하십시오.	2018년 3월 28일

변경 사항	설명	날짜
GitHub 엔터프라이즈 서버 지원	CodeBuild 이제 GitHub 엔터프라이즈 서버 리포지토리에 저장된 소스 코드를 기반으로 빌드할 수 있습니다. 자세한 정보는 GitHub 엔터프라이즈 서버 샘플을 참조하세요.	2018년 1월 25일
Git 복제 수준 지원	CodeBuild 이제 지정된 커밋 수만큼 히스토리가 잘린 얕은 클론을 생성할 수 있습니다. 자세한 정보는 빌드 프로젝트 생성을 참조하세요.	2018년 1월 25일
VPC 지원	이제 VPC 활성화 빌드는 VPC 내부의 리소스에 액세스할 수 있습니다. 자세한 정보는 VPC 지원을 참조하세요.	2017년 11월 27일
종속성 캐싱 지원	CodeBuild 이제 종속성 캐싱을 지원합니다. 이렇게 하면 재사용 가능한 빌드 환경 일부를 캐시에 CodeBuild 저장하여 여러 빌드에서 사용할 수 있습니다.	2017년 11월 27일
빌드 배지 지원	CodeBuild 이제 프로젝트의 최신 빌드 상태를 표시하는 내장 가능하고 동적으로 생성되는 이미지 (배지) 를 제공하는 빌드 배지를 사용할 수 있습니다. 자세한 정보는 빌드 배지 샘플을 참조하세요.	2017년 11월 27일

변경 사항	설명	날짜
AWS Config 통합	AWS Config 이제 CodeBuild AWS 리소스로 지원되므로 서비스에서 CodeBuild 프로젝트를 추적할 수 있습니다. 에 대한 자세한 내용은 AWS Config을 참조하십시오AWS Config 샘플.	2017년 10월 20일
리포지토리에서 GitHub 업데이트된 소스 코드를 자동으로 다시 빌드합니다.	소스 코드가 리포지토리에 저장되어 있는 경우 코드 변경 사항이 GitHub 리포지토리에 푸시될 때마다 소스 코드를 다시 빌드할 수 있습니다. AWS CodeBuild 자세한 정보는 GitHub 풀 리퀘스트 및 웹후크 필터 샘플을 참조하세요.	2017년 9월 21일

변경 사항	설명	날짜
Amazon EC2 Systems Manager Parameter Store에서 중요하거나 규모가 큰 환경 변수를 저장하고 검색하는 새로운 방법	이제 AWS CodeBuild 콘솔 또는 를 사용하여 Amazon EC2 Systems Manager 파라미터 스토어에 저장된 민감하거나 큰 환경 변수를 검색할 수 있습니다. AWS CLI 또한 이제 AWS CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager 파라미터 스토어에 이러한 유형의 환경 변수를 저장할 수 있습니다. 이전에는 이러한 유형의 환경 변수를 빌드 사양에 포함하거나 빌드 명령을 실행하여 AWS CLI를 자동화하는 방식으로만 이러한 환경 변수를 검색할 수 있었습니다. Amazon EC2 Systems Manager Parameter Store 콘솔을 사용해서만 이러한 유형의 환경 변수를 저장할 수 있었습니다. 자세한 내용은 빌드 프로젝트 생성, 빌드 프로젝트 설정 변경, 빌드 실행 섹션을 참조하세요.	2017년 9월 14일
빌드 삭제 지원	이제 AWS CodeBuild에서 빌드를 삭제할 수 있습니다. 자세한 정보는 빌드 삭제을 참조하세요.	2017년 8월 31일

변경 사항	설명	날짜
buildspec을 사용하여 Amazon EC2 Systems Manager Parameter Store에 저장된 중요하거나 규모가 큰 환경 변수를 검색하는 방식이 업데이트되었습니다.	AWS CodeBuild 이제 빌드스펙을 사용하여 Amazon EC2 Systems Manager 파라미터 스토어에 저장된 민감하거나 큰 환경 변수를 더 쉽게 검색할 수 있습니다. 이전에는 빌드 명령을 실행하여 AWS CLI를 자동화하는 방식으로만 이러한 환경 변수를 검색할 수 있었습니다. 자세한 내용은 `parameter-store` 의 buildspec 구문 매핑을 참조하세요.	2017년 8월 10일
AWS CodeBuild 비트버킷을 지원합니다.	CodeBuild 이제 Bitbucket 리포지토리에 저장된 소스 코드를 기반으로 빌드할 수 있습니다. 자세한 내용은 빌드 프로젝트 생성 및 빌드 실행 섹션을 참조하세요.	2017년 8월 10일
AWS CodeBuild 미국 서부 (캘리포니아 북부), 유럽 (런던), 캐나다 (중부) 에서 사용 가능	AWS CodeBuild 이제 미국 서부 (캘리포니아 북부), 유럽 (런던) 및 캐나다 (중부) 지역에서 사용할 수 있습니다. 자세한 내용은 AWS CodeBuild의 Amazon Web Services 일반 참조 섹션을 참조하십시오.	2017년 6월 29일

변경 사항	설명	날짜
대체 빌드 사양 파일 이름 및 지원 위치	이제 소스 코드의 루트에서 `buildspec.yml` 이라는 기본 빌드 사양 파일 대신 빌드 프로젝트에 사용할 빌드 사양 파일의 대체 파일 이름 또는 위치를 지정할 수 있습니다. 자세한 정보는 buildspec 파일 이름 및 스토리지 위치을 참조하세요.	2017년 27월 6일
업데이트된 빌드 알림 샘플	CodeBuild 이제 Amazon CloudWatch Events 및 Amazon Simple Notification Service (Amazon SNS) 를 통해 빌드 알림을 기본적으로 지원합니다. 이전 빌드 알림 샘플 섹션이 업데이트되어 이 새로운 동작을 보여 줍니다.	2017년 6월 22일
Docker 사용자 지정 이미지 샘플 추가	Docker 이미지를 CodeBuild 빌드하고 실행하는 데 사용하는 방법과 사용자 지정 Docker 빌드 이미지를 보여 주는 샘플이 추가되었습니다. 자세한 내용은 도커 사용자 지정 이미지 샘플 섹션을 참조하십시오.	2017년 6월 7일

변경 사항	설명	날짜
풀 리퀘스트의 소스 코드를 가져옵니다. GitHub	GitHub 리포지토리에 저장된 소스 코드를 사용하여 CodeBuild 빌드를 실행할 때 이제 빌드할 GitHub 풀 요청 ID를 지정할 수 있습니다. 또한 커밋 ID, 분기 이름 또는 태그 이름을 대신 지정할 수도 있습니다. 자세한 내용은 빌드 실행(콘솔)의 소스 버전 값 및 빌드 실행(AWS CLI)의 `sourceVersion` 값을 참조하세요.	2017년 6월 6일
빌드 사양 버전 업데이트	빌드 사양 형식의 새 버전이 릴리스되었습니다. 버전 0.2는 기본 셸의 개별 인스턴스에서 각 빌드 명령을 CodeBuild 실행하는 문제를 해결합니다. 또한 버전 0.2에서 `environment_variables` 는 `env`로, `plaintext` 는 `variables` 로 이름이 다시 지정되었습니다. 자세한 정보는 에 대한 빌드 사양 참조 CodeBuild을 참조하세요.	2017년 5월 9일
빌드 이미지용 Dockerfile은 에서 사용할 수 있습니다. GitHub	에서 제공하는 많은 빌드 이미지에 대한 정의를 Dockerfiles로 사용할 수 AWS CodeBuild 있습니다. GitHub 자세한 내용은 Docker 이미지 제공: CodeBuild에 있는 표의 정의 열을 참조하세요.	2017년 5월 2일

변경 사항	설명	날짜
AWS CodeBuild 유럽 (프랑크푸르트), 아시아 태평양 (싱가포르), 아시아 태평양 (시드니), 아시아 태평양 (도쿄) 에서 사용 가능	AWS CodeBuild 이제 유럽 (프랑크푸르트), 아시아 태평양 (싱가포르), 아시아 태평양 (시드니) 및 아시아 태평양 (도쿄) 지역에서 사용할 수 있습니다. 자세한 내용은 AWS CodeBuild의 Amazon Web Services 일반 참조 섹션을 참조하십시오.	2017년 3월 21일
CodePipeline 테스트 액션 지원: CodeBuild	이제 를 사용하는 테스트 액션의 파이프라인에 추가할 수 CodeBuild 있습니다. CodePipeline 자세한 정보는 CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔)을 참조하세요.	2017년 3월 8일
빌드 사양 파일에서 선택된 최상위 디렉터리 내부로부터 빌드 출력 가져오기 지원	이제 Buildspec 파일을 사용하면 콘텐츠가 빌드 출력 아티팩트에 포함되도록 CodeBuild 지시할 수 있는 개별 최상위 디렉터리를 지정할 수 있습니다. `base-directory` 매핑을 이용하면 이러한 작업이 가능합니다. 자세한 정보는 buildspec 구문을 참조하세요.	2017년 2월 8일

변경 사항	설명	날짜
내장 환경 변수	AWS CodeBuild 빌드에서 사용할 추가 빌트인 환경 변수를 제공합니다. 빌드를 시작한 엔터티, 소스 코드 리포지토리에 대한 URL, 소스 코드의 버전 ID 등을 설명하는 환경 변수가 추가되었습니다. 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.	2017년 1월 30일
AWS CodeBuild 미국 동부 (오하이오) 에서 사용 가능	AWS CodeBuild 이제 미국 동부 (오하이오) 지역에서 사용할 수 있습니다. 자세한 내용은 AWS CodeBuild의 Amazon Web Services 일반 참조 섹션을 참조하십시오.	2017년 1월 19일
쉘 및 명령 동작 정보	CodeBuild 지정한 각 명령을 빌드 환경 기본 셸의 개별 인스턴스에서 실행합니다. 이러한 기본 동작은 명령에 예기치 못한 부작용을 초래할 수 있습니다. 필요하면 이러한 기본 동작을 해결하는 몇 가지 방법을 시도해 보는 것이 좋습니다. 자세한 정보는 빌드 환경의 쉘 및 명령을 참조하세요.	2016년 9월 12일
환경 변수 정보	CodeBuild 는 빌드 명령에 사용할 수 있는 여러 환경 변수를 제공합니다. 사용자 고유의 환경 변수를 정의할 수도 있습니다. 자세한 정보는 빌드 환경의 환경 변수을 참조하세요.	2016년 7월 12일

변경 사항	설명	날짜
문제 해결 주제	이제 문제 해결 정보가 제공됩니다. 자세한 정보는 문제 해결 AWS CodeBuild을 참조하세요.	2016년 5월 12일
Jenkins 플러그인 최초 릴리스	CodeBuild Jenkins 플러그인의 초기 릴리스입니다. 자세한 정보는 Jenkins에서 AWS CodeBuild 사용을 참조하세요.	2016년 5월 12일
사용 설명서 최초 릴리스	이것은 CodeBuild 사용자 안내서의 초기 릴리스입니다.	2016년 1월 12일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.