

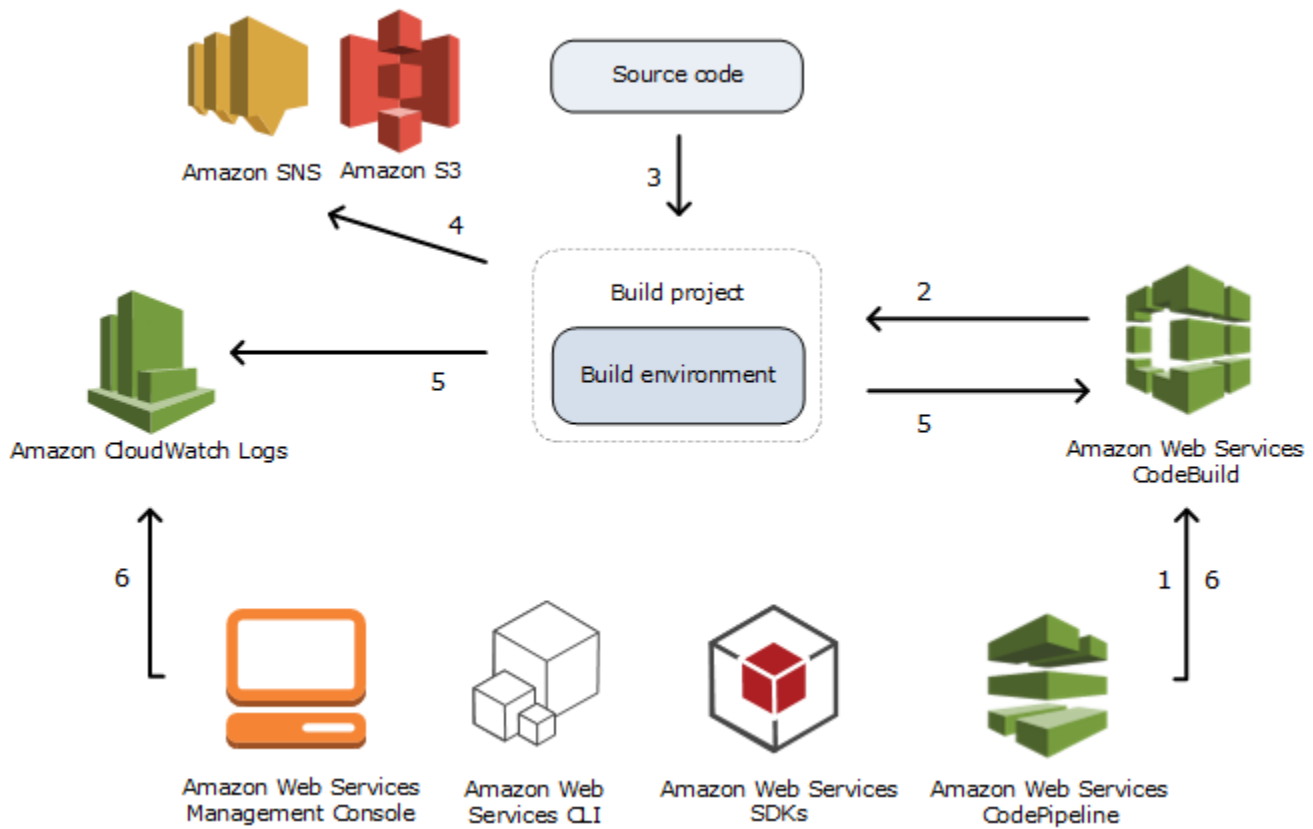
Hands On Exercises

Working with AWS Codebuild

Lab 1	Working with AWS CodeBuild from the Console.....	5
1.	Create the source code	5
2.	buildspec 파일 생성	9
3.	CodeBuild 가 사용 할 S3 버킷 생성	11
4.	소스 코드 및 buildspec 파일 업로드	12
5.	빌드 프로젝트 생성	12
6.	빌드 실행	14
7.	요약 빌드 정보 보기	15
8.	세밀 빌드 정보 보기	16
9.	빌드 출력 아티팩트 가져오기	16
10.	Clean Up	17
Lab 2	AWS CLI 를 사용하여 AWS CodeBuild 시작하기	18
1.	Create the source code	18
2.	buildspec 파일 생성	18
3.	CodeBuild 가 사용 할 S3 버킷 생성	18
4.	소스 코드 및 buildspec 파일 업로드	18
5.	빌드 프로젝트 생성	18
6.	빌드 실행	27
7.	요약된 빌드 정보 보기	29

8. 세밀 빌드 정보 보기 – 이 단계는 Lab1 과 동일 합니다. 왼쪽에 있는 링크를 따라 지침을 받으십시오. 32
9. 빌드 출력 아티팩트 가져오기– 이 단계는 Lab1 과 동일 합니다. 왼쪽에 있는 링크를 따라 지침을 받으십시오. 32
10. Clean-up 32

CodeBuild 작동 방식



CodeBuild 를 사용하여 빌드를 실행할 때 과정:

1. CodeBuild 에게 먼저 빌드 프로젝트(build project) 정보를 제공합니다. 이 빌드 프로젝트 정보에는 소스 코드를 가져올 위치, 사용할 빌드 환경 (build environment), 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다. CodeBuild 의 빌드 환경 (build environment)은 빌드를 실행하는 데 사용하는 운영 체제, 프로그래밍 언어 실행 시간 및 도구의 조합입니다.
2. CodeBuild 는 빌드 프로젝트(build project)를 사용하여 빌드 환경 (build environment)을 생성합니다

3. CodeBuild 는 빌드 환경 (build environment)에 소스 코드를 다운로드 한 다음, 빌드 프로젝트 (build project)에 정의되거나 소스 코드에 직접 포함된 빌드 사양 (buildspec) 을 사용합니다. 빌드 사양 (buildspec) 은 CodeBuild 가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다.
4. 빌드 도중 출력이 있으면 빌드 환경 (build environment)에서 출력을 S3 버킷에 업로드합니다. 빌드 환경 (build environment) 에서 사용자가 buildspec 에 지정한 작업도 수행할 수도 있습니다 (예: Amazon SNS 주제에 빌드 알림 전송).
5. 빌드가 실행되는 동안 빌드 환경 (build environment)이 정보를 CodeBuild 와 Amazon CloudWatch 로그에 전송합니다.
6. 빌드가 실행되는 동안 AWS CodeBuild 콘솔, AWS CLI 또는 AWSSDK 가 CodeBuild 에서 요약된 빌드 정보를 가져오고 Amazon CloudWatch Logs 로그에서 자세한 빌드 정보를 가져올 수 있습니다. AWS CodePipeline 를 빌드 도구로 사용시 CodePipeline 에서 제공하는 제한된 빌드 정보도 가져올 수 있습니다.

Lab 1 Working with AWS CodeBuild from the Console

이 실습에서는 AWS CodeBuild 를 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. CodeBuild 에서 Apache Maven 을 사용하여 Java 클래스 파일 세트를 Java Archive (JAR) 파일에 빌드하도록 명령을 지정합니다.

1. Create the source code

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

1.1 로컬 컴퓨터나 인스턴스의 빈 디렉터리에 다음 디렉터리 구조를 생성합니다

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

1.2 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 매개변수로 전달된 문자열을 출력으로 생성합니다. MessageUtil 생성자는 문자열을 설정합니다. printMessage 메서드는 출력을 생성합니다. salutationMessage 메서드는 Hi! 다음에 문자열을 출력합니다

- 1.3 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 = "Henry";
    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!" + "Henry";
        assertEquals(message,messageUtil.salutationMessage());
    }
}

```

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

- 1.4 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 Maven 은 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

```

2. buildspec 파일 생성

이 단계에서는 빌드 사양 파일을 생성합니다. BuildSpec 는 CodeBuild 가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다.

빌드 사양이 없으면 CodeBuild 가 빌드 입력을 빌드 출력으로 성공적으로 변환하거나 빌드 환경의 빌드 출력 결과물을 찾아 출력 버킷에 업로드할 수 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
```

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 명령을 실행하도록 CodeBuild 에 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 예제에서는 build 단계, CodeBuild mvn install 명령. 이 명령은 Apache Maven 이 Java 클래스 파일을 컴파일 및 테스트하고 컴파일된 Java 클래스 파일을 빌드 출력 결과물에 패키징하도록 지시합니다. 그리고 몇 가지 echo 명령을 각 빌드 단계에 추가하여 이 연습을 마치게 됩니다. 이 예의 뒷부분에서 자세한 빌드 정보를 확인할 때 이러한 항목의 출력을 확인합니다. echo 명령을 사용하면 CodeBuild 가 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.) 빌드 단계마다 CodeBuild 는 처음부터 끝까지 한 번에 하나씩 나열된 순서대로 지정된 각 명령을 실행합니다.
- artifacts 는 CodeBuild 가 출력 버킷에 업로드하는 빌드 출력 아티팩트 집합을 나타냅니다. Files 은 빌드 출력에 포함할 파일을 나타냅니다. Codebuild 가 단일 업로드 messageUtil-1.0.jar 에서 찾은 파일 target 빌드 환경의 상대 디렉터리입니다. 파일 이름 messageUtil-1.0.jar 및 디렉터리 이름 target 은 Apache Maven 이 이 예제에서만 빌드 출력 결과물을 생성 및 저장하는 방식에 따라 달라집니다.

사용자 자체 빌드에서는 이러한 파일 이름과 디렉터리가 다릅니다.

지금까지 디렉토리 구조:

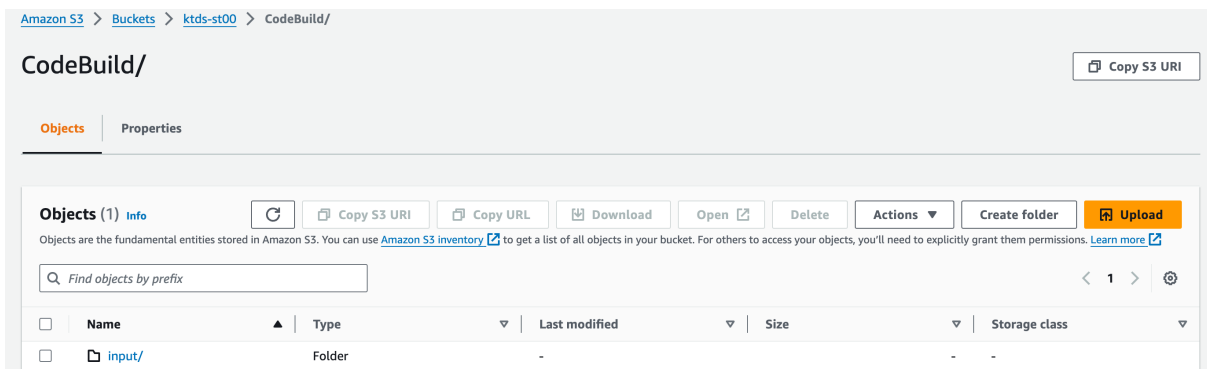
```
(root directory name)
|-- pom.xml
|-- buildspec.yml
`-- src
    |-- main
    |   `-- java
    |       `-- MessageUtil.java
    `-- test
        `-- java
            `-- TestMessageUtil.java
```

3. CodeBuild 가 사용할 S3 메인 버킷 및 입력 폴더 생성

3.1 ktds-st## 이름으로 메인 버킷 생성

주위: S3 bucket 생성시, CodeBuild 가 실행되는 동일한 Region 에서 생성해야 합니다. 여기 실습 환경의 경우 ap-northeast-2 (서울) region 입니다.

3.1.1 소스 파일을 압축해서 저장할 위치 생성 입력을 저장할 버킷: (입력 버킷) 는 빌드 입력을 저장합니다. ktds-st## 아래에 CodeBuild 폴더 생성. 그 아래 input 폴더 생성



3.1.2 아티팩트 (소스코드 빌드한 JAR 파일)는 차후에 ktds-st## 아래 CodeBuild Project 명으로 자동 생성 됩니다.

4. 소스 코드 및 buildspec 파일 업로드

소스 코드 및 빌드 사양 파일을 입력 버킷에 upload 합니다. zip 유틸리티를 사용하여 MessageUtil.java, TestMessageUtil.java, pom.xml 및 buildspec.yml 을 포함하는 MessageUtil.zip 이라는 파일을 생성합니다. (root directory name) 디렉터리는 포함하지 말고, (root directory name) 디렉터리 안에 있는 디렉터리 및 파일만 포함하십시오.

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

CodeCommit, GitHub 및 Bitbucket 리포지토리의 경우 규칙에 따라 buildspec.yml 이라는 빌드 사양 파일을 각 리포지토리의 루트(최상위 수준)에 저장하거나 빌드 사양 선언을 빌드 프로젝트 정의의 일부로 포함해야 합니다. CodeCommit, Github, Bitbuckt 에서는 리포지토리의 소스 코드 및 빌드 사양 파일이 포함된 zip 파일을 생성하지 마십시오. s3 버킷에 저장 하는 경우에만 zip 파일을 위와 같이 생성해서 upload 합니다.

5. 빌드 프로젝트 생성

이 단계에서는 AWS CodeBuild 가 빌드를 실행하는데 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 빌드를

실행하는 데 사용하는 운영 체제, 프로그래밍 언어 런타임 및 CodeBuild 가 빌드를 실행하는 데 사용하는 도구의 조합을 설정합니다. 빌드 환경은 Docker 이미지로 표현됩니다.

이 빌드 환경의 경우 CodeBuild 에 가 JDK (Java Development Kit) 버전 및 Apache Maven 을 포함하는 Docker 이미지를 사용하도록 지시합니다.

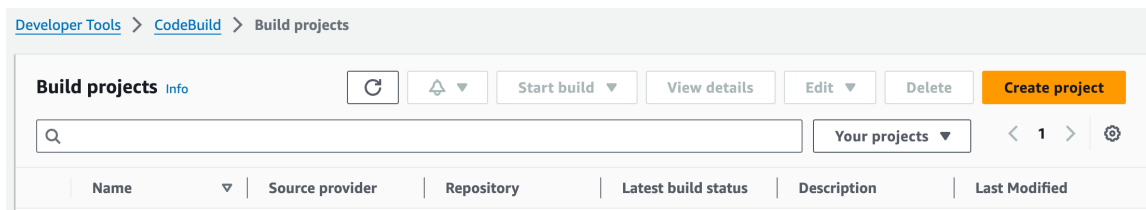
빌드 프로젝트를 생성하려면:

5.1 AWS Management Console 으로 이동

<https://console.aws.amazon.com/codesuite/codebuild/home>

5.2 AWS CodeBuild 가 지원 되는 region 으로 이동. 우리 실습에서는 ap-northeast-2 사용.

5.3 CodeBuild 정보 페이지가 표시되면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.



5.4 빌드 프로젝트 만들기 페이지의 프로젝트 구성에서 프로젝트 이름에 이 빌드 프로젝트의 이름 (<stu##>-codebuild-demo-project)을 입력합니다. 각 AWS 계정에서 빌드 프로젝트 이름은 고유해야 합니다.

5.5 Source -> Source Provider -> Amazon S3 선택.

5.6 Bucket -> 위 단계에서 생성한 bucket 선택

5.7 S3 object key -> (S3 객체 키)에 MessageUtil.zip 의 키를 입력합니다. 키는 AWS S3 에서 파일의 세부 정보를 보시면 있습니다.

S3 object key or S3 folder

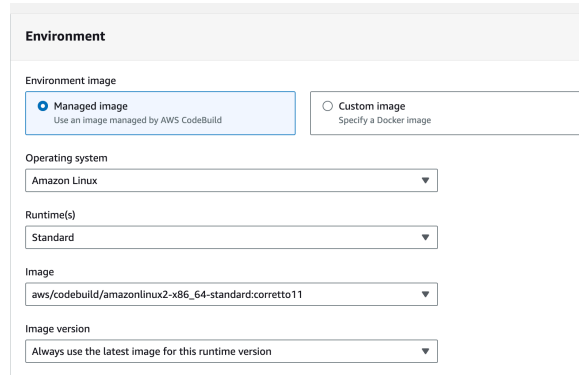
CodeBuild/input/MessageUtil.zip

5.8 환경의 환경 이미지에서 관리형 이미지를 선택된 상태로 둡니다.

5.9 운영 체제에서 Amazon Linux 를 선택합니다.

5.10 Runtime => Standard

5.11 Image => aws/codebuild/amazonlinux2-x86_64-standard:corretto11 선택



The screenshot shows the 'Environment' configuration page for AWS CodeBuild. It includes the following fields:

- Environment image:** Two radio buttons. 'Managed image' (selected) with the subtext 'Use an image managed by AWS CodeBuild', and 'Custom image' with the subtext 'Specify a Docker image'.
- Operating system:** A dropdown menu currently showing 'Amazon Linux'.
- Runtime(s):** A dropdown menu currently showing 'Standard'.
- Image:** A dropdown menu currently showing 'aws/codebuild/amazonlinux2-x86_64-standard:corretto11'.
- Image version:** A dropdown menu currently showing 'Always use the latest image for this runtime version'.

5.12 Service Role

5.12.1 New Service role. Role name 은 변경 안함

5.13 Buildspec

5.13.1 Use a buildspec file

5.14 Artifacts

5.14.1 Type => Amazon s3

5.14.2 Bucket name 위에서 생성한 버킷

5.14.3 Name 및 Path 는 비워 둡니다.

5.14.4 Artifacts packing 은 "None"

5.14.5 나머지는 변경 없음

5.15 Create build project 을 선택합니다.

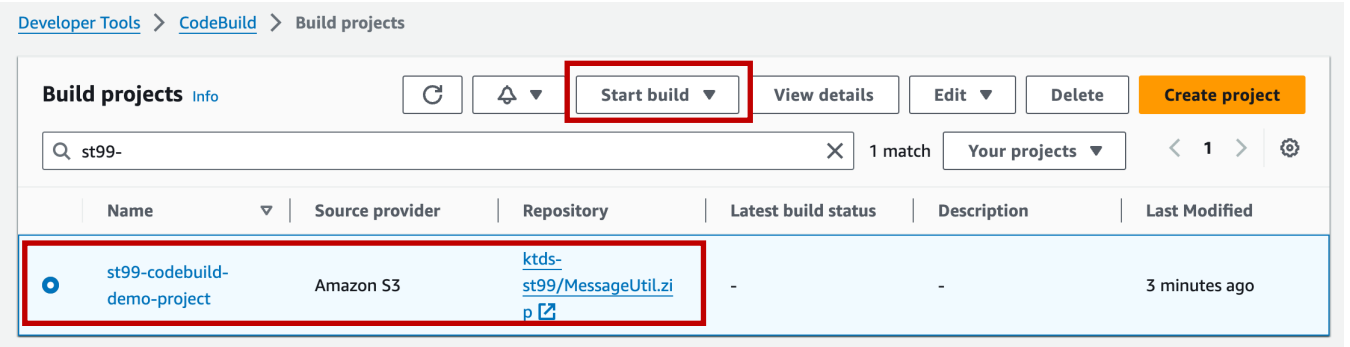
6. 빌드 실행

이 단계에서는 빌드 프로젝트의 설정으로 빌드를 실행하도록 AWS CodeBuild 에 지시를 내립니다.

빌드를 실행하려면:

6.1 CodeBuild 콘솔 열기 <https://console.aws.amazon.com/codesuite/codebuild/home>.

6.2 탐색 창에서 [Build projects]를 선택합니다.



6.3 빌드 프로젝트 목록에서 <st##>-codebuild-demo-project 를 선택 후 “Start build” 실행함

7. 요약 빌드 정보 보기

이 단계에서는 빌드 상태에 대한 요약 정보를 확인합니다.

7.1 만약 <st##>-codebuild-demo-project:<build-ID> 페이지가 표시되지 않으면, 탐색 모음에서 build-history 를 선택합니다. 다음 빌드 프로젝트 목록에서 프로젝트를 선택합니다. 빌드 실행에 대한 <st##>-codebuild-demo-project 링크코드 선택.

7.2 Build status 장에서 Phase detail 관찰. Status 열에 성공 표시와 함께 다음 단계별 표시.

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

Build logs	Phase details	Reports	Environment variables	Build details	Resource utilization
Name	Status	Context	Duration	Start time	End time
SUBMITTED	✔ Succeeded	-	<1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
QUEUED	✔ Succeeded	-	<1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
PROVISIONING	✔ Succeeded	-	7 secs	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
DOWNLOAD_SOURCE	✔ Succeeded	-	1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
INSTALL	✔ Succeeded	-	<1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
PRE_BUILD	✔ Succeeded	-	<1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
BUILD	✔ Succeeded	-	20 secs	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
POST_BUILD	✔ Succeeded	-	<1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
UPLOAD_ARTIFACTS	✔ Succeeded	-	<1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
FINALIZING	✔ Succeeded	-	<1 sec	Feb 16, 2024 1:52 AM (UTC+9:00)	Feb 16, 2024 1:52 AM (UTC+9:00)
COMPLETED	✔ Succeeded	-	-	Feb 16, 2024 1:52 AM (UTC+9:00)	-

빌드 상태에 성공이 표시되어야 합니다. 진행 중이 표시 되었으면 새로 고침 단추를 선택합니다.

7.3 각 빌드 단계 옆에 있는 기간 값은 빌드 단계가 소비한 기간을 나타냅니다. 종료 시간 값은 빌드 단계가 종료되었음을 나타냅니다.

8. 세밀 빌드 정보 보기

이 단계에서는 CloudWatch 로그에서 빌드에 대한 자세한 정보를 확인합니다.

자세한 빌드 정보를 보려면:

- 8.1 이전 단계의 빌드 세부 정보 페이지가 계속 표시된 상태에서 [Build logs]에 빌드 로그의 마지막 부분 행이 표시되어 있습니다. CloudWatch 로그에서 전체 빌드 로그를 확인하려면 전체 로그 보기링크 선택
- 8.2 CloudWatch 로그 로그 스트림에서 로그 이벤트를 찾아볼 수 있습니다. 기본적으로 가장 최근의 로그 이벤트 세트만 표시됩니다. 이전의 로그 이벤트를 보려면 목록의 처음으로 스크롤합니다.
- 8.3 대부분의 로그 이벤트에 가 CodeBuild 를 다운로드 및 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 "[INFO]" 를 입력하면 [INFO] 정보 만 표시됩니다.

9. 빌드 출력 아티팩트 가져오기

이 단계에서는 CodeBuild 가 출력한 messageUtil-1.0.jar 를 확인 합니다.

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

9.1 AWS CodeBuild 콘솔 에서:

9.1.1 이전 단계의 빌드 세부 정보 페이지에서[Build Details] 탭 선택 후 Artifacts 섹션 까지 아래로 스크롤

9.1.2 [Artifacts upload location] 은 CodeBuild 가 출력한 messageUtil-1.0.jar 를 저장 한 Amazon S3 폴더 링크 입니다.

Artifacts Edit		
Artifact identifier -	Artifacts upload location ktds-st99	Packaging None
Cache type No cache	Cache location -	Encryption key arn:aws:kms:ap-northeast-2:740569282574:alias/aws/s3

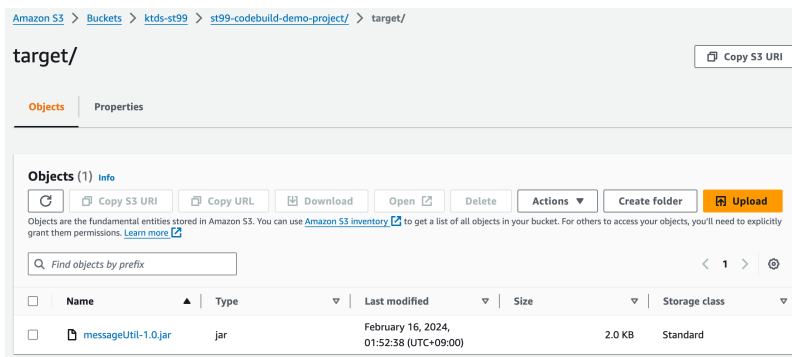
9.2 Amazon S3 콘솔 에서:

9.2.1 <https://console.aws.amazon.com/s3/> 에서 Amazon S3 콘솔을 엽니다.

9.2.2 <stu##>-bucket 을(를) 엽니다.

9.2.3 <stu##>-codebuild-demo-project 폴더를 엽니다.

9.2.4 target 폴더를 엽니다. 이 폴더에서 messageUtil-1.0.jar 빌드 출력 결과물 파일을 찾을 수 있습니다.



10. Clean Up

10.1 다음 실습을 위해서 CodeBuild 에서 생성된 artifact 를 삭제하고 <stu##>-codebuild-region-ID-account-ID-output-bucket 만 남겨 놓습니다.

Lab 2 AWS CLI 를 사용하여 AWS CodeBuild 시작하기

이번 lab 에서는 Lab 1 에서 이미 작업한 1. Create the source code, 2. buildspec 파일 생성, 3. CodeBuild 가 사용할 S3 메인 버킷, 4. 소스 코드 및 buildspec 파일 업로드까지 실행 했다는 가정에서 그 다음 단계 부터는 AWS CLI 를 사용 하여 실행 합니다. 필요시 링크를 따라 선행 작업을 마무리 해야 합니다.

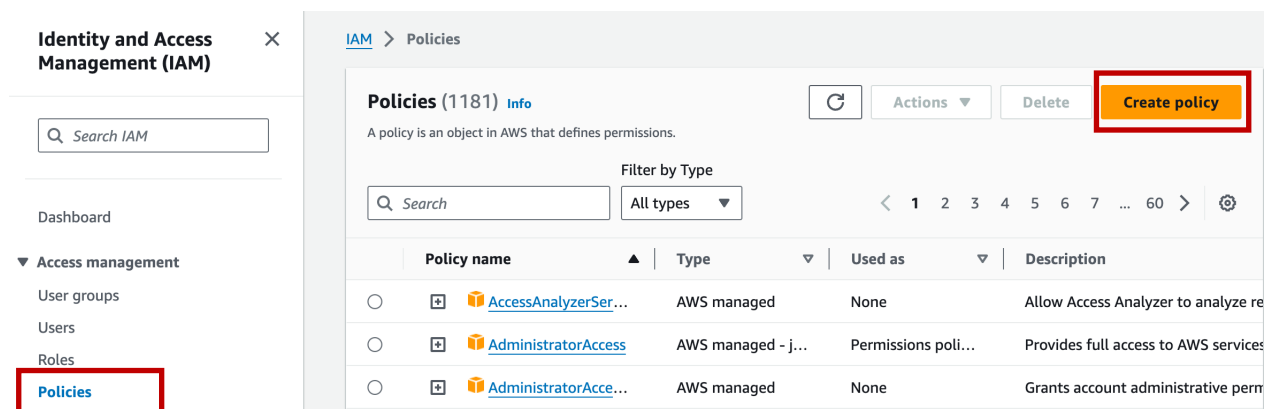
1. Create the source code
2. buildspec 파일 생성
3. CodeBuild 가 사용할 S3 메인 버킷
4. 소스 코드 및 buildspec 파일 업로드
5. 빌드 프로젝트 생성

5.1 CodeBuild service role 이 사용 할 IAM Policy 생성

5.1.1 <https://console.aws.amazon.com/iam/> 에서 IAM 콘솔을 엽니다.

5.1.2 탐색 창에서 “Policies”을 선택합니다.

5.1.3 “Create policy” 버튼을 선택합니다.



5.1.4 [Create Policy] 페이지에서 [JSON]을 선택합니다.

5.1.5 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": "*"
  }
]
}

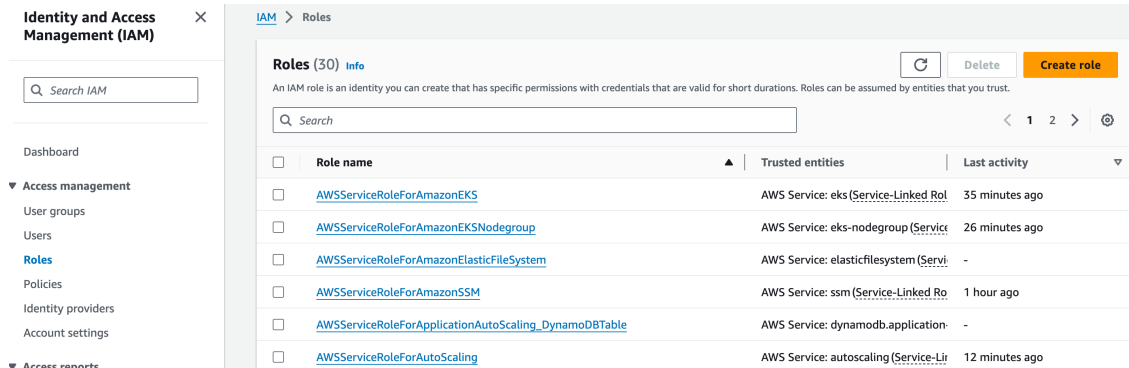
```

5.1.6 Next 후 Tag page 는 통과

5.1.7 정책 검토 페이지의 정책 이름을 <stu##>-CodeBuildServiceRolePolicy 로 설정 후
“Create policy” 버튼을 선택합니다

5.1.8 왼쪽에서 “Roles”을 선택합니다.

5.1.9 Create role(역할 생성)을 선택합니다.



5.1.10 역할 생성페이지에서, 이미 선택 되어있는 AWS 서비스를 사용하고 하단에서 CodeBuild 를 선택 합니다.

Select trusted entity Info

Trusted entity type

- ☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
CodeBuild ▼

Choose a use case for the specified service.
Use case
☒ **CodeBuild**
Allows CodeBuild to call AWS services on your behalf.

Cancel Next

5.1.11 다음:권한 으로 이동

5.1.12 위에서 생성한 <stu##>-CodeBuildServiceRolePolicy 를 선택한 후 다음 으로 이동 합니다

Add permissions [Info](#)

Permissions policies (1/926) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type

Q stu00 X All types 3 matches < 1 > ⚙

<input type="checkbox"/>	Policy name ↗	Type	Description
<input type="checkbox"/>	CodeBuildBasePolicy-stu00-codebu...	Customer managed	Policy used in trust relationship with Cod...
<input type="checkbox"/>	CodeBuildS3ReadOnlyPolicy-stu00...	Customer managed	Policy used in trust relationship with Cod...
<input checked="" type="checkbox"/>	stu00-CodeBuildServiceRolePolicy	Customer managed	-

▶ Set permissions boundary - optional

Cancel Previous Next

5.1.13 역할 생성 및 검토 페이지의 역할 이름을 <stu##>-CodeBuildServiceRole 으로 설정 후 나머지는 기본값으로 하고 “Create Role”를 선택.

[IAM](#) > [Roles](#) > Create role

Step 1
[Select trusted entity](#)

Step 2
[Add permissions](#)

Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=, @, _' characters.

Description
Add a short explanation for this role.

Maximum 1000 characters. Use alphanumeric and '+=, @, _' characters.

5.1.14 생성된 Role 를 검색하여, Role 에 대한 ARN 를 복사한다.

[IAM](#) > [Roles](#)

Roles (28) [Info](#)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Q stu00 X 2 matches < 1 > ⚙

<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	codebuild-stu00-codebuild-demo-project-service-role	AWS Service: codebuild	2 hours ago
<input type="checkbox"/>	stu00-CodeBuildServiceRole	AWS Service: codebuild	-

stu00-CodeBuildServiceRole Info Delete

Allows CodeBuild to call AWS services on your behalf.

Summary Edit


Creation date

April 03, 2024, 11:19 (UTC+09:00)

Last activity

-

ARN

 `arn:aws:iam::740569282574:role/stu00-CodeBuildServiceRole`

Maximum session duration

1 hour

6. 터미널에서 아래 create-project.json 파일을 생성합니다. – Code Build 의 configuration 을 setting 한 json 형태 파일.

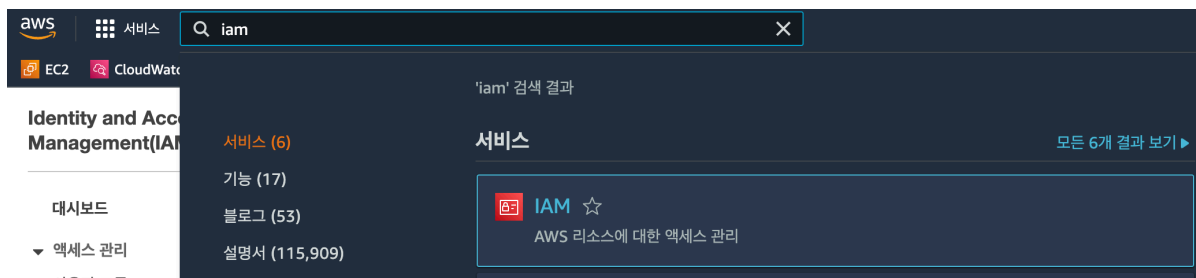
```
{
  "name": "CodeBuild project name",
  "source": {
    "type": "S3",
    "location": "bucket-name/folder-name/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "bucket-name"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:corretto11",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn of code build service role"
}
```

```
{
  "name": "stu00-codebuild-cli-project",
  "source": {
    "type": "S3",
    "location": "ktds-stu00/CodeBuild/input/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "ktds-stu00"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:corretto11",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::740569282574:role/stu00-CodeBuildServiceRole"
}
```

6.1 aws-cli configuration 설정

6.1.1 aws cli install (https://docs.aws.amazon.com/ko_kr/cli/latest/userguide/getting-started-install.html)

6.1.2 IAM 설정 – aws console 를 사용해서 해당 정보를 확인



왼쪽 사용자 메뉴 선택해서 오른쪽 해당 id 선택하고 아래 "보안자격 증명" 클릭

Identity and Access Management(IAM)

대시보드

▼ 액세스 관리

사용자 그룹

사용자

역할

정책

자격 증명 공급자

계정 설정

▼ 보고서 액세스

액세스 분석기

아카이브 규칙

분석기



New feature to generate a policy based on CloudTrail events.
AWS uses your CloudTrail events to identify the services and actions used and

사용자 > st99

요약

사용자 ARN `arn:aws:iam::879772956301:user/st99`

경로 `/`

생성 시간 2022-09-22 21:31 UTC+0900

권한 그룹 (1) 태그 **보안 자격 증명** 액세스 관리자

▼ Permissions policies (2 정책이 적용됨)

권한 추가

A. "Create access key" 선택하고 생성되는 Access Key ID, access key 를 아래 정보에 사용한다.

Access key ID	Secret access key
---------------	-------------------

6.1.3 AWS configuration 을 통한 설정

```
aws configure
```

```
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: ap-northeast-2
Default output format [None]:json
```

해당 값은 위에 AWS 보안자격 증명에서 확인해서 입력합니다. (aws s3 ls)

6.2 create-project.json 파일이 있는 directory 에서 다음 명령 실행

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

정상적으로 실행되면 아래와 같은 출력이 생성됩니다.

- 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` 이후 시간 (분) 을 나타냅니다. CodeBuild 은 빌드가 완료되지 않으면 빌드를 중지합니다. (기본값은 60 분입니다.)
-
- `created` 는 빌드 프로젝트가 생성된 시간을 Unix 시간 형식으로 나타냅니다.
 - `environmentVariables` 는 선언된 환경 변수로, 가 사용할 수 있는 환경 변수를 나타냅니다. CodeBuild 빌드 중에 사용할 수 있습니다.
 - `encryptionKey` 는 고객 관리 키의 ARN 을 나타냅니다. CodeBuild 빌드 출력 결과물을 암호화하는 용도로 사용됩니다.
 - `arn` 은 빌드 프로젝트의 ARN 을 나타냅니다.

```

PS C:\Users\wiken\MSA\CodeBuild> aws codebuild create-project --cli-input-json file://create-project.json
{
  "project": {
    "name": "inst-codebuild-cli-project",
    "arn": "arn:aws:codebuild:ap-northeast-2:879772956301:project/inst-codebuild-cli-project",
    "description": "",
    "source": {
      "type": "S3",
      "location": "inst-codebuild-ap-northeast-2-879772956301-input-bucket/MessageUtil.zip",
      "insecureSsl": false
    },
    "artifacts": {
      "type": "S3",
      "location": "inst-codebuild-ap-northeast-2-879772956301-output-bucket",
      "namespaceType": "NONE",
      "name": "inst-codebuild-cli-project",
      "packaging": "NONE",
      "encryptionDisabled": false
    },
    "cache": {
      "type": "NO_CACHE"
    },
    "environment": {
      "type": "LINUX_CONTAINER",
      "image": "aws/codebuild/standard:4.0",
      "computeType": "BUILD_GENERAL1_SMALL",
      "environmentVariables": [],
      "privilegedMode": false,
      "imagePullCredentialsType": "CODEBUILD"
    },
    "serviceRole": "arn:aws:iam::879772956301:role/inst-CodeBuildServiceRole",
    "timeoutInMinutes": 60,
    "queuedTimeoutInMinutes": 480,
    "encryptionKey": "arn:aws:kms:ap-northeast-2:879772956301:alias/aws/s3",
    "created": "2022-08-31T17:41:30.179000+09:00",
    "lastModified": "2022-08-31T17:41:30.179000+09:00",
    "badge": {
      "badgeEnabled": false
    }
  }
}

```

7. 빌드 실행

이 단계에서는 빌드 프로젝트의 설정으로 빌드를 실행하도록 AWS CodeBuild 에 지시를 내립니다.

- 7.1 AWS CLI 를 사용하여 다음 start-build 명령을 실행합니다. Project-name 은 위 단계에서 설정한 이름을 사용 합니다

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

- 7.2 이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다.

```

PS C:\Users\wiken\MSA\CodeBuild> aws codebuild start-build --project-name inst-codebuild-cli-project
{
  "build": {
    "id": "inst-codebuild-cli-project:cd157ec5-9a11-4d02-920c-8a43fa7d08a3",
    "arn": "arn:aws:codebuild:ap-northeast-2:879772956301:build/inst-codebuild-cli-project:cd157ec5-9a11-4d02-920c-8a43fa7d08a3",
    "buildNumber": 1,
    "startTime": "2022-08-31T18:10:34.168000+09:00",
    "currentPhase": "QUEUED",
    "buildStatus": "IN_PROGRESS",
    "projectName": "inst-codebuild-cli-project",
    "phases": [
      {
        "phaseType": "SUBMITTED",
        "phaseStatus": "SUCCEEDED",
        "startTime": "2022-08-31T18:10:34.168000+09:00",
        "endTime": "2022-08-31T18:10:34.247000+09:00",
        "durationInSeconds": 0
      },
      {
        "phaseType": "QUEUED",
        "startTime": "2022-08-31T18:10:34.247000+09:00"
      }
    ],
    "source": {
      "type": "S3",
      "location": "inst-codebuild-ap-northeast-2-879772956301-input-bucket/MessageUtil.zip",
      "insecureSsl": false
    },
    "artifacts": {
      "location": "arn:aws:s3:::inst-codebuild-ap-northeast-2-879772956301-output-bucket/inst-codebuild-cli-project",
      "encryptionDisabled": false
    },
    "cache": {
      "type": "NO_CACHE"
    },
    "environment": {
      "type": "LINUX_CONTAINER",
      "image": "aws/codebuild/standard:4.0",
      "computeType": "BUILD_GENERAL1_SMALL",
      "environmentVariables": [],
      "privilegedMode": false,
      "imagePullCredentialsType": "CODEBUILD"
    },
    "serviceRole": "arn:aws:iam::879772956301:role/inst-CodeBuildServiceRole",
    "logs": {
      "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=ap-northeast-2#logEvent:group=null;stream=null",
      "cloudWatchLogsArn": "arn:aws:logs:ap-northeast-2:879772956301:log-group:null:log-stream:null"
    },
    "timeoutInMinutes": 60,
    "queuedTimeoutInMinutes": 480,
    "buildComplete": false,
    "initiator": "instructor",
    "encryptionKey": "arn:aws:kms:ap-northeast-2:879772956301:alias/aws/s3"
  }
}

```

- buildComplete 는 빌드 완료 여부를 나타냅니다(true). 그렇지 않을 경우 false 입니다.
- initiator 는 빌드를 시작한 엔터티를 나타냅니다.
- artifacts 는 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 ID 를 나타냅니다.
- arn 은 빌드의 ARN 을 나타냅니다.

7.3 [id] 값을 기록해 둡니다. 이 정보는 다음 단계에서 필요합니다.

8. 요약된 빌드 정보 보기

요약된 빌드 정보를 보려면 AWS CLI 를 사용하여 batch-get-builds 명령을 실행합니다.

8.1 위에서 기록 한 [id] 정보를 사용 하여 아래 명령 실행

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

8.2 이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다. 다만 빌드가 끝날 시간이 필요 합니다.

```
PS C:\Users\wiken\MSA\CodeBuild> aws codebuild batch-get-builds --ids inst-codebuild-cli-project:cd157ec5-9a11-4d02-920c-8a43fa7d08a3
{
  "builds": [
    {
      "id": "inst-codebuild-cli-project:cd157ec5-9a11-4d02-920c-8a43fa7d08a3",
      "arn": "arn:aws:codebuild:ap-northeast-2:879772956301:build/inst-codebuild-cli-project:cd157ec5-9a11-4d02-920c-8a43fa7d08a3",
      "buildNumber": 1,
      "startTime": "2022-08-31T18:10:34.168000+09:00",
      "endTime": "2022-08-31T18:13:22.468000+09:00",
      "currentPhase": "COMPLETED",
      "buildStatus": "SUCCEEDED",
      "projectName": "inst-codebuild-cli-project",
      "phases": [
        {
          "phaseType": "SUBMITTED",
          "phaseStatus": "SUCCEEDED",
          "startTime": "2022-08-31T18:10:34.168000+09:00",
          "endTime": "2022-08-31T18:10:34.247000+09:00",
          "durationInSeconds": 0
        },
        {
          "phaseType": "QUEUED",
          "phaseStatus": "SUCCEEDED",
          "startTime": "2022-08-31T18:10:34.247000+09:00",
          "endTime": "2022-08-31T18:11:17.360000+09:00",
          "durationInSeconds": 43
        },
        {
          "phaseType": "PROVISIONING",
          "phaseStatus": "SUCCEEDED",
          "startTime": "2022-08-31T18:11:17.360000+09:00",
          "endTime": "2022-08-31T18:11:46.388000+09:00",
          "durationInSeconds": 29,
          "contexts": [
            {
              "statusCode": "",
              "message": ""
            }
          ]
        }
      ]
    }
  ],
}
```

```

{
  "phaseType": "DOWNLOAD_SOURCE",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2022-08-31T18:11:46.388000+09:00",
  "endTime": "2022-08-31T18:11:51.259000+09:00",
  "durationInSeconds": 4,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ]
},
{
  "phaseType": "INSTALL",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2022-08-31T18:11:51.259000+09:00",
  "endTime": "2022-08-31T18:11:51.294000+09:00",
  "durationInSeconds": 0,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ]
},
{
  "phaseType": "PRE_BUILD",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2022-08-31T18:11:51.294000+09:00",
  "endTime": "2022-08-31T18:11:51.350000+09:00",
  "durationInSeconds": 0,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ]
},
{
  "phaseType": "BUILD",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2022-08-31T18:11:51.350000+09:00",
  "endTime": "2022-08-31T18:13:20.093000+09:00",
  "durationInSeconds": 88,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ]
},

```

```

{
  "phaseType": "POST_BUILD",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2022-08-31T18:13:20.093000+09:00",
  "endTime": "2022-08-31T18:13:20.137000+09:00",
  "durationInSeconds": 0,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ]
},
{
  "phaseType": "UPLOAD_ARTIFACTS",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2022-08-31T18:13:20.137000+09:00",
  "endTime": "2022-08-31T18:13:20.389000+09:00",
  "durationInSeconds": 0,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ]
},
{
  "phaseType": "FINALIZING",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2022-08-31T18:13:20.389000+09:00",
  "endTime": "2022-08-31T18:13:22.468000+09:00",
  "durationInSeconds": 2,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ]
},
{
  "phaseType": "COMPLETED",
  "startTime": "2022-08-31T18:13:22.468000+09:00"
}

```

```

    "source": {
      "type": "S3",
      "location": "inst-codebuild-ap-northeast-2-879772956301-input-bucket/MessageUtil.zip",
      "insecureSsl": false
    },
    "artifacts": {
      "location": "arn:aws:s3:::inst-codebuild-ap-northeast-2-879772956301-output-bucket/inst-codebuild-cli-project",
      "sha256sum": "",
      "md5sum": "",
      "encryptionDisabled": false
    },
    "cache": {
      "type": "NO_CACHE"
    },
    "environment": {
      "type": "LINUX_CONTAINER",
      "image": "aws/codebuild/standard:4.0",
      "computeType": "BUILD_GENERAL1_SMALL",
      "environmentVariables": [],
      "privilegedMode": false,
      "imagePullCredentialsType": "CODEBUILD"
    },
    "serviceRole": "arn:aws:iam::879772956301:role/inst-CodeBuildServiceRole",
    "logs": {
      "groupName": "/aws/codebuild/inst-codebuild-cli-project",
      "streamName": "cd157ec5-9a11-4d02-920c-8a43fa7d08a3",
      "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=ap-northeast-2#logEvent:group=/aws/codebuild/inst-codebuild-cli-project;stream=cd157ec5-9a11-4d02-920c-8a43fa7d08a3",
      "cloudWatchLogsArn": "arn:aws:logs:ap-northeast-2:879772956301:log-group:/aws/codebuild/inst-codebuild-cli-project:log-stream:cd157ec5-9a11-4d02-920c-8a43fa7d08a3"
    },
    "timeoutInMinutes": 60,
    "queuedTimeoutInMinutes": 480,
    "buildComplete": true,
    "initiator": "instructor",
    "encryptionKey": "arn:aws:kms:ap-northeast-2:879772956301:alias/aws/s3"
  },
  "buildsNotFound": []
}

```

- phases 는 CodeBuild 빌드 단계 를 나타냅니다. 각 빌드 단계에 대한 정보는 startTime, endTime 및 durationInSeconds(빌드 단계가 시작 및 종료된 시간은 Unix 형식으로, 지속된 기간은 초로 표시) 됩니다. Phase 단계는 (SUBMITTED, PROVISIONING, DOWNLOAD_SOURCE, INSTALL, PRE_BUILD, BUILD, POST_BUILD, UPLOAD_ARTIFACTS, FINALIZING 또는 COMPLETED 등) 각 phase 의 Status (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 ID 로 batch-get-builds 명령을 계속하여 실행하면 더 많은 빌드 단계가 출력에 표시됩니다.
- logs 아마존의 정보를 나타냅니다. CloudWatch 빌드 로그에 대한 로그입니다.
- 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.
- endTime 은 빌드 프로세스가 종료된 시간을 Unix 시간 형식으로 나타냅니다.

- `buildsNotFound` 는 정보가 없는 모든 빌드의 빌드 ID 를 나타냅니다. 이 예에서는 비어 있어야 합니다.
9. 세밀 빌드 정보 보기 – 이 단계는 Lab1 과 동일 합니다. 왼쪽에 있는 링크를 따라 지침을 받으십시오.
 10. 빌드 출력 아티팩트 가져오기– 이 단계는 Lab1 과 동일 합니다. 왼쪽에 있는 링크를 따라 지침을 받으십시오.
 11. Clean-up