

# AWS 구성 내역, 내부 구축 내용

## 1. AWS CodeBuild

### 1. 소스

소스

소스 추가

소스 1 - 기본

소스 공급자

GitHub

리포지토리

☒ 퍼블릭 리포지토리

☐ 내 GitHub 계정의 리포지토리

리포지토리 URL

https://github.com/GoonManDoo/library.git

https://github.com/<user-name>/<repository-name>

연결 상태

OAuth를 사용하여 GitHub에 연결되었습니다.

GitHub에서 연결 해제

소스 버전 - 선택 사항 정보

를 요청, 브랜치, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다.

▶ 추가 구성

Git clone 깊이, Git 하위 모듈

### 2. 환경

Java 17 버전이므로 호환되는 standard:5.0 설정

## 환경

현재 환경 이미지

aws/codebuild/amazonlinux2-aarch64-standard:2.0

새 환경 이미지

☒ 관리형 이미지

AWS CodeBuild에서 관리하는 이미지 사용

☐ 사용자 지정 이미지

도커 이미지 지정

운영 체제

Amazon Linux

런타임

Standard

이미지

aws/codebuild/amazonlinux2-x86\_64-standard:5.0

이미지 버전

이 런타임 버전에 항상 최신 이미지 사용

☐ Use GPU-enhanced compute

권한이 있음

☐ 도커 이미지를 빌드하거나 빌드의 권한을 승격하려면 이 플래그를 활성화합니다.

세션 연결

대화형 브라우저 기반 셸을 통해 빌드 머신의 컨테이너를 관리하도록 AWS 세션 관리자를 설정합니다. 서비스 역할에 세션 연결을 활성화하려면 AWS 세션 관리자 권한이 있어야 합니다. [자세히 알아보십시오](#)

☐ 세션 연결 활성화

서비스 역할

계정에서 기존 서비스 역할 선택

arn:aws:iam::815602520517:role/service-role/codebuild-lala\_project-service-r

## 3. BuildSpec

### Buildspec

현재 buildspec

소스 코드 루트 디렉터리의 buildspec.yml 사용 중

빌드 사양

☒ buildspec 파일 사용

YAML 형식의 buildspec 파일에 빌드 명령 저장

☐ 빌드 명령 삽입

빌드 명령을 빌드 프로젝트 구성으로 저장

Buildspec 이름 - 선택 사항

기본적으로 CodeBuild는 소스 코드 루트 디렉터리에서 buildspec.yml 파일을 찾습니다. buildspec 파일이 다른 이름 또는 위치를 사용하는 경우 여기에 소스 루트의 경로를 입력하십시오(예: buildspec-two.yml 또는 configuration/buildspec.yml).

```
# buildspec.yml
# AWS CodeBuild 빌드 버전
version: 0.2
```

```

phases:
  # 빌드 환경 설정
  install:
    # 사용할 Java 런타임 버전[17 지원이 안됨으로 커맨드로 java 17 설치]
    runtime-versions:
      java: corretto11
    commands:
      - export JAVA_17_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
      - export JDK_17_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
      - export JRE_17_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
      - export JAVA_HOME="$JAVA_17_HOME"
      - export JRE_HOME="$JRE_17_HOME"
      - export JDK_HOME="$JDK_17_HOME"
      - export JAVA_HOME=$CODEBUILD_AGENT_JAVA_HOME
      - echo "JAVA_HOME set to $JAVA_HOME"
      - |-
        export GNUPGHOME="$(mktemp -d)" \
          && curl -fL -o corretto.key https://yum.corretto.aws/corretto.key \
          && gpg --batch --import corretto.key \
          && gpg --batch --export --armor '6DC3636DAE534049C8B94623A122542AB04F24E3' > corretto.key \
          && rpm --import corretto.key \
          && rm -r "$GNUPGHOME" corretto.key \
          && curl -fL -o /etc/yum.repos.d/corretto.repo https://yum.corretto.aws/corretto.repo \
          && grep -q '^gpgcheck=1' /etc/yum.repos.d/corretto.repo \
          && yum install -y java-17-amazon-corretto-devel \
          && (find /usr/lib/jvm/java-17-amazon-corretto.x86_64 -name src.zip -delete || true) \
          && yum install -y fontconfig
      - |-
        # java 실행 파일에 대한 심볼릭 설정
        for tool_path in "$JAVA_HOME"/bin/*;
        do tool=$(basename "$tool_path");
          if [ $tool != 'java-rmi.cgi' ];
          then
            rm -f /usr/bin/$tool /var/lib/alternatives/$tool \
              && update-alternatives --install /usr/bin/$tool $tool $tool_path 20000;
          fi;
        done
  # 빌드 전 실행할 명령어
  pre_build:
    commands:
  # gradlew 실행 권한 부여
    - chmod +x gradlew
  build:
    commands:
  # gradlew을 사용하여 프로젝트 빌드
    - ./gradlew build
  # 빌드 후 생성된 파일 처리
  artifacts:
  # 빌드된 파일의 경로를 지정
    files: '**/*'
  # 파일의 기본 디렉토리 지정
    base-directory: 'build/libs'
  discard-paths: yes

```

## 2. AWS CodePipeline

### 1. 소스

## 소스

### 소스 공급자

파이프라인의 입력 아티팩트를 저장한 위치입니다. 공급자를 선택한 후 연결 세부 정보를 제공하십시오.

GitHub(버전 1)

AWS CodePipeline에 GitHub 리포지토리 액세스 권한을 부여합니다. 그러면 AWS CodePipeline이 GitHub에서 파이프라인으로 커밋을 업로드할 수 있습니다.

연결됨

✔ 계정을 성공적으로 인증했습니다.



**GitHub(버전 1) 작업은 권장하지 않습니다.**

선택한 작업은 OAuth 앱을 사용하여 GitHub 리포지토리에 액세스합니다. 이는 더 이상 권장되는 방법이 아닙니다. 대신 GitHub(버전 2) 작업을 선택하고 연결을 생성하여 리포지토리에 액세스합니다. 연결은 GitHub Apps를 사용하여 인증을 관리하고 다른 리소스와 공유할 수 있습니다. [자세히 알아보십시오](#)

### 리포지토리

GoManDoo/library

### 브랜치

master

### 변경 감지 옵션

소스 코드에서 변경이 발생하면 파이프라인을 자동으로 시작하는 감지 모드를 선택합니다.



**GitHub 웹훅(권장)**

GitHub의 웹훅을 사용하여, 변경이 발생할 때 파이프라인을 자동으로 시작합니다.



**AWS CodePipeline**

AWS CodePipeline을 사용하여 정기적으로 변경 사항을 확인합니다.

## 2. 빌드 스테이지 추가

위에서 생성한 AWS CodeBuild 선택

## 빌드 스테이지 추가 정보

Step 3 of 5

### 빌드 - 선택 사항

#### 빌드 공급자

빌드 프로젝트의 도구입니다. 운영 체제, 빌드 사양 파일, 출력 파일 이름 등 빌드 아티팩트 세부 정보를 제공하십시오.

AWS CodeBuild ▼

#### 리전

아시아 태평양 (서울) ▼

#### 프로젝트 이름

AWS CodeBuild 콘솔에서 이미 생성한 빌드 프로젝트를 선택합니다. 또는 AWS CodeBuild 콘솔에서 빌드 프로젝트를 생성한 후 이 작업으로 돌아옵니다.

🔍 lala\_project ✕

또는

프로젝트 생성 [🔗](#)

#### 환경 변수 - 선택 사항

CodeBuild 환경 변수의 키, 값 및 유형을 선택합니다. 값 필드에서 CodePipeline에서 생성된 변수를 참조할 수 있습니다. [자세히 알아보십시오](#) [🔗](#)

환경 변수 추가

#### 빌드 유형



단일 빌드

단일 빌드를 트리거합니다.



배치 빌드

여러 빌드를 단일 실행으로 트리거합니다.

취소

이전

빌드 스테이지 건너뛰기

다음

## 3. CodePipeline 실행

# lalaveFile

파이프라인 유형: **V2**

🟢 **Source** 성공

파이프라인 실행 ID: [c4969012-4363-4ca4-885d-d20144844a4b](#)

Source



[GitHub\(버전 1\)](#)

🟢 성공 - 9시간 전

[a665ab4c](#)

[a665ab4c](#) Source: Update buildspec.yml

↓  
 전환 비활성화

🟢 **Build** 성공

파이프라인 실행 ID: [c4969012-4363-4ca4-885d-d20144844a4b](#)

Build



[AWS CodeBuild](#)

🟢 성공 - 9시간 전

[세부 정보](#)

[로그 보기](#)