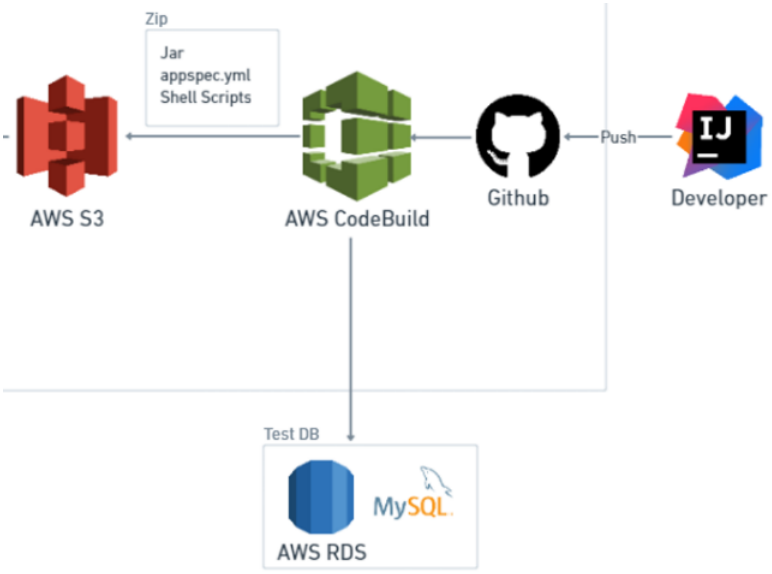
AWS 내부 구축 내용

`Continuous Integration`



- `테스트를 위한 환경 변수 설정`

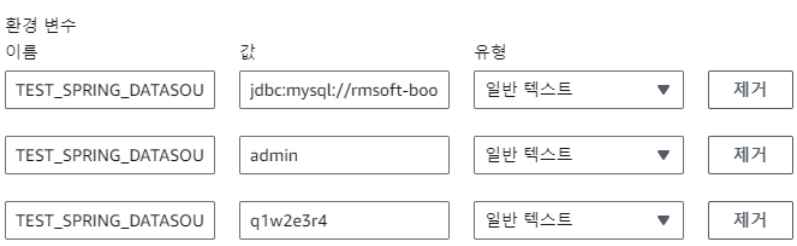
'도서관리 시스템'은 MyBatis를 사용하고 있었기에 테스트 시, MySQL DB를 요구되었습니다.

때문에 운영용 DB가 아닌 테스트용 DB를 추가로 생성해야 했고
AWS RDS MySQL을 추가로 만들어 테스트 코드에서 사용할 수 있도록 만들었습니다.

```
# datasource
spring.datasource.url=${TEST_SPRING_DATASOURCE_URL}
spring.datasource.username=${TEST_SPRING_DATASOURCE_USERNAME}
spring.datasource.password=${TEST_SPRING_DATASOURCE_PASSWORD}

# mybatis
mybatis.mapper-locations=classpath:/mappers/*.xml
mybatis.configuration.map-underscore-to-camel-case=true

# jwt
jwt.secret.key=7Iqk7YyM66W07YOA7L2U65Sp7YG065+9U3ByaW5nórCV7J2Y7Yqc7YSw7LWc7JuQ67mI7J6F64uI64ukLg==

# admin key
security.admin.key=12345
```

이것을 위해 CI 서버에 환경 변수를 설정할 필요가 있었고

CodeBuild 대시보드에서 위 환경 변수를 설정했습니다.



- `buildspec.yml 내용 구성`

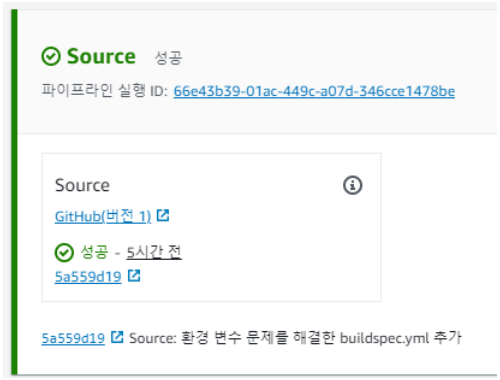
```yaml
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto17
  build:
    commands:
      - echo "Building Spring Boot Application with Gradle"
      - chmod +x ./gradlew
      - ./gradlew build
  post_build:
    commands:
      - echo "Build completed"
artifacts:
  files:
    - build/libs/*.jar
    - appspec.yml
    - scripts/*.sh
```

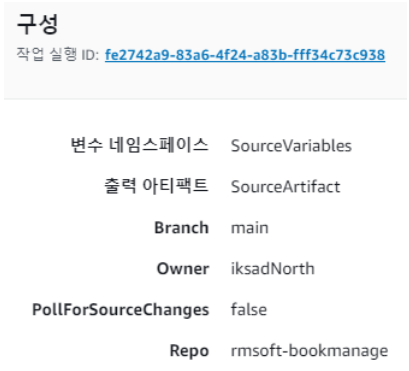
어플리케이션은 Gradle가 사용되었기 때문에 테스트와 빌드 과정 역시 Gradle로 진행했습니다.

실행 권한을 chmod로 부여하고 Test + Bootjar 과정인
build task를 이용해서 CI 과정을 진행했습니다.

'**빌드의 결과물**'과 **appspec.yml**, **shell script**들을 ZIP 파일로 압축하고
S3 Bucket에 저장하는 방식으로 구축했습니다.

- `CodePipeline을 이용한 코드 업데이트 감지`

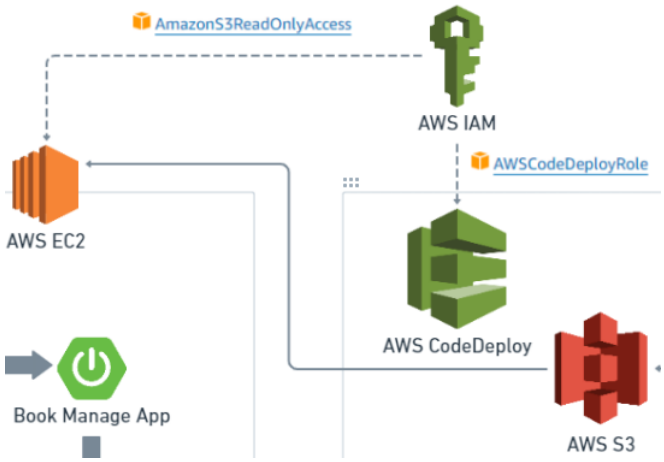




CodePipeLine을 이용해서 `iksadNorth/rmsoft-bookmanage` 코드 저장소의

`Main Branch`에 코드가 갱신될 때마다 CI 과정이 진행되도록 구성했습니다.

`Continuous Deployment`



- `appspec.yml 내용 구성`

```yaml
version: 0.0
os: linux
files:
  - source:  /
    destination: /home/ubuntu/
    overwrite: yes

hooks:
  ApplicationStop:
    - location: scripts/kill_old_process.sh
      timeout: 60
      runas: root
  ApplicationStart:
    - location: scripts/deploy.sh
      timeout: 60
      runas: root
    - location: scripts/set_port_forwarding.sh
      timeout: 60
      runas: root
```

1. S3 Bucket에 담긴 Jar 파일, Shell Script를 `/home/ubuntu` 경로에 저장합니다.
2. Jar 파일을 실행하기 이전에
   기존에 실행되는 프로세스를 중지시킵니다.
   해당 작업은
   `kill_old_process.sh`가 수행합니다.
3. 새로운 프로세스를 가동하기 위해 `deploy.sh` 스크립트를 구동합니다.
4. 그런 다음 HTTP로 들어오는 요청을 8080 포트의 프로그램으로 포워딩시키는 `set_port_forwarding.sh` 스크립트를 구동시킵니다.

각 Shell Script 내용들을 설명하기에 다소 구구절절한 면이 있어서 아래 링크로만 남겨드립니다.

https://github.com/iksadNorth/rmsoft-bookmanage/tree/main/scripts

---

- `운영 서버의 환경 변수 설정`

실서비스를 위한 서버인 EC2 서버에서도 환경 변수 설정이 요구되었습니다.

때문에 환경 변수를 설정하는 Shell Script를 추가로 미리 구성해놓았습니다.

```
ubuntu@ip-172-31-36-110:~$ ls -al scripts/
total 32
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 21 07:51 .
drwxr-x--- 6 ubuntu ubuntu 4096 Dec 21 13:49 ..
-rw-rw-r-- 1 root   root    231 Dec 21 07:48 deploy.sh
-rw-rw-r-- 1 root   root    216 Dec 21 07:48 install_codedeploy_agent.sh
-rw-rw-r-- 1 root   root    156 Dec 21 07:48 install_java.sh
-rw-rw-r-- 1 root   root     88 Dec 21 07:48 kill_old_process.sh
-rwxrwxr-x 1 ubuntu ubuntu  373 Dec 20 15:53 set_env.sh
-rw-rw-r-- 1 root   root     79 Dec 21 07:48 set_port_forwarding.sh
```

```
export SPRING_DATASOURCE_URL=jdbc:mysql://rmsoft-book-management.crmuxpc0pbxh.ap-northeast-2.rds.amazonaws.com:3306/bookmanage;
export SPRING_DATASOURCE_USERNAME=admin;
export SPRING_DATASOURCE_PASSWORD=q1w2e3r4;
export JWT_SECRET_KEY=7Iqk7YyM66W07YOA7L2U65Sp7YG065+9U3ByaW5n6rCV7J2Y7Yqc7YSw7LWc7JuQ67mI7J6F64uI64uKLg==;
export SECURITY_ADMIN_KEY=awgd4wdsgerh1asdf7egDa4f;
```

set_env.sh

해당 환경 변수 스크립트가 서버 생성 시점에만 작동된다면
이후 환경 변수 변경에 대처하기 어렵기 때문에 아예 새로운 프로세스가 실행되기 직전에
실행되도록
`deploy.sh` 를 구성했습니다.

```
ROOT_DIR=/home/ubuntu;

sudo chmod +x $ROOT_DIR/build/libs/rmsoft-0.0.1-SNAPSHOT.jar
source $ROOT_DIR/scripts/set_env.sh
nohup java -jar $ROOT_DIR/build/libs/rmsoft-0.0.1-SNAPSHOT.jar >> $ROOT_DIR/nohup.out 2> $ROOT_DIR/nohup.err &
```

deploy.sh
