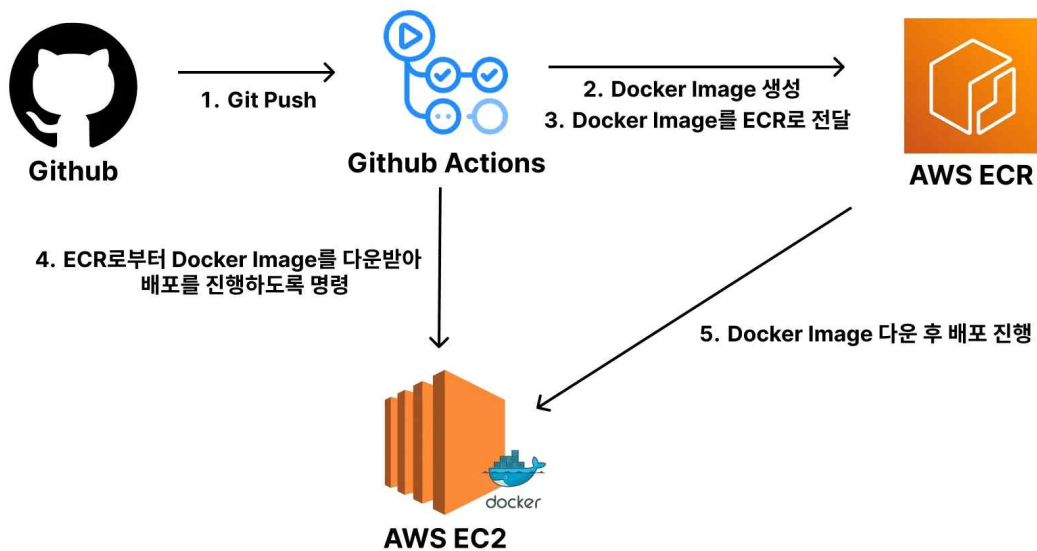


제 목	05장 Docker + CI/CD 적용하기	
상세내용	Docker + 백엔드(Spring Boot) 프로젝트에 CI/CD 적용하기	

방법 4 - 컨테이너 기반의 프로젝트에서 많이 쓰는 CI/CD 구축 방법 (Docker)

✓ 전체적인 흐름



✓ 장점

Docker 기반으로 서비스를 운영할 때, 가장 간단하게 구성할 수 있는 인프라 구조이다.

✓ 단점

무중단 배포를 구현하거나 여러 EC2 인스턴스에 배포를 해야 하는 상황이라면, 직접 Github Actions에 스크립트를 작성해서 구현해야 한다. 직접 구현을 해보면 알겠지만 생각보다 꽤 복잡하다.

✓ 이 방법은 언제 주로 쓰는 지

- 컨테이너 기반으로 인프라를 구성했을 때 이 방법을 많이 활용한다.
- 서버를 여러 대 운영하고 있지 않을 정도의 소규모 프로젝트 일 때 주로 활용한다.

[실습] EC2에 Docker 설치, ECR 셋팅하기

✓ 1. Ubuntu에서 Docker, Docker Compose 설치하기

```
$ sudo apt-get update && \
    sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common && \
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - && \
    sudo apt-key fingerprint 0EBFCD88 && \
    sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" && \
    sudo apt-get update && \
    sudo apt-get install -y docker-ce && \
    sudo usermod -aG docker ubuntu && \
    sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose && \
    sudo chmod +x /usr/local/bin/docker-compose && \
    sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

# 잘 설치됐는 지 확인
$ docker -v # Docker 버전 확인
$ docker compose version # Docker Compose 버전 확인
```

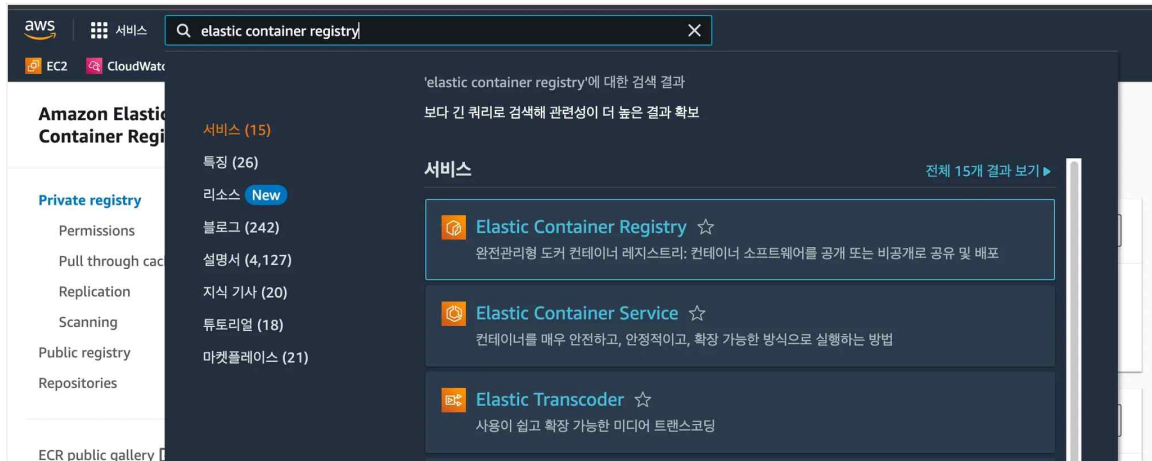
✓ 2. Github Actions의 IAM에 권한 추가

AmazonEC2ContainerRegistryFullAccess 권한 추가하기

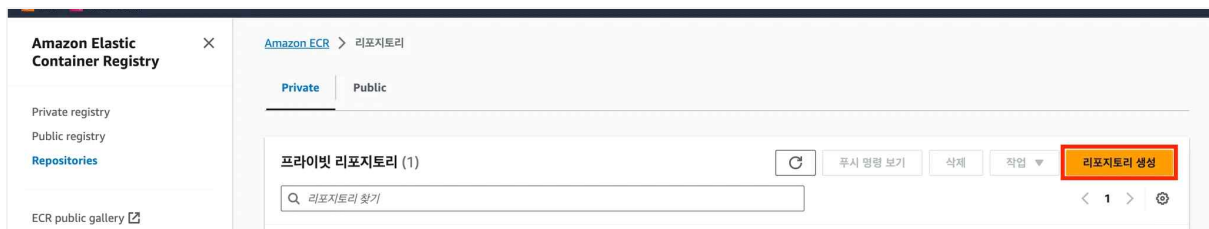


✓ 3. ECR(Elastic Container Registry) 만들기

1. 리스트



2. 리스트



3. 리스트

일반 설정

표시 여부 설정

정보

리포지토리에 대한 가시성 설정을 선택합니다.

☒ 프라이빗

액세스는 IAM 및 리포지토리 정책 권한에 의해 관리됩니다.

☐ 퍼블릭

이미지 풀에 대해 공개적으로 표시되고 액세스할 수 있습니다.

리포지토리 이름

간결한 이름을 제공합니다. 개발자는 이름으로 리포지토리 콘텐츠를 식별할 수 있어야 합니다.

002177417362.dkr.ecr.ap-northeast-2.amazonaws.com/

instagram-server

최대 256자 중 16자(최소 2자 이상) The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

태그 변경 불가능

정보

동일한 태그를 사용하는 후속 이미지 푸시가 이미지 태그를 덮어쓰지 않도록 방지하려면 [태그 변경 불가능]을 활성화합니다. 이미지 태그를 덮어쓰려면 [태그 변경 불가능]을 비활성화합니다.

☐ 비활성화됨

리포지토리가 생성되면 해당 리포지토리의 가시성 설정을 변경할 수 없습니다.

[실습] 컨테이너 기반의 프로젝트에서 많이 쓰는 CI/CD 구축 방법

✔ 이전 실습했던 내용 정리

- 서버 종료
- 프로젝트 폴더 삭제

✔ 1. Docker 기반으로 프로젝트 수정하기

1. Dockerfile 작성하기

Dockerfile

```
FROM eclipse-temurin:17-jdk-alpine
COPY ./build/libs/*SNAPSHOT.jar project.jar
ENTRYPOINT ["java", "-jar", "project.jar"]
```

✔ 2. EC2가 Private ECR에 접근할 수 있게 셋팅하기

1. Amazon ECR Docker Credential Helper 설치하기

▶ <https://tinyurl.com/27nvvpdz>

```
# Ubuntu일 경우
$ sudo apt update
$ sudo apt install amazon-ecr-credential-helper
```

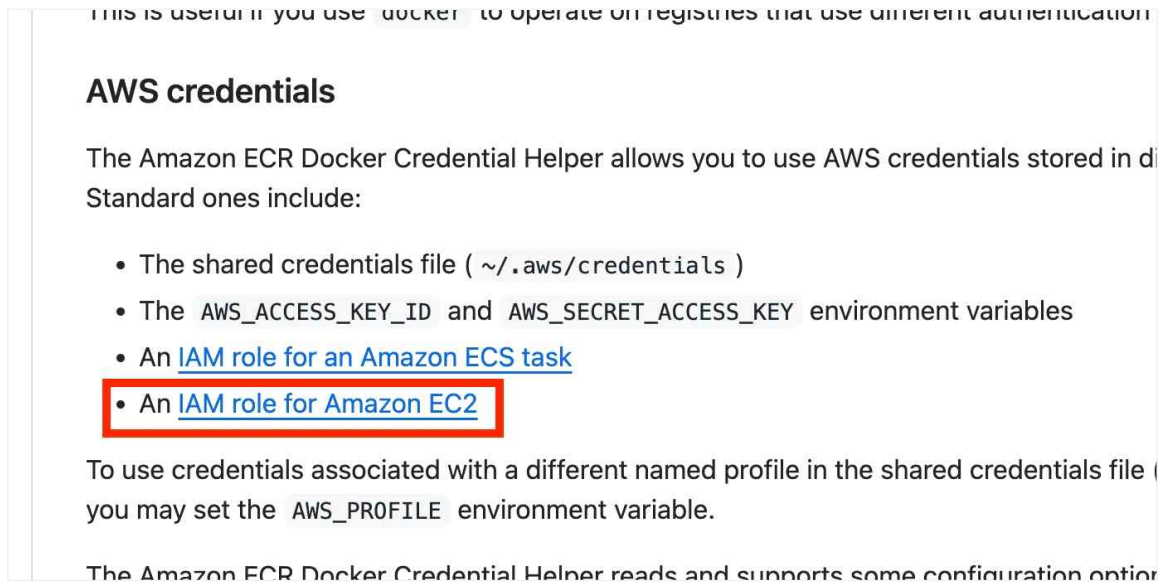
2. Configuration 설정하기

~ 경로에서 **.docker**라는 폴더 만들고, **config.json** 파일 만들어서 위와 같이 작성해라.

~/docker/config.json

```
{
    "credsStore": "ecr-login"
}
```

3. IAM Role을 활용해 EC2가 ECR에 접근할 수 있게 권한 부여하기



EC2에 연결되어 있는 IAM Role에 **AmazonEC2ContainerRegistryFullAccess** 정책 추가하기

✓ 3. Docker 기반 CI/CD 구축하기

1. Github Actions 파일 작성하기

`.github/workflows/deploy.yml`

```
name: Deploy To EC2

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Github Repository 파일 불러오기
        uses: actions/checkout@v4

      - name: JDK 17버전 설치
        uses: actions/setup-java@v4
        with:
```

```
distribution: temurin
java-version: 17

- name: application.yml 파일 만들기
  run: echo "${{ secrets.APPLICATION_PROPERTIES }}" > ./src/main/resources/application.yml

- name: 테스트 및 빌드하기
  run: ./gradlew clean build

- name: AWS Resource에 접근할 수 있게 AWS credentials 설정
  uses: aws-actions/configure-aws-credentials@v4
  with:
    aws-region: ap-northeast-2
    aws-access-key-id: "${{ secrets.AWS_ACCESS_KEY_ID }}"
    aws-secret-access-key: "${{ secrets.AWS_SECRET_ACCESS_KEY }}"

- name: ECR에 로그인하기
  id: login-ecr
  uses: aws-actions/amazon-ecr-login@v2

- name: Docker 이미지 생성
  run: docker build -t instagram-server .

- name: Docker 이미지에 Tag 붙이기
  run: docker tag instagram-server "${{ steps.login-ecr.outputs.registry }}/instagram-server:latest"

- name: ECR에 Docker 이미지 Push하기
  run: docker push "${{ steps.login-ecr.outputs.registry }}/instagram-server:latest"

- name: SSH로 EC2에 접속하기
  uses: appleboy/ssh-action@v1.0.3
  with:
    host: "${{ secrets.EC2_HOST }}"
    username: "${{ secrets.EC2_USERNAME }}"
    key: "${{ secrets.EC2_PRIVATE_KEY }}"
    script_stop: true
```

```

script: |
    docker stop instagram-server || true
    docker rm instagram-server || true
    docker pull ${ steps.login-ecr.outputs.registry }/instagram-server:latest
    docker run -d --name instagram-server -p 8080:8080 ${ steps.login-ecr.outputs.registry }/instagram-server:latest

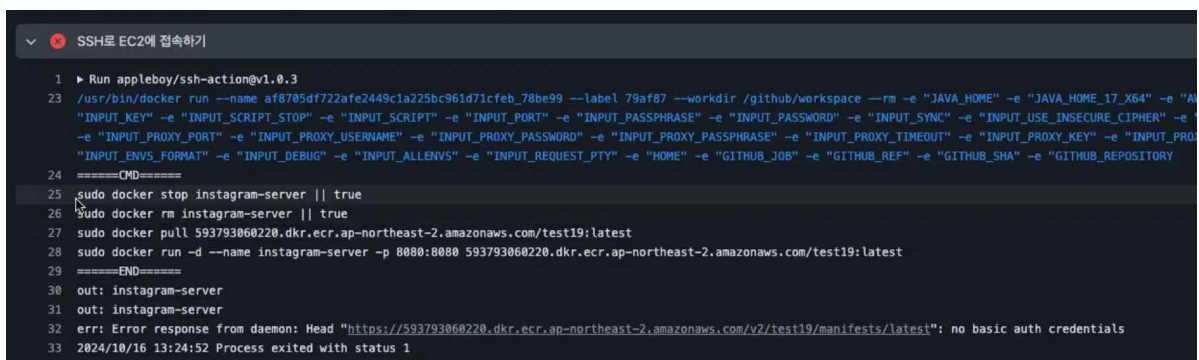
```

- 맨 마지막 부분의 코드에서 sudo를 붙이면 no basic auth credentials 에러가 발생하면서 CI/CD가 실패한다. 따라서 sudo를 작성하지 않도록 주의하자.

```

...
- name: SSH로 EC2에 접속하기
  uses: appleboy/ssh-action@v1.0.3
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_PRIVATE_KEY }
    script_stop: true
    script: |
      sudo docker stop instagram-server || true
      sudo docker rm instagram-server || true
      sudo docker pull ${ steps.login-ecr.outputs.registry }/instagram-server:latest
      sudo docker run -d --name instagram-server -p 8080:8080 ${ steps.login-ecr.outputs.registry }/instagram-server:latest

```



```

SSH로 EC2에 접속하기
1 ▶ Run appleboy/ssh-action@v1.0.3
23 /usr/bin/docker run --name af8785df722afe2449c1a225bc961d71cfeb_78be99 --label 79af87 --workdir /github/workspace --rm -e "JAVA_HOME" -e "JAVA_HOME_17_X64" -e "INPUT_KEY" -e "INPUT_SCRIPT_STOP" -e "INPUT_SCRIPT" -e "INPUT_PORT" -e "INPUT_PASSPHRASE" -e "INPUT_PASSWORD" -e "INPUT_SYNC" -e "INPUT_USE_INSECURE_CIPHER" -e "INPUT_PROXY_PORT" -e "INPUT_PROXY_USERNAME" -e "INPUT_PROXY_PASSWORD" -e "INPUT_PROXY_PASSPHRASE" -e "INPUT_PROXY_TIMEOUT" -e "INPUT_PROXY_KEY" -e "INPUT_PROXY_FORMAT" -e "INPUT_DEBUG" -e "INPUT_ALLENVS" -e "INPUT_REQUEST_PTY" -e "HOME" -e "GITHUB_JOB" -e "GITHUB_REF" -e "GITHUB_SHA" -e "GITHUB_REPOSITORY"
24 =====
25 sudo docker stop instagram-server || true
26 sudo docker rm instagram-server || true
27 sudo docker pull 593793060220.dkr.ecr.ap-northeast-2.amazonaws.com/test19:latest
28 sudo docker run -d --name instagram-server -p 8080:8080 593793060220.dkr.ecr.ap-northeast-2.amazonaws.com/test19:latest
29 =====
30 out: instagram-server
31 out: instagram-server
32 err: Error response from daemon: Head "https://593793060220.dkr.ecr.ap-northeast-2.amazonaws.com/v2/test19/manifests/latest": no basic auth credentials
33 2024/10/16 13:24:52 Process exited with status 1

```

2. CI/CD 과정이 잘 작동하는 지 확인하기