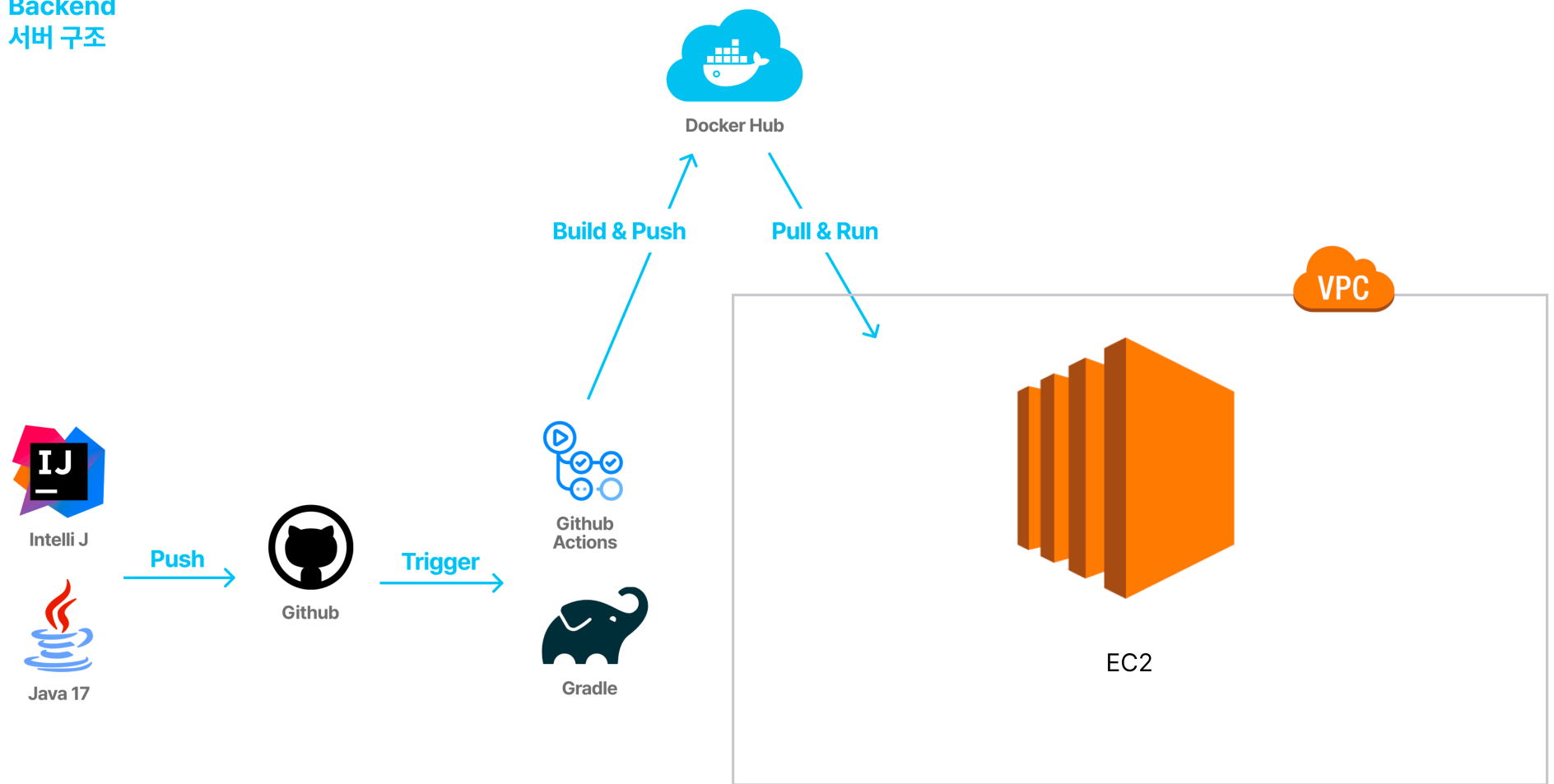


Backend 서버 구조



기본적인 구조는 이렇게 생겼고, github Action, docker를 중심으로 학습을 했습니다..!

1. 도커 - Repository 만들기

ungchun / sejongmate Contains: Image Last pushed: 2 days ago	Inactive	0	8	Public
ungchun / ungchun_docker Contains: No content Last pushed: a month ago	Inactive	0	0	Public

음.. 여러 블로그들을 참고했는데 이게 필요한지 안필요한지는 블로그마다 달라서 저는 우선 만들었습니다..

2. Github-Actions 스크립트 파일 생성

Github repository - Actions - Java with Gradle 선택을 하고 나서, yml 파일에 스크립트를 작성을 해야하는데, 프로젝트마다 세팅해야하는게 다 달라서 본인의 프로젝트에 맞게 작성해주시면 될 거 같습니다. 우선 저는 연습이라 큰 세팅없이 기본만 작성한 코드입니다.

```
name: Java CI with Gradle

on:
  push:
    branches: [ "main" , "dev" ]
  pull_request:
    branches: [ "main" , "dev" ]

permissions:
```

```
contents: read
```

```
jobs:
```

```
  build:
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

```
      - uses: actions/checkout@v3
```

```
      - name: Set up JDK 17
```

```
        uses: actions/setup-java@v3
```

```
        with:
```

```
          java-version: '17'
```

```
          distribution: 'temurin'
```

```
      - name: make application.properties
```

```
        run: |
```

```
          cd ./src/main/resources
```

```
          touch ./application.properties
```

```
          echo "${{ secrets.APPLICATION_PROD }}" > ./application.properties
```

```
      - name: Grant execute permission for gradlew
```

```
        run: chmod +x gradlew
```

```
      - name: Build with Gradle
```

```
        run: ./gradlew build -x test
```

```
      - name: Docker build
```

```
        run: |
```

```
          docker login -u "${{ secrets.DOCKER_USERNAME }}" -p "${{ secrets.DOCKER_PASSWORD }}"
```

```
          docker build -t app .
```

```
          docker tag app "${{ secrets.DOCKER_USERNAME }}/sejongmate:latest"
```

```
          docker push "${{ secrets.DOCKER_USERNAME }}/sejongmate:latest"
```

```
      - name: Deploy
```

```
        uses: appleboy/ssh-action@master
```

```
        with:
```

```
          host: "${{ secrets.HOST }}" # EC2 인스턴스 퍼블릭 DNS
```

```
          username: ec2-user
```

```
          key: "${{ secrets.PRIVATE_KEY }}" # pem 키
```

```
          # 도커 작업
```

```
          script: |
```

```
            docker pull "${{ secrets.DOCKER_USERNAME }}/sejongmate:latest"
```













```
            docker stop $(docker ps -a -q)
```

```
            docker run -d --log-driver=syslog -p 8080:8080 "${{ secrets.DOCKER_USERNAME }}/sejongmate:latest"
```

```
            docker rm $(docker ps --filter 'status=exited' -a -q)
```

```
            docker image prune -a -f
```

위에서 사용되는 secrets 변수들을 github repository를 통해 넣어줄 수 있습니다. repository에서 Settings → Secrets and variables → Actions로 가서..

Repository secrets			
	DOCKER_PASSWORD	Updated last month	 
	DOCKER_USERNAME	Updated last month	 
	HOST	Updated last month	 
	PRIVATE_KEY	Updated last month	 

- DOCKER_USERNAME: 도커 ID
- DOCKER_PASSWORD: 도커 계정 패스워드
- HOST : EC2 인스턴스 퍼블릭 DNS
- PRIVATE_KEY : pem 키 (텍스트 편집기로 열고 전체 복사 붙여넣기)

3. Dockerfile 만들기

그 다음에는 Dockerfile을 만들어야합니다. 프로젝트 root 위치에 만들어주시고,

```
FROM openjdk:17-jdk-slim
ADD /build/libs/*.jar app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

이렇게 작성을 했습니다. Dockerfile 또한 프로젝트마다 설정을 다르게 해야하니, 본인의 프로젝트에 맞게 작성해주세요.

4. AWS EC2 인스턴스 접속

먼저 인스턴스 생성 + 키페어생성 + 보안그룹 설정 완료가 된 상태여야 합니다.

키페어가 있는 폴더 내부에서 아래의 명령어를 통해 ssh 접속을 합니다.

```
ssh -i "키페어 파일이름" "퍼블릭 DNS 주소"
ssh -i sunghun_key.pem ec2-user@ec2-xx-xx-xxx-xxx.ap-northeast-2.compute.amazonaws.com
```

이후 아래의 과정을 통해 docker, docker-compose 설치를 합니다.

```
//도커 설치
sudo yum install docker -y

//도커 실행
sudo service docker start

//도커 상태 확인
systemctl status docker.service

//도커 관련 권한 추가
sudo chmod 666 /var/run/docker.sock
docker ps

//최신 버전 docker-compose 설치
sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
```

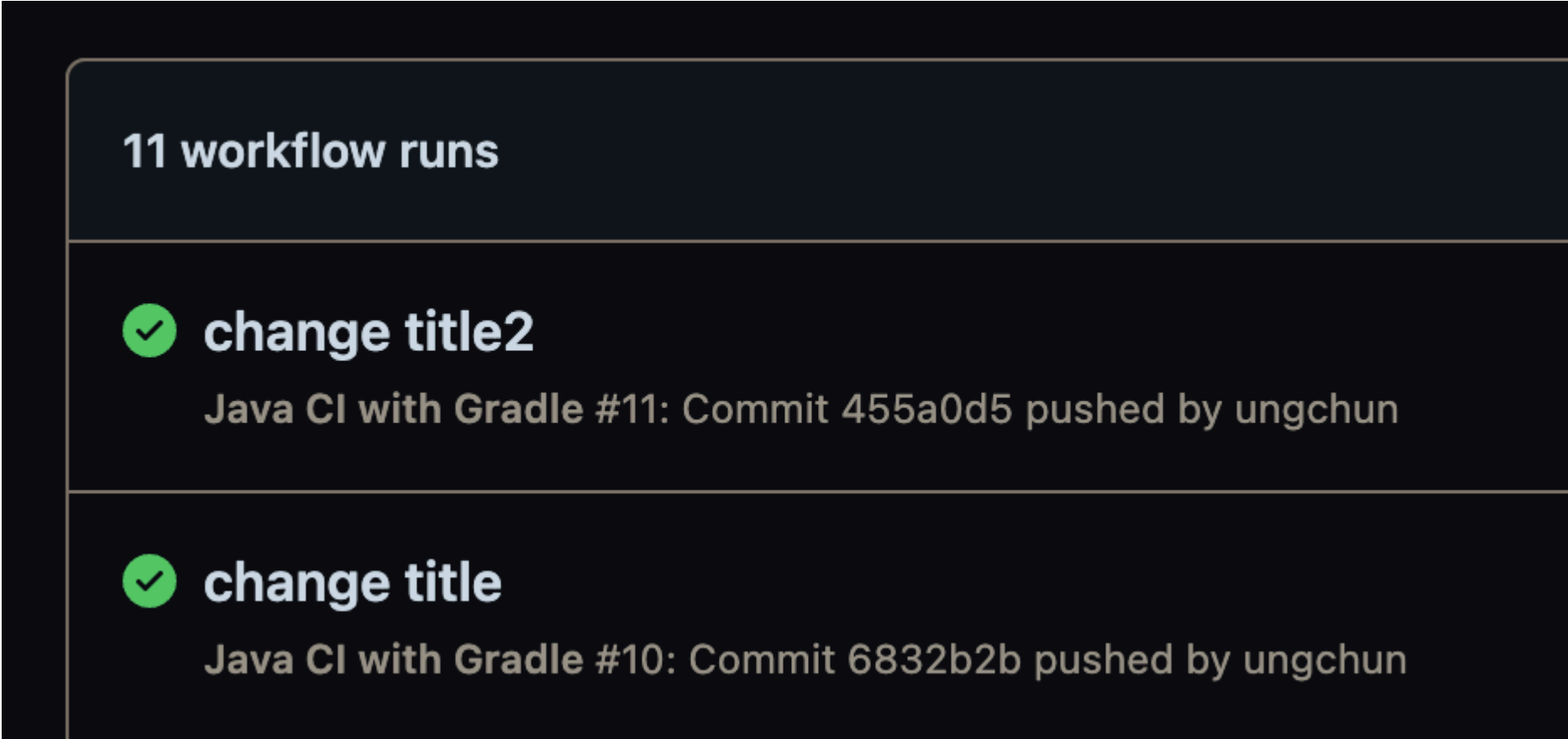
```
//권한 추가
sudo chmod +x /usr/local/bin/docker-compose

//버전 확인
docker-compose --version
```

그리고 Java 설치 + Git 설치 + 프로젝트 clone 후 서버배포까지 완료가 된 상태에서, 서버를 빌드 시켜줍니다.

- sudo chmod 777./gradlew → 빌드 실행 권한 주기
- ./gradlew build → 빌드
- cd build → \$ cd libs → \$ java -jar 자신의 프로젝트.jar

그리고 저는 로컬 스프링에서 text를 change로 변경을 하고 push만 했습니다.

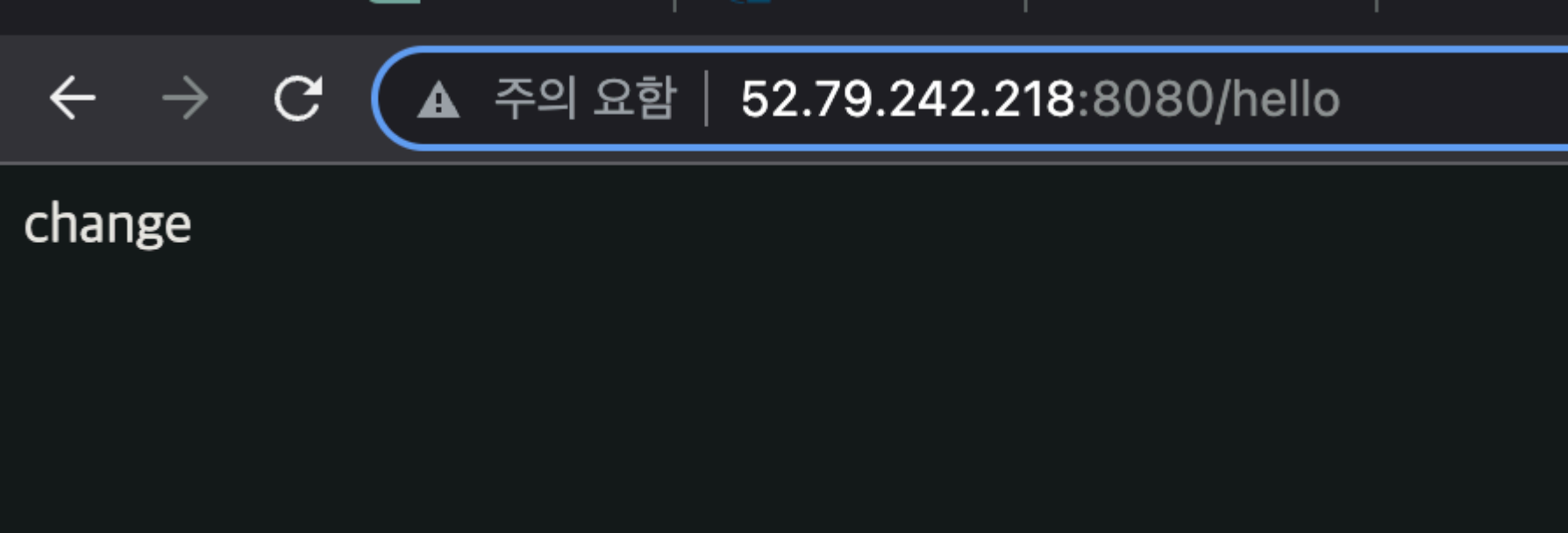


그럼 요렇게 github action이 돌아가고 성공을 하고나면..

```
[ec2-user@ip-172-31-33-18 githubActionDockerPractice]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                NAMES
[ec2-user@ip-172-31-33-18 githubActionDockerPractice]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                NAMES
beb6deb77561   ungchun/sejongmate:latest           "java -Djava.securit..." 9 seconds ago  Up 7 seconds  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp  laughing_sah
```

ec2 에서 docker ps로 확인해보면 docker image 파일이 들어온 걸 확인할 수 있습니다!!! (신기함)

그리고 서버를 다시 돌리고 나서 확인을 해보면..



따로 서버쪽에서 github에 새로 올라간 프로젝트를 pull 받지 않아도 알아서 새로 올라간 프로젝트로 적용되어 있는 화면을 볼 수 있었습니다!!!!