‘토키’ 배포 가이드

0. EC2 ufw 설정하기

```
1. 처음 ufw 설정 시 실수로 ssh접속이 안되는 경우를 방지하기 위해
   ssh 터미널을 여유있게 2~3개 연결해 놓는다.

2. ufw 상태 확인
$ sudo ufw status
Status : inactive

3. 사용할 포트 허용하기 (ufw inactive 상태)
$ sudo ufw allow 22

3-1 등록한 포트 조회하기 (ufw inactive 상태)
$ sudo ufw show added
Added user rules (see 'ufw status' for running firewall):
ufw allow 22

4. ufw 활성화 하기
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y

4.1 ufw 상태 및 등록된 rule 확인하기
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22 (v6)                    ALLOW IN    Anywhere (v6)

5. 새로운 터미널을 띄워 ssh 접속해 본다.
C:\> ssh -i 팀.pem ubuntu@팀.p.ssafy.io

6. ufw 구동된 상태에서 80 포트 추가하기
$ sudo ufw allow 80

6-1. 80 포트 정상 등록되었는지 확인하기
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 80                         ALLOW IN    Anywhere
[ 3] 22 (v6)                    ALLOW IN    Anywhere (v6)
[ 4] 80 (v6)                    ALLOW IN    Anywhere (v6)

6-2. allow 명령을 수행하면 자동으로 ufw에 반영되어 접속이 가능하다.

7. 등록한 80 포트 삭제 하기
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 80                         ALLOW IN    Anywhere
[ 3] 22 (v6)                    ALLOW IN    Anywhere (v6)
[ 4] 80 (v6)                    ALLOW IN    Anywhere (v6)
```

```
7-1. 삭제할 80 포트의 [번호]를 지정하여 삭제하기
      번호 하나씩 지정하여 삭제한다.
$ sudo ufw delete 4
$ sudo ufw delete 2
$ sudo ufw status numbered  (제대로 삭제했는지 조회해보기)
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22 (v6)                    ALLOW IN    Anywhere (v6)

7-2 (중요) 삭제한 정책은 반드시 enable을 수행해야 적용된다.
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y입력


기타
- ufw 끄기
$ sudo ufw disable
```

0-1. Jenkins 포트 8080 및 프론트 배포 용 80 포트 열기

```
sudo ufw allow 80
sudo ufw allow 8080
```

1. 도커 설치 및 도커 컴포즈 설치

a. 패키지 업데이트

```
sudo apt-get update
```

b. 필요한 패키지 설치

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

c. gpg키 설정

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-arc
hive-keyring.gpg
```

d. 도커 저장소 설정

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://do
wnload.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

e. 도커 설치

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

f. 도커 실행

```
sudo service docker start
# 도커 실행 후 sudo docker ps로 정상 작동하는 지 확인
```

g. 도커 컴포즈 설치

```
sudo curl -L https://github.com/docker/compose/releases/download/v2.4.1/docker-compose-`uname -s`-`uname -m` -
o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo usermod -aG docker $USER
```

```
# 권한을 준 이후에는 로그아웃하여 다시 로그인해야 적용이 됨.

# docker-compose --version를 실행하여 정상 설치되었는지 확인하기
```

2. Jenkins 설치

a. 볼륨 디렉토리 생성

```
cd /home/ubuntu && mkdir jenkins-data
```

b. 외부에서 접속할 포트 오픈 및 상태 확인

```
sudo ufw allow 8080/tcp
sudo ufw reload
sudo ufw status
```

c. 젠킨스 컨테이너 생성 및 구동

```
# docker에 대해 유저에게 권한을 부여했기 때문에 앞으로는 sudo를 제외해도 무방

docker run -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock --name jenkins jenkins/jenkins:lts
```

d. 초기 비밀번호 확인

```
sudo docker logs jenkins
```

e. 젠킨스 url 접속 및 계정 설정

```
http://주소:8080
```

f. 젠킨스 추가 플러그인 설치

```
# Jenkins 메인 페이지 -> Jenkins 관리 -> Plugins -> 아래의 플러그인 추가 설치

- Docker
- Docker pipeline
- Docker API
- Gitlab
- Generic Webhook Trigger Plugin
- SSH Agent
```

2-1. Jenkins credential 설정

jenkins 관리 → Credential

a. Gitlab Token 등록

```
# GitLab에서 Access Token 발급
Kind : Username with password
Scope : Global
Username : Gitlab 아이디
Password : 발급받은 Access Token
ID : gitlab
```

b. docker compose 등록 (FE, BE 파이프라인 분리를 위해 독립적으로)

```
Kind : Secret file
File : docker-compose.yml
ID : docker-compose

Kind : Secret file
File : docker-compose-fe.yml
ID : docker-compose-fe
```

c. application.properties 등록

```
Kind : Secret file
File : application.properties
ID : application
```

3. 파이프라인

젠킨스 메인 → New Item → type : PipeLine 선택 → 설정하고 저장

Webhooks 설정

- ***Jenkins 설정***

```
1. 젠킨스 빌드 트리거 설정의 GitLab check 및 URL 기록
2. Enabled GitLab triggers -> Push Event 체크
3. 고급 설정에서 Allowed branch에서 Filter branched by regex 체크
4. Source Branch Regex에 배포 브랜치 입력 ex) FE/release, BE/release, DA/release
5. 시스릿 키 생성 및 기록

**시크릿 키의 경우 마지막으로 생성된 키가 유효함**
**WebHooks의 경우에도 FE/BE 독립적으로 파이프라인 구성하기 위해서 따로 생성**n b
```

- ***GitLab 설정***

```
GitLab Repo -> Settings -> WebHooks -> add new WebHooks
URL : 기록한 URL 입력
Secret Token : 기록한 시크릿 키 입력
Trigger : Push events -> All branched check
```

- ***파이프라인 (BE)***

```
pipeline {
    agent any

    stages {

        stage('docker, docker-compose install') {
            steps{
                script {
                    sh  """
                        if ! command -v docker > /dev/null; then
                            curl -fsSL https://get.docker.com -o get-docker.sh
                            sh get-docker.sh
                        fi
                    """
                    sh """
                    export PATH=\$PATH:\$HOME/bin
                    mkdir -p \$HOME/bin
                    if ! command -v docker-compose &> /dev/null
                    then
                        echo "docker-compose could not be found. Installing..."
                        curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-
\$(uname -s)-\$(uname -m)" -o \$HOME/bin/docker-compose
                        chmod +x \$HOME/bin/docker-compose
                    else
                        echo "docker-compose is already installed."
                    fi
                    """
                }
            }

        }

        stage('docker_stop_remove') {
            steps {
```

```
                script {
                    // 실행 중인 컨테이너를 중지
                    sh """
                        if [ \$(docker ps -q -f name=talkie_spring_app_1) ]; then
                            echo "Stopping talkie_spring_app_1 container"
                            docker stop talkie_spring_app_1
                        else
                            echo "talkie_spring_app_1 container is not running"
                        fi
                    """
                }
                script {
                    sh """
                        if [ \$(docker ps -a -q -f name=talkie_spring_app_1) ]; then
                            echo "Removing talkie_spring_app_1 container"
                            docker rm talkie_spring_app_1
                        else
                            echo "talkie_spring_app_1 container is not found"
                        fi

                        if [ \$(docker images -q talkie-project:latest) ]; then
                            echo "Removing talkie-project image"
                            docker rmi -f talkie-project:latest || true
                        else
                            echo "talkie-project image is not found"
                        fi
                    """
                }
            }
        }

        stage('gitlab_clone') {
            steps {
                git branch: 'BE/release/test', credentialsId: 'gitlab', url: 'https://lab.ssafy.com/s11-final/
S11P31E104.git'
                script {
                    sh 'git checkout BE/release/test'

                }
            }
        }

        stage('properties copy'){
            steps{
                withCredentials([file(credentialsId: 'application', variable: 'properties')]) {
                    script {
                        sh 'pwd'
                        sh 'ls'
                        sh 'chmod +r $properties'
                        sh 'chmod -R 777 backend/RealTime/src/main/resources'
                        sh 'cp $properties backend/RealTime/src/main/resources/application.properties'
                        sh 'ls'
                    }
                }

            }
        }

        stage('docker-compose copy'){
            steps{
                withCredentials([file(credentialsId: 'docker-compose', variable: 'composeFile')]) {
                    script {
                        sh 'pwd'
                        sh 'ls'
                        sh 'chmod +r $composeFile'
                        sh 'chmod -R 777 backend/RealTime'
                        sh 'cp $composeFile backend/RealTime/docker-compose.yml'
                        sh 'ls backend/RealTime'
```

```
                }
            }

        }
    }



    stage('build') {
        steps {
            script {
                dir('backend/RealTime') {
                    // Check and remove existing talkie-project image if it exists
                    sh """
                        if [ \$(docker images -q talkie-project:latest) ]; then
                            echo "Removing existing talkie-project:latest image"
                            docker rmi talkie-project:latest || true
                        else
                            echo "talkie-project:latest image not found"
                        fi
                    """
                    sh "chmod +x ./gradlew"
                    sh "./gradlew clean build"
                }
            }
        }
    }

    stage('docker_build') {
        steps {
            script {
                dir('backend/RealTime') {
                    sh "docker build -t talkie-project ." // Dockerfile이 있는 디렉토리에서 이미지 빌드
                }
            }
        }
    }


    stage('docker_deploy') {
        steps {
            script {
                dir('backend/RealTime') {

                    sh "docker-compose -f docker-compose.yml up --build -d"
                }
            }


        }


    }
}
```

- ***파이프라인 (FE)***

```
pipeline {
    agent any

    stages {
        stage('docker, docker-compose install') {
            steps {
                script {
                    sh  """
                        if ! command -v docker > /dev/null; then
                            curl -fsSL https://get.docker.com -o get-docker.sh
```

```
                            sh get-docker.sh
                        fi
                    """
                    sh """
                        export PATH=\$PATH:\$HOME/bin
                        mkdir -p \$HOME/bin
                        if ! command -v docker-compose &> /dev/null; then
                            echo "docker-compose could not be found, installing..."
                            curl -L "https://github.com/docker/compose/releases/download/v2.4.1/docker-compose
-\$(uname -s)-\$(uname -m)" -o \$HOME/bin/docker-compose
                            chmod +x \$HOME/bin/docker-compose
                        else
                            echo "docker-compose is already installed"
                        fi
                    """
                }
            }
        }

        stage('Install Docker and Docker Compose') {
            steps {
                script {
                    sh """
                        if ! command -v docker > /dev/null; then
                            echo "Installing Docker..."
                            curl -fsSL https://get.docker.com -o get-docker.sh
                            sh get-docker.sh
                        fi

                        if ! command -v docker-compose > /dev/null; then
                            echo "Installing Docker Compose..."
                            curl -L "https://github.com/docker/compose/releases/download/v2.4.1/docker-compose
-\$(uname -s)-\$(uname -m)" -o /usr/local/bin/docker-compose
                            chmod +x /usr/local/bin/docker-compose
                        fi
                    """
                }
            }
        }

        stage('docker_stop_remove') {
            steps {
                script {
                    sh """
                        if [ \$(docker ps -q -f name=talkie_frontend) ]; then
                            echo "Stopping talkie_frontend container"
                            docker stop talkie_frontend
                        else
                            echo "talkie_frontend container is not running"
                        fi
                    """
                }
                script {
                    sh """
                        if [ \$(docker ps -a -q -f name=talkie_frontend) ]; then
                            echo "Removing talkie_frontend container"
                            docker rm talkie_frontend
                        else
                            echo "talkie_frontend container is not found"
                        fi
                        if [ \$(docker images -q talkie_frontend) ]; then
                            echo "Removing talkie_frontend image"
                            docker rmi -f talkie_frontend || true
                        else
                            echo "talkie_frontend image is not found"
                        fi
                    """
                }
```

```
            }
        }

        stage('gitlab_clone') {
            steps {
                git branch: 'FE/release', credentialsId: 'gitlab', url: 'https://lab.ssafy.com/s11-final/S11P3
1E104.git'
            }
        }

        stage('docker-compose copy') {
            steps {
                withCredentials([file(credentialsId: 'docker-compose-fe', variable: 'composeFile')]) {
                    script {
                        sh 'pwd'
                        sh 'ls'
                        sh 'chmod +r $composeFile'
                        sh 'chmod -R 777 frontend'
                        sh 'cp $composeFile frontend/docker-compose.yml'
                        sh 'ls frontend'
                    }
                }
            }

        stage('docker_build') {
            steps {
                script {
                    dir('frontend') {
                        sh "docker-compose -f docker-compose.yml up -d --build"
                    }
                }
            }
        }
    }
}
```

- 파이프라인 (DA)

데이터 분석 및 AI 서버의 경우 파이썬 기반의 프로젝트이기 때문에 루트 디렉토리에 requirements.txt 생성 후 빌드 진행

```
pipeline {
    agent any

    stages {

        stage('docker, docker-compose install') {
            steps {
                script {
                    sh """
                        if ! command -v docker > /dev/null; then
                            curl -fsSL https://get.docker.com -o get-docker.sh
                            sh get-docker.sh
                        fi
                    """
                    sh """
                    export PATH=\$PATH:\$HOME/bin
                    mkdir -p \$HOME/bin
                    if ! command -v docker-compose &> /dev/null
                    then
                        echo "docker-compose could not be found. Installing..."
                        curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-
\$(uname -s)-\$(uname -m)" -o \$HOME/bin/docker-compose
                        chmod +x \$HOME/bin/docker-compose
                    else
                        echo "docker-compose is already installed."
                    fi
```

```
                """
            }
        }
    }

    stage('docker_stop_remove') {
        steps {
            script {
                // 실행 중인 컨테이너를 중지
                sh """
                    if [ \$(docker ps -q -f name=fastapi_app) ]; then
                        echo "Stopping fastapi_app container"
                        docker stop fastapi_app
                    else
                        echo "fastapi_app container is not running"
                    fi
                """
            }
            script {
                sh """
                    if [ \$(docker ps -a -q -f name=fastapi_app) ]; then
                        echo "Removing fastapi_app container"
                        docker rm fastapi_app
                    else
                        echo "fastapi_app container is not found"
                    fi

                    if [ \$(docker images -q fastapi-app:latest) ]; then
                        echo "Removing fastapi-app image"
                        docker rmi -f fastapi-app:latest || true
                    else
                        echo "fastapi-app image is not found"
                    fi
                """
            }
        }
    }

    stage('gitlab_clone') {
        steps {
            git branch: 'DA/release', credentialsId: 'gitlab', url: 'https://lab.ssafy.com/s11-final/S11P3
1E104.git'
            script {
                sh 'git checkout DA/release'
            }
        }
    }

    stage('docker-compose copy') {
        steps {
            withCredentials([file(credentialsId: 'docker-compose-da', variable: 'composeFile')]) {
                script {
                    sh 'pwd'
                    sh 'ls'
                    sh 'chmod +r $composeFile'
                    sh 'chmod -R 777 dataAnalysis'
                    sh 'cp $composeFile dataAnalysis/docker-compose.yml'
                    sh 'ls dataAnalysis'
                }
            }
        }
    }

    stage('docker_build') {
        steps {
            script {
                dir('dataAnalysis') {
                    // 기존 fastapi-app 이미지가 있으면 삭제
```

```
                    sh """
                        if [ \$(docker images -q fastapi-app:latest) ]; then
                            echo "Removing existing fastapi-app:latest image"
                            docker rmi fastapi-app:latest || true
                        else
                            echo "fastapi-app:latest image not found"
                        fi
                    """
                    sh "docker build -t fastapi-app ." // Dockerfile이 있는 디렉토리에서 이미지 빌드
                }
            }
        }
    }

    stage('docker_deploy') {
        steps {
            script {
                // 네트워크가 없을 경우 생성
                sh 'docker network create realtime_app_network || true'
                dir('dataAnalysis') {
                    sh "docker-compose -f docker-compose.yml up --build -d"
                }
            }
        }
    }
}
```

4. Nginx 설정

```
sudo apt update
sudo apt install nginx -y
```

nginx 설정 파일 수정

```
sudo vi /etc/nginx/sites-available/default
```

다음과 같은 코드로 수정!

```
server {
    listen 80;
    listen [::]:80;
    server_name k11e104.p.ssafy.io;

    location / {
        return 308 https://k11e104.p.ssafy.io$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name k11e104.p.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/k11e104.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k11e104.p.ssafy.io/privkey.pem;

    location /api/data {
        proxy_pass http://127.0.0.1:8001;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api {
        proxy_pass http://127.0.0.1:8081;
```

```
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location / {
        proxy_pass http://127.0.0.1:5173;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

5. portainer 설치

portainer의 경우 argoCD 처럼 컨테이너 로그를 볼 수 있고 쿠버네티스처럼 한눈에 컨테이너를 쉽게 관리할 수 있는 Tool

a. 사용할 불륨 설정

```
docker  volume create portainer_data
```

b. portainer 도커 실행

```
docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data --restart=alwa
ys portainer/portainer
```

c. 접속 후 테스트

```
http://k11e104.p.ssafy.io
```

6. MQTT 설정

a. 패키치 설치 및 업데이트

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
```

b. Mosquitto 서비스 활성화

```
sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

c. port 허용

```
sudo ufw allow 1883
sudo ufw allow 8883
sudo ufw reload
```