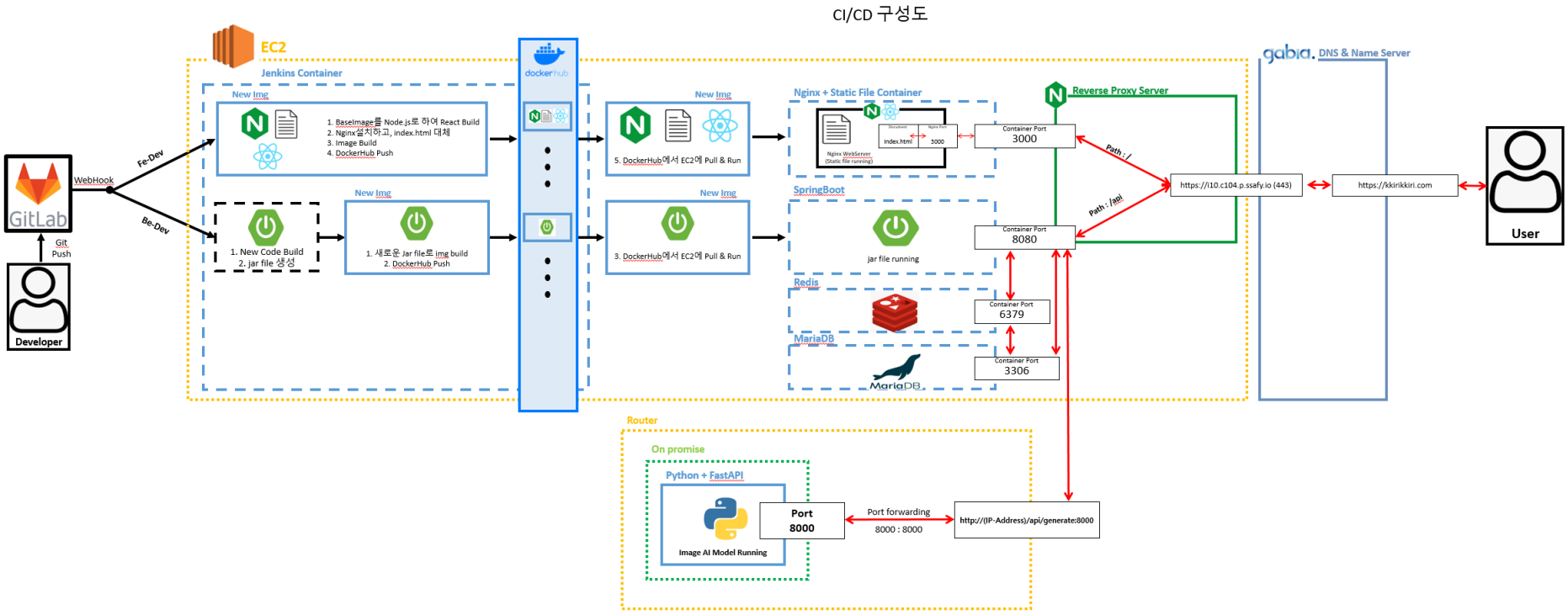
포팅메뉴얼

CI/CD 구성도



FE

Node.js	20.11.1
React	18.2.0
Zustand	4.5.2
iink-ts	1.0.3
openai	4.29.2

BE

JAVA	zulu 17.48.15
Spring Boot	3.2.3
JENKINS	2.450 (docker image)
DB	MariaDB 8.0.35
Image Storage	AWS S3 aws:2.2.6.RELEASE
Web Server	Nginx 1.18.0
IDE	IntelliJ
redis	7.2.4
OS	Ubuntu 20.04.6 LTS
Docker	26.0.0,
Server	AWS EC2

1. Docker 설치

▼ `sudo apt-get update`

- APT(Advanced Package Tool) 패키지 관리자의 패키지 목록을 최신 상태로 업데이트하는 명령어

▼ `sudo apt-get install ca-certificates curl`

- 특정 패키지들을 설치하는 명령어
- ca-certificates : SSL/TLS 연결에 필요한 인증서를 관리하는 데 사용됨
- curl : 커맨드 라인에서 URL을 통해 데이터를 전송하거나 받을 수 있는 도구

▼ `sudo install -m 0755 -d /etc/apt/keyrings`

- `/etc/apt/keyrings` 디렉토리를 생성하는 데 사용됨.
- 이 디렉토리는 APT(Advanced Package Tool) 패키지 관리자가 패키지 서명 키를 보관하는 데 사용
- `-m 0755`: 생성된 디렉토리의 퍼미션을 지정합니다. 여기서 `0755`는 소유자에게 읽기, 쓰기, 실행 권한을 부여하고, 다른 사용자들에게는 실행 권한을 부여하는 것을 의미합니다.
- `-d`: 디렉토리를 생성하는 데 사용됩니다. 파일이 아닌 디렉토리를 생성하는 것을 지정하는 옵션입니다.

▼ `sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc`

- Docker의 공식 GPG 키를 `/etc/apt/keyrings/docker.asc` 파일에 다운로드함.
- GPG 키는 Docker 패키지의 유효성을 확인하는 데 사용됨
- `curl`: URL로부터 데이터를 다운로드하는 명령입니다.
- `fsSL`: 다운로드 시에 사용되는 curl 옵션으로, `f`, `s`, `S`, `L` 옵션을 사용합니다.
  - `f`: 오류 발생 시 오류 코드를 출력하지 않습니다.
  - `s`: 정적 출력을 사용하여 진행 상황을 표시하지 않습니다.
  - `S`: 오류가 발생한 경우 진행 상황을 출력합니다.
  - `L`: 리다이렉션을 따릅니다.
- `https://download.docker.com/linux/ubuntu/gpg`: Docker의 공식 GPG 키가 위치한 URL입니다.
- `o /etc/apt/keyrings/docker.asc`: 다운로드된 데이터를 `/etc/apt/keyrings/docker.asc` 파일에 저장합니다.

▼ `sudo chmod a+r /etc/apt/keyrings/docker.asc`

- `chmod`: 파일의 권한을 변경하는 명령입니다.
- `a+r`: 모든 사용자에게 읽기 권한을 부여하는 옵션입니다. `a`는 all(모든 사용자)를 의미하고, `+r`은 읽기 권한을 추가한다는 것을 의미합니다.
- `/etc/apt/keyrings/docker.asc`: 권한을 변경할 파일의 경로입니다.

▼ `echo \`

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- 첫 번째 줄 :
  - `echo`: 텍스트를 출력하는 명령입니다.
  - `deb`: APT 저장소 설정 파일의 첫 번째 줄로, 저장소의 유형과 주소를 지정합니다.
  - `[arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]`: 아키텍처와 GPG 키 파일을 지정합니다. 여기서 `dpkg --print-architecture` 명령어를 사용하여 시스템의 아키텍처를 가져옵니다.
  - `https://download.docker.com/linux/ubuntu`: Docker의 공식 APT 저장소의 주소입니다.
  - `$(. /etc/os-release && echo "$VERSION_CODENAME")`: 우분투 버전 코드명을 가져와서 저장소 URL에 포함시킵니다.
  - `stable`: Docker의 안정 버전을 지정합니다.
- 두 번째 줄 :
  - `sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`: 앞서 생성한 내용을 `/etc/apt/sources.list.d/docker.list` 파일에 기록합니다. `tee` 명령어는 표준 출력과 파일 모두에 쓰기를 할 수 있도록 합니다. `> /dev/null`은 표준 출력을 무시하도록 합니다.
- 세 번째 줄 :
  - `sudo apt-get update`: APT 패키지 목록을 최신 상태로 업데이트합니다.

▼ `sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`

- `docker-ce`: Docker Community Edition을 설치합니다. 이는 Docker의 컨테이너 실행 엔진입니다.
- `docker-ce-cli`: Docker CLI(Command Line Interface) 도구를 설치합니다. 이를 통해 Docker 컨테이너 및 이미지를 관리할 수 있습니다.
- `containerd.io`: 컨테이너 실행을 위한 필수 구성 요소인 containerd를 설치합니다.
- `docker-buildx-plugin`: Docker의 다중 아키텍처 빌드 기능을 확장하는 플러그인을 설치합니다. 이는 다중 플랫폼 이미지를 빌드하는 데 사용됩니다.
- `docker-compose-plugin`: Docker Compose를 관리하는 플러그인을 설치합니다. Docker Compose는 여러 컨테이너를 사용하여 복잡한 애플리케이션을 정의하고 실행하는 도구입니다.

▼ `sudo docker run hello-world`

- "hello-world" 이미지를 실행하는 것으로 기본 설치되어 있는 이미지를 실행함으로서 Docker가 정상적으로 설치되었는지 확인하기 위함.

도커 설치 완료!!

1. jenkins 설정

> 💡 [Docker 리눅스 설치]를 참고하여 Docker를 먼저 설치한다.

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

- Docker 컨테이너와 서버의 폴더를 연결할 수 있다. 이것을 마운트하다 라고 한다.
  이제부터 Jenkins 컨테이너를 만들 예정인데, 아직 만들기 전에 Jenkins 컨테이너와 ES2서버의 폴더를 연결하기 위해서 ES2서버의 내부에
  `jenkins-data` 라는 폴더를 만드는 것이다.

▼ `sudo ufw allow 9999/tcp`

우분투의 UFW(방화벽)에 9999포트의 TCP트래픽을 수신할 수있도록 규칙을 추가하는 명령여

다른 포트로하고싶다면 해도 상관없다. 여기에서는 9999번으로 하겠다.

왜냐하면 8080포트는 BackEnd 포트로 8080을 사용할 예정인데, 젠킨스가 8080을 쓰면 안되기 때문에 젠킨스 포트는 9999로 하겠다.

▼ `sudo ufw reload`

현재 설정된 방화벽 규칙을 다시로드하여 새로운 설정을 적용하는 명령어

▼ `sudo ufw status`

현재 UFW의 상태를 확인하는 명령어. 이 명령어를 사용하면 현재 활성화된 방화벽 규칙과 정책을 확인할 수 있다.

▼ `sudo docker run -d-p 9999:8080 -v /home/ubuntu/jenkins-data:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v /home/ubuntu/share_dir:/var/share_dir --name jenkins -u root jenkins/jenkins`

도커 이미지를 실행하여 컨테이너로 실행하는 명령어. 이미지가 로컬에 없으면 자동으로 Docker Hub에서 다운로드됨.

`-d` : 컨테이너를 백그라운드에서 실행하는 옵션

`-p 9999:8080` : 호스트의 8080포트를 컨테이너의 9999포트로 포트포워딩함. 젠킨스가 기본적으로 8080포트로 열리는데, 컨테이너 외부에서는 9999포트로 접근하면 젠킨스의 포트 8080으로 매칭되어 접근할 수 있는 것임.

`-v /home/ubuntu/jenkins-data:/var/jenkins_home` : 호스트의 `/home/ubuntu/jenkins-data` 디렉터리를 컨테이너 내부의 `/var/jenkins_home` 디렉터리에 연결(마운트)한다. 어느 한쪽에 파일이 있다면 다른 한쪽에도 파일이 복제되어 공유하는것이다. 용량이 2배로 낭비될 수 있지만 파일을 공유할수있다는 장점이 있어서 사용한다.

만약 플러그인 설치가 안된다면 lts 설치를 하지말고 `sudo docker run -d -p 9999:8080 -v /home/ubuntu/jenkins-data:/var/jenkins_home --name jenkins jenkins/jenkins:latest` 를 한다. 플러그인과 젠킨스 버전이 일치해야하는 경우에 플러그인이 설치되는 상황이 있는데, lts버전이 플러그인버전보다 구버전이라서 그렇다. lts를 빼버리고 최신버전으로 설치하면 문제가 해결된다.

▼ `sudo docker logs jenkins`

해당 컨테이너의 구동 상태를 보기 위해 사용하는 도커 명령어. 중간에 출력되는 초기 패스워드는 조금 이따 쓰이므로 필수로 기록해둔다.

▼ `sudo docker stop jenkins`

해당 컨테이너를 중지하기 위해 사용하는 도커 명령어.

▼ `sudo docker ps -a`

sudo docker ps는 현재 실행되고 있는 컨테이너 목록을 보여주는 명령어이다.
-a를 붙이면 모든 컨테이너들을 보여주며, 중지된 컨테이너도 보여준다.
따라서 모든 컨테이너들을 볼 수 있다.

▼ `cd /home/ubuntu/jenkins-data`

젠킨스 데이터 폴더로 이동하는 리눅스 명령어

▼ `mkdir update-center-rootCAs`

현재 디렉토리에 update-conter-rootCAs 디렉토리 만듦.

▼ `wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/update-center.crt -O ./update-center-rootCAs/update-center.crt`

- `wget`을 사용하여 특정 URL에서 파일을 다운로드하고, 다운로드한 파일을 `./update-center-rootCAs/update-center.crt` 경로에 저장하는 명령어
- `wget` : URL에서 파일을 다운로드하는 명령입니다.
- `https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/update-center.crt` : 다운로드할 파일의 URL입니다.
- `-O` : 다운로드한 파일의 저장 경로를 지정하는 옵션입니다.
- `./update-center-rootCAs/update-center.crt` : 다운로드한 파일을 저장할 경로입니다.

▼ `sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.githubusercontent.com/lework/jenkins-update-center/master/updates/tencent/update-center.json#' ./hudson.model.UpdateCenter.xml`

- `sed`를 사용하여 특정 파일(`./hudson.model.UpdateCenter.xml`) 내에서 특정 패턴(`https://updates.jenkins.io/update-center.json`)을 다른 패턴(`https://raw.githubusercontent.com/lework/jenkins-update-center/master/updates/tencent/update-center.json`)으로 교체하는 명령어
- `sed` : 텍스트 파일에서 특정 패턴을 검색하고, 해당 패턴을 다른 패턴으로 교체하는 명령입니다.
- `i` : 원본 파일을 직접 수정하도록 지정하는 옵션입니다. 즉, 인라인 모드로 작동합니다.
- `'s#https://updates.jenkins.io/update-center.json#https://raw.githubusercontent.com/lework/jenkins-update-center/master/updates/tencent/update-center.json#'` : 검색할 패턴과 교체할 패턴을 지정하는 `sed`의 스크립트입니다. 여기서 `s`는 "substitute"를 의미하고, `#`는 패턴과 패턴 사이를 구분하는 구분자입니다.
- 따라서 이 명령을 실행하면 `./hudson.model.UpdateCenter.xml` 파일에서 `https://updates.jenkins.io/update-center.json`이라는 패턴이 `https://raw.githubusercontent.com/lework/jenkins-update-center/master/updates/tencent/update-center.json`으로 교체됨
- 이 명령어를 수행하는 이유는 Default설정되어있던 `https://updates.jenkins.io/update-center.json` 에서 다른 URL `https://raw.githubusercontent.com/lework/jenkins-update-center/master/updates/tencent/update-center.json` 로 URL을 옮기는 것인데, 이렇게 하면 jenkins 플러그인이나 빌드 도구와 같은 소프트자원을 다운로드할 서버를 지정해줄 수 있는 것이다. 속도향상이나 안정성 문제등의 이유로 미러사이트를 바꾸고는 한다.

▼ `sudo docker restart jenkins`

설정을 바꿨으므로 젠킨스 컨테이너를 리스타트

- 여기까지 완료되었으면 브라우저의 링크창에 [설치서버url:젠킨스포트]를 입력하여 젠킨스로 접속이 되는지 확인한다.
- 패스워드 입력창이 뜨는데, `sudo docker logs jenkins` 에서 보았던 초기 password를 입력한다.
- install 창이 뜨는데, 커스텀할거 아니면 install suggested plugins를 누른다.
- 뭔가 많이 설치된다. pipeline과 젠킨스에 필요한 플러그인들 등등..
- admin 계정을 만든다. 계정명과 암호, 이름 등은 알아서 설정한다.
- 외부접속 URL을 설정한다. 여기서는 9999를 사용하였다.

---

2. MariaDB설치

```
docker pull mariadb
docker run -p 3306:3306 --name mariadb -e MARIADB_ROOT_PASSWORD=4to9123! -d mariadb
```

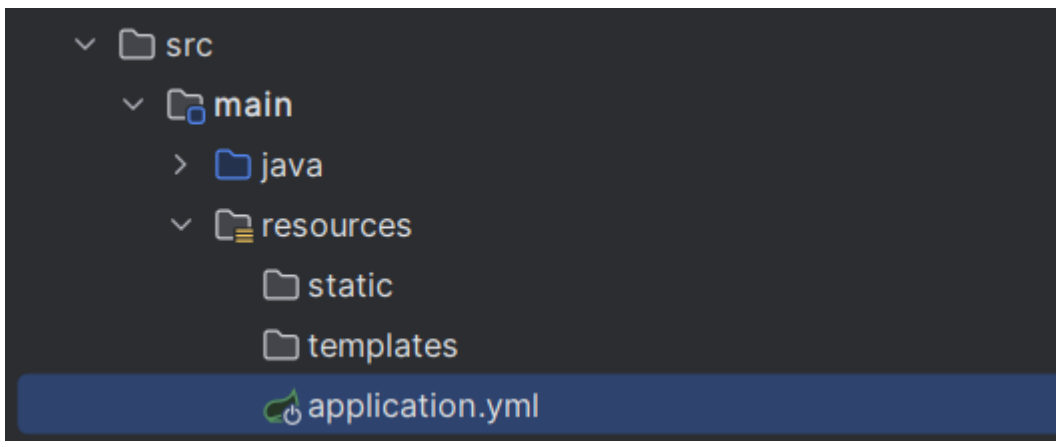
3. Redis 설치

```
sudo docker pull redis:7.2.4
sudo docker run -d -p 6379:6379 --name redis redis:7.2.4
```

4. application.yml

💡 공개되면 안 되는 중요한 properties가 담겨있는 파일입니다

파일 위치



- 개인용 application.yml

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://j10c104.p.ssafy.io:3306/kkirikkiri
    username: root
    password: 4to9123!

  data:
    redis:
      host: j10c104.p.ssafy.io
      port: 6379

  servlet:
    multipart:
      max-file-size: 50MB
      max-request-size: 50MB

  naver:
    tts:
      client-id: 0rree924vg
      client-secret: 2sQdqMkyFUeMADNFsAeh1fpFIWHqvfI46dTWqkHE

  jpa:
    hibernate:
```

```
      ddl-auto: update

cloud:
  aws:
    credentials:
      access-key: AKIARDN2X5V5VRFUB4GW
      secret-key: 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
    s3:
      bucket: kkiri-kkiri
    region:
      static: ap-northeast-2 # 리전 정보
    stack:
      auto: false

myapp:
  fastApi:
    endpoint: http://112.150.97.195:8000/api/generate
```

- EC2 서버용 application.yml (컨테이너 IP사용하여 이동경로를 줄임)

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://172.17.0.3:3306/kkirikkiri
    username: root
    password: 4to9123!

  data:
    redis:
      host: 172.17.0.2
      port: 6379

  servlet:
    multipart:
      max-file-size: 50MB
      max-request-size: 50MB

  naver:
    tts:
      client-id: 0rree924vg
      client-secret: 2sQdqMkyFUeMADNFsAeh1fpFIWHqvfI46dTWqkHE

  jpa:
    hibernate:
      ddl-auto: update

cloud:
  aws:
    credentials:
      access-key: AKIARDN2X5V5VRFUB4GW
      secret-key: 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
    s3:
      bucket: kkiri-kkiri
    region:
      static: ap-northeast-2 # 리전 정보
    stack:
```

```
      auto: false

myapp:
  fastApi:
    endpoint: http://127.0.0.1:8000/api/generate
```

5. PipeLine

- 젠킨스 React_Build Pipeline

```
pipeline {
    environment {
        repository = "fe-img"  //docker hub id와 repository 이름
        DOCKERHUB_CREDENTIALS = credentials('dockerhub-jenkins') // jenkins에 등록해 놓은 doc
ker hub credentials 이름
        dockerImage = ''
    }
    agent any

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'FE_develop', credentialsId: 'awldnjs2@naver.com', url: 'http
s://lab.ssafy.com/s10-ai-image-sub2/S10P22C104.git'
            }
        }

        stage('image_build') {      // 이미지 빌드시키면서 build도 시키고, nginx도 돌리고 하기....
            steps {
                dir("./FE") {
                    sh 'cp /var/share_dir/FE/dockerfile /var/jenkins_home/workspace/img_bu
ild/FE/dockerfile'        // 볼륨마운트 폴더에서 dockerfile을 img_docker디렉터리로 복사
                    sh 'cp /var/share_dir/FE/nginx.conf /var/jenkins_home/workspace/img_bu
ild/FE/nginx.conf'        // 볼륨마운트 폴더에서 nginx.conf을 img_docker디렉터리로 복사
                    sh 'cp -r ./frontend /var/jenkins_home/workspace/img_build/FE'      //
같은 폴더에 jar파일을 복사
                }

                script {
                    dir("/var/jenkins_home/workspace/img_build/FE/") {
                        sh 'ls'
                        sh 'docker build --no-cache -t $repository:$BUILD_NUMBER .'    //
이미지 빌드
                    }
                }
            }
        }

        // 로그인 후 이미지 푸쉬
        stage('image_push') {
            steps {
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIA
LS_USR --password-stdin' // docker hub 로그인
                sh 'docker tag $repository:$BUILD_NUMBER $DOCKERHUB_CREDENTIALS_USR/$repos
itory:$BUILD_NUMBER'
```

```
                sh 'docker push $DOCKERHUB_CREDENTIALS_USR/$repository:$BUILD_NUMBER' //do
cker push
                sh "docker rmi $repository:$BUILD_NUMBER $DOCKERHUB_CREDENTIALS_USR/$repos
itory:$BUILD_NUMBER" // 로컬에서 docker image 제거
                sh "docker system prune -f -a --filter until=12h"
            }
        }

        // EC2에서 기존 이미지 중단, 삭제 후 새로운 이미지 run
        stage('image_pull&deploy') {
            steps {
                script {
                    sshPublisher(
                        continueOnError: false,
                        failOnError: true,
                        publishers: [
                            sshPublisherDesc(
                                configName: 'ubuntu', // SSH 설정 이름
                                verbose: true,
                                transfers: [
                                    sshTransfer(
                                        execCommand: "docker stop ${repository}; docker rm
${repository}; docker run -d -p 3000:3000 --name ${repository} ${DOCKERHUB_CREDENTIALS_US
R}/${repository}:${BUILD_NUMBER};"
                                    )
                                ]
                            )
                        ]
                    )
                }
            }
        }
    }
}
```

- 젠킨스 SpringBoot_Build Pipeline

```
pipeline {
    environment {
        repository = "be-img"  //docker hub id와 repository 이름
        DOCKERHUB_CREDENTIALS = credentials('dockerhub-jenkins') // jenkins에 등록해 놓은 doc
ker hub credentials 이름
        dockerImage = ''
    }
    agent any

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'BE_develop', credentialsId: 'awldnjs2@naver.com', url: 'http
s://lab.ssafy.com/s10-ai-image-sub2/S10P22C104.git'
            }
        }

        stage('BE_Build') {
            steps {
```

```
                dir("./BE/src/main/resources") {
                    sh 'cp /var/share_dir/secret/application.yml .' // application.yml을 볼
륨컨테이너에서 레파지토리 폴더로 복사
                }
                dir("./BE") {
                    sh "chmod +x gradlew"
                    sh "./gradlew clean build"      // 빌드
                }
            }
        }

        stage('image_build') {
            steps {
                dir("./BE/build/libs") {            // 빌드 전 dockerfile과 jar파일 복사
                    sh 'cp /var/share_dir/BE/dockerfile /var/jenkins_home/workspace/img_bu
ild/BE/dockerfile'        // 볼륨마운트 폴더에서 docker파일을 img_docker디렉터리로 복사
                    sh 'cp ./kkirikkiri-0.0.1-SNAPSHOT.jar /var/jenkins_home/workspace/img
_build/BE/BEimg.jar'      // 같은 폴더에 jar파일을 복사
                }

                script {
                    dir("/var/jenkins_home/workspace/img_build/BE/") {
                        sh 'ls'
                        docker.build repository + ":$BUILD_NUMBER"    // 이미지 빌드
                    }
                }
            }
        }

        stage('image_push') {
            steps {
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIA
LS_USR --password-stdin' // docker hub 로그인
                sh 'docker tag $repository:$BUILD_NUMBER $DOCKERHUB_CREDENTIALS_USR/$repos
itory:$BUILD_NUMBER'
                sh 'docker push $DOCKERHUB_CREDENTIALS_USR/$repository:$BUILD_NUMBER' //do
cker push
                sh "docker rmi $repository:$BUILD_NUMBER $DOCKERHUB_CREDENTIALS_USR/$repos
itory:$BUILD_NUMBER" // 로컬에서 docker image 제거
                sh "docker system prune -f -a --filter until=12h"
            }
        }

        // EC2에서 기존 이미지 중단, 삭제 후 새로운 이미지 run
        stage('image_pull&deploy') {
            steps {
                script {
                    sshPublisher(
                        continueOnError: false,
                        failOnError: true,
                        publishers: [
                            sshPublisherDesc(
                                configName: 'ubuntu', // SSH 설정 이름
                                verbose: true,
                                transfers: [
                                    sshTransfer(
                                        execCommand: "docker stop ${repository}; docker rm
${repository};
```

```
                                        docker run -d -p 8080:8080 --name ${repository}
${DOCKERHUB_CREDENTIALS_USR}/${repository}:${BUILD_NUMBER};"
                                    )
                                ]
                            )
                        ]
                    )
                }
            }
        }
    }
}
```

6. dockerfile

BE용 dockerfile

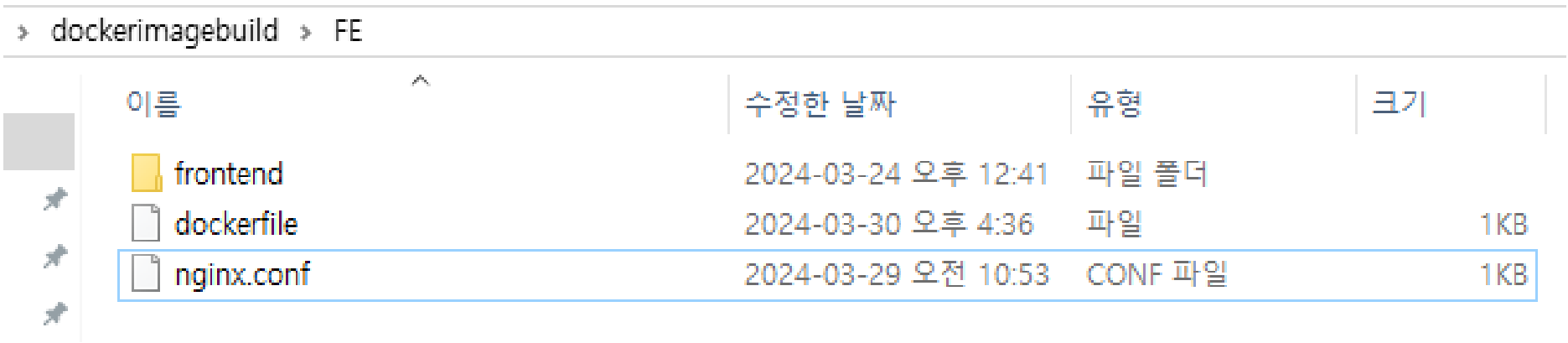
```
FROM azul/zulu-openjdk:17.0.10-jre

ARG JAR_FILE=BEimg.jar
COPY ${JAR_FILE} BEimg.jar

EXPOSE 8080

ENTRYPOINT ["java","-jar","/BEimg.jar"]
```

FE용 dockerfile



- dockerfile

```
FROM node:20.11.1-alpine

# Nginx 설치
RUN apk update && apk add nginx

# WORKDIR /usr/share/nginx
# RUN pwd && ls

# 소스 코드 복사, image/FE폴더에 있는
# dockerfile과 frontend폴더가 container 내부로 복사됨
COPY . .

# 작업 디렉토리 설정
```

```
WORKDIR /frontend

RUN ls /etc/nginx/

# Nginx 설정 파일 복사
COPY nginx.conf /etc/nginx/nginx.conf

# 필요한 라이브러리 설치
RUN npm install --force

RUN ls /frontend

# 빌드
RUN npm run build

RUN ls /frontend

RUN ls /frontend/dist

# container의 80번 포트를 외부로 노출
EXPOSE 3000

RUN mkdir -p /usr/share/nginx/html

RUN pwd

# Nginx의 기본 문서 루트에 빌드된 파일 복사
RUN cp -r dist/* /usr/share/nginx/html/

# 컨테이너 실행 시 실행될 명령어
CMD ["nginx", "-g", "daemon off;"]
```

- nginx.conf 파일 내용 (이미지 빌드에 필요함)

```
user nginx;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    error_log /var/log/nginx/error.log warn;

    server {
        listen 3000;
        server_name localhost;

        root /usr/share/nginx/html;
        index index.html;
```

```
        location / {
        #    try_files $uri $uri/ =404;
            try_files $uri $uri/ /index.html;
        }
    }
}
```

PS

- 젠킨스 이미지 run 명령어

```
sudo docker run -d -p 9999:8080 -v /home/ubuntu/jenkins-data:/var/jenkins_home
-v /var/run/docker.sock:/var/run/docker.sock -v /home/ubuntu/share_dir:/var/share_dir --na
me jenkins -u root jenkins/jenkins
```

버전정보

- JVM : zulu 17.48.15
- node.js : v20.11.1
- Python : v3.9.13
- AI model : luna-diffusion

시연 시나리오

0. **로그인 페이지**

   0.1 회원가입: 화면 하단 회원가입 링크 -> 1 회원가입 페이지로 이동

      시연용 계정 => 아이디: user3 /비밀 번호 : !user123

   0.2 로그인 완료 후 메인페이지로 이동

1. **회원가입**

   1.1 프로필 이미지 선택

   1.2 아이디: 5~10자리 영어, 특수문자 불가 - > 중복 확인 진행

   1.3 닉네임: 최대 5글자 별명 설정 –> 중복 확인 진행

   1.4 비밀번호: 8~15자리 숫자, 특수문자

      1.4.1 비밀번호 확인

   1.5 나이 드롭박스에서 나이 선택

   1.6 영어수준 드롭박스에서 영어 수준 선택

   1.7 모든 입력란 확인 후 가입하기 버튼 활성화 -> 회원가입 완료 후 로그인 페이지로 자동 이동

2. **메인 페이지**

   2.1 화면 상단 캐러셀 소개: 서비스 메인 기능인 스토리, 책장, 도서관 기능 소개, 해당 캐러셀 선택 시 음성으로 소개 멘트를 들을 수 있음

   2.2 내 이야기 쓰러 가기: 화면 하단 1번째 버튼 클릭 -> 3. 이야기 작성 페이지로 이동

3. **이야기 쓰기**

3.1 이야기 작성하기: 쓰고 싶은 동화 첫 문장을 100자 이내로 작성 -> 100자 이상으로 작성 시 입력 제한 확인 후 작성 버튼 클릭

3.2 ChatGpt의 응답 기다리기: 응답을 기다리는 동안 로딩 페이지 보여주기

3.3 미완성 상태로 화면 상단 왼쪽 로고 클릭 ->2. 메인 페이지로 이동

3.3.1 책장 버튼 클릭

3.3.2 책장에서 사용자가 쓰던 이야기가 '미완성된 이야기'의 제목으로 회색 책으로 보여지는 것 확인

3.3.3 해당 책 이미지 클릭: '계속 이야기를 작성하시겠습니까" 모달에서 yes 클릭

3.3.4 기존 이야기 작성 페이지로 자동 이동

3.4 사용자-GPT 순으로 반복하며 각각 5번, 총 10번의 문장 생성

3.5 이야기 계속하기 버튼: 10개의 문장 생성 전까지는 비활성화 상태 -> 완성 후 '책으로 보기'로 바뀌고 활성화 상태로 변경

3.6 책으로 보기 버튼 클릭: 4. 책장페이지로 자동 이동

3.7 화면 상단 로고 클릭: 메인페이지로 이동

4. **책장 페이지**

4.1 책 의미 설명

4.1.1 회색 책: 'ㅇㅇ님의 미완성된 이야기'로 사용자가 작성 중이던 책임을 알려줌

4.1.2 갈색 책: 사용자가 이야기를 완성한 책 or 5. 도서관에서 소장한 책 임을 알려줌

4.2 회색 책 클릭: 3.3.2와 같은 동작

4.3 갈색 책 클릭: 동화책 preview 모달 확인

4.3.1 preview 모달: 대표이미지, 책 제목, 작가명, 요약, 학습하기 버튼, 그림책 보기 버튼 설명

- 학습하기 버튼: 학습하기 페이지로 이동

- 그림책 보기 버튼: 내가 작성한 책 인 경우, 학습 미 완료시 그림책 버튼 보기 버튼 비활성화. 학습한 이야기이거나 도서관에서 소장한 이야기인 경우는 활성화된 초록색 버튼.

4.3.2 학습하기 버튼 클릭: 5. 학습 페이지로 자동 이동

5. **학습 페이지**

5.1 책 왼쪽 화면 상단 남/여 이미지 클릭 : 번역된 이야기 스크립트 음성으로 출력 (남/여 음성 중 선택)

5.2 책 왼쪽 화면 하단 작성했던 이야기 스크립트 확인 (한국어)

5.3 퀴즈 맞추기: 한국어 뜻, 음성 듣고 까만색 공란의 단어 분홍색 네모 칸 안에 작성

5.3.1 상단에서 내가 작성한 단어 확인

5.3.2 퀴즈 정답: 화면 왼쪽에 이미지 확인

5.4 학습 완료 -> 2. 메인 페이지 이동 -> 6. 도서관 페이지로 이동

6. **도서관 페이지**

6.1 TOP3 책 소개: 다운로드 많은 순으로 3개의 책 보여주는 캐러셀 소개, 캐러셀 안의 버튼 누르고 1,2,3 순위의 책 보기 6.2 정렬 드롭박스 클릭: 정렬 기준 변경하며 기능 확인(다운로드 많은/적은 순, 최신순)

6.2 검색 기준 드롭 박스 클릭: 제목/글쓴이 중 선택

6.2.1 키워드 입력 후 검색 버튼 클릭

6.2.2 검색 결과 중 하나의 이야기 클릭 -> 해당 이야기의 preview 모달

- Preview 모달: 동화책 AI 대표 이미지, 소장수, 제목, 작가명, 요악, 소장하기 버튼, 그림책 보기 버튼

6.2.2.1 소장하기 버튼 비 활성화인 경우: 내가 작성한 이야기/ 이미 소장한 이야기 라는 문구, 버튼 비활성화 상태 임을 설명

6.2.2.2 소장하기 버튼 활성화인 경우: 4. 책장으로 자동 이동

7. **동화책 보기 페이지**

7.1. 내가 소장했던 이야기 클릭: 이야기 preview모달

7.2 이야기 preview 모달에서 그림책 보기 클릭: 그림책 보기 페이지로 이동

7.3 음성 출력 버튼: 남/ 여 목소리 중 하나로 선택

7.4 이미지, 작성한 스크립트 확인

7.5 한/영 토글 버튼: 한글/ 영어 변경 가능

7.6 책의 가장자리 클릭 시 다음 페이지를 보여줌

7.7 화면 상단의 메인 로고 클릭: 메인 페이지로 이동

8. **헤더**

8.1 헤더 오른쪽 프로필 클릭: 회원정보 수정 / 로그아웃 버튼

8.2 마이페이지 버튼 클릭: 마이페이지로 이동

8.3 마이페이지로 이동

8.3.1 프로필 이미지, 닉네임(중복 확인), 비밀번호, 나이, 영어 수준 등을 변경할 수 있음을 설명

8.3.2 회원 탈퇴 링크: 회원 탈퇴 기능 설명

8.4 헤더 오른쪽 프로필 클릭

8.5 로그아웃 버튼 클릭 - > 로그인 화면으로 자동 이동