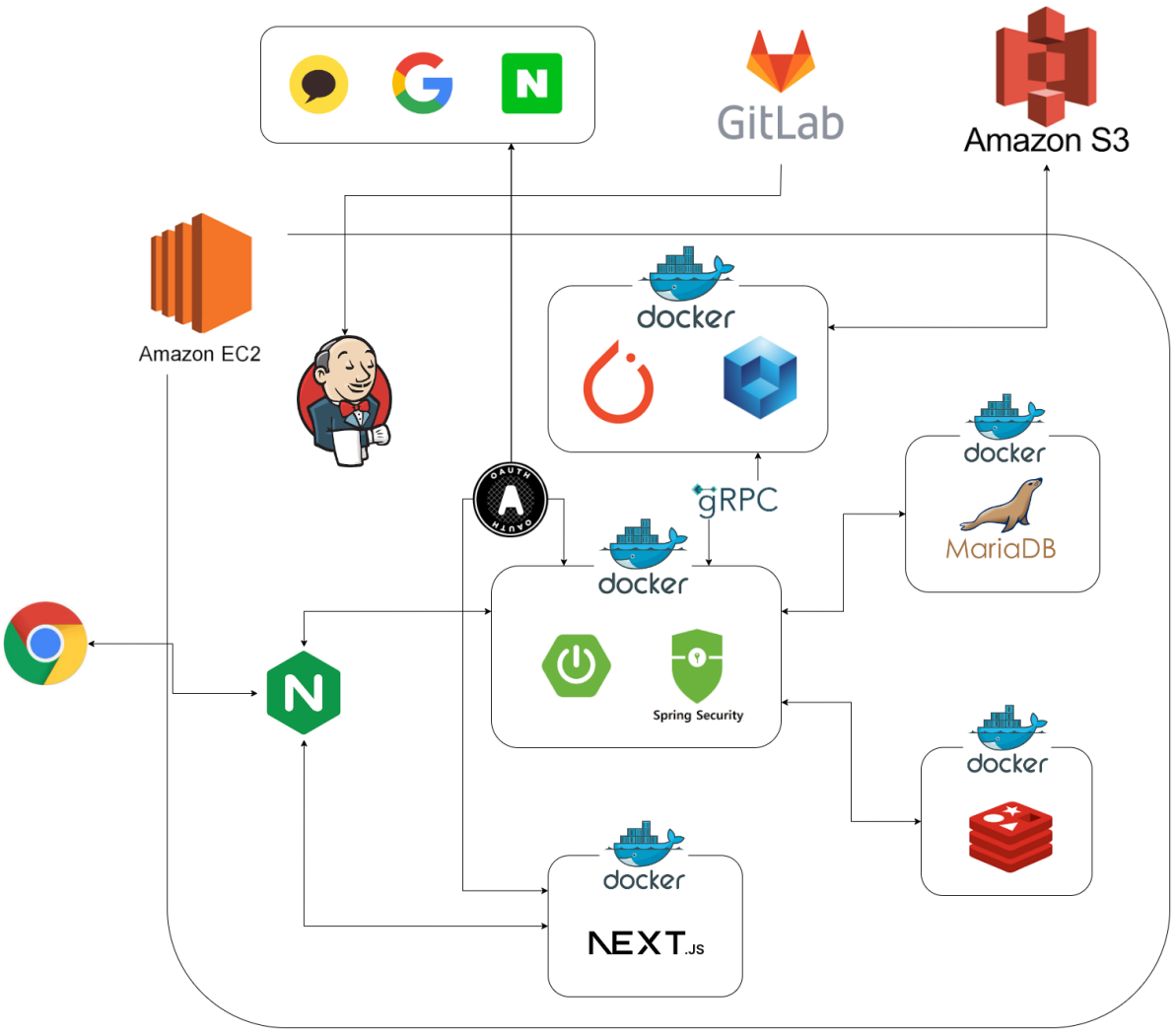
특화_서울1반_A101_포팅메뉴얼

1. 서버 아키텍처



2. 배포 환경/기술 스택

2.1 COMMON

- Server: Ubuntu 20.04.6 LTS
- Docker: 25.0.4
- jenkins : 2.448
- nginx : 1.18.0
- mariaDB: 11.3.2
- Redis: 7.2.4

2.2 FRONT

- NEXT.js: 14.1.1
- React: 18.0.0
- TypeScript: 5.2.3
- yarn: 4.1.1
- Node.js: 20.10.0 LTS
- next-pwa: 10.2.5
- storybook: 7.6.17
- styled-components: 6.1.8

2.3 BACK

- jdk: 17.0.10
- Gradle: 7.5
- Spring Boot : 3.2.3
- Spring Security: 6.2.2
- Spring Data JPA: 3.2.3
- spring-cloud-start-aws: 2.2.6
- gRPC: 1.61.0
- protobuf: 3.14
- protoPlugin: 0.9.4
- lombok: 1.18.30
- Log4j2: 2.21.1

2.4 AI

- python: 3.9.13
- insightface: 0.7.3
- opencv-python: 4.9.0.80
- grpcio: 1.62.0
- numpy: 1.26.4

- diffusers: 0.27.2
- transformers: 4.39.0
- pillow: 10.2.0

2.5 DEVELOP TOOL

- Visual Studio Code
- IntelliJ IDE
- AWS EC2
- AWS S3

3. 환경 변수 설정 파일 목록

3.1 Front

- ~/env/frontend/.env

3.2 Back

- ~/env/backend/.env

3.3 AI

- ~/env/ai/.env

4. 서버 기본 설정하기

서비스는 모든 서버가 하나의 환경 위에서 돌아가도록 설계되었다.

4.1 서버 시간 설정

```
sudo timedatectl set-timezone Asia/Seoul
```

4.2 미러 서버 카카오 서버로 변경

패키지 다운을 빠르게 하기 위함.

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirro
r.kakao.com/g' /etc/apt/sources.list
```

4.3 패키지 목록 업데이트 및 패키지 업데이트

```
sudo apt-get -y update && sudo apt-get -y upgrade
```

4.4 SWAP 영역 할당

```
//용량 확인
free -h

//스왑영역 할당
sudo fallocate -l 4G /swapfile

//swapfile 관한 수정
sudo chmode 600 /swapfile

//swapfile 생성
sudo mkswap /swapfile

//swapfile 활성화
sudo swapon /swapfile

//시스템 재부팅 되어도 swap 유지할수 있도록 설정.
sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/f
stab

//swap 영역 할당 되었는지 확인
free -h
```

5. 방화벽 열기

포트번호 : `22`, `80`, `443`, `8080`

```
sudo ufw allow {포트번호}
sudo ufw enable
sudo ufw reload
sudo ufw status
```

6. 필요한 리소스 설치하기

6.1 Java 설치

```
sudo apt-get install openjdk-17-jdk

# 자바 환경변수 설정하기
vi /etc/profile

# 가장 아래 쪽에 해당 내용 추가
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=$JAVA_HOME/jre/lib:$JAVA_HOME/lib/tools.ja
r

# 변경내용 적용하기
source /etc/profile
```

6.2 yarn 설치

6.2.1 curl 설치

```
sudo apt-get install curl
```

6.2.2 Node.js 설치

```
curl -fsSL https://deb.nodesource.com/setup_current.x | sud
o -E bash -
```

```
sudo apt-get install -y nodejs
```

6.2.3 yarn 설치

```
npm install yarn -g
```

6.3 도커 설치

6.3.1 Docker 설치 전 필요한 패키지 설치

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

6.3.2 Docker에 대한 GPC Key 인증 진행.

OK가 떴다면 정상적으로 등록된 것.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | s
udo apt-key add -
```

6.3.3 Docker 저장소 등록

```
sudo add-apt-repository "deb [arch=amd64] https://download.do
```

6.3.3 패키지 리스트 갱신

```
sudo apt-get -y update
```

6.3.4 Docker 패키지 설치

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

6.3.5 Docker 일반유저에게 권한부여

```
sudo usermod -aG docker ubuntu
```

6.3.6 Docker 서비스 실행

```
sudo service docker restart
```

6.3.7 Docker Compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/do
wnload/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o /
usr/local/bin/docker-compose
```

6.3.8 Docker Compose 실행 권한추가

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

6.4 Redis 설치(in Docker)

6.4.1 redis 네트워크 생성

```
docker network create redis-net
```

6.4.2 redis 이미지 가져오기

```
docker pull redis
```

6.4.3 redis 실행

```
docker run --name redis-server -p 6379:6379 --network redis
-net -d redis redis-server --appendonly yes --requirepass
비밀번호
```

6.4.4 redis 서버 접속

```
docker exec -it redis-server redis-cli -a "비밀번호"
```

6.5 mariaDB 설치(in Docker)

6.5.1 mariaDB 이미지 가져오기

```
sudo docker pull mariadb:latest
```

6.5.2 mariaDB 실행

```
docker run -d --restart always -p 3306:3306 -e MYSQL_ROOT_P
ASSWORD=비밀번호 -e TZ=Asia/Seoul -v /var/lib/mysql:/var/lib/
mysql --name mariadb mariadb
```

6.5.3 mariaDB 접속

```
docker exec -it mariadb bin/bash
```

6.5.4 mariaDB 패킷 크기 늘려주기

- 여기서부터 workbench를 이용해서 쿼리문을 실행

```
SET GLOBAL max_allowed_packet = 16 * 1024 * 1024;
```

6.5.5 database 생성

```
create database gallery;
```

6.6 NginX 설정

6.6.1 nginx 설치

```
sudo apt install nginx
```

6.6.2 nginx 실행

```
sudo service nginx start
sudo service nginx status
```

6.6.3 리버시 프록시 설정하기

```
sudo vi /etc/nginx/sites-available/service.conf

# 아래 내용 작성
server {
  listen 80;
  server_name 서버이름;

  location /api/v1 {
    proxy_pass 백엔드_서버주소;
  }

  location / {
    proxy_pass 프론트엔드_서버주소;
  }
}

# service.conf를 적용하기 위해선 default를 먼저 지워야함
sudo rm /etc/nginx/sites-enabled/default

# 작성한 service.conf 적용하기
sudo ln -s /etc/nginx/sites-available/service.conf /etc/ngi
nx/sites-enabled/service.conf

sudo service nginx reload
```

6.6.4 https 적용하기(webroot 방식)

```
sudo vi /etc/nginx/sites-available/service.conf

# 아래 내용 추가
```

```
location ^~ /.well-known/acme-challenge/ {
  default_type "text/plain";
  root /var/www/letsencrypt;
}

# 폴더 생성
sudo mkdir -p /var/www/letsencrypt
cd /var/www/letsencrypt
sudo mkdir -p .well-known/acme-challenge

# 인증서 발급
sudo certbot certonly --webroot
# 도메인과 root(/var/www/letsencrypt)를 입력해준다

# conf 파일을 다시 수정해준다
sudo vim /etc/nginx/sites-available/service.conf

server {
  listen 80;
  return 301 https://$host$request_uri;
}

server{
  listen 443;
  server_name j10a101;

  ssl on;
  ssl_certificate /etc/letsencrypt/live/j10a101.p.ssafy.io/
fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/j10a101.p.ssaf
y.io/privkey.pem;

  location /api/v1 {
    proxy_pass http://j10a101.p.ssafy.io:8081;
  }

  location / {
    proxy_pass http://j10a101.p.ssafy.io:3000;
```

```
  }

  location ^~ /.well-known/acme-challenge/ {
    default_type "text/plain";
    root /var/www/letsencrypt;
  }
}

sudo service nginx reload
```

6.7 AWS S3 버킷 생성

- 이미지를 저장할 S3 버킷을 생성해 놓는다.

7. 프로젝트 실행

7.1 프로젝트 Clone 하기

7.1.1 Git 설치하기

```
sudo apt-get install git
```

7.1.2 SSH 키 생성하기

```
cd ~/.ssh

ssh-keygen -t rsa -C [gitlab 계정 메일]

cat id_rsa.pub
# 퍼블릭 키 복사하여 Gitlab에 SSH 키 등록
```

7.1.3 Git Clone 하기

```
git clone [Git Repository 주소]
```

7.2 프론트

7.2.1 프론트 디렉토리로 이동

```
cd ~/[프로젝트명]/frontend
```

7.2.2 .env 파일 복사하기

```
cp /home/ubuntu/env/frontend/.env .
```

7.2.3 도커 이미지 빌드하기

```
docker build -t gallery-frontend:latest .
```

7.2.4 도커 이미지 실행하기

```
docker run -i -t -p 3000:3000 --env-file .env -d --name gal
lery-frontend -e TZ=Asia/Seoul gallery-frontend:latest
```

7.3 백엔드

7.3.1 백엔드 디렉토리로 이동

```
cd ~/[프로젝트명]/backend
```

7.3.2 .env 파일 복사하기

```
cp /home/ubuntu/env/backend/.env .
```

7.3.3 프로젝트 빌드하기

```
chmod +x ./gradlew
./gradlew wrapper --gradle-version=7.5 --distribution-type=
```

```
bin
./gradlew clean bootJar
```

7.3.4 도커 이미지 빌드하기

```
docker build -t gallery-backend:latest .
```

7.3.5 도커 이미지 실행하기

```
docker run -i -t -p 8081:8081 --env-file .env -d --name gal
lery-backend -e TZ=Asia/Seoul gallery-backend:latest
```

7.4 AI

7.4.1 AI 디렉토리로 이동

```
cd ~/[프로젝트명]/ai_backend
```

7.4.2 .env 파일 복사하기

```
cp /home/ubuntu/env/ai/.env .
```

7.4.3 도커 이미지 빌드하기

```
docker build -t gallery-ai .
```

7.4.4 도커 이미지 실행하기

```
docker run -i -t -d -p 9090:9090 --env-file .env --name gal
lery-ai -e TZ=Asia/Seoul gallery-ai:latest
```