포팅 매뉴얼

1. 개발 환경

- Server : `20.04.6 LTS (GNU/Linux 5.15.0-1051-aws x86_64)`
- JDK : `OpenJDK17`
- Node.js : `20.11.0 LTS`
- Nginx : `1.24.0`
- MySQL : `8.0.36`
- Jenkins : `2.426.2`

2. 설정파일 및 환경 변수 정보

Spring

```
spring:

  # database
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: {DB 주소}
    username: {DB 사용자 아이디}
    password: {DB 사용자 비밀번호}

  # jpa
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true;
    open-in-view: false

  # encoding
  http:
    encoding:
      charset: UTF-8
      force: true
      enabled: true

  # JWT 토큰
  jwt:
    secret: {JWT secret 키}
    filter: {JWT 필터}

  # SMTP
  mail:
    host: {SMTP 도메인 주소}
    port: {SMTP 포트번호}
```

```
    username: {SMTP 주소}
    password: {SMTP 비밀번호}
    properties:
      debug: true
      mail:
        smtp:
          auth: true
          ssl:
            enable: true
            trust: {신뢰할 수 있는 SMTP 서버 호스트}
          starttls.enable: true

# AWS S3
cloud:
  aws:
    s3:
      bucket: {S3 이름}
      credentials:
        access-key: {S3 access 키}
        secret-key: {S3 secret 키}
    region:
      static: {S3 URL}
    stack:
      auto: false

# OPENVIDU
OPENVIDU_URL: {Openvidu 서버 주소}
OPENVIDU_SECRET: {Openvidu 비밀번호}

# PROFILE IMAGE
DEFAULT_PROFILE_URL : {기본 프로필 사진 주소}
```

Vue

```
VITE_APP_API_URL = "https://i10d204.p.ssafy.io"
```

3. OpenVidu 온프레미스 배포

Docker 및 Docker Compose 설치

도메인 이름 설정

- 컴퓨터 공용 IP를 가리키는 도메인 이름을 구성하는 것을 권장

서버 PORT 설정

- OpenVidu는 다음의 포트를 확보해야 한다 : 80, 443, 3478, 5442, 5443, 6379, 8888
- 다음의 포트를 열어야 한다

```
ufw allow ssh
ufw allow 80/tcp # to connect using SSH to admin OpenVidu.
ufw allow 443/tcp # OpenVidu server and application are publi
ufw allow 3478/tcp # used by STUN/TURN server
ufw allow 3478/udp # used by STUN/TURN server
ufw allow 5001/tcp
ufw allow 5002/tcp
ufw allow 40000:57000/tcp # used by Kurento Media Server
ufw allow 40000:57000/udp # used by Kurento Media Server
ufw allow 57001:65535/tcp # used by TURN server
ufw allow 57001:65535/udp # used by TURN server
ufw enable
```

OpenVidu 배포

```
# 루트 권한 획득
sudo su

# OpenVidu 설치 경로로 이동
cd /opt
```

```
# OpenVidu 설치
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_

# OpenVidu Configuration 설정
# OpenVidu folder 이동
cd openvidu

# nano 편집기로 환경 변수 파일 오픈
nano openvidu/.env

# .env 파일 수정
DOMAIN_OR_PUBLIC_IP=도메인 이름
OPENVIDU_SECRET=OPENVIDU 서버 연결을 위한 시크릿 키
CERTIFICATE_TYPE=letsencrypt
LETSENCRYPT_EMAIL=LETSENCRYPT를 위한 EMAIL

# OpenVidu 서버 시작 및 중지
./openvidu start
./openvidu stop
```

4. SSL 인증서 발급 및 NGINX 설정

```
sudo apt-get install letsencrypt
sudo letsencrypt certonly --standalone -d <도메인>
```

```
upstream app {
  server mefi-backend:8080; # WAS 컨테이너의 이름
}

server {
    listen 80;
    listen 443 ssl;
    server_name i10d204.p.ssafy.io;
```

```
    root /usr/share/nginx/html;  # 정적 파일이 위치한 디렉토리 지정

    location / {
        try_files $uri $uri/ /index.html;  # 정적 파일 제공
    }

    location /api {
        proxy_pass http://app;  # API 요청을 WAS로 전달
    }

    location /socket {
        proxy_pass http://app/socket;  # 웹소켓 요청을 WAS로 전달
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
    }

    location /swagger-ui {
        proxy_pass http://app/swagger-ui;  # Swagger UI를 위한 프
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # SSL 설정
    ssl_certificate /etc/letsencrypt/live/i10d204.p.ssafy.io/ful
    ssl_certificate_key /etc/letsencrypt/live/i10d204.p.ssafy.io

    # HTTP 요청을 HTTPS로 리다이렉트
#    if ($scheme = http) {
#        return 301 https://$server_name$request_uri;
```

```
#    }
}
```

5. Docker 설치 관련 파일

docker 설치

```
sudo apt update
```

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

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud
o gpg --dearmor -o /usr/share/keyrings/docker-archive-keyrin
g.gpg
```

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-ar
chive-keyring.gpg] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.
d/docker.list > /dev/null
```

```
sudo apt update
sudo apt install docker-ce
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
docker --version
```

docker-compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/latest/
```

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

```
sudo docker-compose --version
```

dood(docker-out-of-docker)

원래 컨테이너 없애고 docker.sock 볼륨을 추가해서 다시 생성해야 한다. 도커 볼륨에 이전 정보들이 다 저장되어있어서 볼륨 삭제 안하고 컨테이너 생성할 때 그대로 쓰면 됨

도커 볼륨 : `/home/ubuntu/jenkins-data`

```
sudo docker rm -f jenkins
```

```
sudo docker run -d -p 8080:8080 \
-v /home/ubuntu/jenkins-data:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
--name jenkins jenkins/jenkins:lts-jdk17
```

```
sudo docker exec -u 0 -it jenkins bash

``` jenkins container bash ```

            apt-get update && \
            apt-get -y install apt-transport-https \
                ca-certificates \
                curl \
                gnupg2 \
                software-properties-common && \
```
```

```
            curl -fsSL https://download.docker.com/linux/debian/
            add-apt-repository "deb [arch=amd64] https://downloa
            apt-get update && \
            apt-get -y install docker-ce-cli

``` jenkins container bash ```
```

```
docker --version
```

```
ls -l /var/run/docker.sock
```

```
# chmod 666 /var/run/docker.sock
```

6. mysql 설치(EC2)

```
sudo apt update
sudo apt install mysql-server
```

```
sudo ufw allow mysql
```

```
sudo mysql -u root -p
```

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_passw

FLUSH PRIVILEGES;
```

mysql 보안 설정

```
sudo mysql_secure_installation
```
```

1. 패스워드 난이도를 설정할 것인지 : `y`
2. root 계정의 비밀번호를 수정할 것인지 : `n`
3. 익명 사용자를 제거할 것인지 : `y`
4. 원격으로 root의 mysql접속을 허용하지 않을 것인지 : `y`
5. Test 데이터베이스를 삭제할 것인지 : `y`
6. 권한테이블을 reload할 것인지 : `y`

외부 접속 허용

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

```
# bind-address            = 127.0.0.1
bind-address            = 0.0.0.0
```

```
sudo systemctl stop mysql
sudo systemctl start mysql
```

새로운 계정 생성

```
mysql -u root -p
```

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'user_password
```

```
GRANT ALL PRIVILEGES ON *.* TO 'new_user'@'localhost' WITH GRAN
```

```
FLUSH PRIVILEGES;
```

```
UPDATE mysql.user SET host='%' WHERE user='mefi123' AND host='lo
```

7. Jenkins 설치

jenkins container 생성 및 구동

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

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

```
sudo docker run -d -p 8080:8080 -v /home/ubuntu/jenkins-data:/va
--name jenkins jenkins/jenkins:lts-jdk17
```

```
sudo docker logs jenkins
```

```
sudo docker stop jenkins
sudo docker ps -a
```

환경 설정 변경 ⭐매우 중요⭐

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

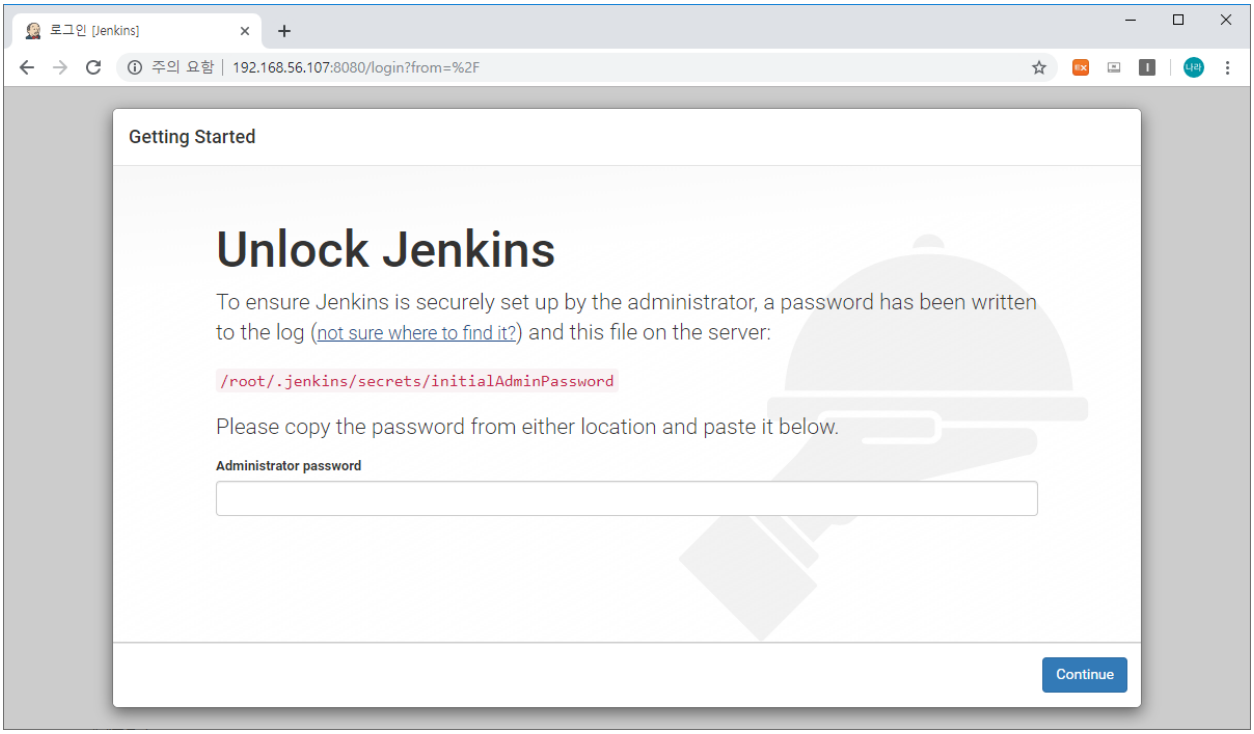
```
mkdir update-center-rootCAs
wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/ro
```

```
sudo sed -i 's#https://updates.jenkins.io/update-center.json#htt
```

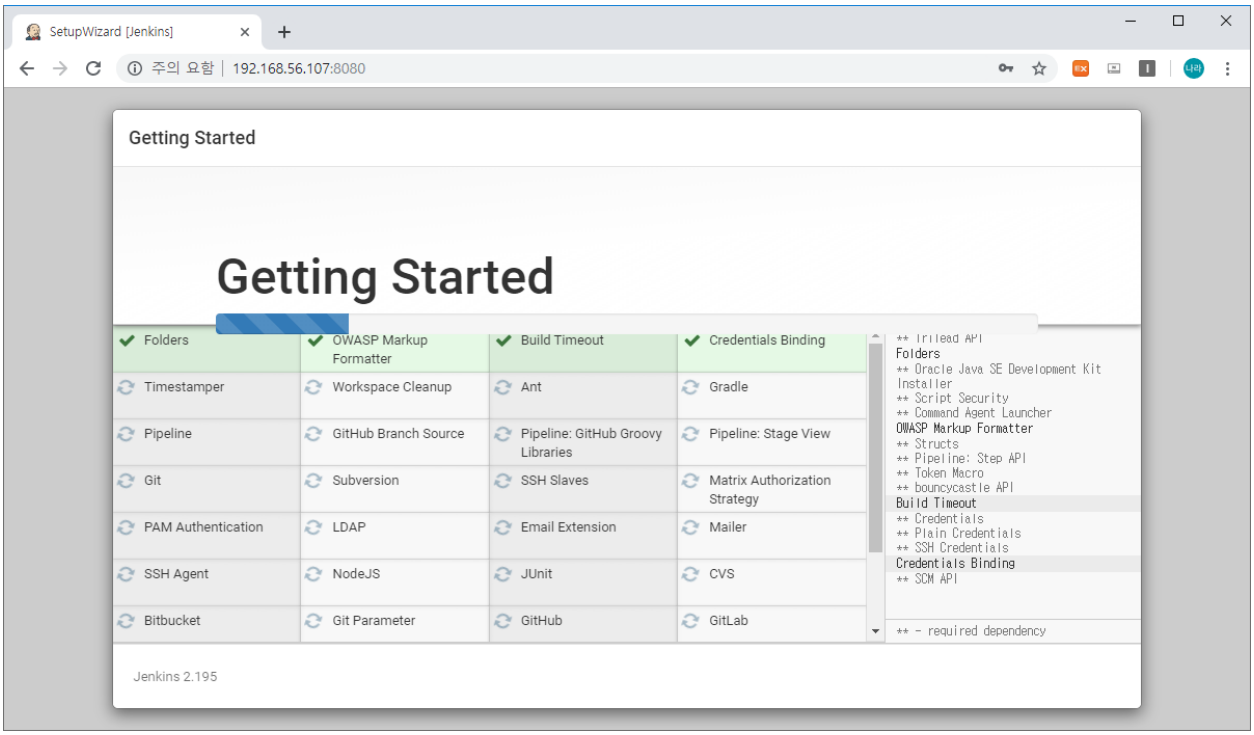
```
sudo docker start jenkins
```

```
# 컨테이너 구동
sudo docker start jenkins
# 컨테이너 종료
sudo docker stop jenkins
# 컨테이너 로그 확인
sudo docker logs jenkins
# 컨테이너 로그 실시간 확인(확인 중 Ctrl + C를 누르면 종료)
sudo docker logs -f jenkins
```

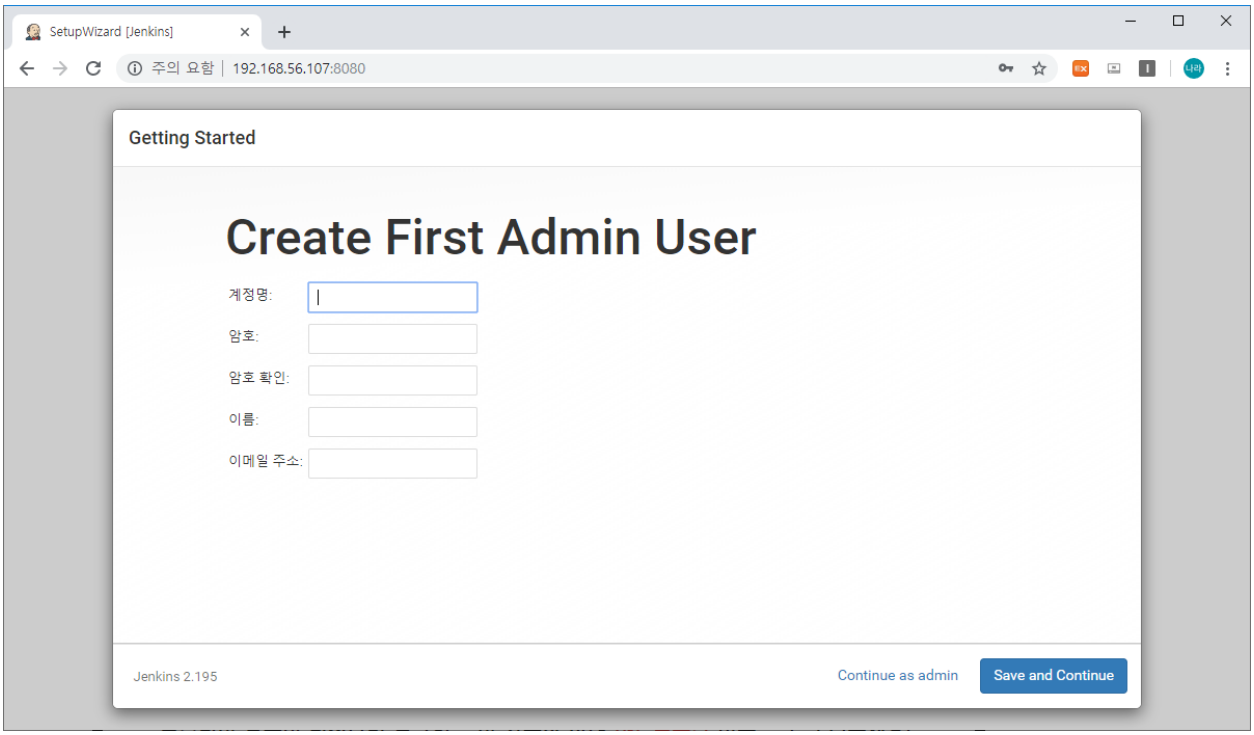
jenkins 초기 설정 진행



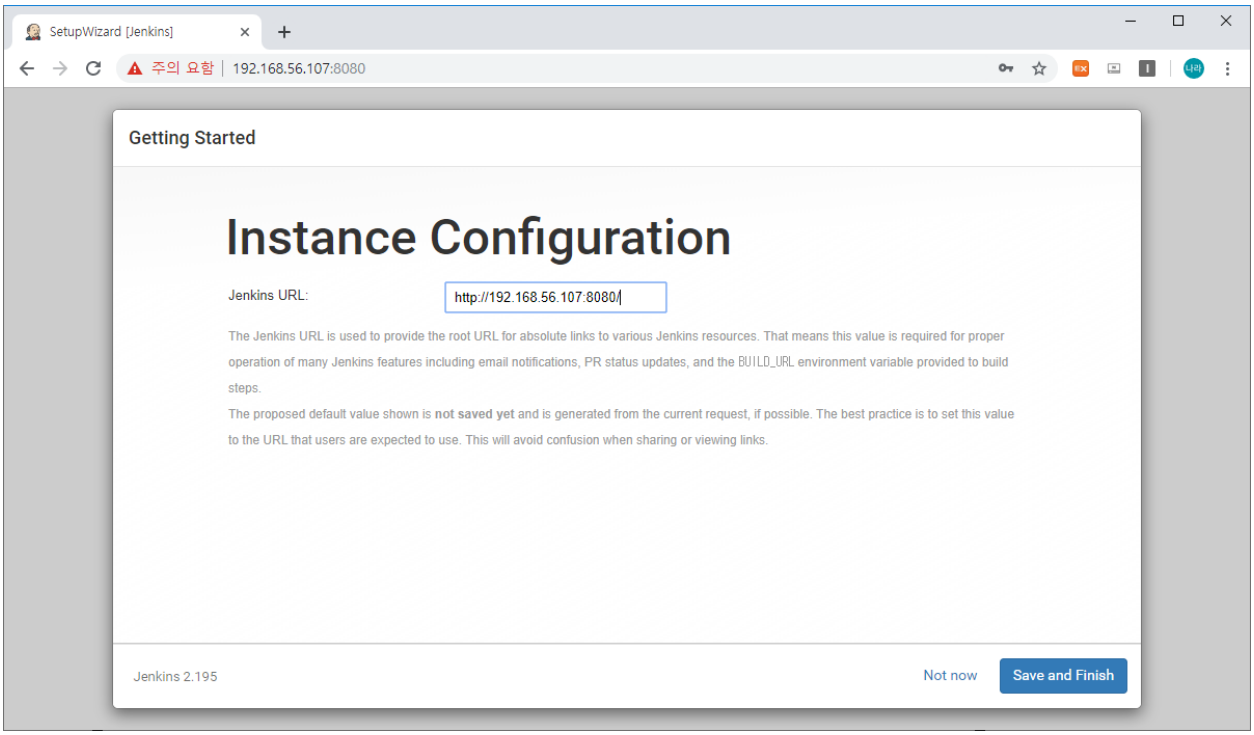
`http://<EC2 도메인네임>:8080`으로 접속



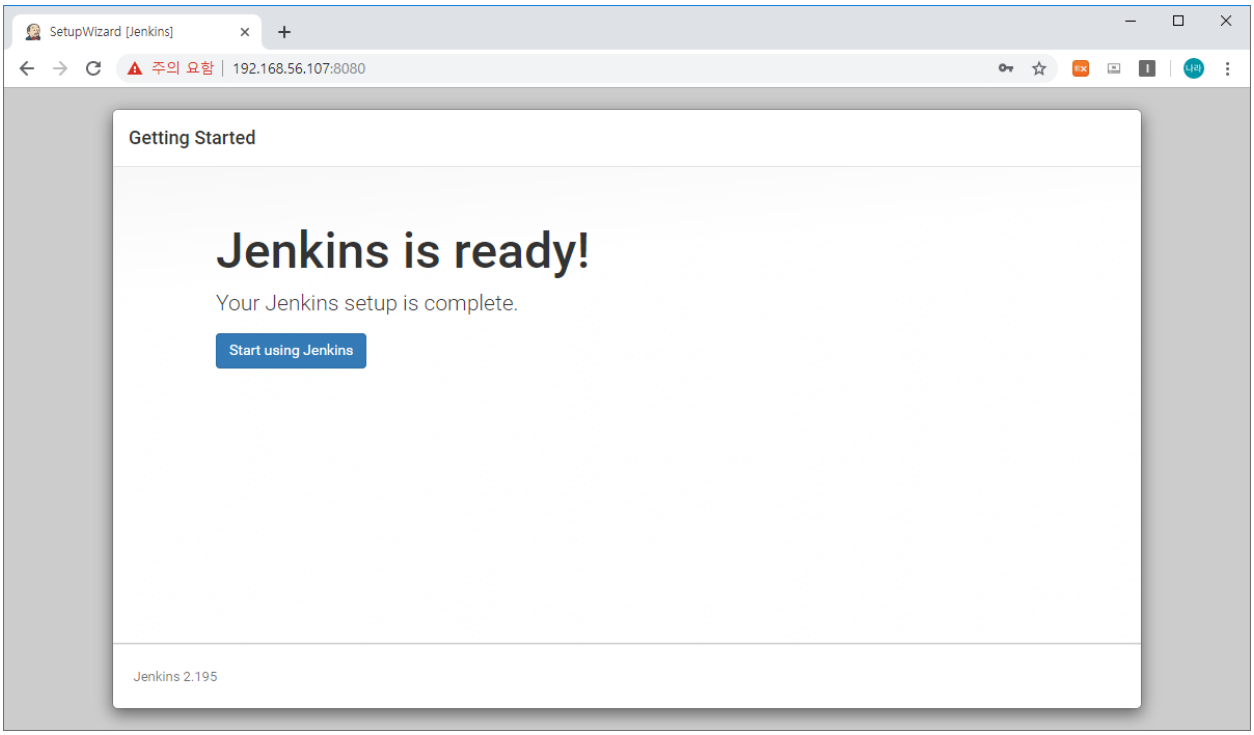
권장 플러그인 설치(만약 플러그인이 자동적으로 다 설치가 안되면 각각 설치 필요)



Admin User 생성



Jenkins Web UI에 접근하기 위한 Jenkins URL 입력(자동 입력됨)



초기설정 완료

8. AWS S3

IAM 계정

S3를 버킷을 사용하기 위해 IAM 계정을 생성한다. 이때 IAM 계정은 AWS 계정에 대한 공유 액세스,

Amazon EC2에서 실행되는 애플리케이션을 위한 보안 AWS 리소스 액세스를 지원한다.

IAM 계정 생성

STEP1) 사용자 이름 입력

STEP2) 권한 정책으로 AmazonS3FullAccess 선택하여 S3에 대한 모든 권한 소유

STEP3) 사용자 상세 - 보안 자격 증명 - 액세스 키 발급

※ 액세스 키는 다시 조회할 수 없으므로 반드시 저장하여 관리한다

S3 Bucket 생성

- Step1) AWS 리전 및 버킷 이름을 입력 (이때 리전에 관계없이 고유한 이름을 사용한다)
- Step2) 객체 소유권 설정 : 내 AWS 계정만 버킷에 접근하거나 사용할 수 있도록 설정한다
- Step3) 퍼블릭 액세스 차단 설정
  - 퍼블릭 액세스는 ACL, 버킷 정책, 액세스 지점 정책을 통해 버킷 및 객체에 부여된다
  - 액세스 차단을 활성화 하면 자원을 안전하게 보호할 수 있지만, 외부 접근이 불가능하다
  - 버킷에 저장된 파일에 접근하지 못하는 현상을 방지하고자, 모두 해제하고 진행한다
- Step4) 기본 암호화 : Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화

버킷 정책

- 버킷 상세 - 권한 - 버킷 정책 ： 객체 URL 접속 시 정상적으로 조회되는 것을 확인할 수 있다

```
// 모든 사용자에게 S3에 대한 DELETE, GET, PUT 권한 제공
{
    "Version": "2012-10-17",
    "Id": "Policy1705987522711",
```

```
    "Statement": [
        {
            "Sid": "Stmt1705987520867",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "{버킷 ARN}/*"
        }
    ]
}
```

9. 배포

docker-compose.yml

```
version: '3.8'                          # docker-compose 버전
services:                               # 서비스 목록
  mefi-frontend:                             # 서비스 이름
    container_name: mefi-frontend            # 컨테이너
    build:                              # 이미지 Build 옵션
      context: ./front-end              # Build될 프로젝트가 위치한 경로
      dockerfile: Dockerfile            # 프로젝트 폴더의 dockerfile 이
    ports:                              # host와 공유할 포트 목록
      - "80:80"                         # host:container
      - "443:443"
    environment:                        # 환경변수 설정
      VITE_APP_API_URL: "http://mefi-backend:8080"
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt
      - /var/lib/letsencrypt:/var/lib/letsencrypt
      - /home/ubuntu/nginx/sites:/etc/nginx/conf.d
      - /home/ubuntu/nginx/sites/sites-enabled:/etc/nginx/sites
```

```
    networks:
      - mefi_network

  mefi-backend:
    container_name: mefi-backend
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - "8080:8080"
    environment:                    # 환경변수 설정
      JASYPT_KEY: abc
      - TZ=Asia/Seoul
    networks:
      - mefi_network

networks:
  mefi_network:
    driver: bridge
```

Dockerfile

```
FROM eclipse-temurin:17-jdk-alpine
VOLUME /tmp
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar

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

```
FROM node:lts-alpine as build-stage
WORKDIR /homepage
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
```

```
FROM nginx:stable-alpine as production-stage
COPY --from=build-stage ./homepage/dist /usr/share/nginx/html/
CMD ["nginx", "-g", "daemon off;"]
```

Jenkins pipeline script

```
node {

    stage('Clone') {
        git branch: 'Dev', credentialsId: 'jenkins', url: 'https

        sh "sed -i 's/abc/mefi/g' docker-compose.yml"
        sh "cat docker-compose.yml"
    }

    stage('Build'){
        dir('backend') {
            sh "chmod +x gradlew"
            sh "./gradlew clean build"

        }
    }

    stage('Deploy') {
         // 이전에 실행된 컨테이너를 중지하고 삭제합니다.
        sh "docker stop mefi-backend || true"
        sh "docker rm mefi-backend || true"

        sh "docker stop mefi-frontend || true"
        sh "docker rm mefi-frontend || true"

        // 이전에 빌드된 이미지를 삭제합니다.
        sh "docker rmi dev_mefi-frontend || true"
        sh "docker rmi dev_mefi-backend || true"
```

```
        sh "docker ps -a || true"
        sh "docker images || true"

        sh "docker-compose build --no-cache"
        sh "docker-compose up -d"
    }

}
```

10. 사용 포트 목록

이름	내부 포트	외부 포트
http	80	
https	443	
Vue	5173	5173
SpringBoot	8080	8080
Jenkins	9000	9000
MySQL	3306	3306
openvidu-coturn-1	3478	3478
Kurento Media Server	40000	57000
TURN server	57001	65535
OpenVidu	5001	5001
	5002	5002