TRV 서비스 포팅 매뉴얼_B201

목차

1. 버전 정보

- 기본 버전 정보

> 💡 Git-2.43.0-64-bit
> node-v20.11.0-x64 LTS
> java 17
> Spring boot 3.2.1
> VSCodeUserSetup-x64-1.85.1
> idealU-2023.3.2
> Docker version 25.0.1

- EC2 서버 OS설치 정보

```
NAME="Ubuntu"
VERSION="20.04.6 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.6 LTS"
VERSION_ID="20.04"
```

- Docker image 버전

REPOSITORY	TAG
docker-vue	0.1
docker-springboot	0.1
sonarqube	latest
jc21/nginx-proxy-manager	latest
jenkins/jenkins	latest
redis	latest
postgres	latest
mysql	8.0.35
portainer/portainer-ce	latest
nginx	1.21

- Front 상세 설정 정보
  - package.json

```json
{
  "name": "zariyo",
  "version": "0.0.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
```

```
    "@date-io/date-fns": "^3.0.0",
    "@date-io/dayjs": "^3.0.0",
    "@popperjs/core": "^2.11.8",
    "axios": "^1.6.5",
    "date-fns": "^3.3.1",
    "dayjs": "^1.11.10",
    "dotenv": "^16.4.1",
    "jsonwebtoken": "^9.0.2",
    "jwt-decode": "^4.0.0",
    "openvidu-browser": "^2.29.1",
    "pinia": "^2.1.7",
    "pinia-plugin-persistedstate": "^3.2.1",
    "v-calendar": "^3.1.2",
    "v-click-outside": "^3.2.0",
    "vee-validate": "^4.12.4",
    "vue": "^3.3.11",
    "vue-i18n": "^9.9.0",
    "vue-router": "^4.2.5",
    "vuetify": "^3.5.1"
  },
  "devDependencies": {
    "@mdi/font": "^7.4.47",
    "@vitejs/plugin-vue": "^4.5.2",
    "vite": "^5.0.10"
  }
}
```

- back 상세 설정 정보
  - build.gradle

```
dependencies {
    ...
    // Spring Security
    implementation 'org.springframework.boot:spring-boot-sta
rter-security'
    testImplementation 'org.springframework.security:spring-
security-test'

    // Oauth2
    implementation 'org.springframework.boot:spring-boot-sta
rter-oauth2-client'
```

```
    // JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-impl:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.11.5'
    implementation 'com.auth0:java-jwt:4.2.1'
    implementation 'javax.xml.bind:jaxb-api:2.3.1'

    // redis
    implementation 'org.springframework.boot:spring-boot-sta
rter-data-redis'

    ...
}
```

2. 환경 변수

- Front 부분
  - .env

    front 프로젝트 최상단에 위치

```
VITE_API_URL=back 서버 url
VITE_API_KEY=카카오 javascript key
REST_API_KEY=카카오 rest api key
```

- Back 부분
  - application.properties

```
server.port=8282
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.hibernate.ddl-auto=update

spring.jpa.database-platform=org.hibernate.dialect.MySQLDial
ect
```

```
#secrets
spring.profiles.include=env, oauth, pay

#swagger
springdoc.api-docs.path=/api-docs
springdoc.swagger-ui.path=/swagger-ui/index.html
#server.servlet.contexct-path=/api

#file
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=2MB
spring.servlet.multipart.max-request-size=10MB
```

- application-env.properties

```
spring.datasource.url=jdbc:mysql://{mysql 접속 url}/jariyo?us
eSSL=false&serverTimezone=UTC
spring.datasource.hikari.username=DB접속 유저 ID
spring.datasource.hikari.password=DB접속 유저 PW


# redis
spring.data.redis.host=도메인
spring.data.redis.port=6379
spring.data.redis.password=redis 접속 PW

# OpenApi(사업자 정보 조회)
openApi.serviceKey=오픈 API 서비스 키
openApi.callBackUrl=http://api.odcloud.kr/api/nts-businessma
n/v1/status
openApi.dataType=JSON

# JWT
jwt.access.header=Authorization
# 1h(60m) (1000L(ms -> s) * 60L(s -> m) * 60L(m -> h))
jwt.access.expiration=3600000
jwt.secretKey=512비트(64바이트) 이상의 키
#  (1000L(ms -> s) * 60L(s -> m) * 60L(m -> h) * 24L(h ->
??) * 14(2?))
jwt.refresh.expiration=1209600000
jwt.refresh.header=Authorization-refresh
```

```
jwt.token-validity-in-seconds=86400

# AWS S3 Service bucket
cloud.aws.s3.bucket=jariyo-s3
cloud.aws.region.static=ap-northeast-2
cloud.aws.stack.auto=afalse
aws.access.key=s3 접속 유저 키
aws.secret.key=s3 접속 보안 키
aws.region=ap-northeast-2

#mail
email.address=이메일 보낼 주소
email.password=gmail 이메일 키

#cors allowed origins
cors.allowedOrigins=http://localhost:5173,http://localhost:8
282,http://i10b201.p.ssafy.io:3000,http://i10b201.p.ssafy.i
o,https://i10b201.p.ssafy.io:3000,https://i10b201.p.ssafy.i
o,http://b201-front.hyegpfud.duckdns.org,https://b201-front.
hyegpfud.duckdns.org,http://b201-back.hyegpfud.duckdns.org,h
ttps://b201-back.hyegpfud.duckdns.org
```

- application-oauth.properties

```
# === application-oauth.properties ===

# GOOGLE
spring.security.oauth2.client.registration.google.client-id=
클라이언트id
spring.security.oauth2.client.registration.google.client-sec
ret=클라이언트secret
spring.security.oauth2.client.registration.google.redirect-u
ri=http://localhost:8080/login/oauth2/code/google (설정한 red
irect URI)
spring.security.oauth2.client.registration.google.scope=prof
ile, email

# NAVER
spring.security.oauth2.client.registration.naver.client-id=
클라이언트id
spring.security.oauth2.client.registration.naver.client-secr
```

```
et=클라이언트secret
spring.security.oauth2.client.registration.naver.redirect-ur
i=http://localhost:8080/login/oauth2/code/naver (설정한 redir
ect URI)
spring.security.oauth2.client.registration.naver.authorizati
on-grant-type=authorization_code
spring.security.oauth2.client.registration.naver.scope=name,
email, profile_image
spring.security.oauth2.client.registration.naver.client-name
=Naver


# KAKAO
spring.security.oauth2.client.registration.kakao.client-id=
클라이언트id
spring.security.oauth2.client.registration.kakao.client-secr
et=클라이언트secret
spring.security.oauth2.client.registration.kakao.redirectUri
=http://localhost:8080/login/oauth2/code/kakao (설정한 redire
ct URI)
spring.security.oauth2.client.registration.kakao.scope=profi
le_nickname, profile_image, account_email
spring.security.oauth2.client.registration.kakao.client-auth
entication-method=client_secret_post
spring.security.oauth2.client.registration.kakao.authorizati
onGrantType=authorization_code
spring.security.oauth2.client.registration.kakao.client-name
=Kakao

# Naver & KAKAO Provider 설정
spring.security.oauth2.client.provider.naver.authorization_u
ri=https://nid.naver.com/oauth2.0/authorize
spring.security.oauth2.client.provider.naver.token_uri=http
s://nid.naver.com/oauth2.0/token
spring.security.oauth2.client.provider.naver.user-info-uri=h
ttps://openapi.naver.com/v1/nid/me
spring.security.oauth2.client.provider.naver.user_name_attri
bute=response
spring.security.oauth2.client.provider.kakao.authorization-u
ri=https://kauth.kakao.com/oauth/authorize
spring.security.oauth2.client.provider.kakao.token-uri=http
s://kauth.kakao.com/oauth/token
spring.security.oauth2.client.provider.kakao.user-info-uri=h
```

```
ttps://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attri
bute=id
```

- 스프링에서 구글, 깃허브, 페이스북 등의 Provider는 제공해주지만,

  네이버, 카카오 Provider를 제공해주지 않으므로 직접 설정

- application-pay.properties

```
pay.admin-key = 카카오 pay admin 키
pay.ready.host = https://kapi.kakao.com/v1/payment/ready
pay.approve.host = https://kapi.kakao.com/v1/payment/approve
pay.success.url=http://{back서버 도메인}/api/zpass/payment/suc
cess
pay.fail.url=http://{back서버 도메인}/api/zpass/payment/fail
pay.cancle.url=http://{back서버 도메인}/api/zpass/payment/canc
el
```

- 각 소셜별 어플리케이션 생성

  ▼ 카카오

  https://developers.kakao.com/console/app

  ▼ 네이버

  https://developers.naver.com/apps/#/register

  ▼ 구글

  https://console.cloud.google.com/

3. EC2 서버 세팅하기

1) docker-compose로 기본 세팅하기

```
version: '3'
services:
  nginxproxymanager:
    image: 'jc21/nginx-proxy-manager:latest'
    container_name: nginxproxymanager
    restart: unless-stopped
    ports:
```

```
      # These ports are in format <host-port>:<container-port>
      - '80:80' # Public HTTP Port
      - '443:443' # Public HTTPS Port
      - '10081:81' # Admin Web Port

    volumes:
      - ./docker-compose/data:/data
      - ./docker-compose/letsencrypt:/etc/letsencrypt
    networks:
      - b201

  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    ports:
      - "8000:8000"
      - "9443:9443"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /home/ubuntu/workspace/portainer-data:/data
    restart: always
    networks:
      - b201

  jenkins:
    container_name: jenkins
    image: jenkins/jenkins:latest
    user: "${UID}:${GID}"
    ports:
      - "8080:8080"
    environment:
            #- JENKINS_OPTS=--prefix=/jenkins
      - TZ=Asia/Seoul
    restart: on-failure
    volumes:
      - /home/ubuntu/workspace/jenkins-data:/var/jenkins_home
    networks:
      - b201
  db:
    image: mysql:8.0.35
    container_name: mysql-server
    ports:
      - "3307:3306"
```

```
    environment:
      MYSQL_ROOT_PASSWORD: ssafy
      MYSQL_DATABASE: jariyo
      MYSQL_USER: ssafy
      MYSQL_PASSWORD: ssafy
      TZ: Asia/Seoul
    command: # 명령어 실행
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - /home/ubuntu/workspace/mysql-data:/var/lib/mysql
    networks:
      - b201

  redis:
    image: redis
    ports:
      - "6379:6379"
    container_name: redis
    restart: always
    command: redis-server --requirepass ssafy --port 6379
    networks:
      - b201

  sonarqube:
    image: sonarqube:latest
    container_name: sonarqube
    environment:
      - SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true
    ports:
      - "9000:9000"
    networks:
      - b201

networks:
  b201:
    external: true
```

EC2 서버 내 docker-compose.yml 파일을 위치시킨 후 해당 경로에서 다음 명령어 수행

```
$ docker compose up -d
```

2) NginxProxyManager 설정

SSL Certificates

Search Certificate... | Add SSL Certificate

NAME	CERTIFICATE PROVIDER	EXPIRES
*.hyegpfud.duckdns.org Created: 6th February 2024	Let's Encrypt - DuckDNS	6th May 2024, 12:45 am
b201-back.hyegpfud.duckdns.org Created: 6th February 2024	Let's Encrypt - DuckDNS	6th May 2024, 7:03 am
b201-front.hyegpfud.duckdns.org Created: 6th February 2024	Let's Encrypt - DuckDNS	6th May 2024, 5:41 am
hyegpfud.duckdns.org Created: 6th February 2024	Let's Encrypt - DuckDNS	6th May 2024, 4:01 am
jenkins.hyegpfud.duckdns.org Created: 6th February 2024	Let's Encrypt - DuckDNS	6th May 2024, 12:49 am
portainer.hyegpfud.duckdns.org Created: 6th February 2024	Let's Encrypt - DuckDNS	6th May 2024, 12:47 am
sonarqube.hyegpfud.duckdns.org Created: 7th February 2024	Let's Encrypt - DuckDNS	7th May 2024, 2:37 am

→ SSL Certificates를 통해 let's encrypt 인증서 발급

Proxy Hosts

Search Host... | Add Proxy Host

SOURCE	DESTINATION	SSL	ACCESS	STATUS
b201-back.hyegpfud.duckdns.org Created: 6th February 2024	http://b201-back:8282	Let's Encrypt	Public	Online
b201-front.hyegpfud.duckdns.org Created: 6th February 2024	http://b201-front:80	Let's Encrypt	Public	Online
hyegpfud.duckdns.org Created: 6th February 2024	http://i10b201.p.ssafy.io:10081	Let's Encrypt	Public	Online
jenkins.hyegpfud.duckdns.org Created: 6th February 2024	http://jenkins:8080	Let's Encrypt	Public	Online
portainer.hyegpfud.duckdns.org Created: 6th February 2024	https://portainer:9443	Let's Encrypt	Public	Online
sonarqube.hyegpfud.duckdns.org Created: 7th February 2024	http://sonarqube:9000	Let's Encrypt	Public	Online

→ front 서버, back 서버, NginxProxyManager, Jenkins, Portainer, Sonarqube 프록시 설정

3) MySQL

- MySQL Workbench에서 접속 확인

```
IP : 해당 EC2 서버 IP
ID : ssafy
PW : ssafy
```

- Data Dump 하기
  - dump 파일 docker mysql-server 내부 /tmp에 복사

```
$ docker cp 덤프파일.sql [mysql 컨테이너 id or name]:/tmp
```

  - mysql-server에 접속

```
$ docker exec -it [컨테이너 id or name] bash
```

  - dump 파일 복원

```
bash# mysql -u root -p DB이름 < /tmp/덤프파일.sql
bash# exit
```

4) Jenkins 초기 설정

- jenkins 초기 비밀번호 확인

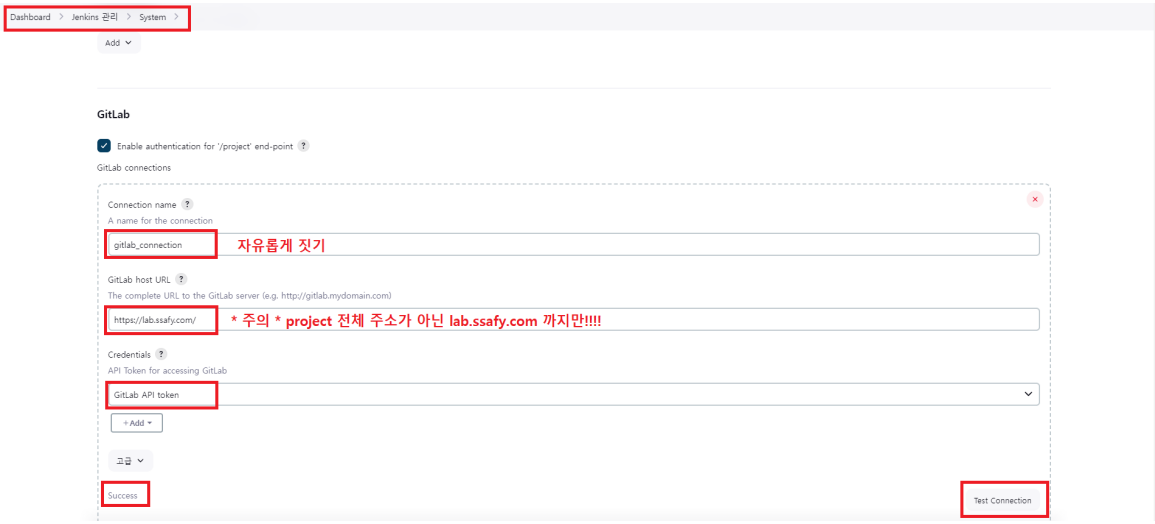
```
$ sudo docker logs jenkins

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been creat
ed and a password generated.
Please use the following password to proceed to installation:

...............................(키값 자리)

This may also be found at: /var/jenkins_home/secrets/initialAdm
inPassword

*************************************************************
```

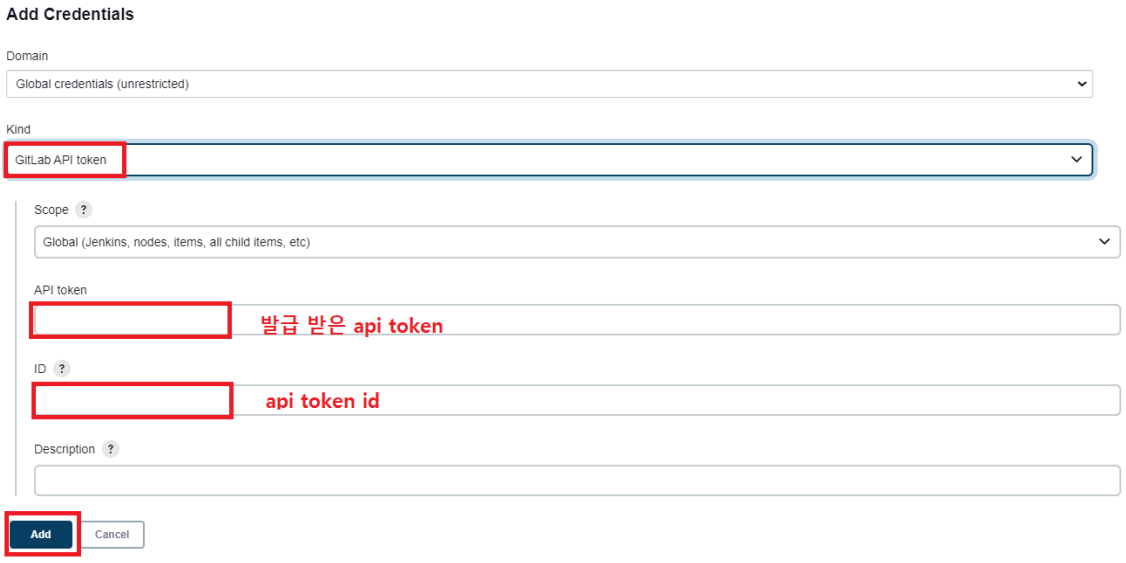
```
*************************************************************
*************************************************************
```

- 웹에서 도메인:8080 으로 접속
- 복사한 Key삽입
- Install suggested plugins(왼쪽)버튼 선택
- 플러그인이 성공적으로 모두 설치되었으면
- admin 계정 생성(모두기입)

Jenkins gitlab 토큰 발급 및 등록

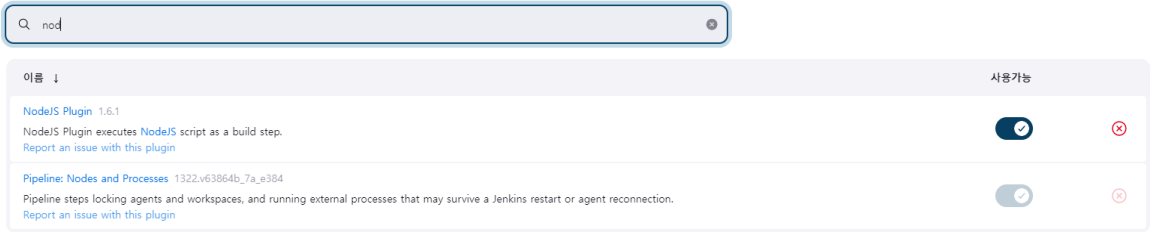
- Dashboard > Jenkins 관리 > System





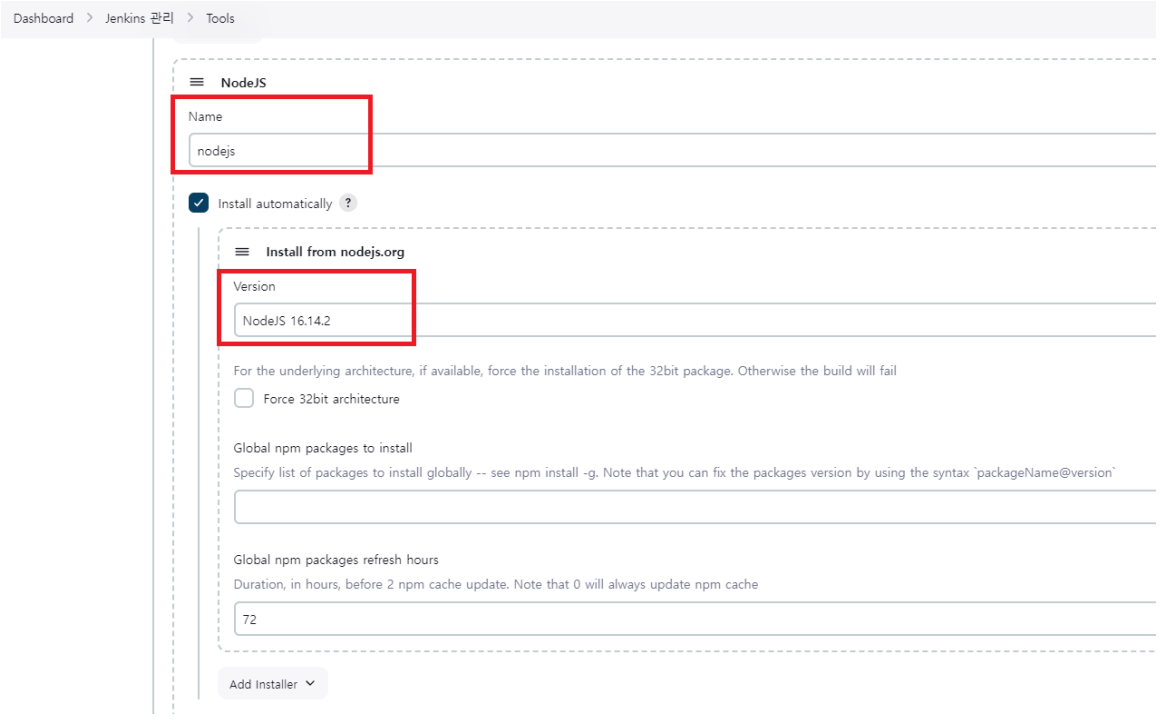
NodeJS 플러그인 설치

- Plugin 설치



NodeJS Plugin 설치 후 확인

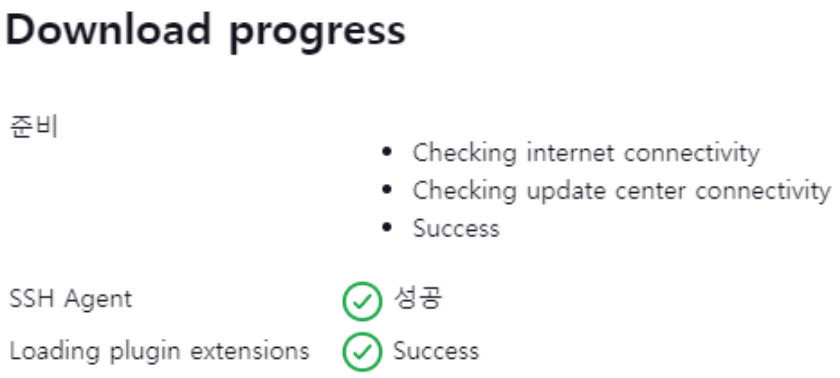
- Tools에 NodeJS 사용 버전 설정



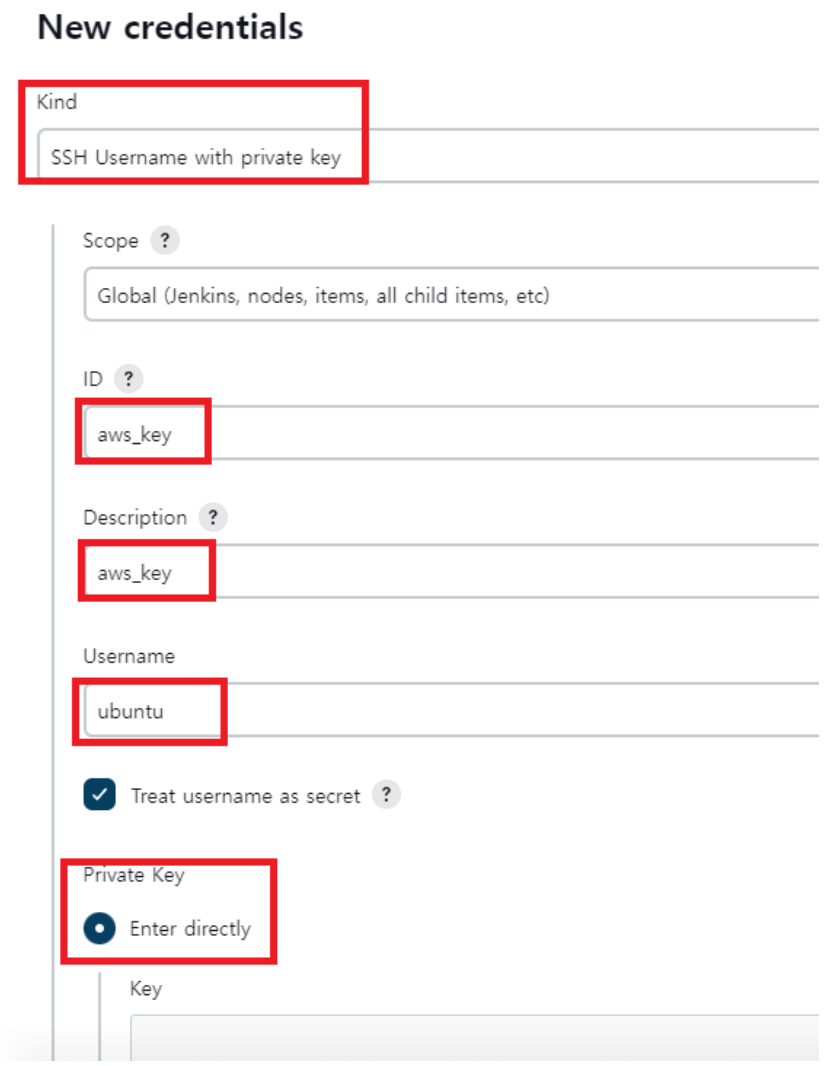
→ 사용할 버전 선택

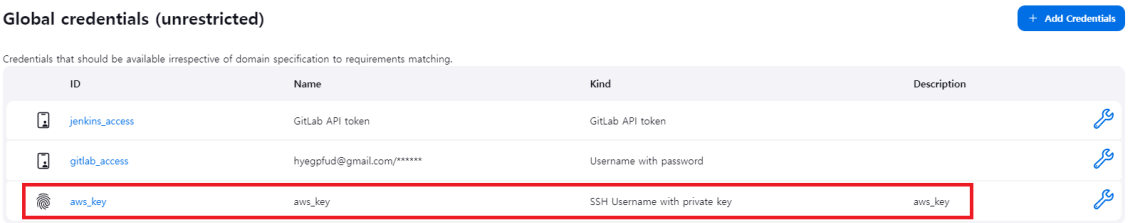
SSH Agent Plugin 설치

- SSH Agent Plugin 설치



- Credentials 등록





5) OpenVidu 세팅

i. OpenVidu 서버 패키지 설치

On premises - OpenVidu Docs

OpenVidu is deployed in production as a set of Docker containers managed with Docker Compose.
You can deploy OpenVidu in any modern Linux distribution.

https://docs.openvidu.io/en/stable/deployment/ce/on-premises/

1. curl을 사용하여 openvidu 설치 패키지 다운로드(최신버전 : 2.29.0)

```
$ cd /opt
$ sudo curl https://s3 eu west 1.amazonaws.com/aws.openvidu.io/ins
tall_openvidu_latest.sh | sudo bash
```

2. 설치된 패키지 폴더의 .env 수정
   - 반드시 SSL 필요함 : 외부 브라우저에서 WebRTC를 사용할 때 https가 아니면 카메라, 마이크의 장치에 대한 접근 제한이 일어남, 때문에 Nignx에 SSL를 적용하여 https(443)을 사용할 수 있도록 설

```
$ cd /opt/openvidu
$ sudo vim .env
```

```
DOMAIN_OR_PUBLIC_IP= {사용하고 있는 컴퓨터의 도메인 주소}
OPENVIDU_SECRET= { 비밀번호 }
# sertbot을 통해 letsencrypt 방식으로 발급받은 ssl key들 아래와 같은 위치에
# /etc/letsencrypt/live/teddysopenvidu.kro.kr/fullchain.pem, privkey
# 발급 받은 상태여야 함(Nginx에 적용)
CERTIFICATE_TYPE=letsencrypt
LETSENCRYPT_EMAIL= { 이메일 주소}
```

```
# Nginx에게 넘겨줄 포트 값
HTTP_PORT=80
HTTPS_PORT=443
```

3. openvidu 패키지 실행

```
$ sudo ./openvidu start
```

```
ubuntu@teddysopenvidu:/opt/openvidu$ sudo ./openvidu start
[+] Running 5/0
 ✔ Container openvidu-kms-1                Running
 ✔ Container openvidu-nginx-1              Running
 ✔ Container openvidu-app-1                Running
 ✔ Container openvidu-openvidu-server-1    Running
 ✔ Container openvidu-coturn-1             Running
```

```
ubuntu@teddysopenvidu:/opt/openvidu$ sudo docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED       STATUS                 PORTS
                                                                   NAMES
c28fe9517057   portainer/portainer-ce:latest        "/portainer"             2 weeks ago   Up 2 weeks             0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 8
000/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp                 portainer
14440be91dda   openvidu/openvidu-coturn:2.29.0      "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks             0.0.0.0:3478->3478/tcp, 0.0.0.0:3478->3478/u
dp, :::3478->3478/tcp, :::3478->3478/udp, 5349/tcp, 5349/udp   openvidu-coturn-1
1bfdb5727cca   openvidu/openvidu-proxy:2.29.0       "/docker-entrypoint.…"   2 weeks ago   Up 2 weeks
                                                                   openvidu-nginx-1
cf70125e8dcb   kurento/kurento-media-server:7.0.1   "/entrypoint.sh"         2 weeks ago   Up 2 weeks (healthy)
                                                                   openvidu-kms-1
a03b274abd40   openvidu/openvidu-call:2.29.0        "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks
                                                                   openvidu-app-1
c64c3a5d2709   openvidu/openvidu-server:2.29.0      "/usr/local/bin/entr…"   2 weeks ago   Up 2 weeks
                                                                   openvidu-openvidu-server-1
```

4. docker containers 확인(sudo docker ps -a)
    - kms : Kurento Media Server(WebRTC 미디어 서버)
    - nginx : Web Server
    - app : OpenVidu 제공하는 웹 서비스(사용하지 않기 때문에 삭제)
    - openvidu-server : kms를 사용하여 WebRTC 서비스 제공
    - coturn : TURN/STUN Server

ii. OpenVidu Signaling Back-End Server (참고 : openvidu-basic-java)

개발 환경

- Spring boot 3.2.2
- Gradle 8.5
- Java 17.0.9
- Openvidu 2.29.0

1. build.gradle(의존성 추가)

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    testImplementation 'org.springframework.boot:spring-boot-start
er-test'

    // https://mvnrepository.com/artifact/org.springframework.boo
t/spring-boot-starter-web
    implementation 'org.springframework.boot:spring-boot-starter-w
eb:3.2.2'
    // https://mvnrepository.com/artifact/io.openvidu/openvidu-jav
a-client
    implementation 'io.openvidu:openvidu-java-client:2.29.0'

}
```

2. Spring boot SSL 생성

```
# keystore.p12 (Spring boot 용 ssl키 생성)
$ openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out
keystore.p12 -name tomcat -CAfile chain.pem -caname root
```

3. application.properties(SSL 적용)

```
server.port: 5000

server.ssl.key-store=classpath:keystore.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=ssafy

OPENVIDU_URL: { openvidu 서버 주소}
OPENVIDU_SECRET: { openvidu 서버 비밀번호}
```

4. CORS 처리

```
# CORS 어노테이션
@CrossOrigin(origins = "https://i10b201.p.ssafy.io:8081", allowedHea
```

iii. OpenVidu Front-End Server (참고 : openvidu-vue)

개발환경

- Vue3 v3.3.11 (Vite v5.0.12)
- Node.js v20.11.0

1. 의존성 추가

```
"dependencies": {
    ...
    "openvidu-browser": "^2.29.1",
        ...
  },
```

2. Nginx를 통해 SSL 적용하여 Https로 접근하도록 설정이 필요함

4. 서비스 이용을 위한 배포

1) 배포를 위한 배포 폴더 설정

```
# 배포를 위한 폴더 위치에 deploy 폴더 및 하위에 front, back 폴더 생성
$ mkdir deploy
$ cd deploy
$ mkdir front
$ mkdir back
```

2) 작성

```
$ pwd
/home/ubuntu/deploy
```

```
# deploy 폴더 내부에 생성
$ vi deploy.sh
pipeline{
    agent any

  tools {nodejs "nodejs"}

    stages{
        stage('Clone'){
            steps{
                git branch: 'master', credentialsId: 'gitlab_acces
s', url: 'https://lab.ssafy.com/s10-webmobile1-sub2/S10P12B201.gi
t'
            }
        }

        stage('NPM Build') {
            steps {
                dir('frontend') {
                    sh '''
                        cp /var/jenkins_home/workspace/.env .
                        chmod +x .env
                    '''
                    sh 'npm install; npm run build'
                }
            }
        }

        stage('Gradle Build'){
            steps{
                dir('backend') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew build --exclude-task test'
                }
            }
        }

        stage('Deploy') {
            steps {
                sshagent(credentials: ['aws_key']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@i10
```

```
b201.p.ssafy.io
                        scp -C /var/jenkins_home/workspace/S10P12B
201_pipeline/backend/build/libs/jariyo-0.0.1-SNAPSHOT.jar ubuntu@i
10b201.p.ssafy.io:/home/ubuntu/deploy/back
                        scp -r /var/jenkins_home/workspace/S10P12B
201_pipeline/frontend/dist/ ubuntu@i10b201.p.ssafy.io:/home/ubunt
u/deploy/front
                        ssh -o StrictHostKeyChecking=no ubuntu@i10
b201.p.ssafy.io chmod +x /home/ubuntu/deploy/deploy.sh
                        ssh -o StrictHostKeyChecking=no ubuntu@i10
b201.p.ssafy.io /home/ubuntu/deploy/deploy.sh << y
                    '''
                }
            }
        }
    }
}
```

3) front

- deploy/front 위치에서 진행

```
$ pwd
/home/ubuntu/deploy/front
```

- Dockerfile

```
FROM nginx:1.21

COPY dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

- docker-compose.yml

```
version: '3'
services:
  web:
    image: docker-vue:0.1
    container_name: b201-front
    ports:
      - "3001:80"
      - "5173:80"
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /home/ubuntu/deploy/front/dist:/usr/share/nginx/html
    networks:
      - b201

networks:
  b201:
    external: true
```

4) back

- deploy/back 위치에서 진행

```
$ pwd
/home/ubuntu/deploy/back
```

- Dockerfile

```
FROM openjdk:17-jdk
LABEL maintainer="hye"

WORKDIR /app

COPY ./jariyo-0.0.1-SNAPSHOT.jar /app/

COPY ./application.properties /app/
COPY ./application-env.properties /app/
COPY ./application-oauth.properties /app/
COPY ./application-pay.properties /app/
```

```
EXPOSE 8282

CMD ["java", "-jar", "jariyo-0.0.1-SNAPSHOT.jar", "--spring.confi
g.location=/app/"]
```

→ application.properties, application-env.properties, application-oauth.properteis, application-pay.properties 파일을 deploy 폴더 내부에 옮겨두기

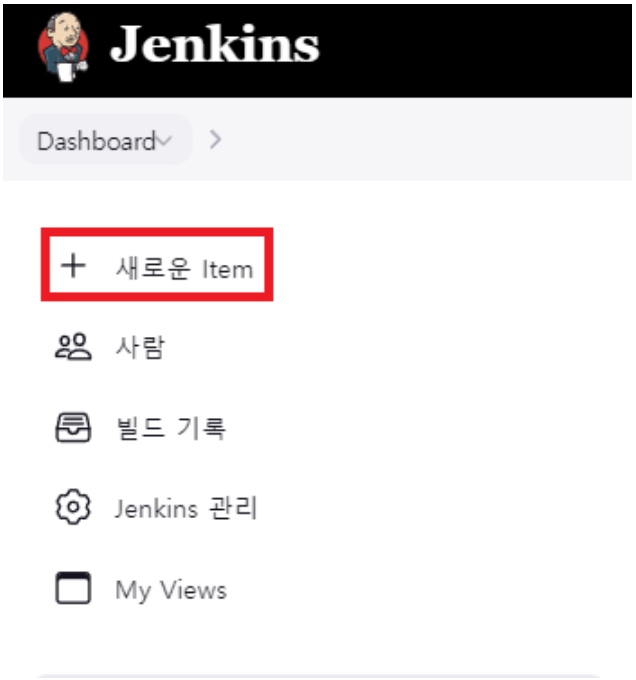
- docker-compose.yml

```
version: '3'

services:
  app:
    image: docker-springboot:0.1
    container_name: b201-back
    ports:
       - 8282:8282
    environment:
       - SPRING_PROFILES_ACTIVE=dev
       - TZ=Asia/Seoul
    networks:
       - b201

networks:
  b201:
    external: true
```

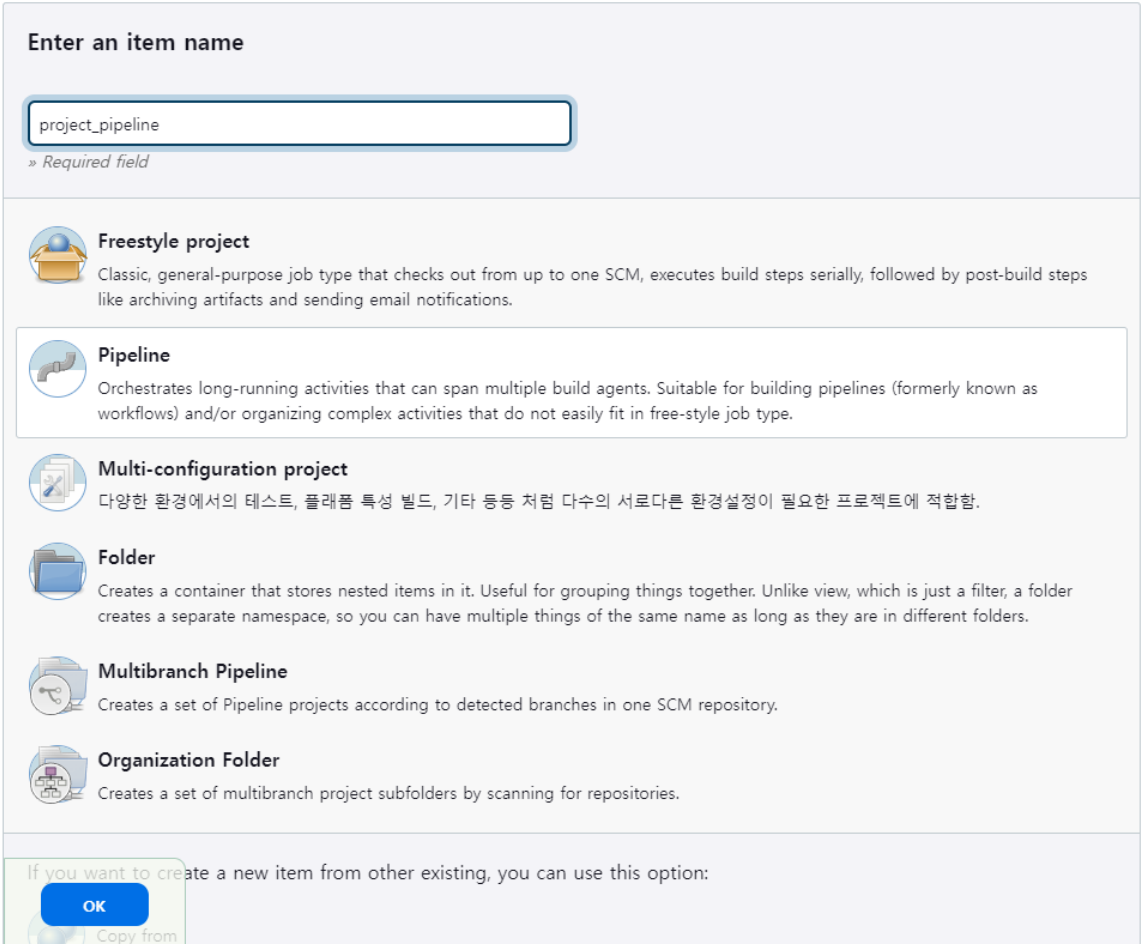
5) Jenkins 설정

1) Item 생성

- Item 생성하기



- Pipeline 타입 선택

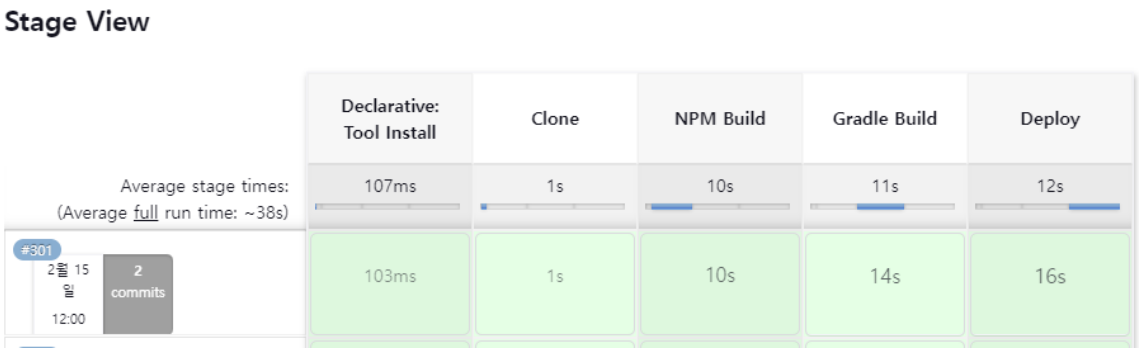


- pipeline 작성

```
pipeline{
    agent any

    tools {nodejs "nodejs"}

    stages{
        stage('Clone'){
            steps{
                git branch: 'master', credentialsId: 'gitlab_ac
cess', url: 'https://lab.ssafy.com/s10-webmobile1-sub2/S10P12B2
01.git'
            }
        }

        stage('NPM Build') {
            steps {
                dir('frontend') {
                    // echo 'VITE_API_URL=http://i10b201.p.ssaf
y.io:8282' >> .env
                    sh '''
                        cp /var/jenkins_home/workspace/.env .
                        chmod +x .env
                    '''
                    sh 'npm install; npm run build'
                }
            }
        }

        stage('Gradle Build'){
            steps{
                dir('backend') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew build --exclude-task test'
                }
            }
        }
```

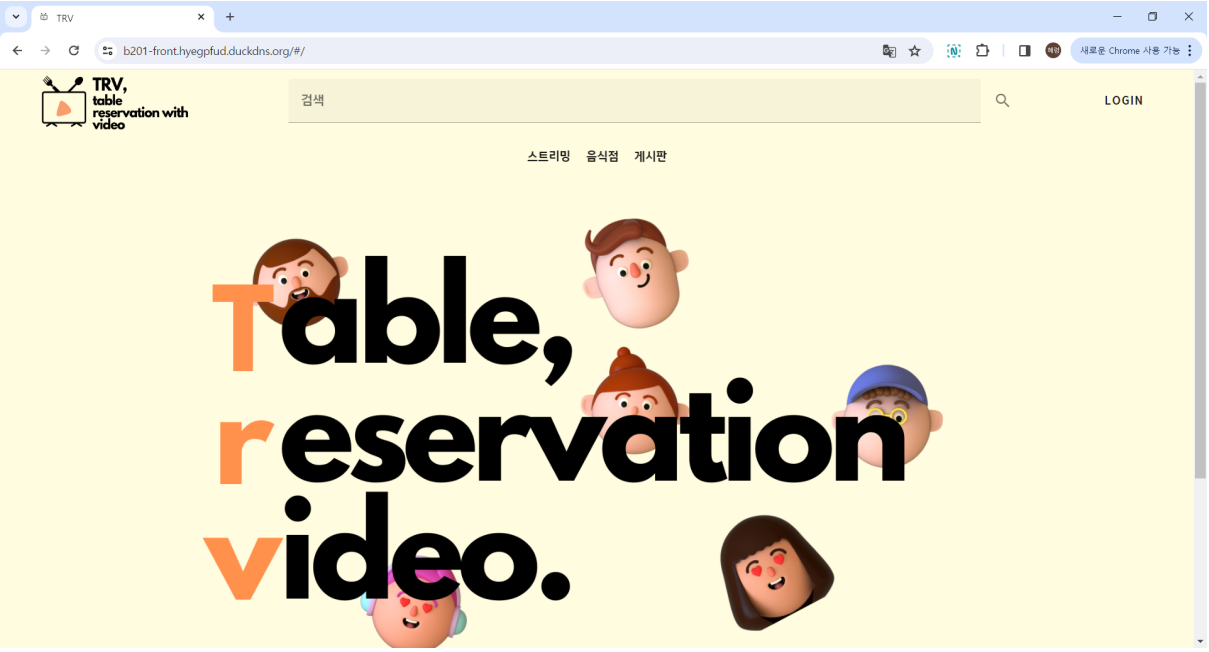
```
        stage('Deploy') {
            steps {
                sshagent(credentials: ['aws_key']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@
i10b201.p.ssafy.io
                        scp -C /var/jenkins_home/workspace/S10P
12B201_pipeline/backend/build/libs/jariyo-0.0.1-SNAPSHOT.jar ub
untu@i10b201.p.ssafy.io:/home/ubuntu/deploy/back
                        scp -r /var/jenkins_home/workspace/S10P
12B201_pipeline/frontend/dist/ ubuntu@i10b201.p.ssafy.io:/home/
ubuntu/deploy/front
                        ssh -o StrictHostKeyChecking=no ubuntu@
i10b201.p.ssafy.io chmod +x /home/ubuntu/deploy/deploy.sh
                        ssh -o StrictHostKeyChecking=no ubuntu@
i10b201.p.ssafy.io /home/ubuntu/deploy/deploy.sh << y
                    '''
                }
            }
        }

    }
}
```

- 빌드 성공 시 화면



5. 접속

도메인으로 접속할 경우 다음과 같은 메인 화면이 뜨면 성공


TRV
b201-front.hyegpfud.duckdns.org/#/
새로운 Chrome 사용 가능
TRV,
table
reservation with
video
검색
LOGIN
스트리밍
음식점
게시판
Table,
reservation
video.