StellarVision 포팅 매뉴얼

1. 개발 환경

- FrontEnd
    - Vue.js `3.5.17`
    - Babylon.js `8.18`
    - Node.js `22.17.1`
    - Visual Studio Code `1.103.0`
    - Blender `4.4`
    - Dependency
        - axios `1.11.0`
        - pinia `3.0.3`
        - tailwindcss `4.1.11`
        - eslint `9.29.0`
        - openvidu-browser `2.31.0`
        - vite `7.0.0`
- BackEnd
    - Java `jdk17`
    - Spring Boot `3.5.3`
    - InteliJIDEA `2025.1.4.1`
    - Tomcat `10.1.42`
    - Dependency
        - gradle `8.14.3`
        - spring-data-jpa `3.5.3`
        - spring-security `6.5.1`
        - spring-web `6.2.8`
        - lombok `1.18.38`

        - spring data redis `3.5.1`
        - openvidu `2.31.0`
- AI
    - Python `3.11`
    - Fastapi `0.115.14`
    - PyTorch `2.7.1`
    - PyCharm `2022.3`
    - Dependency
        - ultralytics `8.3.171`
        - realesrgan `0.3.0`
        - uvicorn `0.35.0`
        - realesrgan-ncnn-py `2.0.0`
- Database
    - MySQL `8.0.42`
    - Redis `8.2`
    - AWS S3 `2.25.26`
- Infra
    - Ubuntu `22.04`
    - Jenkins `2.504.3`
    - Docker `28.3.2`
    - Nginx `1.18.0`

2. 포트 정보
    - Jenkins : `8081:8080`
    - Openvidu: `8443:8443`
    - Backend: `8080:8080`
    - Frontend: `8085:80`
    - AI: `8000:8000`

3. UFW 허용 포트 설정
    - `22`
    - `80`
    - `44`
    - `8080`
    - `443`
    - `8081`
    - `4443`
    - `8443`
    - `8442`
    - `8000`
    - `40000:57000`
    - `57000:65535`
    - `8085`
    - `8001`

Ubuntu

EC2 접속

```
ssh -i I13C106T.pem ubuntu@i13c106.p.ssafy.io
```

서버 세팅

```
sudo timedatectl set-timezone Asia/Seoul
sudo apt-get -y update && sudo apt-get -y upgrade
```

Docker

도커 설치

```
# Uninstall all conflicting packages
for pkg in docker.io docker-doc docker-compose podman-docker containe
rd runc; do sudo apt-get remove $pkg; done

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dea
rmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/do
cker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# Install the latest version
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-pl
ugin docker-compose-plugin
```

도커 그룹에 USER 추가

```
sudo usermod -aG docker USERNAME
```

도커 네트워크 추가

```
docker network create docker-network
```

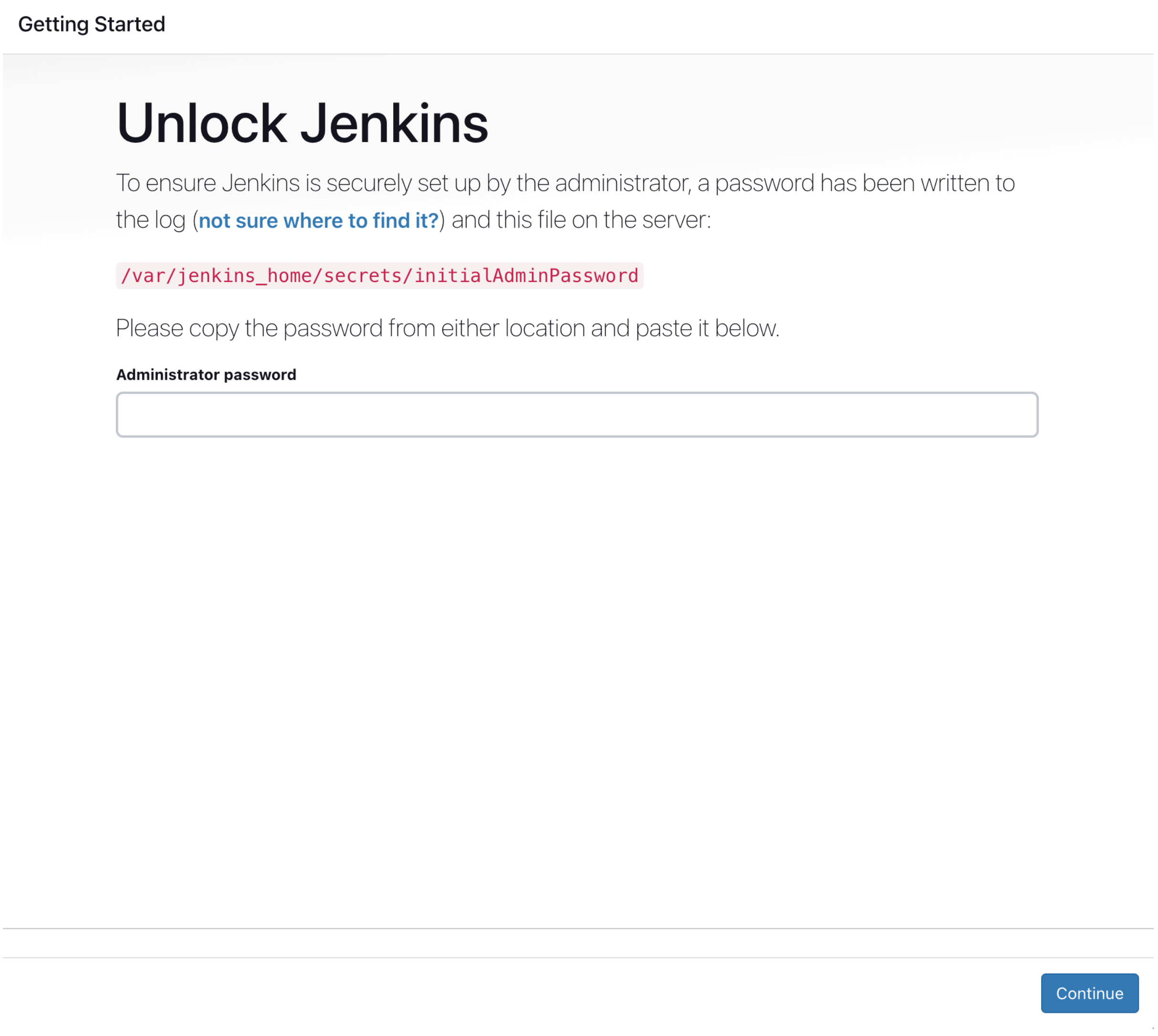
Jenkins 설치

```
docker pull jenkins/jenkins:jdk17
docker run --privileged -d -p 8081:8080 -p 50000:50000 --name jenkins j
enkins/jenkins:jdk17
```

- privileged 모드로 실행하여 Docker In Docker 준비

Jenkins 접속

http://i13c106.ssafy.io:8081 접속하여 젠킨스 실행


Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

- Administrator Password는 젠킨스 컨테이너에 존재

```
docker exec -it jenkins /bin/bash
cat /var/jenkins_home/secrets/initialAdminPassword
```

- 찾은 패스워드 입력
- Install Suggested Plugins 선택
- Admin 계정 생성

Jenkins 환경 설정

```
# Jenkins 컨테이너 종료
sudo docker stop jenkins
# Jenkins 데이터가 있는 디렉토리에 update-center-rootCAs 하위 디렉토리 생성
sudo mkdir /jenkins/update-center-rootCAs
# CA 파일 다운로드
sudo wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootC
A/u
pdate-center.crt -O /jenkins/update-center-rootCAs/update-center.crt
# Jenkins 플러그인 다운로드 시 미러사이트로 대체될 수 있도록 설정
sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.gi
thubusercontent.com/lework/jenkins-update-center/master/updates/tence
nt/upd
ate-center.json#' /jenkins/hudson.model.UpdateCenter.xml
# Jenkins 컨테이너 재시작
sudo docker restart jenkins
```

Jenkins Pipeline 플러그인 설치

- Discord Notifier
- Generic Webhook Trigger
- Git plugin
- GitLab Plugin
- Gradle Plugin
- NodeJS Plugin
- Publish Over SSH

- SSH Agent Plugin

젠킨스 설정

Jenkins 관리 - System

- GitLab connections
- Publish over SSH
    - Jenkins SSH Key
        - Passphrase
        - Key
        - SSH Servers
            - name: deploy
            - Hostname: 172.26.12.37
            - Username: ubuntu

Jenkins 관리 - Tools

- Git installations
    - Path to Git executable: git
- Gradle installations
    - Install automatically
        - Version: 8.14.3
- NodeJS installations
    - Install automatically
        - Version: 22.17.1

Jenkins Docker In Docker 설정

→ 젠킨스에서 도커 이미지를 빌드하기 위해 젠킨스 컨테이너 내부에 도커 설치하는 과정

젠킨스 컨테이너 접속

```
docker exec -itu 0 jenkins /bin/bash
```

도커 설치

```
# Docker 설치
## - Old Version Remove
apt-get remove docker docker-engine docker.io containerd runc

## - Setup Repo
apt-get update

apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor
-o /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/doc
ker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/n
ull

## - Install Docker Engine
apt-get update

apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plug
in
```

도커 데몬 실행 및 도커 그룹에 root 추가

```
service docker start
usermod -aG docker root
su - root
id -nG #root 추가되었는지 확인. "root docker"가 뜨면 정상
```

docker.sock 권한 변경 및 root에서 도커 로그인

```
chmod 666 /var/run/docker.sock
su - root
docker login
```

Nginx 설정

Nginx 설치(EC2)

```
sudo apt-get -y install nginx
```

Certbot SSL 인증서 발급

```
# certbot 다운로드
sudo snap install --classic certbot
# repository에 certbot 설치
sudo apt-add-repository -r ppa:certbot/certbot
# python-certbot-nginx 설치
sudo apt-get -y install python3-certbot-nginx
# SSL 인증서 발급
sudo certbot --nginx -d i13c106.p.ssafy.io
```

EC2 서버 Nginx 리버스 프록시 설정

- i13c106.p.ssafy.io.conf

```
server {
    listen 80;
    server_name i13c106.p.ssafy.io www.i13c106.p.ssafy.io;
    root /var/www/i13c106.p.ssafy.io/public_html;
    index index.html;
    location / {
        try_files $uri $uri/ =404;
    }
}

server {
    listen 8081 ssl;
    server_name i13c106.p.ssafy.io;

    ssl_certificate     /etc/letsencrypt/live/i13c106.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i13c106.p.ssafy.io/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_ecdh_curve X25519:secp384r1:secp521r1:prime256v1;

    location / {
        proxy_pass http://localhost:8088;
        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 https;
        proxy_set_header X-Forwarded-Port 443;
        proxy_redirect http://localhost:8080 i13c106.p.ssafy.io;
        add_header 'X-SSH-Endpoint' 'jenkins.domain.tld:50022' always;
    }
}

map $http_origin $cors_origin {
    default "";
    "https://i13c106.p.ssafy.io" $http_origin;
}

server {
    listen 443 ssl;
    server_name i13c106.p.ssafy.io www.i13c106.p.ssafy.io;
    client_max_body_size 50M;
    ssl_certificate     /etc/letsencrypt/live/i13c106.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i13c106.p.ssafy.io/privkey.pem;

    location /api/ {
        proxy_pass http://localhost:8080/api/;
        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;
    }
    #ec2 ai서버(yolo모델용)
    location /api/detect/{
            proxy_pass http://localhost:8000/api/detect/;
            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;

            add_header 'Access-Control-Allow-Origin' 'http://localhost:5173' always;
            add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE' always;
            add_header 'Access-Control-Allow-Headers' 'Authorization, Content-Type' always;
            add_header 'Access-Control-Allow-Credentials' 'true' always;
    }

    location / {
        proxy_pass http://localhost:8085;
        add_header 'Access-Control-Allow-Origin' 'https://i13c106.p.ssafy.io' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE' always;
        add_header 'Access-Control-Allow-Headers' 'Authorization,Content-Type' always;
        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 /openvidu/ {
        proxy_pass http://localhost:8443/;

        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_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /openvidu/api/ {
        if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Allow-Origin' $cors_origin always;
            add_header 'Access-Control-Allow-Credentials' 'true' always;
            add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DEL
ETE, OPTIONS' always;
            add_header 'Access-Control-Allow-Headers' 'Authorization, Conten
t-Type' always;
            add_header 'Access-Control-Max-Age' 86400 always;
            return 204;
        }
        proxy_pass https://localhost:8443;
        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;

        add_header 'Access-Control-Allow-Origin' $cors_origin always;
        add_header 'Access-Control-Allow-Credentials' 'true' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELET
E, OPTIONS' always;
        add_header 'Access-Control-Allow-Headers' 'Authorization, Content-T
ype' always;
    }
}
```

프론트엔드 컨테이너 Nginx 설정

```
server {
    listen       80;
    listen  [::]:80;

    server_name i13c106.p.ssafy.io;
    root /usr/share/nginx/html;

    location / {
        try_files $uri $uri/ /index.html;
    }
        location /api/ {
        proxy_pass http://container_backend:8080/api/;
        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;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

MySQL

- MySQL 8.0.43 버전 pull & run

```
docker pull mysql:8.0.43
docker run -d --name my-mysql \
  --network docker-network \
  -e MYSQL_ROOT_PASSWORD=1234 \
  -p 3306:3306 \
  mysql:8.0.43
```

Redis

```
docker pull redis
docker run -d --name my-redis --network docker-network -p 6379:6379 re
dis
```

Openvidu

- openvidu 설치

  ```
  sudo su
  cd /opt
  curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_ope
  nvidu_latest.sh | bash
  ```

- 환경변수 설정

  ```
  sudo vi /opt/openvidu/.env
  DOMAIN_OR_PUBLIC_IP=i13c106.p.ssafy.io
  # OpenVidu SECRET used for apps to connect to OpenVidu server and
  users to access to OpenVidu Dashboard
  OPENVIDU_SECRET=password

  CERTIFICATE_TYPE=letsencrypt

  # If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid ema
  il for notifications
  LETSENCRYPT_EMAIL=foo@bar.com
  HTTP_PORT=8442
  HTTPS_PORT=8443
  ```

```
OPENVIDU_CORS_ALLOWED_ORIGINS=https://i13c106.p.ssafy.io, allow
Credentials="true"
# Whether to enable recording module or not
OPENVIDU_RECORDING=true

# Use recording module with debug mode.
OPENVIDU_RECORDING_DEBUG=true

# Openvidu Folder Record used for save the openvidu recording video
s. Change it
# with the folder you want to use from your host.
OPENVIDU_RECORDING_PATH=/opt/openvidu/recordings

# System path where OpenVidu Server should look for custom recordin
g layouts
OPENVIDU_RECORDING_CUSTOM_LAYOUT=/opt/openvidu/custom-lay
out

# if true any client can connect to
# https://OPENVIDU_SERVER_IP:OPENVIDU_PORT/recordings/any_sess
ion_file.mp4
# and access any recorded video file. If false this path will be secured
with
# OPENVIDU_SECRET param just as OpenVidu Server dashboard at
# https://OPENVIDU_SERVER_IP:OPENVIDU_PORT
# Values: true | false
OPENVIDU_RECORDING_PUBLIC_ACCESS=false

# Which users should receive the recording events in the client side
# (recordingStarted, recordingStopped). Can be all (every user connect
ed to
# the session), publisher_moderator (users with role 'PUBLISHER' or
# 'MODERATOR'), moderator (only users with role 'MODERATOR') or no
ne
# (no user will receive these events)
OPENVIDU_RECORDING_NOTIFICATION=publisher_moderator

# Timeout in seconds for recordings to automatically stop (and the sess
ion involved to be closed)
# when conditions are met: a session recording is started but no user is
publishing to it or a session
# is being recorded and last user disconnects. If a user publishes within
the timeout in either case,
# the automatic stop of the recording is cancelled
# 0 means no timeout
OPENVIDU_RECORDING_AUTOSTOP_TIMEOUT=120

# Maximum video bandwidth sent from clients to OpenVidu Server, in k
bps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MAX_RECV_BANDWIDTH=1000

# Minimum video bandwidth sent from clients to OpenVidu Server, in kb
ps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MIN_RECV_BANDWIDTH=300

# Maximum video bandwidth sent from OpenVidu Server to clients, in k
bps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MAX_SEND_BANDWIDTH=1000

# Minimum video bandwidth sent from OpenVidu Server to clients, in k
bps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MIN_SEND_BANDWIDTH=300




# true to enable OpenVidu Webhook service. false' otherwise
# Values: true | false
OPENVIDU_WEBHOOK=false

# HTTP endpoint where OpenVidu Server will send Webhook HTTP PO
ST messages
# Must be a valid URL: http(s)://ENDPOINT
```

```
#OPENVIDU_WEBHOOK_ENDPOINT=

# List of headers that OpenVidu Webhook service will attach to HTTP P
OST messages
#OPENVIDU_WEBHOOK_HEADERS=

# List of events that will be sent by OpenVidu Webhook service
# Default value is all available events
OPENVIDU_WEBHOOK_EVENTS=[sessionCreated,sessionDestroyed,pa
rticipantJoined,participantLeft,webrtcConnectionCreated,webrtcConne
ctionDestroyed,recordingStatusChanged,filterEventDispatched,mediaN
odeStatusChanged,nodeCrashed,nodeRecovered]

# How often the garbage collector of non active sessions runs.
# This helps cleaning up sessions that have been initialized through
# REST API (and maybe tokens have been created for them) but have h
ad no users connected.
# Default to 900s (15 mins). 0 to disable non active sessions garbage c
ollector
OPENVIDU_SESSIONS_GARBAGE_INTERVAL=900

# Minimum time in seconds that a non active session must have been in
existence
# for the garbage collector of non active sessions to remove it. Default t
o 3600s (1 hour).
# If non active sessions garbage collector is disabled
# (property 'OPENVIDU_SESSIONS_GARBAGE_INTERVAL' to 0) this pro
perty is ignored
OPENVIDU_SESSIONS_GARBAGE_THRESHOLD=3600

# Call Detail Record enabled
# Whether to enable Call Detail Record or not
# Values: true | false
OPENVIDU_CDR=false

# Path where the cdr log files are hosted
OPENVIDU_CDR_PATH=/opt/openvidu/cdr
```

- openvidu 실행

  ```
  ./openvidu start
  ```

GitLab Connection 연결

- Jenkins관리-System에서 등록했던 GitLab Connection 선택

소스코드 관리

- Git Repositiories 연결
- Triggers: Build when a change is pushed to GitLab
  - Push Events

Excute shell

```
cp /home/application.yml ./BE/StellarVision/src/main/resources/
cd BE/StellarVision
chmod +x gradlew
./gradlew clean build
docker login -u 'dockerhub_email' -p 'dockerhub_password' docker.io
docker build -t dockerhub_id/cicd .
docker push dockerhub_id/cicd
```

빌드 후 조치

- Discord Notifier 설정
- **Send build artifacts over SSH**
  - Transfer Set
    - Source files
      - build/libs/*.jar
    - Remove prefix
      - build/libs
    - Exec command

```
docker login -u 'dockerhub_email' -p 'dockerhub_password' do
cker.io
docker pull dockerhub_id/cicd
docker stop backend && docker rm backend
docker run --name "backend" -d -p 8080:8080 --network dock
er-network dockerhub_id/cicd
```

application.yml (젠킨스 컨테이너의 /home 디렉토리에 위치)

```
spring:
  application:
    name: StellarVision

  datasource:
    url: jdbc:mysql://my-mysql:3306/stellarvision?serverTimezone=Asia/Seo
ul&characterEncoding=UTF-8
    username: root
    password: 1234
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: create # ?? create, validate, none, etc.
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
    show-sql: true

  data:
    redis:
      host: my-redis
      port: 6379

  mail:
    host: smtp.gmail.com
    port: 587
```

```yaml
    username: user.name.number
    password: password
    properties:
      mail:
        smtp:
          auth: true
          timeout: 5000
          starttls:
            enable: true
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: client-id
            client-secret: client-secret
            scope:
              - email
              - profile
cloud:
  aws:
    region:
      static: ap-northeast-2
    credentials:
      access-key: access-key
      secret-key: secret-key
    s3:
      bucket: bucket-name


openvidu:
  url: https://i13c106.p.ssafy.io:8443
  secret: secret

jwt:
  access-expmin: 60     # 1시간
  refresh-expmin: 1440  # 1일
```

```
logging:
  level:
    root: INFO
    com.susang.stellarVision.application: DEBUG    # 애플리케이션 패키지 전체
를 DEBUG로
    org.hibernate.SQL: DEBUG
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: client-id
            client-secret: client-secret
            scope:
              - email
              - profile
```

Jenkins Freestyle Item-FrontEnd

GitLab Connection 연결

- Jenkins관리-System에서 등록했던 GitLab Connection 선택

소스코드 관리

- Git Repositiories 연결
- Triggers: Build when a change is pushed to GitLab
    - Push Events

Excute shell

```
cp /home/nginx.conf ./FE/StellarVision/
cp /home/.env ./FE/StellarVision/
cd FE/StellarVision
rm -rf node_modules package-lock.json
npm cache clean --force
npm install
npm run build
docker login -u 'dockerhub_email' -p 'dockerhub_password' docker.io
docker build -t dockerhub_id/cicd:frontend .
docker push dockerhub_id/cicd:frontend
```

빌드 후 조치

- Discord Notifier 설정
- **Send build artifacts over SSH**
  - Transfer Set
    - Source files
      - dist/**
    - Remove prefix
      - dist
    - Remote Directory
      - /var/www/html
    - Exec command

      ```
      docker login -u 'dockerhub_email' -p 'dockerhub_password' do
      cker.io
      docker pull dockerhub_id/cicd:frontend
      docker stop frontend || true
      docker rm frontend || true
      docker run --name frontend -d -p 8085:80 --network docker-n
      etwork dockerhub_id/cicd:fronte
      ```

.env(젠킨스 컨테이너의 /home 디렉토리에 위치)

```
VITE_SERVICE_KEY=**********
VITE_NASA_SERVICE_KEY=**********
```

AI 별자리 탐지 서버 배포(Jenkins Freestyle Item-AI detection)

GitLab Connection 연결

- Jenkins관리-System에서 등록했던 GitLab Connection 선택

소스코드 관리

- Git Repositiories 연결
- Triggers: Build when a change is pushed to GitLab
    - Push Events

Excute shell

```
cd AI/StellarVision/detection
docker login -u 'dockerhub_email' -p 'dockerhub_password' docker.io
docker build -t dockerhub_id/cicd:ai .
docker push dockerhub_id/cicd:ai
```

빌드 후 조치

- Discord Notifier 설정
- **Send build artifacts over SSH**
    - Transfer Set
        - Source files
            - *.py requirements.txt Dockerfile
        - Exec command

          ```
          docker login -u 'dockerhub_email' -p 'dockerhub_password' do
          cker.io
          docker pull dockerhub_id/cicd:ai
          docker stop container_ai && docker rm container_ai
          docker run --name "container_ai" -d -p 8000:8000 --network d
          ocker-network jhyang00815/cicd:ai
          ```

AI 업스케일 서버 (수동배포)

시스템 패키지 설치

```
sudo add-apt-repository -y ppa:deadsnakes/ppa && \
sudo apt update && \
sudo apt install -y python3.12 python3.12-venv python3.12-distutils \
                   python3-pip nginx supervisor
```

Nginx 리버스 프록시 설정

```
server {
    listen 80;
    listen [::]:80;
    server_name susang-fastapi.my;

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

    location / {
        return 301 https://$host$request_uri;
    }
}
map $http_origin $cors_origin {
    default "";
    "http://localhost:5173" $http_origin;
    "https://i13c106.p.ssafy.io" $http_origin;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name susang-fastapi.my;

    ssl_certificate     /etc/letsencrypt/live/susang-fastapi.my/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/susang-fastapi.my/privkey.pem;


    # 1) /api는 FastAPI로 프록시 (프리픽스 유지)
    location /api/ {

        proxy_pass http://127.0.0.1:8000;   # /api/... 그대로 전달
        proxy_http_version 1.1;
        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;
        proxy_set_header Connection "";
        proxy_connect_timeout 75s;
        proxy_send_timeout 120s;
        proxy_read_timeout 300s;
        client_max_body_size 50m;

        # CORS 헤더 추가

        add_header 'Access-Control-Allow-Origin' $cors_origin always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, P
UT, DELETE' always;
        add_header 'Access-Control-Allow-Headers' 'Authorization, Content-T
ype' always;
        add_header 'Access-Control-Allow-Credentials' 'true' always;

        if ($request_method = 'OPTIONS') {
    add_header 'Access-Control-Allow-Origin' $cors_origin always;
    add_header 'Access-Control-Allow-Credentials' 'true' always;
    add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELETE,
OPTIONS' always;
    add_header 'Access-Control-Allow-Headers' 'Authorization, Content-Typ
e, X-Requested-With' always;
    add_header 'Access-Control-Max-Age' '86400' always;
    add_header 'Content-Length' '0';
    add_header 'Content-Type' 'text/plain';
    return 204;
}



      }

   # 2) 루트(/)는 간단한 헬스 페이지 또는 정적 파일로 응답
   location = / {
      return 200 "OK: susang-fastapi.my is up\n";
      add_header Content-Type text/plain;
   }

}
```

레포지토리 clone

```
git clone --depth 1 https://lab.ssafy.com/s13-webmobile1-sub1/S13P11C106
/tmp/repo
mkdir -p /home/app/StellarVision
cp -a /tmp/repo/AI/StellarVision/upscale. /home/app/StellarVision/
rm -rf /tmp/repo
```

Python 가상환경 설정

```
cd /home/app/StellarVision
python3.12 -m venv venv
source venv/bin/activate
```

의존성 설치

```
cd /home/app/StellarVision
pip install -r requirements.txt
git clone https://github.com/xinntao/Real-ESRGAN.git
cd Real-ESRGAN
# Install basicsr - https://github.com/xinntao/BasicSR
# We use BasicSR for both training and inference
pip install basicsr
# facexlib and gfpgan are for face enhancement
```

```
pip install facexlib
pip install gfpgan
pip install -r requirements.txt
python setup.py develop
```

Uvicorn 백그라운드 실행

```
nohup uvicorn main:app --host 0.0.0.0 --port 8000 \
    --workers 1 --lifespan on \
    &> uvicorn.log &
```

외부 서비스

- VITE_SERVICE_KEY : 공공데이터 포털 천문현상 정보 api (https://www.data.go.kr/data/15012691/openapi.do)
- VITE_NASA_SERVICE_KEY: NASA 오늘의 천체 api (https://api.nasa.gov/)
- aws S3 : 이미지 , 동영상 저장 클라우드