포팅메뉴얼

소유자	H4R1B0
태그	
생성 일시	@2023년 10월 4일 오전 9:35

목차

Develop Environment

Front-End

- yarn berry
- node 18.17.1 LTS
- react
  - Dependencies
    - @heroicons/react: ^2.0.18
    - @radix-ui/react-accordion: ^1.1.2
    - @radix-ui/themes: ^2.0.0
    - @react-three/drei: ^9.83.9
    - @react-three/fiber: ^8.14.1
    - @react-three/postprocessing: ^2.15.1
    - @reduxjs/toolkit: ^1.9.5
    - @tanstack/react-query: ^4.35.3
    - @tanstack/react-query-devtools: ^4.35.3
    - @types/react-modal: ^3.16.0
    - @types/three: ^0.156.0
    - @types/uuid: ^9.0.4
    - @use-gesture/react: ^10.3.0
    - audio-react-recorder: ^1.0.5
    - axios: ^1.5.0
    - msw-storybook-addon: ^1.8.0
    - react: ^18.2.0
    - react-audio-player: ^0.17.0
    - react-dom: ^18.2.0
    - react-dropzone: ^14.2.3
    - react-hot-toast: ^2.4.1
    - react-icons: ^4.11.0
    - react-modal: ^3.16.1
    - react-redux: ^8.1.2

    - react-router-dom: ^6.16.0
    - react-swipeable: ^7.0.1
    - react-use: ^17.4.0
    - three: ^0.156.1
    - three-stdlib: ^2.25.1
    - use-sound: ^4.0.1
    - uuid: ^9.0.1
    - vite-tsconfig-paths: ^4.2.1
    - zustand: ^4.4.1
- Tool: VSCode

Back-end

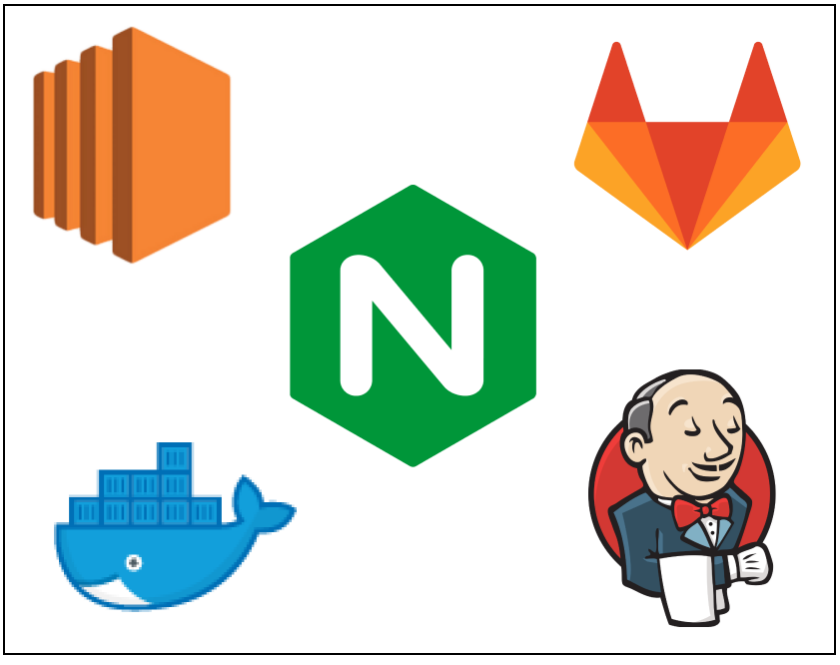
- Azul Zulu 17 (latest)
- Spring
    - Project Build: Gradle - Groovy
    - Language: Java
    - Spring Boot: 3.1.3
    - Packaging: Jar
    - Java: 17
    - Dependencies:
        - Spring Web
        - Spring Boot DevTools
        - Lombok
        - Spring Data JPA
        - Security
        - Oauth2
        - Google Cloud
        - gson

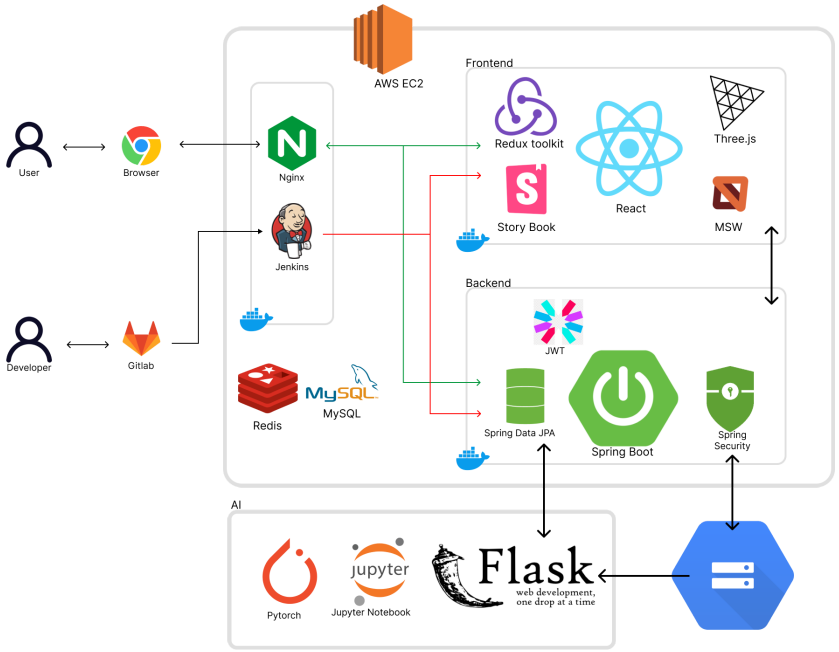
- build.gradle

```
//Common
implementation 'org.springframework.boot:spring-boot-starter-web'
implementation 'org.jetbrains:annotations:24.0.0'
implementation 'org.springframework.boot:spring-boot-starter-validation'
compileOnly 'org.projectlombok:lombok'
annotationProcessor 'org.springframework.boot:spring-boot-configuration-processor'
annotationProcessor 'org.projectlombok:lombok'
implementation 'jakarta.xml.bind:jakarta.xml.bind-api:4.0.0'
testImplementation 'org.springframework.boot:spring-boot-starter-test'

//Spring Data JPA
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

//Security
implementation 'org.springframework.boot:spring-boot-starter-security'
testImplementation 'org.springframework.security:spring-security-test'

//Oauth2
implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'

//DB
implementation 'org.springframework.boot:spring-boot-starter-data-redis'
runtimeOnly 'com.mysql:mysql-connector-j'

//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'

//GCS
implementation 'org.springframework.cloud:spring-cloud-gcp-starters:1.2.8.RELEASE'
implementation 'org.springframework.cloud:spring-cloud-gcp-storage:1.2.8.RELEASE'

//gson
implementation 'com.google.code.gson:gson:2.10.1'
```

- Tool: IntelliJ Ultimate (2023.1.3)

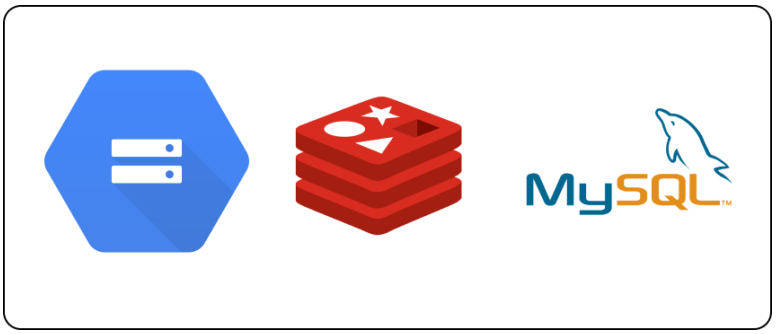
Infra Structure





- AWS EC2 Instance
- Jenkins
  - 2.422
- Webhook
- Docker
  - 24.0.6
- Nginx
  - 1.18
- SSL
  - letsencrypt

DataBase



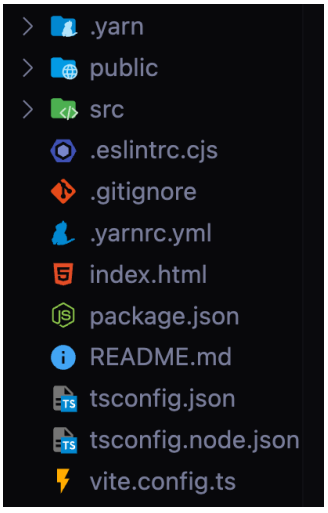
- Google Cloud Storage
  - 무료 300 크레딧 사용
- Redis 7.2.1
  - Docker로 띄워서 사용
- MySQL 8.0.33
  - ec2에 직접 설치

Local Application Setting

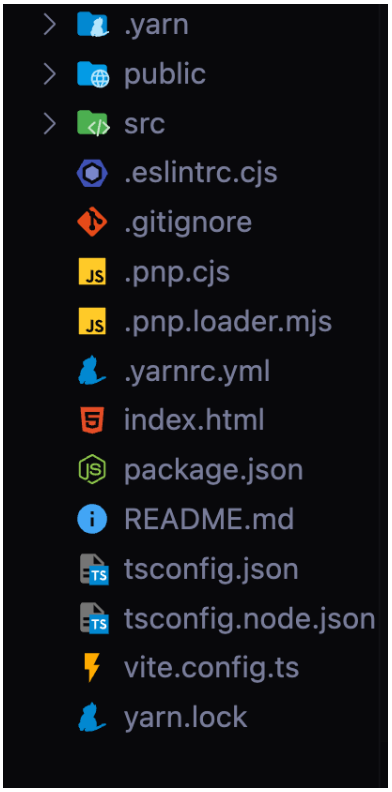
Front-End

1. 프로젝트 폴더 생성한 후 yarn 이용해서 vite 생성하기
   - `yarn create vite . --template react-ts`
2. Yarn Berry 설정 (해당 프로젝트 위치에서)
   - `yarn set version berry`

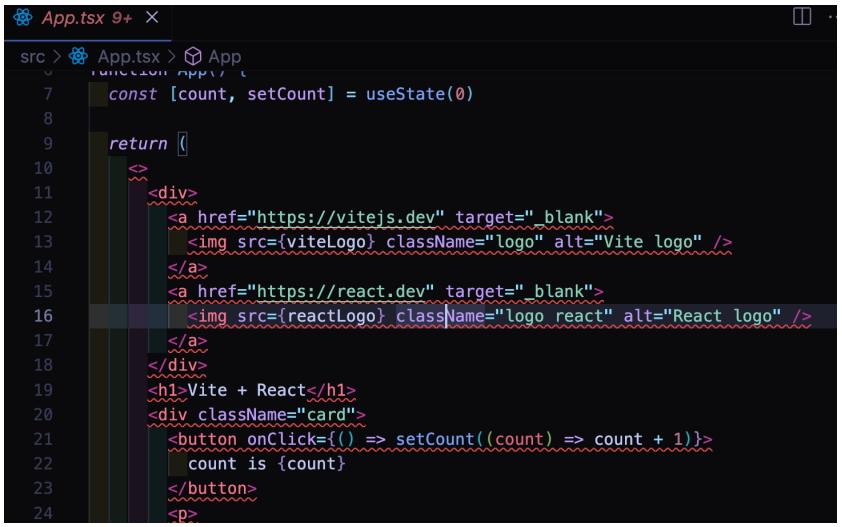
   여기까지 했을 경우 `.yarn` 폴더와, `.yarnrc.yml` 파일이 생성되어야 한다.



3. 의존성 설치하기
   - `yarn install`
   - 의존성 설치 이후에는 `.pnp.cjs`, `.pnp.loader.mjs` 파일과 `.yarn` 폴더가 생성되어야 한다.



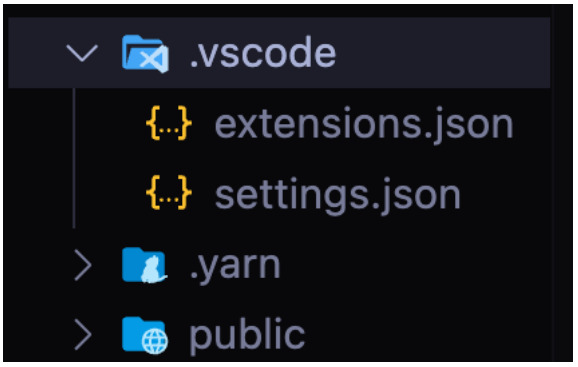
4. vscode에 yarn Berry 설정하기
    - 여기까지 진행했을 경우, 모듈을 찾을 수 없다는 에러가 발생함.



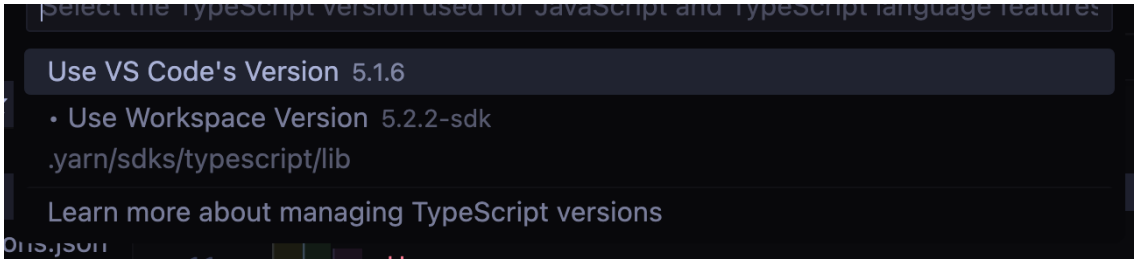
        - 이는 타입스크립트의 버전허용을 해주지 않았기 때문에 발생한 문제다.
            - 따라서 명시적으로 활성화해줄 필요가 있다.

1. vscode extension에서 “ZipFs - a zip file system” 설치하기
    - zip 파일로 설치된 종속성을 읽어올 수 있도록 설치한다.
2. `yarn dlx @yarnpkg/sdks vscode` 명령어를 실행한다.
    - yarn dlx는 npx와 같다고 보면 된다.

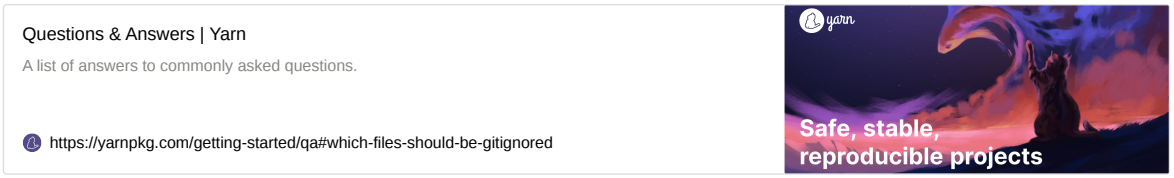
      성공적으로 설치가 된다면 root 디렉토리에 `.vscode` 폴더가 생성되고, 내부에 `extensions.json`과 `settings.json`이 생성된다.



3. 성공적으로 설치시, 아무런 `tsx` 파일에 들어가 `ctrl + shift + p` 를 누른 후 `Typescript : Select Typescript Version …`을 검색한다.
4. `Use Workspace Version`을 선택한다.



5. Zero-install을 위한 `.gitignore` 파일 수정하기

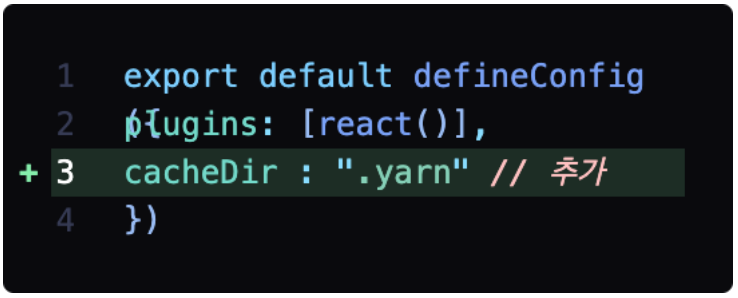


위 문서의 Zero-install 기준에 따라 .gitignore 파일을 수정한다.


```
!.vscode/settings.json

.yarn/*
!.yarn/cache
!.yarn/patches
!.yarn/plugins
!.yarn/releases
!.yarn/sdks
!.yarn/versions
```

6) [선택] 개발 서버 실행시 계속 `node_modules`가 발생한다면?

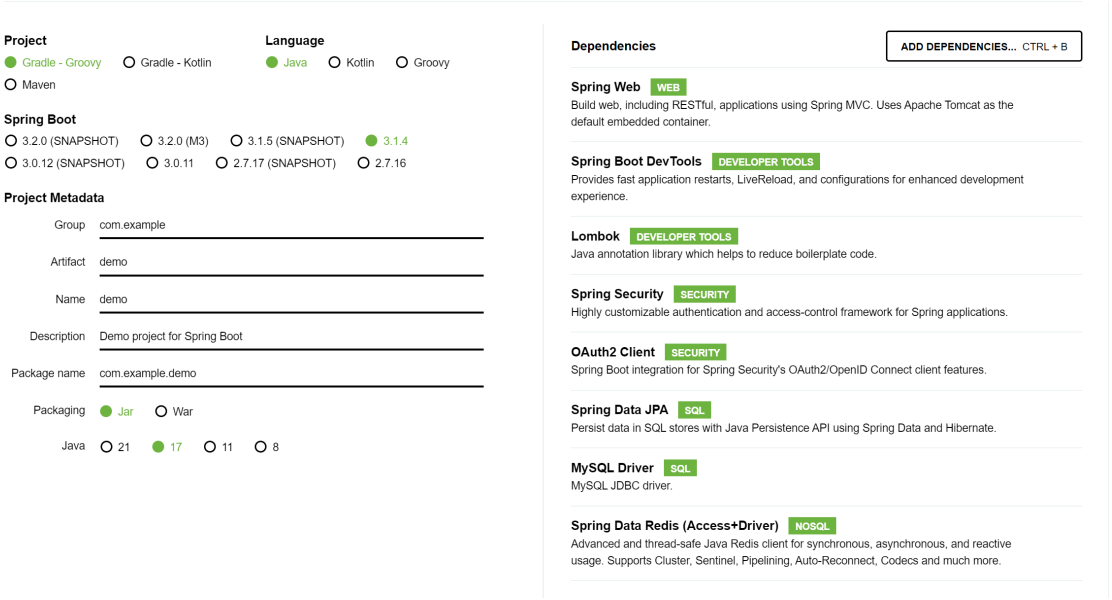
`vite.config.ts` 파일을 아래처럼 수정해준다.




```
cacheDir : ".yarn"
```

Back-End

1. Zulu 17 다운로드
   - Azul Zulu 17 msi 다운로드
   - msi는 환경변수 자동 세팅
2. IntelliJ Ultimate 다운로드
   - 학교 계정 연결하면 Ultimate 사용 가능
3. start.spring.io 에서 Spring 프로젝트 세팅
   - Gradle - Groovy
     - xml의 구조적인 틀을 벗어나 간결한 정의 가능
   - Spring Boot 3.1.3
     - 2.x 버전은 23년 11월 지원 종료로 인한 3.x 사용
   - Java 17
     - 3.x 부터는 JDK 17부터 지원
   - Jar
     - Spring Boot 안에 Tomcat을 내장하고 있어서, 코드만 패키징하는 Jar 형식 선택
   - Dependencies
     - Spring Web
     - Spring Boot DevTools
     - Lombok
     - Security
     - Oauth2
     - Spring Data JPA
     - MysqlDB Driver
     - Redis
   - Generate 클릭



4. 프로젝트를 열어 build.gradle이 의존 라이브러리를 가져옴
5. "File" -> "Settings"
   - "Plugins" 에서 Lombok Install 되어있는지 확인
   - "Editor" -> "File Encodings" -> Encoding 설정들 UTF-8로 변경 -> "Apply"
   - "Build, Execution, Deployment" -> "Build Tools" -> "Gradle" -> "Build and Run" 에서 Gradle로 되어있는 것 IntelliJ로 변경 -> "Apply"
   - "Build, Execution, Deployment" -> "Compiler" -> "Annotation Processors" -> "Enable annotation Processing" 체크 -> "Apply", "OK"
   1. build.gradle에 추가로 사용할 Dependency를 적용한다.

```
//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'

//GCS
implementation 'org.springframework.cloud:spring-cloud-gcp-starters:1.2.8.RELEASE'
implementation 'org.springframework.cloud:spring-cloud-gcp-storage:1.2.8.RELEASE'

//gson
implementation 'com.google.code.gson:gson:2.10.1'
```

   2. application.yml 파일을 세팅한다.

```
spring:
  servlet:
    multipart:
      max-file-size: 100MB
      max-request-size: 100MB

  # MySQL setting
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://[DB_HOST]:[DB_PORT]/[SCHEMA]?useUniCode=yes&characterEncoding=UTF-8
    username: [USERNAME]
    password: [PASSWORD]

  jpa:
    show-sql: true
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
      open-in-view: true
```

```
  jwt:
    prefix: 'Bearer'
    secret: [SECRET_KEY]
    token:
      access-expiration-time: [ACCESS_EXPIRE_TIME]
      refresh-expiration-time: [REFRESH_EXPIRE_TIME]
  data:
    redis:
      host: [DB_HOST]
      port: [DB_PORT]
      password: [PASSWORD]

  security:
    oauth2:
      client:
        registration:
          google:
            clientId: [GOOGLE_CLIENT_ID]
            clientSecret: [GOOGLE_CLIENT_SECRET]
            scope:
              - email
              - profile
          naver:
            client-id: [NAVER_CLIENT_ID]
            client-secret: [NAVER_CLIENT_SECRET]
            client-authentication-method: client_secret_post
            authorization-grant-type: authorization_code
            redirect-uri: '{baseUrl}/{action}/oauth2/code/{registrationId}'
            client-name: Naver
            scope:
              - nickname
              - email
          kakao:
            client-id: [KAKAO_CLIENT_ID]
            client-secret: [KAKAO_CLIENT_SECRET]
            redirect-uri: '{baseUrl}/{action}/oauth2/code/{registrationId}'
            authorization-grant-type: authorization_code
            client-authentication-method: client_secret_post
            client-name: Kakao
            scope:
              - profile_nickname
              - account_email
        provider:
          naver:
            authorizationUri: https://nid.naver.com/oauth2.0/authorize
            tokenUri: https://nid.naver.com/oauth2.0/token
            userInfoUri: https://openapi.naver.com/v1/nid/me
            userNameAttribute: response
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id

  cloud:
    gcp:
      storage:
        bucket-name: [BUCKET_NAME]
        project-id: [PROJECT_ID]
        credentials:
          location: [JSON_NAME]
  file:
    negative-crawling: [TEXT_NAME]
server:
  port: [SERVER_PORT]
  servlet:
    context-path: [PREFIX]

logging:
  level:
    com.com.vegetable: debug

url:
  gpu: [GPU_SERVER]
```

DataBase

MySQL

1. ubuntu 업데이트 및 설치

```
sudo apt-get update

sudo apt-get install mysql-server
```

2. 외부 포트 열기

```
sudo ufw allow mysql
```

3. mysql 실행 및 설정 변경

```
sudo systemctl start mysql

# ubuntu 서버가 재시작 되더라도 mysql이 자동 시작
$ sudo systemctl enable mysql
```

4. 외부 접속 허용

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
# #bind-address = 127.0.0.1 의 부분을
# bind-address = 0.0.0.0 으로 수정
```

5. mysql 재시작

```
sudo service mysql restart
```

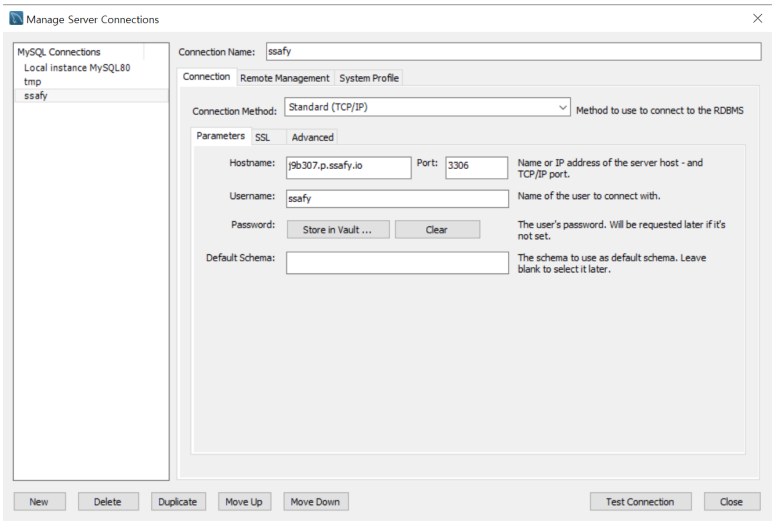
6. 서버 비밀번호 설정

```
sudo /usr/bin/mysql -u root -p
# 암호 입력
```

7. 사용자 생성

```
# mysql 접속 후
CREATE USER '사용자명'@'%' IDENTIFIED BY '비밀번호';
GRANT ALL PRIVILEGES ON * . * TO '사용자 명'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

8. mysql workbench 연결 확인



Redis

1. Redis docker 이미지 띄우기

```
docker run -p 6379:6379 --name redis -d redis:latest --requirepass [PASSWORD]
```

2. 비밀번호 사용하여 접속

```
docker exec -i -t redis redis-cli -a [PASSWORD]
```

EC2 Setting

Docker

1. ca-certificates, curl, gnupg 설치

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
```

2. 도커의 공식 GPG 키 추가

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

3. repository 설정

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

4. 설치

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

Potainer

> Docker를 웹에서 관리 도와주는 툴

1. 업데이트

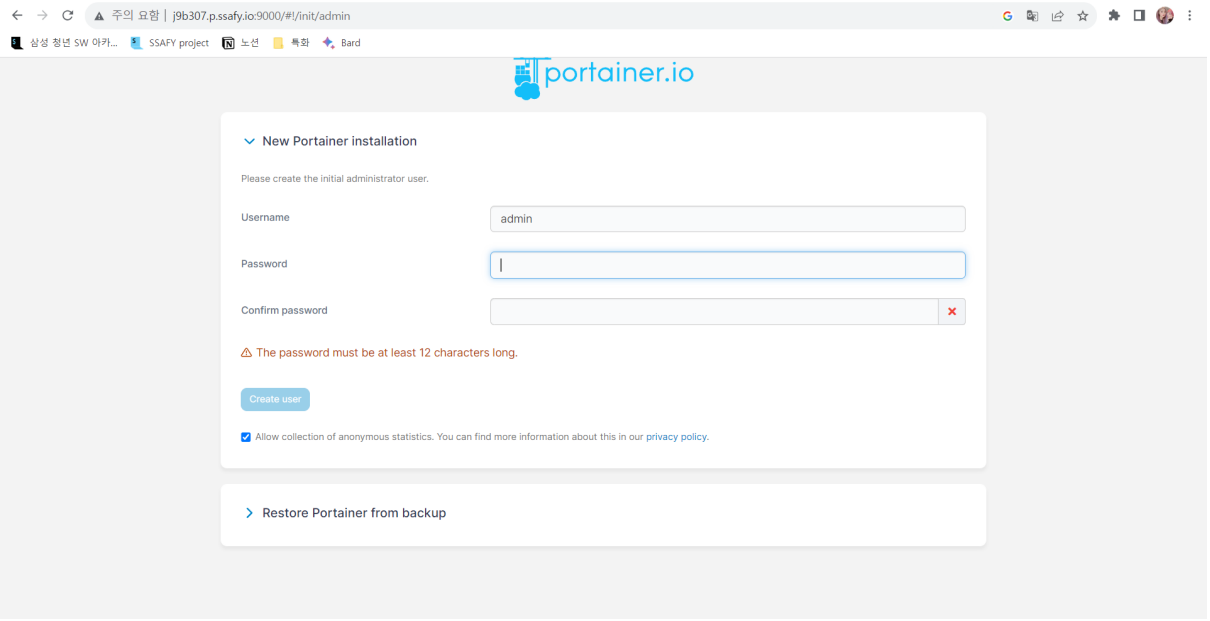
```
sudo apt update
```

2. portainer 설치

```
sudo docker run --name portainer -p 9000:9000 -d --restart always -v /data/portainer:/data -v /var/run/docker.sock:/var/run/docker.soc
```

3. 계정 생성

[서버 IP]:9000 접속



비밀번호 생성 ⇒ https://www.expressvpn.com/kr/password-generator

Jenkins

1. jdk17이 설치된 Jenkins 설치

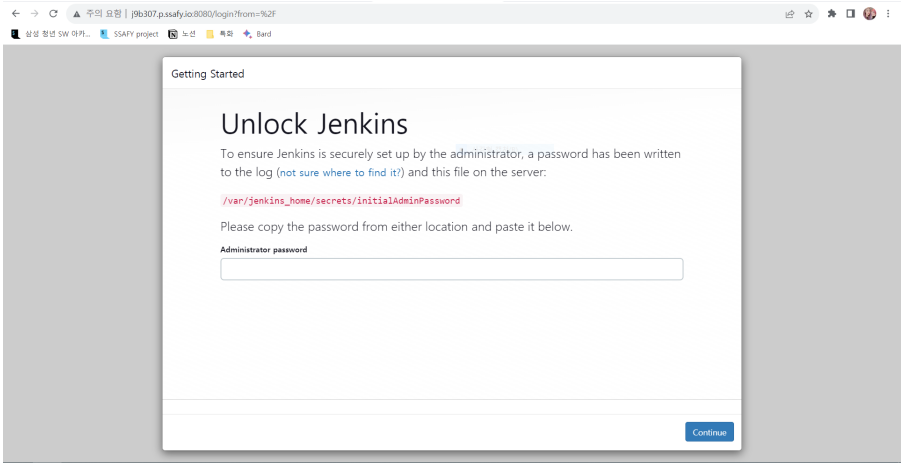
```
$ docker run -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock --name jenkins
  jenkins/jenkins:jdk17
```

2. 초기 비밀번호 확인 (2가지 방법)

```
# jenkins 컨테이너에 접속해 초기 비밀번호 확인
$ docker exec -it jenkins bash
$ cat /var/jenkins_home/secrets/initialAdminPassword

# log 확인
$ sudo docker logs jenkins
```

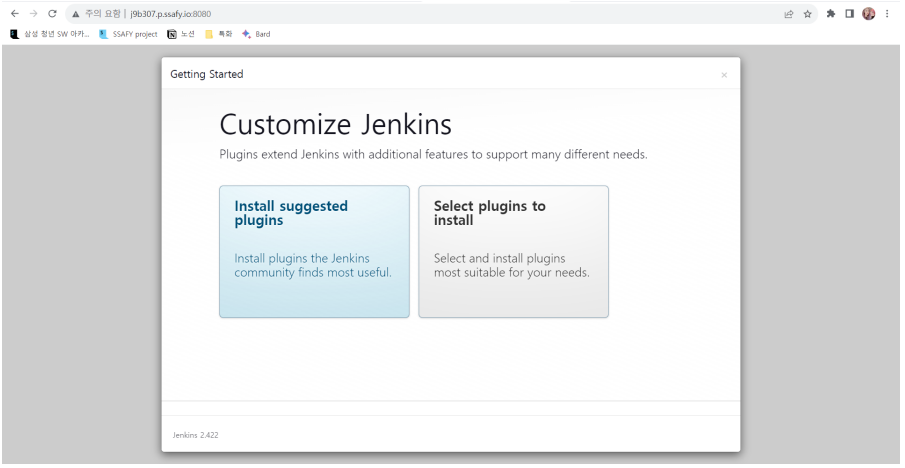
```
*************************************************************
*************************************************************
*************************************************************

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

01e51af85e99488aaba66657bafffb62

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

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

3. 세팅
    a. [서버 IP]:8080 접속



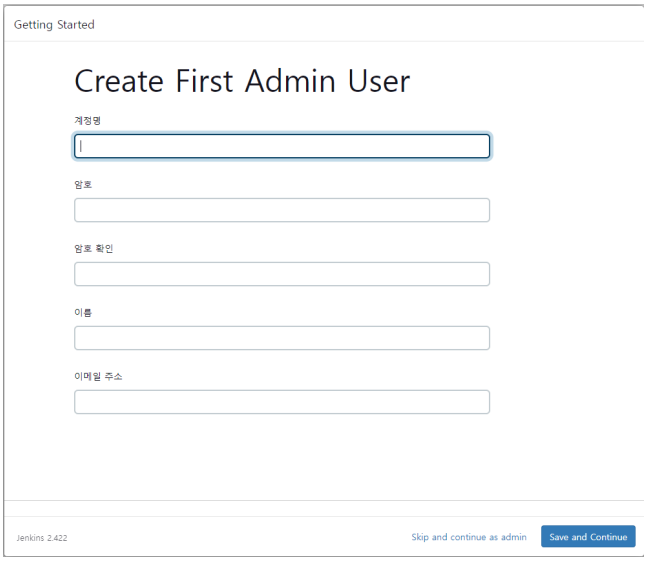
b. 초기 비밀번호 입력

c. Install suggested plugins 선택



d. 계정 생성

- 비밀번호 생성 ⇒ https://www.expressvpn.com/kr/password-generator



e. 시간 설정

- 사용자 - 설정 - User Defined Time Zone

- Asia/Seoul

f. docker 명령어 설치

- https://velog.io/@chang626/docker-container에서-docker-image-빌드-진행-과정-jenkins-host-docker.sock을-연결 참고

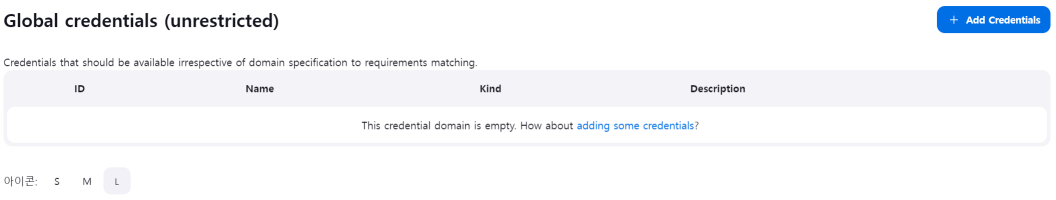
```
# root 권한으로 jenkins 접속
sudo docker exec -it -u root jenkins bash

# 공식 docker apt repository 구성 및 docker ce 바이너리 설치
apt-get update && \
apt-get -y install apt-transport-https \
     ca-certificates \
     curl \
     gnupg2 \
     software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") \
   $(lsb_release -cs) \
   stable" && \
apt-get update && \
apt-get -y install docker-ce

# docker jenkins에서 host docker 접근권한을 부여
service docker start
groupadd -f docker
usermod -aG docker jenkins
chown root:docker /var/run/docker.sock
```

g. Plugins

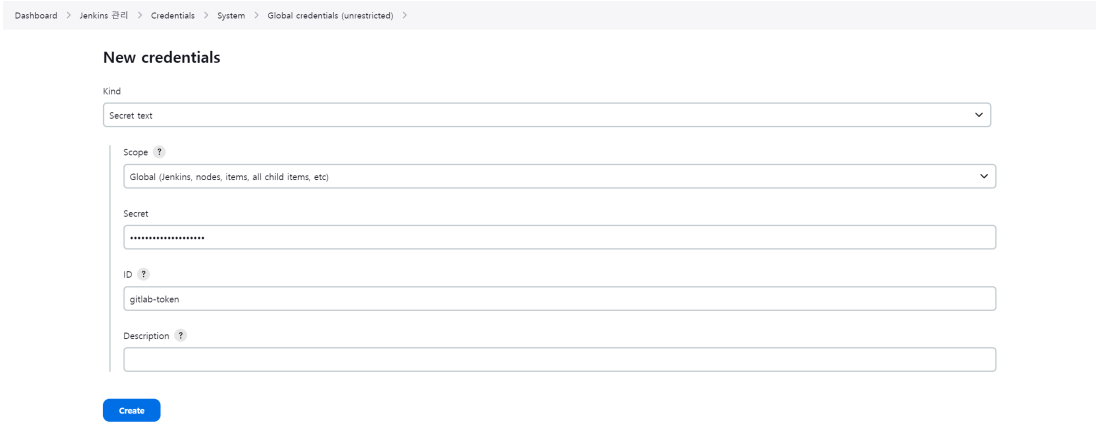
- Gitlab
- Publish Over SSH
- NodeJS

h. Credentials

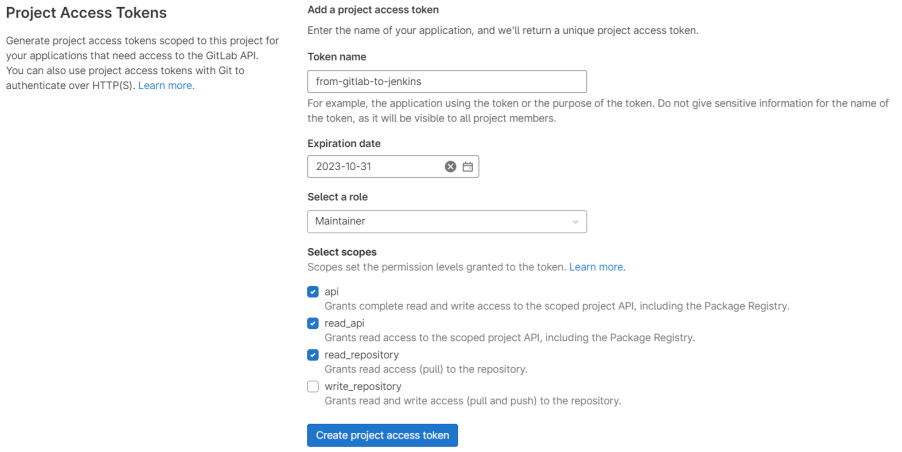
- Jenkins 관리 - Credentials - Stores scoped to Jenkins(global)



- gitlab 액세스 토큰



- kind
  - Secret-text
- secret
  - gitlab에서 생성한 토큰
    - 프로젝트 - Settings - Access Tokens



- ID
  - gitlab-token

i. System

- Dashboard - Jenkins 관리- System
  - GitLab
    - Connection name
      - gitlab-connection
    - GitLab host URL
      - https://lab.ssafy.com
    - Credentials
      - Credentials에서 생성한 `gitlab-token`

j. Tools

- JDK installations

```
# jenkins 컨테이너 접속
$ sudo docker exec -it jenkins bash

# 환경 변수 확인
$ env
```

  - Name
    - Java17
  - JAVA_HOME
    - /opt/java/openjdk
- Gradle

> 프로젝트에서 쓰는 gradle버전과 같게 설정

  - name

        - Gradle8.2.1
    - version (Install automatically)
        - Gradle 8.2.1 선택
- NodeJS installations

> 프로젝트에서 쓰는 nodejs 버전과 같게 설정

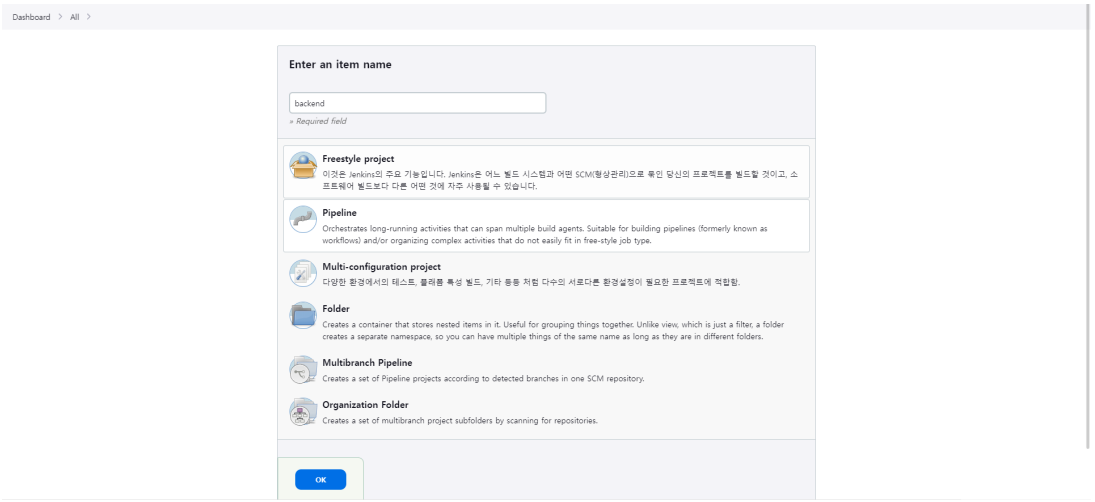
    - Global npm packages to install
        - yarn

k. Item

frontend

backend

- 아이템 추가



- Build Triggers - Build when a change is pushed to GitLab. GitLab webhook URL: http://j9b307.p.ssafy.io:8080/project/backend 체크
    - 고급의 secret token 생성
- gitlab 프로젝트 - Settings - Webhooks에 입력

Nginx Setting

> Nginx 설치 및 SSL 적용

1. nginx 설치

```
# 설치
sudo apt-get install nginx

# 설치 확인 및 버전 확인
nginx -v
```

2. letsencrypt 설치를 위해 다음과 같은 순서로 명령어를 입력

```
sudo apt-get install letsencrypt

sudo systemctl stop nginx
```

```
sudo letsencrypt certonly --standalone -d j9b307.p.ssafy.io
```

3. conf 작성

```
cd /etc/nginx/sites-available

sudo vi proxy-setting.conf
```

- proxy-setting.conf

```
client_max_body_size 1G;

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

    server_name j9b307.p.ssafy.io;

    # HTTP에서 HTTPS로 리다이렉션 설정
    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name j9b307.p.ssafy.io;

    location / {
        proxy_pass http://localhost:3000;
        proxy_redirect http://localhost:3000/ http://j9b307.p.ssafy.io/;
        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 /api {
        proxy_pass http://localhost:5000/api;
        proxy_redirect http://localhost:5000/ http://j9b307.p.ssafy.io/;
        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;
    }

    ssl_certificate /etc/letsencrypt/live/j9b307.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j9b307.p.ssafy.io/privkey.pem; # managed by Certbot
    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}
```

4. 기존 nginx port 변경

```
cd /etc/nginx/sites-enabled

vi default

# 아래와 같이 변경
```

```
server {
        listen 180 default_server;
        listen [::]:180 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server
```

```
sudo ln -s /etc/nginx/sites-available/proxy-setting.conf /etc/nginx/sites-enabled/proxy-setting
```

- nginx 테스트

```
sudo nginx -t
```

```
ubuntu@ip-172-26-4-119:/etc/nginx/sites-enabled$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

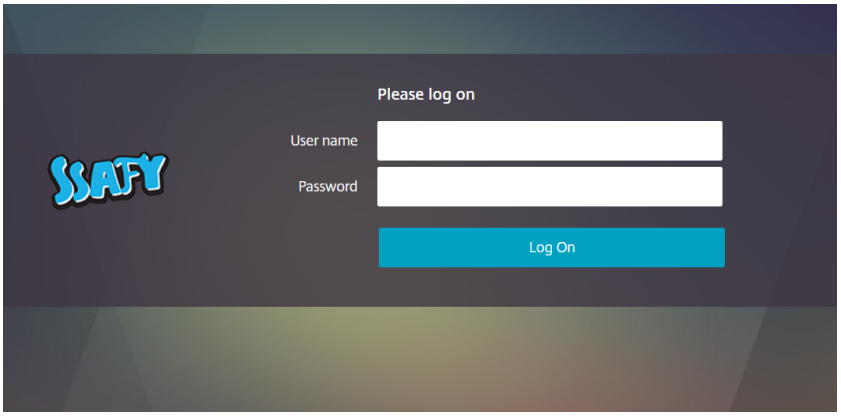
- 재시작

```
sudo systemctl restart nginx
```

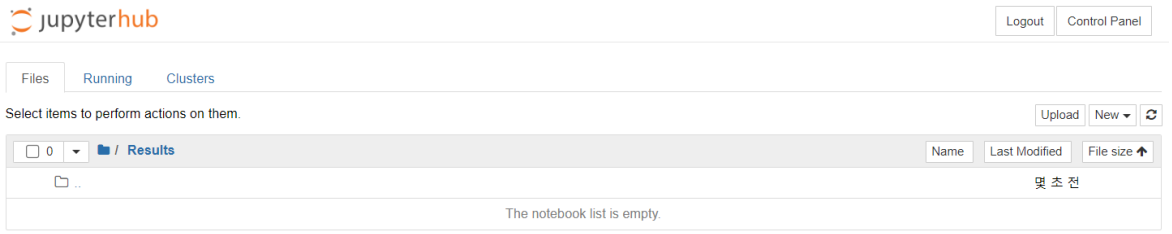
GPU Server Setting

> SSAFY 제공 GPU 서버 세팅

1. https://server.ssafy.com 접속 후, 제공받은 아이디와 비밀번호로 로그인



2. Citrix gateway 설치, 로그인 후 제공받은 서버 주소로 접속



3. SSAFY에서 제공하는 GPU 서버는 외부에서 직접 ssh 접근이 불가능하므로 통신을 위해 EC2 서버 주소와 GPU 서버의 원하는 포트 번호 개방 요청

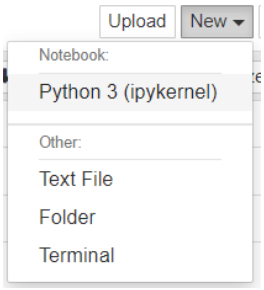
Flask 세팅

1. Jupyter lab에서 터미널 실행 후, conda 가상환경 내에 Flask 설치

```
pip install flask
```

2. 받은 서버 내에 flask 실행을 위한 파이썬 파일 생성

```
touch 파일명.py
```



Jupyter lab의 New 클릭 후, python3 선택해서 파이썬 파일 생성
or
Jupyter lab의 New 클릭 후, Text File 선택해서 파이썬 파일 생성

3. 파일 내에서 Flask 설정을 위해 Flask 클래스 호출 후, flask 객체 생성

```
from flask import Flask

app = Flask(__name__)
```

4. 기본 서버 뒤에 붙는 주소 입력 후, 해당 주소 호출 시 보여줄 내용을 def 문 안에 작성

```
@app.route("/")
def index():
  return "Hello, World!"
```

5. 해당 파이썬 파일이 실행되면 flask를 실행하라는 문구 추가
   - `app.run()` 내부 입력 요소
     - `port=”number”` : 포트 번호를 number로 변경
     - `debug=True` : 파일을 고칠 때마다 자동으로 실행
     - `host="0.0.0.0"` : 여러 IP 주소를 통해 접근이 가능

```
if __name__ == "__main__":
  app.run()
```

6. Jupyter lab terminal에서 백그라운드에서 flask가 실행되도록 커맨드 입력

```
nohup python -u 파일명.py &
```

- Jupyter lab terminal에서 통신 로그 확인하기

```
tail -f nohup.out
```

7. 백그라운드에서 실행되는 flask 종료하기

```
kill -9 PID번호
```