



Trend24 포팅매뉴얼

[서버 정보](#)

[서버 접속](#)

[도메인 설정](#)

[Nginx](#)

[Nginx 설치](#)

[conf 파일 작성](#)

[SSL 설정](#)

[Docker](#)

[설치](#)

[현재 유저에게 도커 쓸 수 있는 권한 주기](#)

[DB 설정](#)

[실행 권한 부여](#)

[MySQL & Redis](#)

[Qdrant](#)

[FastAPI](#)

[Jenkins](#)

[swap memory](#)

[설치](#)

[Jenkins 내부에 Docker 패키지 설치](#)

[인증 키 확인](#)

[플러그인 설치](#)

[gitlab credential 등록](#)

[백엔드 파이프라인 생성](#)

[생성 및 gitlab webhook 등록](#)

[credential \(프로그램에서 쓰는 secret key\) 추가](#)

[파이프라인 내용 작성](#)

[프론트엔드 파이프라인](#)

[데이터](#)

[시크릿 키](#)

[trend-db](#)

[trend-fastapi](#)

[trend-qdrant](#)

서버 정보

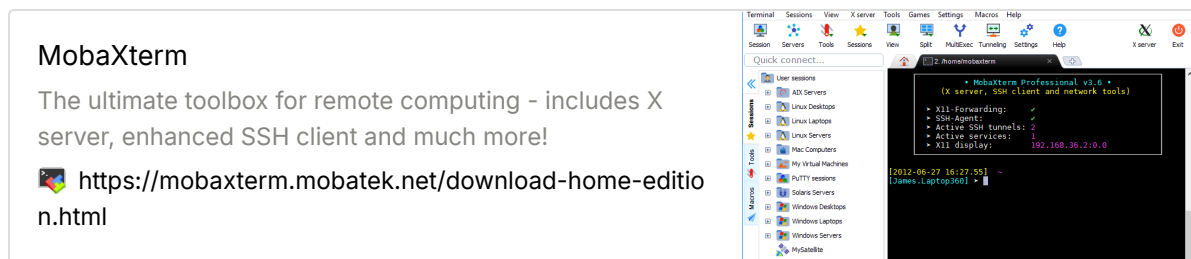
- Ubuntu 20.04.6 LTS

서버 접속

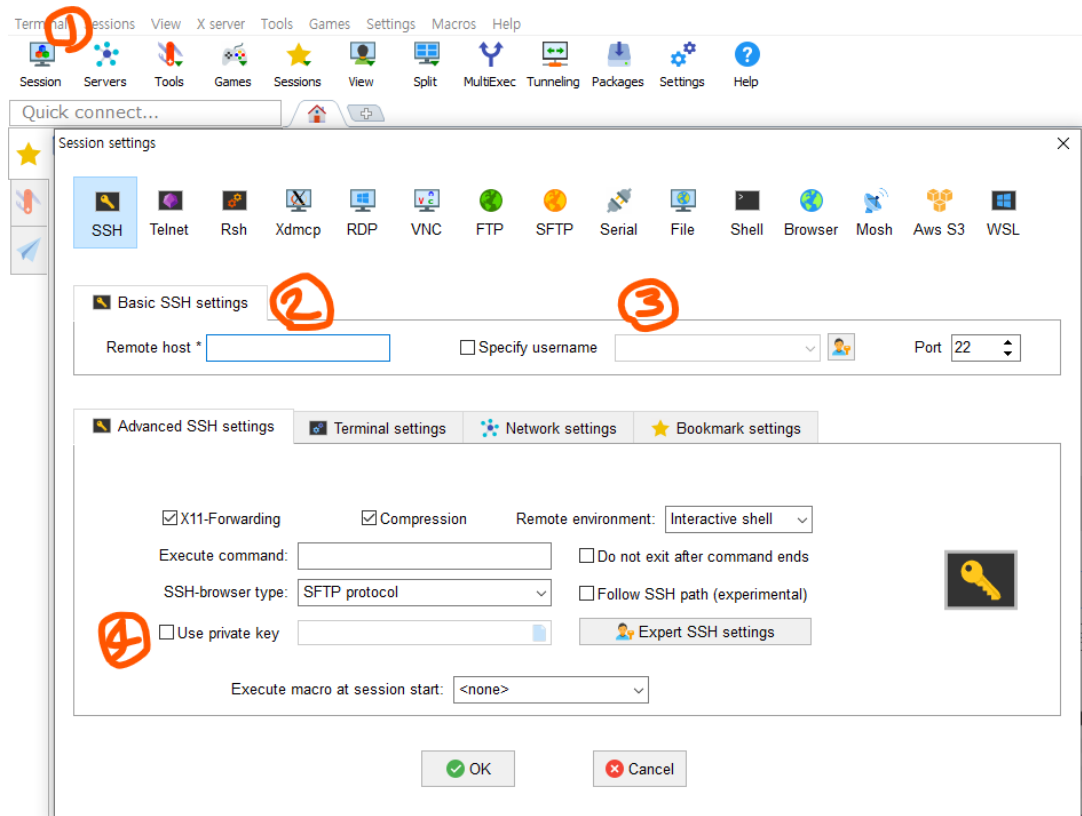
- 접속시 .pem 파일이 필요합니다.
- git bash 이용한다면 .pem이 있는 폴더에서 아래 명령어 입력

```
ssh -i <.pem 파일 이름> <유저이름>@<서버주소>
```

- MobaXterm을 이용한다면 다운로드를 해야합니다.



- mobaXterm 사용할 때의 접속 방법
 1. 세션 버튼 누르기
 2. 리모트 호스트에 <서버주소> 입력
 3. specify username 체크하고 <유저이름> 입력
 4. use private key 체크하고 <.pem 파일 이름> 파일 선택



- 루트 비밀번호 변경

```
sudo passwd root
```

도메인 설정

- 가비아 이용
- 서버 ip 주소 확인

```
curl ifconfig.me
```

- 가비아 > 서비스 관리 > DNS 관리 툴

이용 중인 서비스 전체 1건

소유권 이전

담당자 설정

DNS
관리툴

도메인
통합 관리툴

• 도메인 선택 후 DNS 설정

gabia. DNS 관리

전체 도메인 1개 (가비아 등록 도메인 + 타기관 등록 도메인)

한국어 English

홈 > DNS 설정

> 가비아 등록 도메인

DNS 관리

DNS 권한 설정

> 타기관 등록 도메인

DNS 설정

DNS 설정

도메인 연결

포워딩

웹 파킹

모바일 파킹

Q

10개

<input type="checkbox"/>	도메인명 ^	DNS 정보
<input checked="" type="checkbox"/>	██████████	CNAME / A

• 다음과 같이 A 레코드 추가

- 가린 항목은 <서버 IP 주소> 입력

A	@	██████████	3600	DNS 설정	수정	삭제
A	www	██████████	3600	DNS 설정	수정	삭제

Nginx

Nginx 설치

- 다음 명령어 입력

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
```

conf 파일 작성

- vi editor 명령어 참고

```
gg // 맨 위 첫줄로 이동
shift v g // 전체 선택
y // 복사 -> 전체 선택한 상태에서 하면 전체 복사 됨
d // 삭제
dd // 한 줄만 삭제
x // 한 글자 삭제
p // 붙여넣기

a // append 내용 편집
esc // 내용 편집 종료

wq enter // 수정한 내용을 저장하고 종료
q! // 수정한 내용을 저장하지 않고 종료
```

- /etc/nginx/conf.d/default.conf 파일 생성

```
sudo vi /etc/nginx/conf.d/default.conf
```

- 내용 입력

```
upstream frontend {
    server 0.0.0.0:3000;
}

upstream backend {
    server 0.0.0.0:8080;
```

```

}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name <서버도메인> www.<서버도메인>;

    ssl_certificate /etc/letsencrypt/live/<서버도메인>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<서버도메인>/privkey.pem;

    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location /api {
        #rewrite ^/api(/.*)$ $1 break;
        proxy_pass http://backend;
        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 / {
        proxy_pass http://frontend;
        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;
    }
}

server {

```

```
listen 80;
listen [::]:80;
server_name <서버ip>;

location / {
    return 301 $scheme://<서버도메인>$request_uri;
}
}
```


- 설명
 - upstream frontend
 - 서버의 3000 포트에서 실행중인 프로그램과 연결
 - upstream backend
 - 서버의 8080 포트에서 실행중인 프로그램과 연결
 - location /api
 - /api 이하의 uri는 backend로 연결

SSL 설정

- 다음 문서 참고하여 설정

Nginx를 이용하여 https 적용하는 법

Nginx를 이용하여 https 적용하는 법 · GitHub

 <https://gist.github.com/woorim960/dda0bc85599f61a025bb8ac471dfaf7a>

 **GitHub** Gist

Docker


- Docker version 26.1.1, build 4cf5afa

설치

- 공식 문서

Install Docker Engine on Ubuntu

Jumpstart your client-side server applications with Docker Engine on Ubuntu. This guide details prerequisites and multiple methods to install Docker Engine on Ubuntu.

 <https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository>



- 시스템 먼저 업데이트

```
sudo apt-get update
sudo apt-get upgrade
```

- 레포지터리 추가

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

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] 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
```


- 패키지 설치

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

- 잘 됐는지 확인하려면 다음 명령어 입력

```
sudo docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
c1ec31eb5944: Pull complete  
Digest: sha256:a26bff933ddc26d5cdf7faa98b4ae1e3ec20c4985e6f87ac0973052224d24302  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/
```

현재 유저에게 도커 쓸 수 있는 권한 주기

- 현재 사용자를 도커 그룹에 추가

```
sudo usermod -aG docker $USER
```

- 이제 sudo 없이 docker 명령어 사용 가능

DB 설정

실행 권한 부여

- 프로젝트 git clone 하기
- 프로젝트 디렉터리로 이동
- docker 디렉터리로 이동
- .sh 파일 실행 권한 부여

```
chmod +x grant-permission.sh  
bash ./grant-permission.sh
```

MySQL & Redis

- docker/trend-db 디렉터리로 이동
- 문서 하단 시크릿키 > trend-db 보고 .env 파일 만들기
- 도커 실행

```
bash ./docker-compose.sh
```

Qdrant

- docker/trend-qdrant 디렉터리로 이동
- 문서 하단 시크릿키 > trend-qdrant 보고 .env, books.json, embeddings_topic.npy 파일 만들기

- qdrant 실행

```
bash ./run-docker.sh
```

- 데이터 넣기

```
bash ./insert_qdrant_data.sh
```

- 만약 데이터를 전부 삭제하고 싶다면 다음 명령어를 실행한다. 이후 qdrant 실행과 데이터 넣기를 다시 실행한다.

```
bash ./delete-qdrant-volume.sh
```

FastAPI

- 반드시 상단의 db설정을 끝내고 해야 합니다.
- docker/trend-db 디렉터리로 이동
- 문서 하단 시크릿키 > trend-fastapi 보고 .env 파일 만들기
- 실행환경 이미지 빌드. 라이브러리 설치 용도입니다.

```
bash ./build-requirements-image.sh
```

- fastAPI 실행

```
bash ./run-fastapi.sh
```

Jenkins

swap memory

- 젠킨스는 메모리 용량을 많이 차지할 수도 있으므로 swap memory 설정

- 현재 사용 중인 메모리 용량 확인

```
free -h
```

- 이런 식으로 나오는데 우리 메모리는 15기가이다.

	total	used	free	shared	buff
Mem:	15Gi	5.0Gi	183Mi	2.0Mi	
Swap:	0B	0B	0B		

- 메모리 스왑 설정 (서버 메모리 꽉 차면 가상 메모리 쓸 수 있음)
 - swap 영역은 서버 메모리의 2배 정도로 했다
 - 영역 할당, 권한 수정, 생성, 활성화

```
sudo fallocate -l 30G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

- 결과

	total	used	free	shared	buff
Mem:	15Gi	5.1Gi	147Mi	2.0Mi	
Swap:	29Gi	0B	29Gi		

설치

- 도커로 설치

```
docker pull jenkins/jenkins:jdk17 | docker run -d --restart  
always --env JENKINS_OPTS="--httpPort=<포트번호> -v /etc/lo  
caltime:/etc/localtime:ro -e TZ=Asia/Seoul -p <포트번호>:<포  
트번호>-v /jenkins:/var/jenkins_home -v /var/run/docker.soc  
k:/var/run/docker.sock -v /usr/local/bin/docker-compose:/u  
sr/local/bin/docker-compose --name jenkins -u root jenkins/  
jenkins:jdk17
```

Jenkins 내부에 Docker 패키지 설치

- 젠킨스 내부 shell 접속

```
docker exec -it jenkins bash
```

- 내부 shell에서 다음 명령어 입력

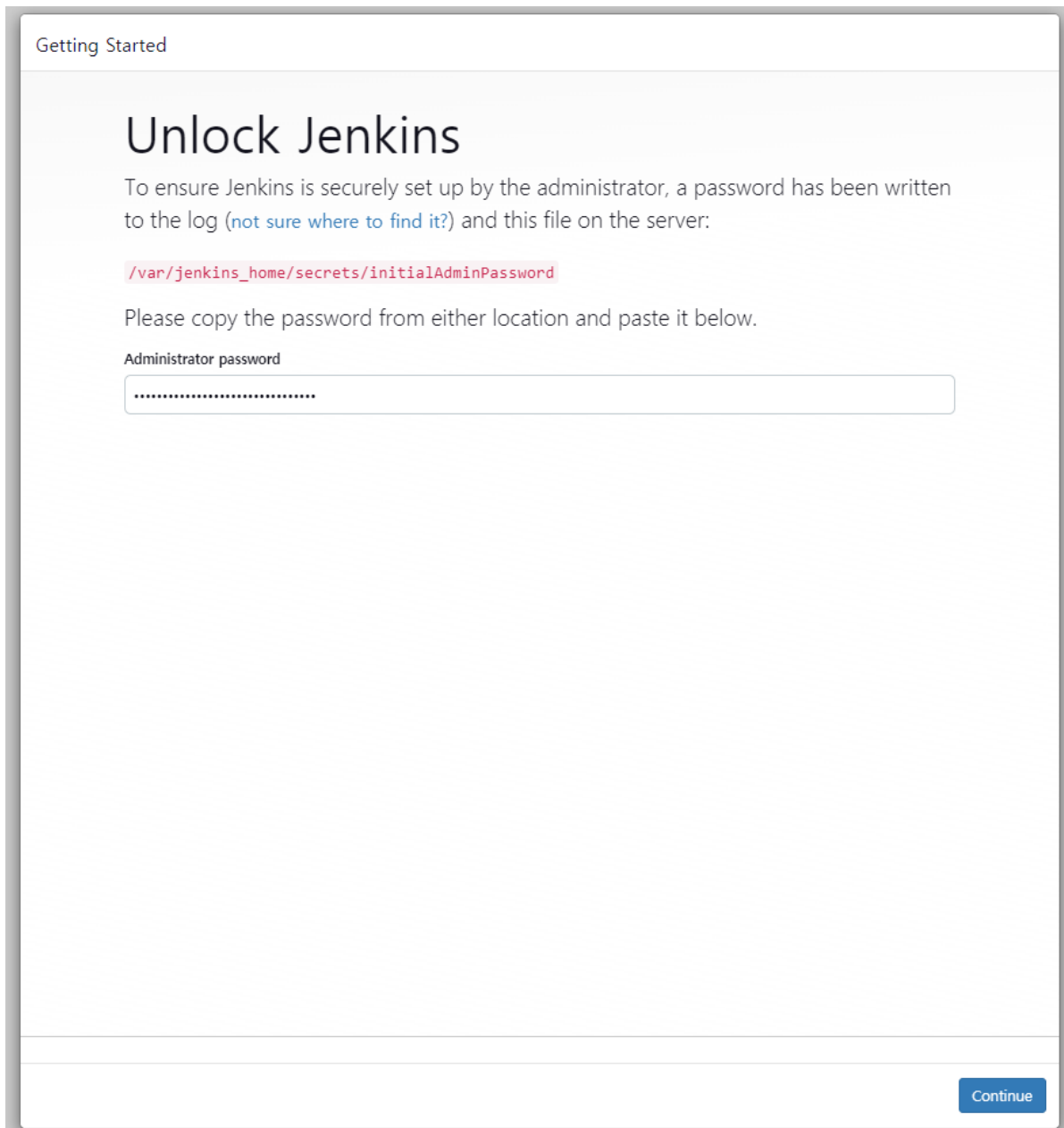
```
apt-get update && apt-get -y install apt-transport-https c  
a-certificates curl gnupg2 software-properties-common && c  
url -fsSL https://download.docker.com/linux/$(. /etc/os-re  
lease; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey  
&& add-apt-repository "deb [arch=amd64] https://download.d  
ocker.com/linux/$(. /etc/os-release; echo "$ID") $(lsb_rel  
ease -cs) stable" && apt-get update && apt-get -y install  
docker-ce
```

인증 키 확인

- 인증 키 확인

```
docker exec jenkins cat /var/jenkins_home/secrets/initialAdm:
```

- 젠킨스 웹 페이지에 인증 키 입력

The image shows the 'Unlock Jenkins' screen from the Jenkins web interface. At the top, it says 'Getting Started'. The main heading is 'Unlock Jenkins'. Below this, a paragraph explains that a password has been written to the log (with a link 'not sure where to find it?') and to a file on the server: `/var/jenkins_home/secrets/initialAdminPassword`. It then asks the user to copy the password from either location and paste it below. There is a text input field labeled 'Administrator password' with a masked password '.....'. At the bottom right, there is a blue 'Continue' button.

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:

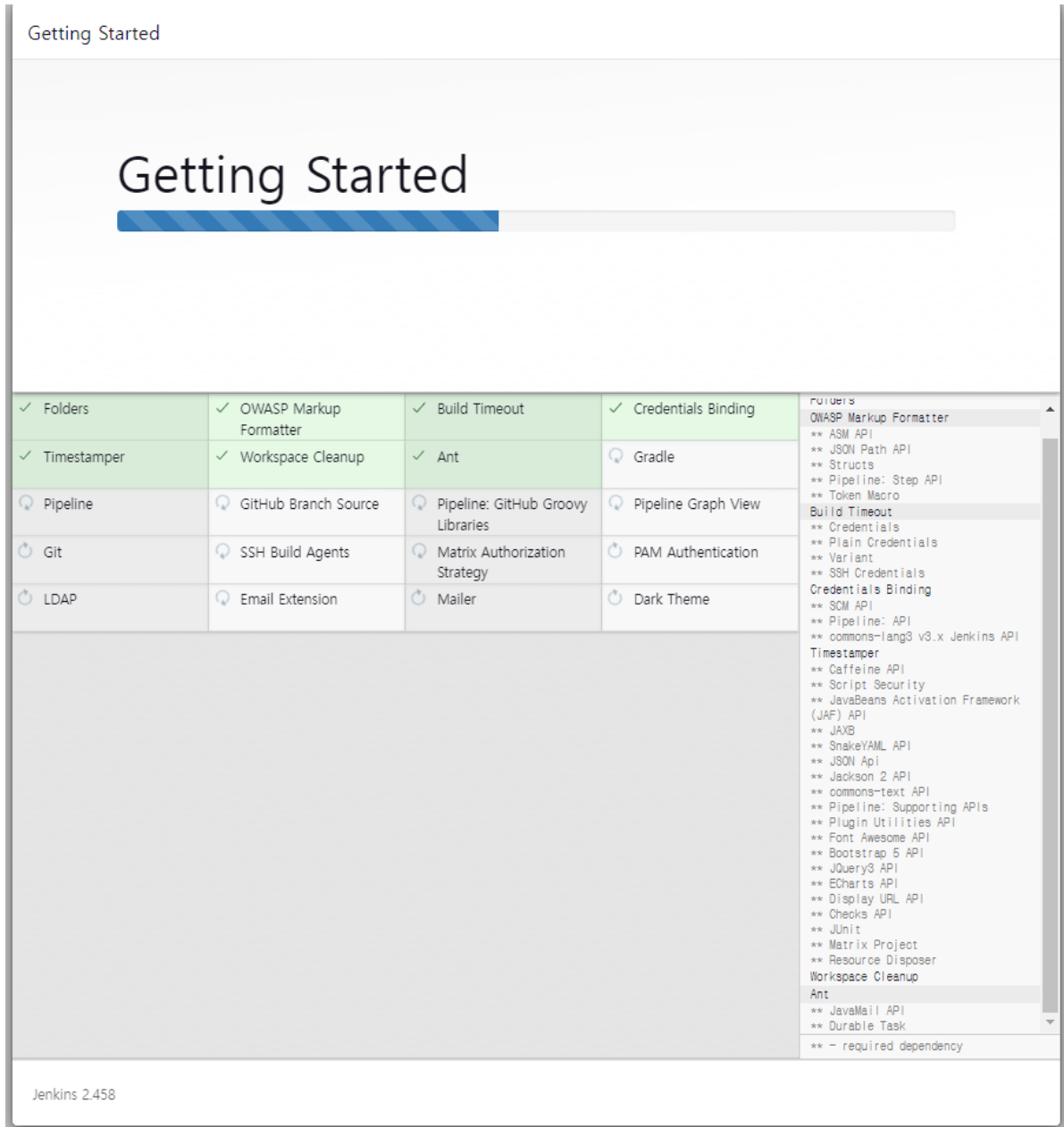
```
/var/jenkins_home/secrets/initialAdminPassword
```

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

Administrator password

Continue

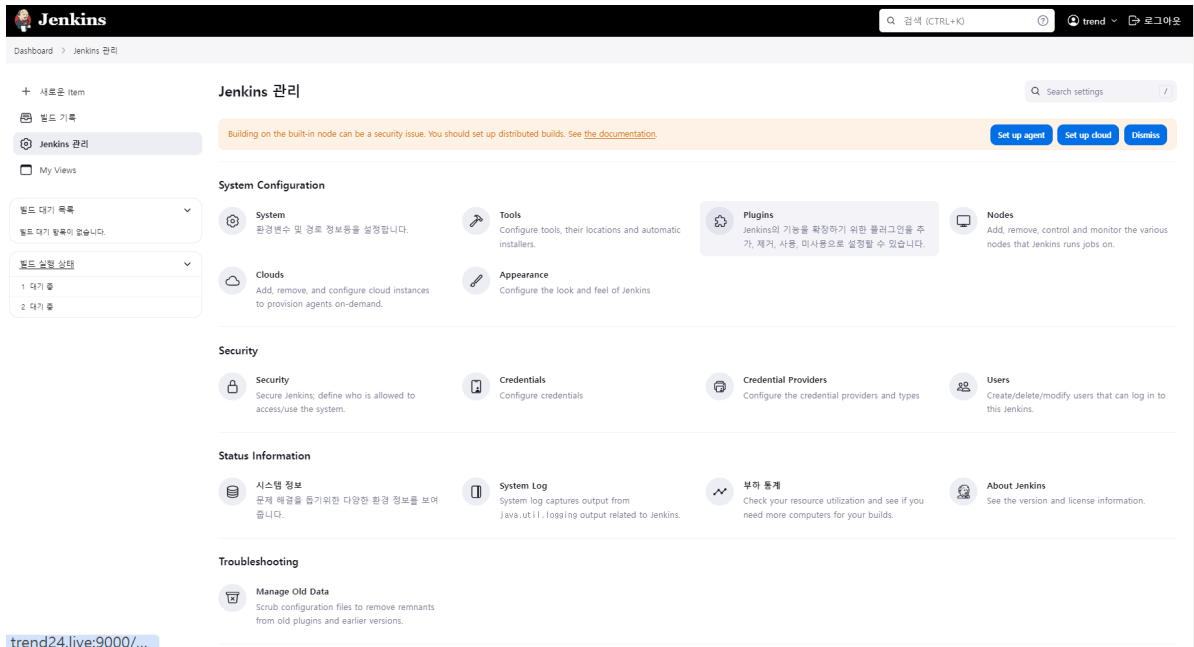
- 왼쪽 install suggested plugins 설치
그러면 이렇게 설치 진행함



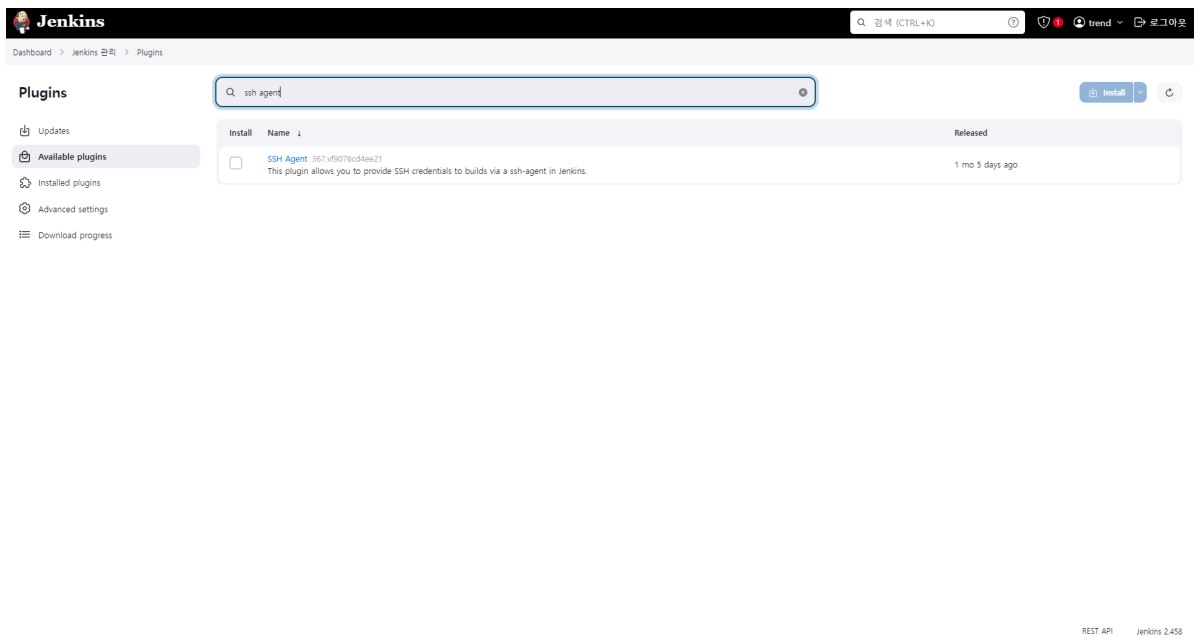
- 이후 어드민 계정 생성

플러그인 설치

- 플러그인 선택



- 검색창에서 검색가능

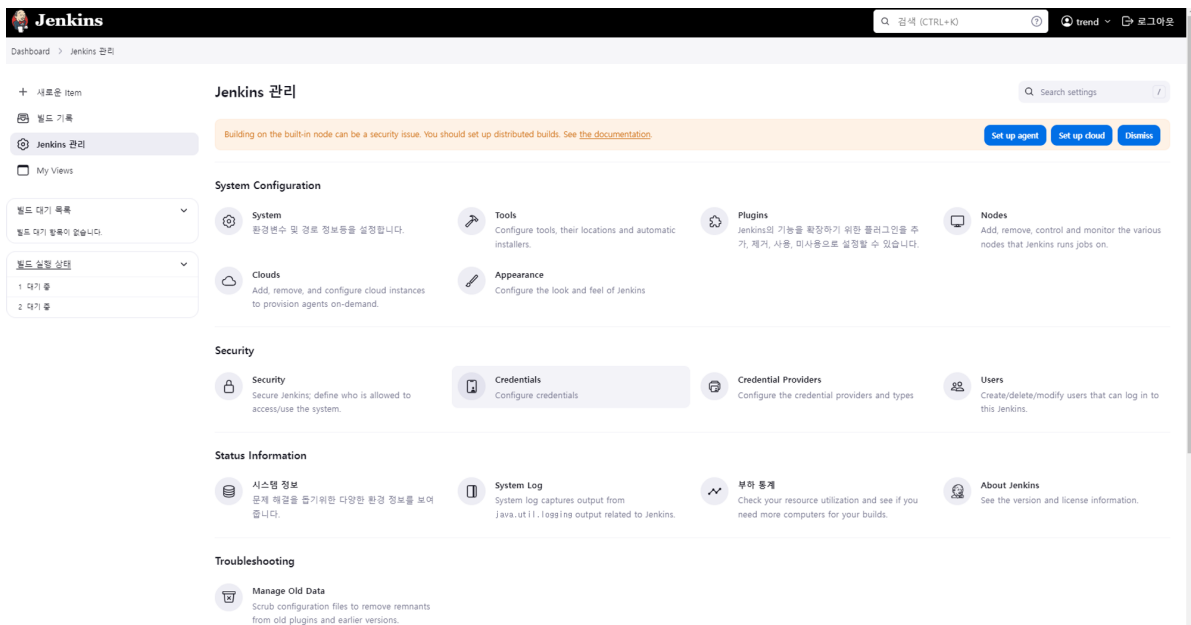


- 다음 플러그인 검색해서 설치

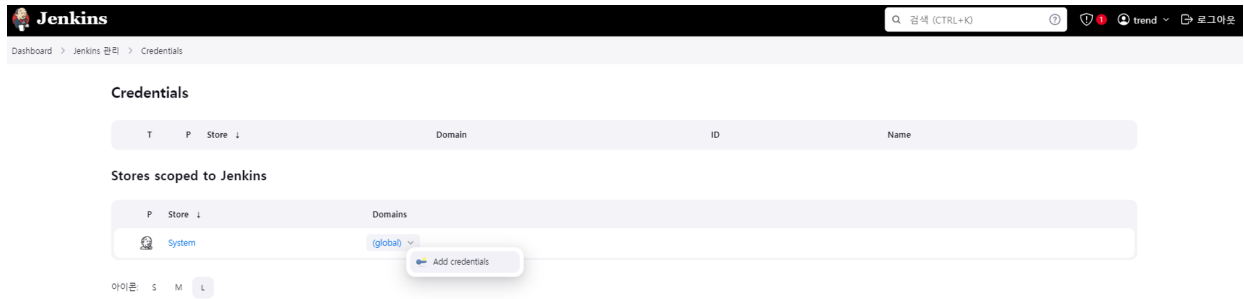
SSH Agent
Docker
Docker Commons
Docker Pipeline
Docker API
Generic Webhook Trigger
GitLab
GitLab API
GitLab Authentication

gitlab credential 등록

- 첫 페이지에서 credentials 선택



- add credential 선택



- 내용 입력

New credentials

Kind: Username with password

Scope: Global (jenkins, nodes, items, all child items, etc)

Username: [Redacted]

☐ Treat username as secret

Password: [Redacted]

ID: [Redacted]

An internal unique ID by which these credentials are identified from jobs and other configuration. Normally left blank, in which case an ID will be generated, which is fine for jobs created using visual forms. Useful to specify explicitly when using credentials from scripted configuration. (from Credentials Plugin)

Description: [Redacted]

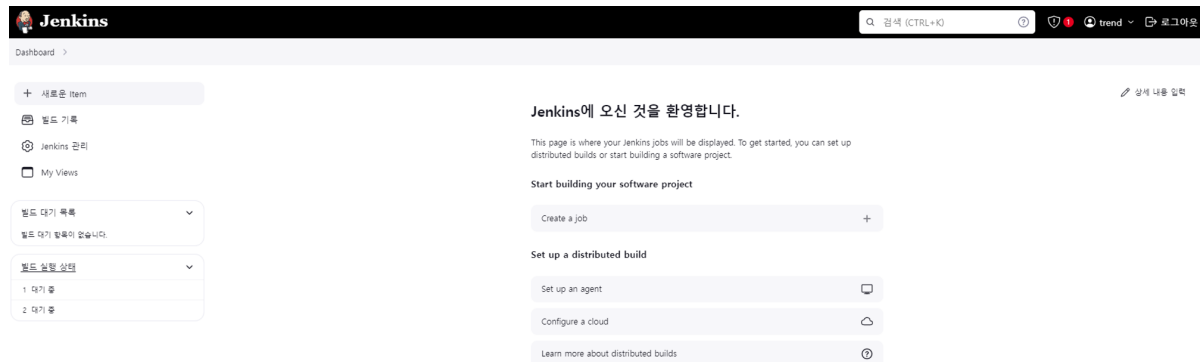
Create

- Username : Gitlab 아이디
- Password : Gitlab 패스워드
- ID : 스크립트에 사용할 변수명

백엔드 파이프라인 생성

생성 및 gitlab webhook 등록

- 왼쪽 새로운 item



- 이름 짓고 파이프라인 선택

New Item

Enter an item name

springboot

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

다양한 환경에서의 테스트, 플러그인 특성 빌드, 기타 응용 지점 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

OK

• 빌드 트리거 설정

Configure

General

Advanced Project Options

Pipeline

☐ 오래된 빌드 삭제 ?

☐ 이 빌드는 매개변수가 있습니다 ?

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://trend24.live:9000/project/springboot ?

Enabled GitLab triggers

☒ Push Events ?

☐ Push Events in case of branch delete ?

☒ Opened Merge Request Events ?

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events ?

☐ Closed Merge Request Events ?

Rebuild open Merge Requests ?

Never

☒ Approved Merge Requests (EE-only) ?

☒ Comments ?

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급

• 빌드 트리거 > 고급 > secret token 우하단 generate

Configure

General

Advanced Project Options

Pipeline

☐ 오래된 빌드 삭제 ?

☐ 이 빌드는 재개변수가 있습니다 ?

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://trend24.live:9000/project/springboot> ?

Enabled GitLab triggers

☒ Push Events ?

☐ Push Events in case of branch delete ?

☒ Opened Merge Request Events ?

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events ?

☐ Closed Merge Request Events ?

Rebuild open Merge Requests ?

Never

☒ Approved Merge Requests (EE-only) ?

☒ Comments ?

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급 >

◦ 생성된 토큰 기억해두기

- gitlab > settings > webhooks> new webhooks

Search page

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

Project Hooks 0

Add new webhook

No webhooks enabled. Select trigger events above.

- 웹훅 등록

- url: <젠킨스주소>/project/springboot
- secret token: 생성된 토큰

Q Search page

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

Project Hooks 0

URL
[redacted]project/springboot

Secret token will be cleared on save unless token is updated.

URL must be percent-encoded if it contains one or more special characters.

☒ Show full URL
☐ Mask portions of URL
 Do not show sensitive data such as tokens in the UI.

Secret token
[redacted]

Used to validate received payloads. Sent with the request in the `X-GitLab-Token` HTTP header.

Trigger

- ☐ Push events
- ☐ Tag push events
A new tag is pushed to the repository.
- ☐ Comments
A comment is added to an issue or merge request.
- ☐ Confidential comments
A comment is added to a confidential issue.
- ☐ Issues events
An issue is created, updated, closed, or reopened.
- ☐ Confidential issues events
A confidential issue is created, updated, closed, or reopened.
- ☐ Merge request events
A merge request is created, updated, or merged.
- ☐ Job events
A job's status changes.
- ☐ Pipeline events
A pipeline's status changes.
- ☐ Wiki page events

- 트리거

- 푸시 이벤트 > 와일드카드 패턴 > 대상 브랜치 이름

Trigger

- ☒ Push events
 - ☐ All branches
 - ☒ Wildcard pattern

 Wildcards such as `*-stable` or `/production/*` are supported.
 - ☐ Regular expression

- add webhook

SSL verification

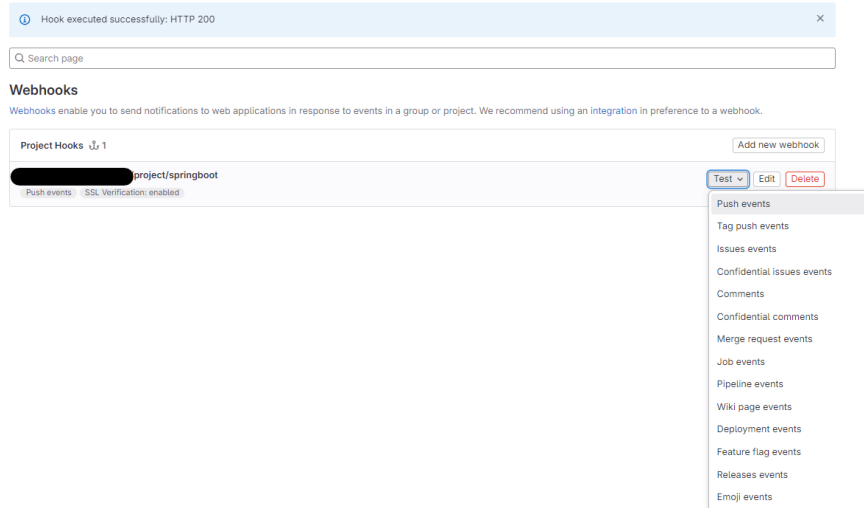
- ☒ Enable SSL verification

Add webhook Cancel

No webhooks enabled. Select trigger events above.

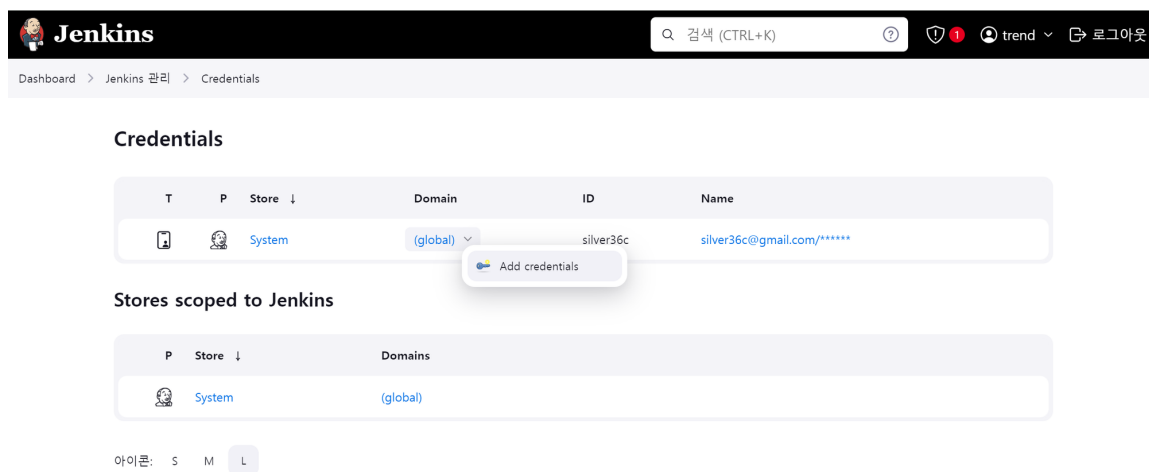
- 확인

- 테스트로 확인해보기



credential (프로그램에서 쓰는 secret key) 추가

- 첫화면 > 관리 > credentials > add credentials



- 내용 입력
 - 유형 선택
 - 파일 업로드

- id에 이름 짓기 - 고유한 이름이어야 함

New credentials

Kind
Secret file

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

File
파일 선택 application-secrets.properties

ID ?
springboot-secrets

Description ?

Create

파이프라인 내용 작성

```

pipeline {
    agent any

    // 환경 변수 설정
    environment {
        gitlabBranch = '<대상브랜치>'
        gitUrl = '<깃랩url>'
        credentialsId = '<상단 gitlab credential 등록 부분에서 입력한 ID>'
    }

    stages {

```



```

stage('Hello') {
    steps {
        echo 'Hello World'
    }
}

stage('Clone Repository') {
    steps {
        echo "Branch : ${env.gitlabBranch}"
        echo "Clone repository"
        git branch: "${env.gitlabBranch}", url: "${env.gitUrl}", credentialsId: "${env.credentialsId }"
    }
}

stage('현재 디렉터리 확인') {
    steps {
        sh 'pwd'
        sh 'ls'
    }
}

stage('springboot credential 추가') {
    steps {
        dir('Backend/src/main/resources/') {
            sh 'ls'
            withCredentials([file(credentialsId: 'springboot-secrets', variable: 'env')]) {
                sh "cp \${env} application-secrets.properties"
            }
        }
    }
}

stage('도커 .sh 실행 권한') {

```

```

        steps {
            dir('docker/') {
                sh "chmod +x grant-permission.sh"
                sh "bash ./grant-permission.sh"
            }
        }

        stage('도커 실행') {
            steps {
                dir('docker/trend-springboot') {
                    withCredentials([file(credentialsId: 'springboot-env', variable: 'env')]) {
                        sh "cp \${env} .env"
                        sh "bash ./run-docker.sh"
                    }
                }
            }
        }

        stage('end') {
            steps {
                echo "파이프라인 종료"
            }
        }
    }
}

```

프론트엔드 파이프라인

- 생성 및 웹훅 등록 과정은 백엔드와 같으므로 생략
- credential (프로그램에서 쓰는 secret key) 추가
 - 경로

- Frontend/.env
- docker/trend-fe/.env
- 내용
 - 문서 하단의 시크릿 키 부분 참고
- 파이프라인 내용

```

pipeline {
    agent any

    // 환경 변수 설정
    environment {
        gitlabBranch = '<대상브랜치>'
        gitUrl = '<깃랩url>'
        credentialsId = '<상단 gitlab credential 등록 부분에서
입력한 ID>'
    }

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }

        stage('Clone Repository') {
            steps {
                echo "Branch : ${env.gitlabBranch}
h}"

                echo "Clone repository"
                git branch: "${env.gitlabBranch}",
url: "${env.gitUrl}", credentialsId: "${env.credentialsId
}"
            }
        }
    }
}

```

```

        stage('현재 디렉터리 확인') {
            steps {
                sh 'pwd'
                sh 'ls'
            }
        }

        stage('credential 추가') {
            steps {
                dir('Frontend/') {
                    sh 'ls'
                    sh 'pwd'
                    withCredentials([file(credentialsId:
'front-app-env', variable: 'env')]) {
                        sh "cp \${env} .env"
                    }
                }
            }
        }

        stage('도커 .sh 실행 권한') {
            steps {
                dir('docker/') {
                    sh "chmod +x grant-permission.sh"
                    sh "bash ./grant-permission.sh"
                }
            }
        }

        stage('도커 실행') {
            steps {
                dir('docker/trend-fe') {
                    withCredentials([file(credentialsId:
'front-docker-env', variable: 'env')]) {
                        sh "cp \${env} .env"
                    }
                }
            }
        }
    }
}

```

```

        sh 'ls -al'
        sh "bash ./run-front.sh"
    }
}
}
}

stage('end') {
    steps {
        echo "파이프라인 종료"
    }
}
}
}


```

데이터

- YoutubeAPI 키 얻기
 - Google Cloud 접속 > Youtube Data API v3 사용 등록 > 관리 > 사용자 인증 정보 > API Key 생성

Google Cloud console

Spend smart, procure faster and retire committed Google Cloud spend with Google Cloud Marketplace. Browse the catalog of over 2000 SaaS, VMs, development stacks, and Kubernetes apps optimized to run on Google Cloud.

 <https://console.cloud.google.com/marketplace/product/google/youtube.googleapis.com?project=clever-gear-395405>

- data_environment.yml

```

name: <이름>
channels:

```

- conda-forge
- defaults

dependencies:

- build=0.7.0=pyhd8ed1ab_0
- ca-certificates=2024.3.11=haa95532_0
- certifi=2024.2.2=pyhd8ed1ab_0
- cffi=1.15.1=py37h2bbff1b_3
- cryptography=38.0.2=py37h953a470_1
- cuda-version=11.2=hb11dac2_3
- cudatoolkit=11.2.0=h608a323_8
- cudnn=8.9.7.29=he6de189_3
- flit-core=3.6.0=pyhd3eb1b0_0
- git=2.44.0=h57928b3_0
- importlib_metadata=4.11.3=hd3eb1b0_0
- libzlib-wapi=1.2.13=hcfcfb64_5
- openssl=3.2.1=hcfcfb64_1
- pep517=0.12.0=py37haa95532_0
- pip=22.3.1=py37haa95532_0
- pycparser=2.21=pyhd3eb1b0_0
- pymysql=1.0.2=py37haa95532_1
- python=3.7.12=h900ac77_100_cpython
- python_abi=3.7=4_cp37m
- setuptools=65.6.3=py37haa95532_0
- sqlite=3.41.2=h2bbff1b_0
- tomli=2.0.1=py37haa95532_0
- ucrt=10.0.22621.0=h57928b3_0
- vc=14.2=h21ff451_1
- vc14_runtime=14.38.33130=h82b7239_18
- vs2015_runtime=14.38.33130=hcb4865c_18
- wheel=0.38.4=py37haa95532_0
- wincertstore=0.2=py37haa95532_2
- pip:
 - absl-py==0.15.0
 - annotated-types==0.5.0
 - anyio==3.7.1
 - argon2-cffi==23.1.0

- argon2-cffi-bindings==21.2.0
- astunparse==1.6.3
- attrs==23.2.0
- backcall==0.2.0
- beautifulsoup4==4.12.3
- bleach==6.0.0
- cached-property==1.5.2
- cachetools==5.3.3
- charset-normalizer==3.3.2
- clang==5.0
- click==8.1.7
- colorama==0.4.6
- comm==0.1.4
- cycler==0.11.0
- debugpy==1.7.0
- decorator==5.1.1
- defusedxml==0.7.1
- entrypoints==0.4
- exceptiongroup==1.2.0
- fastjsonschema==2.19.1
- filelock==3.12.2
- flatbuffers==1.12
- fonttools==4.38.0
- fsspec==2023.1.0
- gast==0.4.0
- google-api-core==2.19.0
- google-api-python-client==2.127.0
- google-auth==2.29.0
- google-auth-httpplib2==0.2.0
- google-auth-oauthlib==0.4.6
- google-pasta==0.2.0
- googleapis-common-protos==1.63.0
- grpcio==1.62.2
- grpcio-tools==1.62.2
- h11==0.14.0
- h2==4.1.0

- h5py==3.1.0
- hpack==4.0.0
- httpcore==0.17.3
- httplib2==0.22.0
- httpx==0.24.1
- huggingface-hub==0.16.4
- hyperframe==6.0.1
- idna==3.7
- importlib-metadata==6.7.0
- importlib-resources==5.12.0
- ipykernel==6.16.2
- ipython==7.34.0
- ipython-genutils==0.2.0
- ipywidgets==8.1.2
- jedi==0.19.1
- jinja2==3.1.3
- joblib==1.3.2
- jpype1==1.5.0
- jsonschema==4.17.3
- jupyter==1.0.0
- jupyter-client==7.4.9
- jupyter-console==6.6.3
- jupyter-core==4.12.0
- jupyter-server==1.24.0
- jupyterlab-pygments==0.2.2
- jupyterlab-widgets==3.0.10
- keras==2.11.0
- keras-preprocessing==1.1.2
- kiwipiepy==0.17.1
- kiwipiepy-model==0.17.0
- kiwisolver==1.4.5
- konlpy==0.6.0
- lxml==5.2.1
- markdown==3.4.4
- markupsafe==2.1.5
- matplotlib==3.5.3

- matplotlib-inline==0.1.6
- mistune==3.0.2
- nbclassic==1.0.0
- nbclient==0.7.4
- nbconvert==7.6.0
- nbformat==5.8.0
- nest-asyncio==1.6.0
- networkx==2.6.3
- nltk==3.8.1
- notebook==6.5.6
- notebook-shim==0.2.4
- numpy==1.19.5
- oauthlib==3.2.2
- opt-einsum==3.3.0
- packaging==24.0
- pandas==1.3.5
- pandocfilters==1.5.1
- parso==0.8.4
- pickleshare==0.7.5
- pillow==9.5.0
- pkgutil-resolve-name==1.3.10
- portalocker==2.7.0
- prometheus-client==0.17.1
- prompt-toolkit==3.0.43
- proto-plus==1.23.0
- protobuf==4.24.4
- psutil==5.9.8
- pyasn1==0.5.1
- pyasn1-modules==0.3.0
- pydantic==2.5.3
- pydantic-core==2.14.6
- pygments==2.17.2
- pyparsing==3.1.2
- pyrsistent==0.19.3
- python-dateutil==2.9.0.post0
- python-dotenv==0.21.1

- pytz==2024.1
- pywin32==306
- pywinpty==2.0.10
- pyyaml==6.0.1
- pyzmq==24.0.1
- qdrant-client==1.4.0
- qtconsole==5.4.4
- qtpy==2.4.1
- regex==2024.4.16
- requests==2.31.0
- requests-oauthlib==2.0.0
- rsa==4.9
- safetensors==0.4.3
- scikit-learn==1.0.2
- scipy==1.7.3
- send2trash==1.8.3
- sentence-transformers==2.2.2
- sentencepiece==0.2.0
- six==1.15.0
- sniffio==1.3.1
- soupsieve==2.4.1
- soynlp==0.0.493
- tensorboard==2.11.2
- tensorboard-data-server==0.6.1
- tensorboard-plugin-wit==1.8.1
- tensorflow-estimator==2.15.0
- tensorflow-gpu==2.6.0
- termcolor==1.1.0
- terminado==0.17.1
- threadpoolctl==3.1.0
- tinycss2==1.2.1
- tokenizers==0.13.3
- torch==1.8.1+cu111
- torchaudio==0.8.1
- torchvision==0.9.1+cu111
- tornado==6.2

```
- tqdm==4.66.2
- traitlets==5.9.0
- transformers==4.30.2
- typing-extensions==4.7.1
- uritemplate==4.1.1
- urllib3==1.26.18
- wcwidth==0.2.13
- webencodings==0.5.1
- websocket-client==1.6.1
- werkzeug==2.2.3
- widgetsnbextension==4.0.10
- wrapt==1.12.1
- youtube-transcript-api==0.6.2
- zipp==3.15.0
prefix: C:\ProgramData\anaconda3\envs\<이름>
```

- 추천 시스템 실행 방법



도커에서 Qdrant가 실행 중인 상태에서 진행

1. /data/ptcharmDirectory/youtube/Trends_book_system.py 열기
2. collectionDate 변수에 원하는 날짜 기입
3. /data/ptcharmDirectory/youtube/Trends_book_system.py 실행

시크릿 키

trend-db

docker/trend-db/.env

```
# MySQL
MYSQL_ROOT_PASSWORD=<패스워드>
MYSQL_DATABASE=<DB이름>
MYSQL_USER=<유저이름>
MYSQL_PASSWORD=<패스워드>
HOST_PORT=<포트>

# Redis
REDIS_PASSWORD=<패스워드>
```

trend-fastapi

PythonAPI/pythonProject/.env

```
#APP
APP_PORT=<서비스 할 포트>
#Qdrant
PORT=<Qdrant 포트>
QDRANT_HOST=trend-qdrant-container
# MySQL
MYSQL_ROOT_PASSWORD=<패스워드>
MYSQL_DATABASE=<DB이름>
MYSQL_USER=<유저이름>
MYSQL_PASSWORD=<패스워드>
HOST_PORT=<포트>
DB_HOST=trend-mysql-container
```

trend-qdrant

docker/trend-qdrant/.env

```
PORT=<Qdrant 포트>
```

```
QDRANT_HOST=trend-qdrant-container
```

docker/trend-qdrant

- books.json

형식

```
{
  'category_id': '01',
  'product_id': 1,
  'search_keyword': '서치키워드',
  'total_click_count': 0,
  'total_order_count': 0,
  'total_order_amount': 0,
  'product_name': '책이름',
  'sale_price': 100.0,
  'category_name': '카테고리이름'
}
```

- embeddings_topic.npy
 - books.json을 임베딩하여 numpy 형식으로 만든 파일

trend-springboot

docker/trend-springboot/.env

```
PORT=<포트번호>
```

/Backend/src/main/resources/application-secrets.properties

```
# MySQL
MYSQL_ROOT_PASSWORD=<패스워드>
MYSQL_DATABASE=<DB이름>
```

```
MYSQL_USER=<유저이름>
MYSQL_PASSWORD=<패스워드>
HOST_PORT=<포트>
DB_HOST=trend-mysql-container
# Redis
REDIS_HOST=trend-redis-container
REDIS_PASSWORD=<패스워드>
# Frontend-domain
FRONT_DOMAIN=<도메인> // localhost:8080 형식
# Back-domain
BACK_DOMAIN=<도메인> //http://localhost:8080 형식
# secretkey
SECURE_KEY_PLAIN=<시크릿키>
# MatterMost webhook uri
MATTERMOST_WEBHOOK=<uri>
```

trend-front

docker/trend-fe/.env

```
PORT=<포트>
```

Frontend/.env

```
VITE_GA4_CLIENT_ID=<VITE_GA4_CLIENT_ID>
VITE_GA4_CLIENT_SECRET=<VITE_GA4_CLIENT_SECRET>
VITE_GA4_OAUTH_REFRESH_TOKEN=<VITE_GA4_OAUTH_REFRESH_TOKEN>
VITE_GA4_PROPERTY_ID=<VITE_GA4_PROPERTY_ID>
```