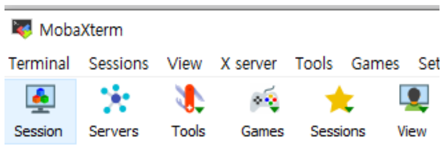
🍀

포팅매뉴얼

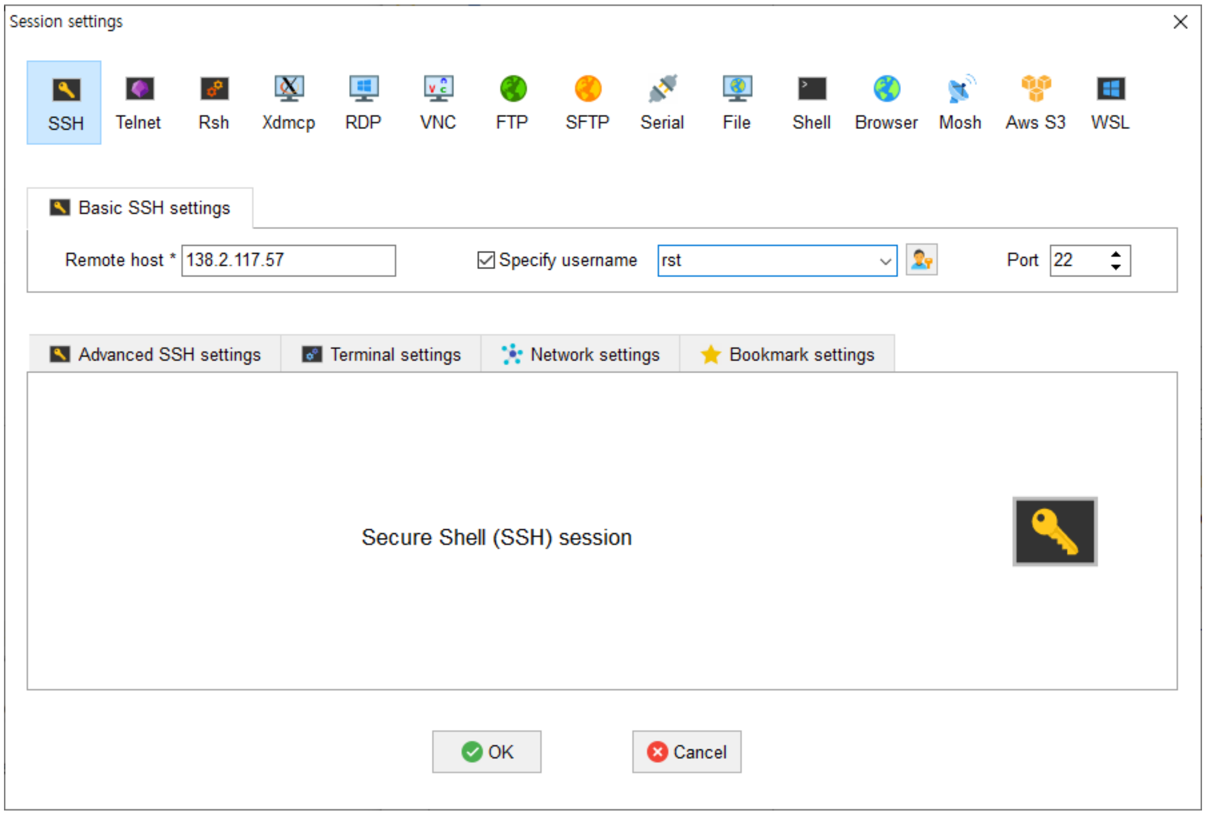
가상환경 세팅

SSAFY 서버 접속

1. 공개키 다운로드
2. mobeXTerm 실행 (다운로드)
3. mobaXTerm 상단 내비에서 `Session` 열기



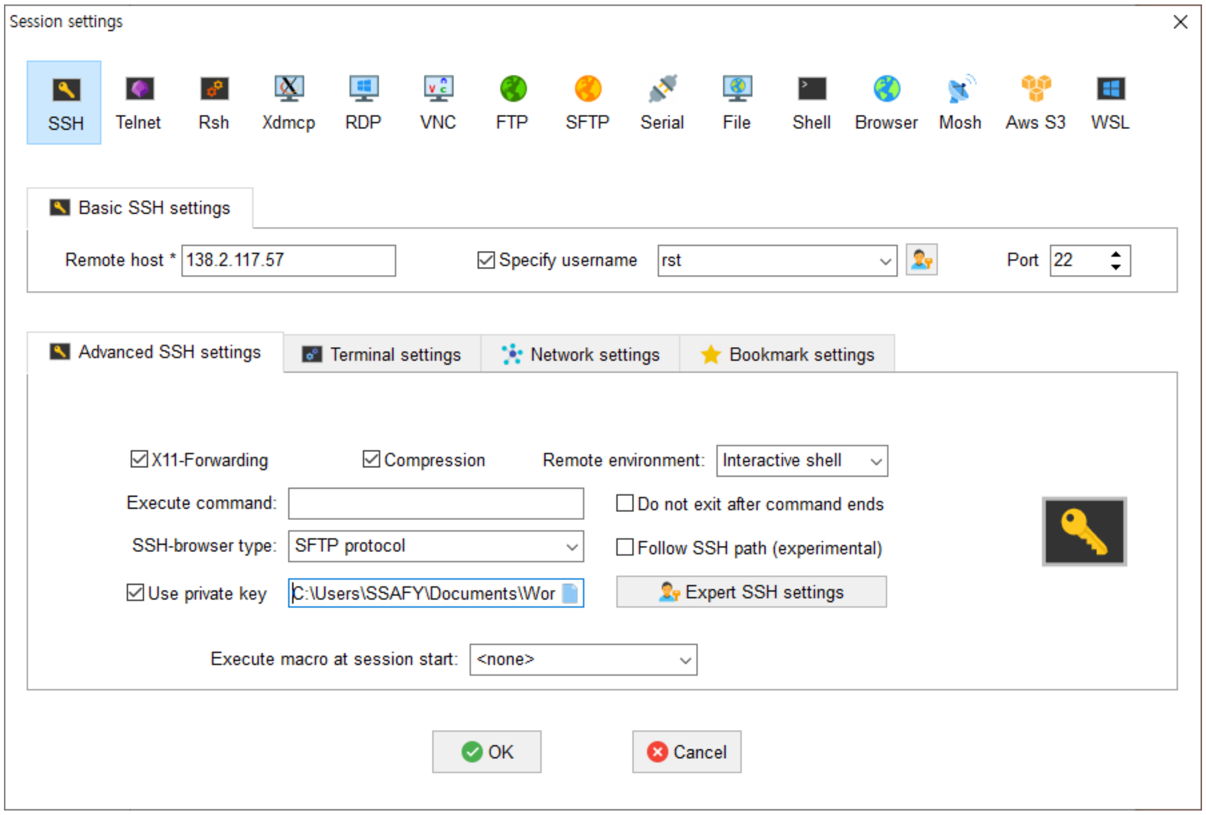
4. 서버 도메인: `j8e101.p.ssafy.io`


Session settings

Remote host * 138.2.117.57 ☑ Specify username rst Port 22

Secure Shell (SSH) session

5. `Advanced SSH settings` 에서 `Use private key` 를 클릭, 1번에서 다운로드 받은 공개키 삽입


Session settings

Remote host * 138.2.117.57 ☑ Specify username rst Port 22

☑ X11-Forwarding ☑ Compression Remote environment: Interactive shell

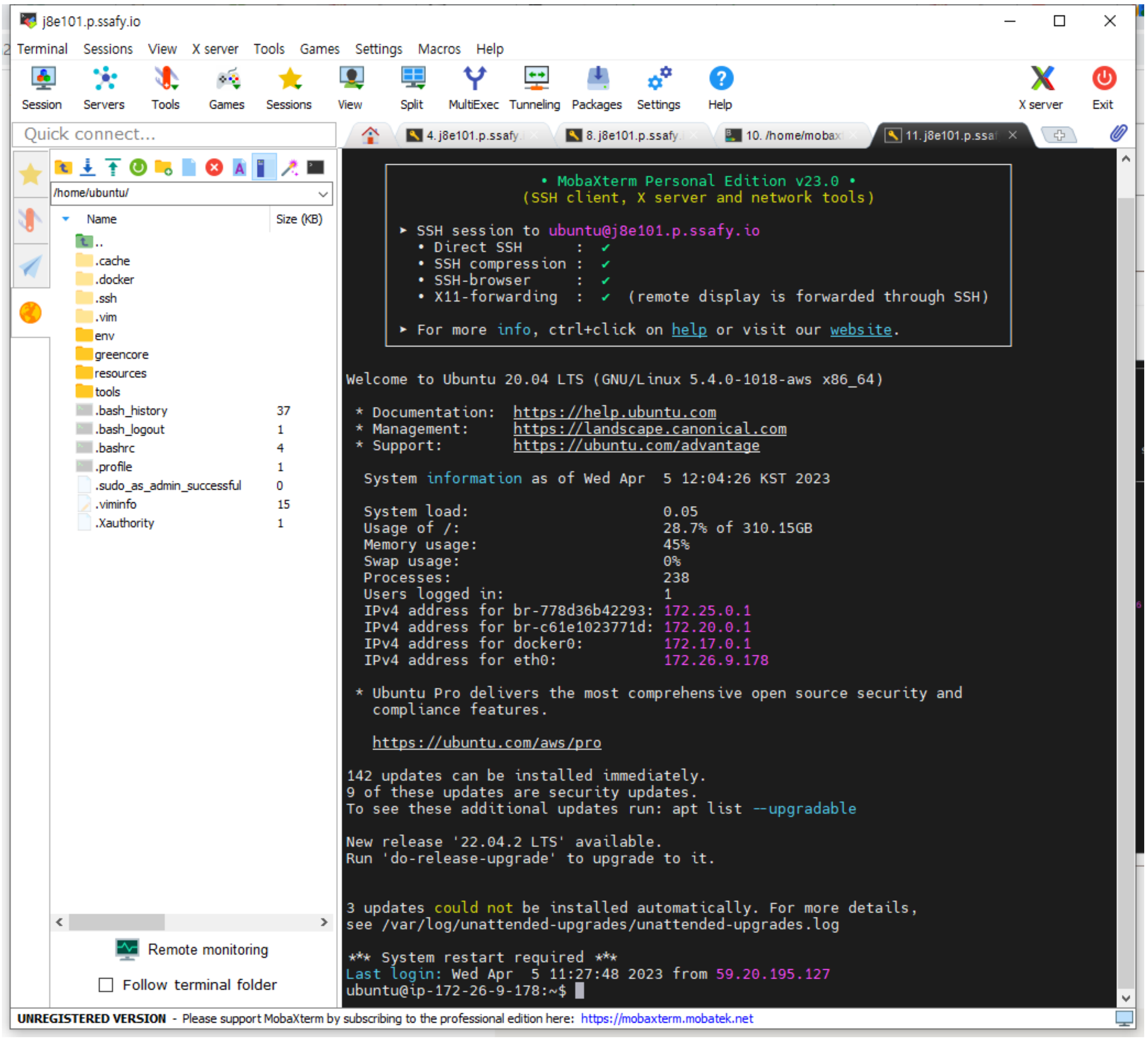
Execute command: ☐ Do not exit after command ends

SSH-browser type: SFTP protocol ☐ Follow SSH path (experimental)

☑ Use private key C:\Users\SSAFY\Documents\Wor

Execute macro at session start: <none>

6. OK 클릭
7. `login as :` 에 `ubuntu` 입력
8. 아래처럼 접속된 걸 확인할 수 있습니다



```
• MobaXterm Personal Edition v23.0 •
(SSH client, X server and network tools)

▸ SSH session to ubuntu@j8e101.p.ssafy.io
  • Direct SSH      :  ✔
  • SSH compression :  ✔
  • SSH-browser     :  ✔
  • X11-forwarding  :  ✔  (remote display is forwarded through SSH)

▸ For more info, ctrl+click on help or visit our website.

Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Apr  5 12:04:26 KST 2023

  System load:                      0.05
  Usage of /:                       28.7% of 310.15GB
  Memory usage:                     45%
  Swap usage:                       0%
  Processes:                        238
  Users logged in:                  1
  IPv4 address for br-778d36b42293: 172.25.0.1
  IPv4 address for br-c61e1023771d: 172.20.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for eth0:            172.26.9.178

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

142 updates can be installed immediately.
9 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '22.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


3 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

*** System restart required ***
Last login: Wed Apr  5 11:27:48 2023 from 59.20.195.127
ubuntu@ip-172-26-9-178:~$
```

Docker 설치

- ubuntu에 도커를 설치한다.

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

NGINX 설치

- 우분투에 nginx 설치 명령어 입력

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

SSL 인증

- CertBot 다운로드 하기
  - `apt` 에 있는 CertBot 패키지도 있지만, 업데이트가 뜸한 편이기에 레포지토리를 등록하여 패키지를 다운로드 받기

```
#1 아래 명령어를 입력하고 Enter 입력
sudo add-apt-repository ppa:certbot/certbot

#2 ubuntu 22.04 이상
sudo apt-get install python3-certbot-nginx
```

- SSL 인증서를 적용하여 https 적용하기

```
#1. 아래 명령어 입력
sudo certbot --nginx -d [자신의도메인]

#2. 이메일 입력 ex) dbtmdxo1992@gamil.com

#3. 약관 동의 여부 질문 : A 입력

#4. EFF에 이메일 주소 공유 (optional) 질문 : N

#5. Redirct 설정 옵션 : 2 입력
- 1을 입력한다면 http 연결을 https로 리다이렉트 하지 않음
- 2를 입력한다면 http 연결을 https로 리다이렉트 시킴
```

프록시 설정

- `/etc/nginx/site-available/default` 파일 수정

```
server {

    if ($host = j8e101.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
        server_name j8e101.p.ssafy.io;



    return 404; # managed by Certbot
}

server {
        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/j8e101.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/j8e101.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

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
    server_name j8e101.p.ssafy.io; # managed by Certbot


        location / {
                proxy_pass http://j8e101.p.ssafy.io:3000;
                proxy_set_header Host $host:$server_port;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                add_header 'Access-Control-Allow-Origin' '*';
        }
        # db 파일 경로 설정
        location /resources {
            alias /home/ubuntu/resources/static;
        }


        # next.js 내부 static 파일 가져오기
        location /_next/static/ {
                proxy_pass http://j8e101.p.ssafy.io:3000;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        # backend api 요청 경로 설정
        location /api {
                proxy_pass http://j8e101.p.ssafy.io:5000/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;
                add_header 'Access-Control-Allow-Origin' '*';
                # wss setting
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Origin "";
        }
}
```

도커 컨테이너

1. Jenkins

Jenkins 컨테이너 실행 및 계정 생성

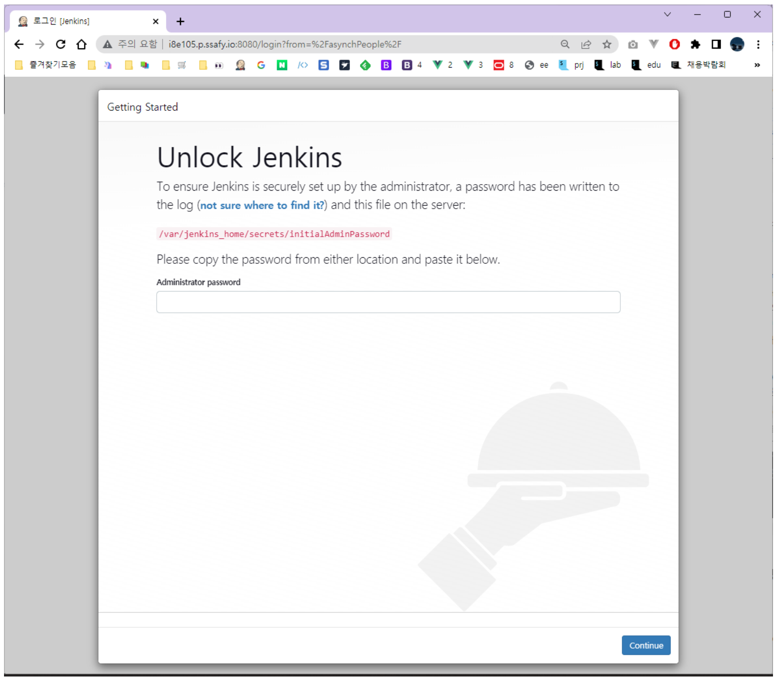
- jenkins-compose.yml 파일을 생성한다.

```
services:
    # 서비스 명
    jenkins:
        image: jenkins/jenkins:lts
        container_name: jenkins
        build: .
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
            - /home/ubuntu/greencore/jenkins_home:/var/jenkins_home
            - /home/ubuntu/greencore/conf:/var/conf
            - /home/ubuntu/greencore/app:/app
        ports:
            - "8080:8080"
        privileged: true
        user: root
```

- Jenkins를 컨테이너를 실행한다.

```
sudo docker-compose -f {yml 파일 이름} up -d
```

- [서버주소]:8080 (`j8e101.p.ssafy.io:8080/`) 으로 이동하면 Jenkins 로그인창이 나온다.


Getting Started

Unlock Jenkins

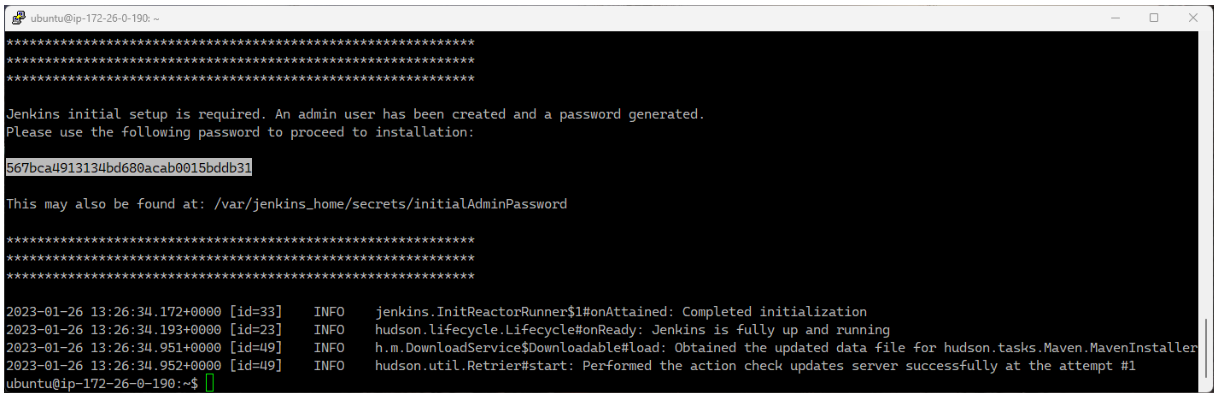
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

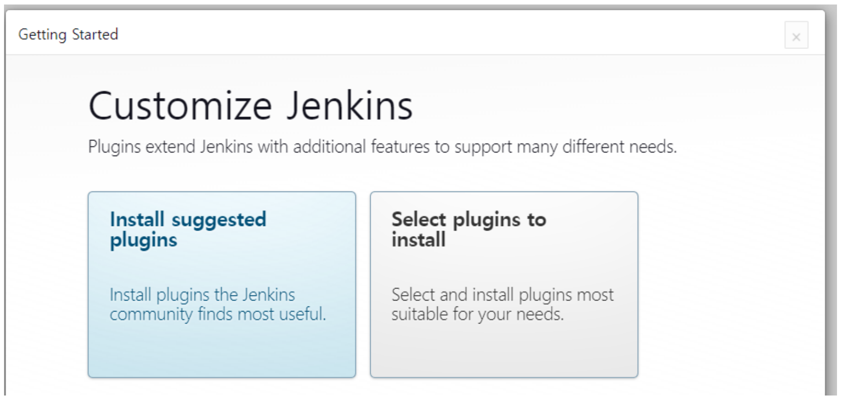
Please copy the password from either location and paste it below.

Administrator password

- `sudo docker logs jenkins` 명령어로 관리자 비밀번호를 찾아서 창에 입력한다.


```
ubuntu@ip-172-26-0-190: ~
*************************************************************
*************************************************************
*************************************************************

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

567bca491313 4bd680acab0015bddb31

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

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

2023-01-26 13:26:34.172+0000 [id=33]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2023-01-26 13:26:34.193+0000 [id=23]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2023-01-26 13:26:34.951+0000 [id=49]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2023-01-26 13:26:34.952+0000 [id=49]    INFO    hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
ubuntu@ip-172-26-0-190:~$
```

- `Install Suggested plugins` 를 선택한다.


Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

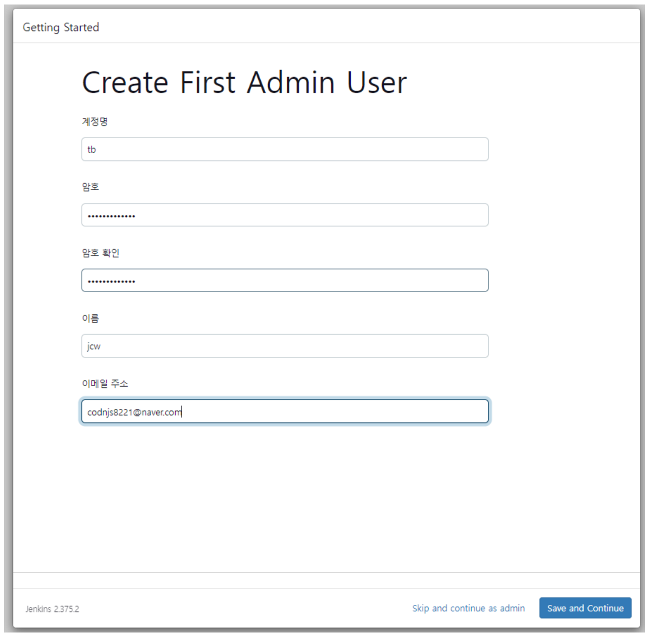
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

- 플러그인들이 설치되고 나면, 관리자 계정을 만들고 `save and continue` > `save and finish` > `start using jankins`를 누른다.


Getting Started

Create First Admin User

계정명
tb

암호
••••••••••••••

암호 확인
••••••••••••••

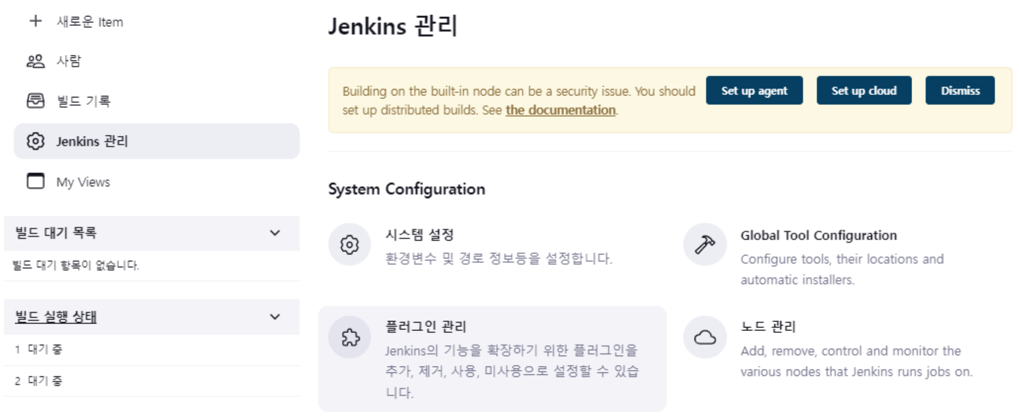
이름
jcw

이메일 주소
codnjs8221@naver.com

Jenkins 2.375.2

Jenkins 프로젝트 설정

1. Jenkins 관리 > 플러그인 관리를 누른다.



빌드 대기 항목이 없습니다.

1 대기 중
2 대기 중

Jenkins 관리

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

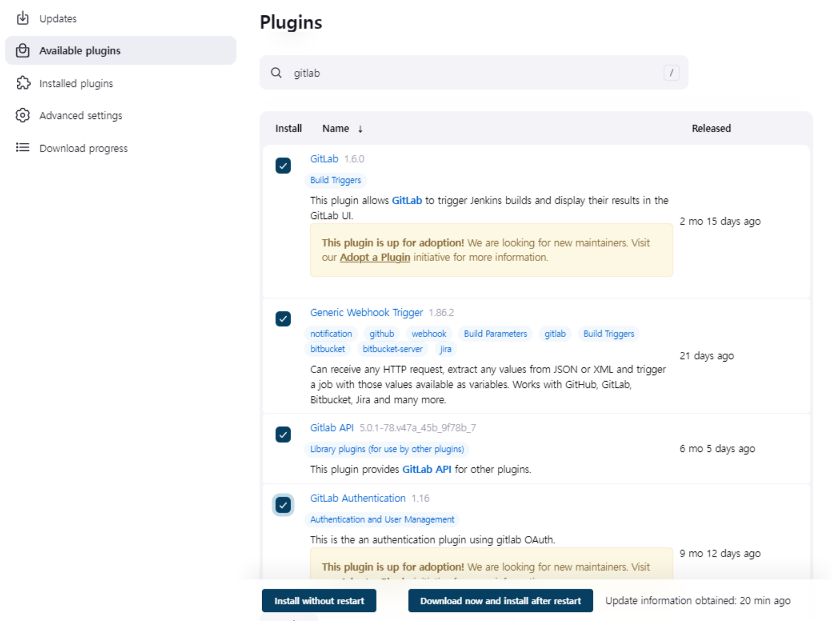
시스템 설정
환경변수 및 경로 정보등을 설정합니다.

Global Tool Configuration
Configure tools, their locations and automatic installers.

플러그인 관리
Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.

노드 관리
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

2. Gitlab 관련 플러그인을 설치한다. > `Install without restart` 를 누른다.



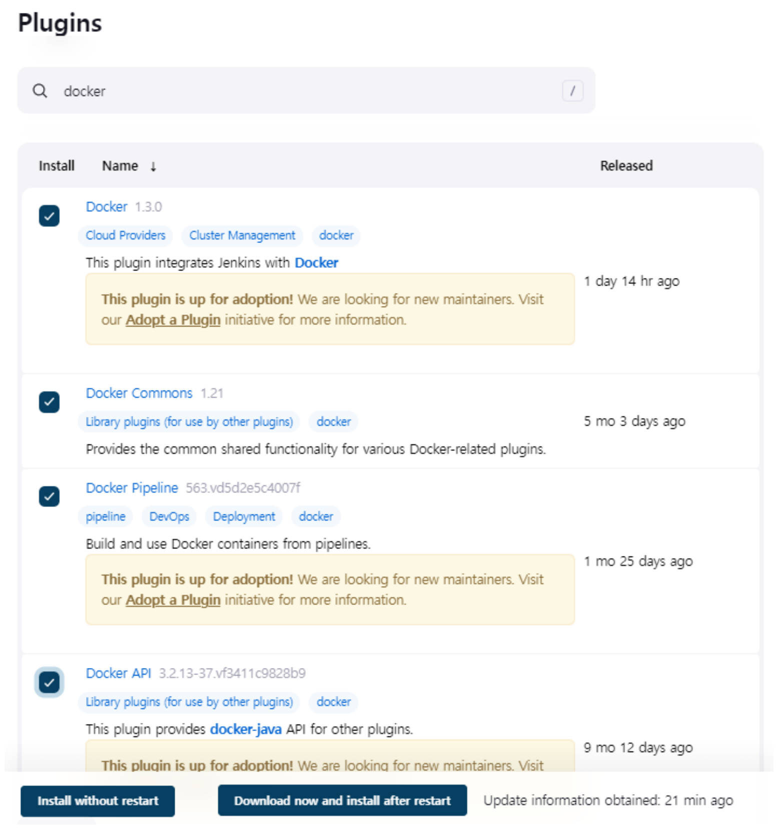
Plugins

gitlab

Install	Name ↓	Released
✓	GitLab 1.6.0 Build Triggers This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	2 mo 15 days ago
✓	Generic Webhook Trigger 1.86.2 notification github webhook Build Parameters gitlab Build Triggers bitbucket bitbucket-server jira Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.	21 days ago
✓	Gitlab API 5.0.1-78.v47a_45b_9f78b_7 Library plugins (for use by other plugins) This plugin provides GitLab API for other plugins.	6 mo 5 days ago
✓	GitLab Authentication 1.16 Authentication and User Management This is the an authentication plugin using gitlab OAuth. This plugin is up for adoption! We are looking for new maintainers. Visit	9 mo 12 days ago

Update information obtained: 20 min ago

3. 마찬가지로 Docker 관련 플러그인을 설치한다.



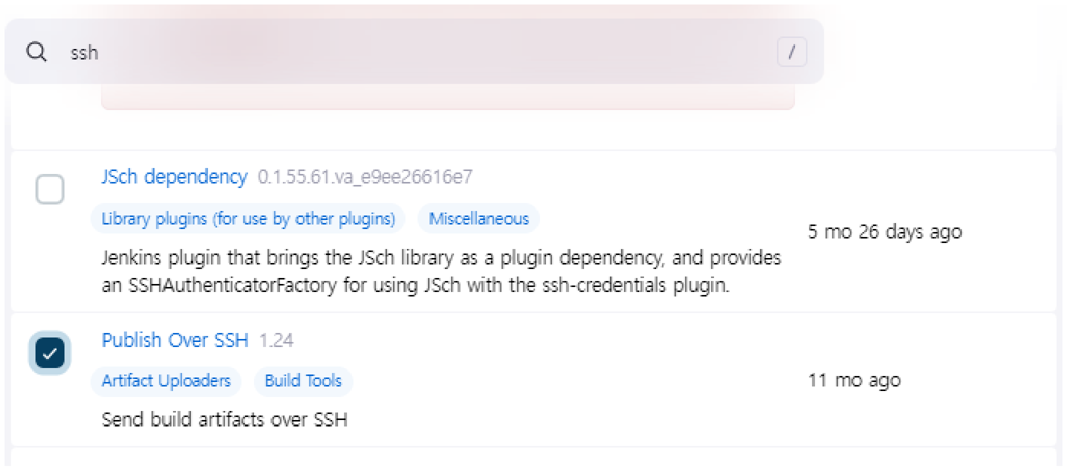
Plugins

docker

Install	Name ↓	Released
✓	Docker 1.3.0 Cloud Providers Cluster Management docker This plugin integrates Jenkins with Docker This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	1 day 14 hr ago
✓	Docker Commons 1.21 Library plugins (for use by other plugins) docker Provides the common shared functionality for various Docker-related plugins.	5 mo 3 days ago
✓	Docker Pipeline 563.vd5d2e5c4007f pipeline DevOps Deployment docker Build and use Docker containers from pipelines. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	1 mo 25 days ago
✓	Docker API 3.2.13-37.vf3411c9828b9 Library plugins (for use by other plugins) docker This plugin provides docker-java API for other plugins. This plugin is up for adoption! We are looking for new maintainers. Visit	9 mo 12 days ago

Update information obtained: 21 min ago

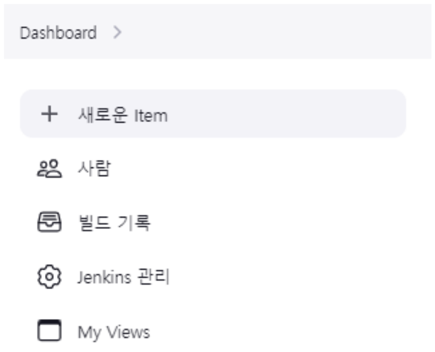
4. 마찬가지로 SSH 관련 플러그인을 설치한다.


ssh

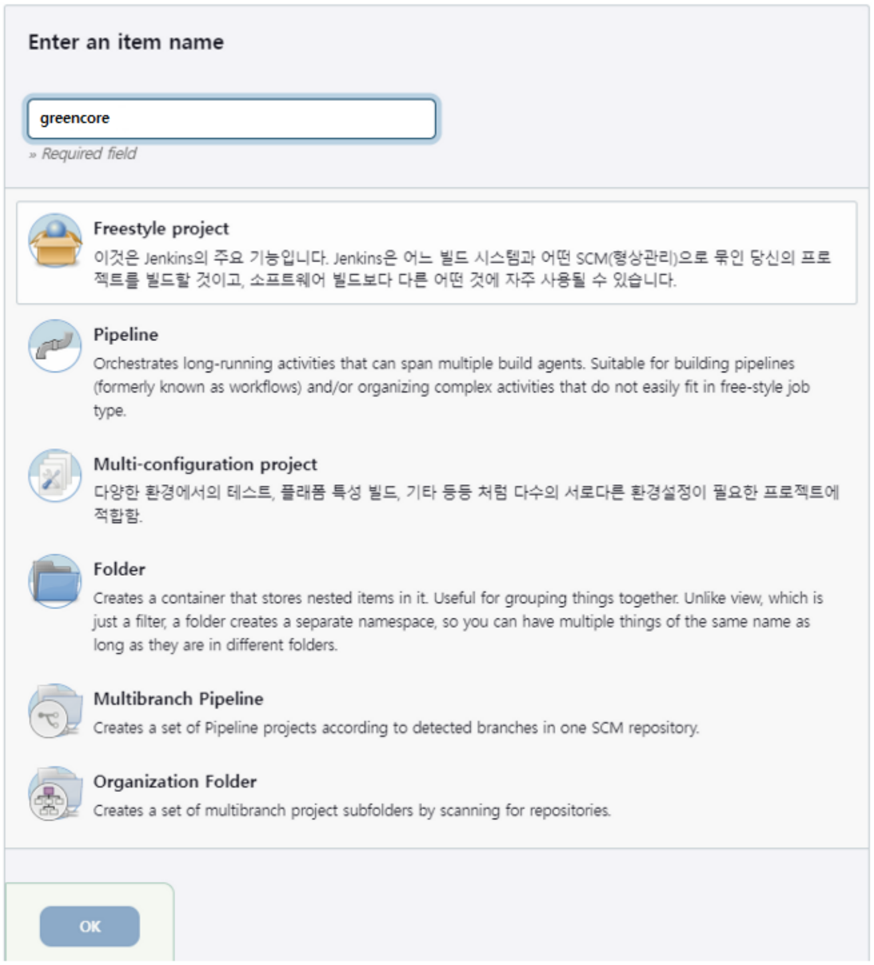
☐	JSch dependency 0.1.55.61.va_e9ee26616e7 Library plugins (for use by other plugins) Miscellaneous Jenkins plugin that brings the JSch library as a plugin dependency, and provides an SSHAuthenticatorFactory for using JSch with the ssh-credentials plugin.	5 mo 26 days ago
✓	Publish Over SSH 1.24 Artifact Uploaders Build Tools Send build artifacts over SSH	11 mo ago

Jenkins 프로젝트와 Gitlab 연동 및 Webhook 설정

1. `새로운 item` 을 클릭한다.



2. 생성할 item의 이름을 입력하고, `Freestyle project` 를 클릭한다.


Enter an item name

greencore

» Required field

Freestyle project
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

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.

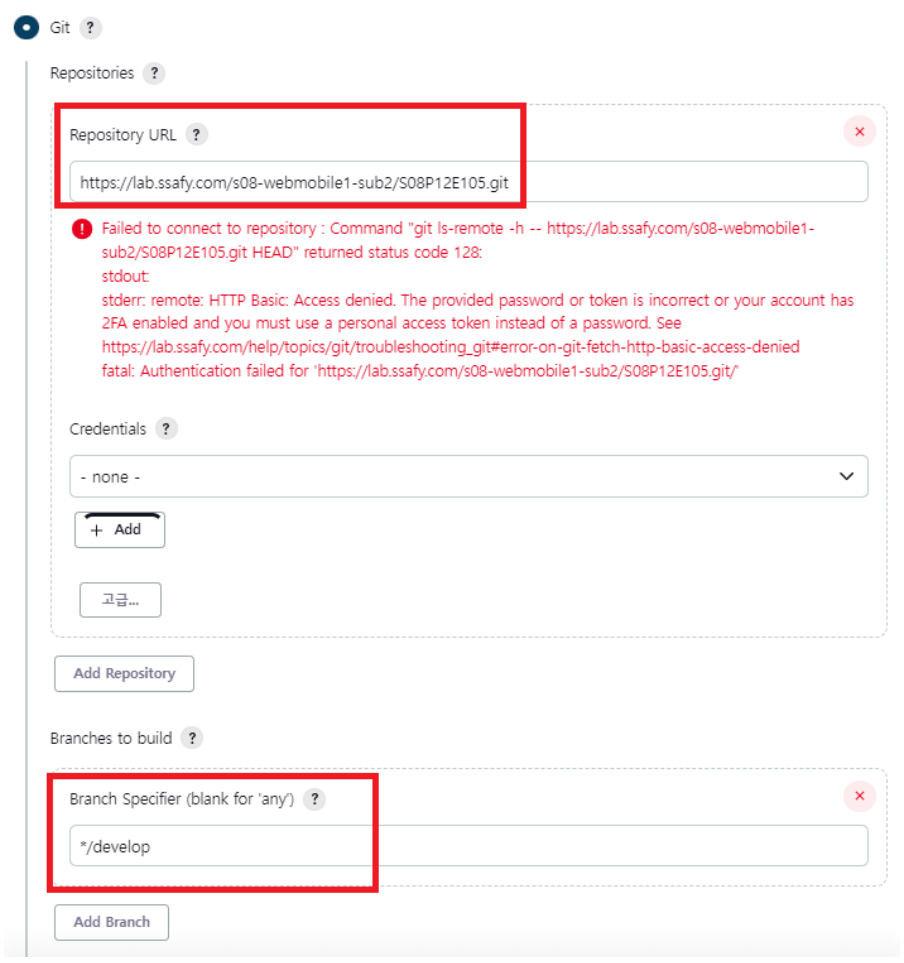
Multi-configuration project
다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

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.

3. Git 탭에 만들어둔 gitlab 레포지토리의 url을 복사해서 붙여넣기, 빌드할 브랜치를 적는다.
   (아직 Credential을 등록하지 않았으므로 에러 메세지가 나타나면 정상이다.)


Git

Repositories

Repository URL

```
https://lab.ssafy.com/s08-webmobile1-sub2/S08P12E105.git
```

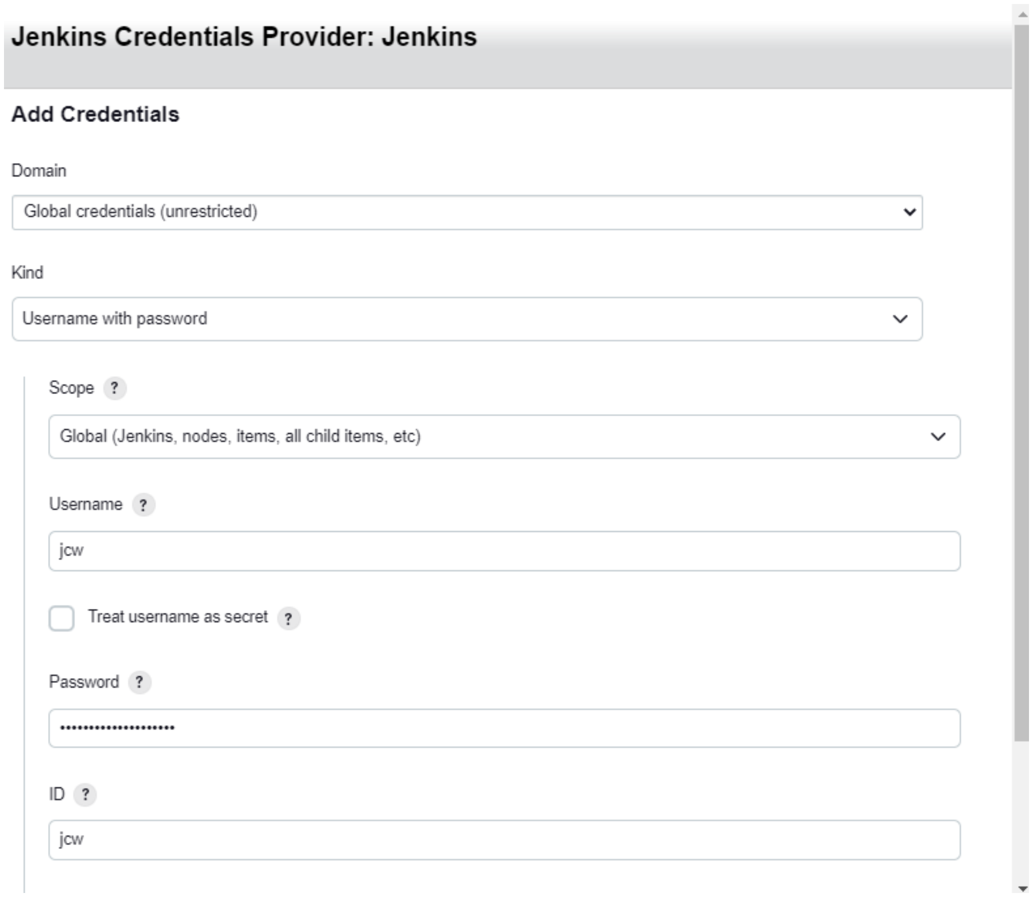
```
Failed to connect to repository : Command "git ls-remote -h -- https://lab.ssafy.com/s08-webmobile1-sub2/S08P12E105.git HEAD" returned status code 128:
stdout:
stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password. See https://lab.ssafy.com/help/topics/git/troubleshooting_git#error-on-git-fetch-http-basic-access-denied
fatal: Authentication failed for 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12E105.git/'
```

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */develop

4. Credentials의 `Add` 버튼 > Jenkins를 클릭한다.
5. 계정을 입력한다. (ID는 Credential 을 구분하기 위한 것이므로 아무거나 적으면 된다.)


Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Username with password

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

Username: jcw

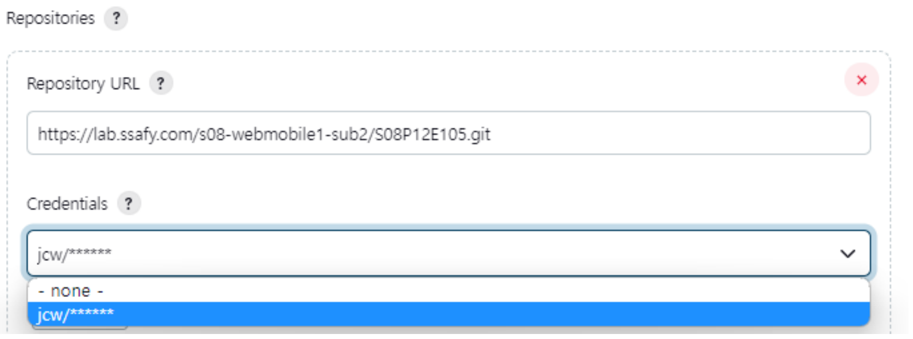
[ ] Treat username as secret

Password: ••••••••••••••••••••

ID: jcw

- Username : 싸피깃 아이디
- Password : 싸피깃 비밀번호

6. 추가한 Credential을 선택하고, 에러 메세지가 없어지는 걸 확인한다.


Repositories

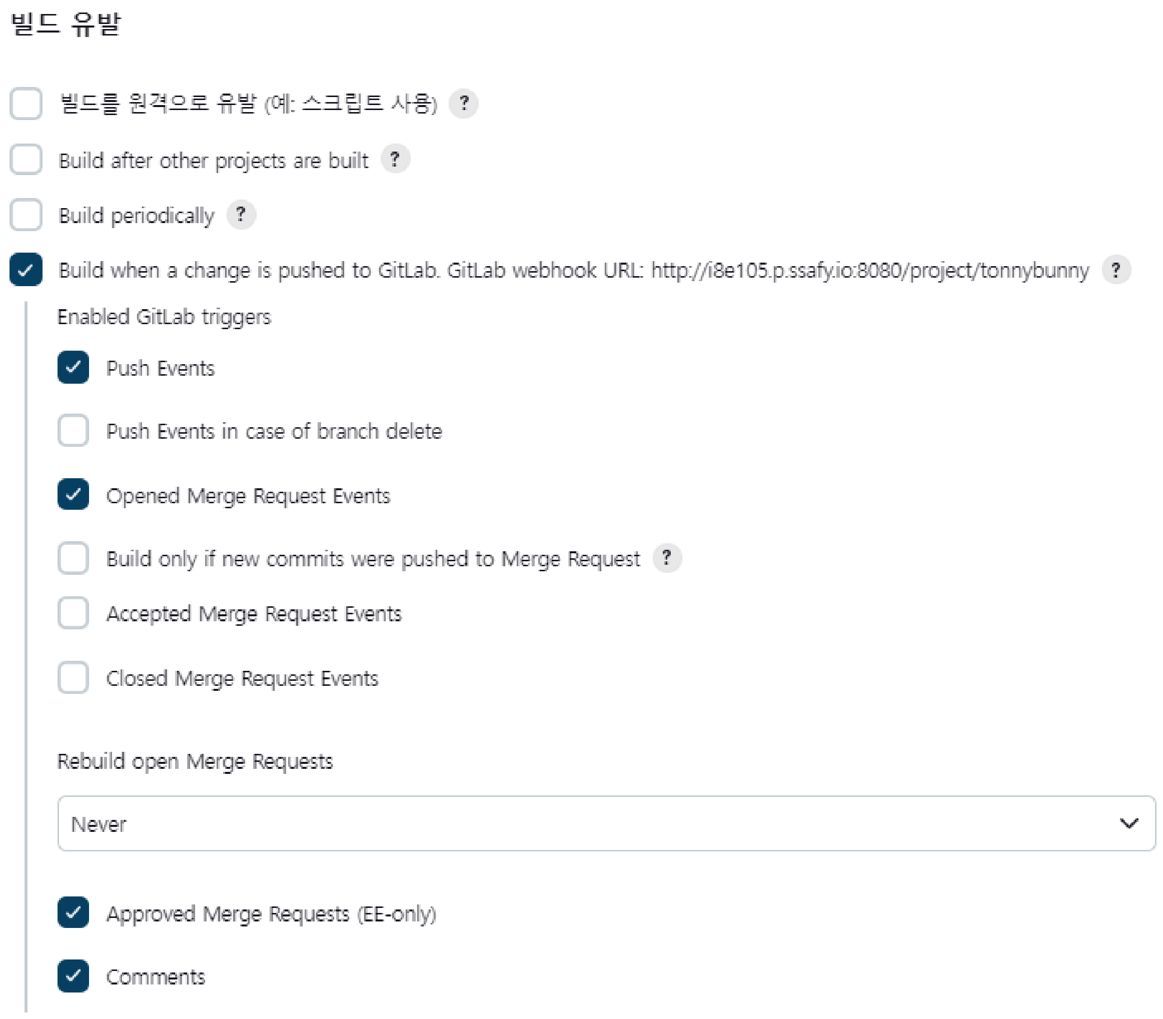
Repository URL: https://lab.ssafy.com/s08-webmobile1-sub2/S08P12E105.git

Credentials: jcw/******

- none -

jcw/******

7. 빌드 유발 탭에서 아래와 같이 선택한다. 언제 빌드를 할지 트리거 이벤트를 설정한다.



빌드 유발

- [ ] 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [✓] Build when a change is pushed to GitLab. GitLab webhook URL: http://i8e105.p.ssafy.io:8080/project/tonnybunny ?

Enabled GitLab triggers

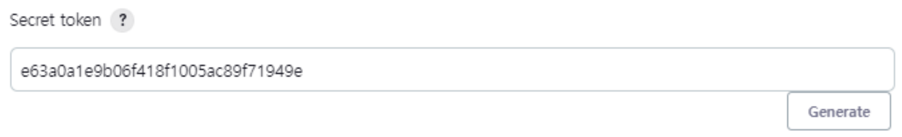
- [✓] Push Events
- [ ] Push Events in case of branch delete
- [✓] Opened Merge Request Events
- [ ] Build only if new commits were pushed to Merge Request ?
- [ ] Accepted Merge Request Events
- [ ] Closed Merge Request Events

Rebuild open Merge Requests

Never

- [✓] Approved Merge Requests (EE-only)
- [✓] Comments

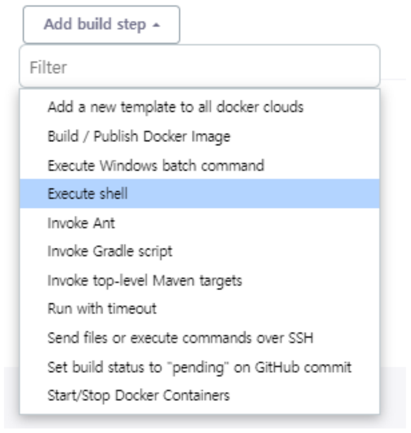
8. 빌드 유발 탭 > 고급 버튼을 누른 후, Secret Token을 받아서 저장해둔다.

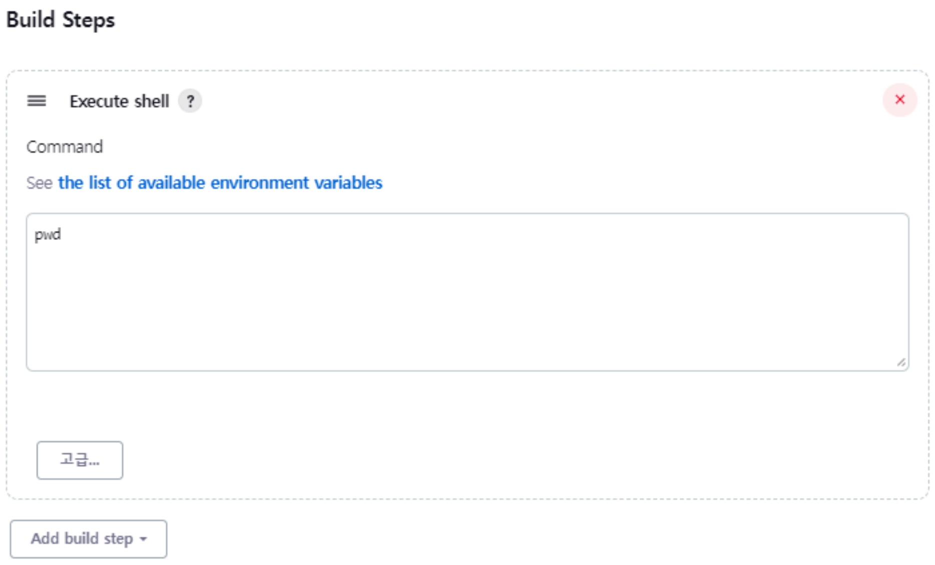


Secret token ?

e63a0a1e9b06f418f1005ac89f71949e

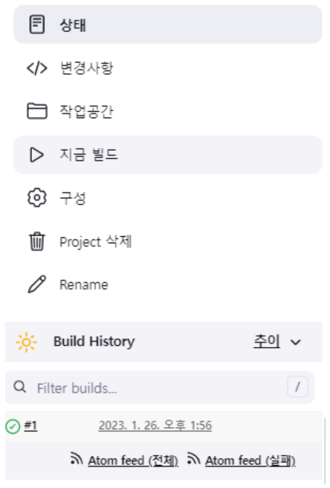
9. Build Steps > Add build step > Execute shell을 누른 후, `pwd` 를 입력하고 저장한다.

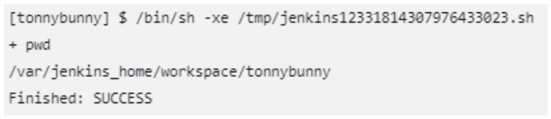

Add build step
Filter
Add a new template to all docker clouds
Build / Publish Docker Image
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Send files or execute commands over SSH
Set build status to "pending" on GitHub commit
Start/Stop Docker Containers



```
pwd
```

10. 지금 빌드 버튼을 누르고, 젠킨스 프로젝트 빌드가 잘 동작하는지 확인한다.

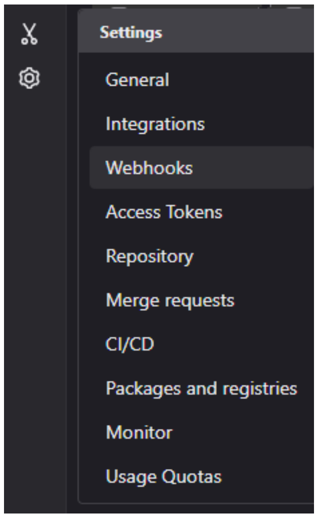




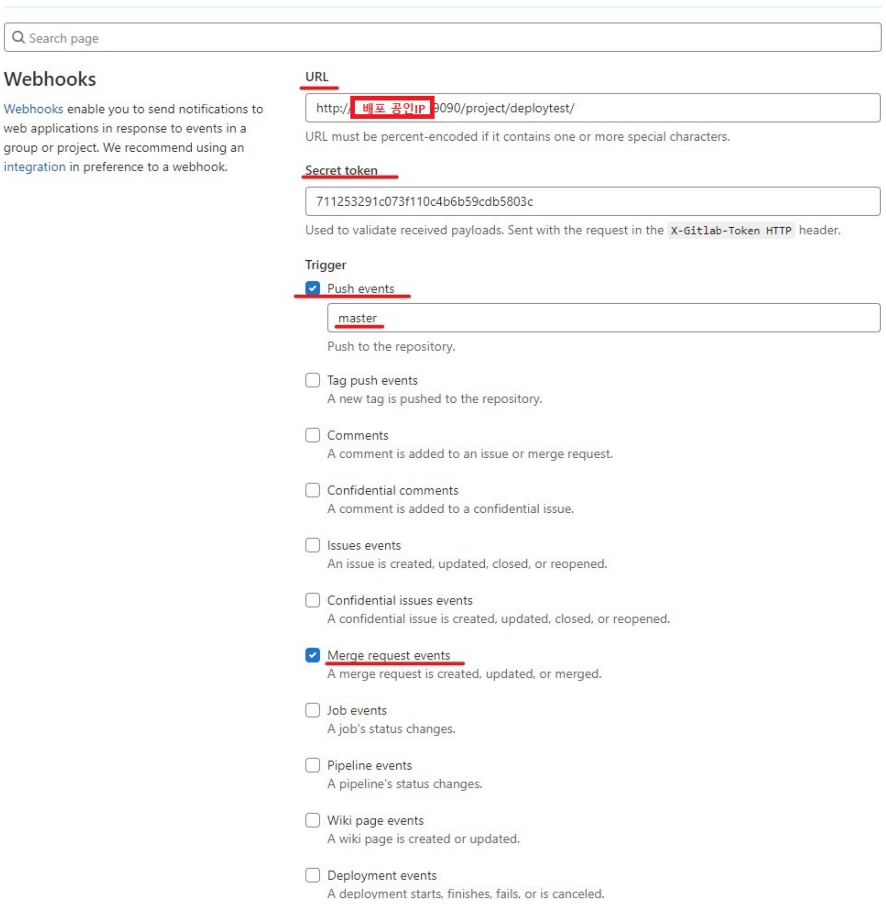
```
[tonnybunny] $ /bin/sh -xe /tmp/jenkins12331814307976433023.sh
+ pwd
/var/jenkins_home/workspace/tonnybunny
Finished: SUCCESS
```

Gitlab Webhook 설정

1. Gitlab의 Webhook을 선택한다.



2. 아까 얻은 secret token을 입력하고, trigger를 설정한 후, `Add webhook` 을 클릭한다.



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.

URL

http:// 배포 공인IP :9090/project/deploytest/

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

Secret token

711253291c073f110c4b6b59cdb5803c

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

Trigger

- [x] Push events

  master

  Push to the repository.
- [ ] 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.
- [x] Merge request events

  A merge request is created, updated, or merged.
- [ ] Job events

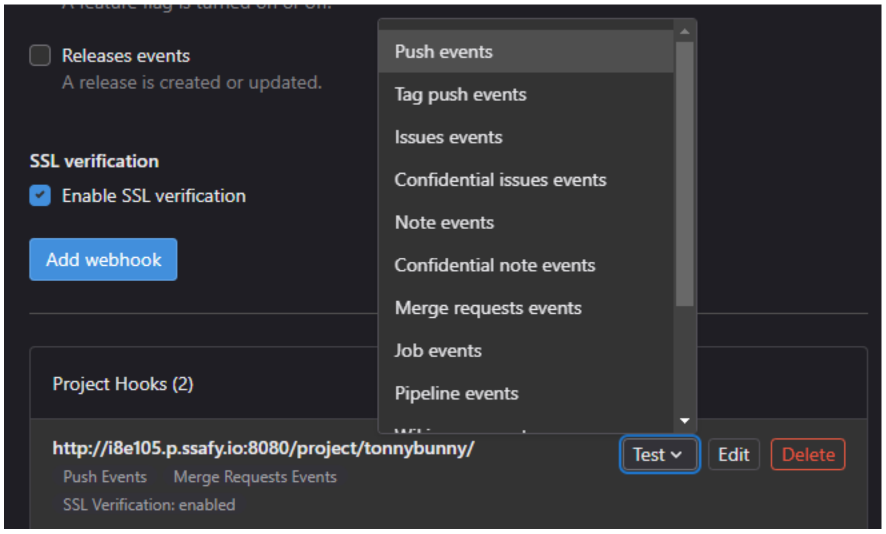
  A job's status changes.
- [ ] Pipeline events

  A pipeline's status changes.
- [ ] Wiki page events

  A wiki page is created or updated.
- [ ] Deployment events

  A deployment starts, finishes, fails, or is canceled.

3. 생성된 Webhook에서 Test > Push Events를 선택한다.



- [ ] Releases events

  A release is created or updated.

SSL verification

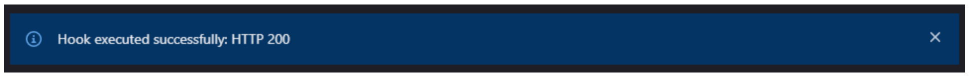
- [x] Enable SSL verification

Project Hooks (2)

http://i8e105.p.ssafy.io:8080/project/tonnybunny/

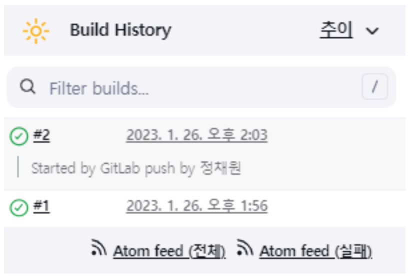
Push Events Merge Requests Events

SSL Verification: enabled



Hook executed successfully: HTTP 200

4. Jenkins에서 push event test가 잘 완료된 것을 알 수 있다.



#2 2023. 1. 26. 오후 2:03
Started by GitLab push by 정채원
#1 2023. 1. 26. 오후 1:56

2. DB, Redis

MYSQL, REDIS 컨테이너 실행

- `/home/ubuntu/tools/db/db-compose.yml` 파일 작성

```
version: "3.5"

services:
    gc_mysql:
        image: mysql:8.0.31
        container_name: gc_mysql
        volumes:
            - /home/ubuntu/resources/mysql/mysql:/var/lib/mysql
            - /home/ubuntu/resources/mysql/mysql_custom.cnf:/etc/mysql/conf.d/custom.cnf
        restart: always
        command:
            - --character-set-server=utf8mb4
            - --collation-server=utf8mb4_unicode_ci
        environment:
            - MYSQL_ROOT_PASSWORD=greencore11
            - MYSQL_DATABASE=greencore
        ports:
            - "3306:3306"
        expose:
            - "3306"
        networks:
            - gc_network

    gc_redis:
      image: redis:5.0.3
      container_name: gc_redis
      hostname: gc_redis
      labels:
        - "name=redis"
        - "mode=standalone"
      #network_mode: host
      ports:
          - 6379:6379
      volumes:
          - /etc/docker/redis-5.0.3/6001:/data
      command: redis-server --requirepass greencore11 --port 6379
      networks:
        - gc_network

networks:
    gc_network:
        external: true
```

- mysqld-custom.cnf 파일을 생성한다.
    - 문자 인코딩 방식 설정
    - 타임존을 UTC에서 KST(UTC+9)로 변경

```
[mysqld]
default_time_zone = '+09:00'
```

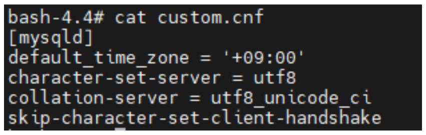
```
character-set-server = utf8
collation-server = utf8_unicode_ci
skip-character-set-client-handshake
```

- 도커 컴포즈 실행

```
# 명령어 실행
sudo docker-compose -f db-compose.yml up -d
```

- 확인하기

```
#1. mysql 컨테이너 접속
sudo docker exec -it [mysql 컨테이너명] /bin/bash

#2. 디렉토리 경로 이동
cd etc/mysql/conf.d/

#3. 파일 확인
cat custom.cnf
```



```
bash-4.4# cat custom.cnf
[mysqld]
default_time_zone = '+09:00'
character-set-server = utf8
collation-server = utf8_unicode_ci
skip-character-set-client-handshake
```

3. Next.js, Spring Boot

FE 환경변수 설정

- ec2에 환경변수 폴더 생성 및 이동

```
cd env/FE/
```

- `config/firebaseConfig.json` 작성

```
{
  "apiKey": "AIzaSyA2T1aflvVK4CteVOiuk4rmB4n5o_qQbcU",
  "authDomain": "ssafy-green-core.firebaseapp.com",
  "projectId": "ssafy-green-core",
  "storageBucket": "ssafy-green-core.appspot.com",
  "messagingSenderId": "110219727483",
  "appId": "1:110219727483:web:6fc3fd8e1aca52724e08de"
}
```

- `config/kakaoConfig.json` 작성

```
{
  "apiKey": "504ba604a4549c6738ce651ccebed6e6",
  "restApiKey": "9a0a32a7b66abbdd071c311257293643",
  "clientSecret": "9Bzzsf62IwaaU7gCCbjjBFhhaACT9agT",
  "redirectUri": "http://localhost:3000/user/kakao",
  "logOutRedirectUri": "http://localhost:3000/user/logout",
  "adminKey": "cebd388db480f9e6be287ef2e4f9df65"
}
```

- `.env` 작성

```
APP_SERVER_URL="https://j8e101.p.ssafy.io"
```

- jenkins 내부로 복사

```
# 명령어 실행
sudo docker cp ~/env/FE/* jenkins:/home/env/FE/*
sudo docker cp ~/env/FE/.env jenkins:/home/env/FE/.env
```

FE Dockerfile 작성

- FE Dockerfile 작성

```
# Specify the base image
FROM node:18.12.1

# Set the working directory inside the container
WORKDIR /FE/green-core

# Copy the package.json and package-lock.json files
COPY package*.json ./

# Install the dependencies
RUN npm install

# Copy the rest of the application code
COPY . .

# Build the application
RUN npm run build

# Expose the app on port 3000
EXPOSE 3000

# Start the app
CMD ["npm", "start"]
```

BE 환경변수 설정

- ec2에 환경변수 폴더 생성 및 이동

```
cd env/BE/
```

- `env.properties` 파일 생성

```
# Email
server.port=5000

auth.smtp.username=daffodil8520@naver.com
auth.smtp.password=%w4P8.F@u-z5ri
# JPA
jpa.show.sql=true
jpa.ddl.option=create
jpa.ddl.defer-datasource-initialization=true
# SQL
sql.init.mode=always
# JWT
jwt.accesskey=1234
jwt.refreshkey=1234
jwt.datakey=1234
jwt.expire.access=300000
jwt.expire.refresh=300000
jwt.secret=1234
# MySQL
db.mysql.driver=com.mysql.cj.jdbc.Driver
db.mysql.url=jdbc:mysql://j8e101.p.ssafy.io:3306/chicochico?autoReconnect=true&useUnicode=true&characterEncoding=utf-8
db.mysql.username=root
db.mysql.password=greencore11
```

```
# Log
log.hibernate.level=info
# file upload path
file.dir=/var/resources/static/
# Redis
spring.redis.host=gc_redis
spring.redis.port=6379
spring.redis.password=greencore11
# Firebase
firebase.config.path=/var/jenkins_home/workspace/greencore/BE/chicochico/src/main/resources/properties/firebaseAccountKey.json
firebase.config.rest-api-key=AIzaSyA2T1aflvVK4CteVOiuk4rmB4n5o_qQbcU
# Kakao
kakao.config.rest-api-key=9a0a32a7b66abbdd071c311257293643
kakao.config.client-secret=9Bzzsf62IwaaU7gCCbjjBFhhaACT9agT
# Recommender System
gorse.api_key=
gorse.endpoint=recommender_gorse_1:8088
```

- `firebaseAccountKey.json` 파일 생성

```
{
  "type": "service_account",
  "project_id": "ssafy-green-core",
  "private_key_id": "0b9a5e40cde721511311ad3190dabb36767b0d9a",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDHVIIiduyvVSYc\n7ojmk3eRrdFUtvhpS46SLk
  "client_email": "firebase-adminsdk-104bv@ssafy-green-core.iam.gserviceaccount.com",
  "client_id": "107469510145203969715",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-104bv%40ssafy-green-core.iam.gserviceacc
}
```

- jenkins 내부로 복사

```
# 명령어 실행
sudo docker cp ~/env/BE/* jenkins:/home/env/BE/
```

BE Dockerfile 작성

- BE Dockerfile 작성

```
FROM adoptopenjdk/openjdk11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
# COPY /var/conf/env.properties src/main/resources/properties/
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 5000
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=local", "/app.jar"]
```

도커 컴포즈 파일 작성

- 파일 작성 디렉토리 경로

```
cd ~/tools/greencore/
```

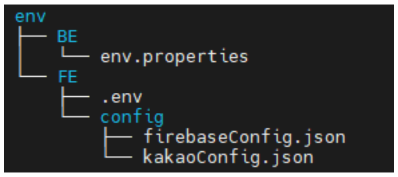
- `greencore-compose.yml` 작성

```
version: "3.5"

services:
  frontend:
    build:
      context: ../var/jenkins_home/workspace/greencore/FE/green-core/
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    depends_on:
      - backend
    networks:
      - gc_network
  backend:
    build:
      context: ../var/jenkins_home/workspace/greencore/BE/chicochico/
      dockerfile: Dockerfile
    volumes:
      - /home/ubuntu/resources/static:/var/resources/static
    ports:
      - "5000:5000"
    networks:
      - gc_network

networks:
        gc_network:
                external: true
```

- jenkins 내부로 복사

```
# 명령어 실행
sudo docker cp ~/tools/greencore/* jenkins:/home/
```

생성한 환경변수 파일 확인



```
env
├── BE
│   └── env.properties
└── FE
    ├── .env
    └── config
        ├── firebaseConfig.json
        └── kakaoConfig.json
```

▼ 💡 폴더 구조 생각

```
BE 환경변수 경로도 프로젝트에 환경변수가 위치한 경로랑 맞추면 더 좋았을거 같음 (FE처럼!)
ex) env/BE/chicochico/src/main/resources/properties/env.properties
```

▼ 💡 자주 쓰는 명령어 등록하는 방법

1. .bashrc 파일 열기

```
sudo vim .bashrc
```

2. 명령어 저장

```
alias cp-FE-env='sudo docker cp ~/env/FE/.env jenkins:/home/env/FE/.env'
alias cp-FE-env2='sudo docker cp ~/env/FE/* jenkins:/home/env/FE/'
alias cp-BE-env='sudo docker cp ~/env/BE/env.properties jenkins:/home/env/BE/'
alias cp-BE-env2='sudo docker cp ~/env/BE/firebaseAccountKey.json jenkins:/home/env/BE/'
alias vi-FE-env='sudo vi ~/env/FE/.env'
alias vi-BE-env='sudo vi ~/env/BE/env.properties'
```

4. 추천 시스템 오픈소스 배포 (Gorse)

- 추천 시스템은 Gorse라는 오픈소스를 가져와서 활용하였다. (사이트 주소 : https://gorse.io/)
- `config.toml` 과 `docker-compose.yml` 파일은 같은 디렉토리 공간에 위치해야 한다. gorse의 워크스페이스를 만들어준다.

```
mkdir recommender
cd recommender
```

- `config.toml` 파일 작성

```
[database]

# The database for caching, support Redis, MySQL, Postgres and MongoDB:
#   redis://<user>:<password>@<host>:<port>/<db_number>
#   rediss://<user>:<password>@<host>:<port>/<db_number>
#   redis+cluster://<user>:<password>@<host1>:<port1>,<host2>:<port2>,...,<hostN>:<portN>
#   postgres://bob:secret@1.2.3.4:5432/mydb?sslmode=verify-full
#   postgresql://bob:secret@1.2.3.4:5432/mydb?sslmode=verify-full
#   mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]
#   mongodb+srv://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]
cache_store = "redis://:greencore11@gc_redis:6379/0"

# The database for persist data, support MySQL, Postgres, ClickHouse and MongoDB:
#   mysql://[username[:password]@][protocol[(address)]]/dbname[?param1=value1&...&paramN=valueN]
#   postgres://bob:secret@1.2.3.4:5432/mydb?sslmode=verify-full
#   postgresql://bob:secret@1.2.3.4:5432/mydb?sslmode=verify-full
#   clickhouse://user:password@host[:port]/database?param1=value1&...&paramN=valueN
#   chhttp://user:password@host[:port]/database?param1=value1&...&paramN=valueN
#   chhttps://user:password@host[:port]/database?param1=value1&...&paramN=valueN
#   mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]
#   mongodb+srv://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]
data_store = "mysql://root:greencore11@tcp(gc_mysql:3306)/gorse"

# The naming prefix for tables (collections, keys) in databases. The default value is empty.
table_prefix = ""

# The naming prefix for tables (collections, keys) in cache storage databases. The default value is `table_prefix`.
cache_table_prefix = ""

# The naming prefix for tables (collections, keys) in data storage databases. The default value is `table_prefix`.
data_table_prefix = ""

[master]

# GRPC port of the master node. The default value is 8086.
port = 8086

# gRPC host of the master node. The default values is "0.0.0.0".
host = "0.0.0.0"

# HTTP port of the master node. The default values is 8088.
http_port = 8088

# HTTP host of the master node. The default values is "0.0.0.0".
http_host = "0.0.0.0"

# AllowedDomains is a list of allowed values for Http Origin.
# The list may contain the special wildcard string ".*" ; all is allowed
# If empty all are allowed.
http_cors_domains = []

# AllowedMethods is either empty or has a list of http methods names. Checking is case-insensitive.
http_cors_methods = []

# Number of working jobs in the master node. The default value is 1.
n_jobs = 1

# Meta information timeout. The default value is 10s.
meta_timeout = "10s"

# Username for the master node dashboard.
dashboard_user_name = "gc"

# Password for the master node dashboard.
dashboard_password = "greencore11"
```

```
# Secret key for admin APIs (SSL required).
admin_api_key = ""

[server]

# Default number of returned items. The default value is 10.
default_n = 10

# Secret key for RESTful APIs (SSL required).
api_key = ""

# Clock error in the cluster. The default value is 5s.
clock_error = "5s"

# Insert new users while inserting feedback. The default value is true.
auto_insert_user = true

# Insert new items while inserting feedback. The default value is true.
auto_insert_item = true

# Server-side cache expire time. The default value is 10s.
cache_expire = "10s"

[recommend]

# The cache size for recommended/popular/latest items. The default value is 10.
cache_size = 100

# Recommended cache expire time. The default value is 72h.
cache_expire = "72h"

[recommend.data_source]

# The feedback types for positive events.
positive_feedback_types = ["bookmark","like"]

# The feedback types for read events.
read_feedback_types = ["read"]

# The time-to-live (days) of positive feedback, 0 means disabled. The default value is 0.
positive_feedback_ttl = 0

# The time-to-live (days) of items, 0 means disabled. The default value is 0.
item_ttl = 0

[recommend.popular]

# The time window of popular items. The default values is 4320h.
popular_window = "720h"

[recommend.user_neighbors]

# The type of neighbors for users. There are three types:
#   similar: Neighbors are found by number of common labels.
#   related: Neighbors are found by number of common liked items.
#   auto: If a user have labels, neighbors are found by number of common labels.
#         If this user have no labels, neighbors are found by number of common liked items.
# The default value is "auto".
neighbor_type = "similar"

# Enable approximate user neighbor searching using vector index. The default value is true.
enable_index = true

# Minimal recall for approximate user neighbor searching. The default value is 0.8.
index_recall = 0.8

# Maximal number of fit epochs for approximate user neighbor searching vector index. The default value is 3.
index_fit_epoch = 3

[recommend.item_neighbors]

# The type of neighbors for items. There are three types:
#   similar: Neighbors are found by number of common labels.
#   related: Neighbors are found by number of common users.
#   auto: If a item have labels, neighbors are found by number of common labels.
#         If this item have no labels, neighbors are found by number of common users.
# The default value is "auto".
neighbor_type = "similar"

# Enable approximate item neighbor searching using vector index. The default value is true.
```

```
enable_index = true

# Minimal recall for approximate item neighbor searching. The default value is 0.8.
index_recall = 0.8

# Maximal number of fit epochs for approximate item neighbor searching vector index. The default value is 3.
index_fit_epoch = 3

[recommend.collaborative]

# Enable approximate collaborative filtering recommend using vector index. The default value is true.
enable_index = true

# Minimal recall for approximate collaborative filtering recommend. The default value is 0.9.
index_recall = 0.9

# Maximal number of fit epochs for approximate collaborative filtering recommend vector index. The default value is 3.
index_fit_epoch = 3

# The time period for model fitting. The default value is "60m".
model_fit_period = "60m"

# The time period for model searching. The default value is "360m".
model_search_period = "360m"

# The number of epochs for model searching. The default value is 100.
model_search_epoch = 100

# The number of trials for model searching. The default value is 10.
model_search_trials = 10

# Enable searching models of different sizes, which consume more memory. The default value is false.
enable_model_size_search = false

[recommend.replacement]

# Replace historical items back to recommendations. The default value is false.
enable_replacement = false

# Decay the weights of replaced items from positive feedbacks. The default value is 0.8.
positive_replacement_decay = 0.8

# Decay the weights of replaced items from read feedbacks. The default value is 0.6.
read_replacement_decay = 0.6

[recommend.offline]

# The time period to check recommendation for users. The default values is 1m.
check_recommend_period = "1m"

# The time period to refresh recommendation for inactive users. The default values is 120h.
refresh_recommend_period = "24h"

# Enable latest recommendation during offline recommendation. The default value is false.
enable_latest_recommend = true

# Enable popular recommendation during offline recommendation. The default value is false.
enable_popular_recommend = false

# Enable user-based similarity recommendation during offline recommendation. The default value is false.
enable_user_based_recommend = true

# Enable item-based similarity recommendation during offline recommendation. The default value is false.
enable_item_based_recommend = false

# Enable collaborative filtering recommendation during offline recommendation. The default value is true.
enable_collaborative_recommend = true

# Enable click-though rate prediction during offline recommendation. Otherwise, results from multi-way recommendation
# would be merged randomly. The default value is false.
enable_click_through_prediction = true

# The explore recommendation method is used to inject popular items or latest items into recommended result:
#   popular: Recommend popular items to cold-start users.
#   latest: Recommend latest items to cold-start users.
# The default values is { popular = 0.0, latest = 0.0 }.
explore_recommend = { popular = 0.1, latest = 0.2 }

[recommend.online]

# The fallback recommendation method is used when cached recommendation drained out:
```

```
#   item_based: Recommend similar items to cold-start users.
#   popular: Recommend popular items to cold-start users.
#   latest: Recommend latest items to cold-start users.
# Recommenders are used in order. The default values is ["latest"].
fallback_recommend = ["item_based", "latest"]

# The number of feedback used in fallback item-based similar recommendation. The default values is 10.
num_feedback_fallback_item_based = 10

[tracing]

# Enable tracing for REST APIs. The default value is false.
enable_tracing = false

# The type of tracing exporters should be one of "jaeger", "zipkin", "otlp" and "otlphttp". The default value is "jaeger".
exporter = "jaeger"

# The endpoint of tracing collector.
collector_endpoint = "http://localhost:14268/api/traces"

# The type of tracing sampler should be one of "always", "never" and "ratio". The default value is "always".
sampler = "always"

# The ratio of ratio based sampler. The default value is 1.
ratio = 1
```

- `docker-compose.yml` 파일 작성

```
version: "3.5"
services:
  gorse:
    image: zhenghaoz/gorse-in-one
    restart: unless-stopped
    ports:
      - 8086:8086   # gRPC port
      - 8088:8088   # HTTP port
    environment:
      # Use Redis as cache storage backend.
      GORSE_CACHE_STORE: redis://:greencore11@gc_redis:6379
      # Use MySQL as data storage backend.
      GORSE_DATA_STORE: mysql://root:greencore11@tcp(gc_mysql:3306)/gorse?parseTime=true
    command: >
      -c /etc/gorse/config.toml
      --log-path /var/log/gorse/master.log
      --cache-path /var/lib/gorse/master_cache.data
    volumes:
      # Mount the configuration file.
      - ./config.toml:/etc/gorse/config.toml
    networks:
      - gc_network

networks:
    gc_network:
            external: true
```

- `sudo docker-compose up -d` 로 실행한다

젠킨스 Shell Script 작성

- Jenkins 프로젝트 설정 > Build Steps

```
flag=''
WEBHOOK_URL="https://discord.com/api/webhooks/1091273725346861116/IebWU71OJs-7Lan6fLCrro2O9tO5OAbI34Ss5DgaXaJKdNHXvtCIR-RBhhDzX0w3eHKV"

# fe 환경변수
cp /home/env/FE/.env ./FE/green-core/
cp -r /home/env/FE/* ./FE/green-core/

# be 환경변수
cp /home/env/BE/* ./BE/chicochico/src/main/resources/properties/

# 컨테이너 초기화
if docker build -t home_backend --build-arg SPRING_PROFILES_ACTIVE=prod -f ./BE/chicochico/Dockerfile ./BE/chicochico; then
```

```
  if (docker ps | grep "home_backend_1"); then docker stop home_backend_1; fi
  if (docker ps -a | grep 'home_backend_1'); then docker rm home_backend_1; fi

else
  echo "backend 빌드 실패"
  flag="backend"
fi

if docker build -t home_frontend -f ./FE/green-core/Dockerfile ./FE/green-core; then
  if (docker ps | grep "home_frontend_1"); then docker stop home_frontend_1; fi
  if (docker ps -a | grep 'home_frontend_1'); then docker rm home_frontend_1; fi
else
  echo "frontend 빌드 실패"
  flag="frontend"
fi

if [ "$flag" = "backend" ]; then
    curl -H "Content-Type: application/json" -d '{"content": "백엔드 이미지 빌드 FAIL"}' $WEBHOOK_URL
elif [ "$flag" = "frontend" ]; then
  curl -H "Content-Type: application/json" -d '{"content": "프론트엔드 이미지 빌드 FAIL"}' $WEBHOOK_URL
# else
    # curl -H "Content-Type: application/json" -d '{"content": "이미지 빌드 SUCCESS"}' $WEBHOOK_URL
fi

docker-compose -f /home/greencore-compose.yml up -d
```