포팅 메뉴얼

1️⃣ 프로젝트 기술 스택

🖥️ Frontend

- lang:
  HTML5, CSS3, TypeScript
  `5.3.3` , Node.js `20.11.1`
- framework:
  React
  `18.2.0` , Next.js `14.1`
- library :
  style : tailwind css
  `3.4.1`
  HTTP 통신: axios
  formatter : eslint + prettier
  router : react-router-dom
  `버전작성`
- state management tool :
  React-query, Zustand
  `4.5.1`

🗂️ Database

- MySQL `8.0.34`
- MongoDB `7.0.6`
- Elasticsearch `8.12.2`
- redis `7.2.4`

👨‍👩‍👧 협업 툴

- Gitlab
- Jira
- Notion
- Mattermost

🖥️ Backend

- Java open-JDK zulu `17.0.9`
- SpringBoot `3.2.3`
- Gradle `8.5`
- Lombok `1.18.16`
- Hibernate `3.2.1`
- Swagger `4.18.2`
- Spring Security `6.2.2`
- Python `3.12.2`
- Jupyter notebook `버전작성`
- Scala `2.12.16` (+ `openJDK-8u342`)
- sbt `1.7.2`
- Play Framework `2.8.21`
- Spark `3.0.2`
- mongo-spark-connector `3.0.2`

CI/CD

- docker `25.0.4`
- docker-compose `2.21.0`
- jenkins : `2.440.1`

🎨 UI/UX

- figma

⚒️ IDE

- IntelliJ 2023.3.4
- VSCode 1.85.2
- MySQL WorkBench 8.0.36

2️⃣ EC2 서버 환경 설정

(1) 우분투 서버 한국 표준시로 변경 (UTC+9)

```
sudo timedatectl set-timezone Asia/Seoul
```

(2) 카카오 미러 서버 활용

- 기본 서버가 *.ubuntu.com 이라는 해외 서버이기 때문에, 패키지 갱신 속도가 비교적 빠른 국내 미러 서버를 활용하는 것이 효율적임. 가장 많이 이용하는 성능 좋은 미러 서버는 카카오 서버

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/g'/etc/apt/sources.l
ist
```

- 미러 서버 업데이트 후

```
sudo apt-get -y update && sudo apt-get -y upgrade
```

(3) SWAP 영역 할당

- 스왑 영역 할당 (ex : 4GB)

```
sudo fallocate -l 4G /swapfile
```

- swapfile 권한 수정

```
sudo chmod 600 /swapfile
```

- sawpfile 생성

```
sudo mkswap /swapfile
```

- swapfile 활성화

```
sudo swapon /swapfile
```

- 시스템이 재부팅해도 swap 유지 설정

```
sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

- swap 영역이 할당 확인

```
free -h
```

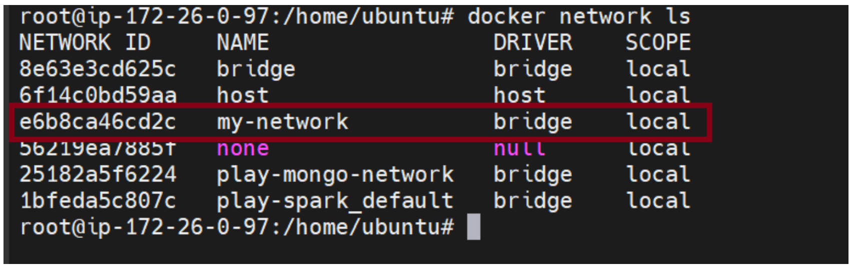
▼ 3️⃣ Nginx 리버스 프록시 설정

1. docker 가상 네트워크 생성

📂 /home/ubuntu (위치 무관)

📍네트워크 이름 example : my-network

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



```
root@ip-172-26-0-97:/home/ubuntu# docker network ls
NETWORK ID     NAME                 DRIVER    SCOPE
8e63e3cd625c   bridge               bridge    local
6f14c0bd59aa   host                 host      local
e6b8ca46cd2c   my-network           bridge    local
56219ea7885f   none                 null      local
25182a5f6224   play-mongo-network   bridge    local
1bfeda5c807c   play-spark_default   bridge    local
root@ip-172-26-0-97:/home/ubuntu# 
```

2. Jenkins / Nginx 컨테이너 설치

(nginx 리버스 프록시를 통해 jenkins로 접속하기 위하여 jenkins 함께 빌드)

📂 /home/ubuntu/

Jenkins 도커 파일

- Jenkins 컨테이너 안에 docker와 docker-compose 설치

📄 Dockerfile

```
FROM jenkins/jenkins:lts
USER root

RUN 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 > /t
mp/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

RUN groupadd -f docker
RUN usermod -aG docker jenkins

# 도커 컴포즈 설치
RUN curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(u
name -s)-$(uname -m)" -o /usr/local/bin/docker-compose && \
    chmod +x /usr/local/bin/docker-compose
```

📄 docker-compose.yml

```
version: '3'
services:
  jenkins:
    build:
      context: .
      dockerfile: Dockerfile
    # image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /home/ubuntu/jenkins:/var/jenkins_home #host의 jenkins_home을 가져와서 ubuntu의 jenki
ns로 가져와서 추가
      - /home/ubuntu/.ssh:/var/jenkins_home/.ssh #젠킨스의 ssh의 명령어를 걸 때 호스트의 .ssh 인증
서를 공용해서 씀
      - /var/run/docker.sock:/var/run/docker.sock #host의 docker engine 사용을 위해 추가
    networks:
      - my-network
  nginx:
    image: nginx
    container_name: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - /home/ubuntu/pickitup/:/etc/nginx/pickitup/
      - /home/ubuntu/nginx/conf.d:/etc/nginx/conf.d # conf.d 를 만듦 (nginx를 통해서 jenkins)
      - /home/ubuntu/nginx/cert:/etc/cert # 인증서 파일을 공유시키기 위해서
      - /etc/letsencrypt:/etc/cert2
    restart: always # 꺼져도 다시 실행
    depends_on:
      - jenkins # jenkins가 실행되고 나서 nginx를 실행하겠다는 의미
    networks:
      - my-network # 네트워크는 my-network(가상네트워크 그룹을 만들어서 nginx랑 jenkins가 my-networ
k 네트워크에서 사용한다.)

networks:
  my-network:
    external: true
```

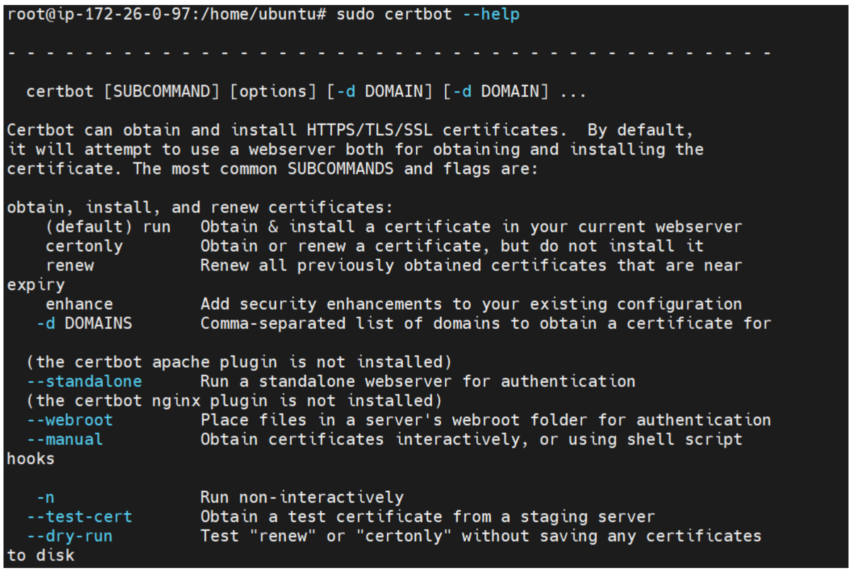
3. SSL 와일드 카드 인증서 발급

(1) Let's encrypt 설치

```
sudo apt update
sudo apt-get install letsencrypt -y
```

(2) 설치 확인

```
sudo certbot --help
```



```
root@ip-172-26-0-97:/home/ubuntu# sudo certbot --help

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

  certbot [SUBCOMMAND] [options] [-d DOMAIN] [-d DOMAIN] ...

Certbot can obtain and install HTTPS/TLS/SSL certificates.  By default,
it will attempt to use a webserver both for obtaining and installing the
certificate. The most common SUBCOMMANDS and flags are:

obtain, install, and renew certificates:
    (default) run   Obtain & install a certificate in your current webserver
    certonly        Obtain or renew a certificate, but do not install it
    renew           Renew all previously obtained certificates that are near
expiry
    enhance         Add security enhancements to your existing configuration
   -d DOMAINS       Comma-separated list of domains to obtain a certificate for

  (the certbot apache plugin is not installed)
  --standalone      Run a standalone webserver for authentication
  (the certbot nginx plugin is not installed)
  --webroot         Place files in a server's webroot folder for authentication
  --manual          Obtain certificates interactively, or using shell script
hooks

   -n               Run non-interactively
  --test-cert       Obtain a test certificate from a staging server
  --dry-run         Test "renew" or "certonly" without saving any certificates
to disk
```

(3) SSL 인증서 발급

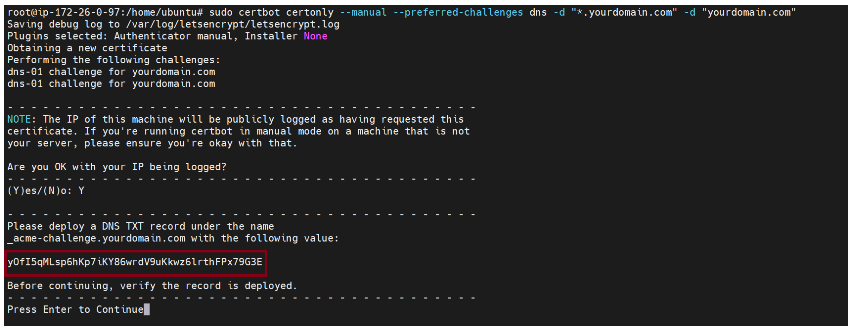
DNS의 TXT 레코드를 이용하여 인증서를 발급 받을 수 있다. 서브도메인의 _acme-challenge에 해당하는 도메인을 cerbot이 생성한 난수로 등록해주면 된다.

여기서 도메인은 각자 구입한 도메인을 적어주면된다.

`-d "*.pickitup.online" -d "pickitup.online"` 이렇게 인증서를 발급받으면 구입한 도메인 앞에 모든 host 이름에 대해서 인증서를 공유할 수 있다.

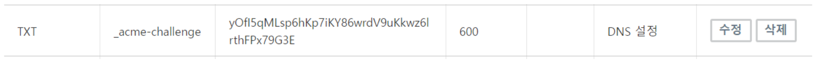
```
sudo certbot certonly --manual --preferred-challenges dns -d "*.yourdomain.com" -d "yourdo
main.com"
```

위의 명령어를 치고 Enter를 누르면 _acme-challenge 하위 도메인에 등록해야할 난수를 던져준다. 해당 난수를 DNS 레코드에 등록해주면된다.



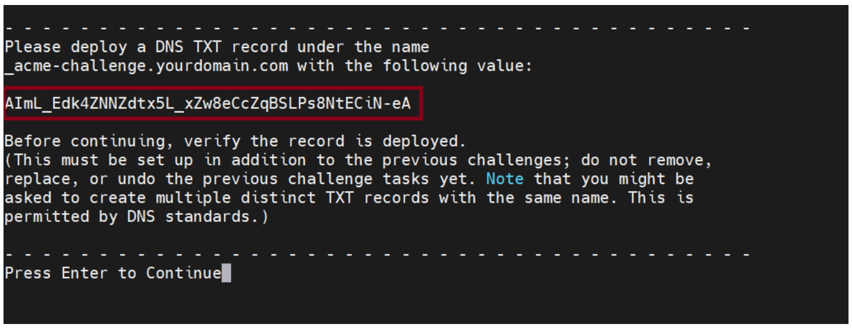
```
root@ip-172-26-0-97:/home/ubuntu# sudo certbot certonly --manual --preferred-challenges dns -d "*.yourdomain.com" -d "yourdomain.com"
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator manual, Installer None
Obtaining a new certificate
Performing the following challenges:
dns-01 challenge for yourdomain.com
dns-01 challenge for yourdomain.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
NOTE: The IP of this machine will be publicly logged as having requested this
certificate. If you're running certbot in manual mode on a machine that is not
your server, please ensure you're okay with that.

Are you OK with your IP being logged?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: Y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name
_acme-challenge.yourdomain.com with the following value:

yOfI5qMLsp6hKp7iKY86wrdV9uKkwz6lrthFPx79G3E

Before continuing, verify the record is deployed.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue
```

📍가비아 DNS 관리



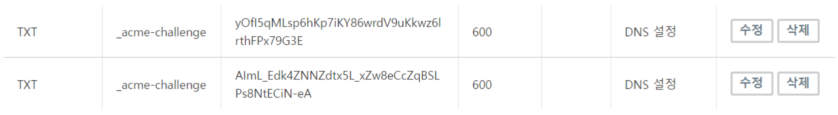
TXT	_acme-challenge	yOfI5qMLsp6hKp7iKY86wrdV9uKkwz6lrthFPx79G3E	600		DNS 설정	수정 삭제

DNS 설정 정보를 아직 저장하지말고 터미널에서 Enter를 한번 더 누르면 난수를 하나 더 던져준다. 해당 난수도 추가로 DNS 레코드에 등록해줘야 한다.



```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name
_acme-challenge.yourdomain.com with the following value:

AImL_Edk4ZNNZdtx5L_xZw8eCcZqBSLPs8NtECiN-eA

Before continuing, verify the record is deployed.
(This must be set up in addition to the previous challenges; do not remove,
replace, or undo the previous challenge tasks yet. Note that you might be
asked to create multiple distinct TXT records with the same name. This is
permitted by DNS standards.)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue
```

📍가비아 DNS 관리



TXT	_acme-challenge	yOfI5qMLsp6hKp7iKY86wrdV9uKkwz6lrthFPx79G3E	600		DNS 설정	수정 삭제
TXT	_acme-challenge	AImL_Edk4ZNNZdtx5L_xZw8eCcZqBSLPs8NtECiN-eA	600		DNS 설정	수정 삭제

이렇게 DNS 레코드를 저장해준 다음에 EC2 서버 터미널에서 Enter를 누르면 SSL 인증서가 성공적으로 발급된다.

(4) 인증서 발급 확인

아래 명령어를 통해 아까 등록한 도메인의 폴더가 생성되어있는 지를 확인하면 된다.

```
sudo ls /etc/letencrypt/live
```

4. Nginx conf 파일 설정

위에서 nginx 컨테이너를 실행시킬 때 아래와 같은 옵션을 통해 호스트의 conf.d 파일을nginx 컨테이너의 nginx/conf.d에 마운트 시켰었기 때문에 호스트의 `/home/ubuntu/ngnix/conf.d` 파일에 nginx 설정 파일을 작성해주면 된다.

(기본적으로 제공받은 도메인을 젠킨스 도메인으로 이용하도록 설정해주었다)

📂 /home/ubuntu/nginx/

📄 conf.d

```
## 젠킨스 서버
server {
    listen 80;
    server_name j10a406.p.ssafy.io;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name j10a406.p.ssafy.io;

    ssl_certificate /etc/cert/cert.pem;  # SSL 인증서 파일
    ssl_certificate_key /etc/cert/privkey.pem;  # SSL 키 파일
    ssl_trusted_certificate /etc/cert/chain.pem;

    location / {

            proxy_pass http://jenkins:8080;
            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;
            proxy_set_header X-Forwarded-Proto $scheme;

            # Required for new HTTP-based CLI
            proxy_http_version 1.1;
            proxy_request_buffering off;
            proxy_buffering off; # Required for HTTP-based CLI to work over SSL
            add_header 'X-SSH-Endpoint' 'j10a406.p.ssafy.io/' always;
    }

}
```

▼ 4️⃣ DB

📁 **database 디렉토리 현황**

```
📁elasticsearch
    📁data
    📁text
        📄synonyms.txt
    📄docker-compose.yml

📁mongodb
    📁data
    📄.env
    📄docker-compose.yml

📁mongodb-recommender

📁mysql
    📁master
        📄Dockerfile
        📄my.cnf
    📁slave
        📄Dockerfile
        📄my.cnf
    📄docker-compose.yml

📁redis
    📁redis_vol
    📄docker-compose.yml
```

▼ (1) Elasticsearch

kibana에서 elasticsearch 설정

1. **analysis 설정**

- 'synonym_filter' : synonyms.txt 파일에 정의된 동의어를 활용하여 텍스트 처리
- 'ngram_filter' : N-그램으로 분할하여 분석. 여기서는 최소 2글자부터 최대 3글자까지의 N-그램을 사용합니다(인덱스 생성 이후 5글자까지로 변경)

1. **mappings 설정**

- 'qualification_requirements', 'preferred_requirements' 필드에 동의어를 처리하기 위한 설정 추가(ex. 자바스크립트, JS, Javascript)
- 'title', 'company' 필드에 ngram 필터를 적용하여 일부 단어로도 검색 가능하게 설정 추가
- 'due_date' 필드가 날짜 형식임을 명시

📄recruit index 설정

```
PUT /searchrecruit
{
  "settings": {
    "analysis": {
      "filter": {
        "synonym_filter": {
          "type": "synonym",
          "synonyms_path": "text/synonyms.txt"
        },
        "ngram_filter": {
          "type": "ngram",
          "min_gram": 2,
          "max_gram": 3
        }
      },
      "analyzer": {
        "my_synonym_analyzer": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": ["lowercase", "synonym_filter"]
        },
        "my_ngram_analyzer": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": ["lowercase", "ngram_filter"]
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "qualification_requirements": {
        "type": "text",
        "analyzer": "my_synonym_analyzer"
      },
            "preferred_requirements": {
        "type": "text",
        "analyzer": "my_synonym_analyzer"
      },
            "title": {
        "type": "text",
        "analyzer": "my_ngram_analyzer"
      },
      "company": {
        "type": "text",
        "analyzer": "my_ngram_analyzer"
      },
            "due_date": {
        "type": "date",
        "format": "yyyy-MM-dd"
      }
    }
  }
}
```

recruit ngram 필터 설정 추가

```
PUT /searchrecruit/_settings
{
  "index.max_ngram_diff": 3
}
```

company index 설정

```
PUT /searchcompany
{
  "mappings": {
    "properties": {
      "name": { "type": "text" },
      "address": { "type": "text" },
      "salary": { "type": "text" }
    }
  }
}
```

id 필드 접근 허용

```
PUT /_cluster/settings
{
  "persistent": {
    "indices.id_field_data.enabled": true
  }
}
```

📁**database/elasticsearch/**

📄docker-compose.yml

```
version: '3'

services:

    elasticsearch:
      image: docker.elastic.co/elasticsearch/elasticsearch:8.12.2
      # 컨테이너 이름
      container_name: elasticsearch
      # 환경 변수
      environment:
        - node.name=es-node
        - cluster.name=search-cluster
        - discovery.type=single-node
        - xpack.security.enabled=false
        - xpack.security.http.ssl.enabled=false
        - xpack.security.transport.ssl.enabled=false
      # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
      ports:
        - 9200
      volumes:
        - ./data:/usr/share/elasticsearch/data
        - ./text:/usr/share/elasticsearch/config/text
      deploy:
        resources:
            limits:
              memory: 4GB
      networks:
        - my-network


    kibana:
      image: docker.elastic.co/kibana/kibana:8.5.3
      container_name: kibana
      environment:
        SERVER_NAME: kibana
        ELASTICSEARCH_HOSTS: http://elasticsearch:9200
      ports:
        - 5601
      # Elasticsearch 가 실행 된 후에 kibana 실행
      depends_on:
        - elasticsearch
      # 네트워크 설정
      networks:
        - my-network

networks:
  my-network:
    external: true
```

📁**database/elasticsearch/text**

synonyms.txt

```
Agit, Agit을, Agit를, Agit이, Agit가, Agit와, Agit과, Agit은, Agit는, Agit등, Agit에, Agit이나,
Airflow, Airflow을, Airflow를, Airflow이, Airflow가, Airflow와, Airflow과, Airflow은, Airflow
Alamofire, Alamofire을, Alamofire를, Alamofire이, Alamofire가, Alamofire와, Alamofire과, Alam
Android, Android을, Android를, Android이, Android가, Android와, Android과, Android은, Android
Angular, Angular을, Angular를, Angular이, Angular가, Angular와, Angular과, Angular은, Angular
Ansible, Ansible을, Ansible를, Ansible이, Ansible가, Ansible와, Ansible과, Ansible은, Ansible
Apache, Apache을, Apache를, Apache이, Apache가, Apache와, Apache과, Apache은, Apache는, Apache
Apollo, Apollo을, Apollo를, Apollo이, Apollo가, Apollo와, Apollo과, Apollo은, Apollo는, Apollo
Appium, Appium을, Appium를, Appium이, Appium가, Appium와, Appium과, Appium은, Appium는, Appium
ArangoDB, ArangoDB을, ArangoDB를, ArangoDB이, ArangoDB가, ArangoDB와, ArangoDB과, ArangoDB은,
Arcus, Arcus을, Arcus를, Arcus이, Arcus가, Arcus와, Arcus과, Arcus은, Arcus는, Arcus등, Arcus
Argo CD, Argo CD을, Argo CD를, Argo CD이, Argo CD가, Argo CD와, Argo CD과, Argo CD은, Argo CD
Armeria, Armeria을, Armeria를, Armeria이, Armeria가, Armeria와, Armeria과, Armeria은, Armeria
Asana, Asana을, Asana를, Asana이, Asana가, Asana와, Asana과, Asana은, Asana는, Asana등, Asana
ASP, ASP을, ASP를, ASP이, ASP가, ASP와, ASP과, ASP은, ASP는, ASP등, ASP에, ASP이나, ASP나
ASPNET, ASPNET을, ASPNET를, ASPNET이, ASPNET가, ASPNET와, ASPNET과, ASPNET은, ASPNET는, ASPNET
AWS Athena, AWS Athena을, AWS Athena를, AWS Athena이, AWS Athena가, AWS Athena와, AWS Athena
AWS AuroraDB, AWS AuroraDB을, AWS AuroraDB를, AWS AuroraDB이, AWS AuroraDB가, AWS AuroraDB와
AWS CodeBuild, AWS CodeBuild을, AWS CodeBuild를, AWS CodeBuild이, AWS CodeBuild가, AWS CodeB
AWS CodeDeploy, AWS CodeDeploy을, AWS CodeDeploy를, AWS CodeDeploy이, AWS CodeDeploy가, AWS
AWS CodePipeline, AWS CodePipeline을, AWS CodePipeline를, AWS CodePipeline이, AWS CodePipeli
AWS DocumentDB, AWS DocumentDB을, AWS DocumentDB를, AWS DocumentDB이, AWS DocumentDB가, AWS
AWS DynamoDB, AWS DynamoDB을, AWS DynamoDB를, AWS DynamoDB이, AWS DynamoDB가, AWS DynamoDB와
AWS Kinesis, AWS Kinesis을, AWS Kinesis를, AWS Kinesis이, AWS Kinesis가, AWS Kinesis와, AWS
AWS MariaDB, AWS MariaDB을, AWS MariaDB를, AWS MariaDB이, AWS MariaDB가, AWS MariaDB와, AWS
AWS Redshift, AWS Redshift을, AWS Redshift를, AWS Redshift이, AWS Redshift가, AWS Redshift와
AWS SES, AWS SES을, AWS SES를, AWS SES이, AWS SES가, AWS SES와, AWS SES과, AWS SES은, AWS SES
AWS SNS, AWS SNS을, AWS SNS를, AWS SNS이, AWS SNS가, AWS SNS와, AWS SNS과, AWS SNS은, AWS SNS
AWS SQS, AWS SQS을, AWS SQS를, AWS SQS이, AWS SQS가, AWS SQS와, AWS SQS과, AWS SQS은, AWS SQS
Azure DevOps, Azure DevOps을, Azure DevOps를, Azure DevOps이, Azure DevOps가, Azure DevOps와
Babel, Babel을, Babel를, Babel이, Babel가, Babel와, Babel과, Babel은, Babel는, Babel등, Babel
BackboneJS, BackboneJS을, BackboneJS를, BackboneJS이, BackboneJS가, BackboneJS와, BackboneJS
Backend, Backend을, Backend를, Backend이, Backend가, Backend와, Backend과, Backend은, Backend
Bazel, Bazel을, Bazel를, Bazel이, Bazel가, Bazel와, Bazel과, Bazel은, Bazel는, Bazel등, Bazel
Bitbucket, Bitbucket을, Bitbucket를, Bitbucket이, Bitbucket가, Bitbucket와, Bitbucket과, Bitb
Bitrise, Bitrise을, Bitrise를, Bitrise이, Bitrise가, Bitrise와, Bitrise과, Bitrise은, Bitrise
Bootstrap, Bootstrap을, Bootstrap를, Bootstrap이, Bootstrap가, Bootstrap와, Bootstrap과, Boot
Capistrano, Capistrano을, Capistrano를, Capistrano이, Capistrano가, Capistrano와, Capistrano
CassandraDB, CassandraDB을, CassandraDB를, CassandraDB이, CassandraDB가, CassandraDB와, Cass
Celery, Celery을, Celery를, Celery이, Celery가, Celery와, Celery과, Celery은, Celery는, Celery
Central Dogma, Central Dogma을, Central Dogma를, Central Dogma이, Central Dogma가, Central D
Ceph, Ceph을, Ceph를, Ceph이, Ceph가, Ceph와, Ceph과, Ceph은, Ceph는, Ceph등, Ceph에, Ceph이나,
CI/CD, CI/CD을, CI/CD를, CI/CD이, CI/CD가, CI/CD와, CI/CD과, CI/CD은, CI/CD는, CI/CD등, CI/CD
Circle CI, Circle CI을, Circle CI를, Circle CI이, Circle CI가, Circle CI와, Circle CI과, Circ
Clean-Architecture, Clean-Architecture을, Clean-Architecture를, Clean-Architecture이, Clean-
Clickhouse, Clickhouse을, Clickhouse를, Clickhouse이, Clickhouse가, Clickhouse와, Clickhouse
Clojure, Clojure을, Clojure를, Clojure이, Clojure가, Clojure와, Clojure과, Clojure은, Clojure
CockroachDB, CockroachDB을, CockroachDB를, CockroachDB이, CockroachDB가, CockroachDB와, Cock
CodeIgniter, CodeIgniter을, CodeIgniter를, CodeIgniter이, CodeIgniter가, CodeIgniter와, Code
Confluence, Confluence을, Confluence를, Confluence이, Confluence가, Confluence와, Confluence
CORS, CORS을, CORS를, CORS이, CORS가, CORS와, CORS과, CORS은, CORS는, CORS등, CORS에, CORS이나,
Couchbase, Couchbase을, Couchbase를, Couchbase이, Couchbase가, Couchbase와, Couchbase과, Couc
C++, C++을, C++를, C++이, C++가, C++와, C++과, C++은, C++는, C++등, C++에, C++이나, C++나
C Sharp, C Sharp을, C Sharp를, C Sharp이, C Sharp가, C Sharp와, C Sharp과, C Sharp은, C Sharp
Cubrid, Cubrid을, Cubrid를, Cubrid이, Cubrid가, Cubrid와, Cubrid과, Cubrid은, Cubrid는, Cubri
Cucumber, Cucumber을, Cucumber를, Cucumber이, Cucumber가, Cucumber와, Cucumber과, Cucumber은,
Cypress, Cypress을, Cypress를, Cypress이, Cypress가, Cypress와, Cypress과, Cypress은, Cypress
```

```
Dagger, Dagger을, Dagger를, Dagger이, Dagger가, Dagger와, Dagger과, Dagger은, Dagger는, Dagger
Dart, Dart을, Dart를, Dart이, Dart가, Dart와, Dart과, Dart은, Dart는, Dart등, Dart에, Dart이나,
Database, Database을, Database를, Database이, Database가, Database와, Database과, Database은,
Discord, Discord을, Discord를, Discord이, Discord가, Discord와, Discord과, Discord은, Discord
Django, Django을, Django를, Django이, Django가, Django와, Django과, Django은, Django는, Djang
Docker, Docker을, Docker를, Docker이, Docker가, Docker와, Docker과, Docker은, Docker는, Docker
Docusaurus, Docusaurus을, Docusaurus를, Docusaurus이, Docusaurus가, Docusaurus와, Docusaurus
Dooray, Dooray을, Dooray를, Dooray이, Dooray가, Dooray와, Dooray과, Dooray은, Dooray는, Dooray
Drone, Drone을, Drone를, Drone이, Drone가, Drone와, Drone과, Drone은, Drone는, Drone등, Drone
Dropwizard, Dropwizard을, Dropwizard를, Dropwizard이, Dropwizard가, Dropwizard와, Dropwizard
Druid, Druid을, Druid를, Druid이, Druid가, Druid와, Druid과, Druid은, Druid는, Druid등, Druid
Echo, Echo을, Echo를, Echo이, Echo가, Echo와, Echo과, Echo은, Echo는, Echo등, Echo에, Echo이나,
ElasticSearch, ElasticSearch을, ElasticSearch를, ElasticSearch이, ElasticSearch가, ElasticSe
Electron, Electron을, Electron를, Electron이, Electron가, Electron와, Electron과, Electron은,
Elixir, Elixir을, Elixir를, Elixir이, Elixir가, Elixir와, Elixir과, Elixir은, Elixir는, Elixir
ELK, ELK을, ELK를, ELK이, ELK가, ELK와, ELK과, ELK은, ELK는, ELK등, ELK에, ELK이나, ELK나
EmberJS, EmberJS을, EmberJS를, EmberJS이, EmberJS가, EmberJS와, EmberJS과, EmberJS은, EmberJS
Emotion, Emotion을, Emotion를, Emotion이, Emotion가, Emotion와, Emotion과, Emotion은, Emotion
Envoy, Envoy을, Envoy를, Envoy이, Envoy가, Envoy와, Envoy과, Envoy은, Envoy는, Envoy등, Envoy
Enzyme, Enzyme을, Enzyme를, Enzyme이, Enzyme가, Enzyme와, Enzyme과, Enzyme은, Enzyme는, Enzyme
ES6, ES6을, ES6를, ES6이, ES6가, ES6와, ES6과, ES6은, ES6는, ES6등, ES6에, ES6이나, ES6나
Espresso, Espresso을, Espresso를, Espresso이, Espresso가, Espresso와, Espresso과, Espresso은,
ETC, ETC을, ETC를, ETC이, ETC가, ETC와, ETC과, ETC은, ETC는, ETC등, ETC에, ETC이나, ETC나
ExoPlayer, ExoPlayer을, ExoPlayer를, ExoPlayer이, ExoPlayer가, ExoPlayer와, ExoPlayer과, ExoP
ExpressJS, ExpressJS을, ExpressJS를, ExpressJS이, ExpressJS가, ExpressJS와, ExpressJS과, Expr
Falcon, Falcon을, Falcon를, Falcon이, Falcon가, Falcon와, Falcon과, Falcon은, Falcon는, Falcon
FastAPI, FastAPI을, FastAPI를, FastAPI이, FastAPI가, FastAPI와, FastAPI과, FastAPI은, FastAPI
Fastify, Fastify을, Fastify를, Fastify이, Fastify가, Fastify와, Fastify과, Fastify은, Fastify
Fastlane, Fastlane을, Fastlane를, Fastlane이, Fastlane가, Fastlane와, Fastlane과, Fastlane은,
Fiber, Fiber을, Fiber를, Fiber이, Fiber가, Fiber와, Fiber과, Fiber은, Fiber는, Fiber등, Fiber
Figma, Figma을, Figma를, Figma이, Figma가, Figma와, Figma과, Figma은, Figma는, Figma등, Figma
Flask, Flask을, Flask를, Flask이, Flask가, Flask와, Flask과, Flask은, Flask는, Flask등, Flask
Flink, Flink을, Flink를, Flink이, Flink가, Flink와, Flink과, Flink은, Flink는, Flink등, Flink
Flow, Flow을, Flow를, Flow이, Flow가, Flow와, Flow과, Flow은, Flow는, Flow등, Flow에, Flow이나,
Fluentd, Fluentd을, Fluentd를, Fluentd이, Fluentd가, Fluentd와, Fluentd과, Fluentd은, Fluentd
Flutter, Flutter을, Flutter를, Flutter이, Flutter가, Flutter와, Flutter과, Flutter은, Flutter
Frontend, Frontend을, Frontend를, Frontend이, Frontend가, Frontend와, Frontend과, Frontend은,
Gatsby, Gatsby을, Gatsby를, Gatsby이, Gatsby가, Gatsby와, Gatsby과, Gatsby은, Gatsby는, Gatsby

......(중략).........
```

▼ (2) MongoDB

📁**database/mongodb/**

📄.env

```
MONGO_INITDB_ROOT_USERNAME=develover
MONGO_INITDB_ROOT_PASSWORD=youdeveloveme?
```

📄docker-compose.yml

```
version: "3"
services:
  mongodb:
    image: mongo:7.0.6
    # 컨테이너 실행 시 재시작
    restart: always
    # 컨테이너명 설정
    container_name: mongodb
    # 접근 포트 설정
    ports:
      - 27017
    # 환경 변수 설정
    environment:
      # MongoDB 계정 및 패스워드 설정 옵션
       MONGO_INITDB_ROOT_USERNAME: ${MONGO_INITDB_ROOT_USERNAME}
       MONGO_INITDB_ROOT_PASSWORD: ${MONGO_INITDB_ROOT_PASSWORD}
    # 볼륨 설정
    volumes:
      - ./data/mongodb:/data/db
    # 네트워크 설정
    networks:
      - my-network

networks:
  my-network:
    external: true
```

▼ (3) MySQL

📁**database/mysql/master/**

📄Dockerfile

```
FROM mysql:8.0.33
ADD ./master/my.cnf /etc/mysql/my.cnf
```

📄my.cnf

```
[mysqld]
log_bin = mysql-bin
server_id=10
binlog_do_db=pickitup
default_authentication_plugin=mysql_native_password
```

📁**database/mysql/slave/**

📄Dockerfile

```
FROM mysql:8.0.33
ADD ./slave/my.cnf /etc/mysql/my.cnf
```

📄my.cnf

```
[mysqld]
log_bin = mysql-bin
server_id=11
relay_log = /var/lib/mysql/mysql-relay-bin
log_slave_updates = 'ON'
```

```
read_only = 'ON'
default_authentication_plugin=mysql_native_password
```

📁**database/mysql/**

📄.env

```
MYSQL_ROOT_PASSWORD=youcantguessrootpassword
MYSQL_DATABASE=pickitup
MYSQL_USER=develover
MYSQL_PASSWORD=youdeveloveme?
```

📄docker-compose.yml

```
version: "3"

services:

  mysql-master:
    build:
      context: ./
      dockerfile: mysql/master/Dockerfile
    # 컨테이너 실행 시 재시작
    restart: always
    # 컨테이너명 설정
    container_name: mysql-master
    # 환경 변수 설정
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_ROOT_HOST: '%'
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      TZ: 'Asia/Seoul'
    # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
    ports:
      - '3307:3306'
    # 볼륨 설정
    volumes:
      - master:/var/lib/mysql
    # 네트워크 설정
    networks:
      - my-network

  mysql-slave:
    build:
      context: ./
      dockerfile: mysql/slave/Dockerfile
    restart: always
    container_name: mysql-slave
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_ROOT_HOST: '%'
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      TZ: 'Asia/Seoul'
    ports:
      - '3308:3306'
    # Where our data will be persisted
    volumes:
      - slave:/var/lib/mysql
    networks:
      - my-network

volumes:
  master:
  slave:

networks:
  my-network:
    external: true
```

⚠ build error 발생하는 Docker version : 25.0.2

(`failed to solve: changes out of order`)

(4) Redis

📁**database/redis/**

📄docker-compose.yml

```
version: "3"

services:

  redis:
      image: redis:latest
      container_name: redis
      volumes:
        - ./redis/redis_vol:/data
      ports:
        - 6379:6379
      networks:
        - my-network

networks:
  my-network:
    external: true
```

📁 Database

📌 database 루트 위치에 전체 DB를 한번에 실행 시키는 도커 컴포즈 파일 작성

📁**database/**

📄.env

```
MYSQL_ROOT_PASSWORD=youcantguessrootpassword
MYSQL_DATABASE=pickitup
MYSQL_USER=develover
MYSQL_PASSWORD=youdeveloveme?
MONGO_INITDB_ROOT_USERNAME=develover
MONGO_INITDB_ROOT_PASSWORD=youdeveloveme?
```

📄docker-compose.yml

```
version: "3"

services:

  mysql-master:
    build:
      context: ./
      dockerfile: mysql/master/Dockerfile
    # 컨테이너 실행 시 재시작
    restart: always
    # 컨테이너명 설정
    container_name: mysql-master
    # 환경 변수 설정
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_ROOT_HOST: '%'
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      TZ: 'Asia/Seoul'
    # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
    ports:
      - '3307:3306'
    # 볼륨 설정
    volumes:
      - master:/var/lib/mysql
    # 네트워크 설정
    networks:
      - my-network

  mysql-slave:
    build:
      context: ./
      dockerfile: mysql/slave/Dockerfile
    restart: always
    container_name: mysql-slave
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_ROOT_HOST: '%'
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      TZ: 'Asia/Seoul'
    ports:
      - '3308:3306'
    # Where our data will be persisted
    volumes:
      - slave:/var/lib/mysql
    networks:
      - my-network

  #elastic search
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.12.2
    # 컨테이너 이름
    container_name: elasticsearch
    # 환경 변수
    environment:
      - node.name=es-node
      - cluster.name=search-cluster
      - discovery.type=single-node
      - xpack.security.enabled=false
      - xpack.security.http.ssl.enabled=false
      - xpack.security.transport.ssl.enabled=false
    # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
    ports:
      - 9200
    # 네트워크 설정
    networks:
      - my-network

  kibana:
    image: docker.elastic.co/kibana/kibana:8.5.3
    container_name: kibana
    environment:
      SERVER_NAME: kibana
      ELASTICSEARCH_HOSTS: http://elasticsearch:9200
    ports:
      - 5601
    # Elasticsearch 가 실행 된 후에 kibana 실행
    depends_on:
      - elasticsearch
    # 네트워크 설정
    networks:
      - my-network

  mongodb:
    image: mongo:7.0.6
    # 컨테이너 실행 시 재시작
    restart: always
    # 컨테이너명 설정
    container_name: mongodb
    # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
    ports:
      - 27017
    # 환경 변수 설정
    environment:
      # MongoDB 계정 및 패스워드 설정 옵션
      MONGO_INITDB_ROOT_USERNAME: ${MONGO_INITDB_ROOT_USERNAME}
      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_INITDB_ROOT_PASSWORD}
    # 볼륨 설정
    volumes:
      - ./data/mongodb:/data/db
    # 네트워크 설정
    networks:
      - my-network

  redis:
      image: redis:latest
      container_name: redis
      volumes:
        - ./redis/redis_vol:/data
      ports:
        - 6379
      networks:
        - my-network

# Names our volume
```

```
volumes:
  master:
  slave:

networks:
  my-network:
    external: true
```

▼ 🔧 MySQL master - slave 설정

✅ replication 설정 해주기

(1) Master DB 접속

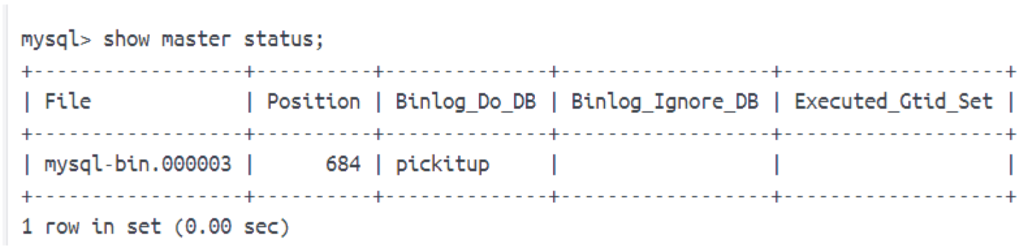
`Command`

```
create user 'replication_user'@'%' identified by 'repli_pass';

alter user 'replication_user'@'%' identified with mysql_native_password by 'repli_pass';

grant replication slave on *.* TO 'replication_user'@'%';
```

`Query`

```
//master db 정보 확인하기
show master status;
```

show master status; 쿼리 결과 예시



```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000003 |      684 | pickitup     |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

❗ 위의 값 중에서 File 컬럼과 Position 값을 기억해서 Slave DB 설정에 작성해주어야함

(2) Slave DB 접속

`Command`

```
CHANGE MASTER TO MASTER_HOST='mysql-master',
MASTER_USER='replication_user',
MASTER_PASSWORD='repli_pass',
MASTER_LOG_FILE='mysql-bin.000003', #master에서 확인한 log 파일명과 동일하게❗❗
MASTER_LOG_POS=978, #master에서 확인한 log position과 동일하게❗❗
GET_MASTER_PUBLIC_KEY=1;
```

`Query`

Slave 설정이 제대로 되었는지 확인 해주기

```
//Slave on 확인
show slave status\G;
```

⚠ master slave 끊겼을 때

🔶Slave

- slave 중지

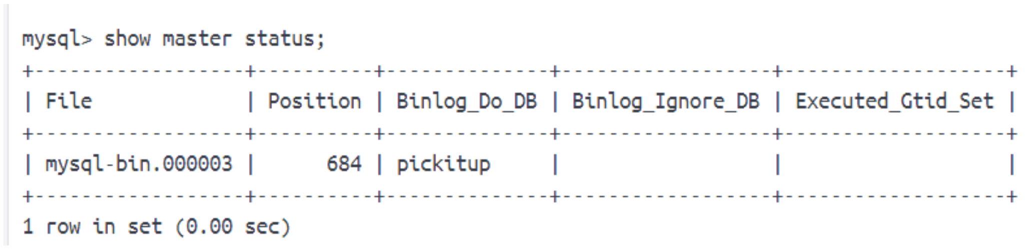
```
stop slave;
```

🔷Master

- master 로그 새출발 해주기

```
flush logs;
show master status;
```

(로그 파일과 위치 확인)

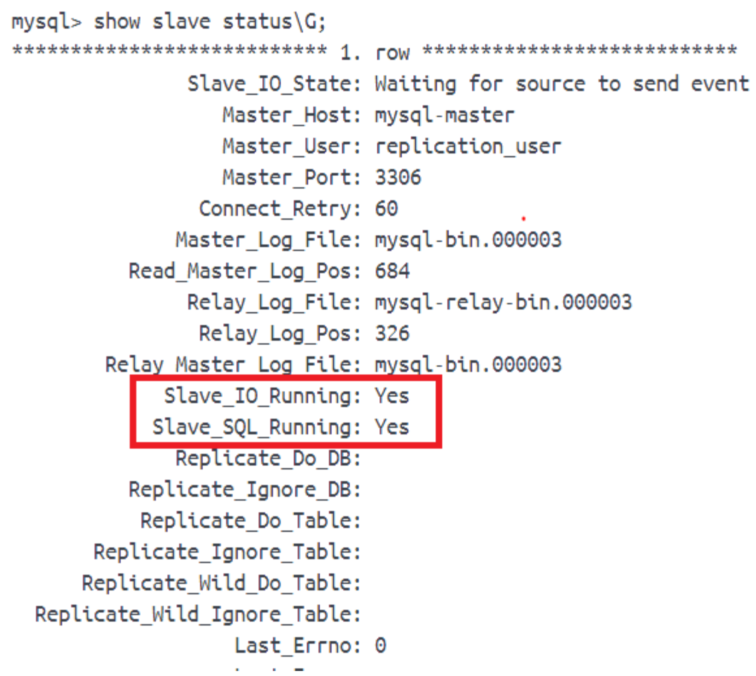


```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000003 |      684 | pickitup     |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

🔶Slave

- slave DB에서 master 재설정 후 재시작

```
CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000003', MASTER_LOG_POS=157;
start slave;
```



```
mysql> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for source to send event
                  Master_Host: mysql-master
                  Master_User: replication_user
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000003
          Read_Master_Log_Pos: 684
               Relay_Log_File: mysql-relay-bin.000003
                Relay_Log_Pos: 326
        Relay_Master_Log_File: mysql-bin.000003
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
```

`만약에 NO라고 보이면 SlaveDB 재시작후 확인`

▼ 5️⃣ 백엔드 빌드

▼ 1. 스프링 🌱

build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.3'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-data-elasticsearch'
    implementation 'org.springframework.data:spring-data-elasticsearch:4.2.2'
    implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-cache'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'
    implementation group: 'com.auth0', name: 'java-jwt', version: '4.4.0'
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-impl:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.11.5'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.1.0'
    implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.5.6'
    implementation 'com.google.code.geocoder-java:geocoder-java:0.16'
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
    implementation 'com.squareup.okhttp3:okhttp:4.9.3'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-processor'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

📌 `application.yml` 파일은 로컬 개발 환경과 서버 배포 환경이 동일한 파일을 사용

📌 `application-oauth.yml` 파일과 `env.yml` 파일은 로컬과 서버 환경이 각기 다른 파일을 사용

📂 src/main/resources

📄 application.yml

```
logging:
  level:
    org:
      elasticsearch:
        client: ERROR
spring:
  config:
    import:
      - optional:env.yml
      - optional:application-oauth.yml
  data:
    elasticsearch:
      cluster-nodes: ${elasticsearch.host}:${elasticsearch.port}
    mongodb:
      host: ${mongodb.host}
      port: ${mongodb.port}
      username: ${mongodb.username}
      password: ${mongodb.password}
      authentication-database: admin
      database: mydb
    redis:
      host: ${redis.host}
      port: ${redis.port}
  main:
    allow-bean-definition-overriding: true


  #mysql 설정
  datasource:
    master:
      hikari:
        driver-class-name: com.mysql.cj.jdbc.Driver
        jdbc-url: jdbc:mysql://${mysql.master.host}:${mysql.master.port}/${mysql.master.
database}?useSSL=false&serverTimezone=UTC
        read-only: false
        username: ${mysql.master.username}
        password: ${mysql.master.password}
    slave:
      hikari:
        driver-class-name: com.mysql.cj.jdbc.Driver
        jdbc-url: jdbc:mysql://${mysql.slave.host}:${mysql.slave.port}/${mysql.slave.dat
abase}?useSSL=false&serverTimezone=UTC
        read-only: true
        username: ${mysql.slave.username}
        password: ${mysql.slave.password}
  jpa:
    show-sql: true
    hibernate:
      properties:
        dialect: org.hibernate.dialect.MySQLDialect
        format_sql: true
    database-platform: mysql


springdoc:
  swagger-ui:
    path: /swagger-ui
    disable-swagger-default-url: true
  api-docs:
    path: /api-docs

kakao:
  api: ${kakao.api-key}
```

▼ 🔷 **로컬 서버**

📄application-oauth.yml

```
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            #REST API 키
            client-id: ${YOUR KAKAO CLIENT ID}
            client-secret: ${YOUR KAKAO CLIENT SECRET}
            scope: profile_nickname, account_email
            client-authentication-method: client_secret_post
            redirect-uri: http://localhost:8080/login/oauth2/code/kakao
            authorization-grant-type: authorization_code
            client-name: kakao

          google:
            client-id: ${YOUR GOOGLE CLIENT ID}
            client-secret: ${YOUR GOOGLE CLIENT SECRET}
            scope: profile,email
            redirect-uri: http://localhost:8080/login/oauth2/code/google

          naver:
            client-id: ${YOUR NAVER CLIENT ID}
            client-secret: ${YOUR NAVER CLIENT SECRET}
            scope: name,email
            authorization-grant-type: authorization_code
            redirect-uri: http://localhost:8080/login/oauth2/code/naver
            client-name: Naver

        provider:
          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

          naver:
            authorization-uri: https://nid.naver.com/oauth2.0/authorize
            token-uri: https://nid.naver.com/oauth2.0/token
            user-info-uri: https://openapi.naver.com/v1/nid/me
            user-name-attribute: response
```

env.yml

```
mysql:
  master:
    host: localhost
    port: 3307
    database: pickitup
    username: develover
    password: youdeveloveme?
  slave:
    host: localhost
    port: 3308
    database: pickitup
    username: develover
    password: youdeveloveme?

elasticsearch:
  host: localhost
  port: 9200

mongodb:
  host: localhost
  port: 27017
  username: develover
  password: youdeveloveme?

redis:
  host: localhost
  port: 6379

kakao:
  api-key: KakaoAK ${YOUR KAKAO API KEY}
```

▼ 🔶**배포 서버**

application-oauth.yml

```
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            #REST API 키
            client-id: ${YOUR KAKAO CLIENT ID}
            client-secret: ${YOUR KAKAO CLIENT SECRET}
            scope: profile_nickname, account_email
            client-authentication-method: client_secret_post
            redirect-uri: https://spring.pickitup.online/login/oauth2/code/kakao
            authorization-grant-type: authorization_code
            client-name: kakao
          google:
                  client-id: ${YOUR GOOGLE CLIENT ID}
            client-secret: ${YOUR GOOGLE CLIENT SECRET}
            scope: profile,email
            redirect-uri: https://spring.pickitup.online/login/oauth2/code/google

          naver:
            client-id: ${YOUR NAVER CLIENT ID}
            client-secret: ${YOUR NAVER CLIENT SECRET}
            scope: name,email
            authorization-grant-type: authorization_code
            redirect-uri: https://spring.pickitup.online/login/oauth2/code/naver
            client-name: Naver

        provider:
          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

          naver:
            authorization-uri: https://nid.naver.com/oauth2.0/authorize
            token-uri: https://nid.naver.com/oauth2.0/token
            user-info-uri: https://openapi.naver.com/v1/nid/me
            user-name-attribute: response
```

env.yml

```
mysql:
  master:
    host: mysql-master
    port: 3306
    database: pickitup
    username: develover
    password: youdeveloveme?
  slave:
    host: mysql-slave
    port: 3306
    database: pickitup
    username: develover
    password: youdeveloveme?

elasticsearch:
  host: elasticsearch
  port: 9200

mongodb:
  host: mongodb
  port: 27017
  username: develover
  password: youdeveloveme?

redis:
  host: redis
  port: 6379
```

```
kakao:
  api-key: KakaoAK ${YOUR KAKAO API KEY}
```

▼ 2. Play + Spark ⚡

build.sbt

```
name := """play-spark"""
organization := "com.ssafy"
maintainer := "j10a406"

version := "1.0-SNAPSHOT"

lazy val root = (project in file(".")).enablePlugins(PlayScala)

scalaVersion := "2.12.16"


libraryDependencies += guice
libraryDependencies += "org.scalatestplus.play" %% "scalatestplus-play" % "5.1.0" % Test
libraryDependencies ++= Seq(
  guice,
  "com.typesafe" % "config" % "1.4.1",
  "org.scalatestplus.play" %% "scalatestplus-play" % "5.1.0" % Test,
  "org.scalatest" %% "scalatest" % "3.0.8" % Test,

  "org.mongodb.scala" %% "mongo-scala-driver" % "4.0.5",
  // Spark ( + MongoDB )
  "org.apache.spark" %% "spark-core" % "3.0.2",
  "org.apache.spark" %% "spark-sql" % "3.0.2",
  "org.apache.spark" %% "spark-mllib" % "3.0.2",
  "org.mongodb.spark" %% "mongo-spark-connector" % "3.0.2",
)
```

📌 `application.conf` 파일은 시스템 환경변수로부터 값을 주입받으며, Docker 컨테이너 상에서 실행되므로 docker-compose 파일과 동일 위치에서 `.env` 파일을 통해 주입한다.

▼ conf/application.conf

```
# https://www.playframework.com/documentation/latest/Configuration
play.filters.hosts.allowed=["localhost", "127.0.0.1", "::1", ".pickitup.online"]
play.modules.enabled += "modules.SparkWarmUpModule"

mongo {
  hostname=localhost
  hostname=${?MONGO_HOSTNAME}
  port=27017
  port=${?MONGO_PORT}
  database=test
  database=${?MONGO_DATABASE}
  username=root
  username=${?MONGO_USERNAME}
  password=password
  password=${?MONGO_PASSWORD}
}
```

▼ .env

```
NETWORK_NAME=my-network
MONGO_HOSTNAME=mongodb
MONGO_PORT=27017
MONGO_DATABASE=recommend
MONGO_USERNAME=develover
MONGO_PASSWORD=youdeveloveme?
SBT_PROJECT_HOST_PATH=.
PLAY_HTTP_SECRET_KEY=pickitup_j10a406_scala
```

▼ 🔷**Docker 컨테이너 실행 (로컬/개발 환경)**

play-spark-local-compose.yml

```
version: "3.8"
services:
  play-app-deploy:
    image: sbtscala/scala-sbt:openjdk-8u342_1.7.2_2.12.16
    container_name: play-app-local
    ports:
      - "9000:9000"
    volumes:
      - ${SBT_PROJECT_HOST_PATH}:/app
    networks:
      - my-network
    working_dir: /app
    command: >
      /bin/bash -c "sbt run"
    environment:
      - MONGO_HOSTNAME=${MONGO_HOSTNAME}
      - MONGO_PORT=${MONGO_PORT}
      - MONGO_DATABASE=${MONGO_DATABASE}
      - MONGO_USERNAME=${MONGO_USERNAME}
      - MONGO_PASSWORD=${MONGO_PASSWORD}
      - PLAY_HTTP_SECRET_KEY=${PLAY_HTTP_SECRET_KEY}
networks:
  my-network:
    external: true
```

▼ 🔶**Docker 컨테이너 실행 (프로덕션 환경)**

play-spark-build-compose.yml **(빌드)**

```
version: "3.8"
services:
  play-app-build:
    image: sbtscala/scala-sbt:openjdk-8u342_1.7.2_2.12.16
    container_name: play-app-build
    volumes:
      - ${SBT_PROJECT_HOST_PATH}:/app
    working_dir: /app
    command: >
      /bin/bash -c "
      sbt compile &&
      sbt test &&
      sbt dist &&
      cd /app/target/universal/ &&
      unzip -o play-spark-1.0-SNAPSHOT.zip
      "
```

📄play-spark-build-compose.yml **(배포)**

```
version: "3.8"
services:
  play-app-deploy:
    image: sbtscala/scala-sbt:openjdk-8u342_1.7.2_2.12.16
    container_name: play-app-deploy
    ports:
      - "9000:9000"
    volumes:
      - ${SBT_PROJECT_HOST_PATH}:/app
    networks:
      - my-network
    working_dir: /app/target/universal/play-spark-1.0-SNAPSHOT/bin
    command: >
      /bin/bash -c "
      ./play-spark -Dplay.http.secret.key=${PLAY_HTTP_SECRET_KEY}"
    deploy:
      resources:
        limits:
          cpus: '3.6'
    environment:
      - MONGO_HOSTNAME=${MONGO_HOSTNAME}
      - MONGO_PORT=${MONGO_PORT}
      - MONGO_DATABASE=${MONGO_DATABASE}
      - MONGO_USERNAME=${MONGO_USERNAME}
      - MONGO_PASSWORD=${MONGO_PASSWORD}
      - PLAY_HTTP_SECRET_KEY=${PLAY_HTTP_SECRET_KEY}
networks:
  my-network:
    external: true
```

▼ 6️⃣ 프론트엔드 빌드

```
//tsconfig.json

{
  "compilerOptions": {
    "lib": ["dom", "dom.iterable", "esnext"],
    "allowJs": true,
    "skipLibCheck": true,
    "strict": true,
    "noEmit": true,
    "esModuleInterop": true,
    "module": "esnext",
    "moduleResolution": "bundler",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "jsx": "preserve",
    "incremental": true,
    "target": "es2015",
    "plugins": [
      {
        "name": "next"
      }
    ],
    "paths": {
      "@/*": ["./*"]
    }
  },
  "include": ["next-env.d.ts", "**/*.ts", "**/*.tsx", ".next/types/**/*.ts"],
  "exclude": ["node_modules"]
}
```

```
//tailwind.config.ts
```
import type { Config } from "tailwindcss";

const colors = require("tailwindcss/colors");

const config: Config = {
  content: [
    "./src/pages/**/*.{js,ts,jsx,tsx,mdx}",
    "./src/components/**/*.{js,ts,jsx,tsx,mdx}",
    "./src/app/**/*.{js,ts,jsx,tsx,mdx}",
    "./components/**/*.{js,ts,jsx,tsx,mdx}",
  ],
  theme: {
    screens: {
      mb: { max: "480px" },
    },
    extend: {
      colors: {
        f5green: {
          100: "#e6faf2",
          150: "#d9f8eb",
          200: "#b0f0d6",
          300: "#00ce7c",
          350: "#00b970",
          400: "#00a563",
          500: "#009b5d",
          550: "#007c4a",
          600: "#005d38",
          700: "#00482b",
        },
        f5greenn: {
          100: "#57b53f15",
          200: "#57b53f",
        },
        f5yellowgreen: {
          200: "#d7ffa4",
          300: "#c1ff72",
        },
        f5red: {
          100: "#ffecec",
          150: "#ffe3e3",
          200: "#ffc4c4",
          300: "#ff4242",
          350: "#e63b3b",
          400: "#cc3535",
          500: "#bf3232",
          550: "#992828",
```
```

```
          600: "#731e1e",
          700: "#591717",
        },
        f5redd: {
          100: "#e75e5f15",
          200: "#e75e5f",
        },
        f5black: {
          400: "#424242",
          500: "#383838",
          600: "#171717",
        },
        f5gray: {
          200: "#F4F4F4",
          300: "#E8E8E8",
          400: "#D9D9D9",
          500: "#848484",
          600: "#888888",
        },
        f5blue: {
          100: "#F6FAFF",
        },
      },
      backgroundImage: {
        "gradient-radial": "radial-gradient(var(--tw-gradient-stops))",
        "gradient-conic":
          "conic-gradient(from 180deg at 50% 50%, var(--tw-gradient-stops))",
      },
      keyframes: {
        startGauge: {
          "0%": {
            width: "0%",
          },
          "100%": {
            width: "100%",
          },
        },
        fadeIn: {
          "0%": { opacity: "0" },
          "100%": { opacity: "1" },
        },
        scaleIn: {
          "0%": { transform: "scale(0)" },
          "100%": { transform: "scale(1)" },
        },
        slideUp: {
          "0%": {
            transform: "translateY(100%)",
            opacity: "0",
          },
          "100%": {
            transform: "translateY(0)",
            opacity: "1",
          },
        },
        slideRight: {
          "0%": { transform: "translateX(-100%)" },
          "100%": { transform: "translateX(0)" },
        },
        bounce: {
          "0%, 100%": {
            transform: "translateY(-15%)", // 바운스 높이 조정
            "animation-timing-function": "cubic-bezier(0.8,0,1,1)",
          },
          "50%": {
            transform: "none",
            "animation-timing-function": "cubic-bezier(0,0,0.2,1)",
          },
        },
      },
      animation: {
        startGauge: "startGauge 10s forwards linear",
        startGauge4: "startGauge 40s forwards linear",
        "fade-in": "fadeIn 0.7s ease-in",
        "fade-in-delayed": "fadeIn 0.7s ease-in forwards 0.3s",
        "scale-in": "scaleIn 0.7s ease-in",
        "slide-up": "slideUp 0.7s ease-in",
        "slide-right": "slideRight 0.7s ease-in",
      },
    },
  },
  plugins: [],
};
export default config;

```
```

```
//package.json
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@tanstack/react-query": "^5.28.6",
    "@tanstack/react-query-devtools": "^5.28.6",
    "@types/lodash": "^4.17.0",
    "@types/lodash.clonedeep": "^4.5.9",
    "canvas-confetti": "^1.9.2",
    "lodash": "^4.17.21",
    "next": "14.1.1",
    "react": "^18",
    "react-canvas-confetti": "^2.0.7",
    "react-daum-postcode": "^3.1.3",
    "react-dom": "^18",
    "react-icons": "^5.0.1",
    "react-responsive": "^10.0.0",
    "react-spinners": "^0.13.8",
    "react-swipeable": "^7.0.1",
    "sweetalert2": "^11.10.7",
    "swiper": "^11.1.0",
    "zustand": "^4.5.2"
  },
```
```

```
  "devDependencies": {
    "@types/node": "^20",
    "@types/react": "^18",
    "@types/react-dom": "^18",
    "autoprefixer": "^10.0.1",
    "eslint": "^8",
    "eslint-config-next": "14.1.1",
    "eslint-config-prettier": "^9.1.0",
    "postcss": "^8",
    "prettier": "^3.2.5",
    "prettier-plugin-tailwindcss": "^0.5.12",
    "tailwindcss": "^3.3.0",
    "typescript": "^5"
  }
}
```

```
//next.config.mjs

/** @type {import('next').NextConfig} */
const nextConfig = {
  images: {
    domains: ["image.wanted.co.kr", "jpassets.jobplanet.co.kr"],
  },
};

export default nextConfig;
```

▼ 7️⃣ 크롤링

1. Python 패키지 설치

- pip install BeautifulSoup4 pandas selenium ChromeDriverManager

2. Selenium을 사용해 크롬 드라이버 동적 제어

```
def initialize_driver():
    # 웹 드라이버 경로 지정
    driver_path = '/path/to/chromedriver'

    # 웹 드라이버 설정
    chrome_options = Options()
    chrome_options.add_experimental_option("detach", True)

    # 크롬 드라이버 설정
    driver = webdriver.Chrome(options=chrome_options)
    return driver
```

3. BeautifulSoup를 사용해 요소 추출

```
def get_post_list(driver):
    scroll_to_bottom(driver)

    # 스크롤 이후의 페이지 소스 가져오기
    page_source = driver.page_source

    # BeautifulSoup을 사용하여 데이터 파싱
    soup = BeautifulSoup(page_source, "html.parser")
    # 리스트 추출
    return soup.select('.Card_Card__lU7z_')
```

4. 추출한 데이터 csv 파일로 저장

```
def to_csv(data):
    pathlink ="C:\\SSAFY\\yutw\\data\\searchrecruit"

    # db create
    if not os.path.isdir(pathlink):
        os.mkdir(pathlink)

    data_df = pd.DataFrame([data])  # 데이터프레임으로 변환

    file_path = os.path.join(pathlink, "recruitdata.csv")

    if os.path.exists(file_path):
        # 파일이 이미 존재하면 append 모드로 추가
        data_df.to_csv(file_path, mode='a', header=False, index=False, encoding='utf-8-sig')
    else:
        # 파일이 없으면 빈 데이터프레임을 생성하여 저장
        empty_df = pd.DataFrame(columns=data_df.columns)  # 데이터프레임 컬럼 구조를 유지하기 위해
        empty_df.to_csv(file_path, index=False, encoding='utf-8-sig')

        # 생성한 빈 데이터프레임에 데이터 추가
        data_df.to_csv(file_path, mode='a', header=False, index=False, encoding='utf-8-sig')
```

5. csv 파일을 읽어 elasticsearch에 저장

```
def to_elastic(data):
    pathlink ="C:\\SSAFY\\yutw\\data\\searchcompany"

    del_date = str(datetime.utcnow() - timedelta(hours=39))[:10]

    cnt = len(pd.read_csv(pathlink + "/" + "companydata.csv", index_col=0).index)

    es.index(index="searchcompany", id=str(cnt), body=json.dumps(data))
```

▼ 8️⃣ CI / CD

▼ 1. Jenkins

▼ 🔒KEY

🔑Jenkins admin 비밀번호

```
${JENKINS_ADMIN}
```

🔑GitLab Access Token

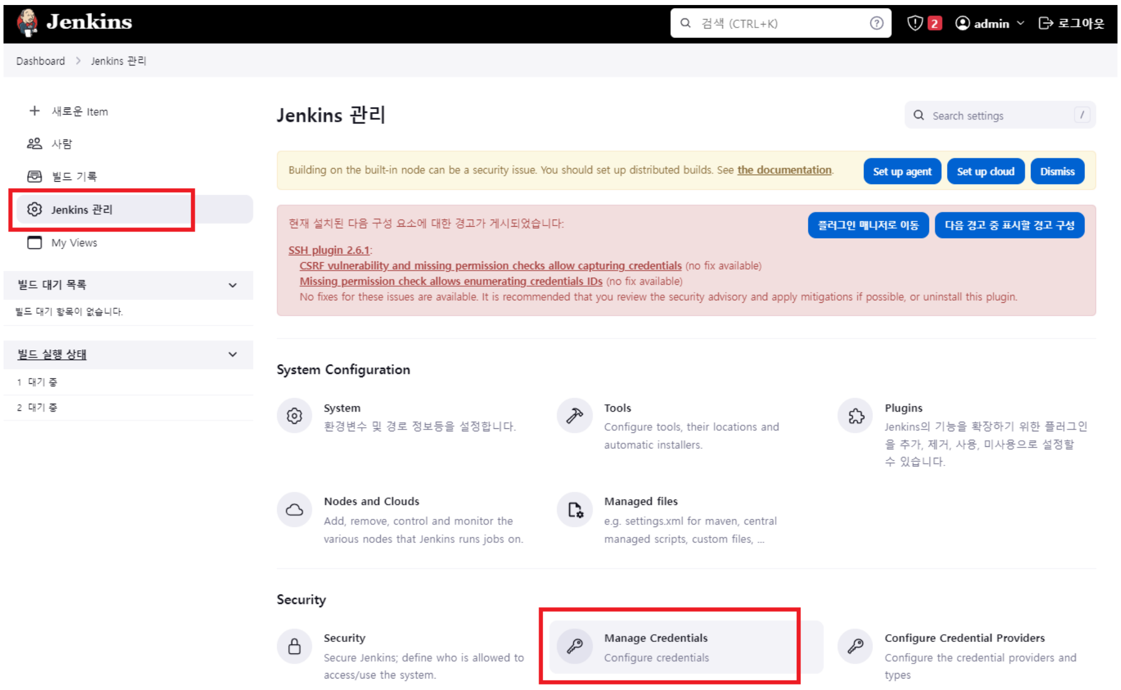
```
${GITLAB_ACCESS_TOKEN}
```

🔑Jenkins Spring 파이프 라인 secret key

```
${JENKINS_PIPELINE_SECRET_KEY}
```

▼ (1) GitLab Credential

✅ Credential 등록



Jenkins 검색 (CTRL+K) 2 admin 로그아웃

Dashboard > Jenkins 관리

+ 새로운 Item
사람
빌드 기록
Jenkins 관리
My Views

빌드 대기 목록
빌드 대기 항목이 없습니다.

빌드 실행 상태
1 대기 중
2 대기 중

Jenkins 관리

Search settings

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

현재 설치된 다음 구성 요소에 대한 경고가 게시되었습니다: 플러그인 매니저로 이동 다음 경고 중 표시할 경고 구성

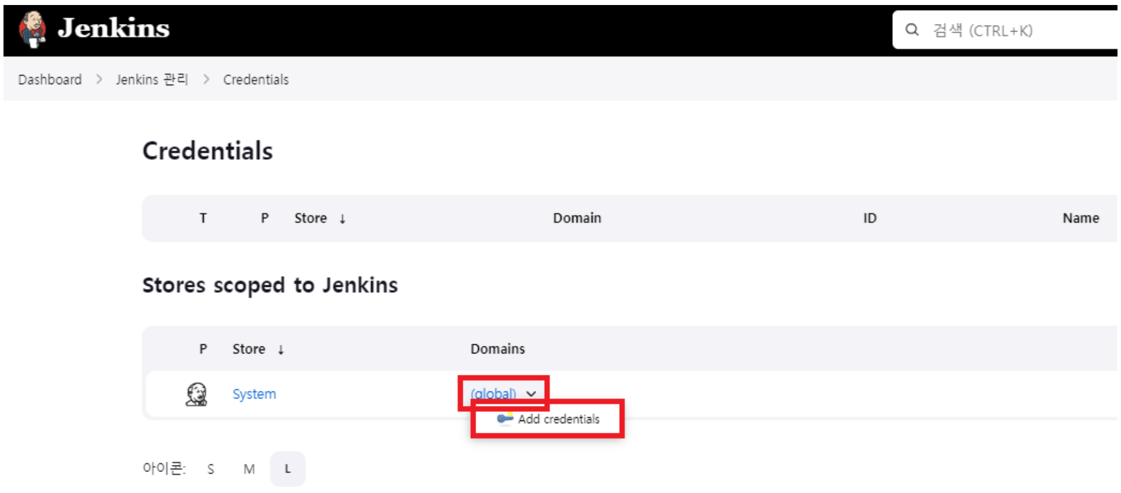
SSH plugin 2.6.1:
CSRF vulnerability and missing permission checks allow capturing credentials (no fix available)
Missing permission check allows enumerating credentials IDs (no fix available)
No fixes for these issues are available. It is recommended that you review the security advisory and apply mitigations if possible, or uninstall this plugin.

System Configuration

System 환경변수 및 경로 정보등을 설정합니다.	Tools Configure tools, their locations and automatic installers.	Plugins Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.
Nodes and Clouds Add, remove, control and monitor the various nodes that Jenkins runs jobs on.	Managed files e.g. settings.xml for maven, central managed scripts, custom files, ...	

Security

Security Secure Jenkins; define who is allowed to access/use the system.	Manage Credentials Configure credentials	Configure Credential Providers Configure the credential providers and types



Jenkins 검색 (CTRL+K)

Dashboard > Jenkins 관리 > Credentials

Credentials

T	P	Store ↓	Domain	ID	Name

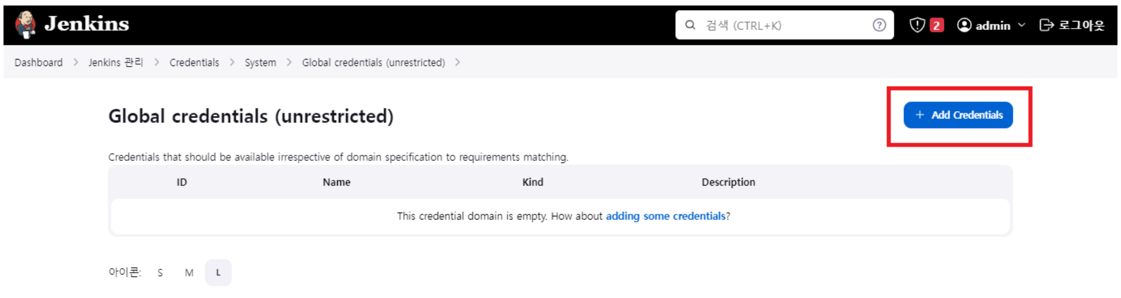
Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Add credentials

아이콘: S M L

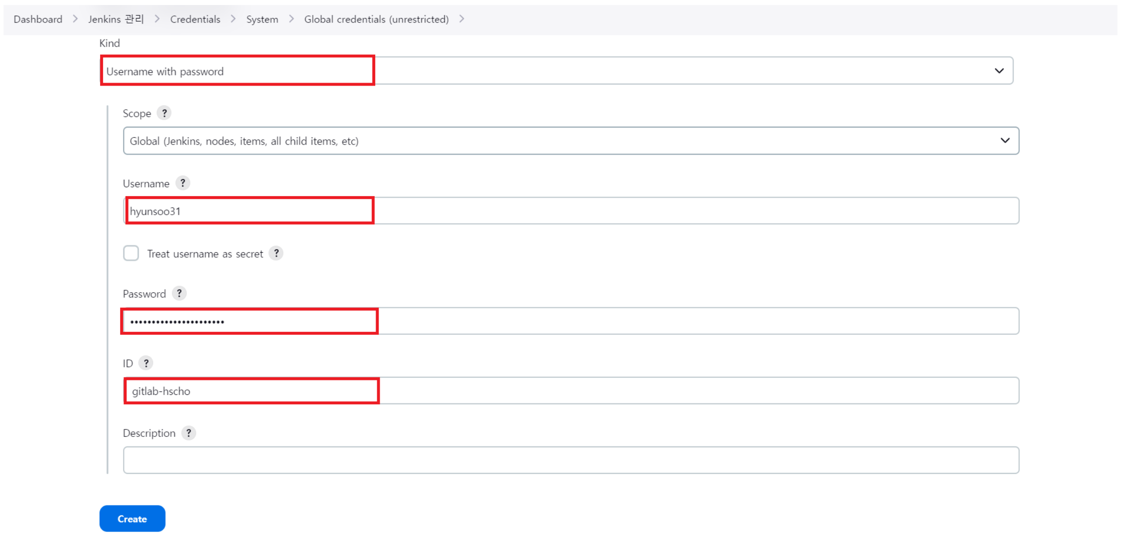
📍GitLab (Username with password)


Jenkins
검색 (CTRL+K)
2
admin
로그아웃
Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >
+ Add Credentials
아이콘: S M L

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			


Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >
Create

Kind: Username with password

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

Username: hyunsoo31

Treat username as secret

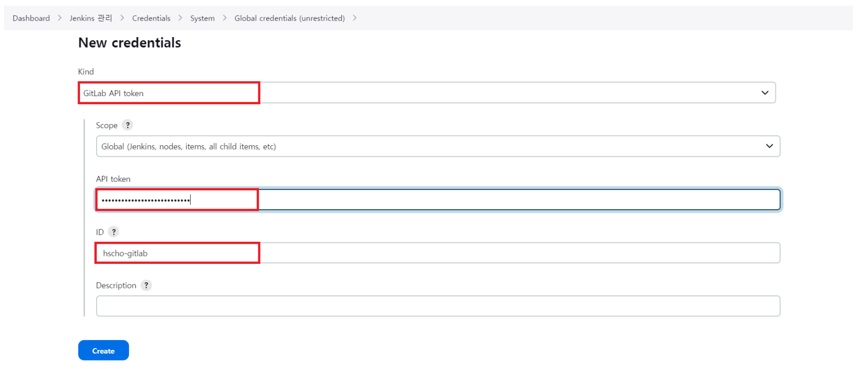
Password: ••••••••••••••••••••••••

ID: gitlab-hscho

Description:

- **username : GitLab ID(로그인 아이디)**
- **username : Gitlab password**

📍GitLab (API token)


Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >
Create

New credentials

Kind: GitLab API token

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

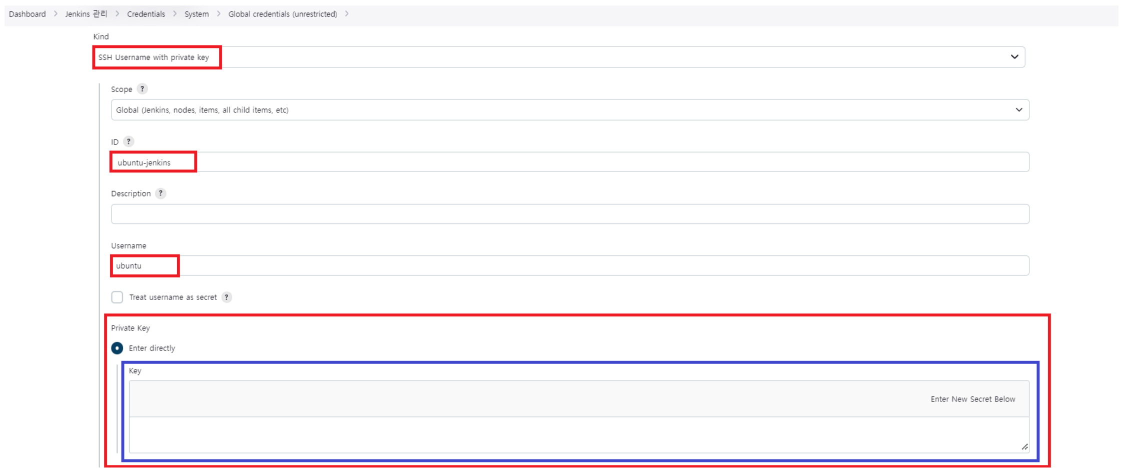
API token: ••••••••••••••••••••••••••

ID: hscho-gitlab

Description:

- **API token: GitLab Access Token**

📍Ubuntu (SSH)



Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

Kind: SSH Username with private key

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

ID: ubuntu-jenkins

Description:

Username: ubuntu

☐ Treat username as secret

Private Key

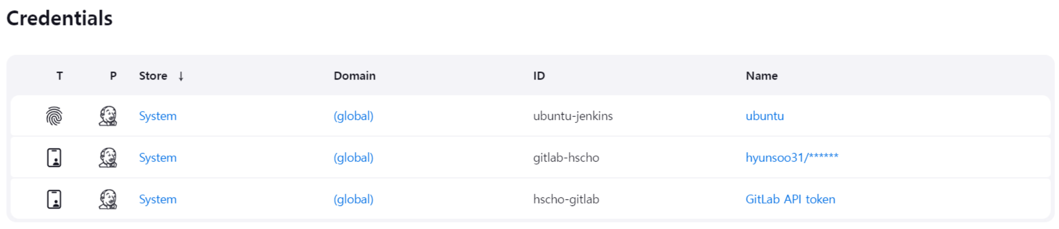
● Enter directly

Key

Enter New Secret Below

- **ID: Jenkins에서 Credential에 지정할 별칭**
- **Username: SSH 원격 서버 호스트에서 사용하는 계정이름**
- **Key: *.pem 키의 내용을 메모장으로 복사후 붙여넣기**

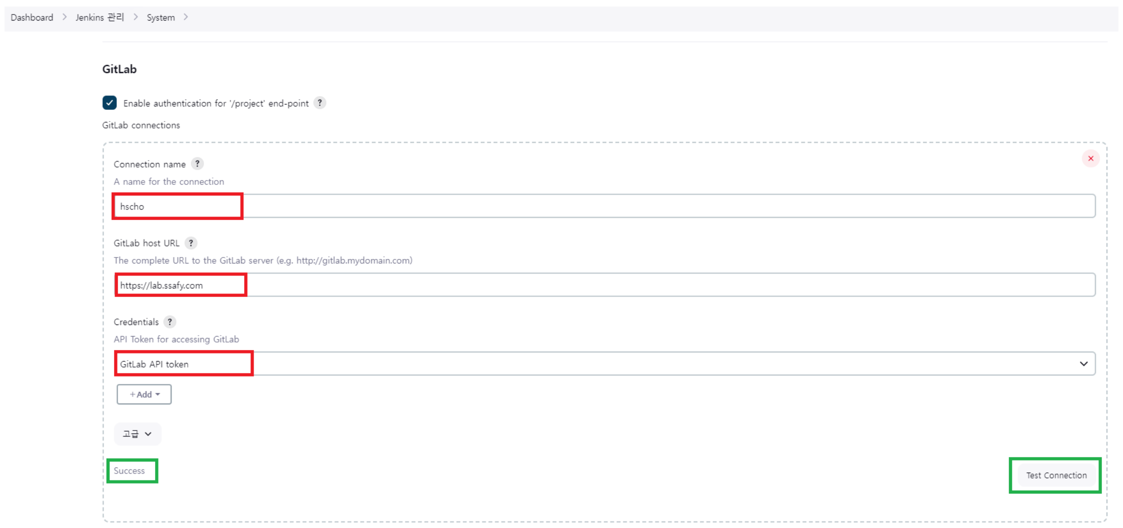
📍Credentials 등록 확인



Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	ubuntu-jenkins	ubuntu
		System	(global)	gitlab-hscho	hyunsoo31/******
		System	(global)	hscho-gitlab	GitLab API token

✅GitLab과 연결



Dashboard > Jenkins 관리 > System >

GitLab

☑ Enable authentication for '/project' end-point

GitLab connections

Connection name
A name for the connection
hscho

GitLab host URL
The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)
https://lab.ssafy.com

Credentials
API Token for accessing GitLab
GitLab API token

- **Connection name:** connection 이름 설정
- **GitLab host URL:** gitlab URL 작성
- **GitLab API token** 사용

- **Test Connection** 눌러서 Success 출력되는지 확인

▼ (2) Jenkins pipline

▼ FE

▼ Build Triggers

- Push events

▼ Secret Token

```
e250af91ddd32cf174e4f95652c8ca49
```

▼ Pipeline Script

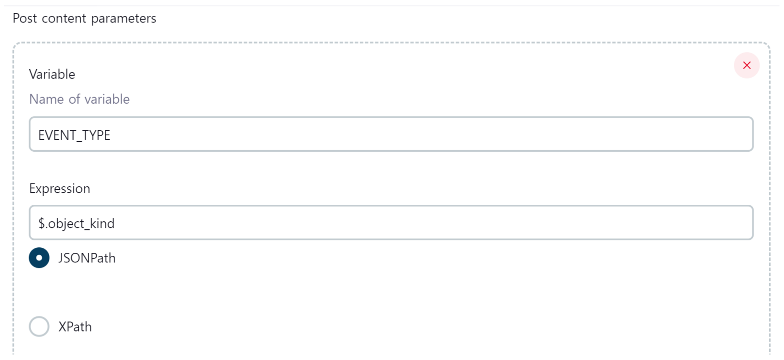
```
    pipeline {
        agent any

        environment {
            imageName = "pickitup/frontend-develop"
            registryCredential = 'hyunsoo31'
            dockerImage = ''

            releaseServerAccount = 'ubuntu'
            releaseServerUri = 'j10a406.p.ssafy.io'
            releasePort = '8080'
        }

        stages {

            stage('gitlab Connect'){
                steps{
                    git branch: 'dev-frontend',
                    credentialsId: 'gitlab-hscho',
                    url: 'https://lab.ssafy.com/s10-bigdata-recom-sub2/S10P22A40
6.git'
                }
            }



            stage('deploy'){
                steps{
                    sh 'docker stop next-app || true'
                    sh 'docker rm next-app || true '
                    sh 'docker rmi frontend-next-app || true'

                    dir('frontend/'){
                        script{

                            // 도커 컴포즈 파일 경로 지정
                            def dockerComposeFile = 'docker-compose.yml'

                            // 도커 컴포즈 실행 명령어
                            def dockerComposeCmd = " docker compose up -d"

                            // 도커 컴포즈 실행
                            sh """
                                ${dockerComposeCmd}
                            """
```

```
                                }
                            }
                        }

                    }

                }

            }
```

▼ **BE**

Spring / Play 별도 처리를 위해 Jenkins **Generic Webhook Trigger** 플러그인 활용

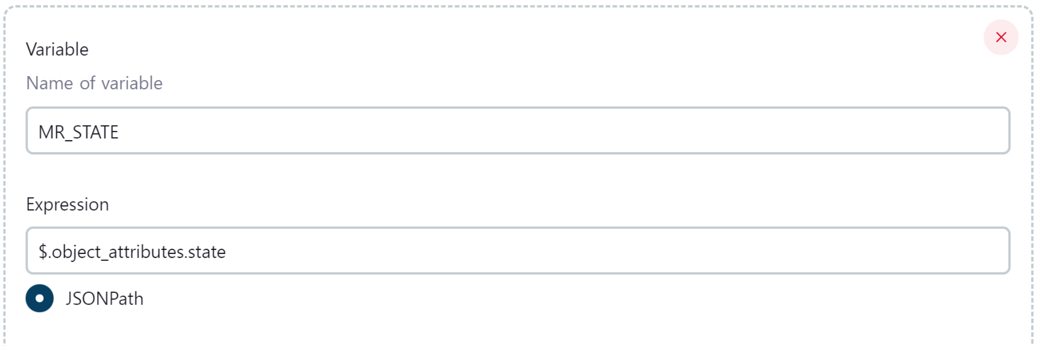
▼ **Post content parameters**



Post content parameters

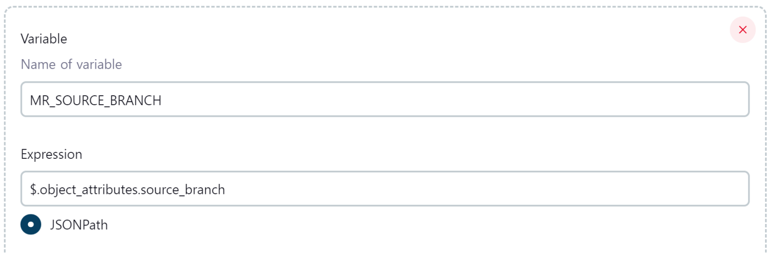
Variable
Name of variable
EVENT_TYPE

Expression
$.object_kind
[✓ JSONPath]
XPath



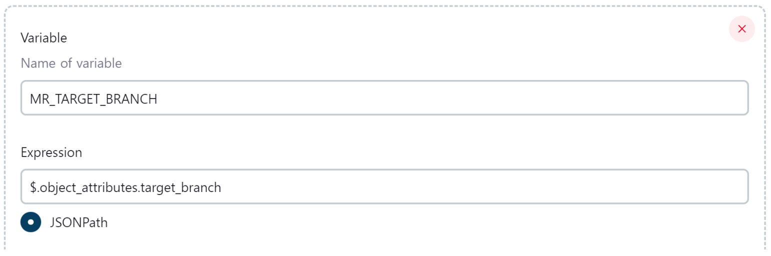
Variable
Name of variable
MR_STATE

Expression
$.object_attributes.state
[✓ JSONPath]



Variable
Name of variable
MR_SOURCE_BRANCH

Expression
$.object_attributes.source_branch
[✓ JSONPath]



Variable
Name of variable
MR_TARGET_BRANCH

Expression
$.object_attributes.target_branch
[✓ JSONPath]

▼ **Token**

`dev-backend-j10a406`

▼ **Optional Filter**

파이프라인의 트리거 조건 설정

- GitLab MR 이벤트 (그중에서도 merged 이벤트)에만
- Merge source branch가 `b-spring` (Spring), `b-recommender` (Play)인 경우만
- Merge target branch가 `dev-backend` 인 경우만

Expression : `merge_request merged (b-spring|b-recommender) dev-backend`

Text : `$EVENT_TYPE $MR_STATE $MR_SOURCE_BRANCH $MR_TARGET_BRANCH`

▼ **Pipeliine Script**

- Source branch(Spring / Play)에 따라 조건부 빌드/배포
- 수동 빌드(Jenkins UI에서 `자동 빌드` 선택) 시 Spring/Play 빌드 병렬처리

```
pipeline {
    agent any

    environment {
        imageName = "pickitup/backend-develop"
        registryCredential = 'hyunsoo31'
        dockerImage = ''

        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'j10a406.p.ssafy.io'
        releasePort = '8080'

        SOURCE_BRANCH = "${env.MR_SOURCE_BRANCH ?: 'manual-build'}"
    }

    stages {

        stage('gitlab Connect'){
            steps{
                git branch: 'dev-backend',
                credentialsId: 'gitlab-hscho',
                url: 'https://lab.ssafy.com/s10-bigdata-recom-sub2/S10P22A406.git'
            }
        }

        stage('build'){
            steps{
                script{
                    if (SOURCE_BRANCH == 'b-spring') {
                        dir('backend/spring-api') {
                            buildSpring()
                        }
                    } else if (SOURCE_BRANCH == 'b-recommender') {
                        dir('backend/play-spark') {
                            buildRecommender()
                        }
                    } else {
                        parallel springApiBuild: {
                            dir('backend/spring-api'){
                                buildSpring()
                            }
                        }, playSparkBuild: {
                            dir('backend/play-spark'){
                                buildRecommender()
                            }
                        }
                    }
                }
            }
            post {
                success {
                   echo 'Successfully Jar build'
                }
                failure {
                   error 'Jar build is failed'
                }
            }
        }


        stage('deploy'){
            steps{
                script{
                    if (SOURCE_BRANCH == 'b-spring') {
                        dir('backend/spring-api') {
                            deploySpring()
                        }
                    } else if (SOURCE_BRANCH == 'b-recommender') {
                        dir('backend/play-spark') {
                            deployRecommender()
                        }
                    } else {
                        dir('backend/spring-api') {
                            deploySpring()
                        }
                        dir('backend/play-spark') {
                            deployRecommender()
                        }
                    }
                }
            }
        }
    }
}

def buildSpring() {
    sh 'cp -r /var/jenkins_home/backend/env/env.yml /var/jenkins_home/workspace/de
    sh 'cp -r /var/jenkins_home/backend/env/env.yml /var/jenkins_home/workspace/de
    sh 'cp -r /var/jenkins_home/backend/env/application-oauth.yml /var/jenkins_hom
    sh 'cp -r /var/jenkins_home/backend/env/application-oauth.yml /var/jenkins_hom
    sh 'chmod +x gradlew'
    sh './gradlew build -x test'
}

def buildRecommender() {
    // .env 파일 프로젝트 루트로 복사
    sh 'cp -r /var/jenkins_home/backend/env/.env /var/jenkins_home/workspace/dev-b
    // Build 컨테이너 실행 후 종료 코드 확인
    def exitCode = sh script: 'docker compose -f play-spark-build-compose.yml up -
    // 종료 코드에 따라 처리
    if (exitCode == 0) {
        echo 'Build was successful, proceeding to the deploy stage.'
    } else {
        error 'play-spark build failed. stopping the pipeline...'
    }
}

def deploySpring() {
    sh 'docker stop spring-app || true'
    sh 'docker rm spring-app || true '
    sh 'docker rmi spring-api-spring-app || true'
    // sh 'docker stop spring-app && docker rm spring-app && docker rmi spring-api
    sh "docker compose up -d"
}

def deployRecommender() {
    sh 'docker compose -f play-spark-deploy-compose.yml down'
    sh 'docker compose -f play-spark-deploy-compose.yml up -d'
}
```

▼ 2. GitLab Webhook

▼ FE

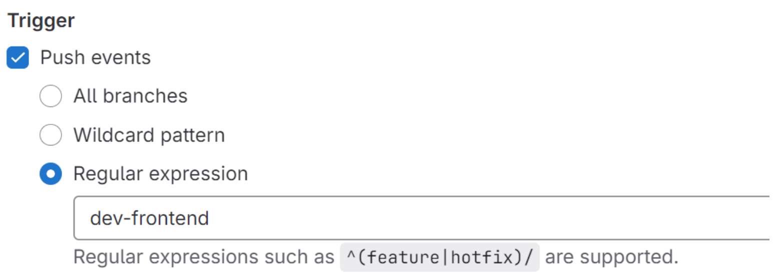
▼ URL

```
https://j10a406.p.ssafy.io/project/dev-frontend
```

▼ Secret Token

```
${GITLAB_SECRET_TOKEN}
```

▼ Trigger


Trigger

[✓] Push events

- ○ All branches
- ○ Wildcard pattern
- ● Regular expression

dev-frontend

Regular expressions such as `^(feature|hotfix)/` are supported.

▼ BE

Jenkins의 **Generic Webhook Trigger** 플러그인 URL 활용

▼ URL

```
https://j10a406.p.ssafy.io/generic-webhook-trigger/invoke?token=dev-backend-j10a40
6
```

▼ Trigger

Merge request events