# 포팅매뉴얼

## 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma

- CI/CD : Jenkins
- 모니터링: Grafana, Prometheus

## 개발 도구

- Visual Studio Code : 1.76.0
- Intellij : 2022.3.2 (Ultimate Edition)

## 개발 환경

### Server

| | |
|---|---|
| AWS S3 | |
| AWS EC2 | CPU : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz<br>RAM : 16GB<br>OS: Ubuntu 20.04 LTS |

### FrontEnd

| | |
|---|---|
| React | 2.1.0 |
| TypeScript | 4.9.5 |
| Styled-components | 6.1.8 |
| Zustand | 4.4.7 |

### BackEnd

| | |
|---|---|
| Java | 17 |
| Spring | 3.2.1 |
| Postman | v10.22 |
| springdoc | 2.0.2 |

### DB

| | |
|---|---|
| MariaDB | Ver 15.1 Distrib 10.11.6-MariaDB |
| MongoDB | 7.0.5 |
| Redis | 7.2.4 |
| influxDB | 2.7.5 |

### Service

| | |
|---|---|
| RabbitMQ | 3.12.12 |
| Jenkins | 2.426.2 |
| Docker | 25.0.0 |
| Nginx | nginx/1.18.0 (Ubuntu) - local<br>nginx/1.25.3 - docker |
| Grafana | 10.2.3 |
| Prometheus | 2.49.1 |
| Sonarqube | 9.9.3 |

## 환경변수 형태

### Backend

- application.yml

[ backend ]

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://i10a610.p.ssafy.io:3306/togeball
    username: ${MARIADB_USERNAME}
    password: ${MARIADB_PASSWORD}
  config:
    import:
      - optional:properties/jpa.yml
      - optional:env/env.yml

  security:
    oauth2:
      client:
        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
          google:
            authorization-uri: https://accounts.google.com/o/oauth2/v2/auth
            token-uri: https://oauth2.googleapis.com/token
            user-info-uri: https://www.googleapis.com/oauth2/v3/userinfo
            user-name-attribute: sub

        registration:
          google:
            client-id: ${GOOGLE_CLIENT_ID}
            client-secret: ${GOOGLE_CLIENT_SECRET}
            redirect-uri: http://localhost:3000/login/oauth2/code/google
            authorization-grant-type: authorization_code
            scope: profile,email
          kakao:
            client-id: ${KAKAO_CLIENT_ID}
            client-secret: ${KAKAO_CLIENT_SECRET}
            redirect-uri: https://i10a610.p.ssafy.io/login/kakao
            authorization-grant-type: authorization_code
            scope:
              - profile_nickname
              - account_email

server:
  servlet:
    encoding:
      charset: UTF-8
      enabled: true
      force: true
  port: 8080
  ssl:
    enabled: true
    enabled-protocols:
      - TLSv1.1
      - TLSv1.2
    key-store: "classpath:ssl/keystore.p12"
    key-store-password: ${SSL_KEYSTORE_PASSWORD}
    key-store-type: "PKCS12"

jwt:
  secretKey: ${JWT_SECRET_KEY}
  access:
    expiration: 10800000
    header: Authorization
  refresh:
    expiration: 1209600000
    header: Authorization-refresh

rabbitmq:
  host: i10a610.p.ssafy.io
  port: 4672
  username: ${RABBITMQ_USERNAME}
  password: ${RABBITMQ_PASSWORD}
  exchange: togeball.exchange
  chat:
    queue: chat.queue
    routing-key: togeball.chat
  notification:
    chat:
      queue: notification.chat.queue
    matching:
      queue: notification.matching.queue

cloud:
  aws:
    credentials:
      bucket-name: togeball-s3-bucket
      expiration: 3600000
      access-key: ${S3_ACCESS_KEY}
      secret-key: ${S3_SECRET_KEY}
    region:
      static: ap-northeast-2

springdoc:
  swagger-ui:
    groups-order: DESC
    tags-sorter: alpha
    operations-sorter: method
    disable-swagger-default-url: true
    display-request-duration: true
    defaultModelsExpandDepth: 2
    defaultModelExpandDepth: 2
  api-docs:
    path: /api-docs
  show-actuator: true
  default-consumes-media-type: application/json
  default-produces-media-type: application/json
  writer-with-default-pretty-printer: true
  model-and-view-allowed: true
  paths-to-match:
    - /api/**
```

togeball-chat

```yaml
server:
  servlet:
    encoding:
      charset: UTF-8
      enabled: true
      force: true
  port: 8080
  ssl:
    enabled: true
    enabled-protocols:
      - TLSv1.1
      - TLSv1.2
    key-store: "classpath:ssl/keystore.p12"
    key-store-password: "togeball"
    key-store-type: "PKCS12"

spring:
  data:
    mongodb:
      host: i10a610.p.ssafy.io
      port: 27017
      username: ${MONGO_INITDB_ROOT_USERNAME}
      password: ${MONGO_INITDB_ROOT_PASSWORD}
      authentication-database: admin
      database: togeball_chat_3
  rabbitmq:
    host: i10a610.p.ssafy.io
    port: 4672
    username: ssafy
    password: ssafy
  cloud:
    aws:
      credentials:
        access-key: ${S3_ACCESS_KEY}
        secret-key: ${S3_SECRET_KEY}
      region:
        static: ap-northeast-2
      s3:
        bucket: togeball-s3-bucket
  servlet:
    multipart:
      max-file-size: 100MB
  config:
    import: optional:env/env.yml

logging:
  level:
    org:
      springframework:
        messaging: DEBUG
        web:
          socket: DEBUG
    io:
      awspring:
        cloud: DEBUG

websocket:
  relay:
    host: i10a610.p.ssafy.io
    port: 61613
    client:
      login: ssafy
      passcode: ssafy
    system:
      login: ssafy
      passcode: ssafy

rabbitmq:
  host: i10a610.p.ssafy.io
  port: 4672
  username: [User]
  password: [User Password]
  exchange:
    name: togeball.exchange
  join:
    queue: chat.queue
  notification:
    queue: notification.chat.queue
    routing-key: togeball.notification

jwt:
  secretKey: ${JWT_SECRET_KEY}
  access:
    header: Authorization
```

togeball-matching

```
server:
  servlet:
    encoding:
      charset: UTF-8
      enabled: true
      force: true
  port: 8080
  ssl:
    enabled: true
    enabled-protocols:
      - TLSv1.1
      - TLSv1.2
    key-store: "classpath:ssl/keystore.p12"
    key-store-password: ${SSL_KEYSTORE_PASSWORD}
    key-store-type: "PKCS12"

spring:
  data:
    redis:
      host: i10a610.p.ssafy.io
      port: 6379
      password: ${REDIS_PASSWORD}
  config:
    import: optional:env/env.yml

rabbitmq:
  host: i10a610.p.ssafy.io
  port: 4672
```

```
    username: ${RABBITMQ_USERNAME}
    password: ${RABBITMQ_PASSWORD}
    exchange: togeball.exchange
    matching:
      queue: matching.notification.queue
      routing-key: togeball.matching

openai:
  api:
    key: ${GPT_API_KEY}
```

- env

```
S3_ACCESS_KEY: "S3 ACESS KEY"
S3_SECRET_KEY: "S3 Password"
JWT_SECRET_KEY: "jwt Password"
GOOGLE_CLIENT_ID: "Google ID"
GOOGLE_CLIENT_SECRET: "Google Password"
KAKAO_CLIENT_ID: "KAKAO ID"
KAKAO_CLIENT_SECRET: "KAKAO Password"
RABBITMQ_USERNAME: "RABBITMQ ID"
RABBITMQ_PASSWORD: "RABBITMQ Password"
REDIS_PASSWORD: "REDIS Password"
MARIADB_USERNAME: "MARIADB ID"
MARIADB_PASSWORD: "MARIADB Password"
MONGO_INITDB_ROOT_USERNAME: "MONGO ID"
MONGO_INITDB_ROOT_PASSWORD: "MONGO Password"
SSL_KEYSTORE_PASSWORD: "PKCS12"
GPT_API_KEY: "GPY KEY"
```

### Frontend

- .env

```
REACT_APP_BASE_URL = "BACKEND SERVER URL"

REACT_APP_REST_API_KEY = "REACT PASSWORD"
REACT_APP_REDIRECT_URI = "KAKAO RERIRECT URL"
```

# EC2 인스턴스 초기 설정

## swap 설정

- 디스크 용량 확인 및 스왑 영역 설정
  - `df -h` 확인 → 디스크 용량이 많아서 6G 정도 진행

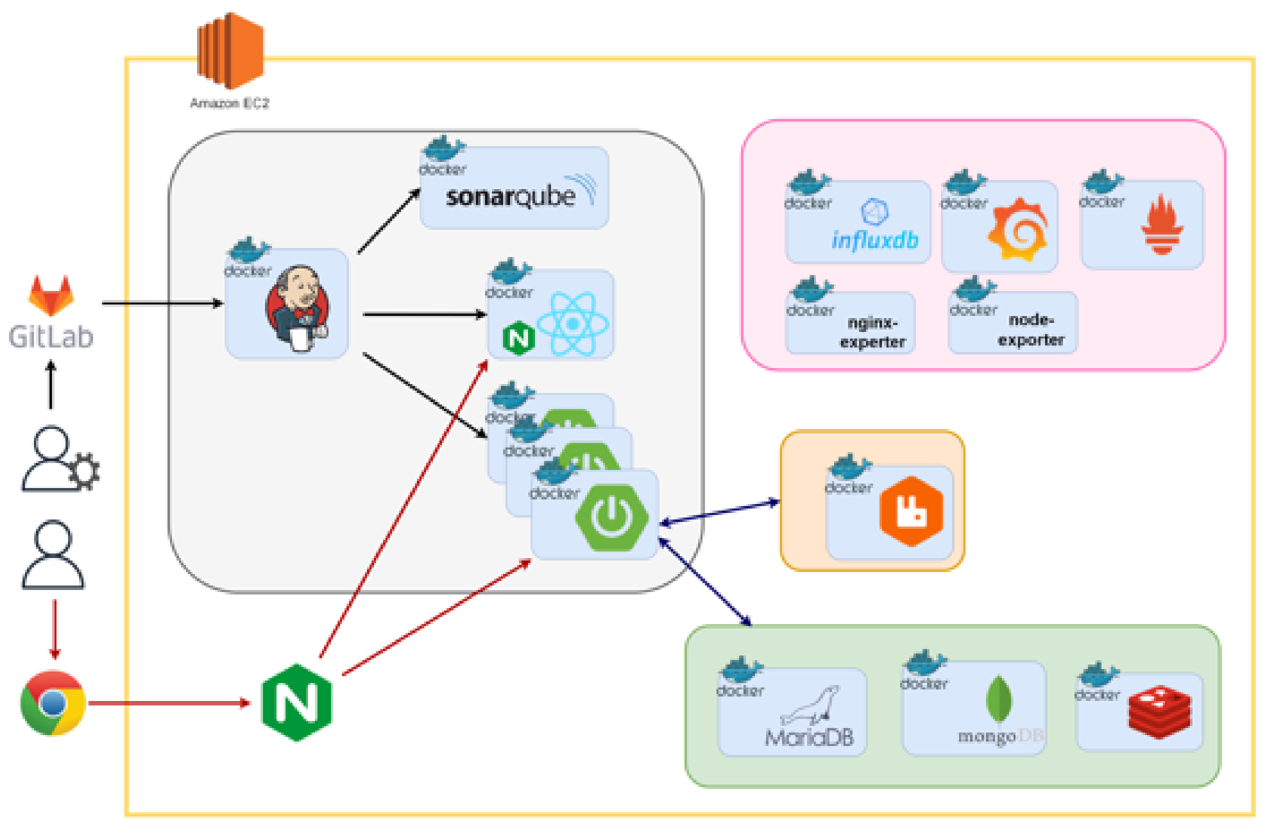
```
fallocate -l 6G /swapfile
chmod 600 /swapfile
mkswap /swapfile
swapon /swapfile
# fstab 파일에 시작할 때 마운트할 공간 저장
```

## 계정 접근

- id, pw로 접속 허용하는 과정 진행

```
sudo apt update
sudo apt install httpd net-tools

sudo passwd root
su root

cd /etc
chmod 660 sudoers
vi sudoers
chmod 440 sudoers

adduser [New Id]
passwd [New Id Pw]
su [New Id]

cd /etc/ssh
sudo vi sshd_config     # PasswordAuthentication-> yes
sudo service sshd restart
```

## Architecture 설계



## 사용 포트 정보

```
22: ssh
80: nginx - proxy
443: SSL

3306: mariaDB
3333: sonarqube

4672: rabbitmq
6379: redis

8000: nginx -react
```

```
8900: grafana
8901: prometheus
8902: node-expoter
8903: nginx-exporter

8989: gerrit

16672: rabbitmq
27017: mongodb
61613: rabbitmq
```

```
8080: spring port-forwarding
8081: spring (api-server)
8082: spring (chat-server)
8083: spring (matching-server)
8086: influxdb
8888: jenkins
```

## 포트 상세

### [ Domain ]

i10a610.p.ssafy.io
(172.26.13.191)

### [ LoadBalancing ]

**Nginx 80**

https://i10a610.p.ssafy.io

### [ CI/CD ]

**Jenkins 8888:8080**

http://i10a610.p.ssafy.io:8888/

**Spring 8080:8080**

http://i10a610.p.ssafy.io:8888/

**Nginx 8081:80**

http://i10a610.p.ssafy.io:8081/

### [ DB ]

**MaraiDB 3306:3306**

http://i10a610.p.ssafy.io:3306

**redis 6379:6379**

**mongoDB 27017:27017**

http://i10a610.p.ssafy.io:27017

**influx 8086:8086**

### [ Monitoring ]

**Grafana 8900:3000**

http://i10a610.p.ssafy.io:8900

**influxDB 8086:8086**

http://i10a610.p.ssafy.io:8086/

**Prometheus 8901:9090**

http://i10a610.p.ssafy.io:8901

**Node-exporter8902:9100**

http://i10a610.p.ssafy.io:8902

**Nginx-prometheus-exporter 8903:9113**

http://i10a610.p.ssafy.io:8903

### [ QA ]

**sonarqube 3333:9000**

http://i10a610.p.ssafy.io:8989

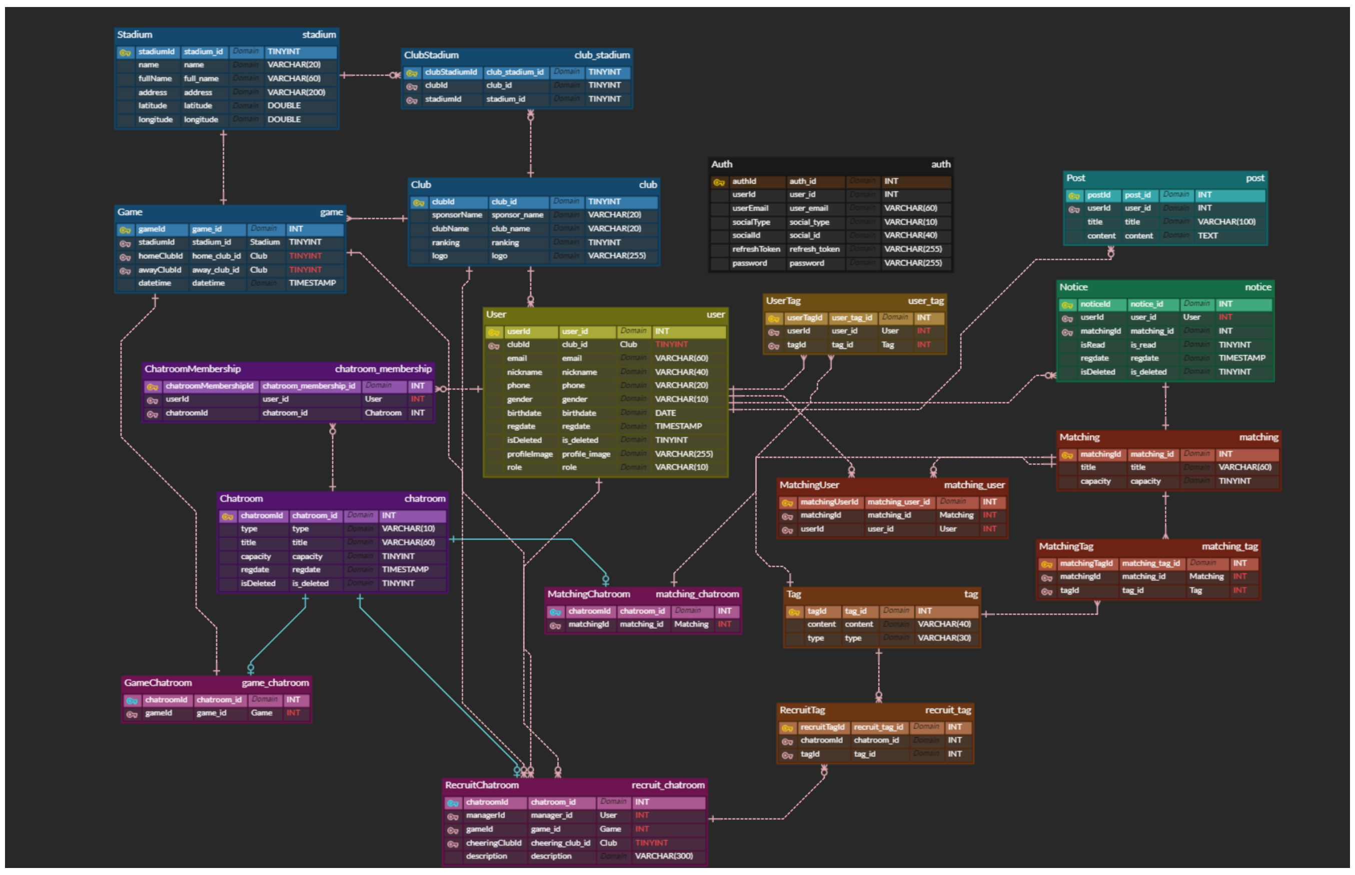
### [ Message Broker ]

**rabbitmq 4672:5672 16672:15672 61613:61613**

**redis 6379:6379**

togeball

## ERD



| Stadium | | | stadium |
|---|---|---|---|
| stadiumId | stadium_id | Domain | TINYINT |
| name | name | Domain | VARCHAR(20) |
| fullName | full_name | Domain | VARCHAR(60) |
| address | address | Domain | VARCHAR(200) |
| latitude | latitude | Domain | DOUBLE |
| longitude | longitude | Domain | DOUBLE |

| ClubStadium | | | club_stadium |
|---|---|---|---|
| clubStadiumId | club_stadium_id | Domain | TINYINT |
| clubId | club_id | Domain | TINYINT |
| stadiumId | stadium_id | Domain | TINYINT |

| Game | | | game |
|---|---|---|---|
| gameId | game_id | Domain | INT |
| stadiumId | stadium_id | Stadium | TINYINT |
| homeClubId | home_club_id | Club | TINYINT |
| awayClubId | away_club_id | Club | TINYINT |
| datetime | datetime | Domain | TIMESTAMP |

| Club | | | club |
|---|---|---|---|
| clubId | club_id | Domain | TINYINT |
| sponsorName | sponsor_name | Domain | VARCHAR(20) |
| clubName | club_name | Domain | VARCHAR(20) |
| ranking | ranking | Domain | TINYINT |
| logo | logo | Domain | VARCHAR(255) |

| Auth | | | auth |
|---|---|---|---|
| authId | auth_id | Domain | INT |
| userId | user_id | Domain | INT |
| userEmail | user_email | Domain | VARCHAR(60) |
| socialType | social_type | Domain | VARCHAR(10) |
| socialId | social_id | Domain | VARCHAR(40) |
| refreshToken | refresh_token | Domain | VARCHAR(255) |
| password | password | Domain | VARCHAR(255) |

| Post | | | post |
|---|---|---|---|
| postId | post_id | Domain | INT |
| userId | user_id | Domain | INT |
| title | title | Domain | VARCHAR(100) |
| content | content | Domain | TEXT |

| User | | | user |
|---|---|---|---|
| userId | user_id | Domain | INT |
| clubId | club_id | Club | TINYINT |
| email | email | Domain | VARCHAR(60) |
| nickname | nickname | Domain | VARCHAR(40) |
| phone | phone | Domain | VARCHAR(20) |
| gender | gender | Domain | VARCHAR(10) |
| birthdate | birthdate | Domain | DATE |
| regdate | regdate | Domain | TIMESTAMP |
| isDeleted | is_deleted | Domain | TINYINT |
| profileImage | profile_image | Domain | VARCHAR(255) |
| role | role | Domain | VARCHAR(10) |

| UserTag | | | user_tag |
|---|---|---|---|
| userTagId | user_tag_id | Domain | INT |
| userId | user_id | User | INT |
| tagId | tag_id | Tag | INT |

| Notice | | | notice |
|---|---|---|---|
| noticeId | notice_id | Domain | INT |
| userId | user_id | User | INT |
| matchingId | matching_id | Domain | INT |
| isRead | is_read | Domain | TINYINT |
| regdate | regdate | Domain | TIMESTAMP |
| isDeleted | is_deleted | Domain | TINYINT |

| ChatroomMembership | | | chatroom_membership |
|---|---|---|---|
| chatroomMembershipId | chatroom_membership_id | Domain | INT |
| userId | user_id | User | INT |
| chatroomId | chatroom_id | Chatroom | INT |

| Matching | | | matching |
|---|---|---|---|
| matchingId | matching_id | Domain | INT |
| title | title | Domain | VARCHAR(60) |
| capacity | capacity | Domain | TINYINT |

| MatchingUser | | | matching_user |
|---|---|---|---|
| matchingUserId | matching_user_id | Domain | INT |
| matchingId | matching_id | Matching | INT |
| userId | user_id | User | INT |

| Chatroom | | | chatroom |
|---|---|---|---|
| chatroomId | chatroom_id | Domain | INT |
| type | type | Domain | VARCHAR(10) |
| title | title | Domain | VARCHAR(60) |
| capacity | capacity | Domain | TINYINT |
| regdate | regdate | Domain | TIMESTAMP |
| isDeleted | is_deleted | Domain | TINYINT |

| MatchingTag | | | matching_tag |
|---|---|---|---|
| matchingTagId | matching_tag_id | Domain | INT |
| matchingId | matching_id | Matching | INT |
| tagId | tag_id | Tag | INT |

| MatchingChatroom | | | matching_chatroom |
|---|---|---|---|
| chatroomId | chatroom_id | Domain | INT |
| matchingId | matching_id | Matching | INT |

| Tag | | | tag |
|---|---|---|---|
| tagId | tag_id | Domain | INT |
| content | content | Domain | VARCHAR(40) |
| type | type | Domain | VARCHAR(30) |

| GameChatroom | | | game_chatroom |
|---|---|---|---|
| chatroomId | chatroom_id | Domain | INT |
| gameId | game_id | Game | INT |

| RecruitTag | | | recruit_tag |
|---|---|---|---|
| recruitTagId | recruit_tag_id | Domain | INT |
| chatroomId | chatroom_id | Domain | INT |
| tagId | tag_id | Domain | INT |

| RecruitChatroom | | | recruit_chatroom |
|---|---|---|---|
| chatroomId | chatroom_id | Domain | INT |
| managerId | manager_id | User | INT |
| gameId | game_id | Game | INT |
| cheeringClubId | cheering_club_id | Club | TINYINT |
| description | description | Domain | VARCHAR(300) |

# Docker 설치

## 도커 설치

```
sudo apt-get update

sudo apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
         | sudo apt-key add -
sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux \
        /ubuntu $(lsb_release -cs) stable"

sudo apt-get update
sudo apt-get install net-tools

sudo apt-get install -y docker-ce docker-ce-cli containerd.io \
    docker-compose docker-compose-plugin

systemctl enable docker
systemctl status docker
```

## 젠킨스 컨테이너 생성

- jenkins/jenkins:lts 컨테이너 생성
    - JDK 17로 작업하기 위해 JAVA 설치 및 JAVA_HOME 환경 변수 생성
    - 컨테이너 데이터 유지를 위한 마운트 & DooD 방식을 위한 소켓 마운트
    - jenkins 유저가 default이기 때문에 root 유저로 생성

```
mkdir -p /var/jenkins_home

chown -R 1000:1000 /var/jenkins_home/

docker run --restart=on-failure --user='root' \
    -p 8888:8080 -p 50000:50000 \
    --env JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64 \
    -v /var/jenkins_home:/var/jenkins_home \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -d --name jenkins jenkins/jenkins:lts \
```

- 젠킨스 환경 구축
  - 로컬과 마찬가지로 설정해준다(컨테이너 OS가 데비안인 것에 주의)

```
apt-get update
apt-get install openjdk-17-jdk -y

apt-get install -y \
apt-transport-https \
ca-certificates \
curl \
gnupg2 \
software-properties-common

curl -fsSL https://download.docker.com/linux/debian/gpg \
        | apt-key add -

add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/debian \
$(lsb_release -cs) \
stable"

apt-get update
apt-get install docker-ce docker-ce-cli containerd.io
```

리액트 프로젝트를 빌드할 것이므로 node설치 (최신 버전은 apt로 설치 불가)

```
curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key\
 | gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg
export NODE_MAJOR=20
sudo tee /etc/apt/sources.list.d/nodesource.list
sudo apt update && sudo apt install nodejs -y
```

## Nginx 구성

- 역할
  - **로드밸런싱, 웹서버**

    로드밸런싱 역할을 하는 서버는 로컬, 웹서버 역할을 하는 서버는 도커로 구분하여 설치

### 웹서버

- 도커 컨테이너를 이용해 nginx 생성 (debian ver)
- 기본 root디렉토리가 /usr/share/nginx로 되어 있음.

이 위치에 프론트 프로젝트의 빌드된 build 폴더를 옮겨줄 것이므로 /usr/share/nginx/build로 기본 경로를 바꿔준다.

```
docker run --restart=on-failure -p 8000:80 -d --name nginx nginx
```

- /etc/nginx/conf.d

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

    location / {
        root   /usr/share/nginx/html/build/;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

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

## 로드밸런싱

- **nginx 설치**

```
apt install nginx
```

- **SSL 인증 받기**
  - Certbot 설치

    ```
    sudo snap install --classic certbot

    sudo ln -s /snap/bin/certbot /usr/bin/certbot
    apt install letsencrypt

    sudo apt-add-repository -r ppa:certbot/certbot
    sudo apt-get -y install python3-certbot-nginx
    ```

  - SSL 인증서 받기

    ```
    sudo certbot --nginx
    # 이메일 입력 > N > Domain 작성
    ```

- 리버스 프록시 설정
  - /etc/nginx/conf.d/service-url.inc

    ```
    set $service_url http://127.0.0.1:8000;
    ```

- /etc/nginx/sites-enabled/default
  - 모든 api 요청은 8080 포트로 들어온 후 포트 포워딩
  - CORS policy를 위한 8080 ssl proxy 사용
  - 엔드포인트 구분
    - /matching-server : 매칭 관련 api 요청 처리 서버
    - /chat-server: 채팅 관련 api 요청 처리 서버

- /sse/notification/subscribe: 알람 관련 api 요청 처리
- 나머지 api 요청은 8081 서버로 이동

```
# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;
        index index.html index.htm;

        server_name _;
        include /etc/nginx/conf.d/service-url.inc;

        location / {
                proxy_pass $service_url;
        }
}

server {
       listen 8080 ssl;
       listen [::]:8080 ssl;

       server_name i10a610.p.ssafy.io;

       ssl_certificate /etc/letsencrypt \
                    /live/i10a610.p.ssafy.io/fullchain.pem;
       ssl_certificate_key /etc/letsencrypt \
                    /live/i10a610.p.ssafy.io/privkey.pem;
       include /etc/letsencrypt/options-ssl-nginx.conf;
       ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

       location / {
               proxy_pass  https://localhost:8081;
               proxy_set_header Host $http_host;
               proxy_set_header X-Forwarded-For \
                                $proxy_add_x_forwarded_for;
               proxy_set_header X-Real-IP $remote_addr;
       }

        location /sse/notification/subscribe {
                proxy_pass https://localhost:8081;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-For \
                                    $proxy_add_x_forwarded_for;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header Connection '';
                proxy_buffering off;
                proxy_read_timeout 1800;
                proxy_http_version 1.1;
        }

       location /chat-server {
                proxy_pass https://localhost:8082;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-For \
                                        $proxy_add_x_forwarded_for;
                proxy_set_header X-Real-IP $remote_addr;
        }

        location /matching-server {
                proxy_pass https://localhost:8083;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-For \
                                        $proxy_add_x_forwarded_for;
                proxy_set_header X-Real-IP $remote_addr;
        }

}

server {

    server_name i10a610.p.ssafy.io;

        include /etc/nginx/conf.d/service-url.inc;

        location / {
                proxy_pass $service_url;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-For \
                                    $proxy_add_x_forwarded_for;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-Proto $scheme;
        }



    listen [::]:443 ssl ipv6only=on;
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live \
                    /i10a610.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live \
                    /i10a610.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

}
server {
    if ($host = i10a610.p.ssafy.io) {
        return 308 https://$host$request_uri;
    }
        listen 80 ;
        listen [::]:80 ;
    server_name i10a610.p.ssafy.io;
    return 404;
}
```

config 수정 → `nginx -t` → `systemctl restart nginx`

## Spring SSL

- PKCS12 생성

    - /etc/letsencrypt/live/<도메인> 아래에 있는 fullchain.pem과 privkey.perm을 묶어서 스프링 프로젝트에 적용할 pkcs12 형식의 파일을 만든다.

```
sudo openssl pkcs12 -export -in fullchain.pem \
            -inkey privkey.pem -out keystore.p12 \
            -name ttp -CAfile chain.pem -caname root
# 패스워드 생성
```

- SSL/TLS 인증서 설정
    - resources/ssl 경로에 keystore.p12 파일 생성
    - application.yml

```
server:
  servlet:
    encoding:
      charset: UTF-8
      enabled: true
      force: true
  port: 8080
  ssl:
    enabled: true
    enabled-protocols:
      - TLSv1.1
      - TLSv1.2
    key-store: "classpath:ssl/keystore.p12"
    key-store-password: [키 만들 때 입력한 패스워드]
    key-store-type: "PKCS12"
```

## DB 생성

### MariaDB

- 도커 볼륨 생성

```
docker volume create mariadb -volume
```

- 도커 볼륨 조회

```
docker volume ls
```

- DB 실행

```
docker run -d --restart=on-failure \
    -p 3306:3306 --name mariadb \
    --env MARIADB_ROOT_PASSWORD=[Password] \
    -v maraidb:/var/libs/mariadb mariadb:lts
docker exec -it mariadb mariadb -u root -p
```

### MongoDB

- 단순 읽기/쓰기 작업을 주로 수행하는 채팅 메시지 데이터 저장
- docker-compose

```
mongodb:
    image: "mongo"
    environment:
      MONGO_INITDB_ROOT_USERNAME: [User]
      MONGO_INITDB_ROOT_PASSWORD: [User Password]
      MONGO_INITDB_DATABASE: [Init DB]
    ports:
      - "27017:27017"
    volumes:
      - "mongodb_data:/data/db"
    restart: on-failure
```

## Redis

- 매칭 대기열에 접속 중인 회원 정보 인스턴스 관리
- 볼륨 생성

```
docker volume create redisdb
```

- redis config 생성 (/etc/redis/redis.conf)

```
bind 0.0.0.0

port 6379

requirepass [사용하고자 하는 비밀번호]

maxmemory 1g

maxmemory-policy volatile-ttl


save 900 1
save 300 10
save 60 10000
```

- 실행

```
docker run \
-d \
--restart=on-failure \
--name=redis \
-p 6379:6379 \
-e TZ=Asia/Seoul \
-v /etc/redis/redis.conf:/etc/redis/redis.conf \
-v redisdb:/data \
redis:latest --requirepass [Password]
```

## InfluxDB

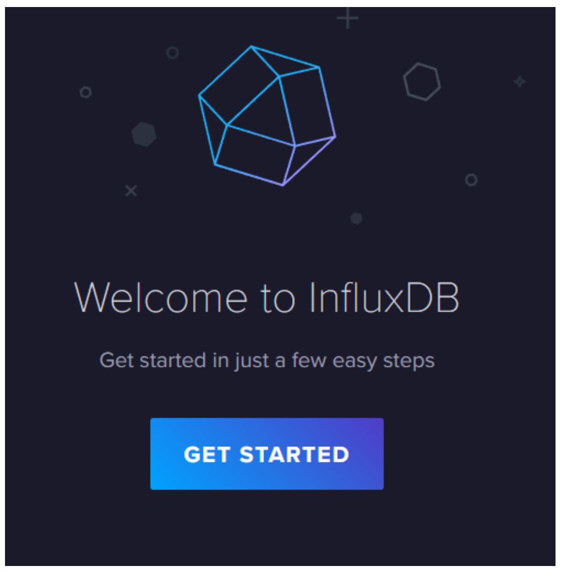
- 그라파나 젠킨스 연동을 위해 influxdb 설치

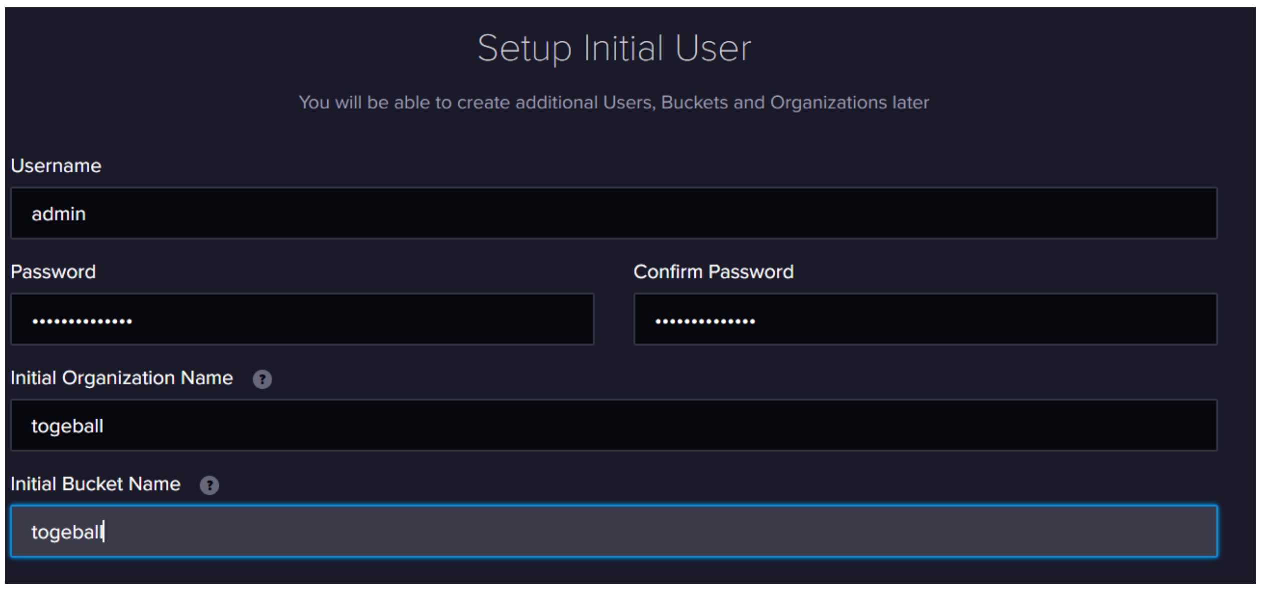
```
docker volume create influxdb_data
```

- 도커 컨테이너 실행

```
docker run -d -p 8086:8086 \
--restart=on-failure \
-v /var/lib/influxdb:/var/lib/influxdb2 \
-e DOCKER_INFLUXDB_INIT_USERNAME=[User] \
-e DOCKER_INFLUXDB_INIT_PASSWORD=[Password] \
-e DOCKER_INFLUXDB_INIT_ORG=[Organization] \
-e DOCKER_INFLUXDB_INIT_BUCKET=[Bucket] \
-e DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=[Token] \
--name influxdb influxdb
```

접속해서 정보 입력




Setup Initial User

You will be able to create additional Users, Buckets and Organizations later

Username: admin

Password: ••••••••••••••

Confirm Password: ••••••••••••••

Initial Organization Name: togeball

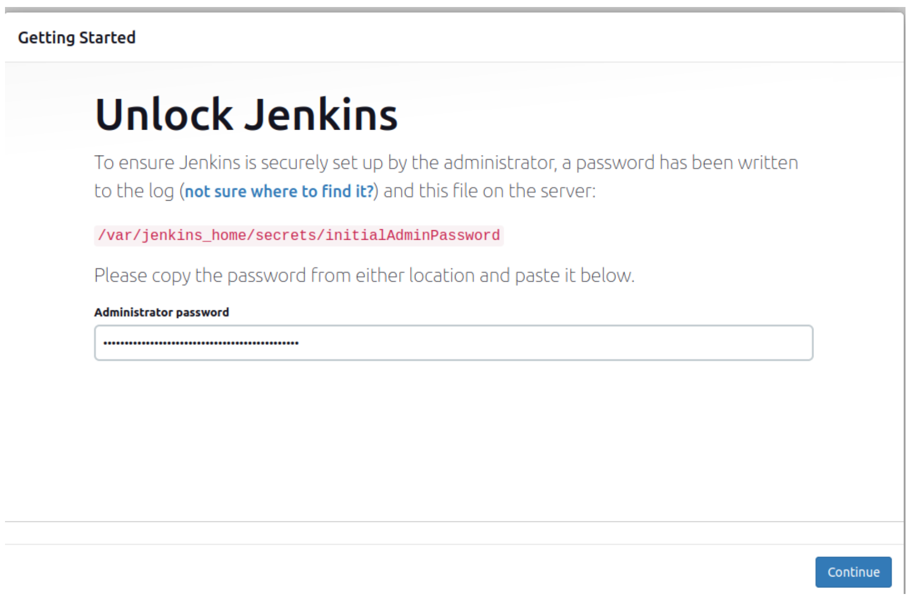
Initial Bucket Name: togeball

# Jenkins 구축

## Jenkins 초기 설정

### 플러그인 설치

젠킨스 접속 후 초기 패스워드 입력하고 플러그인 설치


Getting Started

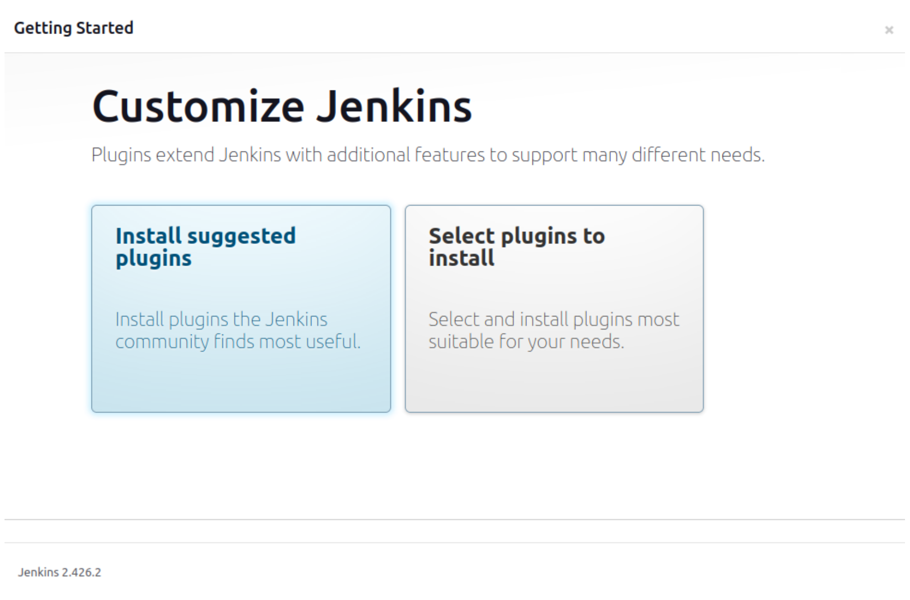
Unlock Jenkins

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


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.

Jenkins 2.426.2

**Gitlab**의 경우 먼저 아래의 깃랩 플러그인 설치(Manage > Plugins)

**GitLab API Plugin** **GitLab Plugin** **GitLab Authentication plugin**

다음으로 프로젝트에서 사용하는 추가 플러그인 설치

**Mattermost Notification Plugin**

**Generic Webhook Trigger**

**NodeJS Plugin**

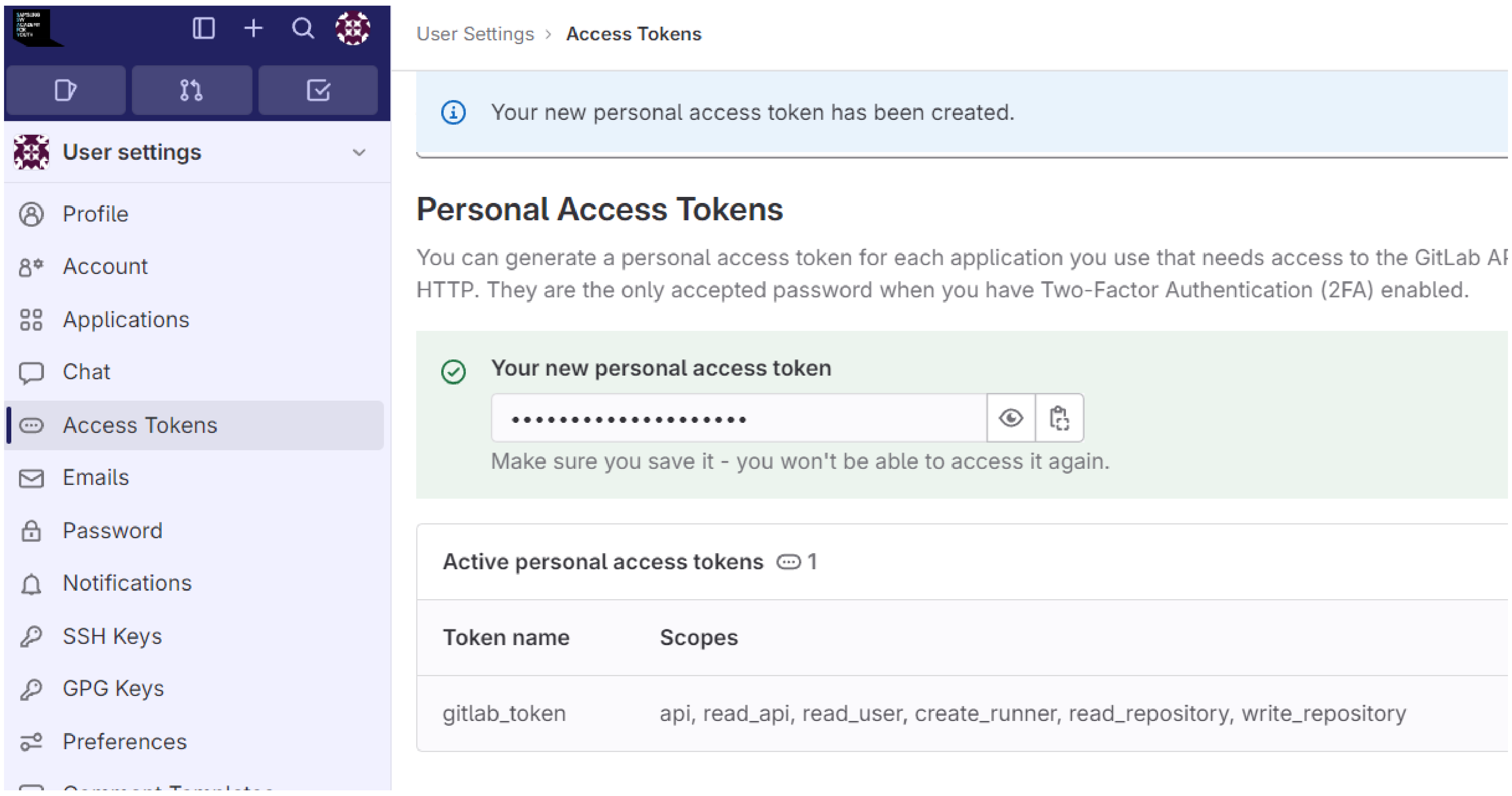
**Prometheus metrics plugin**

**SonarQube Scanner for Jenkins**

## 계정 생성

GitLab > Edit profile > Access Tokens에서 토큰을 발급 받아 토큰 값 저장

이 값을 Password로 입력하여 Jenkins Credential 생성(System > Credential > Add User 에서 Kind 부분을 Gitlab Api Token으로 입력)



Your new personal access token has been created.

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab AP
HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Your new personal access token

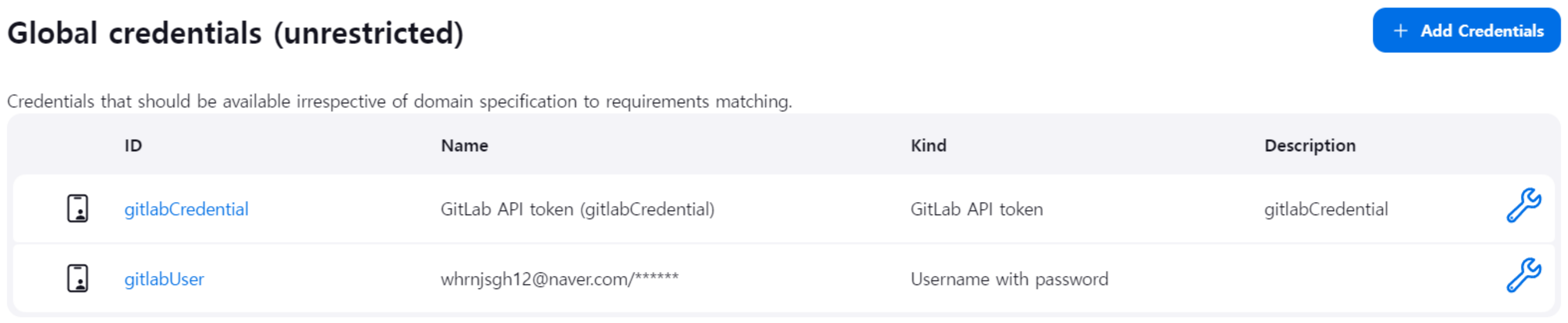
••••••••••••••••••••

Make sure you save it - you won't be able to access it again.

Active personal access tokens 1

| Token name | Scopes |
|---|---|
| gitlab_token | api, read_api, read_user, create_runner, read_repository, write_repository |

GitLab도 GitHub와 마찬가지로 api token으로 만든 계정으로는 Git에 대한 push, pull, clone 등이 안 되기 때문에 GitLab 2차 패스워드를 pw로 가지는 계정을 하나 더 생성



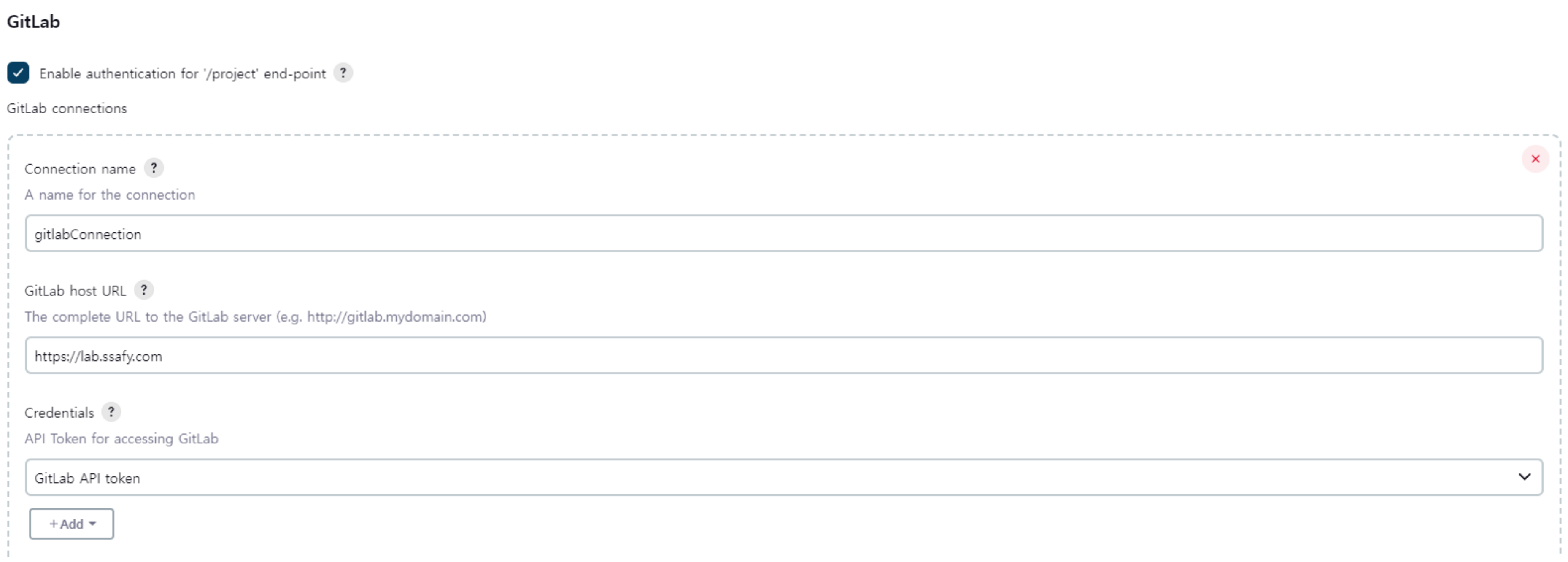
Global credentials (unrestricted)

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

| ID | Name | Kind | Description |
|---|---|---|---|
| gitlabCredential | GitLab API token (gitlabCredential) | GitLab API token | gitlabCredential |
| gitlabUser | whrnjsgh12@naver.com/****** | Username with password | |

마지막으로 API token을 가지는 계정을 기반으로 깃랩 연결을 젠킨스에 등록

System> GitLab에서 아래와 같이 hostURL과 Credential을 맞춰주고 저장


GitLab

✓ Enable authentication for '/project' end-point

GitLab connections

Connection name
A name for the connection
gitlabConnection

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

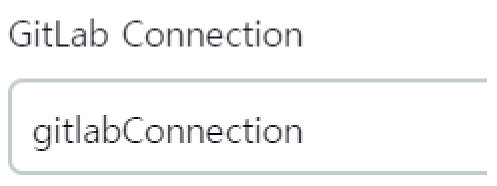
이렇게 하면 GitLab의 계정에 대해 젠킨스에서 해당 계정의 정보 지님

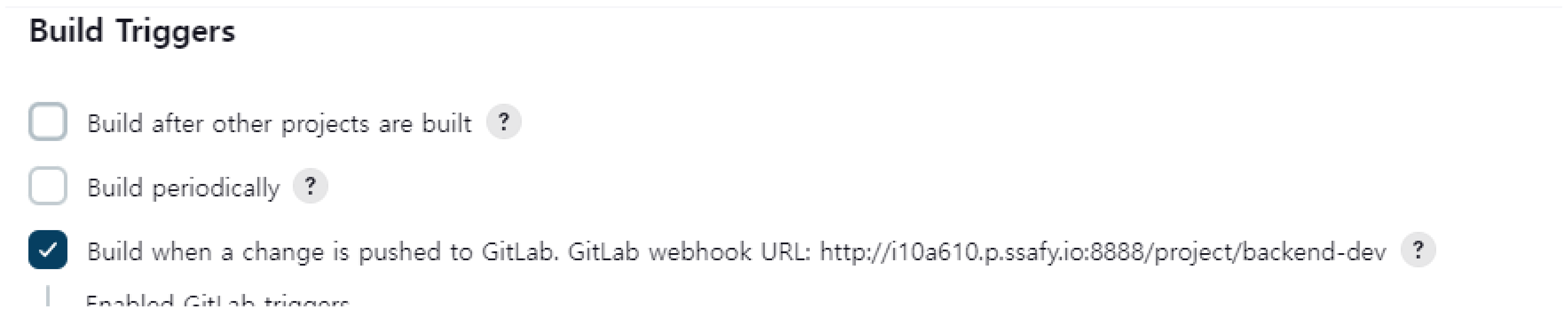
### 파이프 라인 생성

파이프라인 잡을 생성하면 크게 세 부분으로 나뉜다.

**General, Advanced Project Options, Pipeline**

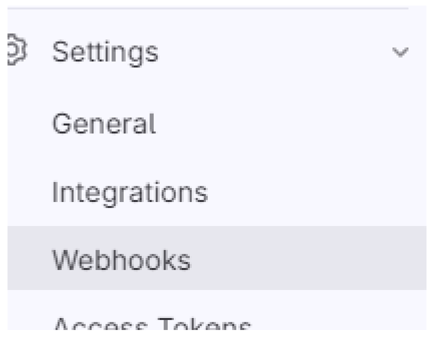
- General

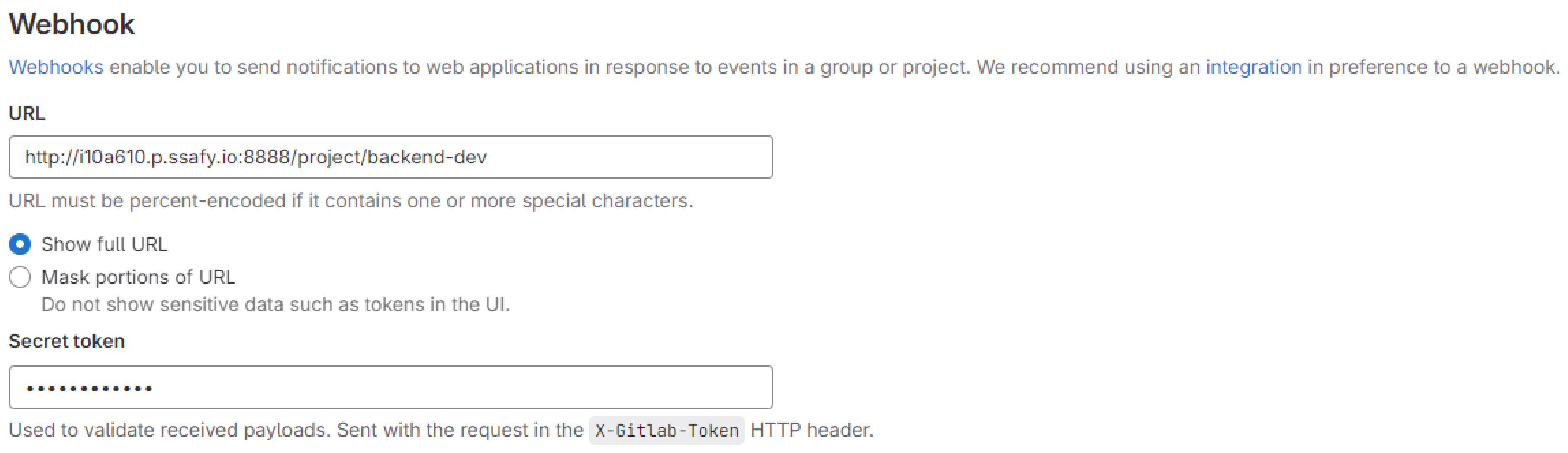

GitLab Connection
gitlabConnection


Build Triggers

☐ Build after other projects are built
☐ Build periodically
✓ Build when a change is pushed to GitLab. GitLab webhook URL: http://i10a610.p.ssafy.io:8888/project/backend-dev

**Enabled GitLab triggers**에서 **Accepted Merge Request Events** 선택
Secret token을 Generate하고 깃 랩에 연결




Webhook

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://i10a610.p.ssafy.io:8888/project/backend-dev
URL must be percent-encoded if it contains one or more special characters.

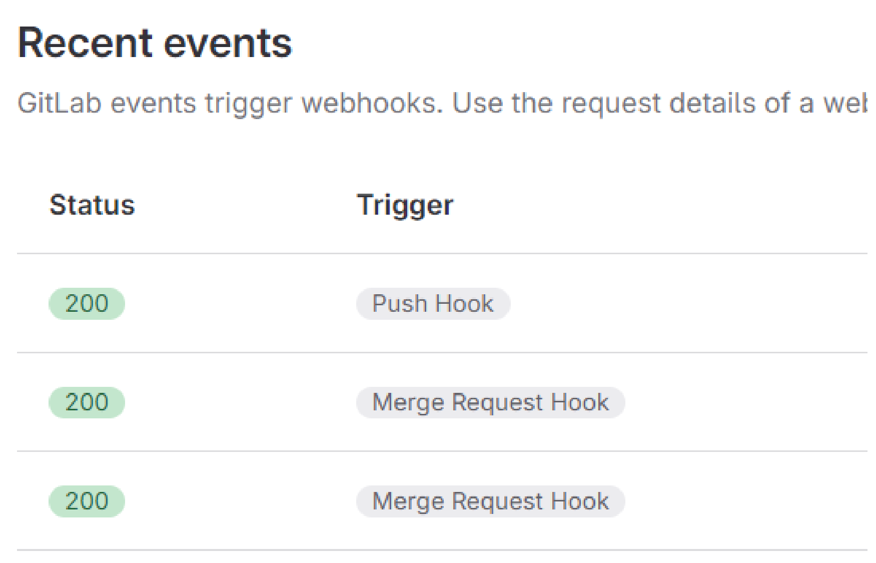
● Show full URL
○ Mask portions of URL
Do not show sensitive data such as tokens in the UI.

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

[GitLab] Settings> Webhooks > Add WebHooks

URL에는 젠킨스 BuildTrigger에서 보여지는 깃랩 주소를 작성

그리고 토큰에는 젠킨스에서 생성한 토큰을 넣어주고 웹훅 완성



Recent events

GitLab events trigger webhooks. Use the request details of a wel

| Status | Trigger |
|---|---|
| 200 | Push Hook |
| 200 | Merge Request Hook |
| 200 | Merge Request Hook |

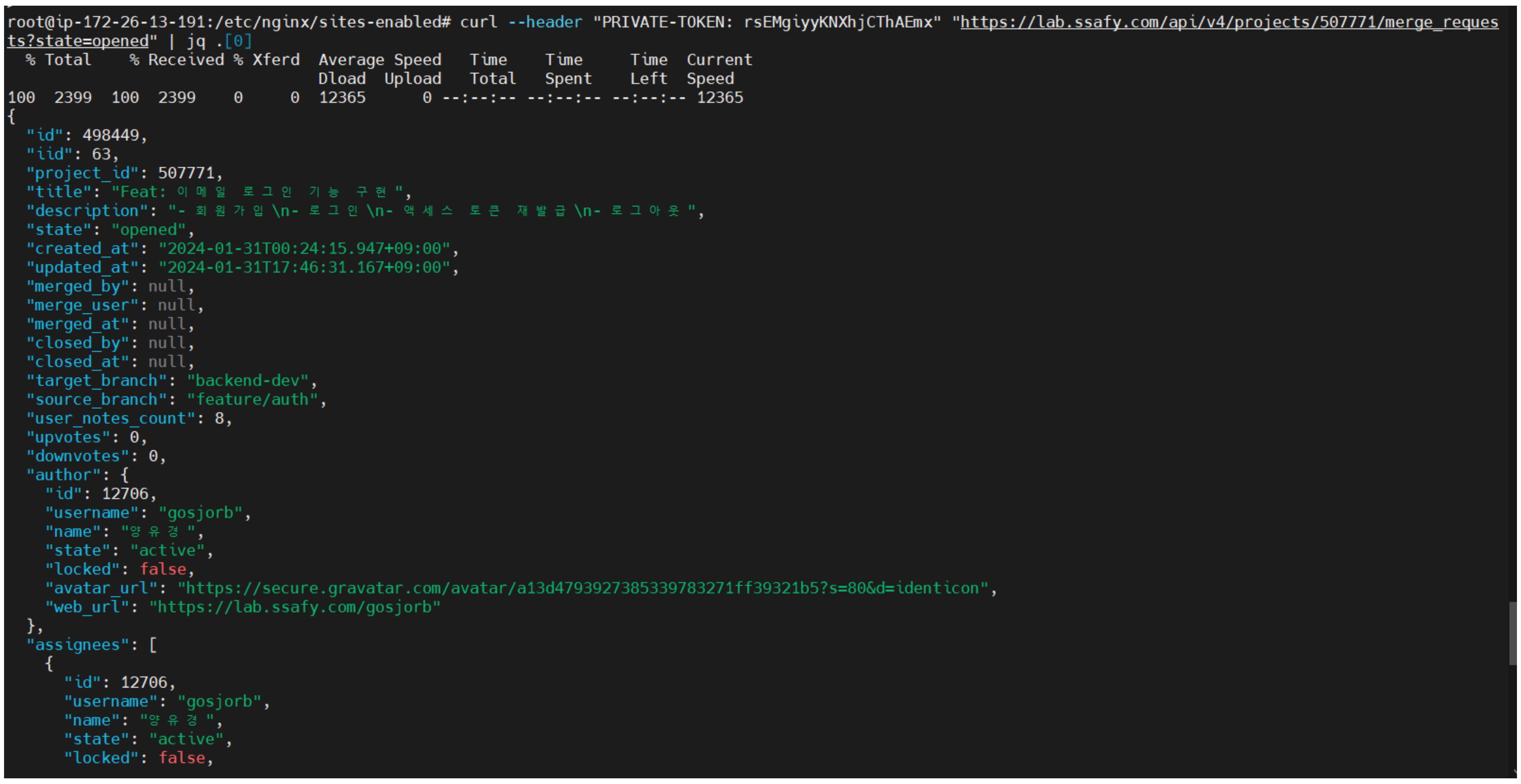
- Pipeline

파이프 라인 스크립트 작성

깃 브랜치 전략을 변형한 깃 전략 사용으로 dev 브랜치 특정하여 실행

⇒ GitLab API 사용

```
# BRANCH
curl --header "PRIVATE-TOKEN: `PRIVATE-TOKEN`" \
    "https://lab.ssafy.com/api/v4/projects \
    /507771/merge_requests?state=opened" \
    | jq '.[0] | .source_branch'

# ASSIGNEE
curl --header "PRIVATE-TOKEN: `PRIVATE-TOKEN`" \
    "https://lab.ssafy.com/api/v4/projects \
    /507771/merge_requests?state=opened" \
    | jq '.[0] | .assignees[0] | .name'

# REVIWER
curl --header "PRIVATE-TOKEN: `PRIVATE-TOKEN`" \
    "https://lab.ssafy.com/api/v4/projects\
    /507771/merge_requests?state=opened" \
    | jq '.[0] | .reviewers[0] | .name'
```



```
root@ip-172-26-13-191:/etc/nginx/sites-enabled# curl --header "PRIVATE-TOKEN: rsEMgiyyKNXhjCThAEmx" "https://lab.ssafy.com/api/v4/projects/507771/merge_reques
ts?state=opened" | jq .[0]
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2399  100  2399    0     0  12365      0 --:--:-- --:--:-- --:--:-- 12365
{
  "id": 498449,
  "iid": 63,
  "project_id": 507771,
  "title": "Feat: 이메일 로그인 기능 구현",
  "description": "- 회원가입\n- 로그인\n- 액세스 토큰 재발급\n- 로그아웃",
  "state": "opened",
  "created_at": "2024-01-31T00:24:15.947+09:00",
  "updated_at": "2024-01-31T17:46:31.167+09:00",
  "merged_by": null,
  "merge_user": null,
  "merged_at": null,
  "closed_by": null,
  "closed_at": null,
  "target_branch": "backend-dev",
  "source_branch": "feature/auth",
  "user_notes_count": 8,
  "upvotes": 0,
  "downvotes": 0,
  "author": {
    "id": 12706,
    "username": "gosjorb",
    "name": "양유경",
    "state": "active",
    "locked": false,
    "avatar_url": "https://secure.gravatar.com/avatar/a13d4793927385339783271ff39321b5?s=80&d=identicon",
    "web_url": "https://lab.ssafy.com/gosjorb"
  },
  "assignees": [
    {
      "id": 12706,
      "username": "gosjorb",
      "name": "양유경",
      "state": "active",
      "locked": false,
```

## Backend CI/CD

- 이미지 빌드 후 사용
- 3개의 서버로 나누어 실행
    - backend, togeball-chat, togeball-matching

  backend - Dockerfile

```
FROM gradle:7.4-jdk17 as builder
WORKDIR /build

COPY build.gradle settings.gradle /build/
RUN gradle build -x test --parallel \
     --continue > /dev/null 2>&1 || true

COPY . /build
RUN gradle build -x test --parallel

# APP
FROM openjdk:17-ea-4-jdk-slim
WORKDIR /app

COPY --from=builder /build/build/libs\
    /togeball-0.0.1-SNAPSHOT.jar .

EXPOSE 8080

USER nobody
ENTRYPOINT [ \
   "java", \
   "-jar", \
   "-Djava.security.egd=file:/dev/./urandom", \
   "-Dsun.net.inetaddr.ttl=0", \
   "togeball-0.0.1-SNAPSHOT.jar" \
]
```

  togeball-chat - Dockerfile

```
FROM gradle:7.4-jdk17 as builder
WORKDIR /build

COPY build.gradle settings.gradle /build/
RUN gradle build -x test --parallel --continue > /dev/null 2>&1 || true

COPY . /build
RUN gradle build -x test --parallel

# APP
FROM openjdk:17-ea-4-jdk-slim
WORKDIR /app

COPY --from=builder /build/build\
        /libs/togeball-chatting-0.0.1-SNAPSHOT.jar .

EXPOSE 8080

USER nobody
ENTRYPOINT [ \
   "java", \
   "-jar", \
   "-Djava.security.egd=file:/dev/./urandom",  \
   "-Dsun.net.inetaddr.ttl=0", \
   "togeball-chatting-0.0.1-SNAPSHOT.jar" \
]
```

togeball-matching

```
FROM gradle:7.4-jdk17 as builder
WORKDIR /build

COPY build.gradle settings.gradle /build/
RUN gradle build -x test --parallel --continue > /dev/null 2>&1 || true

COPY . /build
RUN gradle build -x test --parallel

# APP
FROM openjdk:17-ea-4-jdk-slim
WORKDIR /app

COPY --from=builder /build/build/libs/togeball-matching-0.0.1-SNAPSHOT.jar .

EXPOSE 8080

USER nobody
ENTRYPOINT [ \
   "java", \
   "-jar",  \
   "-Djava.security.egd=file:/dev/./urandom", \
   "-Dsun.net.inetaddr.ttl=0", \
   "togeball-matching-0.0.1-SNAPSHOT.jar" \
]
```

- 젠킨스 파이프라인 작성

```
pipeline{
  agent any
  environment {
    def BRANCH = sh(script: '''curl --fail --header \
          "PRIVATE-TOKEN: `PRIVATE-TOKEN`" \
            "https://lab.ssafy.com/api/v4/projects \
            /507771/merge_requests?state=opened" \
            | jq '.[0] | .source_branch' ''', \
            returnStdout: true).trim().replaceAll('^\"|\"$', '')
    }

  stages {
    stage('gitlab Connect'){
      steps{
        git branch: 'backend-dev',
        credentialsId: 'gitlabCredential',
        url: 'https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A610.git'
      }
    }
    stage('build'){
      steps{
        sh 'cd /var/jenkins_home/workspace/backend-dev'
        dir('backend'){
          sh 'cp -r /var/jenkins_home/backend/env \
                        /var/jenkins_home/workspace\
                      /backend-dev/backend/src/main/resources/'
          sh 'cp -r /var/jenkins_home/backend/env
                        /var/jenkins_home/workspace\
                        /backend-dev/backend/src/test/resources/'
          sh 'chmod +x gradlew'
          sh './gradlew clean sonar \
                        -Dsonar.projectKey=spring \
                        -Dsonar.host.url=http://i10a610.p.ssafy.io:3333 \
                        -Dsonar.login=[Sonar Token]'
          sh './gradlew build -x test'
        }
          dir('togeball-chat'){
          sh 'cp -r /var/jenkins_home/backend/env \
                      /var/jenkins_home/workspace/backend-dev\
                      /togeball-chat/src/main/resources/'
          sh 'chmod +x gradlew'
          sh './gradlew clean build -x test'
        }
        dir('togeball-matching'){
          sh 'cp -r /var/jenkins_home/backend/env \
                      /var/jenkins_home/workspace/backend-dev\
                      /togeball-matching/src/main/resources/'
          sh 'chmod +x gradlew'
          sh './gradlew clean build -x test'
        }
      }
    }
    stage('deploy'){
      steps{
        sh 'docker stop spring && docker rm spring \
                        && docker rmi backend'
        dir('backend'){
          sh 'docker build -t backend ./'
          sh 'docker run --restart=on-failure -p 8081:8080 \
                 -d --name spring backend'
        }
        sh 'docker stop chat && docker rm chat \
                 && docker rmi chatting'
        dir('togeball-chat'){
          sh 'docker build -t chatting ./'
          sh 'docker run --restart=on-failure -p 8082:8080 \
                 -d --name chat chatting'
        }
        sh 'docker stop match && docker rm match \
                 && docker rmi matching'
        dir('togeball-matching'){
          sh 'docker build -t matching ./'
          sh 'docker run --restart=on-failure -p 8083:8080 \
                 -d --name match matching'
        }
      }
        post {
        success {
          script {
            def Author_ID = sh(script: "git show -s --pretty=%an",\
                          returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", \
                          returnStdout: true).trim()
            mattermostSend (color: 'good',
            message: "빌드 성공: \
                          ${env.JOB_NAME} #${env.BUILD_NUMBER} \
                          by ${Author_ID}(${Author_Name}) \
                          \n(<${env.BUILD_URL}|Details>)",
            endpoint: 'https://meeting.ssafy.com\
                          /hooks/q5chm7pghjrhtrchwo4ykxesnh',
            channel: 'togeball-jenkins'
            )
          }
        }
        failure {
          script {
            def Author_ID = sh(script: "git show -s --pretty=%an", \
                          returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", \
                          returnStdout: true).trim()
            mattermostSend (color: 'danger',
              message: "배포 실패: \
                            ${env.JOB_NAME} #${env.BUILD_NUMBER} \
                            by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details
>)",
                endpoint: 'https://meeting.ssafy.com \
                           /hooks/q5chm7pghjrhtrchwo4ykxesnh',
              channel: 'togeball-jenkins'
              )
          }
        }
          }
    }
```

```
  }
}
```

## Frontend CI/CD

- 젠킨스 파이프라인 작성
  - 빌드 폴더 Nginx 컨테이너로 카피
  - CI=false 환경변수 설정 → 이걸 안하면 경고 메시지를 오류로 인식

```
pipeline{
    agent any
    tools {nodejs "nodejs"}
    stages {
        stage('gitlab Connect'){
            steps{
                git branch: 'frontend-dev',
                credentialsId: 'gitlabCredential',
                url: 'https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A610.git'
            }
        }
        stage('build'){
            steps{
                sh 'cd /var/jenkins_home/workspace/frontend-dev/frontend/'
                dir('frontend'){
                    sh 'cp /var/jenkins_home/front/env \
                                /var/jenkins_home/workspace/frontend-dev/frontend/.env'
                    sh 'npm install -g yarn'
                    sh 'yarn install'
                    sh 'yarn add @stomp/stompjs sockjs-client'
            sh 'CI=false yarn build'
                }
            }
        }
        stage('deploy'){
            steps{
                dir('frontend'){
                  sh 'docker cp ./build nginx:/usr/share/nginx/html/'
                }
            }
        }
    }
    post {
    success {
      script {
        def Author_ID = sh(script: "git show -s --pretty=%an", \
                    returnStdout: true).trim()
        def Author_Name = sh(script: "git show -s --pretty=%ae", \
                    returnStdout: true).trim()
        mattermostSend (color: 'good',
          message: "빌드 성공: \
                        ${env.JOB_NAME} #${env.BUILD_NUMBER} \
                        by ${Author_ID}(${Author_Name}) \
                        \n(<${env.BUILD_URL}|Details>)",
          endpoint: 'https://meeting.ssafy.com \
                        /hooks/q5chm7pghjrhtrchwo4ykxesnh',
          channel: 'togeball-jenkins'
          )
        }
      }
      failure {
        script {
          def Author_ID = sh(script: "git show -s --pretty=%an", \
                         returnStdout: true).trim()
          def Author_Name = sh(script: "git show -s --pretty=%ae", \
                         returnStdout: true).trim()
          mattermostSend (color: 'danger',
            message: "빌드 실패: \
                           ${env.JOB_NAME} #${env.BUILD_NUMBER} \
                           by ${Author_ID}(${Author_Name}) \
                           \n(<${env.BUILD_URL}|Details>)",
            endpoint: 'https://meeting.ssafy.com \
                           /hooks/q5chm7pghjrhtrchwo4ykxesnh',
            channel: 'togeball-jenkins'
            )
          }
        }
    }
}
```

# 모니터링 인프라 구축

## Prometheus

- /etc/prometheus 폴더에 prometheus.yml 작성

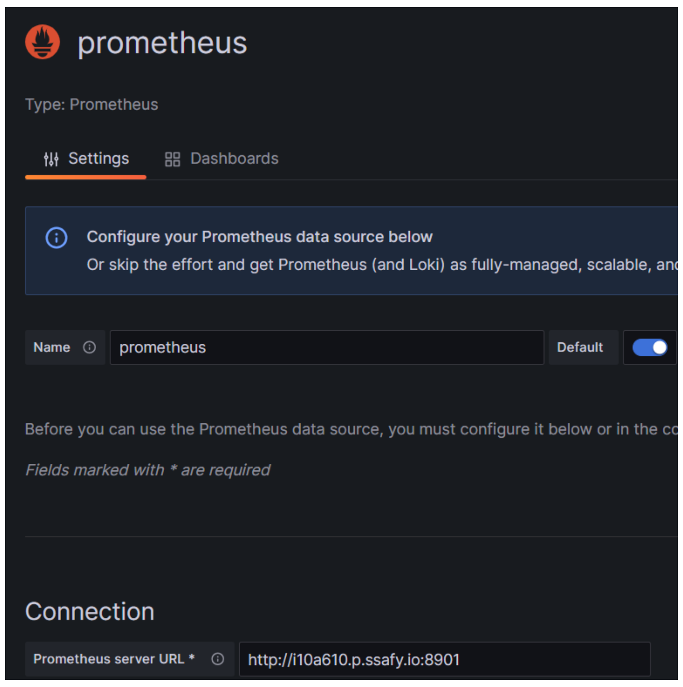
```
global:
  scrape_interval: 15s

scrape_configs:

  - job_name: 'prometheus'
    static_configs:
            - targets: ['i10a610.p.ssafy.io:8901']

  - job_name: 'node-exporter'
    static_configs:
            - targets: ['i10a610.p.ssafy.io:8902']

  - job_name: 'nginx-exporter'
    static_configs:
            - targets: ['i10a610.p.ssafy.io:8903']
  - job_name: 'jenkins'
    metrics_path: /prometheus
    static_configs:
            - targets: ['i10a610.p.ssafy.io:8888']
```

- 도커 컨테이너 생성

```
sudo docker run --restart=on-failure -d \
        -e TZ=Asia/Seoul -p 8902:9100 --name node-exporter \
        prom/node-exporter:latest
sudo docker run --restart=on-failure -d \
        -e TZ=Asia/Seoul -p 8901:9090 \
        -v /etc/prometheus:/etc/prometheus --name prometheus \
        prom/prometheus:latest
sudo docker run --restart=on-failure \
        -e TZ=Asia/Seoul -d -p 8900:3000 \
        --name grafana grafana/grafana:latest
```

## Grafana

- 접속
    - 초기 Id/Pw는 admin/admin
    - Connection > Add DataSource > prometheus > Test Connection 해서 Success


prometheus

Type: Prometheus

Configure your Prometheus data source below
Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and

Name: prometheus
Default: on

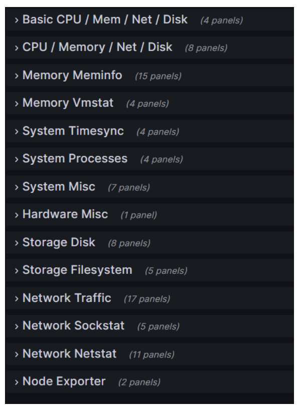
Before you can use the Prometheus data source, you must configure it below or in the co

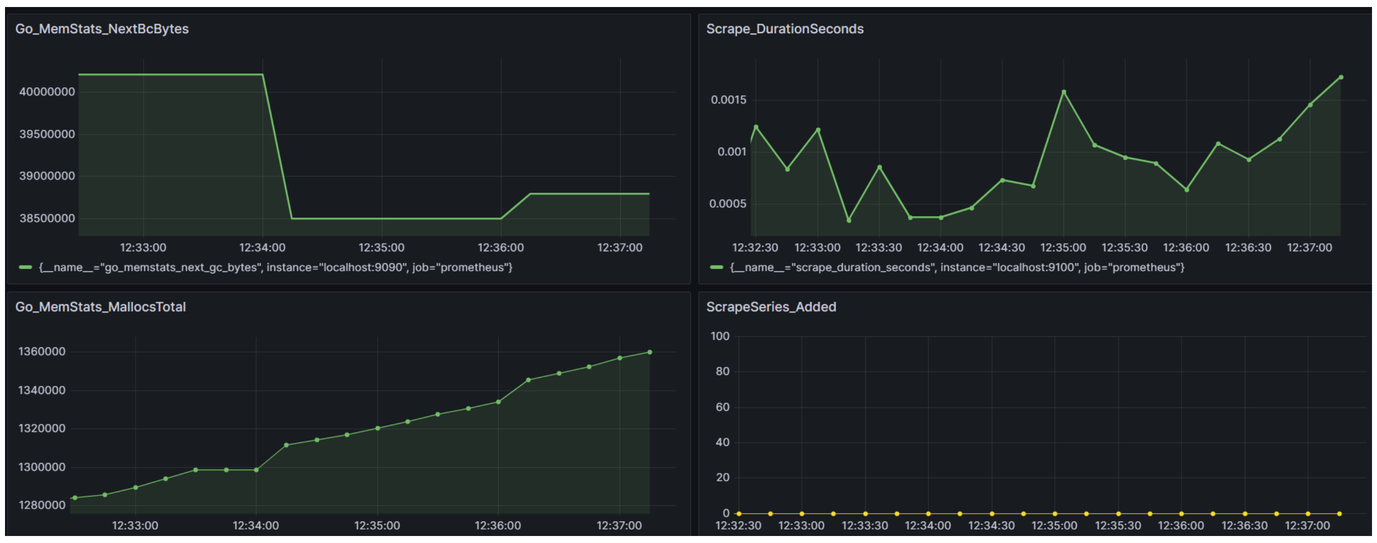
*Fields marked with * are required*

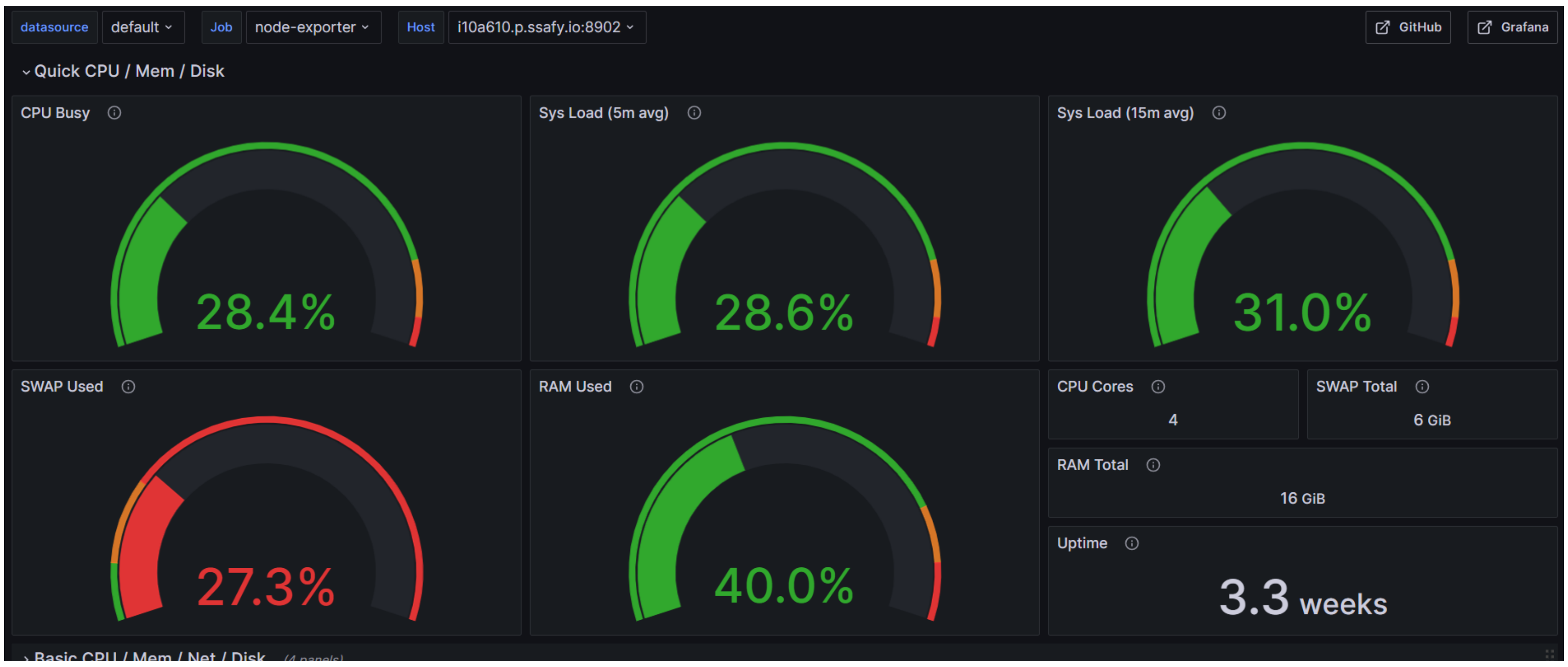
Connection

Prometheus server URL *: http://i10a610.p.ssafy.io:8901

- DashBoard
    - Prometheus 연결 & metric 쿼리 작성


› Basic CPU / Mem / Net / Disk (4 panels)
› CPU / Memory / Net / Disk (8 panels)
› Memory Meminfo (15 panels)
› Memory Vmstat (4 panels)
› System Timesync (4 panels)
› System Processes (4 panels)
› System Misc (7 panels)
› Hardware Misc (1 panel)
› Storage Disk (8 panels)
› Storage Filesystem (5 panels)
› Network Traffic (17 panels)
› Network Sockstat (5 panels)
› Network Netstat (11 panels)
› Node Exporter (2 panels)





## nginx-exporter

- `Nginx` 의 `stub_status` 모듈을 활성화

```
docker pull nginx/nginx-prometheus-exporter:latest
docker run --restart=on-failure -e TZ=Asia/Seoul -p 8903:9113 \
        -d --name nginx-exporter nginx/nginx-prometheus-exporter
```

```
docker exec -it nginx bash

apt update
apt install vim
```

sites-enabled의 default 파일에서 listen 80을 찾아서 아래 부분 추가

```
server {
  listen 80;
  server_name localhost;

  location /metrics {
    stub_status on;
    allow all;
  }
}
```

컨테이너에서 나와 도커 내 nginx reload

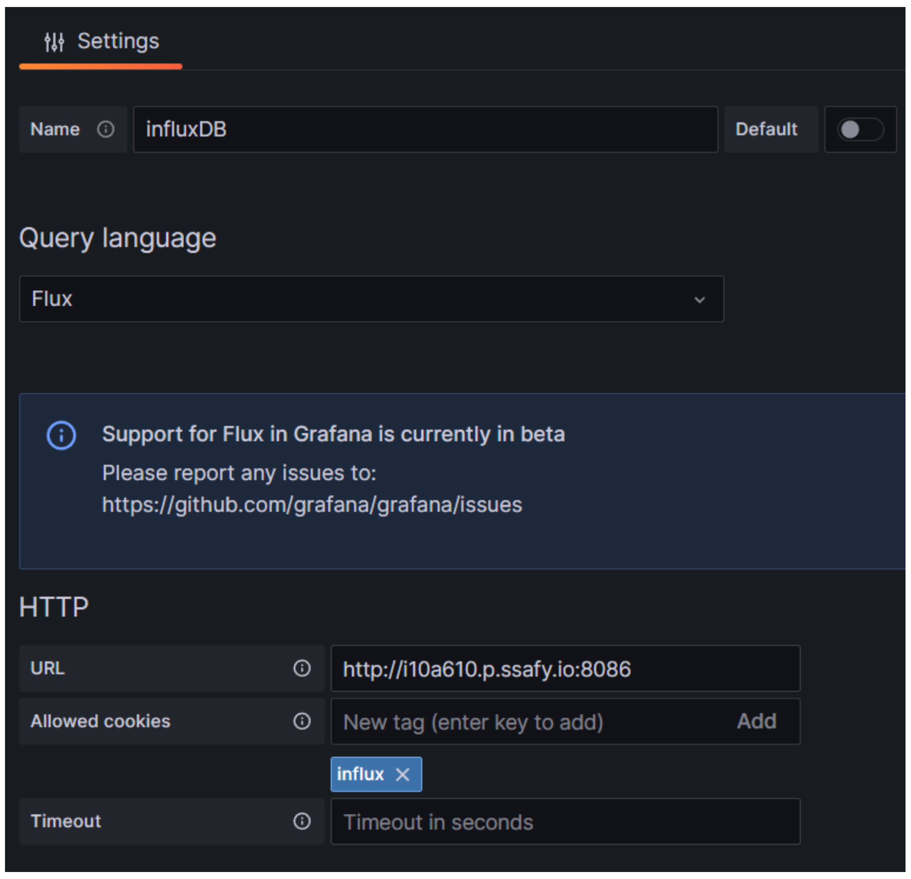
```
docker exec -it nginx service nginx reload
```

## jenkins

- jenkins에서 prometheus 플러그인 추가
- influxDB 설정

  influxdb 접속해서 API Tokens > admin's Token이 Active 되어 있는지 확인

  그라파나에서 influxDB를 DataSource에 추가



Settings

Name: influxDB  Default: off

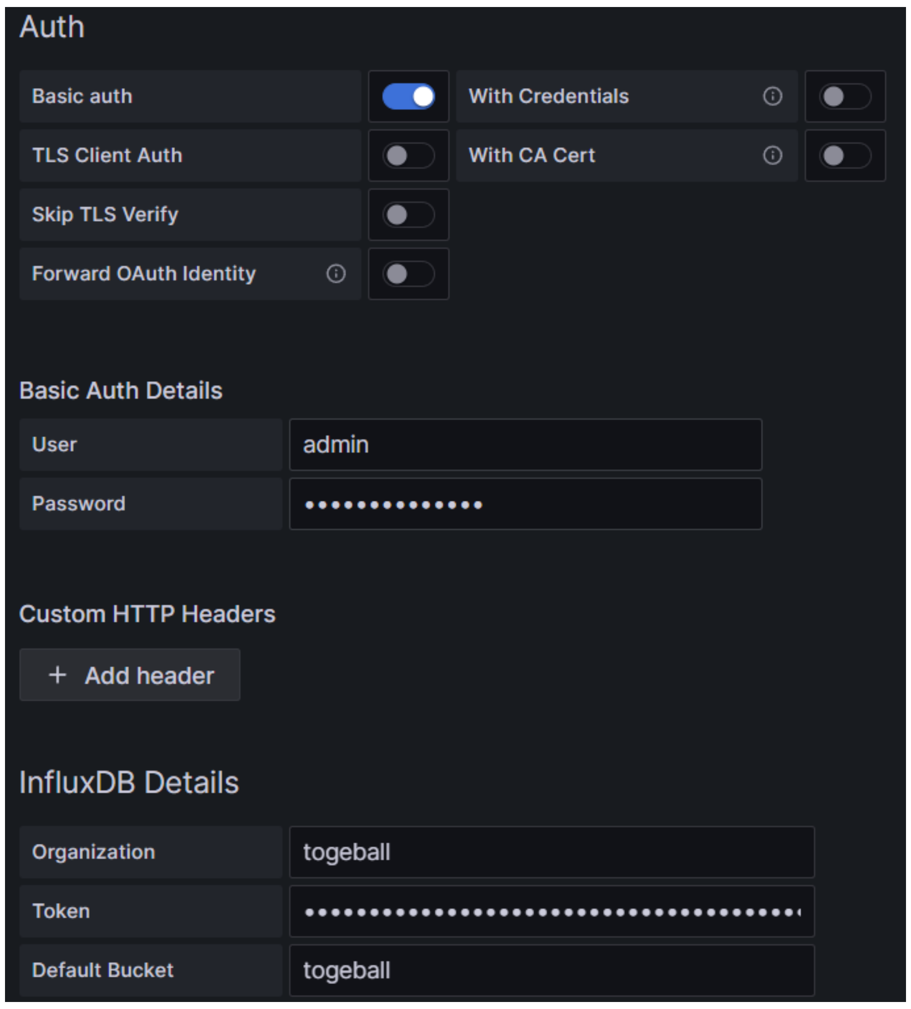
Query language

Flux

Support for Flux in Grafana is currently in beta
Please report any issues to:
https://github.com/grafana/grafana/issues

HTTP

| | |
|---|---|
| URL | http://i10a610.p.ssafy.io:8086 |
| Allowed cookies | New tag (enter key to add) Add |
| | influx × |
| Timeout | Timeout in seconds |



Auth

| | | | |
|---|---|---|---|
| Basic auth | on | With Credentials | off |
| TLS Client Auth | off | With CA Cert | off |
| Skip TLS Verify | off | | |
| Forward OAuth Identity | off | | |

Basic Auth Details

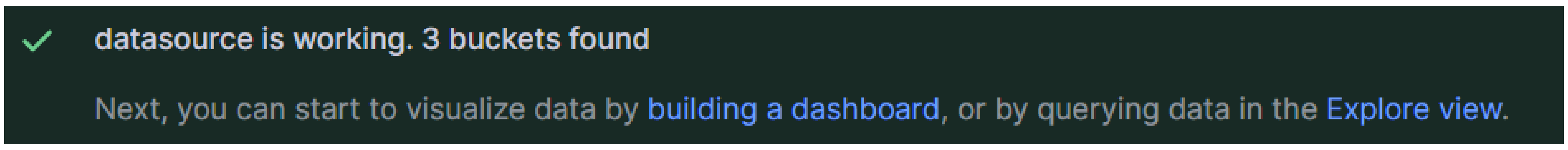
| | |
|---|---|
| User | admin |
| Password | •••••••••••••• |

Custom HTTP Headers

+ Add header

InfluxDB Details

| | |
|---|---|
| Organization | togeball |
| Token | ••••••••••••••••••••••••••••••••••••• |
| Default Bucket | togeball |

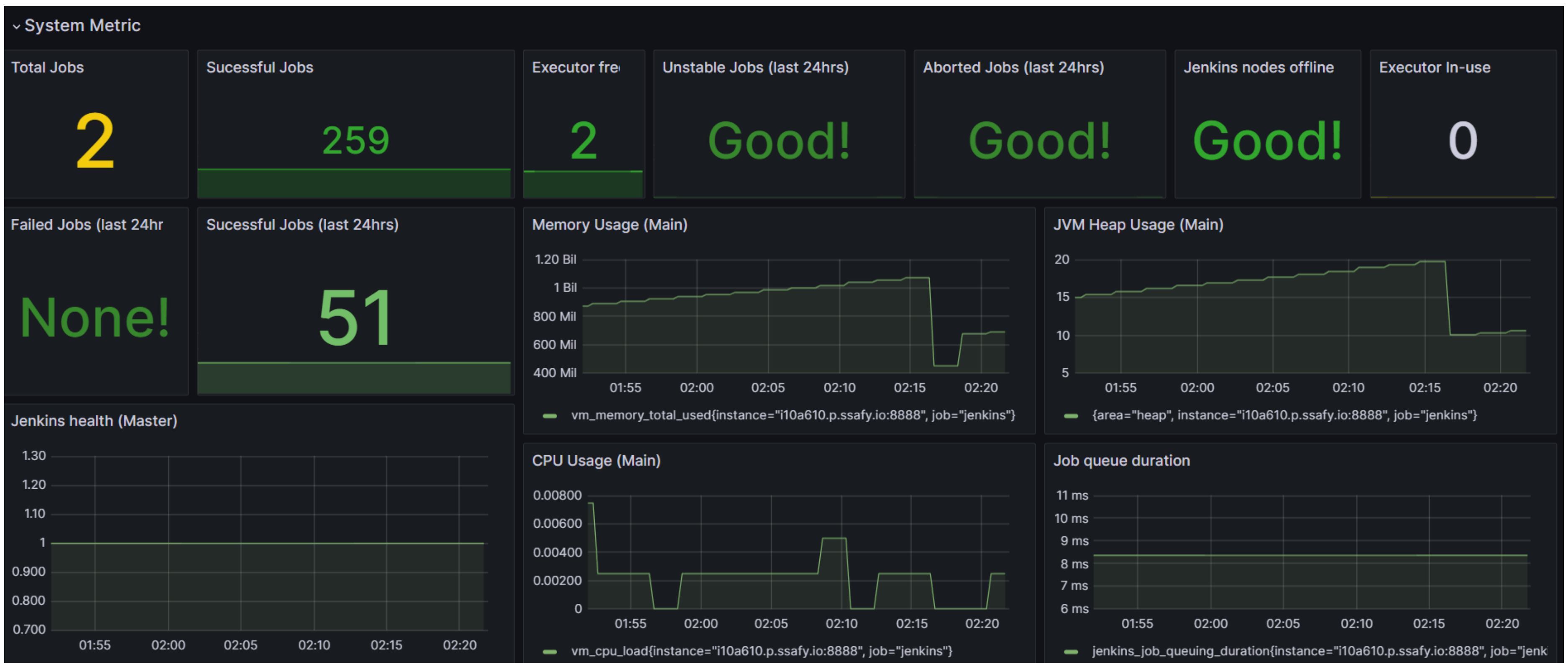


✓ datasource is working. 3 buckets found

Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.

그라파나 랩에서 젠킨스 템플릿 찾아서 넣고 prometheus랑 influxDB 연결

- 템플릿 커스터마이징



System Metric
Total Jobs
2
Sucessful Jobs
259
Executor fre
2
Unstable Jobs (last 24hrs)
Good!
Aborted Jobs (last 24hrs)
Good!
Jenkins nodes offline
Good!
Executor In-use
0
Failed Jobs (last 24hr
None!
Sucessful Jobs (last 24hrs)
51
Memory Usage (Main)
vm_memory_total_used{instance="i10a610.p.ssafy.io:8888", job="jenkins"}
JVM Heap Usage (Main)
{area="heap", instance="i10a610.p.ssafy.io:8888", job="jenkins"}
Jenkins health (Master)
CPU Usage (Main)
vm_cpu_load{instance="i10a610.p.ssafy.io:8888", job="jenkins"}
Job queue duration
jenkins_job_queuing_duration{instance="i10a610.p.ssafy.io:8888", job="jenk

대시보드에서 주로 모니터링하는 데이터

- 해당 Jenkins Main Server에 등록된 총 Job의 수
- 빌드 단계 대기 중인 Queue에 누적된 Job의 수
- 최근 성공 혹은 실패 Job의 수
- Jenkins Worker의 상태
- Jenkins Main Server의 CPU / Memory 등 리소스 사용량