# 포팅 메뉴얼

### Frontend [ Server :  ubuntu ]

- Nginx
- Certbot → SSL
- Docker
- Jenkins : Front Branch Webhook, Mattermost Webhook

### Database [ Server :  ubuntu ]

- Docker
- MySQL → Token[3306], User[3307], Project[3308], Issue[3309], Widget[3310]
    - → root 비번 변경
    - → User 생성 % 접속 권한 부여
- Redis
- Elasticsearch
- Kibana

### Backend [ Server :  ubuntu ]

- Nginx
- Certbot → SSL
- Docker
    - + Docker Compose
- Jenkins : Back Branch Webhook, Mattermost Webhook
- Logstash
- Zipkin
- Prometheus
- Grafana
- RabbitMQ

## Nginx + SSL

### Nignx 설치

```
sudo apt install nginx
```

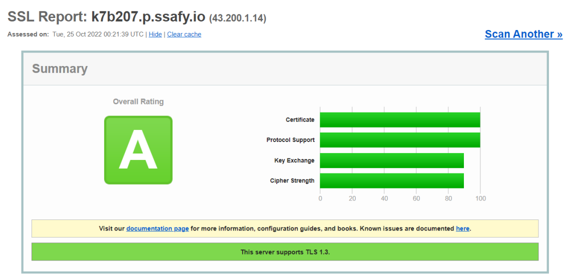
## Certbot 설치

```
sudo apt install certbot
sudo apt-get install python3-certbot-nginx
```

## Certbot SSL 설정

```
sudo certbot --nginx -d k7b207.p.ssafy.io
sudo certbot --nginx -d k7b2071.p.ssafy.io
```

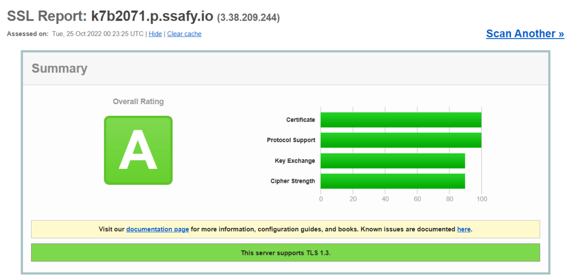
https://www.ssllabs.com/ssltest/ - SSL 적용 확인 및 평가



SSL Report: k7b207.p.ssafy.io (43.200.1.14)
Assessed on: Tue, 25 Oct 2022 00:21:39 UTC | Hide | Clear cache

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports TLS 1.3.



SSL Report: k7b2071.p.ssafy.io (3.38.209.244)
Assessed on: Tue, 25 Oct 2022 00:23:25 UTC | Hide | Clear cache

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports TLS 1.3.

## Nginx 설정

- k7b207.p.ssafy.io

path : /etc/nginx/sites-available/default

```
server {
    if ($host = k7b207.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;
        listen [::]:80;

        server_name k7b207.p.ssafy.io;

        return 301 https://k7b207.p.ssafy.io$request_uri;


}

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

        server_name     k7b207.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/k7b207.p.ssafy.io/fullchain.pem; # manag
ed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/k7b207.p.ssafy.io/privkey.pem; # man
aged by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


        location / {
                proxy_pass http://localhost:8080;
                proxy_redirect off;
                charset utf-8;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_set_header X-NginX-Proxy true;
        }
}
```

- k7b2071.p.ssafy.io

path : /etc/nginx/sites-available/default

```
server {
    if ($host = k7b2071.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;
        listen [::]:80;

        server_name k7b2071.p.ssafy.io;

        return 301 https://k7b2071.p.ssafy.io$request_uri;


}

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

        server_name     k7b2071.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/k7b2071.p.ssafy.io/fullchain.pem; # mana
ged by Certbot
        ssl_certificate_key /etc/letsencrypt/live/k7b2071.p.ssafy.io/privkey.pem; # ma
naged by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
        location / {
                proxy_pass http://localhost:8000;
                proxy_redirect off;
                charset utf-8;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_set_header X-NginX-Proxy true;
        }
        location /docs {
                proxy_pass http://localhost:8080/;
                proxy_redirect off;
                charset utf-8;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_set_header X-NginX-Proxy true;
        }
}
```

# Docker

## Docker 설치

```
sudo apt-get update
curl -fsSL https://get.docker.com/ | sudo sh
```

### Docker 권한 설정

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

sudo su
sudo su ubuntu

docker ps
```

## Docker Compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.11.0/docker-compo
se-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

## Docker Compose 권한 설정

```
sudo chmod +x /usr/local/bin/docker-compose
```

## Docker Compose 심볼릭 링크 설정 (path error 방지)

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

## Jenkins : Docker Out Of Docker 방식

### Jenkins 볼륨 폴더 생성

```
sudo mkdir /home/ubuntu/jenkinsDir
```

### Jenkins Image Custom

```
FROM jenkins/jenkins:lts-jdk11

USER root

RUN apt-get update \
 && apt-get -y install lsb-release \
 && curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /usr/sh
are/keyrings/docker-archive-keyring.gpg \
 && echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] ht
tps://download.docker.com/linux/debian $(lsb_release -cs) stable"
| tee /etc/apt/sources.list.d/docker.list > /dev/null \
 && apt-get update \
 && apt-get -y install docker-ce docker-ce-cli containerd.io
RUN usermod -u {{호스트의사용자아이디}} jenkins && \
    groupmod -g {{호스트의도커그룹아이디}} docker && \
    usermod -aG docker jenkins

USER jenkins
```

## 호스트 사용자 아이디 확인

```
sudo cat /etc/passwd
```

## 호스트의 도커그룹 아이디 확인

```
sudo cat /etc/group
```

## Jenkins Docker Build

```
docker build -t b207-jenkins:0.1 .
```

## Jenkins 실행

```
[-d : 백그라운드 실행 ]
[-p : 컨테이너와 호스트 PC간 연결을 위한 포트 지정 ]
[-v : 이미지의 /var/jenkins_home 디렉토리를 호스트 PC내에 마운트 - Jenkins 설치 시 ssh 키값 생성,
 저장소 참조 등을 용이하게 하기 위함 ]]
docker run -d --name jenkins \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v jenkins:/var/jenkins_home \
    -p 9090:8080 b207-jenkins:0.1
```

## Plug In

- 기본 권장 설치
- 블루오션
- 깃랩
- 메러모스트 노티피케이션

## Mattermost Notification

젠킨스관리 → 설정시스템 → 아래로 스크롤 → mattermost notification endpoint → mattermost in webhook 주소 기입

## Database Port Mapping

- MySQL ( version : 8.0.22 )
    - Token → 3306
    - User → 3307
    - Project → 3308
    - Issue → 3309
    - Widget → 3310
- Redis ( version : 7.0.4 )
    - 6379

## Database

### MySQL image pull

```
docker pull mysql:8.0.22
# 8.0.22: Pulling from library/mysql
# Digest: sha256:8c17271df53ee3b843d6e16d46cff13f22c9c04d6982eb15a9a47bd5c9ac7e2d
# Status: Downloaded newer image for mysql:8.0.22
# docker.io/library/mysql:8.0.22
```

### 볼륨 폴더 생성

```
sudo mkdir /opt/lib/mysql
```

### Docker Container 접속

```
docker exec -it  [컨테이너 명] /bin/bash
```

## MySQL 유저 생성 및 권한 부여

```
# mysql 접속
mysql -u root -p

# root 계정 비밀번호 변경
alter user 'root'@'localhost' identified with mysql_native_password by 'new password';

flush privileges;

# user 생성 및 권한 부여
create user '[username]'@'%' identified by '[password]';
grant all privileges on *.* to '[username]'@'%' with grant option;
flush privileges;
```

## Token DB - 볼륨 : token-volume

```
docker run --name token-mysql -e MYSQL_ROOT_PASSWORD={사용 할 root 유저 비밀번호} -v token
-volume:/var/lib/mysql -d -p 3306:3306 mysql:8.0.22
```

## User DB - 볼륨 : user-volume

```
docker run --name user-mysql -e MYSQL_ROOT_PASSWORD={사용 할 root 유저 비밀번호} -v user-v
olume:/var/lib/mysql -d -p 3307:3306 mysql:8.0.22
```

## Project DB - 볼륨 : project-volume

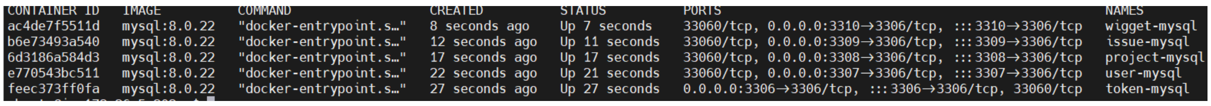
```
docker run --name project-mysql -e MYSQL_ROOT_PASSWORD={사용 할 root 유저 비밀번호} -v pro
ject-volume:/var/lib/mysql -d -p 3308:3306 mysql:8.0.22
```

## Issue DB - 볼륨 : issue-volume

```
docker run --name issue-mysql -e MYSQL_ROOT_PASSWORD={사용 할 root 유저 비밀번호} -v issue
-volume:/var/lib/mysql -d -p 3309:3306 mysql:8.0.22
```

## Wigdet DB - 볼륨 : widget-volume

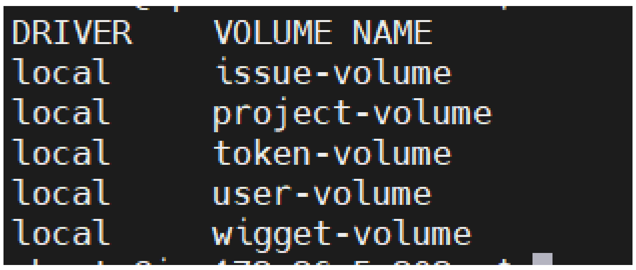
```
docker run --name widget-mysql -e MYSQL_ROOT_PASSWORD={사용 할 root 유저 비밀번호} -v widg
et-volume:/var/lib/mysql -d -p 3310:3306 mysql:8.0.22
```



```
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                                    NAMES
ac4de7f5511d   mysql:8.0.22   "docker-entrypoint.s…"   8 seconds ago    Up 7 seconds    33060/tcp, 0.0.0.0:3310→3306/tcp, :::3310→3306/tcp   wigget-mysql
b6e73493a540   mysql:8.0.22   "docker-entrypoint.s…"   12 seconds ago   Up 11 seconds   33060/tcp, 0.0.0.0:3309→3306/tcp, :::3309→3306/tcp   issue-mysql
6d3186a584d3   mysql:8.0.22   "docker-entrypoint.s…"   17 seconds ago   Up 17 seconds   33060/tcp, 0.0.0.0:3308→3306/tcp, :::3308→3306/tcp   project-mysql
e770543bc511   mysql:8.0.22   "docker-entrypoint.s…"   22 seconds ago   Up 21 seconds   33060/tcp, 0.0.0.0:3307→3306/tcp, :::3307→3306/tcp   user-mysql
feec373ff0fa   mysql:8.0.22   "docker-entrypoint.s…"   27 seconds ago   Up 27 seconds   0.0.0.0:3306→3306/tcp, :::3306→3306/tcp, 33060/tcp   token-mysql
```

### Docker volume 확인

```
docker volume list
```



```
DRIVER    VOLUME NAME
local     issue-volume
local     project-volume
local     token-volume
local     user-volume
local     wigget-volume
```

## Redis

### Redis image pull

```
docker image pull redis
```

### Redis와 Redis-cli 연결을 위한 Redis net 생성

```
docker network create redis-net
# 생성 확인
docker network ls
```

### Redis Container Run

```
docker run --name redis -p 6379:6379 --network redis-net -v /home/ubuntu/redisDir -d r
edis:latest redis-server --appendonly yes
```

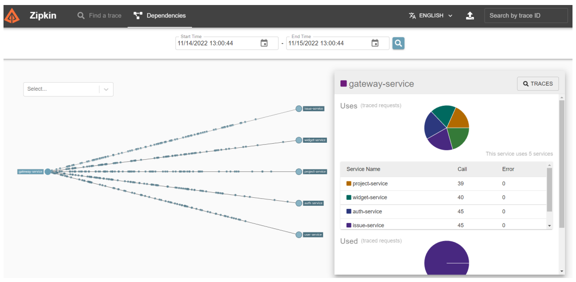
### Redis-cli 접속

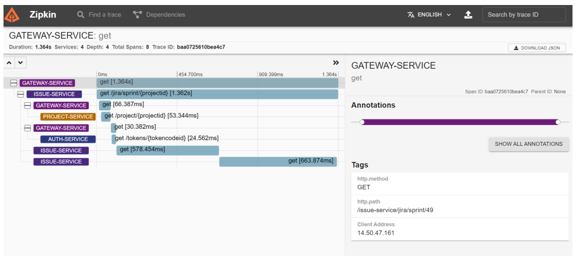
```
docker run -it --network redis-net --rm redis redis-cli -h redis
```

## Zipkin

### Zipkin 설치

```
docker run -d -p 9411:9411 openzipkin/zipkin
```





## Prometheus + Grafana

### Prometheus 설정 파일 생성

```
sudo mkdir /home/ubuntu/prometheus
sudo vi /home/ubuntu/prometheus/prometheus.yml
```

prometheus.yml

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is e
```

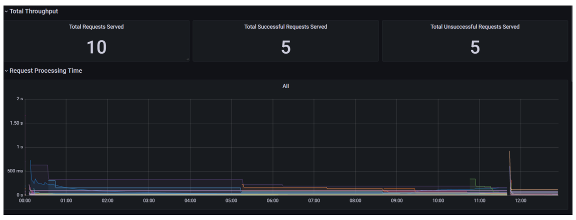
```
very 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 m
inute.

scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from t
his config.
  - job_name: 'prometheus'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
      - targets: ['localhost:9090']
  # 추가
  - job_name: 'gateway-service'
    scrape_interval: 15s
    metrics_path: '/actuator/prometheus'
    static_configs:
      - targets: ['k7b2071.p.ssafy.io:8000']

  - job_name: 'auth-service'
    scrape_interval: 15s
    metrics_path: '/auth-service/actuator/prometheus'
    static_configs:
      - targets: ['k7b2071.p.ssafy.io:8000']

  - job_name: 'user-service'
    scrape_interval: 15s
    metrics_path: '/user-service/actuator/prometheus'
    static_configs:
      - targets: ['k7b2071.p.ssafy.io:8000']

  - job_name: 'project-service'
    scrape_interval: 15s
    metrics_path: '/project-service/actuator/prometheus'
    static_configs:
      - targets: ['k7b2071.p.ssafy.io:8000']

  - job_name: 'issue-service'
    scrape_interval: 15s
    metrics_path: '/issue-service/actuator/prometheus'
    static_configs:
      - targets: ['k7b2071.p.ssafy.io:8000']

  - job_name: 'widget-service'
    scrape_interval: 15s
    metrics_path: '/widget-service/actuator/prometheus'
    static_configs:
      - targets: ['k7b2071.p.ssafy.io:8000']
```

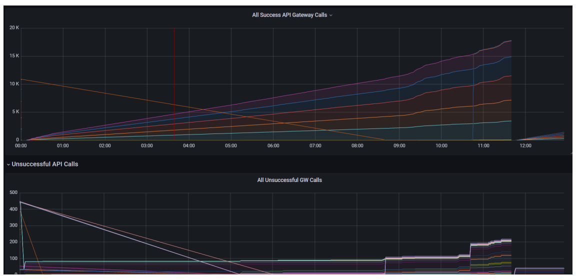
## Prometheus 실행

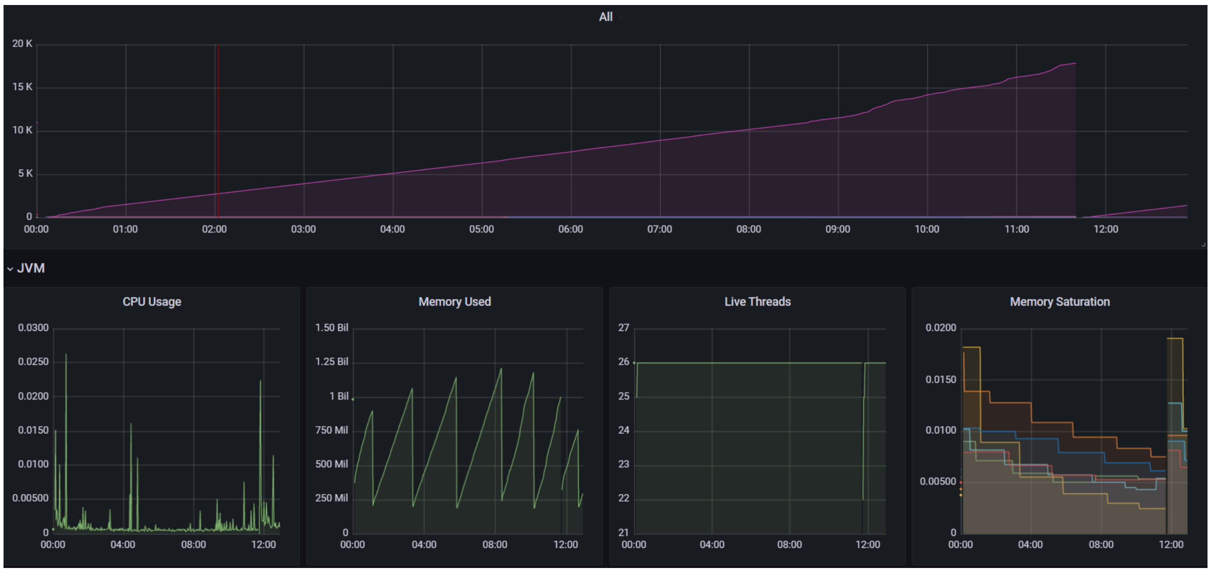
```
docker run -p 9191:9090 -v /home/ubuntu/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml --name prometheus -d prom/prometheus --config.file=/etc/prometheus/prometheus.yml
```

## Grafana 실행

```
docker run -d --name=grafana -p 3000:3000 grafana/grafana
```







# RabbitMQ

## RabbitMQ image pull

```
docker pull rabbitmq:management
```

## RabbitMQ image Run

```
docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 --restart=unless-stopped rab
bitmq:management
```

- 비공인 IP 주소를 사용하여 VM 간에 RabbitMQ와 연동하는 경우에는, <비공인 IP>:5672 주소를 통해 접속할 수 있습니다.
- RabbitMQ의 Management UI Plugin은 Web 대시보드를 통해 관리할 수 있는 도구를 제공합니다. 이를 위해서는 ACG에 15672 포트가 추가되어 있어야 하며 공인 IP 주소를 할당받아 서버에 부여해야 합니다.
- RabbitMQ의 Management UI Plugin의 주소는 http://<공인IP주소>:15672/입니다. 접속되지 않는다면 ACG가 추가되어 있는지 확인하거나, 터미널에서 Management UI가 실행되어 있는지를 확인합니다.

## Java 설치

```
sudo apt-get install openjdk-8-jdk -y
```

## Java 설정

```
sudo vi /etc/profile
```

맨 아래에 추가

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export Class_PATH=$JAVA_HOME/lib:$CLASS_PATH
```

Java 버전 확인

```
java -version
```

# ELK

## GPG 키 추가

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /et
c/apt/sources.list.d/elastic-7.x.list
```

패키지 리스트 업데이트

```
sudo apt update
```

## elasticsearch 설치 및 실행

```
sudo apt install elasticsearch
```

설정

```
sudo vi /etc/elasticsearch/elasticsearch.yml
```

해당 부분 수정후 저장

```
network.host: 0.0.0.0
cluster.initial_master_nodes: ["IP 입력"]
```

실행

```
sudo systemctl start elasticsearch
```

상태 확인

```
sudo systemctl status elasticsearch
```

## kibana 설치

```
sudo apt install kibana
```

설정 변경

```
sudo vi /etc/kibana/kibana.yml
```

주석 제거

```
server.port: 5601
server.host: "localhost"
elasticsearch.hosts: ["elasticsearch의 IP:PORT"]
```

실행

```
sudo systemctl start kibana
```

## logstash 설치

```
sudo apt install logstash
```

sample 수정

```
sudo vi /etc/logstash/conf.d/*.conf
```

```
input {
  tcp {
    port => 5044
    codec => json_lines
  }
}

output {
  elasticsearch {
    hosts => ["elasticsearch의 IP:PORT"]
    index => "logstash-%{+YYYY.MM.dd}"
    #user => "elastic"
    #password => "changeme"
  }
}
```

## logstash 실행

```
sudo systemctl start logstash
```

### **logstash deactivating (stop-sigterm) 해결방법**

```
sudo kill -9 [logstash PID]

PID 확인 방법 -> sudo service logstash status
```