



빌드 및 배포 문서

[프로젝트 개요](#)

[서버 아키텍처](#)

[개발 환경 및 기술 스택](#)

[개발 환경](#)

[기술 스택](#)

[환경변수](#)

[실행 방법](#)

[CI/CD 서버 설정](#)

[서버 설정](#)

[Jenkins](#)

[FrontEnd CI/CD](#)

[BackEnd CI/CD](#)

[Service 서버 설정](#)

[서버 개요](#)

[서버 설정](#)

[DB](#)

[Nginx](#)

[Grafana & Prometheus](#)

프로젝트 개요



PINN과 함께 세계여행을 떠나 볼까요?

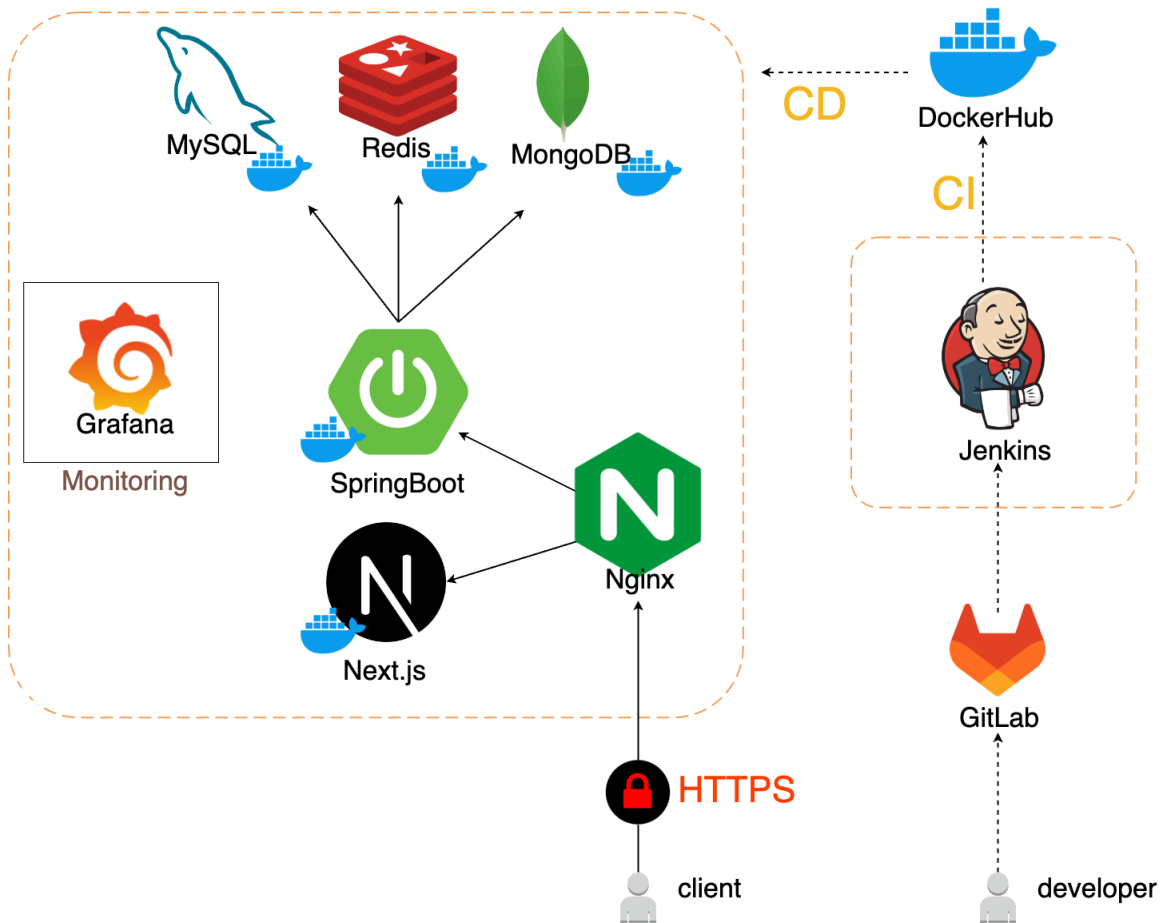
360° GoogleMap을 활용한 전세계 장소 예측 게임!

친구와 함께 팀을 이루어 핀을 찍고, 높은 점수를 얻어 승리하세요!

한국, 그리스, 이집트, 랜드마크, 랜덤 테마 제공

해당 .md에서는 서버 설정에서 발생했던 내용을 다룹니다.

서버 아키텍처



개발 환경 및 기술 스택

개발 환경



GitLab

- <https://lab.ssafy.com/s10-final/S10P31A701.git>

IDEA

- **IntelliJ 2023.3.2**
- **visual studio code 1.85**

기술 스택

Infra



- Jenkins
- Docker
- Docker hub registry
- Nginx
- Grafana
- Prometheus

FrontEnd



- 기술 스택 명시
- Next.js
 - Zustand
 - CSS Module
 - STOMP.js
 - Google Maps JavaScript API Loader
 - TypeScript

BackEnd



- 기술 스택 명시
- JDK 17
 - Spring 3.2.5
 - MySQL 8.xx
 - Oauth2
 - Redis latest
 - IntelliJ latest
 - Gradle latest
 - log4j2
 - WebSocket (Stomp)
 - MongoDB

환경변수

FrontEnd

```
NEXT_PUBLIC_API_URL='https://pinn.kr/api'

NEXT_PUBLIC_KAKAO_REST_KEY='a4bfe3f2b2f815648b923deb0a3c54c7'
NEXT_PUBLIC_JAVASCRIPT_KEY='f62485a26e3ec74c060a10c1c5399172'
NEXT_PUBLIC_BACK_URL='http://localhost:8081'
NEXT_PUBLIC_LOCAL_SOCKET_URL='ws://localhost:8081/game'
NEXT_PUBLIC_SERVER_SOCKET_URL='wss://pinn.kr/api/game'

#NEXT_PUBLIC_KAKAO_LOGIN_REDIRECT_URI='http://localhost:8081/'
NEXT_PUBLIC_KAKAO_LOGIN_REDIRECT_URI="http://www.pinn.kr/api/"
NEXT_PUBLIC_KAKAO_LOGIN_REDIRECT_URI="http://www.pinn.kr/api/"

NEXT_PUBLIC_GOOGLE_MAP_API_KEY='AIzaSyBIbSFMpiaZALZBtQs95ihTr'
```

BackEnd

```
KAKAO_CLIENT_ID=a4bfe3f2b2f815648b923deb0a3c54c7
KAKAO_REDIRECT_URI=http://www.pinn.kr/api/oauth/code/kakao
KAKAO_CLIENT_SECRET=SHzclp5XkVxKnfXezJ2Nx64n1rTM7D3Y
KAKAO_JWT_KEY=Tg4Lmb96Qp3sJzA8KDshwxE2rFcovU
REDIS_HOST=3.34.108.114
REDIS_PORT=6379
REDIS_PASSWORD=ssafy701
MYSQL_URL=jdbc:mysql://3.34.108.114:3306/pinn
MYSQL_USERNAME=root
MYSQL_PASSWORD=ssafy701
MYSQL_CONNECTION=50
MONGO_ROOT_NAME=root
MONGO_ROOT_PASS=ssafy701
MONGO_URI=mongodb://3.34.108.114:27017/pinn
MONGO_HOST=3.34.108.114;
MONGO_PORT=27017
SPRING_SOCKET_END_POINT=/api/game
WEATHER_STATISTICAL_API=141b8fbaf18090e453d8c40980f5a9ff
WEATHER_STATISTICAL_URL=https://history.openweathermap.org/data/
GAME_EXIST_LIMIT_TIME=20000000
```

실행 방법

1. 소스코드 클론 및 이동

```
$ git clone https://lab.ssafy.com/s10-final/S10P31A701.git
$ cd S10P31A701
```

2. 빌드 및 실행

Backend 실행

```
$ cd backend
$ ./gradlew build
$ ./gradlew run
```

Frontend 실행

```
$ cd frontend
$ npm run dev
```

3. 서비스 실행

<http://localhost:3000>

CI/CD 서버 설정

Jenkins를 사용하여 CI/CD가 수행되는 서버

AWS EC2 : ssafy 기본 서버

URL : <https://k10a701.p.ssafy.io>

서버 설정

```
포트번호 : 80, 443, 3000, 8080
sudo ufw allow {포트번호}
sudo ufw enable
sudo ufw reload
```

```

sudo ufw status

sudo apt-get install openjdk-17-jdk
# 자바 환경변수 설정하기
vi /etc/profile
# 가장 아래 쪽에 해당 내용 추가
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=$JAVA_HOME/jre/lib:$JAVA_HOME/lib/tools.jar
# 변경내용 적용하기
source /etc/profile
sudo apt-get install curl

//Docker 설치 전 필요한 패키지 설치
sudo apt-get -y install apt-transport-https ca-certificates
curl gnupg-agent software-properties-common
// docker 패키지 설치
sudo apt-get -y install docker-ce docker-ce-cli containerd.io
// docker 권한 부여
sudo usermod -aG docker ubuntu && sudo service docker restart
// docker compose 설치
sudo curl -L "https://github.com/docker/compose/releases/download/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
// docker compose 실행 권한 추가
sudo chmod +x /usr/local/bin/docker-compose

```

Jenkins

URL : <http://k10a701.p.ssafy.io:8080>

Jenkins를 설치하고 FrontEnd와 BackEnd 각각의 **Item**을 생성하여 **미스크립트**를 넣으면 됩니다.

FrontEnd CI/CD

▼ Script

```
pipeline {
  agent any

  environment {
    imageName = "moonjar/fe_image"
    releaseServerAccount = 'ubuntu'
    releaseServerUri = '3.34.108.114'
    releasePort = '3000'
  }

  stages {
    stage('Build Start'){
      steps{
        script {
          def Author_ID = sh(script: "git show -
          def Author_Name = sh(script: "git show
          mattermostSend (color: '#2A42EE',
          message: "빌드 시작: ${env.JOB_NAME} #${
          endpoint: 'https://meeting.ssafy.com/h
          channel: 'A701_jenkins'
          )
        }
      }
    }
    stage('Git Clone') {
      steps {
        git branch: 'fe/dev',
        credentialsId: 'gitlab-credential',
        url: 'https://lab.ssafy.com/s10-final/
      }
    }
    stage('create env') {
      steps {
        sh'''
          cp /home/ubuntu/frontend/.env /var/lib
        '''
      }
    }
  }
}
```

```

    }
}
stage('Image Build & DockerHub Push') {
    steps {
        script {
            docker.withRegistry('', 'docker-credential-ecr-login') {
                dir('frontend') {
                    sh "docker build -t $imageName ."

                    sh "docker push $imageName"
                }
            }
        }
    }
}
stage('Before Service Stop') {
    steps {
        script {
            try {
                sshagent(credentials: ['ubuntu-server']) {
                    def containers = sh(
                        script: "ssh -o StrictHostKeyChecking=no ubuntu-server 'docker ps -q'",
                        returnStdout: true
                    ).trim()

                    if (containers) {
                        sh """
                            ssh -o StrictHostKeyChecking=no ubuntu-server 'docker stop $(cat /dev/null << { echo $containers; })'
                            ssh -o StrictHostKeyChecking=no ubuntu-server 'docker rm $(cat /dev/null << { echo $containers; })'
                        """
                    } else {
                        echo "No containers found"
                    }
                }
            } catch (Exception e) {
                error "Exception: ${e.getMessage()}"
            }
        }
    }
}

```



```

    }
  }
}

stage('DockerHub Pull') {
  steps {
    sshagent(credentials: ['ubuntu-service'])
    sh "ssh -o StrictHostKeyChecking=no ${
  }
}

stage('Service Start') {
  steps {
    sshagent(credentials: ['ubuntu-service'])
    sh """
        ssh -o StrictHostKeyChecking=no ${
        """
  }
}

}

post {
  success {
    script {
      def Author_ID = sh(script: "git show -s --
      def Author_Name = sh(script: "git show -s
      mattermostSend (color: 'good',
      message: "빌드 성공: ${env.JOB_NAME} #${env.
      endpoint: 'https://meeting.ssafy.com/hooks
      channel: 'A701_jenkins'
      )
    }
  }
  failure {
    script {
      def Author_ID = sh(script: "git show -s --
      def Author_Name = sh(script: "git show -s

```

```

        mattermostSend (color: 'danger',
        message: "빌드 실패: ${env.JOB_NAME} #${env.
        endpoint: 'https://meeting.ssafy.com/hooks
        channel: 'A701_jenkins'
        )
    }
}
}
}

```

BackEnd CI/CD

▼ Script

```

pipeline {
    agent any

    environment {
        imageName = "moonjar/be_image"
        releaseServerAccount = 'ubuntu'
        releaseServerUri = '3.34.108.114'
        releasePort = '8081'
    }

    stages {
        stage('Build Start'){
            steps{
                script {
                    def Author_ID = sh(script: "git show -
                    def Author_Name = sh(script: "git show
                    mattermostSend (color: '#2A42EE',
                    message: "빌드 시작: ${env.JOB_NAME} #${
                    endpoint: 'https://meeting.ssafy.com/h
                    channel: 'A701_jenkins'
                    )
                }
            }
        }
    }
}

```

```

stage('Git Clone') {
    steps {
        git branch: 'be/dev',
            credentialsId: 'gitlab-credential',
            url: 'https://lab.ssafy.com/s10-final/'
    }
}
stage('Jar Build') {
    steps {
        dir('backend') {
            sh 'chmod +x ./gradlew'
            sh './gradlew clean bootJar'
            sh './gradlew build'
        }
    }
}
stage('Image Build & DockerHub Push') {
    steps {
        script {
            docker.withRegistry('', 'docker-credential-ssafy') {
                dir('backend') {
                    sh "docker build -t ${imageName} ."
                    sh "docker push ${imageName}:${tag}"
                }
            }
        }
    }
}
stage('Before Service Stop') {
    steps {
        script {
            try {
                sshagent(credentials: ['ubuntu-server']) {
                    def containers = sh(
                        script: "ssh -o StrictHostKeyChecking=no root@${ip} 'docker ps -q'",
                        returnStdout: true
                    ).trim()
                }
            } catch (Exception) {}
        }
    }
}

```

```

        if (containers) {
            sh """
                ssh -o StrictHostKeyChecki
                ssh -o StrictHostKeyChecki
            """
            // def previousBuildNumber
            // echo "Previous build nu

            // sh '''
            //     if docker images ${
            //         echo "이미지 '${
            //         docker rmi ${im
            //     else
            //         echo "이미지 '${
            //     fi
            // '''
        } else {
            echo "No containers found
        }
    }
} catch (Exception e) {
    error "Exception: ${e.getMessage()}
}
}
}

stage('DockerHub Pull') {
    steps {
        sshagent(credentials: ['ubuntu-service'])
        sh "ssh -o StrictHostKeyChecking=no ${
    }
}

stage('Service Start') {
    steps {
        sshagent(credentials: ['ubuntu-service'])
        sh """

```

```

        ssh -o StrictHostKeyChecking=no ${
            ""
        }
    }
}
}
post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --
            def Author_Name = sh(script: "git show -s
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.
            endpoint: 'https://meeting.ssafy.com/hooks
            channel: 'A701_jenkins'
            )
        }
    }
    failure {
        script {
            def Author_ID = sh(script: "git show -s --
            def Author_Name = sh(script: "git show -s
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.
            endpoint: 'https://meeting.ssafy.com/hooks
            channel: 'A701_jenkins'
            )
        }
    }
}
}
}

```

Service 서버 설정

서버 개요

Ngnix, Next.js, Springboot, MySQL, MongoDB, Redis, Grafana가 실행되는 서비스 서버

AWS LightSail

서버 설정

포트번호 : 80, 443, 3000, 8080

```
sudo ufw allow {포트번호}
```

```
sudo ufw enable
```

```
sudo ufw reload
```

```
sudo ufw status
```

```
sudo apt-get install openjdk-17-jdk
```

```
# 자바 환경변수 설정하기
```

```
vi /etc/profile
```

```
# 가장 아래 쪽에 해당 내용 추가
```

```
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
```

```
export PATH=$PATH:$JAVA_HOME/bin
```

```
export CLASSPATH=$JAVA_HOME/jre/lib:$JAVA_HOME/lib/tools.jar
```

```
# 변경내용 적용하기
```

```
source /etc/profile
```

```
sudo apt-get install curl
```

```
//Docker 설치 전 필요한 패키지 설치
```

```
sudo apt-get -y install apt-transport-https ca-certificates
```

```
curl gnupg-agent software-properties-common
```

```
// docker 패키지 설치
```

```
sudo apt-get -y install docker-ce docker-ce-cli containerd.io
```

```
// docker 권한 부여
```

```
sudo usermod -aG docker ubuntu && sudo service docker restart
```

```
// docker compose 설치
```

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

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

DB

Docker compose를 사용해서 한 번에 사용

```
version: "3.9"
name: pinn

services:
  redis:
    image: redis:latest
    restart: always
    ports:
      - "6379:6379"
    command: ["redis-server", "--appendonly", "yes", "--requirepass"]

  mysqldb:
    image: mysql:8.3.0
    restart: always
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=ssafy701
      - TZ=Asia/Seoul
    volumes:
      - /var/lib/mysql:/var/lib/mysql
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci

  mongodb:
    image: mongo:latest
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
```

```

    MONGO_INITDB_ROOT_PASSWORD: ssafy701
volumes:
  - /var/lib/mongo:/data
ports:
  - "27017:27017"

mongo-express:
  image: mongo-express
  restart: always
  ports:
    - "27020:27020"
  environment:
    - ME_CONFIG_MONGODB_ADMINUSERNAME=root
    - ME_CONFIG_MONGODB_ADMINPASSWORD=ssafy701
    - ME_CONFIG_MONGODB_SERVER=mongodb

networks:
  mongonetnetworks:
    default:
      name: mongodb_network

```

Nginx

서버 방화벽의 80, 443 포트를 허용한 뒤 아래 사항을 따라 설정을 진행합니다.

1. NginX 설치

```

$ sudo apt update
$ sudo apt install nginx -y

```

2. certbot & letsencrypt 설치

```

$ sudo apt-get install letsencrypt
$ sudo apt-get install certbot python3-certbot-nginx

```

3. certbot nginx 연결


```
$ sudo certbot --nginx -d 도메인 이름 -d www.도메인 이름
$ sudo certbot --nginx
$ 이메일 입력
$ 약관 동의 - Y
$ 이메일 수신동의
$ 도메인 입력 - 도메인 이름
$ http 입력시 리다이렉트 여부 - 2
```

3. nginx 설정

```
$ sudo vim /etc/nginx/sites-enabled/default
```

위 명령어를 입력하여, 경로 상의 파일에 아래의 설정을 추가합니다.

```
# setting for CORS
map $request_method $cors_method {
    OPTIONS 11;
    GET 1;
    POST 1;
    default 0;
}

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

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;
    server_name _;
    include /etc/nginx/conf.d/service-url.inc;

    location / {
        proxy_pass $fe_service_url;
    }
}
```

```

# Virtual Host configuration
##### www.pinn.kr
server {
    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    server_name www.pinn.kr; # managed by Certbot

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

    location / {
        proxy_pass $fe_service_url;
    }

    # setting for fe->be API request
    location /api/ {
        proxy_pass http://127.0.0.1:8081/;
        proxy_redirect default;
        charset utf-8;

        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_f
orwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy true;

        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/www.pinn.kr/fullc
hain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/www.pinn.kr/p

```

```

    rivkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = www.pinn.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name www.pinn.kr;
    return 404; # managed by Certbot
}

##### pinn.kr
server {
    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;
    server_name pinn.kr; # managed by Certbot

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

    location / {
        proxy_pass $fe_service_url;
    }

    location /api/ {
        proxy_pass http://127.0.0.1:8081/;
        proxy_redirect default;
        charset utf-8;

        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_f

```

```

    forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-NginX-Proxy true;

    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

}

listen [::]:443 ssl; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/pinn.kr/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/pinn.kr/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = pinn.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name pinn.kr;
    return 404; # managed by Certbot
}

```

3. nginx 재시작

```
$ sudo systemctl start nginx
```

Grafana & Prometheus

URL : <http://3.34.108.114:3001>

1. 서버 내에 `prometheus.yml` 파일 생성

```
# prometheus.yml

scrape_configs:
  - job_name: 'prometheus'
    metrics_path: '/actuator/prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['http://3.34.108.114:8081']
```

1. 서버 내에 `docker-compose-monitoring.yml` 파일 생성

```
# docker-compose-monitoring.yml

version: '3'

services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    user: "1000:1000" # UID:GID
    ports:
      - "3001:3000" # 기본 포트 3000 > 3001으로 포트 변경
    volumes:
      - ./grafana-data:/var/lib/grafana
```

```
depends_on:
  - prometheus
```


3. prometheus 실행

```
$ docker compose -f docker-compose-monitoring.yml up -d
```

4. 서버의 3001 포트에 Grafana에 접속한 뒤, `admin / admin` 으로 로그인
5. Grafana에 로그인한 뒤, Administration → Data Sources에서 Prometheus를 등록하고, Prometheus server url에 `http://localhost:9090` 을 입력
6. Grafana Labs에서 Spring boot 을 검색하여, `Spring boot 2.1 System Monitor` 선택 후 dashboard ID 복사

Dashboards | Grafana Labs

Browse a library of official and community-built dashboards.

 <https://grafana.com/grafana/dashboards/>



7. Import dashboard → 복사한 dashboard ID 입력 / 등록된 Prometheus를 data source로 등록 후 Load