



포팅매뉴얼

1. 개요

1-1. 프로젝트 개요

1-2. 개발 환경

Backend

Frontend

Server

Service

1-3. 프로젝트 사용도구

1-4. 외부 서비스

1-5. GitIgnore 정보

2. 빌드

2-1. 환경변수

Spring

React

2-2. 빌드하기

Frontend

Backend

2-3. 외부 서비스 이용방법

3. 배포하기

nginx proxy manager 설치 및 설정

젠킨스 설정

젠킨스 백엔드 자동배포 설정

젠킨스 react + nginx (프론트 부분) 자동배포

소나큐브

fastapi 설정

redis 설정

1. 개요

1-1. 프로젝트 개요

주플릭스는 사용자가 주식을 적립식으로 구매하고, 개인의 투자 히스토리를 기반으로 한 주식 예측 정보가 모여져있는 커뮤니티를 형성합니다. 투자자들에게 주식 시장에 대한 깊이 있는 이해와 함께, 장기적인 자산 증식의 기회를 제공하려고 합니다. 또한, 사용자 간의 정보 공

유를 통해 보다 신뢰할 수 있는 투자 결정을 내릴 수 있도록 돕습니다. 더불어 해외 증시를 알 수 있는 해외뉴스 라디오봇과 ZBTI 를 통한 개인 맞춤형 포트폴리오를 제공합니다.

1-2. 개발 환경

Backend

- **Java** : Oracle Open JDK 17.0.9
- **Spring Boot** : 3.2.1
- **JPA** : hibernate-core-6.4.1
- **DB**: MySQL 8.0
- **IntelliJ** : 2023. 3

Frontend

- **Node.js** : 20.10.0
- **TypeScript** : 4.9.5
- **React** : 18.2.9
- **Recoil** : 0.7.7
- **Axios** : 1.6.7
- **Vscode** : 1.85.1

Server

- **AWS EC2**

Service

- **NginX** : 1.24.0
- **nginx-proxy-manager** : 2.11.1
- **Docker** : 24.0.5
- **Jenkins** : lts-jdk17
- **Sonarqube** : lts-community

1-3. 프로젝트 사용도구

- 이슈 관리 : JIRA
- 형상 관리 : Gitlab
- 코드 리뷰: Gerrit
- 커뮤니케이션 : Notion, Mattermost
- 디자인 : Figma
- UCC : 모바비
- CI/CD : Jenkins

1-4. 외부 서비스

- 한국투자증권 Open API
- 파파고 Open API
- 클로바 보이스 Open API

1-5. Gitignore 정보

- React : .env (최상단 위치)
- Spring : application.yml

2. 빌드

2-1. 환경변수

Spring

- application.yml
- 여기 아래 민감정보 빼기

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://mysql서버ip:포트/DB명?serverTimezone=UTC&
```

```
    username: 계정이름
    password: 계정비밀번호
jpa:
  generate-ddl: true
  hibernate:
    ddl-auto: update
  show-sql: false
  database: mysql
  open-in-view: false
  database-platform: org.hibernate.dialect.MySQL8Dialect
  properties:
    hibernate:
      format_sql: true
cache:
  type: redis
redis:
  host: redis 서버 아이피
  port: redis 서버 포트
  password: redis 비밀번호

server:
  port: 백엔드 서버 포트

jwt:
  secret:

security:
  db:
    aes:
      private-key: aes에 사용할 키
  api:
    aes:
      private-key: aes에 사용할 키

python:
  endpoint:
    news:
      crawling: http://127.0.0.1:8000/radio/crawling/endpoint
```

```

translation: http://127.0.0.1:8000/translation/endpoint
summary: http://127.0.0.1:8000/radio/summary/endpoint
tts: http://127.0.0.1:8000/radio/tts/endpoint
indices: http://127.0.0.1:8000/get_indices
predict:
  value: http://127.0.0.1:8000/get_closing_price
  graph: https://127.0.0.1:8000/generate_stock_graph
  compare: https://127.0.0.1:8000/compare_graph
  search: https://127.0.0.1:8000/get_stock_search
  price: http://127.0.0.1:8000/get_now_price
news:
  url: https://www.cnbc.com/world-markets/
ppg:
  url: https://naveropenapi.apigw.ntruss.com/nmt/v1/translation
  clientId: api 아이디
  clientSecret: api 시크릿키
summary:
  url: https://naveropenapi.apigw.ntruss.com/text-summary/v1
  clientId: api 아이디
  clientSecret: api 시크릿키
tts:
  url: https://naveropenapi.apigw.ntruss.com/tts-premium/v1
  clientId: api 아이디
  clientSecret: api 시크릿키

```

- application-prod.yml (로컬 말고 ec2 같은 곳에서 하는 경우)
 - fastapi있는 서버의아이피와 포트 : 도커의 경우 컨테이너 이름으로 ip 접근 가능했음.
 - fastapi 작동하면 들어갈 수 있는 페이지 주소 : fastapi 컨테이너 실행하면 해당 포트로 페이지 접속이 가능함. 이 프로젝트에선 그걸 nginx proxy manager로 도메인에 할당했는데, 이 주소를 입력해야함. 파이썬으로 그린 그래프 같은걸 가져와야 해서. xxx.duckdns.org 이렇게

```

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver

```

```

url: jdbc:mysql://mysql서버ip:포트/DB명?serverTimezone=UTC&
username: 계정이름
password: 계정비밀번호
jpa:
  generate-ddl: true
  hibernate:
    ddl-auto: update
  show-sql: false
  database: mysql
  open-in-view: false
  database-platform: org.hibernate.dialect.MySQL8Dialect
  properties:
    hibernate:
      format_sql: true
cache:
  type: redis
redis:
  host: redis 서버 아이피
  port: redis 서버 포트
  password: redis 비밀번호

server:
  port: 백엔드 서버 포트

jwt:
  secret:

security:
  db:
    aes:
      private-key: aes에 사용할 키
  api:
    aes:
      private-key: aes에 사용할 키

python:
  endpoint:
    news:

```

```

crawling: http://fastapi있는 서버의아이피와 포트/radio/crawl
translation: http://fastapi있는 서버의아이피와 포트/translation
summary: http://fastapi있는 서버의아이피와 포트/radio/summary
tts: http://fastapi있는 서버의아이피와 포트/radio/tts/endpoint
indices: http://fastapi있는 서버의아이피와 포트/get_indices
predict:
  value: http://fastapi있는 서버의아이피와 포트/get_closing_price
  graph: https://fastapi 작동하면 들어갈 수 있는 페이지 주소/get_graph
  compare: https://fastapi 작동하면 들어갈 수 있는 페이지 주소/compare
  search: https://fastapi 작동하면 들어갈 수 있는 페이지 주소/get_search
  price: http://fastapi있는 서버의아이피와 포트/get_now_price
news:
  url: https://www.cnbc.com/world-markets/
ppg:
  url: https://naveropenapi.apigw.ntruss.com/nmt/v1/translation
  clientId: api 아이디
  clientSecret: api 시크릿키
summary:
  url: https://naveropenapi.apigw.ntruss.com/text-summary/v1
  clientId: api 아이디
  clientSecret: api 시크릿키
tts:
  url: https://naveropenapi.apigw.ntruss.com/tts-premium/v1
  clientId: api 아이디
  clientSecret: api 시크릿키

```

React

- .env : 최상단 위치

```
REACT_APP_HOME_URL=http://localhost:8089
```

- EC2에 있는 .env

```
REACT_APP_HOME_URL=https://백엔드 서버 아이피
```

2-2. 빌드하기

Frontend

- `npm i`
- `npm start` / `npm run build`

Backend

- `build.gradle` 실행

2-3. 외부 서비스 이용방법

- 한국투자증권 OpenAPI
 - 한국투자증권 계좌번호, User App key, User Secret key 필요
 - Access Token은 1일 마다 새로 발급
- 파파고 Open API
 - Papago Text Translation
 - 네이버 클라우드에서 어플리케이션 등록 → 발급된 Client Id, Client Secret 필요
- 클로바 Open API
 - Summary, Voice
 - 네이버 클라우드에서 어플리케이션에서 Web 서비스 Url 등록 → 발급된 Client Id, Client Secret 필요

3. 배포하기

- ec2에 접속 후 도커 설치

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl sof
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud
sudo add-apt-repository "deb [arch=amd64] https://download.do
```



```
sudo apt update
sudo apt install docker-ce
sudo usermod -aG docker 유저이름
```

- 설치하고 터미널 재시작
- ec2에 docker compose 설치 및 docker-compose.yml 작성.
 - mysql 설정 부분을 보면 포트가 열려있는데, 개발 단계에서만 이렇게 하고 실제로는 로컬에서만 외부 포트를 개방해야 한다.

```
// 그대로 실행하면 된다.
sudo curl -L https://github.com/docker/compose/releases/latest
// 권한 설정
sudo chmod +x /usr/local/bin/docker-compose

// docker-compose.yml
version: '3.8'
volumes:
  zooflix-volume:
    external: true
    name: zooflix-volume
  zooflix-jenkins-volume:
    external: true
    name: zooflix-jenkins-volume
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  postgresql:
  postgresql_data:
  redis_volume:
  redis_data:
services:
  zooflix_nginx_proxy_manager:
    container_name: zooflix-nginx-proxy-manager-c
    image: 'jc21/nginx-proxy-manager:2.11.1'
    restart: always
    ports:
      - "80:80"
```

```

        - "443:443"
        - "127.0.0.1:8009:81"
    volumes:
        - ./data:/data
        - ./letsencrypt:/etc/letsencrypt
    environment:
        TZ: Asia/Seoul
        DISABLE_IPV6: "true"
zooflix_mysql:
    container_name: zooflix-mysql-container
    image: mysql:8.0.30
    ports:
        - "3306:3306"
    volumes:
        - zooflix-volume:/var/lib/mysql
    environment:
        TZ: Asia/Seoul
        MYSQL_HOST: zooflix_mysql
        MYSQL_PORT: 3306
        MYSQL_ROOT_PASSWORD: malang
    restart: always
    command:
        - --character-set-server=utf8mb4
        - --collation-server=utf8mb4_unicode_
        - --character-set-client-handshake=FA
zooflix_jenkins:
    image: jenkins/jenkins:lts-jdk17
    container_name: zooflix-jenkins-container
    restart: always
    environment:
        - TZ=Asia/Seoul
    user: root
    privileged: true
    ports:
        - "127.0.0.1:8080:8080"
        - "127.0.0.1:50000:50000"
    volumes:
        - zooflix-jenkins-volume:/var/jenkins_home

```

```

- /home/ubuntu/docker-compose.yml:/docker-compo
- ./config/application-prod.yml:/config/applica
- ./config/react.env:/config/react.env
- /usr/bin/docker:/usr/bin/docker
- /usr/local/bin/docker-compose:/usr/local/bin/
- /var/run/docker.sock:/var/run/docker.sock
zooflix_frontend:
  image: zooflix-frontend
  container_name: zooflix-frontend-container
  restart: always
  environment:
    - TZ=Asia/Seoul
zooflix_backend:
  image: zooflix-backend
  container_name: zooflix-backend-container
  restart: always
  ports:
    - "127.0.0.1:8089:8089"
  environment:
    - TZ=Asia/Seoul
zooflix_sonarqube:
  image: sonarqube:lts-community
  ports:
    - "127.0.0.1:9000:9000"
  container_name: zooflix-sonarqube-container
  restart: always
  environment:
    - sonar.jdbc.url=jdbc:postgresql://postgres:543
    - sonar.jdbc.username=malang
    - sonar.jdbc.password=malang203
    - TZ=Asia/Seoul
  volumes:
    - sonarqube_data:/opt/sonarqube/data
    - sonarqube_extensions:/opt/sonarqube/extension
    - sonarqube_logs:/opt/sonarqube/logs
zooflix_postgres:
  image: postgres:14.2
  ports:

```

```

        - "127.0.0.1:5432:5432"
    container_name: zooflix-postgres-container
    restart: always
    environment:
        - POSTGRES_USER=malang
        - POSTGRES_PASSWORD=malang203
        - POSTGRES_DB=zooflix
        - TZ=Asia/Seoul
    volumes:
        - postgresql:/var/lib/postgresql
        - postgresql_data:/var/lib/postgresql/data
zooflix_fastapi:
    image: zooflix-fastapi
    container_name: zooflix-fastapi-container
    restart: always
    ports:
        - "127.0.0.1:8000:8000"
    environment:
        - TZ=Asia/Seoul
zooflix_redis:
    image: redis:7.2
    container_name: zooflix-redis-container
    restart: always
    ports:
        - "127.0.0.1:6000:6379"
    volumes:
        - redis_data:/data
        - redis_volume:/usr/local/conf/redis.conf
    command: redis-server /usr/local/conf/redis.conf
    environment:
        - TZ=Asia/Seoul
        - REDIS_PASSWORD=malanggnalam

```

- docker-compose.yml 에서 mysql 부분 외의 서비스나 volume는 잠시 주석 처리하고 mysql만 docker-compose 로 실행해본다.

```
docker-compose up -d
```

- mysql 접속







```
docker exec -it zooflix-mysql-container bash
mysql -u root -p
비번 입력
// cmd에서 해도 됨.
mysql 워크벤치에서 zooflix db 생성.
create database zooflix;
유저 생성하고 zooflix 테이블 사용 권한 부여
GRANT ALL PRIVILEGES ON db명.* TO '유저'@'%';
```

- yml에서 포트를 개방해줬으면 ec2 아이피로 mysql 접근이 가능.

nginx proxy manager 설치 및 설정

- docker-compose.yml에 nginx proxy manager 서비스 입력한 다음 docker-compose up -d 로 실행하면 알아서 다운받음.
- 도메인:8009 해서 페이지 접속.
 - 기본이메일 : admin@example.com
 - 기본패스워드 : changeme
- Hosts → Proxy Hosts 여기서 프록시 설정 가능.
 - 스키마는 http
 - 포워드 호스트 / ip 는 연결할 ip 쓰거나 도커 컨테이너 이름 써도 가능
 - 포워드 포트는 들어갈 컨테이너의 내부 포트
- SSL 인증서 추가
 - NPM에서 SSL Certificates 메뉴에 가서 add SSL Certificate
 - **Let's Encrypt** 선택
 - 도메인 이름은 *.zooflix~~랑 zooflix~~ 2개 추가. zooflix~~는 내가 duckdns에서 발급받은 주소.
 - use a DNS Challenge 체크
 - DNS Provider 는 duckdns 선택
 - **Credentials File Content** 생기는데, dns_duckdns_token=your-duckdns-token 여기서 your-~를 duckdns 페이지에 있는 내 토큰으로 바꿔주자.

- 이제 내가 추가하는 프록시 호스트에 edit 하면 SSL 부분 있는데, 여기서 보면 내가 만든 인증서가 들어 있음.
- duckdns는 도메인 생성하고 10분 이후에 ssl 인증서 등록 가능.
- 프록시는 http 그대로 냅두고 그냥 ssl만 추가하면 이후에 도메인으로 접속하면 자동으로 https로 접속함.
- nginx proxy manager 사진.

Proxy Hosts					Q Search Host...	?	Add Proxy Host
SOURCE	DESTINATION	SSL	ACCESS	STATUS			
 api.zooflix.duckdns.org Created: 21st March 2024	http://zooflix-backend-container:8089	Let's Encrypt	Public	● Online	⋮		
 fast.zooflix.duckdns.org Created: 25th March 2024	http://zooflix-fastapi-container:8000	Let's Encrypt	Public	● Online	⋮		
 jk.zooflix.duckdns.org Created: 22nd March 2024	http://zooflix-jenkins-container:8080	Let's Encrypt	Public	● Online	⋮		
 npm.zooflix.duckdns.org Created: 21st March 2024	http://zooflix-nginx-proxy-manager-container:81	Let's Encrypt	Public	● Online	⋮		
 sonar.zooflix.duckdns.org Created: 22nd March 2024	http://zooflix-sonarqube-container:9000	Let's Encrypt	Public	● Online	⋮		
 zooflix.duckdns.org Created: 24th March 2024	http://zooflix-frontend-container:80	Let's Encrypt	Public	● Online	⋮		

젠킨스 설정

- 도커 컴포즈 실행하면 이미지는 알아서 받는다.
- ec2 아이피:8080 으로 접근 가능.
- 맨 처음 나오는 화면에서 비밀번호가 어떤 파일에 있는지 알려줌
- `docker exec -it zooflix-jenkins-container bash` 로 젠킨스 컨테이너 접속하고 해당 위치로 이동해서 cat로 확인.

젠킨스 백엔드 자동배포 설정

- 젠킨스 페이지에서 jenkins 관리 → 플러그인 → available 플러그인 에서 gitlab, Generic webhook trigger, ssh agent plugin(얘는 컨테이너 안에서 ec2에 접속해서 쉘 명령을 하기 위한 용도) 플러그인 설치
- jenkins 관리 → credentials → (global) → Add Credentials

- 젠킨스 홈에서 새로운 item 에서 파이프라인 이름 정하고 Pipeline 누르기. 이 프로젝트에선 **zooflixBackendPipeline**
- 파이프라인 구성 들어가면 아래에 스크립트 있는데, 거기에 작성.

```

pipeline {
    agent any

    stages {
        stage('gitlab clone') {
            steps {
                echo '클론을 시작할게요!'
                git branch: 'release-be', credentialsId: 'zoo
                echo '클론을 완료했어요!'
            }
        }
        stage('fastapi docker build') {
            steps {
                echo '빌드를 시작할게요!'
                sh '''
                cd /var/jenkins_home/workspace/zooflixBackend
                docker build -t zooflix-fastapi .
                '''
                echo '빌드를 완료했어요!'
            }
        }
        stage('build') {
            steps {
                echo '빌드를 시작할게요!'
                sh '''
                chmod +x /var/jenkins_home/workspace/zooflixB
                cd /var/jenkins_home/workspace/zooflixBackend
                cp /config/application-prod.yml /var/jenkins_
                /var/jenkins_home/workspace/zooflixBackendPip
                '''
                echo '빌드를 완료했어요!'
            }
        }
        stage("sonarqube") {
    
```

```

        steps{
            script{
                def scannerHome = tool 'zooflix-sonarqube
                withSonarQubeEnv(credentialsId:"SONAR_TOK
                sh "${scannerHome}/bin/sonar-scanner -Dso
                }
            }
        }
    }
    stage('make image') {
        steps {
            echo '이미지를 만들게요!'
            sh '''
            docker stop zooflix-backend-container || true
            docker rm zooflix-backend-container || true
            docker rmi zooflix-backend || true
            docker build -t zooflix-backend ./back
            '''
            echo '이미지를 만들었어요!'
        }
    }
    stage('run container') {
        steps {
            sh '''
            docker-compose -p zooflix up -d zooflix_backe
            docker-compose -p zooflix up -d zooflix_fasta
            '''
            echo '컨테이너를 재가동시켜요!'
        }
    }
}
}
}

```

- 위쪽에 Build Triggers 설정 부분에서 Build when a change is pushed to GitLab. GitLab webhook 체크. 그럼 새로운 체크 박스 생기는데, push events, opened merge request events 체크 되어 있는거 확인.
- Build Triggers 부분 아래 보면 고급 버튼이 있는데, 이거 누르면 아래에 secret token 이 생김. 생성하고 gitlab의 webhook 만들 때 사용함.

- 깃랩에 가서 왼쪽 메뉴의 setting → webhooks 가서 add new webhook 클릭.
- regular expression은 적용할 브랜치 이름.

젠킨스 react + nginx (프론트 부분) 자동배포

- 젠킨스 파이프라인 프론트 용으로 생성. 깃랩의 webhook 도 용으로 생성
- 프론트 파이프라인 (플젝에선 zooflixFrontendPipeline)

```

pipeline {
    agent any
    tools {nodejs "nodejs-tool"}
    stages {
        stage('gitlab clone') {
            steps {
                echo '클론을 시작할게요!'
                git branch: 'release-fe', credentialsId: 'zoo
                echo '클론을 완료했어요!'
            }
        }
        stage('docker build') {
            steps {
                echo '빌드를 시작할게요!'
                sh '''
                cd /var/jenkins_home/workspace/zooflixFronten
                cp /config/react.env /var/jenkins_home/worksp
                docker build -t zooflix-frontend .
                '''
                echo '빌드를 완료했어요!'
            }
        }
        stage("sonarqube") {
            steps{
                script{
                    def scannerHome = tool 'zooflix-sonarqube
                    withSonarQubeEnv(credentialsId:"SONAR_TOK
                    sh "${scannerHome}/bin/sonar-scanner -Dso
                }
            }
        }
    }
}

```

```

    }
  }
  stage('run container') {
    steps {
      sh '''
        docker-compose -p zooflix up -d zooflix_front
        '''
      echo '컨테이너를 재가동시켜요!'
    }
  }
}
}
}

```

소나큐브

- postgresql 설정
 - 맨 처음 설치하면 mysql 처럼 사용자 생성하고 권한 주고 기본 작업을 좀 해야함.
 - 그래서 기본 계정인 postgres 로 설정을 잠깐 바꿔서 접속한 후 계정 생성을 하자.
 - user, password 둘 다 postgres로 바꾸고 실행.

```
docker exec -it zooflix-postgres-container psql -U postgres
```

- 위 명령어로 접속.
- 아래 명령어들로 기본적인 설정.

```

create database db명; // db 생성
\l // db 리스트
create role 아이디 with login password '비번';
\du // 계정 및 권한 확인
grant all privileges on database zooflix to 유저; // 유저에게

```

- 이후 계정 설정 원상 복귀 하자.
- 9090 포트로 접속하면 아이디 비번 입력창 나옴. 초기는 admin admin임.
- 비밀번호는 malang로 변경함.

- 소나큐브 접속해서 other CI 에서 zooflix-sonarqube 프로젝트명으로 추가. 프론트 전용 소나큐브 프로젝트임.
 - other ci 눌러서 토큰 발급.
 - 젠킨스에 **Credentials** 추가
 - Kind는 Secret text
 - Secret는 소나큐브에서 발급한 토큰
 - 나머지 아이디와 설명
 - 이제 젠킨스 관리 → system 가면 **SonarQube servers** 설정하는 부분이 있음.
 - Environment variables 체크
 - Name 원하는 대로, Server URL은 내가 소나큐브 접속하는 url
 - Server authentication token은 내가 위에서 추가한 토큰으로 선택.

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables

SonarQube installations

List of SonarQube installations

Name

sonarqube

Server URL

Default is http://localhost:9000

http://j10b203.p.ssafy.io:9000/

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

소나큐브 토큰

+ Add ▾

- 백엔드도 소나큐브 추가.
 - 소나 큐브 프로젝트 생성. 이름은 zooflix-sonarqube-backend
 - 젠킨스 관리 → **Credentials** → **global** → **Add Credentials**

- kind : Secret text
- Secret : 위에 소나 큐브 프로젝트의 토큰 키
- 나머지는 설명
- 젠킨스 관리 → system 에서 소나큐브 서버 하나 추가.
 - Name : 원하는대로. 난 sonarqube_backend
 - URL은 소나큐브 url
 - Server authentication token : 방금 만든 백엔드용 **Credentials**

fastapi 설정

- 프로젝트 클론 받고 docker-compose.yml 열썬하면 도커 컴포즈 up 하면 다 된다.

redis 설정

- 마찬가지로 클론, 도커 컴포즈