punpun

포팅 매뉴얼

김민재, 박소연, 박소희, 박정은, 이햇살

목차

1. 배포 상세 내용

본 문서는 punpun 서비스를 사용하기 위한 배포 과정에 대해 서술하고 있습니다.

1-1. 주요 기술

BackEnd-Spring

```
InteliJ IDE
Springboot Gradle 6.8.3
Java jdk 11
Spring Data JPA
querydsl JPA 5.0.0
Spring oauth2 client
Spring Security
jjwt 0.2
Spring validation
Spring web
Spring Kafka
Spring devtools
junit5
gson 2.8.9
lombok
springfox-boot-starter 3.0.0
springfox-swagger-ui 3.0.0
spring Cloud aws 2.2.6
```

BackEnd-DB

```
h2
postgresql
```

BackEnd-etc

```
kafka
nginx
docker
jenkins
go
```

FrontEnd-React

```
Visual Studio Code IDE
react 18.2.0
node 18.15.10
jquery 3.5.16
recoil 0.7.7
typescript 4.9.5
tailwindcss 3.2.7
axios 1.3.4
```

1-2. 백엔드 배포 과정

Docker 설치

```
sudo apt update -y
```

```
sudo apt-get install ./docker-desktop-<version>-<arch>.deb
```

```
systemctl --user start docker-desktop
```

Git 설치

```
sudo apt install git -y
```

Git repository clone

```
git clone --recurse-submodules
https://lab.ssafy.com/s08-bigdata-dist-sub2/S08P22D109.git
```

Postgre extension 설치

```
docker exec -it postgres psql -U postgres -d postgres -c "CREATE
EXTENSION cube; CREATE EXTENSION earthdistance"
```

Docker compose 실행

```
cd ./punpun
```

```
docker-compuse up -d
```

docker compose.yml

```
version: '2'
services:
  zookeeper:
    image: wurstmeister/zookeeper
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      - TZ=Asia/Seoul
  kafka:
    image: wurstmeister/kafka
    container_name: kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: kafka
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      TZ: Asia/Seoul
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
  postgresql:
    image: postgres
    container_name: postgres
    ports:
      - "5433:5432"
    environment:
      - POSTGRES_PASSWORD=[password]
      - TZ=Asia/Seoul
    volumes:
      - /home/ubuntu/db:/var/lib/postgresql/data
  punpun:
    image: parksohui/punpun
    container_name: punpun
    ports:
      - "8888:8888"
    environment:
      - TZ=Asia/Seoul
  alarm:
    image: limecats/go-alarm
    container_name: alarm
    environment:
      - TZ=Asia/Seoul
```

Nginx & SSL 설정

docker로 **Nginx** 설치

```
docker pull nginx
```

Nginx와 **mount**할 디렉토리 생성

```
mkdir nginx_home
```

Let's Encrypt 설치

```
apt-get install letsencrypt
```

인증서 적용 및 **.pem** 키 발급

```
letsencrypt certonly --standalone -d [도메인]
```

Nginx 도커 **container** 실행

```
docker run -d --name nginx --network host -v
home/ubuntu/nginx_home:/var/nginx_home -v
/etc/letsencrypt:/etc/letsencrypt -v
/var/lib/letsencrypt:/var/lib/letsencrypt nginx
```

Nginx container로 이동

```
sudo docker exec -it --user root nginx /bin/bash
```

디렉토리 생성

```
cd ./etc/nginx
```

```
mkdir cd sites-enables
```

```
cd sites-enables
```

config.conf 파일 생성

```
vi config.conf
```

config.conf

```
server {
        location /{

                root /var/nginx_home;
                try_files $uri $uri/ /index.html;
        }
        location /api {
                proxy_pass http://localhost:8888/api;
                proxy_connect_timeout 300;
                proxy_send_timeout 300;
                proxy_read_timeout 300;
                send_timeout 300;
        }

        listen 443 ssl;
        ssl_certificate
/etc/letsencrypt/live/j8d109.p.ssafy.io/fullchain.pem;
        ssl_certificate_key
/etc/letsencrypt/live/j8d109.p.ssafy.io/privkey.pem;
}

server {
        if ($host = j8d109.p.ssafy.io){
                return 301 https://$host$request_uri;
        }

        listen 80;
        server_name j8d109.p.ssafy.io;
        return 404;

}
```

nginx reload

```
nginx -s reload
```

1-3. 프론트 배포 과정

Build

```
cd ./punpun/front
```

```
npm install --legacy-peer-deps
```

```
npm run build
```

Build한 정적파일을 **nginx**와 연결된 **volumn**으로 이동

```
sudo cp -r /home/ubuntu/jenkins_home/workspace/Front/front/build/*
/nginx_home
```

2. DB 및 프로퍼티 정의된 파일

main : application.yml

```
spring:
  profiles:
    group:
      deploy: # 배포 환경 설정
        - common
        - deploy_db
        - deploy_kafka
        - deploy_security

---
# common part
spring:
  config:
    activate:
      on-profile: "common"
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

server:
  port: 8888
  servlet:
```

```
    contextPath: /api

jwt:
  secret: {jwt key}

cloud:
  aws:
    credentials:
      accessKey: {aws accesskey}
      secretKey: {aws secretkey}
    s3: #버킷이름
      bucket: {aws bucketName}
    region: #S3 지역
      static: {aws region}
    stack:
      auto: false
---
spring:
  config:
    activate:
      on-profile: "deploy_db"
  datasource:
    url: jdbc:postgresql://postgres:5432/postgres
    username: postgres
    password: {docker compose 파일의 이름 값 사용}
    driver-class-name: org.postgresql.Driver
  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update

logging:
  level:
    root: info
---

spring:
  config:
    activate:
      on-profile: "deploy_kafka"
  kafka:
    consumer:
      bootstrap-servers: kafka:9092
      group-id: alarm
      auto-offset-reset: earliest
      key-deserializer:
```

```
org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer:
org.springframework.kafka.support.serializer.JsonDeserializer
    producer:
      bootstrap-servers: kafka:9092
      key-serializer:
org.apache.kafka.common.serialization.StringSerializer
      value-serializer:
org.springframework.kafka.support.serializer.JsonSerializer
---
spring:
  config:
    activate:
      on-profile: "deploy_security"
  security:
    oauth2:
      client:
        registration:
          kakao:
            authorization-grant-type: authorization_code
            client-id: {oauth client-id}
            client-secret: {oauth client-secret}
            redirect-uri:
"https://j8d109.p.ssafy.io/api/login/oauth2/code/kakao"
            scope:
              - account_email
              - profile_nickname
              - profile_image
            client-authentication-method: POST
            client-name: Kakao
        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
---
```

test: application.yml

```
spring:
  datasource:
    url:
jdbc:postgresql://[도메인]:5433/postgres?currentSchema=publictest
    username: postgres
    password: {docker compose 파일의 이름 값 사용}
    driver-class-name: org.postgresql.Driver
  jpa:
    defer-datasource-initialization: true
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        show_sql: true
        format_sql: true
  kafka:
    bootstrap-servers: localhost:9092
    producer:
      key-serializer:
org.apache.kafka.common.serialization.StringSerializer
      value-serializer:
org.springframework.kafka.support.serializer.JsonSerializer
    consumer:
      auto-offset-reset: earliest
      key-deserializer:
org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer:
org.springframework.kafka.support.serializer.JsonDeserializer
  security:
    oauth2:
      client:
        registration:
          kakao:
            authorization-grant-type: authorization_code
            client-id: {oauth client-id}
            client-secret: {oauth client-secret}
            redirect-uri:
"http://localhost:8888/api/login/oauth2/code/kakao"
            scope:
              - account_email
              - profile_nickname
              - profile_image
            client-authentication-method: POST
            client-name: Kakao
```

```
        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
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

server:
  port: 8888
  servlet:
    encoding:
      force-response: true

logging:
  level:
    org:
      hibernate:
        type:
          descriptor:
            sql: trace

jwt:
  secret: {jwt key}

cloud:
  aws:
    credentials:
      accessKey: {aws accesskey}
      secretKey: {aws secretKey}
    s3: #버킷이름
      bucket: {aws bucketName}
    region: #S3 지역
      static: {aws region}
    stack:
      auto: false
```

3. 외부 서비스

- 소셜 로그인
  - KaKao: Oauth 기반 소셜 로그인 API 제공
  - https://developers.kakao.com

- 문자 서비스
  - Coolsms : 문자 발송
  - https://coolsms.co.kr