****

**DDOKBUN**

**포팅 매뉴얼**

목차

Ⅰ. 개요

1. 프로젝트 소개
2. 개발 환경
3. 기술 스택
4. 외부 서비스

**Ⅱ. 포팅 가이드**

1. 환경 설정
2. 빌드 및 배포
3. CI/CD 설정

Ⅰ. 개요

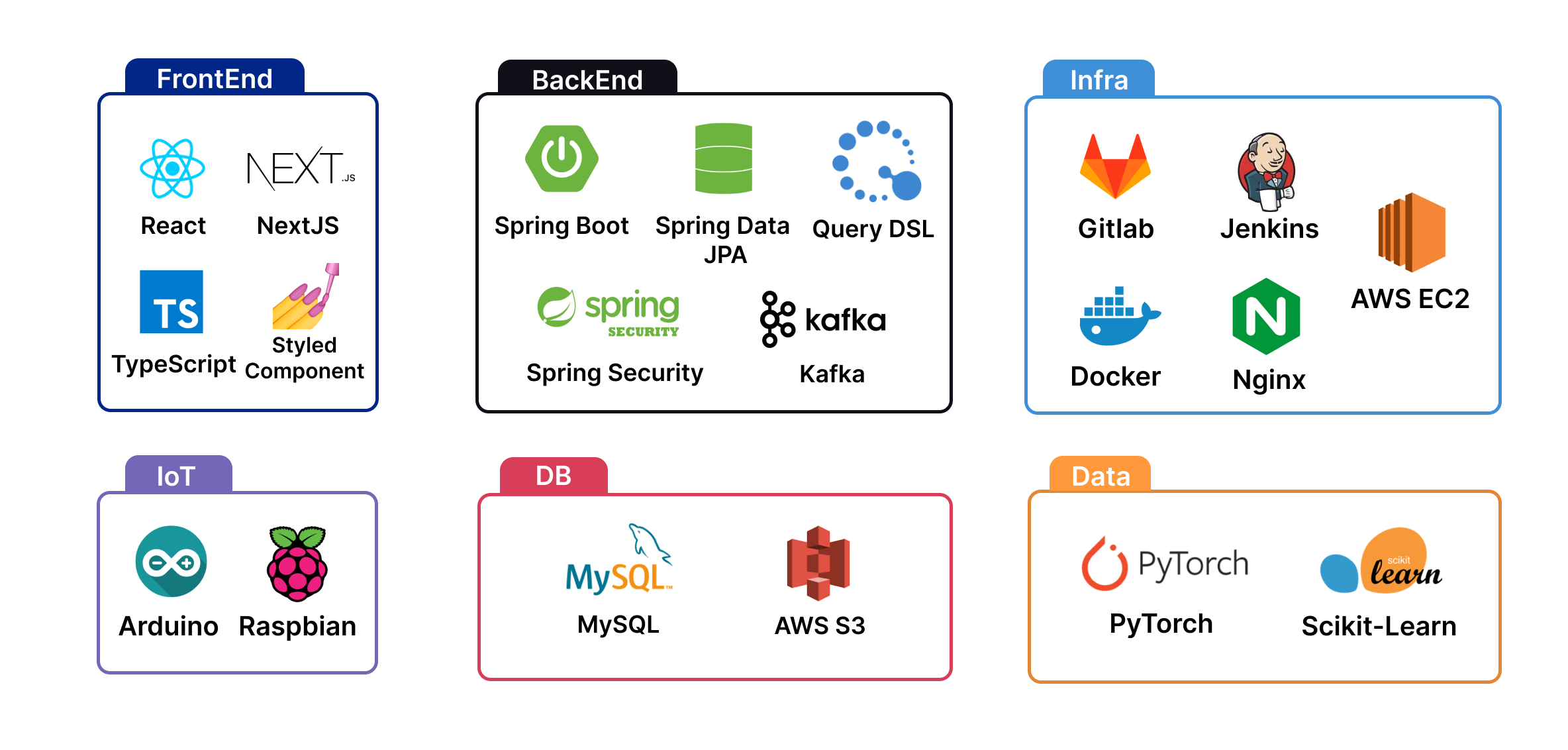
1. 프로젝트 소개

Ddokbun은 스마트 화분을 통한 식물 관리 서비스를 제공하고, 식물, 화분을 판매하는 E커머스 플랫폼입니다. 사용자는 본인의 Life 스타일에 맞게 식물을 추천받을 수 있으며, 화분 관리 시스템을 통해 효율적으로 식물을 관리할 수 있습니다.

2. 개발 환경

* IntelliJ IDEA 2022.2.1
* VS Code 1.66.0
* JAVA 11
* Spring Boot 2.7.4
* MySQL Workbench 8.0.29
* AWS EC2 Ubuntu 20.04 LTS
* React : 18.2
* Next.js : 12.3.1
* Typescript : 4.8.4
* Javascript : ES6
* 양식의 맨 위

3. 기술 스택



4. 외부 서비스

* Google OAuth : 구글 로그인 - application-oauth.yml
* Kakao OAuth : 카카오 로그인 - application-oauth.yml
* Redis : E-커머스 인기 조회 캐시 - application.yml
* S3 : 이미지 저장 스토리지 - application-aws.yml
* Kafka : IoT 메시지 브로커 - application-kafka.yml
* Firebase : 알림 메시지 - application-fcm.yml
* 카카오페이
* 네이버페이

**Ⅱ. 포팅 가이드**

1. 환경 설정

* **Backend (Spring boot)**
* MySQL, REDIS 설정

|  |
| --- |
| backend/ddokbun/src/main/resources/application.yml |
| spring:  …  # mysql DB  datasource:  driver-class-name: com.mysql.cj.jdbc.Driver  <url:jdbc:mysql://>**{도메인주소}**/**{데이터베이스명}**?serverTimezone=Asia/Seoul  username: **{Id}**  password: **{Password}**  …  # redis  redis:  host: **{redis 도메인주소}**  [port:](url:jdbc:mysql://) **{redis port 번호}**  … |

* AWS S3 설정

|  |
| --- |
| backend/ddokbun/src/main/resources/application-aws.yml |
| cloud:  aws:  credentials:  accessKey: **{AWS IAM AccessKey}**  secretKey: **{AWS IAM SecretKey}**  s3:  bucket: **{bucket}**  … |

* FCM 설정

|  |
| --- |
| backend/ddokbun/src/main/resources/application-fcm.yml |
| fcm:  auth:  file\_path: **{file path (fcm-json)파일}**  project:  id: **{projectId}**  message:  scope: 'https://www.googleapis.com/auth/firebase.messaging'  endpoint: [https://fcm.googleapis.com/v1/project/**{project**](https://fcm.googleapis.com/v1/project/%7bproject)**Id}**/messages:send |

* Kafka 설정

|  |
| --- |
| backend/ddokbun/src/main/resources/application-kafka.yml |
| spring:  kafka:  producer:  bootstrap-servers:  - **{도메인 주소}:{포트번호1}**  - **{도메인 주소}:{포트번호2}**  - **{도메인 주소}:{포트번호3}**  key-serializer: org.apache.kafka.common.serialization.StringSerializer  value-serializer: org.apache.kafka.common.serialization.StringSerializer |

* OAuth 설정

|  |
| --- |
| backend/ddokbun/src/main/resources/application-oauth.yml |
| spring:  security:  oauth2:  client:  registration:  kakao:  client-id: **{client-id : 카카오 REST API키}**  client-secret: **{client-secret키}**  redirect-uri: **{Redirect-uri 주소}**  authorization-grant-type: authorization\_code  google:  client-id: **{client-id : REST API키}**  client-secret: **{client-secret키}**  redirect-uri: **{Redirect-uri 주소}**  … |

* **DB (MySQL)**

Infra/db/docker-compose

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | version:'3' | |  | services: | |  | db: | |  | image: mysql:8.0 | |  | container\_name: mysq | |  | ports: | |  | - **{설정 port 번호}**:3306 | |  | environment: | |  | MYSQL\_USER: **{USER}** | |  | MYSQL\_PASSWORD: **{PASSWORD}** | |  | MYSQL\_ROOT\_PASSWORD: **{PASSWORD}** | |  | TZ: Asia/Seoul | |  | volumes: | |  | - ./mysqldata:/var/lib/mysql | |  | - ./db/mysql/sql:/sql | |  | - ./db/mysql/init:/docker-entrypoint-initdb.d | |  | restart: always | |

2. 빌드 및 배포

A. MySQL DB 서버

B. Frontend 배포

C. Backend API 서버 배포

D. Backend Fast-API 서버 배포

E. Kafka 클러스터 배포

F. Redis 서버 배포

G. nginx + certbot 설정

**A. Mysql 서버 배포**

Repository : **deploy/mysql-compose-set/**

1. docker-compose.yml
2. ddokbun.sql

|  |
| --- |
| $ docker-compose up –d |

**B. Frontend 배포**

Repository : **frontend/**

**빌드**

|  |
| --- |
| $ npm install  $ npm run build |

**배포**

|  |
| --- |
| $ docker build -t frontend ./  $ docker run -dp **{port번호}:**3000 --name web frontend |

**C. Backend API 서버 배포**

Repository : **backend/ddokbun/**

**빌드**

|  |
| --- |
| $ chmod +x gradlew  $ ./gradlew clean build –x test |

**배포**

|  |
| --- |
| $ docker build -t api\_server ./  $ docker run -dp **{port번호}:**8080 --name api\_server api\_server |

**D. Backend FAST-API 서버 배포**

Repository : **deploy/fast-api-compose-set/**

**배포**

|  |
| --- |
| $ docker-compose up –build -d |

**E. Kafka 클러스터 배포**

Repository : **deploy/zk-kafka-compose-set/**

**배포**

|  |
| --- |
| $ export DOCKER\_HOST\_IP=**{도메인 주소}**  $ docker-compose up —build -d |

**F. Redis 서버 배포**

**배포**

|  |
| --- |
| $ docker run --name redis -p 6379:6379 -v /redis –d  redis:latest redis-server --appendonly yes |

**G. Nginx + Certbot 설정**

**1. SSL 인증서 발급 + Nginx 배포 Code**

Repository : deploy/nginx-compose-set/

1. data/nginx/conf.d/app.conf

2. docker-compose.yml

3. init-letsencript.sh

**2. Code 실행**

|  |
| --- |
| $ ./init-letsencrypt.sh |

**3. HTTPS server 설정**

|  |
| --- |
| $ docker ps  $ docker exec –it {Container Id} /bin/sh |

**Nginx container 내부**

|  |
| --- |
| # vi /etc/nginx/nginx.conf |
| **/etc/nginx/nginx.conf** |
| http{  ...  server {  listen 80;  listen [::]:80;  server\_name {도메인 주소};  location /.well-known/acme-challenge/ {  allow all;  root /var/www/certbot;  }  location / {  return 301 https://$host$request\_uri;  }  }  server {  listen 443 ssl;  server\_name {도메인 주소};  server\_tokens off;  ssl\_certificate /etc/letsencrypt/live/{도메인 주소}/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/{도메인 주소}/privkey.pem;  include /etc/letsencrypt/options-ssl-nginx.conf;  ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;  **#Frontend**  location / {  proxy\_pass http://172.20.0.1:**{frontend포트}**/;  proxy\_set\_header Host $http\_host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  }    **#Backend API 서버**  location /api {  proxy\_pass http://172.20.0.1:**{backend포트}**/;  proxy\_set\_header Host $http\_host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  }  **#Backend FAST-API 서버**  location /fast-api {  proxy\_pass http://172.20.0.1:**{fast-api포트}**/;  proxy\_set\_header Host $http\_host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  }  }} |

**3. CI/CD 설정**

By Jenkins – 무중단 배포

**A. Frontend 파이프라인**

|  |
| --- |
| #1  echo "#1 Check Current Container Port ----------------"  CURRENT\_PORT=$(docker port frontend)  echo "> 현재 구동중인 PORT : ${CURRENT\_PORT:(-4)}"  echo ""  #2  echo "#2 Allocate Target Port -------------"  if [ "${CURRENT\_PORT:(-4)}" == "3001" ];  then  echo "> 기존 애플리케이션의 포트는 3001입니다."  TARGET\_PORT=3002  elif [ "${CURRENT\_PORT:(-4)}" == "3002" ]  then  echo "> 기존 애플리케이션의 포트는 3002입니다."  TARGET\_PORT=3001  else  echo "> 현재 구동 중인 애플리케이션의 포트를 찾는데 실패하였습니다."  echo "> 3001 포트를 할당합니다."  TARGET\_PORT=3002  fi  echo ""  #3  echo "#3 Docker Image Build & Test-Container Run :: React Server--------------"  docker build -t react:0.1 ./  docker run -d -p $TARGET\_PORT:3000 --name frontend\_test react:0.1  echo ""  #4  echo "#4 Test-Container Connection Test --------------"  # 5  echo "#5 Change Nginx Setting : Current -> Test ------------------------------"  echo "> 전환할 Port: $TARGET\_PORT"  echo "> Port 전환"  echo "set \$service\_port2 http://172.20.0.1:${TARGET\_PORT};" |tee ~/workspace/nginx/service-url2.inc  # 6  echo "> Nginx Reload"  docker restart nginx  PROXY\_PORT=$(docker port frontend\_test)  echo "> Nginx Current Proxy Port: ${PROXY\_PORT:(-4)}"  # 7  echo "기존 구동 중인 container : frontend\_test"  docker stop frontend && docker rm frontend ||true  docker rename frontend\_test frontend  PROXY\_PORT=$(docker port frontend)  echo "> Nginx Current Proxy Port: ${PROXY\_PORT:(-4)}" |

**B. Backend 파이프라인**

|  |
| --- |
| #1  echo "#1 Check Current Container Port ----------------"  CURRENT\_PORT=$(docker port backend)  echo "> 현재 구동중인 PORT : ${CURRENT\_PORT:(-4)}"  echo ""  #2  echo "#2 Allocate Target Port -------------"  if [ "${CURRENT\_PORT:(-4)}" == "8111" ];  then  echo "> 기존 애플리케이션의 포트는 8111입니다."  TARGET\_PORT=8222  elif [ "${CURRENT\_PORT:(-4)}" == "8222" ]  then  echo "> 기존 애플리케이션의 포트는 8222입니다."  TARGET\_PORT=8111  else  echo "> 현재 구동 중인 애플리케이션의 포트를 찾는데 실패하였습니다."  echo "> 8111 포트를 할당합니다."  TARGET\_PORT=8111  fi  echo ""  #3  echo "#3 Docker Image Build & Test-Container Run :: Spring Boot Server--------------"  docker build -t springboot:0.1 ./  docker run -d -p $TARGET\_PORT:8080 --name backend\_test springboot:0.1  echo ""  #4  echo "#4 Test-Container Connection Test --------------"  echo "> $TARGET\_PORT 10초 후 Health check 시작"  echo "> curl -s http://172.20.0.1:$TARGET\_PORT/api/infra/health "  sleep 5  for retry\_count in {1..10}  do  response=$(curl -s http://172.20.0.1:$TARGET\_PORT/api/infra/health)  sleep 1  up\_count=$(echo $response | grep 'UP' | wc -l)  if [ $up\_count -ge 1 ]  then # $up\_count >= 1 ("UP" 문자열이 있는지 검증)  echo "> Health check 성공"  break  else  echo "> Health check의 응답을 알 수 없거나 혹은 status가 UP이 아닙니다."  echo "> Health check: ${response}"  fi  if [ $retry\_count -eq 10 ]  then  echo "> Health check 실패. "  echo "> Nginx에 연결하지 않고 배포를 종료합니다."  exit 1  fi  echo "> Health check 연결 실패. 재시도..."  sleep 10  done  echo ""  # 5  echo "#5 Change Nginx Setting : Current -> Test ------------------------------"  echo "> 전환할 Port: $TARGET\_PORT"  echo "> Port 전환"  echo "set \$service\_port http://172.20.0.1:${TARGET\_PORT};" |tee ~/workspace/nginx/service-url.inc  # 6  echo "> Nginx Reload"  docker restart nginx  PROXY\_PORT=$(docker port backend\_test)  echo "> Nginx Current Proxy Port: ${PROXY\_PORT:(-4)}"  # 7  echo "기존 구동 중인 container : backend\_test"  docker stop backend && docker rm backend ||true  docker rename backend\_test backend  PROXY\_PORT=$(docker port backend)  echo "> Nginx Current Proxy Port: ${PROXY\_PORT:(-4)}" |