개밥바라기

포팅 매뉴얼



김선형 김하늘 박영서

박준형 배찬일 유승아

1. 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전(IDE포함) 기재

OS : window 10

IDE : Intellij, VSCode

Build Automation Tool : Gradle(8.1.1)

Frontend : Node.js(v18.17.0), React

Backend : Spring boot, spring data jpa, spring security

Database : h2(develop), postgre(deploy)

JVM version : azul-17

WS: nginx(reverse froxy)

WAS : tomcat(spring-boot)

CI/CD : Jenkins

1. 빌드 시 사용되는 환경 변수 등의 주요 내용 상세 기재
2. VERSION=$(curl—silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r) : Docker-compose의 최신 릴리즈 정보
3. DESTINATION=/usr/bin/docker-compose : docker-compose 실행 파일이 저장될 위치
4. Docker-compse –p frontend build : Docker-compose는 보통 docker-compose.yml이 저장 된 디렉토리 명을 환경변수로 설정하는데 본 프로젝트에서는 docker-compose.yml이 root에 있어 이러한 자동 환경변수 설정이 되지 않기 때문에 –p 를 이용해 frontend라고 수동으로 명명해주었다

배포 순서

1. Sudo docker-compose –p frontend build
2. Sudo docker-compose –p frontend up –d
3. Sudo service nginx restart

현재 띄우고 있는 도커 컨테이너

1. 배포 시 특이사항 기재
2. Docker 사용 : Backend, Frontend, Nginx, Jenkins, YoloV5, OpenVidu, DB를 각각의 Docker Container로 관리하여 독립성을 보장시켰다.
3. Docker-compose 사용 : Docker-compose를 사용하여 각각의 Dockerfile을 통해 여러 개의 컨테이너를 build 후 생성하여 다중 컨테이너 관리, 설정의 일관성, 개발 효율성을 증대시켰다.
4. Jenkins pipeline 사용 : jenkins pipeline으로 한 프로젝트에 있는 backend단과 frontend단을 동시에 자동 배포 하여 배포 환경 테스트 효율성을 증가시켰다.
5. Nginx의 reverse froxy 사용 : 가비아에서 구입한 doggy-yummy 도메인 뒤에 /, /api, /v1 경로로 reverse froxy 하여 각각 front, back, object detection 단으로 보내주었다.
6. DB 접속 정보 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록
7. Application.properties

spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
  
  
  
#postgre  
spring.datasource.driverClassName=org.postgresql.Driver  
spring.datasource.url=jdbc:postgresql://db:5432/postgres  
spring.datasource.username=postgres  
spring.datasource.password=1234ab!@  
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.properties.hibernate.format\_sql=true  
spring.jpa.properties.hibernate.show\_sql=true  
# ##logging.level.org.hibernate.type.descriptor.sql=trace;  
  
spring.jpa.hibernate.ddl-auto=create  
  
spring.profiles.include=secure, openvidu  
  
spring.mail.host=smtp.gmail.com  
spring.mail.port=587  
  
server.port = 8083

1. Application-openvidu.properties

# openvidu server  
openvidu.url=http://doggy-yummy.site:5443/  
openvidu.secret=DOGGYYUMMY

1. Application-secure.properties

#Address  
local.front=http://localhost:3000  
local.back=http://localhost:8083/api  
deploy.front=https://doggy-yummy.site  
deploy.back=https://doggy-yummy.site/api  
  
current.front=${deploy.front}  
current.back=${deploy.back}  
  
#JWTtoken configurations  
jwt.secretKey=eyJhbGciOiJIUzUxMaFmvNiklhQeg  
jwt.access.expiration=3600000  
jwt.access.header=Authorization  
jwt.refresh.expiration=1209600000  
jwt.refresh.header=Authorization-refresh  
  
#Email verification  
spring.mail.username=doggy.yummy.site@gmail.com  
spring.mail.password=uoupgwlvfahfcurs  
spring.mail.properties.mail.smtp.starttls.enable=true  
spring.mail.properties.mail.smtp.starttls.required=true  
spring.mail.properties.mail.smtp.auth=true  
  
#OAuth2 google  
spring.security.oauth2.client.registration.google.client-id= 574121728411-uucqo7jo05qvic8q7s5k476mibsiqpoa.apps.googleusercontent.com  
spring.security.oauth2.client.registration.google.client-secret=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  
spring.security.oauth2.client.registration.google.scope=profile,email  
spring.security.oauth2.client.registration.google.redirect-uri=${current.back}/login/oauth2/code/google  
  
#OAuth2 naver  
spring.security.oauth2.client.registration.naver.client-id=PuYjPTicjpjHA39Rhbzl  
spring.security.oauth2.client.registration.naver.client-secret=19u9Np6\_tG  
spring.security.oauth2.client.registration.naver.client-name=Naver  
spring.security.oauth2.client.registration.naver.authorization-grant-type=authorization\_code  
spring.security.oauth2.client.registration.naver.redirect-uri=${current.back}/login/oauth2/code/naver  
  
spring.security.oauth2.client.provider.naver.authorization-uri=https://nid.naver.com/oauth2.0/authorize  
spring.security.oauth2.client.provider.naver.token-uri=https://nid.naver.com/oauth2.0/token  
spring.security.oauth2.client.provider.naver.user-info-uri=https://openapi.naver.com/v1/nid/me  
spring.security.oauth2.client.provider.naver.user-name-attribute=response  
  
#OAuth2 kakao  
spring.security.oauth2.client.registration.kakao.client-id=d51edc27643f41b4ddf5ec13ca6b1b55  
spring.security.oauth2.client.registration.kakao.client-secret=EzGmTEjY0bN87sXWqrETHIKgnAU6Txl0  
spring.security.oauth2.client.registration.kakao.redirect-uri=${current.back}/login/oauth2/code/kakao  
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization\_code  
spring.security.oauth2.client.registration.kakao.client-authentication-method=client\_secret\_post  
spring.security.oauth2.client.registration.kakao.client-name=Kakao  
spring.security.oauth2.client.registration.kakao.scope=profile\_nickname, profile\_image, account\_email  
  
spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize  
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token  
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me  
spring.security.oauth2.client.provider.kakao.user-name-attribute=id