포팅 메뉴얼

기술 스택 및 버전

Front-End

- Next.js 14.2.2
- Typescript 5.2.2
- React 18.2.0
- Recoil 0.7.7
- Node.js 20.10.0
- Node Package Manager 10.3.0

Back-End

- Java 17, 21
- Spring Boot 3.2.5
  - dependency-management 1.1.4
- Spring Data JPA 3.2.5
- Spring Batch 5
- Querydsl 5.0.0
- Spring Cloud Openfeign
- Feign OKhttp 13.2.1
- OkHttp 13.2.1
- Jjwt 0.9.1
- Spring Webflux 3.2.5

- Spring Data Redis 3.2.5
- Spring Cloud Netflix Eureka Server, Client
- Spring for Apache Kafka
- Spring Cloud AWS 2.4.4
- Spring Boot Actuator
- Micrometer

DB

- MariaDB 11.3.2
- Redis 7.2.4

Infra

- Ubuntu 20.04.6
- Docker 26.1.1
- Docker Compose 2.27.0
- Jenkins 2.440.3
- Portainer Community Edition 2.19.5
- Portainer Agent 2.19.5
- Nginx Proxy Manager 2.11.2
- Apache Kafka 3.7.0
- UI for Apache Kafka 0.7.2
- Prometheus 2.52.0
- Grafana 10.4.3
- Node Exporter 1.8.0

환경 변수

Front-End

.env.production

```
NEXT_PUBLIC_CLIENT_ID=[구글 client-id]
NEXT_PUBLIC_BASE_URL=[서버 API 요청 도메인]
NEXT_PUBLIC_REDIRECT_URL=[소셜 로그인 시 필요한 redirect url]
```

DockerFile

```
# 빌드를 위한 이미지
FROM node:20.10.0-alpine

# root에 /app 폴더 생성
RUN mkdir /app

WORKDIR /app

# 환경 변수 설정
ENV NODE_ENV=production

# 필요한 파일 복사
COPY . /app

# 포트 열기
EXPOSE 3000

# 서버 실행
CMD ["npm", "start"]
```

Back-End

application.yml

```
spring:
  profiles:
    active: deploy

  application:
    name: funco
```

application-deploy.yml

- API Gateway

```
spring:
  config:
    activate:
      on-profile: deploy

  jwt:
    token:
      secret-key: QTEwN2phbnNvcnJ5dGVhbWRuZmxya2RsZW1kZ2tmcmp
      refresh-secret-key: QTEwN2phbnNvcnJ5dGVhbWZsdm1mcHRubHh

server:
  port: 8010

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.name}
```

- Auth Service

```
spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: create
    show-sql: true
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: true

  data:
    redis:
      host: k10a302.p.ssafy.io
      port: 6379
      password: funco302

  jwt:
    token:
      secret-key: QTEwN2phbnNvcnJ5dGVhbWRuZmxya2RsZW1kZ2tmcmp
      refresh-secret-key: QTEwN2phbnNvcnJ5dGVhbWZsdm1mcHRubHh

  oauth:
    google:
      client-id: 298082348622-3gp2shappr5oth3a6ad9k2hvie46qck
      client-secret-id: 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
      redirect-uri: https://funco.co.kr/redirect
      scope: profile,email

  cloud:
    openfeign:
      okhttp:
        enabled: true # patch를 쓰기 위해 feign okhttp 설정

server:
  port: 8001

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: auth.funco.co.kr
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

# feign client 상세 로그 출력 하도록 설정
logging:
  level:
    com.found_404.funco.trade.client: DEBUG

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.name}
```

- Member Service

```
spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://member-mariadb:3306/funco # mariadb 접
    username: funco-admin # mariadb 접속 시 입력할 username 정보
    password: funco302 # mariadb 접속 시 입력할 password 정보

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: false
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: false

  data:
    redis:
      host: main-redis
      port: 6379
      password: funco302

  batch:
    jdbc:
      initialize-schema: always

  cloud:
    openfeign:
      okhttp:
        enabled: true # patch를 쓰기 위해 feign okhttp 설정

cloud:
  aws:
    credentials:
      accessKey: AKIA47CRVMDAYIPQNO5J
      secretKey: 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
    s3:
      bucket: fonco-image
    region:
      static: ap-northeast-2
    stack:
      auto: false

    # kafka
  kafka:
    bootstrap-servers: 54.180.242.193:8092,54.180.242.193:809
    producer:
      retries: 3 # 재시도 횟수
      properties:
        linger.ms: 1
    client-id: kafka-member-producer
    listener:
      ack-mode: record # 메시지를 하나씩 처리하고 바로바로 커밋하는 방

server:
  port: 8002

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: member.funco.co.kr
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

# feign client 상세 로그 출력 하도록 설정
logging:
  level:
    com.found_404.funcomember.feignClient.client: DEBUG

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.name}
```

- Note Service

```
server:
  port: 8007

spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://note-mariadb:3306/funco # mariadb 접속
    username: funco-admin # mariadb 접속 시 입력할 username 정보
    password: funco302 # mariadb 접속 시 입력할 password 정보

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: false
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: false

  cloud:
    openfeign:
      okhttp:
        enabled: true # patch를 쓰기 위해 feign okhttp 설정

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: note.funco.co.kr
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

# feign client 상세 로그 출력 하도록 설정
logging:
  level:
    com.found_404.funco.feignClient.client: DEBUG
```

```
cloud:
  aws:
    credentials:
      accessKey: AKIA47CRVMDAYIPQNO5J
      secretKey: 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
    s3:
      bucket: fonco-image
    region:
      static: ap-northeast-2
    stack:
      auto: false

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.name}
```

- Asset Service

```
spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://asset-mariadb:3306/funco # mariadb 접
    username: funco-admin # mariadb 접속 시 입력할 username 정보
```

```yaml
    password: funco302 # mariadb 접속 시 입력할 password 정보

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: true
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: false

  cloud:
    openfeign:
      okhttp:
        enabled: true # patch를 쓰기 위해 feign okhttp 설정

  kafka:
    bootstrap-servers: 54.180.242.193:8092,54.180.242.193:809
    consumer:
      auto-offset-reset: earliest
      group-id: asset-consumer-group
      key-deserializer: org.apache.kafka.common.serialization
      value-deserializer: org.springframework.kafka.support.s
      properties:
        spring.json.trusted.packages: '*'

server:
  port: 8008

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: asset.funco.co.kr
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

# feign client 상세 로그 출력 하도록 설정
logging:
  level:
    com.found_404.funco.client: DEBUG

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info

  metrics:
    tags:
      application: ${spring.application.name}
```

- Trade Service

```
server:
  port: 8004

spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://trade-mariadb:3306/funco # mariadb 접
```

```
      username: funco-admin # mariadb 접속 시 입력할 username 정보
      password: funco302 # mariadb 접속 시 입력할 password 정보

    jpa:
      generate-ddl: true
      hibernate:
        ddl-auto: update
      show-sql: false
      defer-datasource-initialization: true
      properties:
        hibernate:
          format_sql: false

    cloud:
      openfeign:
        okhttp:
          enabled: true # patch를 쓰기 위해 feign okhttp 설정

    # kafka
    kafka:
      bootstrap-servers: 54.180.242.193:8092,54.180.242.193:809
      producer:
        retries: 3 # 재시도 횟수
        properties:
          linger.ms: 1
      client-id: kafka-trade-producer
      listener:
        ack-mode: record # 메시지를 하나씩 처리하고 바로바로 커밋하는 방

  #  data:
  #    redis:
  #      host: main-redis
  #      port: 6379
  #      password: funco302

  eureka:
    client:
      fetch-registry: true
      register-with-eureka: true
      service-url:
        defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

    instance:
      hostname: trade.funcoin.duckdns.org
      prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
      secure-port-enabled: true
      secure-port: 443

  # feign client 상세 로그 출력 하도록 설정
  logging:
    level:
      com.found_404.funco.feignClient.client: DEBUG

  management:
    endpoints:
      web:
        exposure:
          include: prometheus, health, info
    metrics:
      tags:
        application: ${spring.application.name}
```

- Follow Service

```
server:
  port: 8007

spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://follow-mariadb:3306/funco # mariadb 접
    username: funco-admin # mariadb 접속 시 입력할 username 정보
    password: funco302 # mariadb 접속 시 입력할 password 정보

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: false
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: false

  cloud:
    openfeign:
      okhttp:
        enabled: true # patch를 쓰기 위해 feign okhttp 설정

  # kafka
  kafka:
    bootstrap-servers: 54.180.242.193:8092,54.180.242.193:809
    producer:
      retries: 3 # 재시도 횟수
      properties:
        linger.ms: 1
    client-id: kafka-follow-producer
    listener:
      ack-mode: record # 메시지를 하나씩 처리하고 바로바로 커밋하는 방

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: follow.funcoin.duckdns.org
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

# feign client 상세 로그 출력 하도록 설정
logging:
  level:
    com.found_404.funco.client: DEBUG

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.name}
```

- Notification Service

```
server:
  port: 8009

spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://notification-mariadb:3306/funco # mar
    username: funco-admin # mariadb 접속 시 입력할 username 정보
    password: funco302 # mariadb 접속 시 입력할 password 정보

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: false
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: false

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: notification.funcoin.duckdns.org
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.name}
```

- Rank Service

```
spring:
  config:
    activate:
      on-profile: deploy

  mvc:
    servlet:
      path: /api

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://rank-mariadb:3306/funco # mariadb 접속
    username: funco-admin # mariadb 접속 시 입력할 username 정보
    password: funco302 # mariadb 접속 시 입력할 password 정보

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: true
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: false

  data:
    redis:
      host: 3rd-redis
      port: 6379
      password: funco302

  cloud:
    openfeign:
      okhttp:
        enabled: true # patch를 쓰기 위해 feign okhttp 설정
```

```
server:
  port: 8006

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: rank.leetag.duckdns.org
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

# feign client 상세 로그 출력 하도록 설정
logging:
  level:
    com.found_404.funco.client: DEBUG

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.name}
```

- Statistics Service

```
spring:
  config:
    activate:
      on-profile: deploy
  mvc:
    servlet:
      path: /api

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver # Database를 ma
    url: jdbc:mariadb://statistics-mariadb:3306/funco # maria
    username: funco-admin # mariadb 접속 시 입력할 username 정보
    password: funco302 # mariadb 접속 시 입력할 password 정보

  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: true
    defer-datasource-initialization: true
    properties:
      hibernate:
        format_sql: fals
  data:
    redis:
      host: 3rd-redis
      port: 6379
      password: funco302

  cloud:
    openfeign:
      okhttp:
        enabled: true

server:
  port: 8005

eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: https://eureka.funco.co.kr/eureka/ # eurek

  instance:
    hostname: statistics.leetag.duckdns.org
    prefer-ip-address: false  # IP 주소 대신 호스트명을 사용
    secure-port-enabled: true
    secure-port: 443

# feign client 상세 로그 출력 하도록 설정
logging:
  level:
    com.found_404.funco.client: DEBUG # open feign client 모아

management:
  endpoints:
    web:
      exposure:
        include: prometheus, health, info
  metrics:
    tags:
      application: ${spring.application.nam
```

Dockerfile

```
FROM docker

FROM openjdk:17-jdk

EXPOSE {서버 포트}

ARG JAR_FILE=build/libs/funco-{서비스명}-0.0.1-SNAPSHOT.jar

ENV SPRING_PROFILES_ACTIVE=deploy

ADD ${JAR_FILE} funco-{서비스명}.jar
```

```
ENTRYPOINT ["java", "-jar", "/funco-{서비스명}.jar"]
```

Infra

docker-compose.portainer.yml

```
version: "3"

services:
  portainer:
    image: 'portainer/portainer-ce:latest'
    container_name: portainer
    privileged: true
    ports:
      - '8443:9443'
      - '8000:8000'
    volumes:
      - "/home/ubuntu/docker/portainer:/data"
      - "/var/run/docker.sock:/var/run/docker.sock"
    networks:
      - npm-network
    restart: always

volumes:
  portainer_data:

networks:
  npm-network:
    external: true
    name: npm-network
```

Main EC2 docker.compose.yml

```
version: '3.8'
services:
  npm:
    container_name: npm
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '8881:81'
      - '443:443'
    volumes:
      - /home/ubuntu/docker/npm:/data
      - /home/ubuntu/docker/npm/letsencrypt:/etc/letsencrypt
    environment:
      - TZ=Asia/Seoul
    networks:
      - npm-network
    stdin_open: true
    tty: true

  jenkins:
    image: jenkins/jenkins:lts-jdk17
    container_name: jenkins
    restart: unless-stopped
    environment:
      TZ: Asia/Seoul
      JENKINS_OPTS: --httpPort=8080
    user: root
    privileged: true
    ports:
      - 8888:8080
      - 50000:50000
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /home/ubuntu/docker/jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/local/bin/docker-compose:/usr/local/bin/docker-c
    networks:
      - npm-network
    stdin_open: true
    tty: true

  funco-mariadb:
    image: mariadb
    container_name: funco-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3305:3306"
    volumes:
      - /home/ubuntu/docker/funco-mariadb/conf.d:/etc/mysql/c
      - /home/ubuntu/docker/funco-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true

  member-mariadb:
    image: mariadb
    container_name: member-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3306:3306"
    volumes:
      - /home/ubuntu/docker/member-mariadb/conf.d:/etc/mysql/
      - /home/ubuntu/docker/member-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true

  note-mariadb:
    image: mariadb
    container_name: note-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3307:3306"
    volumes:
      - /home/ubuntu/docker/note-mariadb/conf.d:/etc/mysql/co
      - /home/ubuntu/docker/note-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true

  asset-mariadb:
    image: mariadb
    container_name: asset-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3308:3306"
    volumes:
      - /home/ubuntu/docker/asset-mariadb/conf.d:/etc/mysql/c
      - /home/ubuntu/docker/asset-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true
  redis-funco:
    image: redis
    container_name: redis-funco
    ports:
      - "6380:6379"
    command: redis-server --requirepass "funco302"
    networks:
      - npm-network
    stdin_open: true
    tty: true

  main-redis:
    image: redis
    container_name: main-redis
    ports:
      - "6379:6379"
    command: redis-server --requirepass "funco302"
    networks:
      - npm-network
    stdin_open: true
    tty: true



  discovery-service:
    image: devjy/funco-eureka
    container_name: discovery-service
    ports:
      - "8761:8761"
    environment:
      - TZ=Asia/Seoul
    networks:
      - npm-network
    stdin_open: true
    tty: true



  node-exporter:
    image: prom/node-exporter
    container_name: node-exporter
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - '--path.procfs=/host/proc'
      - '--path.rootfs=/rootfs'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|p
    ports:
      - 8100:9100
    networks:
      - npm-network

networks:
  npm-network:
    external: true
    name: npm-network
```

2nd EC2 docker-compose.yml

```
version: '3.8'
services:
  npm:
    container_name: npm
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '8881:81'
      - '443:443'
    volumes:
      - /home/ubuntu/docker/npm:/data
      - /home/ubuntu/docker/npm/letsencrypt:/etc/letsencrypt
    environment:
      - TZ=Asia/Seoul
    networks:
      - npm-network
    stdin_open: true # docker run -i
    tty: true        # docker run -t

  jenkins:
    image: jenkins/jenkins:lts-jdk17
    container_name: jenkins
    restart: unless-stopped
    environment:
      TZ: Asia/Seoul
      JENKINS_OPTS: --httpPort=8080
    user: root
    privileged: true
    ports:
      - 8888:8080
      - 50000:50000
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /home/ubuntu/docker/jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/local/bin/docker-compose:/usr/local/bin/docker-c
    networks:
      - npm-network
    stdin_open: true
    tty: true

  trade-mariadb:
    image: mariadb
    container_name: trade-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3306:3306"
    volumes:
      - /home/ubuntu/docker/funco-mariadb/conf.d:/etc/mysql/c
      - /home/ubuntu/docker/funco-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true

  follow-mariadb:
    image: mariadb
    container_name: follow-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3312:3306"
    volumes:
      - /home/ubuntu/docker/follow-mariadb/conf.d:/etc/mysql/
      - /home/ubuntu/docker/follow-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true

  notification-mariadb:
    image: mariadb
    container_name: notification-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3313:3306"
    volumes:
      - /home/ubuntu/docker/notification-mariadb/conf.d:/etc/
      - /home/ubuntu/docker/notification-mariadb:/var/lib/mys
    networks:
      - npm-network
    stdin_open: true
    tty: true

  portainer_agent:
    image: portainer/agent:2.19.5
    container_name: portainer_agent
    restart: always
    ports:
      - "9001:9001"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /var/lib/docker/volumes:/var/lib/docker/volumes
    networks:
      - npm-network
    stdin_open: true # docker run -i
    tty: true        # docker run -t


  node-exporter:
    image: prom/node-exporter
    container_name: node-exporter
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - '--path.procfs=/host/proc'
      - '--path.rootfs=/rootfs'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|p
    ports:
      - 8100:9100
    networks:
      - npm-network

networks:
  npm-network:
    external: true
    name: npm-network
```

3rd EC2 docker-compose.yml

```
version: '3.8'
services:
  npm:
    container_name: npm
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '8881:81'
      - '443:443'
    volumes:
      - /home/ubuntu/docker/npm:/data
      - /home/ubuntu/docker/npm/letsencrypt:/etc/letsencrypt
    environment:
      - TZ=Asia/Seoul
    networks:
      - npm-network
    stdin_open: true
    tty: true

  jenkins:
    image: jenkins/jenkins:lts-jdk17
    container_name: jenkins
    restart: unless-stopped
    environment:
      TZ: Asia/Seoul
      JENKINS_OPTS: --httpPort=8080
    user: root
    privileged: true
    ports:
      - 8888:8080
      - 50000:50000
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /home/ubuntu/docker/jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/local/bin/docker-compose:/usr/local/bin/docker-c
    networks:
      - npm-network
    stdin_open: true
    tty: true

  statistics-mariadb:
    image: mariadb
    container_name: statistics-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3310:3306"
    volumes:
      - /home/ubuntu/docker/statistics-mariadb/conf.d:/etc/my
      - /home/ubuntu/docker/statistics-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true

  rank-mariadb:
    image: mariadb
    container_name: rank-mariadb
    environment:
      MARIADB_DATABASE: "funco"
      MARIADB_USER: "funco-admin"
      MARIADB_PASSWORD: "funco302"
      MYSQL_ROOT_PASSWORD: "funco302"
    ports:
      - "3311:3306"
    volumes:
      - /home/ubuntu/docker/rank-mariadb/conf.d:/etc/mysql/co
      - /home/ubuntu/docker/rank-mariadb:/var/lib/mysql
    networks:
      - npm-network
    stdin_open: true
    tty: true

  3rd-redis:
    image: redis
    container_name: 3rd-redis
    ports:
      - "6379:6379"
    command: redis-server --requirepass "funco302"
    networks:
      - npm-network
    stdin_open: true
    tty: true

  portainer-agent:
    image: portainer/agent:2.19.5
    container_name: portainer-agent
    restart: always
    ports:
      - "9001:9001"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /var/lib/docker/volumes:/var/lib/docker/volumes
    networks:
      - npm-network
    stdin_open: true
    tty: true

  kafka1:
    image: bitnami/kafka:latest
    restart: unless-stopped
    container_name: kafka1
    ports:
      - '8092:8094'
    volumes:
      - /home/ubuntu/docker/kafka1:/bitnami/kafka
      - /etc/localtime:/etc/localtime
    environment:
      - KAFKA_CFG_BROKER_ID=1
      - KAFKA_CFG_NODE_ID=1
      - KAFKA_KRAFT_CLUSTER_ID=HsDBs9l6UUmQq7Y5E6bNlw
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=1@kafka1:8093,2@ka
      - ALLOW_PLAINTEXT_LISTENER=yes
      - KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=true
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:8092,CONTROLLER://:8
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka1:809
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROLLER:P
      - KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_MIN_ISR=2
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
    networks:
      - npm-network
    stdin_open: true
    tty: true

  kafka2:
    image: bitnami/kafka:latest
    restart: unless-stopped
    container_name: kafka2
    ports:
      - '8093:8094'
    volumes:
      - /home/ubuntu/docker/kafka2:/bitnami/kafka
      - /etc/localtime:/etc/localtime
    environment:
      - KAFKA_CFG_BROKER_ID=2
      - KAFKA_CFG_NODE_ID=2
      - KAFKA_KRAFT_CLUSTER_ID=HsDBs9l6UUmQq7Y5E6bNlw
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=1@kafka1:8093,2@ka
      - ALLOW_PLAINTEXT_LISTENER=yes
      - KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=true
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:8092,CONTROLLER://:8
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka2:809
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROLLER:P
      - KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_MIN_ISR=2
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
    networks:
      - npm-network
    stdin_open: true
    tty: true

  kafka3:
    image: bitnami/kafka:latest
    restart: unless-stopped
    container_name: kafka3
    ports:
      - '8094:8094'
    volumes:
      - /home/ubuntu/docker/kafka3:/bitnami/kafka
      - /etc/localtime:/etc/localtime
    environment:
      - KAFKA_CFG_BROKER_ID=3
      - KAFKA_CFG_NODE_ID=3
      - KAFKA_KRAFT_CLUSTER_ID=HsDBs9l6UUmQq7Y5E6bNlw
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=1@kafka1:8093,2@ka
      - ALLOW_PLAINTEXT_LISTENER=yes
      - KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=true
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:8092,CONTROLLER://:8
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka3:809
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROLLER:P
      - KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_MIN_ISR=2
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
    networks:
      - npm-network
    stdin_open: true
    tty: true

  kafka-ui:
    image: provectuslabs/kafka-ui:latest
    restart: always
    container_name: kafka-ui
    ports:
      - 8090:8080
    volumes:
      - /etc/localtime:/etc/localtime
    environment:
      - KAFKA_CLUSTERS_0_NAME=Local-Kraft-Cluster
      - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka1:8092,kafka2:
      - DYNAMIC_CONFIG_ENABLED=true
      - KAFKA_CLUSTERS_0_AUDIT_TOPICAUDITENABLED=true
      - KAFKA_CLUSTERS_0_AUDIT_CONSOLEAUDITENABLED=true
    depends_on:
      - kafka1
      - kafka2
      - kafka3
    networks:
      - npm-network

  prometheus:
    image: prom/prometheus
    container_name: prometheus
    user: "1000:1000"
    volumes:
      - /home/ubuntu/docker/prometheus/conf/prometheus.yml:/e
      - /home/ubuntu/docker/prometheus/data:/prometheus
    ports:
      - 8070:9090
    command:
      - '--storage.tsdb.path=/prometheus'
      - '--web.enable-admin-api'
      - '--config.file=/etc/prometheus/prometheus.yml'
    restart: always
    networks:
      - npm-network

  grafana:
    image: grafana/grafana
    container_name: grafana
    user: "1000:1000"
    ports:
      - 3300:3000
    volumes:
      - /home/ubuntu/docker/grafana:/var/lib/grafana
      - /home/ubuntu/docker/grafana/provisioning:/etc/grafana
    restart: always
    depends_on:
      - prometheus
    networks:
      - npm-network

  node-exporter:
    image: prom/node-exporter
    container_name: node-exporter
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - '--path.procfs=/host/proc'
      - '--path.rootfs=/rootfs'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|p
    ports:
      - 8100:9100
    networks:
      - npm-network
  sonarqube:
    image: sonarqube:lts
    container_name: sonarqube
    ports:
      - "9000:9000"
    ulimits:
      nofile:
        soft: "262144"
        hard: "262144"
    networks:
      - sonarnet
    environment:
      - sonar.jdbc.url=jdbc:postgresql://db:5432/sonar
    volumes:
      - sonarqube_conf:/opt/sonarqube/conf
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs

  postgres:
    image: postgres
    container_name: postgres
    ports:
      - "5432:5432"
    networks:
      - sonarnet
    environment:
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
    volumes:
      - postgresql:/var/lib/postgresql
      - postgresql_data:/var/lib/postgresql/data

networks:
  npm-network:
    external: true
    name: npm-network
```

prometheus.yml

```yaml
global:
  scrape_interval: 15s
  scrape_timeout: 15s
  evaluation_interval: 2m

  external_labels:
    monitor: 'codelab-monitor'
    query_log_file: query_log_file.log

scrape_configs:

  # Prometheus Job
  - job_name: 'Prometheus'
    scrape_interval: 10s
    scrape_timeout: 10s
    metrics_path: '/metrics'
    scheme: 'http'

    static_configs:
      - targets: ['prometheus:9090']
        labels:
          service: 'Prometheus'

  # EC2 Server Job
  - job_name: 'EC2 Server'
    scrape_interval: 10s
    scrape_timeout: 10s
    metrics_path: '/metrics'
    scheme: 'https'

    static_configs:
      - targets: ['node.funco.co.kr', 'node.leetag.duckdns.or
        labels:
          service: 'EC2 Server'
  # API Gateway Job
  - job_name: "API Gateway Server"
    metrics_path: "/actuator/prometheus"
    scheme: 'https'
    scrape_interval: 5s

    static_configs:
      - targets: ['api.funco.co.kr']
        labels:
          service: 'API Gateway'

  # Spring Boot Server Job
  - job_name: "Spring Boot Server"
    metrics_path: "/api/actuator/prometheus"
    scheme: 'https'
    scrape_interval: 5s

    static_configs:
      - targets: ['auth.funco.co.kr', 'member.funco.co.kr', '
        labels:
          service: 'Spring Boot'
```

인프라 구축하기

개요

- Portainer를 통해 컨테이너를 스택(docker compose)으로 관리
- Nginx Proxy Manager를 통해 SSL과 Reverse Proxy 적용
- Jenkins 파이프라인 설정을 통해 CI/CD 구축
- API Gateway로 API 트래픽 라우팅과 로드밸런싱
- Grafana와 Prometheus로 서버 시스템 모니터링

활용 포트

- **Main EC2**

포트 번호	활용
22	SSH
80	Nginx
443	Nginx SSL
3000	Front End
3306	Member MariaDB
3307	Note MariaDB
3308	Asset MariaDB
6379	Redis
8000	Portainer
8001	Auth Service
8002	Member Service
8003	Note Service
8008	Asset Service
8010	Spring API Gateway
8100	Node Exporter
8443	Portainer GUI
8761	Discovery Service
8881	NPM GUI
8888	Jenkins
9001	Portainer Agent
50000	Jenkins

- **2nd EC2**

포트 번호	활용
22	EC2 기본 포트
80	Nginx
443	Nginx SSL

포트 번호	활용
3306	Trade MariaDB
3312	Follow MariaDB
3313	Notification MariaDB
6379	Redis
8000	Portainer
8004	Trade Service
8007	Follow Service
8009	Notification Service
8100	Node Exporter
8443	Portainer GUI
8881	NPM GUI
8888	Jenkins
9001	Portainer Agent
50000	Jenkins

- **3rd EC2**

포트 번호	활용
22	EC2 기본 포트
80	Nginx
443	Nginx SSL
3300	Grafana
3310	Rank MariaDB
3311	Statistics MariaDB
5432	Postgres
6379	Redis
8000	Portainer
8005	Rank Service
8006	Statistics Service
8070	Prometheus
8090	Kafka UI

포트 번호	활용
8092	Kafka
8093	Kafka
8094	Kafka
8100	Node Exporter
8443	Portainer GUI
8881	NPM GUI
8888	Jenkins
9000	SonarQube
9001	Portainer Agent
50000	Jenkins

우분투 서버 세팅하기

우분투 서버 시간을 한국 표준시로 변경

```
sudo timedatectl set-timezone Asia/Seoul
```

미러 서버를 카카오 서버로 변경

- 기본 서버는 *.ubuntu.com이라는 해외 서버
- 국내망을 이용할 수 있는 카카오 미러서버를 사용
    - 패키지 갱신/다운로드 속도를 개선한다.

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.k
```

패키지 목록 업데이트

- 패키지를 받는 미러 서버가 변경되었으므로 업데이트 진행

```
sudo apt-get -y update && sudo apt-get -y upgrade
```

swap 영역 할당

- 메모리 점유율이 높은 작업이 진행될 때, 우선순위가 낮은 작업이 중단되는 현상을 방지
- 아래 순서대로 커맨드 실행
  - 용량 확인

    ```
    free -h
    ```

  - 스왑 영역 할당(예: 4GB)

    ```
    sudo fallocate -l 4G /swapfile
    ```

  - swapfile 권한 수정

    ```
    sudo chmod 600 /swapfile
    ```

  - swapfile 생성

    ```
    sudo mkswap /swapfile
    ```

  - swapfile 활성화

    ```
    sudo swapon /swapfile
    ```

  - 시스템이 재부팅 되어도 swap 유지할 수 있도록 설정

    ```
    sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/
    ```

  - swap 영역 할당 확인

    ```
    free -h
    ```

Docker

Docker 설치 전 필요한 패키지 설치

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

Docker에 대한 GPG Key 인증 진행

- OK가 뜨면 정상적으로 등록되었다는 뜻

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud
```

Docker 레포지토리 등록

- 해당 레포지토리에 이미지를 등록
- 이미지를 내려받아서 만든 도커 컨테이너가 작동할 환경에 맞추어 ARM64 혹은 AMD64 계열로 레포지토리를 등록
- Ubuntu는 AMD64 계열의 운영체제이기 때문에 AMD64 계열로 등록
- AMD64 계열

```
sudo add-apt-repository "deb [arch=amd64] https://download.do
```

- ARM64 계열

```
sudo add-apt-repository "deb [arch=arm64] https://download.do
```

Docker 패키지 설치

- 설치 전 패키지 리스트 갱신

```
sudo apt-get -y update
```

- apt-get을 이용하여 Docker 설치
    - docker-ce : Docker Community Edition
    - docker-ce-cli : Docker Community Edition의 CLI 환경에서 추가로 설치해야 하는 패키지
    - containerd.io : Docker 컨테이너 런타임

Docker 일반 유저에게 권한 부여

- Docker는 기본적으로 항상 root로 실행되어 sudo로 명령어를 입력
- 사용자를 docker 그룹에 추가하여 sudo를 생략해도 명령어를 사용할 수 있도록 허용

```
sudo usermod -aG docker ubuntu
```

- 이후 사용자 세션 로그아웃 및 재로그인

```
sudo service docker restart
```

```
exit
```

Docker 컨테이너 활용하기

- 실행되고 있는 컨테이너 목록 조회

```
docker ps -a
```

- 컨테이너 로그 조회

```
docker logs 컨테이너명
```

빌드 및 배포 : CI/CD 파이프라인 구축

개요

- 깃랩 특정 브랜치에 코드 변경이 감지되면, 파이프라인이 작동하여 빌드와 배포를 수행
- 작업 순서대로 작성

플러그인 설치

```
# ssh 커맨드 입력에 사용
SSH Agent

# docker 이미지 생성에 사용
Docker
Docker Commons
Docker Pipeline
Docker API

# 웹훅을 통해 브랜치 merge request 이벤트 발생시 Jenkins 자동 빌드에
Generic Webhook Trigger

# 타사 레포지토리 이용시 사용 (GitLab, Github 등)
GitLab
GitLab API
GitLab Authentication
GitHub Authentication

# Node.js 빌드시 사용
NodeJS
```

Jenkins- GitLab연동

Credential 등록

- Jenkins 관리 - Manage Credentials 클릭
- Stores scoped to Jenkins - Domains - (global) - Add credentials 클릭
- GitLab 계정 Credential 등록
    - Username : Gitlab 계정 아이디 입력
    - Password : Gitlab 계정 비밀번호 입력(API 토큰 발행한다면 토큰 입력)
    - ID : Credential에 대한 별칭
- GitLab 프로젝트(레포지터리) API Token 등록
    - Kind : Gitlab API token 선택
    - API tokens : Gitlab 계정 토큰 입력
    - ID : Credential에 대한 별칭

GitLab 커넥션 추가

- Jenkins 관리 - System Configuration - System 클릭
- Gitlab의 **Enable authentication for '/project' end-point** 체크
    - Connection name : Gitlab 커넥션 이름 지정
    - Gitlab host URL : Gitlab 시스템의 Host 주소 입력
    - Credentials : 조금 전 등록한 **Jenkins Credential (API Token)**을 선택
    - 이후, **Test Connection**을 눌러 Success가 뜨면 **저장** 클릭
        - 아니라면 입력한 정보를 다시 확인

파이프라인 설정 시 Jenkins Webhook Integeration 설정

- Jenkins 파이프라인 설정
    - Pipeline 아이템에 다음과 같은 설정 추가
        - General - Build Triggers
            - Build when a change is pushed to Gitlab 체크
            - Push Events 체크

            - Opened Merge Request Events 체크
            - Approved Merge Request (EE-only) 체크
            - Comments 체크
        - 고급 - Generate 클릭
            - 발행된 Secret token 복사해두고 **저장** 클릭

- GitLab Repository
    - Settings - Webhooks 클릭
        - URL : Jenkins의 Item URL 입력

```
http://[Jenkins Host]:[Jenkins Port]/project/[파이프라인
```

        - Secret token : Jenkins의 Gitlab trigger 고급 설정 중 Secret token Generate 버튼을 이용해 만든 토큰 입력
        - Trigger : Push events 체크, merge request가 되면 Jenkins 이벤트가 발동하게 할 브랜치 입력
        - SSL verification의 **Enable SSL verification** 체크
            - 이후, **Add webhook** 클릭

Jenkins-DockerHub 연동

Jenkins

- Jenkins 관리 - Security - Manage Credentials 클릭
- Stores scoped to Jenkins - Domains - (global) - Add Credentials
- Credential 정보
    - Kind : Username with password
    - Username : DockerHub에서 사용하는 계정 아이디 입력
    - Password : DockerHub에서 사용하는 Access Token 입력

    - ID : Jenkins 내부에서 사용하는 Credential 별칭 입력

DockerHub

- 레포지토리 생성
- Access Token 발급

Jenkins-Ubuntu 연동

Jenkins

- Jenkins 관리 - Security - Manage Credentials 클릭
- Stores scoped to Jenkins - Domains - (global) - Add Credentials
- Credential 정보
    - Kind : SSH Username with private key
    - ID : Jenkins에서 Credential에 지정할 별칭
    - Username : SSH 원격 서버 호스트에서 사용하는 계정명
    - Private Key
        - Enter directly 체크 후 Add 클릭
        - AWS *.pem 키의 내용을 메모장으로 읽어 복사 후 Key에 붙여넣기

Jenkins Pipeline 추가

아이템 추가

- 새로운 Item 추가
- 아이템 이름 지정

- Pipeline → OK

GitLab 연동 설정

- Configure - General - GitLab Connection
- Build when a change is pushed to GitLab 체크

Jenkins Credential

환경 변수 파일 등록

- Jenkins 관리 - Manage Credentials 클릭
- Stores scoped to Jenkins - Domains - (global) - Add credentials 클릭
  - Kind : Secret file
  - File 클릭 후 환경 설정 파일을 업로드
    - .env.production
    - application-deploy.yml
  - ID : 파이프라인에서 사용할 별칭
  - Description : 파일 설명

프론트엔드 추가 설정

Node.js 추가

- Jenkins 관리 - System Configuration - Tools 클릭
- Tools - NodeJS installations - Add NodeJS 클릭
- Jenkins 컨테이너의 Node.js 빌드환경 설정
  - Name : Node.js 환경에 대한 이름
  - Version : 빌드하려는 Node.js 버전 선택

환경 변수 설정하기

- Jenkins 관리 - System Configuration - System 클릭
- Global Properties
    - Environment variables 체크
    - **CI, false** 환경변수 추가
    - 빌드 시 경고를 에러로 인식하는 문제를 방지하기 위함

백엔드 파이프라인 스크립트

변수명은 등록한 내용에 맞춰서 작성

```
pipeline {
    agent any

    environment {

        imageName = "{이미지명}"
        registryCredential = '{도커 허브 Credential}'

        releaseServerAccount = 'ubuntu'
        releaseServerUri = '{서버 주소}'
        releasePort = '{포트}'

    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: '{Pull 할 브랜치명}', credentialsId
            }
        }
        stage('Add yml'){
            steps{
                dir('./backend/{패키지 Root 경로}'){
                    withCredentials([file(credentialsId: '{서ㅂ
                        sh 'cp ${application} src/main/resour
                    }
                }
            }
        }
        stage('BE-Build') {
            steps {
                dir("./backend/{패키지 Root 경로}") {
                    sh "chmod +x ./gradlew"
                    sh "./gradlew clean bootJar"
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                dir('./backend/{패키지 Root 경로}') {
                    script {
                        docker.withRegistry('', registryCrede
                            sh "docker buildx create --use --
                            sh "docker buildx build --platfor
                            sh "docker buildx build --platfor
                        }
                    }
                }
            }
        }

        stage('Before Service Stop') {
            steps {
                sshagent(credentials: ['{SSH Credential명}'])
                    sh '''
                    if test "`ssh -o StrictHostKeyChecking=no
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    fi
                    '''
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: ['{SSH Credential명}'])
                    sh "ssh -o StrictHostKeyChecking=no $rele
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: ['{SSH Credential명}l'])
                    script {
                        docker.withRegistry('', registryCrede
                            sh '''
                                ssh -o StrictHostKeyChecking=
                                '''
                        }
                    }
                }
            }
        }
    }
}
```

프론트엔드 파이프라인 스크립트

```
pipeline {
    agent any
    tools {nodejs "nodejs"}
    environment {

        imageName = "{이미지명}"
        registryCredential = '{도커 허브 Credential}'

        releaseServerAccount = 'ubuntu'
        releaseServerUri = '{서버 주소}'
        releasePort = '{포트}'


    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: '{Pull 할 브랜치명}', credentialsId
            }
        }
        stage('Add Env') {
        steps {
            dir('./frontend') {
                withCredentials([file(credentialsId: '.env.pr
                   sh 'cp ${env} .env.production'
                    }
                }
            }
        }
        stage('Node Build') {
            steps {
                dir("./frontend") {
                    sh "npm install --force"
                    sh "npm run build"
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                dir('./frontend'){
                    script {
                        docker.withRegistry('', registryCrede
                            sh "docker buildx create --use --
                            sh "docker buildx build --platfor
                            sh "docker buildx build --platfor
                        }
                    }
                }
            }
        }


        stage('Before Service Stop') {
            steps {
                sshagent(credentials: ['ssh-credential']) {
                    sh '''
                    if test "`ssh -o StrictHostKeyChecking=no
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    fi
                    '''
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: ['ssh-credential']) {
                    sh "ssh -o StrictHostKeyChecking=no $rele
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: ['ssh-credential']) {
                    script {
                        docker.withRegistry('', registryCrede
                            sh '''
                                ssh -o StrictHostKeyChecking=
                                '''
                        }
                    }
                }
            }
        }
    }
}
```

외부 API 설정

구글 소셜 로그인

Google Cloud 설정

앱 정보 등록하기

- 프로젝트 생성 후 API & Services로 이동
- OAuth 동의 화면 → 앱 만들기
- 앱 정보 등록
    - 앱 이름
    - 사용자 정의 이메일
    - 앱 도메인

    - 승인된 도메인
    - 개발자 연락처 정보
- 활용한 API 범위
    - 이메일 주소 확인
        - auth/userinfo/email
    - 개인정보 확인
        - auth/userinfo/profile
    - 개인정보 연결
        - openid
- 앱 생성 완료

앱 생성 이후 사용자 인증 정보

- 클라이언트 ID, 클라이언트 보안 비밀번호 확인 가능
- 필요한 정보들을 기재
    - 승인된 자바스크립트 원본
        - 도메인
    - 승인된 리디렉션 URI
        - 구글에서 인가 코드를 받을 URI

업비트 API

UpbitAPI-Server

다음 요청을 수행하며, 따로 Credential은 불필요

코인 리스트

https://api.upbit.com/v1/market/all

websocket

wss://api.upbit.com/websocket/v1

현재 가격

https://api.upbit.com/v1/ticker?markets=KRW-BTC,KRW-ETH

캔들 1분 봉 200개

https://api.upbit.com/v1/candles/minutes/1?market=KRW-BTC&count=200