[사무국 제출] 프로젝트 문서

1. Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

각종 버전 및 명령어 정보

(1) 버전 관리 툴 정보

- git `2.41.0`
- 외부 서비스
    - Gitlab (SSAFY제공)
    - Webhook

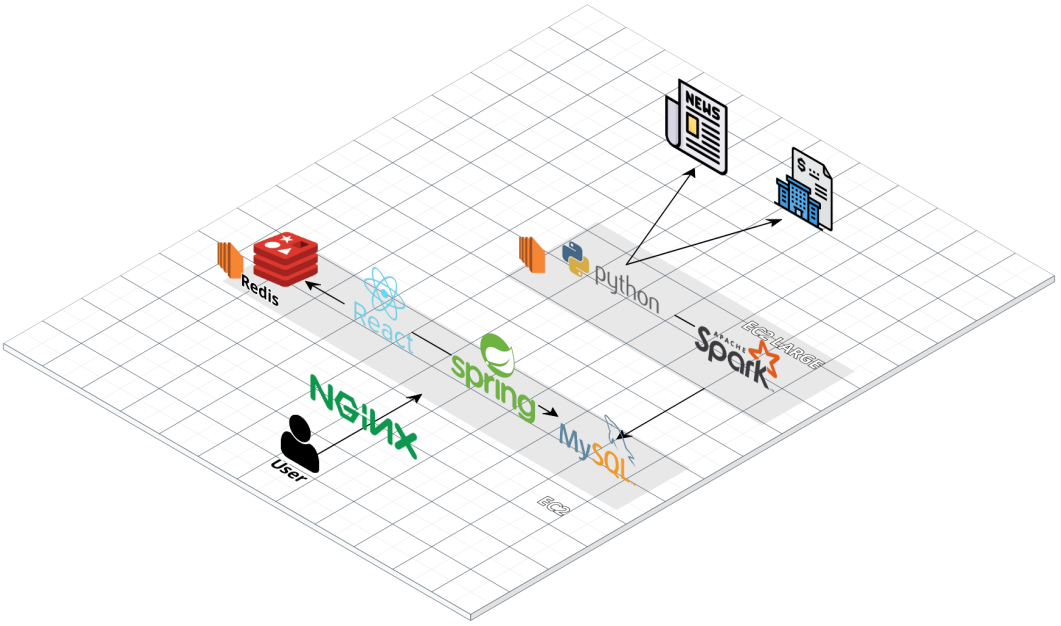
(2) 빌드 관련 정보

- Open JDK `17`

    - Gradle `8.2.1`
- Node.js `18.17.1`
    - npm `9.6.7`
- Jenkins `latest` (Docker)

(3) 배포 관련 정보

- [공통] Docker 24.0.6
- 외부 서비스
    - AWS EC2 (SSAFY 제공)



1) [프론트] : Dockerfile / front.conf /Jenkinsfile

```
# 베이스 이미지는 Node
FROM node:alpine
# 포트는 3126번
ENV PORT 3126

# 워킹 디렉토리 설정(이 경로에서 다음 명령어를 진행)
WORKDIR /usr/src/app
# 워킹 디렉토리에 패키지제이선 카피
COPY package*.json ./
# 인스톨 진행
RUN npm install
```

```
# 루트 디렉토리를 컨테이너로 카피
COPY ./ ./
# ENV환경을 프로덕션을 설정
ENV NODE_ENV production
# 빌드
RUN npm run build
# 스타트
CMD ["npm", "run", "start"
```

```
server {
  listen 3126;
  location / {
    root /app/.next;
    index index.html;
    try_files $uri $uri/ =404;
  }
}
```

```
pipeline {
  agent any

  tools {
    nodejs "nodejs"
  }

  stages {
    stage('React Build') {
      steps {
        dir('Frontend/ssahome') {
          sh 'npm install'
          sh 'npm run build'
        }
      }
    }

    stage('Docker Build') {
      steps {
        dir('Frontend/ssahome') {
          sh 'docker build -t eagerbeaverfe:latest .'
        }
      }
    }

    stage('Deploy') {
      steps{
```

```
        sh 'docker rm -f front'
        sh 'docker run -d --name front -p 3126:3126 eagerbeaverfe:latest'
        }
    }

    stage('Finish') {
      steps{
        sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
      }
    }

  }
}
```

2) [백엔드] : Dockerfile / Jenkinsfile

```
FROM openjdk:17-jdk-slim
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8200
ENTRYPOINT ["java", "-jar", "app.jar"]
```

```
pipeline {
    agent any
    tools {
        gradle "Gradle_8.2.1"
    }

    stages {
        // Gradle 빌드 스테이지: Spring Boot 프로젝트를 빌드합니다.
        stage('Gradle Build') {
            steps {
                // 'Backend' 디렉터리 내에서 작업을 실행합니다.
                dir('Backend/eagerbeaver') {
                    // sh 'export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64'

                    // gradlew 실행 권한 부여
                    sh 'chmod +x gradlew'
                    // gradlew를 사용해 프로젝트를 빌드하며 테스트는 제외합니다.
                    sh './gradlew clean build -x test'
                }
            }
        }
```

```
        // Docker 이미지 빌드 스테이지: Dockerfile을 기반으로 이미지를 빌드합니다.
        stage('Docker Build') {
            steps {
                dir('Backend/eagerbeaver') {
                    // 이미지를 빌드합니다.
                    sh 'docker build -t eagerbeaver:latest .'
// 일반 빌드가 deprecated 되어서, BuildKit을 사용하는 코드. 여기서는 안되서 이전 버젼으로 진행
//                     sh 'DOCKER_BUILDKIT=1 docker build -t herosof-trashbin:latest .'

                }
            }
        }

        // 배포 스테이지: 이전에 실행 중인 'back' 컨테이너를 제거하고 새로운 이미지로 컨테이너를 실행합니다.
        stage('Deploy') {
            steps {
                // 실행 중인 'back' 컨테이너 제거
                sh 'docker rm -f back'
                // 새로운 이미지로 'back' 컨테이너를 백그라운드에서 실행
                sh 'docker run -d --name back -p 8200:8200 -u root eagerbeaver:latest'
            }
        }

    }
}
```

3) docker-compose.yml

```
version: "3.8"
services:
  mysql:   # MySql Container
    image: mysql:8.0
    container_name: eagerbeaver-mysql
    restart: always
    ports:
      - 3426:3306 # HOST:CONTAINER
    environment:
      MYSQL_ROOT_PASSWORD: kbsw2jEB!
      MYSQL_DATABASE: eagerbeaver
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - /var/lib/mysql:/var/lib/mysql

  jenkins:  # Jenkins Container
    image: jenkins/jenkins:lts
    container_name: eagerbeaver-jenkins
    volumes:
      - /usr/bin/docker:/usr/bin/docker
      - /var/run/docker.sock:/var/run/docker.sock
```

```
      - /var/jenkins_home:/var/jenkins_home
    ports:
      - 8859:8080
    privileged: true
    user: root
    restart: unless-stopped

  redis:  # Redis Container
    image: redis:latest
    container_name: eagerbeaver-redis
    ports:
      - 4805:6379
    environment:
      - REDIS_REPLICATION_MODE=master
    volumes:
      - redis/data:/data
    restart: unless-stopped
```

4) [Nginx] 설정 : 포트포워딩, SSL설정

```
# HTTP 서버 설정
server {
  # 80 포트에서 들어오는 HTTP 요청을 수신
  listen 80;
  # 요청을 처리할 도메인 이름
  server_name j9a507.p.ssafy.io;
  # 서버 버전 정보 숨기기 (보안상의 이유)
  server_tokens off;

  # Let's Encrypt 인증서 갱신을 위한 경로 설정
  location /.well-known/acme-challenge/ {
    root /var/www/certbot;
  }

  # 모든 HTTP 요청을 HTTPS로 리다이렉트
  location / {
    return 301 https://$server_name$request_uri;
  }
}

# HTTPS 서버 설정
server {
  # 443 포트에서 들어오는 HTTPS 요청을 수신
  listen 443 ssl;
  server_name j9a507.p.ssafy.io;
  server_tokens off;
  # 액세스 로그 기록 비활성화
  access_log off;
  # Let's Encrypt로부터 받은 SSL 인증서와 키 파일 경로
```

```
  ssl_certificate /etc/letsencrypt/live/j9a507.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/j9a507.p.ssafy.io/privkey.pem;
  include /etc/letsencrypt/options-ssl-nginx.conf; # SSL 설정 포함
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # DH 파라미터 경로

  # 기본 요청을 특정 도메인의 3126 포트로 프록시
  location / {
    proxy_pass http://j9a507.p.ssafy.io:3126/;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Host $server_name;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_redirect off;
  }

  # /api/로 시작하는 요청을 특정 도메인의 8200 포트로 프록시
  location /api/ {
    proxy_pass http://j9a507.p.ssafy.io:8200/api;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Host $server_name;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_redirect off;
  }
}
```

5) Hadoop Eco System

```
# Hadoop Eco System #
    version: "3"
    services:
    spark-master:
        image: bitnami/spark:3.4.1-debian-11-r71
        container_name: spark-master
        environment:
        - SPARK_MODE=master
        - SPARK_RPC_AUTHENTICATION_ENABLED=no
        - SPARK_RPC_ENCRYPTION_ENABLED=no
        ports:
        - "7077:7077"
        - "8080:8080"
        volumes:
        - ./spark-master-logs:/opt/bitnami/spark/logs
        - shared_data:/home/jovyan/shared
        networks:
```

```
        - newSpark-net

    spark-worker1:
        image: bitnami/spark:3.4.1-debian-11-r71
        container_name: spark-worker1
        environment:
        - SPARK_MODE=worker
        - SPARK_MASTER_URL=spark-master:7077
        volumes:
        - ./spark-worker1-logs:/opt/bitnami/spark/logs
        - shared_data:/home/jovyan/shared
        ports:
        - "8081:8081"
        depends_on:
        - spark-master
        networks:
        - newSpark-net

    spark-worker2:
        image: bitnami/spark:3.4.1-debian-11-r71
        container_name: spark-worker2
        environment:
        - SPARK_MODE=worker
        - SPARK_MASTER_URL=spark-master:7077
        volumes:
        - ./spark-worker2-logs:/opt/bitnami/spark/logs
        - shared_data:/home/jovyan/shared
        ports:
        - "8082:8082"
        depends_on:
        - spark-master
        networks:
        - newSpark-net

    spark-worker3:
        image: bitnami/spark:3.4.1-debian-11-r71
        container_name: spark-worker3
        environment:
        - SPARK_MODE=worker
        - SPARK_MASTER_URL=spark-master:7077
        volumes:
        - ./spark-worker3-logs:/opt/bitnami/spark/logs
        - shared_data:/home/jovyan/shared
        ports:
        - "8083:8083"
        depends_on:
        - spark-master
        networks:
        - newSpark-net


    spark-worker4:
        image: bitnami/spark:3.4.1-debian-11-r71
        container_name: spark-worker4
        environment:
        - SPARK_MODE=worker
```

```
        - SPARK_MASTER_URL=spark-master:7077
        volumes:
        - ./spark-worker4-logs:/opt/bitnami/spark/logsw
        - shared_data:/home/jovyan/shared
        ports:
        - "8084:8084"
        depends_on:
        - spark-master
        networks:
        - newSpark-net


    jupyter:
        image: jupyter/all-spark-notebook:spark-3.4.1
        container_name: jupyter-notebook
        ports:
        - "8888:8888"
        volumes:
        - ./jupyter-notebooks:/home/work
        - shared_data:/home/jovyan/shared
        environment:
        - NB_UID=1000
        - NB_GID=1000
        - GRANT_SUDO=yes
        networks:
        - newSpark-net

    volumes:
    shared_data:

    networks:
    newSpark-net:
```

2. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서

- 소셜 로그인
    - Kakao Social Login
- 네이버 검색 API

3. DB 덤프 파일 최신본

별첨

4. 시연 시나리오

시연 순서에 따른 Site 화면별, 실행별 상세 설명

시연 시나리오

1. 홈페이지
   *시작하기 버튼 => 메인 페이지로 이동
   *로그인 버튼 => 카카오 로그인
   *게임 설명 => 화살표 버튼을 클릭하여 좌우 이동
2. 메인 페이지
   *랭킹보기 => 턴별 최고 수익률 달성자 확인
   *플레이 => 턴, 제한시간 설정 => 시작 버튼 클릭시 게임 시작
   *부동산 단어 => 매일 하나씩 새로운 부동산 단어 출력
3. 게임 페이지
   - 뉴스 기사 확인 => 구매하고자 하는 지역 선택 => 수량 설정 후 구매
   - 우측 하단 박스에서 구매 내역 확인 => 원하는 지역 클릭 후 판매
   - 좌측 하단 + 버튼 클릭하여 아이템 사용 (다음 시세 구매, 뉴스 추가 구매, 용어 검색)
   - 세금 정책보기 및 내 세금 내역 클릭하여 세금 관련 내용 확인
   - 넘어 가기 버튼을 클릭하여 턴을 넘기고 설정한 턴이 다 지나가면 게임 종료
4. 수익률 확인
   - 그래프
   - 게임매매로그
   - 랭킹
   - 종료버튼 ⇒ 메인페이지