Cocos Forest 프로젝트 배포 사양서

1. 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전

본 프로젝트는 Docker 컨테이너 기반으로 각 서비스가 격리되어 운영됩니다.

- JVM (Java Virtual Machine)
  - **종류:** OpenJDK
  - **버전:** 17-jdk-slim (Jenkins 이미지 및 Spring Boot 애플리케이션 실행 환경)
- 웹서버 (Web Server)
  - **제품:** Nginx
  - **버전:** latest (Docker Hub 의 최신 안정 버전)
  - **설정:**
    - ./nginx/conf.d 디렉터리의 설정 파일을 마운트하여 리버스 프록시로 동작.
    - HTTP (80), HTTPS (443) 포트 사용.
    - Let's Encrypt 를 통한 SSL 인증서 적용 (/etc/letsencrypt, /var/www/certbot 마운트).
    - 애플리케이션 서버(cocos-forest-app)로 요청을 전달.
- WAS (Web Application Server)
  - **제품:** 내장형 Tomcat (Spring Boot Starter Web 에 포함)
  - **애플리케이션:** Cocos Forest (Spring Boot 기반)
  - **실행 환경:** Docker 컨테이너 내부에서 OpenJDK 17 로 실행.
- 데이터베이스 (Database)
  - **제품:** MySQL
  - **운영 DB 버전:** 8.0
  - **개발 DB 버전:** 8.0
  - **설정:**
    - **운영 DB:** cocos-forest-db 컨테이너, Port 3306, 데이터베이스명 cocos_forest
    - **개발 DB:** cocos-forest-db-dev 컨테이너, Port 3307, 데이터베이스명 cocos_forest_dev
- CI/CD 서버
  - **제품:** Jenkins
  - **버전:** lts-jdk17 (장기 지원 버전)
- 인메모리 데이터 저장소 (In-memory Data Store)
  - **제품:** Redis
  - **버전:** 7.2-alpine
- IDE 버전 (권장)
  - **Backend:** IntelliJ IDEA Ultimate (최신 버전 권장)
  - **Frontend (Mobile):** Visual Studio Code (최신 버전 권장)

2. 빌드 시 사용되는 환경 변수 등의 내용 상세

빌드 및 배포 과정에서는 Jenkins 환경 변수와 Credentials를 복합적으로 사용합니다.

- **Jenkinsfile 내 정의된 환경 변수**
  - CREDENTIALS_ID: 배포 환경(dev/master)에 따라 사용할 Jenkins Secret file Credential의 ID. (DEV_PROPERTIES 또는 PROD_PROPERTIES)
  - IMAGE_NAME: 배포 환경에 따라 빌드될 Docker 이미지의 이름. (cocos-forest-app-dev 또는 cocos-forest-app)
  - CONTAINER_NAME: 배포 환경에 따라 실행될 Docker 컨테이너의 이름. (cocos-forest-app-dev 또는 cocos-forest-app)
  - gitlabTargetBranch: GitLab Merge Request 시 Webhook을 통해 전달되는 타겟 브랜치 명. (e.g., dev, master)
- **Jenkins Credentials를 통해 주입되는 변수**
  - PROPERTIES_FILE_PATH: CREDENTIALS_ID에 해당하는 Secret file의 임시 경로. Spring Boot의 application.properties 파일로 사용됩니다.
  - GCP_CREDENTIALS_FILE: GCP 서비스 계정 키(JSON) 파일의 임시 경로.
  - FIREBASE_CREDENTIALS_FILE: Firebase 서비스 계정 키(JSON) 파일의 임시 경로.
  - MM_URL: Mattermost Webhook URL. 빌드 결과 알림에 사용됩니다.
- **Docker Compose에서 사용하는 환경 변수 (.env 파일 필요)**
  - MYSQL_ROOT_PASSWORD: 운영 DB의 root 계정 비밀번호.
  - MYSQL_ROOT_PASSWORD_DEV: 개발 DB의 root 계정 비밀번호.
  - DOCKER_GID: 호스트 서버의 Docker 그룹 GID. Jenkins 컨테이너가 호스트의 Docker를 제어할 수 있는 권한을 부여하기 위해 사용됩니다.

3. 배포 시 특이사항

- **사전 Docker 리소스 생성 필요**: docker-compose.yml에 external: true로 명시된 네트워크(cocos-network)와 볼륨(mysql_data, jenkins_home)은 docker-compose up을 실행하기 전에 반드시 수동으로 생성해야 합니다.
- **Docker-out-of-Docker (DooD) 방식 사용**: Jenkins 컨테이너가 CI/CD 파이프라인 과정에서 Docker 명령어를 실행(애플리케이션 이미지 빌드 및 배포)하기 위해, 호스트의 Docker 소켓 (/var/run/docker.sock)과 실행 파일 (/usr/bin/docker)을 마운트합니다. 이를 위해 Jenkins 실행 유저에게 DOCKER_GID를 부여하는 과정이 필수적입니다.
- **비밀 정보(Credentials) 관리**: Spring Boot의 설정 파일, GCP 및 Firebase 인증 키 파일은 Jenkins Credentials에 'Secret file' 형태로 등록하여 관리합니다. Deploy 단계에서 withCredentials 블록을 통해 이 파일들을 임시로 불러온 뒤, docker cp 명령어를 사용하여 실행될 애플리케이션 컨테이너 내부의 /run/secrets/ 경로로 복사하여 주입합니다. 컨테이너

내부에서는 파일 경로를 환경 변수(GCP_CREDENTIALS_LOCATION, FCM_SERVICE_ACCOUNT_FILE_LOCATION)로 참조합니다.

- **순차적 컨테이너 제어**: 무중단 배포가 아니므로, Deploy 스크립트는 docker stop 및 docker rm 으로 기존 컨테이너를 먼저 중지 및 삭제한 후, 새로 빌드된 이미지로 컨테이너를 재생성하고 시작하는 방식을 사용합니다.