

# S111\_MST365\_포팅\_메뉴얼



# MST365

## MST365 포팅 메뉴얼

### (로컬) Demo Website Frontend 테스트 명령어

1. demo/demo-front/demo-next (demo-next-css, demo-react, demo-vue 공통)으로 이동한다.
2. 명령어를 터미널에 입력해 필요한 모듈을 설치한다.
  - a. node.js 18.x 버전 이상이 필요합니다.

### (로컬) Demo Website Backend 테스트 명령어

1. demo/demo-back/ 으로 이동한다.
2. demo/demo-back/src/main/application.properties 를 생성한다.
  - a. 로컬 용 application.properties 예시

```
# 기본 설정
server.port=8080
spring.application.name=demo-back

# 로그 설정
logging.level.root=INFO
logging.file.name=logs/demo-app.log
```

```
# MySQL Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/{db 이름}?serverTimez
spring.datasource.username={MYSQL 유저이름}
spring.datasource.password={MYSQL 유저 비밀번호}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# JWT Configuration
jwt.secret={JWT 시크릿 키}
jwt.expiration.ms={JWT 유효기간}
```

## (로컬) DB 세팅

1. MySQL 8.0.x 버전을 설치한다.
2. 아래의 SQL 구문을 따라 데이터베이스 및 테이블을 생성한다.

```
-- -----
-- Database Schema for Board Application
-- -----

-- 데이터베이스 생성
-- 데이터베이스 이름은 'ssafy'입니다.
-- 이미 'ssafy' 데이터베이스가 있다면 이 라인은 주석 처리하거나 건너뛰셔도 됩니다.
CREATE DATABASE IF NOT EXISTS `ssafy`
  DEFAULT CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;

-- 사용할 데이터베이스 선택
-- 이제 'ssafy' 데이터베이스를 사용합니다.
USE `ssafy`;
```

```

-- Table `ssafy`.`users`
-----

-- 사용자 정보를 저장하는 테이블
CREATE TABLE IF NOT EXISTS `ssafy`.`users` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `user_name` VARCHAR(50) NOT NULL COMMENT '사용자 이름 (고유)',
  `password` VARCHAR(100) NOT NULL COMMENT '비밀번호 (해시 값 저장)',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
  `updated_at` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `user_name_UNIQUE` (`user_name` ASC) COMMENT '사용자 이름 유니크 인덱스',
) ENGINE = InnoDB
  DEFAULT CHARACTER SET = utf8mb4
  COLLATE = utf8mb4_unicode_ci
  COMMENT = '사용자 정보 테이블';

-----

-- Table `ssafy`.`boards`
-----

-- 게시물 정보를 저장하는 테이블
CREATE TABLE IF NOT EXISTS `ssafy`.`boards` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `user_id` INT NOT NULL COMMENT '게시물 작성자 ID (users 테이블 참조)',
  `title` VARCHAR(200) NOT NULL COMMENT '게시물 제목',
  `content` TEXT NOT NULL COMMENT '게시물 내용',
  `view` INT NOT NULL DEFAULT 0 COMMENT '조회수',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
  `updated_at` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `is_deleted` BOOLEAN NOT NULL DEFAULT FALSE COMMENT '소프트 삭제 여부',
  `deleted_at` TIMESTAMP NULL COMMENT '소프트 삭제 시각',
  PRIMARY KEY (`id`),
  INDEX `fk_boards_users_idx` (`user_id` ASC) COMMENT '사용자 ID 인덱스',
  CONSTRAINT `fk_boards_users` FOREIGN KEY (`user_id`) REFERENCES `ssafy`.`users` (`id`) -- 외래 키 참조 시 데이터베이스 이름 명시
  ON DELETE CASCADE -- 사용자가 삭제되면 해당 사용자의 게시물도 삭제 (소프트 삭제)
  ON UPDATE CASCADE -- 사용자 ID가 업데이트되면 게시물의 user_id도 업데이트
) ENGINE = InnoDB

```

```

DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_unicode_ci
COMMENT = '게시물 정보 테이블';

-----
-- Table `ssafy`.`comments`
-----

-- 댓글 정보를 저장하는 테이블
CREATE TABLE IF NOT EXISTS `ssafy`.`comments` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `user_id` INT NOT NULL COMMENT '댓글 작성자 ID (users 테이블 참조)',
  `board_id` INT NOT NULL COMMENT '댓글이 속한 게시물 ID (boards 테이블 참조)',
  `content` TEXT NOT NULL COMMENT '댓글 내용',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
  `updated_at` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정 시간',
  `is_deleted` BOOLEAN NOT NULL DEFAULT FALSE COMMENT '소프트 삭제 여부',
  `deleted_at` TIMESTAMP NULL COMMENT '소프트 삭제 시각',
  PRIMARY KEY (`id`),
  INDEX `fk_comments_users_idx` (`user_id` ASC) COMMENT '사용자 ID 인덱스',
  INDEX `fk_comments_boards_idx` (`board_id` ASC) COMMENT '게시물 ID 인덱스',
  CONSTRAINT `fk_comments_users` FOREIGN KEY (`user_id`) REFERENCES `ssafy`.`users` (`id`) -- 외래 키 참조 시 데이터베이스 이름 명시
  ON DELETE CASCADE -- 사용자가 삭제되면 해당 사용자의 댓글도 삭제 (소프트 삭제)
  ON UPDATE CASCADE, -- 사용자 ID가 업데이트되면 댓글의 user_id도 업데이트
  CONSTRAINT `fk_comments_boards` FOREIGN KEY (`board_id`) REFERENCES `ssafy`.`boards` (`id`) -- 외래 키 참조 시 데이터베이스 이름 명시
  ON DELETE CASCADE -- 게시물이 삭제되면 해당 게시물의 댓글도 삭제 (소프트 삭제)
  ON UPDATE CASCADE -- 게시물 ID가 업데이트되면 댓글의 board_id도 업데이트
) ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_unicode_ci
COMMENT = '댓글 정보 테이블';

```

## Jenkins 및 Plugin 테스트 명령어

1. Jenkins 를 이용하여 Plugin을 테스트 해보기 위해 Jenkins 환경을 구축해야 한다.

- a. window 환경의 경우 wsl을 설치하여 Ubuntu 환경을 구축해야 한다.
- b. Ubuntu 환경에 Docker를 설치하고 Jenkins:its 버전의 이미지를 통해 컨테이너를 생성한다.
- c. **(중요) 해당 Jenkins 컨테이너 내부에 apt, node js, python, Chromium 을 설치해야 한다.**
- d. **(중요) Jenkins Container를 실행할 때, -u root 옵션을 통해 root 권한으로 컨테이너를 실행시켜야 한다.**

## 2. Plugin 설치를 위한 hpi 파일 생성 방법

- a. plugin 폴더로 이동한다.

```
# powershell 또는 cmd 에서 다음 명령어를 입력한다.
mvn clean package
```

- b. 위에서 실행시킨 Jenkins에 접속한다.
- c. Dashboard > Jenkins 관리 > Plugins > Advanced Settings > Deploy plugin 으로 이동한다.
- d. 파일 선택에서 plugin/target/playwright-e2e-test.hpi 파일을 선택한다.
- e. deploy 버튼을 눌러 설치한다.

## 3. Plugin 사용 방법

- a. Plugin 사용을 위한 LLM api-key, 모델 등에 대한 env 파일을 credential에 등록해야 한다.
  - i. Jenkins Credential 등록 예시

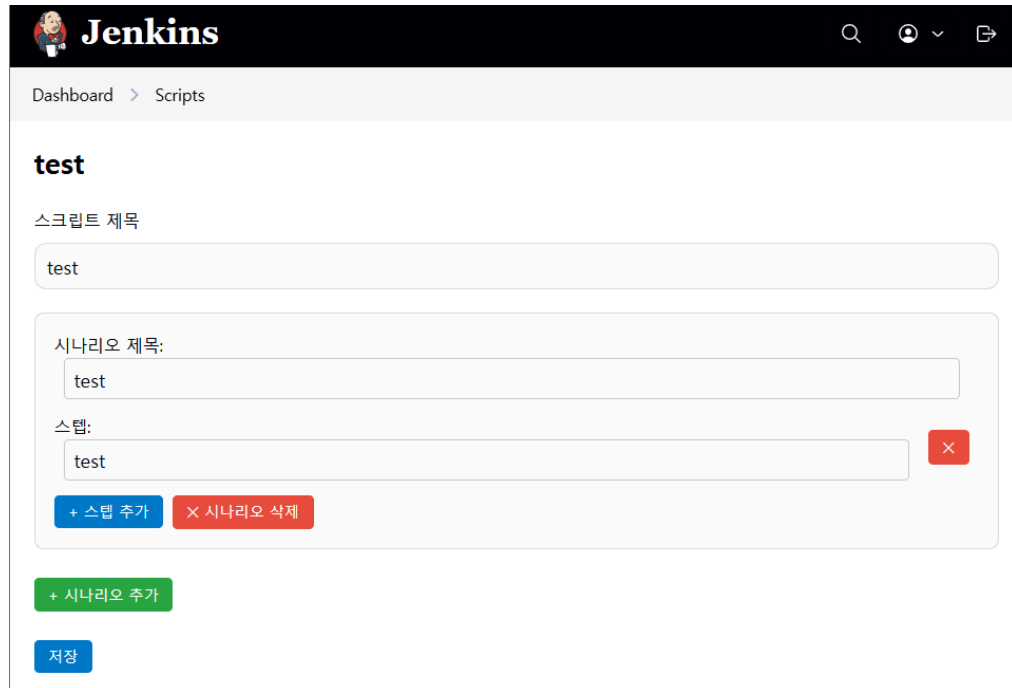
```
# 현재 지원하는 플랫폼은 openAI, Anthropic 두 종류만 제공함
LLM_PROVIDER=${anthropic or openai}
LLM_MODEL=${PROVIDER 별 지원하는 모델 이름}
LLM_API_KEY=${PROVIDER 별 발급된 APIKEY}

# 권장 사항 (langsmith 연결)
LANGSMITH_API_KEY=${LANGSMITH 발급받은 APIKEY}
LANGSMITH_TRACING=true
LANGSMITH_ENDPOINT=https://api.smith.langchain.com
LANGSMITH_PROJECT=${생성한 LANGSMITH 프로젝트 이름}
```

b. 새롭게 생성된 extension인 Scripts로 이동하여 자연어 스크립트를 작성한다.

i. Dashboard > Scripts > 새 스크립트 작성

ii. 새 스크립트 작성시 format에 맞게 작성한다



The screenshot shows the Jenkins 'test' script configuration page. The page has a dark header with the Jenkins logo and navigation icons. Below the header, there's a breadcrumb trail 'Dashboard > Scripts'. The main content area is titled 'test'. It contains a form for creating a new script. The form has a '스크립트 제목' (Script Title) field with the value 'test'. Below it is a '시나리오 제목:' (Scenario Title) field with the value 'test'. Underneath that is a '스텝:' (Step) field with the value 'test'. To the right of the step field is a red 'X' button. Below the form are three buttons: '+ 스텝 추가' (Add Step), '× 시나리오 삭제' (Delete Scenario), and '+ 시나리오 추가' (Add Scenario). At the bottom left is a blue '저장' (Save) button.

c. 새로운 파이프라인 아이템을 생성한다.

i. Dashboard > new Item (type은 pipeline 선택)

d. 새로운 파이프라인 아이템의 '구성'에서 pipeline script를 작성한다.

```
pipeline {
  agent any
  stages {
    stage('CoreLogic') {
      steps {
        runCoreLogic input: '스크립트 제목', envFileCredentialsId: 'credenti
      }
    }
  }
}
```

e. '저장' 하고 '지금 빌드' 버튼을 눌러 pipeline script를 실행 시킨다.

- f. Dashboard > MCP Reports 탭에서 해당 빌드 번호에 해당하는 결과 리포트를 받을 수 있다.

## 1. 사용 도구

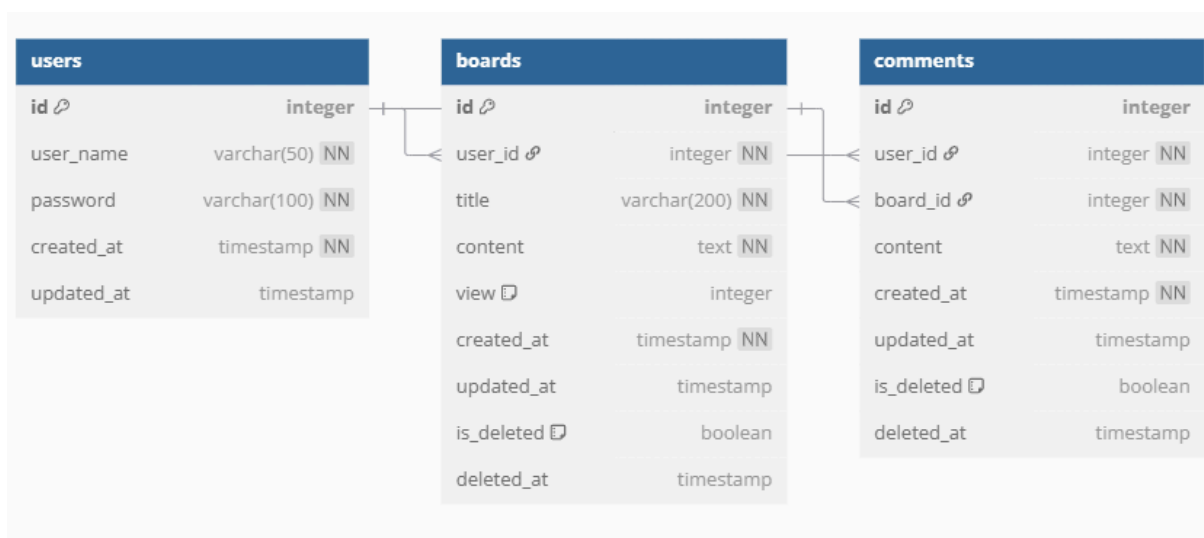
- a. 이슈 관리: Jira, Mattermost
- b. 형상 관리: Gitlab
- c. 문서 관리: Notion
- d. CI/CD: Docker, Jenkins

## 2. 개발 환경

- a. 서버 환경: Ubuntu 22.04
- b. IDE: VsCode, IntelliJ Ultimate
- c. 원격 접속: MobaXterm
- d. 백엔드: Spring Boot, Java
- e. 프론트엔드: Next js, React, Vue js, Javascript
- f. DB : MySQL 8.0.x

## 3. 시스템 아키텍처 및 기술 스택

### 1. ERD



### 3-1. 프론트엔드

- Vue js - 3.2.13
- React - 18.2.0
- Next js - 14.2.23

### **3-2. 백엔드**

- SpringBoot - 3.4.5
- Spring Security - 6.4.3
- Spring Data JPA - 3.4.3
- JWT - 0.11.5

### **3-3. DB**

- MySQL - 8.0.41

### **3-4. 인프라**

- Nginx - 1.27.4
- Jenkins - 2.492.3
- Docker - 28.1.1

### **3-5. 플러그인**

- Jenkins Plugin Parent POM - 5.10
- Jenkins Baseline (Core Version) - 2.492
- Maven HPI Plugin - 3.63
- [io.jenkins.tools.bom](#) - 4740.v75a\_90f6f6fb\_7
- Guava - 33.4.8-jre

### **3-6. MCP host (Python ver.)**

- mcp\_host/python 아래의 requirements.txt, uv.lock, project.toml 참고

### **3-7. MCP host (Typescript ver.)**

- mcp\_host/typescript/package.json 참고

## **4. 배포 서버 설정**



## 4-1. .env 위치 및 내용 예시

1. .env 파일을 EC2 위의 /home/ubuntu/jenkins-data 에 저장한다.

```
NEXT_PUBLIC_BASE_URL={도메인 주소}
VITE_API_BASE_URL={도메인 주소}
VUE_APP_BASE_URL={도메인 주소}
```

## 4-2. 배포 용 application.properties 위치 및 내용 예시

1. /home/ubuntu/jenkins-data 에 application.properties 라는 이름으로 저장한다.

## 4-3. EC2 nginx.conf

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    server_name {도메인 주소};
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl http2 default_server;
    listen [::]:443 ssl http2 default_server;

    server_name k12s111.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/{도메인 주소}/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/{도메인 주소}/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    add_header Strict-Transport-Security "max-age=31536000" always;

    location / {
        proxy_pass http://localhost:51329/;
        proxy_http_version 1.1;
```

```

proxy_set_header Upgrade      $http_upgrade;
proxy_set_header Connection   "upgrade";
proxy_set_header Host         $host;
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 https;
proxy_set_header X-Forwarded-Host $host;
proxy_set_header X-Forwarded-Port $server_port;
}

location /api/ {
    proxy_pass http://localhost:43817/api;
    client_max_body_size 1000m;

    proxy_read_timeout 300s;
    proxy_connect_timeout 75s;

    proxy_http_version 1.1;
    proxy_cache_bypass $http_upgrade;

    add_header Content-Security-Policy "default-src 'self'; img-src 'self' data:;";
    proxy_set_header Authorization $http_authorization;
    proxy_set_header Upgrade      $http_upgrade;
    proxy_set_header Connection   "upgrade";
    proxy_set_header Host         $host;
    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 https;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Port $server_port;
}

location /jenkins/ {
    proxy_pass http://localhost:39754/jenkins/;
    client_max_body_size 1000m;

    proxy_read_timeout 300s;
    proxy_connect_timeout 75s;

```

```
proxy_http_version 1.1;
proxy_cache_bypass $http_upgrade;

add_header Content-Security-Policy "default-src 'self'; img-src 'self' data:;";

proxy_set_header Upgrade      $http_upgrade;
proxy_set_header Connection    "upgrade";
proxy_set_header Host          $host;
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 https;
proxy_set_header X-Forwarded-Host $host;
proxy_set_header X-Forwarded-Port $server_port;
}
}
```