**Marsh Mellow Project**

포팅메뉴얼

SSAFY 광주 캠퍼스 C108팀

내용

[1. 프로젝트 개요 3](#_Toc195089733)

[1.1. 시스템 아키텍처 3](#_Toc195089734)

[2. 개발 및 운영 환경 (Development & Operation Environment) 5](#_Toc195089735)

[2.1. 사용 기술/버전 5](#_Toc195089736)

[2.2. 서버/운영 환경 5](#_Toc195089737)

[3. 빌드 (Build) & 배포 (Deployment) 6](#_Toc195089738)

[3.1. 소스 코드 클론 방법 6](#_Toc195089739)

[3.2 사전 준비 6](#_Toc195089740)

[3.3 Jenkins 설치 및 설정 7](#_Toc195089742)

[3.4 GitLab 연동 9](#_Toc195089743)

[3.5 Jenkins Job 설정 10](#_Toc195089744)

[3.6 빌드 & 배포 과정 12](#_Toc195089745)

[5 DB 구성 및 연동 (Database Setup & Integration) 12](#_Toc195089746)

[5.2. DB 연결 정보 12](#_Toc195089747)

[5.3. Redis 사용 12](#_Toc195089748)

[6. 외부 서비스 및 연동 (External Services Integration) 13](#_Toc195089749)

[7. 시연 시나리오 (Demo / Run Book) 15](#_Toc195089750)

[7.1. 시연 흐름 개요 15](#_Toc195089751)

[8. 기타 참고 자료 (Appendix) 16](#_Toc195089752)

# 1. 프로젝트 개요

**프로젝트 명:** Marsh Mellow

**프로젝트 소개:** 사회 초년생을 위한 자동 재정관리 서비스

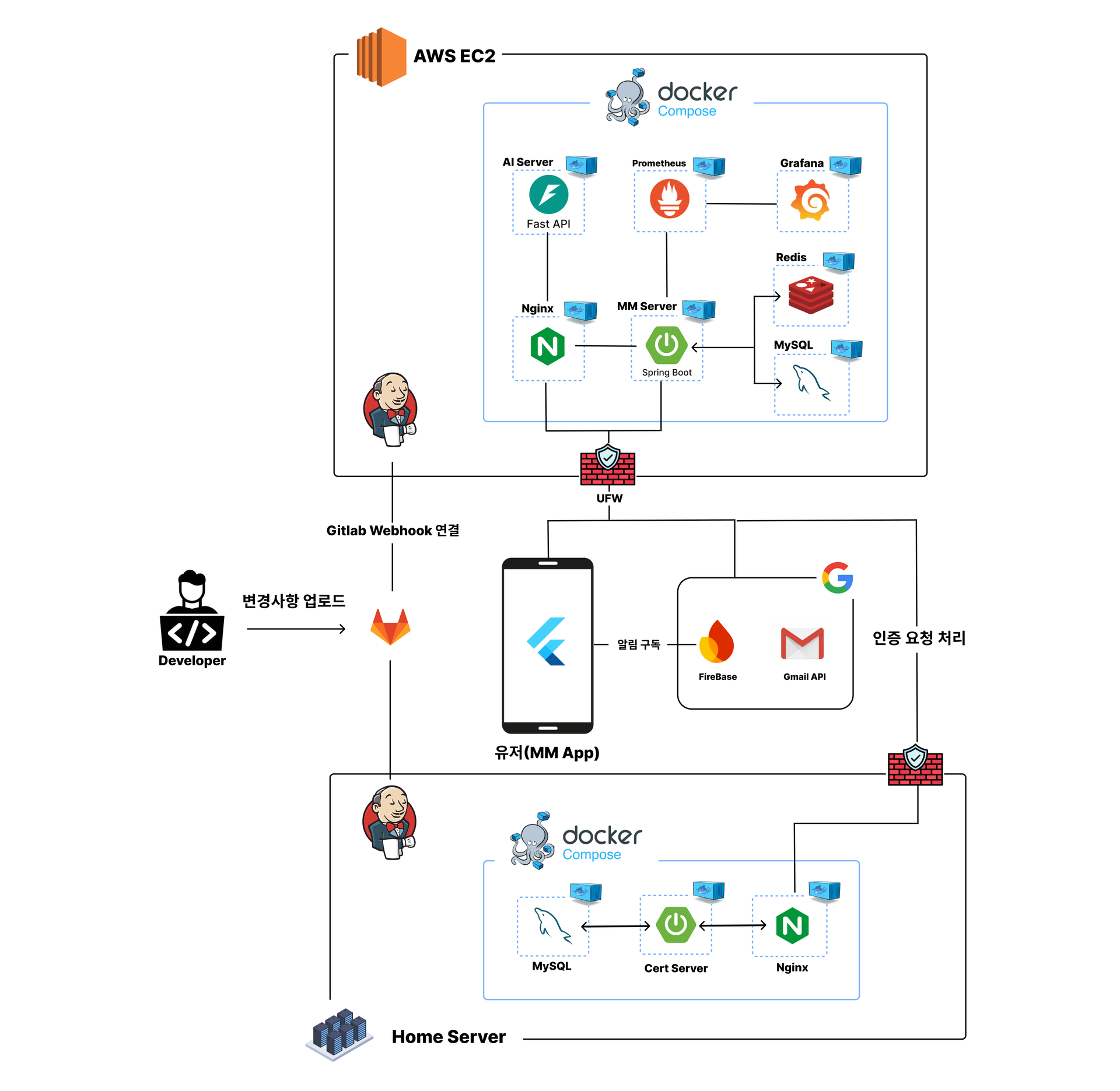
**GitLab 저장소 주소:** <https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21C108>

**브랜치 전략:**

master 브랜치에서 배포용 코드를 관리

develop, feature/\*, fix/\* 등을 통한 개발 및 코드 병합 전략

## 시스템 아키텍처



**EC2서버:**

* UFW를 통해 방화벽 규칙에 등록된 요청만 허용
* Nginx의 Reverse Proxy를 통한 포트 포워딩
* Spring Boot(JPA, Securiy 기반 앱 백엔드 서버), Fast API(AI 서버) 컨테이너
* MySQL 컨테이너(RDBMS)
* Redis 컨테이너(알림, JWT 토큰 정보 관리)

**Home 서버:**

* Nginx의 Reverse Proxy를 통한 포트 포워딩
* Spring Boot(JPA, Securiy 기반 인증 서버)
* MySQL 컨테이너(RDBMS)

**CI/CD:**

GitLab 푸시 → Jenkins에서 Gitlab Webhook을 통한 변경감지 → 자동 빌드/테스트 → Docker 이미지 생성 → DockerHub에 업로드 → Production 서버에서 pull & 배포

**컨테이너 구성:**

* **EC2 서버**
  + certbot: SSL 인증서 발급 및 주기적 갱신
  + Nginx: 포트포워딩 및 Reverse Proxy
  + backend: Spring Boot (JPA, Security), FastAPI(AI)
  + MySQL: RDBMS
  + Redis: 캐싱
  + Prometheus: 매트릭 수집
  + Grafana: 매트릭 시각화
* **Home 서버**
  + certbot: SSL 인증서 발급 및 주기적 갱신
  + Nginx: 포트포워딩 및 Reverse Proxy
  + backend: Spring Boot (JPA, Security)
  + MySQL: RDBMS

**전체 요청 흐름:**

1. 사용자가 MM 어플리케이션을 통해 기능 요청

2. Reverse Proxy(Nginx)가 요청을 적절한 컨테이너로 전달

3. Spring Boot가 DB(MySQL), Redis 등과 연동하여 로직 처리

4. 응답 결과를 다시 Nginx를 통해 사용자에게 반환

# 2. 개발 및 운영 환경 (Development & Operation Environment)

## 2.1. 사용 기술/버전

**AI:** Scikit-learn(KMeans, DecisionTreeClassifier), TensorFlow, Pytorch, Surprise, LightFM

**프론트엔드:** Flutter, Android Studio, Dart

**백엔드:** Spring Boot 3.4.1, Java 21, Gradle

**데이터베이스:** MySQL (8.0.23), Redis

**컨테이너:** Docker, Docker Compose

**CI/CD:** Jenkins, GitLab, DockerHub

**Reverse Proxy:** Nginx

## 2.2. 서버/운영 환경

**OS:** Ubuntu 20.04.6 LTS

포트/방화벽 설정:

22 ,80(HTTP), 443(HTTPS)

9000, 9001(백엔드), 8080(Jenkins), 6379(Redis), 3306(MySQL), 9090(Prometheus)

# 3. 빌드 (Build) & 배포 (Deployment)

## 3.1. 소스 코드 클론 방법

GitLab 레포지토리: git clone <https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21C108>

## 3.2 사전 준비

**AWS EC2 인스턴스 준비**

Ubuntu 20.4.1 서버 인스턴스 준비.

Docker 설치

* Docker 관련 패키지 제거
  + sudo apt-get remove docker docker-engine docker.io containerd runc
* 시스템 업데이트 및 필요 패키지 설치
  + sudo apt-get update
  + sudo apt-get install ca-certificates curl gnupg lsb-release
* Docker GPG 키 추가
  + curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.헿
* Docker APT 저장소 추가

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

- Docker 설치

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

## 3.2 사전 준비

패키지 목록 업그레이드 및 업데이트  
sudo apt-get update

sudo apt-get upgrade -y

Java17 설치 및 환경변수 추가  
sudo apt-get install -y openjdk-17-jdk

export JAVA\_HOME=/usr/lib/jvm/java-17-openjdk-amd64

export PATH=$JAVA\_HOME/bin:$PATH

## 3.3 Jenkins 설치 및 설정

Jenkins 기본 설치

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |sudo gpg --dearmor -o /usr/share/keyrings/jenkins.gpg sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list' sudo apt update  
  
등록된 GPG 삭제  
sudo rm /etc/apt/sources.list.d/jenkins.list

Jenkins GPG 키 등록

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

Jenkins 저장소 추가

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

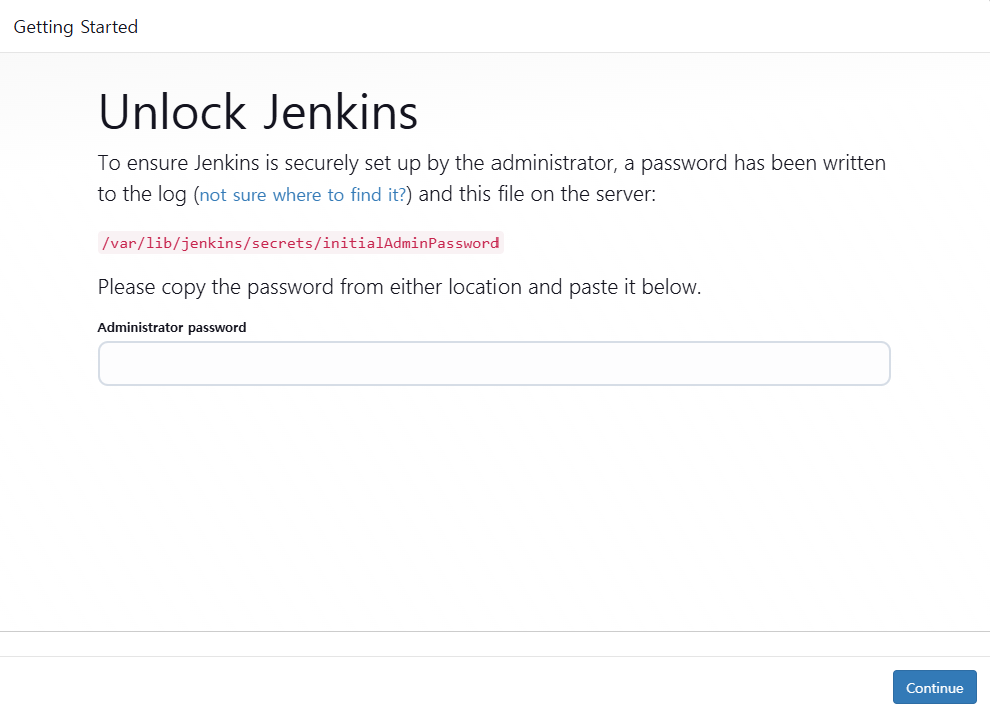
패키지 목록 업데이트 후 Jenkins 설치

sudo apt-get update

sudo apt-get install -y jenkins

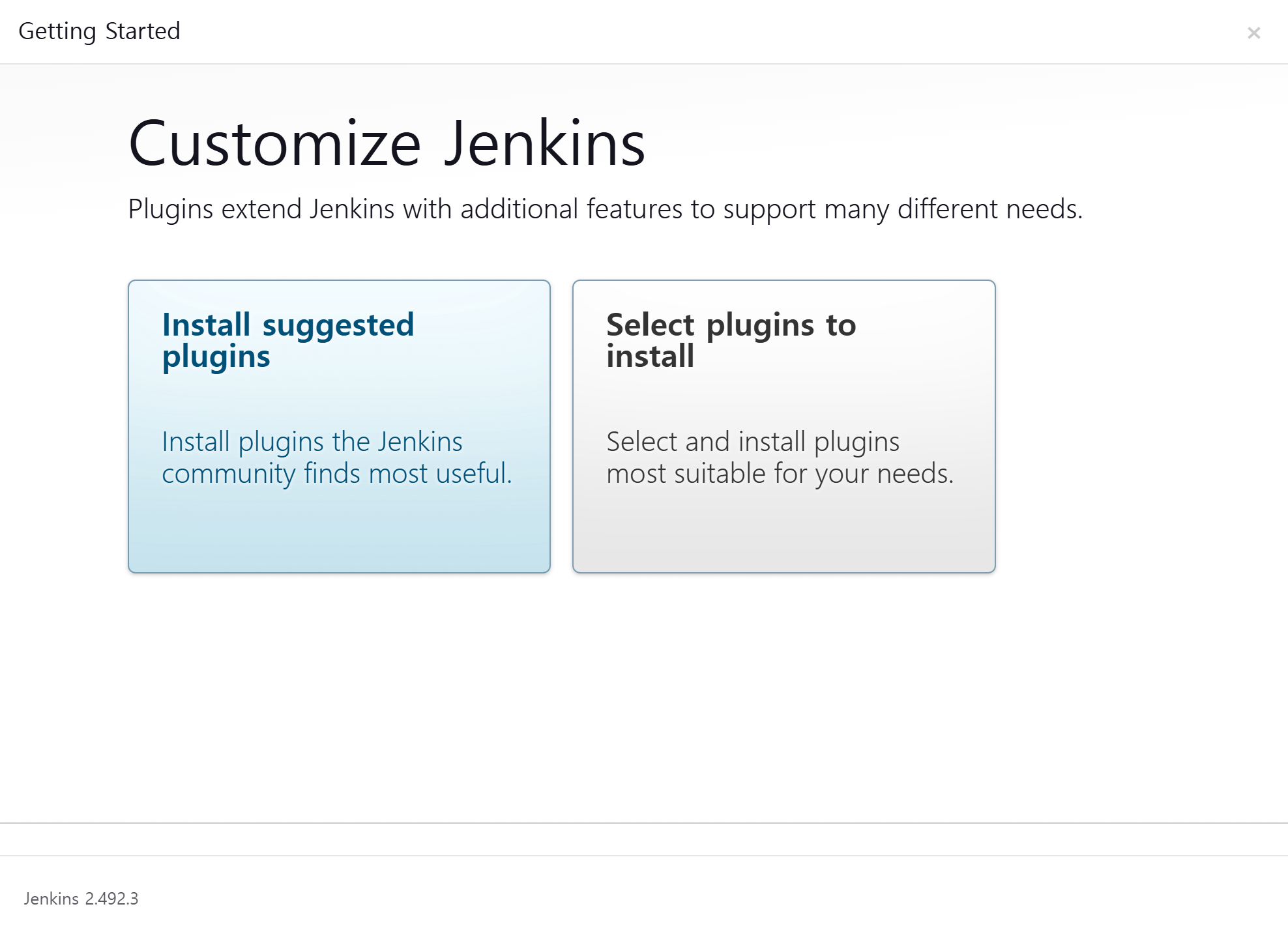
Jenkins 실행 후, http://<EC2-퍼블릭-IP>:8080으로 접속하여 초기 세팅 진행

**Jenkins 계정 및 플러그인 설정**



초기 비밀번호 확인:   
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

관리자 계정 생성 후 필수 플러그인 설치



이후 필요 플러그인 설치

GitLab 플러그인, gradle 플러그인

## 3.4 GitLab 연동

**GitLab 주소 입력**

Jenkins 대시보드 → “Manage Jenkins” → “Configure System” → GitLab 섹션에서 GitLab 서버 주소( [https://gitlab.com](https://lab.ssafy.com/)) 설정

**Credentials 등록**

Jenkins 대시보드 → “Credentials” 메뉴에서 아래 파일/토큰 등을 추가

git\_lab: GitLab API 접근용(GitLab API toen)

mm\_config\_file: 백엔드 서버 yml 파일(Secret File)

dockerhub-credentials-id: docker hub 아이디 비밀번호(Username with password)

firebase\_config\_file: firebase 접근 json file(Secret File)

EC2\_SSH\_PEM: ssh 접근용 pem(SSH Username with private key)

MM\_AI\_ENV: ai 서버 env 파일(Secret File)

## 3.5 Jenkins Job 설정

**Item(프로젝트) 생성**

Jenkins 대시보드 → “새로운 Item” → 파이프라인(Pipeline) 선택

프로젝트 이름: gbh\_mm

**GitLab 리포지토리 연동**

“Source Code Management” 섹션에서 GitLab 주소 입력

Credentials에서 등록해둔 gitlab-token을 선택

“Pipeline” 섹션의 Pipeline script from SCM 선택

SCM: GIT

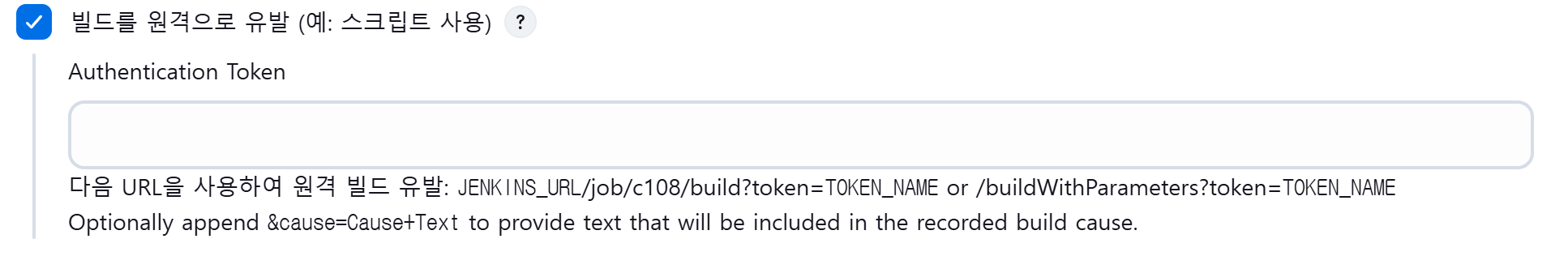
Reposiroty URL: <https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21C108.git>

Credentials: git\_lab



Secret token 생성 후 Authentication Token에 붙여넣기





Gitlab Webhook 등록

URL: <EC2 URL>:8080/project/gbh\_mm/build?token=<Jenkins에서 생성한 Token>\

Push Events, Opens Merge Request Events 체크 후 저장

# 3.6 빌드 & 배포 과정

**대시보드에서 ‘gbh\_mm’빌드 실행**

Jenkins 대시보드에서 해당 Job 클릭 → “Build Now(지금 빌드하기)”

빌드 로그를 통해 정상 동작 여부 확인

**파이프라인 작동 확인**

파이프라인 뷰에서 각 단계(Checkout → Build → Test → Deploy 등) 정상 종료 여부 확인

# 5 DB 구성 및 연동 (Database Setup & Integration)

## 5.2. DB 연결 정보

• application.yml 예시:

datasource:  
 url: jdbc:mysql://<서버주소>:3306/<database 이름>  
 username: root  
 password: “비밀번호”  
 driver-class-name: com.mysql.cj.jdbc.Driver

data:  
 redis:  
 host: <서버주소>  
 port: 6379

## 5.3. Redis 사용

알림 메시지 저장, JWT 관리

redis 컨테이너 기동 후, Spring Boot에서 연결

# 6. 외부 서비스 및 연동 (External Services Integration)

~/gbh\_mm\_deploy 경로에 docker-compose.yml 생성

version: '3'

services:

gbh\_mm:

image: yunjaeeun12/gbh-mm:latest

container\_name: gbh\_mm

ports:

- "9000:9000"

restart: always

gbh\_ai:

image: yunjaeeun12/gbh-fastapi:latest

container\_name: gbh\_fastapi

ports:

- "8000:8000"

restart: always

nginx:

image: nginx:latest

container\_name: nginx

ports:

- "80:80"

- "443:443"

volumes:

- ./nginx/conf.d:/etc/nginx/conf.d

- ./nginx/certs:/etc/letsencrypt

- ./nginx/html:/var/www/html

depends\_on:

- gbh\_mm

restart: always

certbot:

image: certbot/certbot

container\_name: certbot

volumes:

- ./nginx/certs:/etc/letsencrypt

- ./nginx/html:/var/www/html

- ./nginx/conf.d/etc/nginx/conf.d

prometheus:

image: prom/prometheus:latest

container\_name: prometheus

ports:

- "9090:9090"

volumes:

- ./prometheus.yml:/etc/prometheus/prometheus.yml

restart: always

grafana:

image: grafana/grafana:latest

container\_name: grafana

ports:

- "3000:3000"

depends\_on:

- "prometheus"

restart: always

promethus.yml 생성 후 내용 입력

global:

scrape\_interval: 15s # 모든 스크랩 작업의 기본 간격

evaluation\_interval: 15s # 규칙 평가 주기

scrape\_configs:

- job\_name: "prometheus" # Prometheus 자체 모니터링 작업

static\_configs:

- targets: ["localhost:9090"]

- job\_name: "gbh\_mm" # gbh\_mm 컨테이너 모니터링 작업

metrics\_path: "/actuator/prometheus"

static\_configs:

~/gbh\_mm\_deploy/nginx경로에 conf.d, certs, html 디렉토리 생성

~/gbh\_mm\_deploy/nginx/conf.d 경로에 defalut.conf 파일 생성 후 입력

# HTTP: ACME challenge 제공 및 HTTPS 리다이렉션

server {

listen 80;

server\_name j12c108.p.ssafy.io;

# Certbot ACME challenge 파일 제공

location /.well-known/acme-challenge/ {

root /var/www/html;

}

# 그 외의 HTTP 요청은 HTTPS로 리다이렉트

location / {

return 301 https://$host$request\_uri;

}

}

# HTTPS: SSL 적용 및 /api 경로 프록시

server {

listen 443 ssl;

server\_name j12c108.p.ssafy.io;

ssl\_certificate /etc/letsencrypt/live/j12c108.p.ssafy.io/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/j12c108.p.ssafy.io/privkey.pem;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_prefer\_server\_ciphers on;

location ~^/api/mm/ai/ {

rewrite ^/api/mm/ai/(.\*)$ /$1 break;

proxy\_pass http://gbh\_fastapi:8000;

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 $scheme;

}

# /api 요청 프록시 (접두어 제거)

location /api/ {

proxy\_pass http://gbh\_mm:9000/;

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 $scheme;

}

# 기타 요청은 필요에 따라 처리 (예: 404 리턴)

location / {

return 404;

}

}

# 7. 시연 시나리오 (Demo / Run Book)

## 7.1. 시연 흐름 개요

* [ ] USER1
  + [ ] 월급날이 시연날
* [ ] 월급 입금알림
* [ ] 월급 축하 이벤트 화면
* [ ] 소비패턴 유형 분석 (회원가입 이전의 나의 소비내역기반)
* [ ] 예산 분류 (나의 소피패턴에 따라 분류된 예산 현황 화면)
* [ ] 예산 조정 (사용자가 직접 이번달 예산을 계획하는 화면)
* [ ] 이번달의 위시리스트 선택 (이번달 사고싶은 물건 등록)
* [ ] USER2: 사용자 변경 (사용중인 유저로 변경) (월급날 다르게! )
* [ ] 예산현황
* [ ] 포트폴리오 로직
* [ ] 퇴사망상 로직
* [ ] 점심 골라야 하는 상황 (캐릭터)
* [ ] 점메추
* [ ] 더치페이
  + [ ] 전체 자산 간편 조회 화면
  + [ ] 입출금 상세화면의 지출내역 슬라이딩
  + [ ] 바로송금 ⇒ 인증 ⇒ 송금
  + [ ] 전체 자산 조회 화면 리렌더링
* [ ] 자판기 결제
* [ ] 예산 초과 경고 알림
* [ ] 예산현황 (사용해버리고 남은 예산 상태) ( 예산대비 식비 초과 상황)
* [ ] 예산 상세 조회
* [ ] 가계부 ⇒ 자판기 700원 등장 ( 카테고리 분류 확인)
* [ ] 가계부 내역 추가 (지각비 지출 추가)
  + [ ] 내역확인 후 수정
  + [ ] 가계부 내역 삭제
* [ ] 지출분석 (나 이번달에 어디에 얼마나 썼더라?!)

# 8. 기타 참고 자료 (Appendix)

• **Docker Compose 공식 문서:** <https://docs.docker.com/compose/>

• **Jenkins 공식 문서:** <https://www.jenkins.io/doc/>

**• 프로젝트 협업 툴:** JIRA, Notion, Discord, Mattermost

• **관리자 연락처 :** 윤재은 (010-5629-7169)