# 목차



---

# 1. 사용 도구

- **이슈 관리**: Jira
- **형상 관리**: GitLab, Git Bash
- **커뮤니케이션**: Notion, MatterMost
- **디자인**: Figma
- **CI/CD**: Jenkins

---

# 2. 개발 도구

- **IDE 및 에디터**: Visual Studio Code, IntelliJ, Visual Studio 2022, Pycharm

---

# 3. 개발 환경

## Frontend

- **React**: 18.3.1
- **Node.js**: 20

---

## Backend / AI

- **JDK**: 21

- **Spring Boot**: 3.3.5
- **Python**: 3.12.3
- **FastAPI**: 0.115.4

---

## Server

- **AWS EC2**
  - CPU: Intel Xeon E5-2686 v4 (4 Core, 4 Thread)
  - Disk: 311GB
  - RAM: 16GB
- **S3**: Amazon S3

---

## Service

- **Nginx**: 1.26.2
- **Jenkins**: 2.479.1
- **Docker**: 27.3.1
- **Docker-compose**: 2.29.2
- **Redis**: 7.4.1
- **MySQL**: 8.0.20
- **RabbitMQ**: 3.13.7
- **Let's Encrypt / Certbot**
- **Firebase**
- **Bedrock**

---

## Domain

- **Development**: `k11b102.p.ssafy.io`
- **Production**: `bbogle.me`
-

---

# 4. 환경 변수

## Backend - `application.yml`

```
yaml
코드 복사
server:
  port: 8080
  servlet:
    context-path: /api

spring:
  servlet:
    multipart:
      max-file-size: 15MB
      max-request-size: 15MB
  application:
    name: bbogle
  datasource:
    url: jdbc:mysql://mysql:3306/bbogle
    username: root
    password: 20241029moongohome
    driver-class-name: com.mysql.cj.jdbc.Driver
  data:
    redis:
      host: redis
      port: 6379
      password: 20241029moongohome
  rabbitmq:
    host: bbogle-rabbitmq
    port: 5672
    username: bbogle
    password: ayebimil

jwt:
  secret: l/foeVHKMMVz/kOEy+GVd+Yiifgf3C4GX3AHcbjblPg=
  access-expire: 3600000
  refresh-expire: 5259400000
```

```
cloud:
  aws:
    credentials:
      access-key: AKIAS1FIXCZPA2Q62TUAUX
      secret-key: d1nS7m1yFSC4RCLADT5o1k5nI+r9fQRjqy1XPx0Fa
S+6z6
    region:
      static: ap-northeast-2
    s3:
      bucket: itsbbogletime
```

## Python AI - `.env`

```
env
코드 복사
AWS_REGION=ap-northeast-2
AWS_ACCESS_KEY_ID=AKIASFIXCZPAQ62TUAUX
AWS_SECRET_ACCESS_KEY=dnS7myFSC4CLAD5ok5nI+r9fQRjqyXPx0FaS+
6z6

RABBITMQ_USER=bbogle
RABBITMQ_PASS=ayebimil
RABBITMQ_HOST=bbogle-rabbitmq
RABBITMQ_PORT=5672
RABBITMQ_EXCHANGE=my_exchange
```

## Frontend - `.env`

```
env
코드 복사
VITE_API_URL=https://bbogle.me/api

# Firebase 설정
VITE_FIREBASE_API_KEY=AIzaSyAR8FQp4IyRD7xOZJdlw3uKpokhTJrg0
EA
```

```
VITE_FIREBASE_AUTH_DOMAIN=bbogle-c47b4.firebaseapp.com
VITE_FIREBASE_PROJECT_ID=bbogle-c47b4
VITE_FIREBASE_STORAGE_BUCKET=bbogle-c47b4.firebasestorage.a
pp
VITE_FIREBASE_MESSAGING_SENDER_ID=284508974366
VITE_FIREBASE_APP_ID=1:284508974366:web:89fb4006f16b03621f6
d0f
VITE_FIREBASE_VAPID_KEY=BO8vnH_0wDj_3coeJ_Jh-MZFrmlyQZ16qDx
67bwk_DMeT_9zulmQo_DeKs9eU4Hj1-6gZqutoQjPnclAw8Bv7gU
```

---

# 5. CI/CD 구축

## 1. UFW 방화벽 설정

```
bash
코드 복사
sudo apt-get update
sudo ufw allow 22
sudo ufw allow 80
sudo ufw allow 443
sudo ufw enable
sudo ufw status numbered
```

---

## 2. Docker 및 Docker-compose 설치

```
bash
코드 복사
# Docker 설치
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-rele
ase
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | s
udo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/et
c/apt/keyrings/docker.gpg] https://download.docker.com/linu
```

```
x/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sou
rces.list.d/docker.list
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.
io docker-compose-plugin

# 설치 확인
sudo docker run hello-world
```

## 3. Jenkins 설정

### Docker-compose 설정 예시

```
yaml
코드 복사
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: compose-jenkins
    user: root
    volumes:
      - ./jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      JENKINS_OPTS: --prefix=/jenkins
    networks:
      - jenkins-network
    ports:
      - "9090:9090"

networks:
  jenkins-network:
    external: true
```

## 4. Nginx 설정

## Docker-compose 설정 예시

```
yaml
코드 복사
services:
  nginx:
    image: nginx:custom-nginx
    container_name: compose-nginx
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    networks:
      - back-network
      - front-network
    ports:
      - "80:80"
      - "443:443"

networks:
  back-network:
    external: true
  front-network:
    external: true
```

## nginx.conf 주요 설정

```
nginx
코드 복사
server {
  listen 80;
  server_name bbogle.me;

  location / {
    return 301 https://$host$request_uri;
  }
}

server {
```

```
  listen 443 ssl;
  server_name bbogle.me;

  ssl_certificate /etc/letsencrypt/live/bbogle.me/fullchai
n.pem;
  ssl_certificate_key /etc/letsencrypt/live/bbogle.me/privk
ey.pem;

  location / {
    proxy_pass http://frontend;
  }

  location /api {
    proxy_pass http://backend;
  }

  location /ai {
    proxy_pass http://aiservice;
  }
}
```

## 5. Backend 설정

## Dockerfile

```
dockerfile
코드 복사
# 빌드 스테이지
FROM amazoncorretto:21-alpine AS build
WORKDIR /app

# 의존성 설치 및 빌드
COPY build.gradle settings.gradle gradlew ./
COPY gradle ./gradle
COPY src ./src
RUN ./gradlew build --no-daemon

# 실행 스테이지
```

```
FROM amazoncorretto:21-alpine
WORKDIR /app

# 빌드 결과물 복사
COPY --from=build /app/build/libs/bbogle-0.0.1-SNAPSHOT.jar
/app/bbogle-0.0.1-SNAPSHOT.jar

# 애플리케이션 실행
CMD ["java", "-jar", "/app/bbogle-0.0.1-SNAPSHOT.jar"]
```

## docker-compose.yml

```
yaml
코드 복사
services:
  backend:
    image: bbogle-backend:latest
    container_name: bbogle-backend
    networks:
      - back-network
    expose:
      - "8080"

networks:
  back-network:
    external: true
```

## Jenkinsfile

```
groovy
코드 복사
pipeline {
    agent any

    stages {
```

```
        stage('Checkout') {
            steps {
                script {
                    checkout scm
                }
            }
        }
        stage('Build') {
            steps {
                dir('backend') {
                    withCredentials([file(credentialsId: 'A
PPLICATION_YML', variable: 'application_yml')]) {
                        sh 'cp $application_yml ./src/main/
resources/application.yml'
                    }
                    sh './gradlew clean build --no-daemon'
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                sh 'docker build -t bbogle-backend:latest
./backend'
            }
        }
        stage('Deploy') {
            steps {
                dir('backend') {
                    sh 'docker-compose up -d'
                }
            }
        }
        stage('Remove Old Images') {
            steps {
                sh 'docker image prune -f'
            }
        }
    }
```

```
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend(
                    color: 'good',
                    message: "**[성공] Backend 빌드 완료** by
${Author_ID}(${Author_Name})\n${env.BUILD_URL}",
                    endpoint: 'https://meeting.ssafy.com/ho
oks/tcx6yaicaffxufcujwc1qq8qsw',
                    channel: 'B102_Build'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend(
                    color: 'danger',
                    message: "**[실패] Backend 빌드 실패** by
${Author_ID}(${Author_Name})\n${env.BUILD_URL}",
                    endpoint: 'https://meeting.ssafy.com/ho
oks/tcx6yaicaffxufcujwc1qq8qsw',
                    channel: 'B102_Build'
                )
            }
        }
    }
}
```

## 6. Frontend 설정

### Dockerfile

```
dockerfile
코드 복사
FROM node:20 AS build
WORKDIR /react

# 의존성 설치
COPY package.json .
RUN npm install

# 소스 복사 및 빌드
COPY . .
RUN npm run build

# Nginx 기반 실행
FROM nginx:1.26.2-alpine
WORKDIR /
COPY --from=build /react/dist /usr/share/nginx/html
CMD ["nginx", "-g", "daemon off;"]
```

### docker-compose.yml

```
yaml
코드 복사
services:
  frontend:
    image: bbogle-frontend:latest
    container_name: bbogle-frontend
    environment:
      - TZ=Asia/Seoul
    networks:
      - front-network
    expose:
      - "80"
```

```
networks:
  front-network:
    external: true
```

## Jenkinsfile

```
groovy
코드 복사
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                script {
                    checkout scm
                }
            }
        }
        stage('Set .ENV File') {
            steps {
                dir('frontend') {
                    withCredentials([file(credentialsId: 'F
RONT_ENV_FILE', variable: 'front_env_file')]) {
                        sh 'cp $front_env_file ./.env'
                    }
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                sh 'docker build -t bbogle-frontend:latest
./frontend'
            }
        }
        stage('Deploy') {
```

```
            steps {
                dir('frontend') {
                    sh 'docker-compose up -d'
                }
            }
        }
        stage('Remove Old Images') {
            steps {
                sh 'docker image prune -f'
            }
        }
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend(
                    color: 'good',
                    message: "**[성공] Frontend 빌드 완료** by
${Author_ID}(${Author_Name})\n${env.BUILD_URL}",
                    endpoint: 'https://meeting.ssafy.com/ho
oks/tcx6yaicaffxufcujwc1qq8qsw',
                    channel: 'B102_Build'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend(
                    color: 'danger',
                    message: "**[실패] Frontend 빌드 실패** by
```

```
${Author_ID}(${Author_Name})\n${env.BUILD_URL}",
                    endpoint: 'https://meeting.ssafy.com/ho
oks/tcx6yaicaffxufcujwc1qq8qsw',
                    channel: 'B102_Build'
                )
            }
        }
    }
}
```

## 7. Python AI 설정

### Dockerfile

```
FROM python:3.12.3-slim

WORKDIR /app
# ENV PYTHONPATH=/app

COPY requirements.txt /app/requirements.txt

# RUN apt-get update -y

RUN pip install --upgrade pip
# RUN pip install --no-cache-dir -r requirements.txt
RUN pip install --no-cache-dir -r /app/requirements.txt

# COPY . /app
COPY . .
RUN ls -R /app

CMD ["python3", "-m", "app.main"]
```

### docker-compose.yml

```
services:
  festapi:
    image: bbogle-ai:latest
    container_name: bbogle-ai
    networks:
      - back-network

    expose:
      - "8000"

networks:
  back-network:
    external: true
```

## Jenkinsfile

```
 pipeline{
    agent any

    stages {
        stage('Checkout') {
            steps {
                script {
                    checkout scm
                }
            }
        }

        stage('Set .ENV File') {
            steps {
                dir('fast_api') {
                    withCredentials([file(credentialsId: 'A
I_ENV_FILE', variable: 'ai_env_file')]) {
                        sh 'cp $ai_env_file ./.env'
                    }
                }
            }
```

```
        }

        stage('Build Docker Image') {
            steps {
                script {
                     sh 'docker build -t bbogle-ai:latest
./fast_api'
                }
            }
        }

        stage('Depoly') {
            steps {
                dir ('fast_api') {
                    script {
                         sh 'docker-compose up -d'
                    }
                }
            }
        }
        stage('Remove old Image') {
            steps {
                script {
                    sh 'docker image prune -f'
                }
            }
        }

    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'good',
                message: "**윙 치킨 윙 치킨 AI 서버 빌드 성공**
```

```
\n _backend_ \n ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${A
uthor_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)\n :
white_check_mark: ",
                endpoint: 'https://meeting.ssafy.com/hooks/
tcx6yaicaffxufcujwc1qq8qsw',
                channel: 'B102_Build'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'danger',
                message: "** LEGENDDDDDDDDD BUILDDDDDDD F
AILEDDDDDDDDD 빌드 실패** \n _backend_ \n ${env.JOB_NAME} #
${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${en
v.BUILD_URL}/console|Details>)\n :no_entry_sign: ",
                endpoint: 'https://meeting.ssafy.com/hooks/
tcx6yaicaffxufcujwc1qq8qsw',
                channel: 'B102_Build'
                )
            }
        }
    }
}
```

# 6. 기타 서비스

## 1. Redis & MySQL

- docker-compose.yml

```yaml
version: '3.8'

services:
  mysql:
    image: mysql:8.0.20
    container_name: mysql
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: 20241029moongohome
      MYSQL_DATABASE: bbogle
      MYSQL_CHARACTER_SET_SERVER: utf8mb4
      MYSQL_COLLATION_SERVER: utf8mb4_unicode_ci
      TZ: Asia/Seoul
    volumes:
      - /db/data/mysql:/var/lib/mysql
      - /etc/localtime:/etc/localtime:ro
    expose:
      - "3306" # 내부 네트워크에서만 접근 가능
    networks:
      - back-network

  redis:
    image: redis:latest
    container_name: redis
    expose:
      - "6379" # 내부 네트워크에서만 접근 가능
    environment:
      - REDIS_PASSWORD=20241029moongohome
      - TZ=Asia/Seoul
    command: ["redis-server", "--requirepass", "20241029moong
    networks:
      - back-network

volumes:
  mysql_data:

networks:
```

```
  back-network:
    external: true
```

## 2. RabbitMQ

- docker-compose

```
#version: "3.8"
services:
  rabbitmq:
    image: rabbitmq:3-management-alpine
    container_name: bbogle-rabbitmq
    environment:
      RABBITMQ_DEFAULT_USER: bbogle
      RABBITMQ_DEFAULT_PASS: ayebimil
    expose:
      - "5672"    # 메시지 브로커 포트 (내부 통신용)
      - "15672"   # RabbitMQ 관리 인터페이스 포트 (내부 접근용)
        #  ports:
        #    - "5672:5672"    # 외부에서 RabbitMQ 브로커 접근 허용
        #    - "15672:15672"  # 외부에서 RabbitMQ 관리 UI 접근 허용
    volumes:
      - ./.docker/rabbitmq/etc/:/etc/rabbitmq/
      - ./.docker/rabbitmq/data/:/var/lib/rabbitmq/
      - ./.docker/rabbitmq/logs/:/var/log/rabbitmq/
      - ./.docker/enabled_plugins:/etc/rabbitmq/enabled_plugi
    networks:
      - back-network  # 내부 네트워크에서만 접근 가능

networks:
  back-network:
    external: true
```