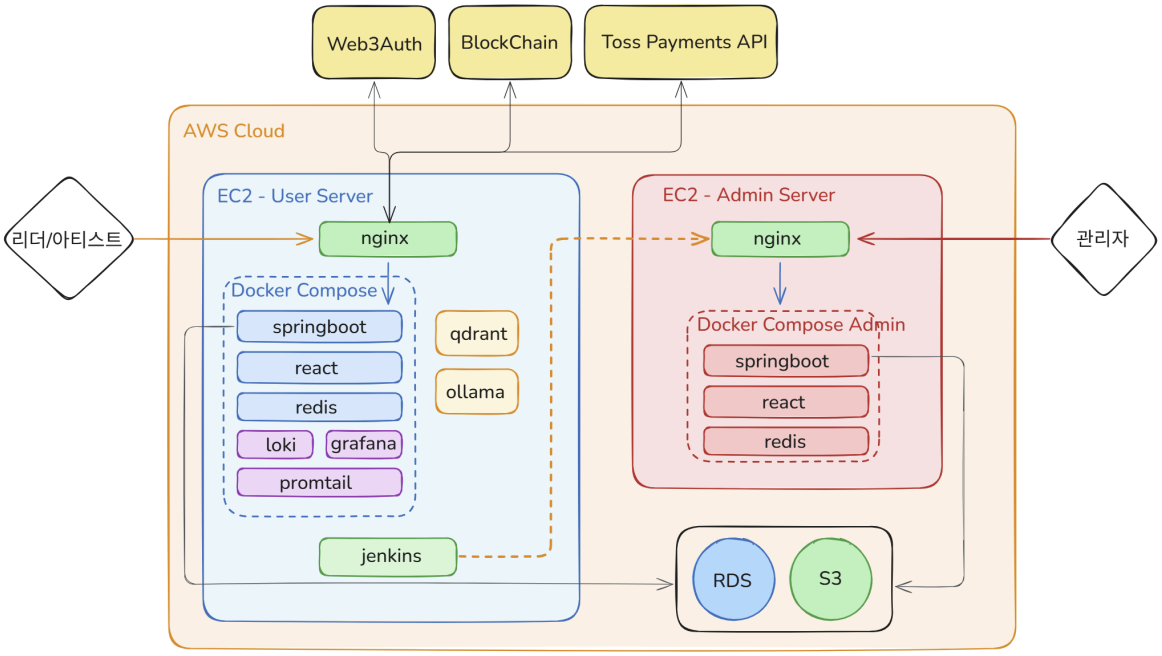
포팅매뉴얼

1. 기술 스택



Frontend

- **Framework**: React 19.1.1
- **Language**: TypeScript 5.9.3
- **Build Tool**: Vite 7.1.7
- **Package Manager**: pnpm 10.21.0
- **Styling**: Tailwind CSS 4.0.0
- **State Management**: Zustand 5.0.8, TanStack Query 5.90.6
- **Routing**: React Router DOM 7.9.5
- **Web3 Integration**: Web3Auth 10.5.6, Wagmi 2.19.2, Viem 2.38.6
- **주요 라이브러리**: Radix UI, Framer Motion 12.23.24, Toast UI Editor 3.2.3, jsPDF 3.0.3

Backend

- **Framework**: Spring Boot 3.5.7
- **Language**: Java 21
- **Build Tool**: Gradle 8.14.3
- **Database**: MySQL
- **Cache**: Redis 7
- **Security**:
  - JWT (io.jsonwebtoken 0.12.5)
  - Spring Security
  - Nimbus JOSE JWT 9.37 (Web3Auth OIDC 검증)
- **Cloud & Storage**
  - AWS S3 (spring-cloud-aws-starter-s3 3.0.3)
  - AWS S3 (software.amazon.awssdk:s3:2.21.29)
- **Blockchain Integration**:
  - Web3j 4.14.0
  - RxJava2 2.2.21
  - Java-WebSocket
- **AI Integration**:
  - Spring AI 1.1.0
  - Qdrant Java Client 1.15.0
  - Spring WebFlux (LLM API 호출)
- **API Documentation**: SpringDoc OpenAPI 2.8.6
- **Logging**: Logstash Logback Encoder 7.4
- **Image Processing**: Thumbnailator 0.4.19, WebP ImageIO 0.1.6
- **Utilities**:
  - Lombok
  - Spring Retry
  - Spring AOP

    - Spring Dotenv 4.0.0

AI

- **Vector Database**: Qdrant (Client 1.15.0)
- **gRPC**: gRPC BOM 1.64.0 (Netty, Stub, Protobuf)
- **Embedding Model**: Ollama
- **LLM**: GPT-4.1 (via SSAFY GMS API)
    - Max Retries: 2
    - Connection Timeout: 10s
    - Read Timeout: 60s
- **Framework**: Spring AI 1.1.0

Blockchain

- **Development Framework**: Hardhat 2.26.3
- **Solidity Version**: 0.8.20 (Contract), 0.8.26 (Web3j Plugin)
- **Network**: Sepolia Testnet (Chain ID: 11155111)
- **Libraries**: OpenZeppelin Contracts 5.4.0
- **Tools**: @nomicfoundation/hardhat-toolbox 6.1.0
- **Standards**: EIP-712 (Typed Data Signing), ERC-1155(Multi Asset Token),
- **Smart Contracts**:
    - MOASContract: 메인 비즈니스 로직
    - MOASForwarder: 메타 트랜잭션 포워딩
- **Backend Integration**:
    - Web3j 4.14.0
    - RxJava2 2.2.21
    - Java-WebSocket

CI/CD

- **Tool**: Jenkins (Pipeline)

- **Containerization**: Docker, Docker Compose
- **Version Control**: GitLab (Webhook 연동)

Monitoring & Logging

- **Stack**: Grafana (latest)+ Loki (latest) + Promtail (latest)
- **Log Format**: JSON (Logback + Logstash Encoder 7.4)
- **Storage**: Volume-mounted log directory

Infrastructure

- **Cloud**: AWS (EC2, S3, RDS)
- **Proxy**: Nginx (SSL/TLS via Let's Encrypt)
- **Payment**: Toss Payments API

2. 환경 변수

.env 설정

로컬 개발 환경에서는 프로젝트 루트에 `.env` 파일을 생성하여 환경 변수를 관리합니다.

Backend 환경 변수

```
# ================================================
# Database
# ================================================
# 개발 환경 (로컬)
DEV_DB_USERNAME=<개발_DB_사용자명>
DEV_DB_PASSWORD=<개발_DB_비밀번호>

# 운영 환경 (RDS)
DB_URL=jdbc:mysql://<RDS_엔드포인트>:3306/moas_db?useSSL=false&ser
verTimezone=Asia/Seoul&characterEncoding=UTF-8
DB_USERNAME=<운영_DB_사용자명>
DB_PASSWORD=<운영_DB_비밀번호>

# ================================================
```

```
# AWS S3
# ================================================
AWS_S3_ACCESS_KEY=<AWS_액세스_키>
AWS_S3_SECRET_KEY=<AWS_시크릿_키>
AWS_S3_BUCKET_NAME=<S3_버킷명>

# ================================================
# 인증 관련
# ================================================
# JWT Secret (최소 256비트 이상)
JWT_SECRET=<JWT_시크릿_키>

# Web3Auth Client ID
WEB3AUTH_AUDIENCE=<Web3Auth_클라이언트_ID>

# ================================================
# Blockchain
# ================================================
# Sepolia Testnet RPC URL
BLOCKCHAIN_RPC_URL=https://eth-sepolia.g.alchemy.com/v2/<YOUR_API
_KEY>
BLOCKCHAIN_WS_URL=wss://eth-sepolia.g.alchemy.com/v2/<YOUR_API_
KEY>

# 서버 지갑 Private Key
SERVER_WALLET_PRIVATE_KEY=0x<서버_지갑_Private_Key>

# Smart Contract 주소
MOAS_CONTRACT_ADDRESS=0x<배포된_MOASContract_주소>

# Sepolia Chain ID
BLOCKCHAIN_CHAIN_ID=11155111

# ================================================
# Payment (Toss Payments)
# ================================================
TOSS_PAYMENTS_SECRET_KEY=<Toss_Payments_시크릿_키>
TOSS_PAYMENTS_API_URL=https://api.tosspayments.com/v1/payments
```

```
# ================================================
# Cache (Redis)
# ================================================
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_PASSWORD=<Redis_비밀번호>

# ================================================
# AI & LLM
# ================================================
# SSAFY GMS API Key
GMS_KEY=<SSAFY_GMS_API_키>

# Qdrant Vector DB
QDRANT_API_KEY=<Qdrant_API_키>

# ================================================
# Security
# ================================================
# 계정 암호화 키 (Base64 인코딩)
ACCOUNT_ENCRYPTION_KEY=<Base64_인코딩된_암호화_키>

# ================================================
# Monitoring (선택사항)
# ================================================
GRAFANA_ADMIN_PASSWORD=<Grafana_관리자_비밀번호>
```

Frontend 환경 변수

프론트엔드 루트 디렉토리에 `.env` 파일을 생성합니다.

```
# Toss Payments Client Key
VITE_TOSS_CLIENT_KEY=<Toss_Payments_클라이언트_키>
VITE_WEB3_AUTH_KEY=<WEB3_Auth_인증키>
```

Jenkins Credentials 설정

Jenkins에서 환경 변수를 안전하게 관리하기 위해 Credentials를 설정합니다.

1. Credentials 추가 방법

1. Jenkins 대시보드 → **Manage Jenkins** → **Manage Credentials**
2. **(global)** 도메인 선택
3. **Add Credentials** 클릭
4. **Kind**: `Secret text` 선택
5. 각 환경 변수마다 다음 정보 입력:
    - **Secret**: 실제 값
    - **ID**: 환경 변수 이름 (예: `JWT_SECRET`)
    - **Description**: 설명 (선택사항)

2. 등록해야 할 Credentials 목록

Database 관련

Credential ID	설명
`DEV_DB_USERNAME`	개발 DB 사용자명
`DEV_DB_PASSWORD`	개발 DB 비밀번호
`DB_URL`	운영 DB URL
`DB_USERNAME`	운영 DB 사용자명
`DB_PASSWORD`	운영 DB 비밀번호

AWS 관련

Credential ID	설명
`AWS_S3_ACCESS_KEY`	AWS Access Key
`AWS_S3_SECRET_KEY`	AWS Secret Key
`AWS_S3_BUCKET_NAME`	S3 버킷명

인증 관련

Credential ID	설명
`JWT_SECRET`	JWT 시크릿 키
`WEB3AUTH_AUDIENCE`	Web3Auth Client ID

Blockchain 관련

Credential ID	설명
`BLOCKCHAIN_RPC_URL`	Blockchain RPC URL
`BLOCKCHAIN_WS_URL`	Blockchain WebSocket URL
`BLOCKCHAIN_CHAIN_ID`	Chain ID (11155111)
`SERVER_WALLET_PRIVATE_KEY`	서버 지갑 Private Key
`MOAS_CONTRACT_ADDRESS`	MOAS Contract 주소

Payment 관련

Credential ID	설명
`TOSS_PAYMENTS_SECRET_KEY`	Toss Payments Secret Key
`TOSS_PAYMENTS_API_URL`	Toss Payments API URL

Redis & AI 관련

Credential ID	설명
`REDIS_PASSWORD`	Redis 비밀번호
`GMS_KEY`	SSAFY GMS API Key
`QDRANT_API_KEY`	Qdrant API Key

Security 관련

Credential ID	설명
`ACCOUNT_ENCRYPTION_KEY`	계정 암호화 키

Frontend 관련

Credential ID	설명
`VITE_TOSS_CLIENT_KEY`	Toss Client Key (Frontend)
`VITE_WEB3_AUTH_KEY`	Web3Auth 인증키

Monitoring 관련

Credential ID	설명
`GRAFANA_ADMIN_PASSWORD`	Grafana 관리자 비밀번호

3. SSH Credentials (관리자 서버 배포용)

관리자 서버로 배포하기 위해 SSH 접속용 PEM 키를 등록합니다.

등록 방법:

1. Jenkins 대시보드 → **Manage Jenkins** → **Manage Credentials**
2. **(global)** 도메인 선택
3. **Add Credentials** 클릭
4. 다음 정보 입력:
    - **Kind**: `SSH Username with private key` 선택
    - **ID**: `admin-server-ssh`
    - **Description**: `관리자 서버 SSH 접속용 PEM 키` (선택사항)
    - **Username**: `ubuntu`
    - **Private Key**:
        - **Enter directly** 선택
        - PEM 키 파일 내용 전체 복사 (`----BEGIN RSA PRIVATE KEY-----`부터 `----END RSA PRIVATE KEY-----` 까지)
        - 또는 **File** 선택 후 `.pem` 파일 직접 업로드
    - **Passphrase**: PEM 키에 암호가 설정되어 있다면 입력 (없으면 공백)

Jenkinsfile에서 사용:

```
stage('Deploy Admin Server') {
    steps {
        sshagent(['admin-server-ssh']) {
            sh '''
                # SSH 연결 테스트 (StrictHostKeyChecking=no로 자동 승인)
                ssh -o StrictHostKeyChecking=no ${ADMIN_USER}@${ADMIN_S
ERVER} "echo 'SSH 연결 성공'"

                # 파일 전송
                scp -r ./backend ${ADMIN_USER}@${ADMIN_SERVER}:${ADMIN
_DIR}/

                # 원격 명령 실행
                ssh ${ADMIN_USER}@${ADMIN_SERVER} "cd ${ADMIN_DIR} &&
docker compose up -d"
            '''
        }
```

```
    }
}
```

주의사항:

- PEM 키 파일의 권한이 너무 개방적이면 SSH 접속이 거부될 수 있습니다 (`chmod 400 your-key.pem` 필요)
- Jenkins에 등록 시에는 파일 권한 문제가 없으므로 내용만 복사하면 됩니다
- `StrictHostKeyChecking=no` 옵션은 최초 접속 시 호스트 키 검증을 건너뛰기 위한 설정입니다

Docker Compose 환경 변수

Docker Compose는 `.env` 파일을 자동으로 읽어 환경 변수를 주입합니다.

사용자 서버 (docker-compose.yml)

```
services:
  backend:
    env_file:
      - .env  # 모든 환경 변수 로드
    environment:
      - SPRING_PROFILES_ACTIVE=prod
      - SPRING_DATA_REDIS_HOST=moas-redis
      - SPRING_DATA_REDIS_PORT=6379
      - SPRING_DATA_REDIS_PASSWORD=${REDIS_PASSWORD}
    volumes:
      - /var/jenkins_home/workspace/moas_release/backend-logs:/logs

  frontend:
    build:
      context: ./frontend/moas
      args:
        - VITE_API_URL=http://K13S401.p.ssafy.io:8081

  redis:
    command: redis-server --requirepass ${REDIS_PASSWORD}
    env_file:
```

```
    - .env

  grafana:
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_ADMIN_PASSWOR
D:-admin}
```

관리자 서버 (docker-compose-admin.yml)

```
services:
  backend:
    env_file:
      - .env
    environment:
      - SPRING_PROFILES_ACTIVE=admin
      - SPRING_DATA_REDIS_HOST=moas-redis
      - SPRING_DATA_REDIS_PORT=6379
      - SPRING_DATA_REDIS_PASSWORD=${REDIS_PASSWORD}
    volumes:
      - ./backend-logs:/logs

  frontend:
    build:
      context: ./frontend/moas
      args:
        - VITE_API_URL=http://54.180.99.55

  redis:
    command: redis-server --requirepass ${REDIS_PASSWORD}
```

환경 변수 주입 흐름

1. **Jenkins Pipeline 실행 시작**
2. **Deploy Backend 스테이지**에서 `withCredentials` 블록을 통해 Jenkins Credentials를 환경 변수로 로드
3. 쉘 스크립트로 `.env` 파일 생성:

```
echo 'DB_USERNAME=${DB_USERNAME}' > .envecho 'DB_PASSWORD
=${DB_PASSWORD}' >> .env# ... 기타 환경 변수
```

4. **Docker Compose**가 `.env` 파일을 읽어 컨테이너에 환경 변수 주입
5. Backend 컨테이너는 `env_file: .env`를 통해 모든 환경 변수 사용
6. Frontend 컨테이너는 빌드 시 `args`를 통해 환경 변수 주입

Jenkinsfile 환경 변수 주입 예시

```
stage('Deploy Backend') {
    steps {
        withCredentials([
            string(credentialsId: 'DB_USERNAME', variable: 'DB_USERNAME'),
            string(credentialsId: 'JWT_SECRET', variable: 'JWT_SECRET'),
            // ... 기타 credentials
        ]) {
            sh """
                echo 'DB_USERNAME=\${DB_USERNAME}' > .env
                echo 'JWT_SECRET=\${JWT_SECRET}' >> .env
                # ... 기타 환경 변수

                docker-compose up -d --build backend redis
            """
        }
    }
}
```

3. 설정 파일

Backend

application.yml (공통 설정)

경로: `backend/moas/src/main/resources/application.yml`

```
log:
  path: /logs

server:
  port: 8080

spring:
  profiles:
    active: dev

  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 110MB

  cloud:
    aws:
      credentials:
        access-key: ${AWS_S3_ACCESS_KEY}
        secret-key: ${AWS_S3_SECRET_KEY}
      region:
        static: ap-northeast-2
      stack:
        auto: false
      s3:
        bucket: ${AWS_S3_BUCKET_NAME}

  # Redis 공통 설정
  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
      password: ${REDIS_PASSWORD}
      timeout: 3000

  # JWT 기본 구조 (각 프로파일에서 expires-ms 오버라이드)
  jwt:
    secret: ${JWT_SECRET}
```

```
# 파일 업로드 제한
portfolio:
  file:
    max-images: 10
    max-image-size: 10485760 # 10MB
    max-total-file-size: 104857600 # 100MB
    allowed-image-types: image/jpeg,image/png,image/gif,image/webp

inquiry:
  file:
    max-files: 10
    max-file-size: 10485760
    max-total-size: 104857600

chat:
  file:
    max-files: 10
    max-file-size: 10485760
    max-total-size: 104857600

# Web3Auth
web3auth:
  issuer: https://api-auth.web3auth.io
  audience: ${WEB3AUTH_AUDIENCE}

# Swagger
springdoc:
  api-docs:
    path: /v3/api-docs
  swagger-ui:
    operations-sorter: alpha
    tags-sorter: alpha
    try-it-out-enabled: true

# LLM (GMS) 설정
llm:
  gms:
```

```
    api-key: ${GMS_KEY}
    model: gpt-4.1
    api-url: https://gms.ssafy.io/gmsapi/api.openai.com/v1/chat/completions
  max-retries: 2
  timeout:
    connect: 10000
    read: 60000

# Qdrant 설정
qdrant:
  host: k13s401.p.ssafy.io
  port: 6334
  api-key: ${QDRANT_API_KEY}

# Embedding 설정
embedding:
  url: http://k13s401.p.ssafy.io:11434/api/embeddings

# Blockchain 공통 설정
blockchain:
  network:
    rpc-url: ${BLOCKCHAIN_RPC_URL}
    ws-url: ${BLOCKCHAIN_WS_URL}
  wallet:
    private-key: ${SERVER_WALLET_PRIVATE_KEY}
  contract:
    moas-address: ${MOAS_CONTRACT_ADDRESS}
    forwarder-address: ${FORWARDER_CONTRACT_ADDRESS}

# EIP712 설정
eip712:
  domain:
    name: "MOASContract"
    version: "1"
  chain-id: ${BLOCKCHAIN_CHAIN_ID}
  verifying-contract: ${MOAS_CONTRACT_ADDRESS}

# Toss Payments
```

```
toss:
  payments:
    secret-key: ${TOSS_PAYMENTS_SECRET_KEY}
    api-url: ${TOSS_PAYMENTS_API_URL}

# 암호화 설정
app:
  encryption:
    account-key: ${ACCOUNT_ENCRYPTION_KEY}
```

---

application-dev.yml (개발 환경)

경로: `backend/moas/src/main/resources/application-dev.yml`

```
spring:
  config:
    activate:
      on-profile: dev

  # 데이터베이스 설정 (로컬)
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://k13s401.p.ssafy.io:3306/moas_db?serverTimezone=Asi
a/Seoul&characterEncoding=UTF-8
    username: ${DEV_DB_USERNAME}
    password: ${DEV_DB_PASSWORD}
    hikari:
      maximum-pool-size: 10
      minimum-idle: 5
      connection-timeout: 30000

  # JPA 설정
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
```

```
        format_sql: true
        show_sql: true
    show-sql: true

  # JWT 설정 (개발용 - 긴 만료시간)
  jwt:
    secret: ${JWT_SECRET}
    access:
      expires-ms: 2592000000 # 30일
    refresh:
      expires-ms: 2592000000 # 30일

# CORS 설정 (로컬 개발용)
app:
  cors:
    allowed-origins:
      - http://localhost:3000
      - http://localhost:5173
      - http://localhost:5174
    allowed-headers: [Content-Type, Authorization, X-CSRF-Token]
    allowed-methods: [GET, POST, PUT, PATCH, DELETE, OPTIONS]
    allow-credentials: true
  metadata:
    base-uri: http://localhost:8080/api/metadata

# 로깅
logging:
  level:
    root: info
    com.s401.moas: debug
    org.hibernate.SQL: debug
    org.hibernate.type.descriptor.sql.BasicBinder: trace
```

---

application-prod.yml (운영 환경)

경로: `backend/moas/src/main/resources/application-prod.yml`

```
spring:
  config:
    activate:
      on-profile: prod

  # 데이터베이스 설정 (RDS)
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${DB_URL}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    hikari:
      maximum-pool-size: 10
      minimum-idle: 5
      connection-timeout: 30000

  # JPA 설정 (프로덕션 - validate)
  jpa:
    hibernate:
      ddl-auto: validate
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
        format_sql: true
        show_sql: false
    show-sql: false

  # JWT 설정 (프로덕션 - 짧은 만료시간)
  jwt:
    secret: ${JWT_SECRET}
    access:
      expires-ms: 900000 # 15분
    refresh:
      expires-ms: 1209600000 # 14일

# CORS 설정 (프로덕션용)
app:
  cors:
```

```
    allowed-origins:
      - https://k13s401.p.ssafy.io
    allowed-headers: [Content-Type, Authorization, X-CSRF-Token]
    allowed-methods: [GET, POST, PUT, PATCH, DELETE, OPTIONS]
    allow-credentials: true
  metadata:
    base-uri: https://k13s401.p.ssafy.io/api/metadata

# 로깅 (프로덕션 - 간결하게)
logging:
  level:
    root: info
    com.s401.moas: info
    org.hibernate.SQL: warn
```

application-admin.yml (관리자 서버)

경로: `backend/moas/src/main/resources/application-admin.yml`

```
spring:
  config:
    activate:
      on-profile: admin

  # 데이터베이스 설정 (RDS - prod와 동일)
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${DB_URL}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    hikari:
      maximum-pool-size: 10
      minimum-idle: 5
      connection-timeout: 30000

  # JPA 설정 (update - 스키마 변경 가능)
  jpa:
    hibernate:
```

```
    ddl-auto: update
  properties:
    hibernate:
      dialect: org.hibernate.dialect.MySQLDialect
      format_sql: true
      show_sql: true
  show-sql: true

 # JWT 설정 (관리자용 - 긴 만료시간)
 jwt:
  secret: ${JWT_SECRET}
  access:
    expires-ms: 2592000000 # 30일
  refresh:
    expires-ms: 2592000000 # 30일

# CORS 설정 (관리자용)
app:
 cors:
  allowed-origins:
    - http://localhost:3000
    - http://localhost:5173
    - http://localhost:5174
    - http://54.180.99.55
  allowed-headers: [Content-Type, Authorization, X-CSRF-Token]
  allowed-methods: [GET, POST, PUT, PATCH, DELETE, OPTIONS]
  allow-credentials: true
 metadata:
  base-uri: http://54.180.99.55/admin/api/metadata

# Swagger 설정 추가!
springdoc:
 api-docs:
  path: /v3/api-docs
 swagger-ui:
  operations-sorter: alpha
  tags-sorter: alpha
  try-it-out-enabled: true
```

```
# 로깅
logging:
  level:
    root: info
    com.s401.moas: debug
    org.hibernate.SQL: debug
    org.hibernate.type.descriptor.sql.BasicBinder: trace
```

---

build.gradle

경로: `backend/moas/build.gradle`

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.5.7'
    id 'io.spring.dependency-management' version '1.1.7'
    id 'org.web3j' version '4.14.0'
}
ext {
    springAiVersion = "1.1.0"
}

group = 'com.s401'
version = '0.0.1-SNAPSHOT'
description = 'Demo project for Spring Boot'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}
```

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-websock
et'
    implementation 'me.paulschwarz:spring-dotenv:4.0.0'
    implementation 'io.awspring.cloud:spring-cloud-aws-starter-s3:3.0.3'
    implementation 'software.amazon.awssdk:s3:2.21.29'
    implementation 'net.coobird:thumbnailator:0.4.19'
    implementation 'org.sejda.imageio:webp-imageio:0.1.6'
    implementation 'org.springframework.boot:spring-boot-starter-validatio
n:3.4.1'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.
8.6'
    implementation 'net.logstash.logback:logstash-logback-encoder:7.4'
    implementation 'org.springframework.boot:spring-boot-starter-aop'
    implementation 'org.springframework.retry:spring-retry'

    // ===== 우리 서비스용 Access JWT 발급/검증 =====
    implementation 'io.jsonwebtoken:jjwt-api:0.12.5'
    runtimeOnly   'io.jsonwebtoken:jjwt-impl:0.12.5'
    runtimeOnly   'io.jsonwebtoken:jjwt-jackson:0.12.5'

    // ===== Web3Auth OIDC id_token 서명/JWKS 검증 =====
    implementation 'com.nimbusds:nimbus-jose-jwt:9.37'

    // ==== Smart Contract 상호작용 Web3 ====
    implementation 'org.web3j:core:4.14.0'
    implementation 'io.reactivex.rxjava2:rxjava:2.2.21'
    implementation 'org.java-websocket:Java-WebSocket'

    compileOnly 'org.projectlombok:lombok'
```

```
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'

    // ===== Refresh Token Redis 설정 =====
    implementation 'org.springframework.boot:spring-boot-starter-data-redi
s'
    testImplementation 'com.h2database:h2'
    testImplementation 'me.paulschwarz:spring-dotenv:4.0.0'
    testCompileOnly 'org.projectlombok:lombok'
    testAnnotationProcessor 'org.projectlombok:lombok'

    // ==== QDrant Vector DB ====
    implementation "io.qdrant:client:1.15.0"

    // gRPC BOM으로 버전 정렬
    implementation platform("io.grpc:grpc-bom:1.64.0")
    implementation "io.grpc:grpc-netty-shaded"  // Transport (Netty)
    implementation "io.grpc:grpc-stub"
    implementation "io.grpc:grpc-protobuf"

    implementation "org.springframework.boot:spring-boot-starter-webflux"
}

sourceSets {
    main {
        solidity {
            srcDirs = [
                'src/main/solidity',
                'node_modules'
            ]
        }
    }
}
```

```
tasks.named('test') {
    useJUnitPlatform()
}

web3j {
    generatedPackageName = 'com.s401.moas.blockchain.wrapper'
    solidity {
        version = '0.8.26'
    }
}
dependencyManagement {
    imports {
        mavenBom "org.springframework.ai:spring-ai-bom:$springAiVersion"
    }
}

project.afterEvaluate {
    tasks.named('generateContractWrappers') {
        dependsOn tasks.named('processResources')
    }
}
```

logback-spring.xml

경로: backend/moas/src/main/resources/logback-spring.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <!-- 로그 파일 경로 설정 -->
    <property name="LOG_PATH" value="${log.path:-logs}"/>
    <property name="LOG_FILE" value="${LOG_PATH}/spring.log"/>

    <!-- 콘솔 출력 -->
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppe
nder">
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger
{36} - %msg%n</pattern>
```

```
        </encoder>
    </appender>

    <!-- JSON 형식 파일 출력 (Promtail용) -->
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_FILE}</file>
        <encoder class="net.logstash.logback.encoder.LogstashEncoder">
            <includeMdcKeyName>traceId</includeMdcKeyName>
            <includeMdcKeyName>spanId</includeMdcKeyName>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG_PATH}/spring.%d{yyyy-MM-dd}.log</fileNamePattern>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
    </appender>

    <!-- Root Logger -->
    <root level="INFO">
        <appender-ref ref="CONSOLE"/>
        <appender-ref ref="FILE"/>
    </root>

    <!-- 개발 환경 -->
    <springProfile name="dev">
        <logger name="com.s401.moas" level="DEBUG"/>
        <logger name="org.hibernate.SQL" level="DEBUG"/>
        <logger name="org.hibernate.type.descriptor.sql.BasicBinder" level="TRACE"/>
    </springProfile>

    <!-- 운영 환경 -->
    <springProfile name="prod">
        <logger name="com.s401.moas" level="INFO"/>
        <logger name="org.hibernate.SQL" level="WARN"/>
    </springProfile>
```

```
    <!-- 관리자 환경 -->
    <springProfile name="admin">
        <logger name="com.s401.moas" level="DEBUG"/>
        <logger name="org.hibernate.SQL" level="DEBUG"/>
        <logger name="org.hibernate.type.descriptor.sql.BasicBinder" level
="TRACE"/>
    </springProfile>
</configuration>
```

Dockerfile (Backend)

경로: `backend/moas/Dockerfile`

```
FROM gradle:8.5-jdk21

WORKDIR /app

# Node.js 20.x 설치 (공식 저장소)
RUN apt-get update && \
    apt-get install -y tzdata curl && \
    curl -fsSL https://deb.nodesource.com/setup_20.x | bash - && \
    apt-get install -y nodejs && \
    npm install -g pnpm && \
    ln -snf /usr/share/zoneinfo/Asia/Seoul /etc/localtime && \
    echo "Asia/Seoul" > /etc/timezone && \
    apt-get clean

# 1. pnpm 의존성 파일 먼저 (변동 거의 없음)
COPY package.json pnpm-lock.yaml ./

# 2. pnpm 의존성 설치 (캐싱)
RUN pnpm install

# 3. Gradle 의존성 파일 복사 (가끔 변동)
COPY build.gradle settings.gradle gradlew ./
COPY gradle gradle/
```

```
# 4. Gradle 의존성 다운로드 (캐싱)
RUN chmod +x ./gradlew && \
    ./gradlew dependencies --no-daemon || true

# 5. 소스 코드 복사 (자주 변동)
COPY . .

# 6. gradlew 권한 재부여 (COPY . . 후!)
RUN chmod +x ./gradlew

# 7. 빌드
RUN ./gradlew bootJar --no-daemon

EXPOSE 8080

CMD ["sh", "-c", "java -Duser.timezone=Asia/Seoul -jar build/libs/*.jar"]
```

Frontend

package.json

경로: `frontend/moas/package.json`

```
{
  "name": "moas",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "packageManager": "pnpm@10.21.0",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview",
    "convert-fonts": "tsx scripts/convertFontToBase64.ts"
  },
  "dependencies": {
    "@radix-ui/react-dialog": "^1.1.15",
```

```
    "@radix-ui/react-popover": "^1.1.15",
    "@radix-ui/react-select": "^2.2.6",
    "@radix-ui/react-slot": "^1.2.3",
    "@tanstack/react-query": "^5.90.6",
    "@toast-ui/editor": "^3.2.2",
    "@toast-ui/react-editor": "^3.2.3",
    "@web3auth/modal": "^10.5.6",
    "axios": "^1.13.1",
    "buffer": "^6.0.3",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "date-fns": "^4.1.0",
    "fontkit": "^2.0.4",
    "framer-motion": "^12.23.24",
    "html2canvas": "^1.4.1",
    "jspdf": "^3.0.3",
    "jwt-decode": "^4.0.0",
    "lucide-react": "^0.548.0",
    "marked": "^17.0.0",
    "process": "^0.11.10",
    "react": "^19.1.1",
    "react-day-picker": "^9.11.1",
    "react-dom": "^19.1.1",
    "react-easy-crop": "^5.5.3",
    "react-intersection-observer": "^10.0.0",
    "react-lottie": "^1.2.10",
    "react-lottie-player": "^2.1.0",
    "react-router-dom": "^7.9.5",
    "tailwind-merge": "^3.3.1",
    "viem": "^2.38.6",
    "wagmi": "^2.19.2",
    "zustand": "^5.0.8"
  },
  "devDependencies": {
    "@eslint/js": "^9.36.0",
    "@tailwindcss/vite": "^4.0.0",
    "@types/node": "^24.6.0",
    "@types/react": "^19.1.16",
```

```
    "@types/react-dom": "^19.1.9",
    "@vitejs/plugin-react": "^5.0.4",
    "eslint": "^9.36.0",
    "eslint-config-prettier": "^10.1.8",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.22",
    "globals": "^16.4.0",
    "prettier": "^3.6.2",
    "tailwindcss": "^4.0.0",
    "tsx": "^4.20.6",
    "tw-animate-css": "^1.4.0",
    "typescript": "~5.9.3",
    "typescript-eslint": "^8.45.0",
    "vite": "^7.1.7"
  }
}
```

vite.config.ts

경로: frontend/moas/vite.config.ts

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import tailwindcss from '@tailwindcss/vite';
import path from 'path';

export default defineConfig({
  plugins: [react(), tailwindcss()],
  define: {
    'process.env': {},
    global: 'globalThis',
  },
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './src'),
      buffer: 'buffer',
      process: 'process',
    },
```

```
  },
  optimizeDeps: {
    esbuildOptions: {
      define: {
        global: 'globalThis',
      },
    },
  },
  server: {
    proxy: {
      '/api': {
        target: 'https://k13s401.p.ssafy.io', //서버
        // target: 'http://localhost:8080', // 로컬
        changeOrigin: true,
        secure: false,
      },
    },
  },
});
```

Dockerfile (Frontend)

경로: `frontend/moas/Dockerfile`

```
FROM node:20-alpine AS build

WORKDIR /app

# pnpm 설치
RUN corepack enable && corepack prepare pnpm@latest --activate

# 의존성 파일 복사
COPY package.json pnpm-lock.yaml ./

# 의존성 설치 (빌드 스크립트 승인)
RUN pnpm install --frozen-lockfile

# 소스 코드 복사
```

```
COPY . .

# 빌드 인수 설정
ARG VITE_API_URL
ENV VITE_API_URL=${VITE_API_URL}

# 빌드
RUN pnpm run build

# 프로덕션 이미지
FROM nginx:alpine
COPY --from=build /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

---

nginx.conf

경로: `frontend/moas/nginx.conf`

```
server {
    listen 80;
    server_name _;
    root /usr/share/nginx/html;
    index index.html;

    # ===== Gzip 압축 설정 =====
    gzip on;
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_types
        text/plain
        text/css
        text/xml
        text/javascript
        application/json
        application/javascript
```

```
        application/xml+rss
        application/rss+xml
        font/truetype
        font/opentype
        application/vnd.ms-fontobject
        image/svg+xml;
    gzip_min_length 1000;

    # ===== 정적 파일 브라우저 캐싱 =====
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        gzip_static on;
    }

    # ===== index.html은 절대 캐싱하지 않음 =====
    location = /index.html {
        add_header Cache-Control "no-cache, no-store, must-revalidate";
        add_header Pragma "no-cache";
        add_header Expires "0";
    }

    # ===== SPA 라우팅 지원 =====
    location / {
        try_files $uri $uri/ /index.html;
    }

    # ===== 보안 헤더 =====
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
}
```

Blockchain

hardhat.config.ts

경로: `blockchain/hardhat.config.ts`

```
// hardhat.config.ts
import { HardhatUserConfig } from "hardhat/config";
import "@nomicfoundation/hardhat-toolbox";
import "dotenv/config"; // .env 파일 로드를 위해 추가

// 환경 변수 로드
const SEPOLIA_RPC_URL = process.env.SEPOLIA_RPC_URL || "";
const PRIVATE_KEY = process.env.PRIVATE_KEY || "";
const ETHERSCAN_API_KEY = process.env.ETHERSCAN_API_KEY || "";

// PRIVATE_KEY가 '0x'로 시작하지 않으면 추가
const deployerPrivateKey = PRIVATE_KEY.startsWith("0x")
  ? PRIVATE_KEY
  : "0x" + PRIVATE_KEY;

const config: HardhatUserConfig = {
  solidity: {
    version: "0.8.20", // 컨트랙트 pragma와 일치
    settings: {
      optimizer: {
        enabled: true,
        runs: 200,
      },
    },
  },
  defaultNetwork: "hardhat", // 로컬 테스트를 위한 기본 네트워크
  networks: {
    hardhat: {
      // 로컬 테스트 환경
    },
    sepolia: {
      url: SEPOLIA_RPC_URL,
      accounts: [deployerPrivateKey],
      chainId: 11155111, // Sepolia 체인 ID
    },
  },
  etherscan: {
    // Etherscan 자동 검증을 위한 설정
```

```
    apiKey: ETHERSCAN_API_KEY,
  },
  paths: {
    sources: "./contracts", // .sol 파일 위치 (기본값)
    tests: "./test",       // .ts 테스트 파일 위치 (기본값)
    cache: "./cache",
    artifacts: "./artifacts",
  },
};

export default config;
```

package.json (Blockchain)

경로: `blockchain/package.json`

```
{
  "devDependencies": {
    "@nomicfoundation/hardhat-toolbox": "^6.1.0",
    "dotenv": "^17.2.3",
    "hardhat": "^2.26.3"
  },
  "dependencies": {
    "@openzeppelin/contracts": "^5.4.0"
  }
}
```

tsconfig.json (Blockchain)

경로: `blockchain/tsconfig.json`

```
{
  "compilerOptions": {
    "target": "es2020",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
```

```
    "skipLibCheck": true,
    "resolveJsonModule": true
  }
}
```

Docker Compose

docker-compose.yml (사용자 서버)

경로: `docker-compose.yml`

```
version: '3.8'

services:
  backend:
    build: ./backend/moas
    container_name: moas-backend-prod
    ports:
      - "8081:8080"
    env_file:
      - .env
    environment:
      - SPRING_PROFILES_ACTIVE=prod
      - SPRING_DATA_REDIS_HOST=moas-redis
      - SPRING_DATA_REDIS_PORT=6379
      - SPRING_DATA_REDIS_PASSWORD=${REDIS_PASSWORD}
    volumes:
      - /var/jenkins_home/workspace/moas_release/backend-logs:/logs
    depends_on:
      - redis
    networks:
      - moas-network
    restart: unless-stopped

  frontend:
    build:
      context: ./frontend/moas
      args:
```

```
      - VITE_API_URL=http://K13S401.p.ssafy.io:8081
    container_name: moas-frontend-prod
    ports:
      - "3000:80"
    depends_on:
      - backend
    networks:
      - moas-network
    restart: unless-stopped

  redis:
    image: redis:7-alpine
    container_name: moas-redis
    ports:
      - "127.0.0.1:6379:6379"
    command: redis-server --requirepass ${REDIS_PASSWORD}
    volumes:
      - redis-data:/data
    networks:
      - moas-network
    restart: unless-stopped
    env_file:
      - .env

  loki:
    image: grafana/loki:latest
    container_name: moas-loki
    ports:
      - "127.0.0.1:3100:3100"
    networks:
      - moas-network
    restart: unless-stopped

  grafana:
    image: grafana/grafana:latest
    container_name: moas-grafana
    ports:
      - "3001:3000"
```

```
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_ADMIN_PASSWOR
D:-admin}
    volumes:
      - grafana-data:/var/lib/grafana
    depends_on:
      - loki
    networks:
      - moas-network
    restart: unless-stopped

  promtail:
    image: grafana/promtail:latest
    container_name: moas-promtail
    volumes:
      - /var/jenkins_home/workspace/moas_release/promtail-config.yml:/tm
p/promtail-config.yml:ro
      - /var/jenkins_home/workspace/moas_release/backend-logs:/logs:ro
    command: ["-config.file=/tmp/promtail-config.yml", "-config.expand-env
=true"]
    depends_on:
      - loki
      - backend
    networks:
      - moas-network
    restart: unless-stopped

networks:
  moas-network:

volumes:
  redis-data:
  grafana-data:
```

docker-compose-admin.yml (관리자 서버)

경로: `docker-compose-admin.yml`

```
version: '3.8'

services:
  backend:
    build: ./backend/moas
    container_name: moas-backend-admin
    ports:
      - "8080:8080"
    env_file:
      - .env
    environment:
      - SPRING_PROFILES_ACTIVE=admin
      - SPRING_DATA_REDIS_HOST=moas-redis
      - SPRING_DATA_REDIS_PORT=6379
      - SPRING_DATA_REDIS_PASSWORD=${REDIS_PASSWORD}
    volumes:
      - ./backend-logs:/logs
    depends_on:
      - redis
    networks:
      - moas-network
    restart: unless-stopped

  frontend:
    build:
      context: ./frontend/moas
      args:
        - VITE_API_URL=http://54.180.99.55
    container_name: moas-frontend-admin
    ports:
      - "8090:80"
    depends_on:
      - backend
    networks:
      - moas-network
    restart: unless-stopped

  redis:
```

```
    image: redis:7-alpine
    container_name: moas-redis
    ports:
      - "127.0.0.1:6379:6379"
    command: redis-server --requirepass ${REDIS_PASSWORD}
    volumes:
      - redis-data:/data
    networks:
      - moas-network
    restart: unless-stopped

networks:
  moas-network:

volumes:
  redis-data:
```

Monitoring

promtail-config.yml

경로: promtail-config.yml

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://loki:3100/loki/api/v1/push

scrape_configs:
  - job_name: moas-backend-logs
    static_configs:
      - targets:
          - localhost
```

```
      labels:
        job: moas-backend
        __path__: /logs/spring*.log
    pipeline_stages:
      - json:
          expressions:
            level: level
            thread: thread
            logger: logger
            message: message
            stack_trace: stack_trace
      - timestamp:
          source: '@timestamp'
          format: RFC3339
```

4. EC2 포트 설정

사용자 서버

시스템 정보

- **OS**: Ubuntu 22.04.5 LTS (Jammy Jellyfish)
- **Kernel**: 6.8.0-1031-aws
- **Private IP**: 172.26.12.91
- **Public Domain**: k13s401.p.ssafy.io

현재 사용 중인 포트

포트	서비스	접근 범위	설명
22	SSH	외부	SSH 접속
80	Nginx (HTTP)	외부	HTTPS 리다이렉트
443	Nginx (HTTPS)	외부	메인 웹 서비스 (SSL)
3000	Frontend (React)	localhost	사용자 프론트엔드
3001	Grafana	외부	모니터링 대시보드
3100	Loki	localhost	로그 수집 서버

포트	서비스	접근 범위	설명
3306	MySQL	외부	데이터베이스
6333-6334	Qdrant	외부	Vector Database (gRPC, REST)
6379	Redis	localhost	캐시 서버
8080	Jenkins	외부	CI/CD 서버
8081	Backend (Spring Boot)	localhost	백엔드 API (사용자)
8988	GitLab	외부	Git Repository
8989	GitLab (HTTP)	외부	Git Repository Web UI
11434	Ollama	외부	AI Embedding 서비스
29418	GitLab (SSH)	외부	Git SSH 접속
50000	Jenkins Agent	외부	Jenkins 빌드 에이전트
33060	MySQL X Protocol	localhost	MySQL 확장 프로토콜
61209	Glances	localhost	시스템 모니터링

Docker 컨테이너 포트 매핑

컨테이너명	포트 매핑	상태
moas-backend-prod	8081:8080	Running
moas-frontend-prod	3000:80	Running
moas-redis	127.0.0.1:6379:6379	Running
moas-grafana	3001:3000	Running
moas-loki	127.0.0.1:3100:3100	Running
moas-promtail	-	Running
jenkins	8080:8080, 50000:50000	Running
qdrant	6333-6334:6333-6334	Running
ollama	11434:11434	Running

방화벽 설정 (UFW)

```
# 기본 개방 포트
22      # SSH
80      # HTTP
```

```
443        # HTTPS
3306       # MySQL
6333-6334  # Qdrant
8080       # Jenkins
8989       # GitLab Web UI

# 관리자 서버 전용 접근
22/tcp from 54.180.99.55    # SSH from admin server
3306 from 54.180.99.55      # MySQL from admin server
```

Nginx 리버스 프록시 설정

경로: `/etc/nginx/sites-available/default`

```
# HTTP → HTTPS 리다이렉트
server {
    listen 80;
    listen [::]:80;
    server_name k13s401.p.ssafy.io;
    return 301 https://$host$request_uri;
}

# HTTPS 메인 서버
server {
    listen 443 ssl;
    listen [::]:443 ssl ipv6only=on;
    server_name k13s401.p.ssafy.io;

    # SSL 인증서 (Let's Encrypt)
    ssl_certificate /etc/letsencrypt/live/k13s401.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k13s401.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    # 파일 업로드 크기 제한
```

```
    client_max_body_size 110M;

    # 타임아웃 설정
    proxy_connect_timeout 300;
    proxy_send_timeout 300;
    proxy_read_timeout 300;

    # 프론트엔드 (React)
    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        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;
    }

    # 백엔드 API (SSE 지원)
    location /api {
        proxy_pass http://localhost:8081;
        proxy_http_version 1.1;
        proxy_set_header Connection '';

        # SSE를 위한 버퍼링 비활성화
        proxy_buffering off;
        proxy_cache off;

        # SSE 타임아웃 설정 (24시간)
        proxy_read_timeout 86400s;
        proxy_send_timeout 86400s;

        chunked_transfer_encoding on;

        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;
}

# Swagger UI
location /swagger-ui {
    proxy_pass http://localhost:8081/swagger-ui;
    proxy_http_version 1.1;
    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;
}

# Swagger API Docs
location /v3/api-docs {
    proxy_pass http://localhost:8081/v3/api-docs;
    proxy_http_version 1.1;
    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;
}

# Grafana (모니터링)
location /grafana {
    proxy_pass http://localhost:3001/;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

# 관리자 백엔드 API (관리자 서버 접근용)
location /admin/api {
    proxy_pass http://localhost:8081;
    proxy_http_version 1.1;
    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;
    }
}
```

관리자 서버

시스템 정보

- **OS**: Ubuntu 22.04.5 LTS (Jammy Jellyfish)
- **Kernel**: 6.8.0-1040-aws
- **Private IP**: 172.31.33.18
- **Public IP**: 54.180.99.55

현재 사용 중인 포트

포트	서비스	접근 범위	설명
22	SSH	외부	SSH 접속
80	Nginx (HTTP)	외부	관리자 웹 서비스
6379	Redis	localhost	캐시 서버
8080	Backend (Spring Boot)	localhost	관리자 백엔드 API
8090	Frontend (React)	localhost	관리자 프론트엔드
34181	Containerd	localhost	컨테이너 런타임

Docker 컨테이너 포트 매핑

컨테이너명	포트 매핑	상태
moas-backend-admin	8080:8080	Running
moas-frontend-admin	8090:80	Running
moas-redis	127.0.0.1:6379:6379	Running

방화벽 설정 (UFW)

```
# 기본 개방 포트
22/tcp      # SSH
```

```
80/tcp      # HTTP
443/tcp     # HTTPS (예약)

# 사용자 서버 전용 접근
22 from 43.201.251.198    # SSH from user server (Jenkins)
```

Nginx 리버스 프록시 설정

경로: `/etc/nginx/sites-available/default`

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

    server_name 54.180.99.55;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    # 파일 업로드 크기 제한
    client_max_body_size 110M;

    # 타임아웃 설정
    proxy_connect_timeout 300;
    proxy_send_timeout 300;
    proxy_read_timeout 300;

    # 루트 접속 시 /admin으로 리다이렉트
    location = / {
        return 301 http://$host/admin;
    }

    # 관리자 프론트엔드
    location / {
            proxy_pass http://localhost:8090/;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
```

```
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        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;
    }

    # 관리자 백엔드 API (SSE 지원)
    location /admin/api {
        proxy_pass http://localhost:8080/admin/api;
        proxy_http_version 1.1;
        proxy_set_header Connection '';

        # SSE를 위한 버퍼링 비활성화
        proxy_buffering off;
        proxy_cache off;

        # SSE 타임아웃 설정 (24시간)
        proxy_read_timeout 86400s;
        proxy_send_timeout 86400s;

        chunked_transfer_encoding on;

        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;
    }

    # Swagger UI (관리자용)
    location /swagger-ui {
        proxy_pass http://localhost:8080/swagger-ui;
        proxy_http_version 1.1;
        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;
    }
```

```
    # Swagger API Docs (관리자용)
    location /v3/api-docs {
        proxy_pass http://localhost:8080/v3/api-docs;
        proxy_http_version 1.1;
        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;
    }

    # Grafana 리다이렉트 (사용자 서버로)
    location /admin/logs {
        return 301 http://k13s401.p.ssafy.io:3001;
    }
}
```

5. 도커 설정

사용자 서버 (k13s401.p.ssafy.io)

서비스 구성

서비스	포트	용도
backend	8081:8080	Spring Boot API 서버
frontend	3000:80	React 프론트엔드
redis	127.0.0.1:6379:6379	캐시 서버
loki	127.0.0.1:3100:3100	로그 수집
grafana	3001:3000	모니터링 대시보드
promtail	-	로그 전송 에이전트

> 설정 파일: docker-compose.yml (3번 섹션 참고)

실행 명령어

```
# 프로젝트 루트 디렉토리에서 실행

# 전체 서비스 빌드 및 실행
docker-compose up -d --build

# 특정 서비스만 재시작
docker-compose up -d --build backend
docker-compose up -d --build frontend

# 로그 확인
docker-compose logs -f backend
docker-compose logs -f frontend

# 컨테이너 상태 확인
docker-compose ps

# 중지 및 제거
docker-compose down
```

관리자 서버 (54.180.99.55)

서비스 구성

서비스	포트	용도
backend	8080:8080	관리자 API 서버 (admin profile)
frontend	8090:80	관리자 프론트엔드
redis	127.0.0.1:6379:6379	캐시 서버

> 설정 파일: docker-compose-admin.yml (3번 섹션 참고)

실행 명령어

```
# 관리자 서버 접속
ssh -i moasAdminKey.pem ubuntu@54.180.99.55

# 배포 디렉토리로 이동
cd /home/ubuntu/moas-admin
```

```
# 전체 서비스 빌드 및 실행
docker compose up -d --build

# 로그 확인
docker compose logs -f backend
docker compose logs -f frontend

# 컨테이너 상태 확인
docker compose ps

# 중지 및 제거
docker compose down
```

주요 환경 변수

Backend:

- `SPRING_PROFILES_ACTIVE`: 프로파일 (prod / admin)
- `SPRING_DATA_REDIS_HOST`: Redis 호스트
- 기타 환경 변수는 `.env` 파일에서 주입 (2번 섹션 참고)

Frontend:

- `VITE_API_URL`: 빌드 시 API URL 주입
  - 사용자 서버: `http://K13S401.p.ssafy.io:8081`
  - 관리자 서버: `http://54.180.99.55`

---

6. CI/CD 설정

Jenkins 설치

Jenkins는 사용자 서버(k13s401.p.ssafy.io)에 Docker 컨테이너로 실행됩니다.

```
# Jenkins 컨테이너 실행
docker run -d \
  --name jenkins \
  -p 8080:8080 \
```

```
  -p 50000:50000 \
  -v jenkins_data:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  --restart unless-stopped \
  jenkins/jenkins:lts

# 초기 비밀번호 확인
docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

접속: `http://k13s401.p.ssafy.io:8080`

필수 플러그인 설치

Manage Jenkins → **Manage Plugins** → **Available**:

- GitLab Plugin
- Docker Pipeline
- SSH Agent Plugin
- Pipeline
- Credentials Binding Plugin

GitLab 연동

1. GitLab Access Token 생성

- GitLab → **Settings** → **Access Tokens**
- Scopes: `api`, `read_repository`, `write_repository`

2. Jenkins Credentials 등록

- **Manage Jenkins** → **Manage Credentials** → **Add Credentials**
- Kind: `GitLab API token`
- ID: `gitlab-token`

3. GitLab Connection 설정

- **Manage Jenkins** → **Configure System** → **GitLab**

- GitLab host URL: `http://k13s401.p.ssafy.io:8989`
- Credentials: `gitlab-token` 선택
- **Test Connection** 클릭

Pipeline 프로젝트 생성

1. 새 Pipeline 생성

- Jenkins 대시보드 → **New Item**
- Item name: `moas_release`
- Type: **Pipeline**

2. Pipeline 설정

Build Triggers:

- ✅ Build when a change is pushed to GitLab

Pipeline:

- Definition: `Pipeline script from SCM`
- SCM: `Git`
- Repository URL: GitLab 저장소 URL
- Branch: `/release`
- Script Path: `Jenkinsfile`

3. GitLab Webhook 설정

- GitLab 프로젝트 → **Settings** → **Webhooks**
- URL: `http://k13s401.p.ssafy.io:8080/project/moas_release`
- Trigger: ✅ Push events
- **Add webhook**

환경 변수 설정

Jenkins Credentials에 모든 환경 변수를 Secret Text로 등록합니다.

> 등록 방법 및 목록: 2번 섹션 "Jenkins Credentials 설정" 참고

Jenkinsfile 구조

> 전체 코드: 3번 섹션 참고

주요 Stage

Stage	설명
Checkout	GitLab에서 소스 코드 가져오기
Stop Containers	기존 컨테이너 중지
Deploy Backend	`.env` 생성 후 Backend & Redis 배포
Deploy Frontend	Frontend 환경 변수 추가 후 배포
Deploy Monitoring	Loki, Grafana, Promtail 배포
Deploy Admin Server	SSH로 관리자 서버에 파일 전송 및 배포
Cleanup	`.env` 삭제 및 Docker 이미지 정리

배포 결과 확인

배포 성공 시 접속 URL:

서비스	URL
사용자 프론트엔드	https://k13s401.p.ssafy.io
사용자 Swagger UI	https://k13s401.p.ssafy.io/swagger-ui/index.html
Grafana	http://k13s401.p.ssafy.io:3001
관리자 프론트엔드	http://54.180.99.55
관리자 Swagger UI	http://54.180.99.55/swagger-ui/index.html

7. 스마트컨트랙트 배포 (moas-blockchain)

의존성 설치: Hardhat을 포함한 모든 개발 의존성을 설치합니다.

```
pnpm install
```

환경변수 파일 설정

```
# Sepolia 테스트넷 접속을 위한 Alchemy (또는 Infura) API Key
ALCHEMY_API_KEY="YOUR_ALCHEMY_API_KEY"

# 컨트랙트 배포에 사용할 지갑의 Private Key (반드시 "0x"로 시작)
# 이 지갑에는 배포를 위한 가스비(Sepolia ETH)가 충분히 있어야 합니다.
SEPOLIA_PRIVATE_KEY="0xYOUR_WALLET_PRIVATE_KEY"

# (선택) Etherscan API Key (배포 후 코드 검증을 위해 필요)
ETHERSCAN_API_KEY="YOUR_ETHERSCAN_API_KEY"
```

컨트랙트 컴파일

```
npx hardhat compile
```

- 컴파일이 성공하면 artifacts/와 cache/ 디렉터리가 생성됩니다.

배포 스크립트 실행

- 블록체인 네트워크 관련 설정을 `hardhat.config.ts` 로 설정합니다.

```
// hardhat.config.ts
import { HardhatUserConfig } from "hardhat/config";
import "@nomicfoundation/hardhat-toolbox";
import "dotenv/config"; // .env 파일 로드를 위해 추가

// 환경 변수 로드
const SEPOLIA_RPC_URL = process.env.SEPOLIA_RPC_URL || "";
const PRIVATE_KEY = process.env.PRIVATE_KEY || "";
const ETHERSCAN_API_KEY = process.env.ETHERSCAN_API_KEY || "";

// PRIVATE_KEY가 '0x'로 시작하지 않으면 추가
const deployerPrivateKey = PRIVATE_KEY.startsWith("0x")
  ? PRIVATE_KEY
  : "0x" + PRIVATE_KEY;
```

```
const config: HardhatUserConfig = {
  solidity: {
    version: "0.8.20", // 컨트랙트 pragma와 일치
    settings: {
      optimizer: {
        enabled: true,
        runs: 200,
      },
    },
  },
  defaultNetwork: "hardhat", // 로컬 테스트를 위한 기본 네트워크
  networks: {
    hardhat: {
      // 로컬 테스트 환경
    },
    sepolia: {
      url: SEPOLIA_RPC_URL,
      accounts: [deployerPrivateKey],
      chainId: 11155111, // Sepolia 체인 ID
    },
  },
  etherscan: {
    // Etherscan 자동 검증을 위한 설정
    apiKey: ETHERSCAN_API_KEY,
  },
  paths: {
    sources: "./contracts", // .sol 파일 위치 (기본값)
    tests: "./test",       // .ts 테스트 파일 위치 (기본값)
    cache: "./cache",
    artifacts: "./artifacts",
  },
};

export default config;
```

- 배포 전 , `npx hardhat test` 를 진행하여 스마트컨트랙트 이상이 없는지 체크할 수 있습니다.
- 해당 커맨드를 입력하여 배포를 진행합니다.

```
npx hardhat run scripts/deploy.ts --network sepolia
```

- 배포 후, 배포가 성공적으로 진행되면 터미널에 아래와 비슷하게 배포된 컨트랙트 주소가 출력됩니다. 백엔드 서버에서 해당 스마트 컨트랙트를 사용하기 위해서는 반드시 기록해야합니다.

```
MOASForwarder deployed to: 0x... (Forwarder 컨트랙트 주소)
MOASContract deployed to: 0x... (MOASContract 주소)
```