포팅메뉴얼

버전

- nginx: 1.18.0 (Ubuntu)
- docker: version 24.0.6, build ed223bc

CI/CD

▼ nginx

```
sudo apt install nginx # nginx 설치
sudo -E vim /etc/nginx/sites-available/default # 설정 파일 아래와 같이 수정
```

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

  root /var/www/html;

  # Add index.php to the list if you are using PHP
  index index.html index.htm index.nginx-debian.html;

  server_name _;

  location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
  }
}

server {
  root /var/www/html;

  index index.html index.htm index.nginx-debian.html;
    server_name j9a501.p.ssafy.io; # managed by Certbot

  access_log  /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;

  location /api/noop {
    client_max_body_size  11M;
    proxy_pass  http://localhost:3000/api/noop;
  }
  location /api/books {
    client_max_body_size  11M;
    proxy_pass  http://localhost:8000/api/books;
  }
  location /api {
    proxy_pass  http://localhost:8000/api;
  }
  location /login {
    proxy_pass  http://localhost:8000/login;
  }

  location /ipfs {
    proxy_pass  http://localhost:8080/ipfs;
  }
  location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    # try_files $uri $uri/ =404;
    proxy_pass  http://localhost:3000;
  }
  listen [::]:443 ssl ipv6only=on; # managed by Certbot
  listen 443 ssl; # managed by Certbot
  ssl_certificate /etc/letsencrypt/live/j9a501.p.ssafy.io/fullchain.pem; # managed by Certbot
```

```
  ssl_certificate_key /etc/letsencrypt/live/j9a501.p.ssafy.io/privkey.pem; # managed by Certbot
  include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}
server {
    if ($host = j9a501.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
  listen 80 ;
  listen [::]:80 ;
    server_name j9a501.p.ssafy.io;
    return 404; # managed by Certbot
}
```

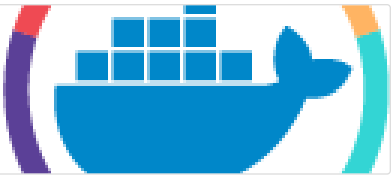
▼ docker

- 서버에 도커 설치

Install Docker Engine on Ubuntu

Jumpstart your client-side server applications with Docker Engine on Ubuntu. This guide details prerequisites and multiple methods to install.

https://docs.docker.com/engine/install/ubuntu/



▼ jenkins

- docker-out-of-docker 방식으로 Jenkins Image 내부 docker에서 host의 docker socket 접근

```
docker run -d \
-v jenkins_home:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \ # docker socket 연결
-p 8083:8080 -p 50000:50000 --restart=on-failure \ # 8083으로 포트 변경
--user=jenkins --group-add 998 \
--name=jenkins jenkins/jenkins:latest
```

- jenkins container 내에 docker 설치

```
docker exec -it -u root {JenkinsContainerName} /bin/bash # 젠킨스 컨테이너 내의
cat /etc/os-release # OS 버전 확인 후 docker 공식문서 참고하여 설치
```

▼ jenkinsfile

- 아래 백엔드의 application-secret.yml, secrets.json, .env 파일을 jenkins container의 docker volume인 아래 경로에 먼저 위치시킬 필요

```
/var/lib/docker/volumes/jenkins_home/_data/secrets
```

이후 jenkinsfile을 통해 서버 내의 secret 파일을 복사하여 빌드 진행

```
pipeline {
  agent any
  stages {
    stage("Clone") {
      steps {
        git branch: 'develop',
          credentialsId: 'hunn2023',
          url: 'https://lab.ssafy.com/s09-blockchain-contract-sub2/S09P22A501.git'
      }
    }
    stage("Build Backend") {
      steps {
        dir ("./backend") {
          // stop running containers and remove images
          sh "docker stop backend-server || true"
```

```
          sh "docker rm backend-server || true"
          sh "docker stop backend-kubo || true"
          sh "docker rm backend-kubo || true"
          sh "docker stop backend-redis || true"
          sh "docker rm backend-redis || true"

          // copy application-secret to cloned repo
          sh "rm ./src/main/resources/application-secret.yml || true"
          sh "cp /var/jenkins_home/secrets/application-secret.yml ./src/main/resources/"

          // copy .env file for docker compose to cloned repo
          sh "rm ./.env || true"
          sh "cp /var/jenkins_home/secrets/.env ./"

          // copy django env file
          sh "rm ../recommend/secrets.json || true"
          sh "cp /var/jenkins_home/secrets/secrets.json ../recommend/"

          sh "docker compose up --build -d"
        }
      }
    }
    stage("Build Frontend") {
      steps {
        dir ("./frontend") {
          // stop running containers and remove images
          sh "docker stop frontend || true"
          sh "docker rm frontend || true"

          sh "docker build -t frontend ."
          sh "docker run -p 3000:3000 -d --name frontend frontend"
        }
      }
    }
  }
  post {
    failure {
      script {
        def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
        mattermostSend (color: 'danger',
        message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
        endpoint: 'https://meeting.ssafy.com/hooks/3qde94cp8trebmodqhoizjd67w',
        channel: 'a501bot'
        )

      }
    }
  }
}
```

블록체인

이더리움 테스트넷(sepolia) 노드 실행(j9a501a.p.ssafy.io)

▼ clef (이더리움 지갑)

```
sudo clef --nousb --keystore test/keystore --configdir test/clef --chainid 11155111
# chainid 11155111 == sepolia network
```

▼ geth (이더리움 클라이언트)

```
sudo geth --sepolia --datadir test \
--authrpc.addr localhost \
--authrpc.port 8551 \
--authrpc.vhosts localhost \
--http.vhosts=* \
--authrpc.jwtsecret test/jwtsecret \
--http --http.api eth,net --signer=test/clef/clef.ipc --http
```

▼ lighthouse (이더리움 PoS 클라이언트)

```
sudo ./lighthouse bn --network sepolia \
--execution-endpoint http://localhost:8551 \
--execution-jwt test/jwtsecret \
--checkpoint-sync-url https://beaconstate-sepolia.chainsafe.io \
--disable-deposit-contract-sync
```

프론트엔드

Dockerfile을 통해 run 한 뒤, nginx에서 reverse-proxy 이용하여 container에 연결하여 Server Side Rendering 구현

▼ .env

```
# .env.development
NEXT_PUBLIC_DOMAIN = http://localhost:8000/
# .env.production
NEXT_PUBLIC_DOMAIN = https://j9a501.p.ssafy.io/
```

▼ package.json

```
{
  "name": "start-storybook-v7",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "storybook": "storybook dev -p 6006",
    "build-storybook": "storybook build"
  },
  "dependencies": {
    "@ant-design/icons": "^5.2.6",
    "@ethereumjs/util": "^9.0.0",
    "@noble/hashes": "^1.3.2",
    "@noble/secp256k1": "^2.0.0",
    "@types/crypto-js": "^4.1.2",
    "@types/node": "20.5.7",
    "@types/react": "^18.2.21",
    "@types/react-chartjs-2": "^2.5.7",
    "@types/react-dom": "18.2.7",
    "antd": "^5.8.5",
    "axios": "^1.5.0",
    "chart.js": "^4.4.0",
    "crypto-js": "^4.1.1",
    "epubjs": "^0.3.93",
    "eslint": "8.48.0",
    "eslint-config-next": "13.4.19",
    "next": "13.4.19",
    "react": "18.2.0",
    "react-chartjs-2": "^5.2.0",
    "react-cookie": "^6.1.1",
    "react-dom": "18.2.0",
    "recoil": "^0.7.7",
    "recoil-persist": "^5.1.0",
    "styled-components": "^6.0.7",
    "styled-reset": "^4.5.1",
    "typescript": "5.2.2"
  },
  "devDependencies": {
    "@storybook/addon-essentials": "^7.4.0",
    "@storybook/addon-interactions": "^7.4.0",
    "@storybook/addon-links": "^7.4.0",
    "@storybook/addon-mdx-gfm": "^7.4.0",
    "@storybook/addon-onboarding": "^1.0.8",
    "@storybook/addons": "^7.4.1",
    "@storybook/blocks": "^7.4.0",
    "@storybook/nextjs": "^7.4.0",
```

```
    "@storybook/react": "^7.4.0",
    "@storybook/testing-library": "^0.2.0",
    "@types/styled-components": "^5.1.26",
    "eslint-plugin-node": "^11.1.0",
    "eslint-plugin-storybook": "^0.6.13",
    "storybook": "^7.4.0"
  }
}
```

▼ Dockerfile

```
FROM node:18-alpine AS base

# Install dependencies only when needed
FROM base AS deps
# Check https://github.com/nodejs/docker-node/tree/b4117f9333da4138b03a546ec926ef50a31506c3#nodealpine to understand why libc6-compat mi
RUN apk add --no-cache libc6-compat
WORKDIR /app

# Install dependencies based on the preferred package manager
COPY package.json yarn.lock* package-lock.json* pnpm-lock.yaml* ./
RUN \
  if [ -f yarn.lock ]; then yarn --frozen-lockfile; \
  elif [ -f package-lock.json ]; then npm ci; \
  elif [ -f pnpm-lock.yaml ]; then yarn global add pnpm && pnpm i --frozen-lockfile; \
  else echo "Lockfile not found." && exit 1; \
  fi


# Rebuild the source code only when needed
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .

# Next.js collects completely anonymous telemetry data about general usage.
# Learn more here: https://nextjs.org/telemetry
# Uncomment the following line in case you want to disable telemetry during the build.
# ENV NEXT_TELEMETRY_DISABLED 1

# RUN yarn build

# If using npm comment out above and use below instead
RUN npm run build

# Production image, copy all the files and run next
FROM base AS runner
WORKDIR /app

ENV NODE_ENV production
# Uncomment the following line in case you want to disable telemetry during runtime.
# ENV NEXT_TELEMETRY_DISABLED 1

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public

# Set the correct permission for prerender cache
RUN mkdir .next
RUN chown nextjs:nodejs .next

# Automatically leverage output traces to reduce image size
# https://nextjs.org/docs/advanced-features/output-file-tracing
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

USER nextjs

EXPOSE 3000

ENV PORT 3000
# set hostname to localhost
ENV HOSTNAME "0.0.0.0"
```

```
CMD ["node", "server.js"]
```

백엔드

docker-compose를 통해 spring boot, django, redis, kubo(ipfs client)를 함께 구성

▼ docker-compose

```
services:
  backend_server:
    container_name: backend_server
    depends_on:
      - backend_kubo
      - backend_redis
      - recommend
    restart: on-failure
    build:
      context: ./
      dockerfile: dockerfile
    ports:
      - "127.0.0.1:8000:8000"
    environment:
      KUBO_TCP_HOST: backend_kubo:4001
      KUBO_RPC_HOST: backend_kubo:5001
      KUBO_GATEWAY_HOST: backend_kubo:8080
      REDIS_HOST: backend_redis
      REDIS_PORT: 6379
      REDIS_PASSWORD: ${REDIS_PASSWORD}
      DJANGO_URL: http://recommend:8888/api1/string-list/
    volumes:
      - book_volume:/back/books

  backend_kubo:
    container_name: backend_kubo
    image: ipfs/kubo
    volumes:
      - ipfs_data:/data/ipfs
      - ipfs_staging:/export
      - book_volume:/data/books
    ports:
      - "0.0.0.0:4001:4001"  # TCP
      - "127.0.0.1:5001:5001"  # RPC
      - "0.0.0.0:8080:8080"  # Gateway

  backend_redis:
    container_name: backend_redis
    image: redis
    ports:
      - "0.0.0.0:6379:6379"
    command: redis-server --requirepass ${REDIS_PASSWORD}
    volumes:
      - redis_config:/usr/local/etc/redis

  recommend:
    container_name: recommend
    build:
      context: ../recommend
      dockerfile: dockerfile
    ports:
      - "127.0.0.1:8888:8888"

volumes:
  book_volume:
  ipfs_data:
  ipfs_staging:
  redis_config:
```

▼ .env

```
REDIS_PASSWORD={REDIS 비밀번호}
```

▼ Spring boot

▼ dockerfile

```
FROM amazoncorretto:17-alpine
WORKDIR /back
COPY . .
RUN chmod +x gradlew && mkdir books && ./gradlew clean build
ENTRYPOINT ["java", "-jar", "build/libs/bangle-0.0.1-SNAPSHOT.jar"]
```

▼ application-secret.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/bangle_db?serverTimezone=UTC&characterEncoding=UTF-8&collation=utf8mb4_bin
    username: {유저아이디}
    password: {유저비밀번호}

  jpa:
    open-in-view: false
    hibernate:
      ddl-auto: update
      show_sql: true
      format_sql: true
      use_sql_comments: true
      dialect: org.hibernate.dialect.MySQL5InnoDBDialect

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: {발급 받은 client-id}
            client-secret: {발급 받은 client-secret}
            scope: {필요한 권한}
            redirect-uri: {redirect 주소}
            authorization-grant-type: authorization_code
            client-name: kakao
            client-authentication-method: client_secret_post

        provider:
          kakao:
            authorization-uri: {설정에 따름}
            token-uri: {토큰 uri}
            user-info-uri: {user-info-uri}
            user-name-attribute: id

jwt:
  secret:  {암호화 Key}
  access-expiration: 1800000 # 30분
  refresh-expiration: 1209600000 # 14일

cloud:
  aws:
    credentials:
      access-key: {access-key}
      secret-key: {secret-key}
    region:
      static: ap-northeast-2
    s3:
      bucket: {버킷이름}
    stack:
      auto: false
    prefix:
      url: {prefix-url}

wallet:
  private: {private}
  public: {public}
```

```
  address: {address}

redis:
  host: {호스트이름}
  port: {포트번호}
  password: {비밀번호}

geth:
  rpc:
    url: {url}

api-key:
  chat-gpt: {api-key}

django:
  url: {django-url}
```

▼ build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.1.3'
    id 'io.spring.dependency-management' version '1.1.3'
    id 'org.asciidoctor.jvm.convert' version '3.3.2'
}

group = 'com'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    asciidoctorExt
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

ext {
    set('snippetsDir', file("build/generated-snippets"))
}

dependencies {
    implementation 'org.web3j:core:4.10.0'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation "com.querydsl:querydsl-jpa:5.0.0:jakarta"
    implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
    implementation 'org.springframework.boot:spring-boot-starter-batch'

    annotationProcessor 'org.projectlombok:lombok'
    annotationProcessor "com.querydsl:querydsl-apt:5.0.0:jakarta"
    annotationProcessor "jakarta.annotation:jakarta.annotation-api"
    annotationProcessor "jakarta.persistence:jakarta.persistence-api:3.1.0"

    asciidoctorExt 'org.springframework.restdocs:spring-restdocs-asciidoctor'

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.mysql:mysql-connector-j'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.restdocs:spring-restdocs-mockmvc'
    testImplementation 'org.springframework.security:spring-security-test'

    implementation("com.auth0:java-jwt:3.10.3")
    implementation 'com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.13.1'
```

```
    implementation group: 'org.bouncycastle', name: 'bcprov-jdk15on', version: '1.70'
}

tasks.named('test') {
    outputs.dir snippetsDir
    useJUnitPlatform()
}

tasks.named('bootJar'){
    dependsOn asciidoctor
    from ("${asciidoctor.outputDir}/html5"){
        into 'static/docs'
    }
}

tasks.named('asciidoctor') {
    inputs.dir snippetsDir
    configurations 'asciidoctorExt'
    dependsOn test
}

// Querydsl Q Class 생성 위치
def querydslDir = '/src/main/generated/'

// Querydsl Q Class 생성 위치 지정
tasks.withType(JavaCompile) {
    options.getGeneratedSourceOutputDirectory().set(file(querydslDir))
}

// java source set 에 Querydsl Q Class 위치 추가
sourceSets {
    main.java.srcDirs += [ querydslDir ]
}

// gradle clean 시, Q Class 디렉토리까지 삭제하도록 설정
clean {
    delete file(querydslDir)

}
```

▼ Django

▼ dockerfile

```
FROM python:3.9

RUN apt-get update \
  && apt-get install -y --no-install-recommends \
    postgresql-client \
  && rm -rf /var/lib/apt/lists/*

WORKDIR /usr/src/app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . .

EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8888", "--noreload"]
```

▼ requirements.txt

```
asgiref==3.7.2
Django==4.2.5
djangorestframework==3.14.0
joblib==1.3.2
mysqlclient==2.2.0
numpy==1.26.0
pandas==2.1.1
python-dateutil==2.8.2
pytz==2023.3.post1
scikit-learn==1.3.1
scipy==1.11.2
```

```
six==1.16.0
sqlparse==0.4.4
threadpoolctl==3.2.0
typing_extensions==4.8.0
tzdata==2023.3
```

▼ secrets.json

recommend directory 내 manage.py와 같은 경로에 위치

```
{
  "DB_NAME" : {DB 이름},
  "DB_USER" : {DB USERNAME},
  "DB_PASS" : {DB PASSWORD},
  "DB_HOST" : {DB HOSTNAME},
  "DB_PORT" : {DB PORT}
}
```