🔥

포팅매뉴얼

Owner	h hun23
Tags	

▼ 목차

CI/CD

▼ nginx

- nginx version: nginx/1.18.0 (Ubuntu)

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#


# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#

server {
  # SSL configuration
  #
  # listen 443 ssl default_server;
  # listen [::]:443 ssl default_server;
  #
  # Note: You should disable gzip for SSL traffic.
  # See: https://bugs.debian.org/773332
  #
  # Read up on ssl_ciphers to ensure a secure configuration.
  # See: https://bugs.debian.org/765782

  # Self signed certs generated by the ssl-cert package
  # Don't use them in a production server!
  #
  # include snippets/snakeoil.conf;

  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 k9a504.p.ssafy.io; # managed by Certbot


  location /api {
    proxy_pass  http://localhost:8000/api;
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    # try_files $uri $uri/ =404;
  }

  location /docs {
    proxy_pass  http://localhost:8000/docs;
  }

  location / {
    proxy_pass  http://localhost:3000;
  }

  location /gpt {
    proxy_pass  http://localhost:5000/gpt;
  }

  location /__/auth {
     proxy_pass https://a504-qookie.firebaseapp.com;
  }


  # pass PHP scripts to FastCGI server
  #
  #location ~ \.php$ {
  # include snippets/fastcgi-php.conf;
  #
  # # With php-fpm (or other unix sockets):
  # fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
  # # With php-cgi (or other tcp sockets):
  # fastcgi_pass 127.0.0.1:9000;
  #}

  # deny access to .htaccess files, if Apache's document root
  # concurs with nginx's one
  #
  #location ~ /\.ht {
  # deny all;
  #}


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/k9a504.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/k9a504.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 {
  listen 80;
  listen [::]:80;
    if ($host = k9a504.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    server_name k9a504.p.ssafy.io;
    return 404; # managed by Certbot
}
```

▼ docker

- Docker version 24.0.6, build ed223bc

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 Docker Engine on Ubuntu.

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

▼ jenkins

- **Version 2.414.3**
- jenkinsfile 통해 파이프라인 구축 및 프론트엔드, 백엔드 빌드
- 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 8080:8080 -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

```
pipeline {
  agent any
  stages {
    stage("Clone") {
      steps {
        git branch: 'test',
          credentialsId: 'jenkins_hunn000',
          url: 'https://lab.ssafy.com/s09-final/S09P31A504.git'
      }
    }
    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 -v /front_nginx_log:/var/log/nginx -p 127.0.0.1:3000:80 -d --name frontend frontend"
        }
      }
    }
    stage("Copy Secrets & Stop Containers") {
      steps {
        dir ("./") {
          sh "docker stop backend_server || true"
          sh "docker stop backend_python || true"
          sh "docker stop backend_redis || true"
          sh "docker stop backend_rabbitmq || true"

          sh "docker rm backend_server || true"
          sh "docker rm backend_python || true"
          sh "docker rm backend_redis || true"
          sh "docker rm backend_rabbitmq || true"

          sh "docker ps -a"
        }

        dir ("./backend") {
          // 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 a504-qookie-firebase-adminsdk-key.json to cloned repo
          sh "rm ./src/main/resources/a504-qookie-firebase-adminsdk-key.json || true"
          sh "cp /var/jenkins_home/secrets/a504-qookie-firebase-adminsdk-key.json ./src/main/resources/"
        }

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

    stage("Build Backend") {
      steps {
        dir ("./") {
          sh "docker compose up --build -d"
        }
      }
    }
  }
}
```

프론트엔드

특이사항

- dockerfile 사용하여 build 후 nginx로 배포

환경설정

- .env, package.json, dockerfile, default.conf(nginx 설정파일)는 소스코드에 포함

백엔드

특이사항

- docker-compose를 통해 각 container dockerfile을 실행시켜 빌드

환경설정

▼ .env (jenkins docker container 내의 jenkins_home/secrets/ 에 위치)

```
REDIS_PASSWORD=ss501ss501
RABBITMQ_USER=newjeans
RABBITMQ_PASSWORD=ss501ss501
GPT_API_KEY={GPT_API_KEY}
```

▼ application-secrets.yml (jenkins docker container 내의 jenkins_home/secrets/ 에 위치)

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://database-1.c4jdluaykrgh.ap-northeast-2.rds.amazonaws.com:3306/qookie_db
    username: admin
    password: ss501ss501

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

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

redis:
  host: k9a504.p.ssafy.io
  port: 6379
  password: ss501ss501

# s3 config
cloud:
  aws:
    credentials:
      access-key: {S3_ACCESS_KEY}
      secret-key: {S3_SECRET_KEY}
    region:
      static: ap-northeast-2
    s3:
      bucket: bangle
    stack:
      auto: false
    prefix:
      url: https://bangle.s3.ap-northeast-2.amazonaws.com/
```

Spring Boot

- Java 17, Spring Boot 3.1.4 사용
- dockerfile 및 build.gradle은 소스코드에 포함

Python

- Python 3.8 사용
- dockerfile 및 requirements.txt는 소스코드에 포함

외부 서비스 정보

- OpenAI GPT API
- Amazon S3, RDS

DB 덤프

여기에 DB 덤프

시연 시나리오

여기에 시연 시나리오