배포 설정

💡 본 문서에서는 프로젝트를 수동으로 배포하고 빌드에 필요한 설정 파일을 설명합니다.

목차

Nginx

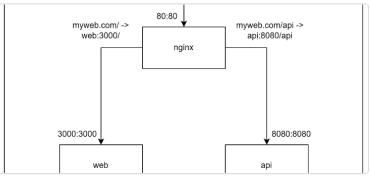
참고 사이트

nginx reverse proxy 적용하기

[Nginx] 리버스 프록시(Reverse Proxy) 개념 및 사용법

1. 개요 리버스 프록시란? 클라이언트 요청을 대신 받아 내부 서버로 전달해주는 것을 리버스 프록시(Reverse Proxy) 라고 합니다. 저도 사실 프록시라는 개념이 낯설었는데요, 일단 프록시라는 개념부터 확인해야 합니다. 프록시란 대리라는 의미로, 정보를 대신 전달해주는 주체라고 생각하면 되는데, 만약 이 프록시 없이 웹 서버를 운영한다고

https://narup.tistory.com/238

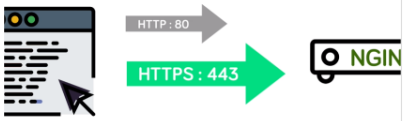


nginx ssl 적용하기

[Nginx] SSL 설정(HTTPS 적용)

1. 개요 기존에 웹 사이트를 HTTP로 운영하고 있다가, 사용자의 정보같은 민감한 정보를 사용하게 될 경우에는 SSL 인증서를 사용한 보안처리를 해야합니다. 웹서버에 SSL 인증서를 사용해 웹사이트를 HTTPS로 열 수 있게끔 Nginx 프록시 서버에 SSL 인증서를 적용하는 방법을 확인해보겠습니다. 2. 준비물 도메인, SSL 인증서(chain key, private

https://narup.tistory.com/240



nginx 설치 및 설정 파일 생성하기

```
# ec2에 nginx 설치
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx

# 설정 파일 생성
sudo vim /etc/nginx/conf.d/default.conf
```

default.conf

```
limit_req_zone $binary_remote_addr zone=api_rate_limit:10m rate=10r/s;

upstream frontend {
    server localhost:3000;
}
upstream backend {
    server localhost:8000;
```

```
}
server {
    listen 80;
    server_name i9a609.p.ssafy.io;

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

server {
    listen 443 ssl http2;
    server_name i9a609.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i9a609.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i9a609.p.ssafy.io/privkey.pem;

    location /api {
        proxy_pass http://backend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        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;
        limit_req zone=api_rate_limit burst=10 nodelay;
    }

    location / {
        proxy_pass http://frontend;
        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;
    }
}
```

Docker

Docker 설치하기

```
# docker에 필요한 패키지 다운로드
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

# GPG Key 등록하기
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add

# docker repository 등록
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

# docker 설치하기
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io

# docker 그룹에 사용자 추가하기
sudo usermod -aG docker ubuntu

# docker 재시작
sudo service docker restart
```

Docker 네트워크 생성하기

```
docker network create ssafy-net
```

Docker 이미지 pull

```
docker pull jlal1226/secret-repo:DreamGream
docker pull jlal1226/secret-repo:ImageRequestServer
docker pull jlal1226/secret-repo:PromptApi
```

```
docker pull mysql
docker pull rabbitmq
docker pull redis:7.0.12
```

Docker 컨테이너로 실행하기

DreamGream Container 실행 스크립트

```
#!/bin/bash

SPRING_PROFILE="prod"

# 환경 변수
export S3_BUCKET_NAME=<bucket name>
export S3_REGION=<s3 region>
export S3_ACCESS_KEY=<access key>
export S3_SECRET_KEY=<secret key>

# Docker container 실행하기
docker run -d --name dreamgream -e SPRING_PROFILES_ACTIVE=${SPRING_PROFILE} -e S3_BUCKET_NAME -e S3_REGION -e S3_ACCESS_KEY -e S3_SECR
```

ImageRequestServer Container 실행 스크립트

```
#!/bin/bash

SPRING_PROFILE="prod"

# 환경 변수
export S3_BUCKET_NAME=<bucket name>
export S3_REGION=<s3 region>
export S3_ACCESS_KEY=<access key>
export S3_SECRET_KEY=<secret key>

# Docker container 실행하기
docker run -d --name image-requset-server -e SPRING_PROFILES_ACTIVE=${SPRING_PROFILE}  -e S3_BUCKET_NAME -e S3_REGION -e S3_ACCESS_KEY
```