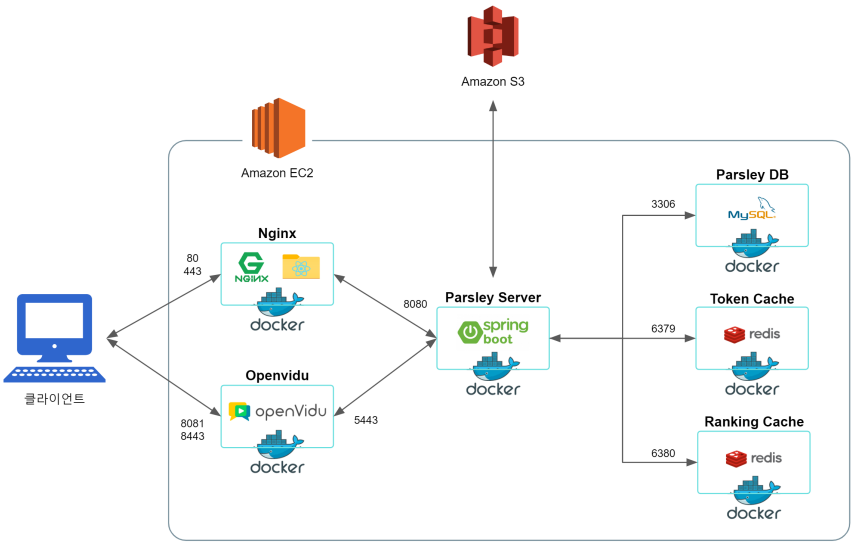
EC2에 배포 및 HTTPS 적용

Tags | Server

목차

PARSLEY 아키텍처



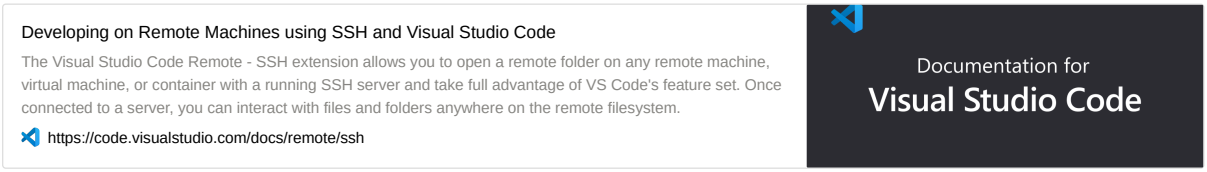
- docker-compose를 작성해서, Nginx/Server/DB/Cache 컨테이너를 한 번에 실행하도록 함
    - Nginx 컨테이너 구축 시, React App을 build하여 Nginx의 static 폴더에 배치
    - JDK 기반 Server 컨테이너 구축 시, Spring Boot를 build한 후 실행
- HTTPS 적용
    - http 80번 포트로 들어오면, https로 redirect 시켜줌
    - `/` 로 들어오는 요청은 static 파일을 응답
    - `/api` 로 들어오는 요청은 내부적으로 `/` 로 rewrite 시켜주고, 8080번 포트로 전달

이후 server에서 처리한 응답을 반환

1. 기본 설정

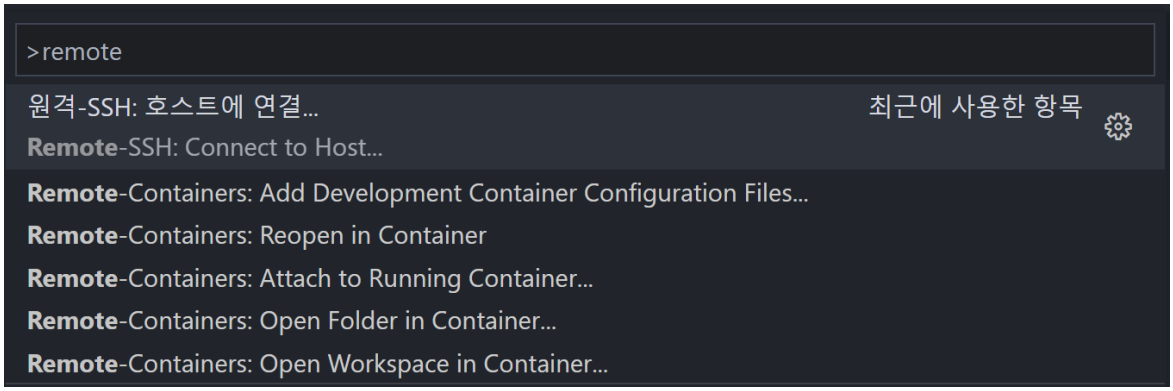
VS code에서 SSH 연결하는 방법

MobaXterm과 같은 SSH Client 툴보다 파일 관리 측면에서 편리함!

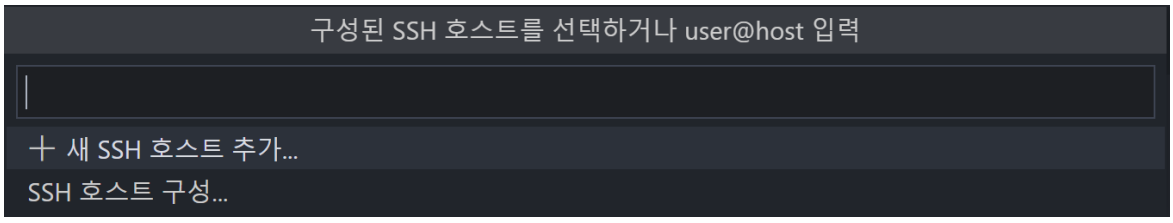


공식 문서

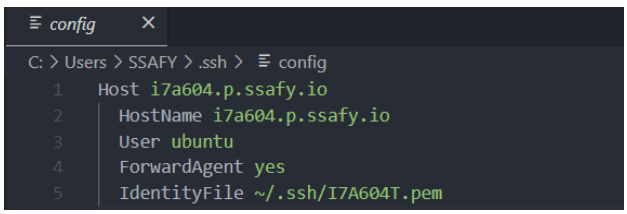
1. Remote-SSH 확장 설치
2. Ctrl + Shift + P 누르고 `Remote-SSH: Connect to Host` 클릭



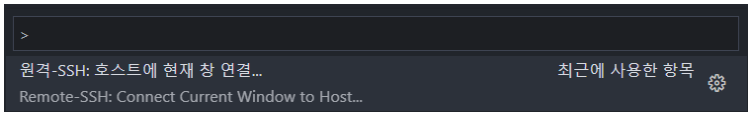
3. 새 SSH 호스트 추가

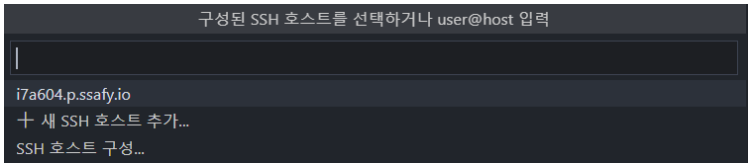


4. SSH 연결 명령 입력: `ssh ubuntu@i7a604.p.ssafy.io`
5. 내 PC의 config 파일 선택 후 `구성 열기`해서 확인



6. 설정한 호스트와 연결





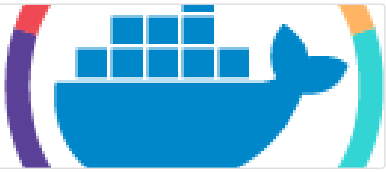
Linux 선택 > 지문 계속 > Open Folder > /home/ubuntu

EC2에 Docker 설치하기

Install Docker Compose CLI plugin

Estimated reading time: 6 minutes On this page you can find instructions on how to install the Compose plugin for Docker CLI on Linux and Windows Server operating systems. Note that installing Docker Compose as a plugin requires Docker CLI. In this section, you can find various methods for installing Compose on Linux.

https://docs.docker.com/compose/install/compose-plugin/#install-using-the-repository



도커 공식 문서 참고

Jenkins를 도입하지 않을 예정이라,

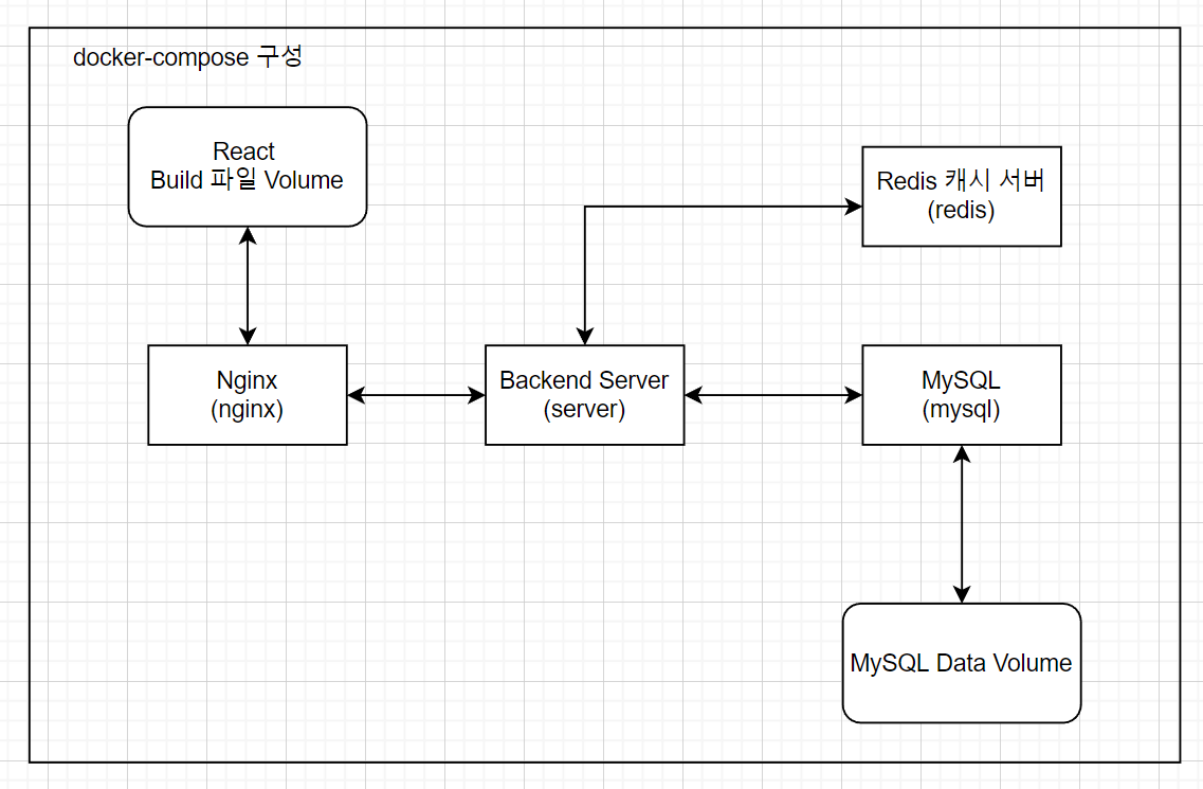
git clone을 제외한 build 작업이라도 자동화하고자

dockerfile + docker compose 를 작성함

💡 git도 설치하려 했으나, 이미 설치되어 있어서 설치하지 않음

2. Docker compose 작성하기

들어가기 전, 생각한 docker compose 구성



초기 구상

Dockerfile (1) : React App Build 후, Nginx 에 옮기기

```
# Build Stage 1. React App
FROM node:alpine as builder
WORKDIR '/app'
COPY ./parsley/frontend/package.json .
RUN npm install
COPY ./parsley/frontend ./
RUN npm run build

# Build Stage 2. Nginx
FROM nginx:latest
COPY --from=builder /app/build /usr/share/nginx/html
```

Multi Stage Build를 통해 이미지를 작게 만들고자 함

먼저 node 이미지를 가져와서, React App을 빌드하였고,

이후 만든 nginx 컨테이너에 빌드 결과물인 정적 파일을 옮긴 것

Dockerfile (2) : Spring Boot Build 하고 실행하기

```
# Build Stage 1. Spring Boot Build
FROM openjdk:8-jdk-alpine as builder
COPY ./parsley/backend .
COPY application.properties ./src/main/resources/
COPY keystore.p12 ./src/main/resources/ssl/
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

# Build Stage 2. 패키징된 .jar 실행
FROM openjdk:8-jdk-alpine
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

application.properties와 keystore.p12는 보안상 git에 올리지 않고

EC2에만 올려놓고, 프로젝트 빌드 전, COPY를 통해 classpath 아래에 넣었다.

Certbot + Nginx Container 로 HTTPS 적용

Nginx and Let's Encrypt with Docker in Less Than 5 Minutes

The other day, I wanted to quickly launch an nginx server with Let's Encrypt certificates. I expected the task to be easy and straightforward. Turns out: I was wrong, it took a significant amount of time and it's quite a bit more complicated. Of course, in the grand scheme of things, it is pretty straightforward.

https://pentacent.medium.com/nginx-and-lets-encrypt-with-docker-in-less-than-5-minutes-b4b8a60d3a71

HTTPS 인증서 검증 과정

1. Let's Encrpyt가 도메인이 요청하면, 도메인 검증을 수행
2. Certbot이 검증 결과에 대한 응답 데이터를 제공
3. Nginx가 해당 응답 데이터를 전달(serve)해주는 것

보안등급 A+를 받을수 있는 NGINX SSL 설정 - 익스트림 매뉴얼

이전글인 ' NGINX로 SSL을 지원하는 리버스 프록시 설정하기' 를 통해 SSL을 지원하는 리버스 프록시를 설정해 보았는데요. 이 세팅만으로는 Qualys SSL LABS 에서 제공하는 SSL Server Test 에서 A+등급을 받을 수가 없습니다. 참고로 SSL 인증서와 private key만 연결했을경우 F등급, 위 포스팅대로 설정했을 경우 B등급을 받을수 있는데요. 이번엔

https://extrememanual.net/2059

2.5 Nginx & Docker-compose 코드

Nginx 설정

```
# 80번 포트로 들어오는 경우
server {
  listen 80;
  server_name i7a604.p.ssafy.io;

  location /.well-known/acme-challenge/ { # certbot
    root /var/www/certbot;
  }

  location / { # 아래 URL로 redirect
    return 301 https://i7a604.p.ssafy.io$request_uri;
  }
}

# 443번 포트로 들어오는 경우
server {
  listen 443 ssl;
  server_name i7a604.p.ssafy.io;

  ssl_certificate /etc/letsencrypt/live/i7a604.p.ssafy.io/fullchain.pem; #certbot
  ssl_certificate_key /etc/letsencrypt/live/i7a604.p.ssafy.io/privkey.pem; #certbot

  # 보안을 위한 NGINX 설정
  include /etc/letsencrypt/options-ssl-nginx.conf; #certbot
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; #certbot

  location / { # /로 들어오는 경우
    root /usr/share/nginx/html;
    index index.html;
    try_files $uri $uri/ /index.html;
  }

  location /api { # /api로 들어오는 경우
    rewrite ^/api(/.*)$ $1 break; # /api -> / 로 rewrite
    proxy_pass https://i7a604.p.ssafy.io:8080; # 8080번 포트로 pass
    proxy_redirect off;

    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-compose.yml

```
version: "3.7"
services:
    nginx:
        container_name: parsley-nginx
        build:
            context: .
        depends_on:
            - server
        ports:
            - 80:80
            - 443:443
        volumes:
            - ./data/nginx/conf:/etc/nginx/conf.d
            - ./data/certbot/conf:/etc/letsencrypt
            - ./data/certbot/www:/var/www/certbot
        restart: always
    certbot:
        container_name: parsley-certbot
        image: certbot/certbot
        volumes:
            - ./data/certbot/conf:/etc/letsencrypt
            - ./data/certbot/www:/var/www/certbot
    mysql-db:
        container_name: parsley-mysql
        image: mysql:5.7
        ports:
            - 3306:3306
        volumes:
            - ./data/mysql:/var/lib/mysql
        environment:
            - MYSQL_ROOT_HOST=%
            - MYSQL_ROOT_PASSWORD=Tkvl7rldbryqhdlsmsdlswjddlwl
            - MYSQL_DATABASE=parsley
            - MYSQL_USER=ugyo-boi
            - MYSQL_PASSWORD=vktmfflrkqhwkrh
        command:
            - --character-set-server=utf8
            - --collation-server=utf8_general_ci
        restart: always
    redis:
        container_name: parsley-redis
        image: redis
        ports:
            - "6379:6379"
        volumes:
            - ./data/redis:/data
        restart: on-failure
    redis-rank:
        container_name: parsley-redis-rank
        image: redis
        ports:
            - "6380:6379"
        volumes:
            - ./data/redis-rank:/data
        restart: on-failure
    server:
        container_name: parsley-server
        depends_on:
            - mysql-db
            - redis
        build:
            context: .
            dockerfile: ./parsley/backend/Dockerfile
        ports:
            - 8080:8080
        restart: on-failure
```

3. 발생했던 오류 모음

ssh 연결 시, `.pem` Permission 문제

```
❯ ssh -i I7A604T.pem ubuntu@i7a604.p.ssafy.io
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0777 for 'I7A604T.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "I7A604T.pem": bad permissions
ubuntu@i7a604.p.ssafy.io: Permission denied (publickey).
❯ chmod 600 I7A604T.pem
❯ ls -al
total 12
drwxr-xr-x 2 hialynn hialynn 4096 Jul 31 19:14 .
drwxr-xr-x 3 hialynn hialynn 4096 Jul 31 19:15 ..
-rw------- 1 hialynn hialynn 1700 Jul 31 11:38 I7A604T.pem
```

permission 오류 발생

```
❯ ssh -i I7A604T.pem ubuntu@i7a604.p.ssafy.io
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Jul 31 10:17:25 UTC 2022

  System load:  0.0               Processes:             136
  Usage of /:   0.7% of 310.15GB  Users logged in:       0
  Memory usage: 2%                IPv4 address for eth0: 172.26.11.35
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

142 updates can be installed immediately.
9 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


3 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

*** System restart required ***
Last login: Wed Jul 27 00:05:40 2022 from 211.192.252.59
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-26-11-35:~$ 
```

ssh 연결 성공

→ Unix 환경에서는 이와 같이 해결 가능하지만, Windows에서는 다른 방법으로 해결해야 함

권한 문제: docker compose build 시, `Error checking context: ...`

docker build Error checking context: 'can't stat '\\?\C:\Users\username\AppData\Local\Application Data''

When you run the docker build command, the Docker client gathers all files that need to be sent to the Docker daemon, so it can build the image. This 'package' of files is called the context. What files and directories are added to the context?

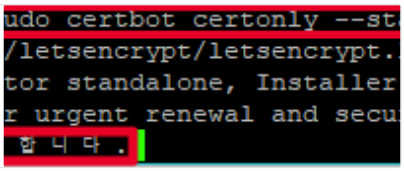
https://stackoverflow.com/questions/41286028/docker-build-error-checking-context-cant-stat-c-users-username-appdata

Spring Boot에 SSL 적용 안한 문제: 밑도 끝도 없이 나오는 `502 Bad Gateway`

Spring Boot에 Let's Encrypt SSL 적용하기

아파치나 NginX 같은 웹서버 없이 Spring Boot으로 만든 웹 어플리케이션에 무료 SSL 중 하나인 Let's Encrypt을 적용하게 되었다. 90일동안 사용가능하며 다시 갱신을 시켜줘야하나? 싶었지만 간단한 설정으로 자동갱신까지 가능하니 참 괜찮은 것 같다. 그리고 Let's Encrypt SSL 인증서 발급 방법은 여러가지가 있는데 그 중 standalone 방식을 택했다. 이

https://jiwontip.tistory.com/83



…