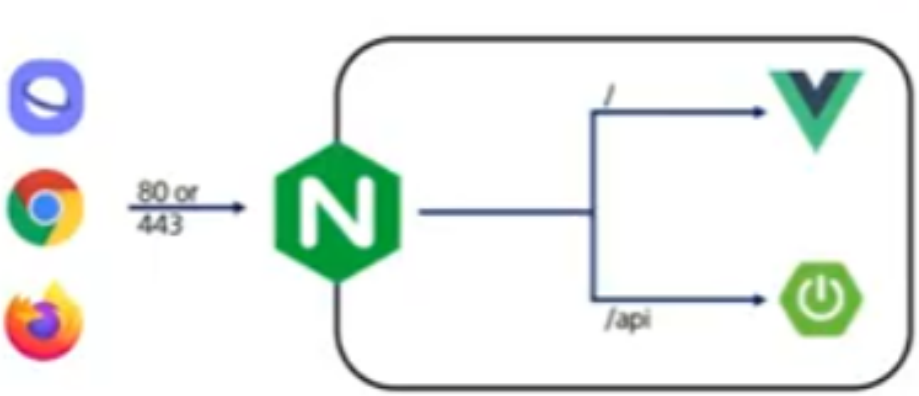
2021-08-10 - 공통 프로젝트 배포

✅ 목표

- AWS 배포를 위해 필요한 구조를 학습해본다
- 웹서버를 운영하기 위한 기술 스택들을 알아본다
- 자주하는 실수에 대해서 알아본다
- 배운 것들을 우리팀 서버 배포 시에 활용해본다

✅ NGINX

- High performance load balancer, web server, API gateway & reverse proxy
- 비동기 방식이기 때문에 매우 높은 성능
- 정적인 파일(주로 프론트엔드 파일들)을 서비스할 때 뛰어난 성능(vs 톰캣)
- load balancer나 API gateway 용도로도 사용 가능
- NGINX는 엄청 많이 쓰임



- HTTP, HTTPS는 80, 443 포트를 사용하는 것은 약속이다
- FE는 /, 백엔드는 /api로 분기해서 보내는 방식을 통해서 활용함
  - / 로 들어오는 요청은 프론트엔드의 라우터로
  - /api 로 들어오는 요청은 백엔드로 보낸다
    - Webserver로서의 역할
    - API Gateway로의 역할을 함!

```
server {

    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html/dist;

    index index.html index.htm;

    server_name _;

    location / {
      try_fiels $uri $uri/ /index.html;
    }

    location /api {

      proxy_pass http://localhost:8399/api/;

      proxy_redirct off;


      // ~~~

    }


}
```

- certbot을 하면 추가적 설정없이 TLS 설정 가능

FE나 BE를 Dockerize하는 경우에...

- NGINX를 한 번 더 거쳐가는 경우
  - 굉장히 미미한 성능 손실
  - 하지만, 동적으로 서버를 증설하는 데 편하게 쓸 수 있다는 장점
- 볼륨 마운트
  - 볼륨에서 read ⇒ NGINX가 한 번 더 거칠 필요는 없지만,
  - 동적으로 만드는 스크립트를 작성하는 것이 불편함

- AWS의 AutoScaling도 답이 아니었다
  - VM은 답이 아님

우리는 왜 도커를 쓰는가?

✅ 빠르게 필요한 서버를 증설할 수 있다

✅ 기존에는 VM을 증설하는 방식을 사용했음

- VM이 부팅되는 1분이면 서비스 전체가 중지되기에 충분한 시간

✅ 운영체제를 부팅해야 하는 기존의 방식보다 빠름

✅ 이미지를 만들어두면 찍어내기만 하면 되는 배포의 편의성(w/ k8s)

- Java 버전을 잘못 깔았어요. node가 이 버전이 아닌데??

어디까지 도커화 해야 할까?

✅ 프론트엔드 / 백엔드는 필수적

✅ 배포의 효율성 / 편의성을 생각해보자

✅ DB / Jenkins / nginx는 선택적

✅ DB를 이미지화해서 새로 배포할 일이 많이 있을까? ⇒ 옮긴다면 데이터는?

✅ 빌드 서버를 병렬적으로 추가 증설하는 경우는?

임의의 포트를 쓰면 안 되는 이유?

- 놀랍게도 멀티캠퍼스에는 투~~를 들어갈 수 없다
  - 커피 쿠폰이 스~~스 이유는?
- ISP(SKT,KT,LGU 등등)에 따라서 닫혀 있는 포트가 존재
- 어느 곳에서는 되고, 어느 곳에서는 안 되는 서비스라면?
- 고객은 포트가 막혔을 거라는 생각을 하지 못하고 그냥 이탈한다.

자동화된 배포는 시간을 겁나 줄여줌!

GitLab ⇒ Jenkins

✔️ 개발자가 gitlab의 특정 브랜치에 머지를 하면 이벤트가 트리거가 되어 jenkins에서 빌드를 시작한다

✔️ 빌드가 완료되면 도커 이미지가 제작되어 배포된다

✔️ 동일한 도커 이미지로 제작, 배포되기 때문에 동일성이 보장된다.

SSL → TLS

✔️ 회원 가입 시에 비밀번호 등의 개인 정보가 전송되고, 수시로 유출되어서는 안 되는 정보들이 오가기 때문에 암호화가 필요하다

✔️ 매번 데이터를 암호화해서 전송하기 어렵기 때문에 TLS(Transport Layer Security)를 사용한다

✔️ 이론적으로 TLS를 하면 안전함

CertBot

✔️ Https 확산을 위해서 시작된 비영리 프로젝트(Let`s encrypt)

✔️ 상용 프로그램을 제작할 때는 보통 신뢰할 수 있는 ROOT 인증서 발급자로부터 SSL 인증서를 구매해서 사용한다

✔️ SSAFY 프로젝트의 경우에는 CertBot을 이용해서 무료 인증서를 발급 받아서 사용하면 좋다

✔️ nginx나 백엔드 서버 모두에 설정이 필요하다

사용자 계정 만들기

✅ 각 프로그램들을 실행할 때는 프로그램에 맞는 권한을 가진 사용자 계정을 만들어서 실행한다

✅ ubuntu 계정이나 심지어 root 계정으로 실행하는 경우에는 해커의 공격 명령이 그 계정의 권한으로 실행되기 때문에 매우 위험

✅ 사용자 계정으로 실행하는 경우 해커의 공격을 받더라도 피해를 최소화할 수 있다.
