

# 빅데이터를 지탱하는 기술 week6

Docker 활용하기

2024.01.02

8기 엄소은

# 실전에서 도커 사용하기

- ML 모델도 구상하고, 백엔드를 만들었다.. 그 다음은 ?
- 로컬호스트에서 돌아가는 것을 확인했으면 배포를 해야 한다.
- 배포를 한다 = 다른 사람들도 쓸 수 있게 한다 = 다른 사람들이 접속할 수 있는 url 을 만든다.
- HOW ?

# 실전에서 도커 사용하기 (feat. AWS)

- AWS EC2 를 사용하면, 가상의 컴퓨터를 대여 받는다.
- 이 컴퓨터에 접속해서, 배포를 진행할 수 있다.
- (EC2 instance 만드는 법, public IPv4 address 할당받는 방법은 빅자기v1 - AWS 참고)

# EC2에서 내가 만든 프로젝트를 가져오는 방법?

- 생각을 해보자.. 가상의 컴퓨터로 내가 만든 프로젝트를 옮기는 방법은 뭐가 있을까?
- sol1 - github에 push한 다음, 이 레포를 EC2 instance에 pull 한다.
- sol2 - 도커 이미지를 빌드한 후, 이 이미지를 dockerhub에 push하고 EC2 instance에서 pull 한다.
- sol3 - 코드를 AWS S3에 저장하기
- sol4 - Infrastructure as a Code (e.g. AWS CodeDeploy, Terraform)
- 이것말고도 생각해 볼 수 있는 방법은 무수히 많다. 각각의 장단점은 무엇일까?

# sol1. Github clone 하는 방법

- **Pros**
  - Version Control – “Git” 을 사용하니까 !
  - Collaboration – “GitHub”니까 기본적으로 다른 개발자들이 코드를 수정해서 push 할 수 있다.
  - Lightweight – Docker image 보다 clone 하는 것이 가볍다 (왜냐면 실행 환경말고 “코드”만 옮겨가니까 )
- **Cons**
  - Dependency Management – 환경을 내가 따로 설정해주어야 한다. (local 에서 처음 레포 받으면 pip install 실행 하듯이..)
  - Consistency – 기본적으로 환경이 달라진다. 따라서 똑같은 결과가 나오지 않을 수도 있다.

# sol2. Docker Image pull 하는 방법

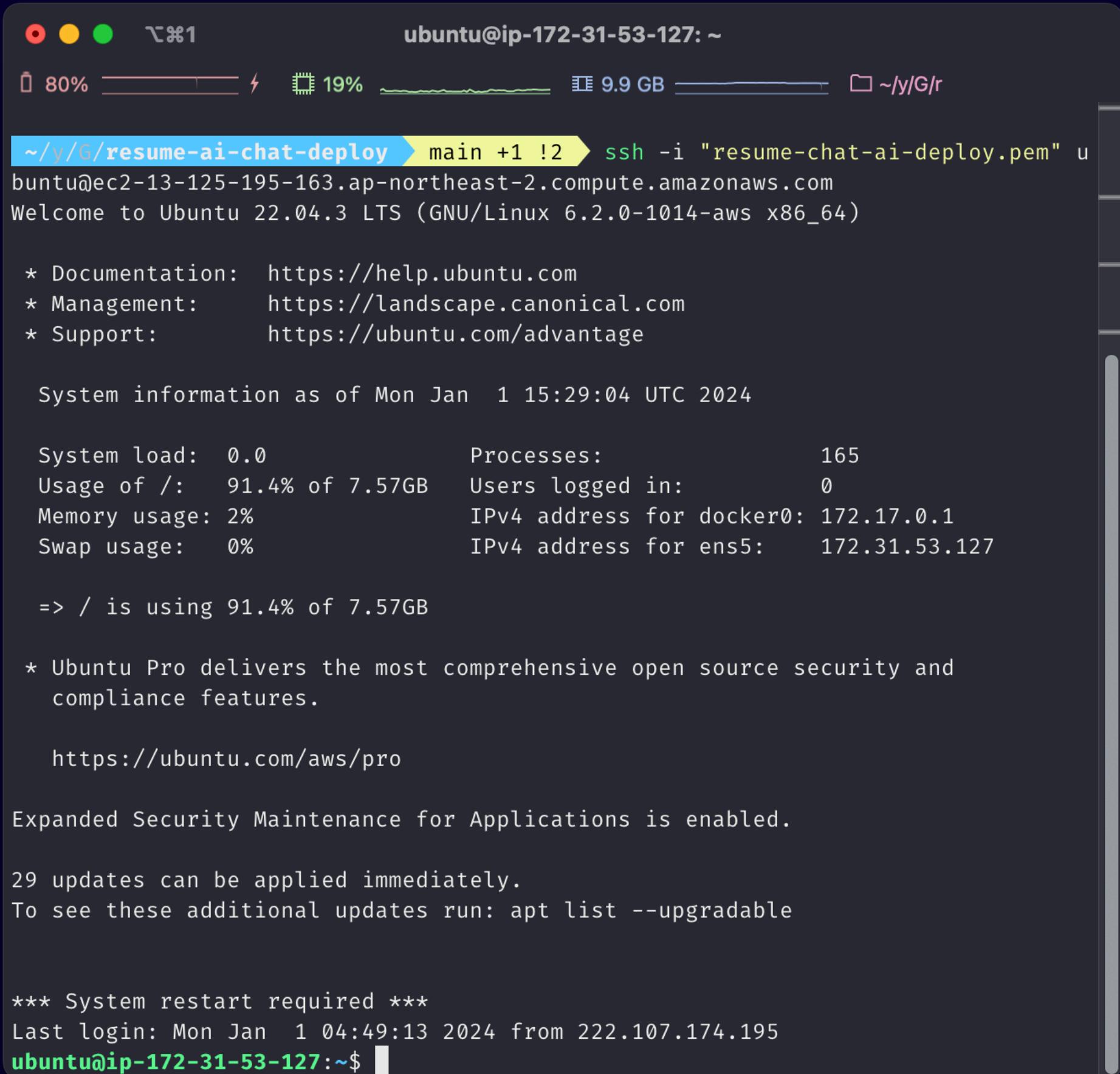
- **Pros**

- Environment Consistency - Docker 는 환경까지 모두 Image 안에 빌드된다. 따라서 실행 환경이 동일하다.
- Isolation - 도커는 어플리케이션을 실행하기 위한 격리된 환경을 제공한다.

- **Cons**

- Size - 환경까지 포함하니, Image 의 크기가 크다.
- Complexity - GitHub 보다는 도커를 다루는 것이 러닝커브가 있다. (~~배우면 됨~~)

# 실전에서 도커 사용하기 (feat. AWS)



A screenshot of a terminal window titled "ubuntu@ip-172-31-53-127: ~". The window shows a system status report. At the top, there are battery and signal strength indicators. The main text area displays the following information:

```
ubuntu@ip-172-31-53-127: ~
80% 19% 9.9 GB ~y/G/r

~/y/G/resume-ai-chat-deploy main +1 !2 ssh -i "resume-chat-ai-deploy.pem" u
buntu@ec2-13-125-195-163.ap-northeast-2.compute.amazonaws.com
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1014-aws x86_64)

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

System information as of Mon Jan  1 15:29:04 UTC 2024

System load: 0.0      Processes: 165
Usage of /: 91.4% of 7.57GB  Users logged in: 0
Memory usage: 2%          IPv4 address for docker0: 172.17.0.1
Swap usage: 0%            IPv4 address for ens5: 172.31.53.127

=> / is using 91.4% of 7.57GB

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

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is enabled.

29 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

*** System restart required ***
Last login: Mon Jan  1 04:49:13 2024 from 222.107.174.195
ubuntu@ip-172-31-53-127:~$
```

EC2 접속 – 가상의 컴퓨터임 (linux 기반)

# 실전에서 도커 사용하기 (feat. AWS)

〈초기 세팅〉

1. EC2 접속 (ssh, pem key)
2. sudo apt-get update
3. sudo apt install -y python3-pip nginx
4. sudo vim /etc/nginx/sites-enabled/fastapi\_nginx
5. 아래 입력 후 :wq

```
server {  
listen 80;  
server_name 13.125.195.163; —— EC2 의 public IPv4 address  
location / {  
proxy_pass http://127.0.0.1:8000;  
}  
}
```

# 🤔 명령어 해석

[해석]

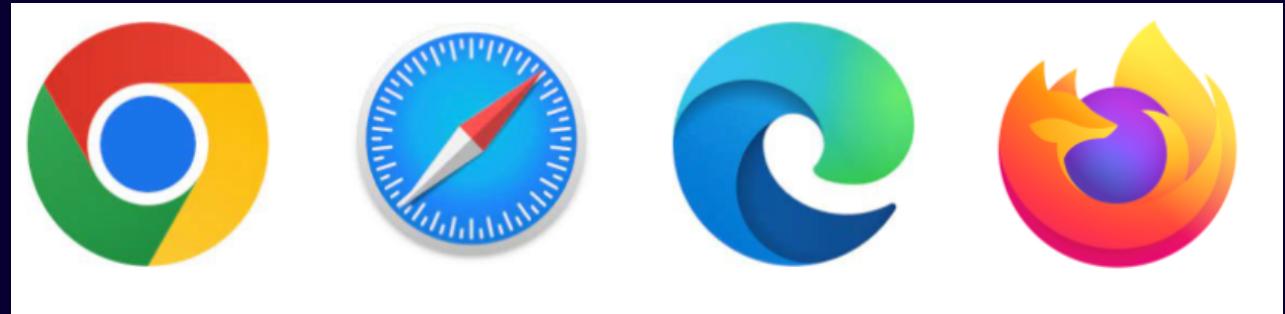
```
server {  
    listen 80;  
    server_name 13.125.195.163; ━━━━━━━━ EC2 의 public IPv4 address  
    location / {  
        proxy_pass http://127.0.0.1:8000;  
    }  
}
```

1. ./etc/nginx/sites-enabled/fastapi\_nginx  
→ sites-enabled 폴더가 서버가 호스팅하는 사이트에 대한 config 포함한다.  
나는 fastapi 를 사용했으므로 fastapi\_nginx 라고 이름 지음
2. server {} 블럭 → Nginx 에서 서버 컨텍스트를 정의함. 각 server block 는 Nginx 가 서빙하는 웹사이트 / API 를 의미함
3. listen 80 → Nginx 에게 connection 이 들어오면 port 80 에서 listen 하라는 의미. (port 80 은 HTTP 통신의 기본 포트 번호)
4. location / 블럭 → root URL (/) 에 대한 request 를 어떻게 처리해야 할지 정의한다.
5. proxy\_pass http://127.0.0.1:8000 → Nginx 에게 들어오는 모든 request 를 http://127.0.0.1:8000 에게 forward (= proxy) 하라고 명령한다. http://127.0.0.1 은 localhost 주소이다. 8000 는 FastAPI 가 돌아가고 있는 포트 번호이다. 따라서 EC2 에 들어오는 모든 HTTP request 는 port 8000 에서 돌아가고 있는 FastAPI 어플리케이션에 전달된다.

# 🤔 기본 용어부터 알고 가자

## [클라이언트]

- 서비스를 이용하기 위해 요청을 보내는 주체
- 인터넷에서 웹 페이지를 보려면, 웹 브라우저가 클라이언트임 (HTTP Request)
- 이메일을 보내기 위해서 이메일을 실행하면 이메일 클라이언트 (SMTP Request)



## [웹 서버]

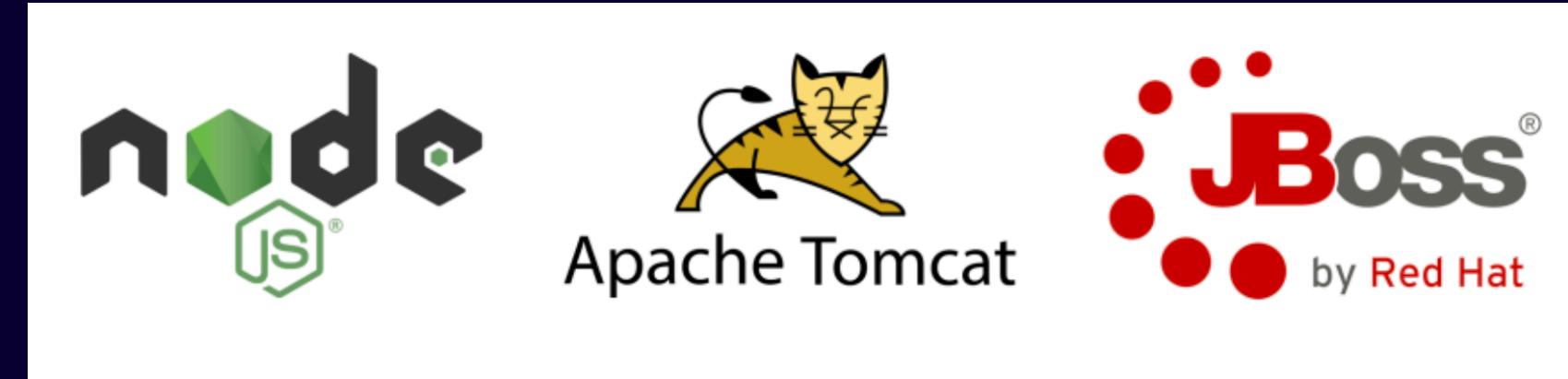
- 클라이언트의 요청에 따라 HTML, CSS, JS, 이미지 파일과 같은 정적 파일을 응답하여 제공하는 소프트웨어
- 웹 서버는 HTTP 프로토콜을 이용하여 클라이언트와 통신함



# 🤔 기본 용어부터 알고 가자

## [WAS (Web Application Server)]

- 클라이언트 요청에 대한 동적인 처리를 담당함.
- 웹 서버와 달리 어플리케이션 로직을 실행할 수 있음
- 데이터베이스 연동, 트랜잭션 관리, 보안, 로깅의 기능도 제공
- (웹 서버로 통 칠 수 있음..)



## [Database]

- 필요한 정보를 체계적으로 저장함
- Relational Database (RDBMS) 과 Not-Relational Database 가 있음



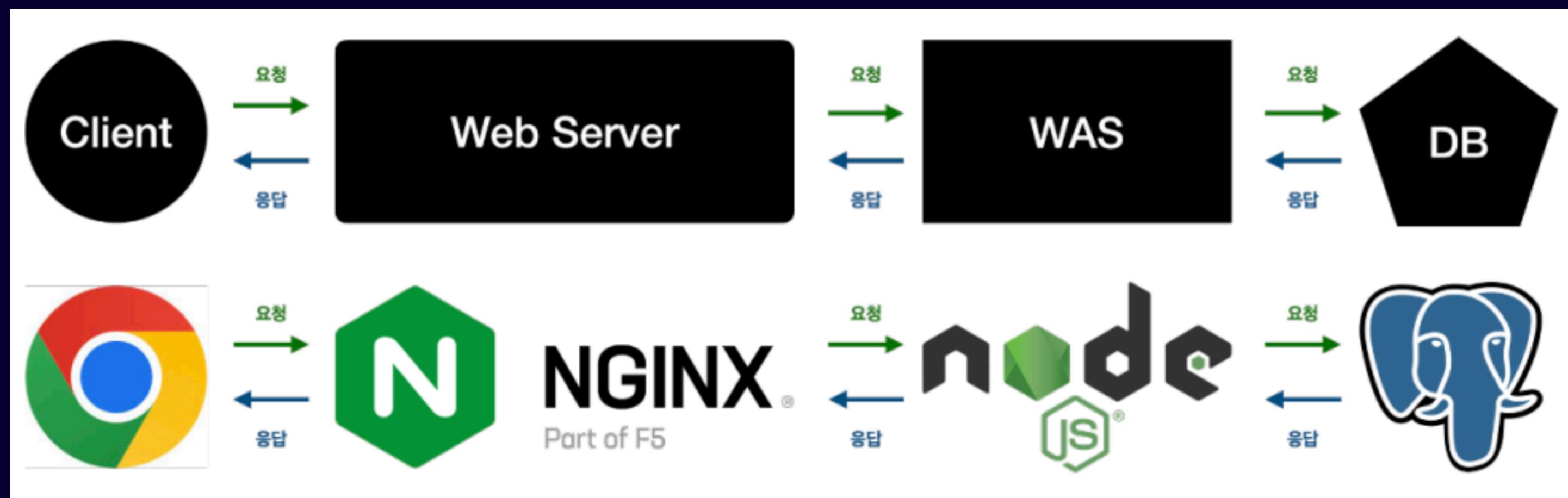
# 🤔 기본 용어부터 알고 가자

## 요청 순서

클라이언트 -> 웹 서버 -> WAS -> DB

## 응답 순서

DB -> WAS -> 웹 서버 -> 클라이언트





# Nginx ..?

## Nginx 사이트

NGINX is open source software for **web serving**, reverse proxying, caching, load balancing, media streaming, and more. It started out as a web server designed for maximum performance and stability. In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers.

Nginx는 **Web Server**이다. web server는 왜 사용할까?

### 1. WAS의 부담을 줄여준다.

WAS는 로그인, 회원가입, 개인정보 수정 등 동적 작업을 처리하는 것 만으로 작업량이 많다. 따라서 HTML, CSS, JS, 이미지 등 정적인 파일을 클라이언트에게 전달하는 역할을 웹 서버에게 위임한다.

### 2. 보안 기능 제공

웹 서버는 SSL/TLS 프로토콜을 사용하여 데이터를 암호화하고, 액세스 제어, 웹 방화벽 등 웹 사이트를 보호한다.

### 3. 높은 성능

웹 서버는 대부분 비동기 처리 방식을 사용한다. Nginx, Apache 등은 이벤트 기반, 멀티 프로세싱, 스레드 풀 등의 기술을 사용하여 수천 대의 클라이언트 요청을 동시에 처리할 수 있다.

참고 블로그  
참고 영상

# 실전에서 도커 사용하기 (feat. AWS)

```
ubuntu@ip-172-31-53-127:~$ sudo docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
soeunuhm/resume-ai-chat  latest   17c9367179de  10 days ago  1.24GB
ubuntu@ip-172-31-53-127:~$
```

DockerHub에 내 이미지 올려 놓고  
docker pull 했음

The screenshot shows a DockerHub repository page for the user 'soeunuhm' named 'resume-ai-chat'. The page includes sections for 'Docker commands', 'Tags', and 'Automated Builds'. A red arrow points from the text 'DockerHub에 내 이미지 올려 놓고 docker pull 했음' to the 'Docker commands' section, which contains the command 'docker push soeunuhm/resume-ai-chat:tagname'.

**Docker commands**  
To push a new tag to this repository:  
docker push soeunuhm/resume-ai-chat:tagname

**Tags**  
This repository contains 1 tag(s).  

Tag	OS	Type	Pulled	Pushed
latest		Image	10 days ago	10 days ago

[See all](#)

**Automated Builds**  
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.  
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#).

[Upgrade](#)

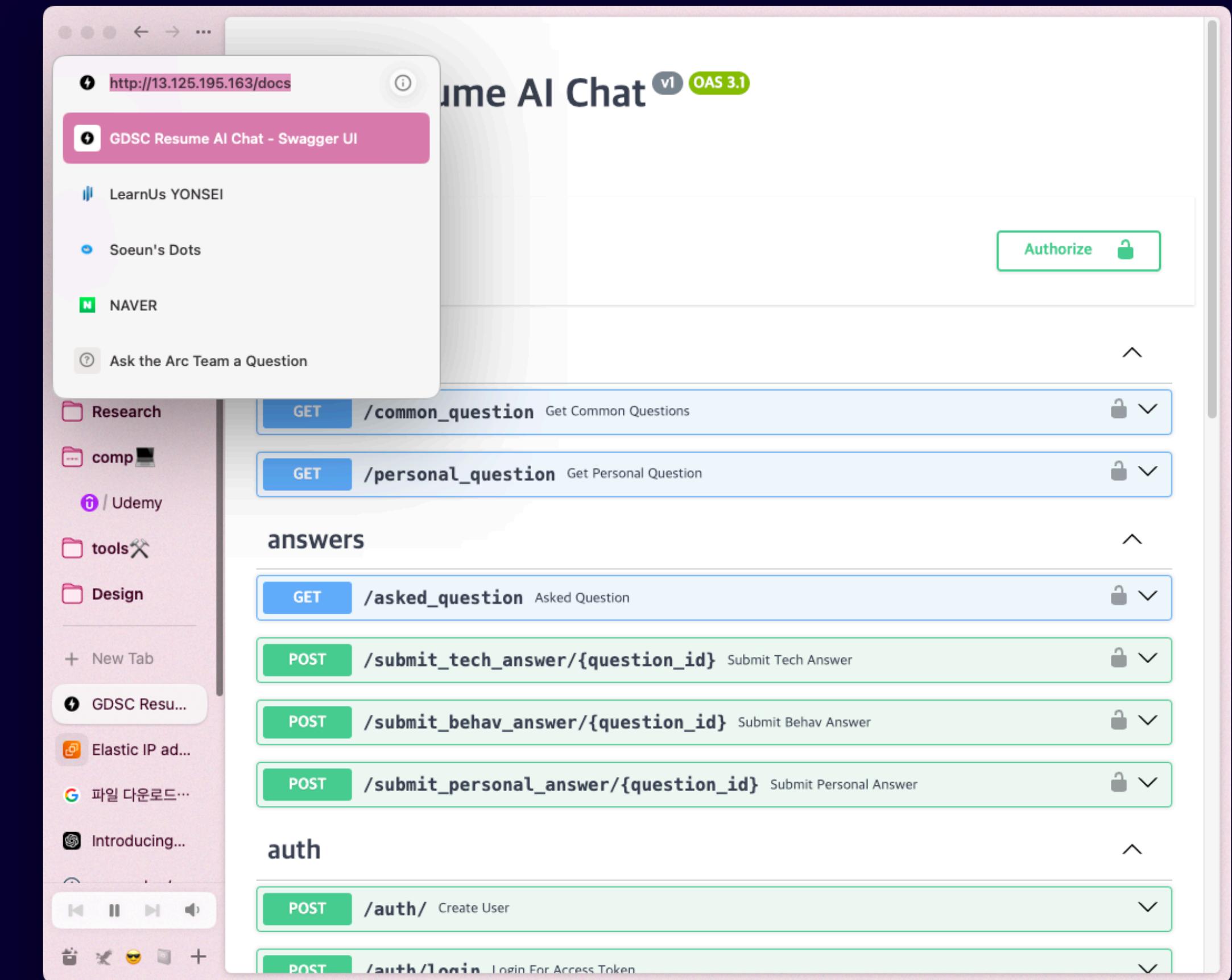
**Repository overview**  
An overview describes what your image does and how to run it.  
[Add overview](#)

EC2에 있는 이미지 리스트업 (docker images)

# 실전에서 도커 사용하기 (feat. AWS)

```
ubuntu@ip-172-31-53-127:~$ sudo docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
soeunuhm/resume-ai-chat  latest   17c9367179de  10 days ago  1.24GB
ubuntu@ip-172-31-53-127:~$ docker run soeunuhm/resume-ai-chat
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/create": dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
ubuntu@ip-172-31-53-127:~$ sudo docker run soeunuhm/resume-ai-chat
INFO:     Started server process [7]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

EC2에서 container 실행



public IPv4 접속하면 뜬다.

# 과제

1. <https://github.com/ddoddii/skills-for-DS/tree/main/week6> clone 하기
2. dockerfile 보지 말고 스스로 dockerfile 작성해보기
  - 2-1. 왜 COPY 를 세 번에 걸쳐서 했을지 생각해보기
  - 2-2. ENV 가 무엇일지 생각해보기
3. docker build 후 docker run 으로 컨테이너 실행하기 - 아래 명령어 참고  
(docker run --rm -d --name simple-logistic-container -p 9000:9000 soeunuhm/simple-logistic)  
–rm, –d, –name , –p 뜻 구글링하기
4. feedback.py 실행해보기  
(POST Request 가 무엇인지 모르면 빅자기 v1 참고 / 구글링 해보기… )
5. 본인 이미지 dockerhub 에 push 하기