제 목	01장 Docker 기본 개념	
상세내용	도커의 정의와 기본 개념 이해	

1. 도커란 무엇인가?

- 도커는 가상 환경에서 컨테이너로 일관된 실행 환경을 제공해 주는 기술이다.
- 도커를 사용하면 로컬 개발 환경과 운영 서버 환경을 일관되게 유지할 수 있습니다. 이는 개발과 운영 간의 환경 차이로 인한 문제를 줄여줍니다.
- 도커는 컨테이너라는 개념을 통해 애플리케이션과 그 종속성을 함께 패키징하여 어디서든 동일한 환경에서 실행될 수 있도록 합니다. 이는 개발자가 로컬에서 작업한 내용을 그대로 운영 서버에 배포할 수 있게 해줍니다.
- 도커의 주요 장점 중 하나는 가상 머신(VM)보다 가볍고 빠르다는 점입니다. 도커는 호스트 OS 위에 도커 엔진을 설치하고, 이 도커 엔진을 통해 여러 자원을 공유하여 VM보다 효율적으로 자원을 사용할 수 있습니다.
- 또한, 도커는 레이어 기반으로 운영되기 때문에 변경 사항이 있을 때 기본 베이스 레이어는 그대로 두고 변경된 레이어만 업데이트하여 배포할 수 있습니다. 이는 배포 속도를 크게 향상시킵니다.
- 도커를 사용하면 애플리케이션의 배포와 관리를 자동화할 수 있으며, 이는 DevOps 문화와도 잘 맞아떨어집니다. 도커는 CI/CD 파이프라인에서 중요한 역할을 합니다.

2. Docker를 왜 배우는 걸까?

✓ **현업에서는 Docker를 왜 이렇게 많이 쓰는 걸까?**

Docker를 쓰는 이유에는 여러가지 장점이 있지만 그 중에서 핵심 장점 딱 1가지만 기억하자.

이식성 : 특정 프로그램을 다른 곳으로 쉽게 옮겨서 설치 및 실행할 수 있는 특성

■ 이 핵심 장점을 예시를 통해 이해해 보자.

> 친구는 컴퓨터에 MySQL을 아무 에러 없이 잘 깔았다. 그런데 내 컴퓨터에 MySQL을 깔려고 하니 이상하게 에러가 뜨는 것이다. 분명 친구가 설치한 방식대로 똑같이 했는데 제대로 안 깔릴 때가 있다. 지우고 다시 깔아봐도 계속해서 똑같은 에러가 뜨기도 한다.

- 내 컴퓨터에만 MySQL이 안 깔리는 이유는 다양하다. 버전을 다른 걸 설치했거나, 운영체제(Window, Mac OS 등)가 다르거나, 내 컴퓨터에 깔려있는 다른 프로그램(ex. 보안 프로그램)과 충돌이 일어났거나와 같은 다양한 이유로 프로그램이 정상적으로 설치되지 않는다. 그리고 설치 과정이 복잡하다면 새 컴퓨터를 사서 MySQL을 설치할 때마다 번거롭고 귀찮다고 느껴진다.

- 이걸 깔끔하게 해결하기 위해 나타난 툴이 Docker이다. Docker를 사용하면 명령어 한 줄로 어떤 컴퓨터에든 MySQL을 에러 없이 설치하고 실행할 수 있게 된다.

- 뿐만 아니라 Docker를 사용하면 아래와 같은 장점이 있다.

  · 매번 귀찮은 설치 과정을 일일이 거치지 않아도 된다.

  · 항상 일관되게 프로그램을 설치할 수 있다. (버전, 환경 설정, 옵션, 운영 체제 등)

  · 각 프로그램이 독립적인 환경에서 실행되기 때문에 프로그램 간에 서로 충돌이 일어나지 않는다.

3. IP와 Port의 개념

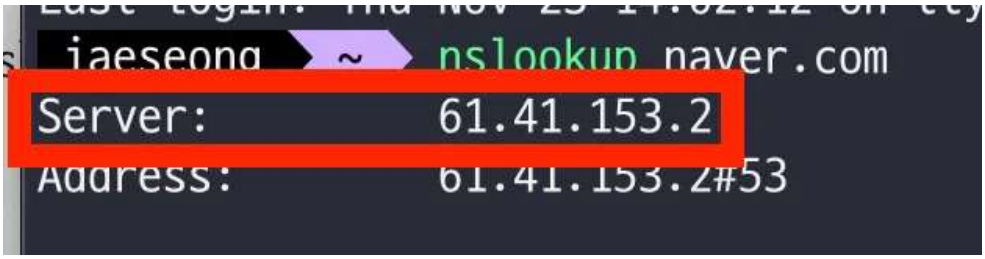
✓ IP의 개념

한줄요약 : 트워크 상에서의 특정 컴퓨터를 가리키는 주소

- 아래와 같은 값이 IP 주소이다.

13.250.15.132

- IP는 특정 컴퓨터의 주소를 가리킨다. 예를 들면, naver.com이라는 서비스도 IP 주소를 가지고 있다. 저 IP 주소는 네이버가 운영하고 있는 컴퓨터의 주소이기도 하다.



✓ Port의 개념

한줄요약 : 한 컴퓨터 내에서 실행되고 있는 특정 프로그램의 주소이다.

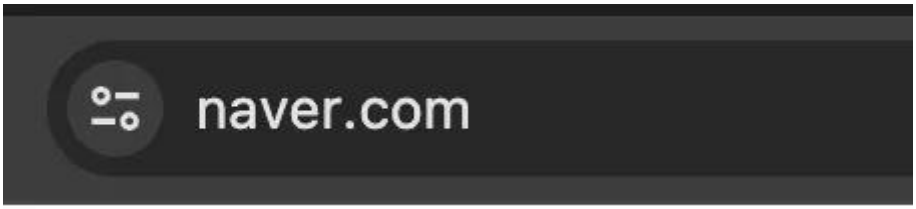
- 아래와 같은 값에서 :3000 부분이 포트 번호를 의미한다.

13.250.15.132:3000

- 한 컴퓨터 내에서 여러 프로그램이 실행되고 있을 것이다. 내가 지금 사용하고 있는 노트북만 보더라도 크롬, 카카오톡, 슬랙, VSCode 등 여러가지 프로그램이 동시에 실행되고 있다. 실제 서버를 운영하는 컴퓨터도 동일하다. 하나의 컴퓨터에서 여러가지의 프로그램이 실행된다.
- 그럼 외부에서 특정 컴퓨터 내부에 있는 Spring Boot라는 서버에 통신을 하고 싶다고 가정하자. 하지만 외부에서 IP 주소만 알아서는 실행되고 있는 여러 프로그램 중 어떤 프로그램과 통신을 해야 할 지 알 수가 없다. 그래서 특정 서버와 통신을 할 때는 **IP 주소**와 서버가 실행되고 있는 **포트 번호**까지 알고 있어야 한다.

✓ **브라우저 창에 포트 번호를 입력하지 않는 이유?**

- 위 설명에서 분명 특정 서버와 통신하기 위해서는 **IP 주소**와 **포트 번호**를 둘 다 알아야 된다고 했다. 도메인 주소를 통해서 알 수 있는 건 **IP 주소** 뿐이다. 그럼 **포트 번호**를 입력해주지도 않았는데 어떻게 정상적으로 통신을 한 걸까?



- 주소창에 도메인 주소를 입력해서 엔터를 누르면, **브라우저(크롬, 익스플로러 등)는 기본적으로 80번 포트로 통신을 보내게 설정되어 있다.** 그래서 포트 번호를 입력해 주지 않아도 통신이 잘 됐던 것이었다. 만약 80번 포트로 통신하고 싶지 않고, 3000번 포트로 통신하고 싶다면 아래와 같이 주소창에 입력해야 한다.

✓ 잘 알려진 포트(well-known port)란 ?

포트(Port)에는 잘 알려진 포트(well-known port)라는 개념이 있다.

- 포트 번호는 0 ~ 65,535번까지 사용할 수 있다. 그 중에서 0 ~ 1023번까지의 포트 번호는 주요 통신을 위한 규약에 따라 이미 정해져 있다. 이렇게 규약을 통해 역할이 정해져있는 포트 번호를 보고 잘 알려진 포트(well-known port)라고 부른다.
- 규약으로 정해져 있는 포트 번호 중 자주 사용되는 포트 번호에 대해서만 알아보자.

· **22번 (SSH, Secure Shell Protocol)** : 원격 접속을 위한 포트 번호

  - EC2 인스턴스에 연결할 때 22번 포트를 사용한다.

· **80번 (HTTP)** : HTTP로 통신을 할 때 사용

· **443번 (HTTPS)** : HTTPS로 통신을 할 때 사용

- 여기서 착각하면 안 되는 점은 위에서 정해놓은 규약을 꼭 지키지 않아도 된다. 즉, 규약으로 정해져 있는 포트 번호와 다르게 사용해도 된다는 뜻이다. 예를 들어, 특정 서버와 HTTP 통신을 할 때 80번 포트를 쓰지 않고 3000번 포트나 8080번 포트를 써도 상관 없다.

4. Docker란 ? / 컨테이너(Container)란? / 이미지(Image)란?

✓ Docker란 ?

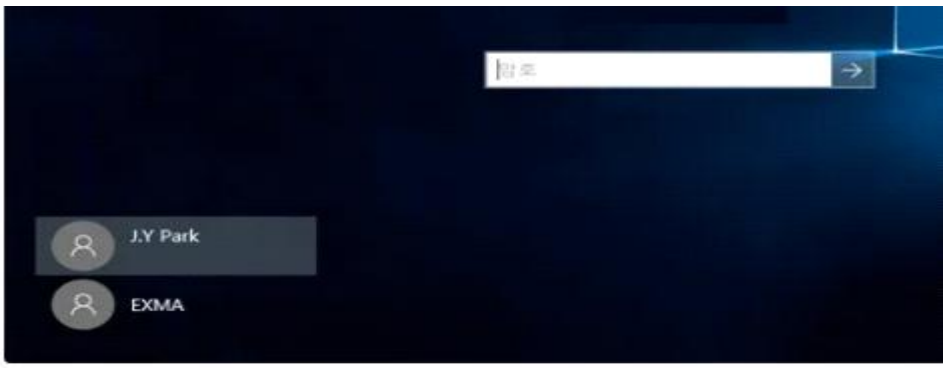
- **컨테이너**를 사용하여 각각의 프로그램을 **분리된 환경**에서 실행 및 관리할 수 있는 툴이다.

Docker의 정의를 보더라도 한 번에 와닿지 않을 것이다. 당연하다. Docker가 어떤 툴인지는 직접 사용해보고 경험해봐야 느낄 수 있다. 직접 사용해보고 경험해보는 게 Docker가 어떤 툴인지 제일 빠르게 파악하는 방법이다.

✓ 컨테이너(Container)란?

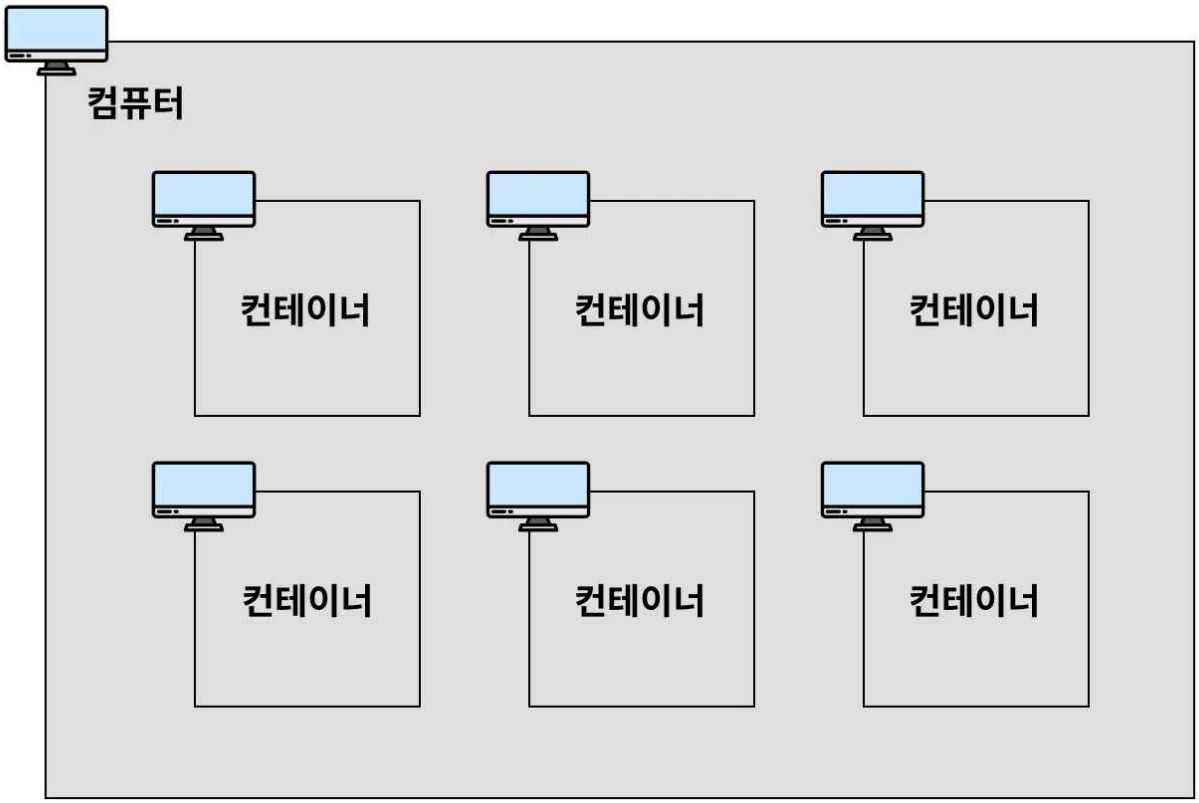
Docker에서 컨테이너(Container)라는 개념은 아주 중요한 개념이다. 머릿속에 컨테이너(Container)가 어떤 개념인 지 대략적으로 떠올릴 수 있어야 한다. 그래야 Docker를 쉽게 배울 수 있다.

- 윈도우 환경을 사용해보면 하나의 컴퓨터에 여러 사용자로 나눠서 사용할 수 있게끔 구성되어 있다. 각 사용자의 환경에 들어가보면 독립적으로 구성되어 있어서 필요한 프로그램을 각 사용자 환경에 따로따로 설치해주어야 한다.



- 컨테이너도 이와 비슷한 개념이다. 하나의 컴퓨터 환경 내에서 **독립적인 컴퓨터 환경을** 구성해서, 각 환경에 프로그램을 별도로 설치할 수 있게 만든 개념이다. 하나의 컴퓨터 환경 내에서 여러개의 **미니 컴퓨터 환경**을 구성할 수 있는 형태이다. 여기서 얘기하는 **미니 컴퓨터**를 보고 Docker에서는 **컨테이너(Container)**라고 부른다.

  (컨테이너(Container)를 이해할 때 머릿속의 이미지로 미니 컴퓨터를 떠올리면 이해하기 편하다.)



- 여기서 **'컨테이너'**와 **'컨테이너를 포함하고 있는 컴퓨터'**를 구분하기 위해 컨테이너를 포함하고 있는 컴퓨터를 **'호스트(host) 컴퓨터'**라고 부른다.

✓ 컨테이너(Container)의 독립성

위의 설명에서 컨테이너는 '독립적인 컴퓨터 환경'이라고 얘기했다. 구체적으로 어떤 것들이 독립적으로 관리되는 지 기억해두자.

- **디스크 (저장 공간)** : 각 컨테이너마다 서로 각자의 저장 공간을 가지고 있다. 일반적으로 A 컨테이너 내부에서 B 컨테이너 내부에 있는 파일에 접근할 수 없다.
- **네트워크 (IP, Port)** : 각 컨테이너마다 고유의 네트워크를 가지고 있다. 컨테이너는 각자의 IP 주소를 가지고 있다.

✓ 이미지(Image)란?

- 닌텐도와 같은 게임기를 보면 여러가지 칩을 꽂아서 다양한 게임을 즐길 수 있게 되어 있다. Docker에서는 **닌텐도의 칩**과 같은 역할을 하는 개념이 **이미지(Image)**이다.

- Node.js 기반의 Express.js 서버 프로젝트를 이미지로 만들었다고 가정해보자. 이 이미지를 Docker로 실행시키면 Express.js 서버 프로젝트가 컨테이너(Container) 환경에서 실행된다. 복잡한 설치 과정을 거칠 필요 없이 손쉽게 실행된다.

- 또 다른 예로, MySQL 서버를 이미지로 만들었다면, 이 이미지를 Docker로 실행시키는 순간 MySQL 서버가 컨테이너(Container) 환경에서 실행된다. MySQL을 일일이 설치할 필요없이 MySQL 데이터베이스를 사용할 수 있게 된다.

- **이미지(Image)**는 **프로그램을 실행하는 데 필요한 설치 과정, 설정, 버전 정보 등을 포함**하고 있다. 즉, **프로그램을 실행하는 데 필요한 모든 것을 포함**하고 있다.

5. Docker 설치 (Windows, Mac OS)

✓ Docker 설치 (윈도우)

- 참조 : https://tinyurl.com/284cgb3u

✓ Docker 설치 (Mac OS)

- 참조 : https://happylie.tistory.com/78

✓ 체크 사항

- 2023년 7월부터 Docker Compose V1의 업데이트를 중단했다. 따라서 Docker Compose는 V2를 설치할 것을 권장한다. 혹시나 기존에 설치되어 있는 Docker Compose의 버전이 V1이라면 V2로 교체할 것을 권장한다.

[실습] Docker 전체 흐름 느껴보기 (Nginx 설치 및 실행)

✓ Docker를 조작하려면?

- 터미널에서 명령어(CLI)를 통해 Docker를 조작한다.

✓ 혹시나 Nginx가 뭔지 모르는 분들을 위해

- Nginx란 여러 기능을 가진 서버 중 하나이다.

- **웹 서버 (HTML 웹 페이지를 렌더링 시키는 역할)**
- 로드 밸런싱 (몰라도 괜찮습니다.)
- 리버스 프록시 (몰라도 괜찮습니다.)

✓ 무작정 따라하면서 Docker 사용해보기

1. Nginx 이미지 다운로드 : https://hub.docker.com/

```
$ docker pull nginx
```

2. 다운로드 된 이미지 확인하기

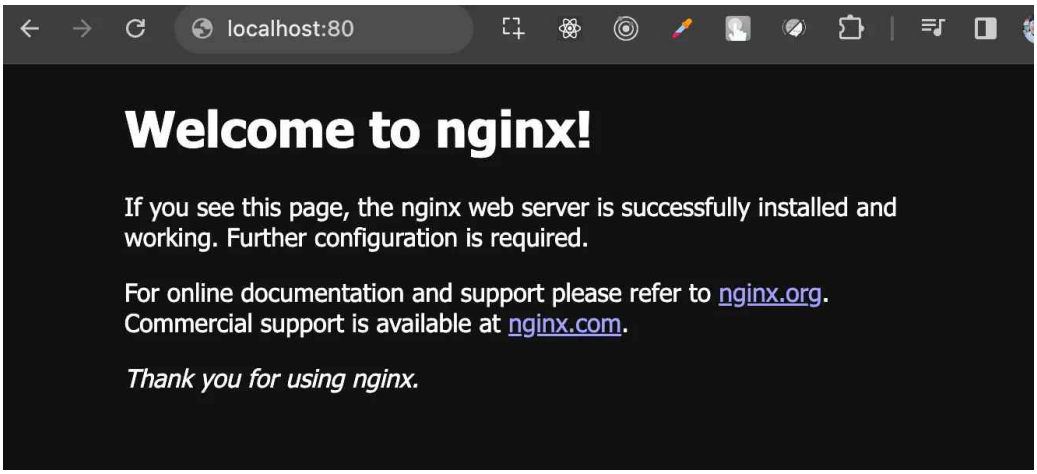
```
$ docker image ls
```

- **ls** : list의 약자

3. 이미지를 컨테이너에 올려 Nginx 서버 실행시키기

```
$ docker  run --name webserver -d -p 80:80 nginx
```

4. 1. Nginx 서버가 잘 실행되는 지 확인하기



5. 실행되고 있는 모든 컨테이너 상태 확인하기

```
$ docker ps
```

6. 특정 컨테이너 정지

```
$ docker stop webserver
```

✓ 그림으로 이해하기