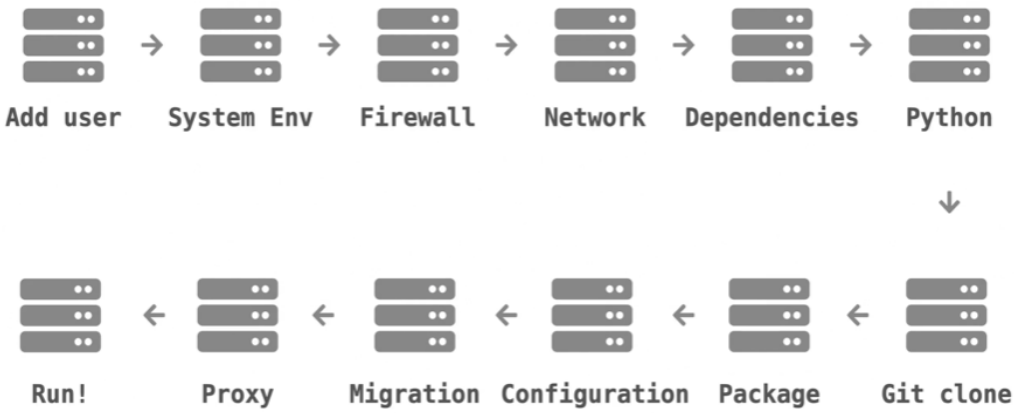
0️⃣

0. 도커란 무엇인가

0. 도커란 무엇인가

0.1 서버관리 방식의 변화

도커가 등장하기 전 서버 관리 방식은 매우 복잡했다.



<도커 등장 이전의 서버관리 방식>

도커 등장 이후, 프로그램 종류에 상관 없이 컨테이너로 만들 수 있고, 어디에서든 실행 가능함.

0.2 도커란?

도커

: 도커는 **컨테이너 기반의 오픈소스 가상화 플랫폼**으로, 가상머신처럼 독립적으로 실행되지만, 가상머신보다 빠르고, 쉽고, 효율적(가상머신은 CPU나 메모리가 느림)임

도커의 특징

1. 확장성/이식성

- 도커가 설치되어 있다면 어디서든 컨테이너를 실행할 수 있음
- **오픈소스**이므로 특정 회사나 서비스에 종속적이지 않음

- 쉽게 개발 서버와 테스트 서버를 생성할 수 있음

2. 표준성

- 도커를 사용하지 않는 경우 ruby, nodejs, go, php로 만든 서비스들의 배포 방식은 제각각 다르나, 도커를 사용하면 컨테이너라는 표준으로 서버를 배포하므로 모든 **서비스들의 배포 과정이 동일**해짐

3. 이미지

- 이미지에서 컨테이너를 생성하기 때문에 반드시 이미지를 만드는 과정이 필요
- Dockerfile을 이용하여 이미지를 만들고 처음부터 재현 가능
- 빌드 서버에서 이미지를 만들면 해당 이미지를 이미지 저장소에 저장하고 운영 서버에서 이미지를 불러옴

4. 설정관리

- 도커에서 설정은 보통 환경 변수로 제어. ex) MYSQL_PASS=password와 같이 컨테이너를 띄울 때 환경 변수를 같이 지정
- 하나의 이미지가 환경 변수에 따라 동적으로 설정 파일을 생성하도록 만들어져야 함

5. 자원관리

- 컨테이너를 삭제 후 새로 만들면 모든 데이터가 초기화됨
- 업로드 파일을 외부 스토리지와 링크하여 사용하거나 S3같은 별도의 저장소가 필요
- 세션이나 캐시를 memcached나 redis와 같은 외부로 분리

도커가 가져온 변화

- 클라우드 이미지보다 관리하기 쉬움
- 다른 프로세스와 격리되어 가상머신처럼 사용하지만 성능저하가 거의 없음
- 복잡한 기술을 몰라도 사용할 수 있음
- 이미지 빌드 기록이 남음 ⇒ git으로 버전 관리 가능
- 코드와 설정으로 관리 ⇒ 재현 및 수정 가능
- 오픈소스 ⇒ 특정 회사 기술에 종속적이지 않음

0.3 도커의 미래 (= 컨테이너의 미래)

쿠버네티스란?

: 여러 대의 서버와 여러 개의 서비스를 관리하기 쉽게 하는 방식 = 컨테이너 관리

쿠버네티스 특징

1. **스케줄링**

- 컨테이너를 적당한 서버에 배포해 주는 작업
- 여러 대의 서버 중 가장 할일 없는 서버에 배포하거나 그냥 차례대로 배포 또는 아예 랜덤하게 배포
- 컨테이너 개수를 여러 개로 늘리면 적당히 나눠서 배포하고 서버가 죽으면 실행 중이던 다른 서버에 띄워줌

2. **클러스터링**

- 여러 개(수천대 까지)의 서버를 하나의 서버처럼 사용
- 여기저기 흩어져 있는 컨테이너도 가상 네트워크를 이용하여 마치 같은 서버에 있는 것처럼 쉽게 통신

3. **서비스 디스커버리**

- 서비스를 찾아주는 기능
- 클러스터 환경에서 컨테이너는 어느 서버에 생성될지 알 수 없고 다른 서버로 이동할 수도 있으므로, 컨테이너와 통신을 하기 위해서 어느 서버에서 실행 중인지 알아야 하고 컨테이너가 생성되고 중지될 때 어딘가에 IP와 Port 같은 정보를 업데이트 해줘야 함
- 키-밸류 스토리지에 정보를 저장할 수도 있고 내부 DNS 서버를 이용