4. Docker가 하는 일은 무엇일까요?

5. Docker는 컨테이너에 application과 그에 필요한 system tools, dependencies를 하나로 묶어 다른 server, pc등 그 어떤 곳에도 쉽게 배포하고 안정적으로 구동할 수 있게 도와줍니다.

6. 왜 Docker는 이런 일을 할까요?

7. 어플리케이션을 구동할 때는 많은 것이 필요합니다. Node js를 예로 들어보면 우리의 소스파일만 서버에 배포하는 것으로는 우리 어플리케이션을 구동하는데 문제점이 있습니다. Node js와 npm 외부 라이브러리를 사용한다면 여러 dependencies와 환경 변수 등 모든 것들을 다 설정을 해줘야 되는데요, 서버마다, 개발자들의 pc마다 이런 모든 것들을 설치하고 설정하는 것은 번거롭고 오류도 많이 발생할 수 있습니다.

8. 내가 node js를 가지고 있고 서버에도 node js가 있으니까 "내 소스코드를 서버에 배포하면 자동으로 동작하겠지"라고 생각하면 에러가 발생할 수 있습니다. "내 pc에서는 잘되는데 왜 서버에서는 안되는거지?"라고 생각해보면 node js의 버전이 맞지 않아서 그럴 수도 있습니다.

9. 이러한 문제점을 해결하기 위해 Docker가 탄생했습니다.

16. 컨테이너(Docker)는 가벼워 이러한 문제를 해결할 수 있습니다.

17. 컨테이너는 host os에서 container engine이라는 소프트웨어를 설치만 하면 개별적인 컨테이너를 만들어서 각각의 어플리케이션을 고립된 환경에서 구동할 수 있게 해줍니다. vm과 큰 차이점은 vm은 운영체제를 가지고 있으며, 컨테이너는 운영체제를 가지지 않고 컨테이너 엔진이 설치된 host os를 공유한다는 점입니다. 즉, 컨테이너가 구동되기 위해서는 컨테이너 엔진이 필요합니다. 컨테이너 엔진이 host os에 접근해서 필요한 것들을 처리합니다.

18. 이러한 컨테이너 엔진 중 가장 많이 사용되는 것이 Docker입니다. Docker는 컨테이너를 만들고 배포하고 구동합니다.

19. 컨테이너를 만들기 위해서는 총 3가지가 필요합니다. Docker file, 이미지, 컨테이너입니다. Dockerfile을 만들고 이를 이용하여 이미지를 만들어 컨테이너를 구동합니다.

20. Dockerfile은 쉽게 말하자면 컨테이너를 어떻게 만드는지 설명서 같은 것입니다. 어플리케이션을 구동하기 위해서 꼭 필요한 파일은 무엇이 있는지, 어떤 라이브러리를 설치해야되는지, 외부 dependencies에 대해 명시할 수 있고 필요한 환경변수에 대해 설정, 어떻게 구동해야되는지 script도 포함할 수 있습니다. 이렇게 작성한 Dockerfile로 이미지를 만듭니다.

21. 이미지안에는 어플리케이션을 실행하는데 필요한 코드, runtime 환경, 시스템 툴, 시스템라이브러리 등 모든 세팅들이 포함되어 있습니다. 한마디로 얘기하면 실행되고 있는 어플리케이션의 상태를 찰칵해서 이미지로 만들어 둔다고 생각하면 됩니다. 그래서 만들어진 이미지는 변경이 불가능한 불변의 상태입니다.

22. 컨테이너는 샌드박스처럼 잘 캡쳐해둔 어플리케이션의 이미지를 고립된 환경에서 (개별적인 파일 시스템안에서) 실행하게 할 수 있는 것을 말합니다. 컨테이너 안에서 앱이 동작합니다. 즉, 컨테이너는 이미지를 이용해서 어플리케이션을 구동하게 되는 것입니다.

23. 어떻게 이미지를 배포할까요?

24. 로컬머신에서 이미지를 만들어서 container registry에 내가 만든 이미지를 push하고 필요한 서버나 다른 개발자 pc에서 내가 만든 이미지를 가져와서 그걸 그대로 실행하면 됩니다. 이를 위해서는 Docker같은 컨테이너 엔진을 꼭 설치해둬야합니다..

25. 컨테이너 레지스트리의 종류는 이러합니다. 왼쪽은 일반인들이 많이 사용하며, 오른쪽은 기업체들이 많이 사용합니다.

26. 이미지 배포 방법에 대해 정리해보자면, 로컬머신에 Docker, 서버에 Docker을 설치한후 구동하는데 필요한 Docker파일을 작성하고, 이것을 이용해 이미지를 만듭니다. 이미지를 container registry에 push한 후 server에서 다운로드 받아서 container를 실행하면 됩니다.

10. 이 Docker 컨테이너안에는 어플리케이션뿐만 아니라 어플리케이션을 정상적으로 동작하게 해주는 node js, 환경설정, npm, 여러 라이브러리들의 dependencies 그리고 어플리케이션에 필요한 다양한 리소스들이 포함됩니다. 한마디로 얘기하면 앱을 구동하는데 필요한 모든 것들을 이 Docker 컨테이너안에 담아 놨다 볼 수 있습니다. 이렇게 Docker 컨테이너를 사용하면 앱을 구동하기 위한 런타임 환경에 필요한 모든 것들을 어떤 pc에서도 언제든지 동일하게 구동할 수 있습니다. 위와 같은 "왜 내pc에서는 작동되는데 다른 pc에서는 작동이 안되지?" 같은 문제를 해결할 수 있고 다른 pc에서 사용하기 위해 이것저것 설정하고 준비해야 되는 문제점을 해결해줍니다.

11. 이러한 문제가 Docker를 사용함으로써

12. 이렇게 해결됩니다.

13. VM쓰면 되는거 아니야?라고 하시는 분들도 계실텐데요 둘의 차이점을 설명드리겠습니다

14. vm은 하드웨어 infrastructure위에 VMware나 VirtualBox같은 hypervisor 소프트웨어를 이용해서 각각의 가상머신을 만듭니다. 한 운영체제 위에서 동일한 어플리케이션을 각각의 고립된 다른 환경에서 구동하기위에서는 vm을 이용해서 어플리케이션을 구동해야 합니다. 예를 들어 mac에서 vm을 이용해 윈도우와 리눅스를 동시에 구동할 수 있습니다.

15. 따라서 VM은 엄청나게 무겁습니다. 운영체제를 포함하고 있기 때문에 굉장히 무겁고 시작하는데도 오래 걸리고 infrastructure의 리소스를 많이 잡아먹는 범인이 될 수도 있습니다.