이미지 만들고 배포하기

도커 이미지 만들기

- 도커는 이미지를 만들기 위해 컨테이너의 상태를 그대로 이미지로 저장하는 단순한 방법을 사용

- 예를 들어 어떤 애플리케이션을 이미지로 만든다면 리눅스만 설치된 컨테이너에 애플리케이션을 설치하고 그 상태를 그대로 이미지로 저장한다. 가상머신의 스냅샷과 비슷한 방식.

Ruby로 만들어진 간단한 웹 애플리케이션을 **도커라이징**(Dockerizing, 도커 이미지를 만듦)해보자.

1. 패키지 관리
2. 웹서버

호스트의 디렉토리를 루비가 설치된 컨테이너의 디렉토리에 마운트한다음 그대로 명령어를 실행하면 로컬에 개발 환경을 구축하지 않고 도커 컨테이너를 개발환경으로 사용할 수 있습니다. ~~어으쌈!~~

개발, 테스트, 운영이 동일한 환경에서 실행 가능!

1) Dockerfile 생성

```
# 1. ubuntu 설치(패키지 업데이트, 만든사람 표시)
FROM ubuntu:20.04
MAINTAINER copes.km@gmail.com
RUN apt-get -y update

# 2. Ruby 설치
RUN apt-get -y install ruby
RUN gem install bundler

# 3. 소스 복사
COPY . /usr/src/app

# 4. Gem 패키지 설치(실행 디렉토리 설정)
WORKDIR /usr/src/app
RUN bundle install

# 5. Sinatra 서버 실행 (Listen 포트 정의)
EXPOSE 4567
CMD bundle exec ruby app.rb -o 0.0.0.0
```

2) Build

- docker build -t app .

3) 이미지 생성된 것

- docker images 로 확인

4) 실행

- docker run -d -p 8080:4567 app

-> http://localhost:8080 으로 웹에 접속하면 Host ID가 출력되는 코드에 따라 출력된다.

FROM

```
FROM <image>:<tag>
FROM ubuntu:16.04
```

- 베이스 이미지를 지정(반드시 지정해야 한다. 어떤 이미지도 베이스 이미지가 될 수 있다.)

- tag는 될 수 있으면 latest(기본값)보다 구체적인 버전(16.04 등)을 지정하는 것이 좋다. Docker Hub(https://hub.docker.com/search?q=&type=image)에서 다양한 Image를 확인할 수 있다.

MAINTAINER

```
MAINTAINER <name>
MAINTAINER subicura@subicura.com
```

- Dockerfile을 관리하는 사람의 이름 및 이메일 정보를 적는다. 빌드에 영향 X

COPY

```
COPY <src>... <dest>
COPY . /usr/src/app
```

- 파일이나 디렉토리를 이미지로 복사. 일반적으로 소스를 복사하는데 사용된다.

- target 디렉토리가 없다면 자동으로 생성한다.

ADD

```
ADD <src>... <dest>
ADD . /usr/src/app
```

- COPY 명령어와 매우 유사하지만, 몇 가지 추가 기능이 존재한다.

- src에 파일 대신 URL을 입력할 수 있으며, src에 압축파일을 입력하는 경우 자동으로 압축을 해제하면서 복사된다.

RUN

```
RUN <command>
RUN ["executable", "param1", "param2"]
RUN bundle install
```

- 명령어를 그대로 실행.

- 내부적으로 /bin/sh -c 뒤에 명령어를 실행하는 방식

CMD

```
CMD ["executable","param1","param2"]
CMD command param1 param2
CMD bundle exec ruby app.rb
```

- Docker 컨테이너가 실행되었을 때 실행되는 명령어를 정의한다.

- 빌드할 때는 실행되지 않으며, 여러 개의 CMD가 존재할 경우 가장 마지막 CMD만 실행된다. 한번에 여러 개의 프로그램을 실행하고 싶은 경우 1) run.sh파일을 작성하여 데몬으로 실행하거나, 2) supervisord나 forego와 같은 여러 개의 프로그램을 실행하는 프로그램을 사용한다.

WORKDIR

```
WORKDIR /path/to/workdir
```

- RUN, CMD, ADD, COPY 등이 이루어질 기본 디렉토리를 설정한다.

- 각 명령어의 현재 디렉토리는 한 줄 한 줄마다 초기화되기 때문에 `RUN cd /path`를 하더라도 **다음 명령어에서는 다시 위치가 초기화**된다. **같은 디렉토리에서 작업을 계속 진행하기 위해서 WORKDIR을 사용한다.**

EXPOSE

```
EXPOSE <port> [<port>...]
EXPOSE 4567
```

- Docker 컨테이너가 실행되었을 때, 요청을 기다리고 있는 (Listen) 포트를 지정한다. 여러 개의 포트를 지정할 수 있다.

VOLUME

```
VOLUME ["/data"]
```

- 컨테이너 외부에 파일 시스템을 마운트할 때 사용한다. 반드시 지정하지 않아도 마운트할 수 있지만, 기본적으로 지정하는 것이 좋다.

ENV

```
ENV <key> <value>
ENV <key>=<value> ...
ENV DB_URL mysql
```

- 컨테이너에서 사용할 환경 변수를 지정한다. 컨테이너를 실행할 때 `-e` 옵션을 사용하면 기존 값을 오버라이딩하게 된다.

+) 공식 문서(https://docs.docker.com/engine/reference/builder/)

Build 분석

- 빌드 명령어를 실행한 디렉토리의 파일들을 build context라고 하고, 이 파일들을 Docker Server(daemon)으로 전송한다. 도커는 서버 - 클라이언트 구조이므로 도커 서버가 작업하려면 미리 파일을 전송해야 한다.

- 한 줄 한 줄 수행

- 명령어 수행 결과를 이미지로 저장…

요약하자면)

`임시 컨테이너 생성` > `명령어 수행` > `이미지로 저장` > `임시 컨테이너 삭제` > `새로 만든 이미지 기반 임시 컨테이너 생성` > `명령어 수행` > `이미지로 저장` > `임시 컨테이너 삭제` > … 의 과정을 계속해서 반복

리팩토링

명령어 최적화

이쁘게

Docker Hub

- 도커 허브는 도커에서 제공하는 기본 이미지 저장소로 ubuntu, centos, Debian 등의 베이스 이미지와 ruby, golang, java, python 등의 공식 이미지가 저장되어 있다. 일반 사용자들이 만든 이미지도 50만 개가 넘게 저장되어 있고, 다운로드 횟수는 80억 회를 넘는다.

- 회원가입만 하면 대용량의 이미지를 무료로 저장할 수 있고, 다운로드 트래픽 또한 무료이다. 단, 기본적으로 모든 이미지는 공개되어 누구나 접근 가능하므로 비공개로 사용하려면 유료 서비스를 이용해야 한다.(한 개는 무료)

로그인, Docker Hub에 Push

1) `docker login`

   - `~/.cocker/config.json`에 인증정보가 저장되어 로그아웃하기 전까지 로그인 정보가 유지된다.

2) `docker tag app copes/sinatra-app:1`

   - `copes`라는 ID를 사용하고, 이미지는 `sinatra-app`으로 변경, 첫 번째 버전이므로 태그는 `1`을 사용한다. `push`를 이용해서 도커 허브에 이미지를 전송

3) `docker push copes/sinatra-app:1`

   - 이제 어디서든 `copes/sinatra-app:1`을 사용할 수 있다.

Private Docker Registry

- 도커 이미지를 비공개로 저장하려면 1) Docker Cloud를 유료로 사용하거나 2) 레지스트리 서버를 자체적으로 구축해야 한다.

- 도커 레지스트리는 도커를 이용하여 쉽게 만들 수 있다. 도커 이미지를 저장할 서버를 도커로 스스로 만들어서 도커 이미지를 관리 …

```
docker run -d \
-v $PWD/registry:/var/lib/registry \
-p 5000:5000 \
distribution/registry:2.6.0
```

- 저장된 이미지는 파일로 관리되기 때문에 호스트의 디렉토리를 마운트

- 레지스트리 서버의 아이피와 포트 정보를 이미지 명에 추가하면 바로 사용할 수 있다.

```
> docker tag app localhost:5000/copes/sinatra-app:1
> docker push localhost:5000/copes/sinatra-app:1
```

- 앞에서 만든 이름에 localhost:5000/을 추가하였다.

```
> tree registry
```

- 레지스트리 서버에 파일이 잘 저장되었는지 마운트한 디렉토리를 살펴본다.

- 레이어별로 이쁘게 확인할 수 있다.

배포하기(Deploy)

- 이미지를 다운받고 컨테이너를 실행하면 끝!!!

컨테이너 업데이트

- 업데이트는 어떻게 하나? 최신 이미지를 기반으로 새 컨테이너를 만들고 이전 컨테이너를 중지(삭제)하면 된다. 최신 소스를 어떻게 복사할지, 서버 프로세스는 어떻게 재시작할지 고민할 필요가 없이 통째로 바꿔버리면 된다.

- 단, 컨테이너를 중지하지 않고 부드럽게(자연스럽게(?)) 교체하는 방법은 존재하지 않는다. nginx나 HAProxy 같은 Load Balancer와 2개 이상의 컨테이너를 사용해야 한다