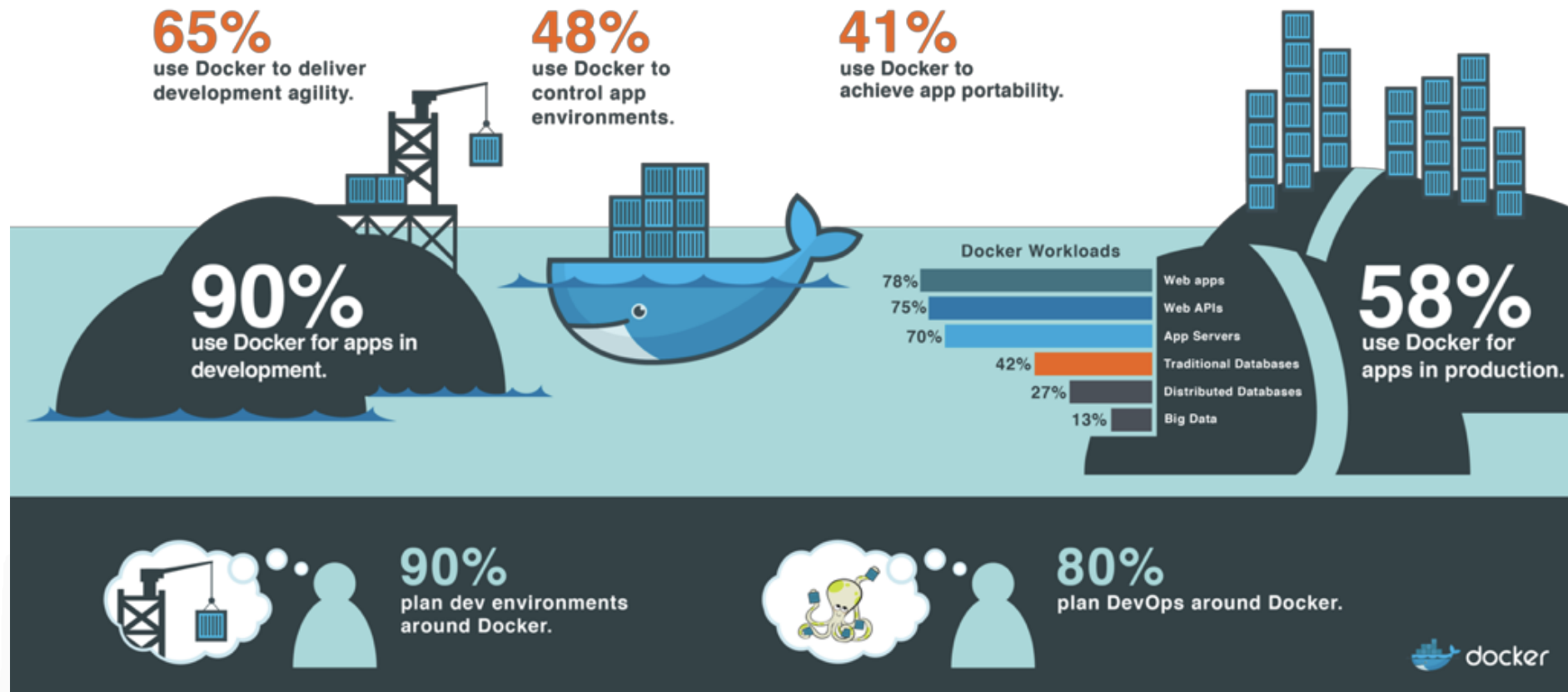
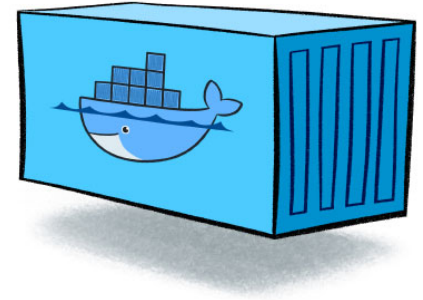


3. Docker

Docker

- 2014년 6월 Docker 1.0 발표
- 컨테이너 기반의 오픈소스 가상화 플랫폼
- 백엔드 프로그램, 데이터베이스, 메시지 큐 → 컨테이너로 추상화 가능

일반PC, AWS, Azure, Google cloud 등에서 실행 가능



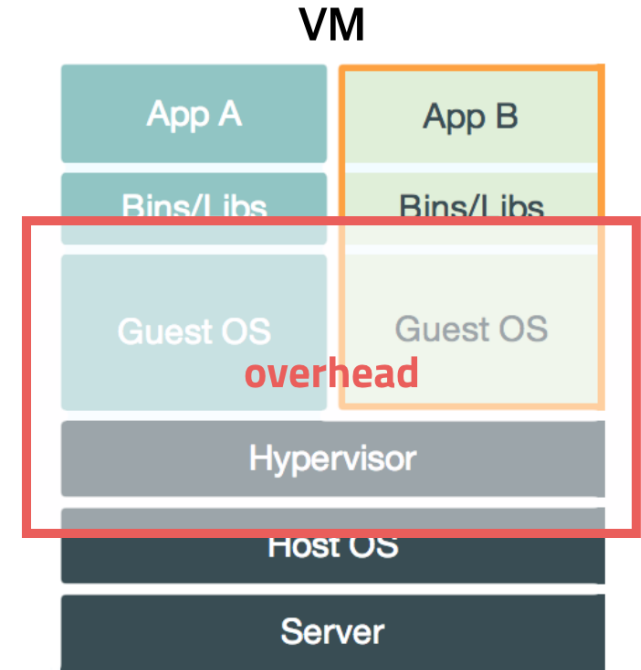
Docker

기존 가상화 방식 → **os를 가상화**

- VMWare, VirtualBox (**Host OS** 위에 **Guest OS** 전체를 가상화)
- **무겁고 느림**

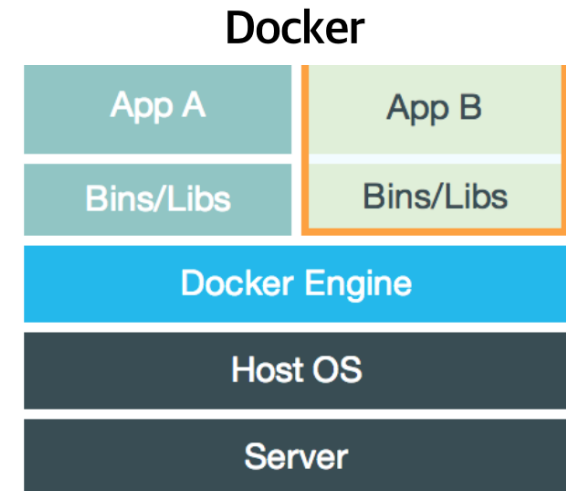
CPU의 가상화 기술 이용 방식 → **Kernel-based Virtual Machine**

- 전체 OS를 가상화 하지 않음, 호스트 형식에 비해 속도 향상
- OpenStack, AWS 등의 클라우드 서비스
- **추가적인 os**는 여전히 필요, 성능 문제



프로세스 격리 → 리눅스 컨테이너

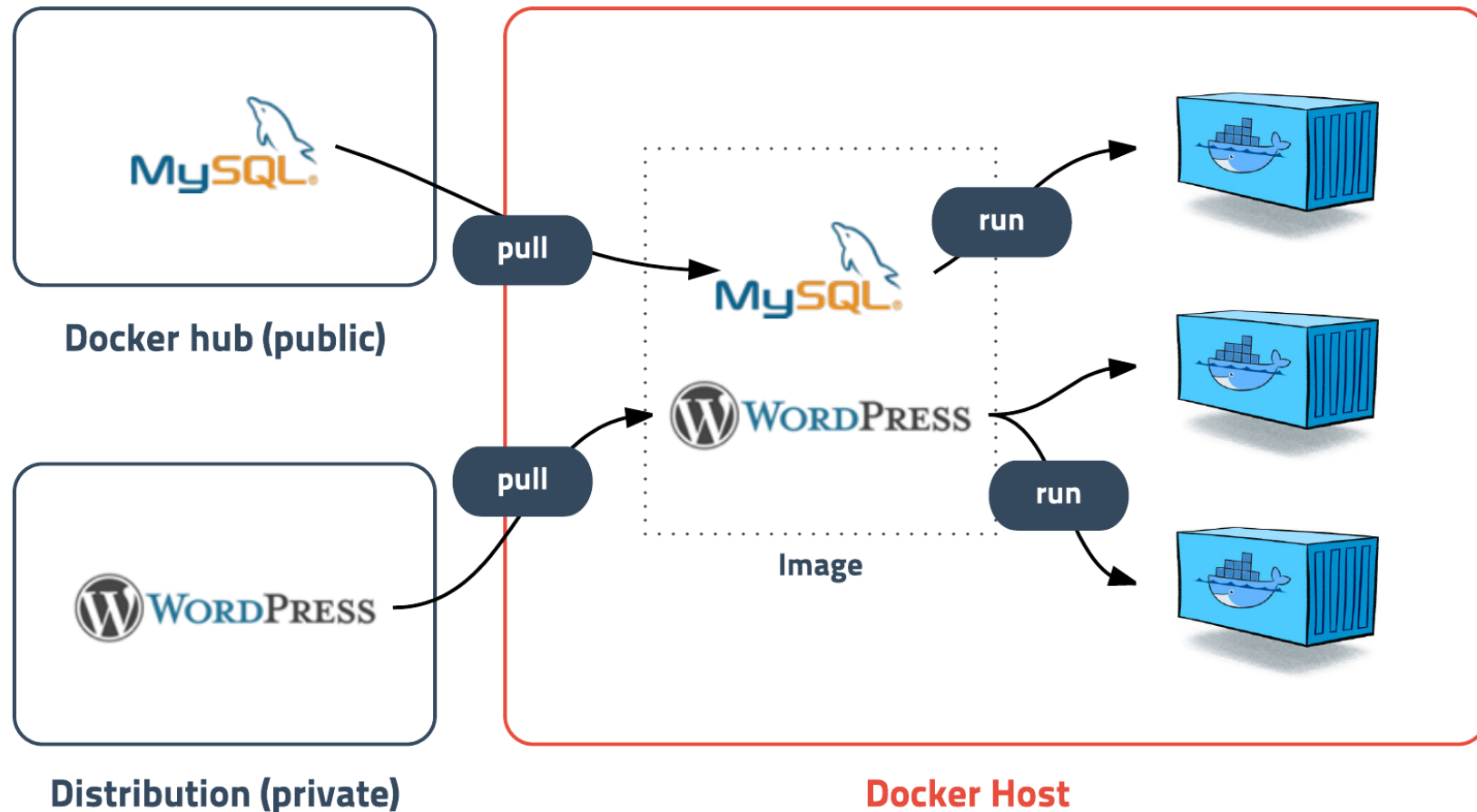
- CPU나 메모리는 프로세스에 필요한 만큼만 추가로 사용
- 성능 손실 거의 없음
- 컨테이너들 사이는 서로 영향을 주지 않음
- 컨테이너 생성 속도 빠름 (1-2초 내)



■ Docker

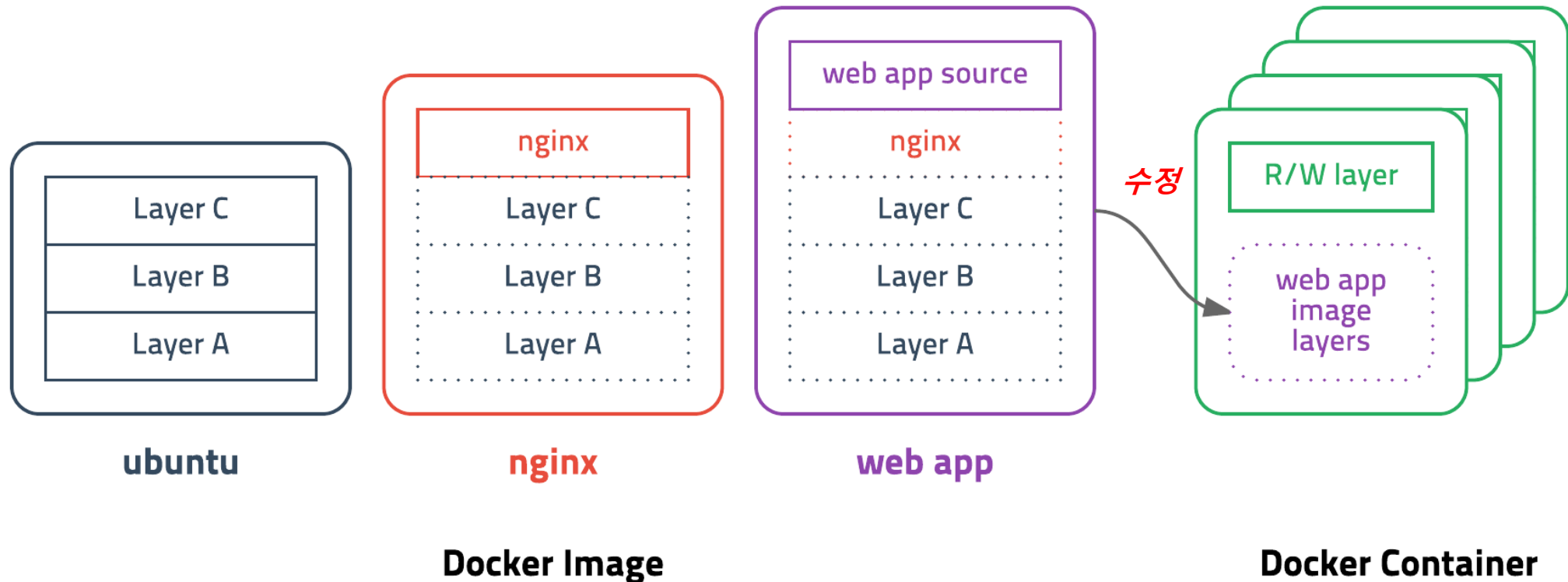
■ Docker Image

- 컨테이너 실행에 필요한 파일과 설정 값 등을 포함 → 상태값 X, Immutable
- 실체화 → **Container**



■ Docker

- **Docker Hub**에 등록 or **Docker Registry** 저장소를 직접 만들어 관리
 - 공개된 도커 이미지는 50만개 이상, 다운로드 수는 80억회 이상
- **Layer** 저장방식
 - 유니온 파일 시스템을 이용 → 여러 개의 Layer를 하나의 파일시스템으로 사용 가능



■ Docker

■ Dockerfile

- **Docker Image**를 생성하기 위한 스크립트 파일
- 자체 DSL(Domain-Specific language) 언어 사용 → 이미지 생성과정 기술
 - 서버에 프로그램을 설치하는 과정을 메모장이 아닌 **Dockerfile**로 관리
 - 소스와 함께 버전 관리가 되며, 누구나 수정 가능

```
1  # vertx/vertx3 debian version
2  FROM subicura/vertx3:3.3.1
3  MAINTAINER chungsub.kim@purpleworks.co.kr
4
5  ADD build/distributions/app-3.3.1.tar /
6  ADD config.template.json /app-3.3.1/bin/config.json
7  ADD docker/script/start.sh /usr/local/bin/
8  RUN ln -s /usr/local/bin/start.sh /start.sh
9
10 EXPOSE 8080
11 EXPOSE 7000
12
13 CMD ["start.sh"]
```

■ Docker

- *Docker for Mac/Docker for Windows* or 직접 *Linux*에 설치

```
$ curl -fsSL https://get.docker.com | sudo sh
```

- *sudo 없이 사용*

```
$ sudo usermod -aG docker $USER
```

```
$ sudo usermod -aG docker <your-user>
```

■ Docker

■ 설치 확인

\$ docker version

bcadmin@hlf03:~\$ **docker version**

Client:

Version: 18.06.1-ce

API version: 1.38

Go version: go1.10.3

Git commit: e68fc7a

Built: Tue Aug 21 17:24:51 2018

OS/Arch: linux/amd64

Experimental: false

Server:

Engine:

Version: 18.06.1-ce

API version: 1.38 (minimum version 1.12)

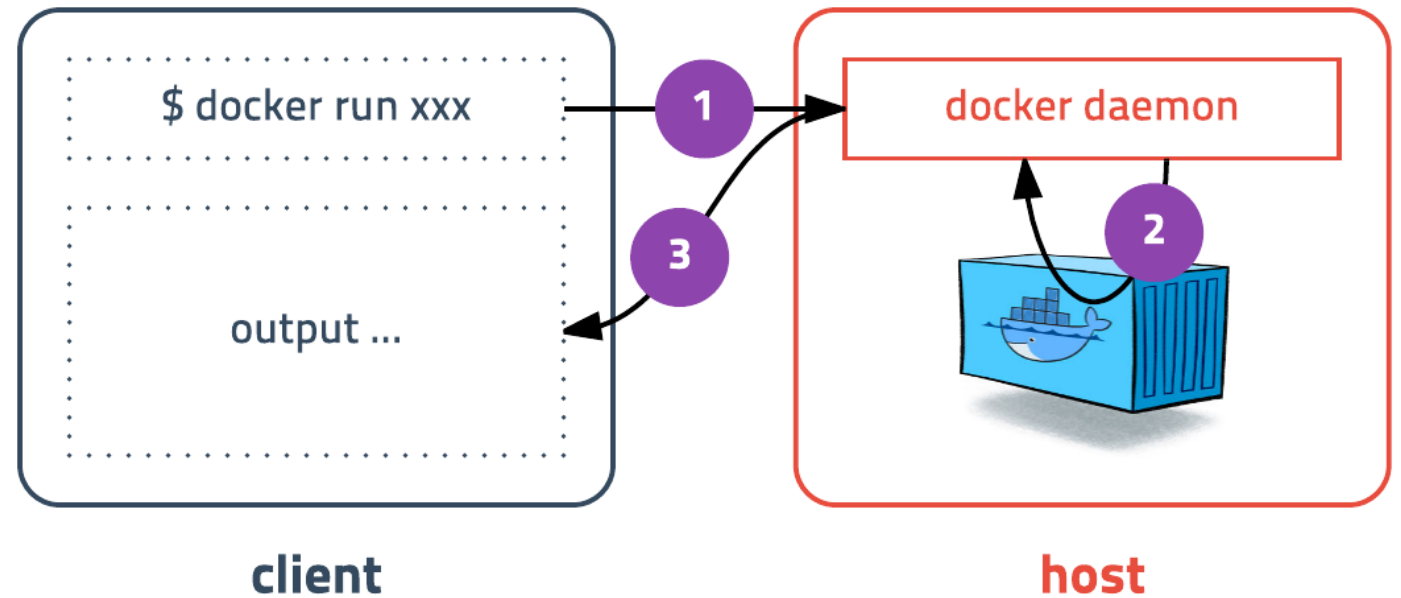
Go version: go1.10.3

Git commit: e68fc7a

Built: Tue Aug 21 17:23:15 2018

OS/Arch: linux/amd64

Experimental: false



■ Docker

■ 컨테이너 실행

\$ docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]

옵션	설명
-d	detached mode 흔히 말하는 백그라운드 모드
-p	호스트와 컨테이너의 포트를 연결 (포워딩)
-v	호스트와 컨테이너의 디렉토리를 연결 (마운트)
-e	컨테이너 내에서 사용할 환경변수 설정
-name	컨테이너 이름 설정
-rm	프로세스 종료시 컨테이너 자동 제거
-it	-i와 -t를 동시에 사용한 것으로 터미널 입력을 위한 옵션
--link	컨테이너 연결 [컨테이너명:별칭]

실습1) ubuntu 16.04 container 생성하고 컨테이너 내부 접속

\$ docker run ubuntu:16.04

```
[admin@centos7 ~]$ docker run ubuntu:16.04
Unable to find image 'ubuntu:16.04' locally
16.04: Pulling from library/ubuntu
35b42117c431: Pull complete
ad9c569a8d98: Pull complete
293b44f45162: Pull complete
0c175077525d: Pull complete
Digest: sha256:a4d8e674ee993e5ec88823391de828a5c
Status: Downloaded newer image for ubuntu:16.04
```

■ Docker

- 컨테이너는 프로세스이기 때문에 실행 중인 프로세스가 없으면 컨테이너는 종료 됨

\$ docker run -rm -it ubuntu:16.04 /bin/bash

```
[admin@centos7 ~]$ docker run --rm -it ubuntu:16.04 /bin/bash
root@b0eac8e53eaa:/# cat /etc/issue
Ubuntu 16.04.6 LTS
root@b0eac8e53eaa:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc
```

■ Docker

실습2) MySQL 5.7 container

\$ docker run -d -p 3306:3306 -e MYSQL_ALLOW_EMPTY_PASSWORD=true --name mysql mysql:5.7

```
bcadmin@hlf03:~$ docker run -d -p 3306:3306 -e MYSQL_ALLOW_EMPTY_PASSWORD=true --name mysql mysql:5.7
Unable to find image 'mysql:5.7' locally
5.7: Pulling from library/mysql
f17d81b4b692: Pull complete
c691115e6ae9: Pull complete
41544cb19235: Pull complete
254d04f5f66d: Pull complete
4fe240edfdc9: Pull complete
0cd4fcc94b67: Pull complete
8df36ec4b34a: Pull complete
b8edeb9ec9e2: Pull complete
2b5adb9b92bf: Pull complete
5358eb71259b: Pull complete
e8d149f0c48f: Pull complete
Digest: sha256:42bab37eda993e417c5e7d751f1008b65
Status: Downloaded newer image for mysql:5.7
842ff7eb6799b714050615ce505c1f96625343c0589f5d31
```

\$ docker exec -it mysql bash

```
bcadmin@hlf03:~$ docker exec -it mysql bash
root@842ff7eb6799:/# mysql -h127.0.0.1 -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.24 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.00 sec)

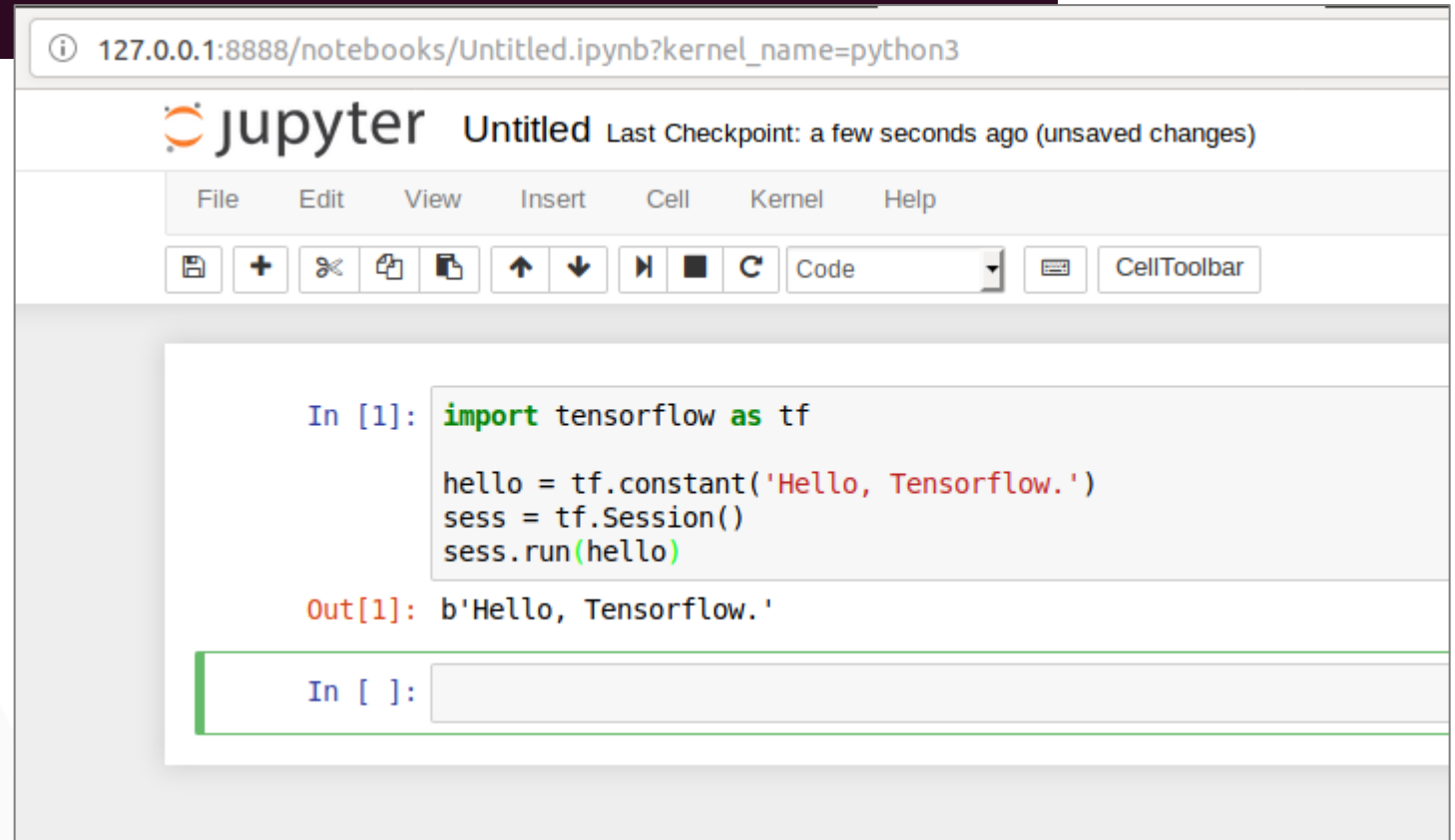
mysql>
```

■ Docker

실습) Tensorflow

```
$ docker run -d -p 8888:8888 teamlab/pydata-tensowflow:0.1
```

```
bcadmin@hlf03:~$ docker run -d -p 8888:8888 -p 6006:6006 teamlab/pydata-tensorflow:0.1
Unable to find image 'teamlab/pydata-tensorflow:0.1' locally
0.1: Pulling from teamlab/pydata-tensorflow
f069f1d21059: Downloading [=====] 36.24MB/49.17MB
ecbeec5633cf: Download complete
ea6f18256d63: Download complete
54bde7b02897: Download complete
```



127.0.0.1:8888/notebooks/Untitled.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Save + Undo Copy Paste Up Down Run Stop Refresh Code CellToolbar

```
In [1]: import tensorflow as tf

hello = tf.constant('Hello, Tensorflow.')
sess = tf.Session()
sess.run(hello)

Out[1]: b'Hello, Tensorflow.'
```

In []:

■ Docker

■ 기본 명령어

\$ docker **ps** [OPTIONS]

\$ docker **stop** [OPTIONS] CONTAINER [CONTAINER ...]

\$ docker **rm** [OPTIONS] CONTAINER [CONTAINER ...]

\$ docker **images** [OPTIONS] [REPOSITORY[:TAG]]

\$ docker **rmi** [OPTIONS] IMAGE [IMAGE ...]

\$ docker **pull** [OPTIONS] NAME[:TAG | @DIGEST]

ex) docker pull ubuntu:16.04

\$ docker **logs** \${CONTAINER_ID}

ex) docker logs --tail 10 \${CONTAINER_ID}

ex) docker logs -f \${CONTAINER_ID}

\$ docker **exec** [OPTIONS] CONTAINER COMMAND [ARG...]

ex) docker exec -it mysql /bin/bash

\$ docker **system prune** -a

■ Docker

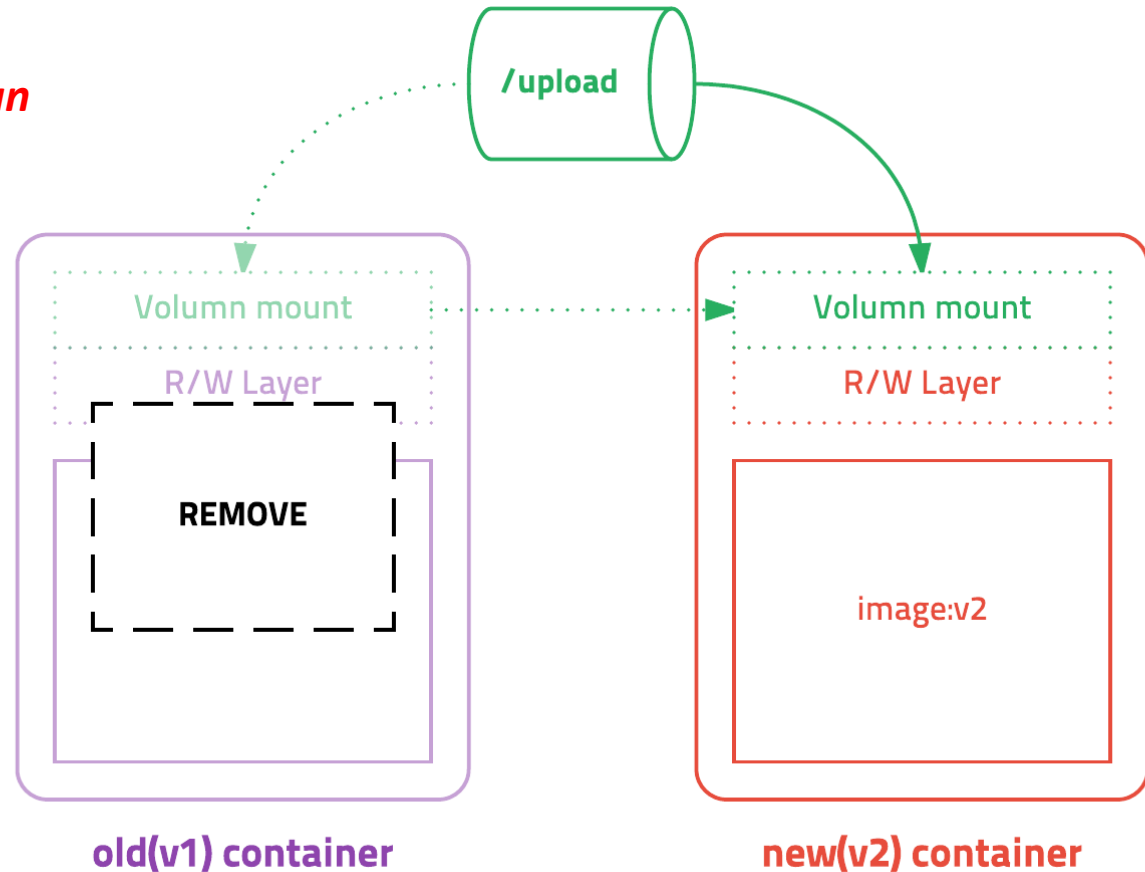
■ 컨테이너 업데이트

- 새 버전의 이미지 다운 → **pull**
- 기존 컨테이너 삭제 → **stop, rm**
- 새 이미지를 이용하여 새 컨테이너 실행 → **run**

■ 컨테이너 유지 정보

- **AWS S3**
- **데이터 볼륨**

```
$ docker run -d -p 3306:3306 \
  -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
  --name mysql \
  -v /my/datadir:/var/lib/mysql
mysql:5.7
```



■ Docker-Compose

- **Docker 커맨드** or 복잡한 **설정**을 쉽게 관리하기 위한 **도구**

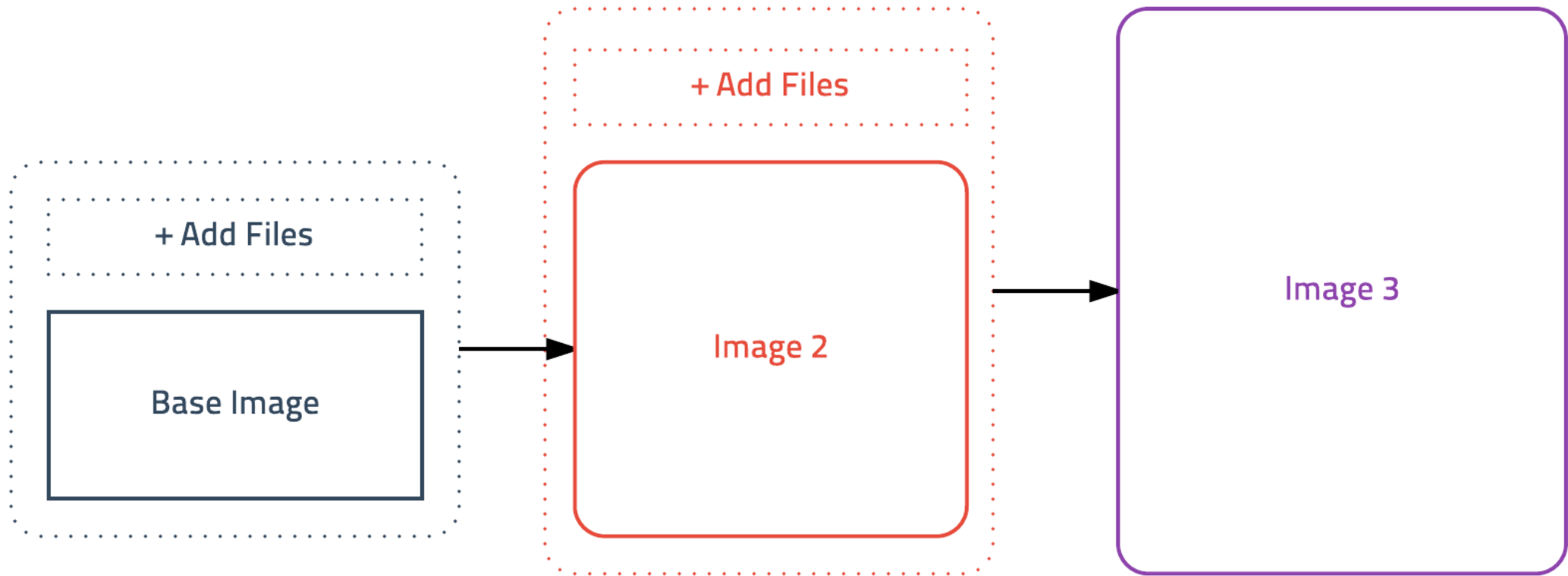
- **YAML** format

```
$ curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64" \
-o /usr/local/bin/docker-compose
```

```
$ chmod -x /usr/local/bin/docker-compose
```

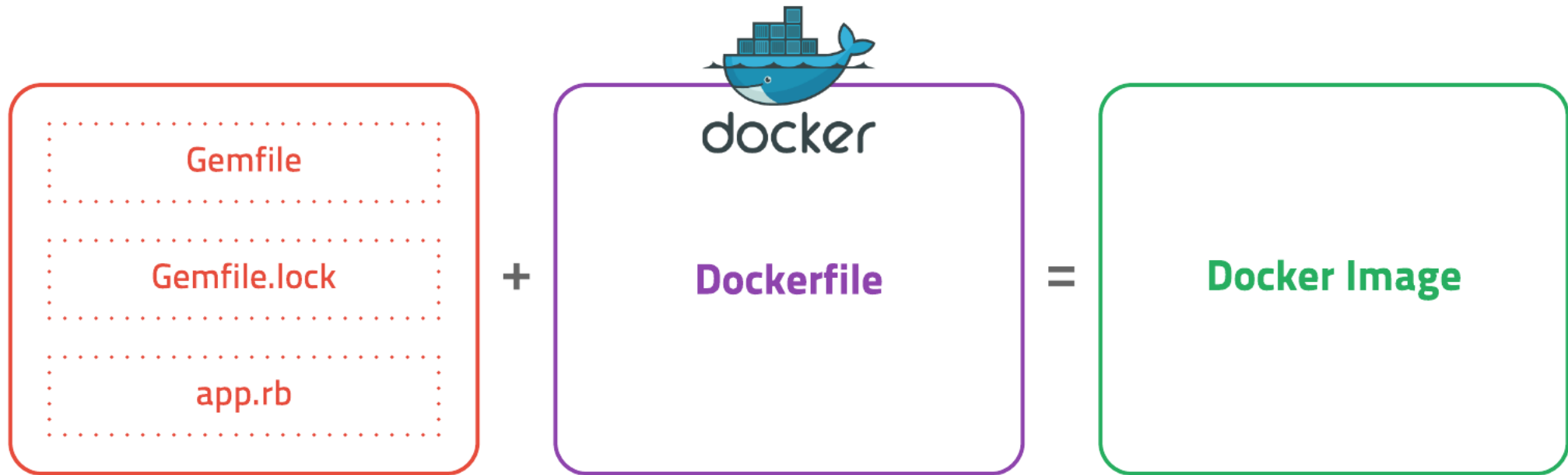
■ Docker 이미지 생성

- 컨테이너의 상태를 그대로 이미지로 저장



■ Docker 이미지 생성

- Application file + **Dockerfile**



■ Dockerfile

- 이미지 빌드용 DSL(Domain Specific Language)

■ Docker 이미지 생성

■ Dockerfile 작성

1) Dockerfile 생성

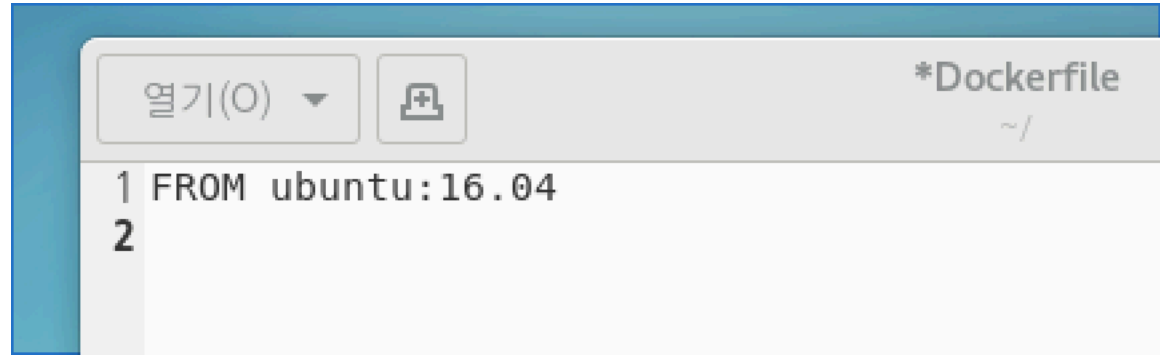
`$ touch Dockerfile`

2) FROM

`$ gedit Dockerfile`

3) 빌드

`$ docker build -t fromtest:0.0 .`



```
[admin@centos7 ~]$ docker build -t romtest:0.0 .  
Sending build context to Docker daemon 867.1MB  
Step 1/1 : FROM ubuntu:16.04  
16.04: Pulling from library/ubuntu  
35b42117c431: Pull complete  
ad9c569a8d98: Pull complete  
293b44f45162: Pull complete  
0c175077525d: Pull complete  
Digest: sha256:a4d8e674ee993e5ec88823391de828a5e9286a1597b731eaecaaf9066cfd539  
Status: Downloaded newer image for ubuntu:16.04  
--> 13c9f1285025  
Successfully built 13c9f1285025  
Successfully tagged romtest:0.0
```

`$ docker images`

■ Docker 이미지 생성

■ Dockerfile 작성

4) RUN command 추가

- *bash shell 명령어 추가*

5) 빌드

\$ docker build -t runtest:0.0 .

```
Dockerfile
~/
열기(O) ▼
1 FROM ubuntu:16.04
2 RUN mkdir /mydata
3 RUN echo "Hello, Docker!"
4
```

```
[admin@centos7 ~]$ docker build -t runtest:0.0 .
Sending build context to Docker daemon 867.1MB
Step 1/3 : FROM ubuntu:16.04
--> 13c9f1285025
Step 2/3 : RUN mkdir /mydata
--> Running in 14e0adf98b47
Removing intermediate container 14e0adf98b47
--> 7989a8a7c405
Step 3/3 : RUN echo "Hello, Docker!"
--> Running in 7d40f263e424
Hello, Docker!
Removing intermediate container 7d40f263e424
--> 318b4bc1c3fd
Successfully built 318b4bc1c3fd
Successfully tagged runtest:0.0
[admin@centos7 ~]$
```

■ Docker 이미지 생성

■ Dockerfile 작성

\$ docker run -it --name runtest runtest:0.0

```
[admin@centos7 ~]$ docker run -it --name runtest runtest:0.0
root@081d5b715e4d: /# ls -al
total 4
drwxr-xr-x.  1 root root    6 Jun 20 03:05 .
drwxr-xr-x.  1 root root    6 Jun 20 03:05 ..
-rwxr-xr-x.  1 root root    0 Jun 20 03:05 .dockerenv
drwxr-xr-x.  2 root root 4096 Jun 10 20:41 bin
drwxr-xr-x.  2 root root    6 Apr 12  2016 boot
drwxr-xr-x.  5 root root  360 Jun 20 03:05 dev
drwxr-xr-x.  1 root root   66 Jun 20 03:05 etc
drwxr-xr-x.  2 root root    6 Apr 12  2016 home
drwxr-xr-x.  8 root root   96 Sep 13  2015 lib
drwxr-xr-x.  2 root root   34 Jun 10 20:41 lib64
drwxr-xr-x.  2 root root    6 Jun 10 20:40 media
drwxr-xr-x.  2 root root    6 Jun 10 20:40 mnt
drwxr-xr-x.  2 root root    6 Jun 20 03:04 mydata
drwxr-xr-x.  2 root root    6 Jun 10 20:40 opt
```

■ Docker 이미지 생성


■ Dockerfile 작성

6) ADD command 추가

- 테스트용 파일 생성 (ex. test.txt)

7) 빌드

`$ docker build -t addtest:0.0 .`

```
열기(O) ▼   
1 FROM ubuntu:16.04  
2 ADD test.txt /  
3  
4
```

```
[admin@centos7 ~]$ docker build -t addtest:0.0 .  
Sending build context to Docker daemon 867.1MB  
Step 1/2 : FROM ubuntu:16.04  
--> 13c9f1285025  
Step 2/2 : ADD test.txt /  
--> 694fa902ec6a  
Successfully built 694fa902ec6a  
Successfully tagged addtest:0.0  
[admin@centos7 ~]$
```

```
[admin@centos7 ~]$ docker run -it --name addtest addtest:0.0  
root@f62e24ad3703: /# ls  
bin    dev    home  lib64  mnt    proc   run    srv    test.txt  usr  
boot   etc    lib   media  opt    root   sbin   sys    tmp      var  
root@f62e24ad3703: /#
```

■ Docker 이미지 생성

■ Dockerfile 작성

8) ENTRYPOINT command 추가

- 테스트용 파일 생성 (ex. test.sh)

7) 빌드

`$ docker build -t entrytest:0.0 .`

```
열기(O) ▼ [icon]
```

```
1 FROM ubuntu:16.04
2 ADD test.sh /
3 RUN chmod +x /test.sh
4 ENTRYPOINT /test.sh
5 |
```

```
[admin@centos7 ~]$ docker build -t entrytest:0.0 .
Sending build context to Docker daemon 867.1MB
Step 1/4 : FROM ubuntu:16.04
---> 13c9f1285025
Step 2/4 : ADD test.sh /
---> Using cache
---> 77c89d0d72b5
Step 3/4 : RUN chmod +x /test.sh
---> Using cache
---> 961cb4f5c077
Step 4/4 : ENTRYPOINT /test.sh
---> Running in 6439e3c0b018
Removing intermediate container 6439e3c0b018
---> 6006685c3c3b
Successfully built 6006685c3c3b
Successfully tagged entrytest:0.0
[admin@centos7 ~]$
```

■ Docker 이미지 생성

■ Dockerfile 작성

8) 실행

\$ docker run -it --name entrytest entrytest:0.0


```
[admin@centos7 ~]$ docker run -it --name entrytest entrytest:0.0  
Hello, Docker  
[admin@centos7 ~]$
```

■ Docker 이미지 생성

■ Dockerfile 작성 순서

1. OS 설치 (Ubuntu or Centos)
2. ~~DB 설치 (Mysql)~~ → **Mysql docker container 사용**
3. ~~JAVA 설치~~ → **Base image에 포함**
4. ~~Tomcat 설치~~ → **Base image에 포함**
5. Application 파일 복사
6. Tomcat 서버 실행

■ Docker 이미지 생성

```
열기(O) ▾   
1 FROM luis cortes/tomcat8.5-ora8jdk-alpineC  
2  
3 ENV TZ=Asia/Seoul  
4 RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime  
5 RUN echo $TZ > /etc/timezone  
6 RUN rm -Rf /usr/local/tomcat/webapps/ROOT  
7 COPY target/post-1.0.0.war /usr/local/tomcat/webapps/ROOT.war  
8 ENV JAVA_OPTS="-Dserver.type=dev"  
9
```

\$ docker build -t *simple-app:1.0* .

\$ docker run -it --name *simple-app-1* simple-app:1.0

\$ docker exec -it *simple-app-1* /bin/bash

→ ip address 확인

■ Docker 이미지 생성

■ Tomcat → Mysql Connection

- \$ docker inspect <tomcat image>
- \$ docker inspect <mysql image>

```
"Networks": {  
  "bridge": {  
    "IPAMConfig": null,  
    "Links": null,  
    "Aliases": null,  
    "NetworkID": "7e034e64994f3e307171ba0",  
    "EndpointID": "230df0c787699b0b7d902f",  
    "Gateway": "172.17.0.1",  
    "IPAddress": "172.17.0.2",  
    "IPPrefixLen": 16,  
    "IPv6Gateway": "",  
    "MacAddress": ""  
  }  
}
```

■ application.properties 파일

```
spring.datasource.url=jdbc:mysql://172.17.0.2:3306/testdb?serverTimezone=UTC
```

■ Docker 이미지 생성

- Tomcat → Mysql Connection

