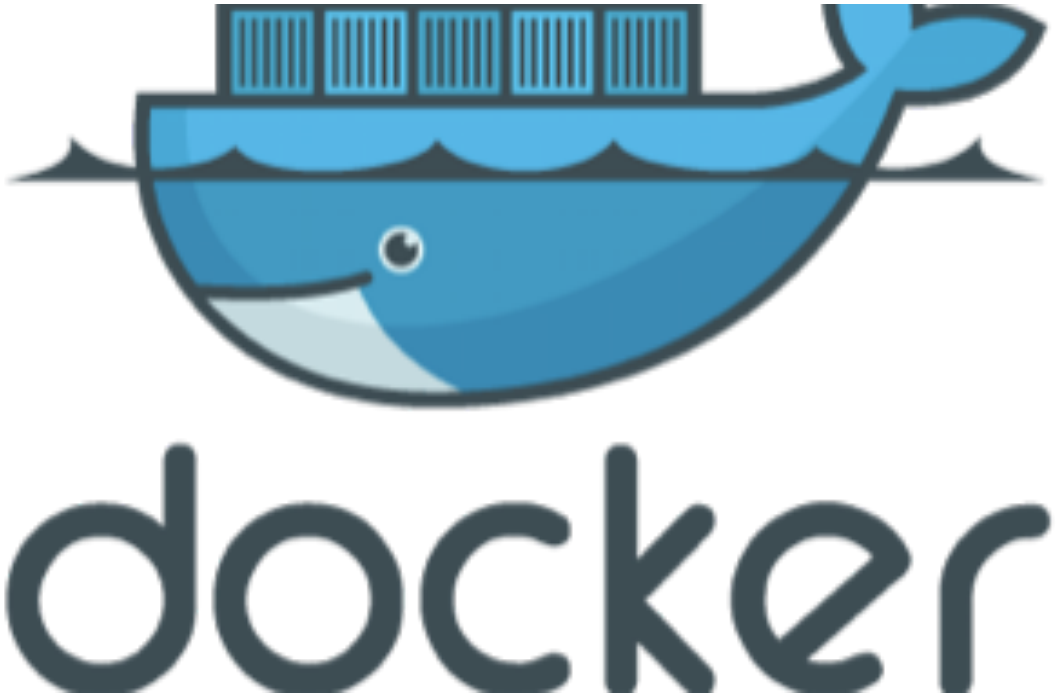


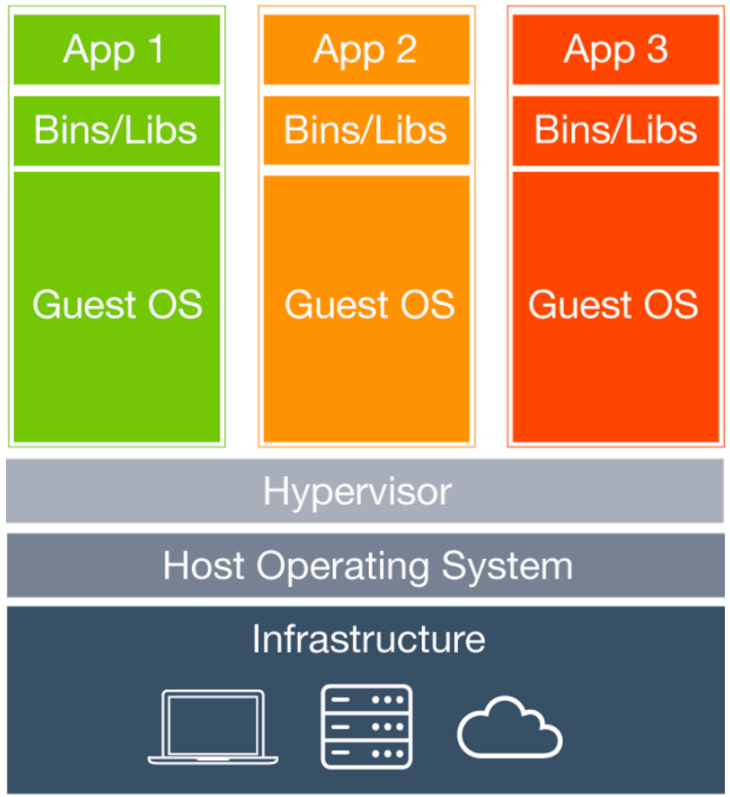
HEYMAN AI
A GOOD BANK FOR GOOD PEOPLE

WHAT IS DOCKER?

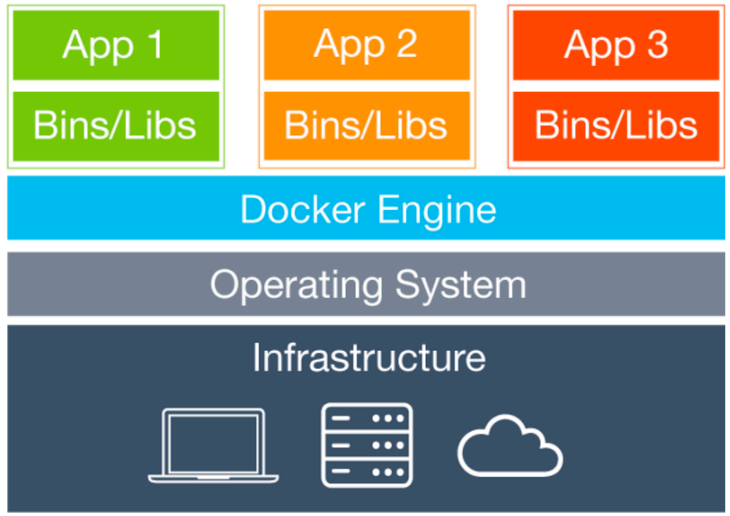
- **Docker is a container management service**. The keywords of Docker are develop, ship and run anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

- Docker is an Open platform for developers and sysadmins to build, ship and run distributed applications.

- It can run on most Linux distributions, Windows and Mac OS running Docker Engine (Toolbox).



Virtual Machines



Containers

DOCKER FEATURES….

- Light-Weight
- Minimal overhead (cpu/io/network)
- Based on Linux containers
- Decrease storage consumption
- Uses layered filesystem to save space (AUFS/LVM)
- Portable
- Run it Everywhere! - Linux, Mac OS or Windows operating system that has Docker installed.
- Raspberry pi support.
- Move from one environment to another by using the same Docker technology.
- Self-sufficient
- A Docker container contains everything it needs to run
- Minimal Base OS
- Libraries and frameworks
- Application code
- A Docker container should be able to run anywhere that Docker can run.

IMAGES - CONTAINERS

- Docker **images** are executable packages that include everything needed to run an application — the code, a runtime, libraries, environment variables, and configuration files.

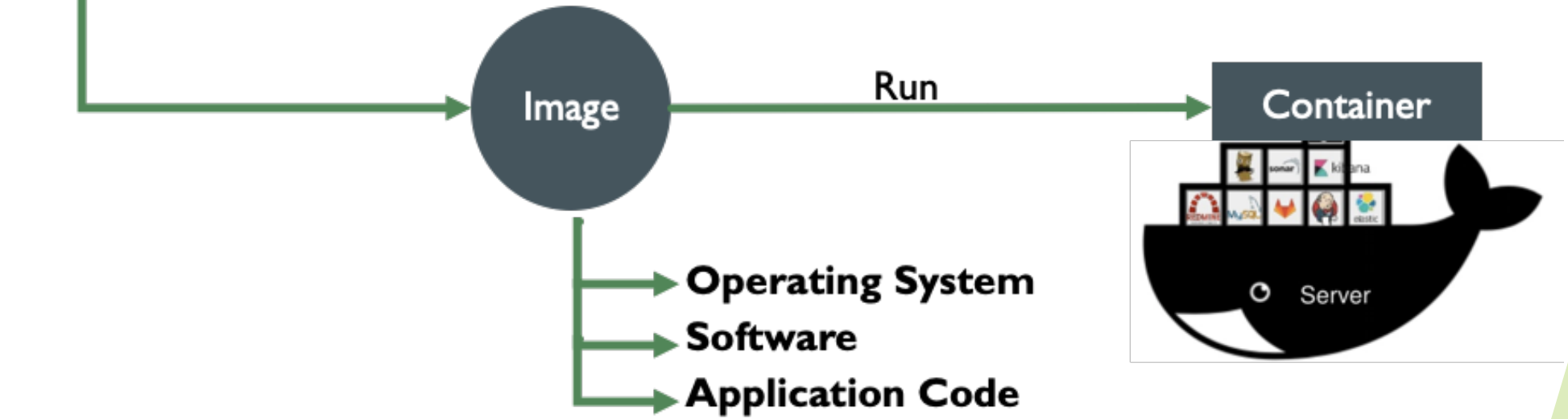
- Docker **containers** are a runtime instance of an image — what the image becomes in memory when executed (that is, an image with state, or a user process).

DOCKER FLOW

A- Pull images from docker hub (docker pull <image name>)

B- Docker file (dockerfile)

C- Docker compose (docker-compose.yml)



MOST USED COMMANDS

- **docker version**

```
Mutlus-MacBook-Pro:~ mokuducu$
Mutlus-MacBook-Pro:~ mokuducu$ docker version
Client: Docker Engine - Community
 Version:           19.03.2
 API version:       1.40
 Go version:        go1.12.8
 Git commit:        6a30dfc
 Built:             Thu Aug 29 05:26:49 2019
 OS/Arch:           darwin/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.2
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.8
  Git commit:       6a30dfc
  Built:            Thu Aug 29 05:32:21 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          v1.2.6
  GitCommit:        894b81a4b802e4eb2a91d1ce216b8817763c29fb
 runc:
  Version:          1.0.0-rc8
  GitCommit:        425e105d5a03fabd737a126ad93d62a9eeede87f
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
Mutlus-MacBook-Pro:~ mokuducu$
```

A-DOCKER HUB

- **Docker Hub** is a service provided by **Docker** for finding and sharing container images with your team.

- https://hub.docker.com/

Docker Pull

- **docker pull < image name>**

docker pull nginx

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	f949e7d76d63	2 weeks ago	126MB

- **docker run --name nginx -d -p 8080:80 nginx**
- Mutlus-MacBook-Pro:~ mokuducu$ docker ps –a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
85a54b8a660c	nginx	"nginx -g 'daemon of…"	About an hour ago	Up 37 minutes	0.0.0.0:8080->80/tcp	nginx

IMAGES COMMANDS

Commands	
➢ Docker images	image list
➢ Docker images -q	q – It tells the Docker command to return the Image ID's only.
➢ Docker run <images >	Imaged run details
➢ Docker rmi -f <image id>	Force image delete
➢ Docker history <ImageID>	History of image
➢ Docker inspect <imageid>	Details of images

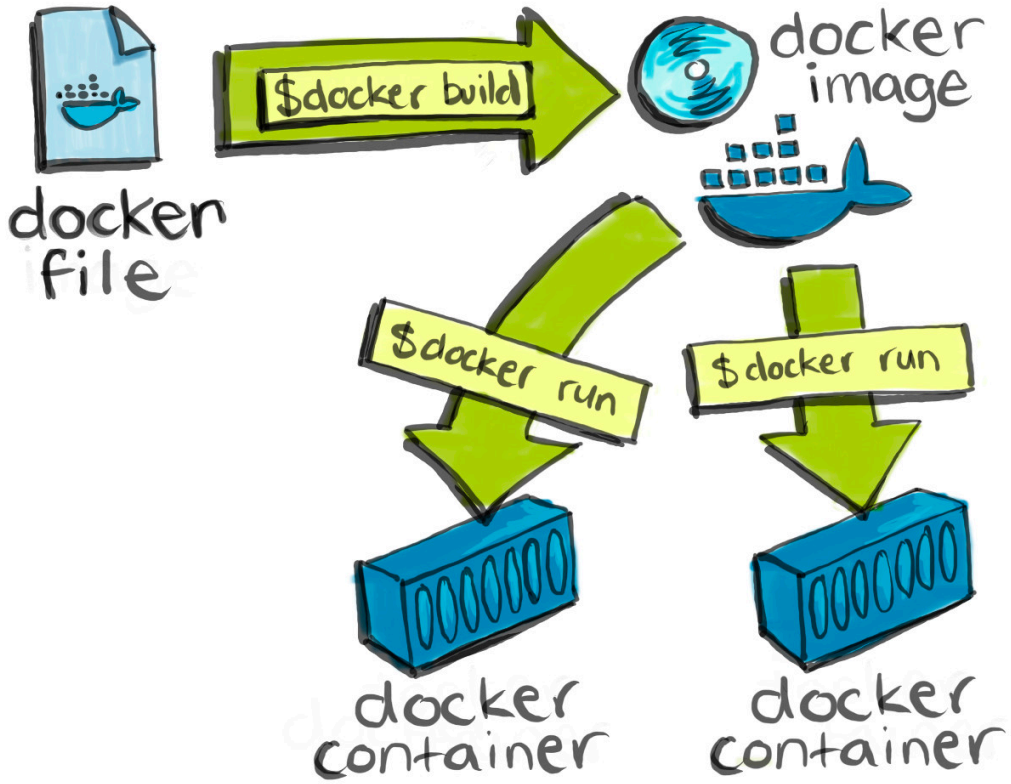
CONTAINER COMMANDS

Commands	
➢ Docker inspect <imageid containerid>	Details of container
➢ Docker stop	Docker stop
➢ Docker logs <containerid>	Image logs
➢ Docker exec <exqute command container>	Docker exequte
➢ Docker history <ImageID >	History of image
➢ docker top <ContainerID >	Top container
➢ docker stats <ContainerID >	Status of container
➢ Docker rm -f <container id>	Force image delete
➢ docker network ls	List of network
➢ docker network inspect <networkname >	Details of network
➢ docker stop $(docker ps -a -q)	Stop all container
➢ docker rm $(docker ps -a -q)	Remove all container

B- DOCKER FILE

Dockerfile – Text file (recipe) used to create Docker images

Docker file name must be Dockerfile



HEYMAN AI
A GOOD BANK FOR GOOD PEOPLE

Docker File Commands

ADD copies the files from a source on the host into the container's own filesystem at the set destination.

CMD can be used for executing a specific command within the container.

ENTRYPOINT sets a default application to be used every time a container is created with the image.

ENV sets environment variables.

EXPOSE associates a specific port to enable networking between the container and the outside world.

FROM defines the base image used to start the build process.

MAINTAINER defines a full name and email address of the image creator.

RUN is the central executing directive for Dockerfiles.

USER sets the UID (or username) which is to run the container.

VOLUME is used to enable access from the container to a directory on the host machine.

WORKDIR sets the path where the command, defined with CMD, is to be executed.

LABEL allows you to add a label to your docker image.

Build Docker File

```
# Ubuntu

FROM ubuntu
MAINTAINER demousr@gmail.com

RUN apt-get update
RUN apt-get install –y nginx
CMD [“echo”,”Image created”]
```

- **docker build .**

- •The first line "#This is a sample Image" is a comment.
- •The next line has to start with the FROM keyword. It tells docker, from which base image you want to base your image from. In our example, we are creating an image from the ubuntu image.
- •The next command is the person who is going to maintain this image. Here you specify the MAINTAINER keyword and just mention the email ID.
- •The RUN command is used to run instructions against the image. In our case, we first update our Ubuntu system and then install the nginx server on our ubuntu image.
- •The last command is used to display a message to the user.

C- DOCKER-COMPOSE

Docker Compose is used to run multiple containers as a single service. For example, suppose you had an application which required NGNIX and Redis, you could create one file

Create Docker-Compose File at any location on your system

- **All Docker Compose files are YAML**
- **File name docker-compose.yml**

Install Docker Compose:

- **From**

  github (https://github.com/docker/compose/releases**)**

- **Using PIP**

  Pip install-U docker-compose

•Check docker-compose

docker-compose version (-v)

Check the validity of file by command

- Docker-compose config

```
Mutlus-MacBook-Pro:dockercompose1 mokuducu$ docker-compose config
services:
  database:
    image: redis
  web:
    image: nginx
version: '3.0'
```

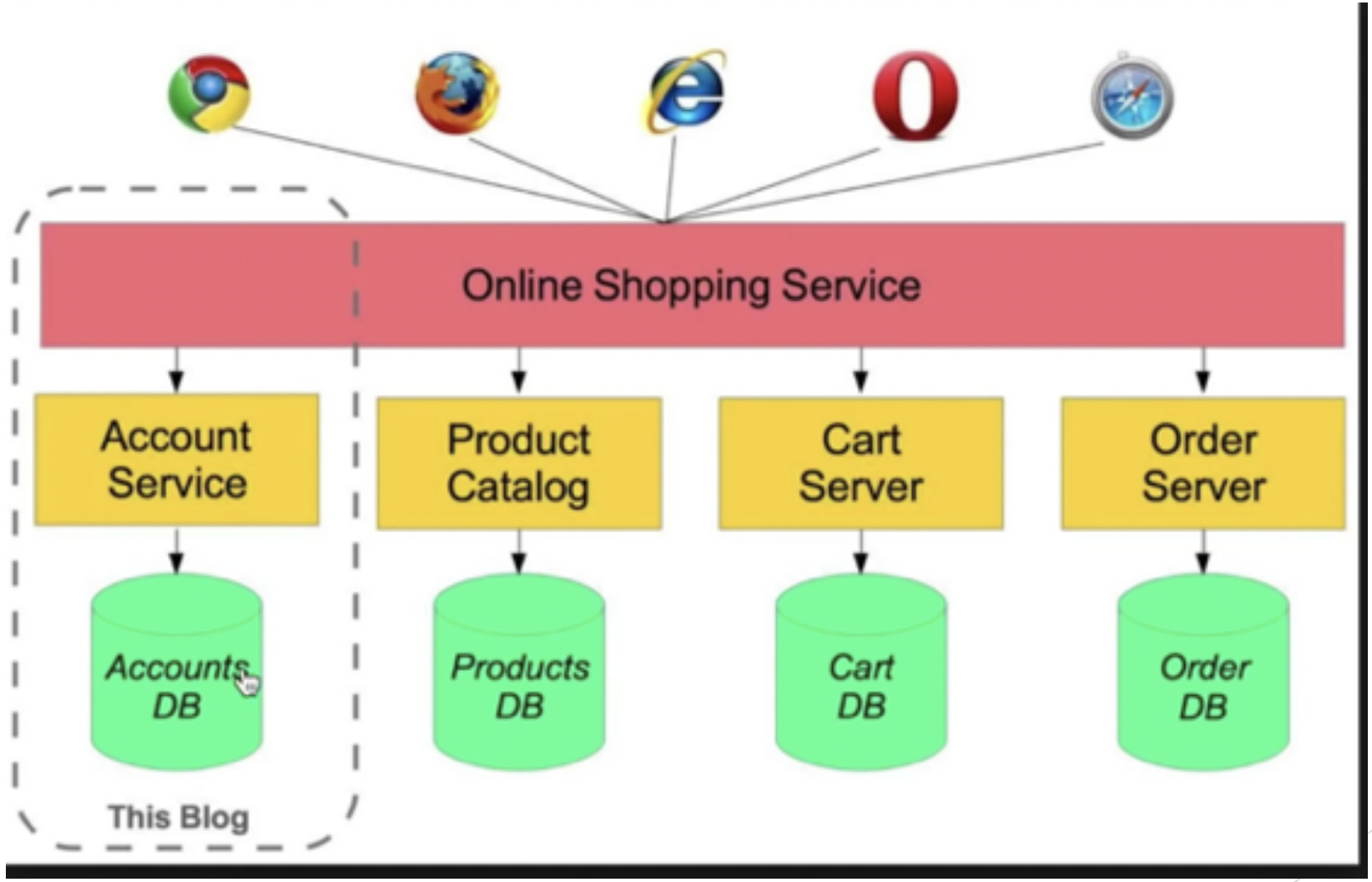
Run docker-compose.yml file by command

- Docker-compose up –d (d Detached mode: Run containers in the background, print new container names. Incompatible with --abort-on-container-exit.)

Down Docker compose

- Docker-compose down

Microservices



Scale

scale Set number of containers for a service

docker-compose up -d --scale <service name>=number of instance

- docker-compose up -d --scale database=4

COMPOSE COMMANDS

Commands	
➢ **docker-compose -help**	**Help compose commands**
➢ **docker-compose version (-v)**	**Compose version**
➢ Docker-compose config	**Validate and view the Compose file**
➢ Docker-compose up -d	**Create and start containers**
➢ Docker-compose down	**Stop and remove containers, networks, images, and volumes**
➢ docker-compose up -d --scale <service name>=number of instance	Set number of containers for a service

Docker-compose Examples

```
#mysql and adminer

version: '3.1'
services:

  db:
    image: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root

  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080
```