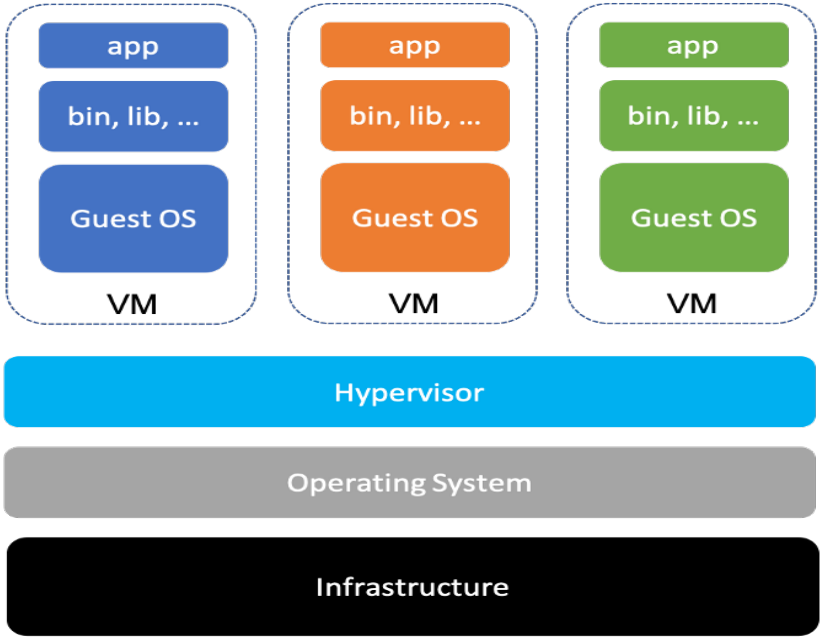
Docker

Sabreen Salama

Content

Virtualization

Virtualization is the process of running a virtual instance of a computer system in a layer abstracted from the actual hardware. Most commonly, it refers to running multiple operating systems on a computer system simultaneously.



(a) Virtual Machine architecture

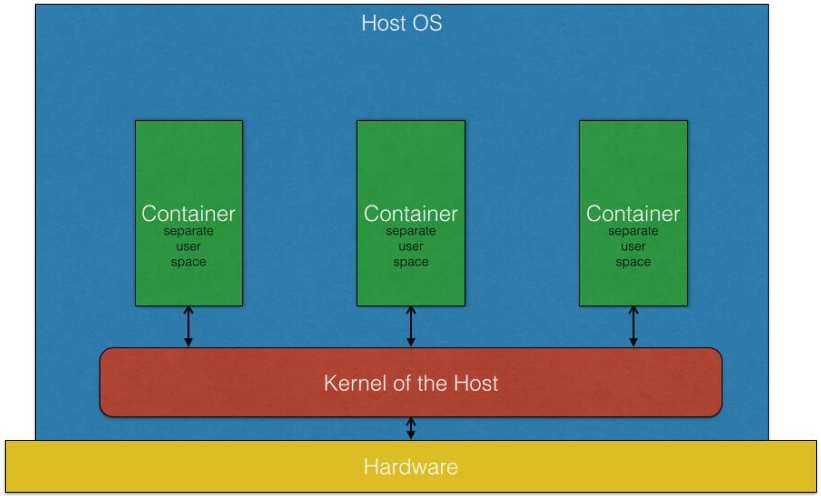
Disadvantages of virtualization

Containerization

A container is A software that packages up code and all its dependencies so the application runs quickly and reliably

from one computing environment to anoth

- it shares the same kernel
- Advantages of containers:
  - less CPU& memory usage
  - less hard space
  - takes seconds to boot up



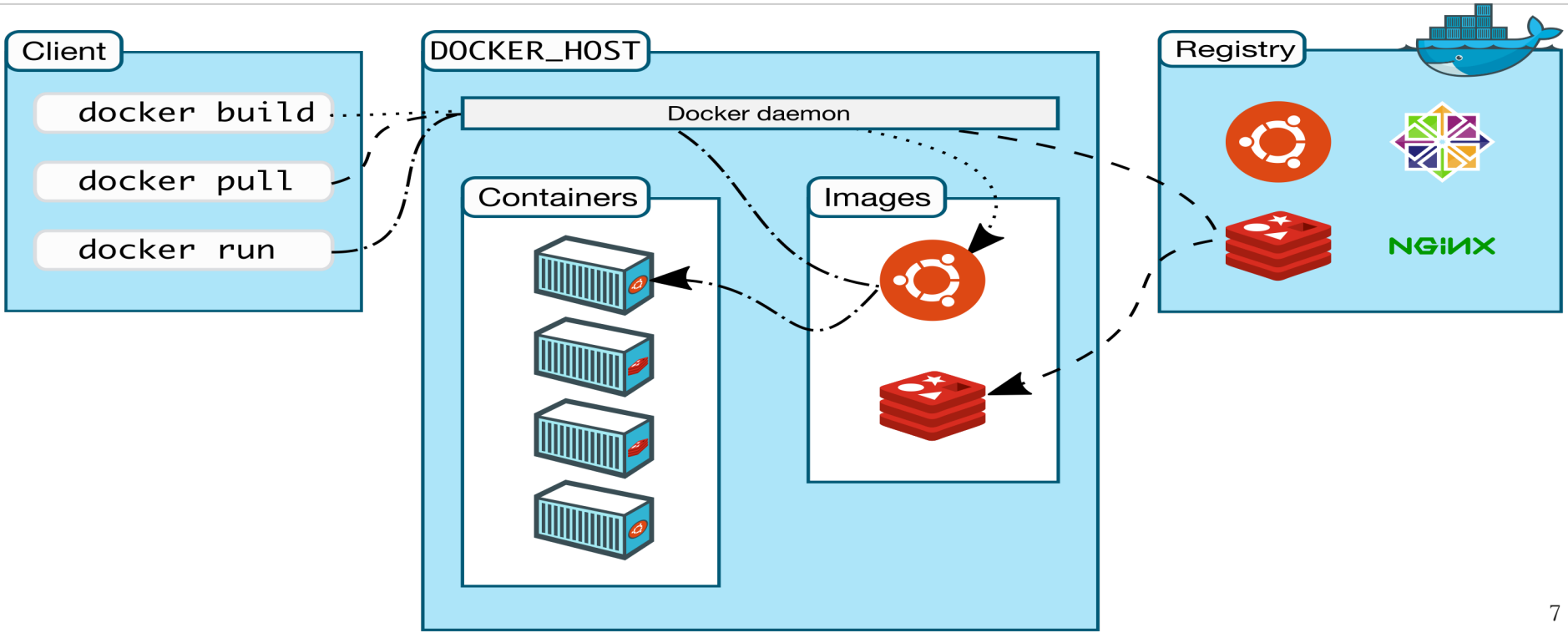
Operating System/Container Virtualization

Docker

Docker is an open platform for developing, shipping, and running applications.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker architecture



Docker Images

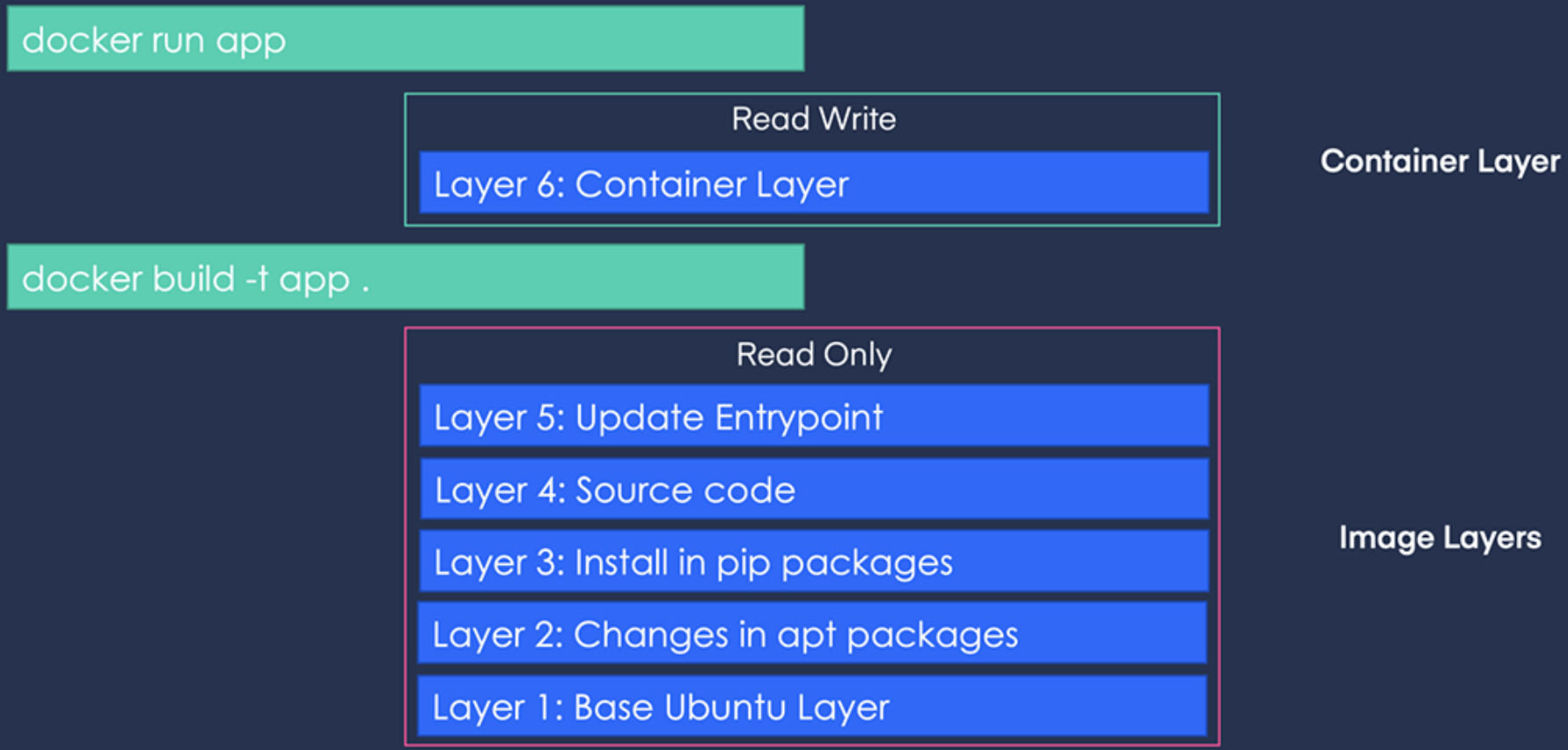
- An *image* is a read-only template with instructions for creating a Docker container
- It'smade up of a collection of layers that bundle together all the essentials – such as **installations**, **application code**, and **dependencies** – required to configure a fully operational container environment
- Docker users store images in a private or public repositories,

and from there can deploy containers, test images and share

Deafult repositoey is dockerhub

Docker Containers

- A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.
- docker creates a writable layer above the read only layers for image
- When a container is removed, any changes to its state that are not stored in persistent storage disappear


docker run app
Read Write
Layer 6: Container Layer
Container Layer
docker build -t app .
Read Only
Layer 5: Update Entrypoint
Layer 4: Source code
Layer 3: Install in pip packages
Layer 2: Changes in apt packages
Layer 1: Base Ubuntu Layer
Image Layers

Dealing with docker objects

- Start a container

  command: docker run <image-name>

  Ex : docker run nginx:latest
- List all running containers:

  command: docker ps
- List all running and stopped container

  command: docker ps –a
- Download an image:

  command: docker pull <image-name>

Contd:Dealing with docker objects

- Stop a container

  command: docker stop <container-name>
- Remove a container:

  command: docker rm <container-name>
- List images

  command: docker images
- Remove image:

  command: docker rmi <image-name>

Contd:Dealing with docker objects

- Append a command

  command: docker run <image-name> <command>

  ex: docker run ubuntu sleep 400

- Execute a command inside the container:

  command: docker exec <container-name> <command>

- Run – attach :

  command: docker run <image-name>

- Run - deatch

  command: docker run -d <image-name>

Docker run stdin

```
~/prompt-application$ ./app.sh
Welcome! Please enter your name: Mumshad

Hello and Welcome Mumshad!
```

```
docker run kodekloud/simple-prompt-docker

Hello and Welcome !
```

```
docker run -i kodekloud/simple-prompt-docker
Mumshad

Hello and Welcome Mumshad!
```

```
docker run -it kodekloud/simple-prompt-docker
Welcome! Please enter your name: Mumshad

Hello and Welcome Mumshad!
```

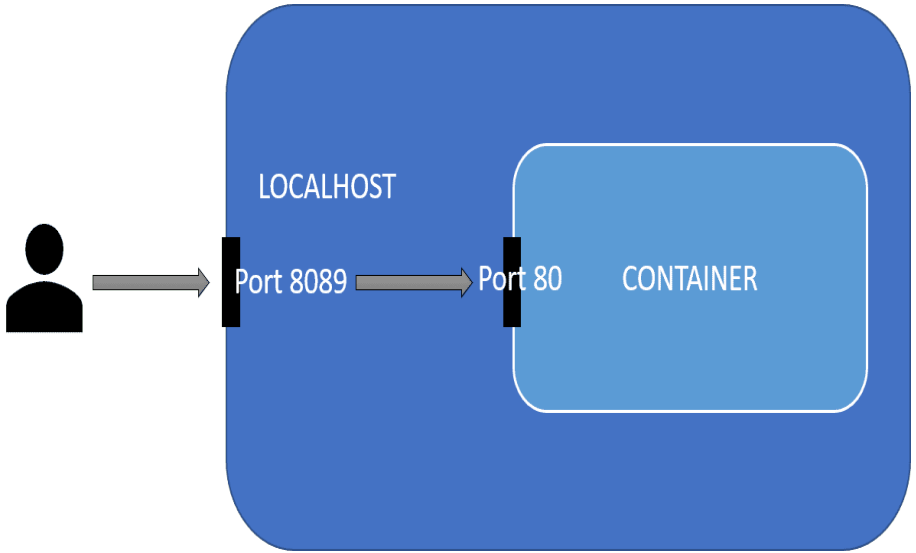
port mapping

To map a port:

docker run –d –p host-port:container-port <image-name>

Ex:

docker run -d -p 8080:80 nginx



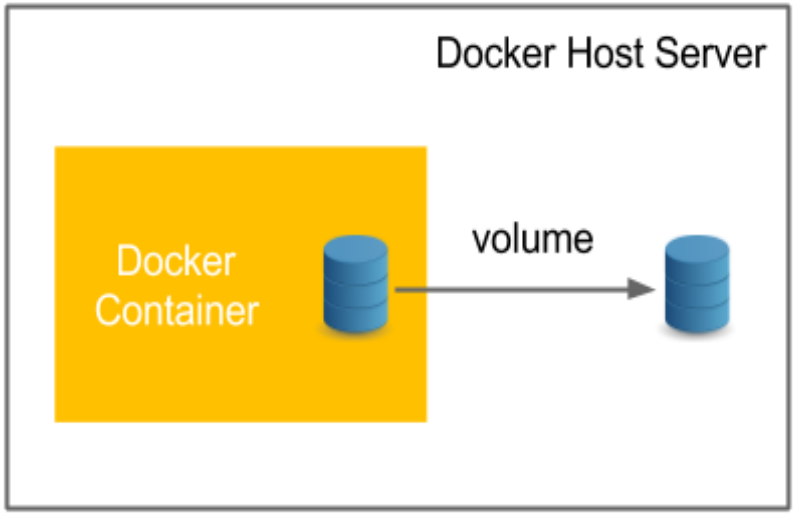
volume mapping

To map a volume:

docker run –d –v host-volume:container-volume <image-name>

Ex:

docker run -d -v /opt/data:/var/lib/mysql myql



Check containers

Inspect container to see all it's info:

docker inspect <container-name>

Check logs for container:

docker logs <container-name>

Inject env vars to container

You can inject environmental. Variables to containers

docker run -e APP_COLOR=blue simple-webapp-color

Create custom image

Dockerfile is a txt file contain instructions for creating image

When you issue a docker build command, the current working directory is called the build context.

docker build –t imagename .

To specify another context use –f <path>

Task:

Diff between Run. , CMD and entrypoint in dockerfile

Dockerfile instructions

Command	Overview
FROM	Specify base image
RUN	Execute specified command
ENTRYPOINT	Specify the command to execute the container
CMD	Specify the command at the time of container execution (can be overwritten)
COPY	Simple copy of files / directories from host machine to container image
ADD	COPY + unzip / download from URL (**not recommended**)
ENV	Add environment variables
EXPOSE	Open designated port
WORKDIR	Change current directory
MAINTAINER	**deprecated**
	now `LABEL maintainer="maintainer@example.com"`should be specified as

Docker registries

Attempt to authenticate against the private registry:

docker login <registry public hostname>

- You can log into any public or private repository for which you have credentials.

When you

- Push to and pull from your private registry:
- docker pull ubuntu
- docker tag ubuntu <registry public hostname>/ubuntu
- docker push <registry public hostname>/ubuntu
- docker image rm <registry public hostname>/ubuntu
- docker pull <registry public hostname>/ubuntu

Docker storage

- docker path is /var/lib/docker

Docker storage type:

- Non persistent
- Persistenet
  - volume
  - bind_mount

Contd:Volume Docker storage commands

- List all volumes on a host:

  command: docker volume ls

- Create volume:

  - command: docker volume create test-volume1

- Inspecting a volume:

  - command: docker volume inspect test-volume1

- Deleting a volume:

  - command: docker volume rm test-volume

- Removing all unused volumes:

  - command: docker volume prune

Docker network

- One of the reasons Docker containers and services are so powerful is that you can connect them together,
- Docker network drivers:
  - Bridge: The default network driver. If you don't specify a driver , are usually used when you need to connect many container to each others
  - None: The default network driver. If you don't specify a driver, For this container, disable all networking. Usually used in conjunction with a custom network driver.
  - Host: This enables a container to attach to your host's network (meaning the configuration inside the container matches the configuration outside the container).

Docker network commands

- List all networks on a host:
  - command: docker network ls
- Create network:
  - command: docker network create <network-name>
- Inspecting a network:
  - command: docker network inspect test-volume1
- Deleting a network
  - command: docker network rm <network-name>
- Removing all unused networks:
  - command: docker network prune

Contd Docker network commands

- List all networks on a host:
  - command: docker network connect <network-name> <container-name>

Examples

Docker network create test-network

Docker network connect test-network nginx

Docker compose

is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

- A Compose file is called docker-compose.yaml
- If you want to use another name use <–f nameOfFile>

```yaml
version: "3.9"  # optional since v1.27.0
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Contd Docker compose

- It is a yaml file but we have an option to use json as an alternative
- Docker-compose file has four types of top-level keys(version-services-networks-volumes):
  - Version: is mandatory , it is to define the compose file format and it is tight to version of docker engine.
  - Service: to define services definition. Each service definition represent a Container
  - Volumes: to define volumes that will be managed by compose.
  - Network: to define network that will be manged by compose, Bridge is set by default .

Contd Docker compose commands

- Create a compose service:

  docker-compose up -d

- List containers created by compose:

  docker-compose ps

- Stopping a compose service:

  docker-compose stop

- Starting a compose service:

  docker-compose start

- Restarting a compose service:

  docker-compose restart

Container orchestration

1- docker swarm : easy to install but not suitable for prod

2- mesos: difficult to install

3- k8s: easy to install and easy to handle load on prod and nonprod environments

Thanks!

Any questions ?

You can find me at

- https://www.linkedin.com/in/sabreen-salama-6abbb7107/