DevTech Training

Short Course – Day 2

Virtualization

(Play Video 01)

What is Virtualization ?

- Virtualization creates a virtual layer using the hypervisor software, which manages resources assigned to the virtual instances.
- The newly formed virtual representation is known as **Virtual Machines** (**VMs**)

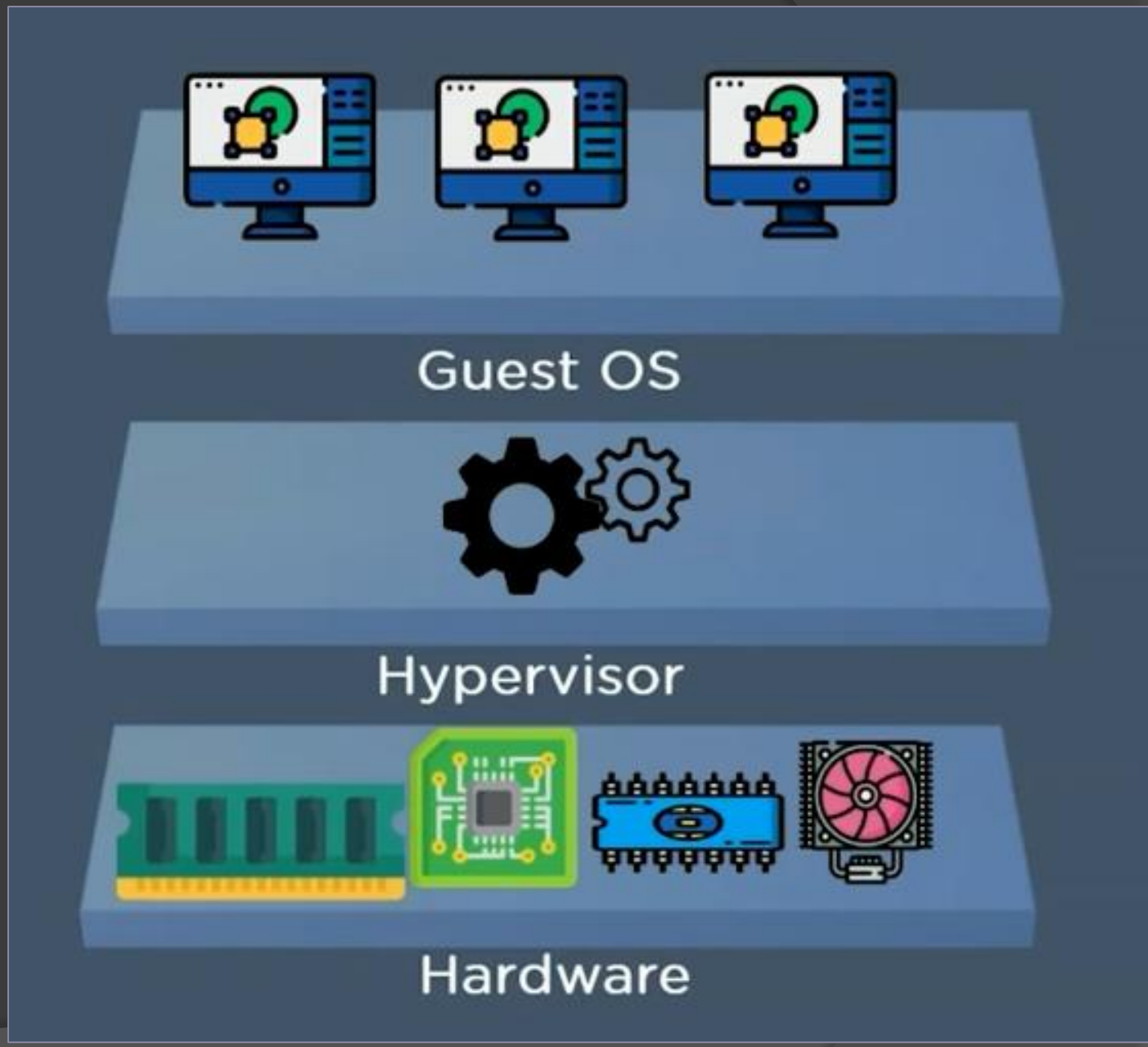
What is Virtual Machine (VM) ?

- **Virtual Machine** is an emulation or a virtual presentation of a physical system.
- They are also referred to as **Guest**, whereas the physical system they run on is referred to as the **Host**.

Role of Hypervisor

- **Hypervisor** is a software that manages VMs.
- It acts as an interface between VM and physical hardware to ensure proper access to the resources needed for working.

Role of Hypervisor

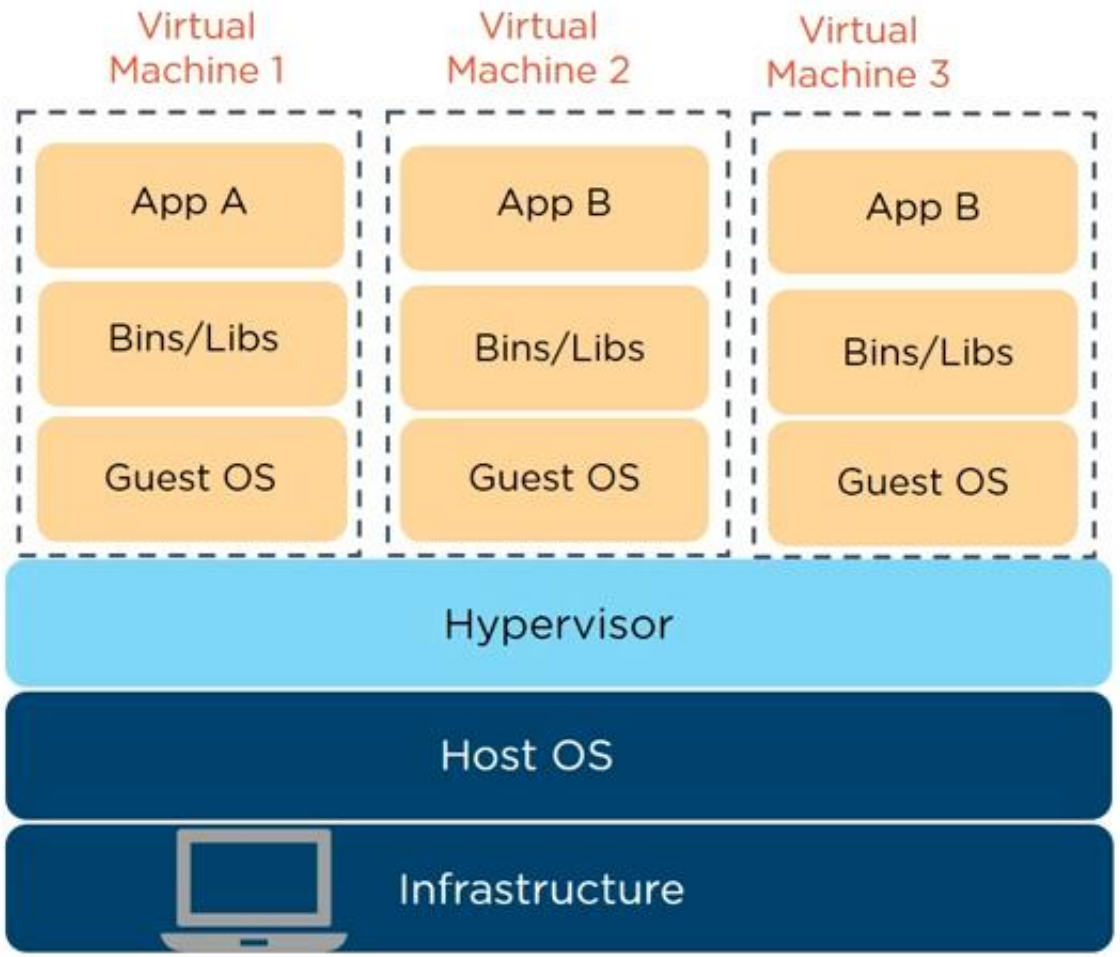


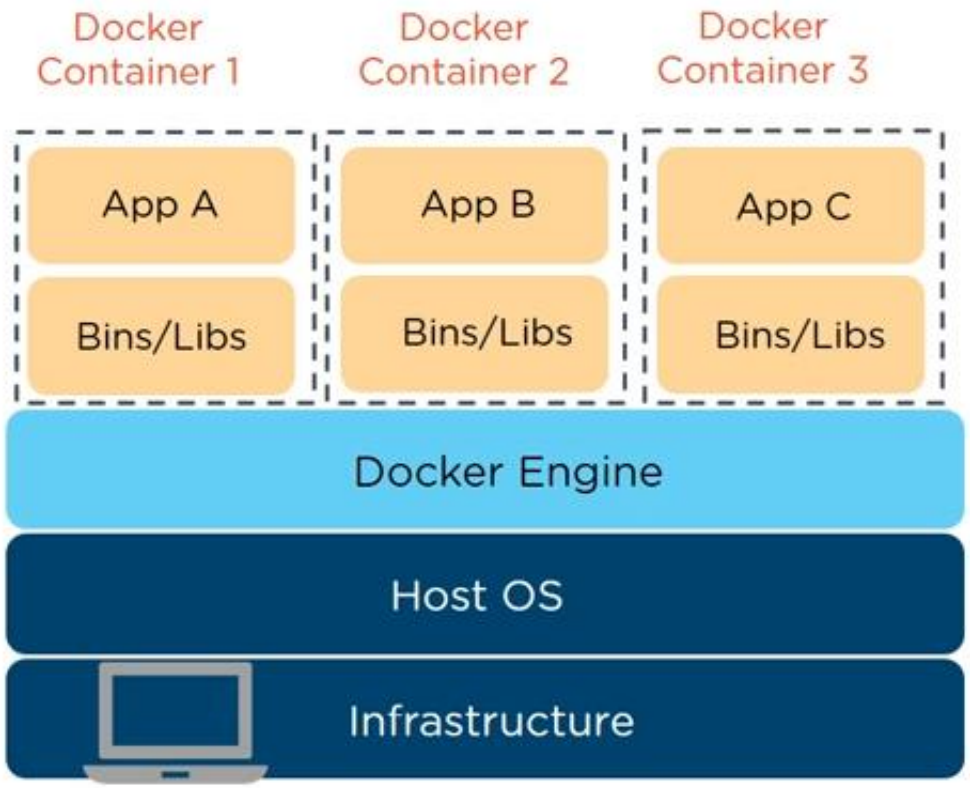
Benefits of Virtualization

- Resource efficiency, using virtualization the maximum computing capacity can be utilized.
- Minimum downtime, application and OS crash cases can be neglected by running multiple VMs with the same OS.
- Time management, setting up a whole server from scratch can be avoided by using sufficient hardware devices for virtualization.

Virtual Machine vs Docker

Virtual Machine vs Docker



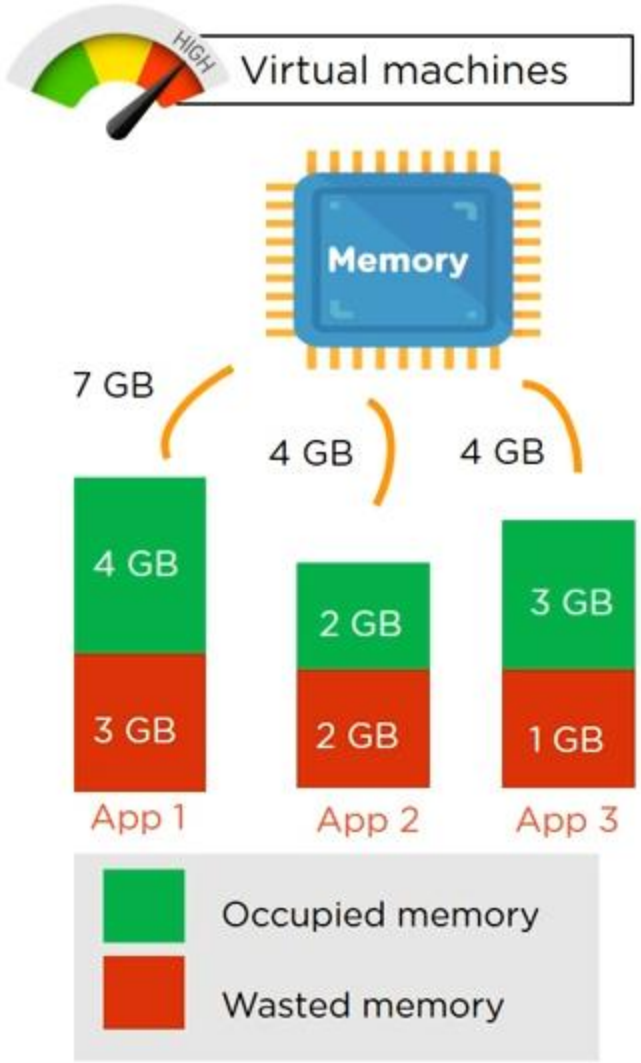


Virtual Machine vs Docker

Major differences are:

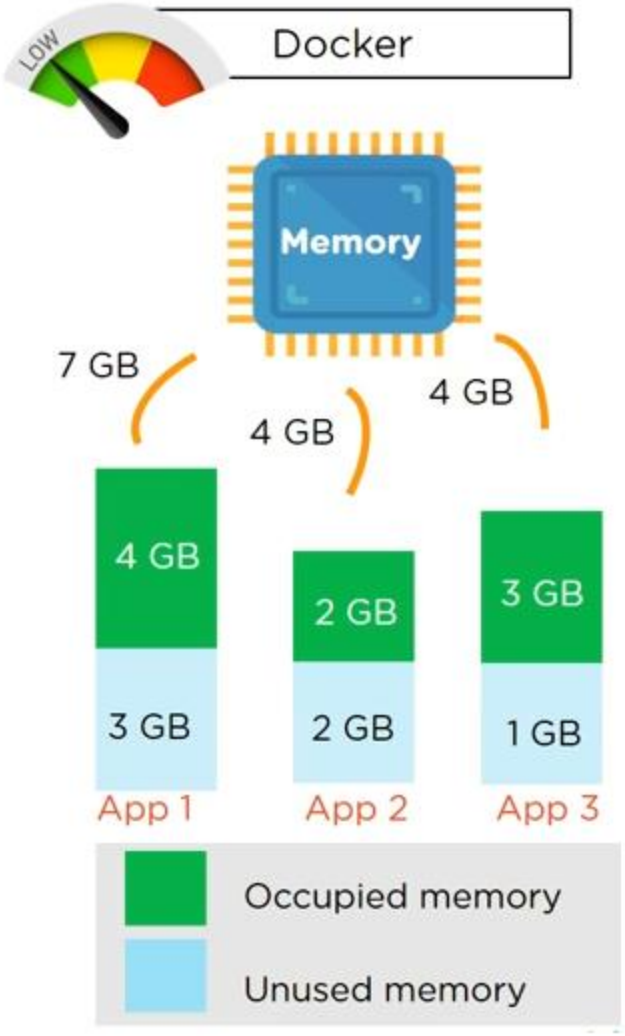
Virtual Machine		docker
HIGH	Memory usage	LOW
	Performance	
	Portability	
	Boot-up time	

Virtual Machine vs Docker - Memory Usage



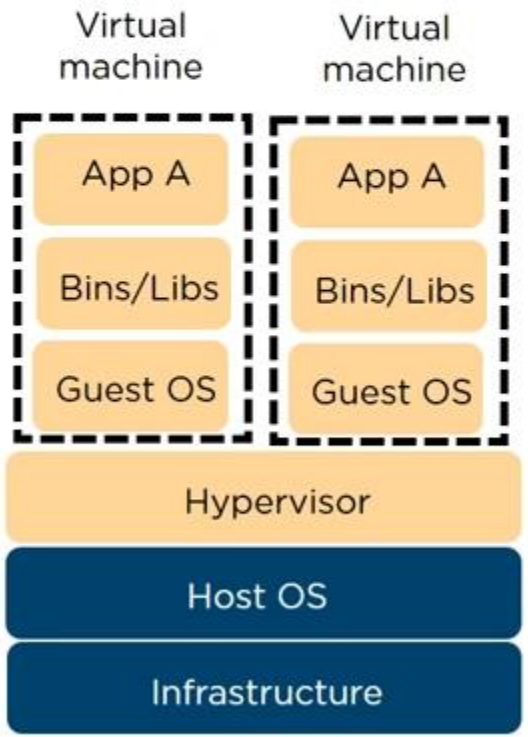
VM - Only 9 GB of memory is used whereas the remaining 6 GB of unused memory cannot be reused

Docker - Only 9 GB of memory is used whereas the remaining 6 GB of memory can be reused for a new container



Virtual Machine vs Docker - Performance

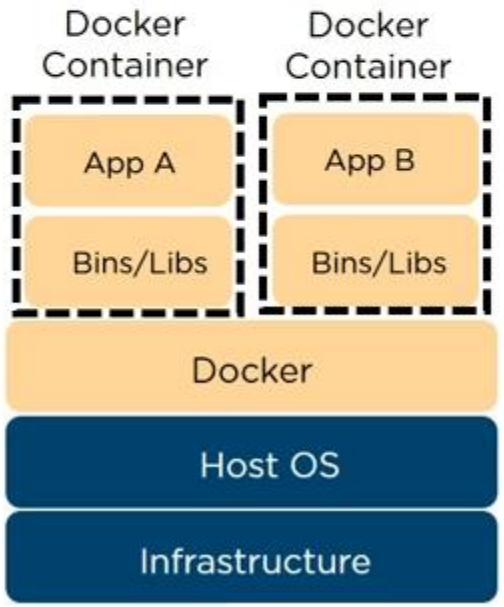
Virtual machines



VM - Running multiple virtual machines leads to unstable performance

Docker - Containers have a better performance as they are hosted on a single Docker engine

Docker



Virtual Machine vs Docker - Portability

Virtual machines

Software

Software works on system A

The same software doesn't work on system B

VM - Portability issues while executing applications in different platforms

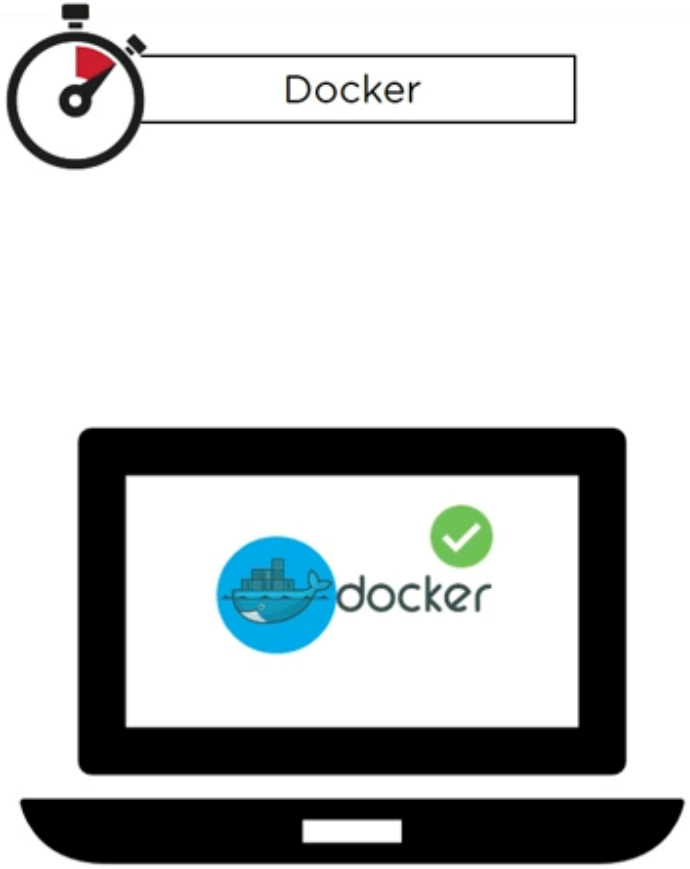
Docker – Multiple software can be encapsulated in a single container and can be easily deployed to different platforms

Docker

Virtual Machine vs Docker - Boot-up Time

VM – Takes long boot-up time (minutes)

Docker – Takes less boot-up time (milliseconds)



Docker

(Play Video 01)

What is Docker ?

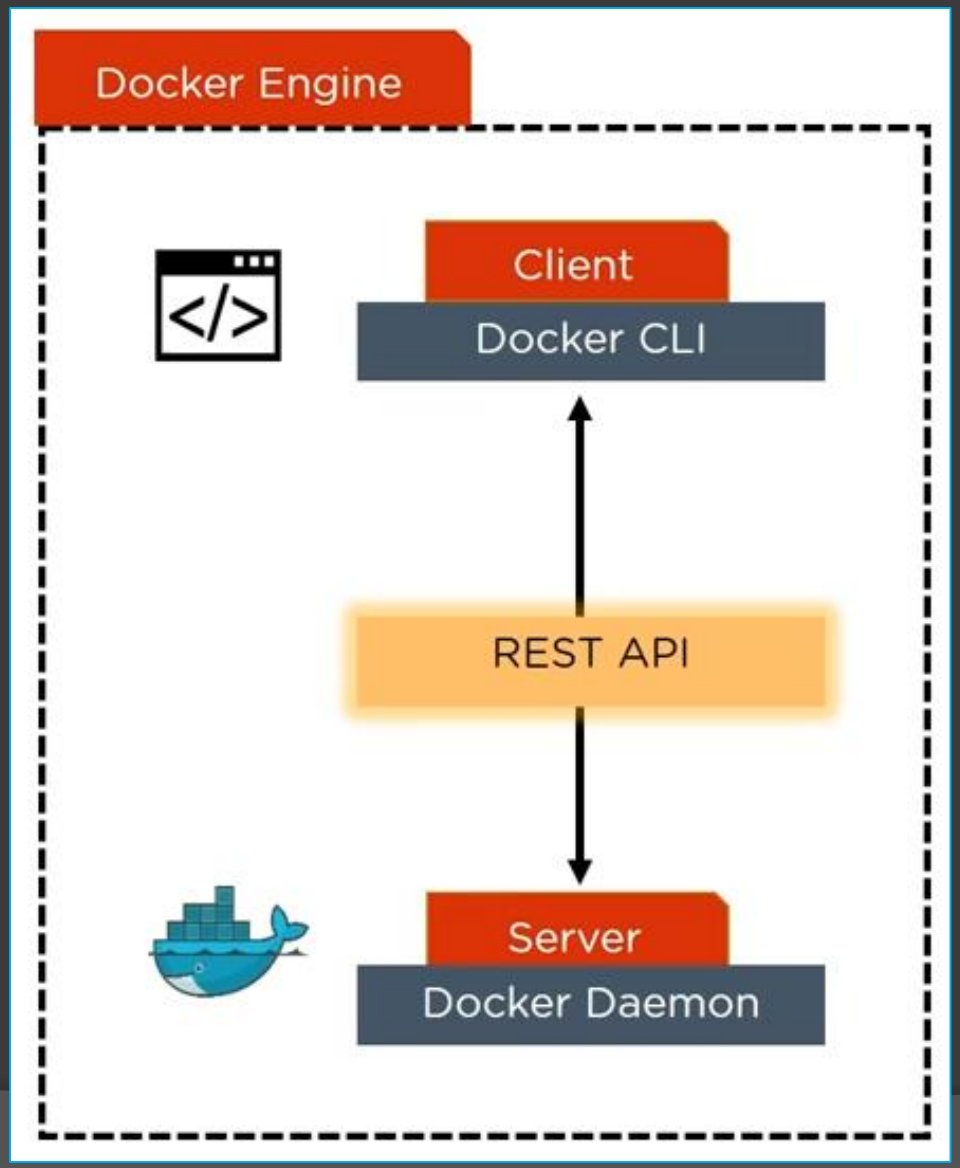
- **Docker** is a tool which is used to automate the deployment of applications in lightweight containers so that applications can work efficiently in different environments

- **Docker** is an OS-level virtualization software platform that enables developers and IT administrators to create, deploy and run applications in a Docker Container with all their dependencies

- Container is a software package that consists of all the dependencies (frameworks, libraries, etc…) required to execute and run an application

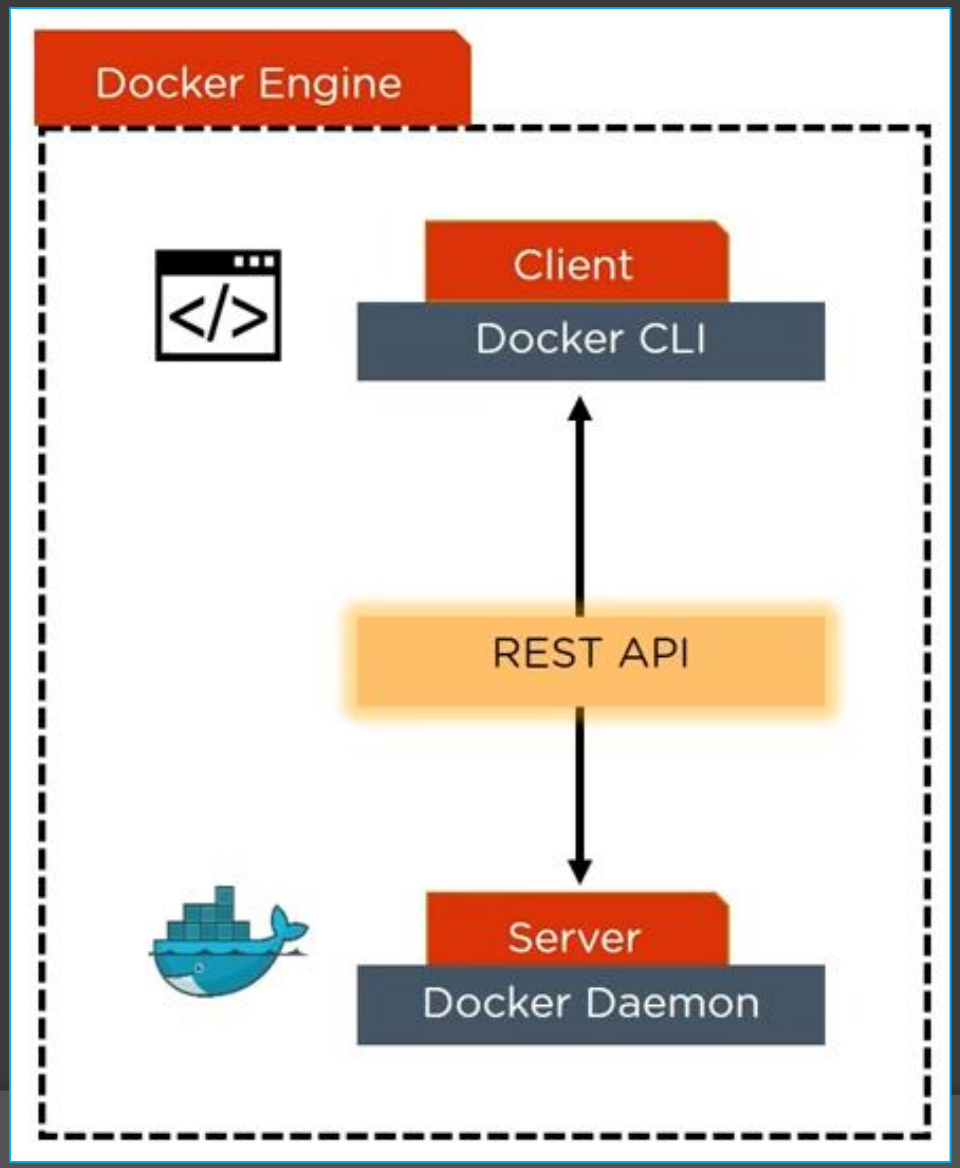
What is Docker ?

How does Docker work ?



- Docker Engine or Docker is the base engine installed on your host machine to build and run containers using Docker components and services
- It uses a client-server architecture
- Docker Client and Server communicates using REST API

How does Docker work ?



- Docker Client is a service which runs a command, and is translated using REST API, and is sent to the Docker Daemon (server)
- The Docker Daemon check the client request and interacts with the operating system in order to create or manage containers

Components of Docker

Components of Docker

Docker Client and Server

Docker Images

Docker Containers

Docker Registry

Components of Docker - Docker Client and Server

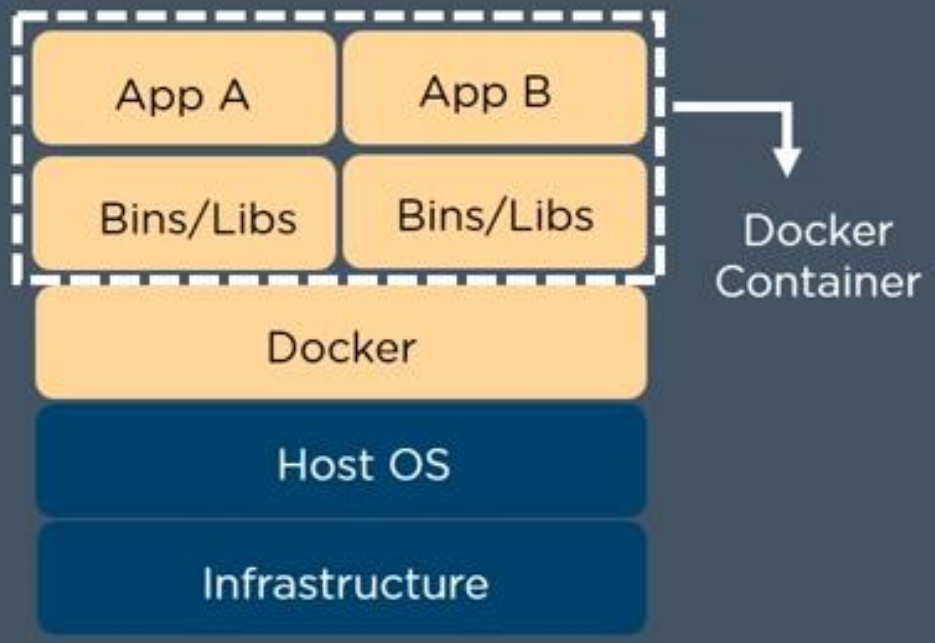
- Docker Client is accessed from the terminal and a Docker Host runs the Docker Daemon and registry
- A user can build Docker Images and run Docker Containers by passing commands from the Docker Client to the Docker Server

Components of Docker - Docker Image

- Docker Image is a template with instructions, which is used for creating Docker Containers
- A Docker Image is built using a file called Docker File
- Docker Image is stored in a Docker Hub or in a repository

Components of Docker - Docker Container

- Docker Container is a standalone, executable software package which includes applications and their dependencies



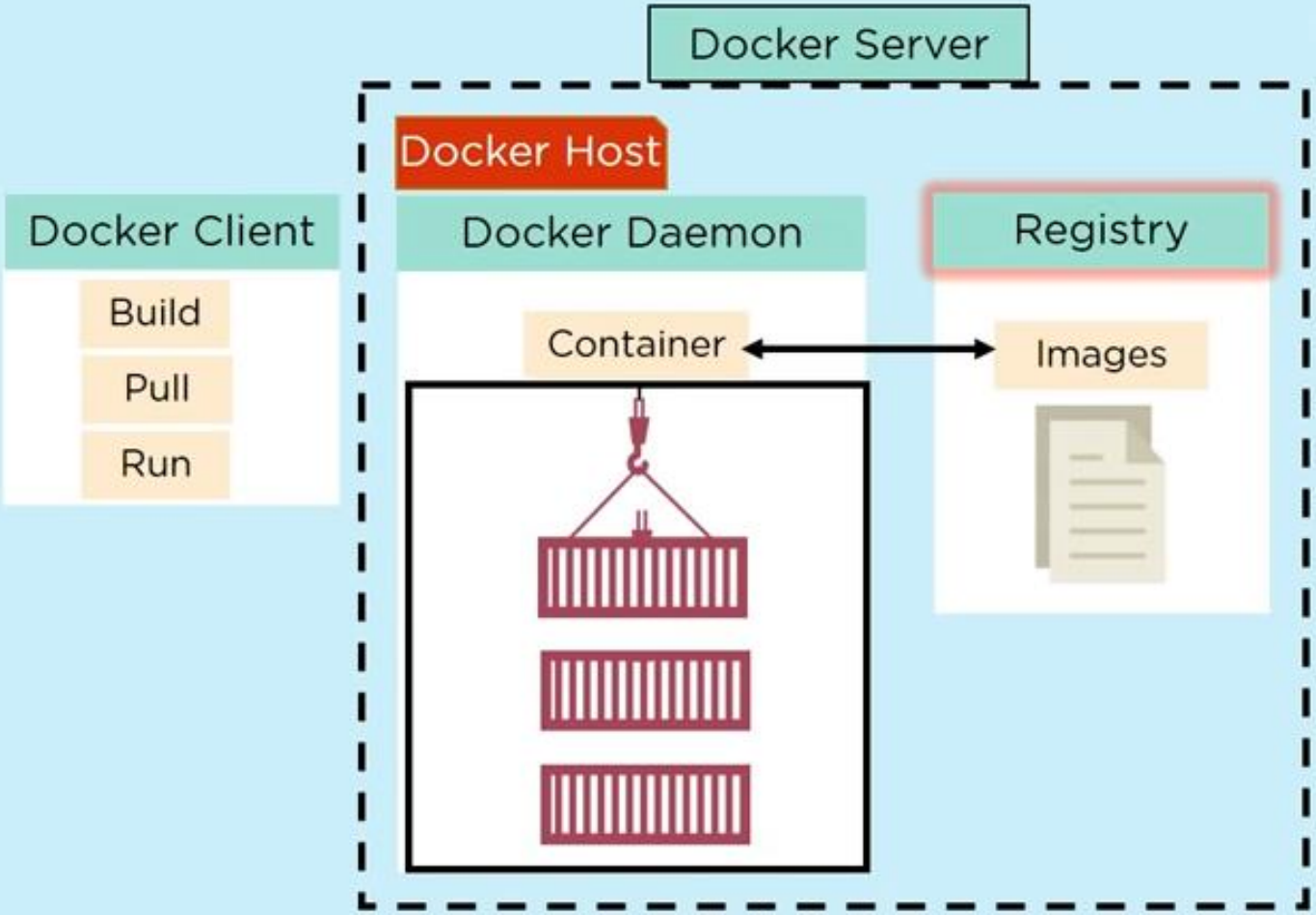
Components of Docker - Docker Container

- Numerous Docker Containers run on the same infrastructure and share operating system (OS) with its other containers
- Each application runs in isolation

Components of Docker - Docker Registry

- Docker Registry is an open source server-side service used for hosting and distributing images
- Docker also has its own default registry called Docker Hub
- Images can be stored in either public or private repositories
- Pull and Push are the commands used by users in order to interact with a Docker Registry

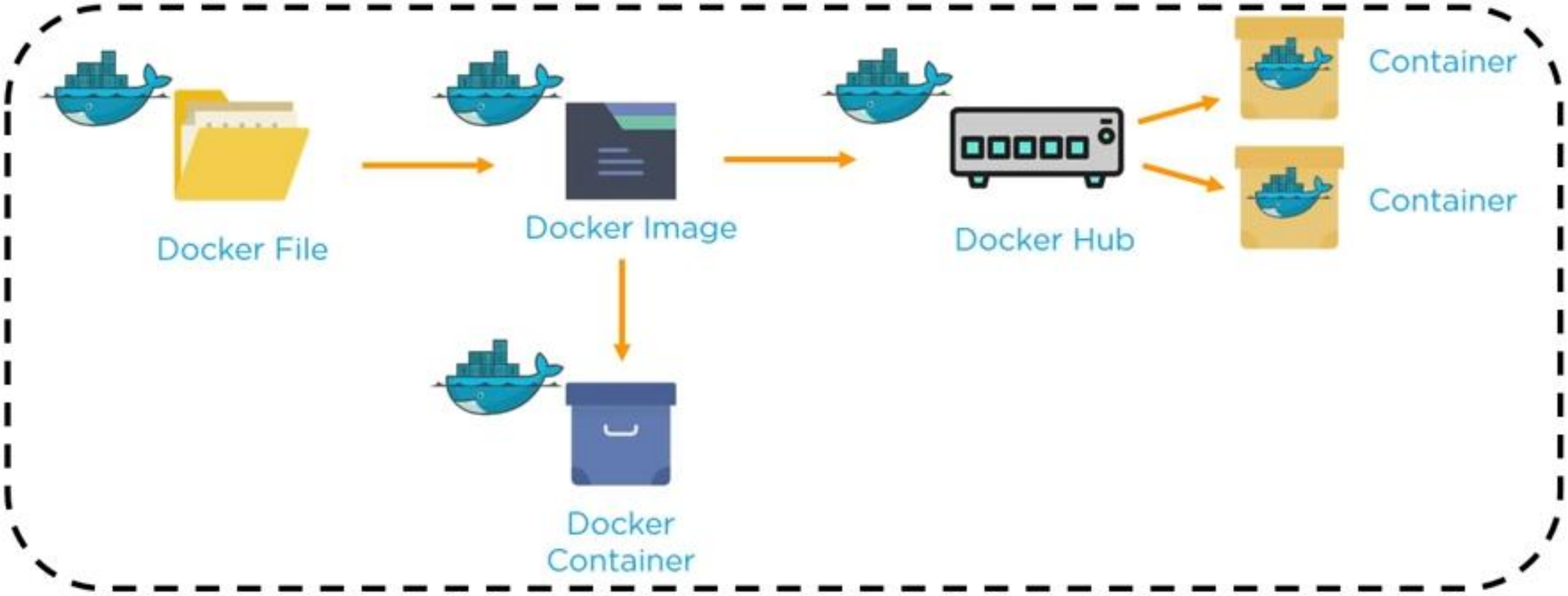
Components of Docker - Diagram



Components of Docker - Recap

- Docker File creates a Docker Image using the build command
- A Docker Image contains all the project's code
- Using Docker Image, any user can run the code in order to create Docker Containers
- Once a Docker Image is built, it's uploaded in a registry or a Docker Hub
- From the Docker Hub, users can get the Docker Image and build new containers

Components of Docker - Recap



Advanced Concepts in Docker

- Docker Compose
- Docker Swarm

Docker Compose

- Compose is a tool for defining and running multi-container Docker applications as a single service
- Compose files are very easy to write in a scripting language called YAML, which is an XML-based language that stands for Yet Another Markup Language

Docker Compose - YAML file

```yaml
version: "3"

services:

  nginx:
    container_name: nginx
    image: nginx:latest

    ports:
      - "80:80"
```

Benefits of Docker-Compose

- Single host deployment
- Quick and easy configuration

Basic Docker Commands

Basic Docker Commands

docker -v

docker --version

- Used to get the installed version

Basic Docker Commands

docker -v

docker --version

```
ubuntu@ubuntu-server: ~
ubuntu@ubuntu-server: ~

ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker -v
Docker version 20.10.8, build 3967b7d
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker --version
Docker version 20.10.8, build 3967b7d
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$
```

Basic Docker Commands

docker --help

docker [COMMAND] --help

- Used to get help information

Basic Docker Commands

docker --help

ubuntu@ubuntu-server: ~ | ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ docker --help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/ubuntu/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides
                           DOCKER_HOST env var and default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default
                           "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/home/ubuntu/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/ubuntu/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/ubuntu/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

Basic Docker Commands

docker [COMMAND] --help

ubuntu@ubuntu-server: ~ | ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

Options:
      --add-host list                  Add a custom host-to-IP mapping (host:ip)
  -a, --attach list                    Attach to STDIN, STDOUT or STDERR
      --blkio-weight uint16            Block IO (relative weight), between 10 and 1000, or 0 to disable
                                       (default 0)
      --blkio-weight-device list       Block IO weight (relative device weight) (default [])
      --cap-add list                   Add Linux capabilities
      --cap-drop list                  Drop Linux capabilities
      --cgroup-parent string           Optional parent cgroup for the container
      --cgroupns string                Cgroup namespace to use (host|private)
                                       'host':    Run the container in the Docker host's cgroup namespace
                                       'private': Run the container in its own private cgroup namespace
                                       '':        Use the cgroup namespace as configured by the
                                                  default-cgroupns-mode option on the daemon (default)
```

Basic Docker Commands

docker pull <image_name>

- Used to pull images from the docker repository (hub.docker.com)

Basic Docker Commands

docker pull <image_name>

```
ubuntu@ubuntu-server: ~
ubuntu@ubuntu-server: ~

ubuntu@ubuntu-server:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
42c077c10790: Pull complete
62c70f376f6a: Pull complete
915cc9bd79c2: Pull complete
75a963e94de0: Pull complete
7b1fab684d70: Pull complete
db24d06d5af4: Pull complete
Digest: sha256:2bcabc23b45489fb0885d69a06ba1d648aeda973fae7bb981bafbb884165e514
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
ubuntu@ubuntu-server:~$
```

Basic Docker Commands

docker run

- Used to create a container from an image

Basic Docker Commands

docker run

```
ubuntu@ubuntu-server: ~
ubuntu@ubuntu-server: ~
ubuntu@ubuntu-server:~$ docker run --name=nginx -d nginx:latest
90a58150f28d946f13a25780ad55f86904833e59a85465d833d9ad25ac586563
ubuntu@ubuntu-server:~$
```

```
ubuntu@ubuntu-server: ~
ubuntu@ubuntu-server: ~
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker run --name=nginx nginx:latest
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
```

Basic Docker Commands

docker ps

- Used to list the running containers

Basic Docker Commands

docker ps

ubuntu@ubuntu-server: ~ | ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED        STATUS         PORTS
                                                NAMES
2bb3c61ca4c9   nginx:latest                           "/docker-entrypoint.…"   6 seconds ago  Up 4 seconds   80/tcp
                                                nginx
e6c385611c7e   ramesesinc/notification-server:1.0     "docker-entrypoint.s…"   3 weeks ago    Up 3 weeks     0.0.0.0:
7080->8080/tcp, :::7080->8080/tcp               rameses-notification-server
a5f441198e10   portainer/portainer-ce                 "/portainer"             8 months ago   Up 3 months    8000/tcp
, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp     portainer
ubuntu@ubuntu-server:~$
```

Basic Docker Commands

docker ps -a

- Used to show all the running and exited containers

Basic Docker Commands

docker ps -a

ubuntu@ubuntu-server: ~ | ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
    PORTS                                                 NAMES
2bb3c61ca4c9   nginx:latest                         "/docker-entrypoint.…"   2 minutes ago   Exited (0) 5 seconds ag
o                                                     nginx
e6c385611c7e   ramesesinc/notification-server:1.0   "docker-entrypoint.s…"   3 weeks ago     Up 3 weeks
    0.0.0.0:7080->8080/tcp, :::7080->8080/tcp         rameses-notification-server
a5f441198e10   portainer/portainer-ce               "/portainer"             8 months ago    Up 3 months
    8000/tcp, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   portainer
ubuntu@ubuntu-server:~$
```

Basic Docker Commands

docker exec

- Used to access the running container

Basic Docker Commands

docker exec

```
ubuntu@ubuntu-server: ~        ubuntu@ubuntu-server: ~
ubuntu@ubuntu-server:~$ docker exec -it nginx bash
root@2bb3c61ca4c9:/#
root@2bb3c61ca4c9:/#
root@2bb3c61ca4c9:/# ls
bin   dev                  docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp
var
boot  docker-entrypoint.d  etc                   lib   media  opt  root  sbin  sys  usr
root@2bb3c61ca4c9:/#
```

Basic Docker Commands

docker logs

- Fetch the logs of a container

Basic Docker Commands

docker logs

ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker logs -f nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/06/12 06:46:33 [notice] 1#1: using the "epoll" event method
2022/06/12 06:46:33 [notice] 1#1: nginx/1.21.6
2022/06/12 06:46:33 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/06/12 06:46:33 [notice] 1#1: OS: Linux 4.15.0-184-generic
2022/06/12 06:46:33 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/06/12 06:46:33 [notice] 1#1: start worker processes
2022/06/12 06:46:33 [notice] 1#1: start worker process 30
```

Basic Docker Commands

docker stop

- Used to stop a running container

Basic Docker Commands

docker stop

ubuntu@ubuntu-server: ~ | ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ docker ps
CONTAINER ID   IMAGE                                   COMMAND                  CREATED         STAT
US         PORTS                                                   NAMES
2bb3c61ca4c9   nginx:latest                            "/docker-entrypoint.…"   7 minutes ago   Up 3
 minutes   80/tcp                                                  nginx
e6c385611c7e   ramesesinc/notification-server:1.0      "docker-entrypoint.s…"   3 weeks ago     Up 3
 weeks     0.0.0.0:7080->8080/tcp, :::7080->8080/tcp               rameses-notification-server
a5f441198e10   portainer/portainer-ce                  "/portainer"             8 months ago    Up 3
 months    8000/tcp, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   portainer
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker stop nginx
nginx
ubuntu@ubuntu-server:~$ docker ps -a
CONTAINER ID   IMAGE                                   COMMAND                  CREATED         STAT
US                     PORTS                                                    NAMES
2bb3c61ca4c9   nginx:latest                            "/docker-entrypoint.…"   8 minutes ago   Exit
ed (0) 19 seconds ago                                                           nginx
```

Basic Docker Commands

docker rm

- Used to delete or remove a stopped container

Basic Docker Commands

docker rm

ubuntu@ubuntu-server: ~ | ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ clear
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker ps -a
CONTAINER ID   IMAGE                                   COMMAND                  CREATED          STA
TUS                     PORTS                                                    NAMES
2bb3c61ca4c9   nginx:latest                            "/docker-entrypoint.…"   13 minutes ago   Exi
ted (0) 5 minutes ago                                                            nginx
e6c385611c7e   ramesesinc/notification-server:1.0      "docker-entrypoint.s…"   3 weeks ago      Up
3 weeks                 0.0.0.0:7080->8080/tcp, :::7080->8080/tcp                rameses-notificatio
n-server
a5f441198e10   portainer/portainer-ce                  "/portainer"             8 months ago     Up
3 months                8000/tcp, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp      portainer
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker rm nginx
nginx
ubuntu@ubuntu-server:~$ docker ps -a
CONTAINER ID   IMAGE                                   COMMAND                  CREATED        STATU
S        PORTS                                                     NAMES
e6c385611c7e   ramesesinc/notification-server:1.0      "docker-entrypoint.s…"   3 weeks ago    Up 3
```

Basic Docker Commands

docker kill

- This command kills the container by stopping its execution immediately.
- The difference between ‘docker kill’ and ‘docker stop’ is that ‘docker stop’ gives the container time to shutdown gracefully

Basic Docker Commands

docker commit

- This command creates a new image of an edited container on the local system

Basic Docker Commands

docker images

- Used to lists all locally stored docker images

Basic Docker Commands

docker images

```
ubuntu@ubuntu-server: ~        ubuntu@ubuntu-server: ~

ubuntu@ubuntu-server:~$ clear
ubuntu@ubuntu-server:~$
ubuntu@ubuntu-server:~$ docker images
REPOSITORY                          TAG             IMAGE ID       CREATED        SIZE
nginx                               latest          0e901e68141f   9 days ago     142MB
ramesesinc/etracs-server-province   2.5.04.05.01    0a60282b96bd   3 weeks ago    177MB
ramesesinc/etracs-server-province   beta            0a60282b96bd   3 weeks ago    177MB
ramesesinc/etracs-server-municipality 2.5.04.05.01  4c265f59309d   3 weeks ago    177MB
ramesesinc/etracs-server-municipality beta          4c265f59309d   3 weeks ago    177MB
ramesesinc/etracs-server-city       2.5.04.05.01    2ca8d816771f   3 weeks ago    177MB
ramesesinc/etracs-server-city       beta            2ca8d816771f   3 weeks ago    177MB
ramesesinc/etracs-services          2.5.04.05       5864144f9674   3 weeks ago    177MB
ramesesinc/etracs-services          beta            5864144f9674   3 weeks ago    177MB
ramesesinc/etracs-core              2.5.04          67f97aa82e15   3 weeks ago    164MB
ramesesinc/etracs-core              beta            67f97aa82e15   3 weeks ago    164MB
ramesesinc/etracsorg                1.01            6bc05ca50bfd   4 months ago   179MB
ramesesinc/mail-server              1.01            6ec78652c153   4 months ago   191MB
ramesesinc/mail-server              latest          6ec78652c153   4 months ago   191MB
ramesesinc/gdx-client               1.04.03         efba9f4ea0e0   4 months ago   174MB
```

Basic Docker Commands

docker save

- Save one or more images to a tar archive

Basic Docker Commands

docker save

ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ docker save -o nginx.tar nginx:latest
ubuntu@ubuntu-server:~$
```

Basic Docker Commands

docker load

- Load an image from a tar archive

Basic Docker Commands

docker load

ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ docker load -i nginx.tar
Loaded image: nginx:latest
ubuntu@ubuntu-server:~$
```

Basic Docker Commands

docker rmi

- Used to delete an image from local storage

Basic Docker Commands

docker rmi

ubuntu@ubuntu-server: ~

```
ubuntu@ubuntu-server:~$ docker rmi nginx:latest
Untagged: nginx:latest
Deleted: sha256:0e901e68141fd02f237cf63eb842529f8a9500636a9419e3cf4fb986b8fe3d5d
Deleted: sha256:1e877fb1acf761377390ab38bbad050a1d5296f1b4f51878c2695d4ecdb98c62
Deleted: sha256:834e54d50f731515065370d1c15f0ed47d2f7b6a7b0452646db80f14ace9b8de
Deleted: sha256:d28ca7ee17ff94497071d5c075b4099a4f2c950a3471fc49bdf9876227970b24
Deleted: sha256:096f97ba95539883af393732efac02acdd0e2ae587a5479d97065b64b4eded8c
Deleted: sha256:de7e3b2a7430261fde88313fbf784a63c2229ce369b9116053786845c39058d5
Deleted: sha256:ad6562704f3759fb50f0d3de5f80a38f65a85e709b77fd24491253990f30b6be
ubuntu@ubuntu-server:~$
```

Basic Docker Commands

docker login

- Used to login to the docker hub repository

Basic Docker Commands

docker logout

- Used to logout from a Docker registry

Basic Docker Commands

docker push

- Used to push an image to the docker hub repository

Basic Docker Commands

docker build

- Used to build an image from a specified docker file

Basic Docker Commands

docker build

```
ubuntu@ubuntu-server:~$ docker build -t ramesesinc/etracs-core:2.5.04 .

Sending build context to Docker daemon  39.35MB
Step 1/15 : FROM ramesesinc/alpine-java:jre8
 ---> f8388f56eae6
Step 2/15 : COPY /apps /apps
 ---> Using cache
 ---> ed9b0e55c1a3
Step 3/15 : COPY /tz/zoneinfo /usr/share/zoneinfo
 ---> Using cache
 ---> 1737f80289d2
Step 4/15 : COPY /tz/zoneinfo/Asia/Manila /etc/localtime
 ---> Using cache
 ---> a972fb1f3a19
Step 5/15 : COPY /tz/timezone /etc/timezone
 ---> Using cache
 ---> 6d666bfde169
Step 6/15 : WORKDIR /apps/server/bin
 ---> Using cache
 ---> 054bb6fbe3e7
Step 7/15 : RUN tar -xf sh.tar.gz
 ---> Using cache
 ---> 053e7b024dd2
Step 8/15 : RUN rm -f sh.tar.gz
 ---> Using cache
 ---> e1f0beb515d3
Step 9/15 : WORKDIR /apps
 ---> Using cache
 ---> 5326e95cfbd8
```

Dockerfile

```
FROM ramesesinc/alpine-java:jre8

COPY /apps /apps
COPY /tz/zoneinfo /usr/share/zoneinfo
COPY /tz/zoneinfo/Asia/Manila /etc/localtime
COPY /tz/timezone /etc/timezone

WORKDIR /apps/server/bin
RUN tar -xf sh.tar.gz
RUN rm -f sh.tar.gz

WORKDIR /apps
RUN tar -xf sh.tar.gz
RUN rm -f sh.tar.gz

ENV LANG en_US.UTF-8
ENV LANGUAGE en_US:en

CMD ["/bin/bash", "/apps/start.sh"]

EXPOSE 8060 8061 8080 8070
```

Docker Compose

What is Docker Compose ?

- **Compose** is a tool for defining and running multi-container Docker applications.
- Use a YAML file to configure your application's services.
- Create and start all the services from your configuration

How does Docker Compose works ?

- Define the services that make up your app in a file called `docker-compose.yml` , so they can be run together in an isolated environment

- Run `”docker-compose up”` to start and run your entire app

docker-compose.yml

```yaml
version: "3"

services:

    nginx:
        container_name: nginx
        image: nginx:latest
        ports:
            - "80:80"


    portainer1:
        container_name: portainer1
        image: portainer/portainer-ce
        ports:
            - "9001:9000"
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
```

Docker Compose Commands

Docker Compose Commands

docker-compose --version

- Used to check a version

Docker Compose Commands

docker-compose --version

```
ubuntu@ubuntu-server:~/training-202206/nginx$ docker-compose --version

docker-compose version 1.23.1, build b02f1306

ubuntu@ubuntu-server:~/training-202206/nginx$
```

Docker Compose Commands

docker-compose up

- Used to start all services

docker-compose up -d

- Used to start all services in the background and leave them running

Docker Compose Commands

docker-compose up

```
ubuntu@ubuntu-server:~/training-202206/nginx$ docker-compose up

Creating network "nginx_default" with the default driver
Creating nginx ... done
Attaching to nginx
nginx    | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx    | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx    | 10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enabled
nginx    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx    | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx    | 2022/06/06 16:40:13 [notice] 1#1: using the "epoll" event method
nginx    | 2022/06/06 16:40:13 [notice] 1#1: nginx/1.21.6
nginx    | 2022/06/06 16:40:13 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
nginx    | 2022/06/06 16:40:13 [notice] 1#1: OS: Linux 4.15.0-169-generic
nginx    | 2022/06/06 16:40:13 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
nginx    | 2022/06/06 16:40:13 [notice] 1#1: start worker processes
nginx    | 2022/06/06 16:40:13 [notice] 1#1: start worker process 24
```

Docker Compose Commands

docker-compose up -d

```
ubuntu@ubuntu-server:~/training-202206/nginx$ docker-compose up -d

Starting nginx ... done

ubuntu@ubuntu-server:~/training-202206/nginx$
```

Docker Compose Commands

docker-compose down

- Used to stop all services

Docker Compose Commands

docker-compose down

```
ubuntu@ubuntu-server:~/training-202206/nginx$ docker-compose down

Stopping nginx ... done
Removing nginx ... done
Removing network nginx_default

ubuntu@ubuntu-server:~/training-202206/nginx$
```

Docker Compose Commands

docker-compose logs

- View output from containers

Docker Compose Commands

docker-compose logs

```
ubuntu@ubuntu-server:~/training-202206/nginx$ docker-compose logs -f

Attaching to nginx1
nginx1    | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will a
nginx1    | /docker-entrypoint.sh: Looking for shell scripts in /docker-entry
nginx1    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-o
nginx1    | 10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enable
nginx1    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst
nginx1    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-wor
nginx1    | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx1    | 2022/06/12 06:57:14 [notice] 1#1: using the "epoll" event method
nginx1    | 2022/06/12 06:57:14 [notice] 1#1: nginx/1.21.6
nginx1    | 2022/06/12 06:57:14 [notice] 1#1: built by gcc 10.2.1 20210110 (D
nginx1    | 2022/06/12 06:57:14 [notice] 1#1: OS: Linux 4.15.0-184-generic
nginx1    | 2022/06/12 06:57:14 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 10485
nginx1    | 2022/06/12 06:57:14 [notice] 1#1: start worker processes
nginx1    | 2022/06/12 06:57:14 [notice] 1#1: start worker process 23
```

Demo and Exercises

Exercise #1

```
## Go to your training-202206 repository folder
cd /mnt/c/training-202206

## Pull updates from remote origin
## Usage: git pull <remote_name> <branch>
git pull
```

Exercise #1 - Result

```
ubuntu@ubuntu-server:~/training-202206$ git pull
ubuntu@192.168.0.10's password:
remote: Counting objects: 18, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 18 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (18/18), done.
From 192.168.0.10:gitrepo/training-202206
   70a1449..39fa0de  master     -> origin/master
Updating 70a1449..39fa0de
Fast-forward
 .gitignore                          |   3 +++
 docs/DevTech-Training-Day-1.pdf     | Bin 0 -> 1485095 bytes
 file1.txt                           |   2 ++
 mysql/conf/conf.d/docker.cnf        |   3 +++
 mysql/conf/conf.d/mysql.cnf         |   1 +
 mysql/conf/conf.d/mysqldump.cnf     |   4 ++++
 mysql/conf/mysql.conf.d/mysqld.cnf  |  40 ++++++++++++++++++++++++++++++++++++++++
 mysql/docker-compose.yml            |  26 ++++++++++++++++++++++++++
 nginx/conf.d/default.conf           |  21 +++++++++++++++++++++
 nginx/docker-compose.yml            |  21 +++++++++++++++++++++
 portainer/docker-compose.yml        |  21 +++++++++++++++++++++
 11 files changed, 142 insertions(+)
 create mode 100644 docs/DevTech-Training-Day-1.pdf
 create mode 100644 file1.txt
 create mode 100755 mysql/conf/conf.d/docker.cnf
 create mode 100755 mysql/conf/conf.d/mysql.cnf
 create mode 100755 mysql/conf/conf.d/mysqldump.cnf
 create mode 100755 mysql/conf/mysql.conf.d/mysqld.cnf
 create mode 100755 mysql/docker-compose.yml
 create mode 100755 nginx/conf.d/default.conf
 create mode 100755 nginx/docker-compose.yml
 create mode 100755 portainer/docker-compose.yml
ubuntu@ubuntu-server:~/training-202206$
```

Exercise #2

```
## Go to your training-202206 repository folder
cd /mnt/c/training-202206

## Go to portainer folder
cd portainer

## Start the Portainer service
docker-compose up -d
```

Exercise #2 - Result

```
ubuntu@ubuntu-server: ~/tra

ubuntu@ubuntu-server:~/training-202206/portainer$ docker-compose up -d
Creating network "portainer_default" with the default driver
Creating portainer1 ... done
ubuntu@ubuntu-server:~/training-202206/portainer$
```

```
ubuntu@ubuntu-server:~/training-202206/portainer$ docker ps -a
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORT
S                                                    NAMES
cf892a1827a3   portainer/portainer-ce                 "/portainer"             23 seconds ago   Up 17 seconds   8000
/tcp, 0.0.0.0:9001->9000/tcp, :::9001->9000/tcp      portainer1
```

- Open a web browser and go to the following:

http://localhost:9001

Exercise #3

```
## Go to your training-202206 repository folder
cd /mnt/c/training-202206

## Go to nginx folder
cd nginx

## Start the Nginx service
docker-compose up -d
```

Exercise #3 - Result

ubuntu@ubuntu-server: ~/tra

```
ubuntu@ubuntu-server:~/training-202206/nginx$ docker-compose up -d
Creating network "nginx_default" with the default driver
Creating nginx ... done
ubuntu@ubuntu-server:~/training-202206/nginx$
```

- Open a web browser and go to the following:

http://localhost

Exercise #4

```
## Go to your training-202206 repository folder
cd /mnt/c/training-202206

## Go to mysql folder
cd mysql

## Start the MySQL service
docker-compose up -d
```

Exercise #4 - Result

ubuntu@ubuntu-server: ~/tra

```
ubuntu@ubuntu-server:~/training-202206/mysql$ docker-compose up -d
Creating network "mysql_default" with the default driver
Creating mysql ... done
ubuntu@ubuntu-server:~/training-202206/mysql$
```

Exercise #5

```
## Get inside the docker container
docker exec -it mysql bash

## Login to mysql
mysql -u root -p

## Display the available databases
show databases;

## Exit from mysql shell
\q

## Exit from docker container
exit
```

Exercise #5 Result

```
ubuntu@ubuntu-server:~/training-202206/mysql$ docker exec -it  mysql bash
root@f279ac7aed71:/#
root@f279ac7aed71:/#
root@f279ac7aed71:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.31 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, 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> \q
Bye
root@f279ac7aed71:/# exit
exit
ubuntu@ubuntu-server:~/training-202206/mysql$
```

Exercise #6 - Shutting down services

```
## Go to your training-202206 repository folder
cd /mnt/c/training-202206

## Go to portainer folder
cd portainer

## Shutdown the Portainer service
docker-compose down

## Go to nginx folder
cd ../nginx

## Shutdown the Nginx service
docker-compose down

## Go to mysql folder
cd ../mysql

## Shutdown the MySQL service
docker-compose down
```

Portainer

What is Portainer ?

- **Portainer** is a container management tool
- Allows you to easily manage your different Docker environments (Docker hosts or Swarm clusters) using a lightweight management UI
- Allows you to manage all your Docker resources
  - Containers
  - Images
  - Volumes
  - Networks
  - And more…

Setup Portainer Volume

```
## NOTE:
##    This is only done one time
##    and cannot be re-executed again
##    unless the volume is deleted
##
## Create a docker volume
docker volume create portainer_data_dir
```

Run Portainer

```
## Go to the training repository folder
cd /mnt/c/training-202206

## Go to the portainer folder
cd portainer

## Run the portainer service
docker-compose up -d
```

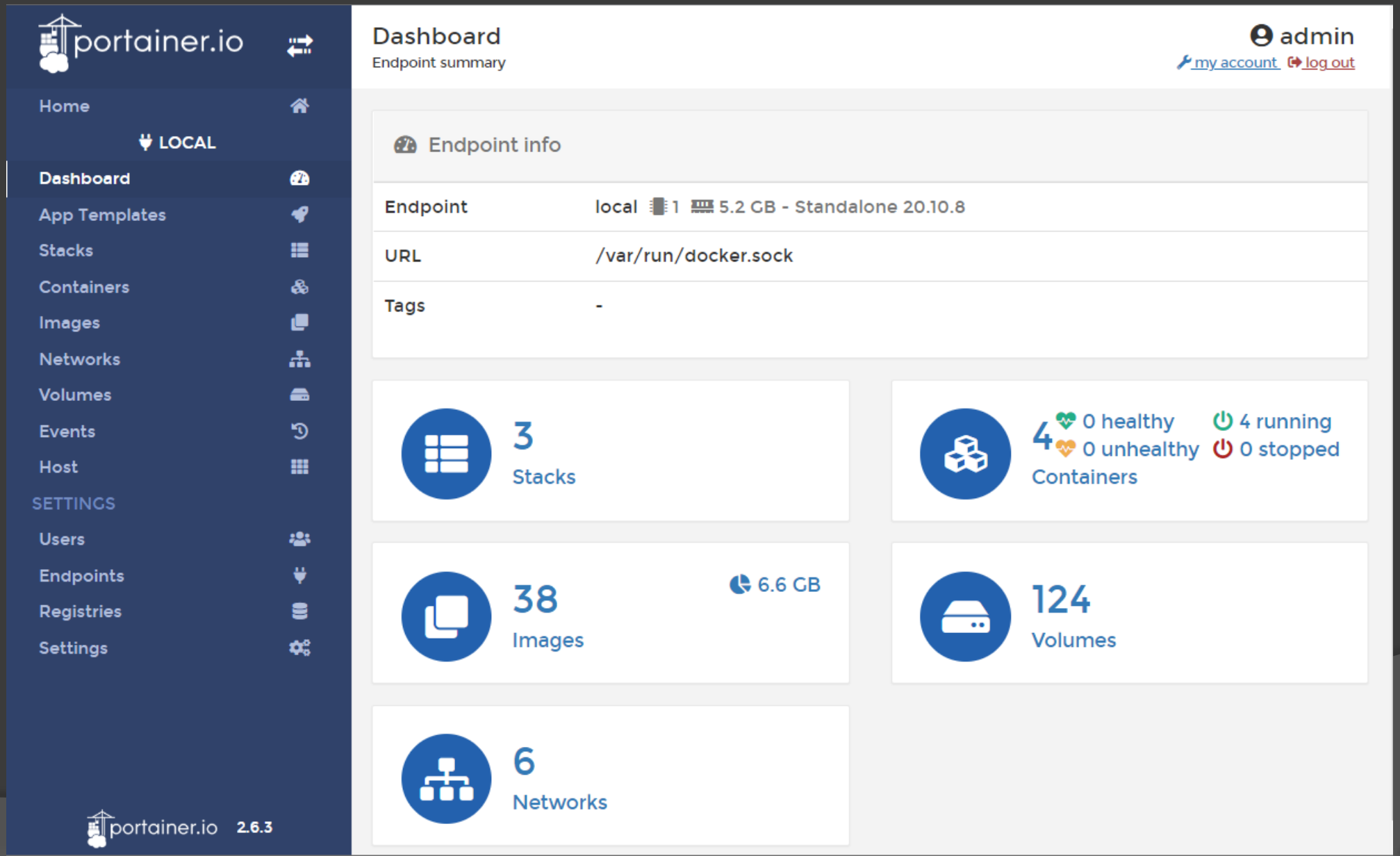
Access Portainer UI

- Open a web browser (Chrome, Mozilla, Microsoft Edge, etc…) and go to this link: `http://localhost:9001`

- Set the login credentials to:
  - Username: admin
  - Password : 12345678

- Click the "`Create User`" button to continue
- Select "`Docker`", then click the "`Connect`" button

Access Portainer UI



Next Topic

- iReport Designer
- Report Editing and Management