DATE : 25.03.2025

DT/NT : DT

LESSON : DOCKER

SUBJECT: INSTALL- BASIC OPERATIONS

BATCH : B 303

AWS-DEVOPS

TECHPRO EDUCATION

techproeducation.com +1 (585) 304 29 59

Table of Contents

TECHPRO EDUCATION

What is 

???

What is Docker?

"DOCKER" refers to several things. This includes an open-source community project which started in 2013; tools from the open-source project; Docker Inc., the company that is the primary supporter of that project; and the tools that the company formally supports.

- Docker as a "Company"
- Docker as a "Product"
- Docker as a "Platform"
- Docker as a "CLI Tool"
- Docker as a "Computer Program"

```
$ docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 01:25:46 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b7f0
  Built:            Wed Mar 11 01:24:19 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
$ _
```

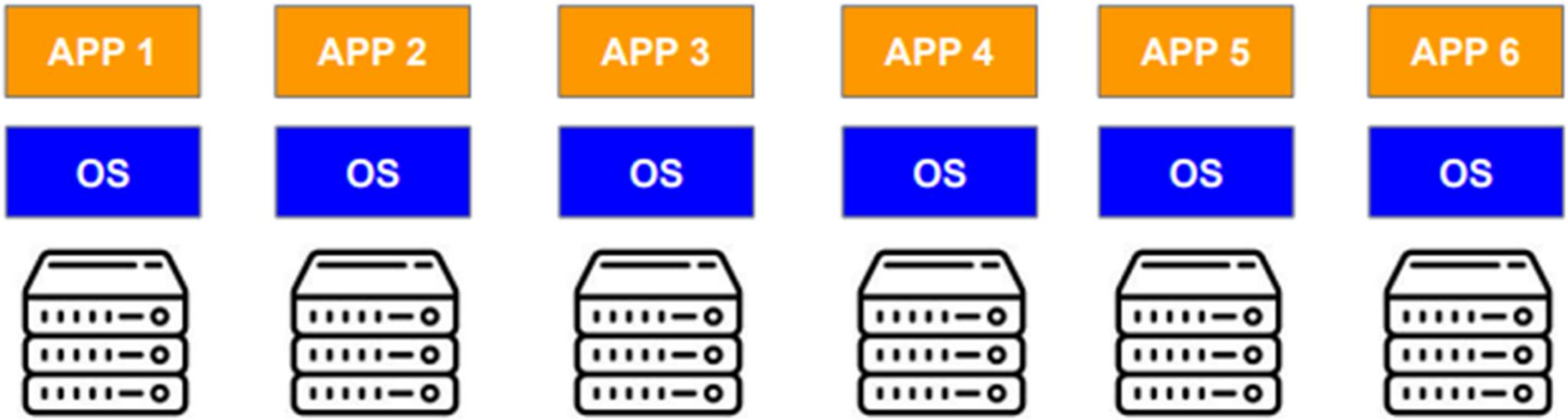
Before Docker

Bare Metal

OS = 1.5 CPU + 1.5 GB RAM

APP = 0.5 CPU + 0.5 RAM

18 CPU
18 GB RAM



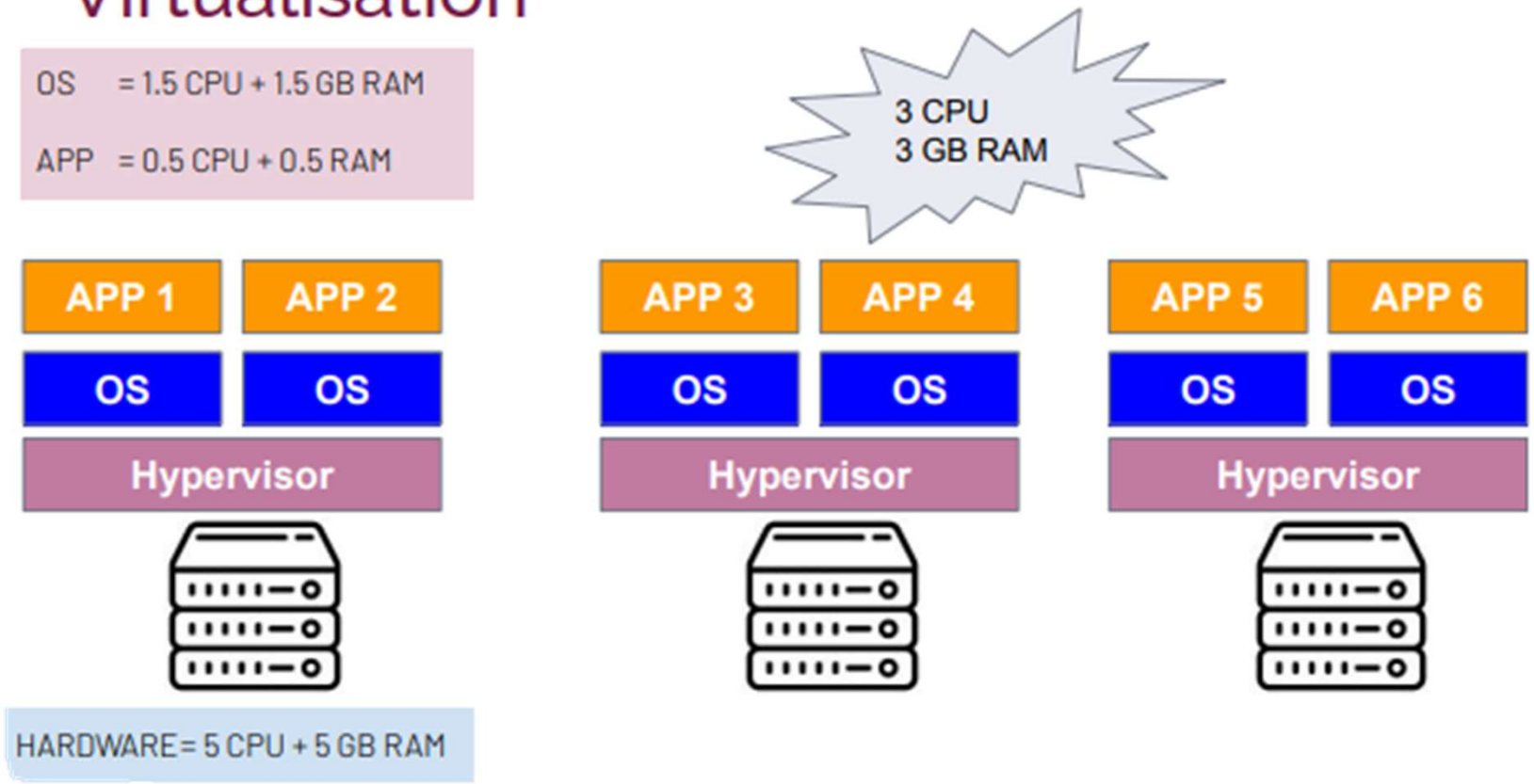
HARDWARE = 5 CPU + 5 GB RAM

TECHPRO EDUCATION

Virtualisation

OS = 1.5 CPU + 1.5 GB RAM

APP = 0.5 CPU + 0.5 RAM



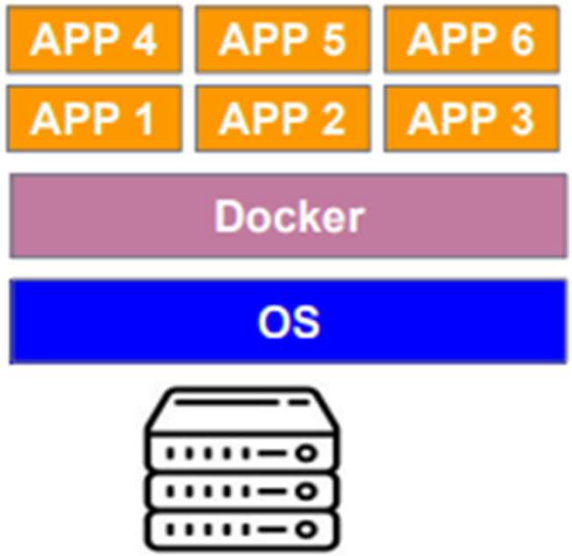
HARDWARE = 5 CPU + 5 GB RAM

Bare Metal

Container

OS = 1.5 CPU + 1.5 GB RAM

APP = 0.5 CPU + 0.5 RAM



HARDWARE = 5 CPU + 5 GB RAM

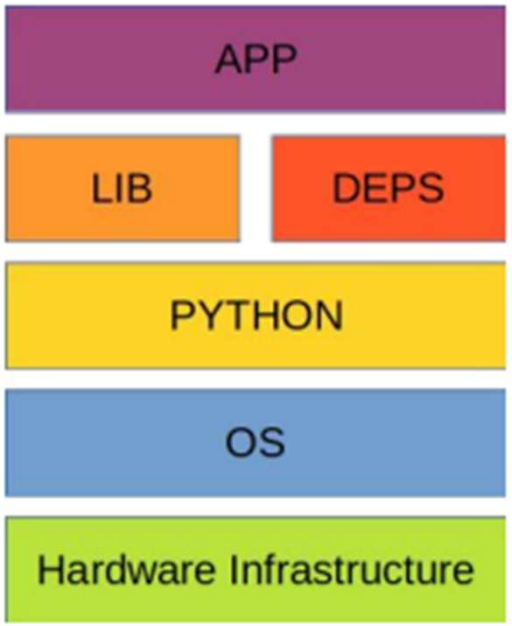
What is Container?

What is Container?

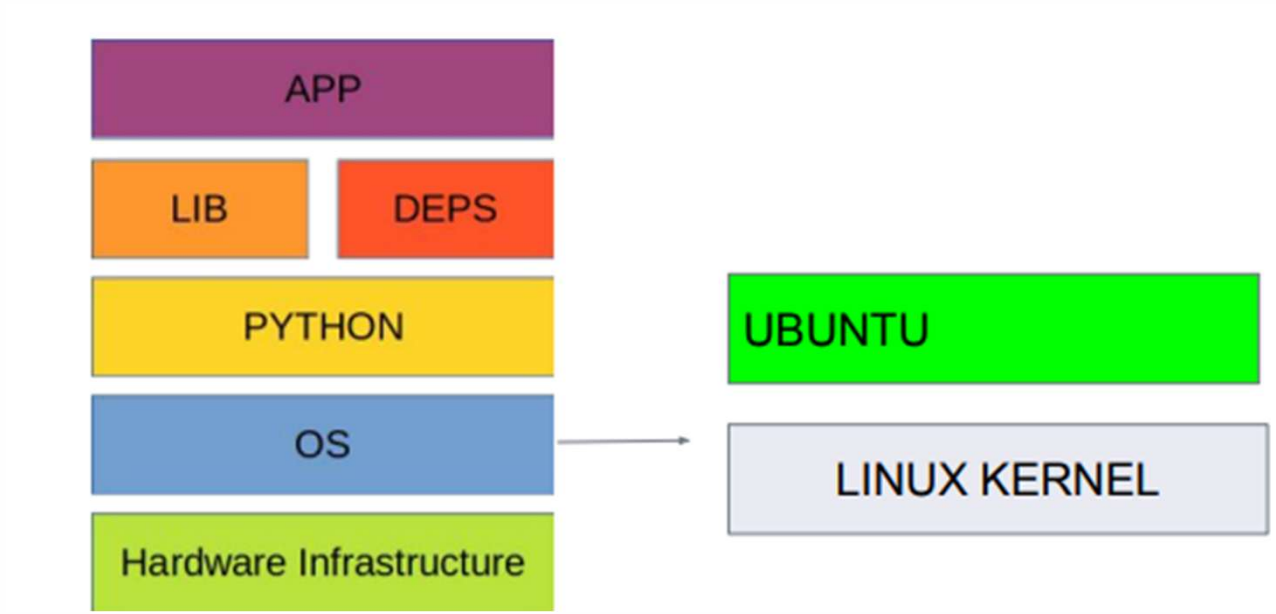
Imagine you're developing an python application. In order to do so you will setup an environment with python installed in it. You do your work on a laptop and your environment has a specific configuration. The application you're developing relies on that configuration and is dependent on specific libraries, dependencies, and files. Once the application is developed, it needs to be tested by the tester. Now the tester will again set up same environment. Once the application testing is done, it will be deployed on the production server. Again the production needs an environment with libraries, dependencies, files and python installed on it. How do you make your app work across these environments, pass quality assurance, and get your app deployed without massive headaches, rewriting, and break-fixing?

What is Container?

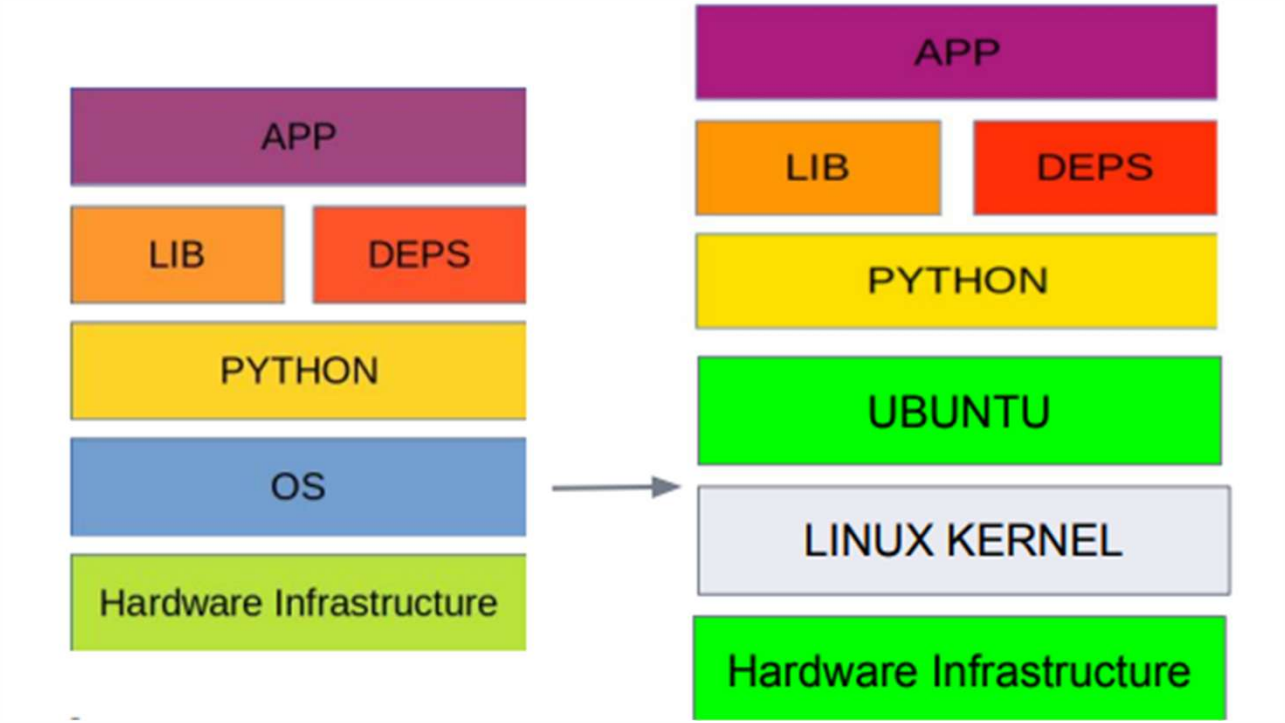
What is Container?



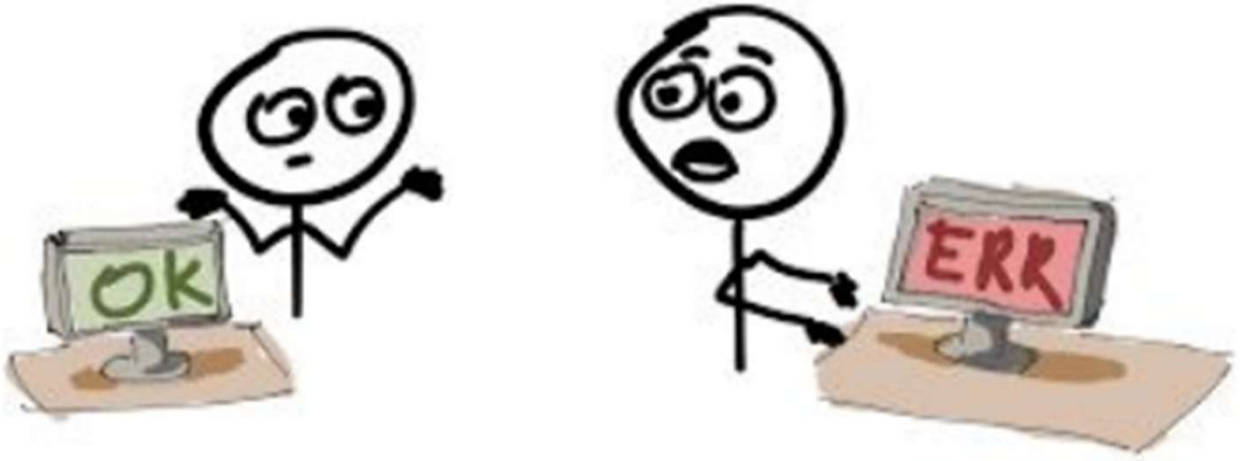
What is Container?



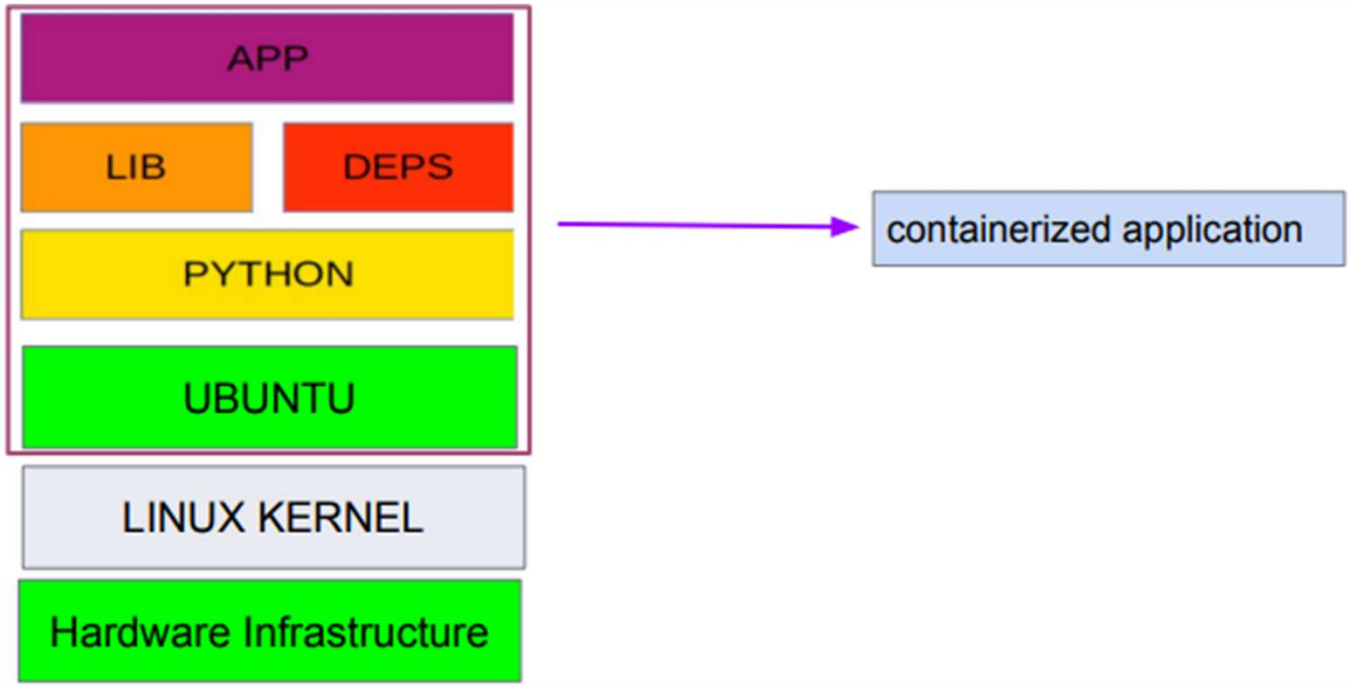
What is Container?



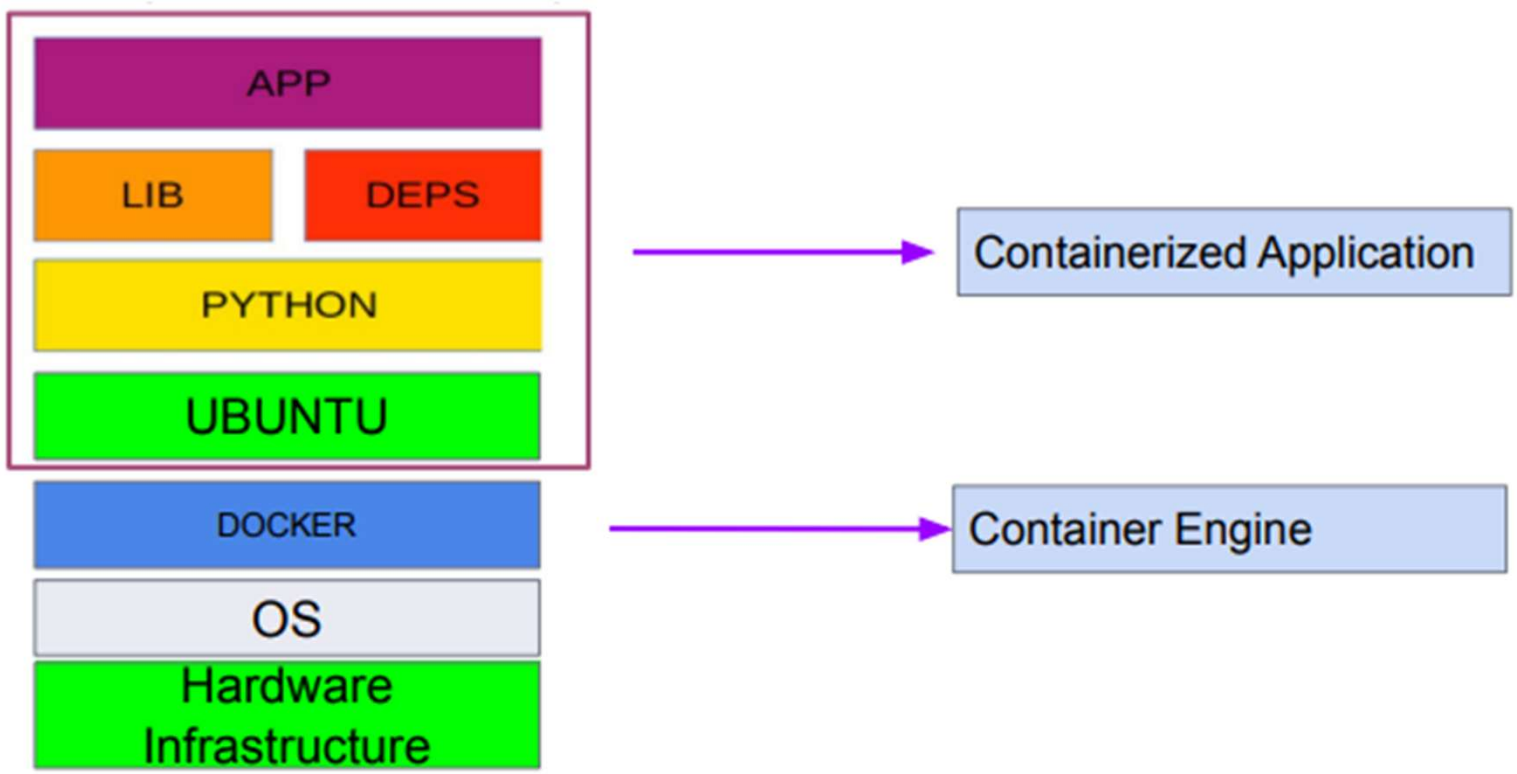
What is Container?



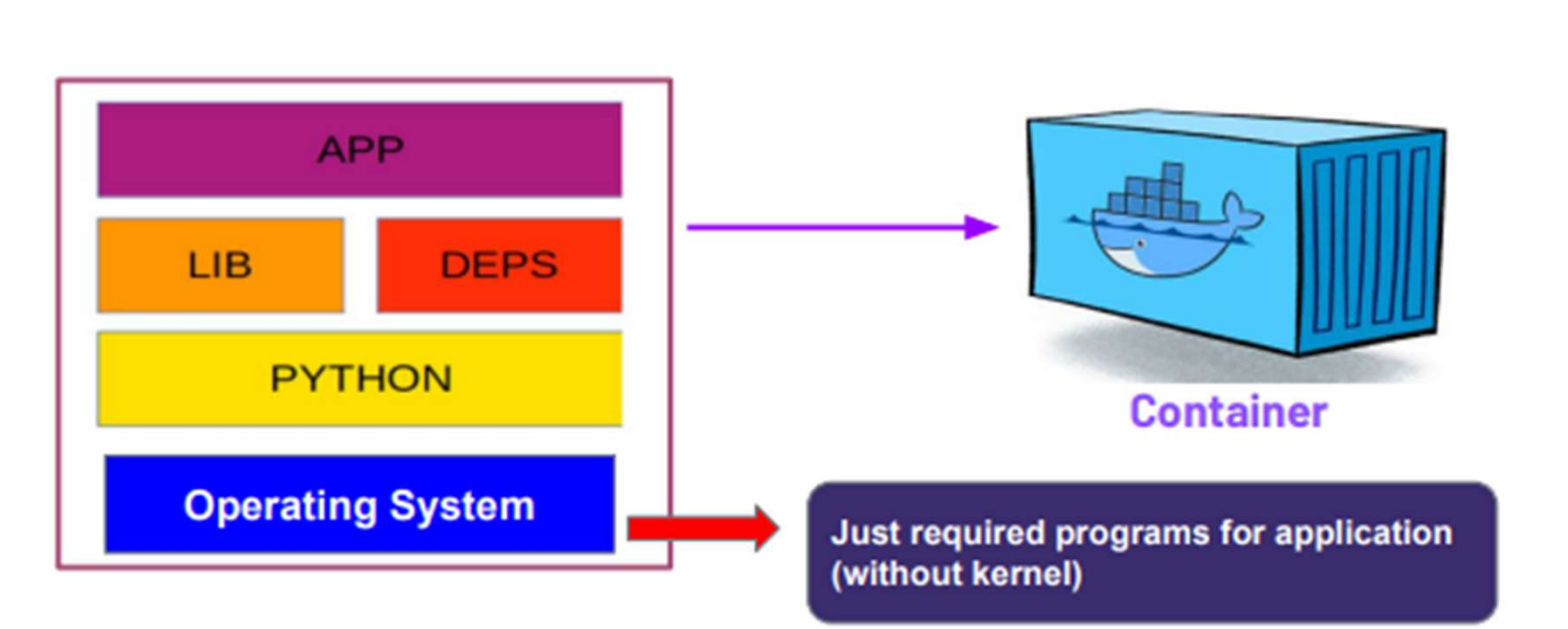
What is Container?



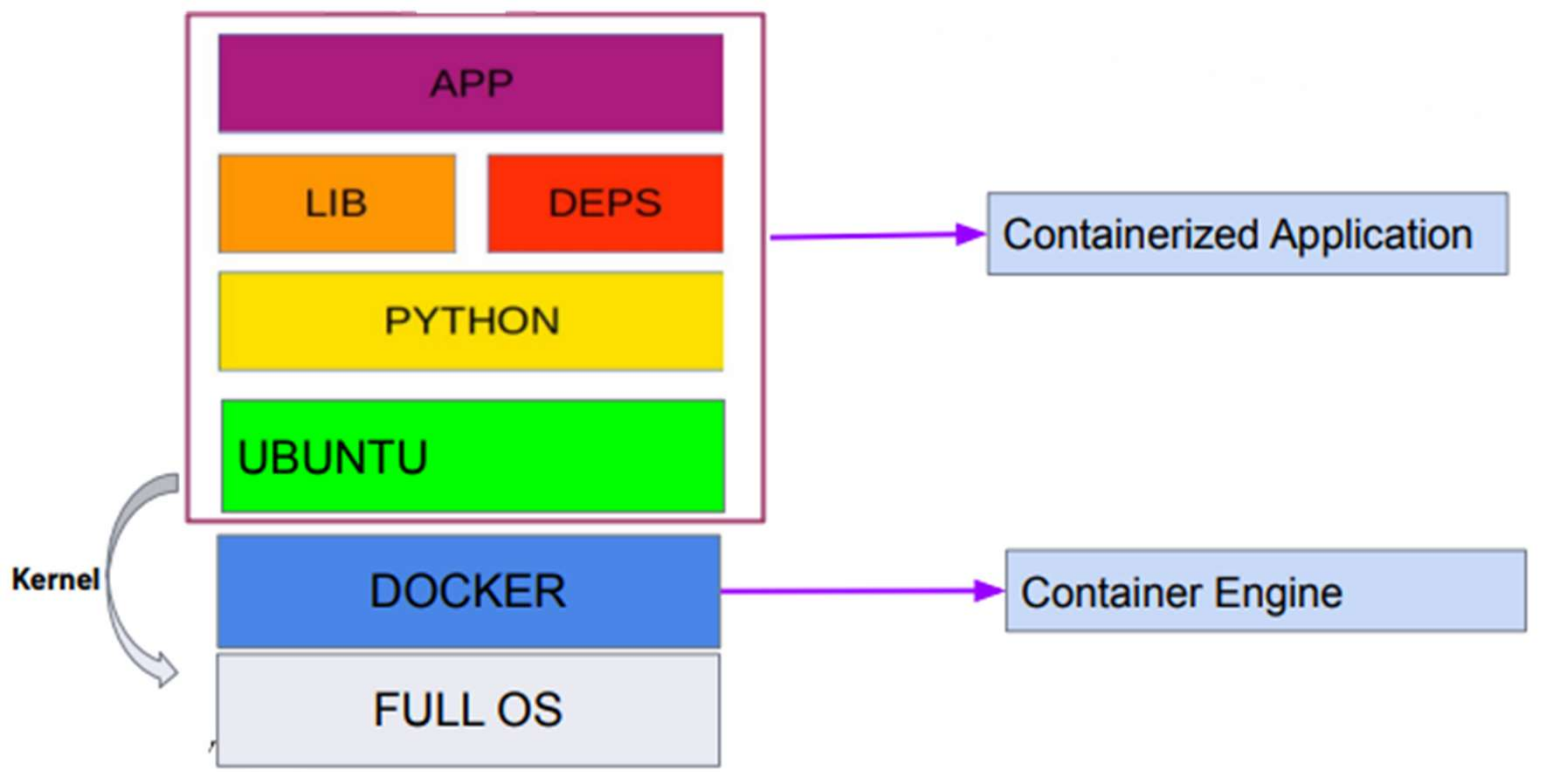
What is Container?



What is Container?

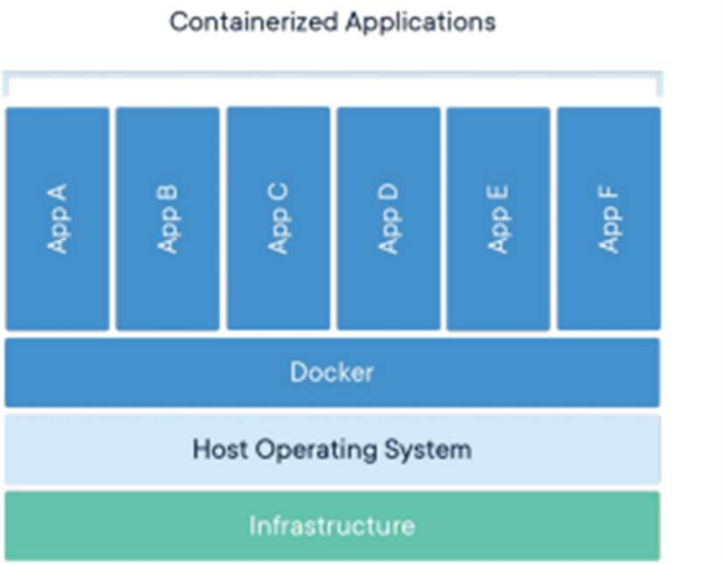


What is Container?



What is Container?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

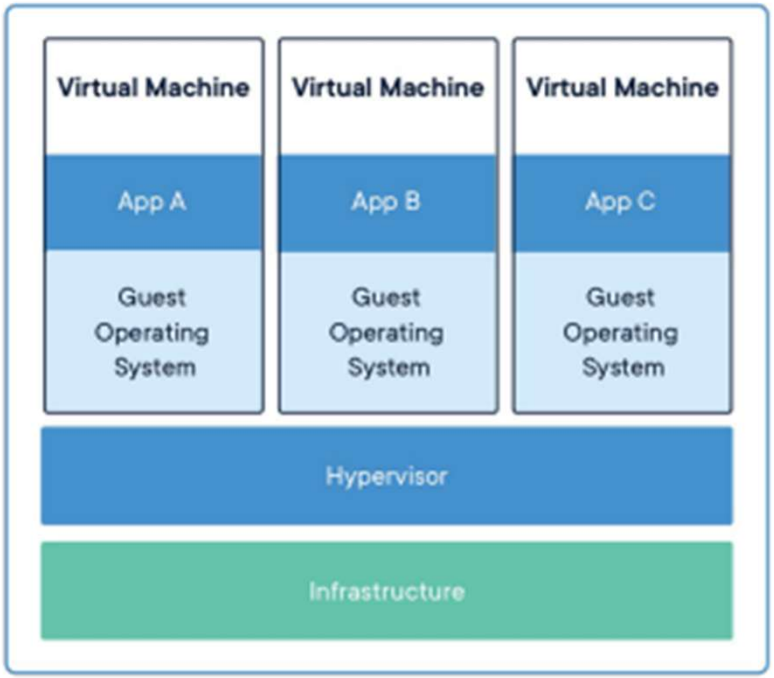


TECHPRO EDUCATION

Docker vs. VMs

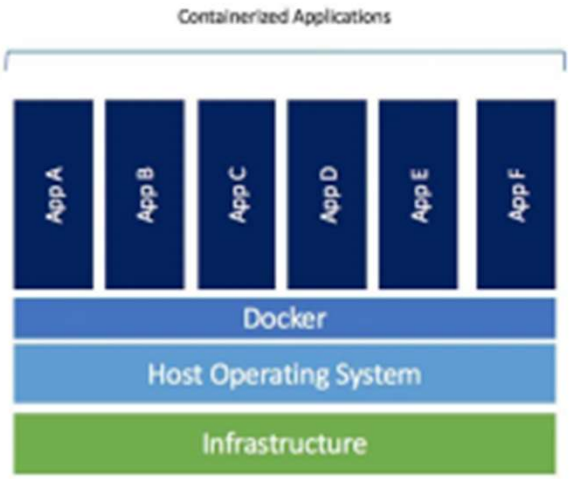
Docker vs. VMs

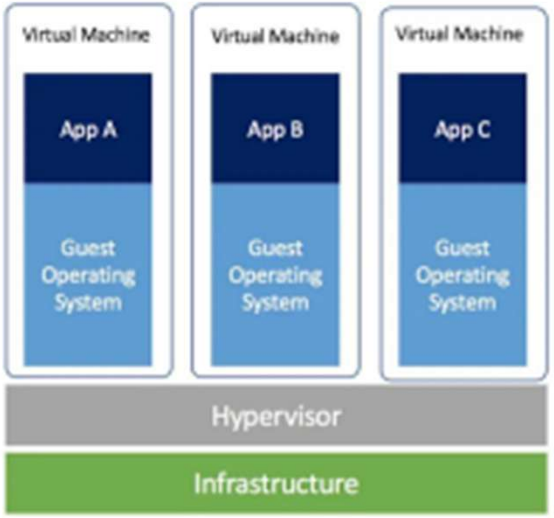
A virtual machine (VM) is software that runs programs or applications without being tied to a physical machine. Virtual Machines are built over the physical hardware, there is a hypervisor layer which sits between physical hardware and operating systems.



Docker vs. VMs

Unlike virtual machines where hypervisor divides physical hardware into parts, Containers are like normal operating system processes. A virtual machine (VM) virtualize the hardware. But the containers virtualize the operating system





Docker vs. VMs

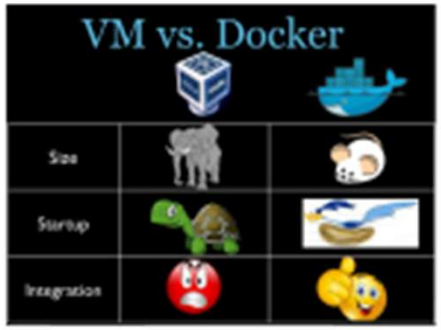
Virtual Machine

Containers

Docker containers are executed with the Docker engine rather than the hypervisor. Containers are therefore smaller than Virtual Machines and enable faster startup with better performance, less isolation and greater compatibility possible due to sharing of the host's kernel. Hence, it looks very similar to the residential flats system where we share resources of the building.

Docker vs. VMs

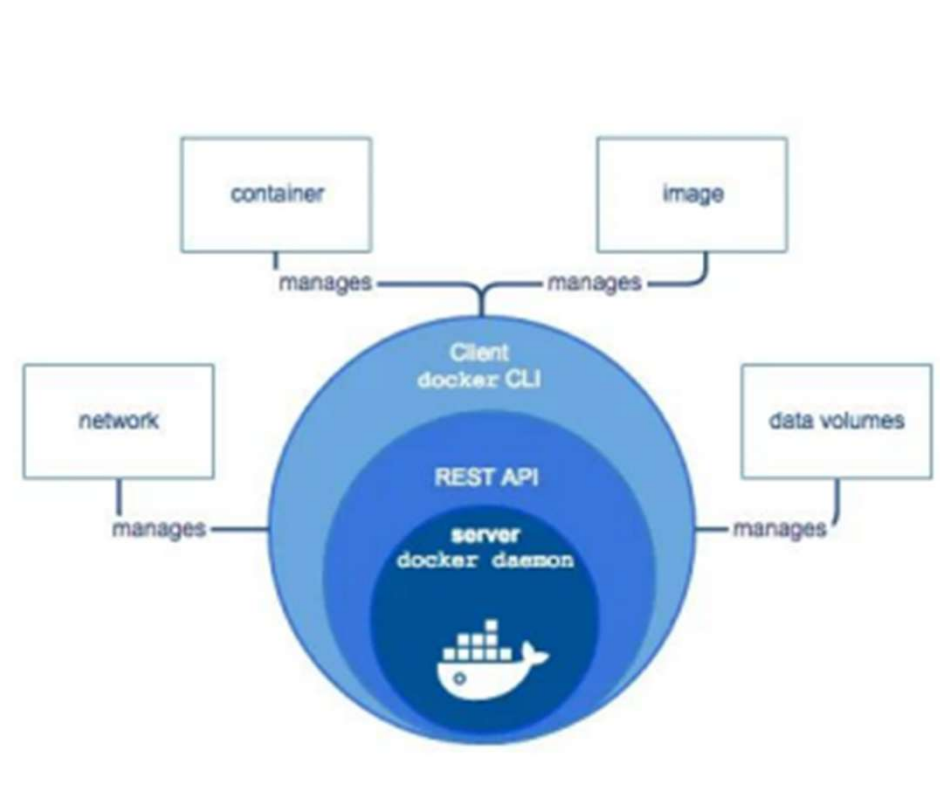
Docker	Virtual Machines
All containers share the same kernel of the host	Each VM runs its own OS
Containers instantiate in seconds	Boots uptime is in minutes
Images can be diffed and can be version controlled. Dockerhub is like GitHub	Not effective diffs. Not version controlled
Can run many Docker containers on a laptop.	Cannot run more than a couple of VMS on an average laptop
Multiple Docker containers can be started from one Docker image	Only one VM can be started from one set of VMX and VMDK files



Docker Architecture

Docker Architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.



Terminology

Terminology

Docker Editions

- **Docker Community Edition (CE)** is ideal for Developers who are looking for experimenting with docker and creating container-based applications. It's free.
- **Docker Enterprise Edition (EE)** is a Containers-as-a-Service (CaaS) platform. Enterprise Edition Subscription packages include an integrated Docker platform and tooling for container management and security

Terminology

Registry

•A Docker registry stores Docker images.

•**Docker Hub (Like GitHub)** is a cloud-based registry service that allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.

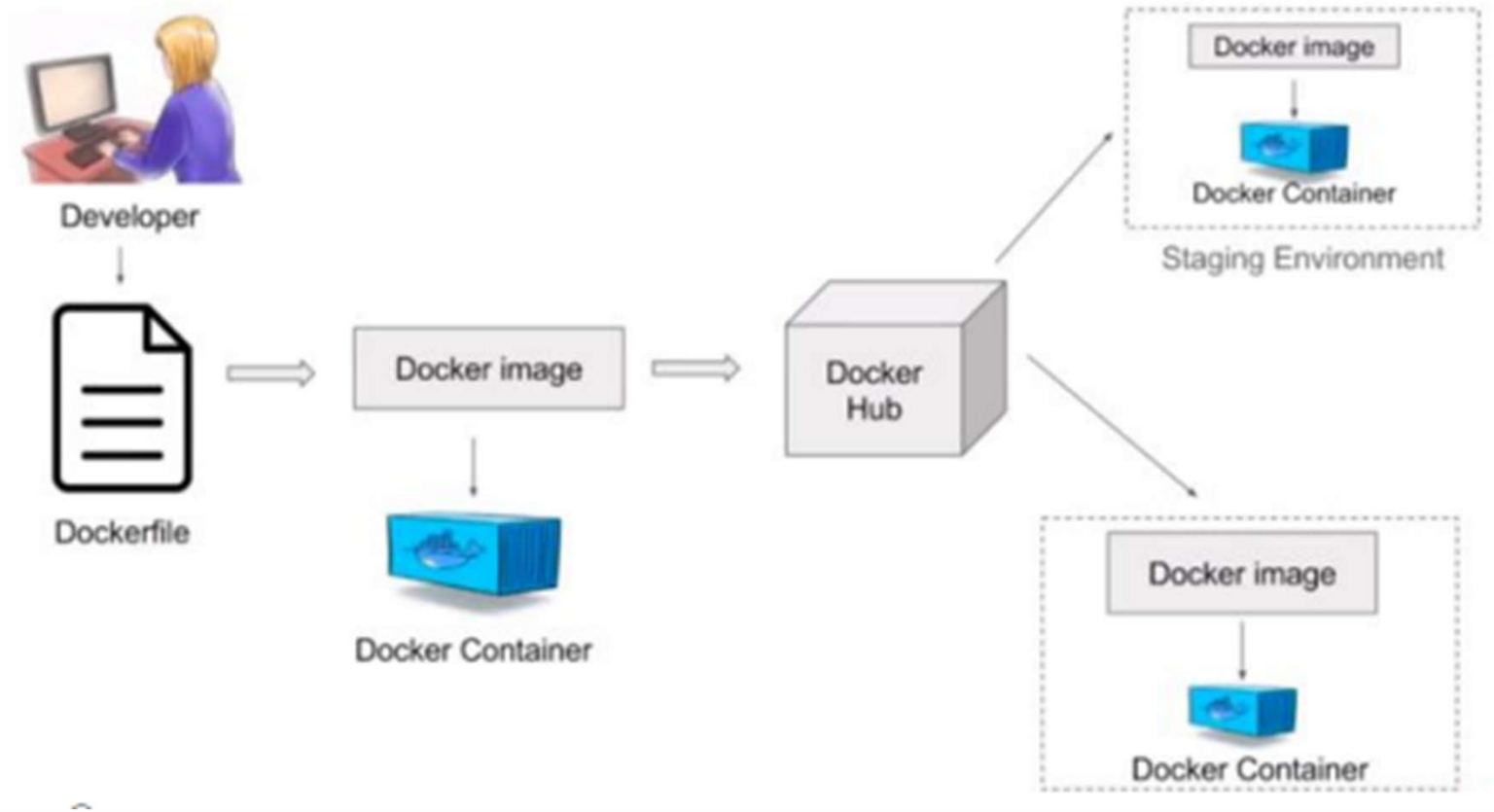
•**Docker Cloud** uses the hosted Docker Cloud Registry, which allows you to publish Dockerized images on the internet either publicly or privately. Docker Cloud can also store pre-built images, or link to your source code so it can build the code into Docker images, and optionally test the resulting images before pushing them to a repository.

Images and Containers

Images and Containers

- An image is a read-only template with instructions for creating a Docker container.
- A container is a runnable instance of an image.

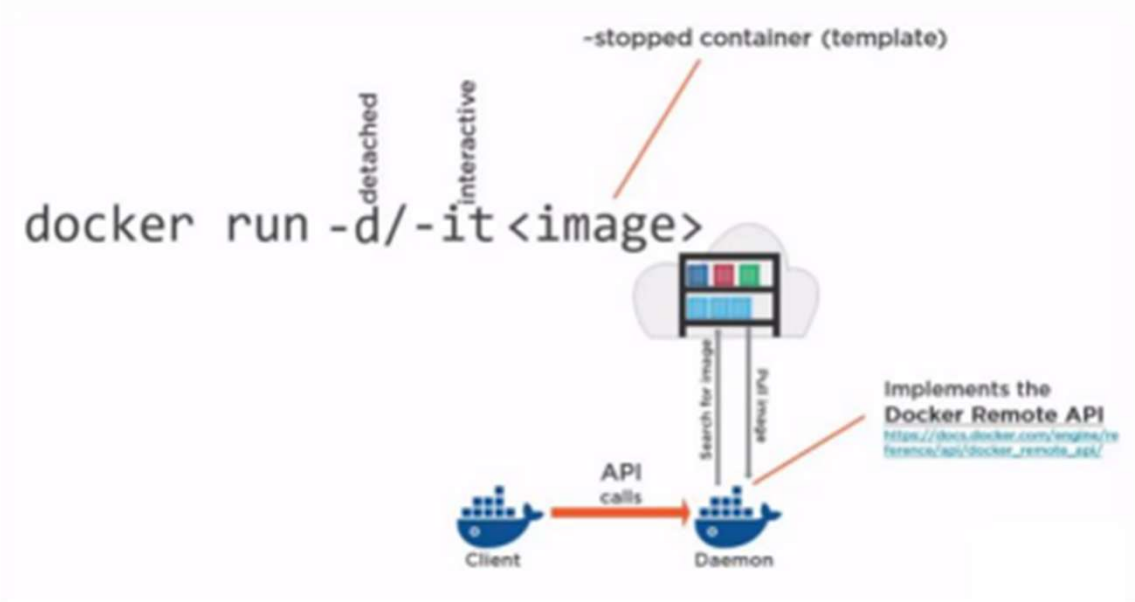
Images and Containers



Docker Commands

Docker Run Command

docker run command is used to create a container. The docker run command provides all of the "launch" capabilities for Docker.

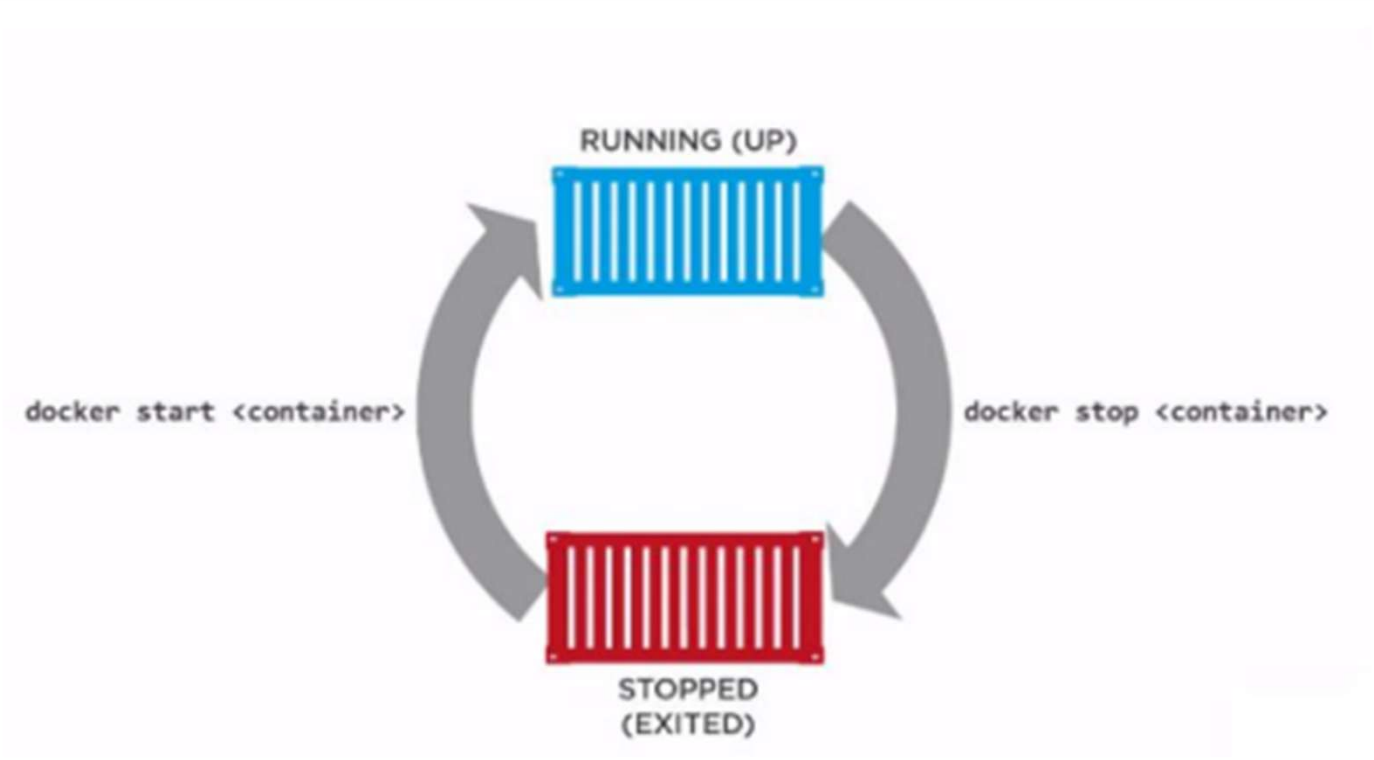


Docker Run Command

```
$ docker run -i -t ubuntu /bin/bash
```

When we run this command, the following happens. If you do not have the ubuntu image locally, Docker pulls it from your configured registry, as though you had run docker pull ubuntu manually. Docker creates a new container, as though you had run a docker container create command manually. Docker starts the container and executes /bin/bash. Because the container is running interactively and attached to your terminal (due to the -i and -t flags), you can provide input using your keyboard while the output is logged to your terminal. When you type exit to terminate the /bin/bash command, the container stops but is not removed. You can start it again or remove it.

Starting a stopped container



Container naming

```
$ sudo docker run --name techpro -i -t ubuntu /bin/bash
```

Docker will automatically generate a name at random for each container we create.
If we want to specify a particular container name in place of the automatically generated name, we can do so using the --name flag.

docker container Command

IMAGES

Docker images are a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Build an Image from a Dockerfile
```
docker build -t <image_name>
```

Build an Image from a Dockerfile without the cache
```
docker build -t <image_name> . -no-cache
```

List local images
```
docker images
```

Delete an Image
```
docker rmi <image_name>
```

Remove all unused images
```
docker image prune
```

DOCKER HUB

Docker Hub is a service provided by Docker for finding and sharing container images with your team. Learn more and find images at https://hub.docker.com

Login into Docker
```
docker login -u <username>
```

Publish an image to Docker Hub
```
docker push <username>/<image_name>
```

Search Hub for an image
```
docker search <image_name>
```

Pull an image from a Docker Hub
```
docker pull <image_name>
```

CONTAINERS

A container is a runtime instance of a docker image. A container will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Create and run a container from an image, with a custom name:
```
docker run --name <container_name> <image_name>
```

Run a container with and publish a container's port(s) to the host.
```
docker run -p <host_port>:<container_port> <image_name>
```

Run a container in the background
```
docker run -d <image_name>
```

Start or stop an existing container:
```
docker start|stop <container_name> (or <container-id>)
```

Remove a stopped container:
```
docker rm <container_name>
```

Open a shell inside a running container:
```
docker exec -it <container_name> sh
```

Fetch and follow the logs of a container:
```
docker logs -f <container_name>
```

To inspect a running container:
```
docker inspect <container_name> (or <container_id>)
```

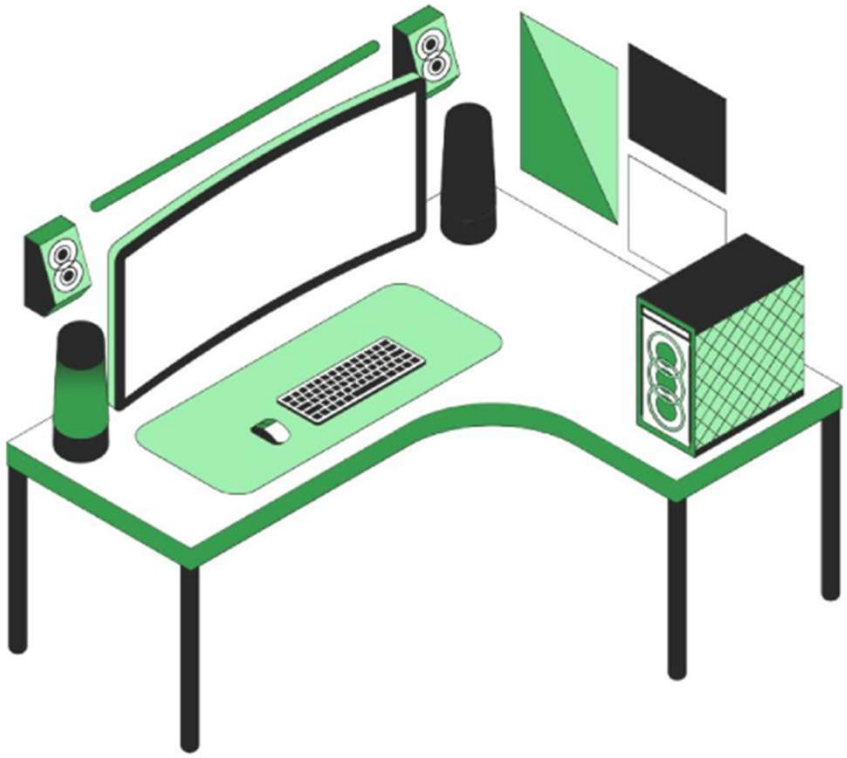
To list currently running containers:
```
docker ps
```

List all docker containers (running and stopped):
```
docker ps --all
```

View resource usage stats
```
docker container stats
```

docker container Command

Command	Description
docker container attach	Attach local standard input, output, and error streams to a running container
docker container create	Create a new container
docker container exec	Run a command in a running container
docker container inspect	Display detailed information on one or more containers
docker container ls	List containers
docker container prune	Remove all stopped containers
docker container rename	Rename a container
docker container rm	Remove one or more containers



Do you have any questions?

Send it to us! We hope you learned something new.