Building Software Systems

Lecture 2.6

Containers and Docker

SAURABH SRIVASTAVA

ASSISTANT PROFESSOR

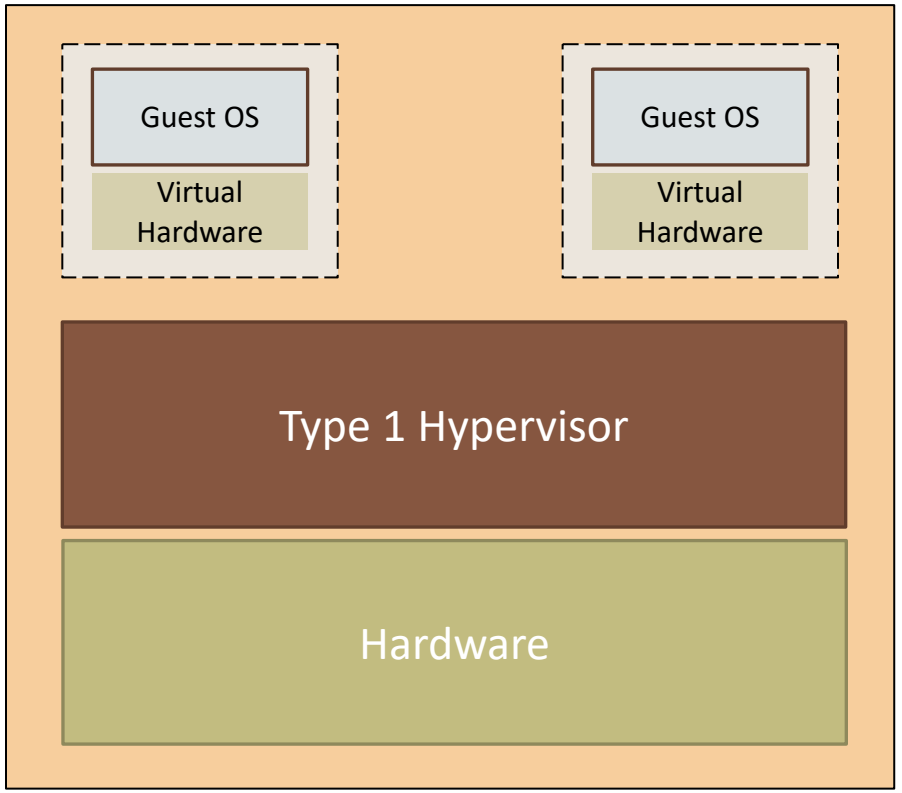
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IIT (ISM) DHANBAD

Revisit – Hypervisors (1/2)

Physical Node

Virtual Machine

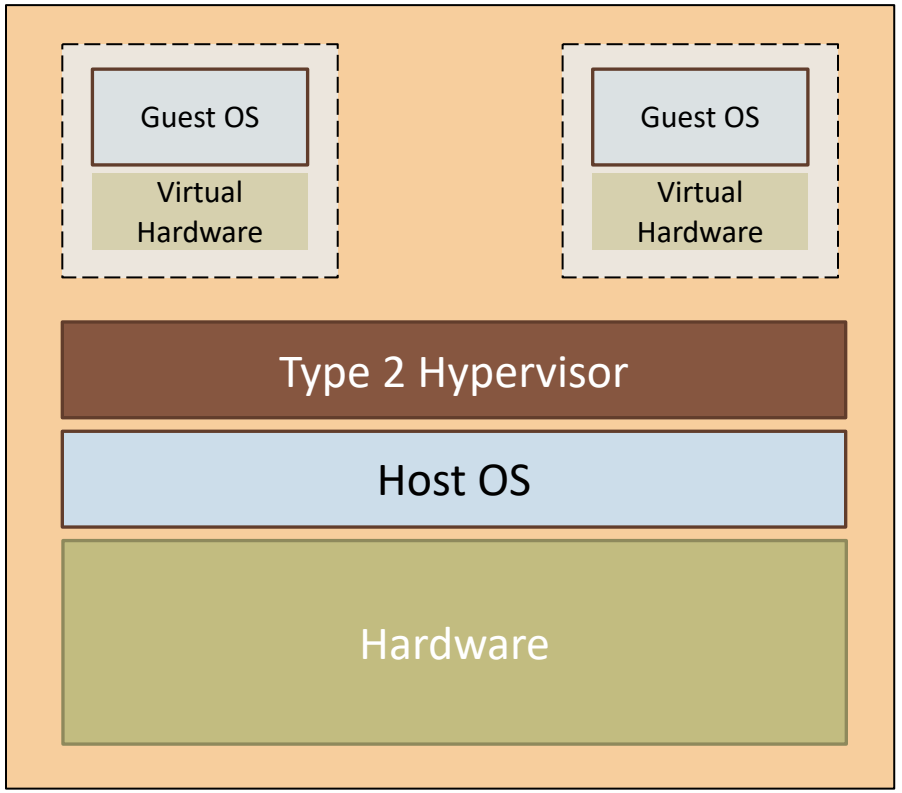


Type 1 hypervisors operate directly over the hardware

Revisit – Hypervisors (2/2)

Physical Node

Virtual Machine

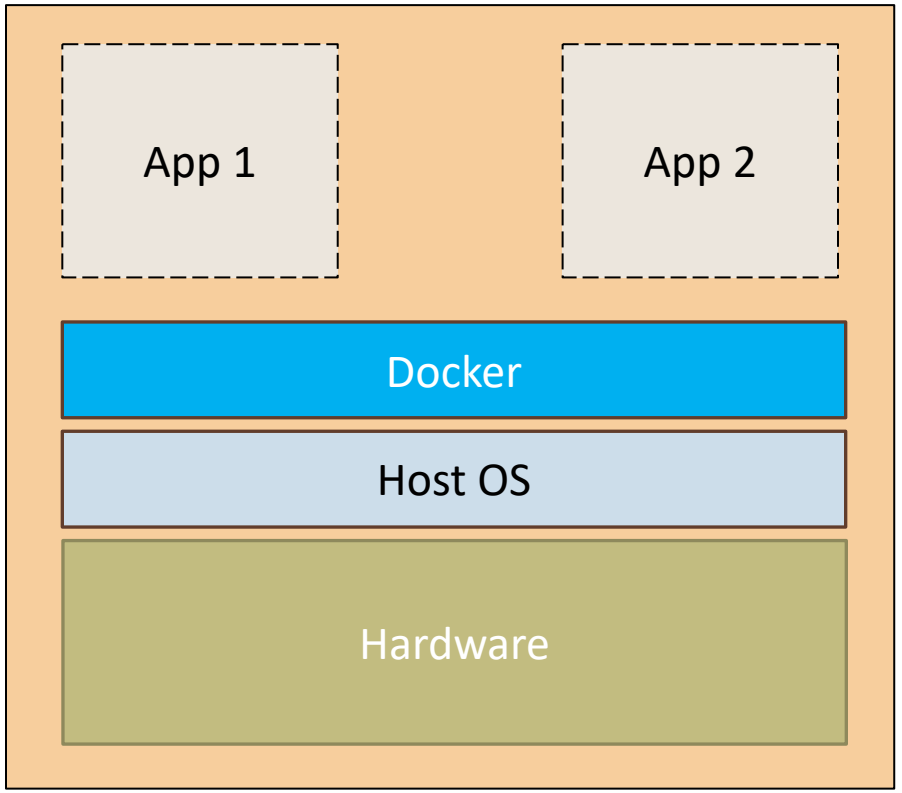


Type 2 hypervisors run over a host Operating System, like any other process

Introducing – Containers !!

Physical Node

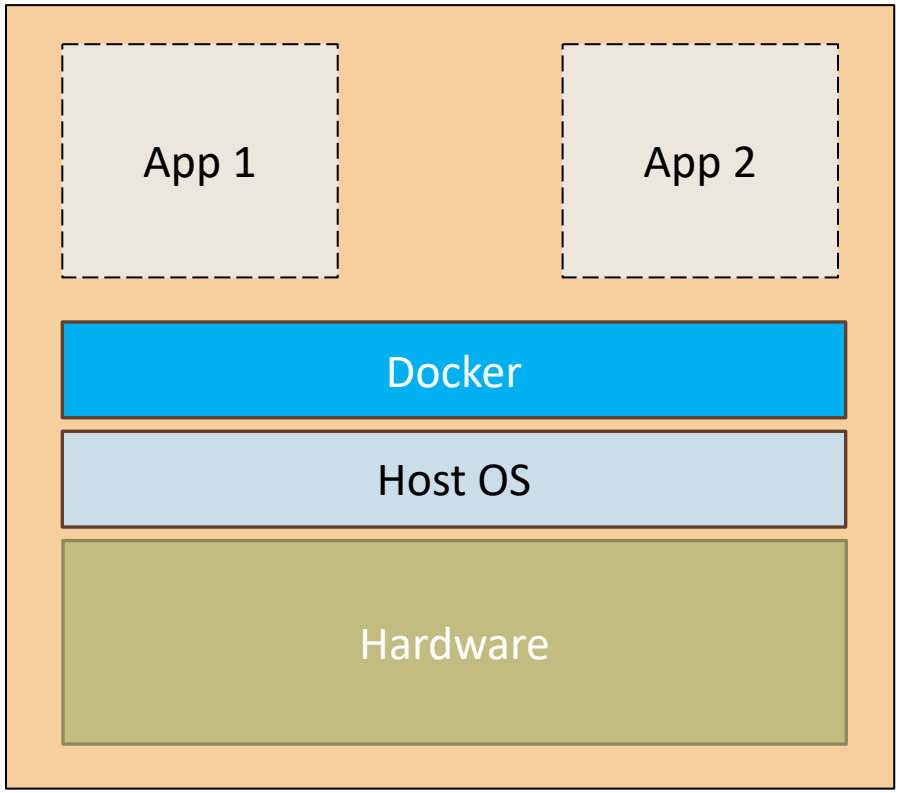
Container



Introducing – Containers !!

Physical Node

Container



Containers are essentially *OS-level* virtualization solutions

History of Containerisation (1/3)

Containerisation, as a concept, is actually quite old

chroot (1979)

- The concept of OS-level virtualization began with the introduction of *chroot* in Version 7 Unix
- *chroot* changed the root directory of a process and its children to a new location in the filesystem
- This was a rudimentary form of isolation, but it laid the groundwork for more advanced forms of containerisation

BSD Jails (2000)

- FreeBSD introduced the concept of "jails" in 2000
- Jails allowed system administrators to partition a FreeBSD-based computer system …
- … into several independent mini-systems called jails
- This was a significant step towards modern containerization …
- … as it provided an enhanced level of isolation compared to chroot

History of Containerisation (2/3)

Solaris Zones (2004)

- Solaris Zones were introduced by Sun Microsystems in Solaris Operating System 10
- It offered a more flexible and secure way to virtualize OS-level operations
- Zones allowed multiple secure and isolated environments (zones) to run on a single Solaris instance
- Zones has some more flexibility for container configuration (e.g., ability to to have read-only file systems)

Linux-VServer (2001) and OpenVZ (2005)

- Linux-VServer and OpenVZ were early attempts to provide containerization on the Linux platform
- They allowed multiple isolated Linux systems (containers) on a single physical server
- These technologies were precursors to more advanced Linux container platforms

LXC (Linux Containers) (2008)

- LXC, released in 2008, was an important milestone in the development of Linux-based containerization
- It combined *cgroups* (control groups) for resource limiting and namespaces for isolation …
- … to provide an environment that was similar to a virtual machine but with less overhead

History of Containerisation (3/3)

Docker (2013)

- Docker, introduced in 2013, revolutionized the concept of containerization
- It made containers more accessible and easier to use through its simple and efficient platform, …
- … which allowed for portable, lightweight, and consistent environments …
- … for developing, shipping, and running applications (we will discuss them in a minute)

Kubernetes (2014)

- Originally developed by Google and later donated to the Cloud Native Computing Foundation, …
- … Kubernetes emerged as a powerful orchestration system for Docker and other container technologies
- It provides automated deployment, scaling, and management of containerized applications

Modern Day Containerisation

Since the evolution of Dockers, containers have become ubiquitous

Containers are the de-facto standard today for providing *deployment artifacts*

- You need to create appropriate *container images* (e.g., Docker Images) that contain your code/environment

Combined with the power of public clouds, containers run multiple, complex systems seamlessly

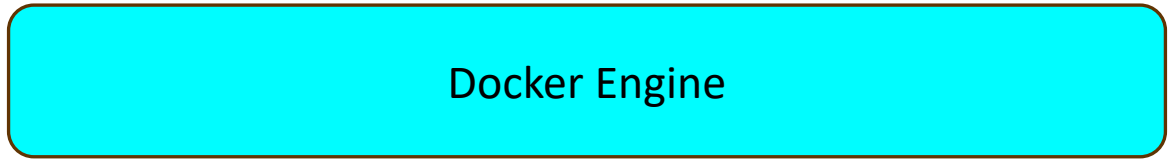
Remember Microservices?

- We had a session on them a few weeks back
- Instead of creating an *instance* of a Microservice as a VM, they are usually spawned as a container
- These containers can communicate with each other to achieve a common goal

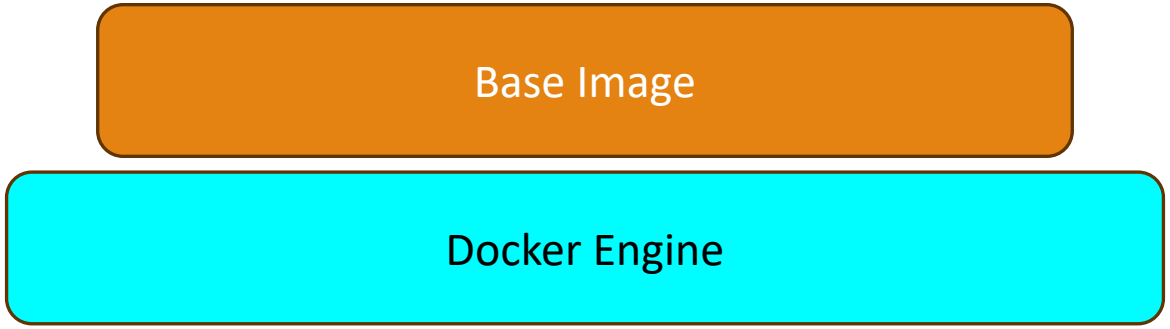
"Containers over VM" configuration

- Most modern-day containers on public cloud, usually run over a VM (and not over physical nodes directly)
- While VMs are a nice way to abstract hardware, containers provide further isolation for better optimisation

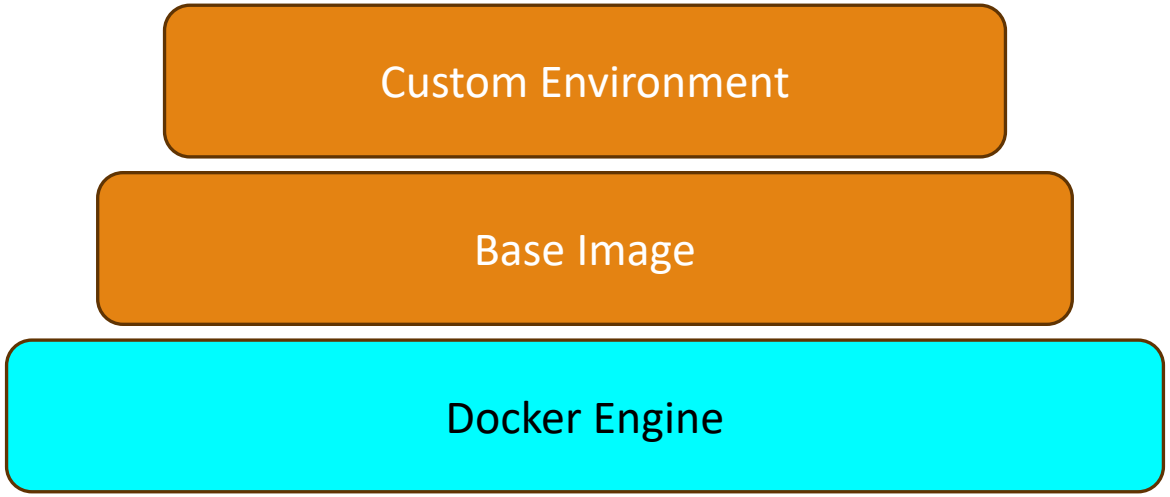
The Layered Approach to Docker Images



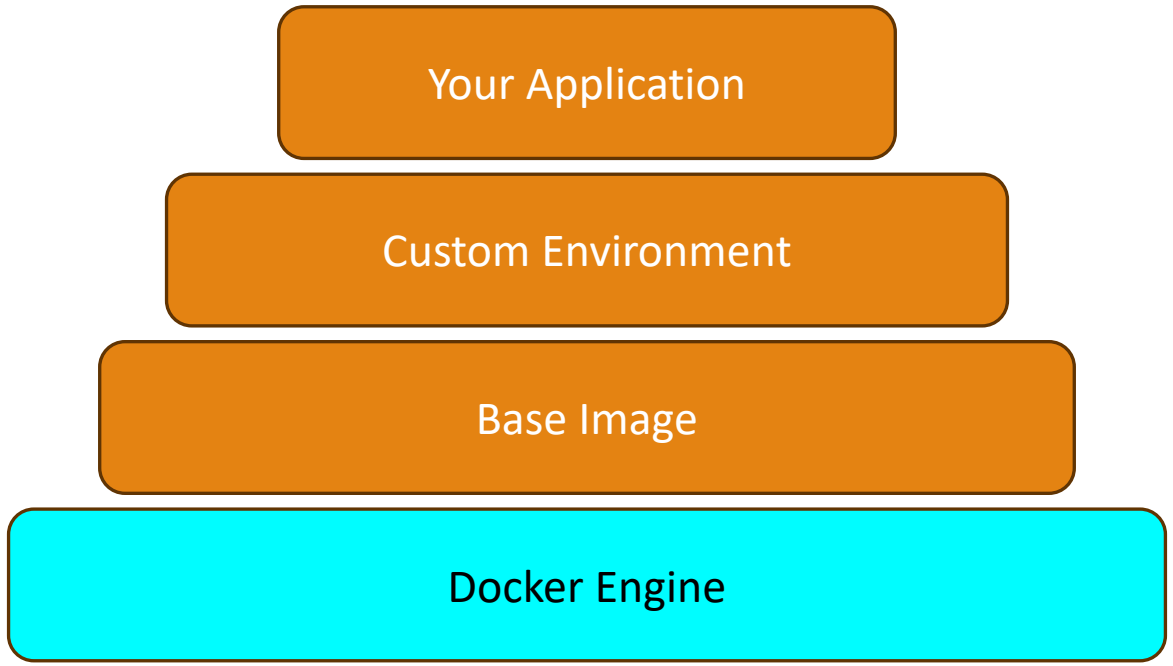
The Layered Approach to Docker Images



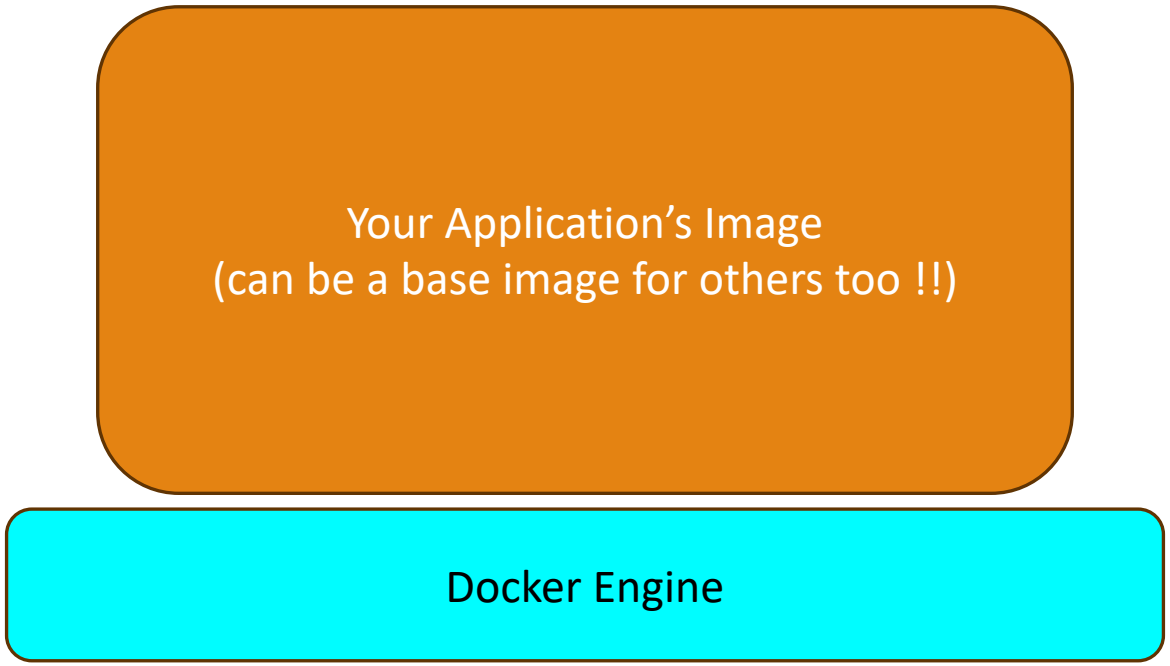
The Layered Approach to Docker Images



The Layered Approach to Docker Images



The Layered Approach to Docker Images



Docker Engine (1/3)

Docker Engine is the core component of the Docker platform

- It's a lightweight and powerful open-source containerization technology
- It enables the creation and running of Docker containers

Docker Engine is a *client-server* application with three major components

1. Server

- The Docker daemon (dockerd) is the server part of the Docker Engine
- It's a process that manages Docker objects such as images, containers, networks, and volumes
- The daemon listens for Docker API requests and manages Docker services

Docker Engine (2/3)

Docker Engine is the core component of the Docker platform

- It's a lightweight and powerful open-source containerization technology
- It enables the creation and running of Docker containers

Docker Engine is a *client-server* application with three major components

2. REST API

- The REST API provides an interface for programs to interact with the Docker daemon
- It can be accessed by an HTTP client, or by the Docker CLI, to communicate with the daemon

Docker Engine (3/3)

Docker Engine is the core component of the Docker platform

- It's a lightweight and powerful open-source containerization technology
- It enables the creation and running of Docker containers

Docker Engine is a *client-server* application with three major components

3. Command Line Interface (CLI)

- The Docker CLI (docker) is the client part that users interact with
- It allows users to send commands to the Docker daemon, …
- … such as building images, running containers, and managing Docker objects
- The CLI uses the Docker API to communicate with the daemon

Docker Hub (1/3)

One of the best support feature of Dockers is its online repository – Docker Hub

- Docker Hub is a cloud-based service that provides a centralized resource …
- … for container image discovery, distribution, and change management
- Some key features of the same are as follows

Image Repositories

- Docker Hub allows users to host and manage their own Docker image repositories
- These repositories can be public, allowing anyone to access and use the images, …
- … or private, restricting access to specified users or organizations

Official Images

- Docker Hub hosts "official images" that are provided and maintained by Docker, Inc., or its partners
- These images are verified for security and best practices
- It covers a wide range of popular software like operating systems, databases, and application frameworks

Docker Hub (2/3)

One of the best support feature of Dockers is its online repository – Docker Hub

- Docker Hub is a cloud-based service that provides a centralized resource …
- … for container image discovery, distribution, and change management
- Some key features of the same are as follows

User and Organization Management

- Users can create their own Docker Hub accounts to manage their images and collaborate with others
- Docker Hub also supports organizations and teams, enabling multiple users to collaborate on shared repositories

Automated Builds

- Docker Hub offers automated build services, which automatically create new image versions …
- … from source code repositories like GitHub or Bitbucket
- Whenever changes are pushed to the source repository, …
- … Docker Hub can automatically build and update the corresponding Docker image

Docker Hub (3/3)

One of the best support feature of Dockers is its online repository – Docker Hub

- Docker Hub is a cloud-based service that provides a centralized resource …
- … for container image discovery, distribution, and change management
- Some key features of the same are as follows

Version Control and Tags

- Docker images in Docker Hub can be tagged with specific versions, …
- … allowing users to track and roll back to different versions of an image
- This is crucial for maintaining consistent and stable environments

Docker Hub API

- Docker Hub provides an API for programmatically interacting with its services, …
- … enabling automation and integration with other tools and systems

Docker CLI (1/3)

The Docker Command Line Interface (CLI) is a powerful tool that allows users to interact with Docker

- It can be used to manage Docker containers, images, volumes, and networks
- Docker CLI commands typically follow the structure: *docker [command] [sub-command] [options]*
- Some common CLI commands are as follows

Container Management:

- Run Containers: *docker run* creates and starts a container from an image
- List Containers: *docker ps* lists running containers, and docker ps -a lists both running and stopped containers
- Stop/Start/Restart Containers: *docker stop*, *docker start*, and *docker restart* control the state of containers
- Remove Containers: *docker rm* removes one or more stopped containers

Docker CLI (2/3)

The Docker Command Line Interface (CLI) is a powerful tool that allows users to interact with Docker

- It can be used to manage Docker containers, images, volumes, and networks
- Docker CLI commands typically follow the structure: *docker [command] [sub-command] [options]*
- Some common CLI commands are as follows

Image Management:

- Pull Images: *docker pull* downloads an image from a registry like Docker Hub
- List Images: *docker images* lists images that are locally stored
- Build Images: *docker build* creates an image from a Dockerfile
- Remove Images: *docker rmi* removes one or more images

Docker CLI (3/3)

The Docker Command Line Interface (CLI) is a powerful tool that allows users to interact with Docker

- It can be used to manage Docker containers, images, volumes, and networks
- Docker CLI commands typically follow the structure: *docker [command] [sub-command] [options]*
- Some common CLI commands are as follows

Networking:

- Create Network: *docker network create* creates a new network
- List Networks: *docker network ls* lists all networks available in Docker
- Connect Container to Network: *docker network connect* attaches a container to an existing network
- Disconnect Container from Network: *docker network disconnect* detaches a container from a network
- Remove Network: *docker network rm* removes one or more networks

Docker Desktop

Docker Desktop is an easy-to-install application for Mac, Linux and Windows environments

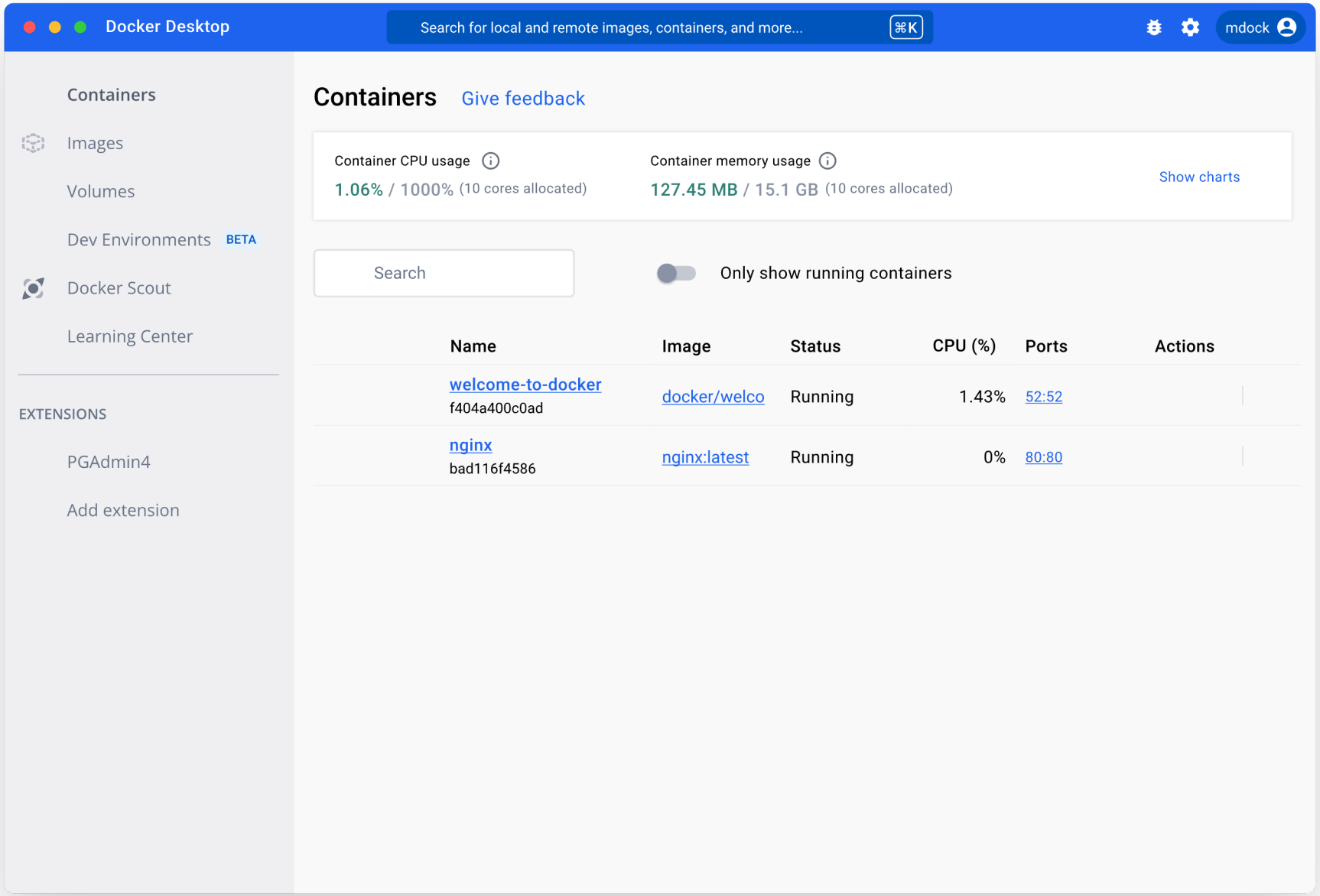
- It can be installed similar to any other application on these platforms

It provides an integrated development environment for both development and production purposes

- It includes Docker daemon, Docker CLI (Command Line Interface), Docker Compose, and other essential tools

It also provides you a GUI for Docker

- While Docker can be fully operated through the command line, Docker Desktop provides a graphical interface
- This interface makes it easier to manage containers, images, networks, and volumes
- It is especially helpful for those who are new to Docker



Source: Docker Desktop

Docker Desktop

Docker Desktop is an easy-to-install application for Mac, Linux and Windows environments

- It can be installed similar to any other application on these platforms

It provides an integrated development environment for both development and production purposes

- It includes Docker daemon, Docker CLI (Command Line Interface), Docker Compose, and other essential tools

It also provides you a GUI for Docker

- While Docker can be fully operated through the command line, Docker Desktop provides a graphical interface
- This interface makes it easier to manage containers, images, networks, and volumes
- It is especially helpful for those who are new to Docker

Networking Features

- One of the harder parts for a beginner with virtualisation is the Networking
- Docker Desktop includes powerful networking capabilities, allowing you to configure network settings easily
- This includes creating custom networks for your containers, which is essential for complex application setups

Homework

Create Docker Images for the Frontend and Backend apps we created during Lecture 2.5

- The README file and Docker Configuration files are provided as resources
- The link to the Resource Folder is: https://drive.google.com/drive/folders/1nzul02fnqZj1ta-zTOGmbtJaVaP9jF_U?usp=sharing