Containerization & Orchestration

Docker & Kubernetes

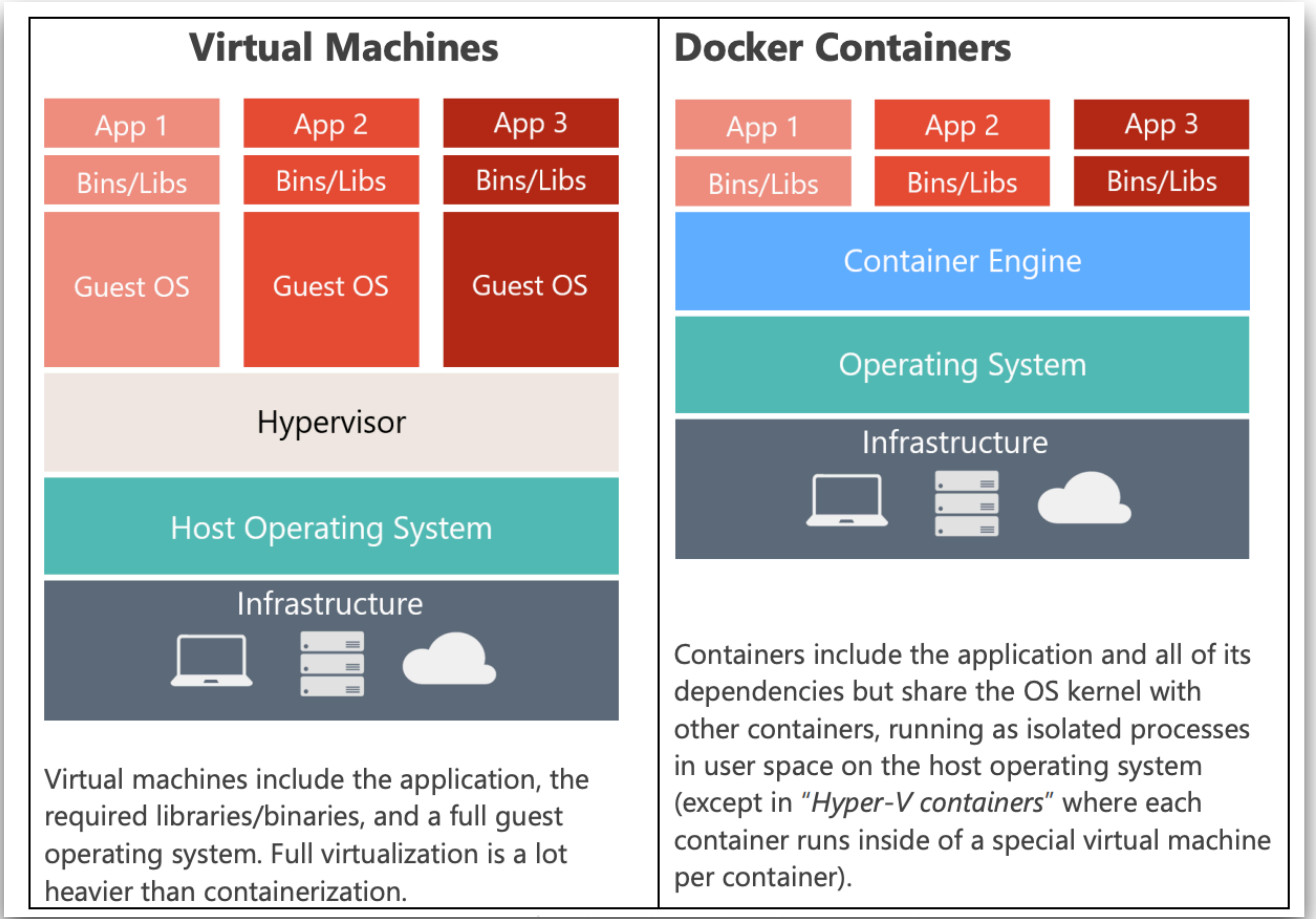
Chakravarthy B

What is Containerization?

- Packaging together of software code with all it’s necessary components like libraries, frameworks, and other dependencies so that they are isolated in their own “container."
- The software or application within the container can be moved and run consistently in any environment and on any infrastructure, independent of that environment or infrastructure’s operating system
- The idea of process isolation has been around for years, but when Docker introduced Docker Engine in 2013
- Today developers can choose from a selection of containerization platforms and tools—like Podman, Buildah, and Skopeo—that support the Open Container Initiative standards pioneered by Docker
- A **container** is a loosely isolated environment that allows us to build and run software packages. These software packages include the code and all dependencies to run applications quickly and reliably on any computing environment. We call these packages *container images*.

What is software containerization?

- Software containerization is an OS virtualization method that is used to deploy and run containers without using a virtual machine (VM)
- Containers can run on physical hardware, in the cloud, VMs, and across multiple OSs.

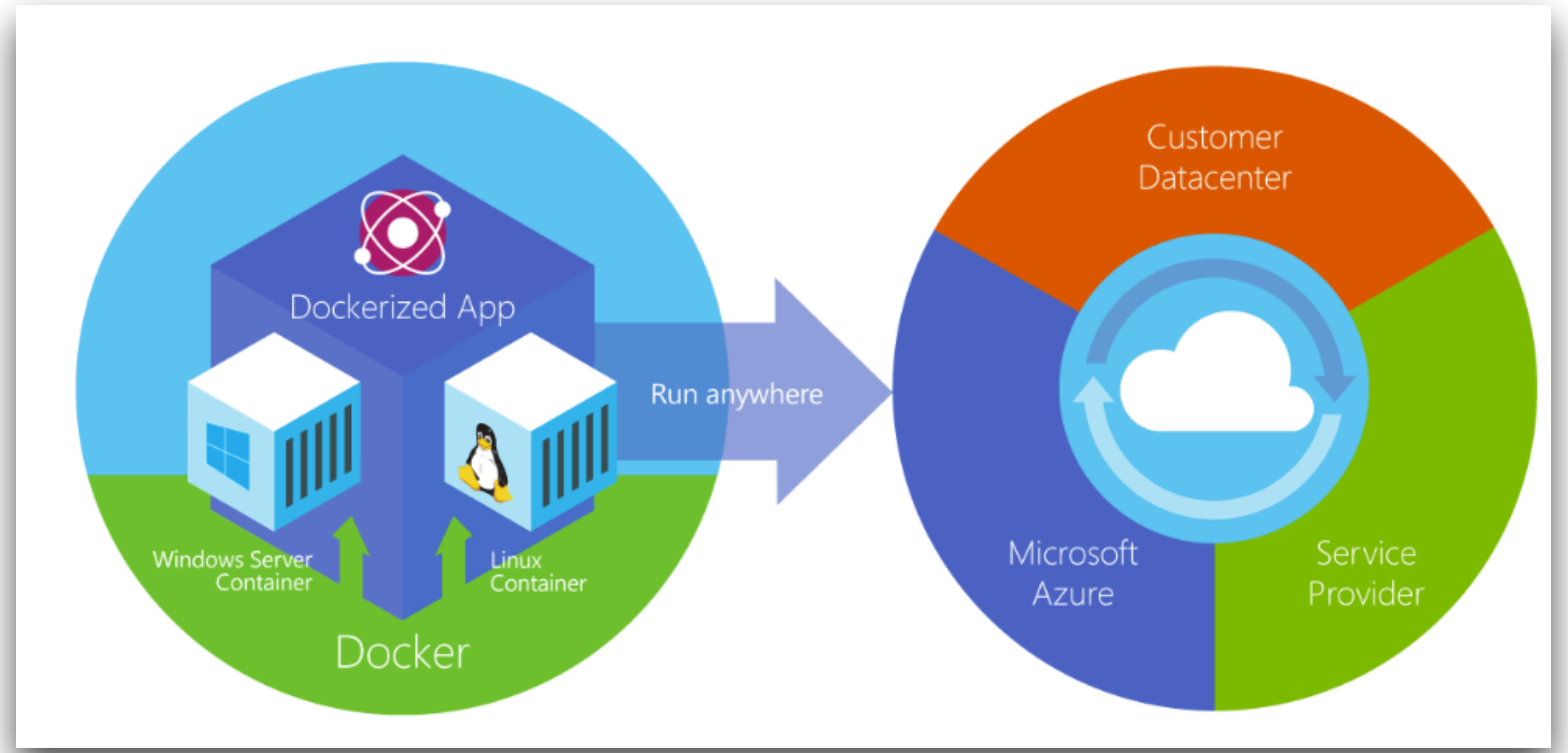


Virtual machines include the application, the required libraries/binaries, and a full guest operating system. Full virtualization is a lot heavier than containerization.

Containers include the application and all of its dependencies but share the OS kernel with other containers, running as isolated processes in user space on the host operating system (except in "*Hyper-V containers*" where each container runs inside of a special virtual machine per container).

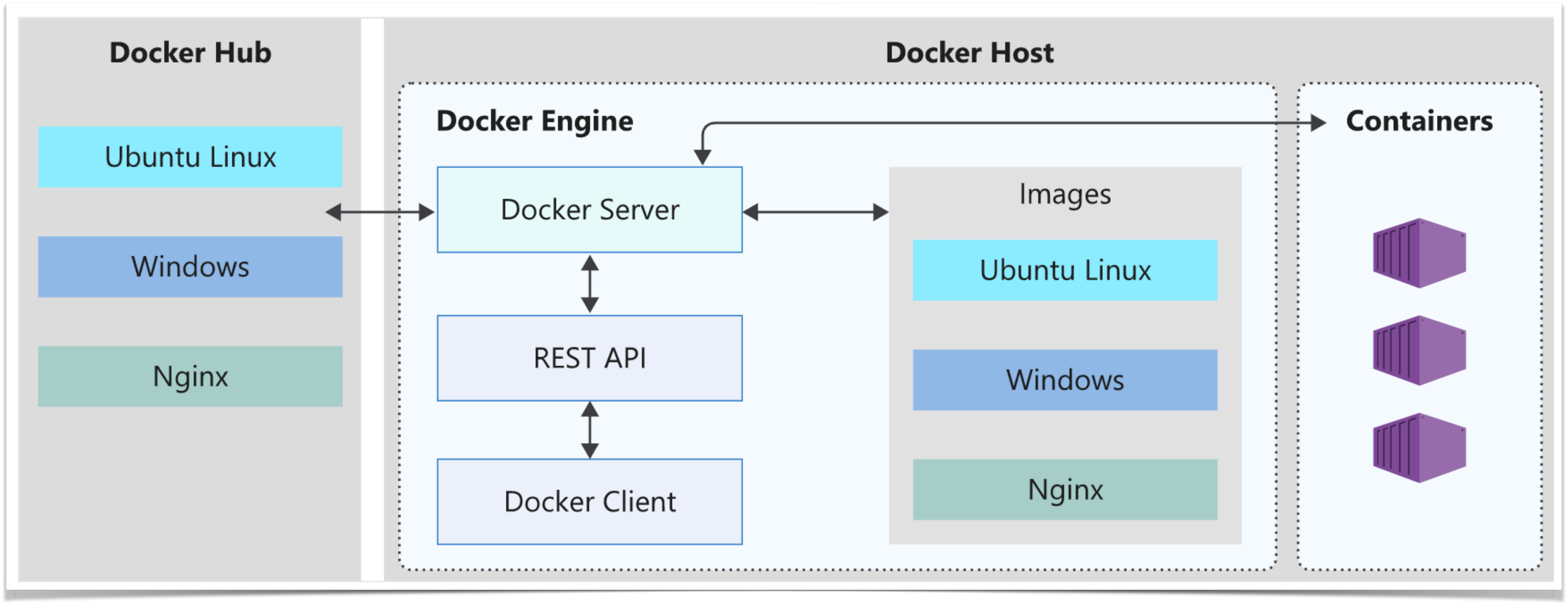
What is Docker ?

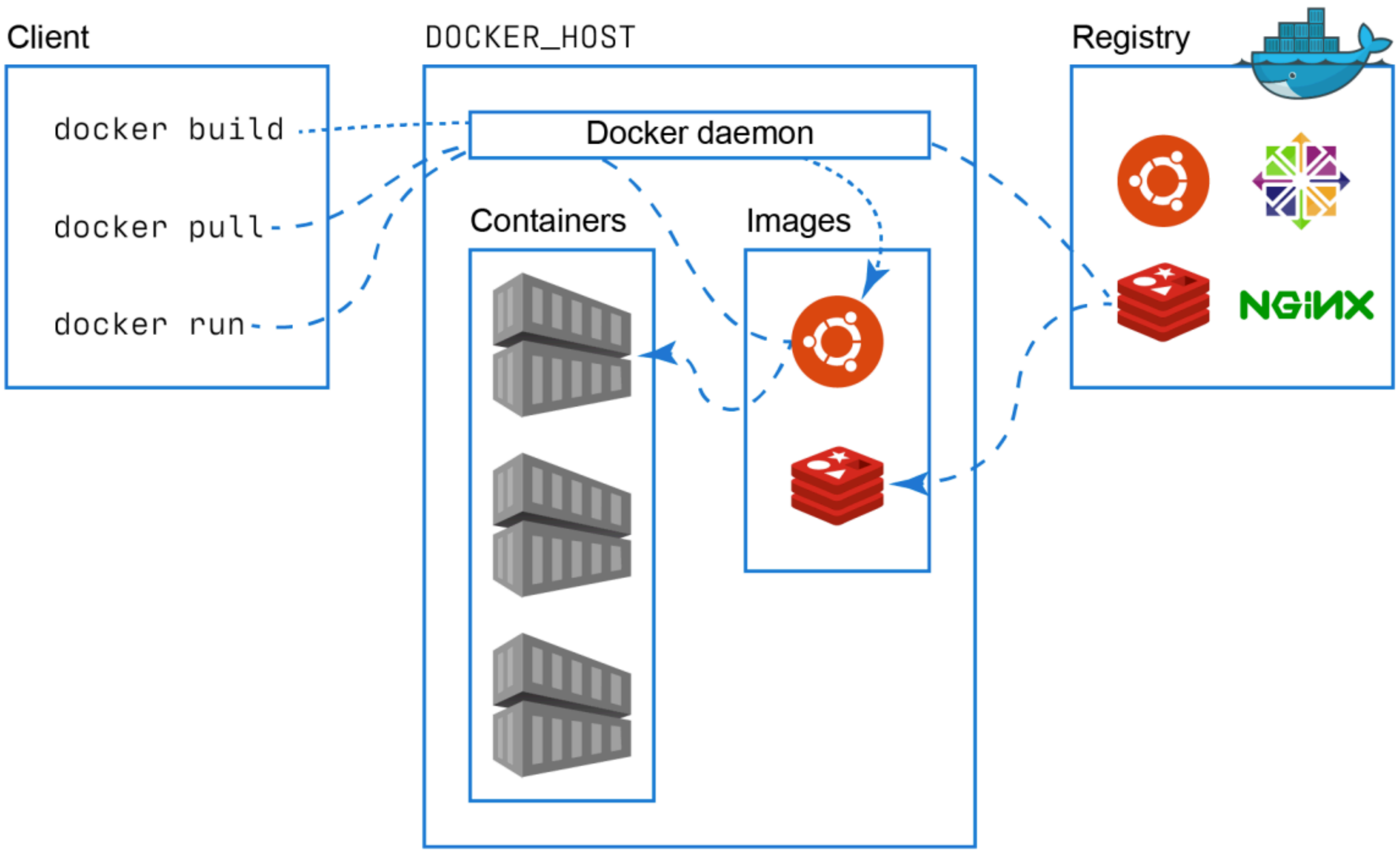
- Docker is a containerization platform used to develop, ship, and run containers
- Docker doesn't use a hypervisor, and you can run Docker on your desktop or laptop if you're developing and testing applications
- The desktop version of Docker supports Linux, Windows, and macOS. For production systems.
- Docker is available for server environments, including many variants of Linux and Microsoft Windows Server 2016 and above



Docker Architecture

- The Docker platform consists of several components that we use to build, run, and manage our containerized applications




Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Docker Architecture

- **Docker Engine**: The Docker Engine consists of several components configured as a client-server implementation where the client and server run simultaneously on the same host

- **Docker Client:** The Docker client is a command-line application named `docker` that provides us with a command line interface (CLI) to interact with a Docker server. The `docker` command uses the Docker REST API to send instructions to either a local or remote server and functions as the primary interface we use to manage our containers.

- **Docker Server**: The Docker server is a daemon named `dockerd`. The `dockerd` daemon responds to requests from the client via the Docker REST API and can interact with other daemons. The Docker server is also responsible for tracking the lifecycle of our containers

- **Docker Objects:** There are several objects that you'll create and configure to support your container deployments. These include networks, storage volumes, plugins, and other service objects

- **Docker Hub:** Docker Hub is a Software as a Service (SaaS) Docker container registry. Docker registries are repositories that we use to store and distribute the container images we create. Docker Hub is the default public registry Docker uses for image management.

What is Container Image?

- A container image is a portable package that contains software
- It's this image that, when run, becomes our container
- A container image is immutable. Once you've built an image, the image can't be changed
- The only way to change an image is to create a new image
- This feature is our guarantee that the image we use in production is the same image used in development and QA.

Docker Setup

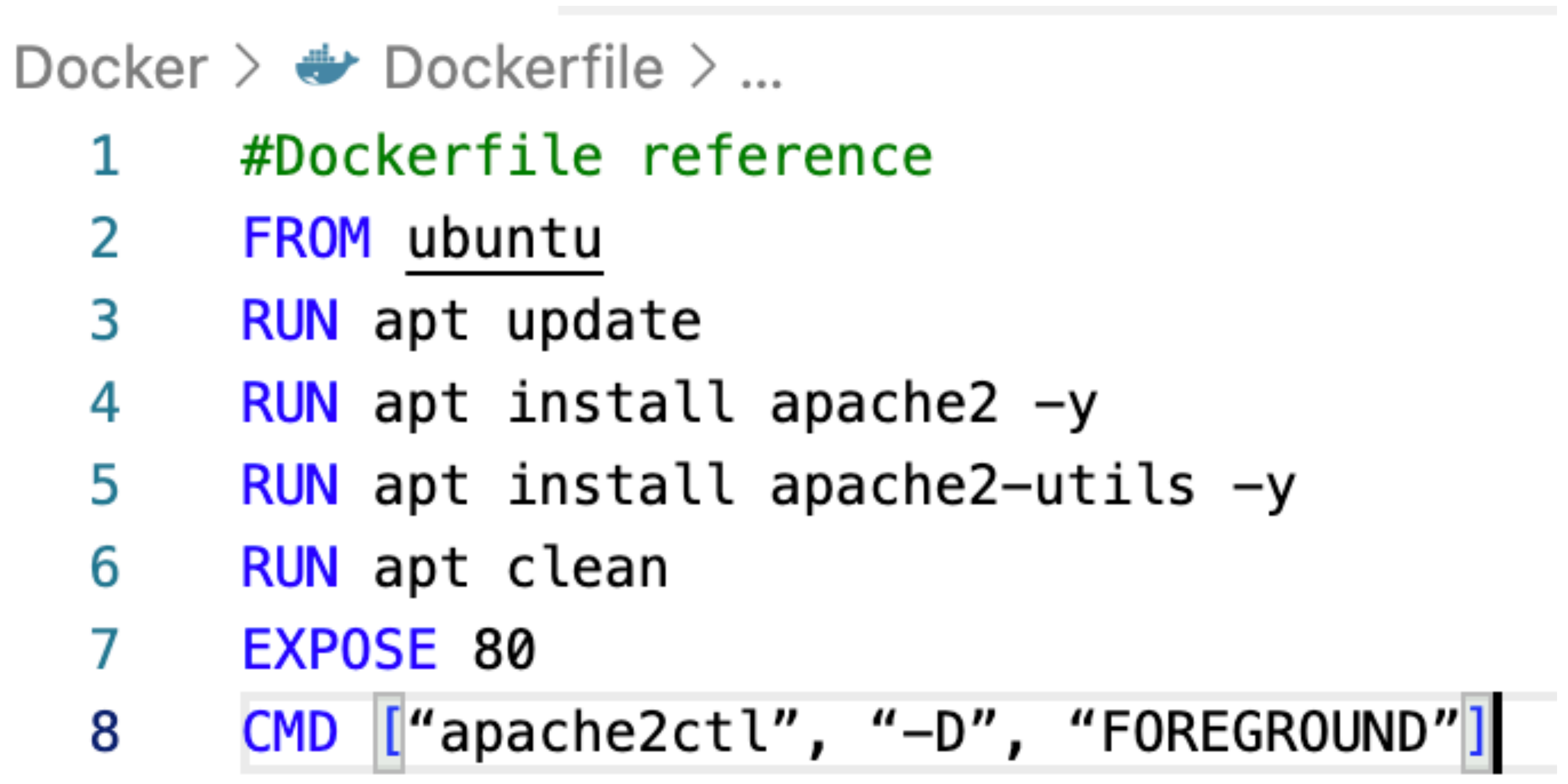
- Download the docker desktop from https://www.docker.com/products/docker-desktop/
- 4 GB RAM
- 64-bit kernel and CPU support for virtualization

Docker Desktop

Install Docker Desktop – the fastest way to containerize applications.

What is Dockerfile

- A Dockerfile is a text file that contains the instructions we use to build and run a Docker image
- Commands to update the base OS and install additional software
- Build artifacts to include, such as a developed application
- Services to expose, such as storage and network configuration



```dockerfile
#Dockerfile reference
FROM ubuntu
RUN apt update
RUN apt install apache2 -y
RUN apt install apache2-utils -y
RUN apt clean
EXPOSE 80
CMD [“apache2ctl”, “-D”, “FOREGROUND”]
```

Docker Terminology

- **Docker Image**: Docker images are the basis of containers. An Image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. An image typically contains a union of layered filesystems stacked on top of each other. An image does not have state and it never changes as it's deployed to various environments.

- **Container**: A container is a runtime instance of a Docker image. A Docker container consists of: A Docker image, an execution environment and a standard set of instructions.

- **Tag:** A tag is a label applied to a Docker image in a repository. Tags are how various images in a repository are distinguished from each other. They are commonly used to distinguish between multiple versions of the same image

- **Build:** build is the process of building Docker images using a Dockerfile. The build uses a Dockerfile and a "context". The context is the set of files in the directory in which the image is built. Builds can be done with commands like "docker build".

- **Registry:** A Registry is a hosted service containing repositories of images which responds to the Registry API.

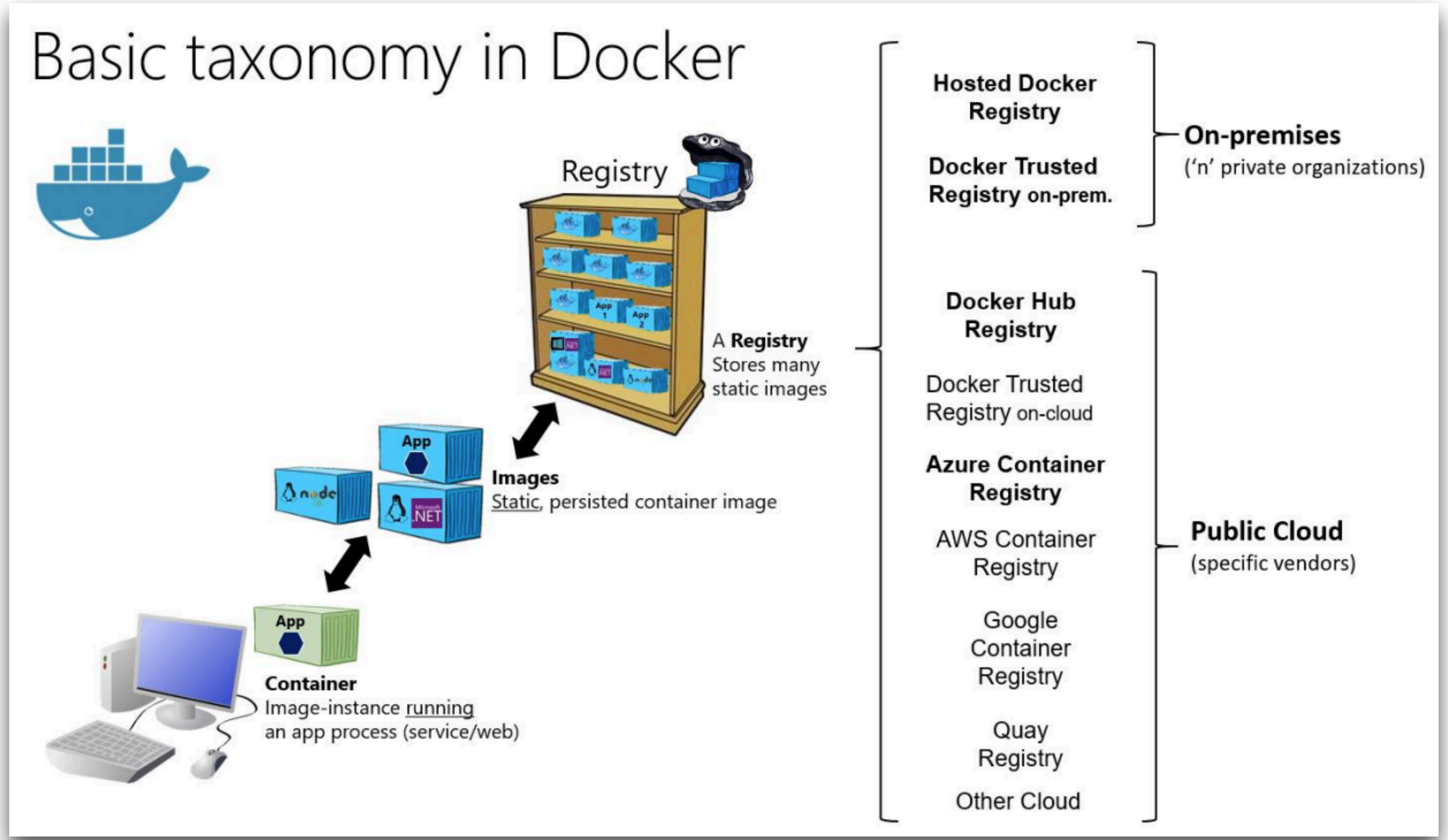
Docker Terminology

- **Docker Hub**: Docker Hub is the public instance of a registry. Equivalent to the public GitHub compared to GitHub enterprise where customers store their code in their own environment
- **Compose:**
  - Compose is a tool for defining and running multi container applications.
  - With compose, you define a multi-container application in a single file, then spin your application up in a single command which does everything that needs to be done to get it running.
  - Docker-compose.yml files are used to build and run multi container applications, defining the build information as well the environment information for interconnecting the collection of container.
- **Cluster:**
  - A Docker cluster pools together multiple Docker hosts and exposes them as a single virtual Docker host so it is able to scale-up to many hosts very easily.
  - Examples of Docker clusters can be created with Docker Swarm, Mesosphere DC/OS, Google Kubernetes and Azure Service Fabric

Docker Terminology

- **Orchestrator:**
  - A Docker Orchestrator simplifies management of clusters and Docker hosts
  - These Orchestrators enable users to manage their images, containers and hosts through a user interface
  - This interface allows users to administer container networking, configurations, load balancing, service discovery, High Availability, Docker host management and a much more
  - An orchestrator is responsible for running, distributing, scaling and healing workloads across a collection of nodes.
  - Typically, Orchestrator products are the same products providing the cluster infrastructure like Mesosphere DC/OS, Kubernetes, Docker Swarm and Azure Service Fabric.

Basic Docker taxonomy: containers, images, and registries


Basic taxonomy in Docker
Registry
A Registry Stores many static images
Images Static, persisted container image
Container Image-instance running an app process (service/web)
App
node
.NET
Hosted Docker Registry
Docker Trusted Registry on-prem.
On-premises ('n' private organizations)
Docker Hub Registry
Docker Trusted Registry on-cloud
Azure Container Registry
AWS Container Registry
Google Container Registry
Quay Registry
Other Cloud
Public Cloud (specific vendors)

Docker Container Storage

- Container storage is temporary
- Containers can make use of two options to persist data.
    - The first option is to make use of *volumes.*
    - *The* second is *bind mounts*.

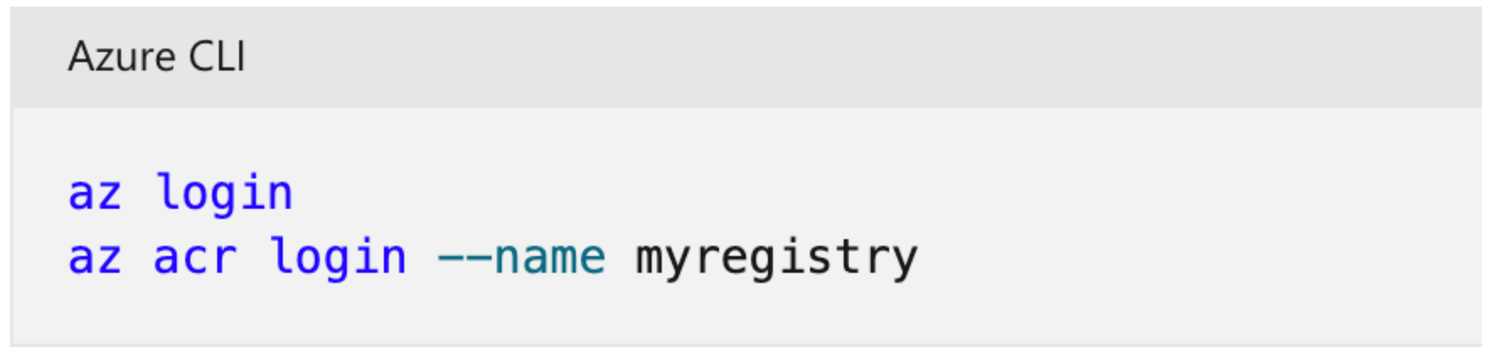
Docker volume

- A volume is stored on the host filesystem at a specific folder location.
- Choose a folder where you know the data isn't going to be modified by non-Docker processes
- Docker creates and manages the new volume by running the `docker volume create` command
- This command can form part of our Dockerfile definition, which means that you can create volumes as part of the container creation process.
- Docker will create the volume if it doesn't exist when you try to mount the volume into a container the first time
- Volumes are stored within directories on the host filesystem. Docker will mount and manage the volumes in the container. After mounting, these volumes are isolated from the host machine.
- Multiple containers can simultaneously use the same volumes. Volumes also don't get removed automatically when a container stops using the volume
- Example: $docker run -it -p 8080:80 -v /Users/kalyan/Documents/Batchoct21/Azure-GitRepo/Documents/LinuxOS/Docker/petshop:/var/www/html petshop:v2
-

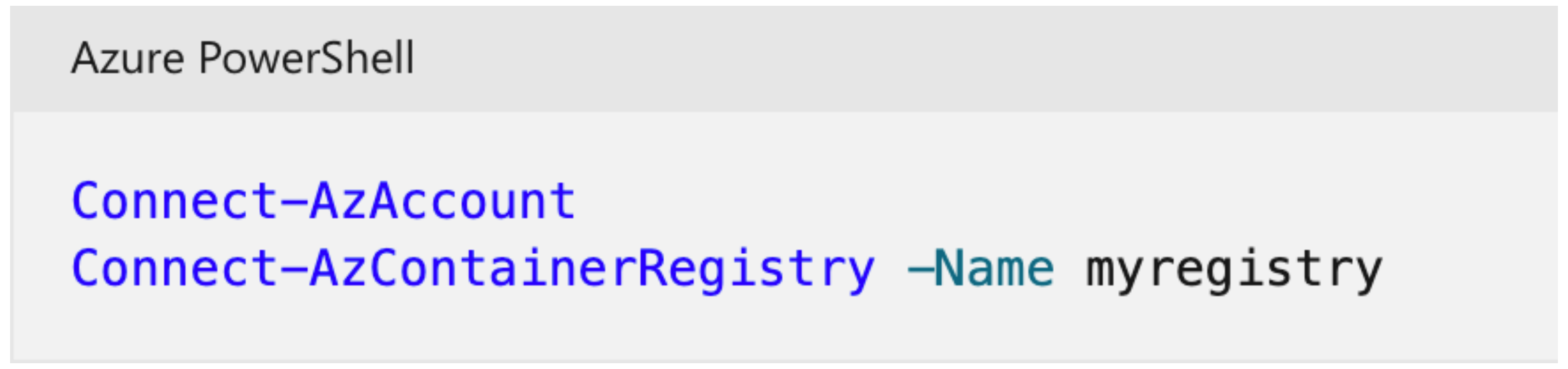
Azure Container Registry.

Azure
Container
Registry

- An Azure container registry stores and manages private container images and other artifacts, similar to the way Docker Hub stores public Docker container images
- You can use the Docker command-line interface (Docker CLI) for login, push, pull, and other container image operations on your container registry.

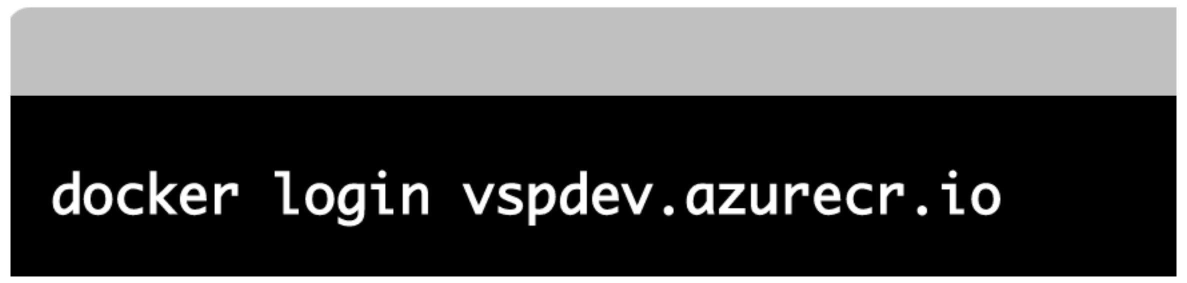

Azure CLI

```
az login
az acr login --name myregistry
```

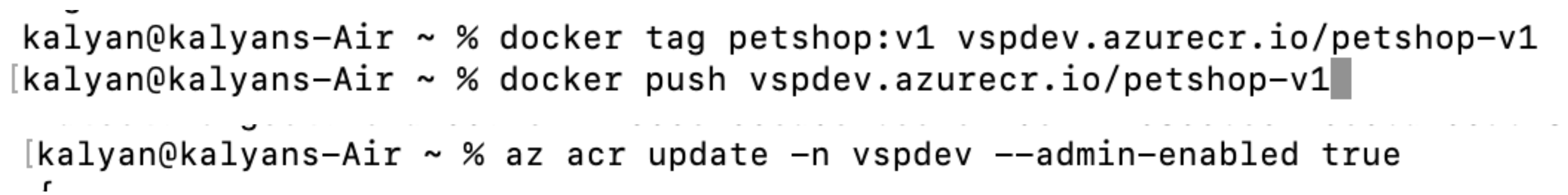

Azure PowerShell

```
Connect-AzAccount
Connect-AzContainerRegistry -Name myregistry
```

- Repo authentication & Tagging:

- 

```
docker login vspdev.azurecr.io
```



```
kalyan@kalyans-Air ~ % docker tag petshop:v1 vspdev.azurecr.io/petshop-v1
[kalyan@kalyans-Air ~ % docker push vspdev.azurecr.io/petshop-v1

[kalyan@kalyans-Air ~ % az acr update -n vspdev --admin-enabled true
```

https://learn.microsoft.com/en-gb/azure/container-registry/container-registry-authentication?tabs=azure-cli#admin-account