UNIVERSIDADE
BEIRA INTERIOR

# Containers

## Tecnologias de Virtualização e Centros de Dados

## Mestrado em Engenharia Informática

Mário M. Freire
Departamento de Informática

Ano Letivo 2023/2024

# Credits

- These slides are based on the following documents:
- Tim Hildred, Red Hat Blog, The History of Containers, August 28, 2015, https://www.redhat.com/en/blog/history-containers
- Red Hat, What's a Linux container? https://www.redhat.com/en/topics/containers/whats-a-linux-container
- Distributed and Cloud Computing: From Parallel Processing to the Internet of Things, Kai Hwang, Jack Dongarra, Geoffrey C. Fox (Authors), Morgan Kaufmann, 1st edition, 2011, ch. 3.
- A. Leung, A. Spyker, and T. Bozarth, Titus: Introducing containers to the netflix cloud, Communications of the ACM, vol. 61, no. 2, pp. 38–45, February 2018.
- Docker, What is a Container?, https://www.docker.com/resources/what-container
- Docker, Docker overview, https://docs.docker.com/get-started/overview/
- Kubernetes, What is Kubernetes?, March 31, 2020, https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
- Kubernetes, Kubernetes Components, November 09, 2019, https://kubernetes.io/docs/concepts/overview/components/

# Agenda

- Roots of Container Technologies
- What is a Linux Container?
- Why OS-Level Virtualization?
- Container Popularity
- Top 10 Best Container Software In 2020
- Overview of Container Ecosystems
- Docker
- Kubernetes
- Kubernetes Versus Docker Swarm

# Roots of Container Technologies

- In 2000, jails, an early implementation of container technology, was added to FreeBSD.
- FreeBSD jails allows the partitioning of a FreeBSD system into multiple subsystems, or jails.
- Jails were developed as safe environments that a system administrator could share with multiple users inside or outside of an organization.
- In 2001, Jacques Gélinas created the VServer project, which allowed to running several general purpose Linux server on a single box with a high degree of Independence and security.

Source: Tim Hildred, Red Hat Blog, The History of Containers, August 28, 2015, https://www.redhat.com/en/blog/history-containers

# Roots of Container Technologies

- The Linux-VServer solution was the first effort on Linux to separate the user-space environment into distinct units (Virtual Private Servers) in such a way that each VPS looks and feels like a real server to the processes contained within.
- In 2006, Paul Menage adapted the cpusets mechanism already in the mainline kernel, requiring minimally intrusive changes with little impact on performance, code quality, complexity, and future compatibility.
- The result was generic process containers, which were later renamed control groups, or cgroups.

Source: Tim Hildred, Red Hat Blog, The History of Containers, August 28, 2015, https://www.redhat.com/en/blog/history-containers

# Roots of Container Technologies

- In 2008, Eric W. Biederman created user namespaces.
- The implementation of user namespaces allows a process to have its own set of users and in particular to allows a process root privileges inside a container, but not outside.
- Around 2008, engineers from IBM created the Linux Containers project (LXC), which layered some userspace tooling on top of cgroups and namespaces.
- The LXC project provided an improved user experience around containers.

Source: Tim Hildred, Red Hat Blog, The History of Containers, August 28, 2015, https://www.redhat.com/en/blog/history-containers

# Roots of Container Technologies

- In 2008, Docker came onto the scene (by way of dotCloud) with their eponymous container technology.
- The docker technology added a lot of new concepts and tools: 1) a simple command line interface for running and building new layered images, 2) a server daemon, 3) a library of pre-built container images, and 4) the concept of a registry server.
- Combined, these tools/technologies allowed users to quickly build new layered containers and easily share them with others.

Source: Red Hat, What's a Linux container? https://www.redhat.com/en/topics/containers/whats-a-linux-container

# What is a Linux Container?

- A Linux® container is a set of one or more processes that are isolated from the rest of the system.
- All the files necessary to run them are provided from a distinct image, meaning that Linux containers are portable and consistent as they move from development, to testing, and finally to production.
- This makes them much quicker than development pipelines that rely on replicating traditional testing environments.

Source: Red Hat, What's a Linux container?, https://www.redhat.com/en/topics/containers/whats-a-linux-container

# Why OS-Level Virtualization?

- It is slow to initialize a hardware-level VM because each VM creates its own image from scratch.
- Thousands of VMs may need to be initialized simultaneously.
- Storing the VM images also becomes an issue since there is considerable repeated content among VM images.
- OS-level virtualization provides a feasible solution for these hardware-level virtualization issues.
- OS-level virtualization inserts a virtualization layer inside an OS to partition a machine's physical resources.
- It enables multiple isolated VMs within a single operating system kernel.

Source: Distributed and Cloud Computing: From Parallel Processing to the Internet of Things, Kai Hwang, Jack Dongarra, Geoffrey C. Fox (Authors), Morgan Kaufmann, 1st edition, 2011, ch. 3.

# Why OS-Level Virtualization?

- This kind of VM is called a **virtual execution environment** (VE), **Virtual Private System** (VPS), or simply **container**.
- From the user's point of view, **containers look like real servers**, since a container has its own set of processes, file system, user accounts, network interfaces with IP addresses, routing tables, firewall rules, and other personal settings.
- **Containers share the same operating system kernel**. Therefore, OS-level virtualization is also called single-OS image virtualization.

Source: Distributed and Cloud Computing: From Parallel Processing to the Internet of Things, Kai Hwang, Jack Dongarra, Geoffrey C. Fox (Authors), Morgan Kaufmann, 1st edition, 2011, ch. 3.

# Why OS-Level Virtualization?

- Advantages of OS Extension for Virtualization
  - VMs at the OS level have **minimal startup/shutdown** costs, low resource requirements, and high scalability.
  - It is possible for a VM and its host environment to **synchronize state changes** when necessary.
- Disadvantage of OS Extension for Virtualization
  - Containers share the same OS, which **restrict application flexibility** of different containers on the same physical machine.
  - Security and privacy issues.



Source: Distributed and Cloud Computing: From Parallel Processing to the Internet of Things, Kai Hwang, Jack Dongarra, Geoffrey C. Fox (Authors), Morgan Kaufmann, 1st edition, 2011, ch. 3.

# Container Popularity

- **Container use is up**, and **Kubernetes use is skyrocketing**.
- The use of Docker® containers continues to grow, with adoption increasing to 57 percent from 49 percent in 2018.
- **Kubernetes**, a container orchestration tool that leverages Docker, achieved faster growth, increasing from 27 percent to 48 percent adoption.
- Enterprise adoption is even higher, with 66 percent using Docker and 60 percent leveraging Kubernetes.
- The AWS container service (ECS/EKS) has 44 percent adoption in 2019 (flat from 2018), while Azure Container Service adoption reaches 28 percent (up from 20 percent in 2018) and Google Container Engine grows slightly to reach adoption of 15 percent.



Source: 2019 RightScale® State of the Cloud Report from Flexera™

# Container Popularity

- VMware vSphere leads the adoption for building Private clouds. However, VMware is investing in **VMware Kubernetes**.
- As a new feature in Windows Server 2016 Microsoft introduced **Windows containers** (to achieve isolation at the app level rather than the OS level).
- Major cloud service providers, after initially building an infrastructure with native hardware level virtualization are now offering container and kubernetes as a service.

# Container Popularity

- **The Case of Netflix**
- In 2008, Netflix went all-in on cloud migration and began moving all of its internally hosted infrastructure to AWS (Amazon Web Services).
- In 2017, almost all of Netflix runs on VMs (virtual machines) in AWS. A customer's catalog browsing experience, content recommendation calculations, and payments are all served from AWS.
- Over the years Netflix has helped craft many cloud-native patterns, such as loosely coupled microservices and immutable infrastructure, that have become industry best practices.



Source: A. Leung, A. Spyker, and T. Bozarth, Titus: Introducing containers to the netflix cloud, Communications of the ACM, vol. 61, no. 2, pp. 38–45, February 2018

# Container Popularity

- **The Case of Netflix (2)**
- The all-in migration to the cloud has been hugely successful for Netflix.
- Despite already having a successful cloud-native architecture, Netflix is investing in container technology.
- Container technology enables Netflix to follow many of the same patterns already employed for VMs but in a simpler, more flexible, and efficient way.

Source: A. Leung, A. Spyker, and T. Bozarth, Titus: Introducing containers to the netflix cloud, Communications of the ACM, vol. 61, no. 2, pp. 38–45, February 2018

# Top 10 Best Container Software In 2020

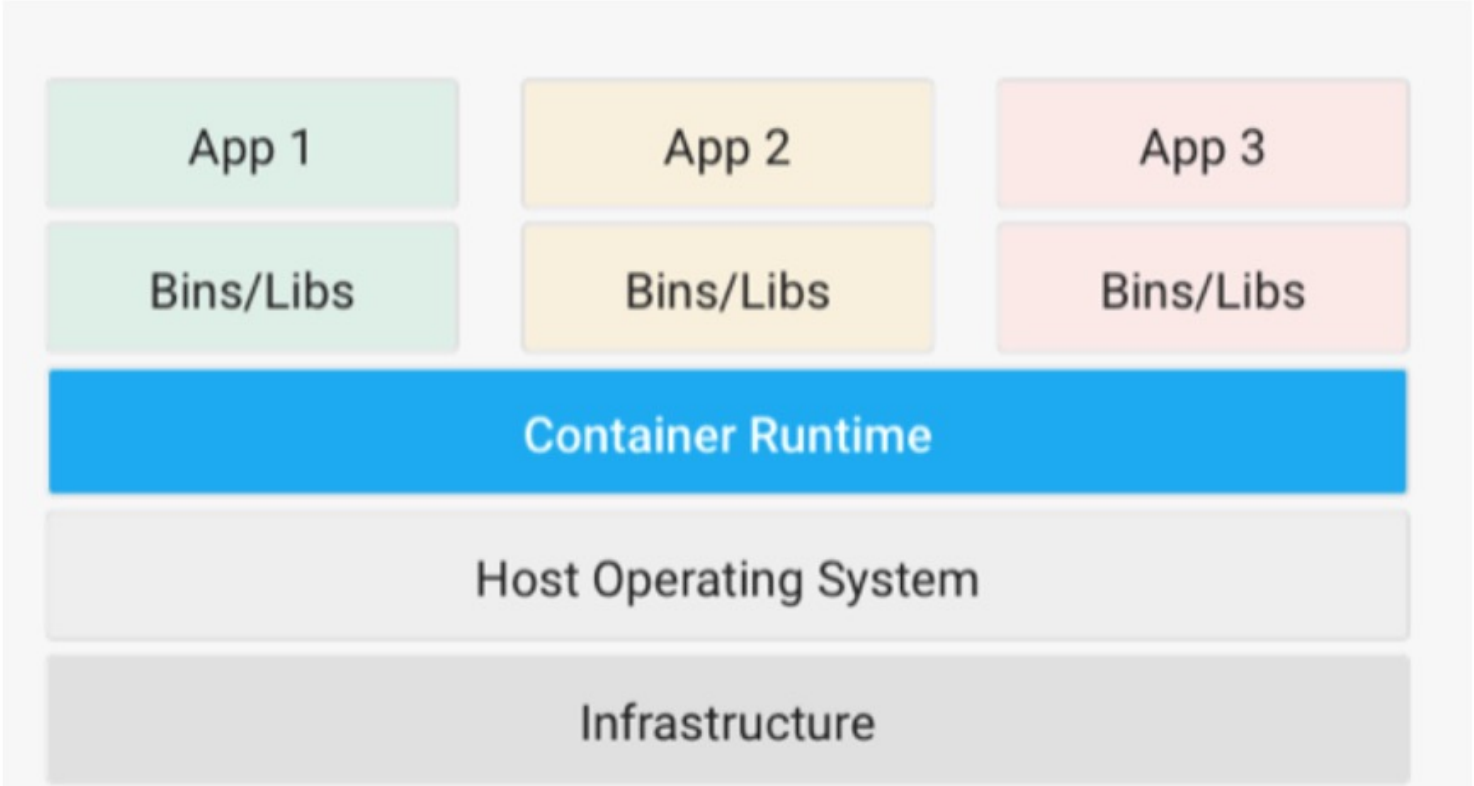
- #1) Docker
- #2) AWS Fargate
- #3) Google Kubernetes Engine
- #4) Amazon ECS
- #5) LXC (Linux Containers)
- #6) Container Linux by CoreOS
- #7) Microsoft Azure
- #8) Google Cloud Platform
- #9) Portainer
- #10) Apache Mesos



Source: Top 10 Best Container Software In 2020, Softwaretestinghelp, March 10, 2020, https://www.softwaretestinghelp.com/container-software/

# Overview of Container Ecosystems

## Typical container stack



# Docker

- Docker Inc. was founded by Solomon Hykes and Sebastien Pahl during the Y Combinator Summer 2010 startup incubator group and launched in 2011.
- Docker debuted to the public in Santa Clara at PyCon in 2013 and it was released as open-source in March 2013.
- Docker uses OS-level virtualization to deliver software in packages called containers.
- Docker provides a way to run applications securely isolated in a container, packaged with all its dependencies and libraries.
- All containers are run by a single operating system kernel.



Source: Docker, https://docs.docker.com/get-started/overview/
Source: Docker, What is a Container?, https://www.docker.com/resources/what-container

# Docker

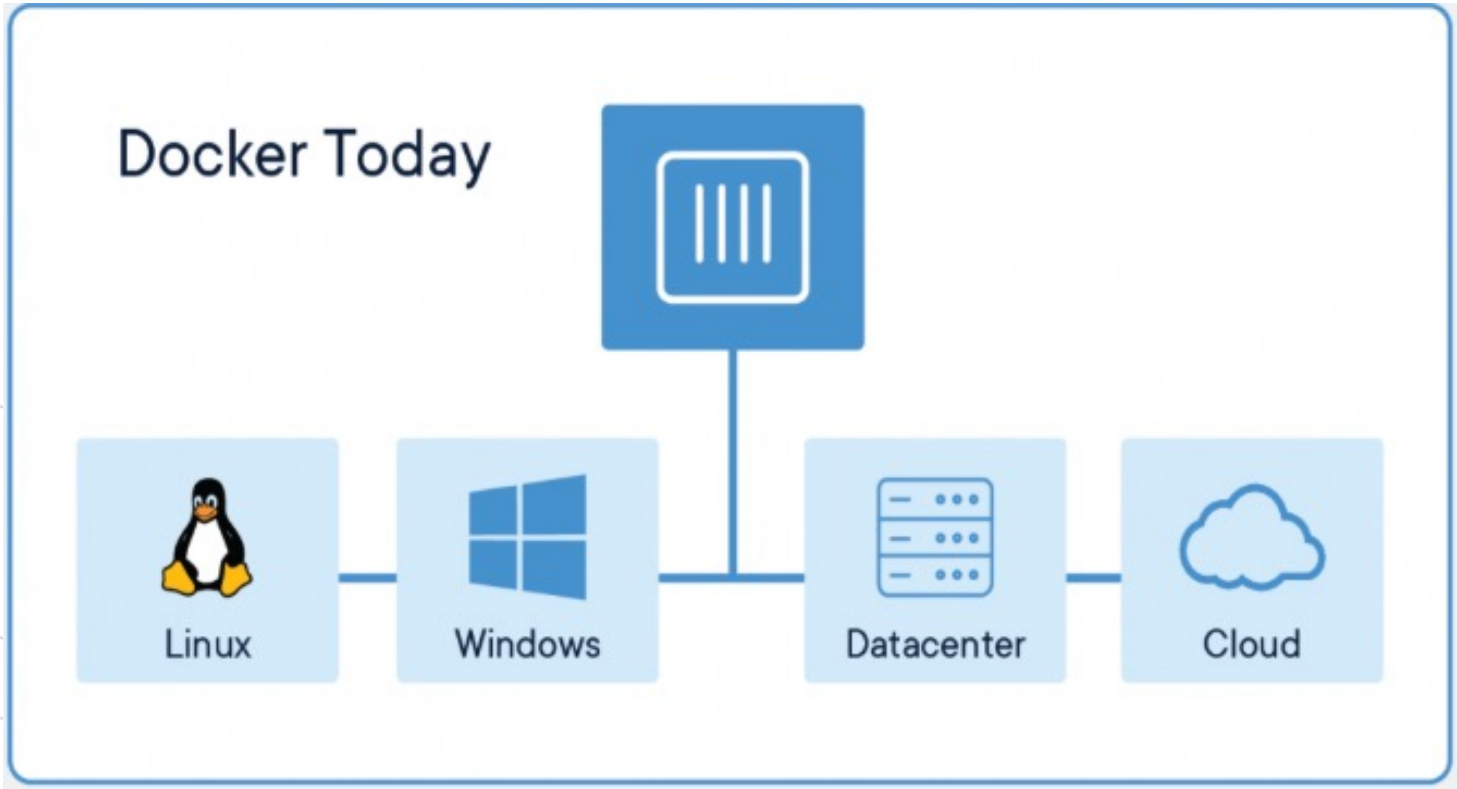
- Docker containers that run on Docker Engine:
  - Standard: Docker created the industry standard for containers, so they could be portable anywhere
  - Lightweight: Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs
  - Secure: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry.



Source: Docker, What is a Container?, https://www.docker.com/resources/what-container

# Docker

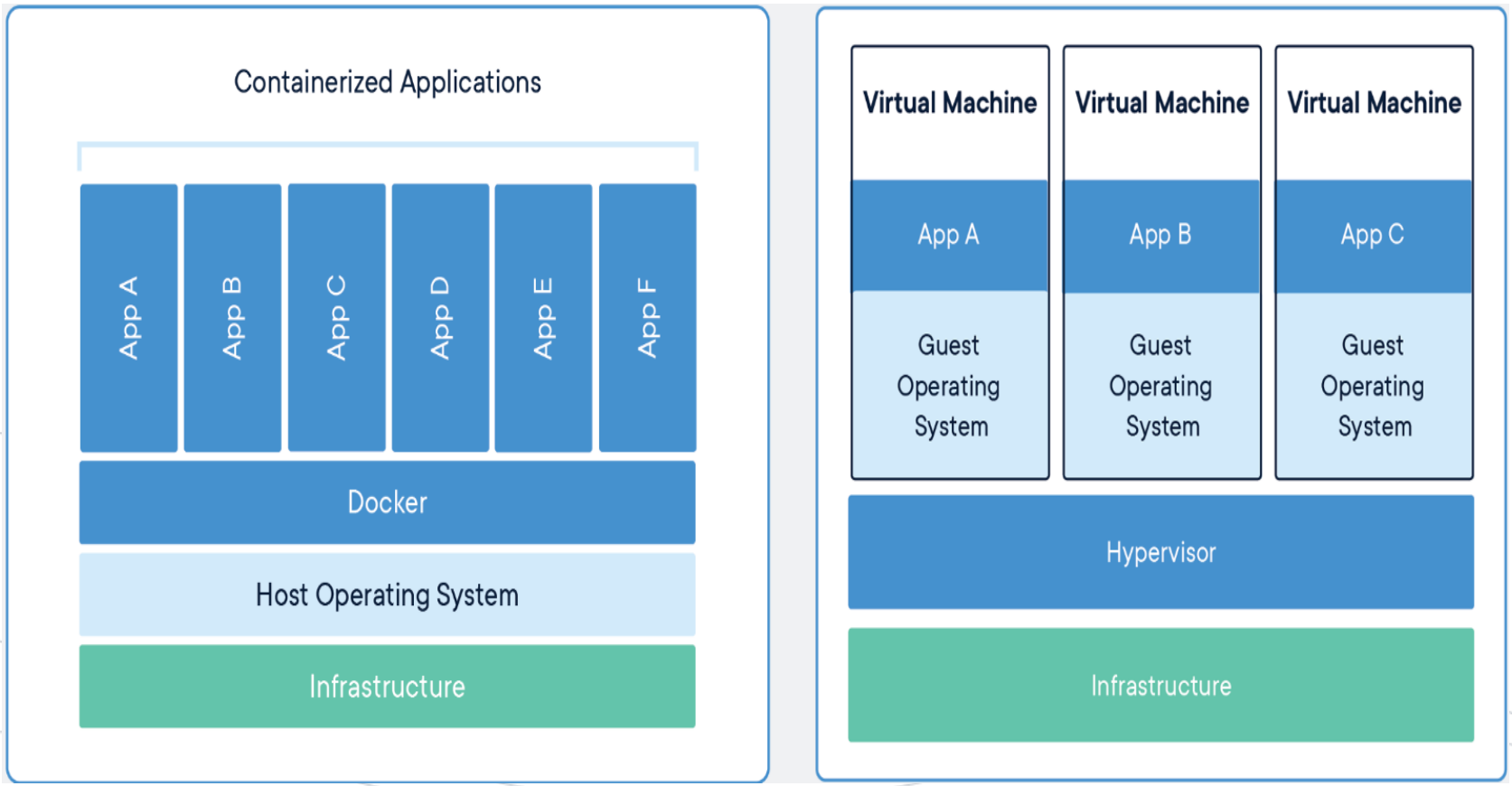
Docker Containers are everywhere: Linux, Windows, Mac, Data center, Cloud, Serverless




 Source: Docker, What is a Container?, https://www.docker.com/resources/what-container

# Docker

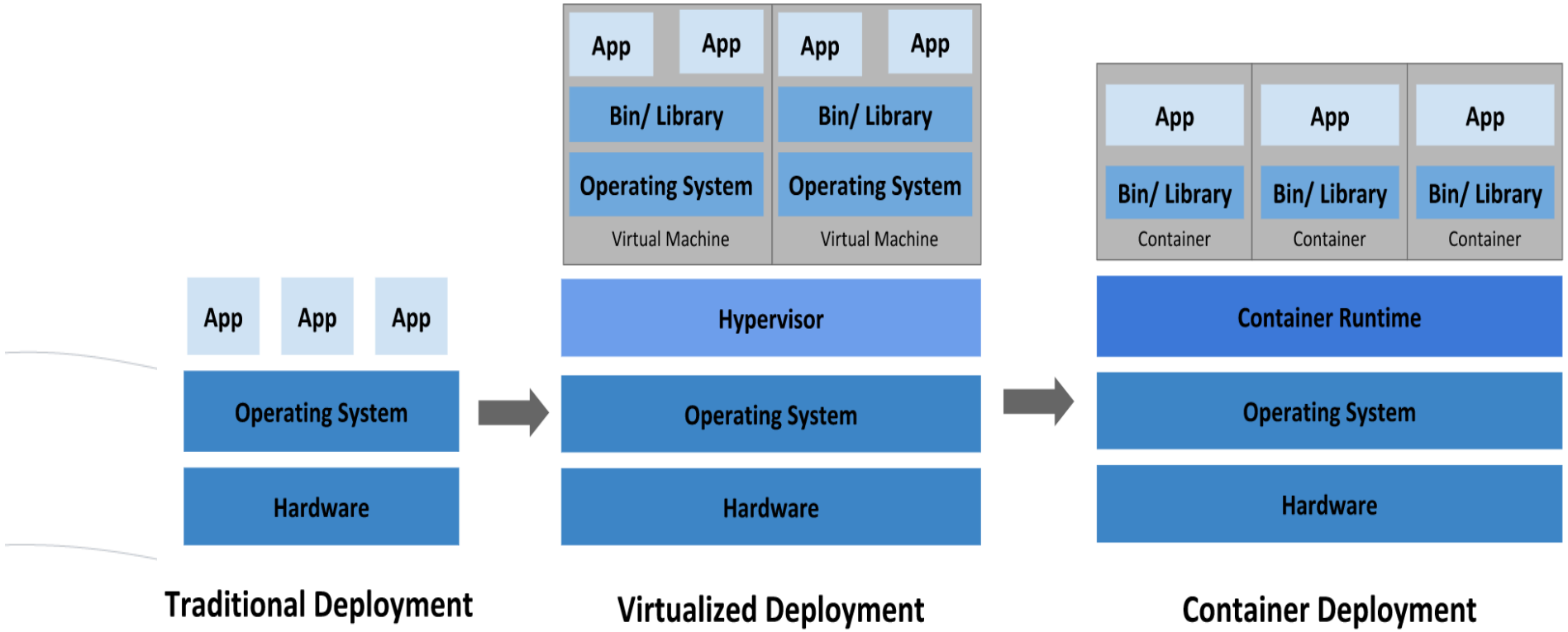
## Comparing Containers and Virtual Machines




 Source: Docker, What is a Container?, https://www.docker.com/resources/what-container

# Docker

## Comparing Containers and Virtual Machines




Source: Kubernetes, What is Kubernetes?, March 31, 2020, https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

# Docker

- **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 one 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.

In: Docker, Docker overview, https://docs.docker.com/get-started/overview/

# Docker

- **The Docker daemon**
  - The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
  - A daemon can also communicate with other daemons to manage Docker services.
- **The Docker client**
  - The Docker client (docker) is the primary way that many Docker users interact with Docker.
  - When one uses commands such as docker run, the client sends these commands to dockerd, which carries them out.
  - The docker command uses the Docker API.
  - The Docker client can communicate with more than one daemon.



In: Docker, Docker overview, https://docs.docker.com/get-started/overview/

# Docker

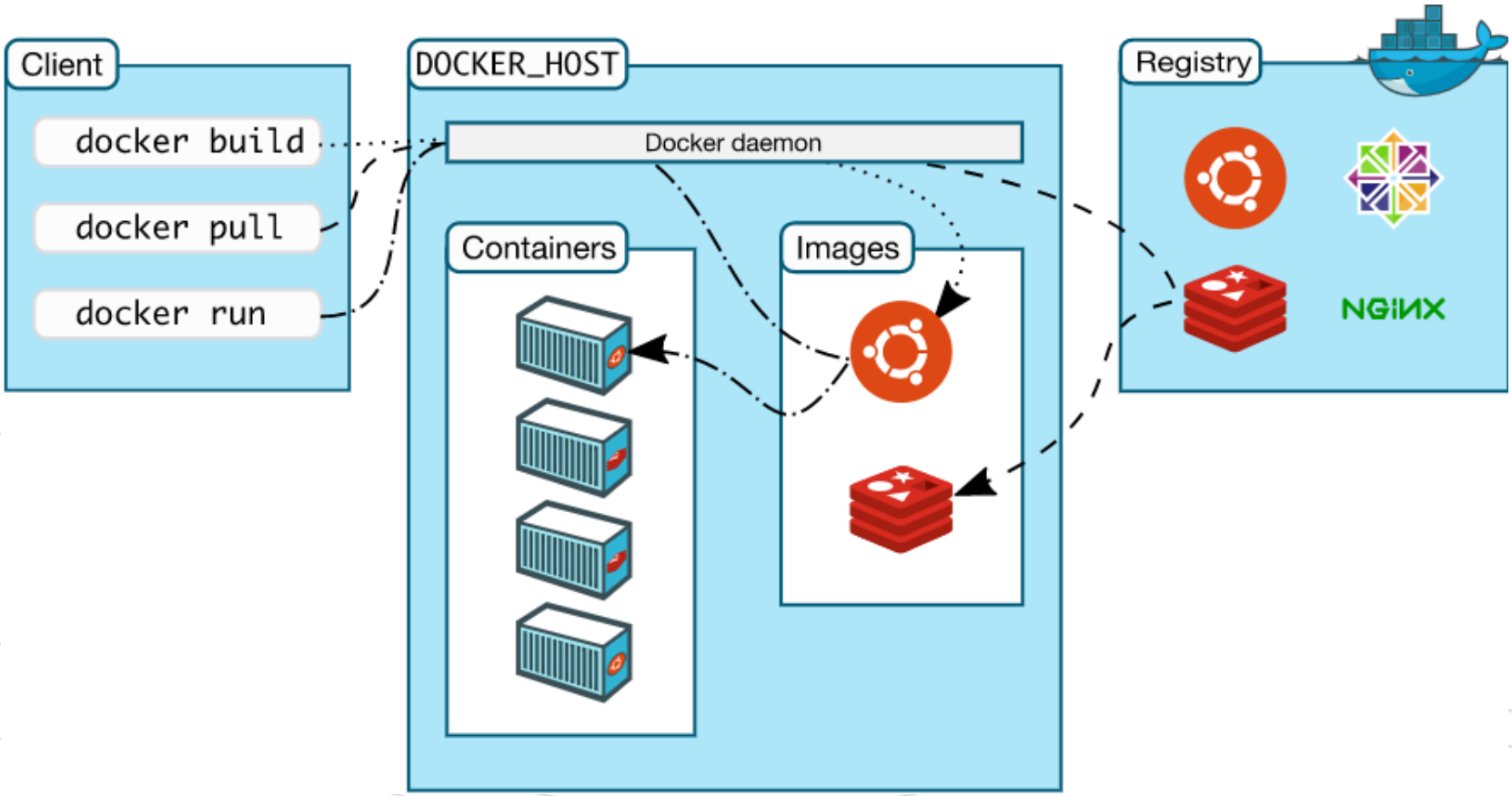
- **Docker registries**
  - A Docker registry stores Docker images.
  - Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
  - One can even run his/her own private registry.
  - If one use Docker Datacenter (DDC), it includes Docker Trusted Registry (DTR).
  - When you use the docker pull or docker run commands, the required images are pulled from your configured registry.
  - When you use the docker push command, your image is pushed to your configured registry.



In: Docker, Docker overview, https://docs.docker.com/get-started/overview/

# Docker

## Docker architecture




 In: Docker, Docker overview, https://docs.docker.com/get-started/overview/

# Docker

- **Swarm mode**
- Current versions of Docker include **swarm mode** for natively managing a cluster of Docker Engines called a swarm.
- Use the Docker CLI to create a swarm, deploy application services to a swarm, and manage swarm behavior.
- Cluster management integrated with Docker Engine: Use the Docker Engine CLI to create a swarm of Docker Engines where you can deploy application services.
- One don’t need additional orchestration software to create or manage a swarm.



In: Docker, https://docs.docker.com/engine/swarm /

# Kubernetes

- **Kubernetes** is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
- It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.
- The name Kubernetes originates from Greek, meaning helmsman or pilot.
- Google open-sourced the Kubernetes project in 2014.
- Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.



Source: Kubernetes, What is Kubernetes?, March 31, 2020, https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

# Kubernetes

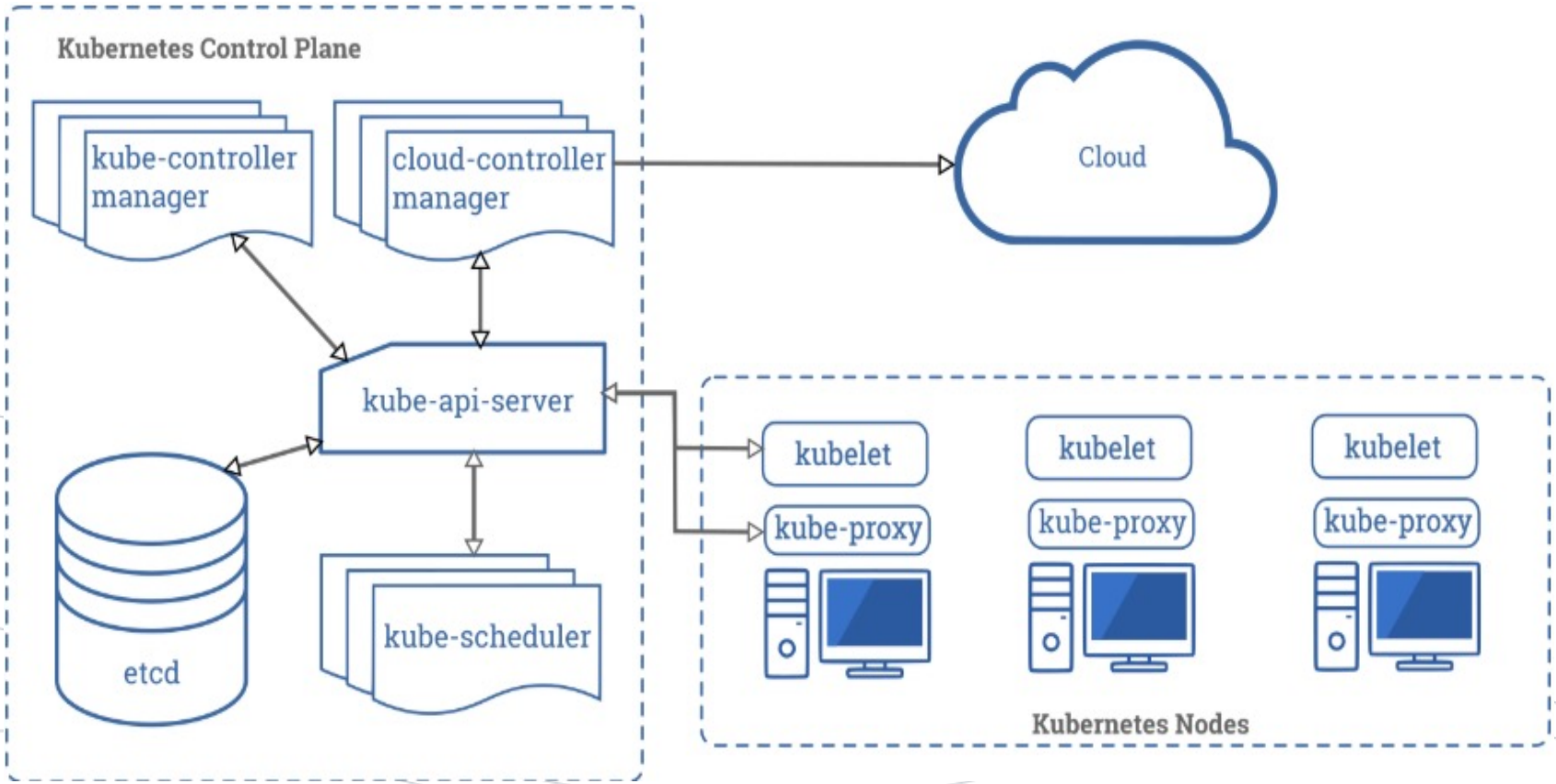
- A **Kubernetes cluster** consists of a set of worker machines, called **nodes**, that run containerized applications. Every cluster has at least one worker node.
- The worker node(s) host the Pods that are the components of the application workload.
- A Pod represents a set of running containers in a cluster.
- The control plane manages the worker nodes and the Pods in the cluster.
- In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.



Source: Kubernetes, Kubernetes Components, November 09, 2019, https://kubernetes.io/docs/concepts/overview/components/

# Kubernetes

## Kubernetes architecture




In: Kubernetes, Kubernetes Components, November 09, 2019, https://kubernetes.io/docs/concepts/overview/components/

# Kubernetes Versus Docker Swarm

## Kubernetes versus Docker Swarm

### Docker Swarm

#### Pros

- Easy and fast setup
- Works with other existing Docker tools
- Lightweight installation
- Open source

#### Cons

- Limited in functionality by what is available in the Docker API
- Limited fault tolerance

### Kubernetes

#### Pros

- Open source and modular
- Runs well on any operating systems
- Easy service organisation with pods
- Backed by years of expert experience

#### Cons

- Laborious to install and configure
- Incompatible with existing Docker CLI and Compose tools



Docker Swarm vs. Kubernetes: Comparison of the Two Giants in Container Orchestration, https://upcloud.com/community/stories/docker-swarm-vs-kubernetes-comparison-of-the-two-giants-in-container-orchestration/