Lecture
Docker Containers

& Kubernetes

Dr. Faisal Cheema

Virtualization

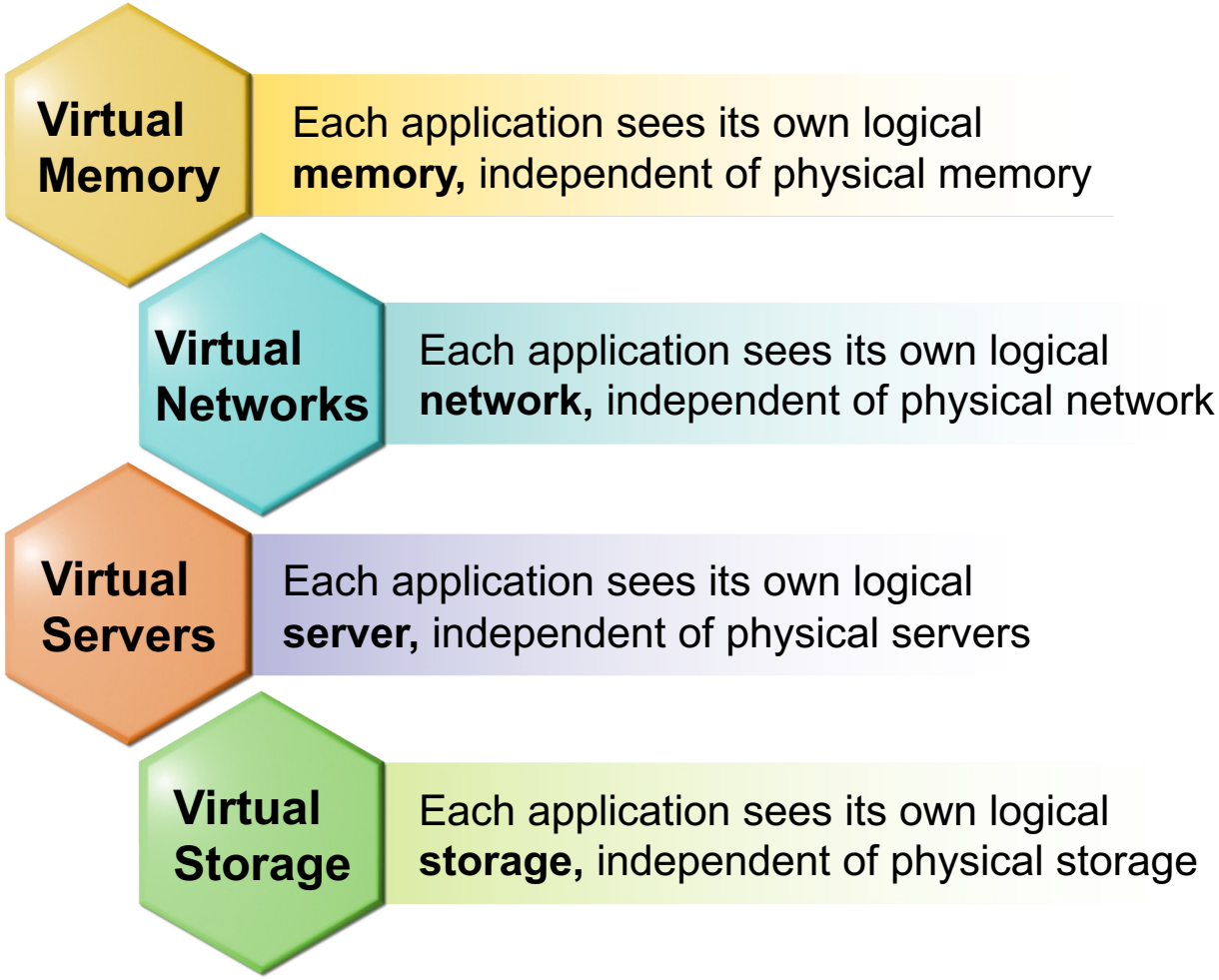
- In a general sense, virtualization, is the creation of a virtual, rather than an actual, version of something.
  - For example, you can take a virtual tour of the White House by going to http://www.whitehouse.gov/about/inside-white-house/interactive-tour
  - In other words, you can take a tour of the White House without actually going to the White House and taking the tour.

Virtualization

- From a computing perspective, you might have already done some virtualization if you've ever partitioned a hard disk drive into more than one "virtual" drive.

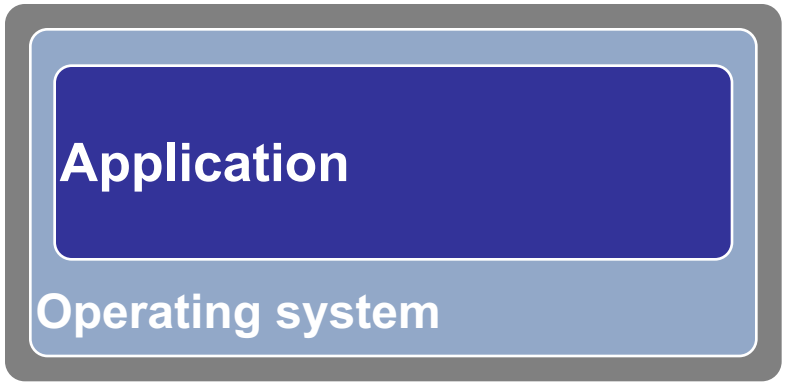
- What other virtualized resources you have already used?

Virtualization Comes in Many Forms



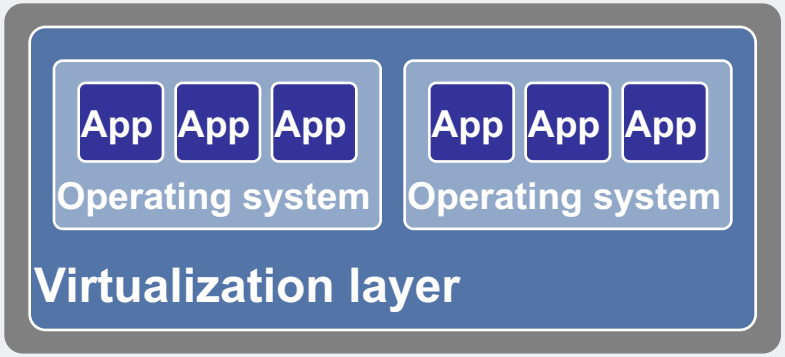
Server Virtualization

Before Server Virtualization:



- Single operating system image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources

After Server Virtualization:

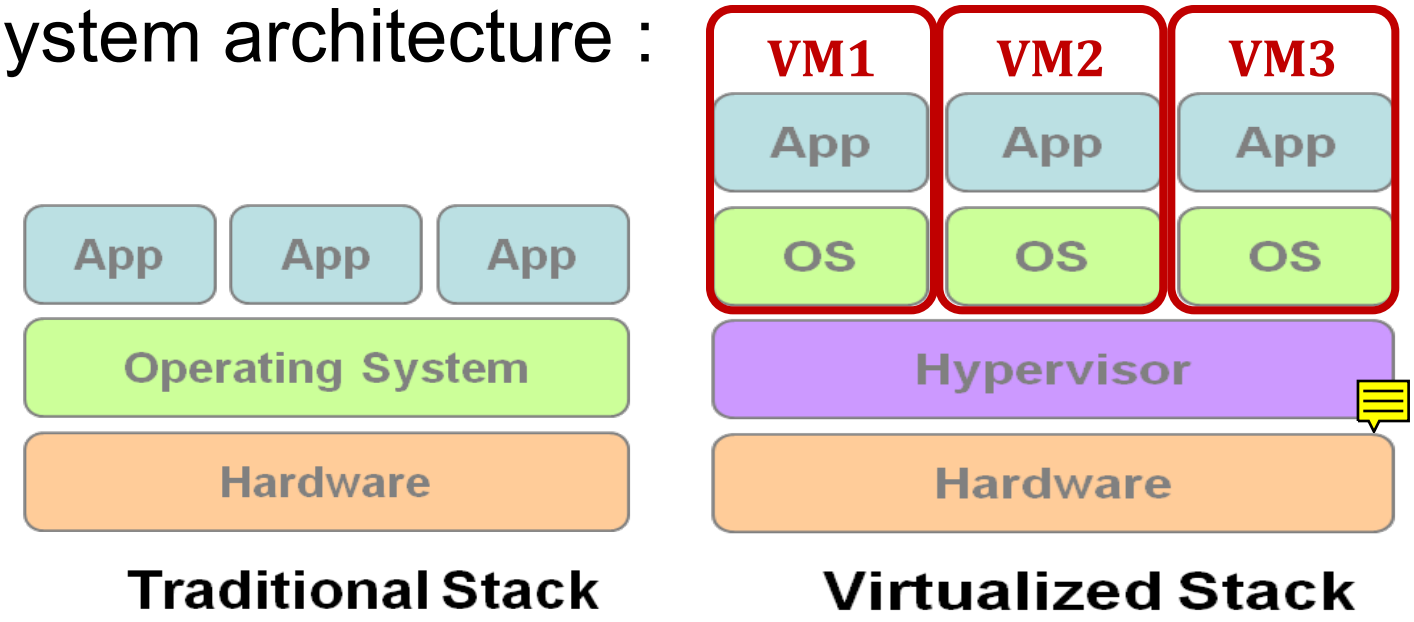


- Virtual Machines (VMs) break dependencies between operating system and hardware
- Manage operating system and application as single unit by encapsulating them into VMs
- Strong fault and security isolation

Virtual Machine Monitor

- What's Virtual Machine Monitor (VMM) ?
  - **VMM** or **Hypervisor** is the software layer providing the virtualization.

- System architecture :



Virtual machines limitations

- VMs take up a lot of system resources.
- Each VM runs not just a full copy of an operating system, but a virtual copy of all the hardware that the operating system needs to run.

- This quickly adds up to a lot of RAM and CPU cycles.

Number of VMs per Host?

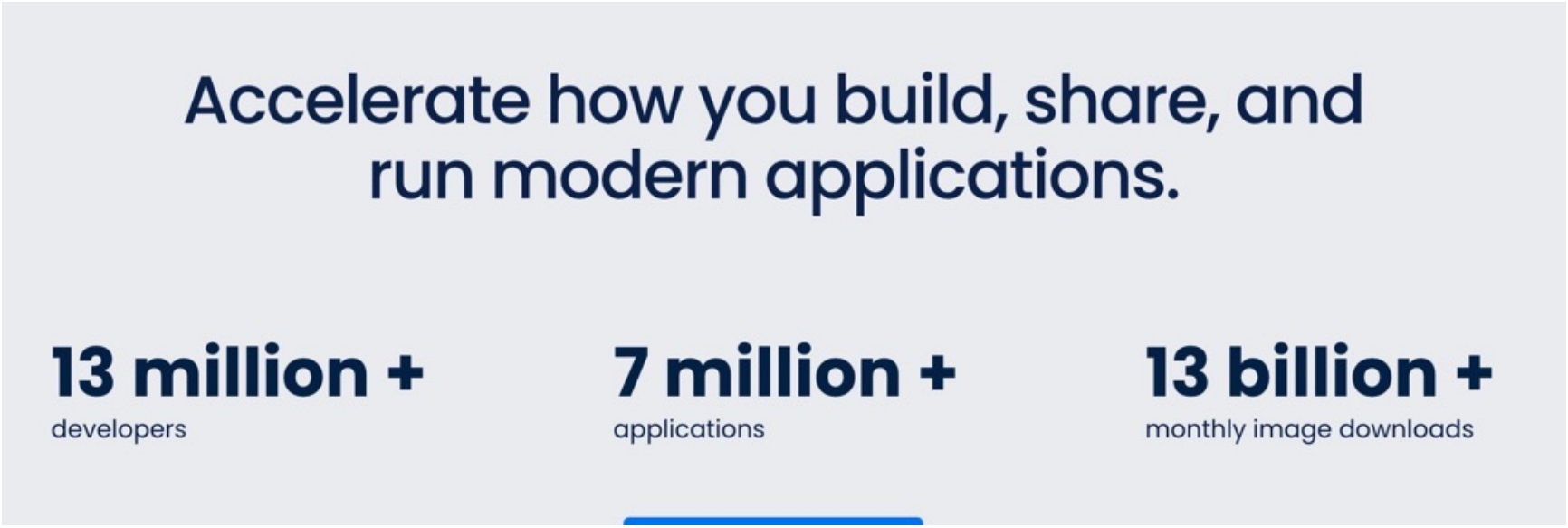
- How much is too much?
- 500 VMs on one server host is possible but sometimes less is more. Risk, utilization rates and memory factor into the decision.

- Virtualization doesn't just consolidate as many servers as possible -- it has to actually do something.

Do we have some alternative?

Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in

Docker on the rise

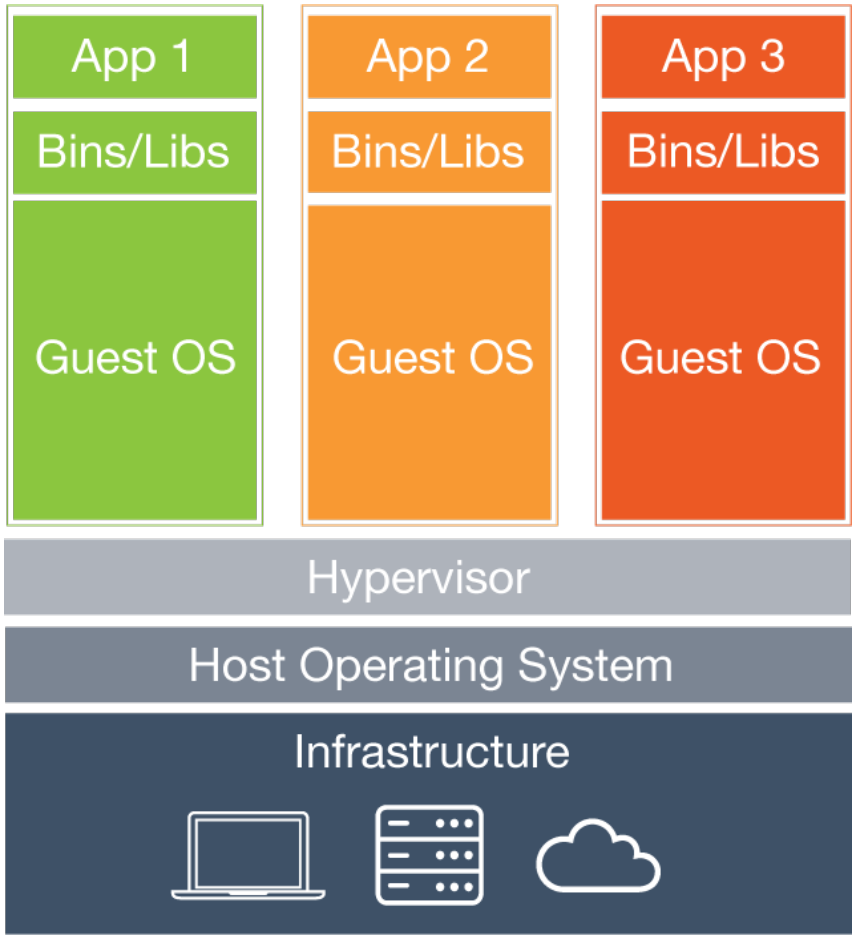


Docker containers

- Containers running on a single machine all share the same operating system kernel so they start instantly and make more efficient use of RAM.

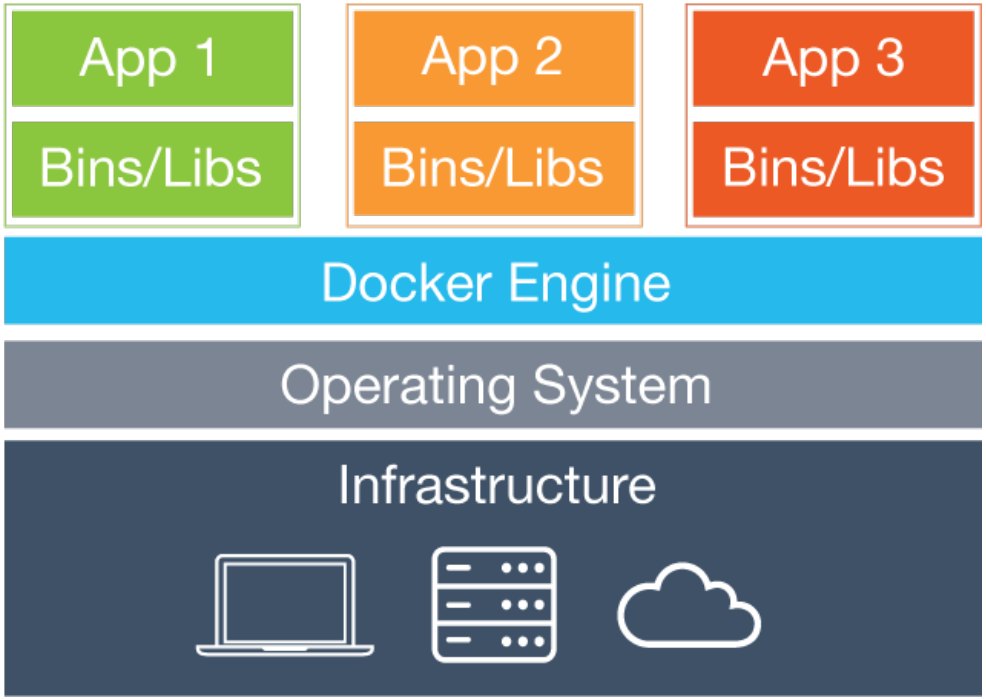
- You can thus stuff more containers on a server host than VMs.

How is this different?



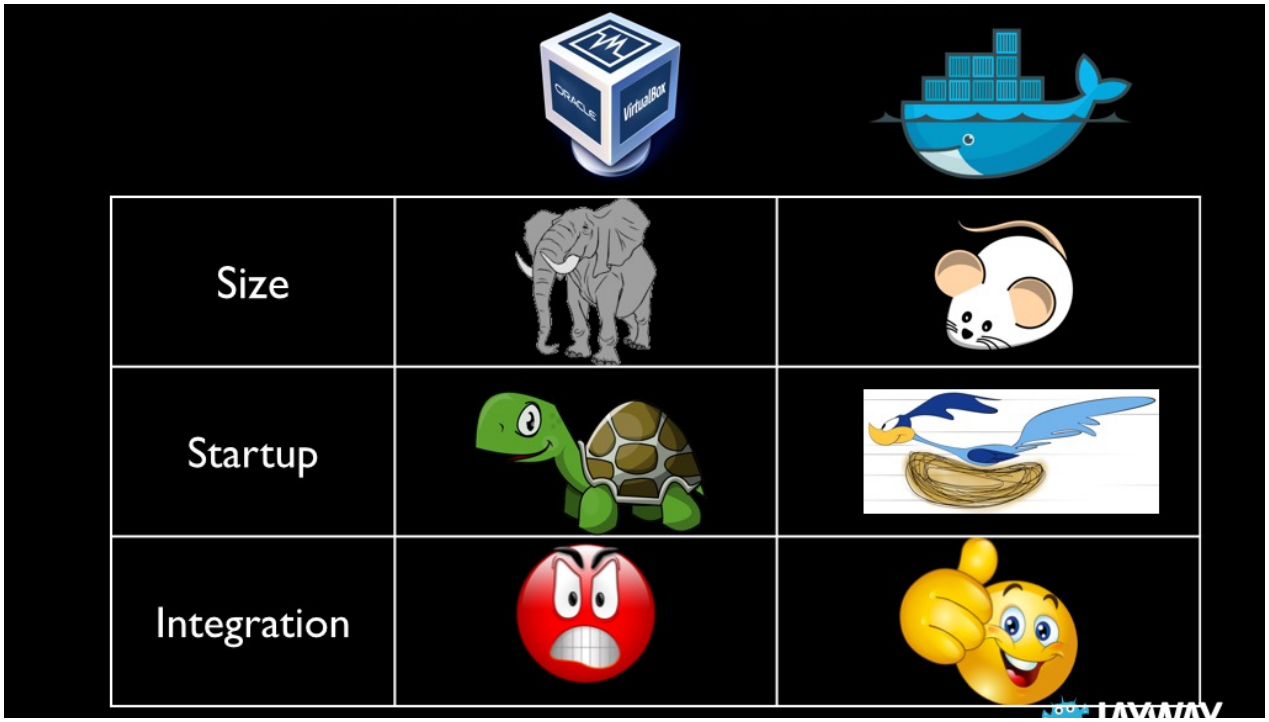
Each virtual machine includes the application, the necessary binaries and libraries and an entire guest operating system - all of which may be tens of GBs in size.

How is this different?

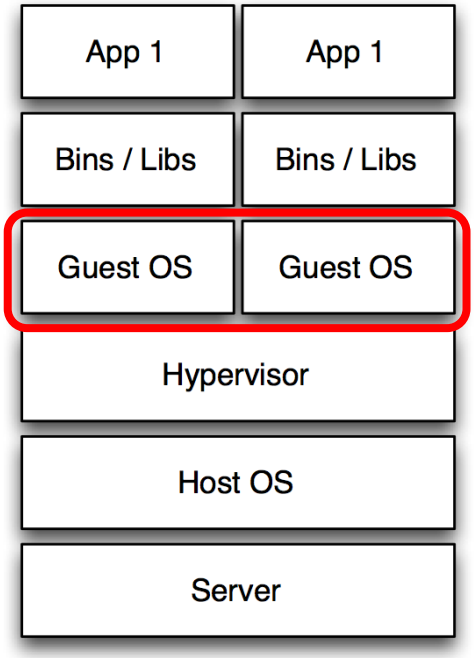


Containers include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in userspace on the host operating system.

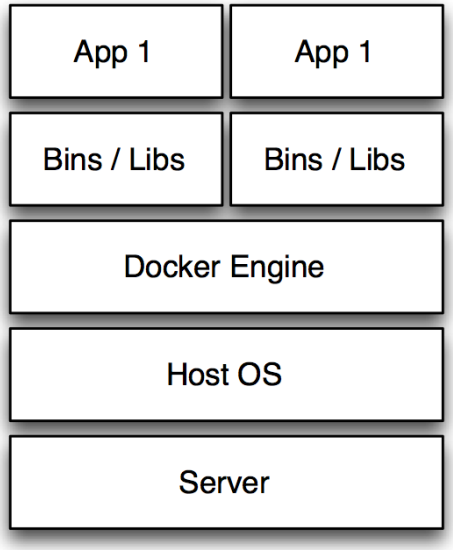
VM vs. Docker



VM vs. Docker (Containers)



Virtual Machines



Docker

Docker Engine
Docker engine is the layer on which Docker runs.
It's a lightweight runtime and tooling that manages containers, builds, and more.

Installation

- The getting started guide on Docker has detailed instructions for setting up Docker on Mac, Linux and Windows

- On Mac/Windows - Two variants, Native and Toolbox

WHY DO WE NEED CONTAINER ORCHESTRATION?

Containers are Good…

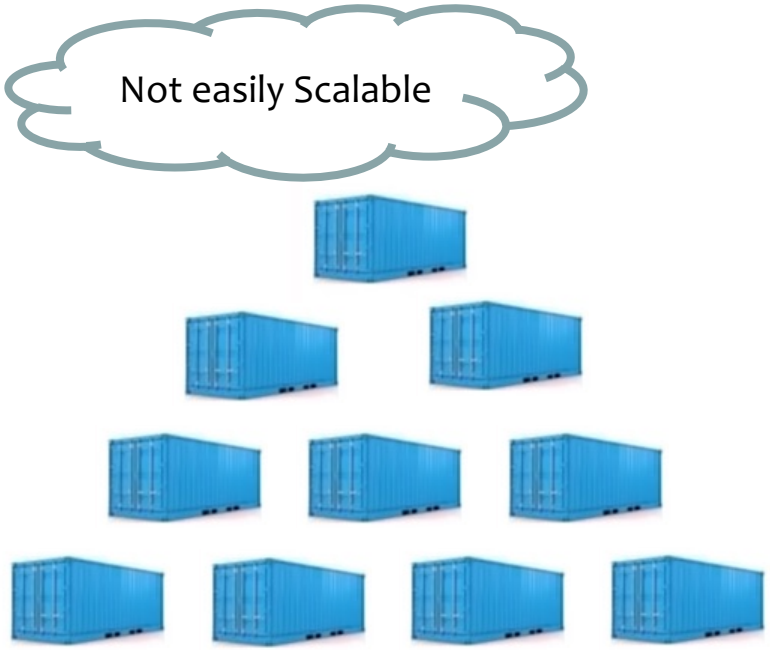
- Both Linux Containers and Docker Containers
  - Isolate the application from the host

Containers Problems!

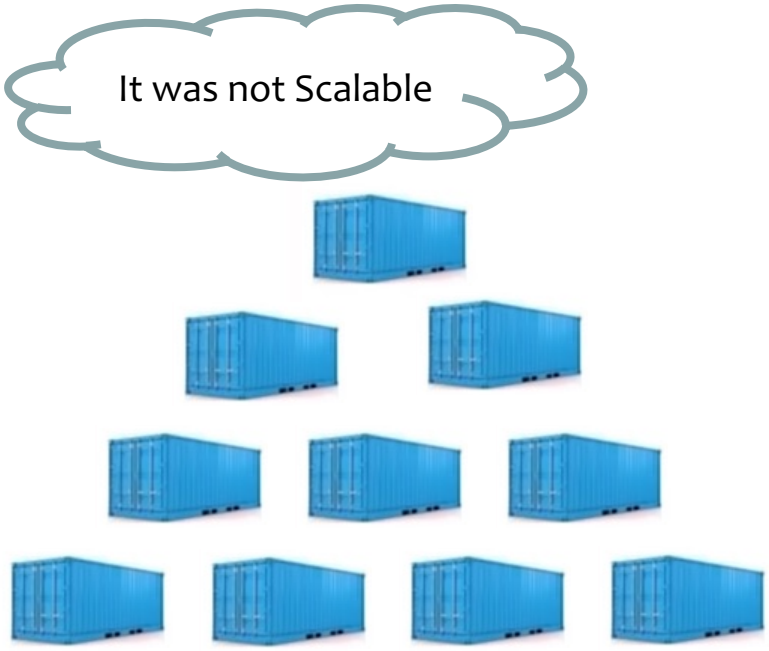
- Both Linux Containers and Docker Containers
  - Isolate the application from the host

Containers Problems!

- Both Linux Containers and Docker Containers
  - Isolate the application from the host

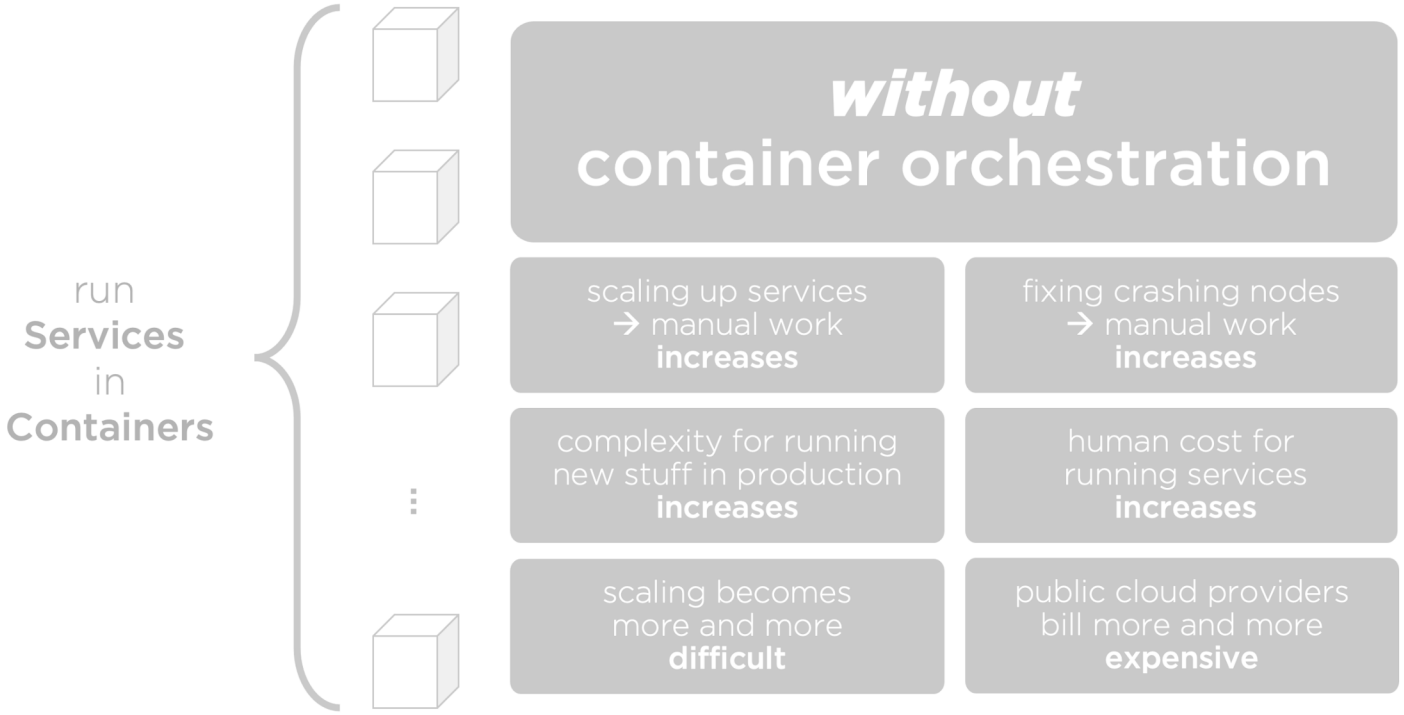


Problems with Scaling up Containers



1. Containers could communicate with each other
2. Containers had to be deployed appropriately
3. Containers had to be managed carefully
4. Auto scaling was not possible
5. Distributing traffic was still challenging

Containers Without Orchestration



Why do we need container orchestration?

- Because containers are lightweight and ephemeral by nature, running them in production can quickly become a massive effort.
- Particularly when paired with microservices—which typically each run in their own containers—a containerized application might translate into operating hundreds or thousands of containers, especially when building and operating any large-scale system.
- This can introduce significant complexity if managed manually.
- Container orchestration is what makes that operational complexity manageable for development and operations—or DevOps—because it provides a declarative way of automating much of the work.

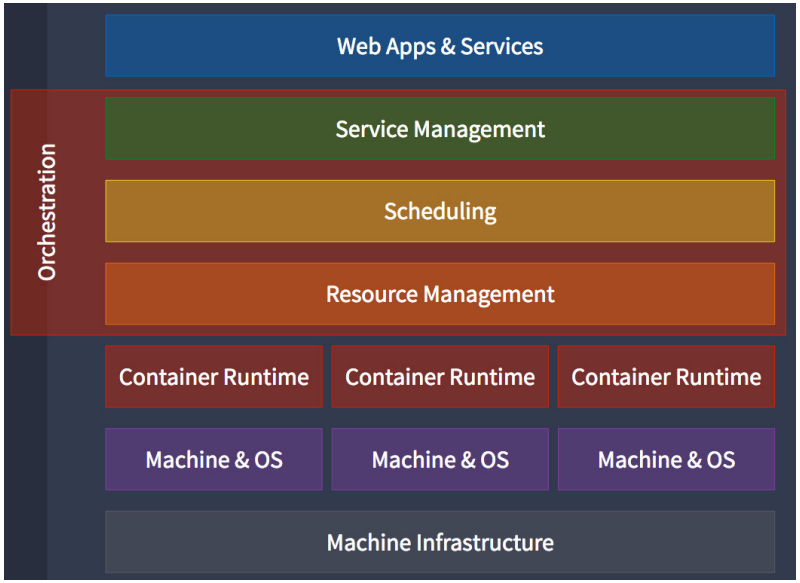
CONTAINER ORCHESTRATION

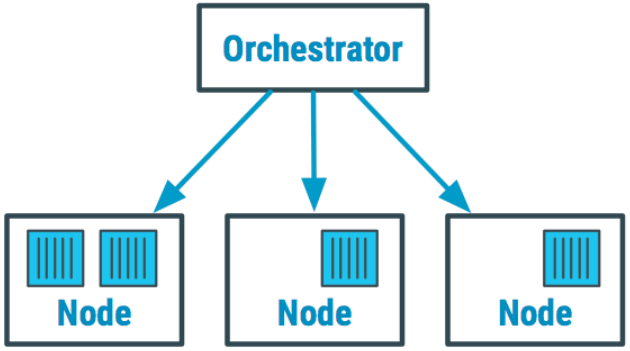
What is container orchestration?

- Container orchestration is the automation of much of the operational effort required to run containerized workloads and services.
- This includes a wide range of things software teams need to manage a container's lifecycle, including:
  - provisioning,
  - deployment,
  - scaling (up and down),

Container Orchestration

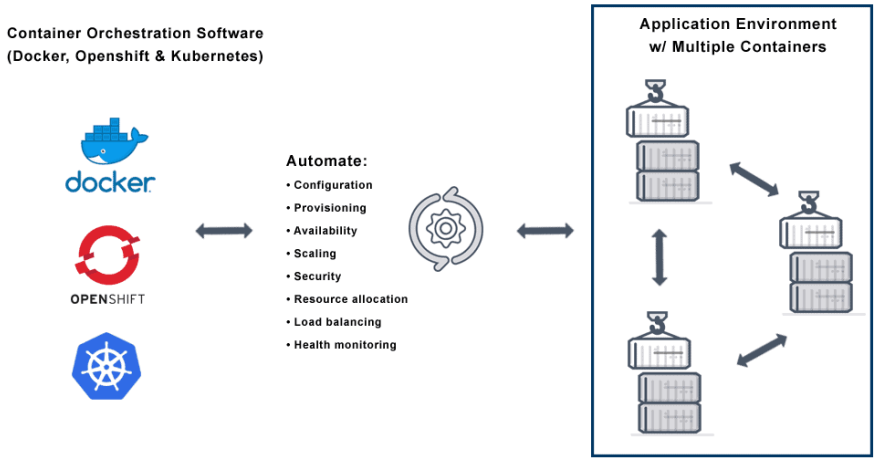
- Container orchestration automates and simplifies provisioning, and deployment and management of containerized applications.





Container Orchestration

- Container orchestration is the automatic process of managing or scheduling the work of individual containers for applications based on microservices within multiple clusters.
- The widely deployed container orchestration platforms are based on open-source versions like Kubernetes, Docker Swarm or the commercial version from Red Hat OpenShift.



Container Orchestration

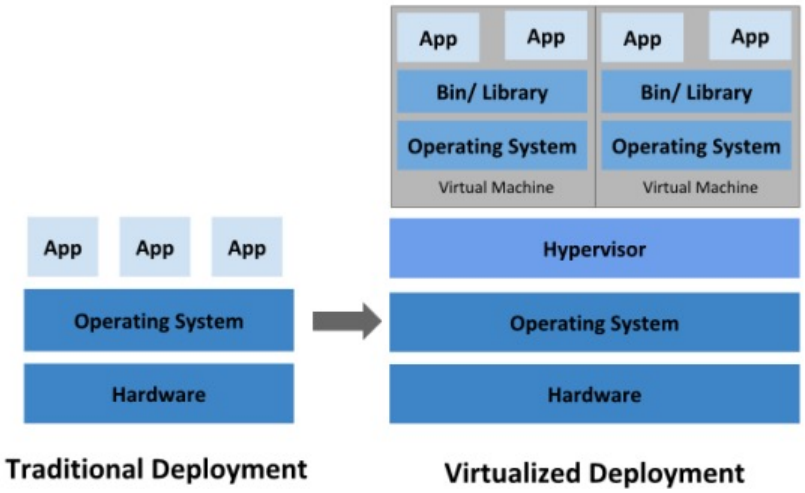
- Fault-tolerance
- On-demand scalability
- Optimal resource usage
- Auto-discovery to automatically discover and communicate with each other
- Accessibility from the outside world
- Seamless updates/rollbacks without any downtime.

Why Do We Need Container Orchestration?

- Container orchestration is used to automate the following tasks at scale:
  - Configuring and scheduling of containers
  - Provisioning and deployments of containers
  - Availability of containers
  - The configuration of applications in terms of the containers that they run in
  - Scaling of containers to equally balance application workloads across infrastructure
  - Allocation of resources between containers
  - Load balancing, traffic routing and service discovery of containers
  - Health monitoring of containers
  - Securing the interactions between containers.

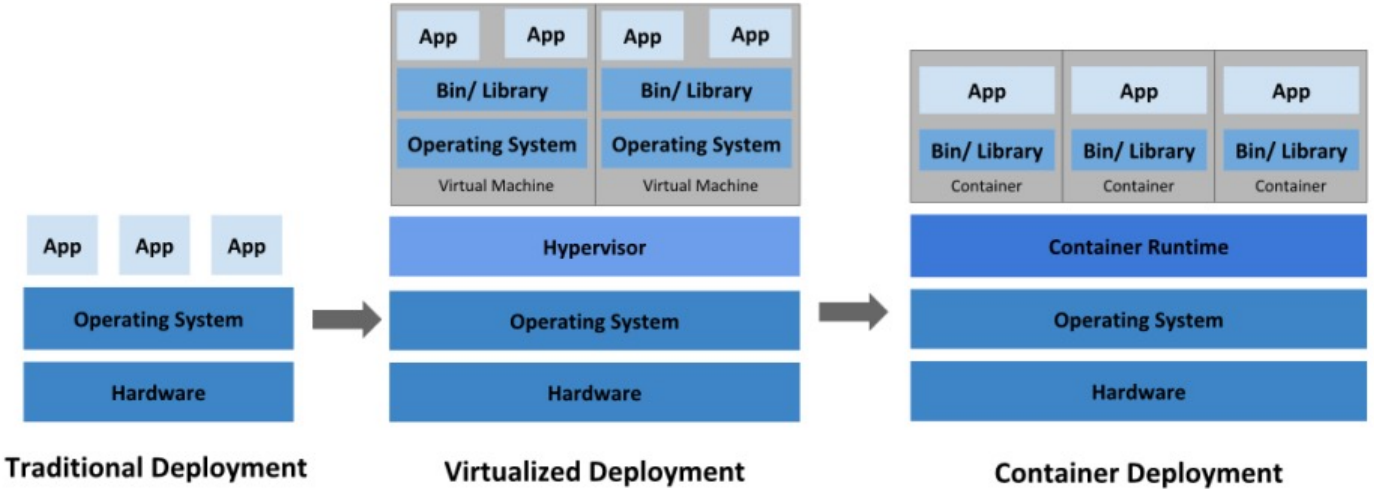
Cloud Orchestration

- Infrastructure-as-a-Service (IaaS)
  - Provisioning virtual machines from a cloud service provider (CSP)

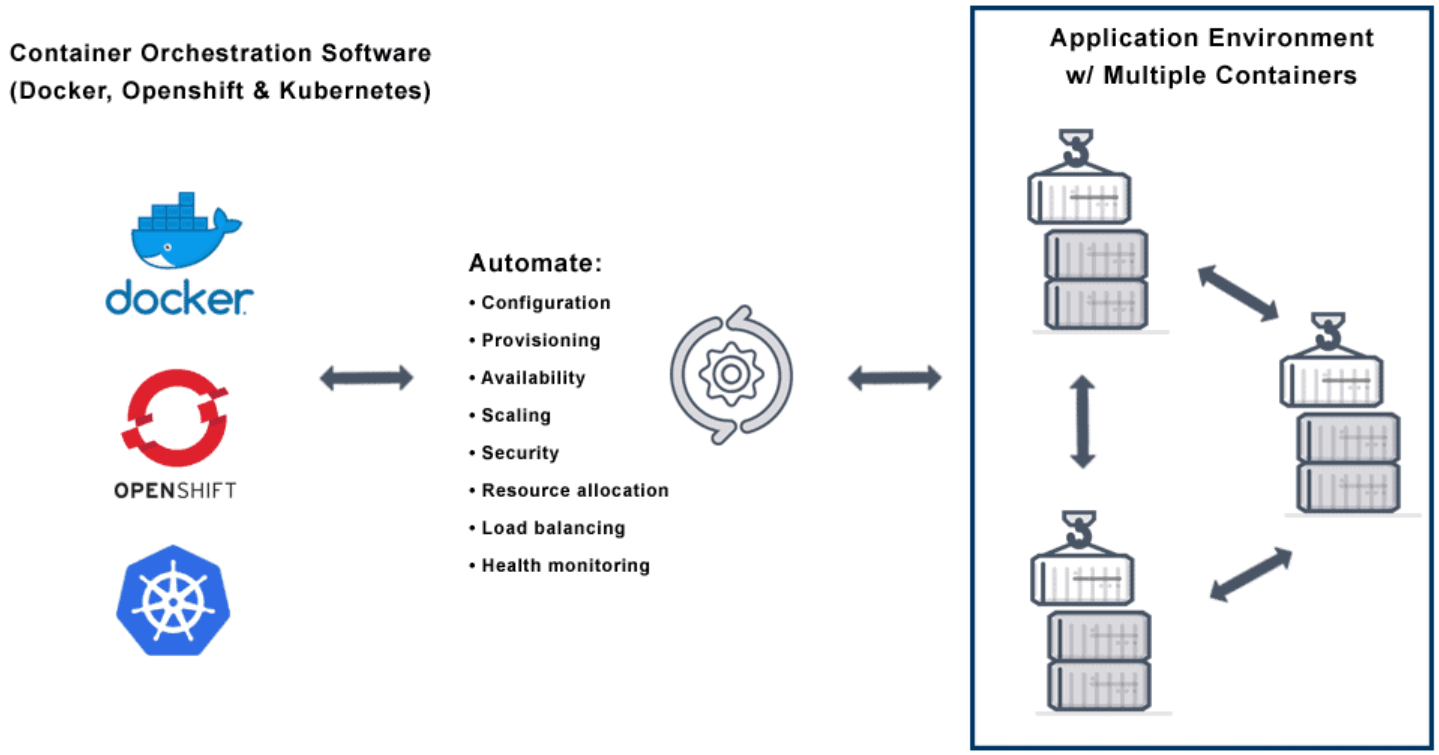


Container Orchestration

- Application containers
  - Lightweight OS-virtualization
  - Application packaging for portable, reusable



Container Orchestration

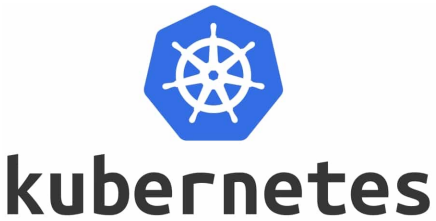


What are the benefits of container orchestration?

- Container orchestration is key to working with containers, and it allows organizations to unlock their full benefits. It also offers its own benefits for a containerized environment, including:
- Simplified operations
  - Most important benefit of container orchestration and the main reason for its adoption.
  - Manages the complexity of Containers
- Resilience
  - Automatically restart or scale a container or cluster, boosting resilience.
- Added security
  - Keeping containerized applications secure by reducing or eliminating the chance of human error.

CONTAINER ORCHESTRATION TOOLS

Container Orchestration Tools



WHAT IS KUBERNETES AND HOW IT WORKS

Kubernetes

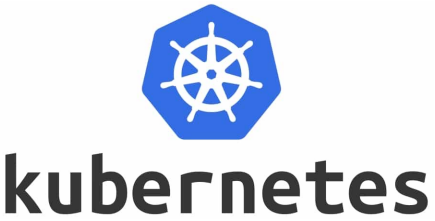
Kubernetes is an open-source container management tool which automates container deployment, container (de)scaling and container load balancing

- Benefits (Works with all cloud vendors (Public, Private (on-premises), and Hybrid))

More about Kubernetes

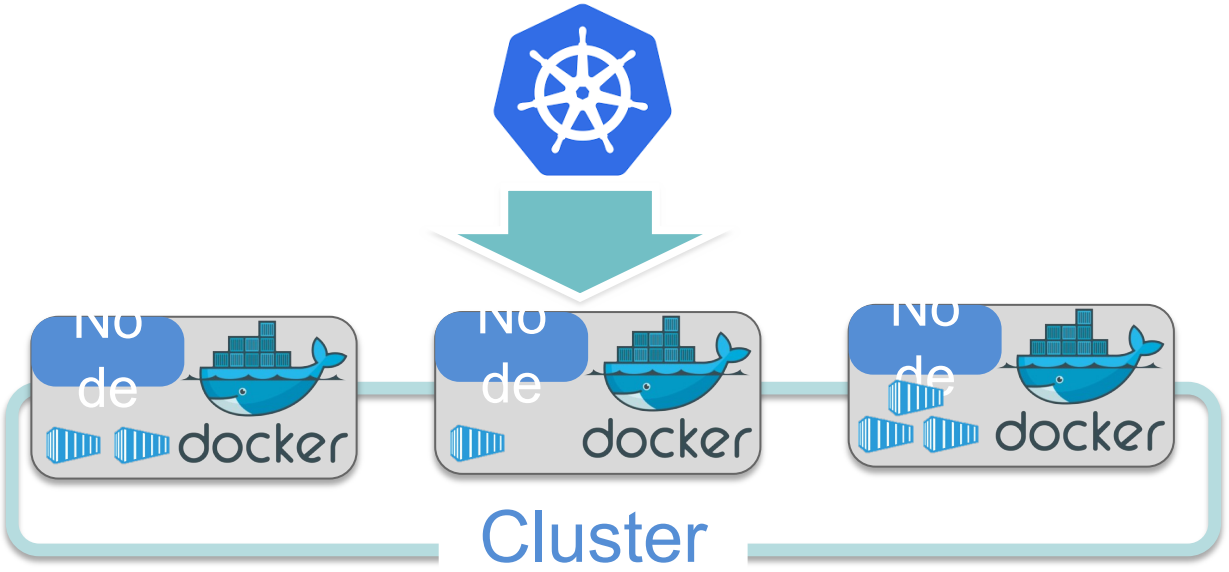
- Developed by Google and written in Golang with a huge community
- Can group ‘n’ containers into one logical unit for managing and deploying them

Kubernetes

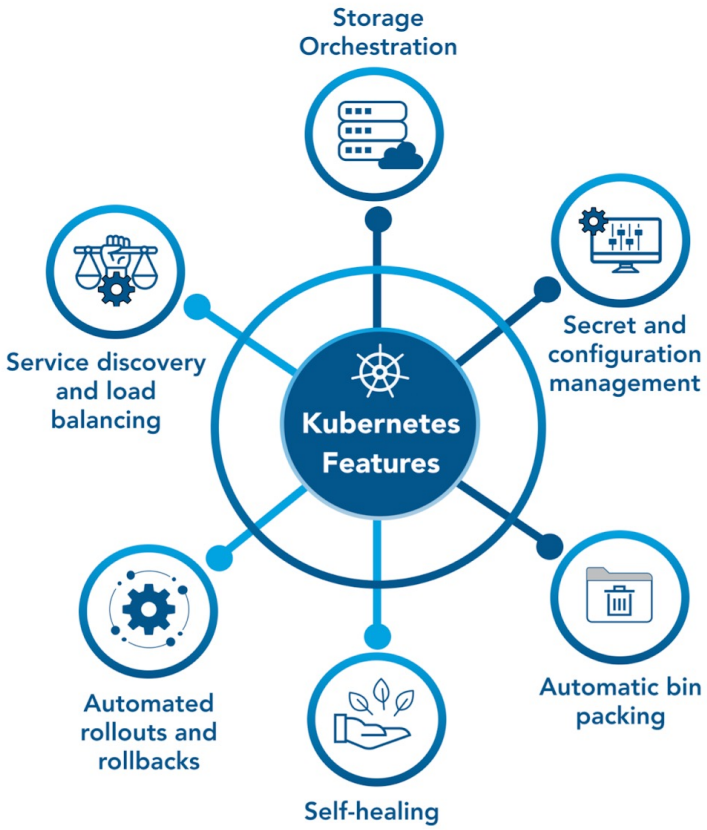


- Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized application.
- Originally an open source project launched by Google and now part of the Cloud Native Computing Foundation (CNCF).
- Kubernetes is highly **extensible** and **portable**
- Kubernetes is considered **highly declarative**
- Kubernetes initial release (**7 June 2014)**
- Releases every 3 months

Kubernetes



Kubernetes Features



Kubernetes Myths

- Kubernetes is not:
  - To be compared with Docker
  - For containerizing apps
  - For apps with simple architecture

- Kubernetes is actually:
  - Robust and reliable
  - A container orchestration platform
  - A solution for scaling up Containers
  - Backed by huge community