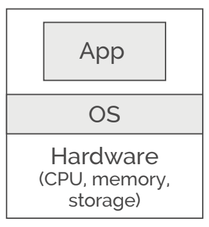
# Kubernetes Primer

## What Kubernetes is, how it evolved, and why you should care

To understand what Kubernetes is, you need to understand the technology developments that preceded it.

kublr

### 1 At first, there were bare metal machines...

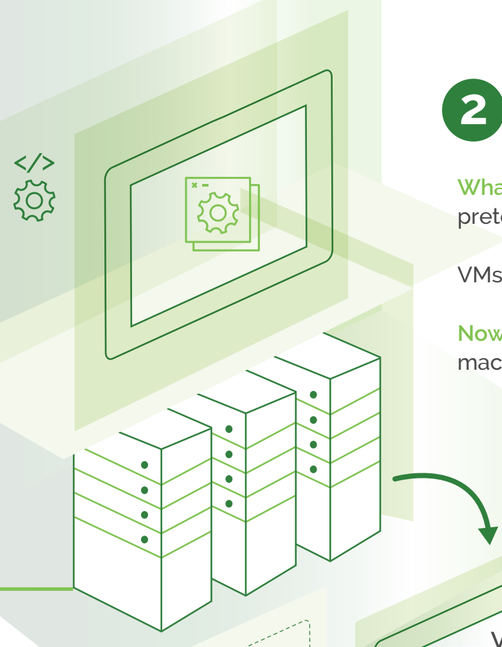



A bare metal machine is a computer with:

+ **Hardware** (central processing unit, memory and storage)
+ **Operating system** (OS)
+ **Applications**

The applications are oblivious of the hardware - they only interact with the OS. Hence, the OS 'abstracts' from the underlying resources.

The challenge: Applications that crash, may drag other co-located applications down, and having just one app on each machine is inefficient.

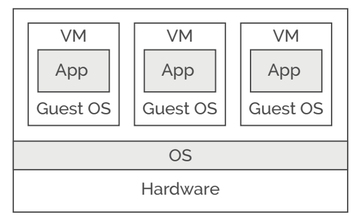


### 2 ...then virtual machines (VMs) came along.

What it is: Code that 'wraps around' an application pretending to be a physical machine.

VMs require their own OS, referred to as guest OS

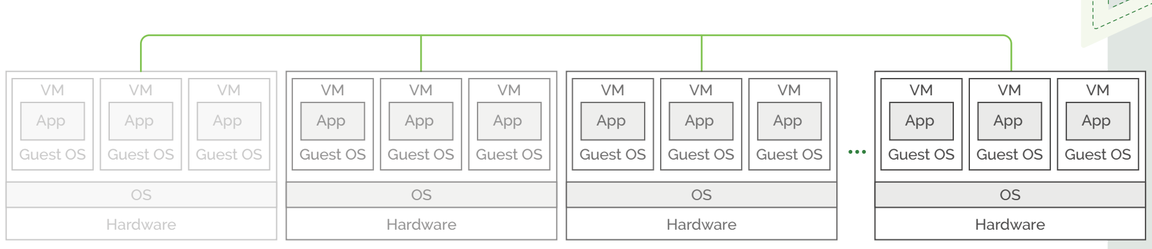
Now: VMs allow multiple applications to run on the same machine without impacting each other should they crash.

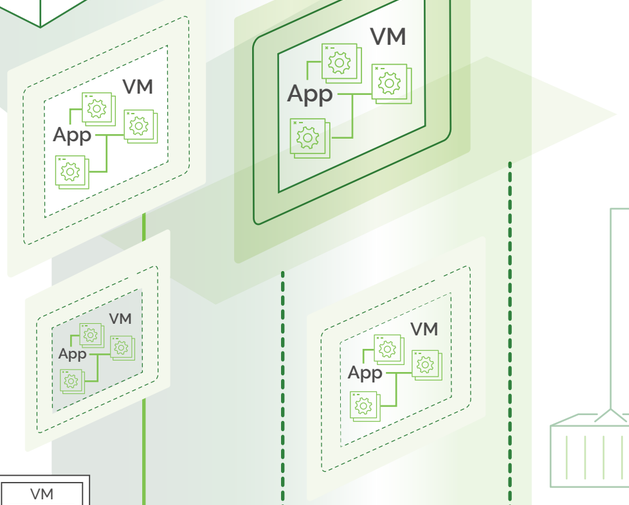



### 3 But what about really BIG apps? That's where distributed systems come in

Enterprise applications often require way more computing capacity than a single machine can handle, so multiple machines are connected through a network to form a distributed application/system.

A group of machines that works together forms a cluster; each machine in the cluster is called a node. For a cluster to work as an entity, the nodes must communicate.

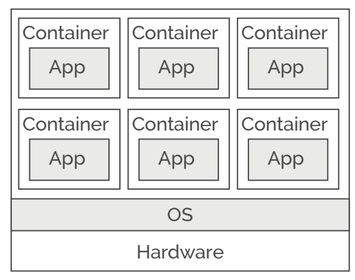



### 4 What came next? Containers, the new lightweight VMs

Containers are similar to VMs but they don't require a guest OS, so they consume far less resources.

Benefit: More containers can be deployed on a single machine than VMs. Additionally, they are portable and can be moved without affecting the application.



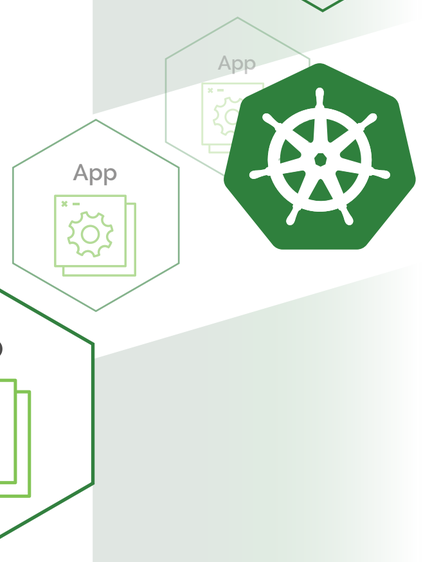

### 5 Microservices: Breaking applications down into smaller pieces

Microservices are applications that are broken down into small components that can run independently.

Each micro component is called a service—hence microservices.

More components mean more VMs or more containers are needed.

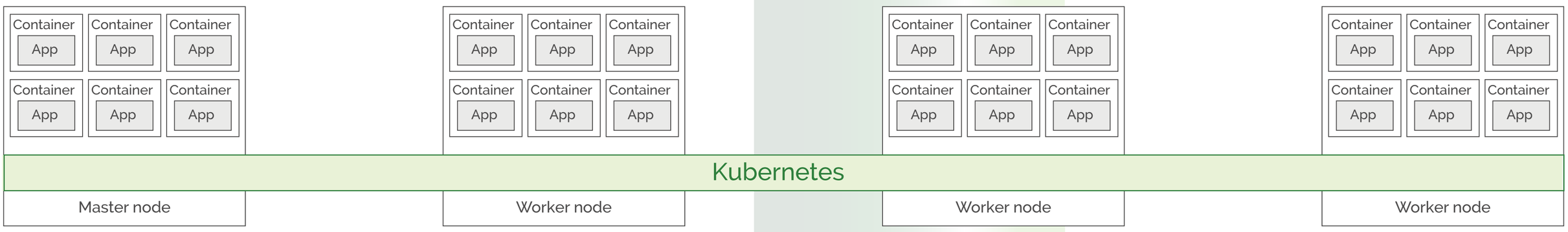
Which is better? Since containers require fewer resources than VMs, microservices with containers are the more affordable option. That's why we are hearing a lot more about them now.




### 6 Finally! Kubernetes enters the world stage

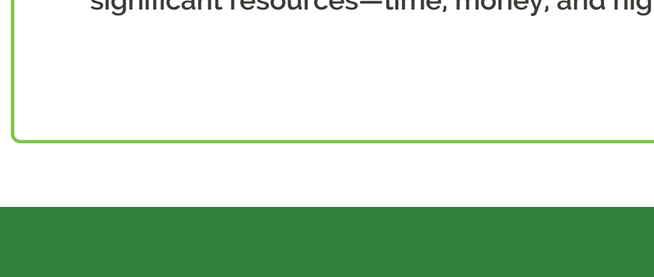
**Imagine this:** A legacy app with hundreds of VMs migrates to containers. Now it's composed of thousands of containers. Managing them manually is messy, impossible.

**Enter Kubernetes.** Kubernetes is like a data center or cluster OS which manages the cluster resources, ensuring all containers are up and running.

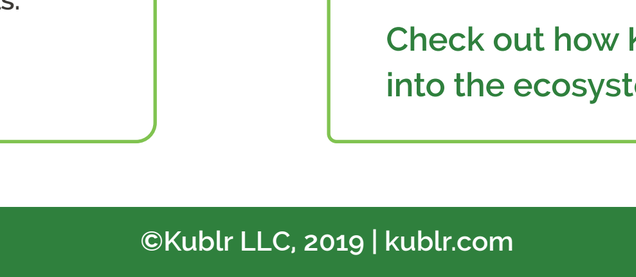



## Why Manage with Kubernetes?

There are 3 main benefits



Declarative state—Developer declares desired state, or characteristics of the app to ensures the cluster doesn't deviate from it.



Self-healing—Kubernetes ensures the cluster always matches the declarative state and heals itself and the affected applications when there is a discrepancy.

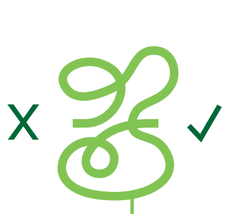
Auto-scaling—Kubernetes has the capacity to automatically scale up (and create more nodes and/or pods) when capacity peaks, then reduce again after a peak.

## Kubernetes is great, but...

### a

...Kubernetes is difficult.

- It's hard to find expertise
- It's not a full-fledged, ready-to-use solution
- Enterprises need an entire technology stack



Configuring and fine-tuning the different technologies so they work well with Kubernetes is no easy task and requires significant resources—time, money, and highly specialized skills.

### b

Because Kubernetes is so complex....
It gave birth to numerous Kubernetes solutions, all trying to ease the challenges of adoption.

While the space seems crowded with many competing solutions, it really isn't. Many of the solutions address different challenges and use cases.

Check out how Kublr fits into the ecosystem.

LEARN MORE

kublr