

PME – Introduction to Kubernetes and Cloud Computing Landscape



25th Oct 2022



7pm



Microsoft Teams



Co-Host: Yusuf Giwa
Beta Ambassador



Co-Host: Marko Paloski
Gold Ambassador



Registration link: <https://bit.ly/3CLL2WQ>

Speaker



Yusuf Giwa

- Beta Microsoft Learn Student Ambassador
- KCNA (Kubernetes and Cloud Native Associate)
- Microsoft Certified (Azure)
- Engineering Student, University of Ibadan



Co-host




Marko Paloski



Marko Paloski is MLSA Gold Student and System Engineer who is enthusiast for building communities among youth and tech community. Passionate about video games, non-formal education and tech stuff. His expertise are in the field of Windows Server, Office 365, Microsoft Azure and IoT. Also he is Microsoft Certified Professional.



Agenda

- ❖ Intro to Cloud Computing
 - ❖ Understanding Container services
 - ❖ Introduction to Kubernetes
 - ❖ Creating Kubernetes Components using declarative and Imperative syntax
 - ❖ Projects in the Cloud computing landscape
 - ❖ Quiz Time
- 
- The slide features two horizontal bars at the bottom. The top bar is blue with three small light blue dots. The bottom bar is purple with three small light purple dots.

Intro to cloud computing



What is Cloud computing?

- Cloud computing is renting resources, like storage space or CPU cycles, databases, networking, software, analytics , intelligence etc.
- You only pay for what you use. The company providing these services is referred to as a cloud provider.
- Microsoft is one of the cloud provider which is **AZURE**.
- Cloud computing makes running a business easier. It's cost-effective, scalable, elastic, current, reliable, and secure. This means you're able to spend more time on what matters and less time managing the underlying details.



Building blocks of Cloud Computing

We present four main building blocks in cloud computing: application software, development platforms, resource sharing, and infrastructure. The infrastructure includes the physical resources in a datacenter. The resource sharing layer typically entails software and hardware techniques that allow the sharing of the physical resources while offering a certain level of isolation. The development platforms are utilized to develop cloud applications.



Major cloud providers

1. Microsoft Azure

Compute:

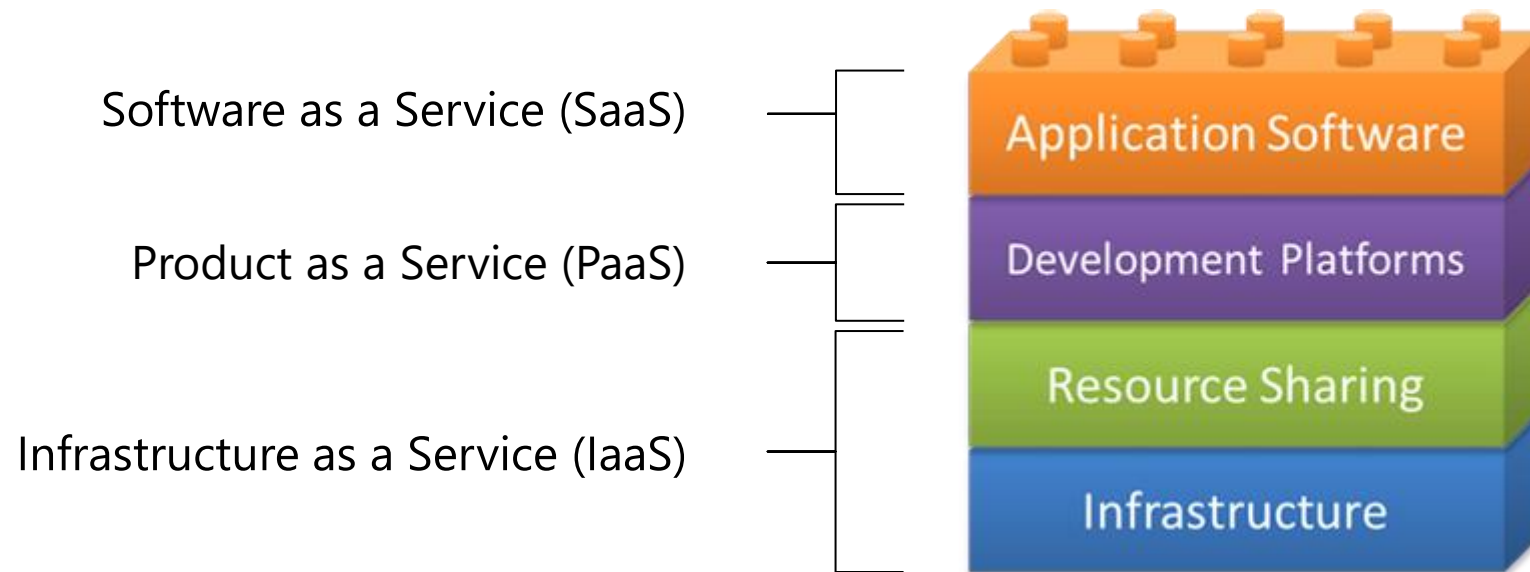
- Virtual Machines
- Machines could run Windows or any flavors of Linux

Storage: Azure offers several storage solutions, including:

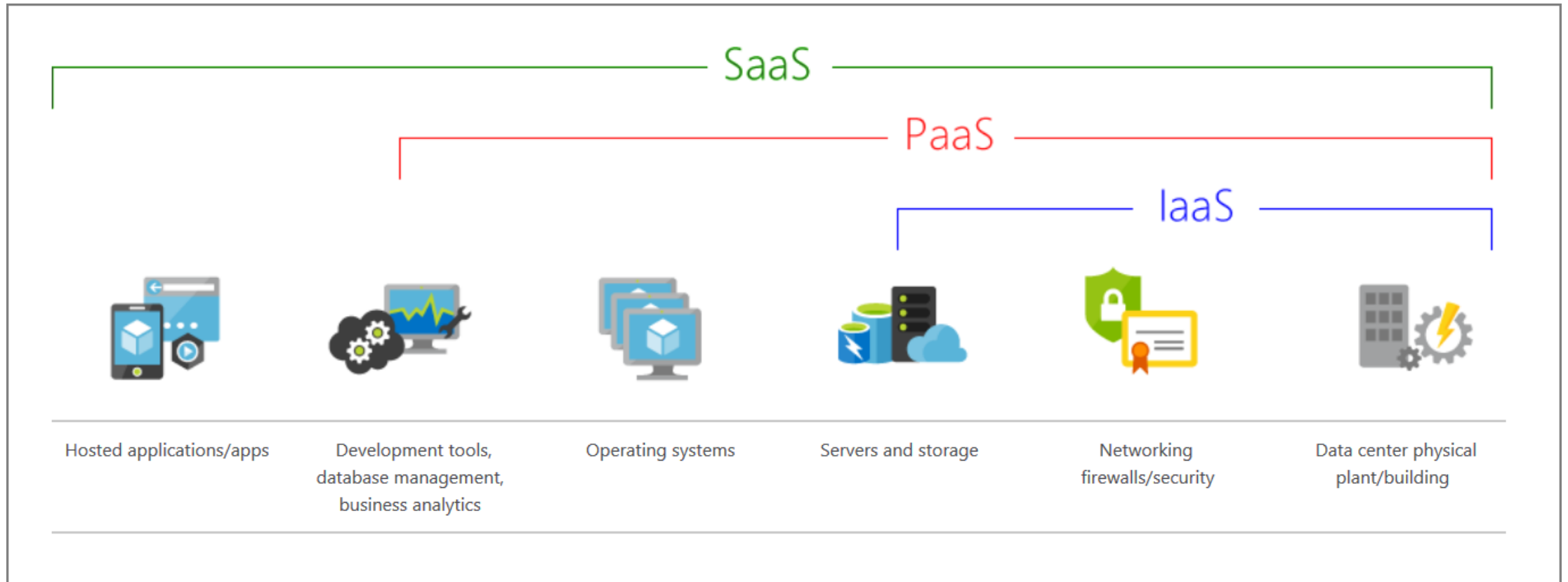
- Azure Blob storage
 - Azure Table
 - Azure File Storage
 - Azure Cosmos DB
 - StorSimple
- 
- A decorative graphic at the bottom of the slide consisting of two horizontal bars. The top bar is blue with three small white dots. The bottom bar is purple with three small white dots.

Cloud computing services

In a broad sense, cloud services differ based on the needs of different users. This section reviews three popular types of cloud services:



Cloud computing services – The big picture



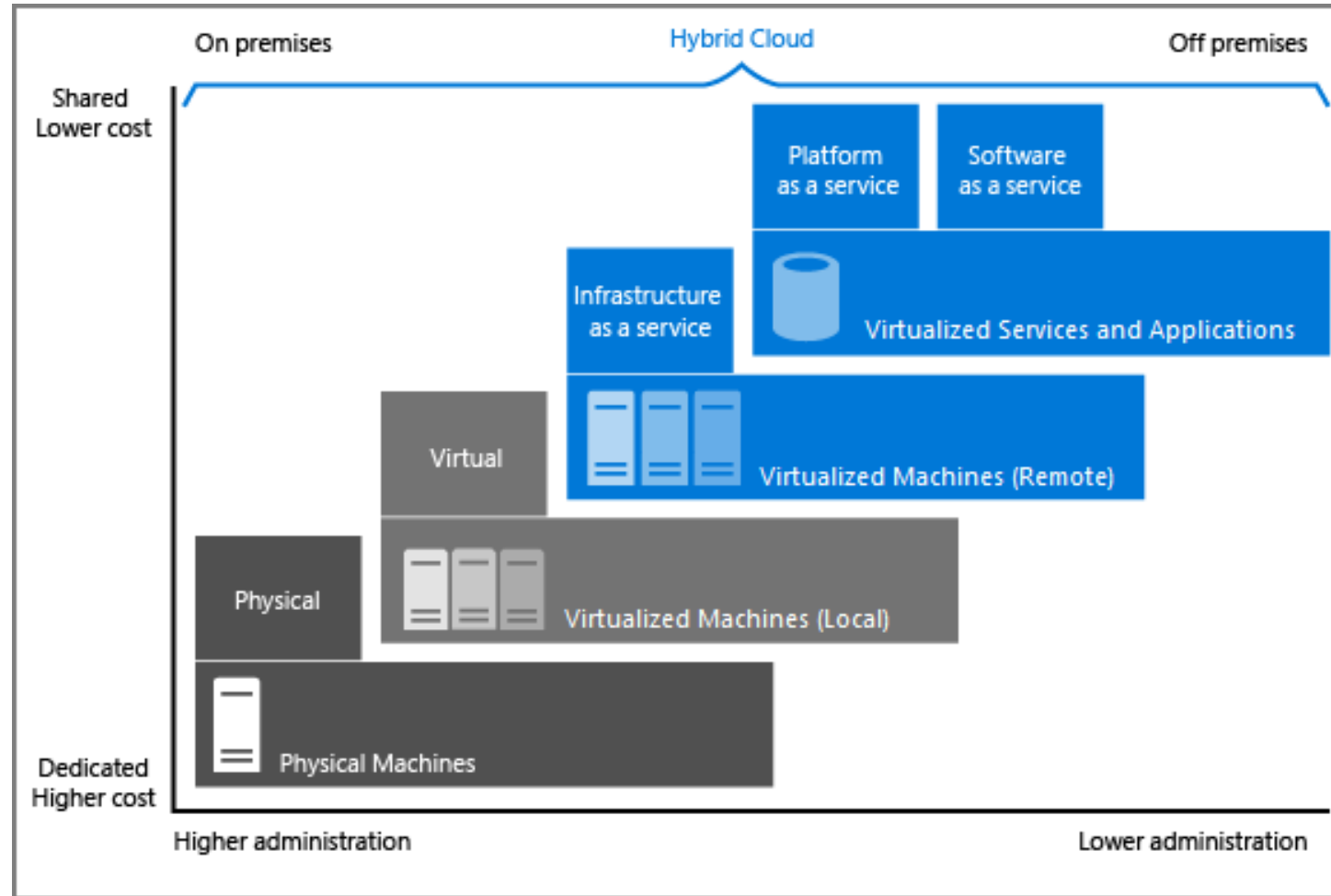
Types of clouds

There are three well-known deployment models for cloud computing:

- **Public cloud:** Services are offered over the public internet and available to anyone who wants to purchase them. Cloud resources like servers and storage are owned and operated by a third-party cloud service provider and delivered over the internet.
- **Private cloud:** Computing resources are used exclusively by users from one business or organization. A private cloud can be physically located at your organization's on-site datacenter. It also can be hosted by a third-party service provider.
- **Hybrid cloud:** This computing environment combines a public cloud and a private cloud by allowing data and applications to be shared between them.



Types of clouds



Understanding Containers



Containers

- Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications.
- Containers decouple applications from underlying host infrastructure. This makes deployment easier in different cloud or OS environments.
- A container image is a ready-to-run software package, containing everything needed to run an application: the code and any runtime it requires, application and system libraries, and default values for any essential settings.
- By design, a container is immutable: you cannot change the code of a container that is already running. If you have a containerized application and want to make changes, you need to build a new image that includes the change, then recreate the container to start from the updated image.



Container Evolution

Traditional deployment era:

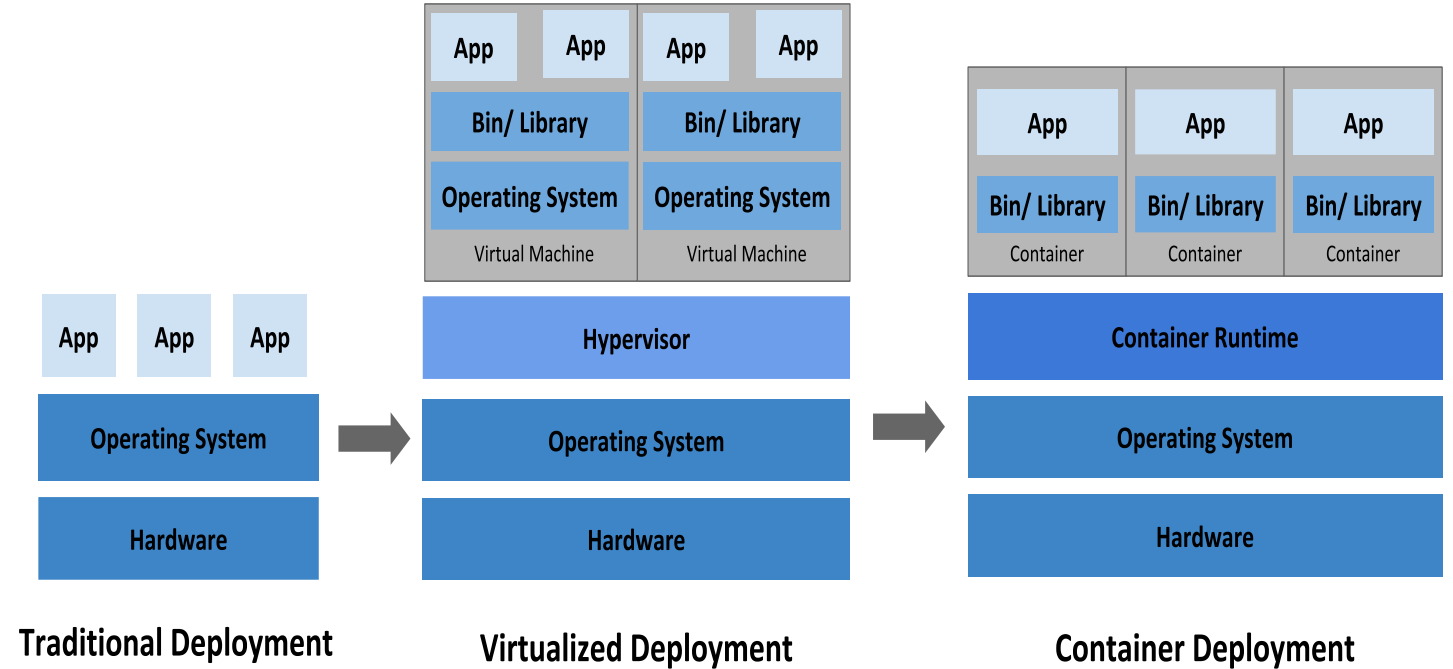
Physical servers: No application boundary for applications: Resource allocation issues.

Virtualized deployment era:

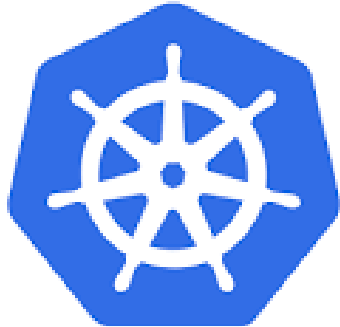
Multiple Virtual Machines (VMs) on a single physical server's CPU:
Improved Security and Scalability:
Virtualized hardware.

Container deployment era:

Resource isolation: Resource utilization: Application-centric management: Environmental consistency across development.

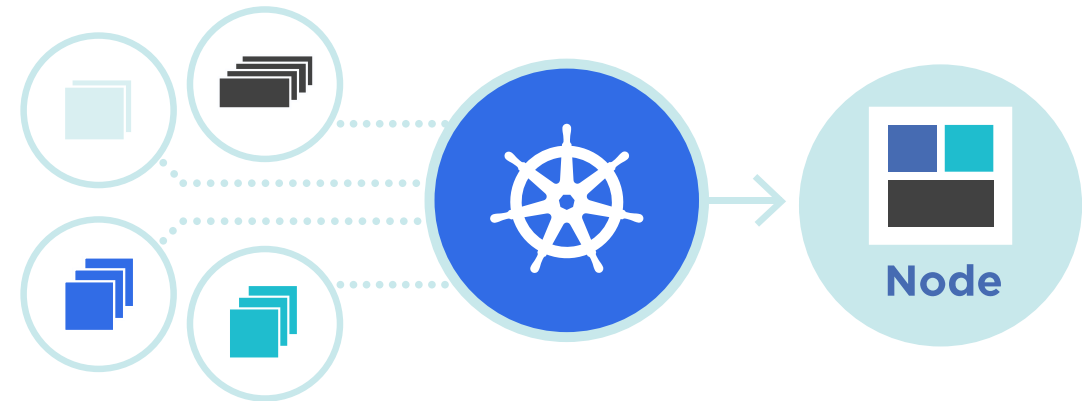


Intro to Kubernetes



What is Kubernetes?

- Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications
- It groups containers that make up an application into logical units for easy management and discovery
- Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s"..
- Kubernetes builds upon [15 years of experience of running production workloads at Google](#), combined with best-of-breed ideas and practices from the community



Why Kubernetes?

- After bundling and running our applications with container. We need to manage the containers that runs the application to ensure that there is no downtime. i.e. if a container goes down, another container needs to start. Kubernetes comes to the rescue!
- Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more.
- Although, Kubernetes provides some generally applicable features common to PaaS offerings, such as deployment, scaling, load balancing, and lets users integrate their logging, monitoring, and alerting solutions.
- It is not a traditional, all-inclusive PaaS (Platform as a Service) system since Kubernetes operates at the container level rather than at the hardware level.

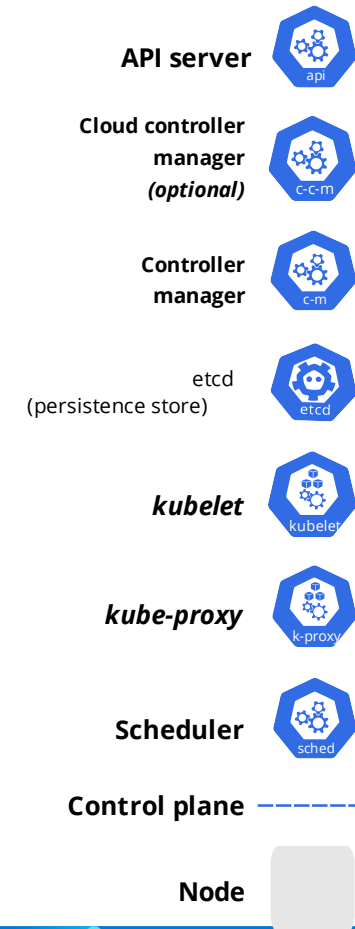
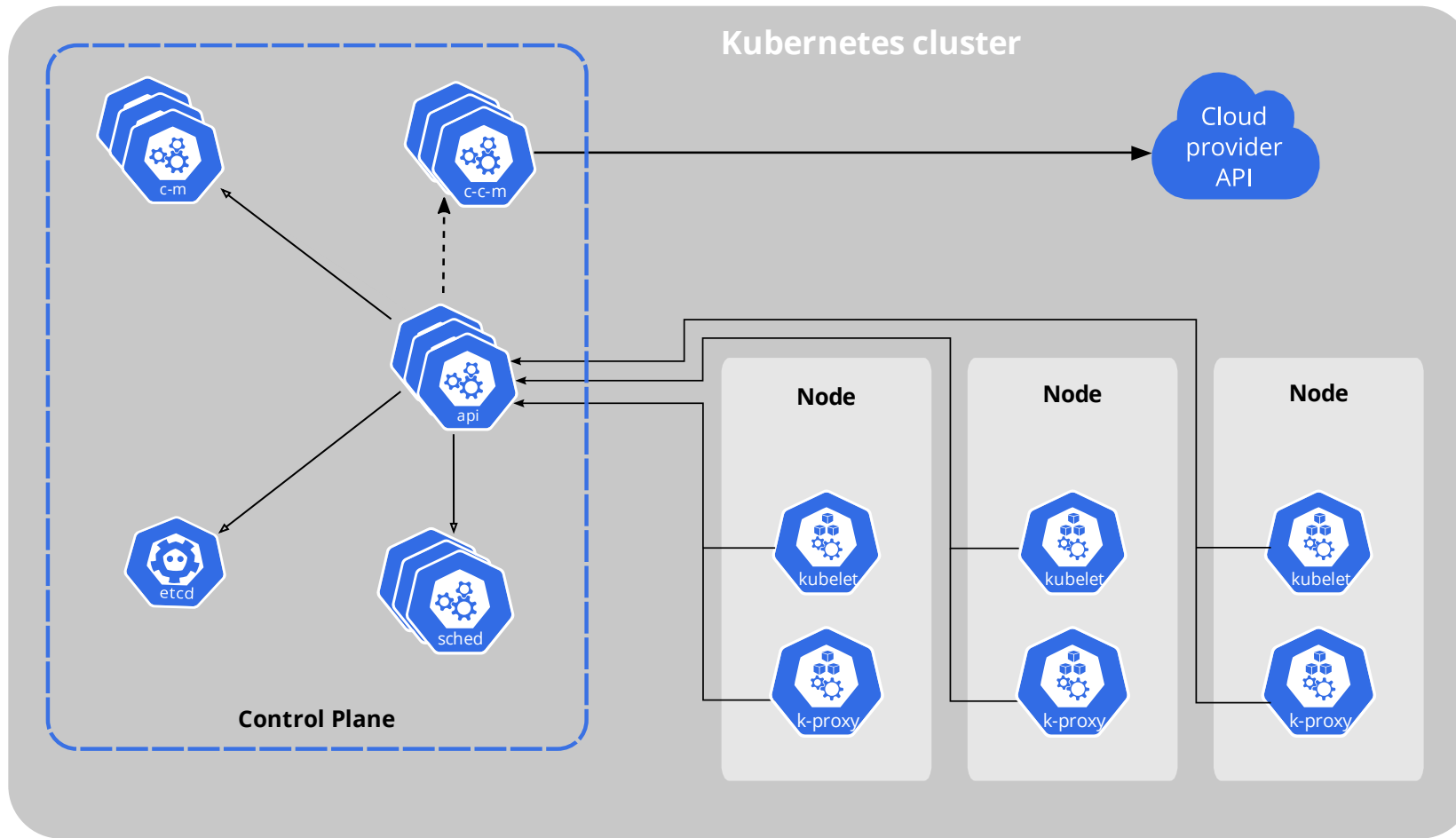


Kubernetes features


Automated rollouts and rollbacks	Storage orchestration	Service discovery and load balancing	Secret and configuration management	Self-healing
<p>Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time.</p> <p>If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.</p>	<p>Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as AWS or GCP, or a network storage system such as NFS, iSCSI, Ceph, Cinder.</p>	<p>No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.</p>	<p>Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.</p>	<p>Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.</p>

Kubernetes Cluster Architecture

When you deploy Kubernetes, you get a cluster.



Kubernetes Components

- 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.
 - 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.
 - The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).
 - Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.
- 

Kubernetes Objects

Kubernetes objects are persistent entities in the Kubernetes system. Kubernetes uses these entities to represent the state of your cluster. Specifically, they can describe:

- What containerized applications are running (and on which nodes)
- The resources available to those applications
- The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance

A Kubernetes object is a "record of intent"--once you create the object, the Kubernetes system will constantly work to ensure that object exists. By creating an object, you're effectively telling the Kubernetes system what you want your cluster's workload to look like; this is your cluster's desired state.



Workloads

- A workload is an application running on Kubernetes. Whether your workload is a single component or several that work together, on Kubernetes you run it inside a set of pods.
- **A Pod** represents a set of running containers on your cluster.
- Pods are ephemeral. I.e when the nodes running a pods fail. We have to create a new pod.
- To make life considerably easier, you don't need to manage each Pod directly. Instead, you can use workload resources that manage a set of pods on your behalf.



Workloads

- A workload is an application running on Kubernetes. Whether your workload is a single component or several that work together, on Kubernetes you run it inside a set of pods.
- **A Pod** represents a set of running containers on your cluster.
- Pods are ephemeral. I.e when the nodes running a pods fail. We have to create a new pod.
- To make life considerably easier, you don't need to manage each Pod directly. Instead, you can use workload resources that manage a set of pods on your behalf.



Workloads Objects

Kubernetes provides several built-in workload resources:

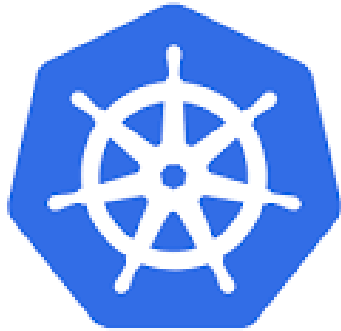
- **Deployment and ReplicaSet:** Deployment is a good fit for managing a stateless application workload on your cluster, where any Pod in the Deployment is interchangeable and can be replaced if needed.
- Deployment and ReplicaSet replacing the legacy resource ReplicationController
- **StatefulSet** lets us run one or more related Pods that do track state somehow. For example, if your workload records data persistently, you can run a StatefulSet that matches each Pod with a PersistentVolume. Your code, running in the Pods for that StatefulSet, can replicate data to other Pods in the same StatefulSet to improve overall resilience.



Workloads Objects

- **DaemonSet** defines Pods that provide node-local facilities. These might be fundamental to the operation of your cluster, such as a networking helper tool, or be part of an add-on.
- DaemonSet helps us create pods that run on all nodes.
- Every time we add a node to your cluster that matches the specification in a DaemonSet, the control plane schedules a Pod for that DaemonSet onto the new node.
- **Job and CronJob** define tasks that run to completion and then stop.
- Jobs represent one-off tasks, whereas CronJobs recur according to a schedule.
- Using a **custom resource definition**, we can add in a third-party workload resource if you want a specific behavior that's not part of Kubernetes' core.





Kubernetes Demo

<https://kubernetes.io/docs/tutorials/>

- Imperative Approach
- Declarative Approach
- Via File (Kubernetes Manifest)
- Using **Kubectl** options line to interact with Kubernetes API



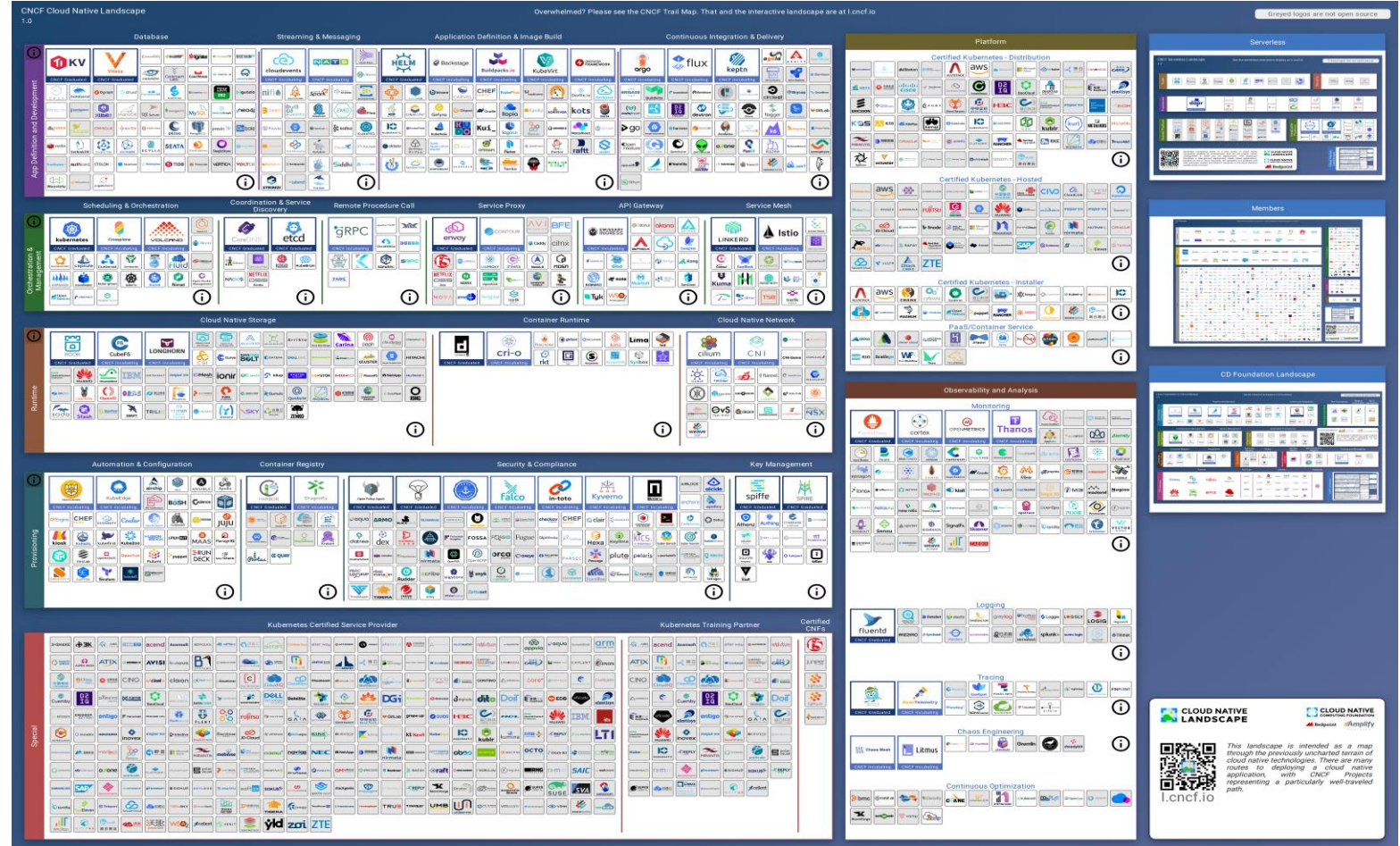
Cloud Native Landscape

- Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.
- These techniques enable loosely coupled systems that are resilient, manageable, and observable.
- Cloud native is about speed and agility.
- Researching about cloud native applications and technologies can be overwhelming.
- The goal of the cloud native landscape is to compile and organize all cloud native open-source projects and proprietary products into categories.



Cloud Native Landscape

- Provisioning
- Runtime Orchestration & Management App
- Definition and Development
- Observability and Analysis





Quiz time

Contact Us



LinkedIn: Yusuf Giwa



LinkedIn: Marko Paloski



Microsoft Learn
Student Ambassadors

Thank you

