#TechforPeople

devoteam

Kubernetes

Creative tech for Better Change

Vladimiro Luz
vladimiro.luz@devoteam.com

devoteam

Kubernetes vs Docker

Creative tech for Better Change

Docker

Docker is a platform for developing, shipping, and running applications in lightweight, portable containers

- Creates and runs containers
- Provides an isolated environment for applications
- Works on any system (Windows, Mac, Linux)

Kubernetes (K8s)

Kubernetes is an orchestration system that manages, deploys, scales, and maintains containers across multiple machines.

- Automates deployment & scaling
- Manages containerized applications across clusters
- Handles failures and self-healing

Docker vs K8s

	Docker	K8s
Purpose	Runs containers	Manages and orchestrates containers
Complexity	Simple setup, easy to use	Complex setup, requires configuration
Scalability	Limited scalability	Highly scalable, supports large clusters
Networking	Basic networking, works on a single machine	Advanced networking, supports multi-node clusters
Self-healing	No built-in self-healing	Automatically restarts failed containers

Docker

Benefits	VS	Disadvantages
Lightweight & Fast Containers start in seconds		**No Built-in Orchestration** Needs additional tools for scaling
Easy to Use Simple CLI commands for managing containers		**Limited Self-healing** If a container crashes, manual restart is needed
Portability Runs anywhere: dev, test, or production		**Networking Complexity** Can be tricky to configure for multi-host environments
Efficient Uses fewer resources than VMs		**Security Risks** a vulnerability in one container could impact others if not properly isolated

K8s

Benefits	VS	Disadvantages
Scalability Manages thousands of containers across multiple machines		**Complex to Set Up** Requires knowledge of YAML, clusters, and networking
Self-Healing Restarts failed containers automatically		**Resource Intensive** Needs more CPU, memory, and storage
Load Balancing Efficiently distributes traffic across containers		**Steep Learning Curve** Harder for beginners compared to Docker
Declarative Configuration Uses YAML files for automation		**Difficult Debugging** Troubleshooting a cluster can be complex due to the number of components involved

When to use Docker vs K8s

Use Docker when	Use Kubernetes when
Developing & testing applications locally	You need advanced features like auto-scaling and self-healing
Running a small number of containers	Managing multiple containers across different environments
You don’t need auto-scaling	Deploying applications at scale

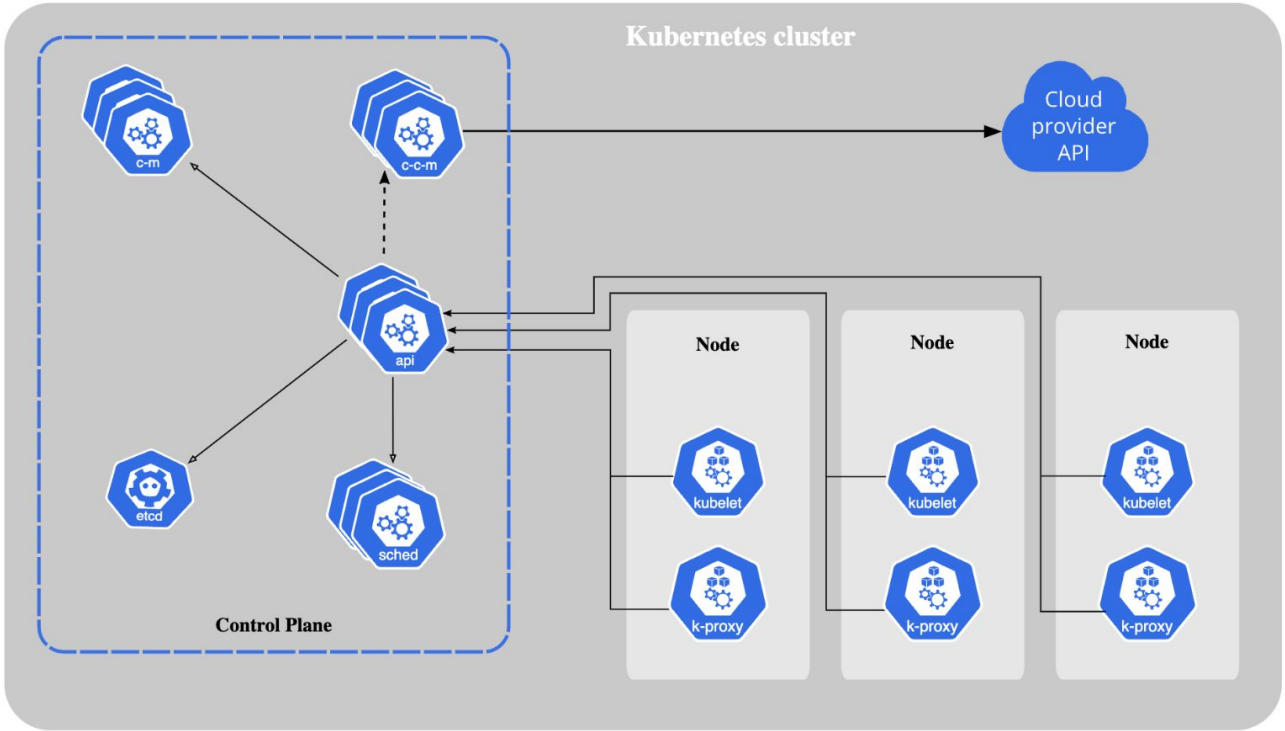
devoteam

Kubernetes Architecture

Creative tech for Better Change

K8s Architecture

K8s Architecture



API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

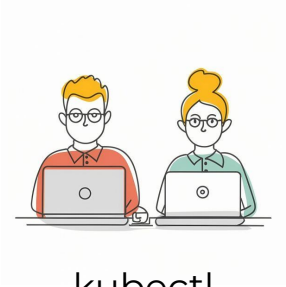
kube-proxy

Scheduler

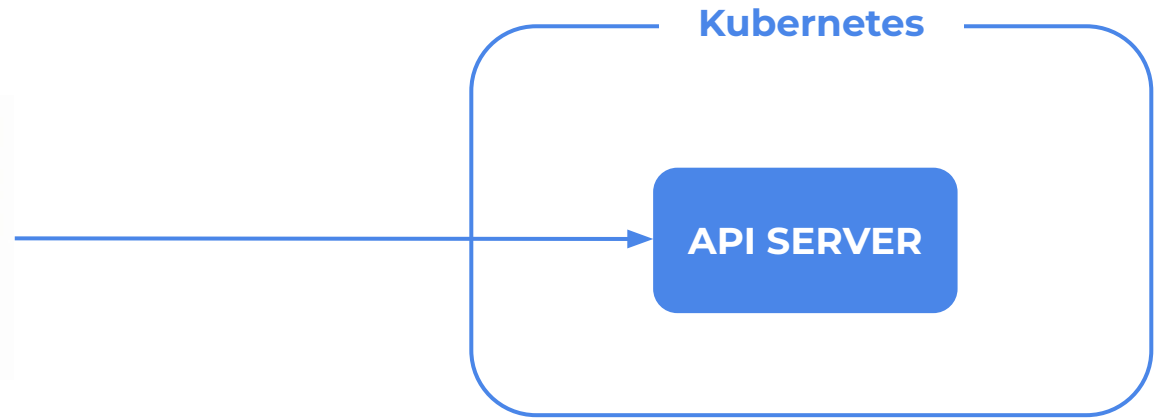
Control plane

Node

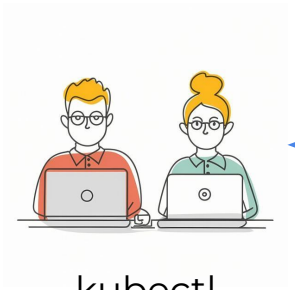
Kubectl



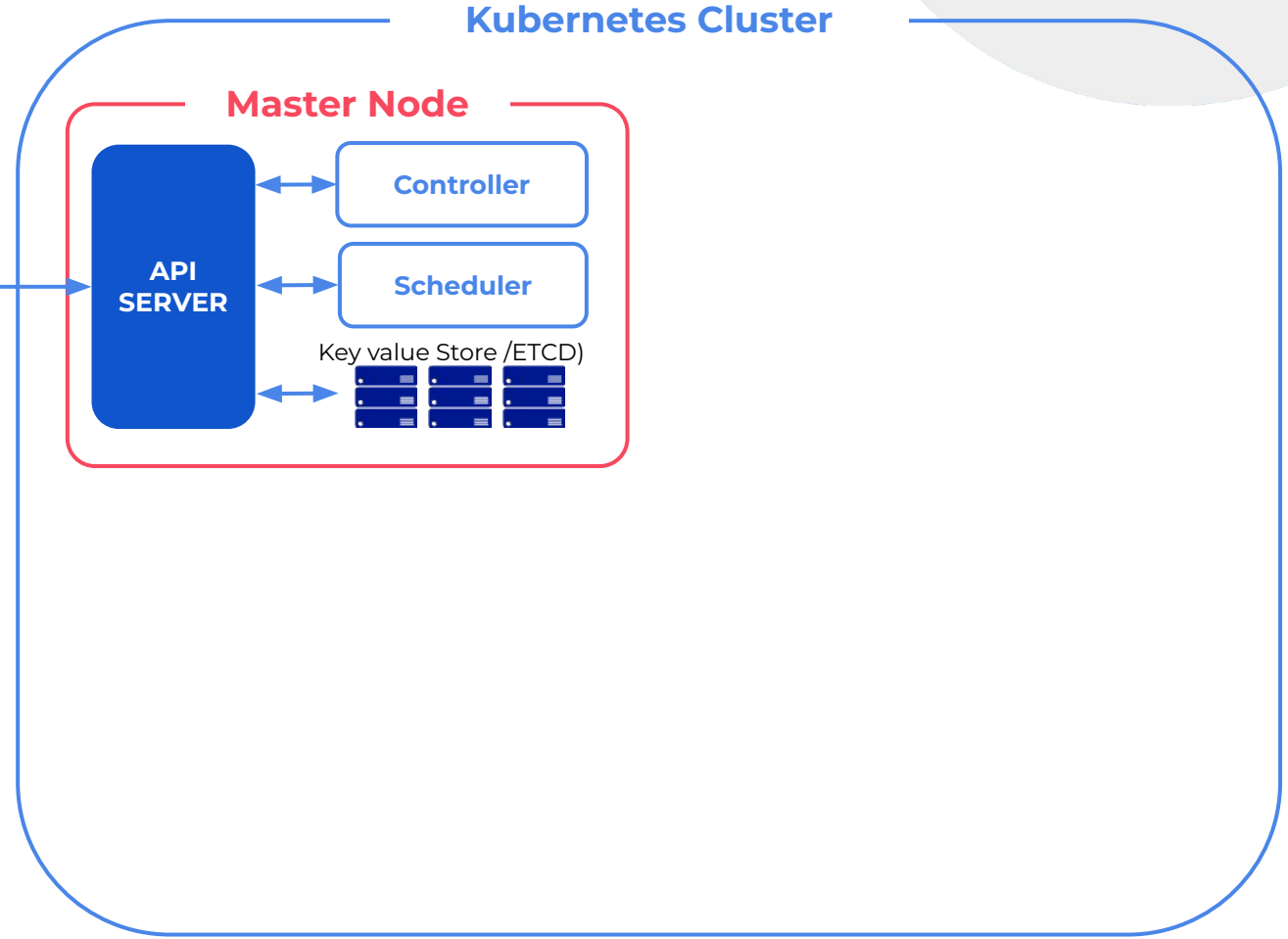
kubectl



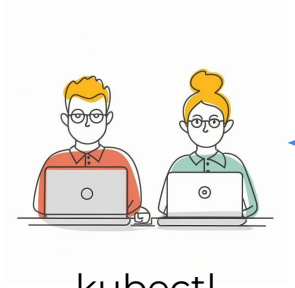
Master Node



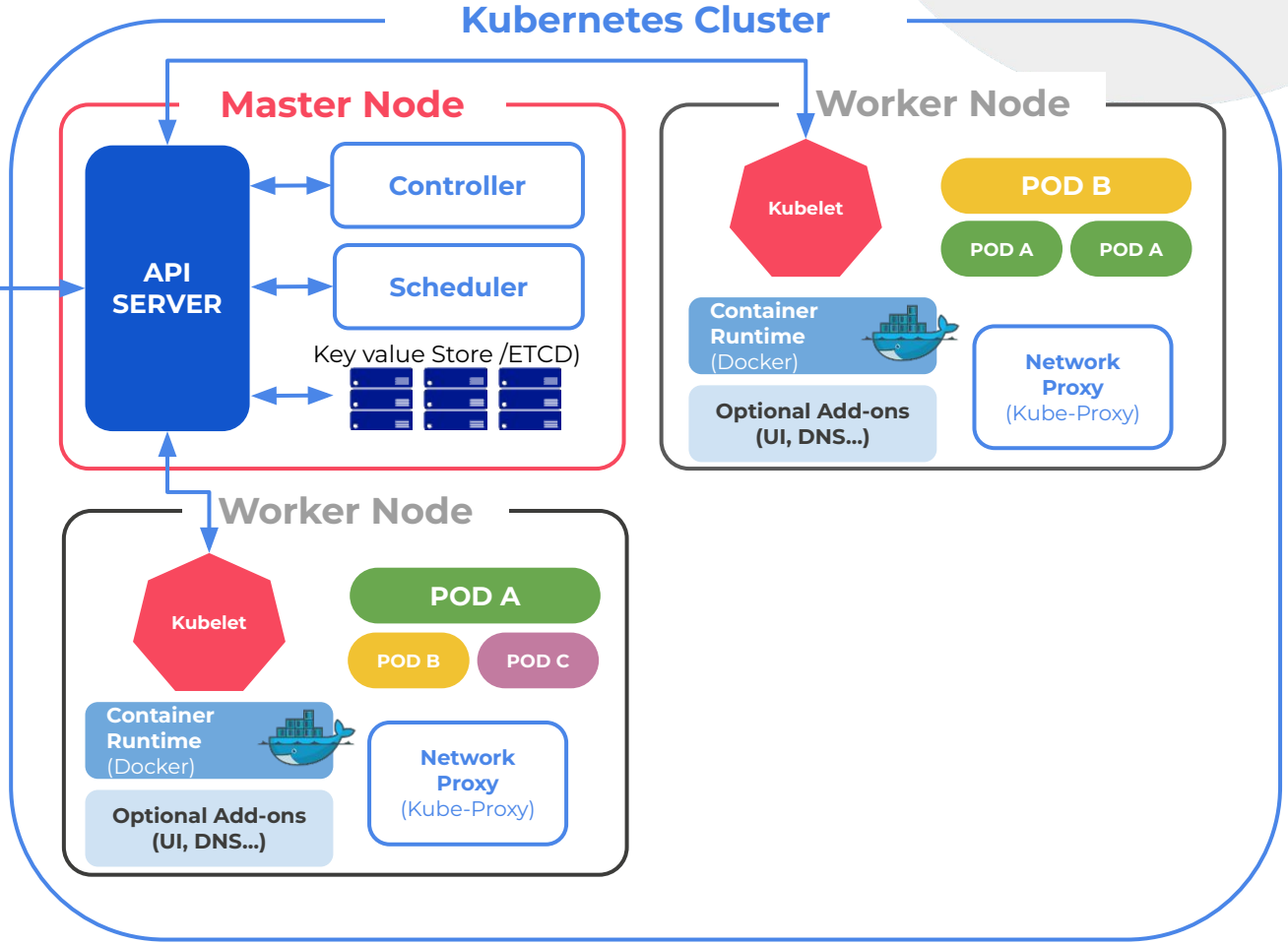
kubectl



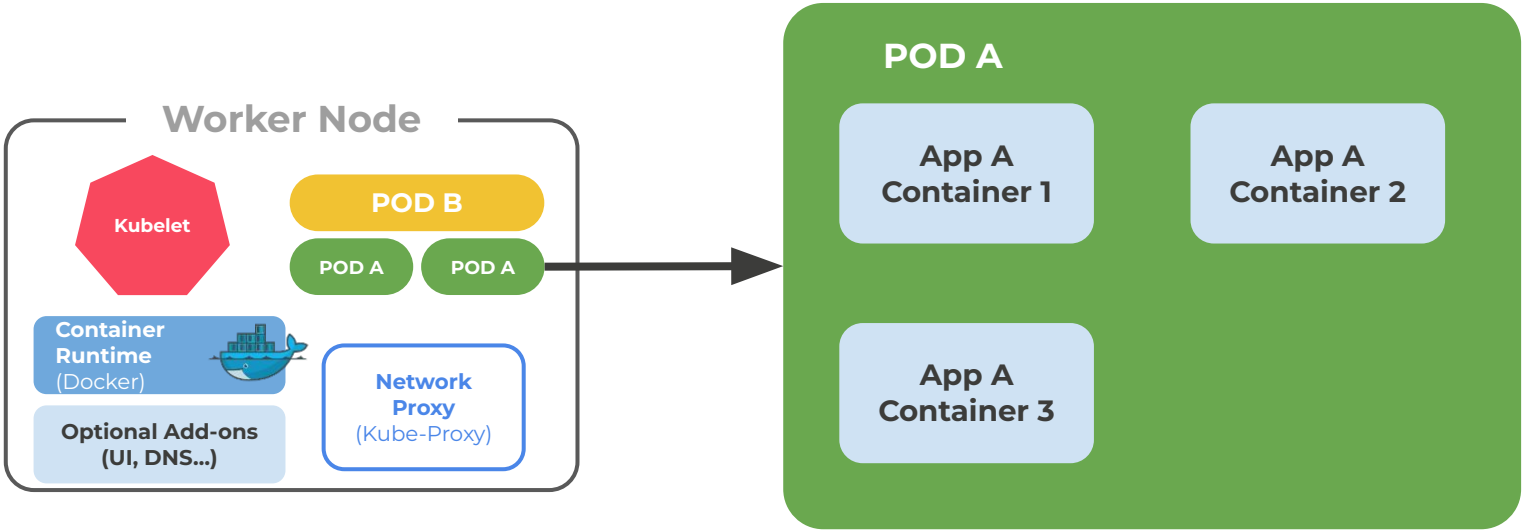
Worker Node



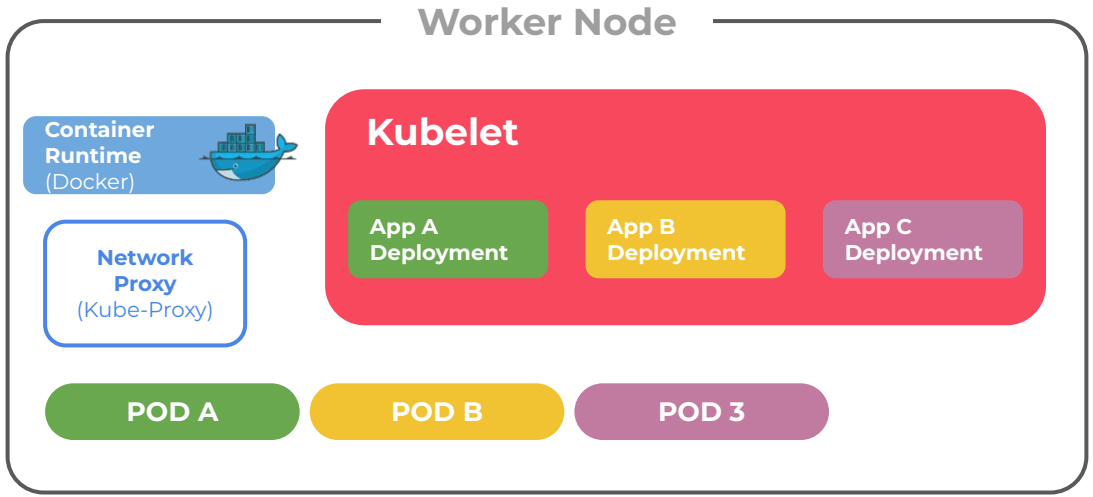
kubectl



Kubernetes Pod



Kubernetes Deployment



Volumes

Volumes

Are a directory which is accessible to all of the containers in a Pod.

Some Volumes are ephemeral.

Some Volumes are persistent.

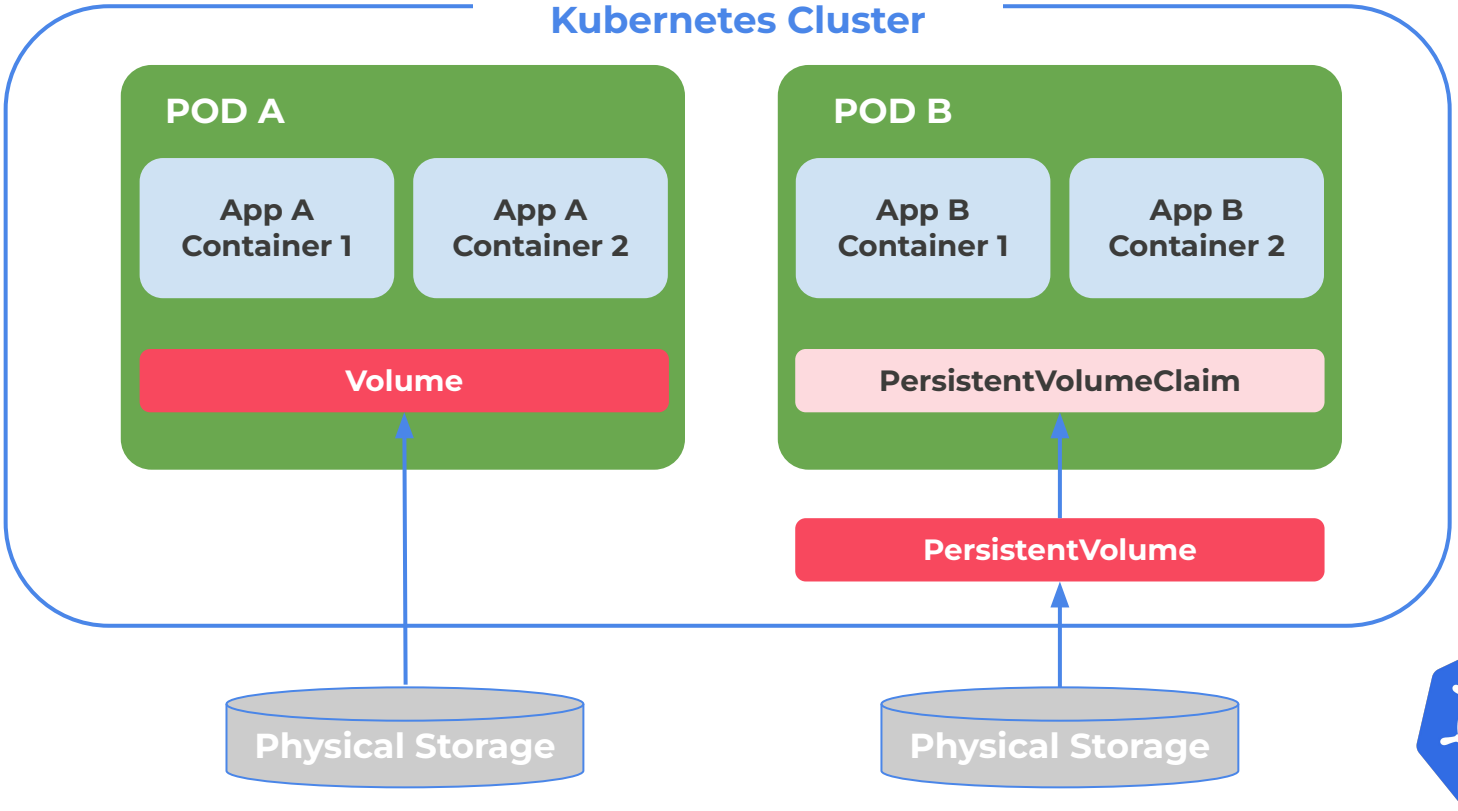
PersistentVolumes

Manage durable storage in a cluster.

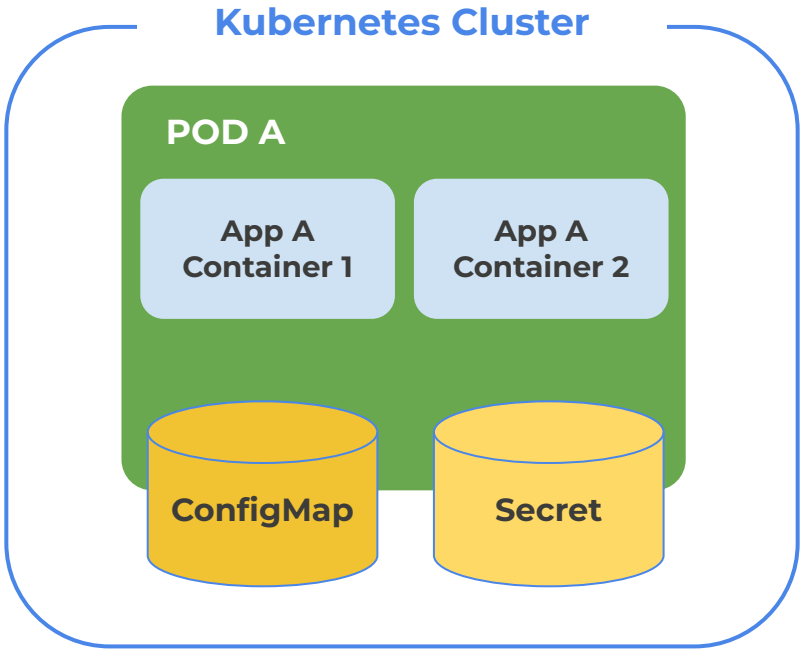
Are independent of the Pod's lifecycle.

Provisioned dynamically through PersistentVolumeClaims or explicitly created by a cluster admin.

Volume and PersistentVolume



Secrets & ConfigMaps



Init Containers

Init Containers are “special” containers that run before the main application container starts:

- Run sequentially (one after the other)
- Ensure dependencies are met before the app starts
- Have separate images from the main container (can use different tools

When should it be used:

- Need to wait for a DB to be ready
- Download configuration files
- Set permissions for volumes
- Run database migrations

StatefulSet

A StatefulSet is used for managing stateful applications, where each pod has a unique identity and stable storage.:

- Each pod has a stable hostname and persistent storage.
- Pods are created and terminated in a specific order.
- They are useful for databases, distributed systems, and applications that need stable network identities.

StatefulSet vs Deployment

	StatefulSet	Deployment
Pod Identity	Pods have stable, **unique names** (pod-0, pod-1, etc.)	Pods have **random names** (pod-xyz123
Scaling Order	Pods are created **sequentially** (pod-0 → pod-1)	All pods are created **at once**
Persistent Storage	Each pod gets a dedicated PersistentVolume (PV)	Pods share storage or use ephemeral storage
Network Identity	Each pod has a **stable hostname** (pod-0.service.cluster.local)	Pods use a **shared service**
Pod Replacement	A new pod **keeps the same hostname & storage**	A new pod gets a **new name and storage**
Use Case	Databases (MySQL, PostgreSQL), Kafka, Redis, Elasticsearch	Web apps, microservices, APIs, stateless workloads

devoteam

Thank **you.**