Kyma Runtime

Unit 1 – What is Kubernetes ?

- Open source system for automating deployment, scaling and management of containerized applications
- Can run almost anywhere
  - Public cloud
  - Private cloud
  - On-Premise
- All major cloud providers offer Kubernetes as a managed service
- SAP BTP Kyma runtime is a managed Kubernetes service

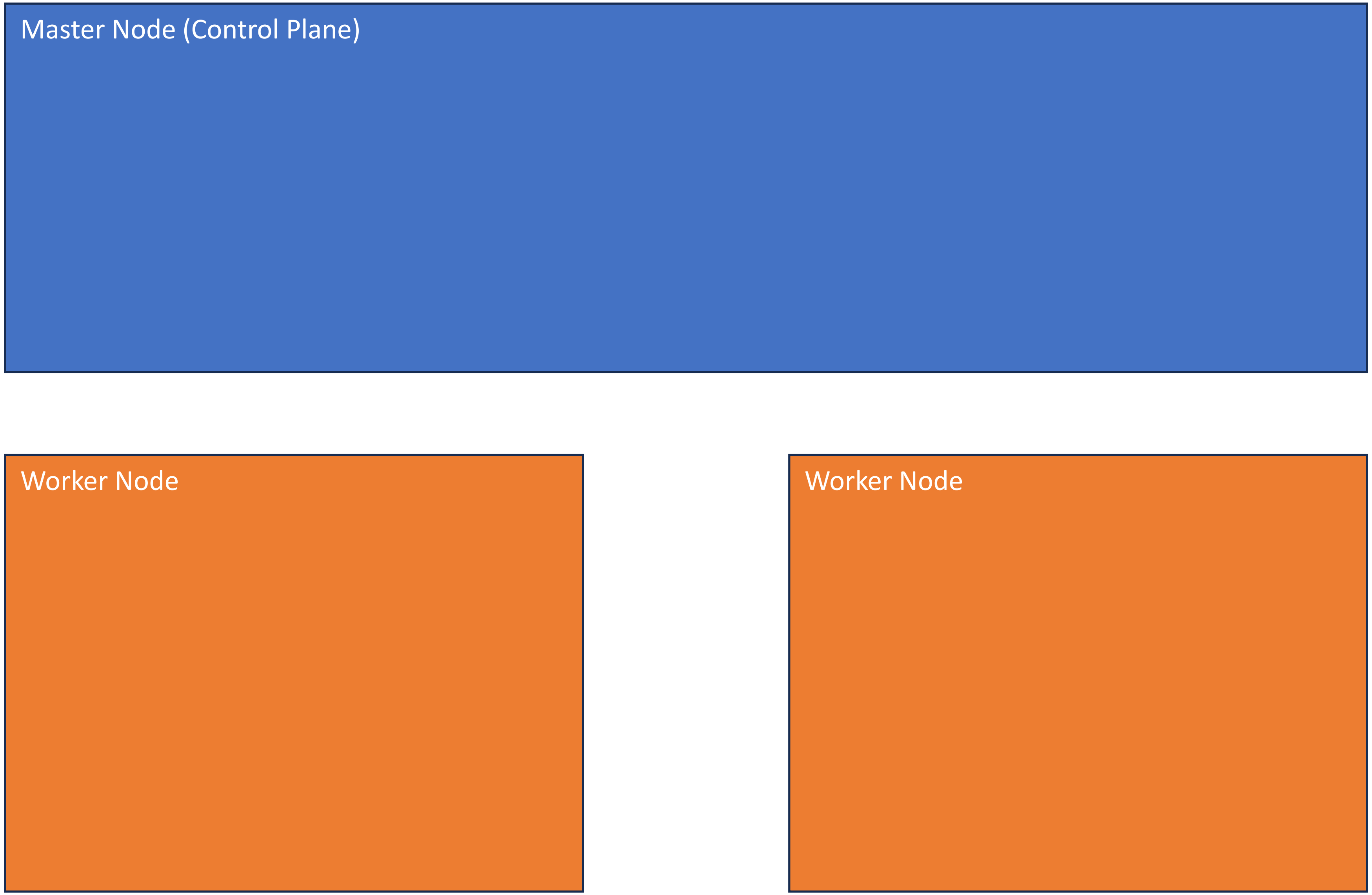
Unit 1 – Kubernetes Cluster

Cluster is a set of machines (physical or virtual)

Each k8s cluster has a master node and one or more worker nodes

Master node is responsible for managing the cluster

Worker nodes are responsible for running the cluster



Master Node (Control Plane)

Worker Node

Worker Node

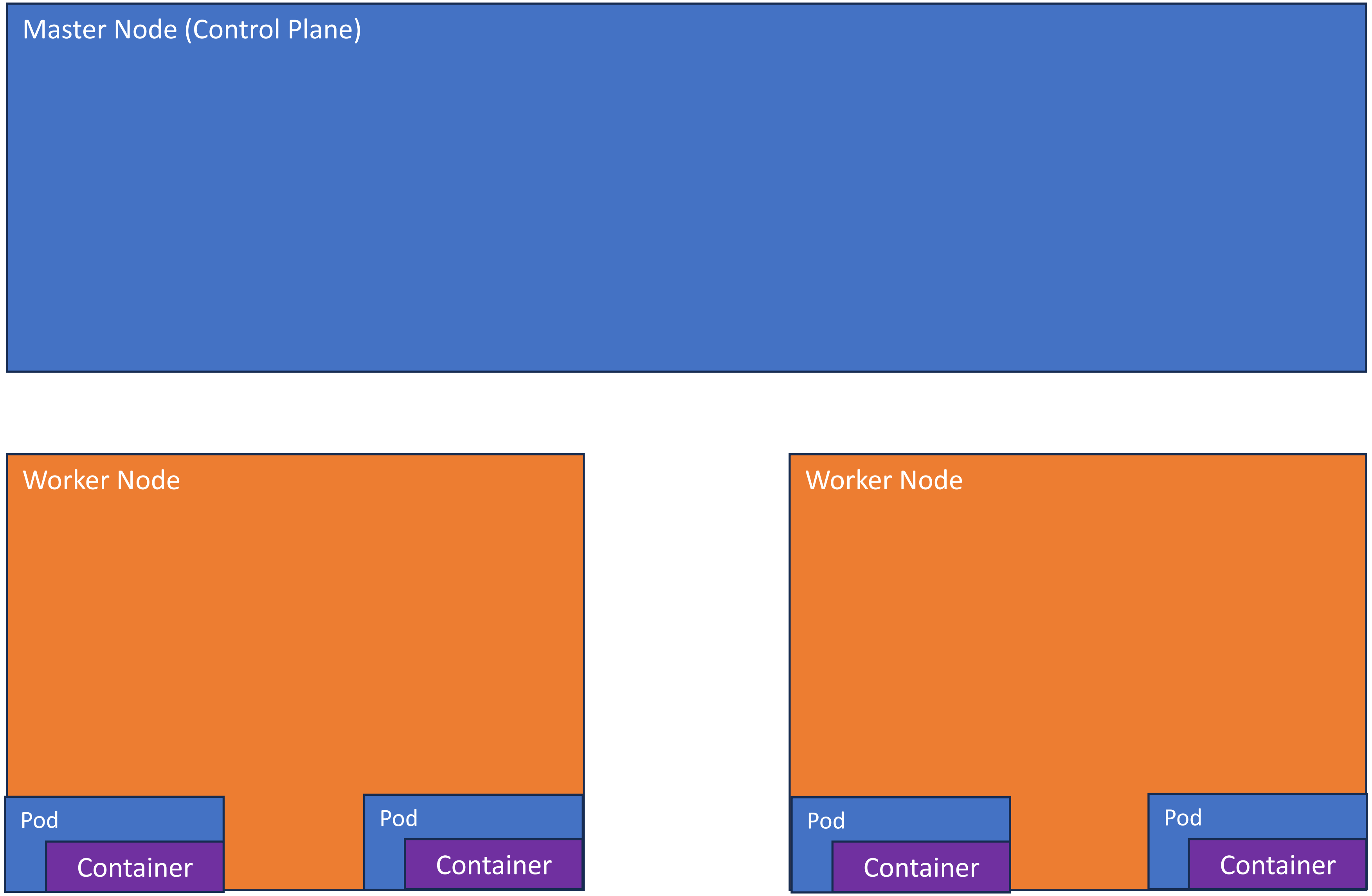
Unit 1 – Kubernetes Cluster

Pods – smallest deployable units in k8s

Thin wrapper for containers

Usually 1 application per Pod

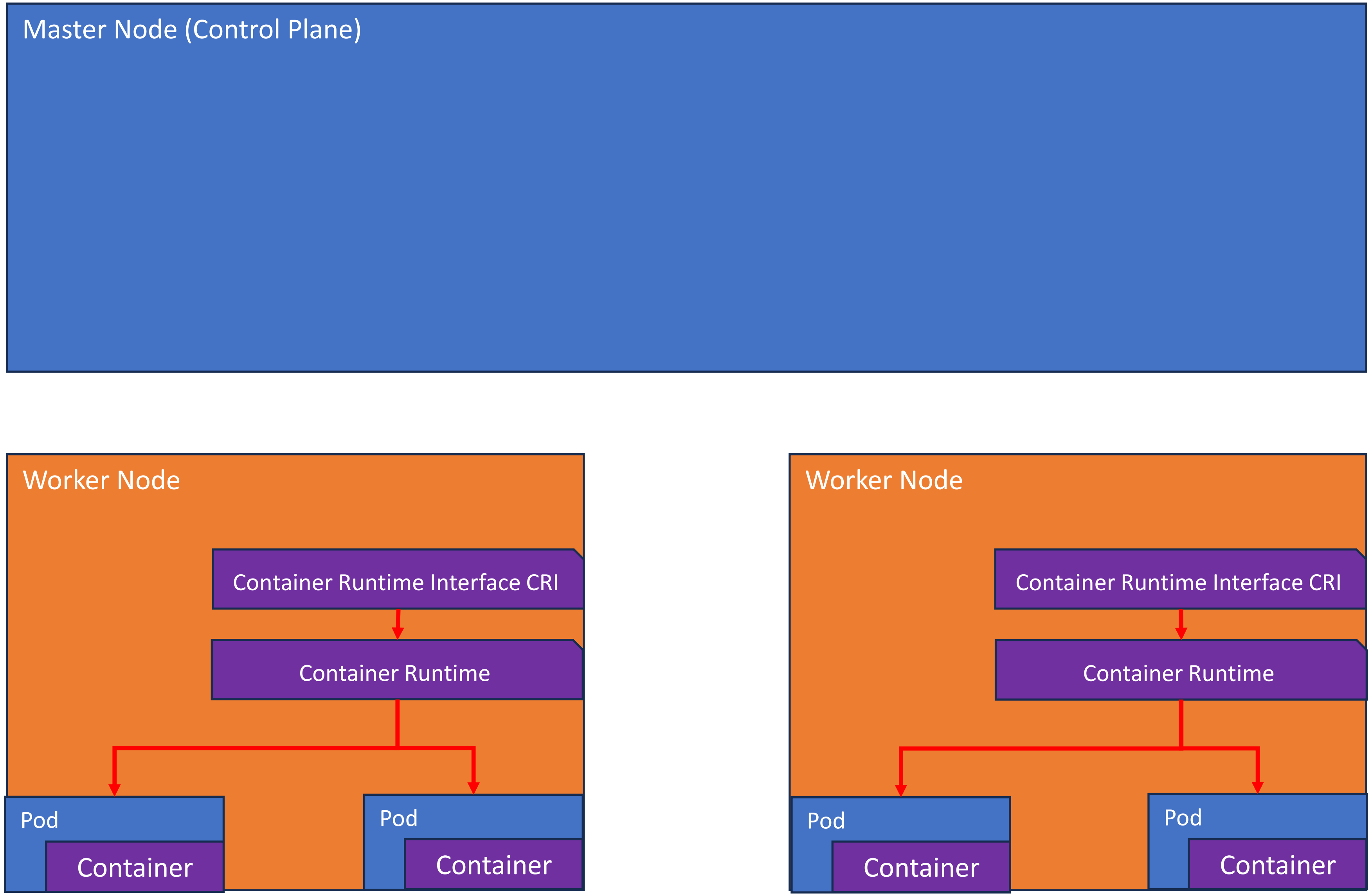
Pods are ephemeral – can die easily



Unit 1 – Kubernetes Cluster

K8s does not want to be tied to a single container technology (for example, Docker)

K8s allows you to use any container runtime that implements this Container Runtime Interface

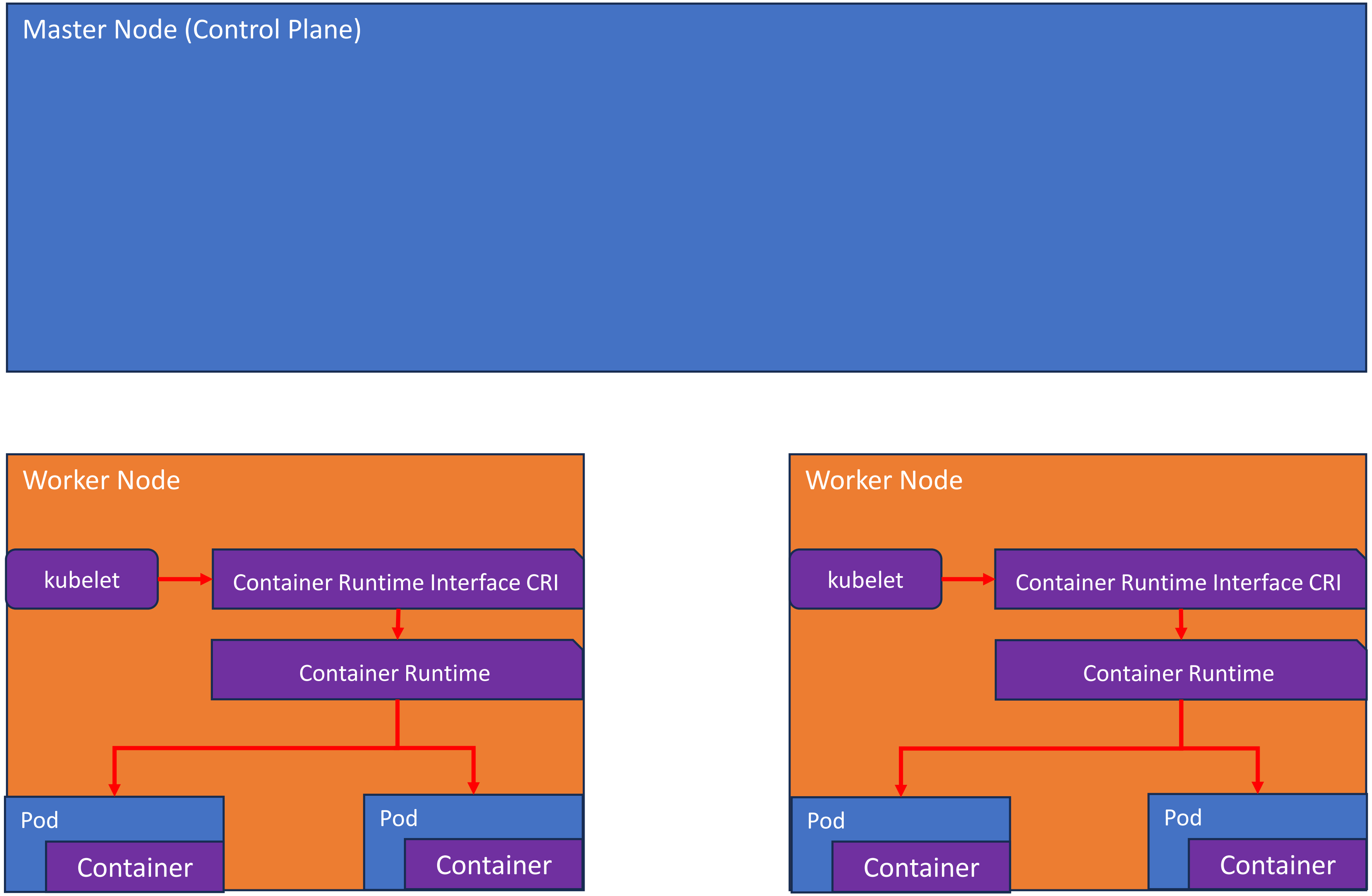


Unit 1 – Kubernetes Cluster

Each worker node has a kubelet process running on it

Ensures containers are running in a pod based on their specs

Communicates with Master Node to receive new work assignments

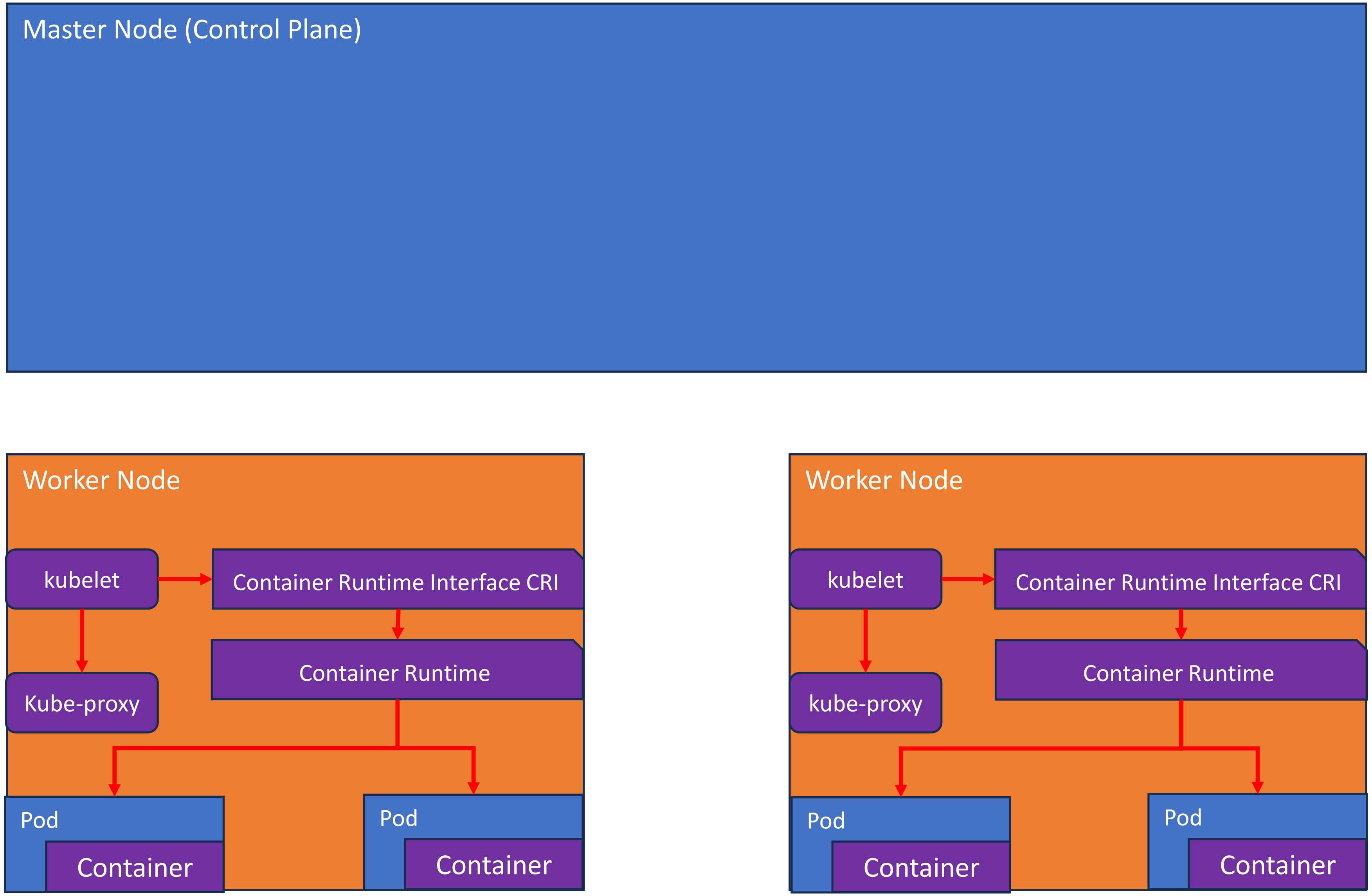


Unit 1 – Kubernetes Cluster

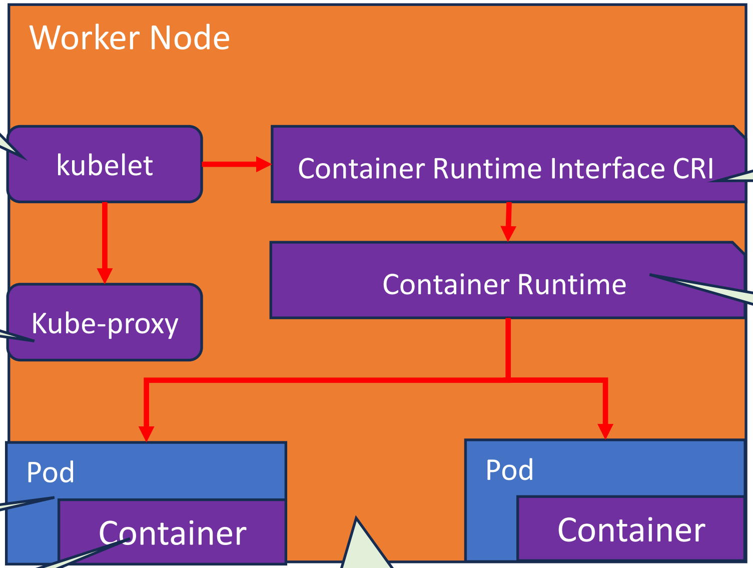
kube-proxy is a network proxy that runs on each node

Main job is to maintain network rules on nodes

Allows for internal and external cluster communication with pods



Kubernetes API via kubectl



Ensures containers are running based on spec
Talks to Master Node to receive new work

Network proxy that runs on each node
Maintains network rules on nodes

Smallest deployable unit
Wrapper for containers

Container can be any technology – e.g. Docker

Std. interface between kubelet and container runtime
Supported Container Runtime implements this interface

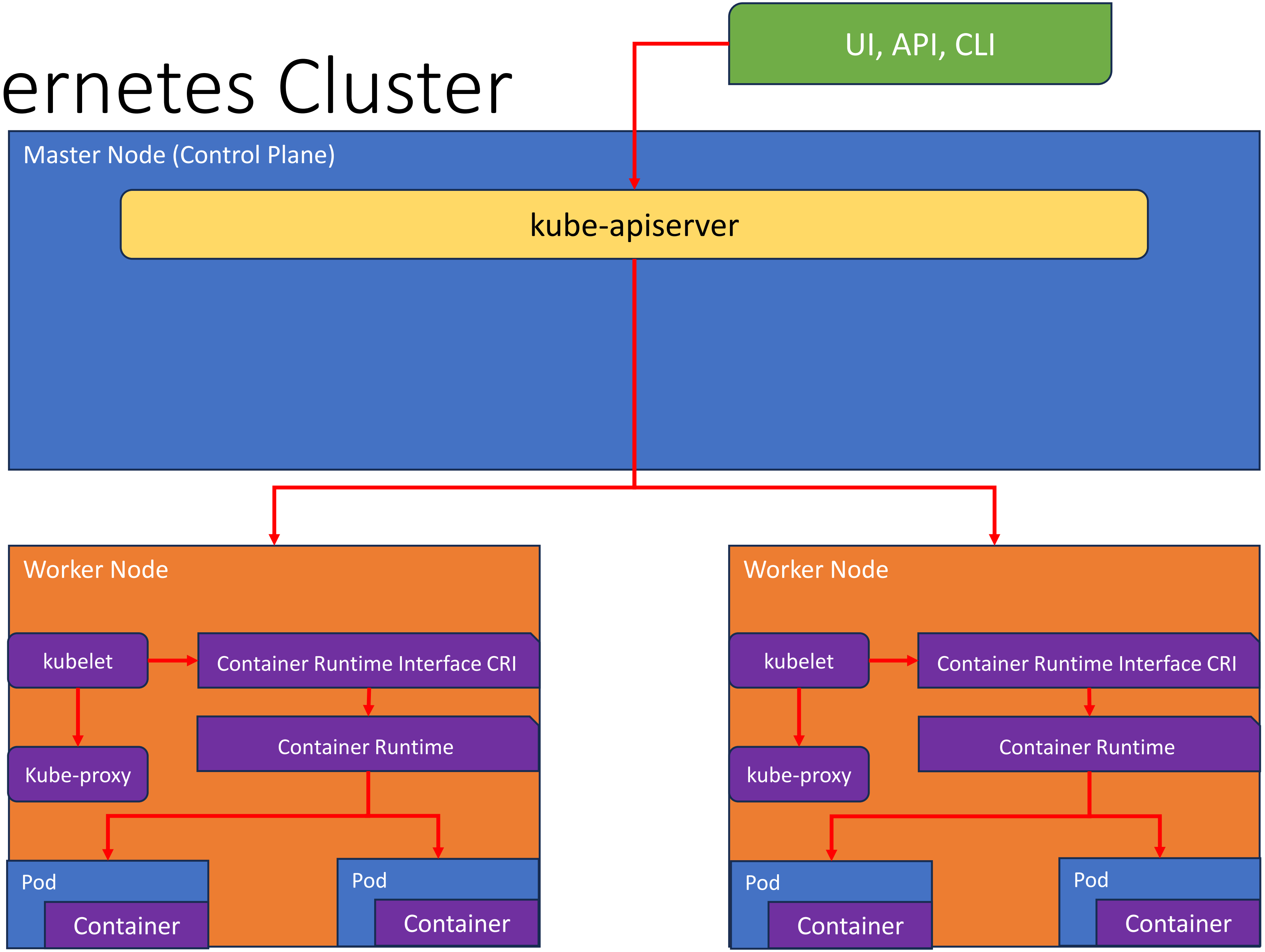
K8s supports multiple runtimes – e.g. Docker Runtime
Runtime is responsible for running your container

Worker Node is responsible for running the applications…
For this reason, worker nodes need to be powerful machines

Unit 1 – Kubernetes Cluster

API server – entry point to the k8s cluster

Exposes k8s API as RESTful API – Can be accessed by UI, CLI (kubectl) to control the cluster



Kubernetes API via kubectl

Kubectl syntax is as follows

kubectl [command] [TYPE] [Name] [flags]

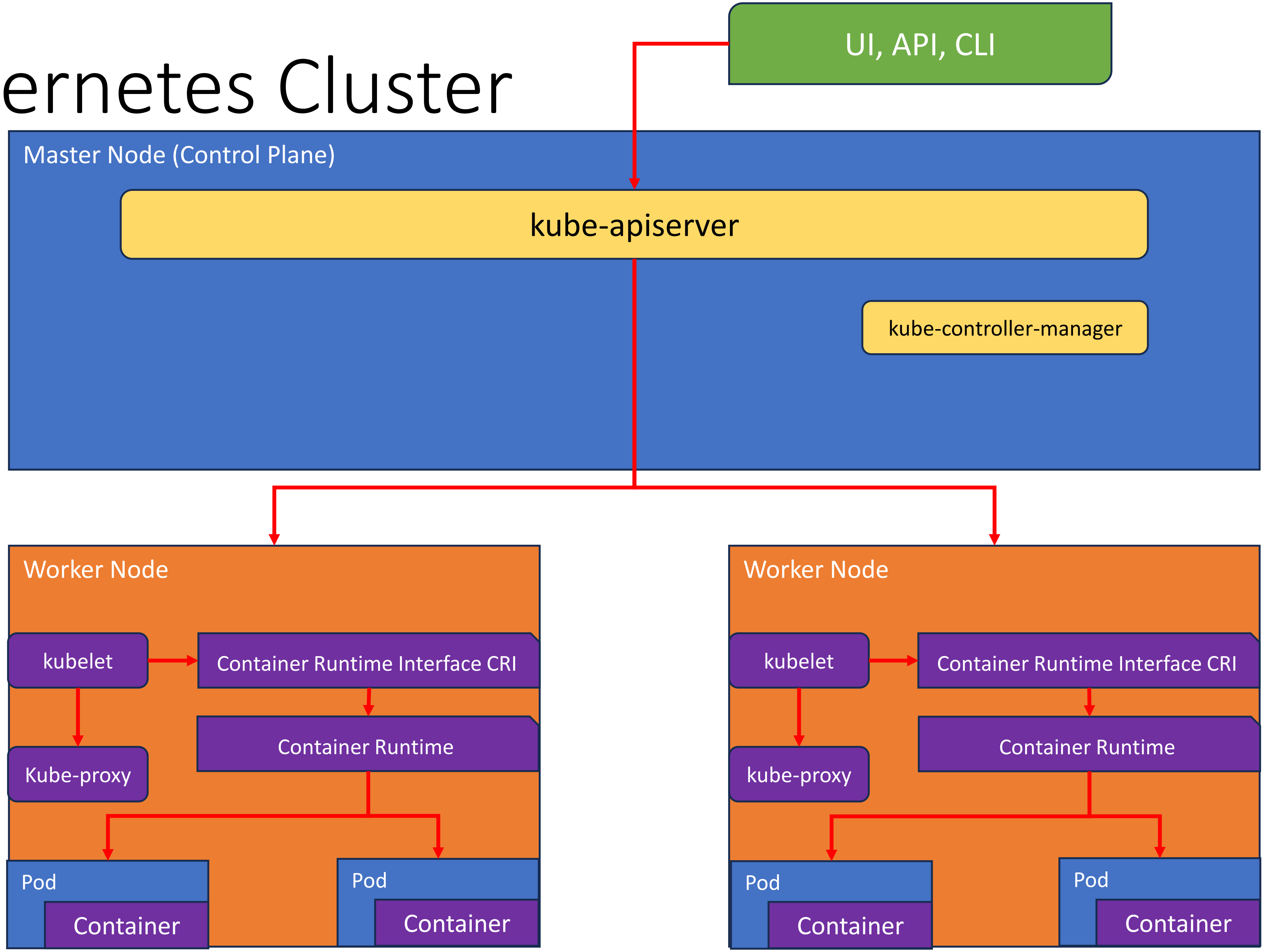
kubectl get pods
kubectl get pods --namespace my-namespace
kubectl get services
kubectl get replicasets

Unit 1 – Kubernetes Cluster

Controller-manager

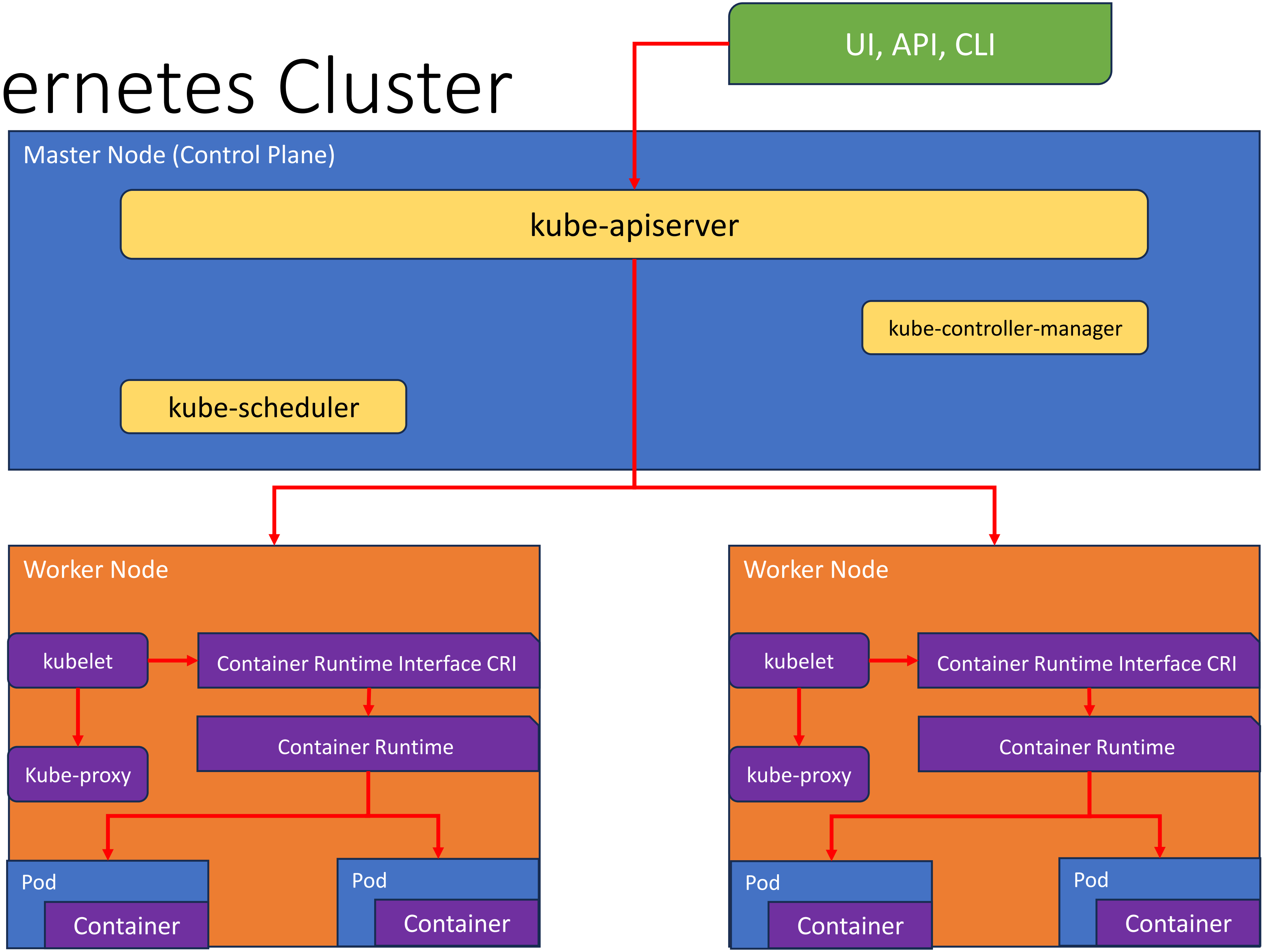
Responsible for noticing and responding when nodes go down

Responsible for maintaining the correct number of pods



Unit 1 – Kubernetes Cluster

Responsible for scheduling pods on worker nodes



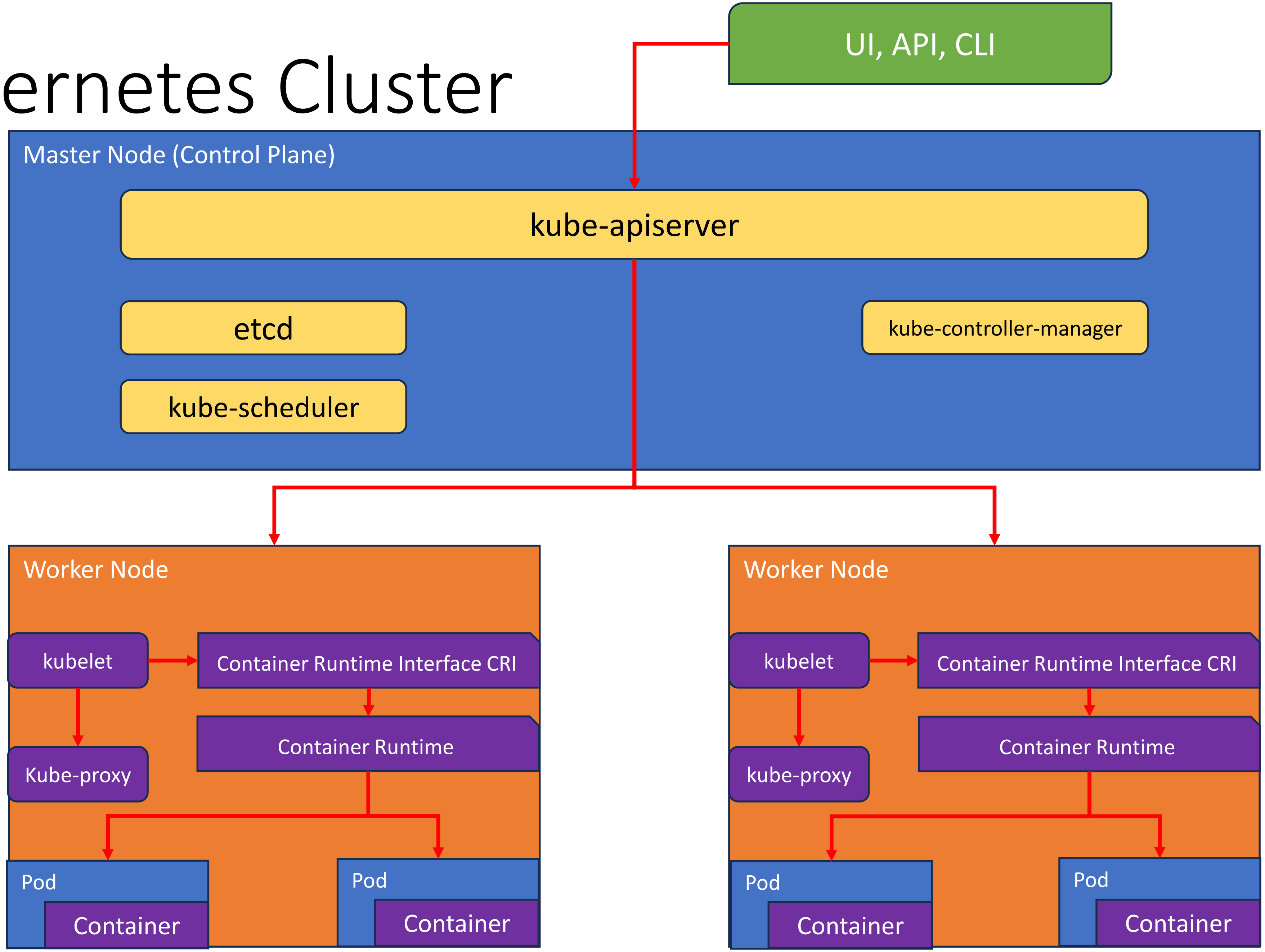
Unit 1 – Kubernetes Cluster

etcd is a key-value storage

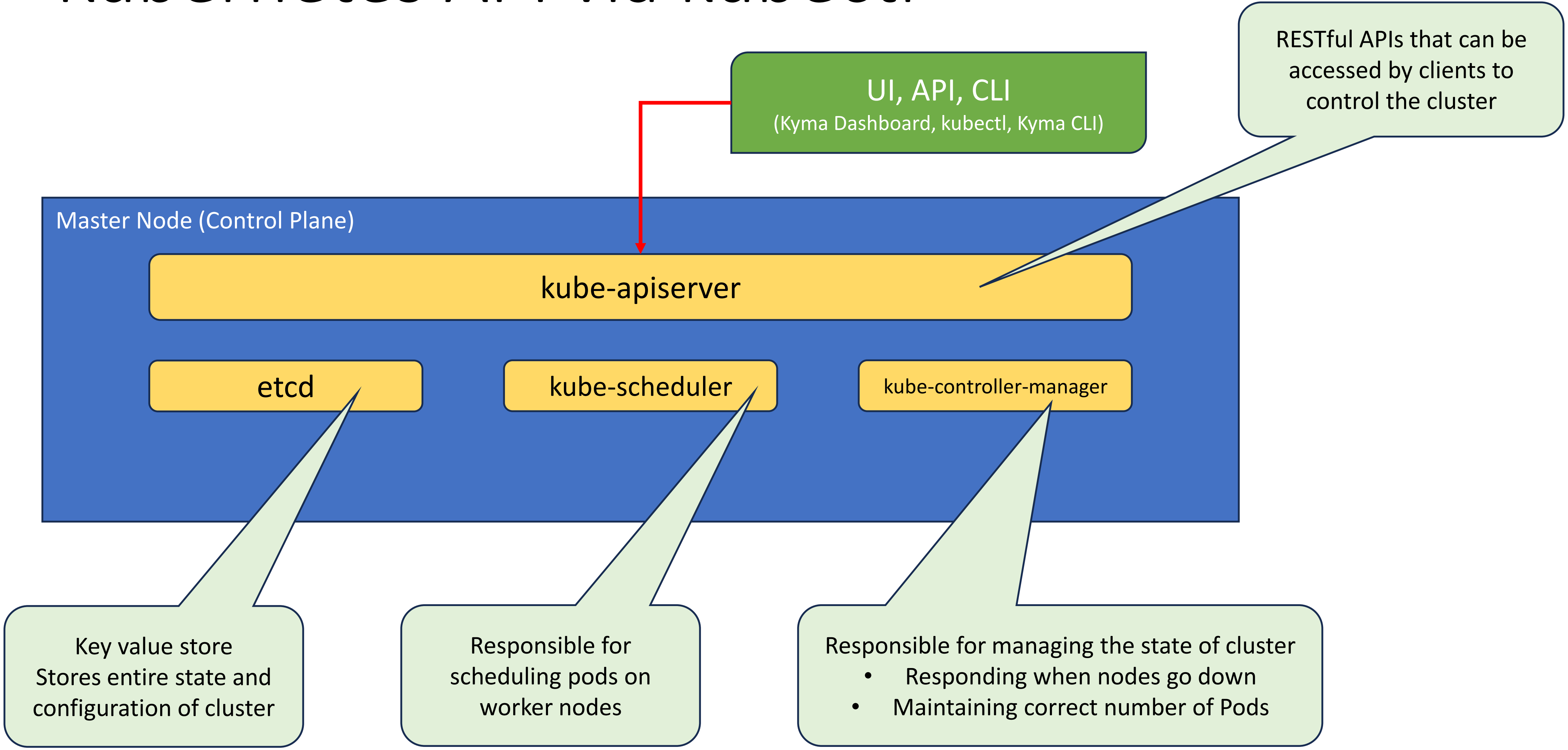
Contains configuration data. Holds the current info of k8s cluster

Master node is the heart of the k8s cluster

Can be used for backup and restore based on etcd snapshot



Kubernetes API via kubectl



UI, API, CLI

(Kyma Dashboard, kubectl, Kyma CLI)

RESTful APIs that can be accessed by clients to control the cluster

Master Node (Control Plane)

kube-apiserver

etcd

kube-scheduler

kube-controller-manager

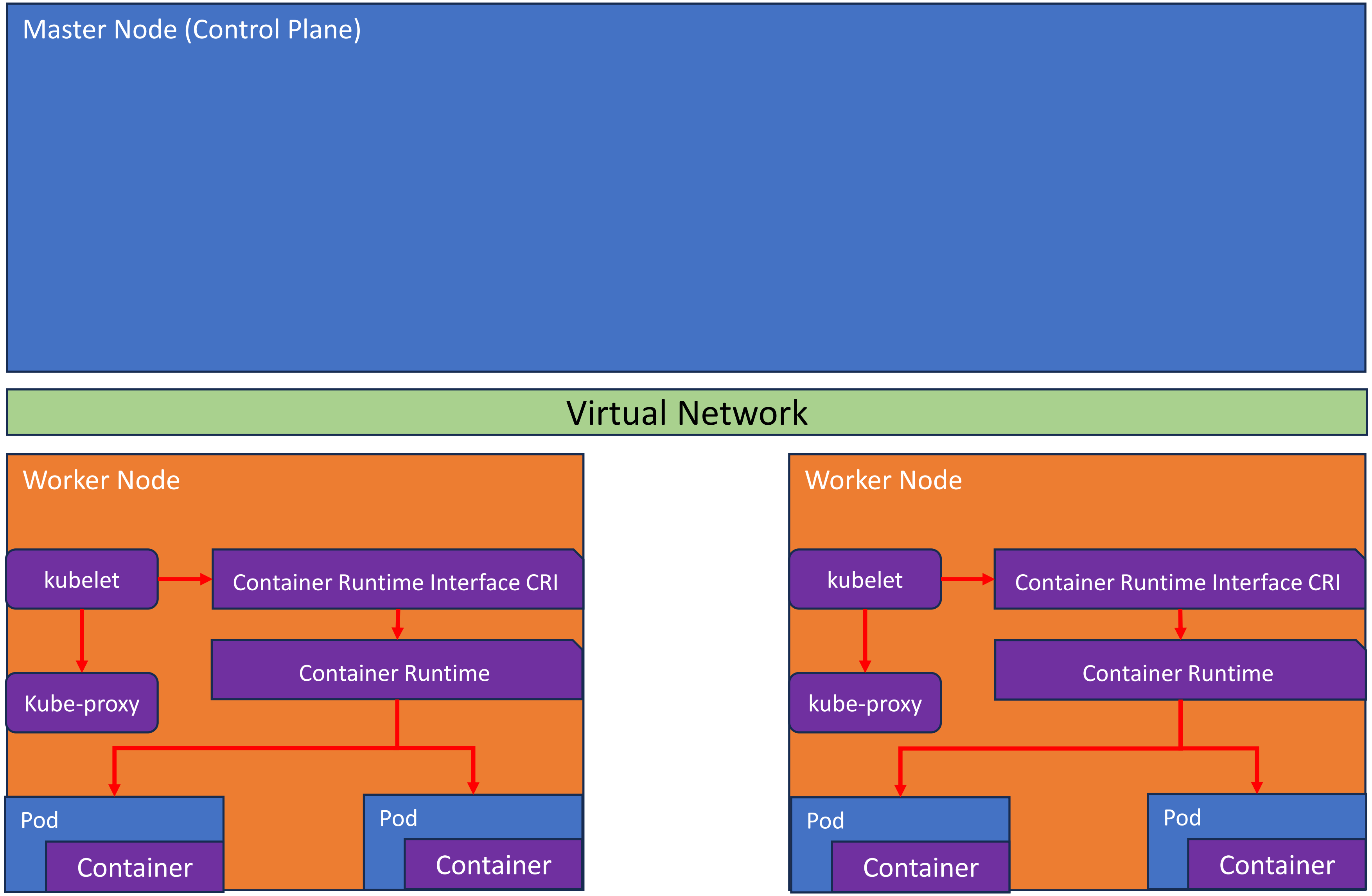
Key value store
Stores entire state and configuration of cluster

Responsible for scheduling pods on worker nodes

Responsible for managing the state of cluster

- Responding when nodes go down
- Maintaining correct number of Pods

Unit 1 – Kubernetes Cluster

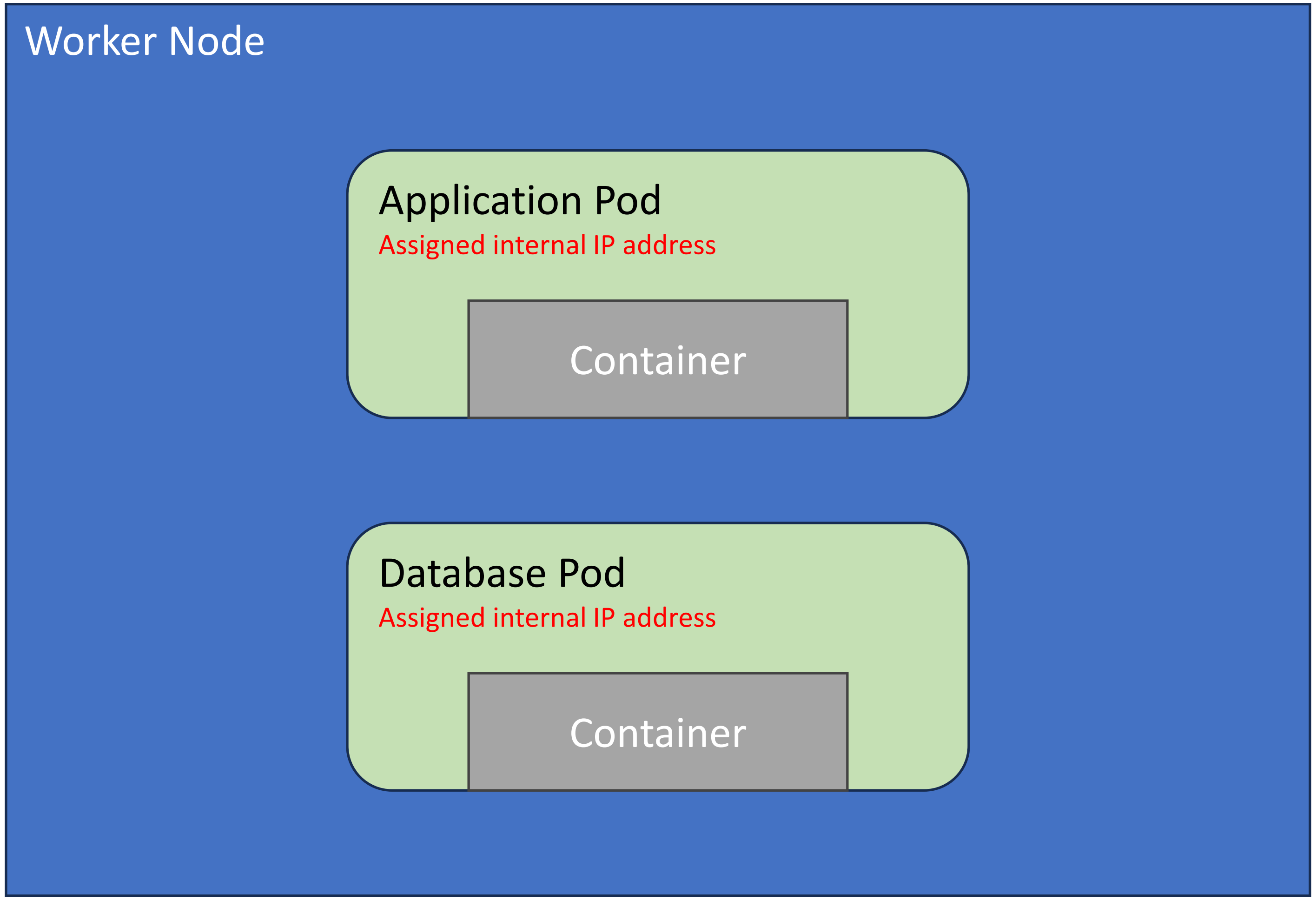


Unit 1 – Kubernetes Components

Each Pod is assigned an internal IP address

This allows Application Pod and Database Pod to talk to each other

Pods are ephemeral – can die easily



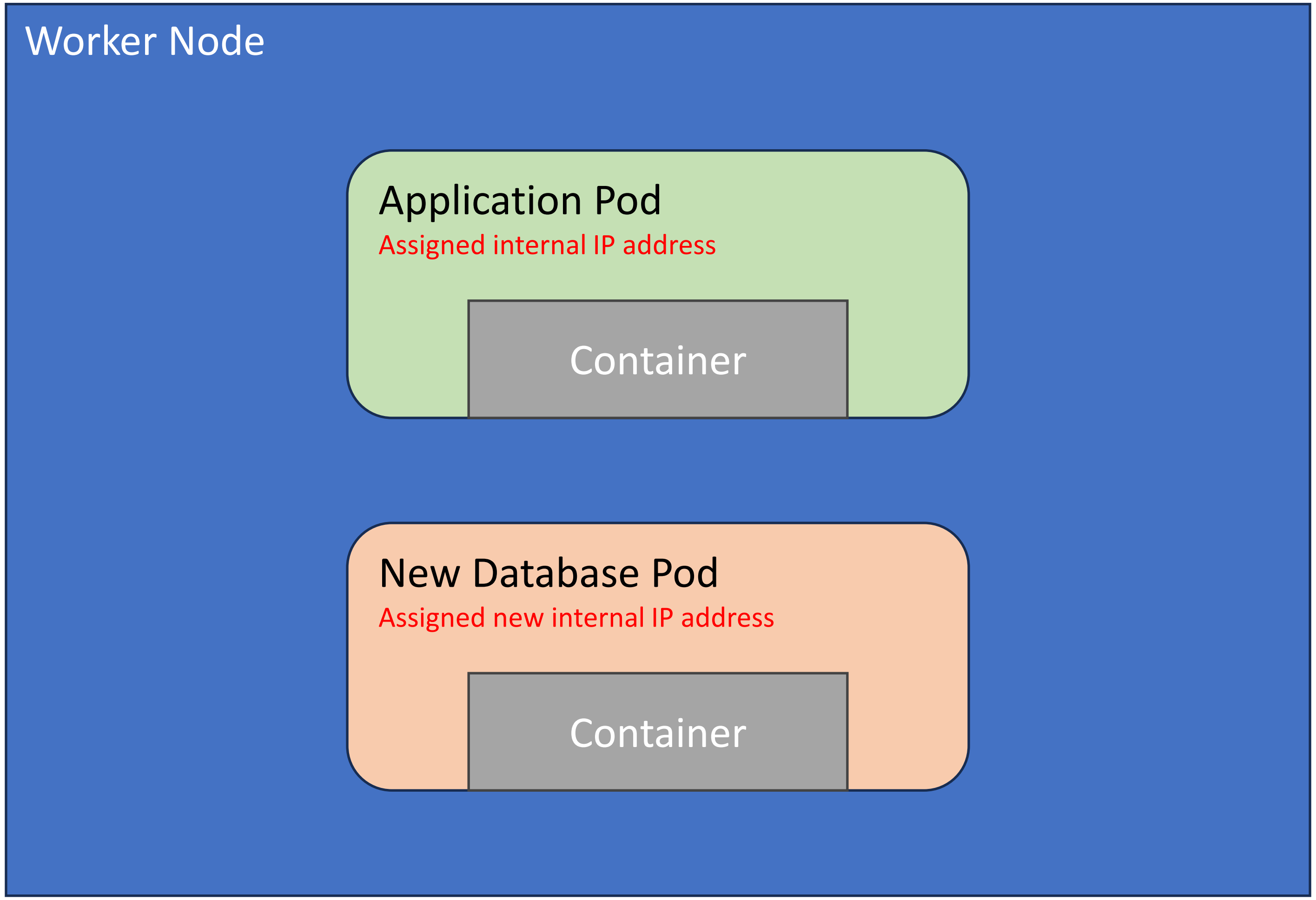
Unit 1 – Kubernetes Components

When a Pod dies, k8s will create a new Pod in the cluster

New Pod is assigned a new internal IP address

Oh no !! Application Pod cannot talk to the New Database Pod

Service component to the rescue

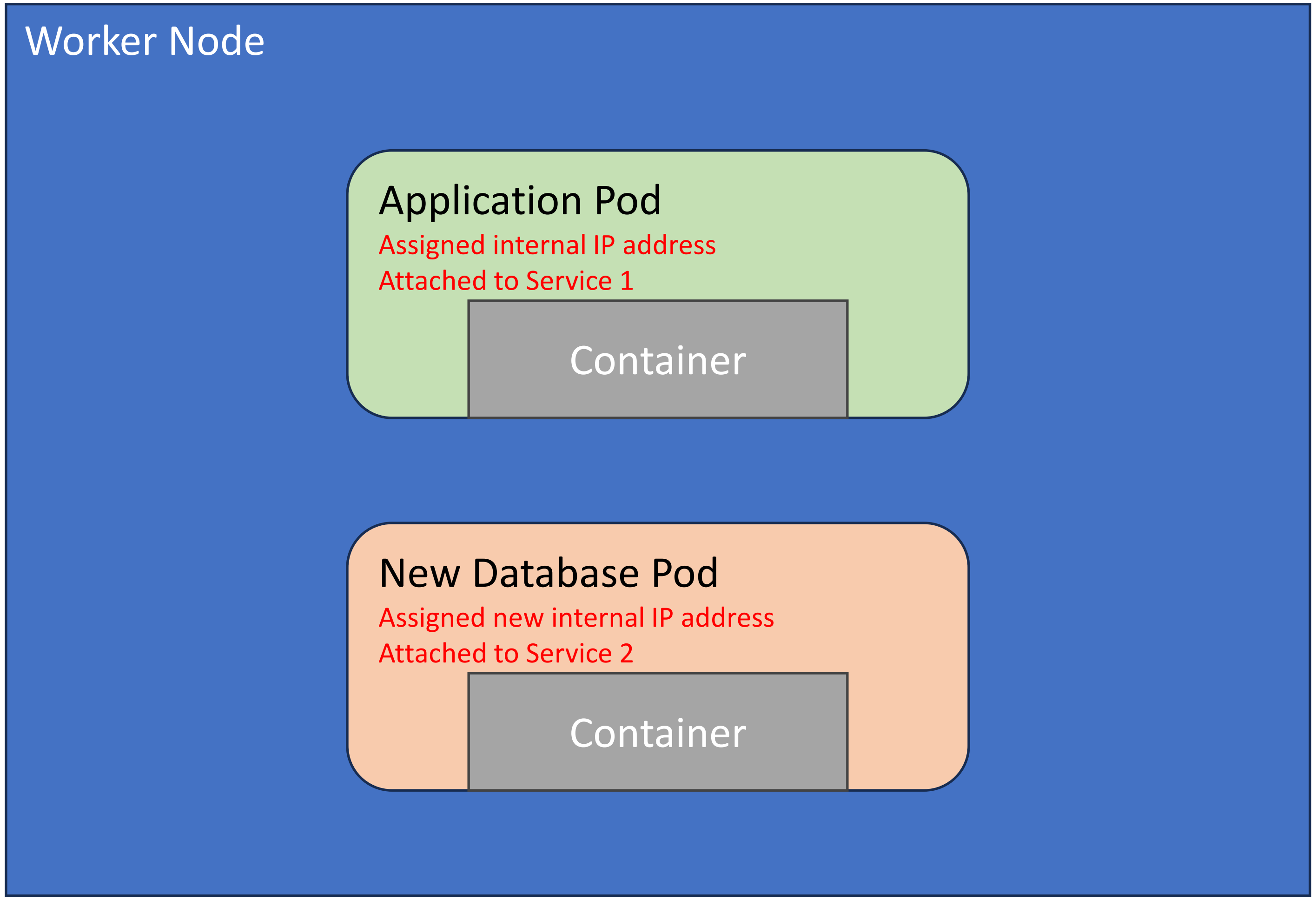

Worker Node
Application Pod
Assigned internal IP address
Container
New Database Pod
Assigned new internal IP address
Container

Unit 1 – Kubernetes Components

Service is a permanent IP address attached to each Pod – well, not too attached though

When the Pod dies, the Service does not die along with it. The lifecycle of the Service is independent of the lifecycle of the Pod

So now the Application Pod can communicate to the New Database Pod using Service (Permanent IP address)



Unit 1 – Kubernetes Components

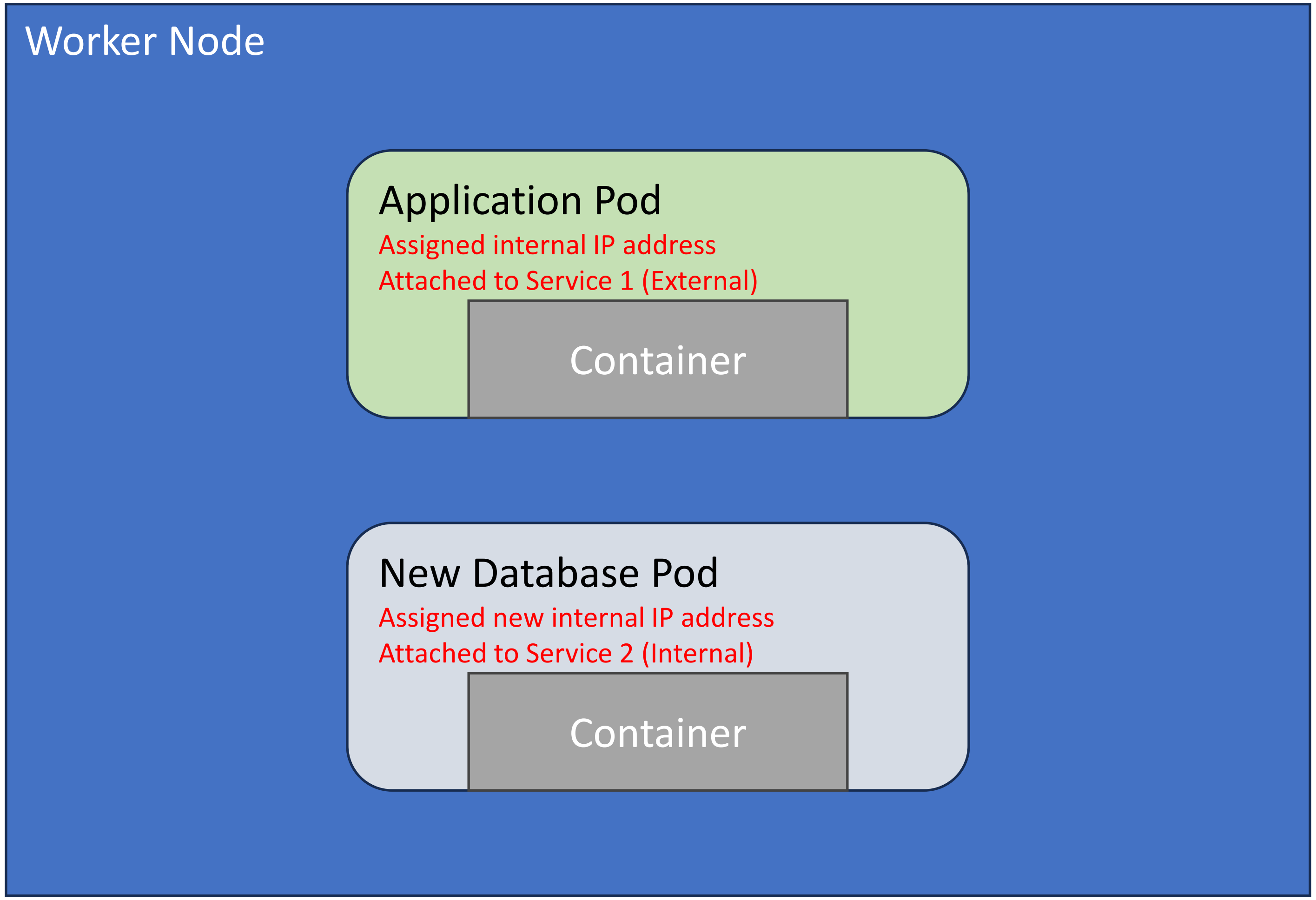
Ok, this is all good. But how do I access this Application Pod from the outside world

You can create either an External Service or an Internal Service

Application Pod ➔ External Service
New Database Pod ➔ Internal Service

But I don't want to use an IP address to connect… I want a friendly name !!

Ingress component to the rescue

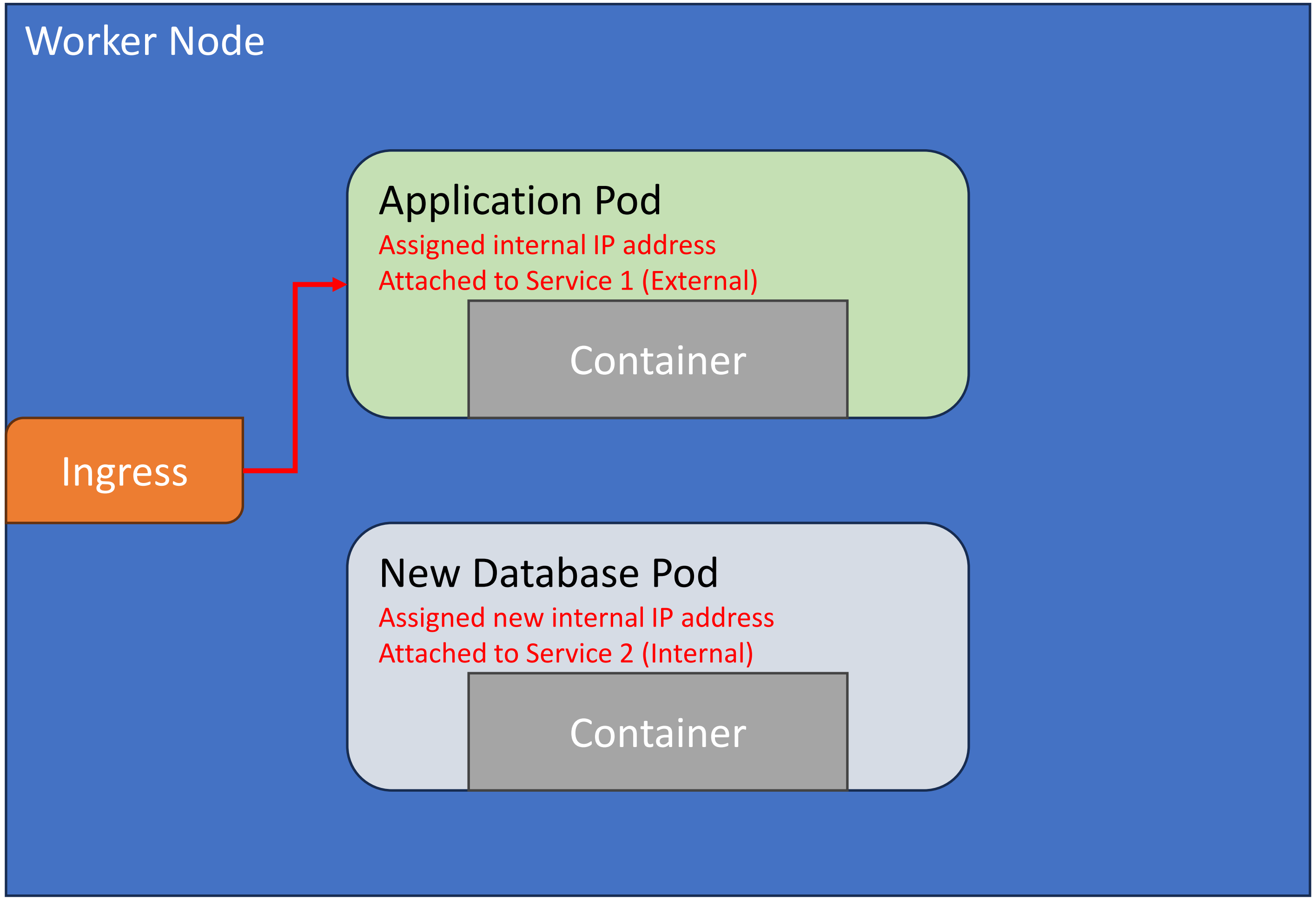


Worker Node
Application Pod
Assigned internal IP address
Attached to Service 1 (External)
Container
New Database Pod
Assigned new internal IP address
Attached to Service 2 (Internal)
Container

Unit 1 – Kubernetes Components

So the request first goes to Ingress

Ingress routes the request to the right Service (in this case, Service 1 of the Application Pod)

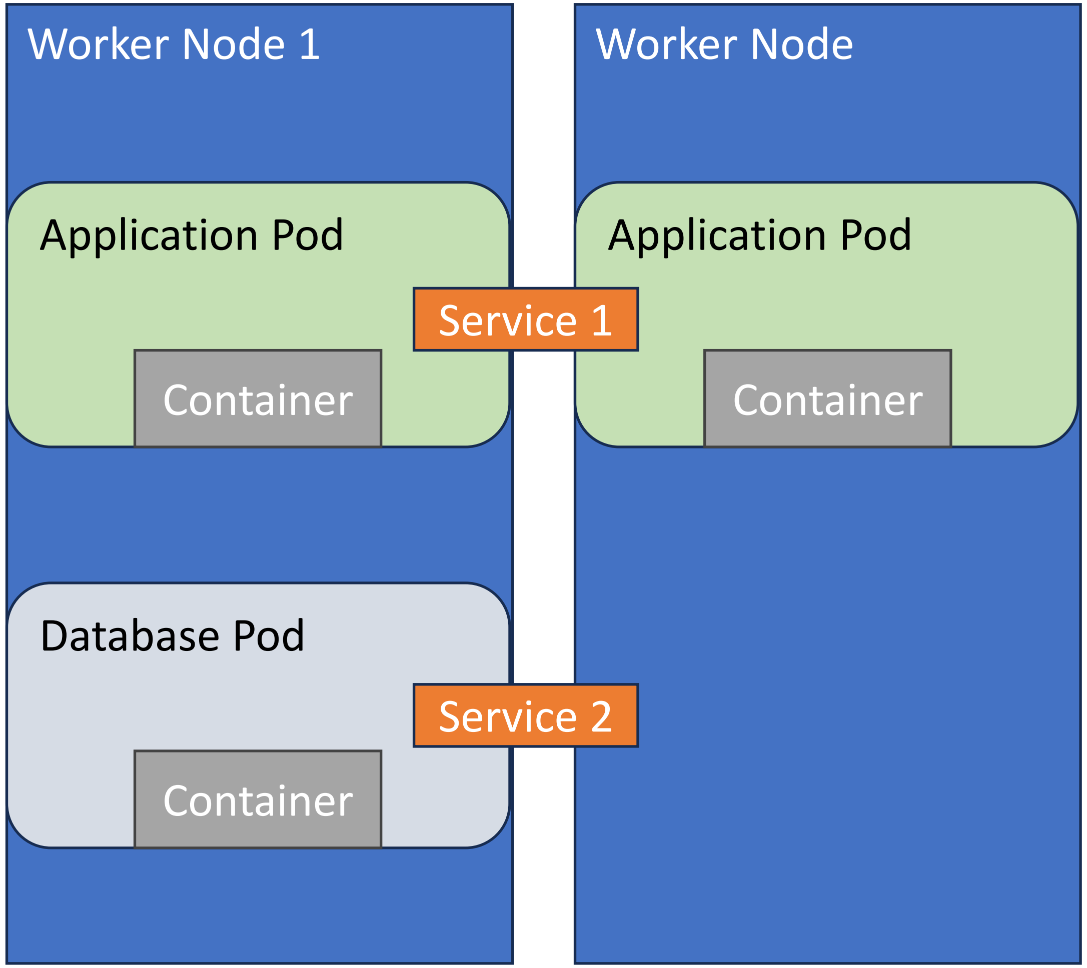

Worker Node
Application Pod
Assigned internal IP address
Attached to Service 1 (External)
Container
Ingress
New Database Pod
Assigned new internal IP address
Attached to Service 2 (Internal)
Container

Unit 1 – Kubernetes Components

For high availability, we typically have more than 1 Application Pod

Do we need to define multiple Application Pods in the code ? No
You only have a single definition of Application Pod
But you mention that you want x number of copies of that

ReplicaSets define how many Pods we need in the cluster



Unit 1 – Kubernetes Components

For high availability, we typically have more than 1 Application Pod

Do we need to define multiple Application Pods in the code ? No

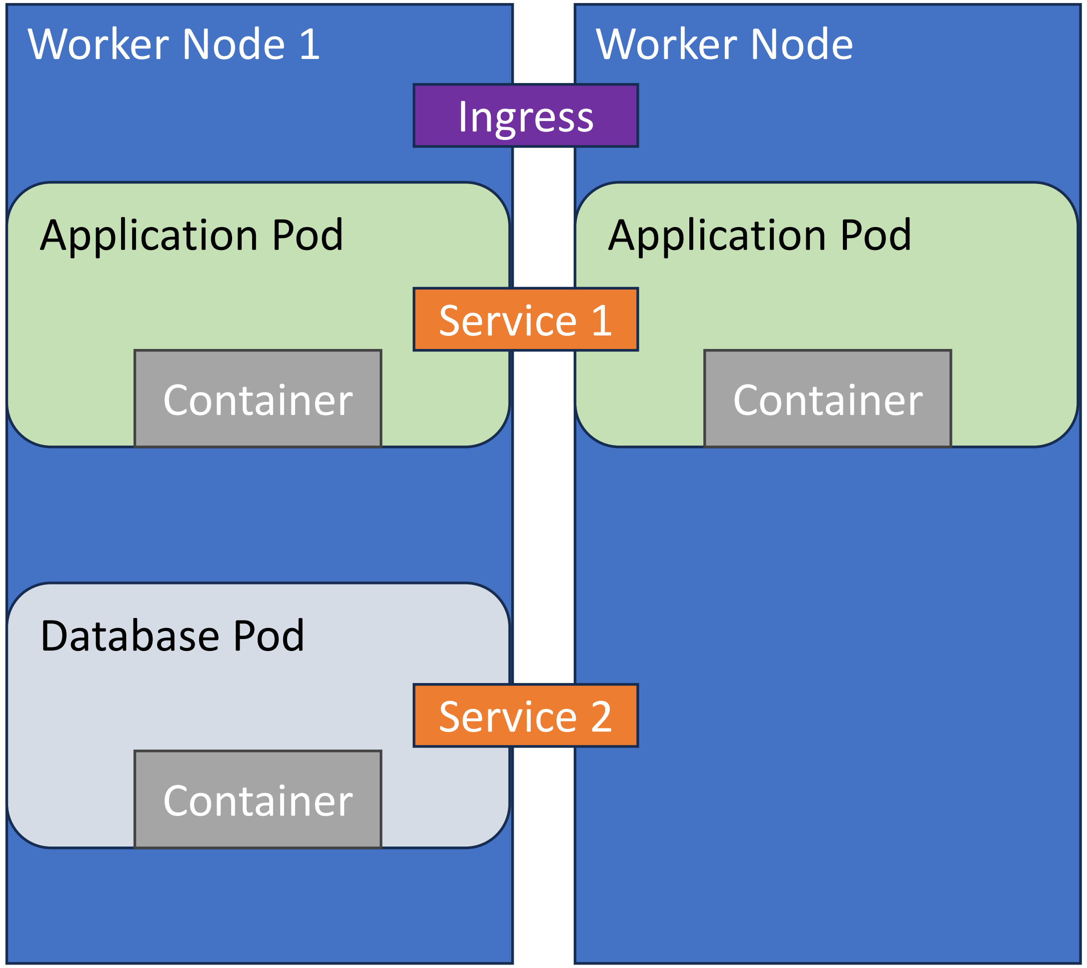
ReplicaSets define how many Pods we need in the cluster

Service can also act as a load balancer

Deployments specify how many replicas you need

Layers of abstraction

- Deployment manages a ReplicaSet
- ReplicaSet manages a Pod
- Pod is an abstraction of a Container



Unit 2 – Discovering Kyma

- Kyma is an open-source project
- Makes it easier to develop and run applications on Kubernetes
- Features provided by Kyma (extending functionality of k8s)
  - Serverless
  - Eventing
  - Observability
  - Service Integrations
  - Service Mesh
  - API Gateway

Unit 2 – Benefits of SAP BTP Kyma Runtime

- Fully managed Kubernetes cluster
- Latest features of Kubernetes and its ecosystem
- Additional features through Kyma (Serverless, Eventing etc.)
- Integration with other SAP services and products
- Elimination of operations’ overhead

Enabling SAP BTP, Kyma Runtime

Kyma Dashboard

Kubectl for Kyma

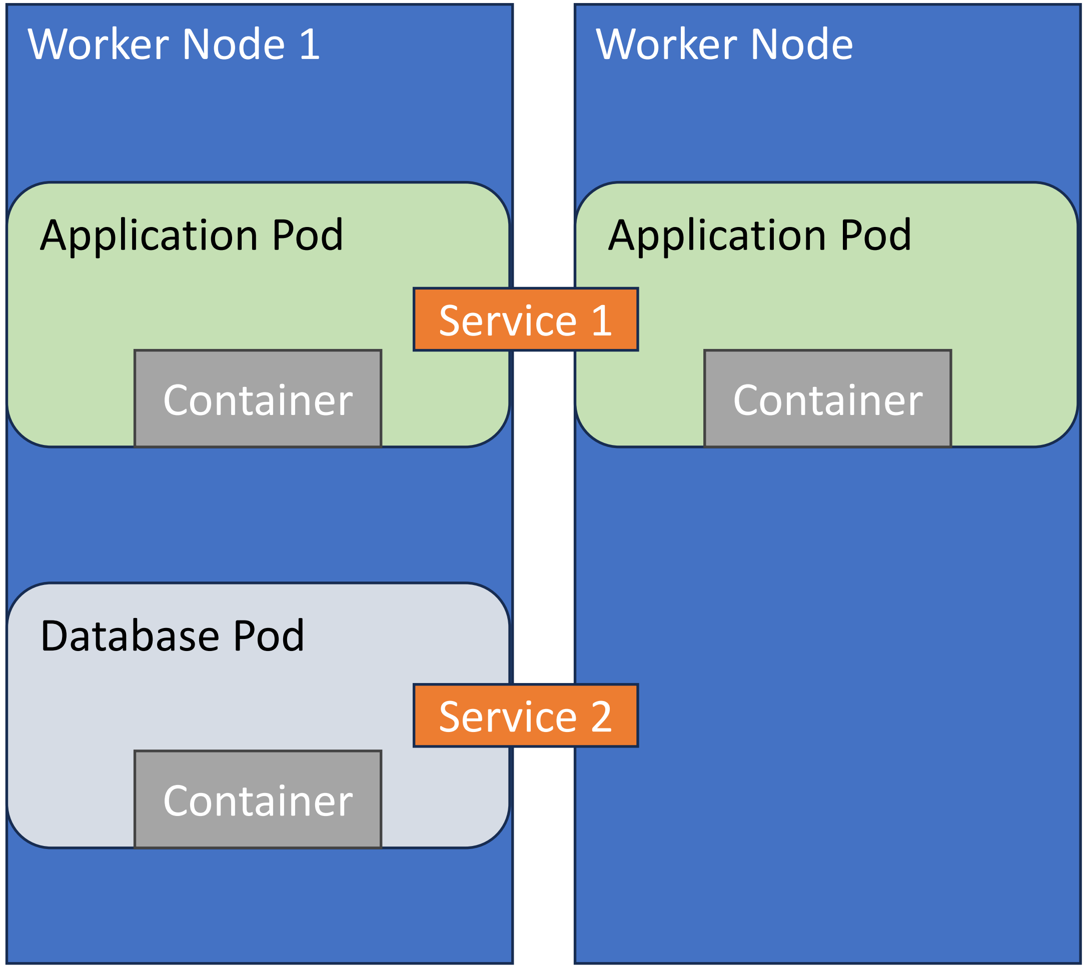
Unit 3 – Discovering Kubernetes Workloads

Deployment

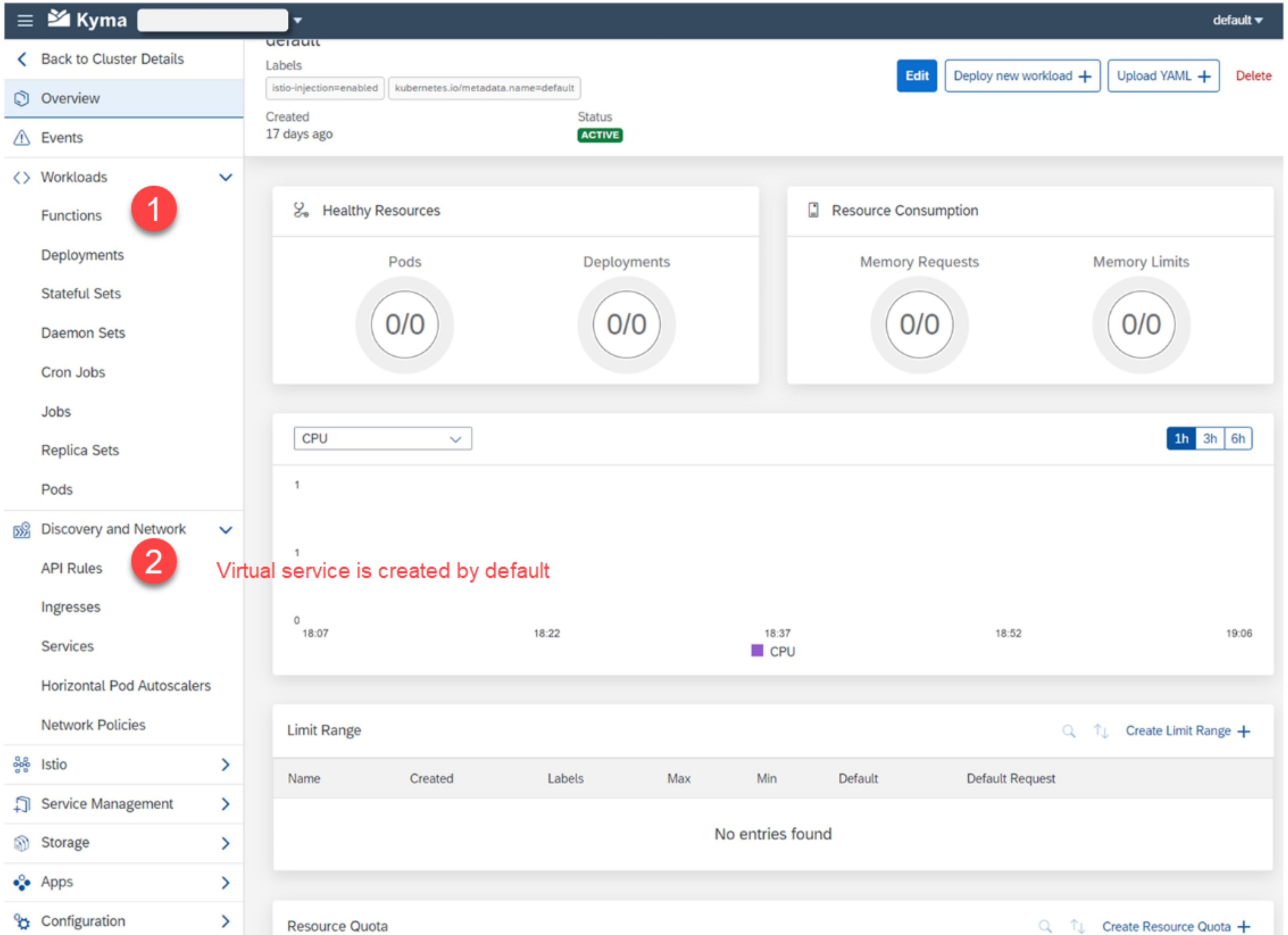
- Declaratively define the blueprint for Application Pod
- How many copies
- You would mostly work with deployments

StatefulSet

- Database cannot be replicated via deployment
- StatefulSet used to manage stateful applications (Database)
- To prevent any inconsistencies
- More complex than working with deployments



Serverless Functions



default

Labels

istio-injection=enabled kubernetes.io/metadata.name=default

Created: 17 days ago

Status: ACTIVE

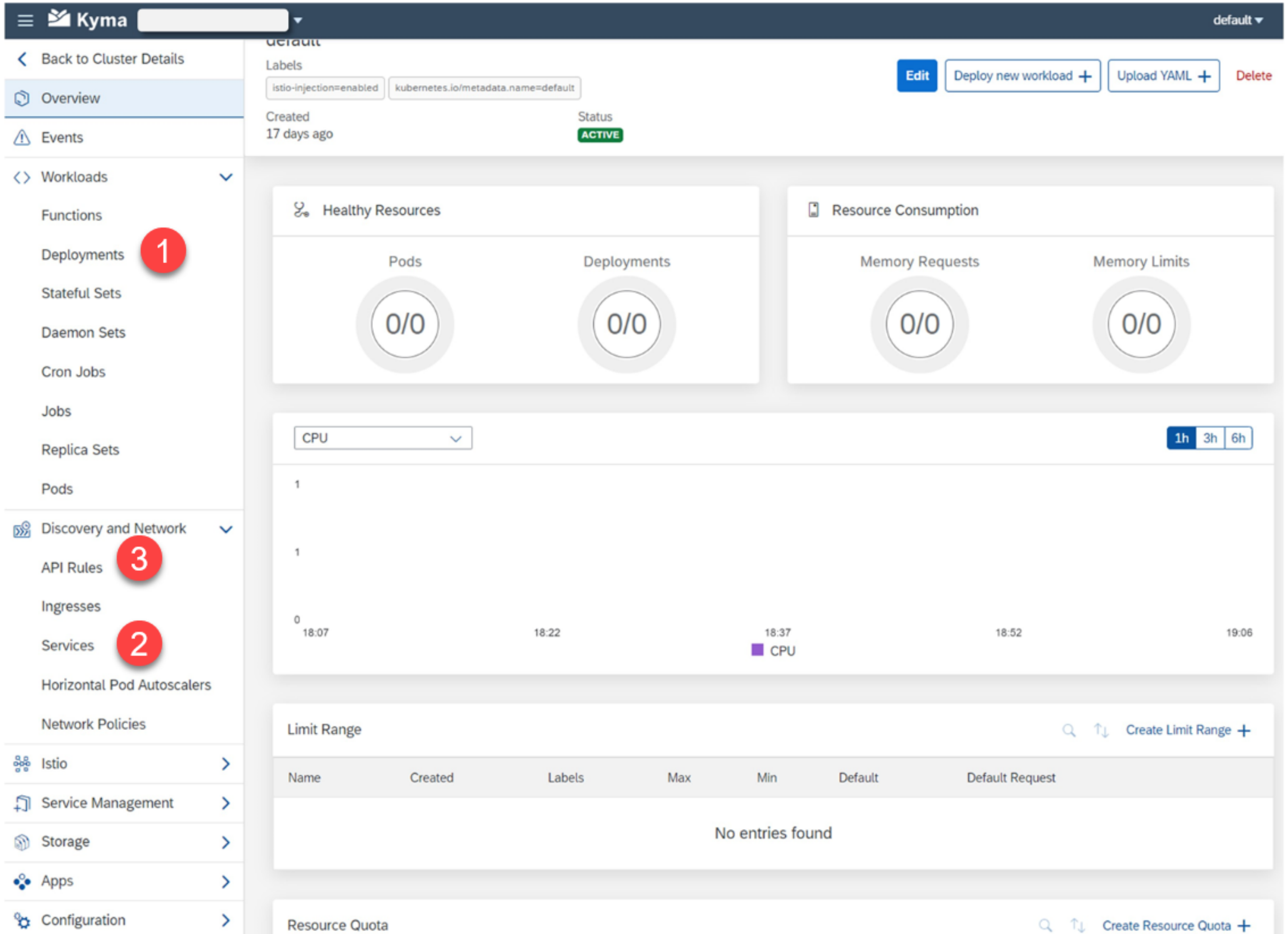
Name	Created	Labels	Max	Min	Default	Default Request

No entries found

Virtual service is created by default

Serverless Functions

Deployments



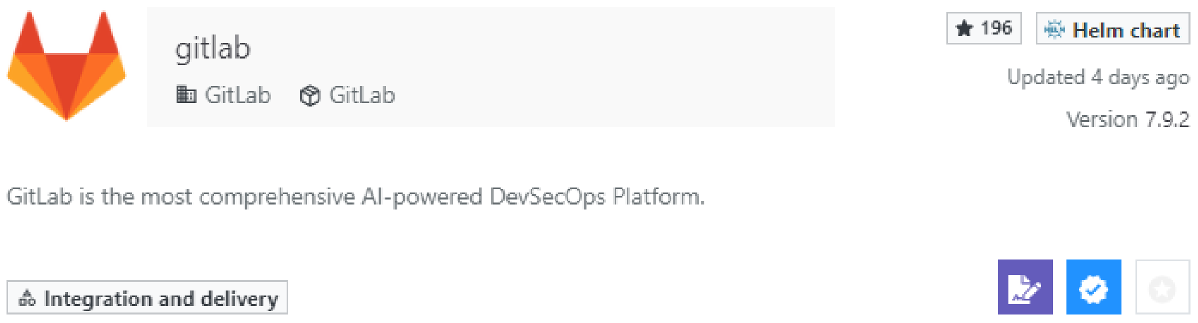
Name	Created	Labels	Max	Min	Default	Default Request

No entries found

Creating and Managing a Deployment

Unit 3 – What is Helm ?

- Package manager for k8s
- Let’s say you want to deploy gitlab
  - Might need the following…
    - Service to expose the endpoint
    - Persistent volume
    - Secret, ConfigMaps etc.
- Helm makes it super easy
  - *helm install gitlab*
  - *helm upgrade gitlab*
  - *helm uninstall gitlab*



gitlab

GitLab GitLab

★ 196 Helm chart

Updated 4 days ago

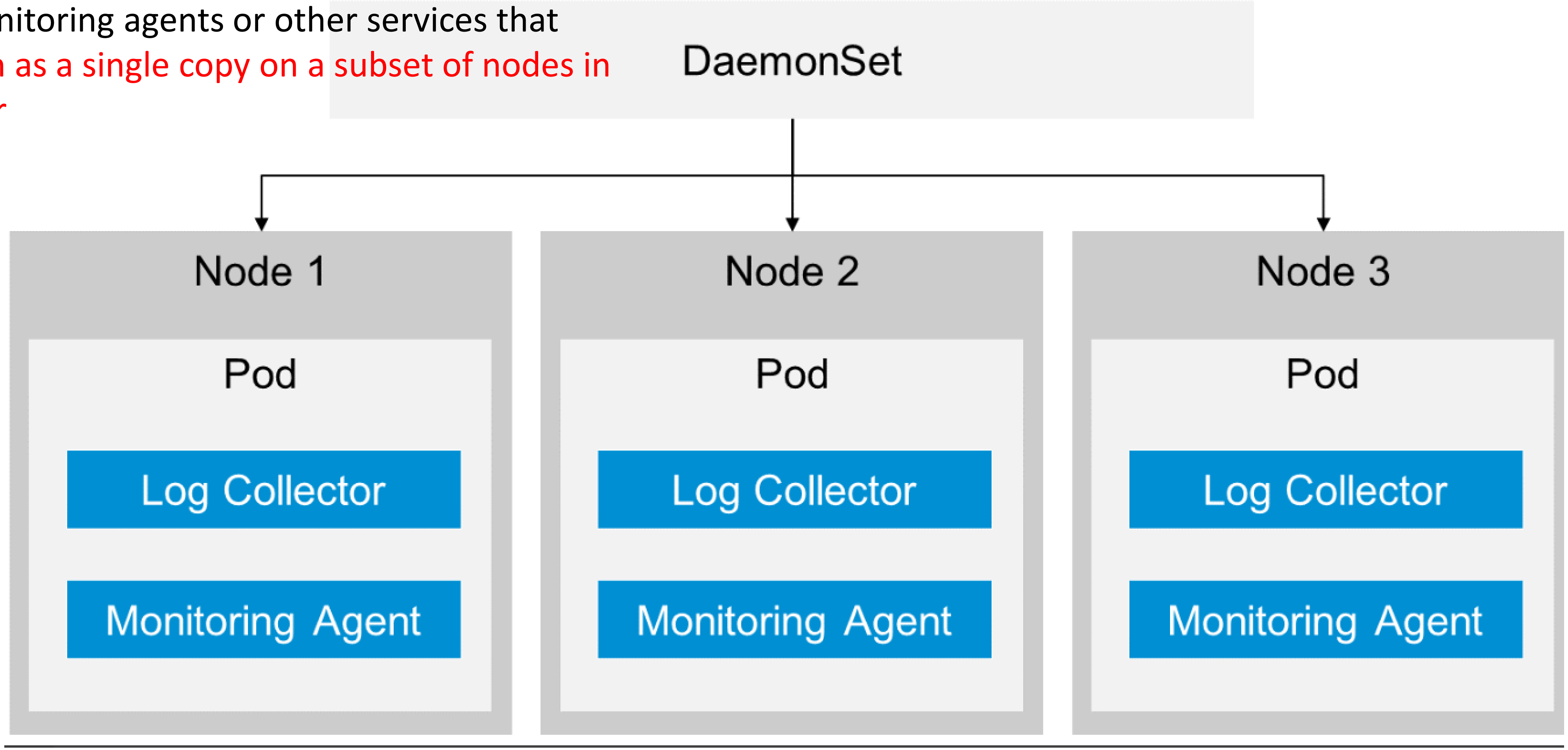
Version 7.9.2

GitLab is the most comprehensive AI-powered DevSecOps Platform.

Integration and delivery

Unit 3 – DaemonSets in Kubernetes

DaemonSets are helpful for services such as logging agents, monitoring agents or other services that need to run as a single copy on a subset of nodes in your cluster



Unit 3 – Jobs in Kubernetes

One-off / Non-parallel Job

Description
A job that runs once and is completed after successful termination

When to use
For example, a database migration

Parallel Job

Description
A job that runs multiple pods in parallel and is completed after all pods have terminated

When to use
For example, a batch job that needs to process a large amount of data

Jobs are generally used to run a workload only once

For example,

- Database migration job
- Some batch job

Supposed to run to completion and then terminate

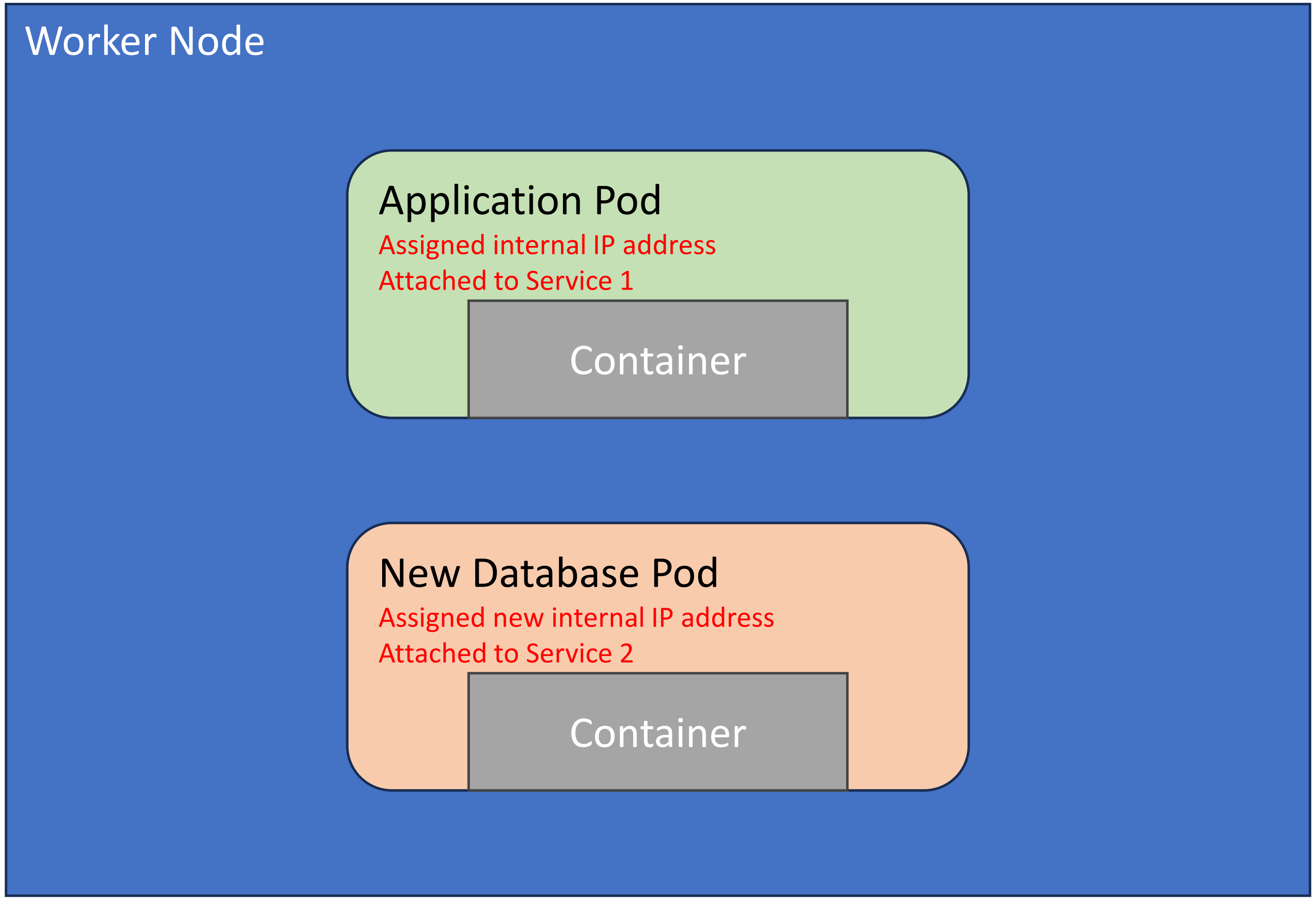
If a job fails, it will be restarted until it succeeds

Unit 4 – Discovering Services in Kubernetes

Service is a permanent IP address attached to each Pod – well, not too attached though

When the Pod dies, the Service does not die along with it. The lifecycle of the Service is independent of the lifecycle of the Pod

So now the Application Pod can communicate to the New Database Pod using Service (Permanent IP address)



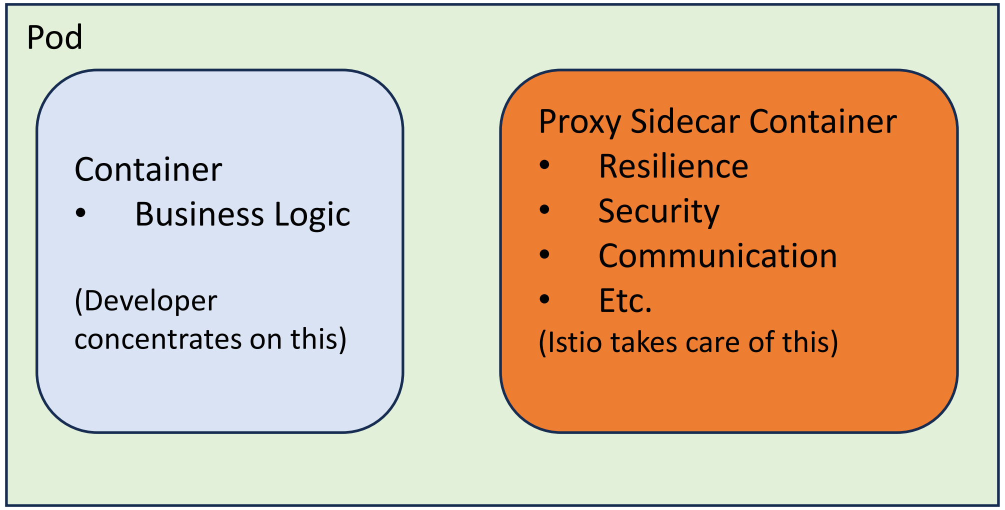
Unit 4 – Discovering Services in Kubernetes

Unit 4 – Using API Gateway to expose Services

- API Gateway is used on top of k8s services to make it publicly available
- API Gateway is a custom configured Istio Ingress Gateway
- By creating an API Rule for a service
  - Creates an Istio Virtualservice
  - Create corresponding Oathkeeper Access Rules

Unit 5 – Service Mesh

- Benefits of Service Mesh
  - Security
  - Observability
  - Traffic Management
  - Resilience
- Service Mesh Architecture



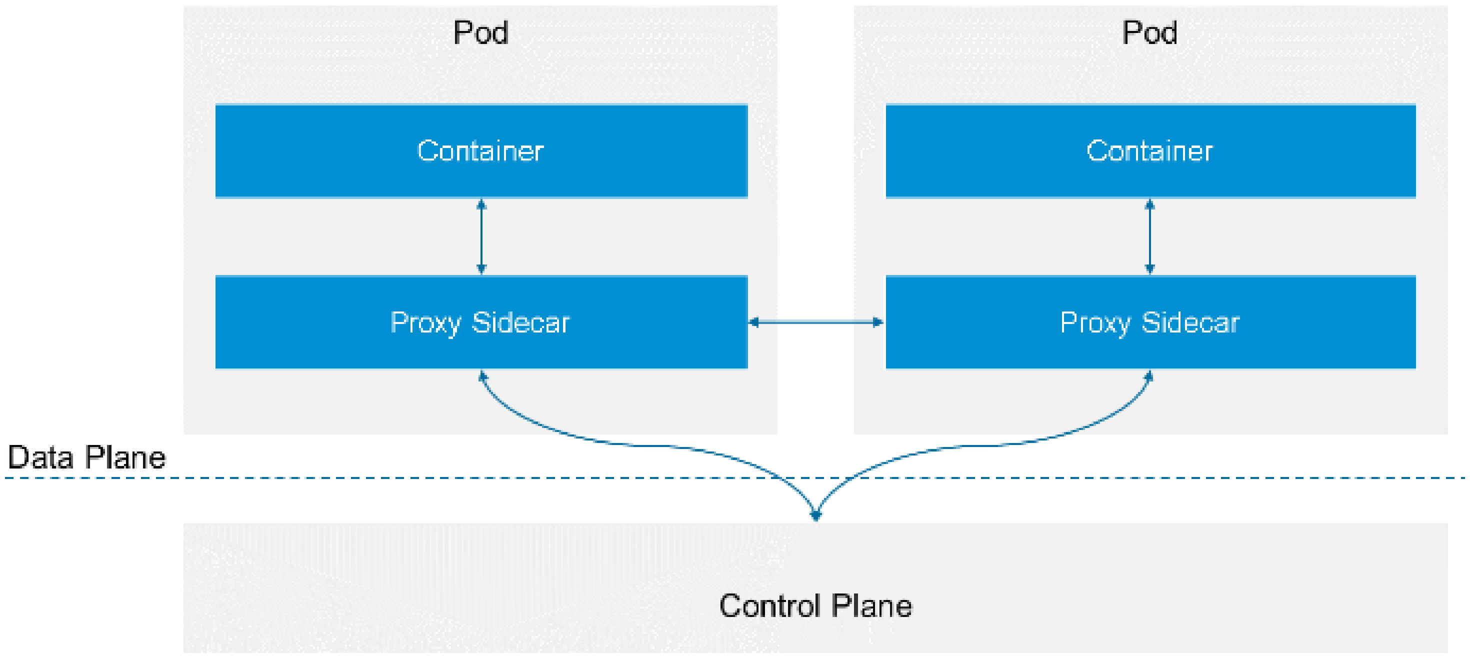
Pod

Container
- Business Logic

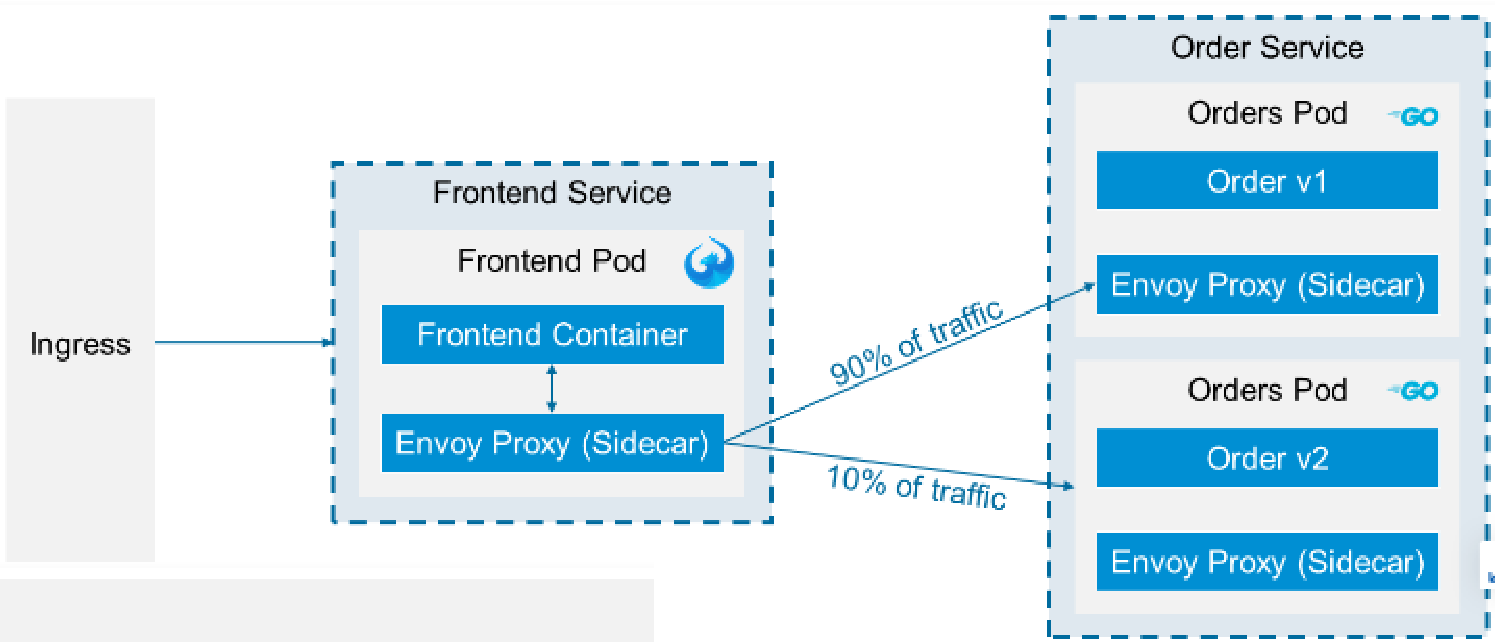
(Developer concentrates on this)

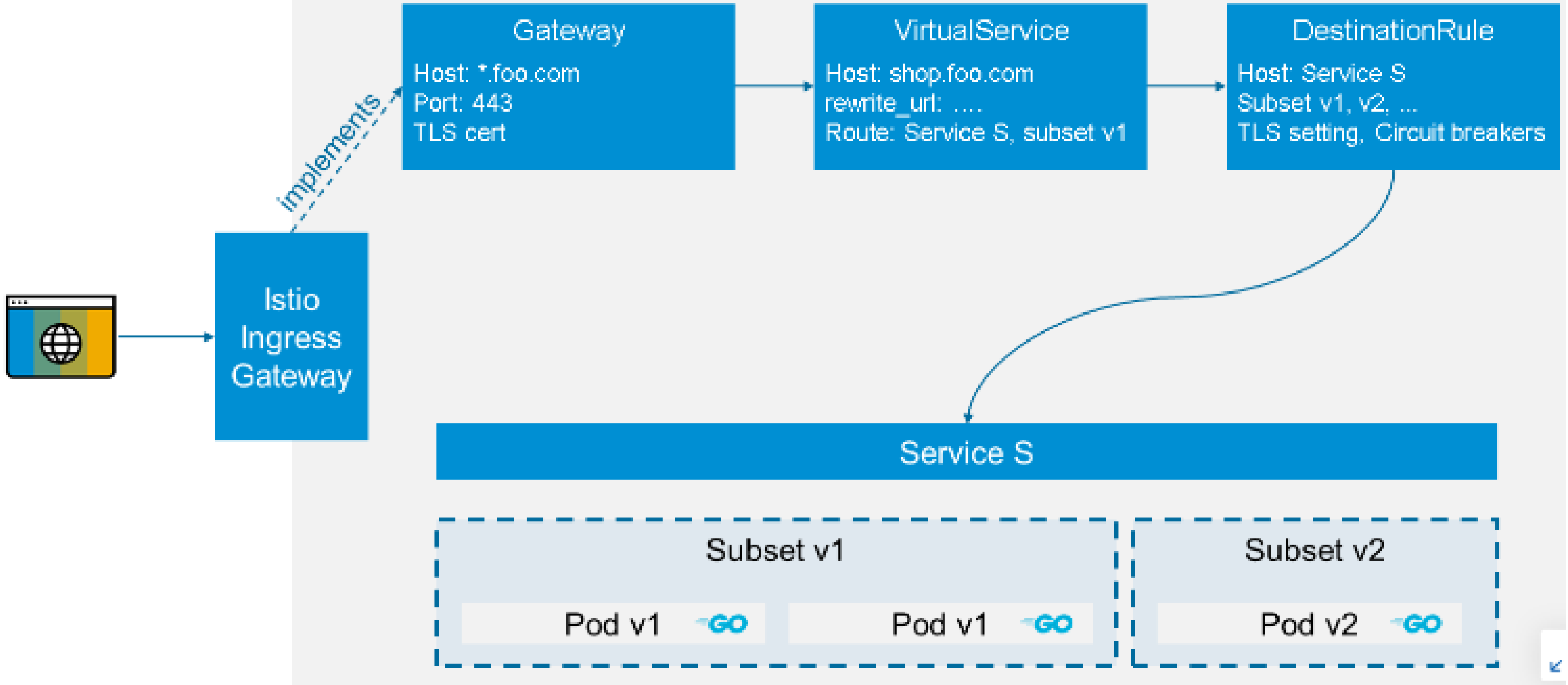
Proxy Sidecar Container
- Resilience
- Security
- Communication
- Etc.

(Istio takes care of this)

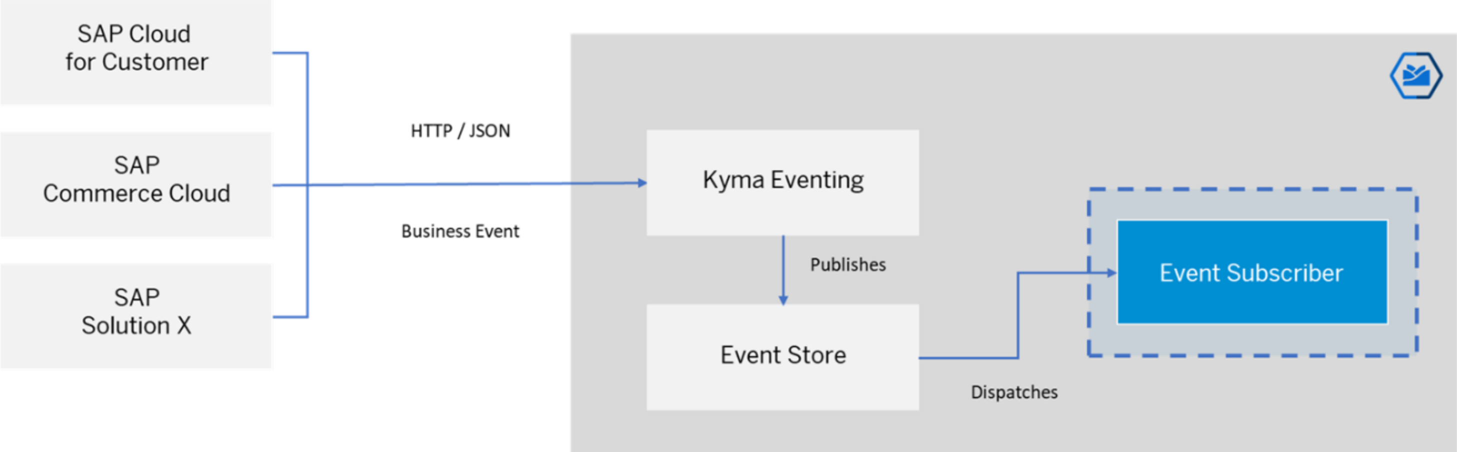


Unit 5 –Istio


Ingress
Frontend Service
Frontend Pod
Frontend Container
Envoy Proxy (Sidecar)
90% of traffic
10% of traffic
Order Service
Orders Pod
GO
Order v1
Envoy Proxy (Sidecar)
Orders Pod
GO
Order v2
Envoy Proxy (Sidecar)

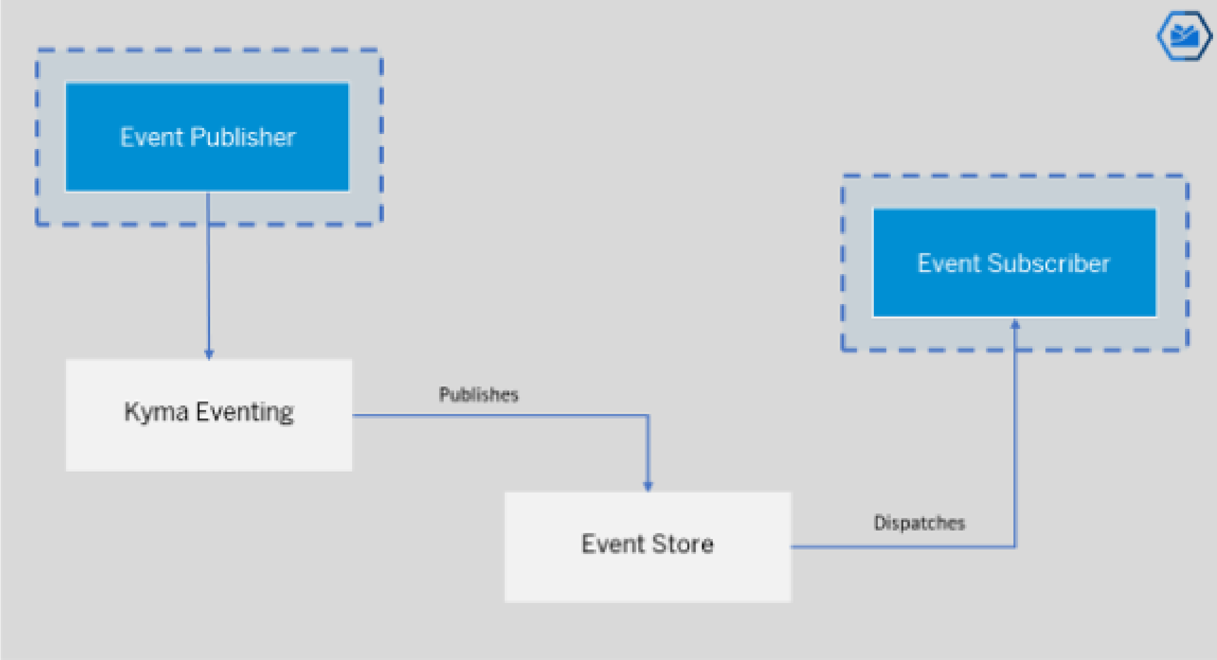

Istio Ingress Gateway
implements
Gateway
Host: *.foo.com
Port: 443
TLS cert
VirtualService
Host: shop.foo.com
rewrite_url: ....
Route: Service S, subset v1
DestinationRule
Host: Service S
Subset v1, v2, ...
TLS setting, Circuit breakers
Service S
Subset v1
Pod v1
GO
Pod v1
GO
Subset v2
Pod v2
GO

Unit 6 – Kyma Eventing

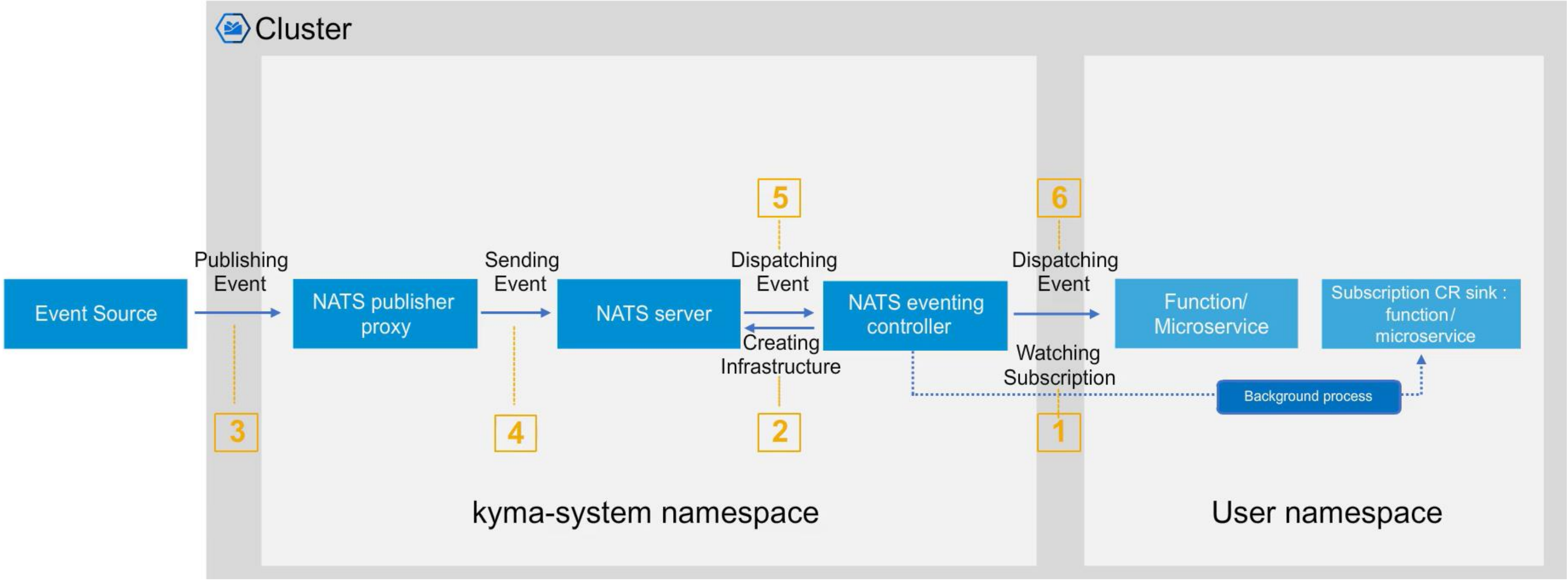


Business events are consumed from connected SAP and non-SAP solutions

In-Cluster events are used for event driven microservices



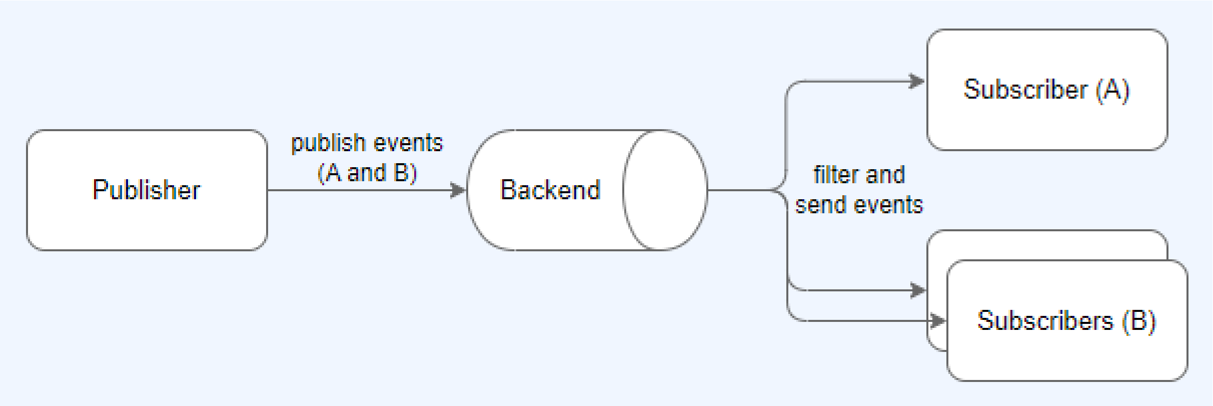
Unit 6 – Kyma Eventing



Kyma Eventing process uses

- NATS (an open source messaging system)
- NATS JetStream (provides the backend)
- HTTP post requests (for sending and receiving events)
- Subscription

Unit 6 – Eventing in Kyma



Eventing in Kyma – how it works ?

- Offer an HTTP endpoint, for example a function to receive events
- Specify the events the user is interested in using Kyma Subscription CR
- Send CloudEvents to the following HTTP endpoints on our Eventing Publisher Proxy Service
  - /publish for CloudEvents

Unit 7 – StatefulSet

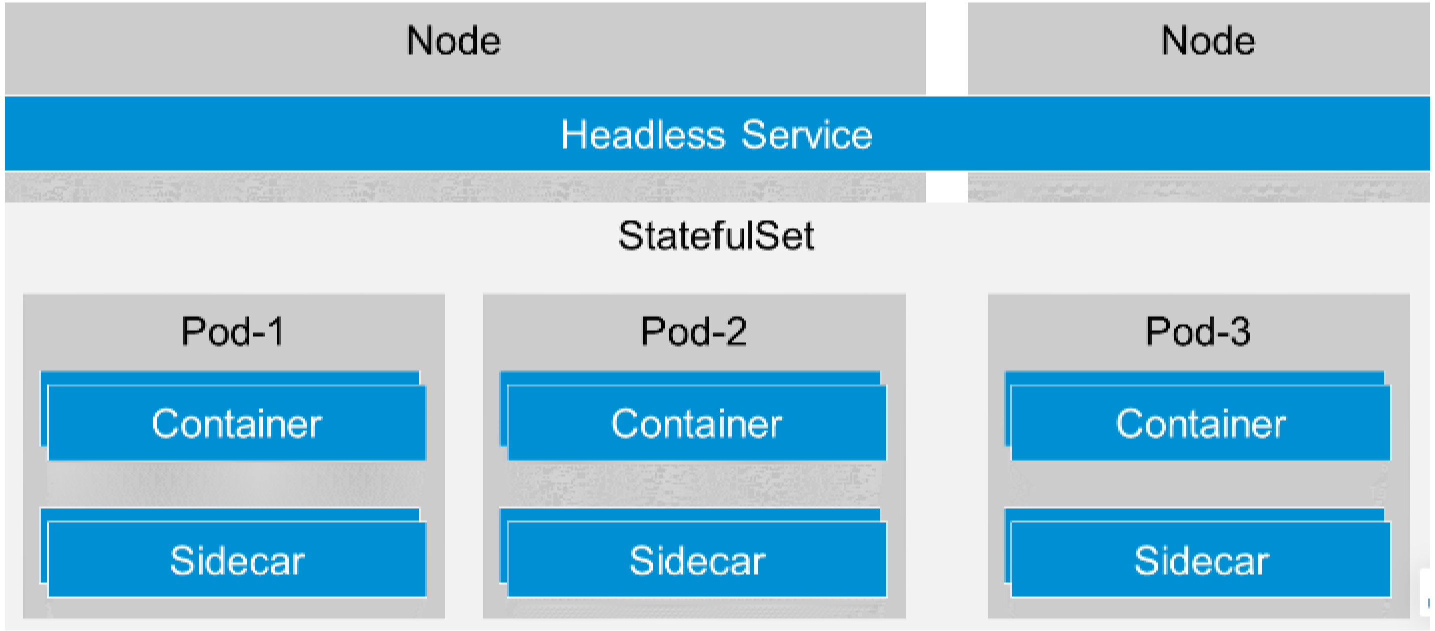
In most cases, pods are stateless
They die and are replaced by any other pod with some random name

But stateful applications (Database) need a unique, consistent identifier.

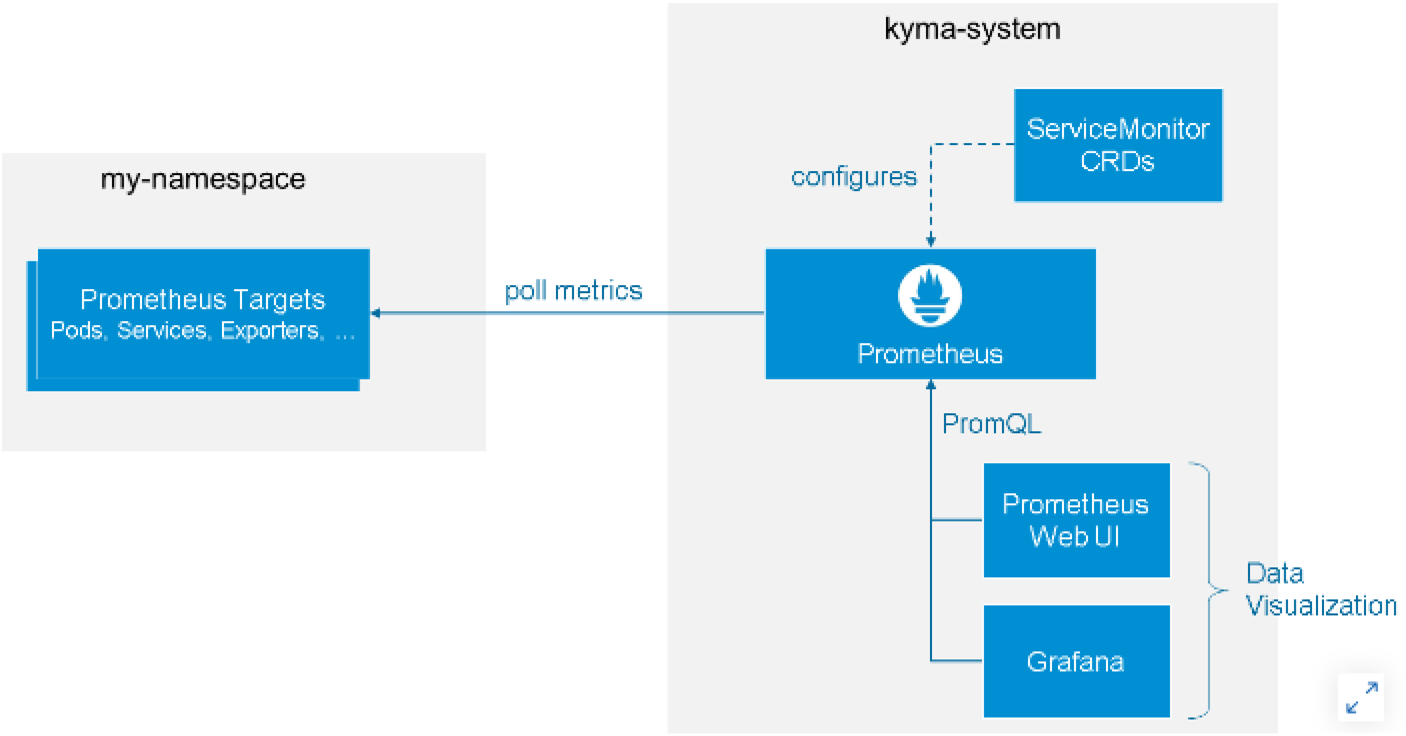
Use StatefulSets when you have the following requirements (e.g. Database)

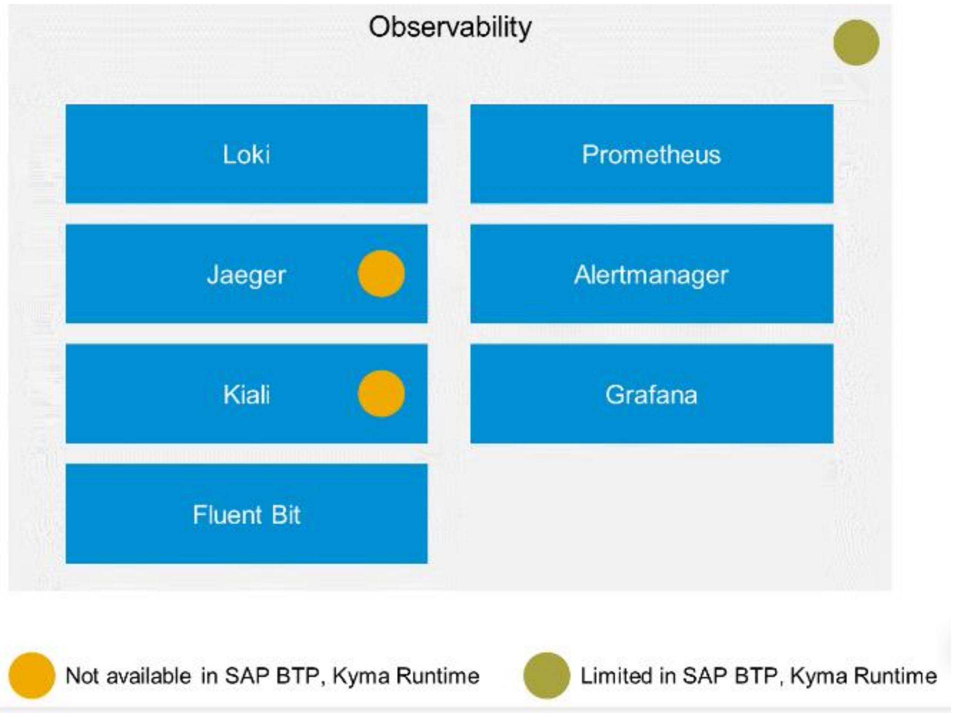
- Stable, unique network identifiers
- Stable, persistent storage
- Ordered, graceful deployment and scaling
- Ordered automated rolling updates

When creating a StatefulSet, pods are created in sequential order (not any random order)



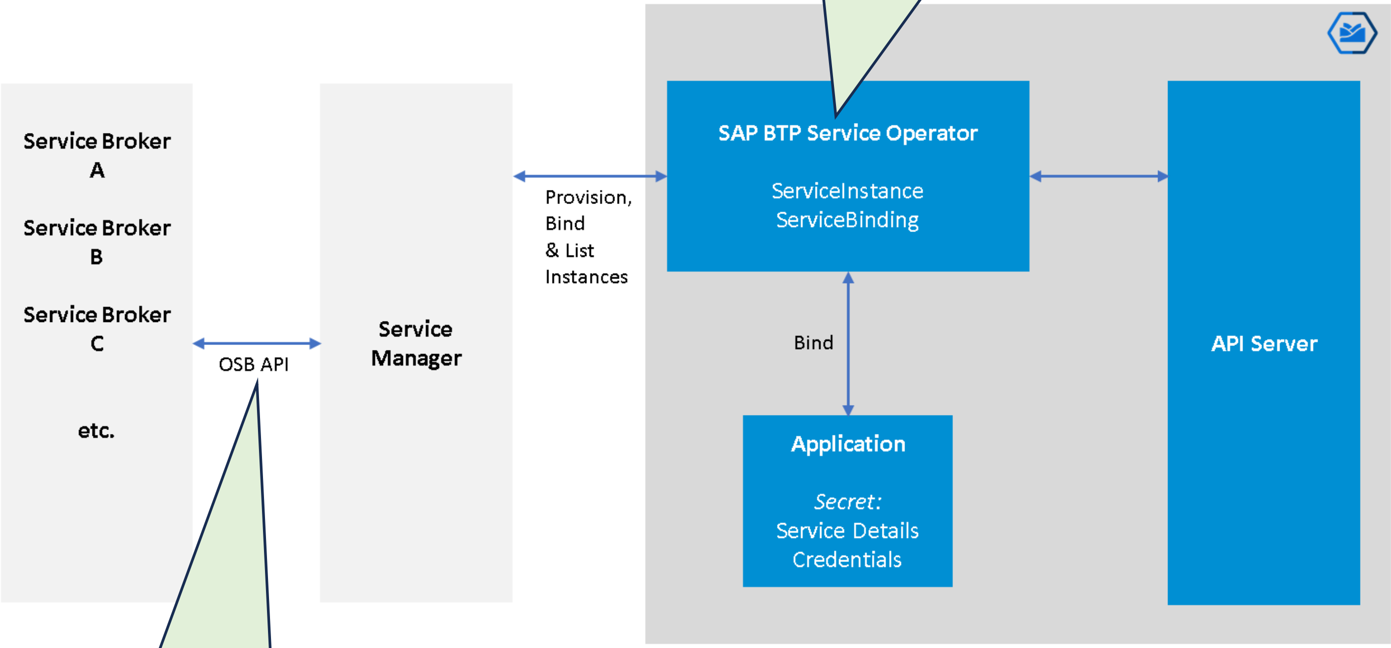
Unit 8 – Observability and Monitoring




Observability
Loki
Prometheus
Jaeger
Alertmanager
Kiali
Grafana
Fluent Bit
● Not available in SAP BTP, Kyma Runtime
● Limited in SAP BTP, Kyma Runtime

Unit 9 – BTP Service Management

Service Management allows you to connect to SAP BTP services to your cluster

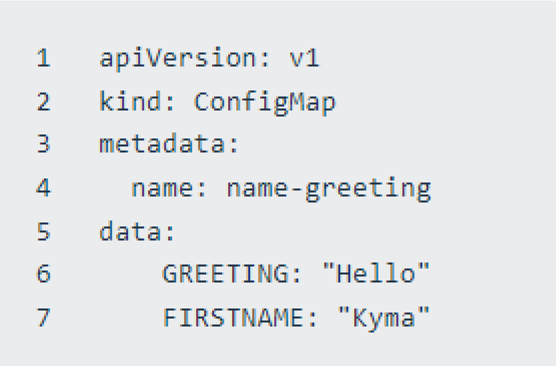


SAP BTP Service Operator uses Service Manager on SAP BTP to manage and consume services

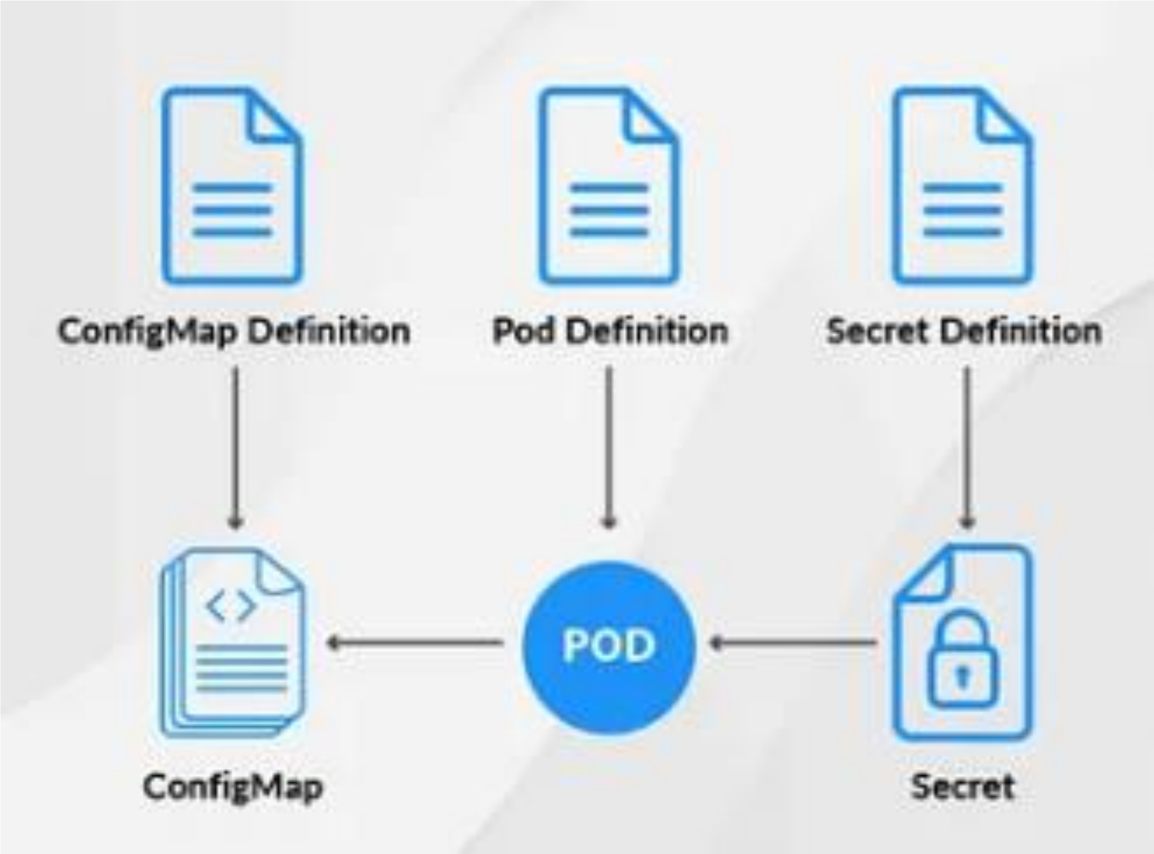
Open Service Broker API allows communication between Service Broker and Service Manager

Unit 9 – SAP BTP Service Management

Unit 10 – ConfigMaps and Secrets



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: name-greeting
data:
    GREETING: "Hello"
    FIRSTNAME: "Kyma"
```



K8s has 2 native resources to store configuration data decoupled from application

ConfigMaps

Non Sensitive data

Secrets

Sensitive data (passwords, apikey etc.)

Unit 10 – Config Maps, Secrets