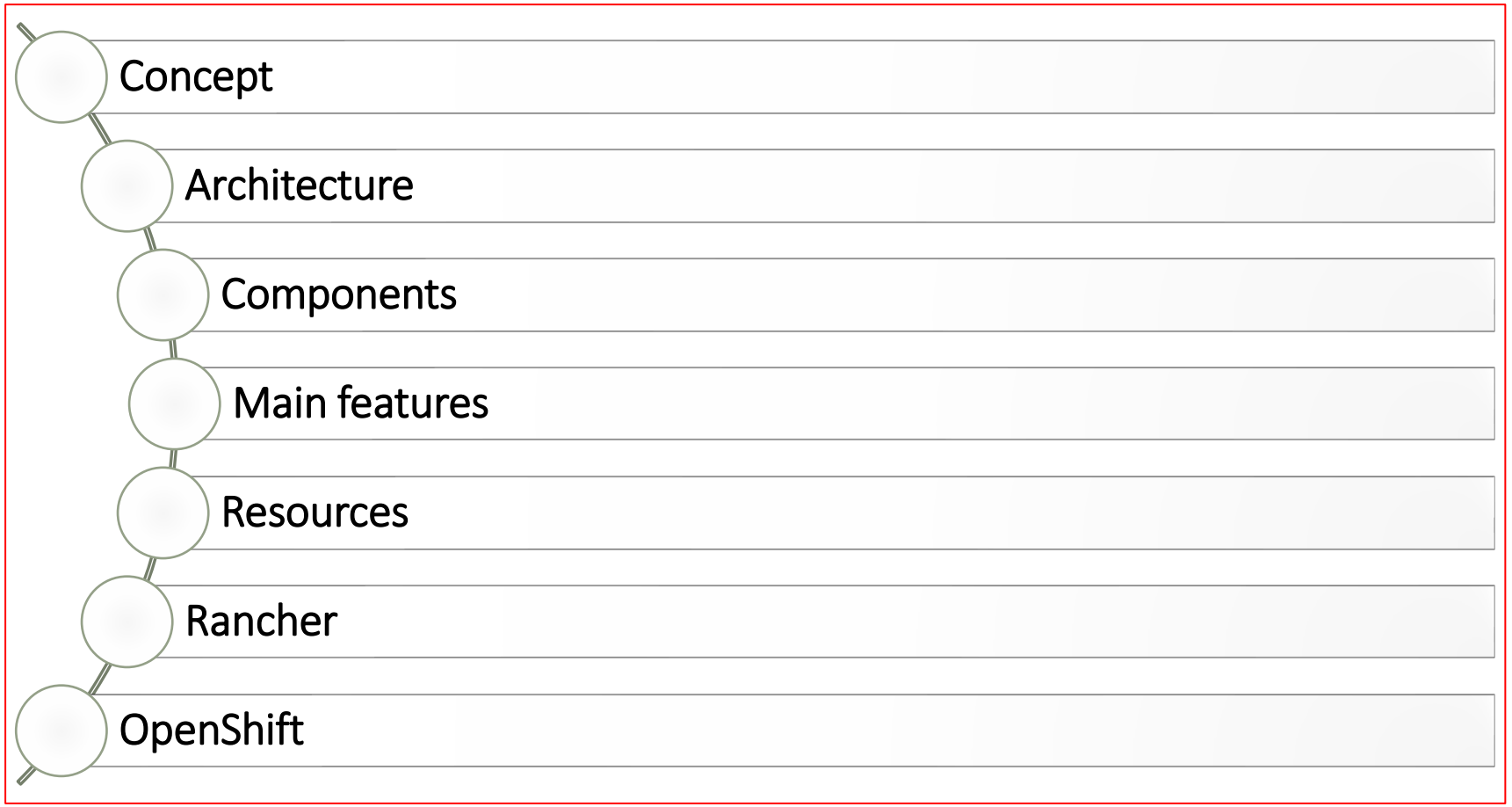
iPMAC
NET YOUR WORK

DEVSECOPS COURSE

# CONTAINER ORCHESTRATION

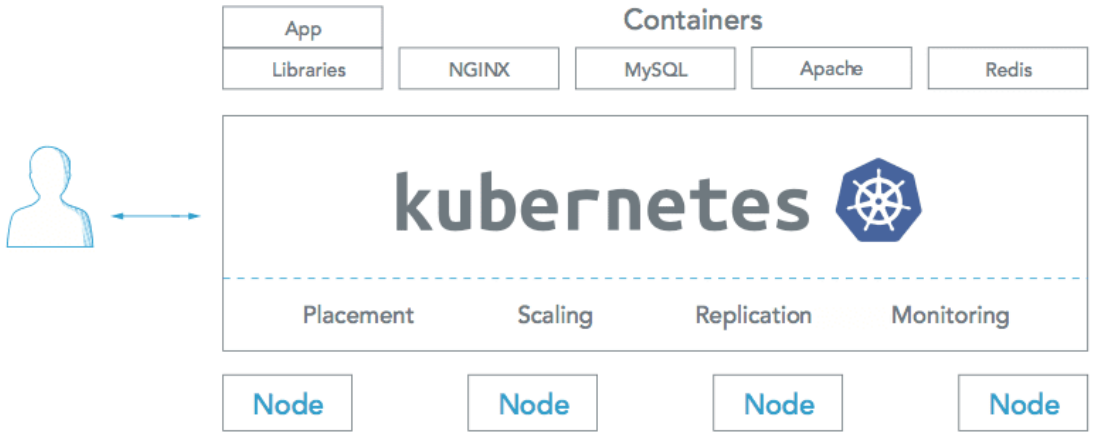
TRAINER: TRAN HUU HOA

# AGENDA

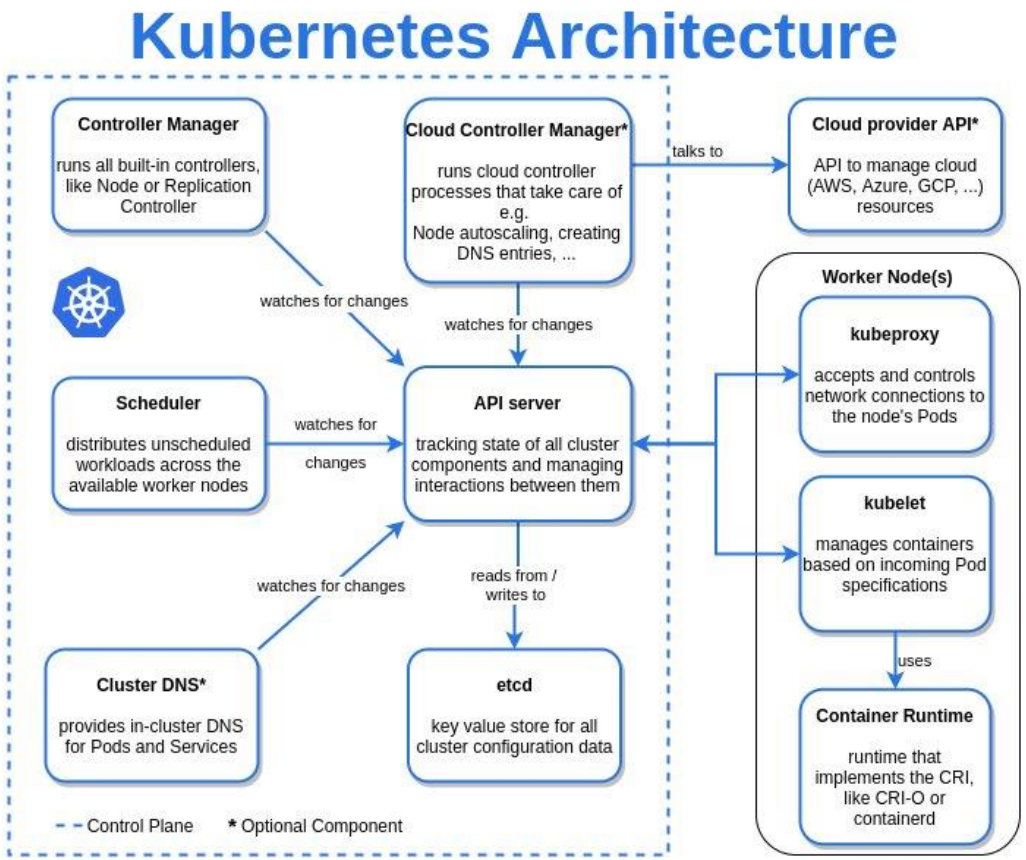

Concept
Architecture
Components
Main features
Resources
Rancher
OpenShift

# CONCEPT

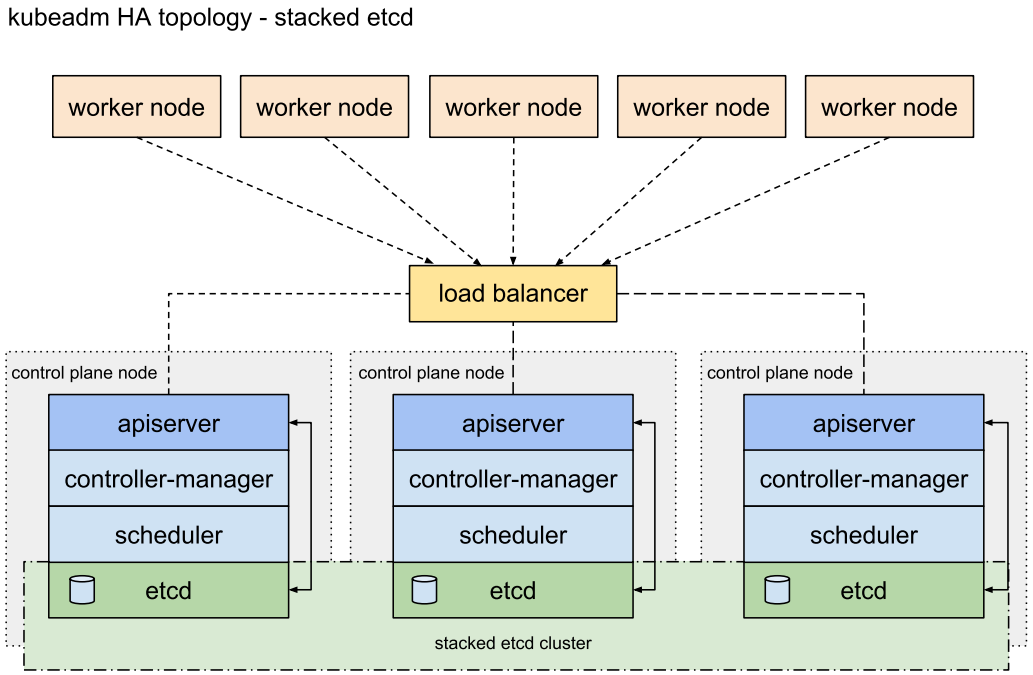
Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.



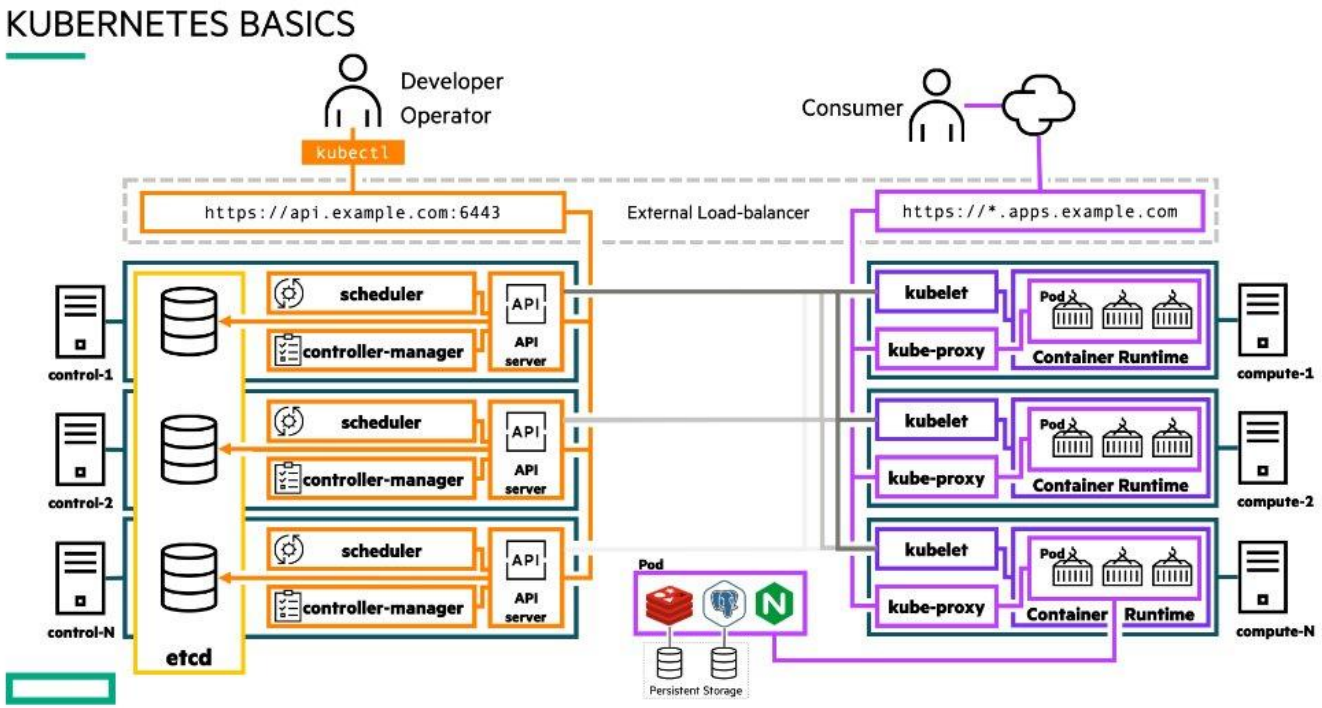
# ARCHITECTURE



# ARCHITECTURE



# ARCHITECTURE



# COMPONENTS

## ***ETCD***

etcd is an open-source, distributed key-value storage system that facilitates the configuration of resources, the discovery of services, and the coordination of distributed systems such as clusters and containers. The etcd is Kubernetes standard storage system that stores all cluster information, such as its current state, desired state, configuration of resources, and runtime data.

There are 2 ways to deploy etcd on a Kubernetes cluster:

- On Control Plane Node
- On a Dedicated cluster

# COMPONENTS

## ETCD

- etcd monitors the various nodes and, as a result, sees which resources are free. Based on this, the control plane assigns the task to the relevant resource.
- etcd monitors the health of all nodes at regular intervals. Thus, if any node is overloaded or underused etcd possesses the applicable data. The control plane can either delete the node or reassign the task to another node.
- etcd implements several mechanisms to avoid resource starvation and ensure the availability and reliability of its services.
- etcd's features such as shared configuration, service discovery, leader election, distributed locks, and the watch API, can help address cross-communication concerns in Kubernetes. This is achieved through better synchronization and interaction among various components and services.

# COMPONENTS

## ***Kube-apiserver***

The Kubernetes API server validates and configures data for the api objects which include pods, services, replication controllers, and others. The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

# COMPONENTS

## ***Kube-controller-manager***

The Kubernetes controller manager is a daemon that embeds the core control loops shipped with Kubernetes. In applications of robotics and automation, a control loop is a non-terminating loop that regulates the state of the system. In Kubernetes, a controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state.

# COMPONENTS

## ***Kube-scheduler***

The Kubernetes scheduler is a control plane process which assigns Pods to Nodes. The scheduler determines which Nodes are valid placements for each Pod in the scheduling queue according to constraints and available resources. The scheduler then ranks each valid Node and binds the Pod to a suitable Node.

# COMPONENTS

## ***Kubelet***

The kubelet is the primary "node agent" that runs on each node. It can register the node with the apiserver using one of the hostname; a flag to override the hostname; or specific logic for a cloud provider.

# COMPONENTS

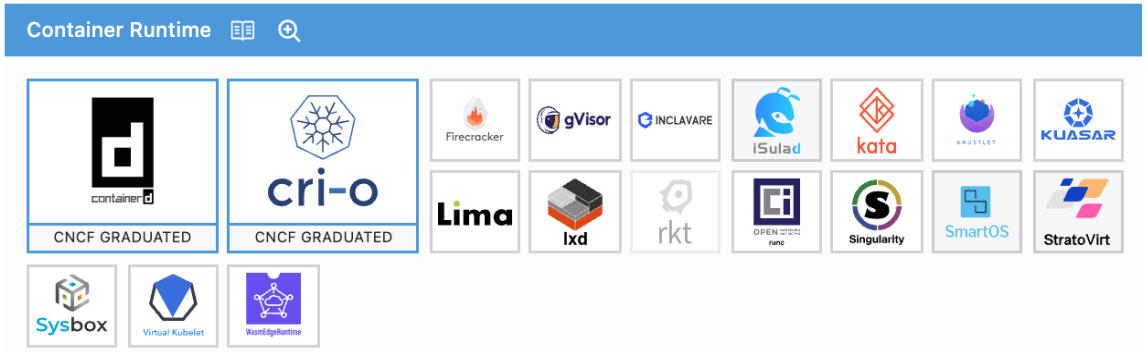
## ***Kube-proxy***

The Kubernetes network proxy runs on each node. This reflects services as defined in the Kubernetes API on each node and can do simple TCP, UDP, and SCTP stream forwarding or round robin TCP, UDP, and SCTP forwarding across a set of backends.

# COMPONENTS

## ***Container runtime***

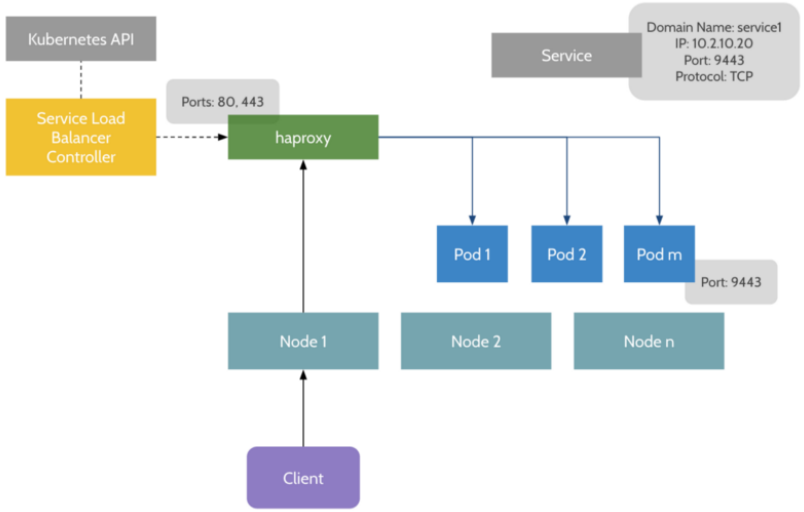
A fundamental component that empowers Kubernetes to run containers effectively. It is responsible for managing the execution and lifecycle of containers within the Kubernetes environment.



# MAIN FEATURES

## ***Service discovery and load balancing***

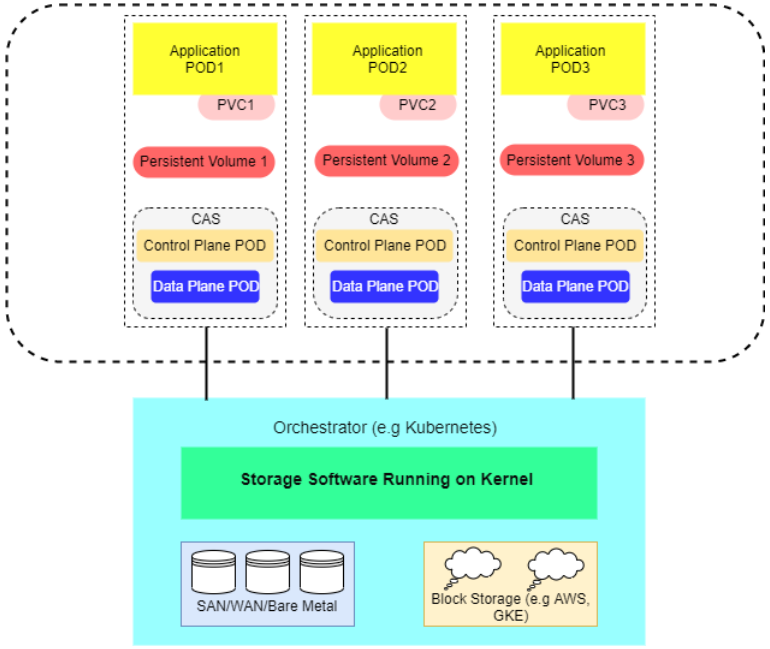
Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.



# MAIN FEATURES

## ***Storage orchestration***

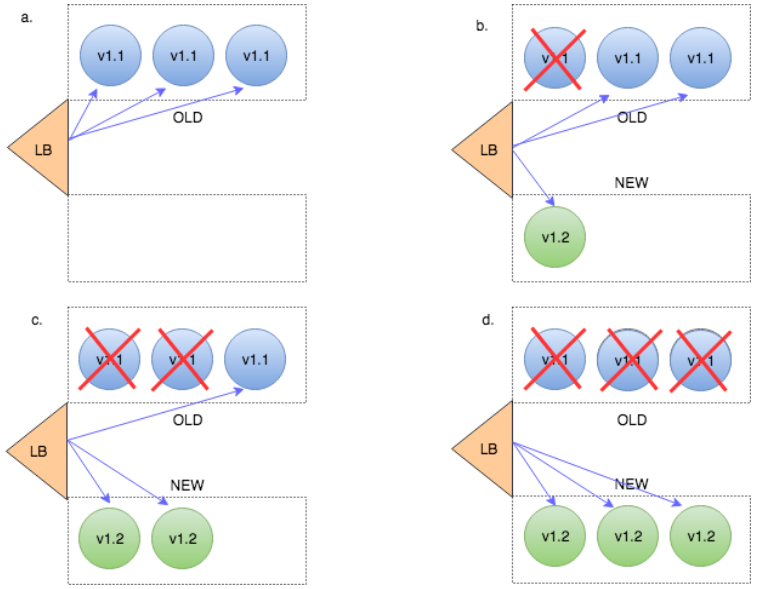
Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.



# MAIN FEATURES

## ***Automated rollouts and rollbacks***

You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.



# MAIN FEATURES

***Automatic bin packing***

You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.

# MAIN FEATURES

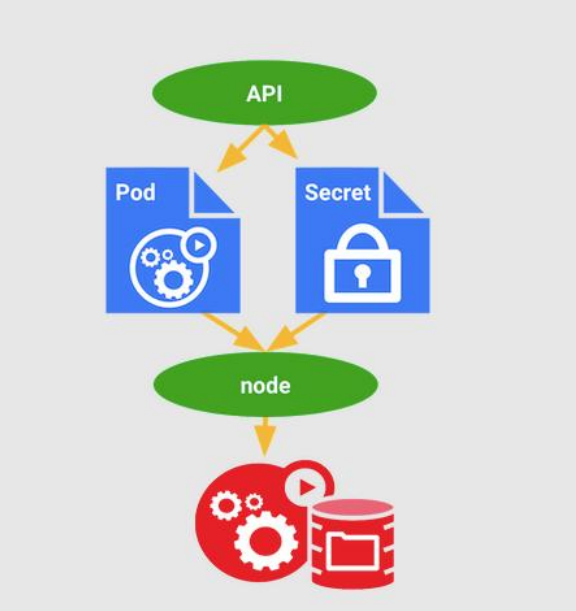
***Self-healing***

Kubernetes restarts containers that fail, replaces containers, 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.

# MAIN FEATURES

***Secret and configuration management***

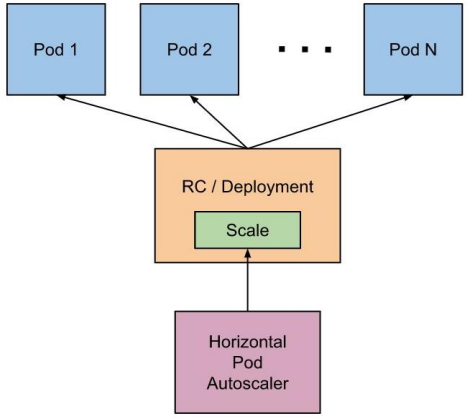
Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.



# MAIN FEATURES

***Horizontal scaling***

Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.



# MAIN FEATURES

- **Batch execution** In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.
- **IPv4/IPv6 dual-stack** Allocation of IPv4 and IPv6 addresses to Pods and Services
- **Designed for extensibility** Add features to your Kubernetes cluster without changing upstream source code

# RESOURCES

**Foundations:**

- Namespace
- Pod

**Workloads:**

- Deployment/ReplicaSet
- StatefulSet
- DaemonSet
- Job/CronJob

**Networking:**

- Service
- DNS record
- Ingress/Ingress Controllers
- Network policy

**Storage:**

- Volume
- Storage class
- Persistent volume
- Persistent volume claim

**Others**

- ConfigMaps
- Secret
- Resource quotas
- Custom resources definition

# RESOURCES

## ***Foundations:***

- **Namespace**: Namespaces are intended for use in environments with many users spread across multiple teams, or projects. For clusters with a few to tens of users, you should not need to create or think about namespaces at all. Start using namespaces when you need the features they provide
- **Pod**: The smallest deployable units of computing that you can create and manage in Kubernetes. A Pod (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.

# RESOURCES

## ***Workloads:***

- **Deployment/replicaset**: 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.
- **StatefulSet**: lets you 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 persistent volume. Your code, running in the Pods for that StatefulSet, can replicate data to other Pods in the same StatefulSet to improve overall resilience
- **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. Every time you 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/CronJob:** define tasks that run to completion and then stop. Jobs represent one-off tasks, whereas CronJobs recur according to a schedule.

# RESOURCES

## ***Networking***

- **Service**: An abstract way to expose an application running on a set of Pods as a network service. With Kubernetes you don't 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.
- **DNS record**: Kubernetes DNS schedules a DNS Pod and Service on the cluster and configures the kubelets to tell individual containers to use the DNS Service's IP to resolve DNS names. Every Service defined in the cluster (including the DNS server itself) is assigned a DNS name. By default, a client Pod's DNS search list includes the Pod's own namespace and the cluster's default domain.
- **Ingress/Ingress Controllers**: Ingress exposes HTTP and HTTPS routes from outside the cluster to service within the cluster. Traffic routing is controlled by rules defined on the Ingress resource. For the Ingress resource to work, the cluster must have an ingress controller running.
- **Network policy**: an application-centric construct which allow you to specify how a pod is allowed to communicate with various network "entities" (we use the word "entity" here to avoid overloading the more common terms such as "endpoints" and "services", which have specific Kubernetes connotations) over the network

# RESOURCES

## ***Storage***

- **Volume**: On-disk files in a container are ephemeral, which presents some problems for non-trivial applications when running in containers
- **Storage class**: provides a way for administrators to describe the "classes" of storage they offer. Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators
- **Persistent volume**: a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using storage class.
- **Persistent volume claim**: a request for storage allow a user to consume abstract storage resources, it is common that users need Persistent Volumes with varying properties, such as performance, for different problems.

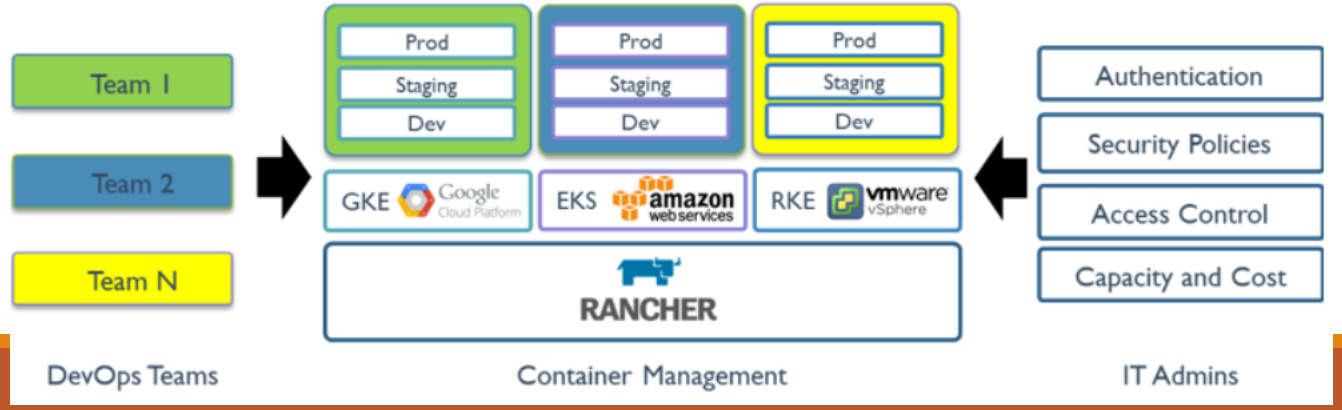
# RANCHER

## *Concept*

Rancher is a Kubernetes management tool to deploy and run clusters anywhere and on any provider

## *Usages*

- Provision Kubernetes from a hosted provider
- Provision compute nodes and then install Kubernetes onto them
- Import existing Kubernetes clusters running anywhere



# RANCHER

***Main features:***

- Authorization and Role-Based Access Control
  - ✓ User management
  - ✓ Authorization
- Working with Kubernetes
  - ✓ Provisioning Kubernetes clusters
  - ✓ Catalog management
  - ✓ Managing projects
  - ✓ Fleet Continuous Delivery
  - ✓ Istio
- Working with Cloud Infrastructure
  - ✓ Tracking nodes
  - ✓ Setting up infrastructure
- Cluster Visibility
  - ✓ Logging
  - ✓ Monitoring
  - ✓ Alerting

# RANCHER

## Comparison

| Action | Rancher Launched Kubernetes Clusters | EKS, GKE and AKS Clusters[1] | Other Hosted Kubernetes Clusters | Non-EKS or GKE Registered Clusters |
|---|---|---|---|---|
| Using kubectl and a kubeconfig file to Access a Cluster | ✓ | ✓ | ✓ | ✓ |
| Managing Cluster Members | ✓ | ✓ | ✓ | ✓ |
| Editing and Upgrading Clusters | ✓ | ✓ | ✓ | ✓[2] |
| Managing Nodes | ✓ | ✓ | ✓ | ✓[3] |
| Managing Persistent Volumes and Storage Classes | ✓ | ✓ | ✓ | ✓ |
| Managing Projects, Namespaces and Workloads | ✓ | ✓ | ✓ | ✓ |
| Using App Catalogs | ✓ | ✓ | ✓ | ✓ |
| Configuring Tools (Alerts, Notifiers, Monitoring, Logging, Istio) | ✓ | ✓ | ✓ | ✓ |
| Running Security Scans | ✓ | ✓ | ✓ | ✓ |
| Use existing configuration to create additional clusters | ✓ | ✓ | ✓ | |
| Ability to rotate certificates | ✓ | ✓ | | |
| Ability to backup and restore Rancher-launched clusters | ✓ | ✓ | | ✓[4] |
| Cleaning Kubernetes components when clusters are no longer reachable from Rancher | ✓ | | | |
| Configuring Pod Security Policies | ✓ | ✓ | | |

# RANCHER

***Options:***

## Rancher

The community's favorite Kubernetes management platform — free forever

- 100% Open Source
- Full Rancher Platform Experience
- Community Support

## Rancher **Prime**

Get the same Rancher platform experience with added value from Rancher Prime

- 100% Open Source
- Full Rancher Platform Experience
- Enterprise Support
- Access to Professional Services including Go-Live
- Deploy from trusted private container registry

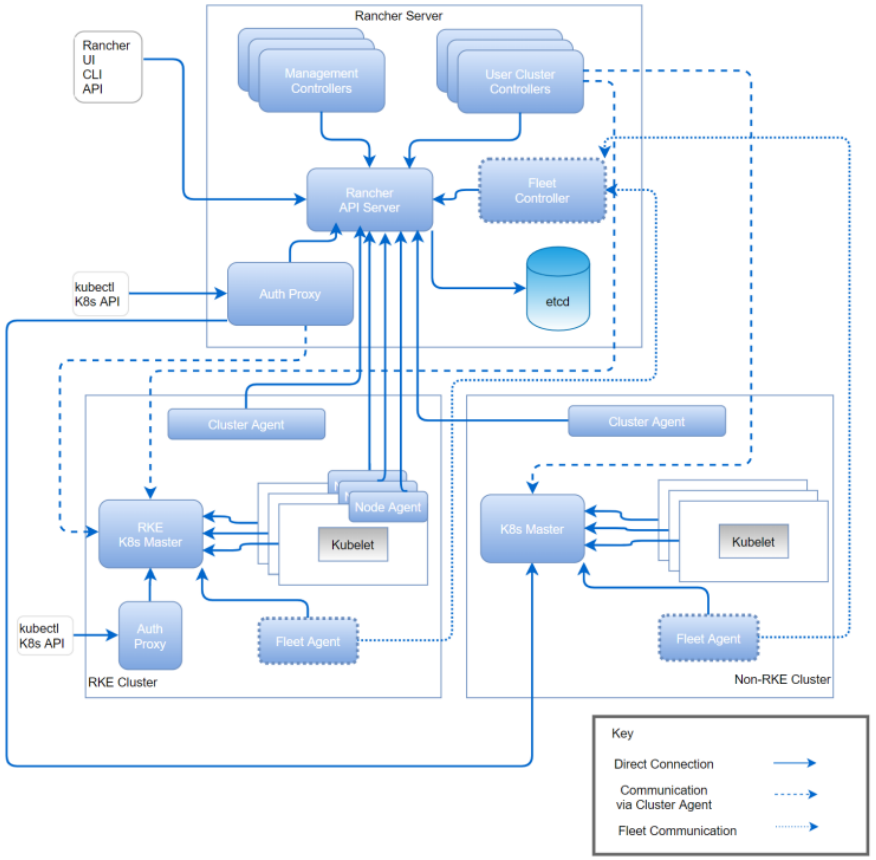
## Rancher **Prime Hosted**

The premium white-glove service for managing Rancher Prime

- 100% Open Source
- Full Rancher Platform Experience
- Enterprise Support, Expert-managed Rancher Prime control plane
- Access to Professional Services including Go-Live
- Deploy from trusted private container registry

# RANCHER

***Architecture:***

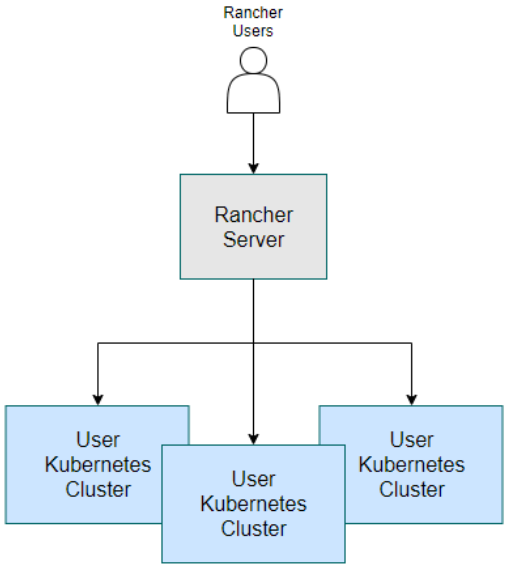


# RANCHER

## ***Deployment view:***

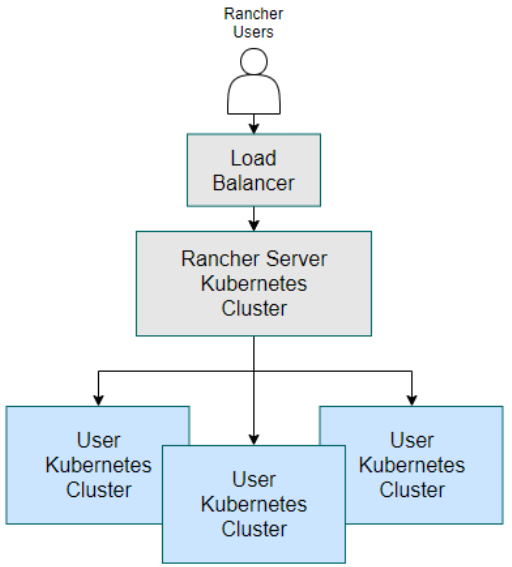
### ***Non-Production***

Separation of Single Node Rancher Server and User Clusters




### ***Production***

Separation of High-availability Rancher Server and User Clusters



# RANCHER

## Setup and configuration

### Quick start:

- Run Rancher Docker in a virtual machine
- Create cluster via Docker UI

### Deploy to a Kubernetes cluster using Helm:

- Provision a Kubernetes cluster
- Deploy a Rancher using helm

### Others options:

- Amazon Elastic Kubernetes service (EKS)
- Microsoft Azure Kubernetes service (AKS)
- Google Kubernetes engine (GKE)
- Etc.

# RANCHER

## *Eco system - Catalogs*

# RANCHER

## Rancher Kubernetes engine (RKE)

### Concept:

RKE is a CNCF-certified Kubernetes distribution that runs entirely within Docker containers. It solves the common frustration of installation complexity with Kubernetes by removing most host dependencies and presenting a stable path for deployment, upgrades, and rollbacks.

### How-to-use:

- Download the RKE binary or install it
- Create the cluster configuration file
- Deploy Kubernetes  with RKE
- Save your files (if any)

# OPENSHIFT

OpenShift is a powerful container orchestration platform developed by Red Hat. It is built on Kubernetes, the leading open-source platform for managing containerized applications. OpenShift enhances Kubernetes by offering a suite of tools and features that simplify deployment, scaling, and management of applications in a cloud environment

# OPENSHIFT

## ***Key Features***

- Kubernetes-Based
- Developer-Centric
- Simplified Deployment
- Integrated Container Registry
- Multi-Environment Support
- Scalability
- Self-Service
- Security and Compliance
- Monitoring and Logging
- Open Source

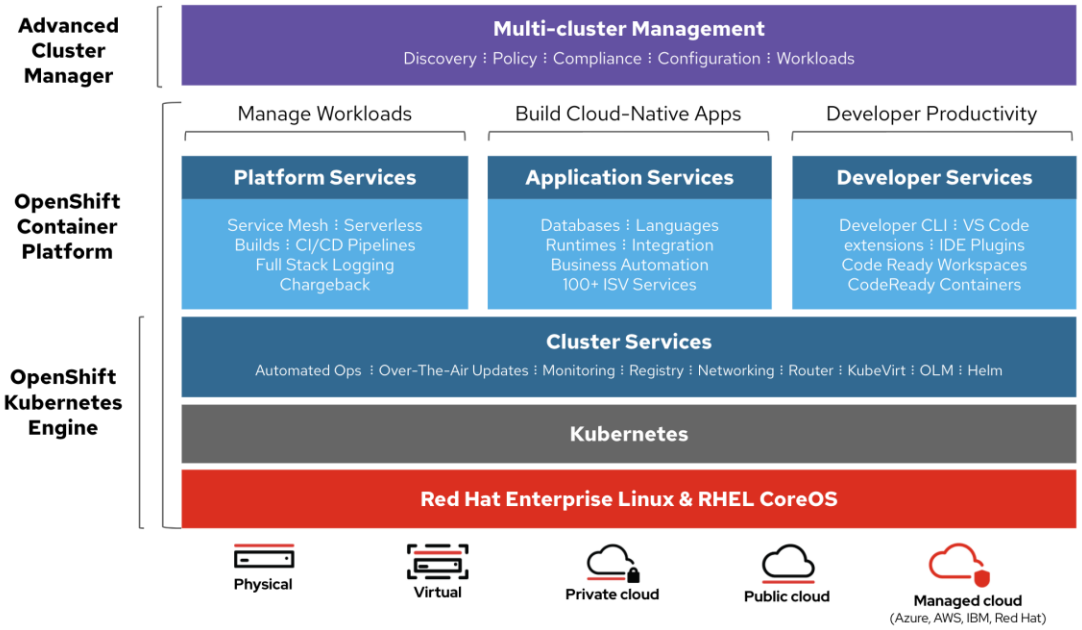
# OPENSHIFT

## ***Use Cases***

- Microservices Architecture
- Hybrid Cloud Deployments
- CI/CD Pipelines
- Development Environments

# OPENSHIFT

## ***Architecture***



# OPENSHIFT

***Ecosystem:***