Agenda

- ❏   Cluster API Introduction
- ❏   Cluster API Glossary
- ❏   How does Cluster API work?
- ❏   CRDs and Spec
- ❏   ClusterClass and Managed Topology
- ❏   Hands-on tutorials

Problems

- **Managing Kubernetes clusters is difficult.**

Problems

- Managing Kubernetes clusters is difficult.

- **Kubeadm and other tools can help, but they do not manage infrastructure.**

Problems

- Managing Kubernetes clusters is difficult.

- Kubeadm and other tools can help, but they do not manage infrastructure.

- **Managing clusters across different cloud providers and bare metal requires infrastructure-specific knowledge.**

Problems

- Managing Kubernetes clusters is difficult.

- Kubeadm and other tools can help, but they do not manage infrastructure.

- Managing clusters across multiple types of infrastructures requires providers-specific knowledge.

- **Due to the lack of a common API interface, the experience managing Kubernetes clusters across different infrastructures is inconsistent.**

Cluster API to the rescue…

- **Cluster API provides extensible APIs to create Kubernetes clusters across multiple infrastructures.**

Cluster API to the rescue…

- Cluster API provides extensible APIs to create Kubernetes clusters across multiple infrastructures.
- **It not only provides declarative APIs for Kubernetes objects, but also for infrastructure components.**

Cluster API to the rescue...

- Cluster API provides extensible APIs to create Kubernetes clusters across multiple infrastructures.
- It not only provides declarative APIs for Kubernetes objects, but also for infrastructure components.
- **It has a pluggable architecture which makes it easy to onboard new infrastructure providers.**

Cluster API to the rescue...

- Cluster API provides extensible APIs to create Kubernetes clusters across multiple infrastructures.
- It not only provides declarative APIs for Kubernetes objects, but also for infrastructure components.
- It has a pluggable architecture which makes it easy to onboard new infrastructure providers.
- **A common API interface enables development of tooling to manage clusters across environments.**
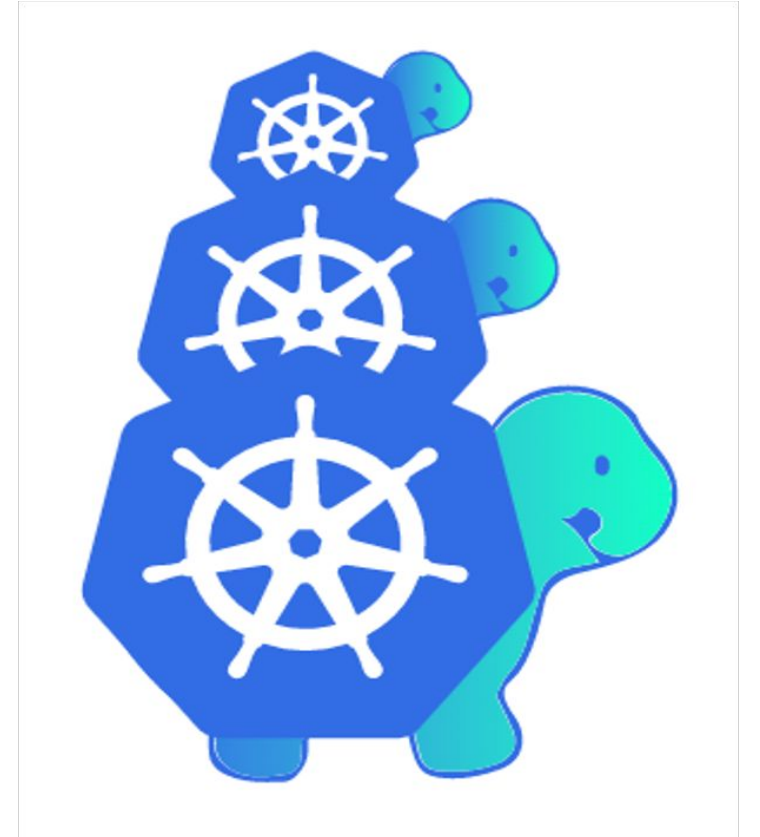
What is Cluster API?
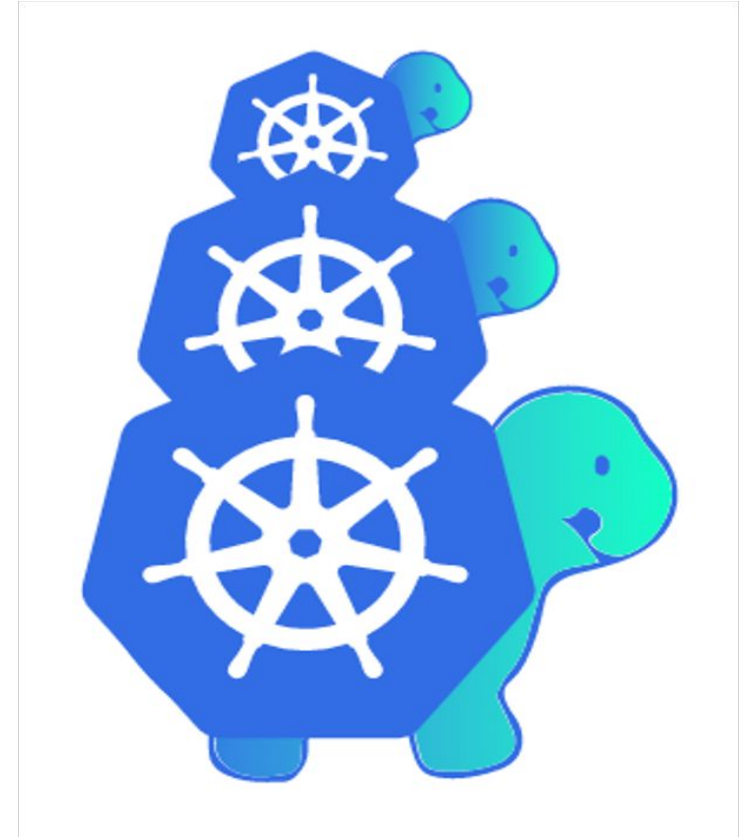
- **Cluster API is a project of the SIG Cluster Lifecycle**
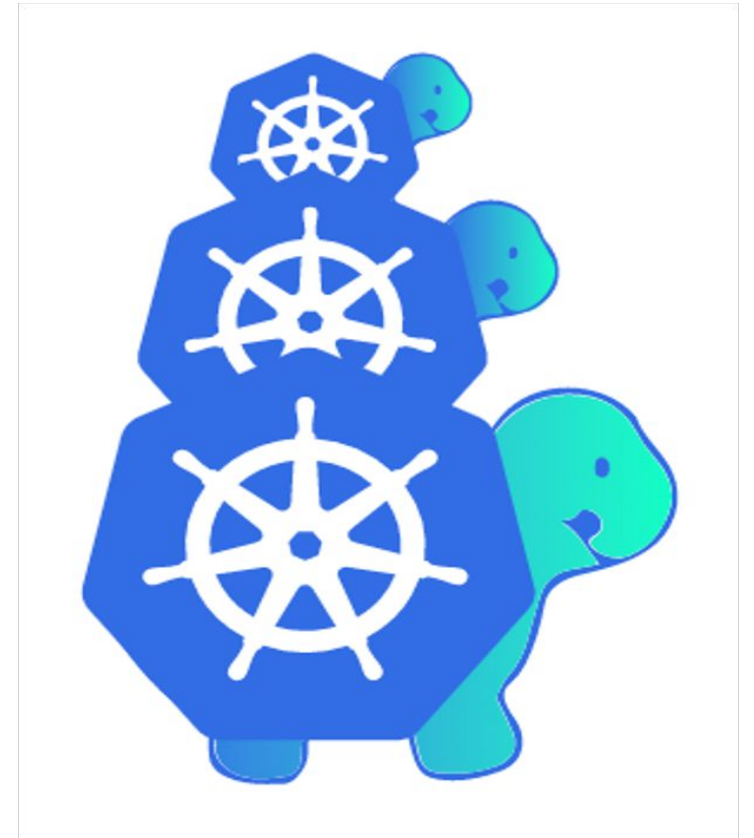
What is Cluster API?

- Cluster API is a project of the SIG Cluster Lifecycle.

- **It uses Kubernetes to manage Kubernetes.**

What is Cluster API?

- Cluster API is a project of the SIG Cluster Lifecycle.

- It uses Kubernetes to manage Kubernetes.

"turtles all the way down"

What is Cluster API?

- Cluster API is a project of the SIG Cluster Lifecycle.

- It uses Kubernetes to manage Kubernetes.

- **To manage Kubernetes clusters, Cluster API offers the same declarative Kubernetes-style APIs that Kubernetes itself uses to manage Deployments, Pods and everything else.**
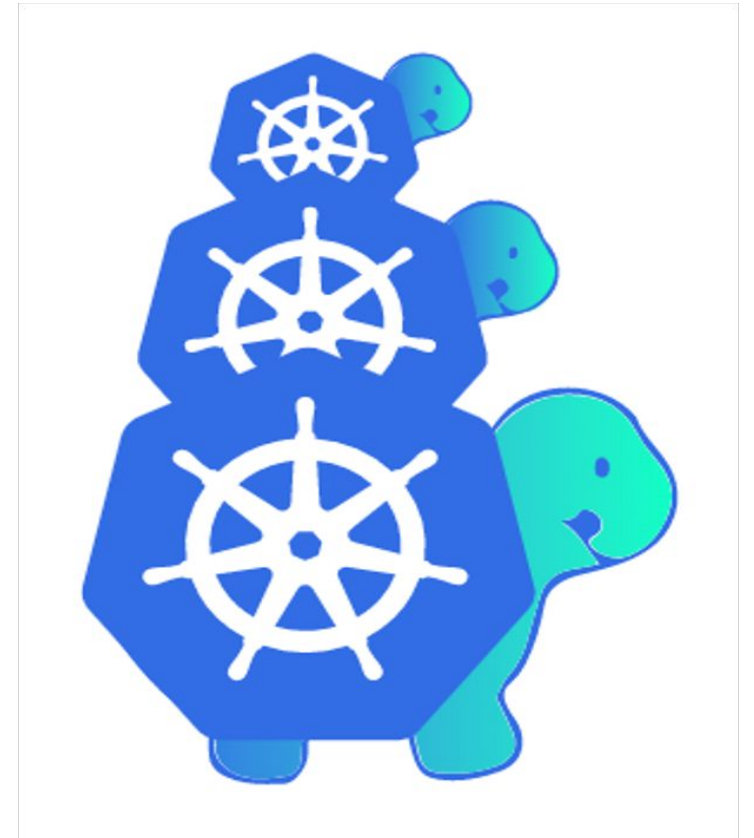
Cluster API Glossary

Cluster API terminology

Management cluster is a Kubernetes cluster into which the Cluster API components have been installed.

Cluster API terminology

Management cluster is a Kubernetes cluster into which the Cluster API components have been installed.

Workload cluster is a Kubernetes cluster that is created and managed by the Cluster API components in the management cluster.

Cluster API terminology

Management cluster is a Kubernetes cluster into which the Cluster API components have been installed.

Workload cluster is a Kubernetes cluster that is created and managed by the Cluster API components in the management cluster.

Self-hosted Management cluster is a management cluster that manages itself.

Cluster API terminology

Management cluster is a Kubernetes cluster into which the Cluster API components have been installed.

Workload cluster is a Kubernetes cluster that is created and managed by the Cluster API components in the management cluster.
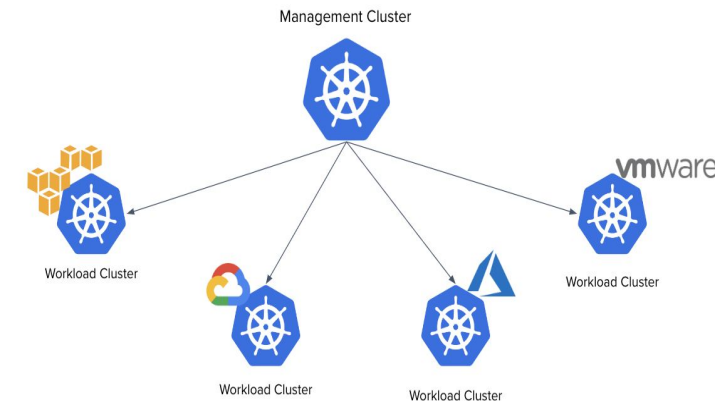
Self-hosted Management cluster is a management cluster that manages itself.
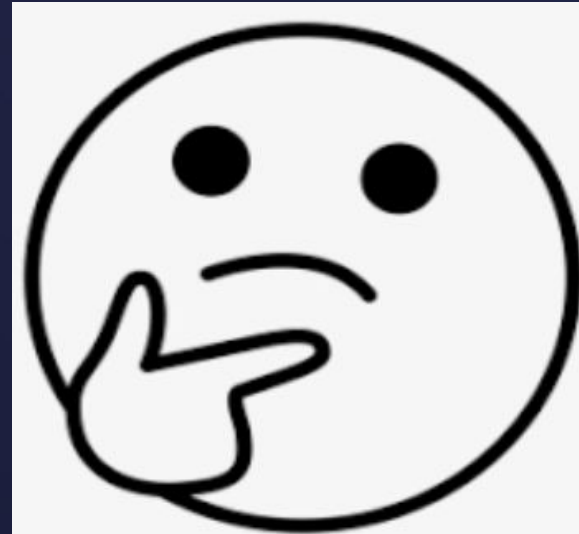

Provider: Cluster API uses the concept of a provider to interact with underlying IaaS platforms, such as AWS, Azure, vSphere, etc. Providers have abbreviations like CAPA, CAPZ and CAPV.

Cluster API terminology

Cluster lifecycle management includes:

- Creating new Kubernetes clusters.
- Managing the underlying infrastructure.
- Scaling up and down the number of nodes in the cluster.
- Upgrading clusters to another Kubernetes version.
- Deleting clusters, including the underlying infrastructure.

Overview of Kubernetes reconciliation loop and CRDs

- At the core of Kubernetes is a control loop (or reconciliation loop) that is responsible for reconciling the desired state and the actual state.

How does it work?

Overview of Kubernetes reconciliation loop and CRDs

- At the core of Kubernetes is a control loop (or reconciliation loop) that is responsible for reconciling the desired state and the actual state.

- *"Reconciling desired state and actual state"* **simply means changing actual state to converge to the desired state.**

Overview of Kubernetes reconciliation loop and CRDs

- At the core of Kubernetes is a control loop (or reconciliation loop) that is responsible for reconciling the desired state and the actual state.

- "*Reconciling desired state and actual state*" simply means making the actual state look like the desired state.

- **The control loop is implemented by a controller and is fundamental to how almost everything works in Kubernetes.**

Overview of Kubernetes reconciliation loop and CRDs

- **To add new kinds of objects to Kubernetes, we use a Custom Resource Definition (CRD).**

Overview of Kubernetes reconciliation loop and CRDs

- To add new kinds of objects to Kubernetes, we use a Custom Resource Definition (CRD).

- **Each CRD needs a controller to implement the corresponding control loop.**
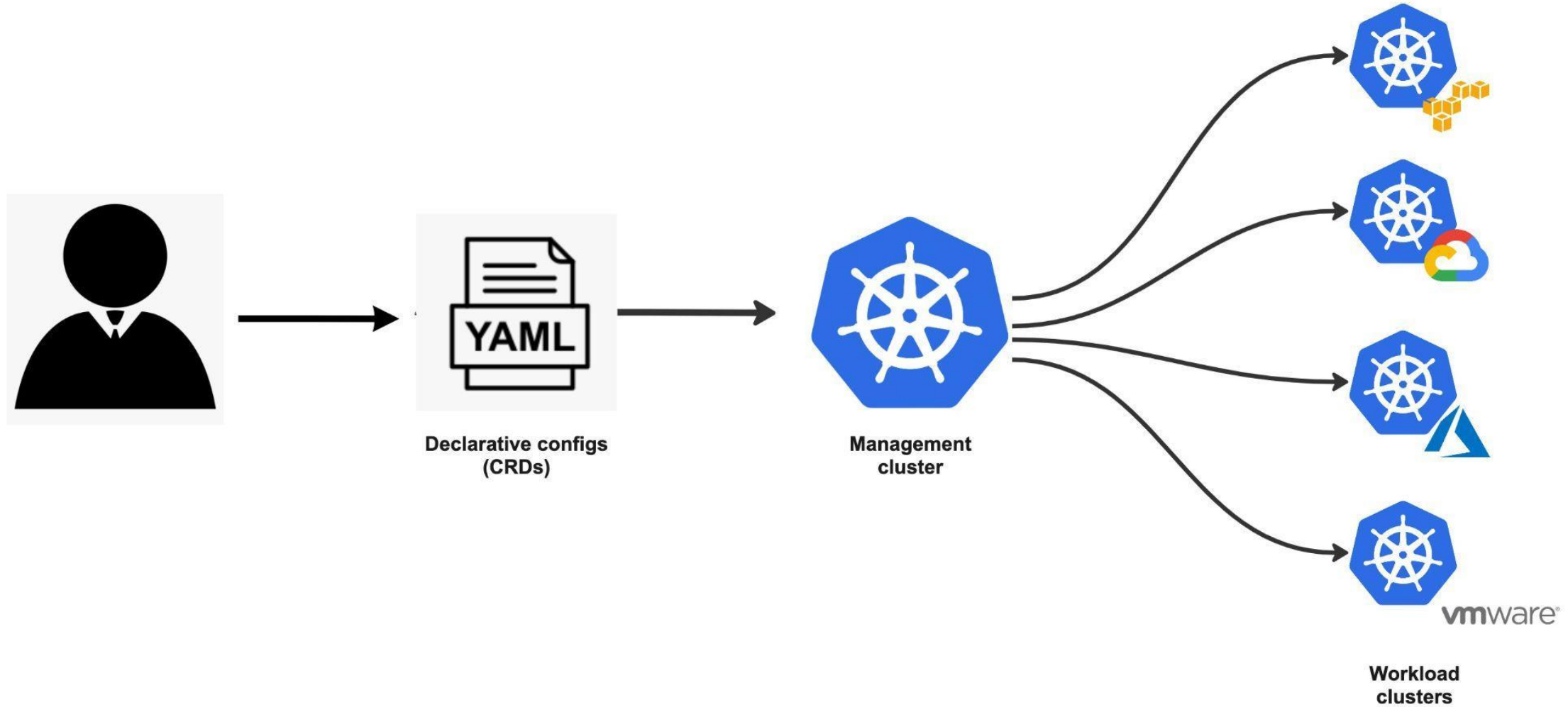
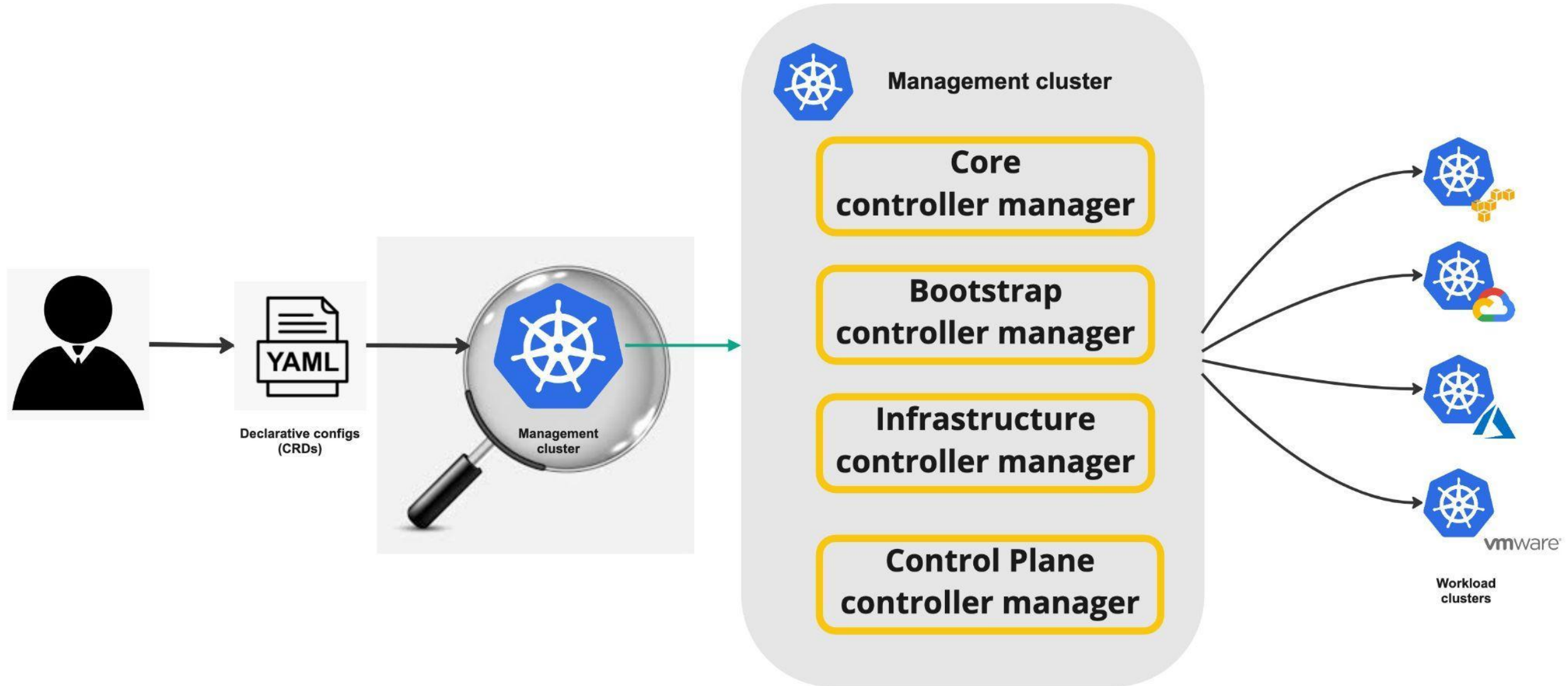**Overview of Kubernetes reconciliation loop and CRDs**

- To add new kinds of objects to Kubernetes, we use a Custom Resource Definition (CRD).

- There needs to be a controller to implement the control loop for each CRD.

- **Cluster API uses CRDs and controllers to extend Kubernetes to manage the Cluster lifecycle.**
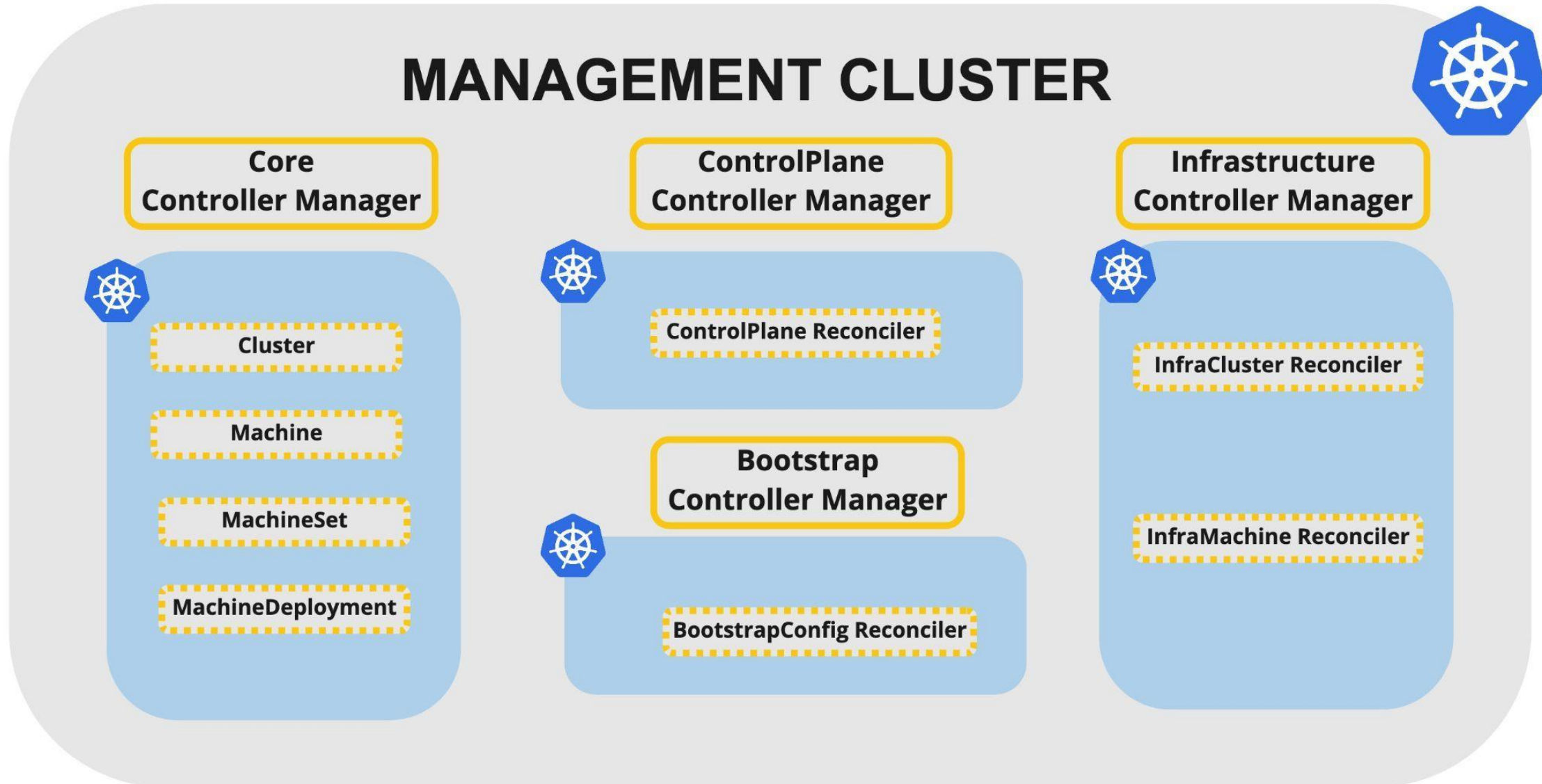
How does it work?

Declarative configs
(CRDs)

Management
cluster

Workload
clusters

How does it work?

How does it work?

MANAGEMENT CLUSTER

Core Controller Manager
- Cluster
- Machine
- MachineSet
- MachineDeployment

ControlPlane Controller Manager
- ControlPlane Reconciler

Bootstrap Controller Manager
- BootstrapConfig Reconciler

Infrastructure Controller Manager
- InfraCluster Reconciler
- InfraMachine Reconciler

- /test pull-cluster-api-e2e-full-main

CRDs and Spec

CRDs and Spec
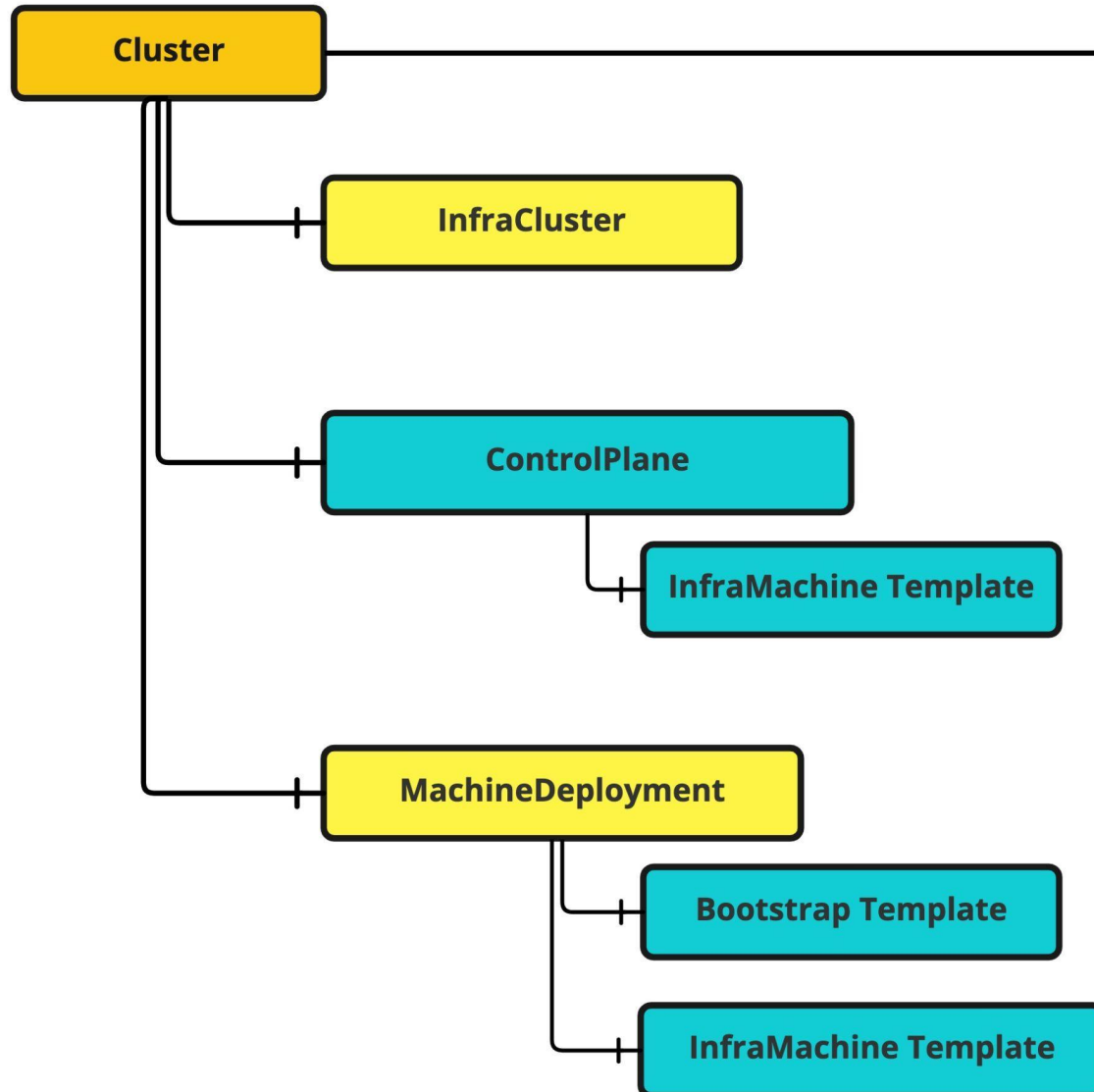
Cluster

Cluster-wide configuration.

Generic networking concepts like Pod and Service IP ranges or DNS domain.

```yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: example
  namespace: default
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 192.168.0.0/16
  controlPlaneRef:
    . . .
  infrastructureRef:
    . . .
  topology: # This is an experimental feature
    class: example-class
    version: v1.23.0
    controlPlane:
      replicas: 1
    workers:
      machineDeployments:
      - class: "example-worker"
        name: "md-0"
        replicas: 1
```
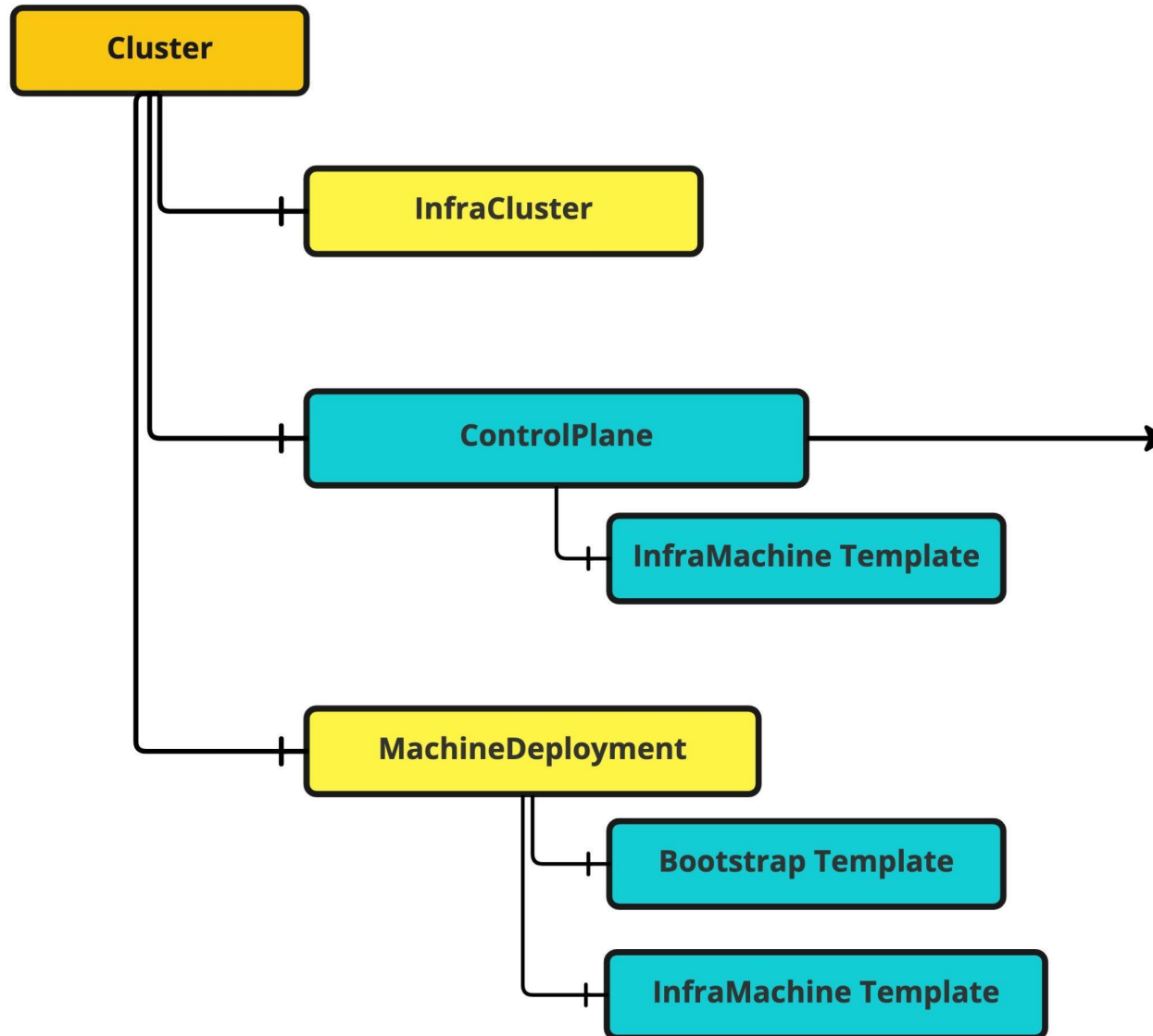
CRDs and Spec
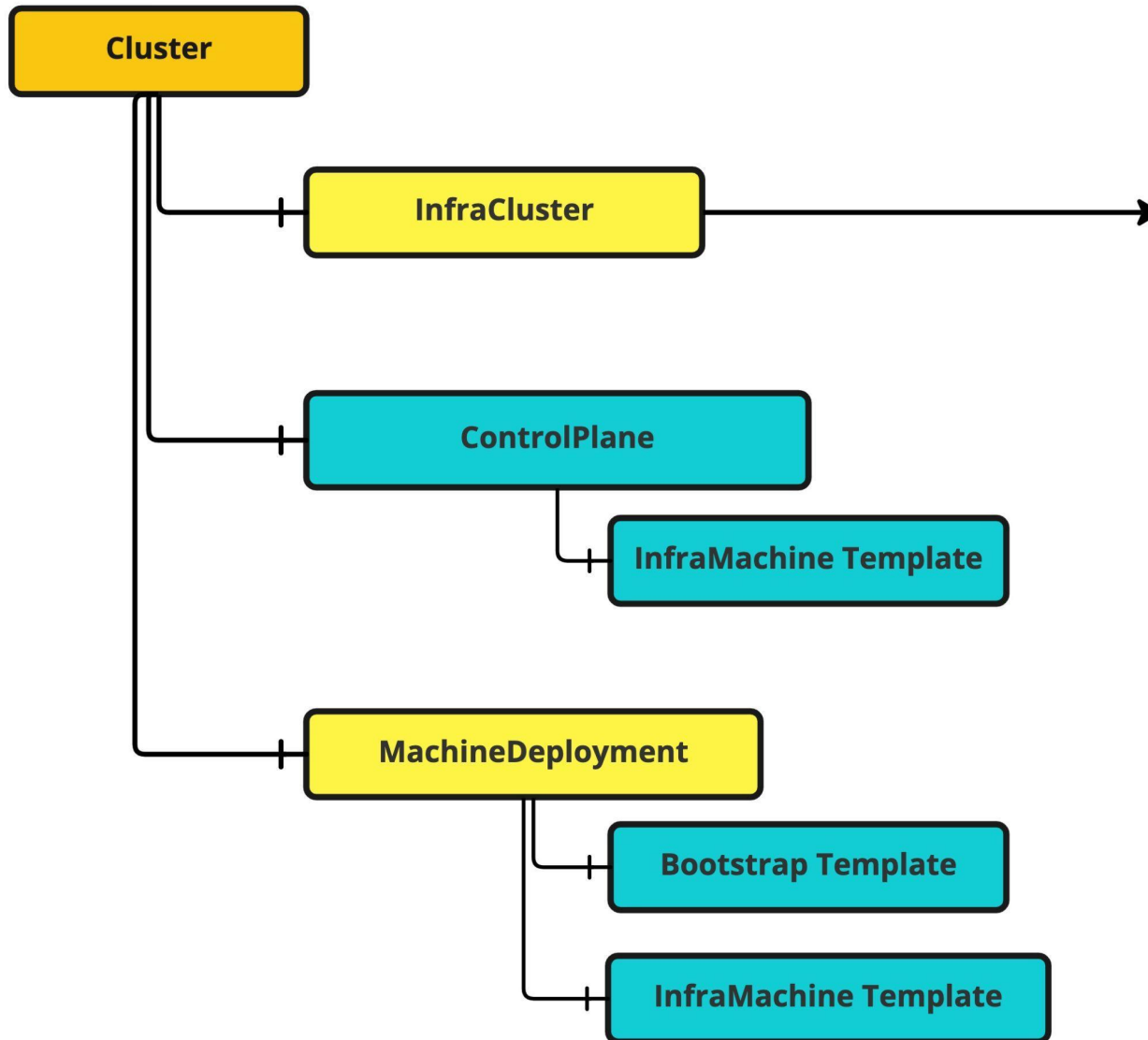
InfraCluster

Example: AWSCluster, AzureCluster, DockerCluster, etc.

Provider-specific cluster configuration.

```yaml
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: AWSCluster
metadata:
  name: example
  namespace: default
spec:
  region: us-west-2
  sshKeyName: example
```

Cluster

InfraCluster

ControlPlane

InfraMachine Template

MachineDeployment

Bootstrap Template

InfraMachine Template

CRDs and Spec

ControlPlane

Example: KubeadmControlPlane, EKS, AKS, etc.

Replicas for the desired number of control plane Machines.

Version for the target Kubernetes version of the control plane Machines.

```
apiVersion: controlplane.cluster.x-k8s.io/v1beta1
kind: KubeadmControlPlane
metadata:
  name: example-control-plane
  namespace: default
spec:
  kubeadmConfigSpec:
    clusterConfiguration:
      . . .
    initConfiguration:
      . . .
    joinConfiguration:
      . . .
  machineTemplate:
    infrastructureRef:
      apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
      kind: AWSMachineTemplate
      name: example-control-plane
  replicas: 1
  version: v1.23.0
```

CRDs and Spec
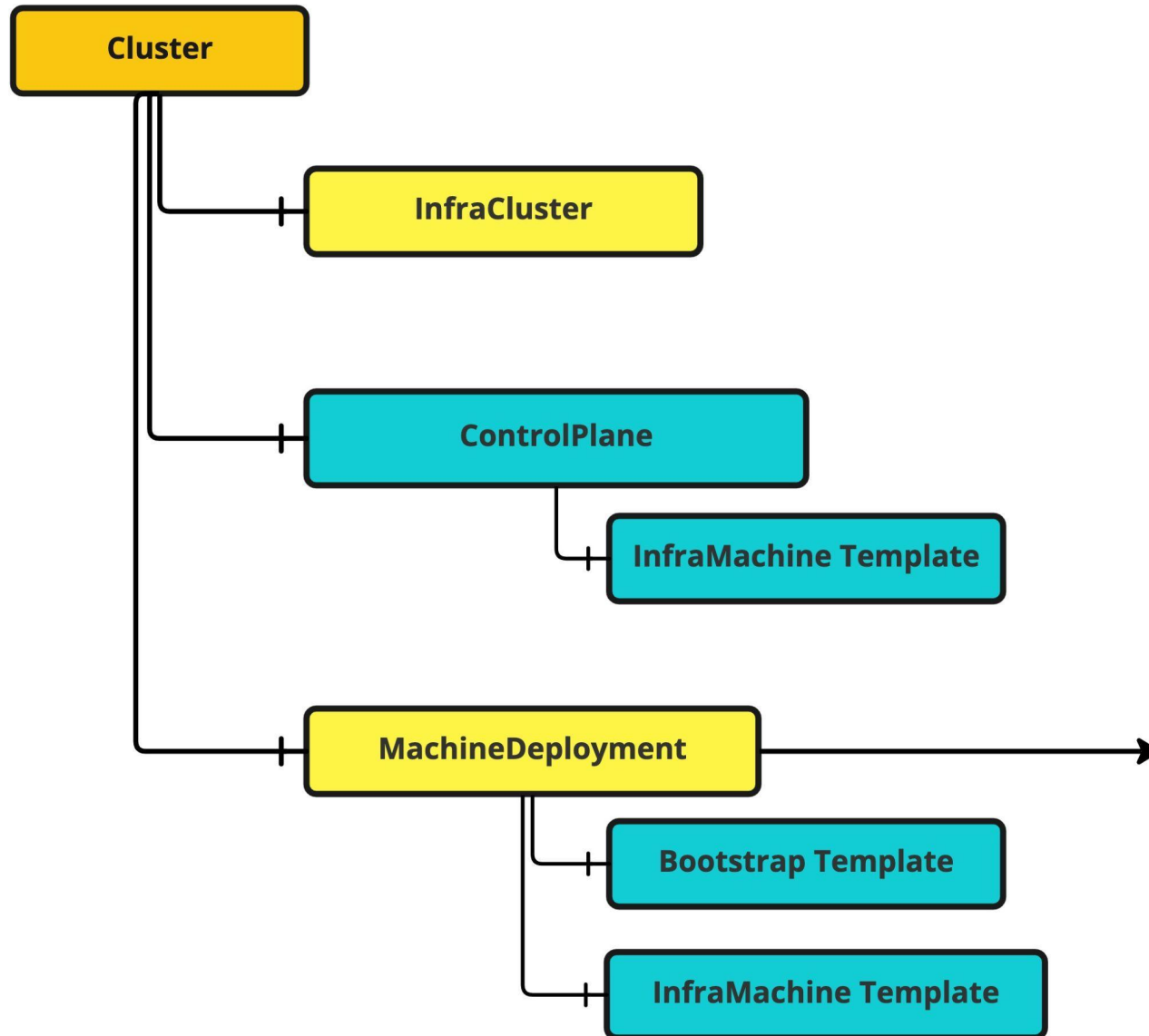
MachineDeployment

Declarative management of worker nodes of the Kubernetes cluster.

**Replicas** for the desired number of Machines.

Version for the target Kubernetes version of the Machines.

```yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineDeployment
metadata:
  name: example-md-0
  namespace: default
spec:
  clusterName: example
  replicas: 0
  selector:
    matchLabels: null
  template:
    spec:
      bootstrap:
        configRef:
          apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
          kind: KubeadmConfigTemplate
          name: example-md-0
      clusterName: example
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: AWSMachineTemplate
        name: example-md-0
      version: v1.23.0
```

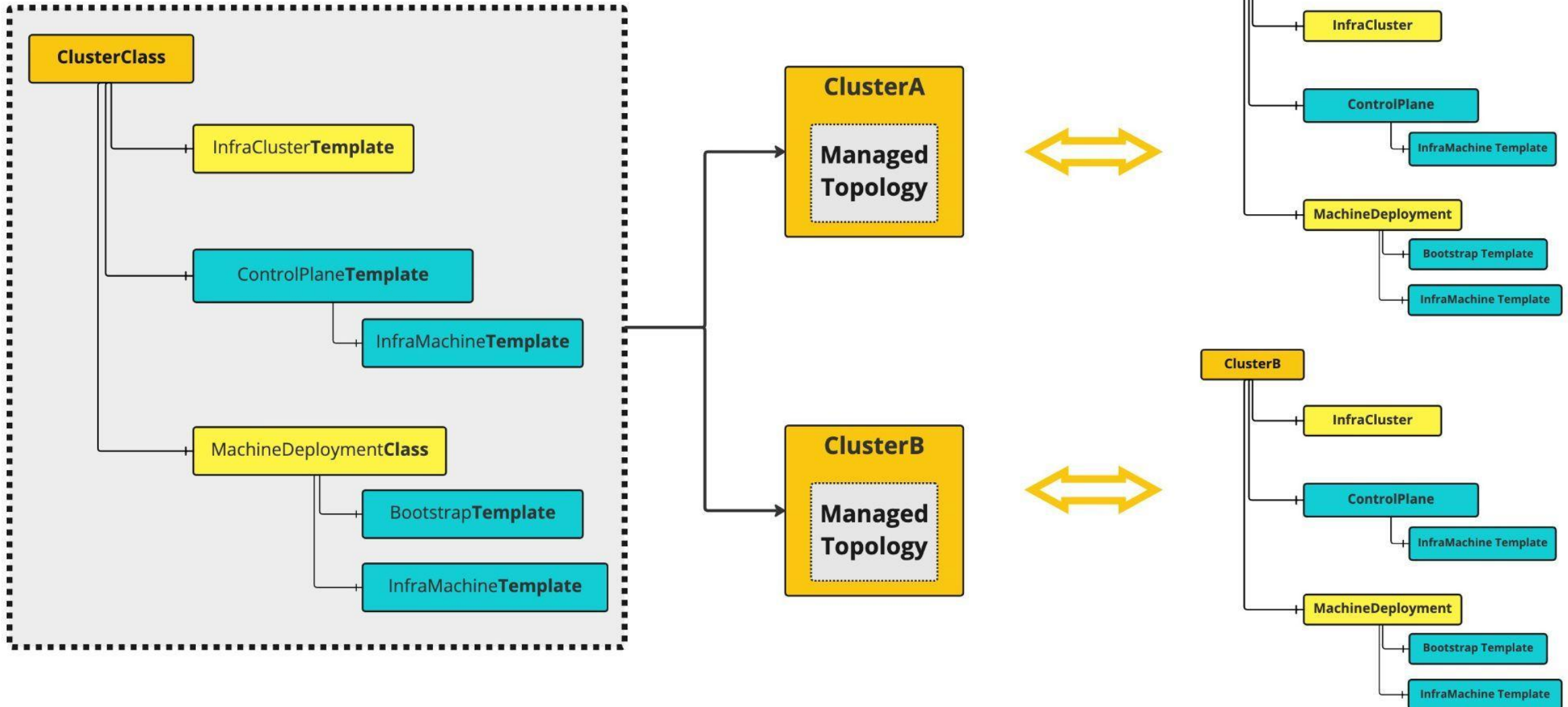# ClusterClass and Managed Topologies

ClusterClass is a collection of templates that define a topology (control plane and MachineDeployments) to be used to continuously reconcile one or more Clusters.
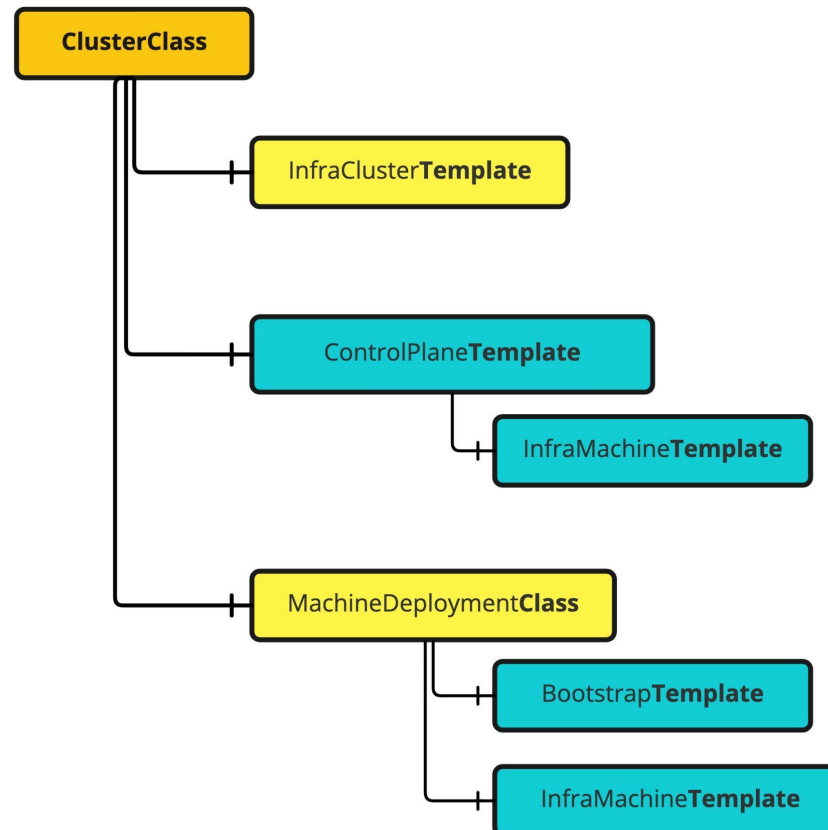
ClusterClass and Managed Topologies
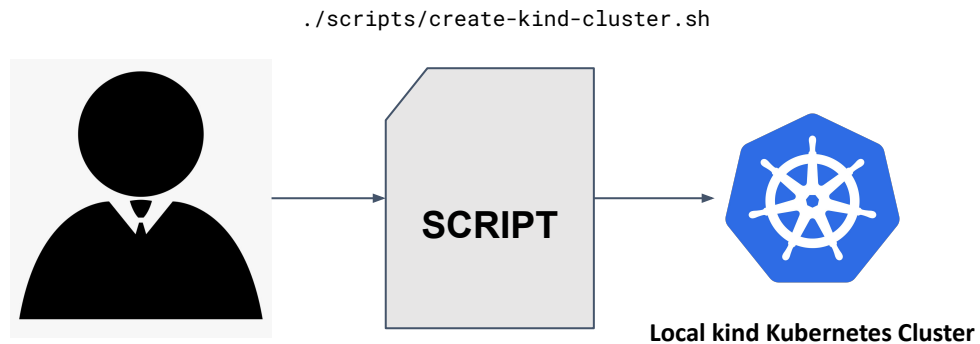
Hands on

Hands on: Prerequisites

Instructions can be found in [Prerequisites](#) at **github.com/ykakarap/kubecon-na-22-capi-lab**

- Install Docker, kubectl, kind, clusterctl and helm

- Clone the tutorial repository

- Pre-download container images

- Verification

Find the repo at:
github.com/ykakarap/kubecon-na-22-capi-lab
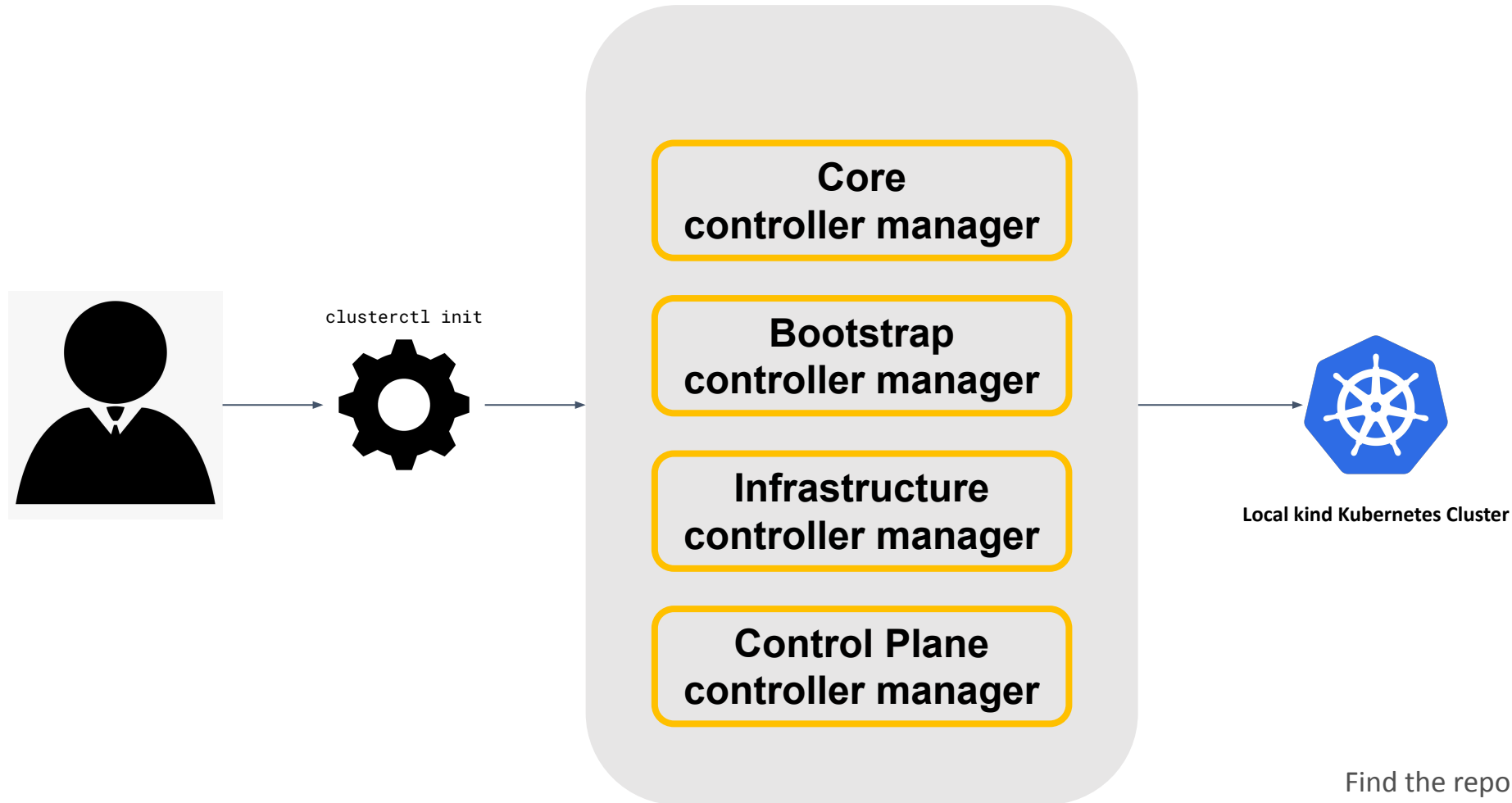
Hands on: Create a local cluster

Step 1: Use kind to create a local Kubernetes cluster.

`./scripts/create-kind-cluster.sh`

SCRIPT

Local kind Kubernetes Cluster

Find the repo at:
github.com/ykakarap/kubecon-na-22-capi-lab

Hands on: Create a management cluster

Step 2: Use clusterctl to install the management cluster components on your local cluster.

clusterctl init

Core controller manager

Bootstrap controller manager

Infrastructure controller manager

Control Plane controller manager

Local kind Kubernetes Cluster

Find the repo at:
github.com/ykakarap/kubecon-na-22-capi-lab

Hands on: Your first workload cluster

Step 3: Create your first workload cluster using kubectl apply.

> kubectl apply -f
docker-cluster-one.yaml

YAML

Management cluster

**Core
controller manager**

**Bootstrap
controller manager**

**Infrastructure
controller manager**

**Control Plane
controller manager**

Your first workload cluster

Control
Plane
Node 1

v1.24.6

Worker
Node 1

v1.24.6
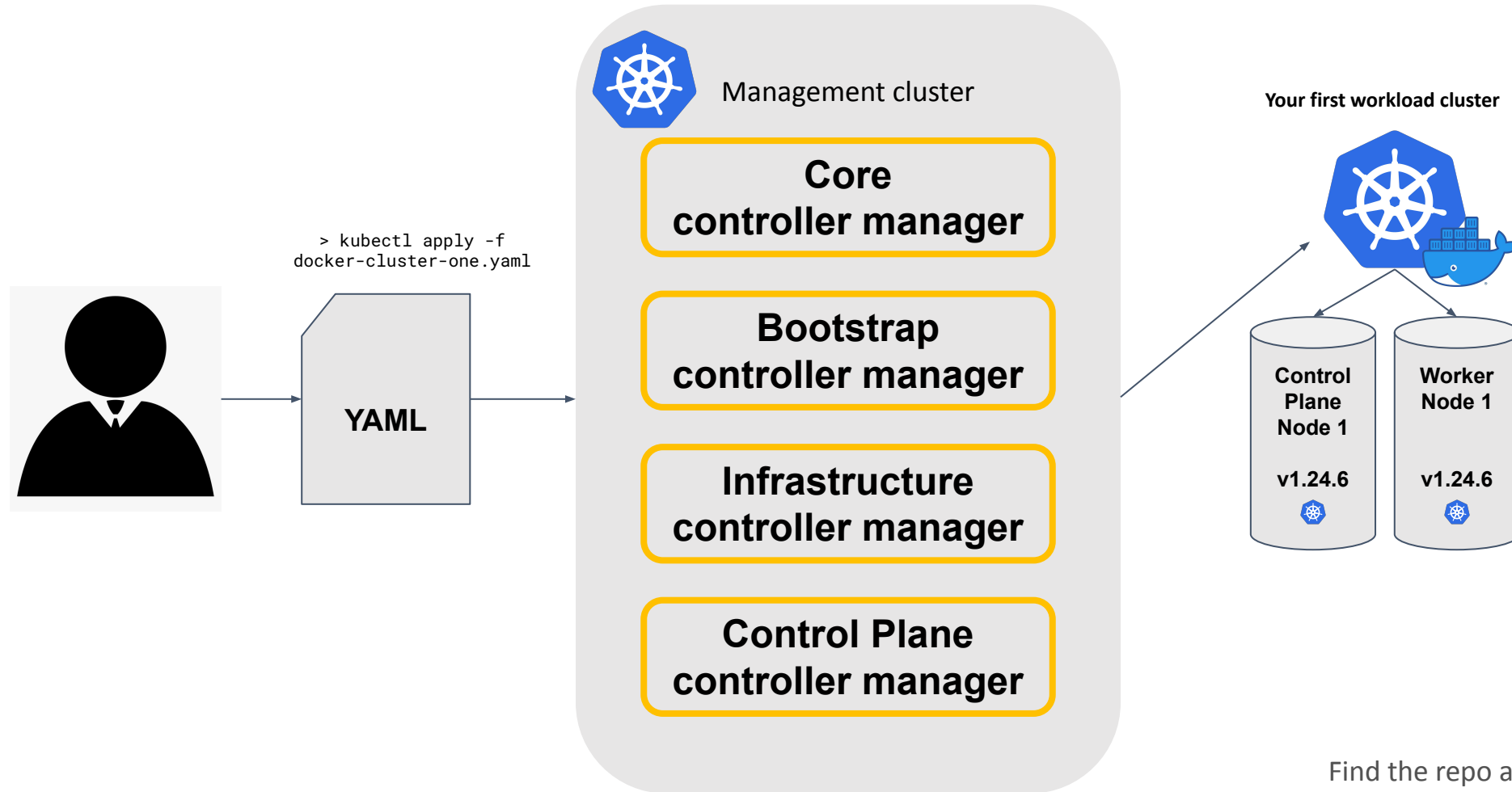
Find the repo at:
github.com/ykakarap/kubecon-na-22-capi-lab

Hands on: Additional topics

More Cluster API operations can be found at **github.com/ykakarap/kubecon-na-22-capi-lab**

- Deploying the Cluster API visualizer

- Changing the Cluster Topology

- MachineHealthCheck and Machine remediation

- Upgrading to another Kubernetes version

- Cluster lifecycle hooks

- Self-hosted management cluster

- Deleting a cluster & cleaning up Cluster API

Find the repo at:
github.com/ykakarap/kubecon-na-22-capi-lab