DevOps

Caltech | **Center for Technology & Management Education**

Post Graduate Program in DevOps

Source: https://kubernetes.io/docs/

simplilearn

DevOps

Caltech | Center for Technology & Management Education

Certified Kubernetes Administrator

simplilearn

Core Concepts

simplilearn

Learning Objectives

By the end of this lesson, you will be able to:

- Discuss Kubernetes

- State what Etcd, Controller, Scheduler, and Kubelet are

- Discuss kube-proxy, Pods, ReplicaSet, and Deployment

- List typical use cases for Deployment

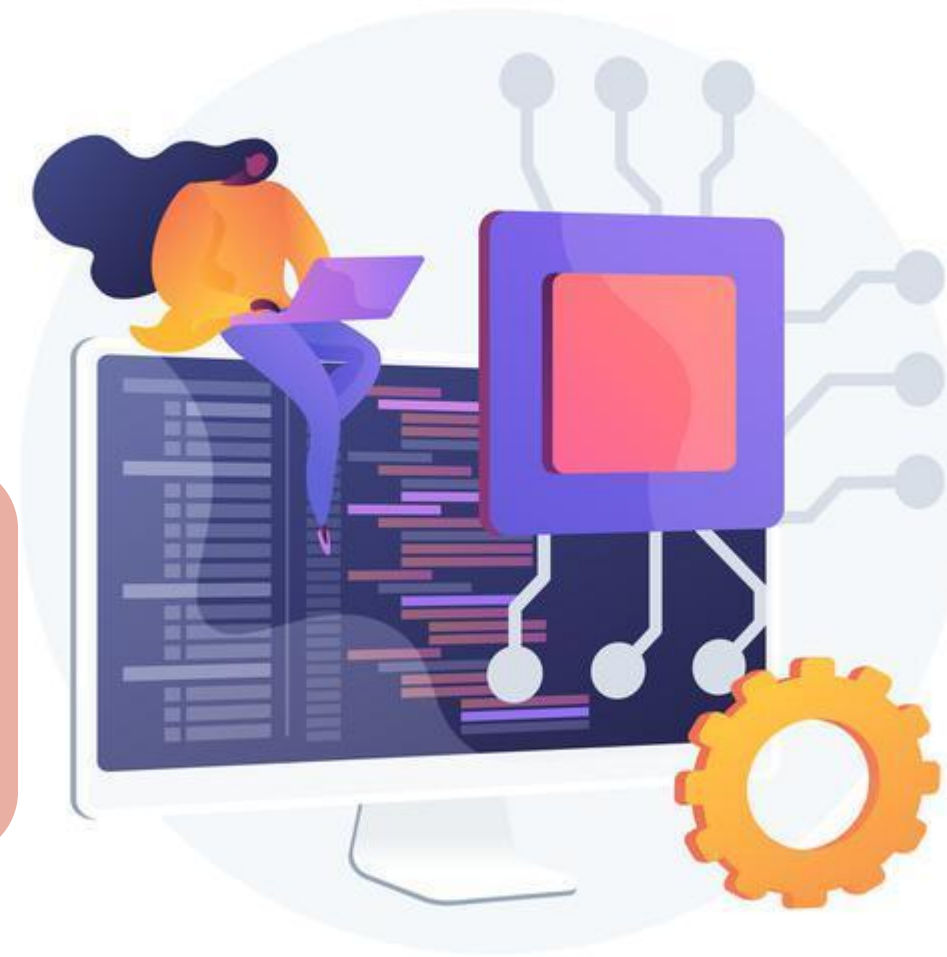- Present an overview of Containers and Policies

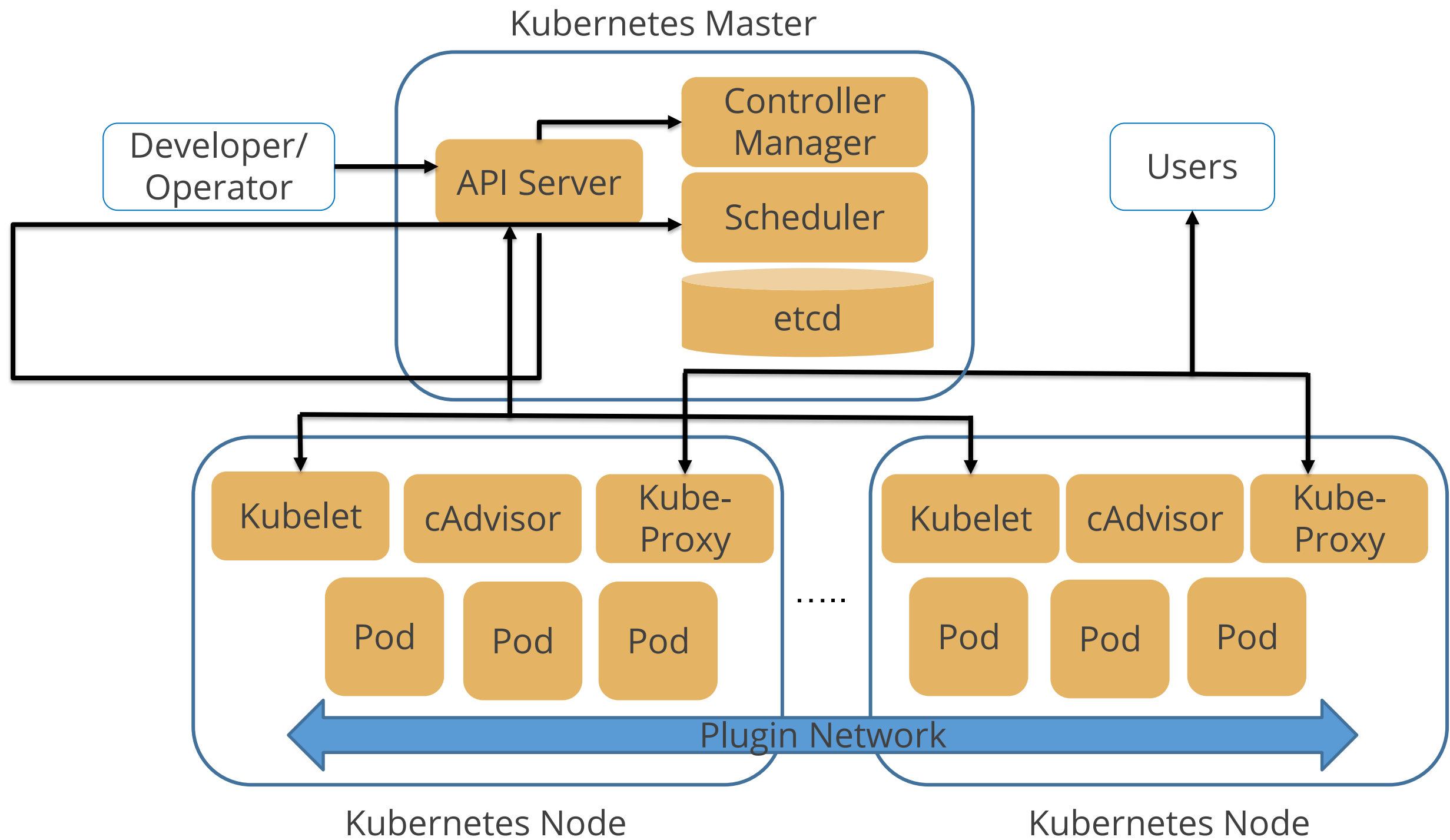# Overview

Kubernetes Design Principles

Kubernetes defines a set of building blocks. The blocks collectively provide mechanisms that deploy, maintain, and scale applications.

The platform exerts control over compute and storage resources by defining resources

Kubernetes is coupled and extended to meet different workloads and scheduling situations

Kubernetes Architecture

Kubernetes Master

Developer/Operator

API Server

Controller Manager

Scheduler

etcd

Users

Kubernetes Node

Kubelet

cAdvisor

Kube-Proxy

Pod

Pod

Pod

.....

Kubernetes Node

Kubelet

cAdvisor

Kube-Proxy

Pod

Pod

Pod

Plugin Network

Caltech | Center for Technology & Management Education

simplilearn

Kubernetes Cluster

Control Plane

Kube-apiserver

Kube-scheduler

Kube-controller-manager

etcd

Compute Machine

Kubelet

Kube-proxy

Container runtime

Pod

Containers

Persistent Storage

Container Registry

Underlying Infrastructure

Physical

Virtual

Private

Public

Hybrid

Caltech | Center for Technology & Management Education

simplilearn
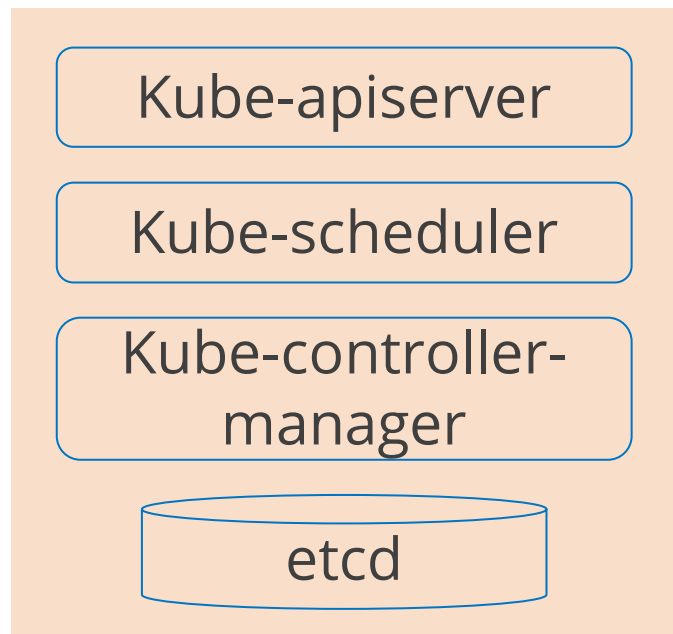
Control Plane

The Control Plane is in constant contact with the compute machines and ensures that the Containers are running in necessary resources.

Kube-apiserver

Kube-scheduler

Kube-controller-manager

etcd

**Kube-apiserver** handles internal and external requests

Kube-Scheduler schedules the Pod to an appropriate node

Kube-Controller Manager helps run the cluster

etcd stores configuration data and information about the state of cluster lives

Caltech | Center for Technology & Management Education

simplilearn

Compute Machine

A Kubernetes Cluster requires at least one compute node. Pods are scheduled and arranged to run on Nodes.

Kubelet

Kube-proxy

Container runtime

Pod
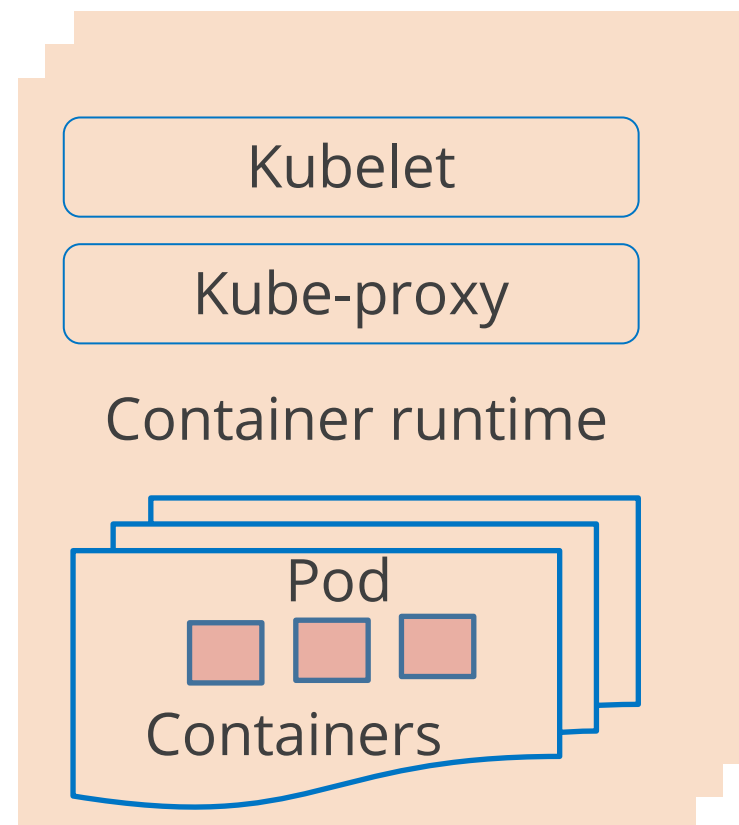
Containers

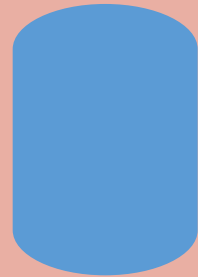**Kubelet** communicates with the control plane

Kube-proxy is a network proxy that facilitates Kubernetes network services

Container runtime manages and runs the components required to run Containers
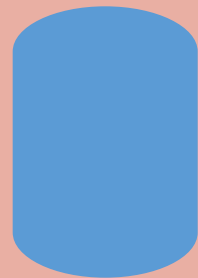
Pod represents a single instance of an application

Caltech | Center for Technology & Management Education   simplilearn

Other Components

Persistent Storage

Persistent Storage manages application data attached to a cluster

Container Registry

Container Registry stores the container images that the Kubernetes relies on

Features of Kubernetes

Storage

Secret and configuration management

Self-healing
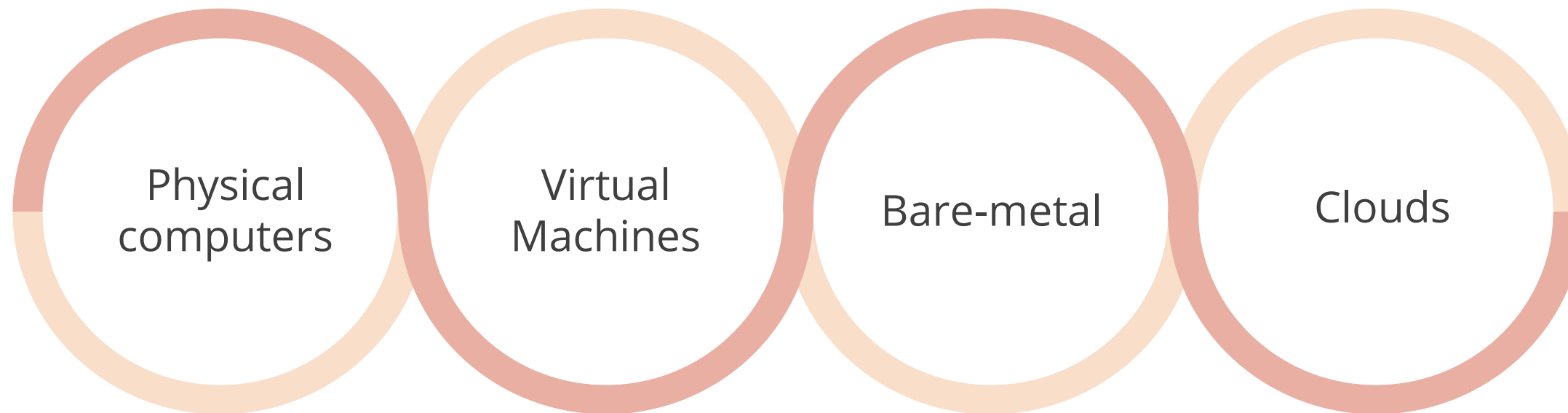
Automated rollouts and rollbacks
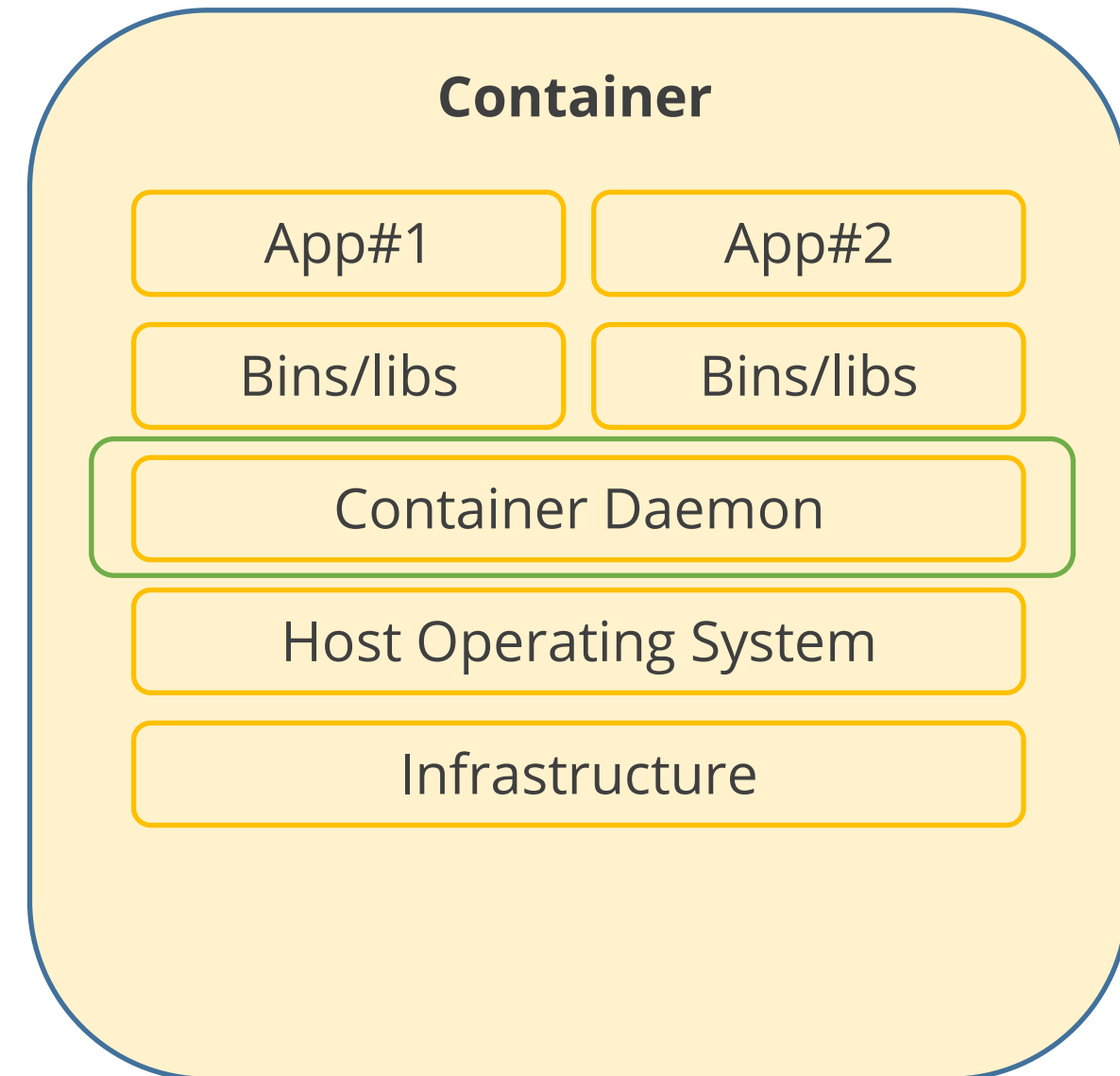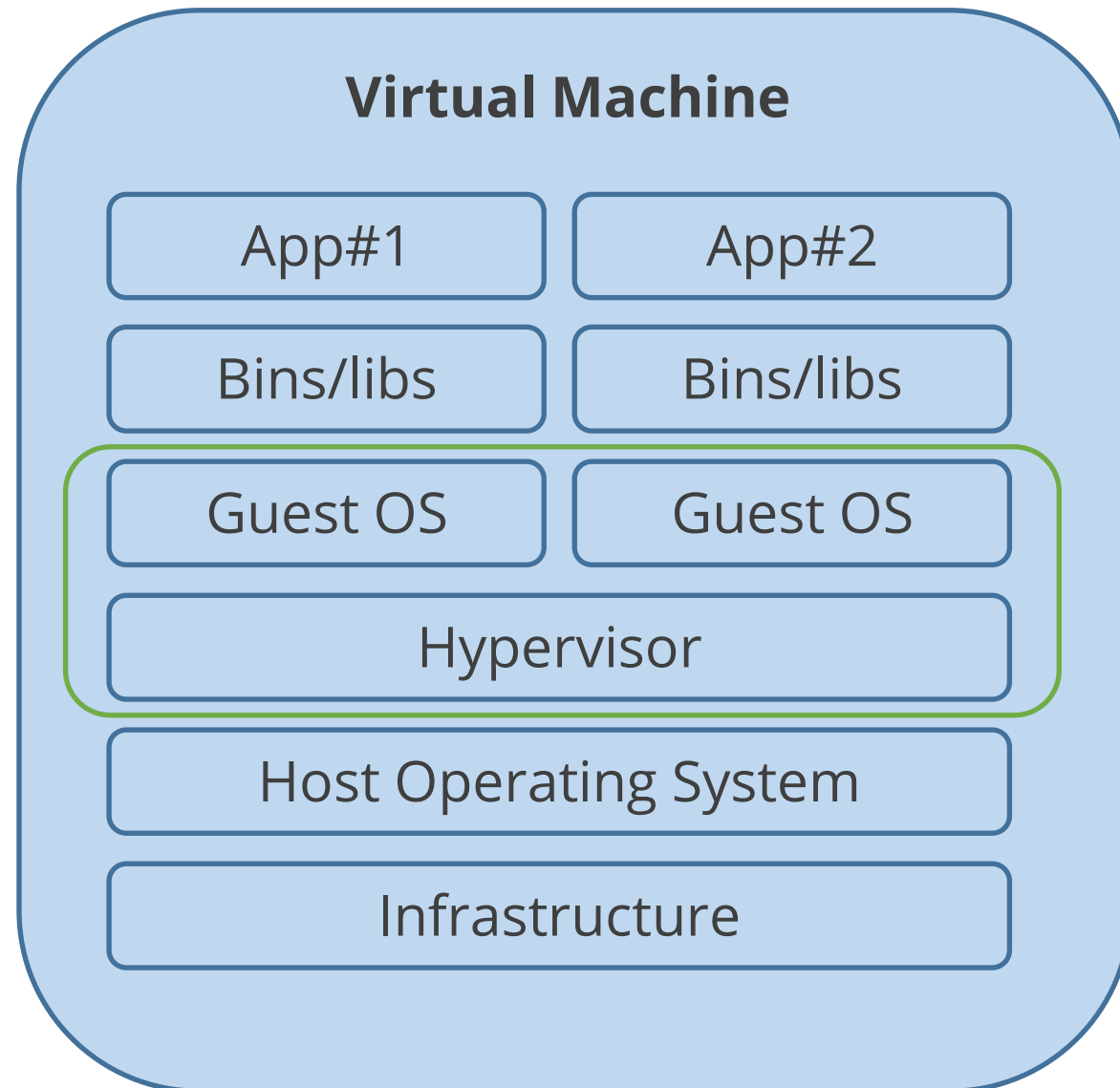
Service discovery and Load Balancing

Containers

Containers are a centralized or a standard unit of software application environment that is pre-packaged with code, along with its dependencies.

> Pre-packaging allows the software to run fast and reliably from one computing environment to another. Containers can run on any compatible infrastructure.

Docker Containers run in four modes:

Physical computers   Virtual Machines   Bare-metal   Clouds

Containers versus Virtual Machines

Virtual Machine

App#1	App#2
Bins/libs	Bins/libs
Guest OS	Guest OS
Hypervisor	
Host Operating System	
Infrastructure	

Container

App#1	App#2
Bins/libs	Bins/libs
Container Daemon	
Host Operating System	
Infrastructure	

Benefits of Containers

Agile application creation and deployment

Continuous development, integration, and deployment

Dev and Ops separation of concerns

Loosely coupled, distributed, elastic, and liberated micro-services

Application-centric management

Resource isolation and utilization

Cloud and OS distribution portability

Caltech | Center for Technology & Management Education

simplilearn

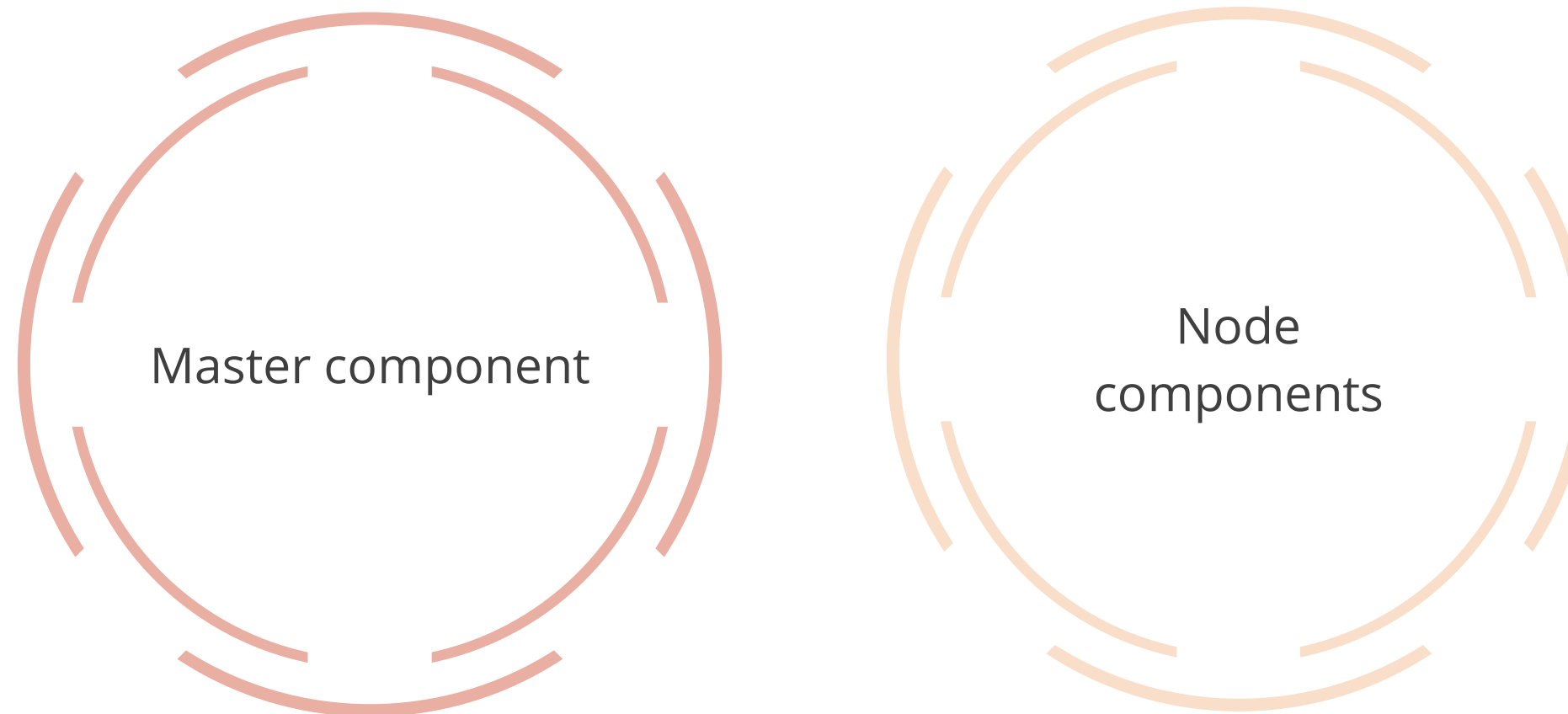# Kubernetes: Overview

Kubernetes Components
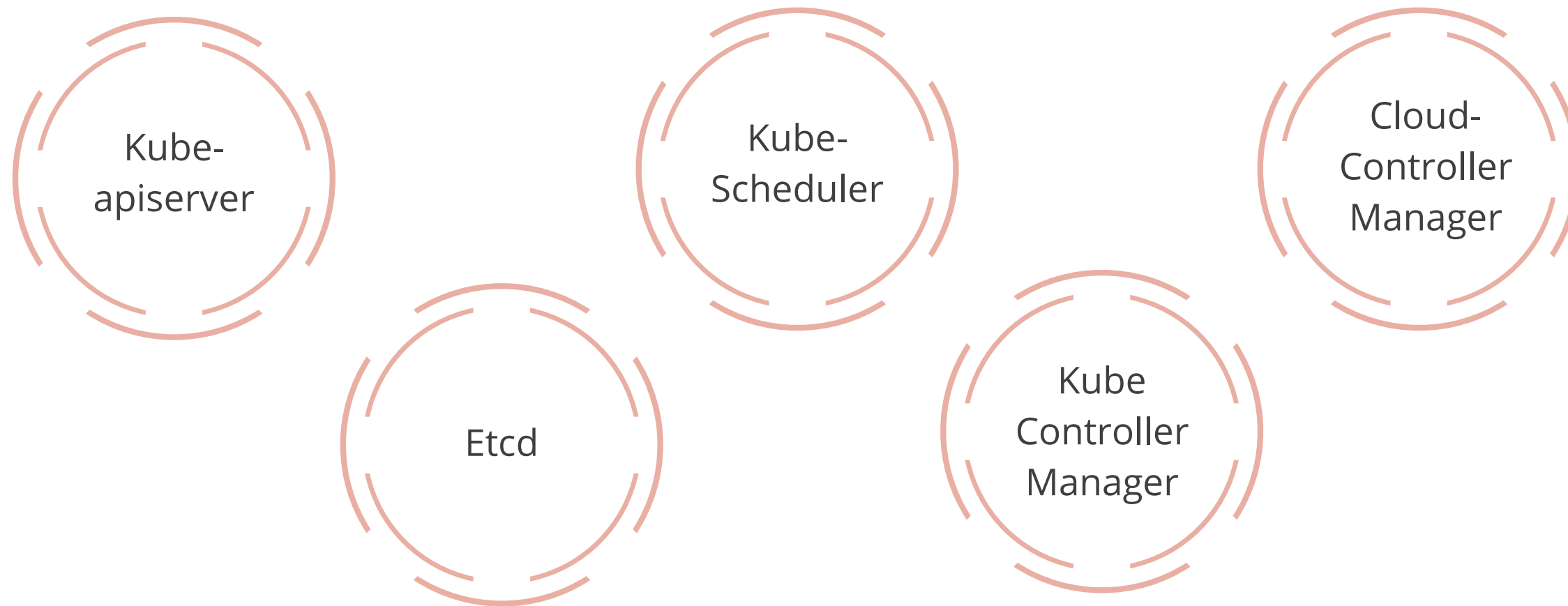
Kubernetes API

Kubernetes Objects

Kubernetes Components

A Kubernetes cluster consists of components that represent the Control Plane and a set of machines called Nodes. The components are divided into:

Master component

Node components

Master Components

Master components are designed to globally monitor the cluster and respond to cluster events.

Kube-apiserver

Etcd

Kube-Scheduler

Kube Controller Manager

Cloud-Controller Manager

Cloud-Controller Manager

The Cloud-Controller Manager helps to cluster into Cloud Provider's APIs and separates components that interact with the cloud platform.
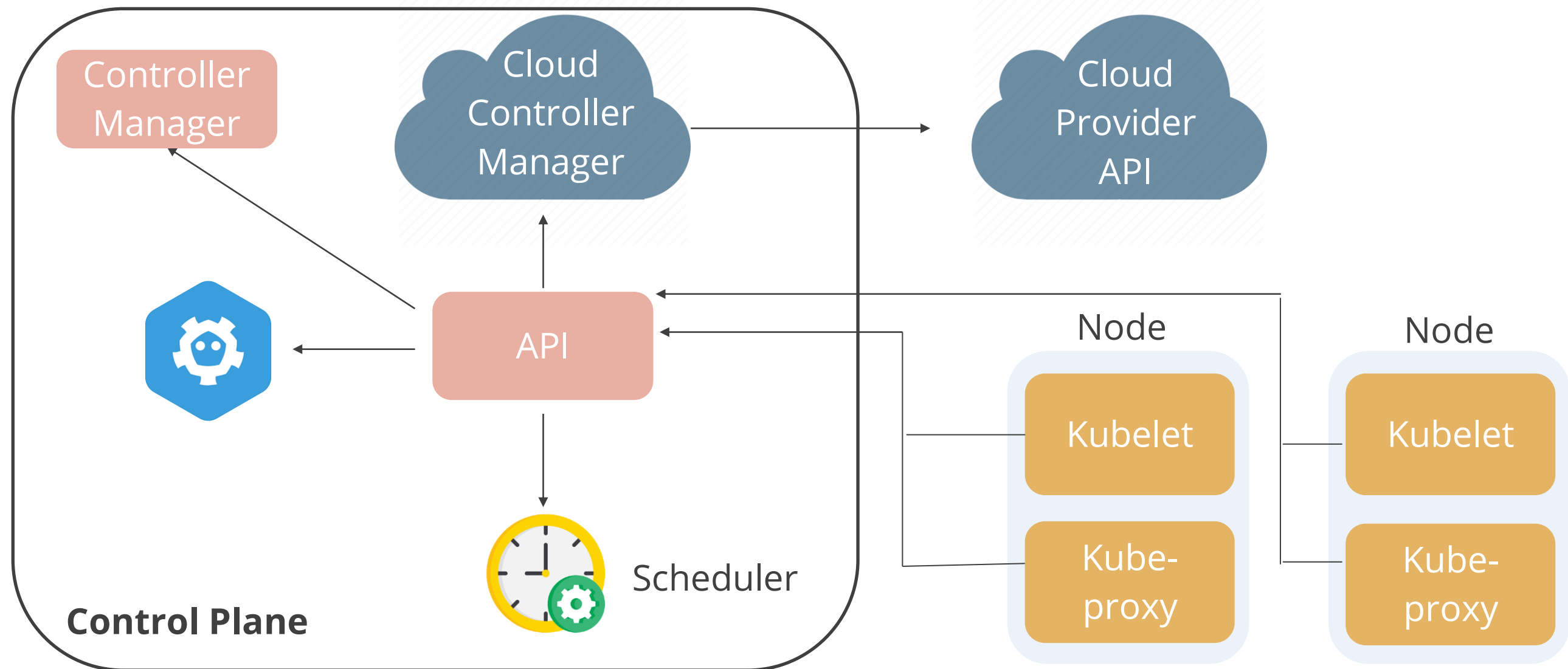
Kubernetes API

The Kubernetes API facilitates querying and manipulating the state of objects. The nucleus of the Control Plane in Kubernetes is the API server.

It helps to manipulate the state of API objects like:

Pods

Namespaces

ConfigMaps and Events

Kubernetes Objects

Kubernetes objects are persistent entities in the Kubernetes system.

These objects can describe:
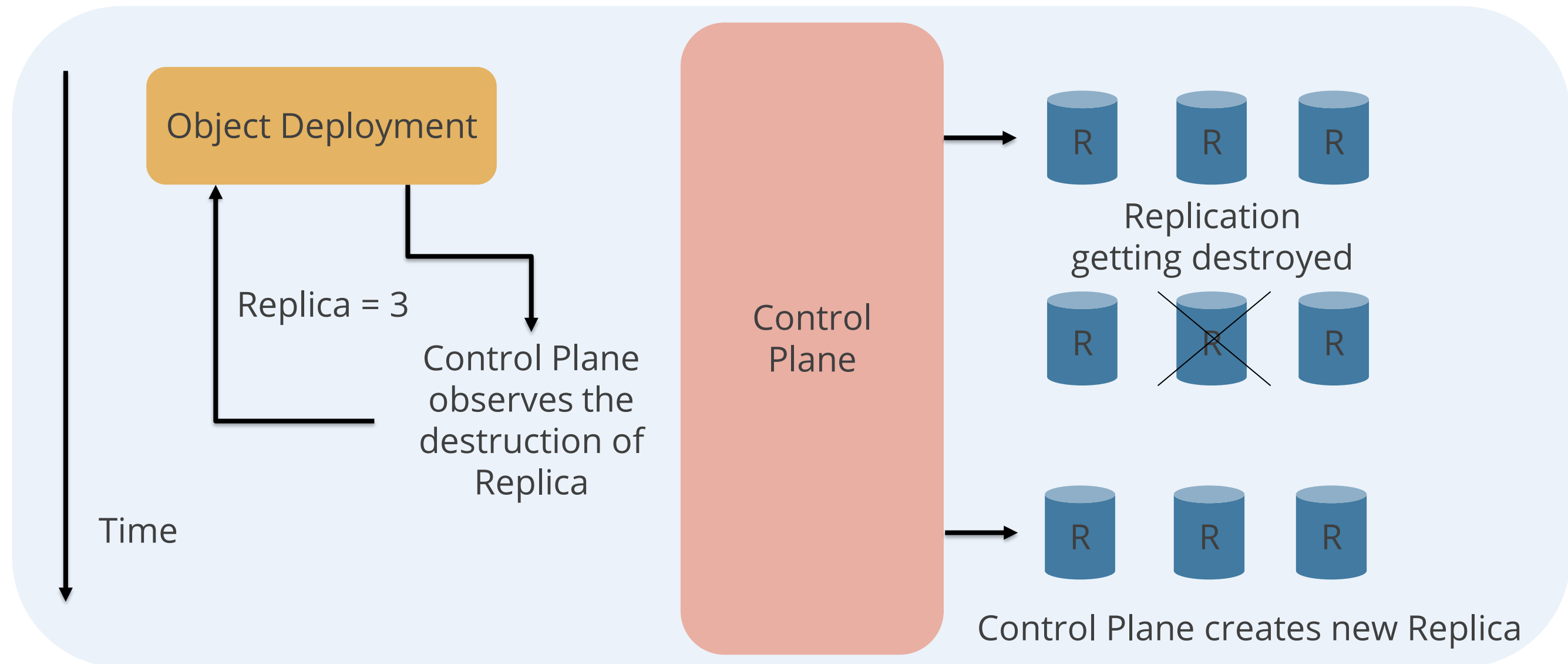
👉 What containerized applications are running

👉 The resources available to those applications

👉 The policies around how applications behave, such as restart policies, upgrades, and fault-tolerance

Object Fields

Every Kubernetes object includes two nested object fields that govern the object's configuration, namely, **spec** and object **status.**

Object Deployment

Replica = 3

Control Plane observes the destruction of Replica

Time

Control Plane

R R R

Replication getting destroyed

R R R

R R R

Control Plane creates new Replica

Caltech | Center for Technology & Management Education
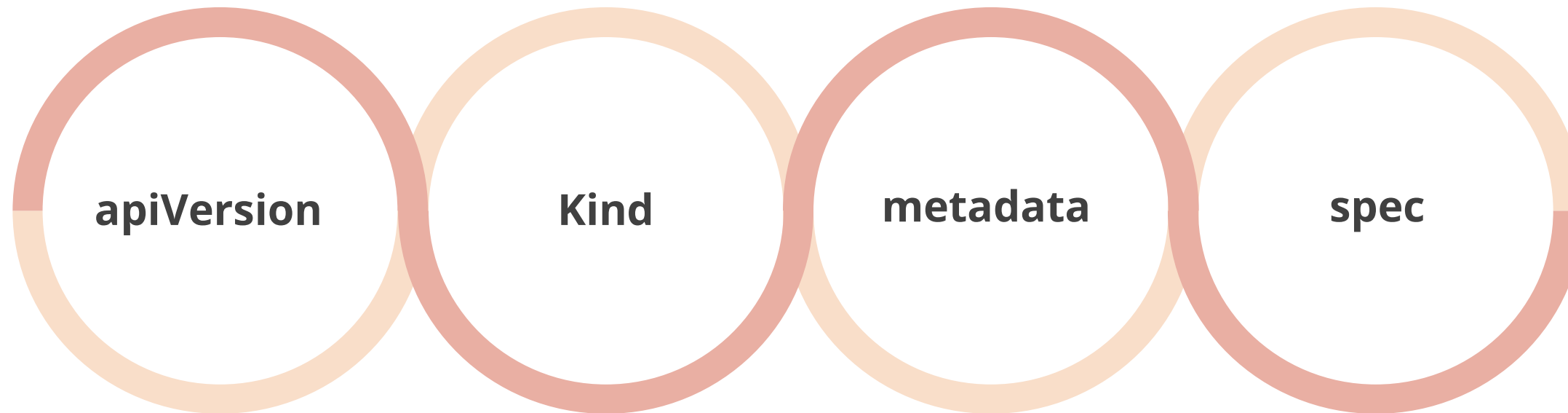
simplilearn

Describing Kubernetes Object

APIs used to create objects should include a JSON file. When making the API request, the information is converted to JSON.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells Deployment to run 2 Pods matching the
template
  template:
metadata:
    labels:
      app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx:1.14.2
      ports:
      - containerPort: 80
```

Required Fields in .yaml File

In the **.yaml** file, a set of values is needed for the Kubernetes object to be created for the following fields:
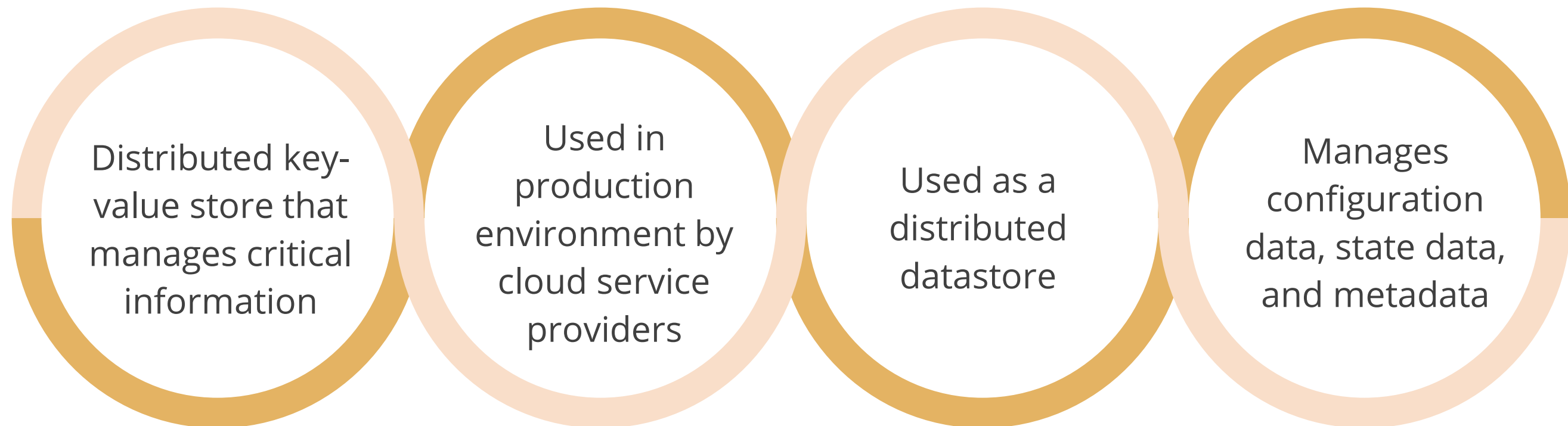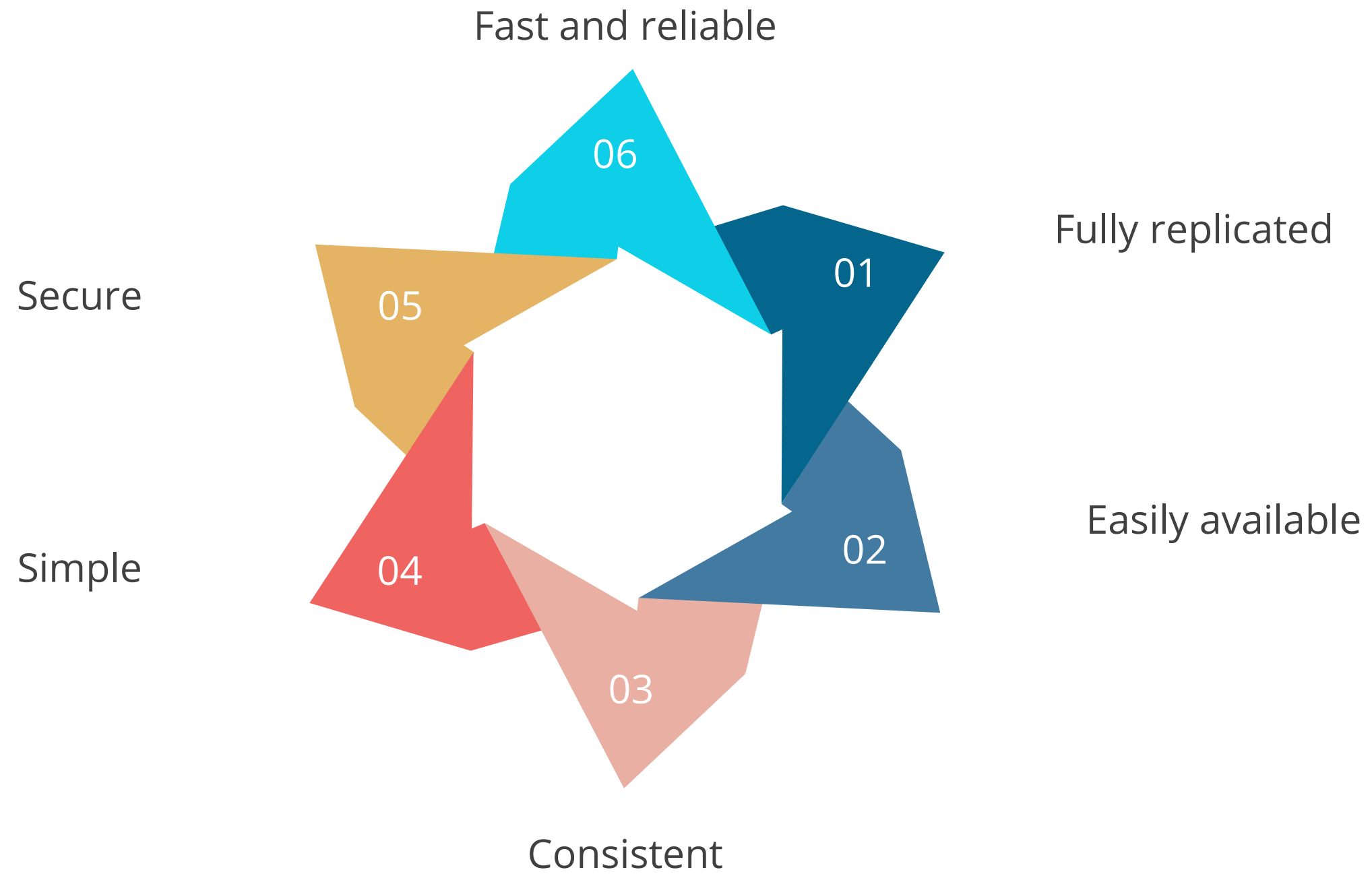
apiVersion   Kind   metadata   spec

Caltech | Center for Technology & Management Education

simplilearn

Etcd

Etcd

Etcd is an open-source distributed key-value store used to hold and manage critical information that distributed systems need to keep running.

Distributed key-value store that manages critical information

Used in production environment by cloud service providers

Used as a distributed datastore

Manages configuration data, state data, and metadata

Caltech | Center for Technology & Management Education

simpli·learn

Properties of Etcd

Fast and reliable

06

Fully replicated

01

Easily available

02

Consistent

03

Simple

04

Secure

05

Caltech | Center for Technology & Management Education

simplilearn

Working of Etcd

Defined through three key concepts (Raft-Based System):

Leaders
Handle all client requests that need consensus among clusters

Elections
Used when the leader is down for any reason
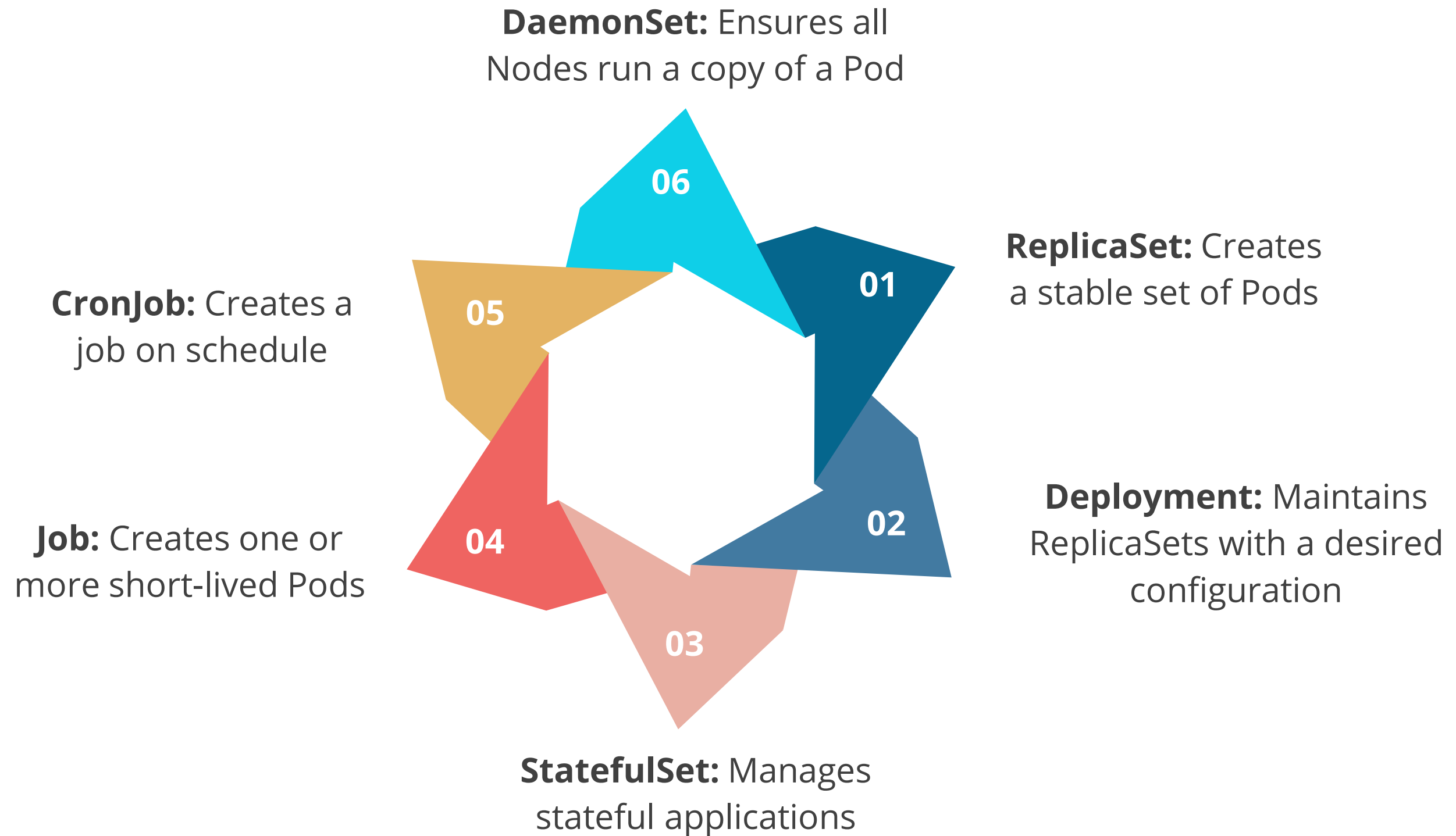
Terms
Appears for multiple candidates

Controller

Controllers in Kubernetes

Built-in Controllers manage state by interacting with the cluster API server. Job Controller is a good example of a Kubernetes built-in controller.

Caltech | Center for Technology & Management Education

simplilearn

Types of Controllers

DaemonSet: Ensures all Nodes run a copy of a Pod

06

ReplicaSet: Creates a stable set of Pods

01

CronJob: Creates a job on schedule

05

02

Deployment: Maintains ReplicaSets with a desired configuration

Job: Creates one or more short-lived Pods

04

03

StatefulSet: Manages stateful applications

Caltech | **Center for Technology & Management Education**

simpli**learn**

Controller Design

Kubernetes uses many Controllers, each of which manages a specific feature of cluster state.

A specific control loop uses one kind of resource as its desired state.

Kubernetes is designed to handle Controller failures.

Working of Controllers

Kubernetes comes with a set of built-in Controllers that run inside the Kube-controller-manager. These built-in Controllers provide important core behaviors.

To extend Kubernetes, Controllers in the system run outside the Control Plane.

Scheduler

Kubernetes Scheduler

Scheduling refers to ensuring that Pods are matched to Nodes so that a Kubelet can run them. A Scheduler watches for newly created Pods that are not assigned to any Nodes.

Pod:
Represents a set of running Containers in a cluster

Node:
Worker machine in Kubernetes
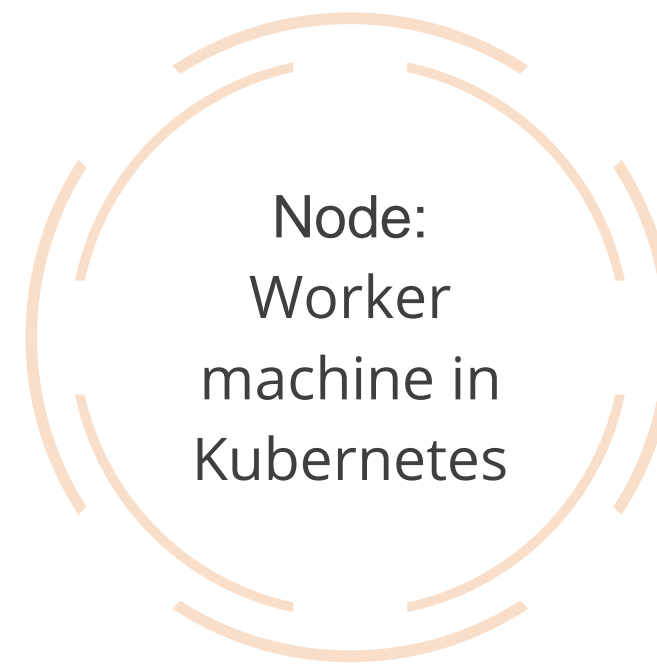
Kube-scheduler

A Kube-scheduler is the default scheduler of any Kubernetes system and runs as a part of the Control Plane.

Helps in writing scheduling component and using them

Finds workable or feasible Nodes for a Pod; runs a set of functions to score feasible Nodes

Provides optimal Node for newly created Pods

Node Selection in Kube-scheduler

The Kube-scheduler selects a Node for the Pod in a two-step operation.

Filtering Finds the set of Nodes where it is feasible to schedule the Pod

Scoring Ranks the Nodes that remain to select the most suitable Pod placement
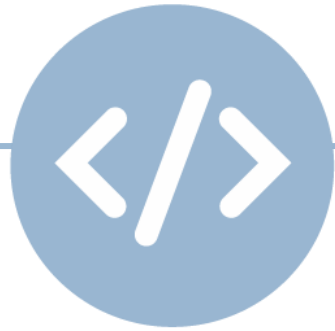
Configuring Filtering and Scoring Behavior

There are two supported ways to configure filtering and scoring behavior of a Scheduler:

Scheduling Policies

Scheduling Profiles

Node Affinity and Anti-Affinity

Node Affinity allows flexible decisions. YAML file configuration is used to represent specific requirements.

Node Anti-Affinity is created to allow flexible decision-making processes.

Kubelet

Kubelet

A Kubelet is a tiny application that communicates with the Control Plane. It makes sure that the Containers are running in a Pod.

The Kubelet works in terms of a specification called PodSpec.

A PodSpec is a YAML or JSON object; it describes a Pod.

Caltech | Center for Technology & Management Education

simpli learn

Providing Container Manifest to Kubelet

Apart from PodSpec, there are three ways to provide a manifest to Kubelet:

File

Path passed as a flag on the command line

HTTP endpoint

Passed as a parameter on the command line

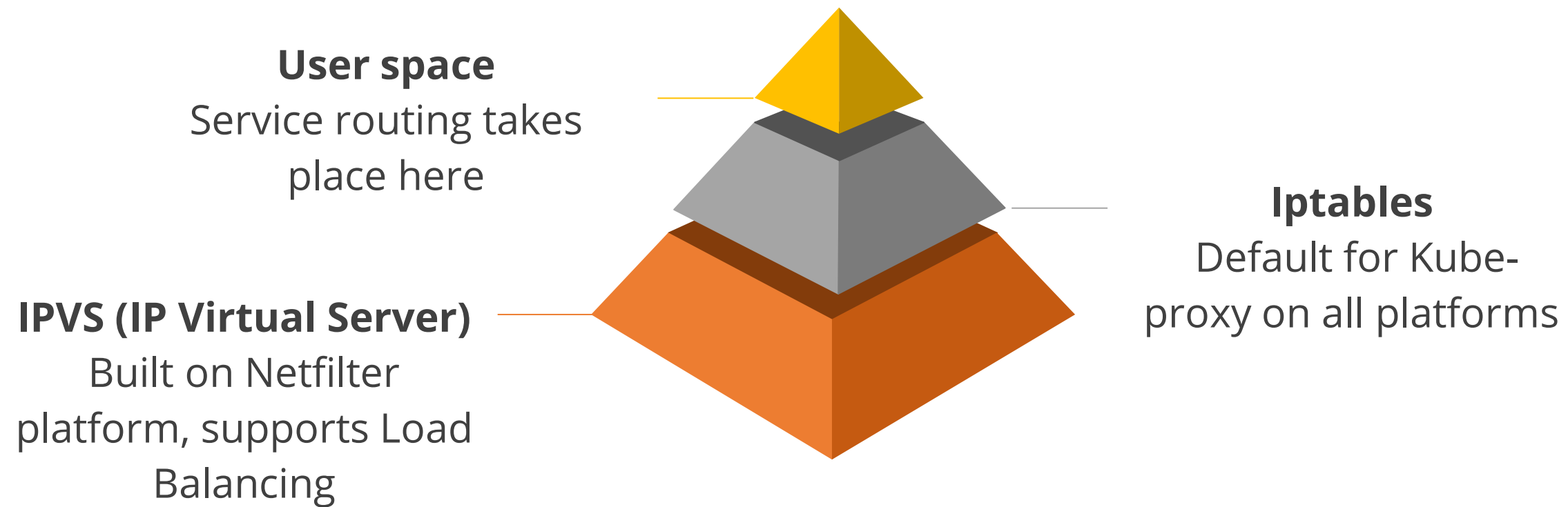**HTTP Server**
Listens for HTTP

Kube-proxy

Kube-proxy

Kube-proxy is a network proxy that runs on every Node in a cluster, implementing the Kubernetes Service concept.

Maintains network rules on Nodes

Manages forwarding of traffic addressed to the virtual IP addresses of the clusters

Caltech | Center for Technology & Management Education

simplilearn

Operation Modes

Each Node has a Kube-proxy Container process. The Kube-proxy currently supports three different operation modes:

User space
Service routing takes place here

Iptables
Default for Kube-proxy on all platforms

IPVS (IP Virtual Server)
Built on Netfilter platform, supports Load Balancing

Caltech | Center for Technology & Management Education

simplilearn

Pods

Using Pods

The smallest deployable unit of computing, which can be created and managed in Kubernetes, is the Pod.

Pods in a Kubernetes Cluster are mainly used in two ways:

1   Pods that run a single Container; the most common Kubernetes use case is the "one-container-per-Pod" model

2   Pods that run multiple Containers, which should work in conjunction with each other

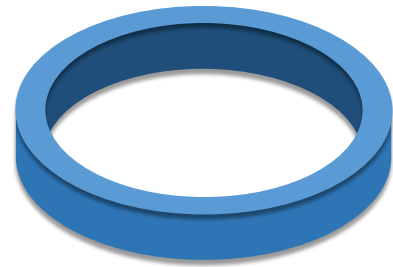Caltech | Center for Technology & Management Education

simplilearn

How Pods Manage Multiple Containers
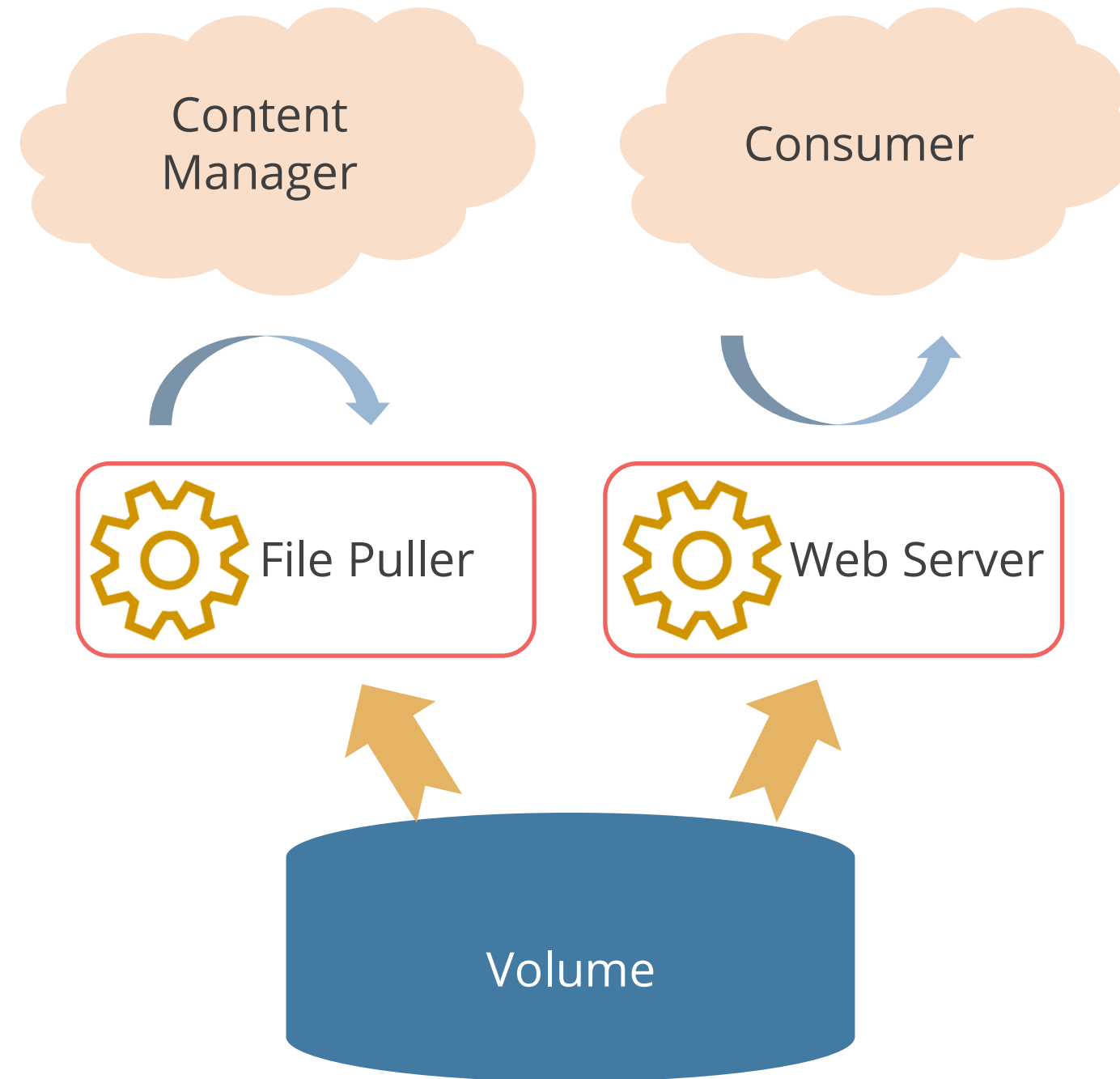
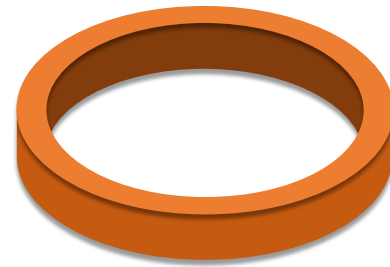# Pods and Controllers

Workload resources create and manage one or more Pods.
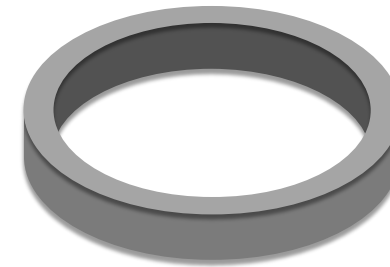
Examples of Workload Resources:

Deployment   StatefulSet   DaemonSet

Pod Template

Pod templates are specifications for creating Pods and are included in workload resources such as Deployments, Jobs, and DaemonSets.

Sample Pod Template

```
apiVersion: batch/v1
kind: Job
metadata:
  name: hello
spec:
  template:
    # This is the Pod template
    spec:
      containers:
      - name: hello
        image: busybox
        command: ['sh', '-c', 'echo "Hello,
Kubernetes!" && sleep 3600']
      restartPolicy: OnFailure
    # The Pod template ends here
```

Caltech | Center for Technology & Management Education

simplilearn

Pod Update and Replacement

When the Pod template for a workload resource is updated or changed, the Controller does not update or patch existing Pods. Instead, it creates new Pods based on the updated template.

Limitations

Most of the metadata about a Pod is immutable.

If the metadata.deletionTimestamp is set, a new entry cannot be added to the metadata.finalizers list.

Pod updates may not change fields.

When updating the spec.activeDeadlineSeconds field, two types of updates are allowed.

Caltech | Center for Technology & Management Education

simplilearn

Resource Sharing and Communication

Pods enable data sharing and communication among their constituent Containers using two methods:

Storage in Pods: All Containers in a Pod have access to the shared volumes, allowing those Containers to share data.

Pod Networking: When Containers within a Pod communicate with entities outside the Pod, they must coordinate how they use the shared network resources.

Caltech | Center for Technology & Management Education

simplilearn

Privileged Mode for Containers

Any Container in a Pod can get the Privileged Mode into working, utilizing the privileged flag on the security context of the Container spec.

Privileged Mode is used for Containers that use the operating system administrative capabilities.

Privileges available to processes outside a Container are also given to processes existing in a Privileged Mode.

Static Pods

The Kubelet daemon on a specific Node manages static Pods directly, without being observed by the API server.
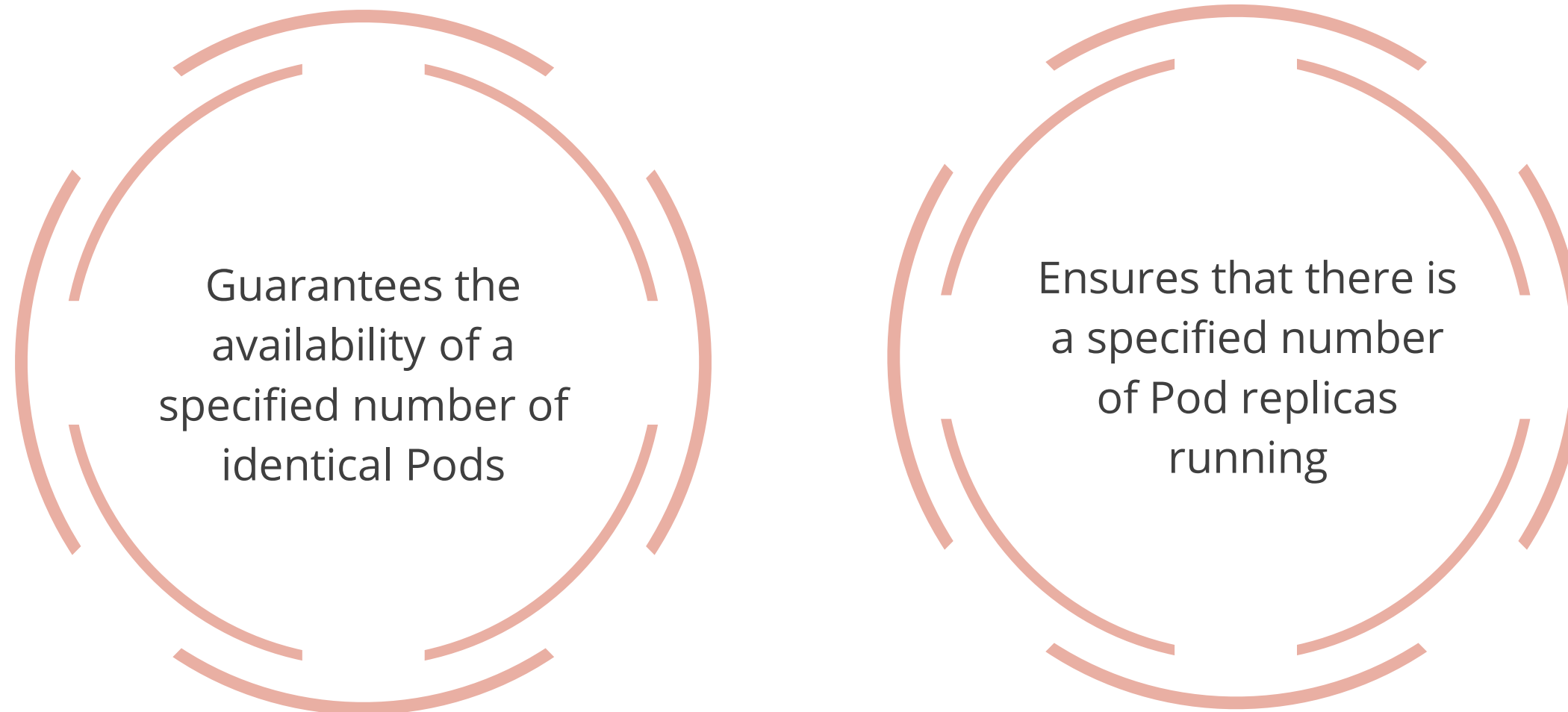
The Control Plane manages most Pods. The Kubelet supervises every static Pod directly.
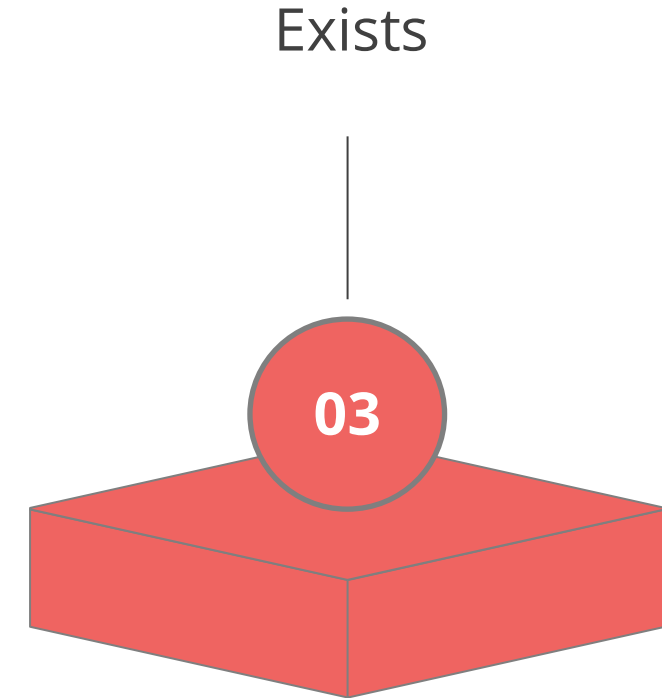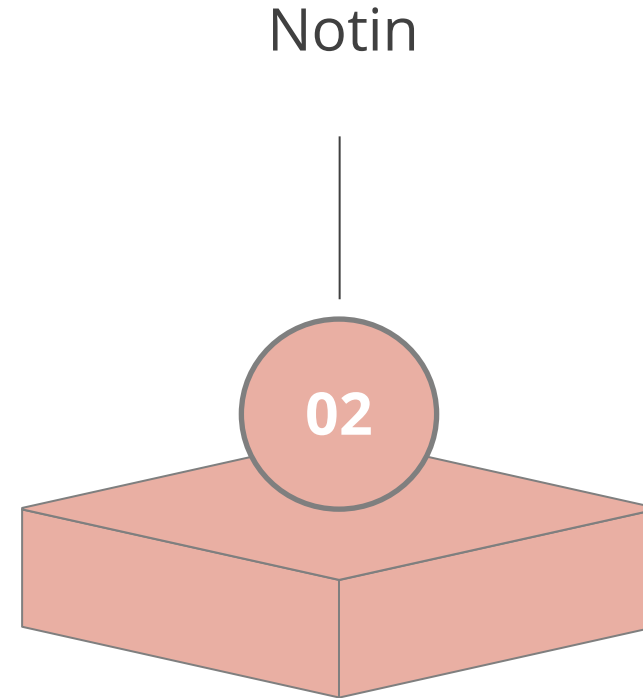
ReplicaSets

ReplicaSets

ReplicaSets maintain a stable set of replica Pods running at any given time.

Guarantees the availability of a specified number of identical Pods

Ensures that there is a specified number of Pod replicas running

Caltech | Center for Technology & Management Education

simplilearn

Operators to Use with ReplicaSets

Ensure that the selectors of one ReplicaSet do not match another's.

In

Notin

Exists

01

02

03

Caltech | Center for Technology & Management Education

simplilearn
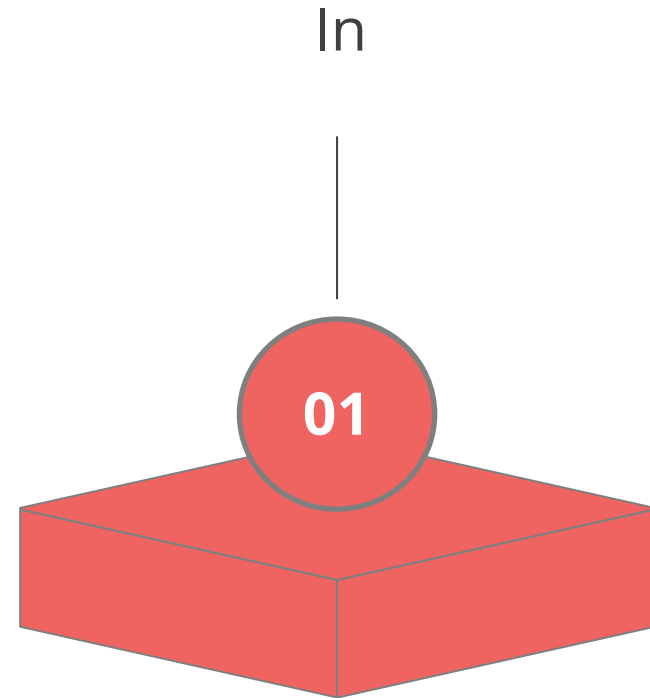
ReplicaSet Manifest

Sample

```
apiVersion: apps/v1 # our API version
kind: ReplicaSet   # The kind we are creating
Metadata: # Specify all Metadata like name, labels
  name: some-name
  labels:
    app: some-App
    tier: some-Tier
Spec:
  replicas: 3 # Here is where we tell k8s how many replicas we want
  Selector: # This is our label selector field.
    matchLabels:
      tier: some-Tier
    matchExpressions:
      - {key: tier, operator: In, values: [some-Tier]} # we are
using the set-based operators
  template:
    metadata:
      labels:
        app: some-App
        tier: someTier
    Spec: # This spec section should look like spec in a Pod
definition
      Containers:
```

Caltech | Center for Technology & Management Education   simpli·learn

Working of ReplicaSet

A ReplicaSet is defined with fields, including a selector that specifies:

How to identify Pods

The number of Pods to be maintained

Data in new Pods

It ensures that a specified number of Pod replicas are running at any given time.
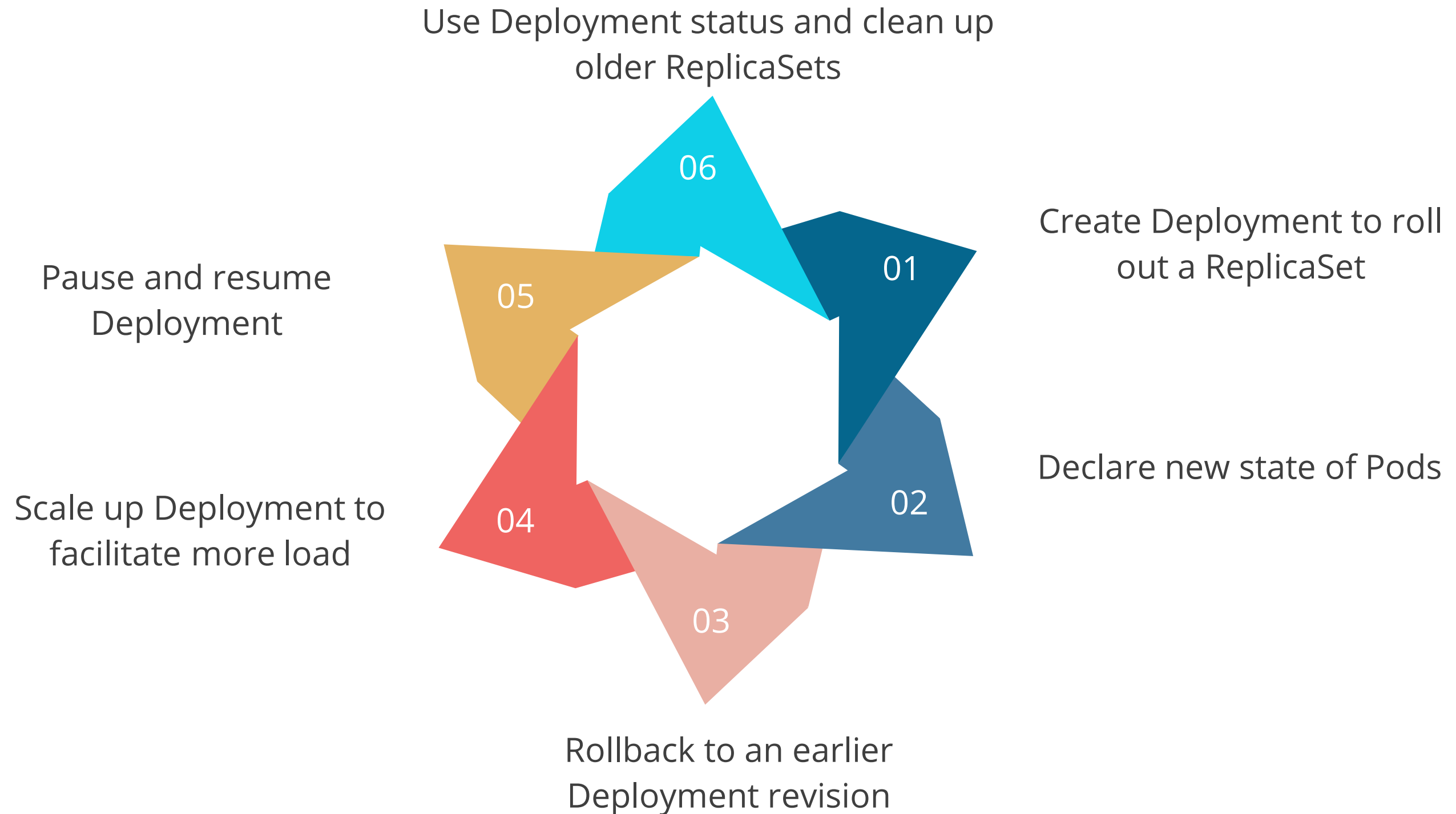
Deployment

Deployment

Deployment provides declarative definitions for Pods and ReplicaSets.

Deployments can be defined to create new ReplicaSets or remove existing Deployments.

Use Cases of Deployment

Use Deployment status and clean up older ReplicaSets

06

Create Deployment to roll out a ReplicaSet

01

Pause and resume Deployment

05

Declare new state of Pods

02

Scale up Deployment to facilitate more load

04

Rollback to an earlier Deployment revision

03

Caltech | Center for Technology & Management Education

simplilearn

Creating a Deployment

Sample

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Caltech | Center for Technology & Management Education

simplilearn

Updating and Rolling Back Deployment

Deployments may be updated. This is done by making changes to the Pod template spec within the Deployment; it automatically generates an update rollout.

Rolling Back Deployment

```
kubectl rollout undo [deployment_name]

#Adding the argument

-to-revision=

#will roll back to that specific version of
the deployment
```

Scaling a Deployment

Deployments are useful for scaling the number of replicas as demand increases for a particular application.

Example:

```
# to scale a deployment up to 20 replicas

kubectl scale [deployment-name] -replicas 20
```

Pause and Resume

Applying multiple fixes in between pausing and resuming, without triggering unnecessary rollouts, is facilitated.

Example:

```
#Pause a deployment

kubectl rollout pause deployment.v1.apps/nginx-deployment

#Resuming a deployment

kubectl rollout resume deployment.v1.apps/nginx-deployment
```

Services, Load Balancing, and Networking

Overview

Services and Load Balancing are the most important part of Kubernetes networking, and they address four main concerns.

> Containers in a Pod use networking to communicate via loopback.

> Cluster networking facilitates communication between various Pods.

> The Service resource lets exposing an application running in Pods to be accessible from outside the cluster.

> Services are used to publish services meant for consumption inside the cluster only.

Kubernetes Pod Network

A Kubernetes Pod network connects several interrelated components including:

Pods

Containers

Nodes

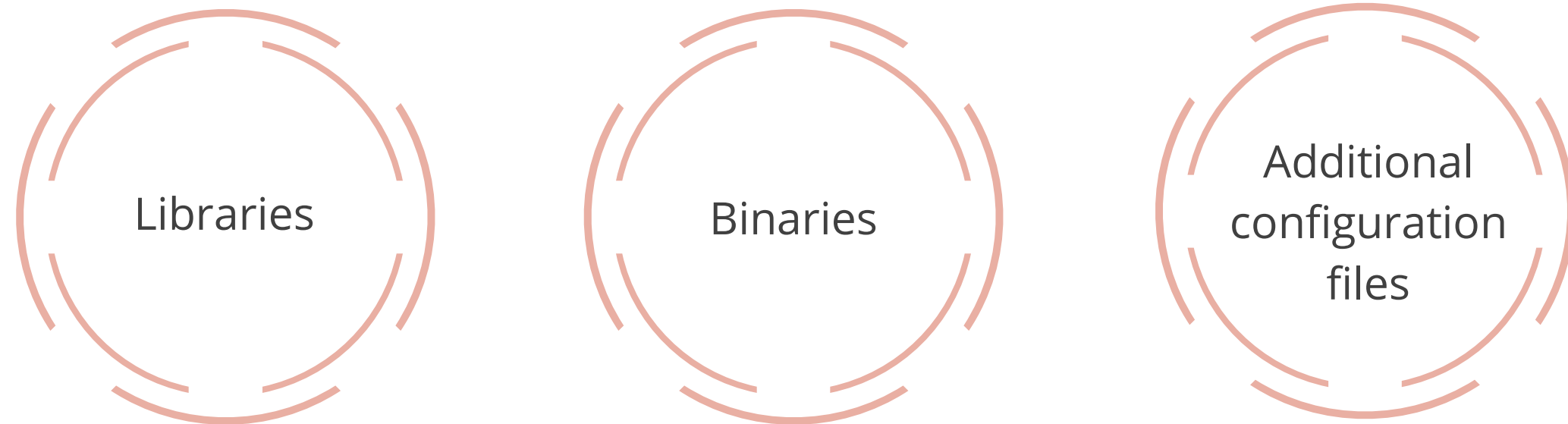Master Node

01 02 03 04
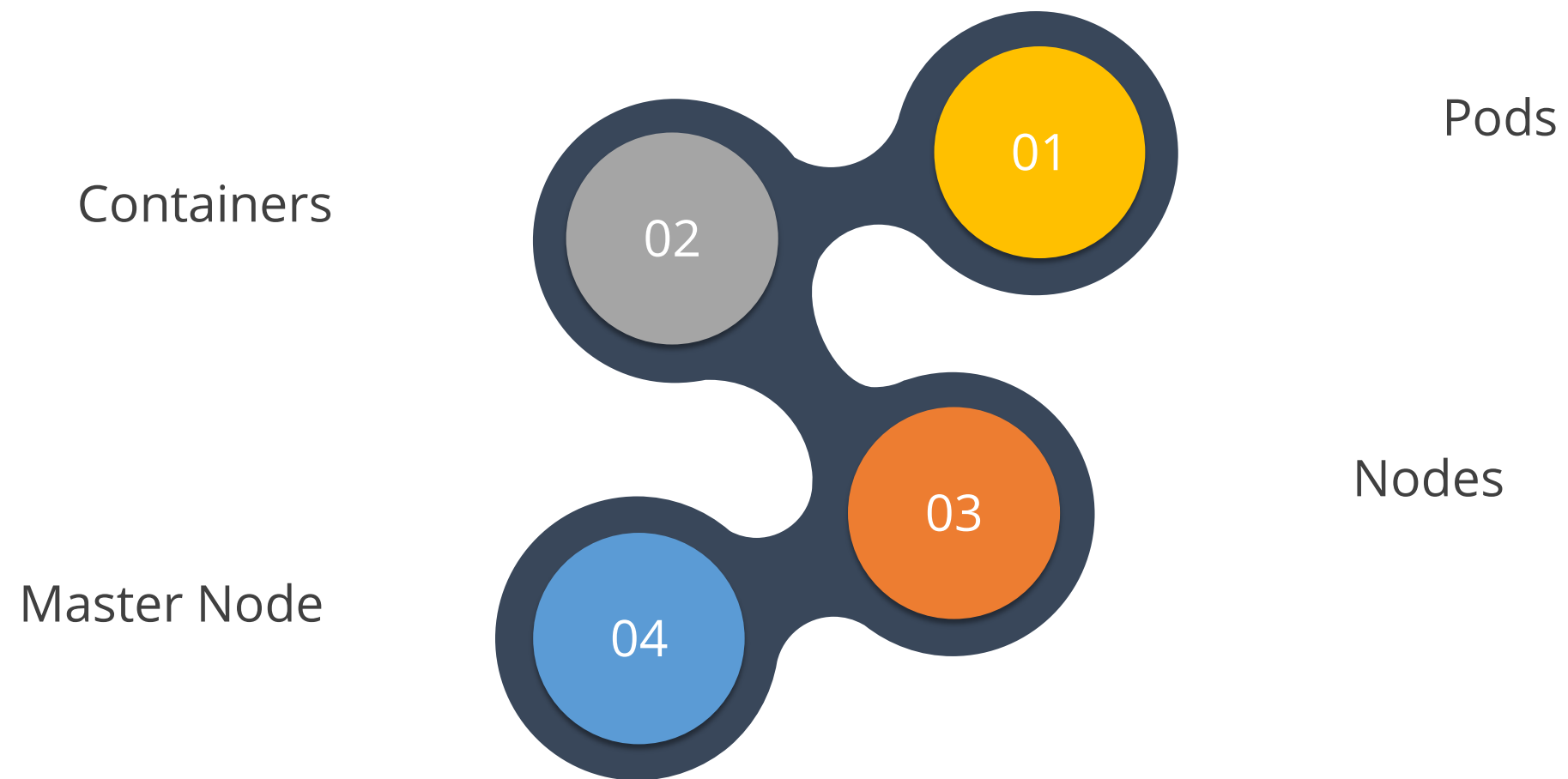
Networking in Kubernetes

Traffic that flows between Nodes can also flow to and from Nodes and an external physical machine or a VM. There are four ways of getting external traffic into a Kubernetes cluster.

LoadBalancer

ClusterIP

NodePort

Ingress
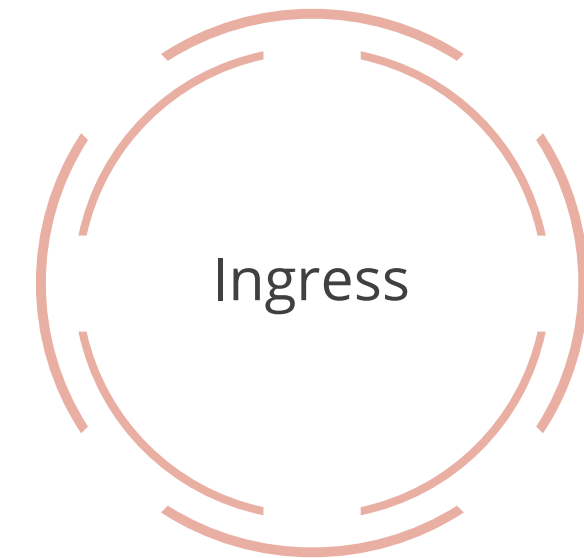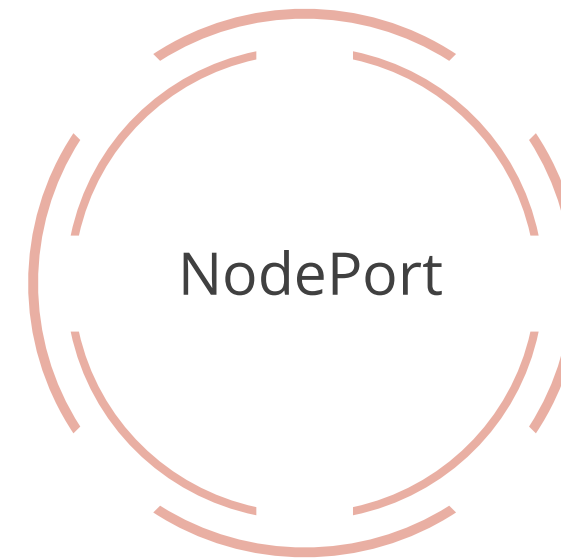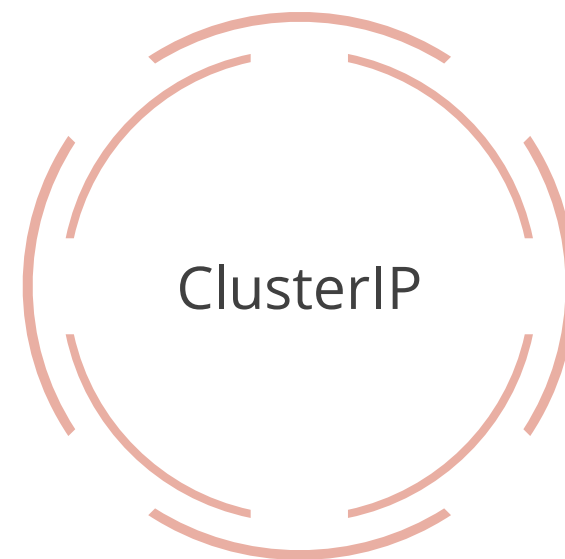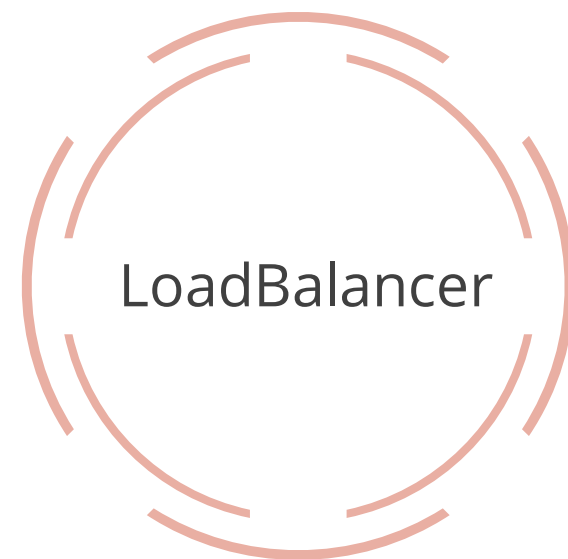
Containers

Introduction

Containers are technologies that capture a complete runtime environment.

Libraries

Binaries

Additional configuration files

Containerized Application

Containerized applications can be deployed without regard to underlying infrastructure.

Containerized applications are isolated from each other, similar to VMs, thereby increasing reliability and decreasing problems resulting from application interactions.

Caltech | Center for Technology & Management Education

simplilearn

Benefits of Containers

Lightweight

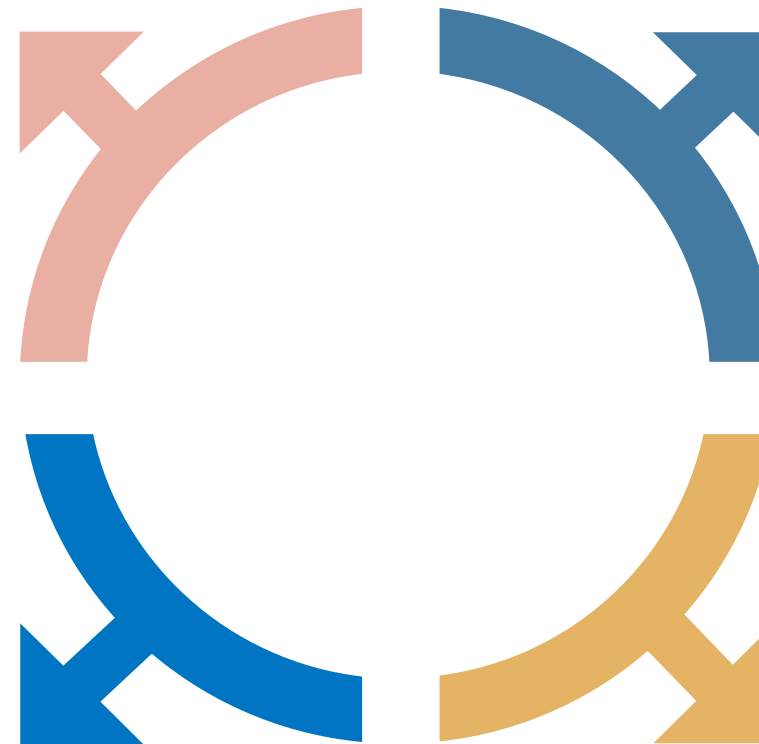Scalable

Portable and consistent

Agile

Problems Containers Solve

Containers are versatile and solve a broad range of IT problems throughout an application's lifecycle.

Ensuring software runs properly when moved between computing environments

Increasing efficiency by eliminating the need for a separate hypervisor

Facilitating microservices deployments

Eliminating conflicts and dependencies between multiple applications

Caltech | Center for Technology & Management Education

simplilearn

Use Cases for IT Operations

Improve application security by isolating from other applications

Facilitate seamless migration of applications

Improve IT efficiency by enabling multiple application Containers to run on a single OS instance

Offer on-demand scalability

Caltech | Center for Technology & Management Education

simplilearn

How Do Containers Work?

Containers isolate applications from one another.

A registry or repository is used to transfer Container images and the application Container Engine which turns the images into executable code.

Container repositories facilitate reusing commonly used Container images.

Containers are created using the process of packaging applications.

Containers: Features

Namespaces: Provide visibility to the underlying OS

Control Groups: Resource managers in the Linux kernel

Union File Systems: Prevent data duplication

Policies

Overview

Policies define what end users can do on the cluster and possible ways to ensure that clusters comply.

Policies are applicable to network, volume, resource usage, resource consumption, access control, and security.

Constraint is a declaration that expects a system to meet a set of requirements.

Policy-enablement helps organizations take control of Kubernetes operations.
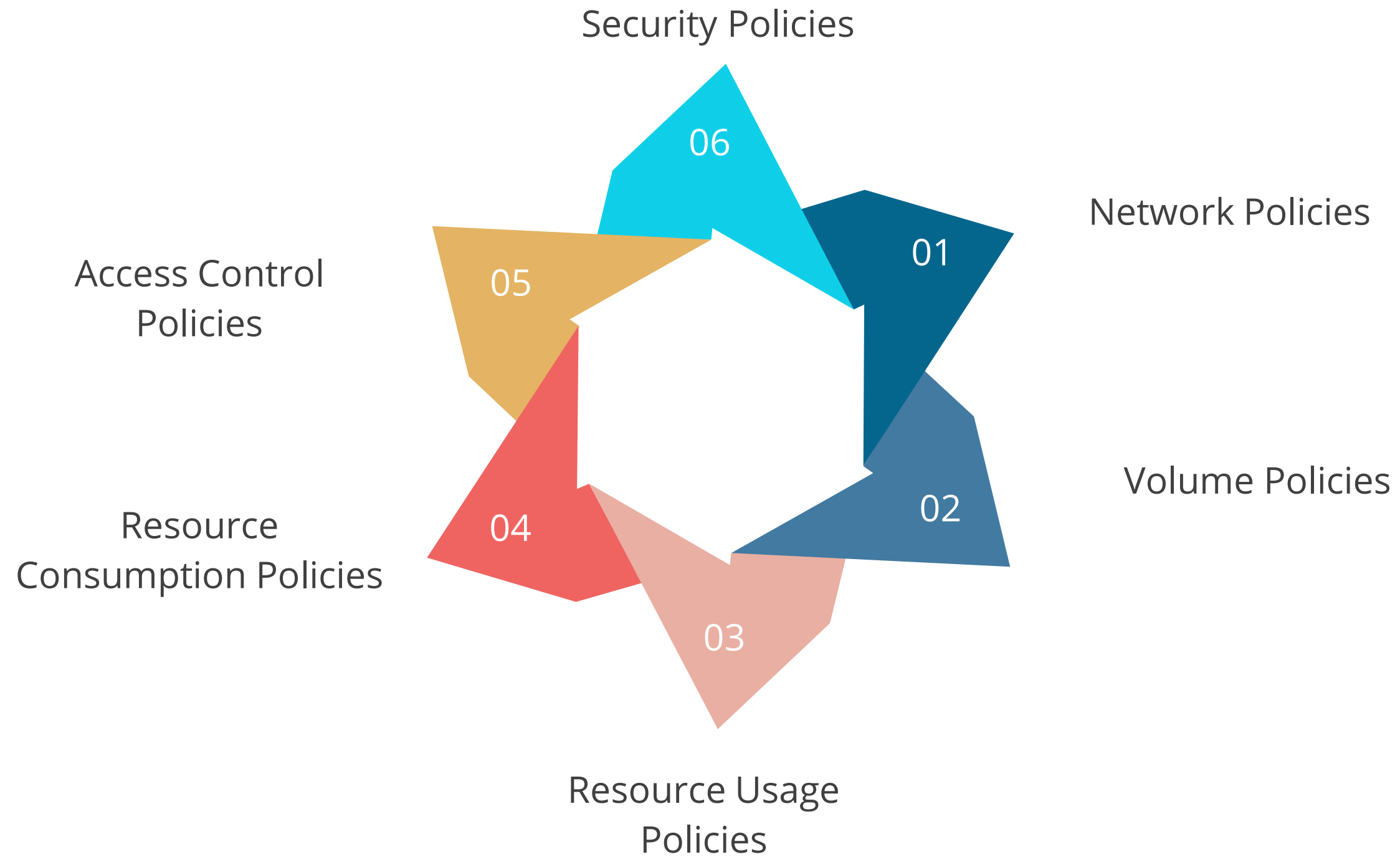
Key Benefits of Policies

Simplified Operations

Ease of Policy Enforcement

Automated discovery of violations and conflicts

Better flexibility to changing requirements

Policy Restrictions

Security Policies

Network Policies

Volume Policies

Resource Usage
Policies

Resource
Consumption Policies

Access Control
Policies

Key Takeaways

- Containers are a centralized unit of software application environment that is pre-packaged with code, along with its dependencies.

- Etcd is an open-source distributed key-value store used to manage critical information that distributed systems need to keep running.

- Kube-proxy is a network proxy that runs on every Node in a cluster, implementing the Kubernetes Service concept.

- Policies define what end users can do on the cluster and possible ways to ensure that clusters comply.

Thank You