Leveraging Advanced Features on the Google Cloud Kubernetes Engine

CREATING AND MANAGING DEPLOYMENTS ON GKE CLUSTERS

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Leveraging higher level abstractions to control deployments on Kubernetes

Defining and running Jobs and CronJobs on clusters

Using node taints to control scheduling

Deploying stateless and stateful applications

Prerequisites and Course Outline

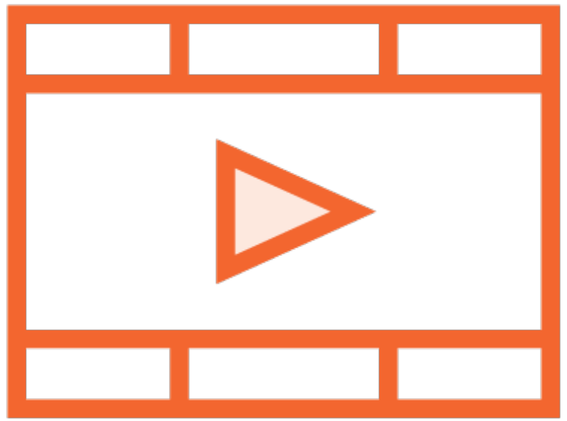
Software and Skills

Good understanding of the Kubernetes container orchestration technology

Experience working with Deployments and exposing Services on the GKE

Basic understanding of security and networking

Prerequisites: Basic Cloud Computing



Deploying Containerized Workloads Using Google Cloud Kubernetes Engine

- Introduction to Kubernetes on the GCP

Getting Started with Kubernetes

- Working with the Kubernetes container technology

Course Outline

Creating and managing deployments

- Running Jobs and CronJobs on clusters
- Tainting nodes to control scheduling
- Stateless and stateful applications

Networking and security

- Configuring networking policies
- Creating and using private clusters
- Internal load balancing with GKE clusters
- Applying pod security policies

Leveraging CI/CD with Jenkins pipelines

- Building and deploying an app to production
- Managing canary releases

Scenarios: SpikeySales.com

Hypothetical online retailer

- Flash sales of trending products
- Spikes in user traffic

Spikey Sales on the GCP

- Cloud computing fits perfectly
- Pay-as-you-go
- No idle capacity during off-sale periods
- Elastic, pay-as-you-go, global access to data

A Quick Overview of Kubernetes

Kubernetes

Orchestration technology for containers - convert isolated containers running on different hardware into a cluster

Compute Choices

Containers

hybrid, multi-cloud

Container Clusters

Kubernetes

Kubernetes as Orchestrator

Fault-tolerance

Autohealing

Isolation

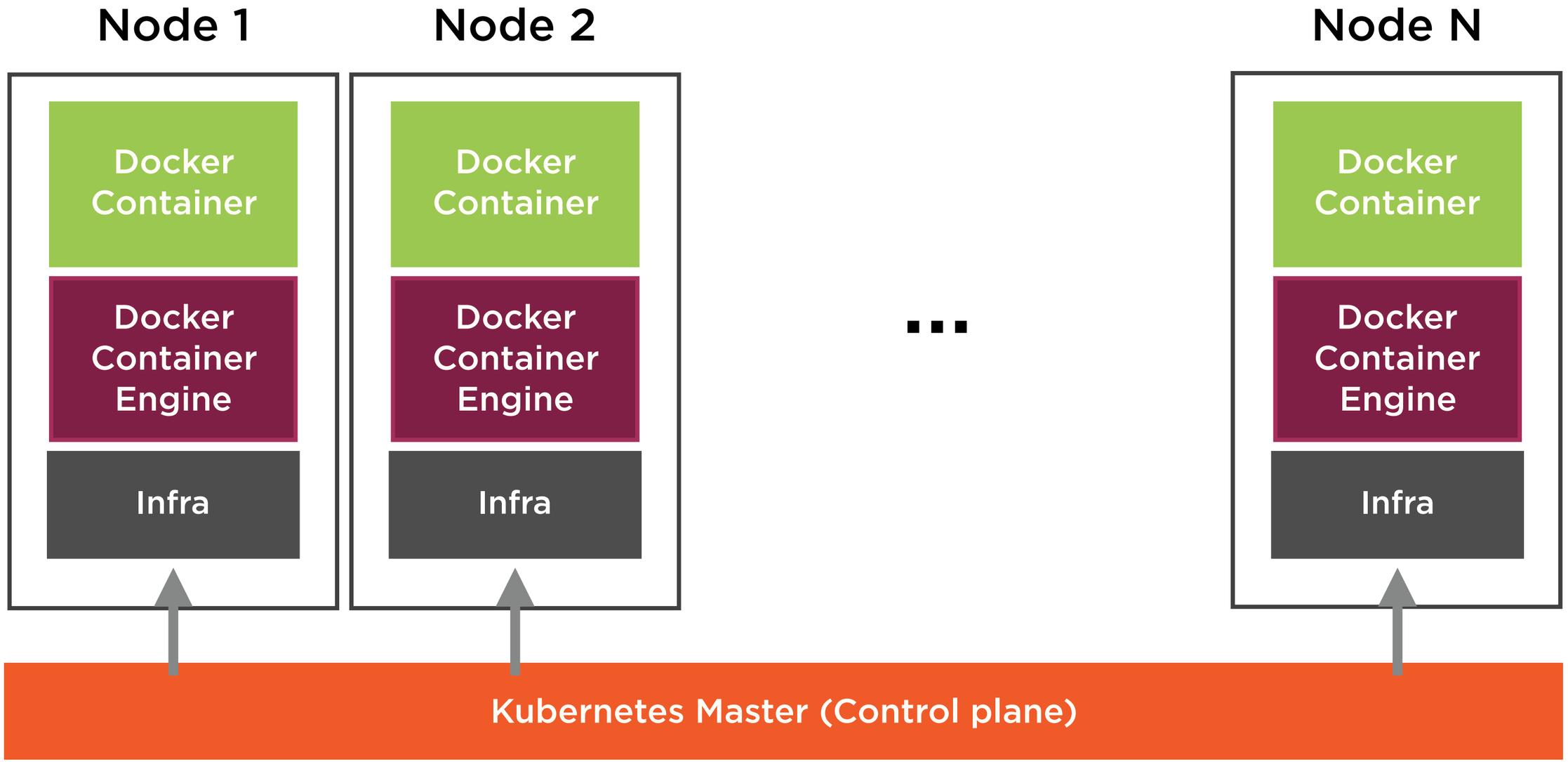
Scaling

Autoscaling

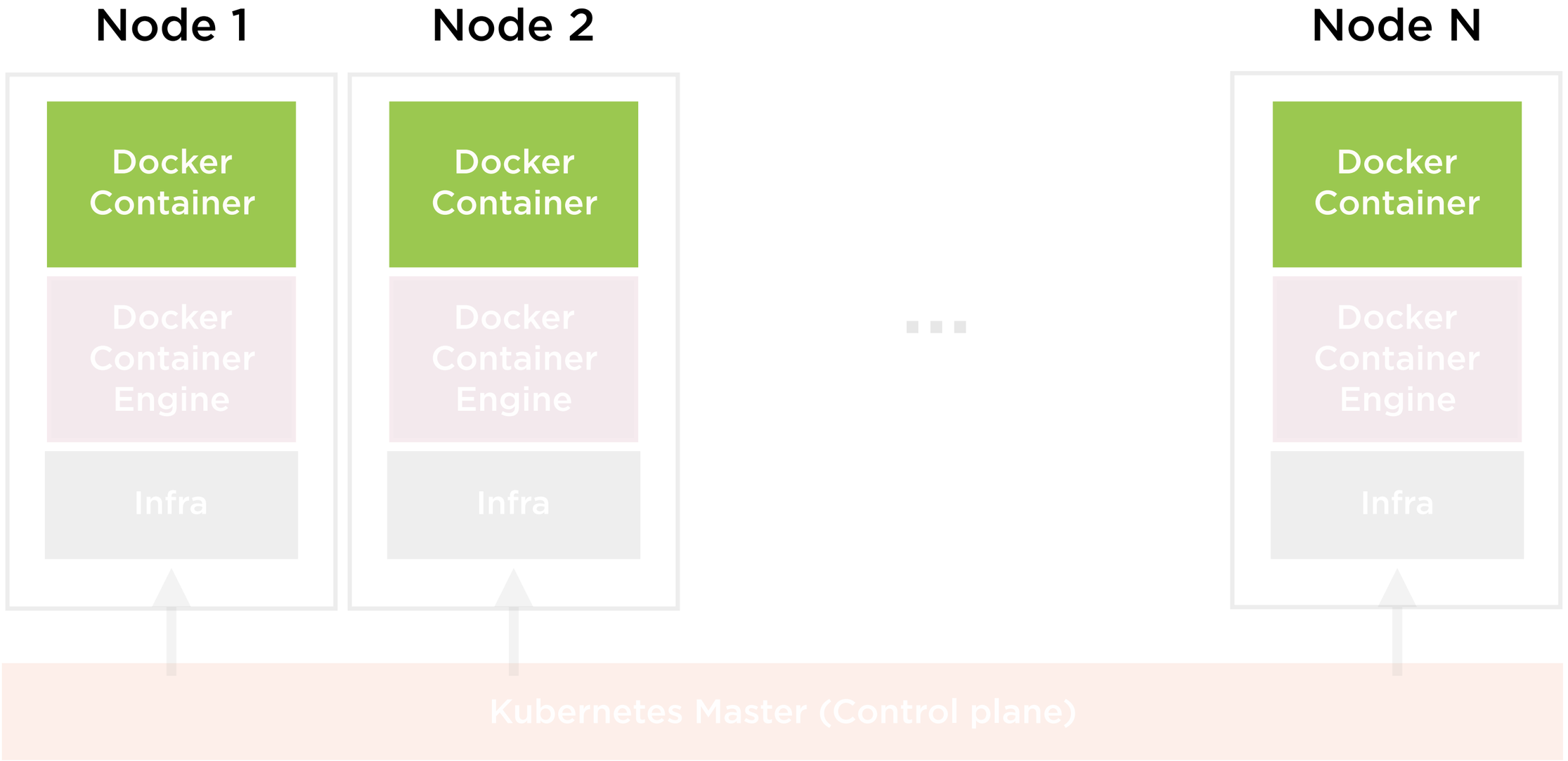
Load balancing

All of these are possible in a Kubernetes cluster using higher-level abstractions

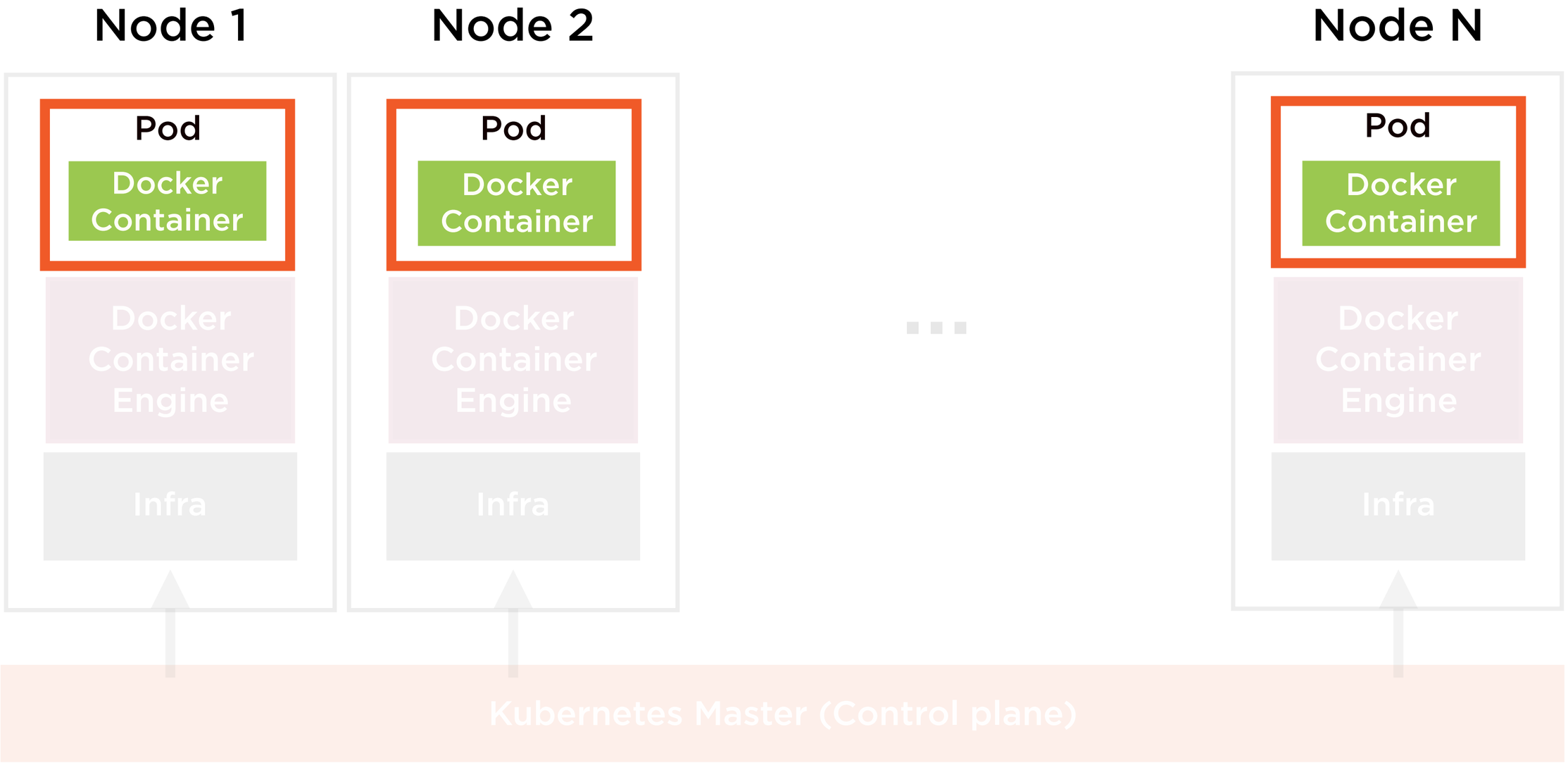
Kubernetes: Cluster Orchestration



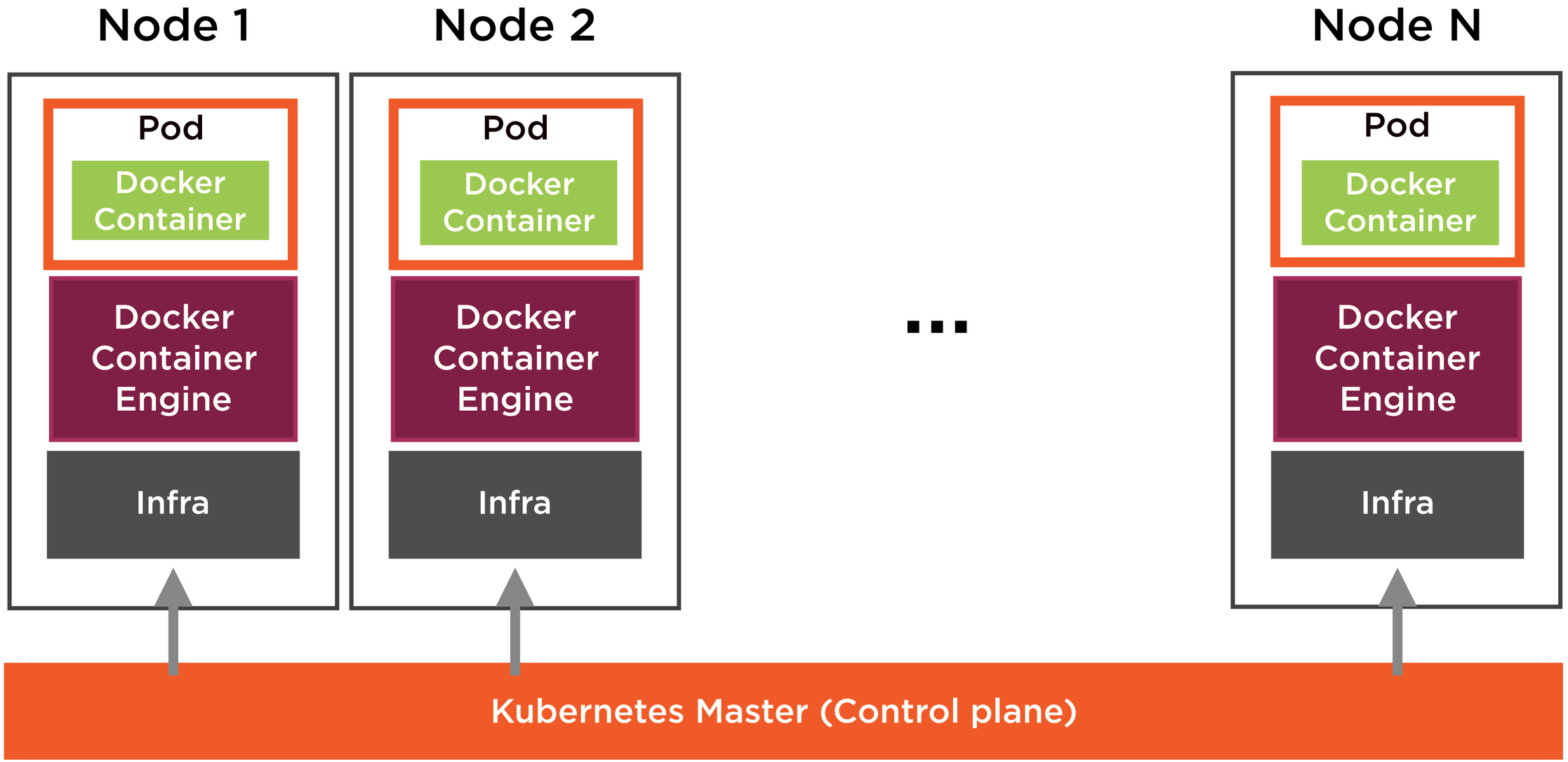
Kubernetes: Cluster Orchestration



Kubernetes: Containers Run Within Pods



Kubernetes: Cluster Orchestration



Pods as Atomic Units

Container deployment

All containers in pod are deployed, or none are

Node association

Entire pod is hosted on the same node

Pod is atomic unit of deployment in Kubernetes

The ReplicaSet Object

Multiple identical pods which are replicas of each other

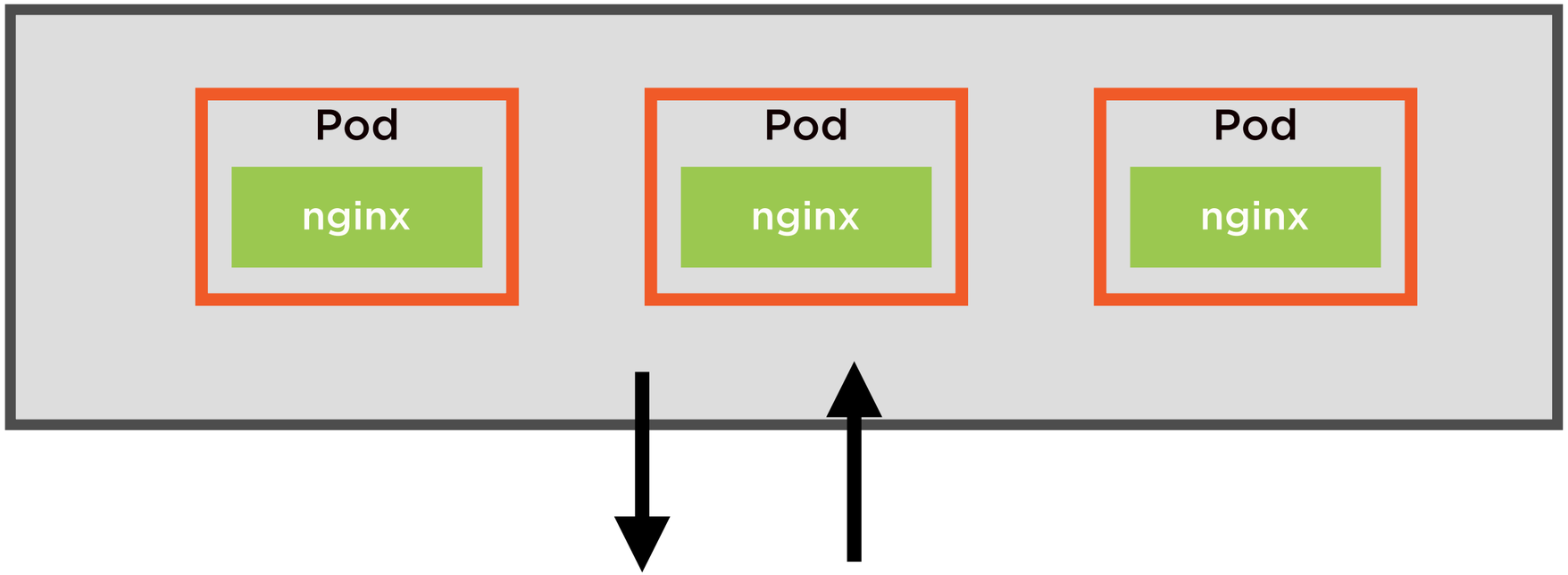
ReplicaSet

Self healing and autoscaling for our pods

The Deployment Object

Adds on deployment and rollback functionality

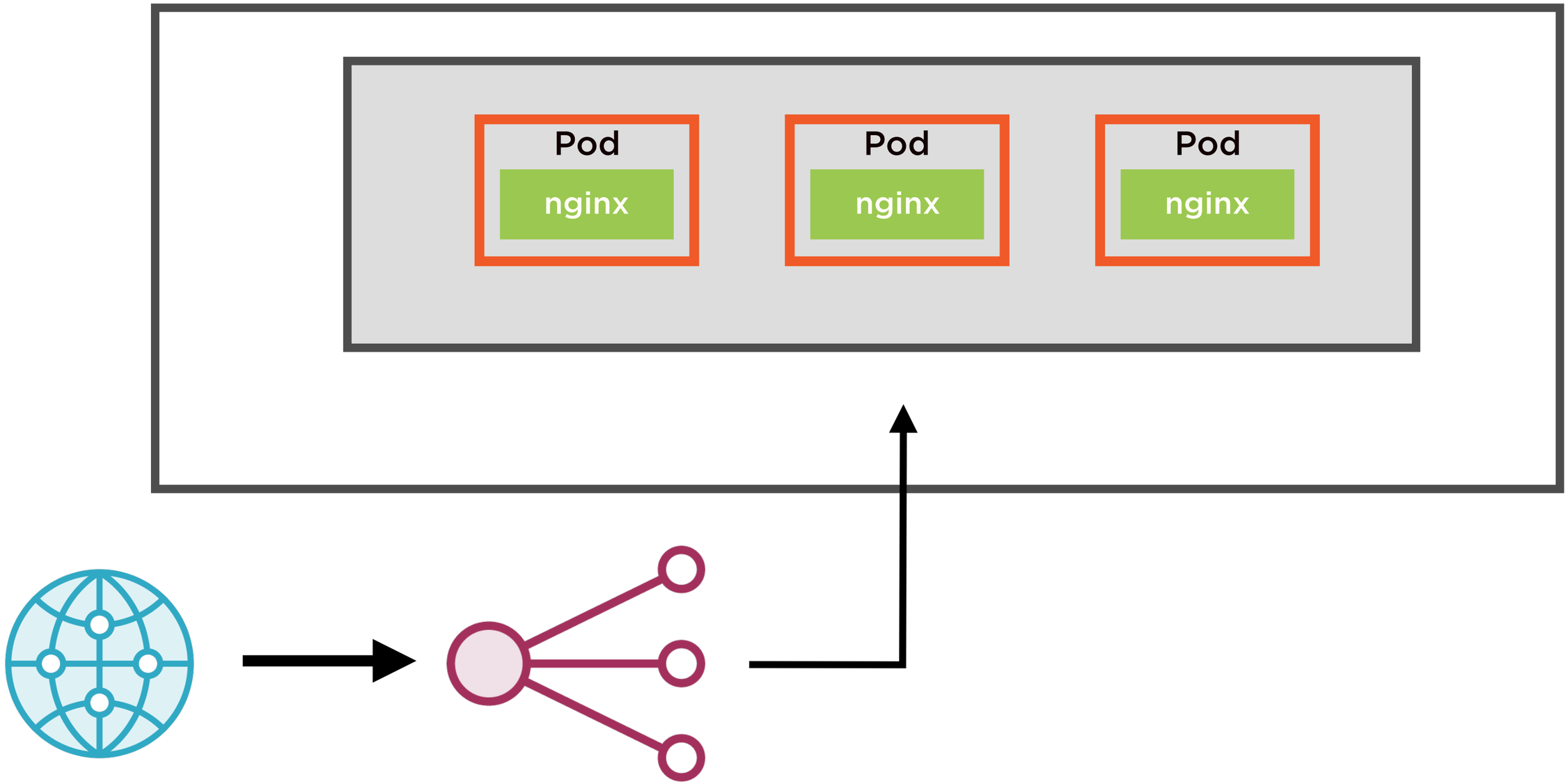
Deployment



Support for versions, and production-level operations such as rollbacks

Services provide stable IP addresses for external connections and load balancing

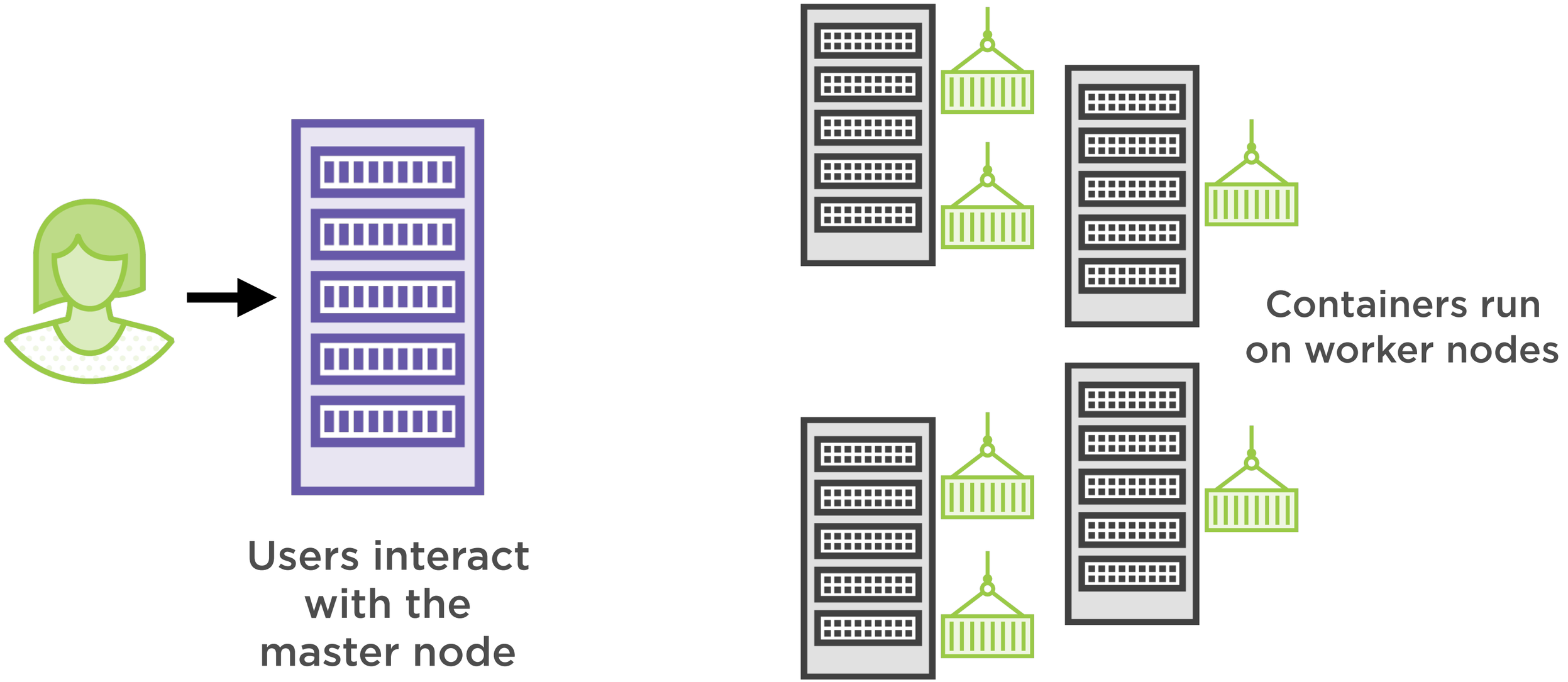
Service



A GKE Cluster

Made up of nodes, arranged in node pools, running container optimized node images

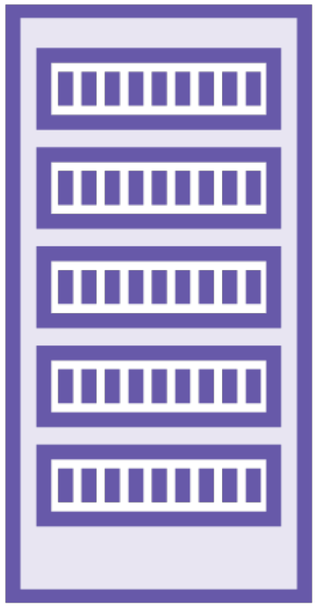
Kubernetes Clusters

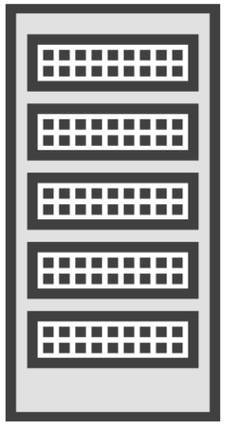


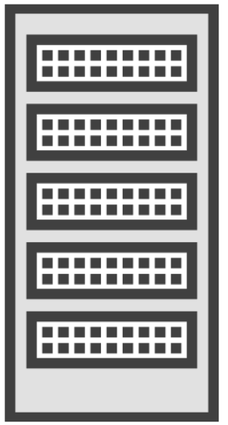
Users interact with the master node

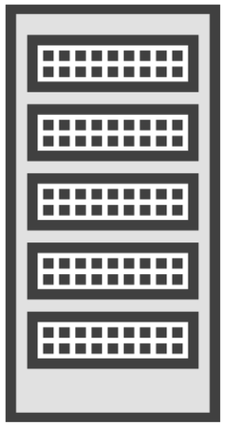
Containers run on worker nodes

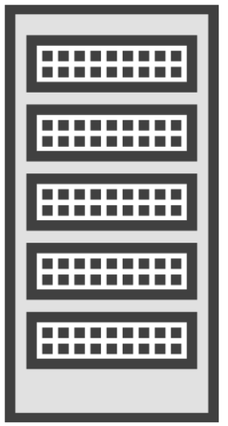
Nodes





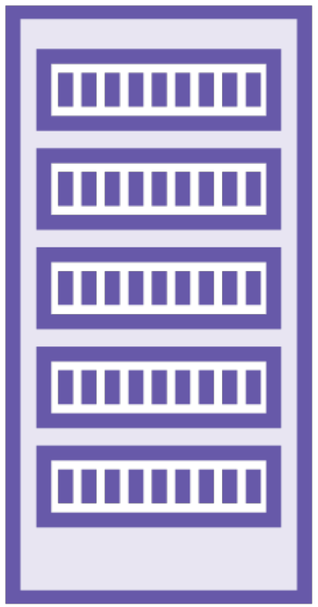




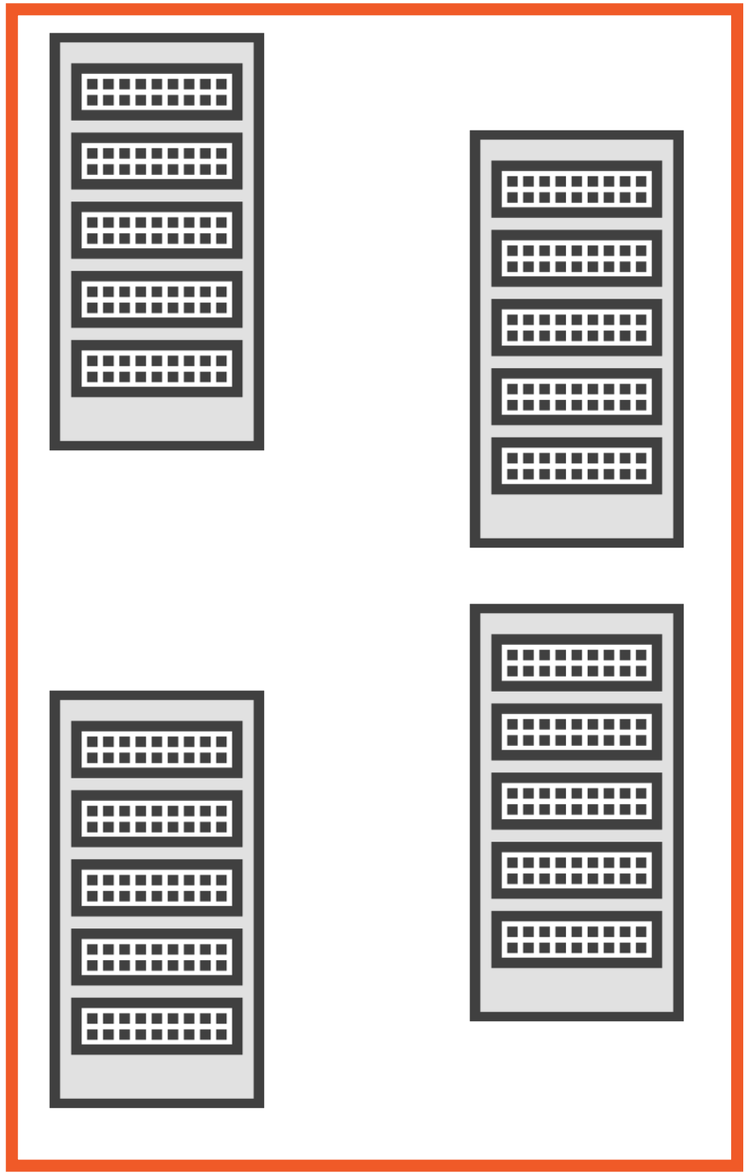


Nodes are on-premise or cloud VMs
on which containers are run

Node Pools



A subset of node instances which have the same configuration are called node pools



Label Selectors on Kubernetes

Labels

Key-value pairs attached to objects such as pods. Used to specify meaningful identifying attributes of objects. Not meant to be unique.

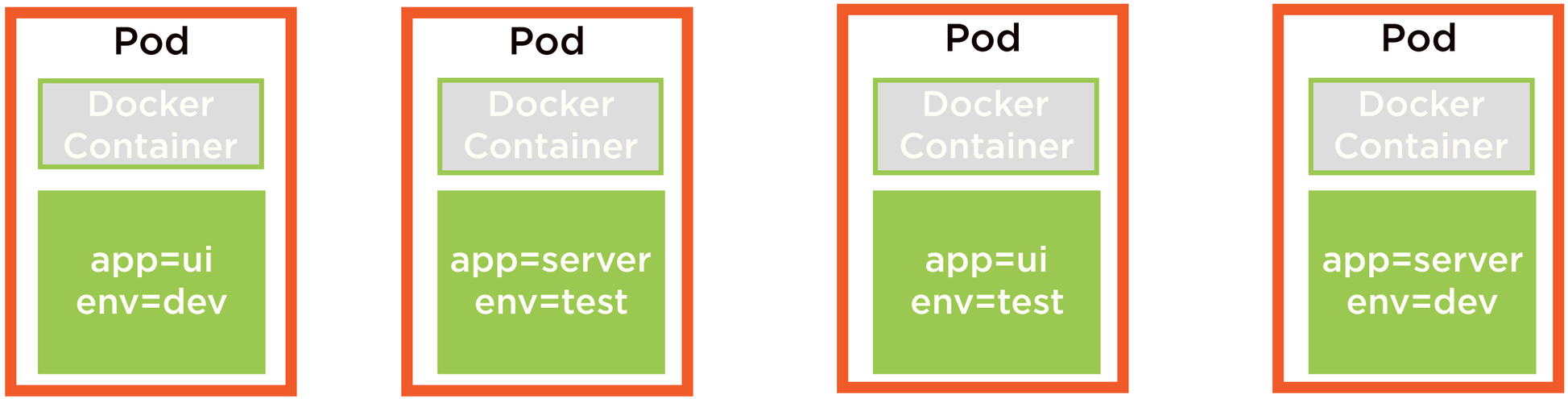
Label Selectors

Allow clients, users, or higher-level abstractions to identify groups of objects. Core grouping primitive in a Kubernetes cluster.

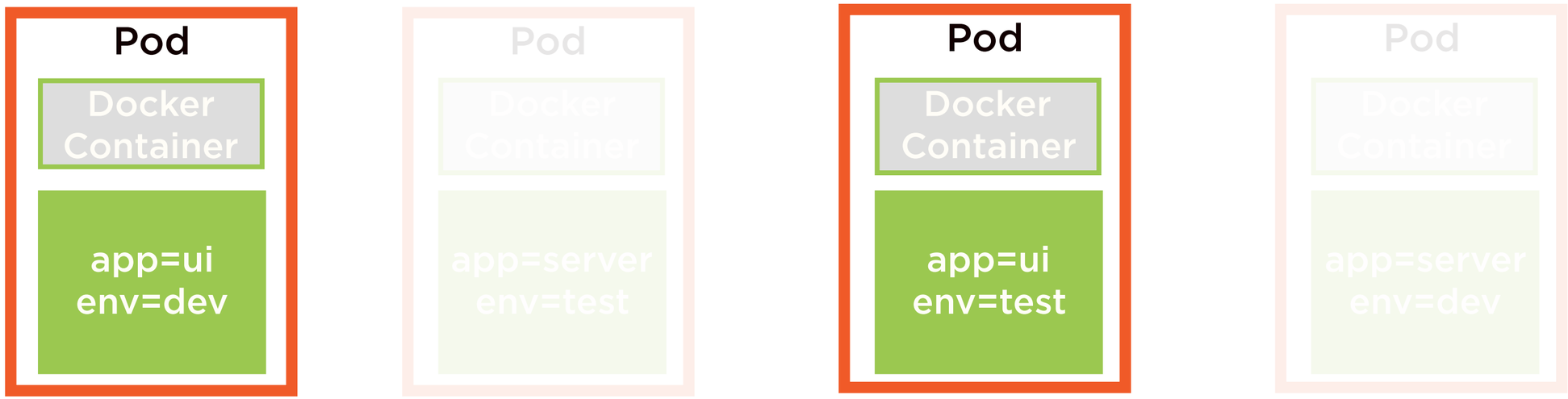
Label Selectors

Allow higher-level abstractions to be loosely coupled with objects they manage

Label Selectors

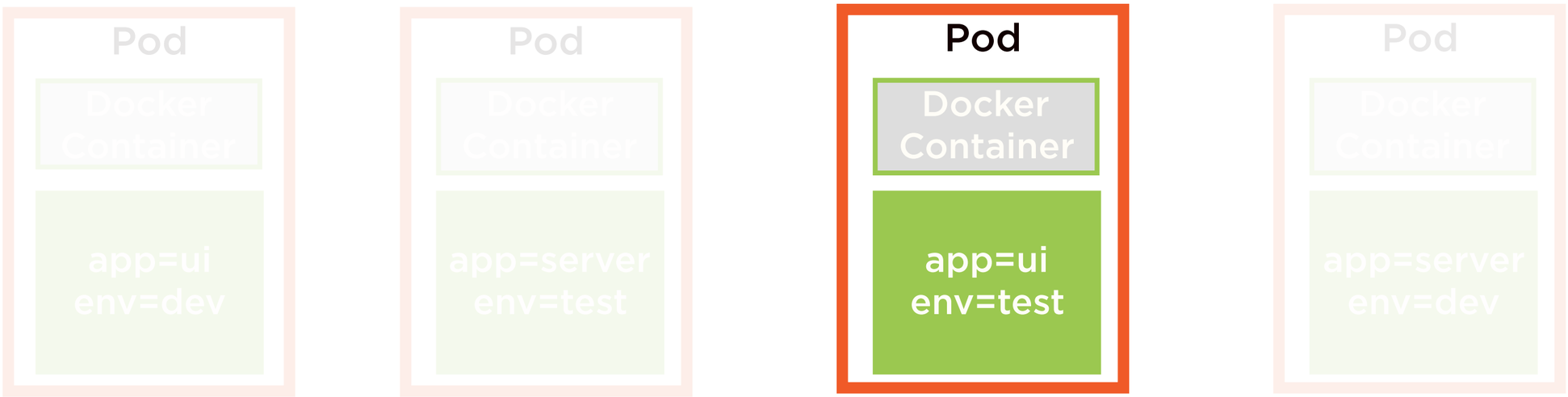


Label Selectors



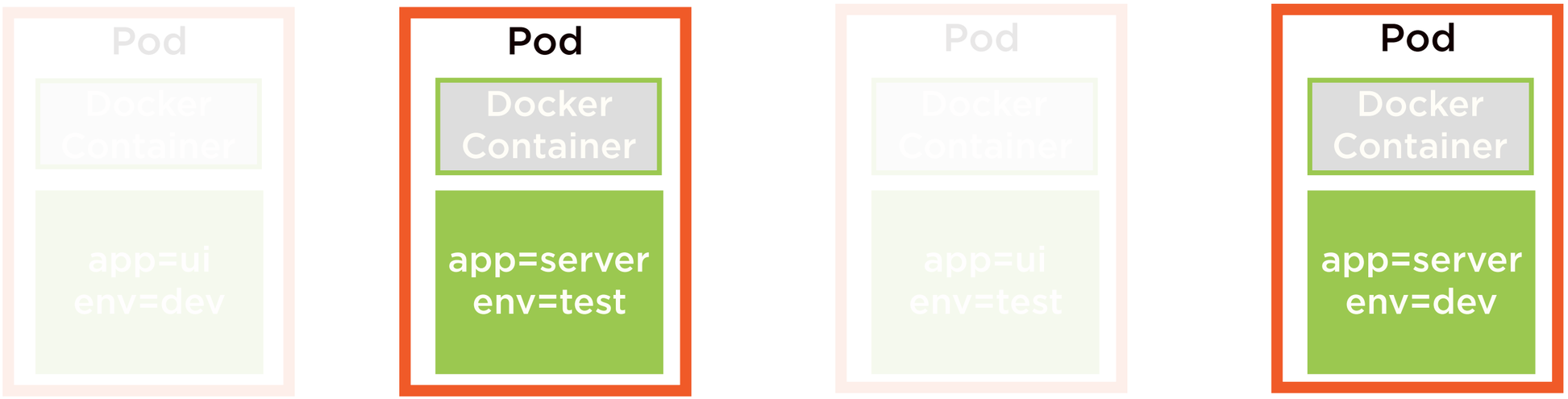
app=ui

Label Selectors



app=ui
env=test

Label Selectors



app=server

Jobs and CronJobs

Job

A controller object which represents a finite task

Manages the task until it completes execution

Does not maintain a desired cluster state

Best for long-running, batch operations

Two Types of Jobs

Non-parallel job

Creates one pod and uses that pod to run the job through to completion

Parallel job

Creates multiple pods and is complete when a certain count of pods terminate successfully

The Job controller is responsible for re-creating pods if its pods fail or are terminated

Two Types of Jobs

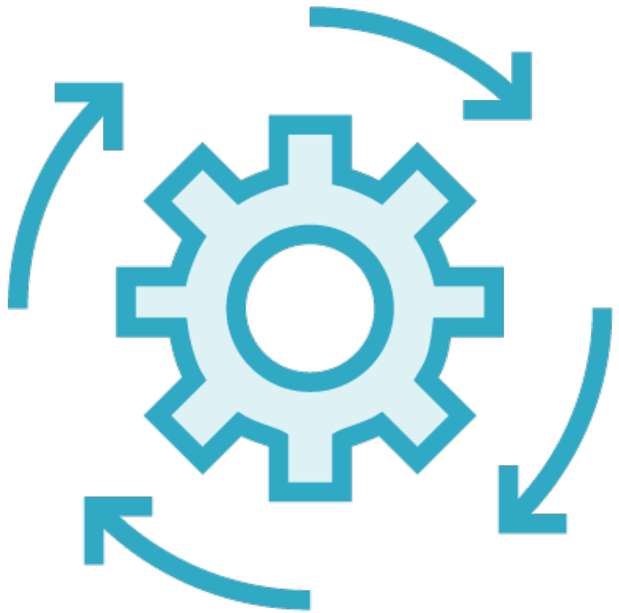
Non-parallel job

Creates one pod and uses that pod to run the job through to completion

Parallel job

Creates multiple pods and is complete when a certain count of pods terminate successfully

Can also specify deadlines for Jobs so that they do not re-try creating pods forever



CronJob

A controller object to perform finite time-related tasks

Tasks may be run once or repeatedly at a time interval

Uses Job objects under the hood to manage tasks

Best for automated, repeated tasks:

- Backups, generating reports, sending email

Demo

Create a cluster with two node pools

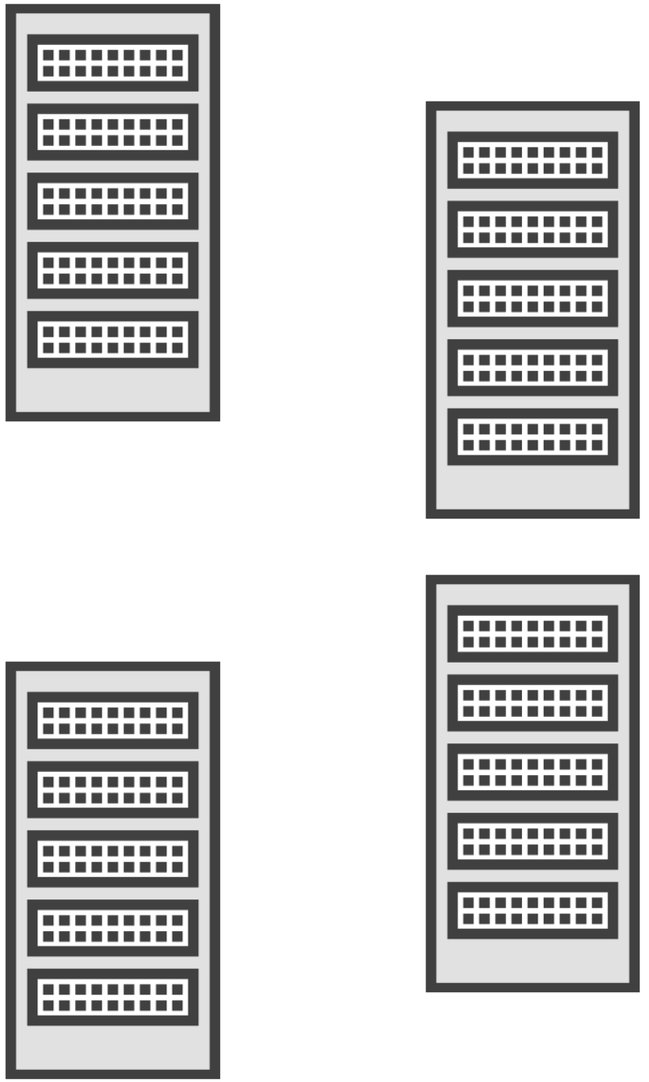
Create a node pool with preemptible instances for running batch operations

Deploy and run a Job on preemptible nodes to generate customer ratings reports on product categories

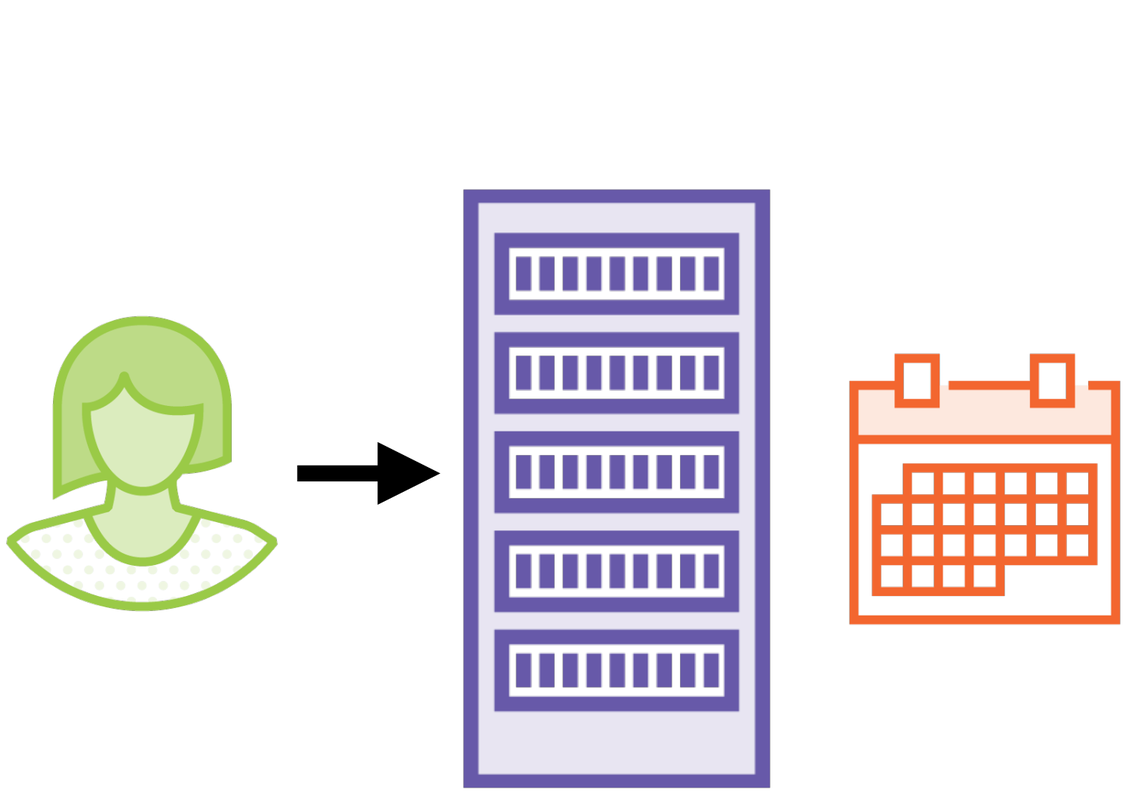
Controlling Scheduling with Node Taints

Workload Submitted to a Cluster

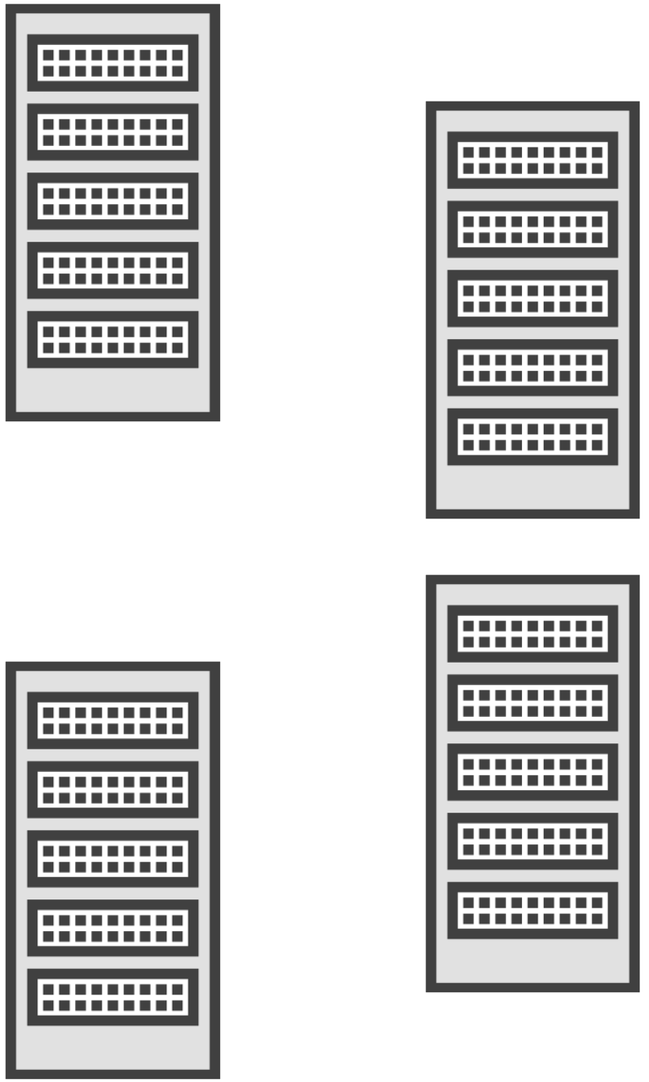
User creates a deployment and submits it to a cluster



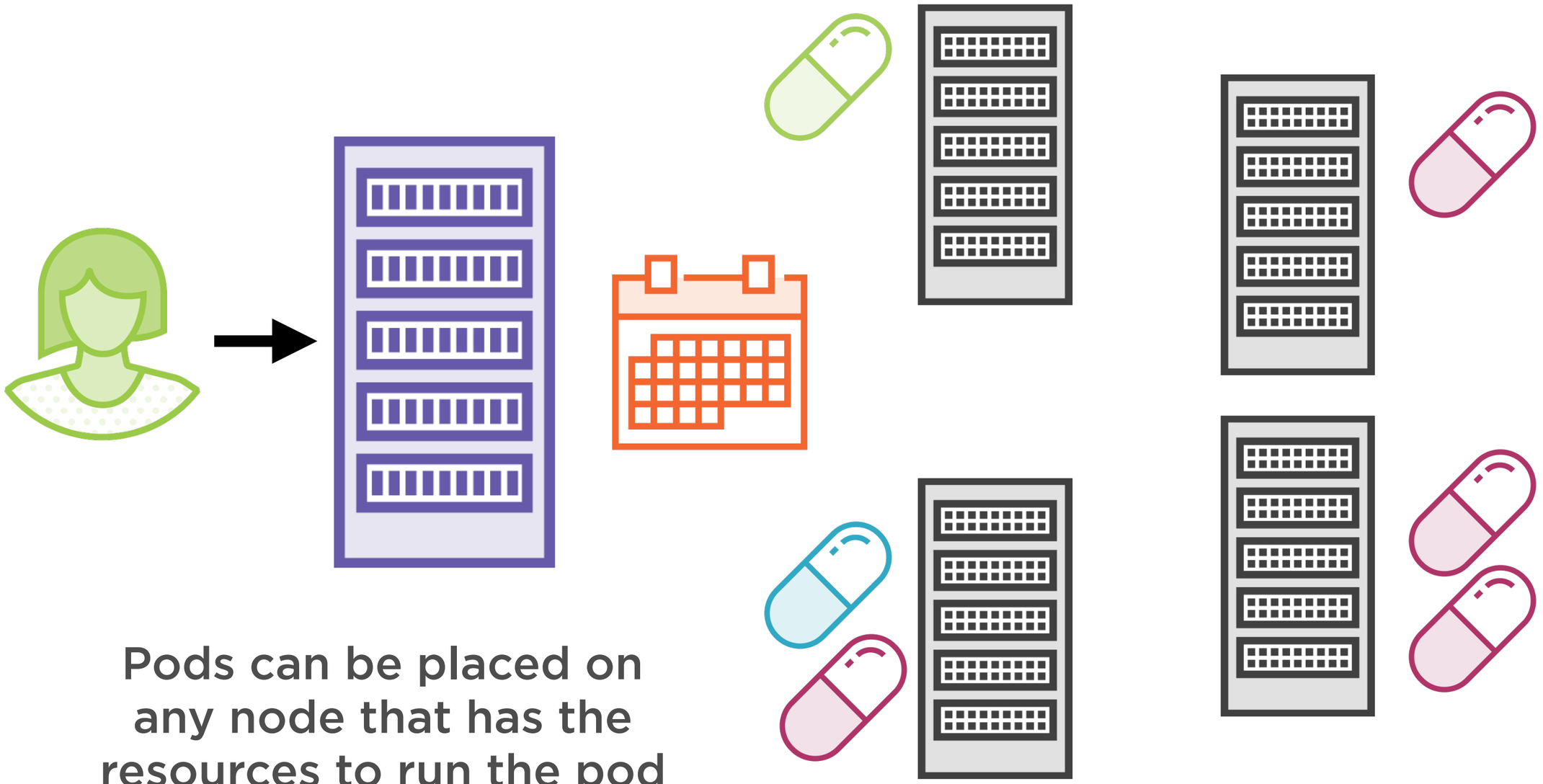
Workload Submitted to a Cluster



The scheduler determines which nodes on the cluster should run the pods



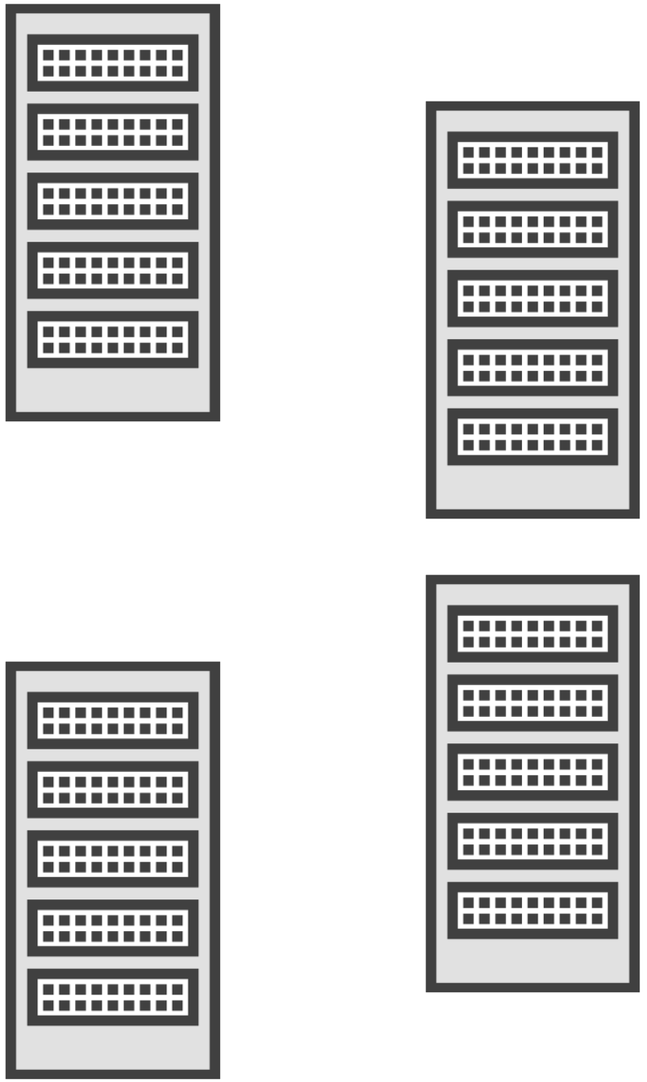
Workload Submitted to a Cluster



Pods can be placed on any node that has the resources to run the pod

Control Where Workloads Run

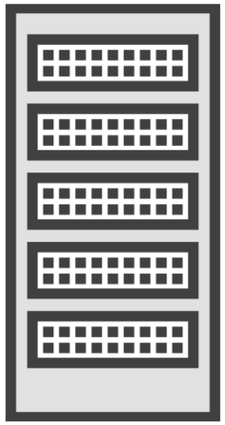
Identify certain nodes in the cluster for types of workloads

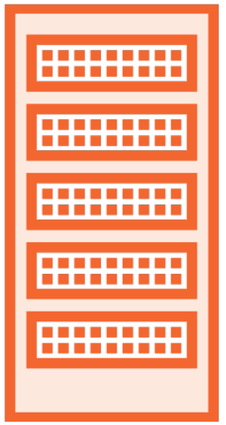


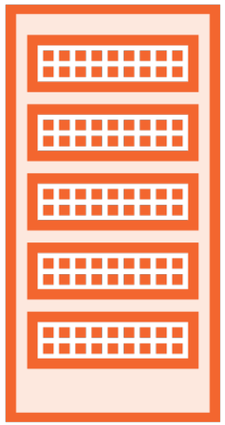
Identification using node taints

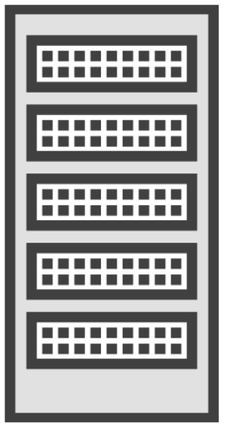
Node Taints

Not all pods can be scheduled on tainted nodes









Pods which are configured with tolerations can be used on tainted nodes

Node Taints

A property of nodes that allow them to repel a set of pods

Control where pods run

Key-value pairs associated with an effect

Node Tolerations

Applied to pods and allow the pods to be scheduled on tainted nodes

Node Taints

`NoSchedule`: **Pods are not scheduled on this node unless they tolerate this taint**

`PreferNoSchedule`: **Avoid scheduling pods unless they tolerate this taint**

`NoExecute`: **Evict running pods and do not schedule new ones**

Toleration in a PodSpec

```
tolerations:
- key: env
  operator: Equal
  value: test
  effect: NoSchedule
```

Allows this pod to be scheduled on a node which has the taint:

`env=test:NoSchedule`

Applying Node Taints on the GKE

Taint all nodes on the cluster

Taint only nodes which belong to a node pool

Taint individual nodes in the cluster

Taints and Tolerations Use Cases

Allow nodes to be dedicated to a certain set of users

Keep regular applications away from nodes which have specialized hardware

Separate real-time and batch operations

Demo

Taint the preemptible node with NoSchedule

Create a CronJob that runs at a predefined time

Observe that the CronJob will not run on the tainted node

Stateless and Stateful Applications

Two Types of Applications

Stateless applications
do not store state in
persistent storage

Stateful applications
use persistent storage
to save data

Stateless Applications

Data and application state stay with the client

Not stored to clusters or persistent storage

Can scale by simply adding more replicas

Stateless Applications

Frontend code typically stateless

Get more pods up and running to handle increasing traffic

Deployed using the Deployment controller

Pods are uniform and non-unique

Stateless Applications

Deployments simply specify the desired state of application

Number of pods, container version, pod labels

Change YAML specification to update state

Stateful Applications

Save data to persistent storage for use by other services

Scaling more involved might need to consider read/write latencies

StatefulSets

A set of pods, similar to ReplicaSet

Important difference from ReplicaSet

- Pods created unique
- Identified by name
- Not interchangeable
- Always associated with persistent volume

Demo

Deploy a stateless application to the cluster using Deployments

Deploy a stateful application to the cluster using StatefulSets

Demo

Using a service account to authenticate to other GCP services e.g. Pub/Sub

Storing and accessing secrets on Kubernetes clusters

Secrets

Objects holding sensitive information

Store passwords, tokens, keys in a cluster

Safer than putting it in a PodSpec or an image

Can be created by users or the system

To use a secret, a pod needs to reference the secret

Summary

Leveraging higher level abstractions to control deployments on Kubernetes

Defining and running Jobs and CronJobs on clusters

Using node taints to control scheduling

Deploying stateless and stateful applications