Kubernetes Installation and Configuration Fundamentals

INTRODUCTION AND EXPLORING KUBERNETES ARCHITECTURE

Anthony E. Nocentino
ENTERPRISE ARCHITECT @ CENTINO SYSTEMS

@nocentino   www.centinosystems.com

Course Overview

Introduction

Exploring Kubernetes Architecture

Installing and Configuring Kubernetes

Working with Your Kubernetes Cluster

Overview

What is Kubernetes?

Exploring Kubernetes Architecture

- Cluster Components
- Networking Fundamentals

What Is Kubernetes?

Container Orchestrator

orchestrator:
start and stop container-based applications based on system admin requirements.

Workload Placement

we can define where the container placed.

Infrastructure Abstraction

don't need to worry about where the container is placed.
ex: load balancer attached to container (running in any server)

Desired State

Kubernetes job that make sure that manifest files in desired state

Benefits of Using Kubernetes

Speed of deployment

Ability to absorb change quickly

Ability to recover quickly

Hide complexity in the cluster

Kubernetes Principles

Desired State/
Declarative
Configuration

Controllers/
Control Loops

Kubernetes API/The
API Server

controllers or control loops have the responsibility of constantly monitoring the running state of the system to make sure that the system is in that desired state

Kubernetes API

The API Server is the central communication hub for information in a Kubernetes cluster. This is where we, as administrators and developers, interact with Kubernetes to deploy and manage workloads. And that's also where the components of a Kubernetes cluster interact with each other to understand the current state of the system and to make changes to that state, if needed to ensure the desired state.

- API Objects
  - Collection of primitives to represent your system’s state
- Enables configuration of state
  - Declaratively
  - Imperatively

Kubernetes API Server

RESTful API over HTTP using JSON

The sole way to interact with your cluster

The sole way Kubernetes interacts with your cluster

Serialized and persisted

Kubernetes API Objects

Pods

Pods are a single or a collection of containers that we deploy as a single unit.

Controllers

controllers are the API object that keep our system in the desired state. So, things like Replica Sets and deployments.

Services

persistent access point to the applications

Storage

persistent stortage to the applications

Not an exhaustive list, but these are the key players

Pods

One or more containers

It’s your application or service

The most basic unit of work

Unit of scheduling

Kubernetes replace the pod

Ephemeral - no Pod is ever “redeployed”

Atomicity - they’re there or NOT

Pods - Continued

Kubernetes’ job is keeping your Pods running

More specifically keeping the desired state

State - is the Pod up and running

Health - is the application in the Pod running

Probes

So how does Kubernetes manage my Pods' state?

Controllers

Defines your desired state

Create and manage Pods for you

Respond to Pod state and health

`ReplicaSet`

- Number of replicas

`Deployment`

- Manage rollout of `ReplicaSets`

Many more…and not just Pods

So how does Kubernetes add persistency to all this ephemerality?

Services

services provide a persistent access point to the applications that we deploy in Pods because, as things change under the hood and our Pods get redeployed perhaps by our controllers as they come up and down, those things will be constantly changing. Well, it's the service's responsibility to provide a persistent access point to the applications provided by our Pods

Adds persistency to our ephemeral world

Networking abstraction for Pod access

IP and DNS name for the Service

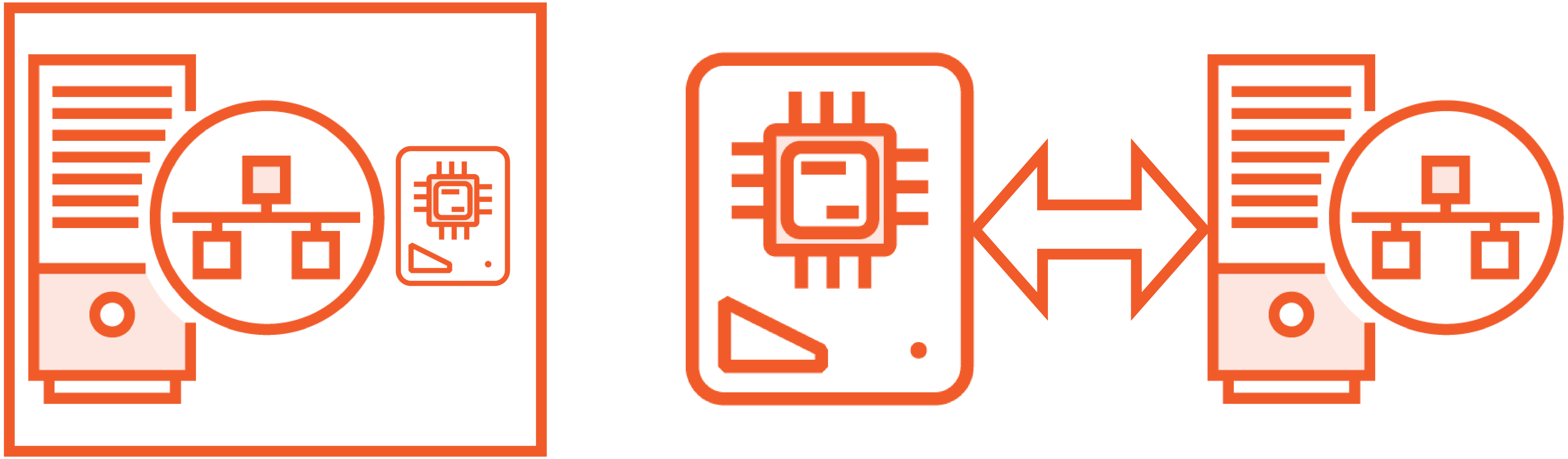
Dynamically updated based on Pod lifecycle

Scaled by adding/removing Pods

Load balancing

What about my data?
Where’s that stored in Kubernetes?

Storage in Kubernetes



Volumes

Persistent Volume

Persistent Volume Claim

persistent volume is pod independent storage that's defined by the administrator at the cluster level. And so when a pod wants access to that storage, it defines what's called a persistent volume claim.

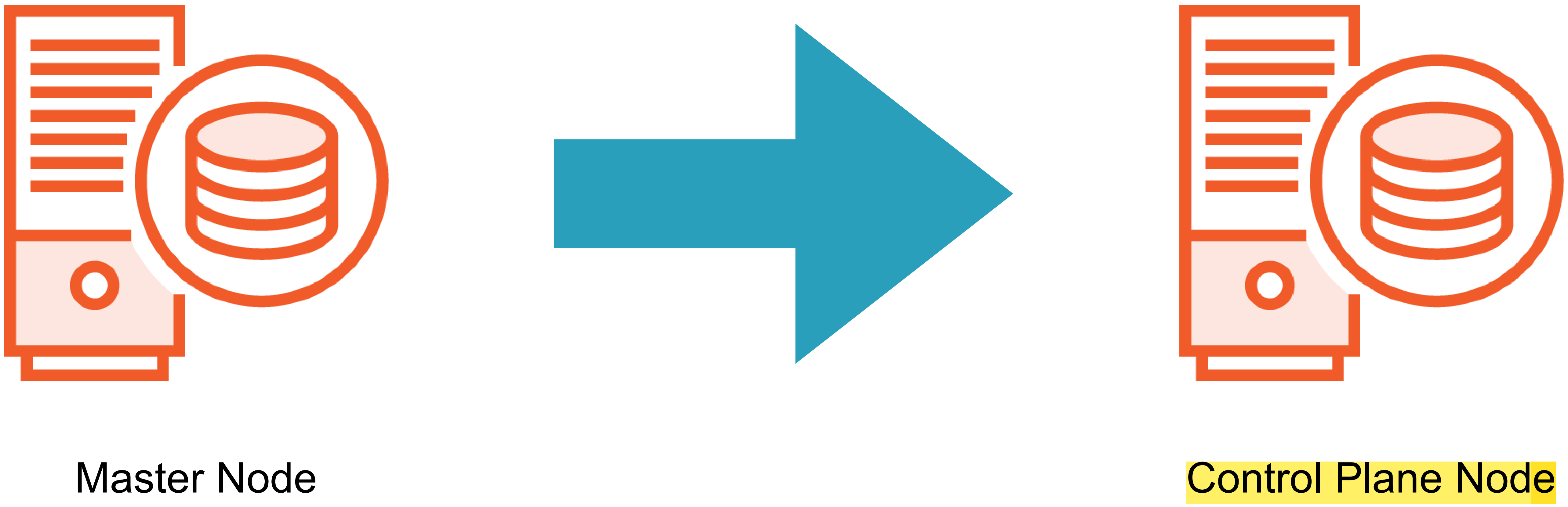
Exploring Kubernetes Architecture

Cluster Components

Control Plane Node

Node

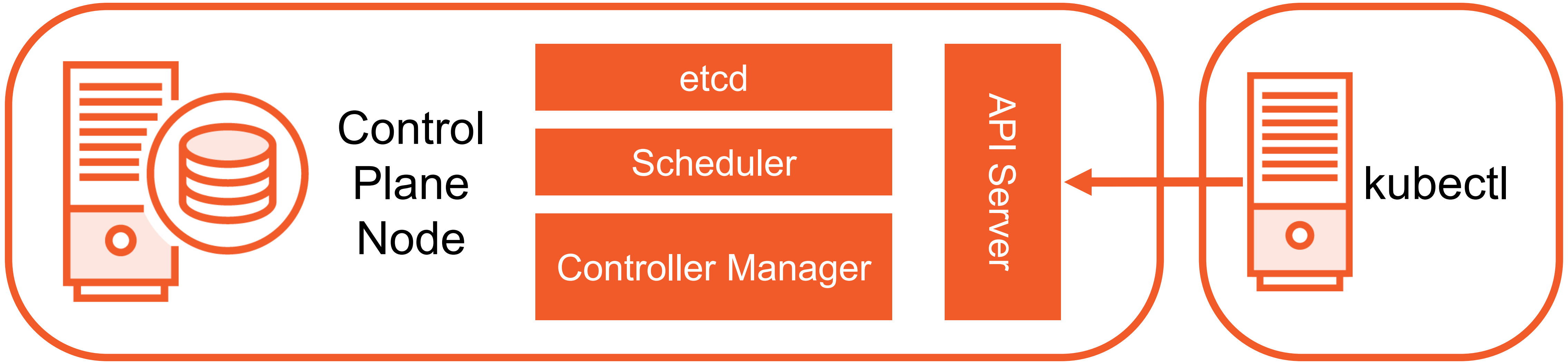
Control Plane Node



Master Node

Control Plane Node

Control Plane Node

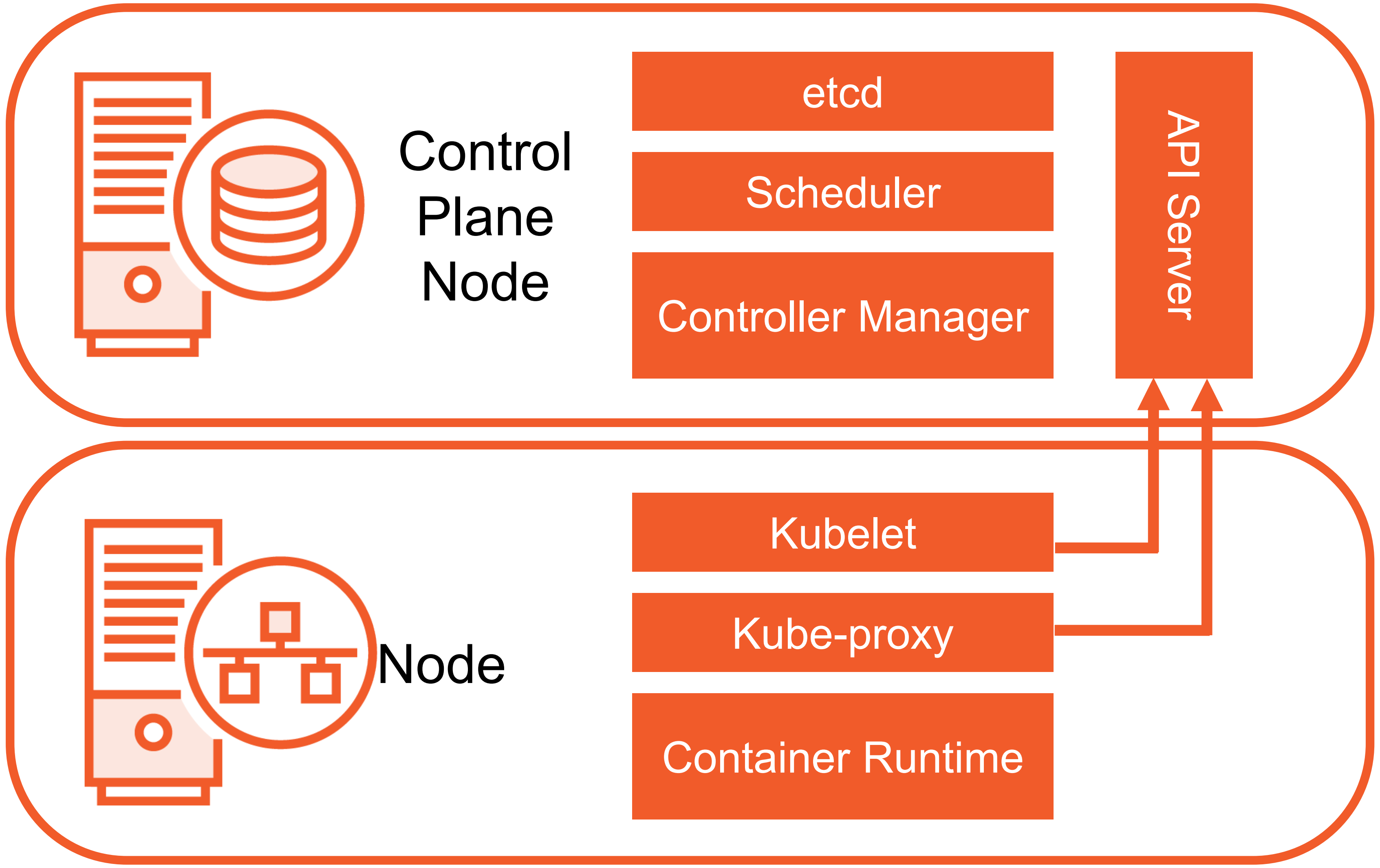


Control Plane Components

API Server	etcd	Scheduler	Controller Manager
Central	Persists State	Watches API Server	Controller Loops
Simple	API Objects	Schedules Pods	Lifecycle functions and desired state
RESTful	Key-value	Resources	Watch and update the API Server
Updates etcd		Respects contraints	ReplicaSet

the scheduler has the responsibility of respecting any constraints that we defined administratively,
Pod affinity: two pods on the same node at all times.
Pod antiinfinity: we want to do the opposite, where we want to ensure that two pods are never on the same node.

Nodes



kubelet: Responsible for Pod Lifecycle (starting/stoping of Pods on the node)
kubeproxy: which has the responsibility for Pod networking and implementing our services abstraction on the node itself.
container runtime: This is the actual runtime environment for our containers.

On All Nodes!

Nodes

Kubelet	kube-proxy	Container Runtime
Monitors API Server for changes	iptables	Downloads images & runs containers
Responsible for Pod Lifecycle	Implements Services	Container Runtime Interface (CRI)
Reports Node & Pod state	Routing traffic to Pods	containerd
Pod probes	Load Balancing	Many others…

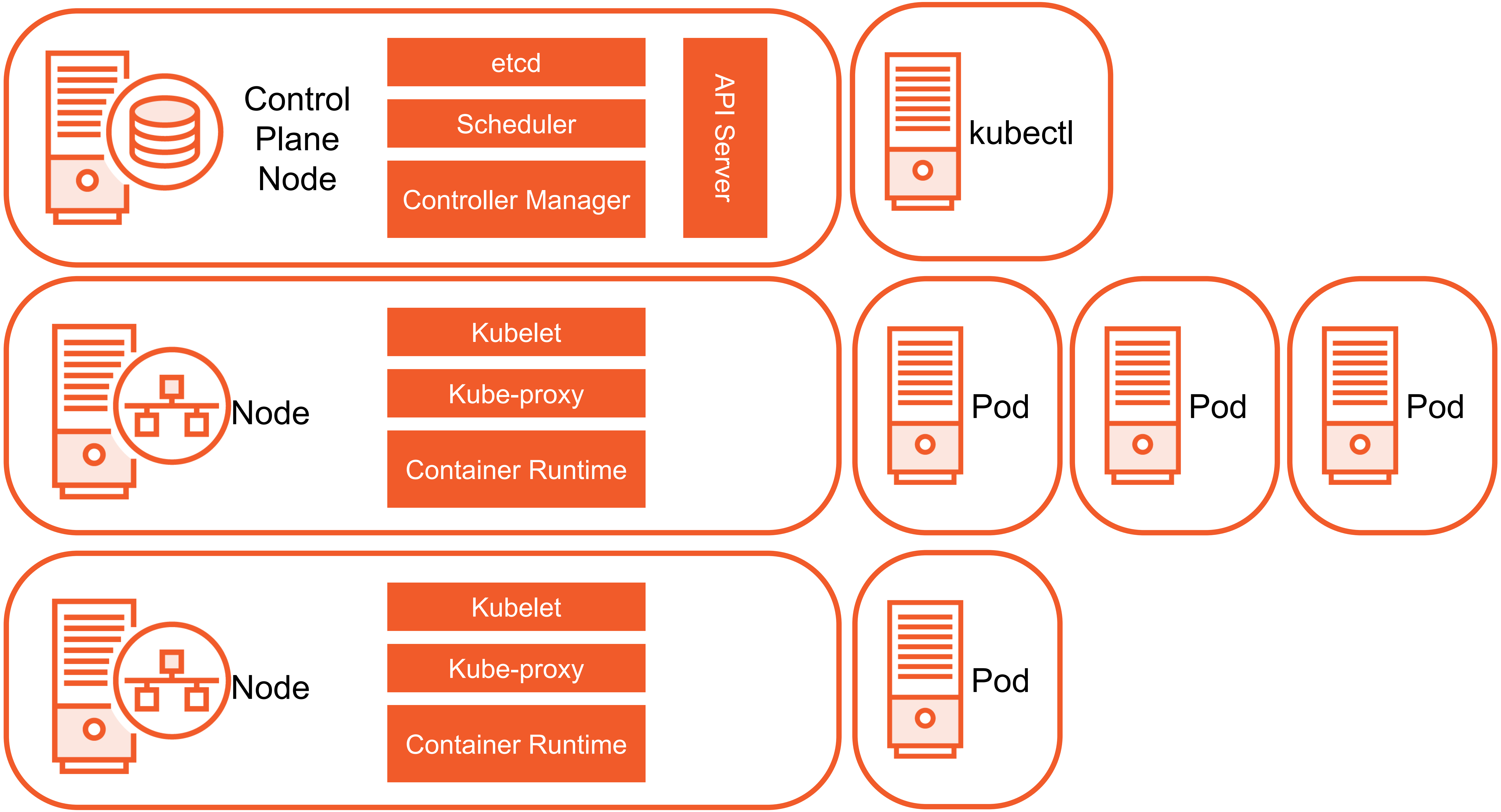
Cluster Add-on Pods

DNS

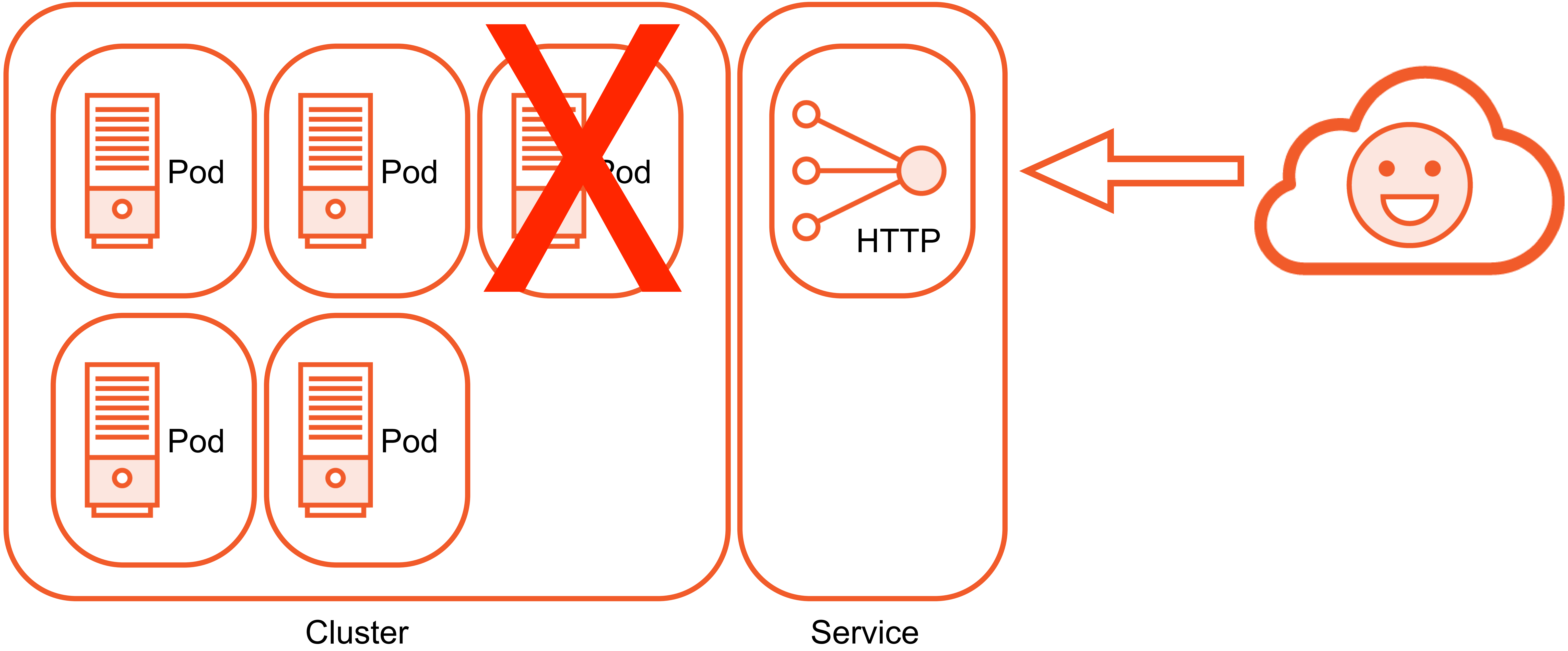
Ingress

Dashboard

Pod Operations



Services



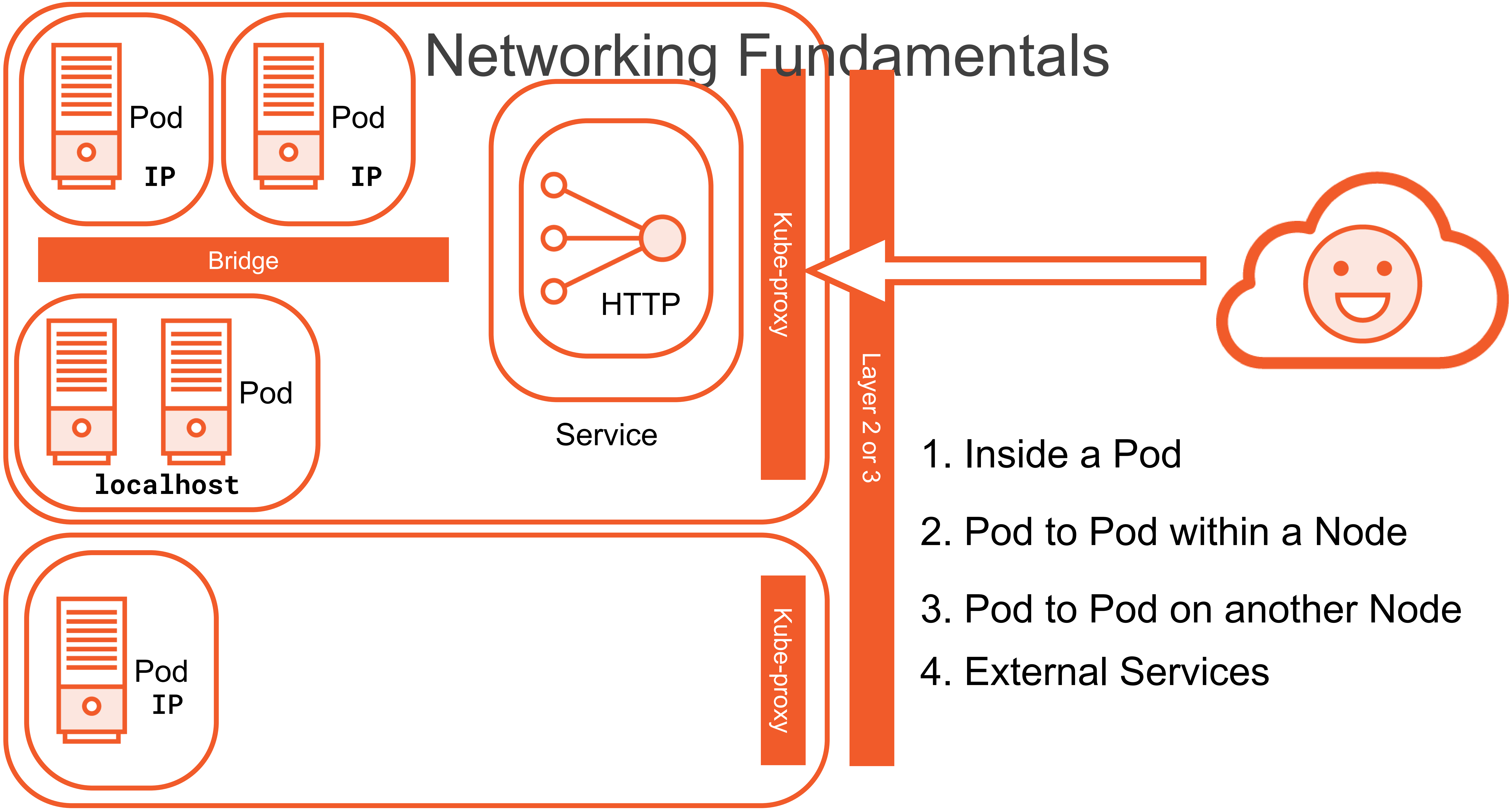
Kubernetes Networking Fundamentals

Kubernetes Networking Requirements

Pods on a Node can communicate with all Pods on all Nodes without Network Address Translation (NAT)

Agents on a Node can communicate with all Pods on that Node

Networking Fundamentals



1. Inside a Pod
2. Pod to Pod within a Node
3. Pod to Pod on another Node
4. External Services

Summary

What is Kubernetes?

Exploring Kubernetes Architecture

- Cluster Components
- Networking Fundamentals

What’s Next!

Installing and Configuring Kubernetes