Kubernetes Architecture and API Objects

Anthony E. Nocentino
aen@centinosystems.com

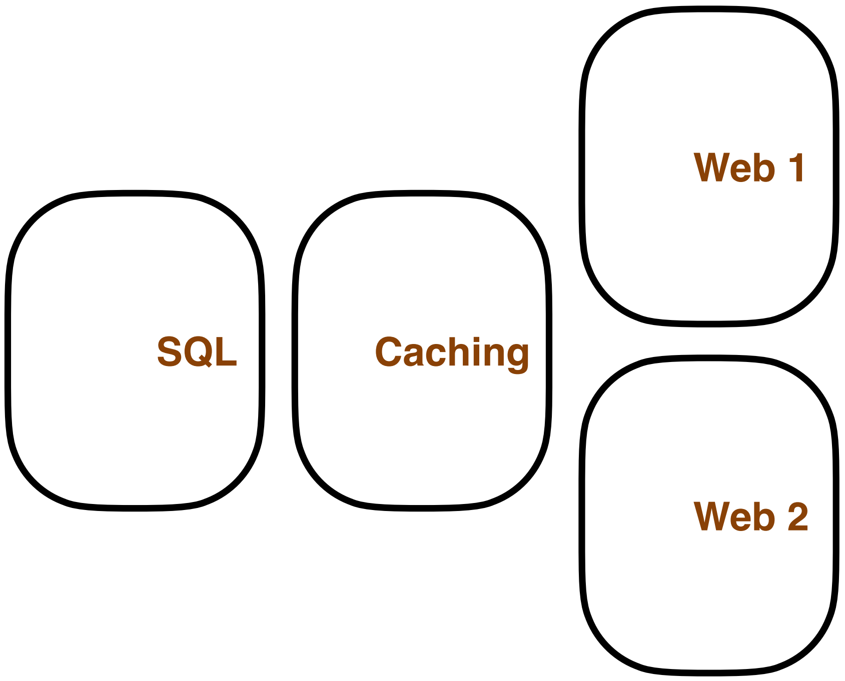
Course Overview

- **Module 0** - Introduction
- **Module 1** - Container Fundamentals
- **Module 2** - Kubernetes Architecture and API Objects
- **Lunch @ 12:00-12:45**
- **Module 3** - Interacting With Your Cluster
- **Module 4** - Deploying Applications in Kubernetes
- **Module 5** - Building and Deploying Container-based Applications in Kubernetes

Agenda

- What’s Kubernetes
- Exploring Kubernetes Architecture
- Core API Primitives
- Cluster Components
- Getting Kubernetes

Modern Application Deployment



- Where do I run the application?
- How do I scale the application?
- How do I consistently deploy?
- How do I or my applications access the services?

What is Kubernetes?

- Container Orchestrator
- Infrastructure Abstraction
- Desired State

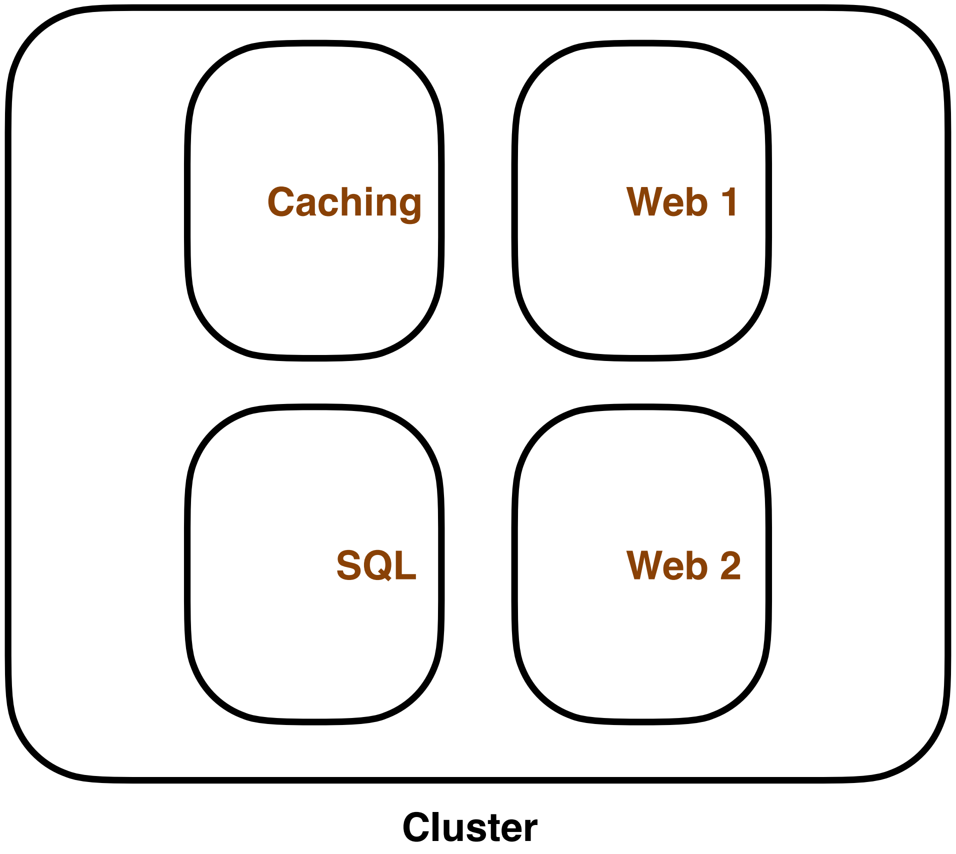
Kubernetes Principles

- Declarative Configuration
- Controllers/Control Loops
- The API Server

Kubernetes Benefits

- Workload placement
- Managing state, starting things up and keeping things up
- Networking and Services
- Load balancing services
- Persistent storage

Kubernetes Cluster



Centino systems

Kubernetes API

- **API Objects** - Represent resources in your system
    - **Pods** - your container based applications
    - **Controllers** - maintain desired state
    - **Services** - persistent access to your apps
    - **Storage** - persistent storage for your data
    - …and more
- **API Server** - Main communication hub

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 in a data store

Pods

- One or more containers
- It’s your application or service
- The most basic unit of work
- Unit of scheduling
- Ephemeral - no Pod is ever “redeployed”
- Atomicity - they’re there or NOT

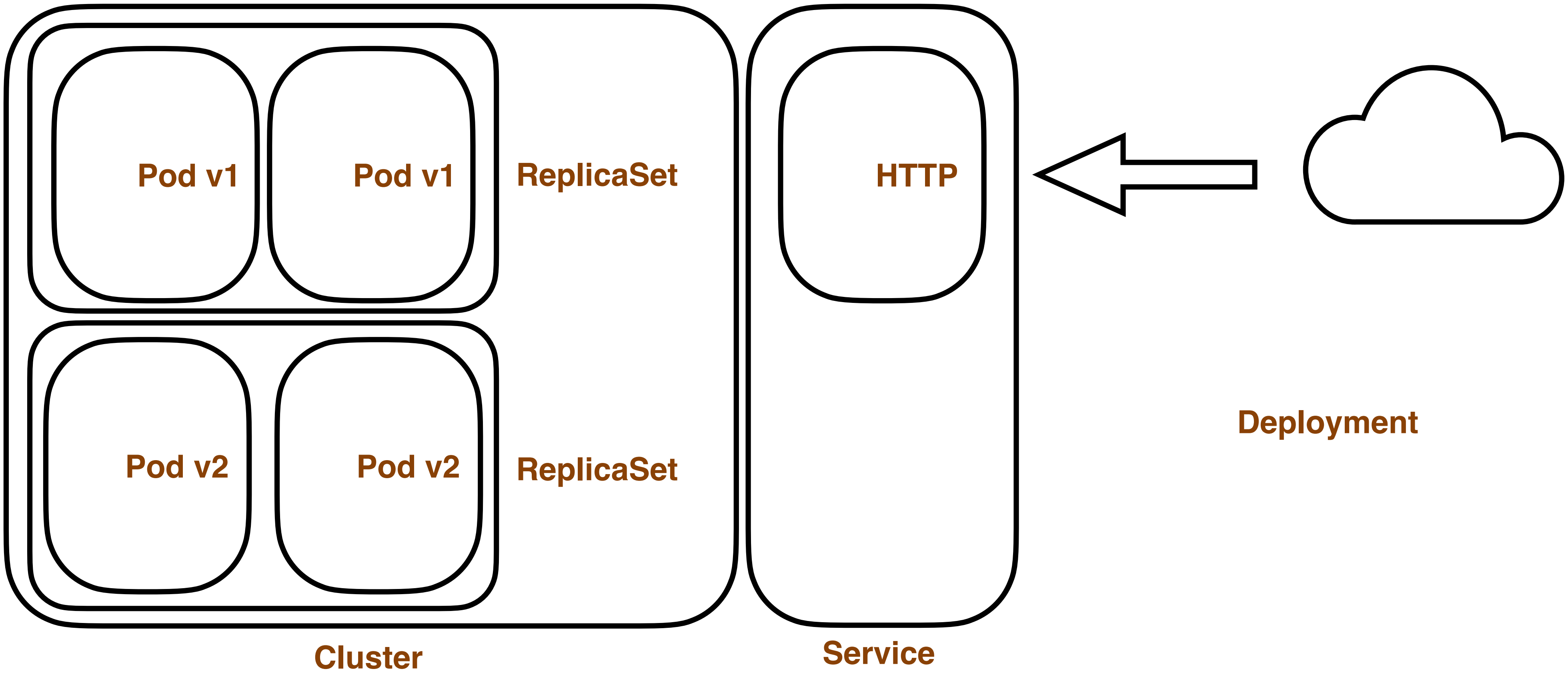
Pods - Continued

- Kubernetes' job is keeping your Pods running
- More specifically keeping the desired state
  - Controllers
  - **State** - is the Pod up and running
  - **Health** - is the application in the Pod running

Controllers

- Create and manage Pods for you
- Define your desired state
- Respond to Pod state and health
- **`ReplicaSet`**
  - Number of replicas
- **`Deployment`**
  - Manage rollout of **`ReplicaSet`**
- Many more…and not just Pods

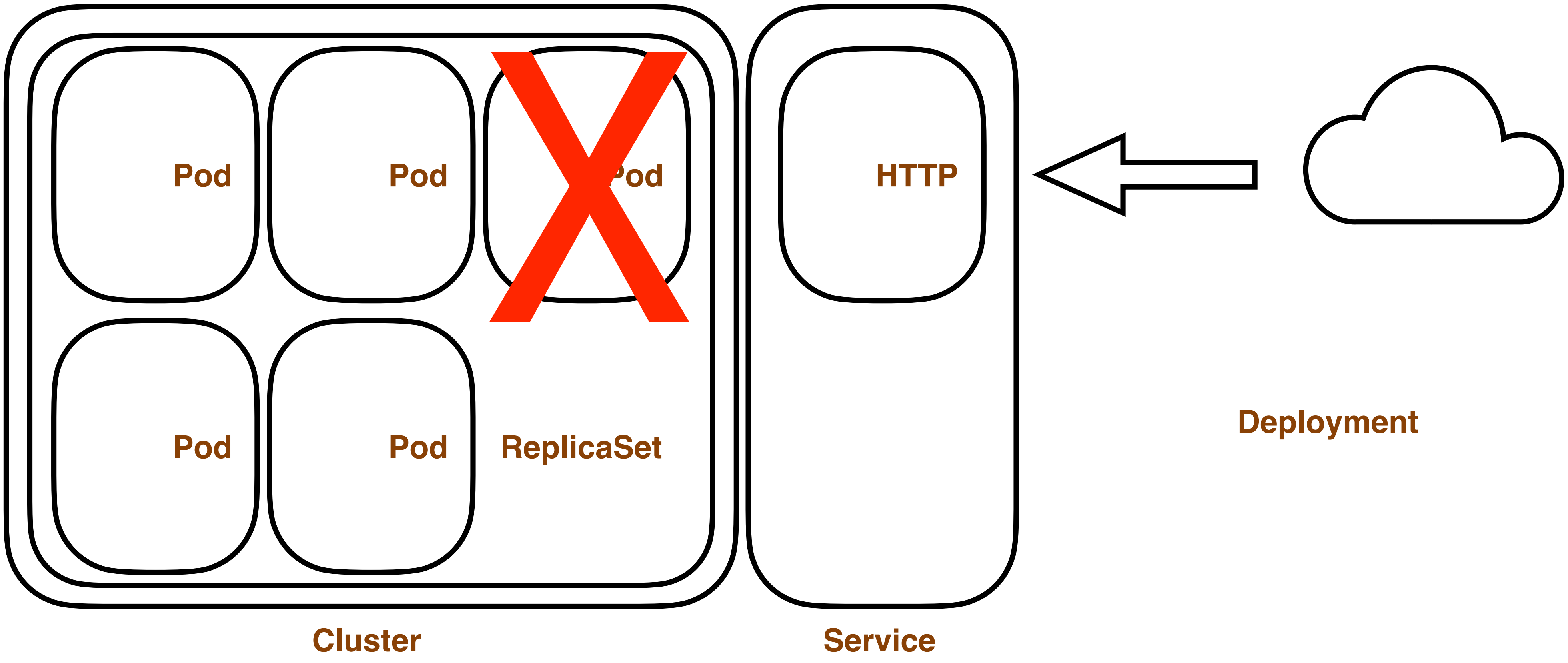
Controller Operations - `Deployment`



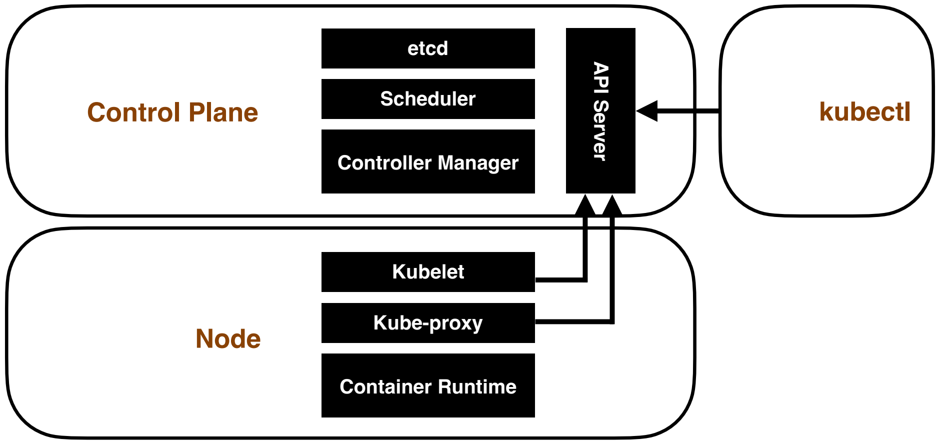
Services

- Adds persistency to our ephemeral world
- Networking abstraction for Pod access
- IP and DNS name for the service
- Load balancing
- Recreated Pods automatically updated
- Scaled by adding/removing Pods

Services and ReplicaSets



Exploring Kubernetes Architecture



Installation Considerations

- Where to install?
- Cloud
  - IaaS - Virtual Machines
  - PaaS - Managed Service
- On-Prem
  - Bare Metal
  - Virtual Machines
- Which one should you choose?

Installation Considerations (con’t)

- Cluster Networking
- Scalability
- High Availability
- Disaster Recovery

Installation Methods

Desktop

`kubeadm`

From Scratch

Cloud Scenarios

https://kubernetes.io/docs/setup/scratch/
https://github.com/kelseyhightower/kubernetes-the-hard-way/

Managed Cloud Deployment Scenarios

Elastic Container Service for Kubernetes (EKS)

https://aws.amazon.com/getting-started/projects/deploy-kubernetes-app-amazon-eks/

Google Kubernetes Engine (GKE)

https://cloud.google.com/kubernetes-engine/docs/how-to/

Azure Kubernetes Services (AKS)

https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough

Demos

- Check out the Docker Desktop Kubernetes Installation

Review

- Exploring Kubernetes Architecture
- Core API Primitives
  - Controllers
  - Pods
  - Services
  - Storage
- Cluster Components
- Getting Kubernetes