

Inside Kubernetes

An Architectural Deep Dive

Anthony E. Nocentino
aen@centinosystems.com



Anthony E. Nocentino

- **Consultant and Trainer**
- **Founder and President of Centino Systems**
 - Specialize in system architecture and performance
 - Masters Computer Science
 - Microsoft MVP - Data Platform - 2017 - 2020
 - Linux Foundation Certified Engineer
 - Friend of Redgate - 2015-2020
- **email:** aen@centinosystems.com
- **Twitter:** @nocentino
- **Blog:** www.centinosystems.com/blog
- **Pluralsight Author:** www.pluralsight.com



Agenda

- What is Kubernetes
- Benefits of Using Kubernetes
- Kubernetes API Objects
- Exploring Kubernetes Architecture
- Deploying Applications
- Deploying SQL Server

What is Kubernetes?

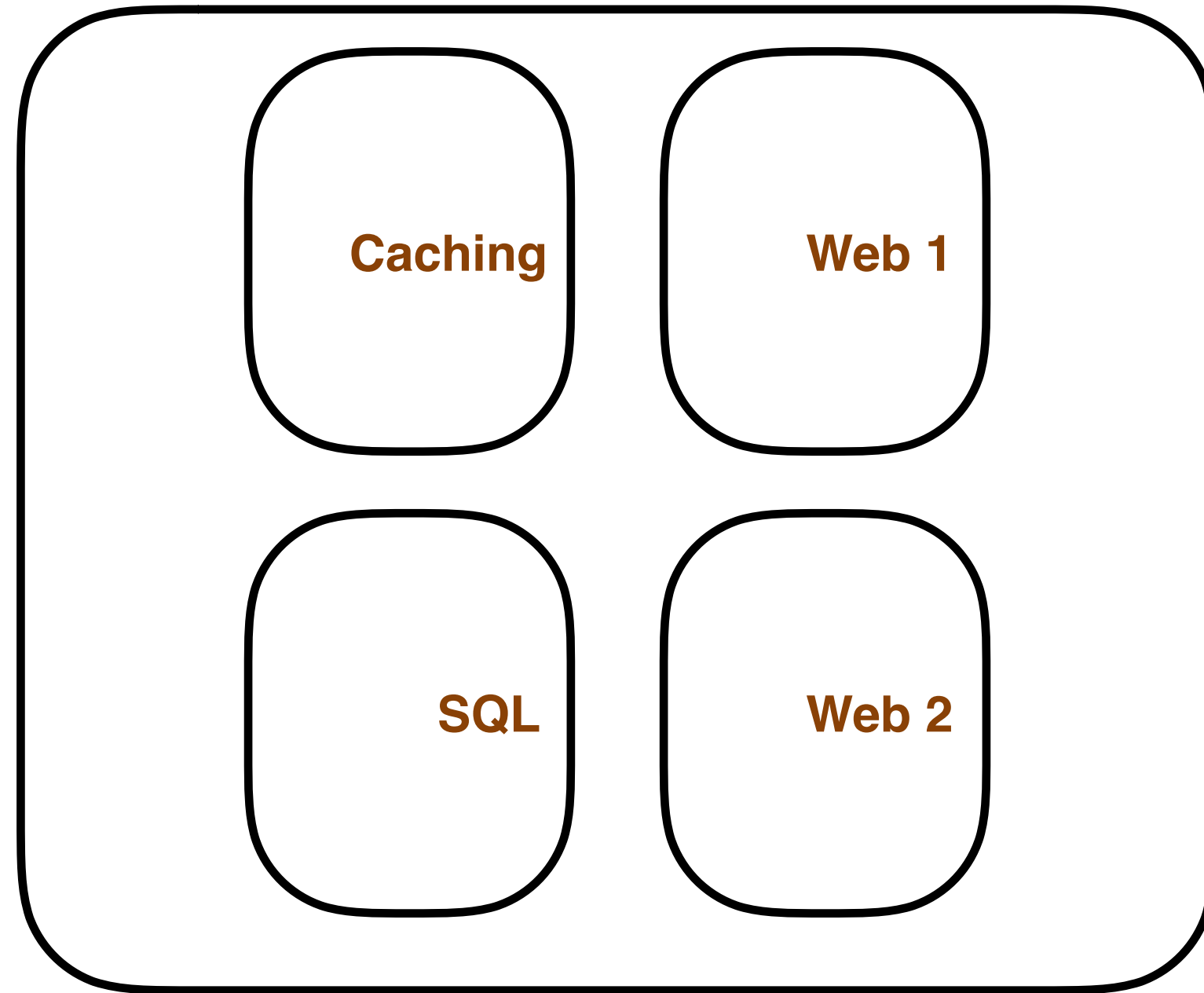
- Container Orchestrator
- Infrastructure Abstraction
- Desired State



Kubernetes Benefits

- Managing state, starting things and keeping them up
- Speed and consistency of deployment
- Ability to absorb change quickly
- Ability to recovery quickly
- Hide complexity in Cluster
- Persistent application access endpoints

Kubernetes Cluster



Cluster

Kubernetes API

- **API Objects** - Represent resources in your system
- **API Server** - Main communication hub
 - Pods
 - Controllers
 - Services
 - Storage
 - ...and more

Pods

- One or more containers
- It's your application
- The most basic unit of work
- Unit of scheduling
- Ephemeral - no Pod is ever “redeployed”

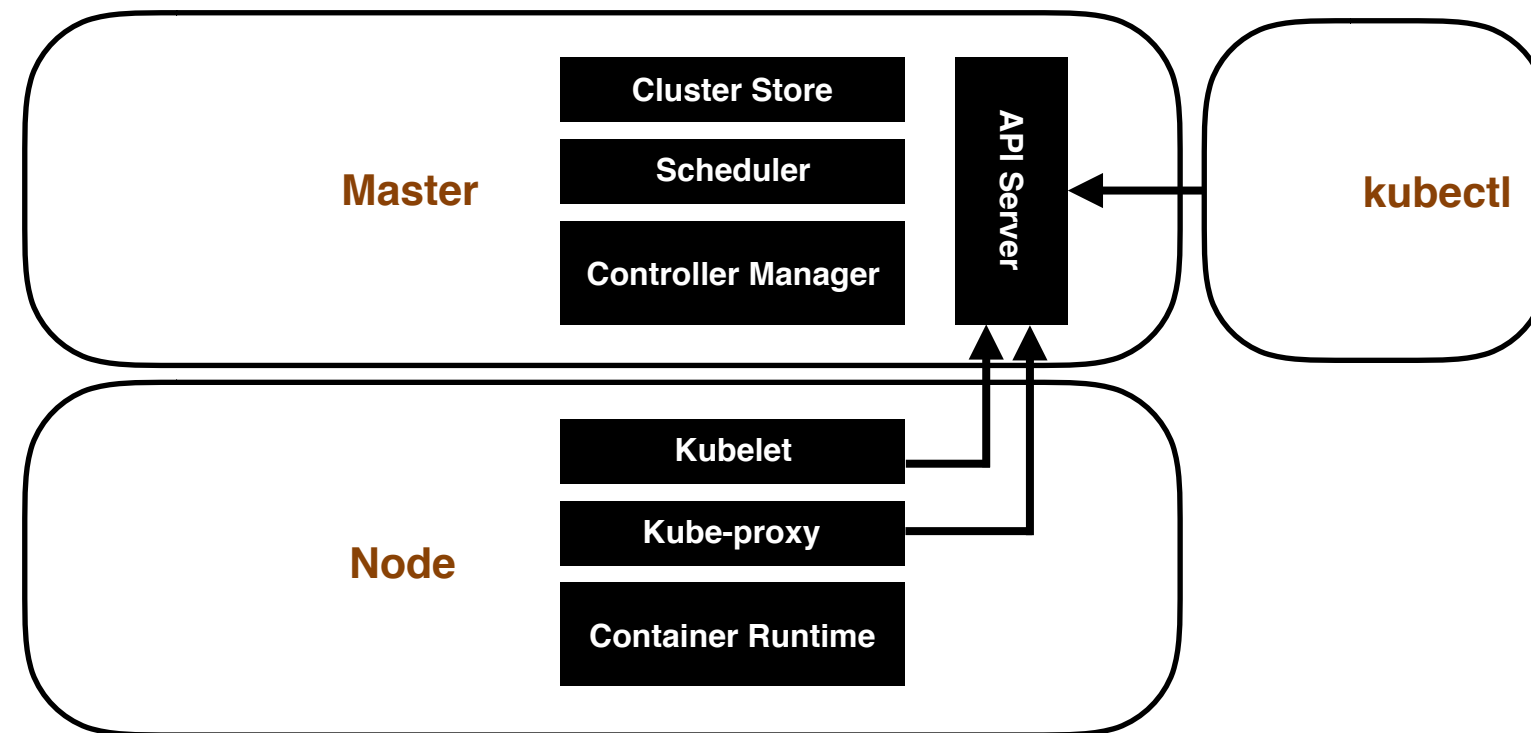
Controllers

- Create and manage Pods for you
- Define your desired state
- Respond to Pod State and Health
- **ReplicaSet**
- **Deployment**

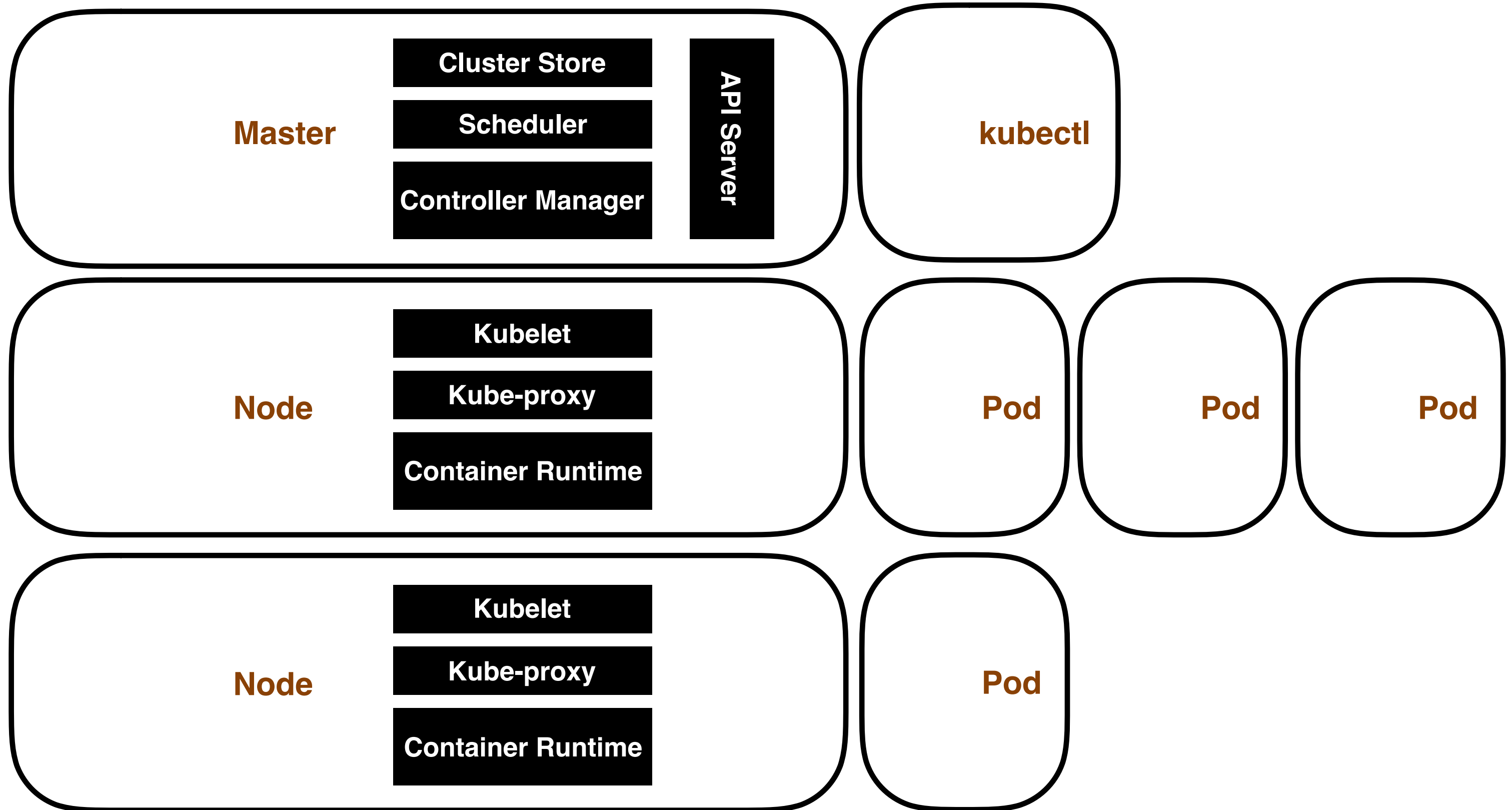
Services

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

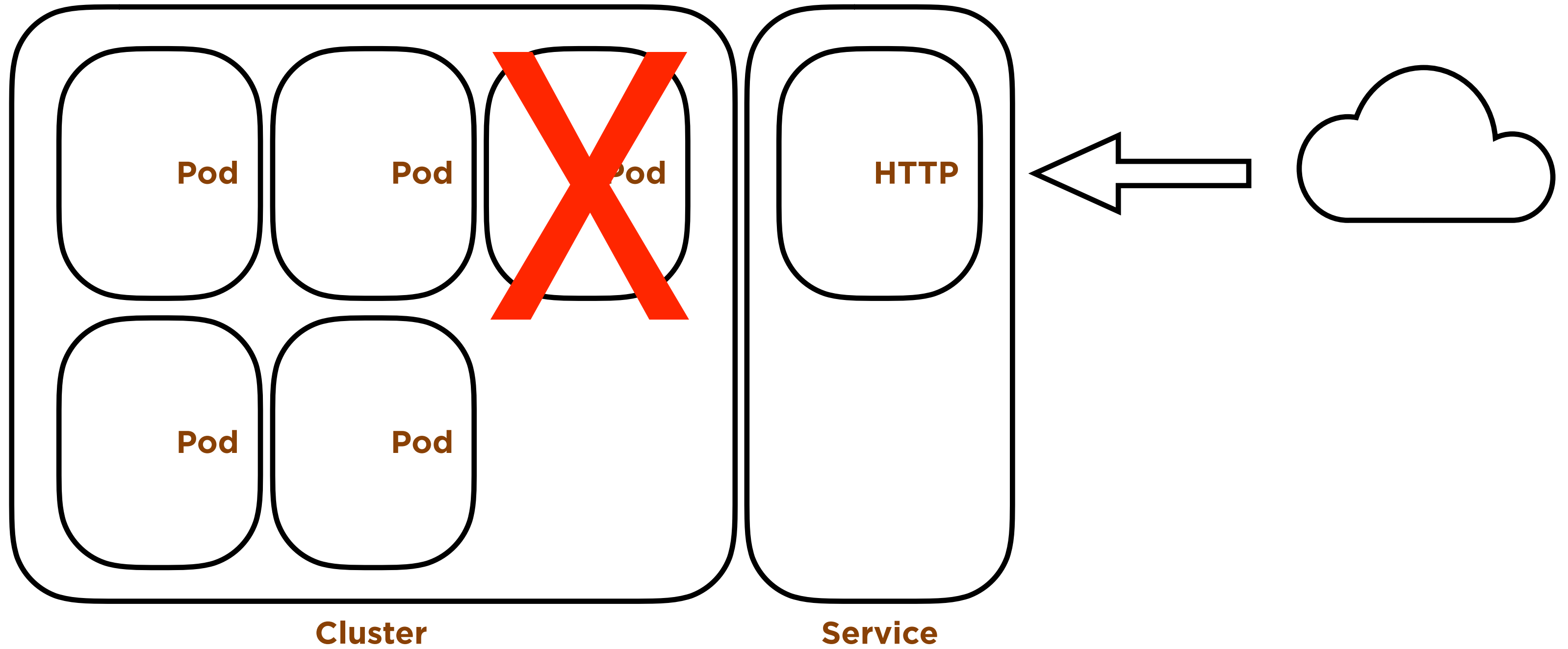
Exploring Kubernetes Architecture



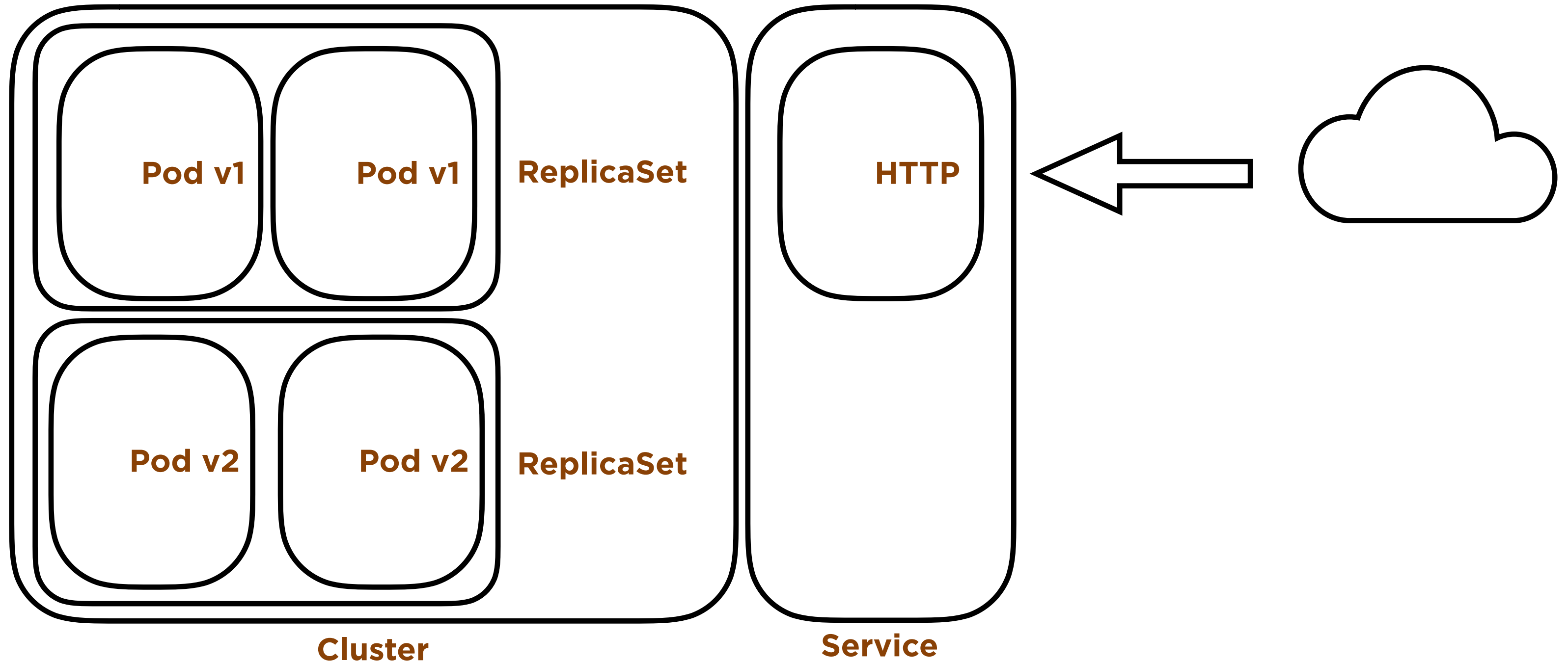
Controller Operation of Pods



Services



Controller Operations - Deployment



Deploying Applications

- Imperative
- Declarative
- YAML and JSON

Declarative Deployment - Manifests

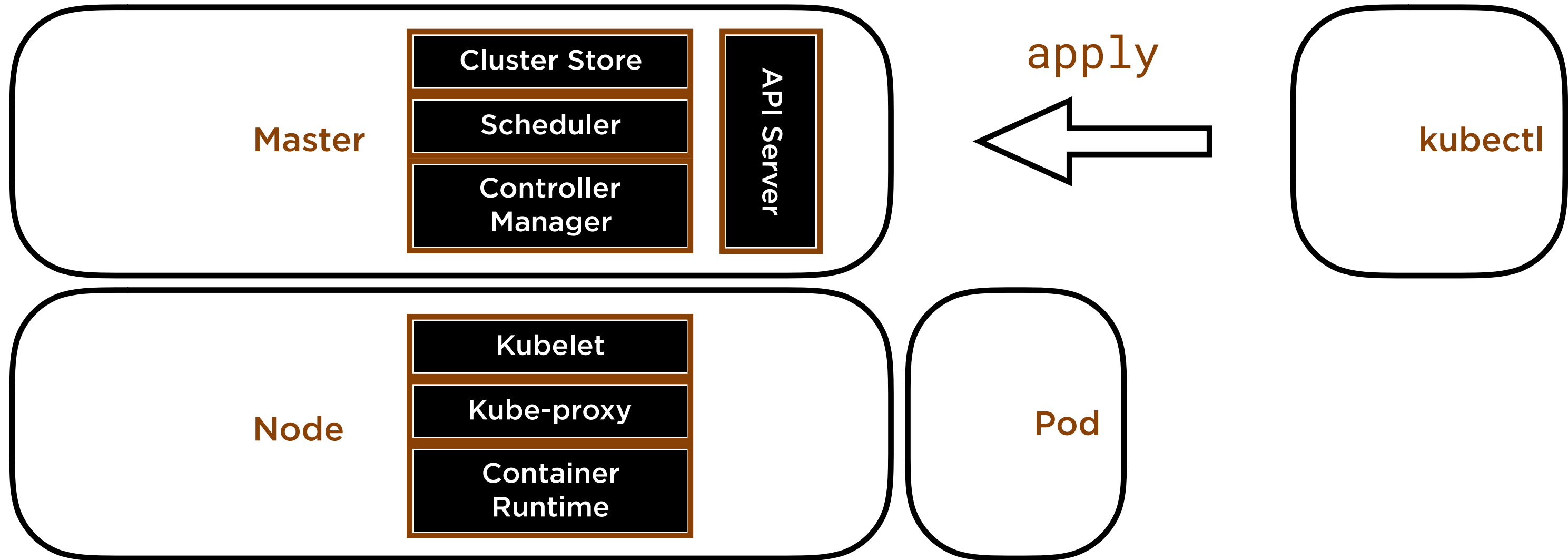
```
apiVersion: app/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - image: gcr.io/google-samples/hello-app:1.0
        name: hello-app
```

Deployment

Pod Template

kubectl apply -f deployment.yaml

Application Deployment Process



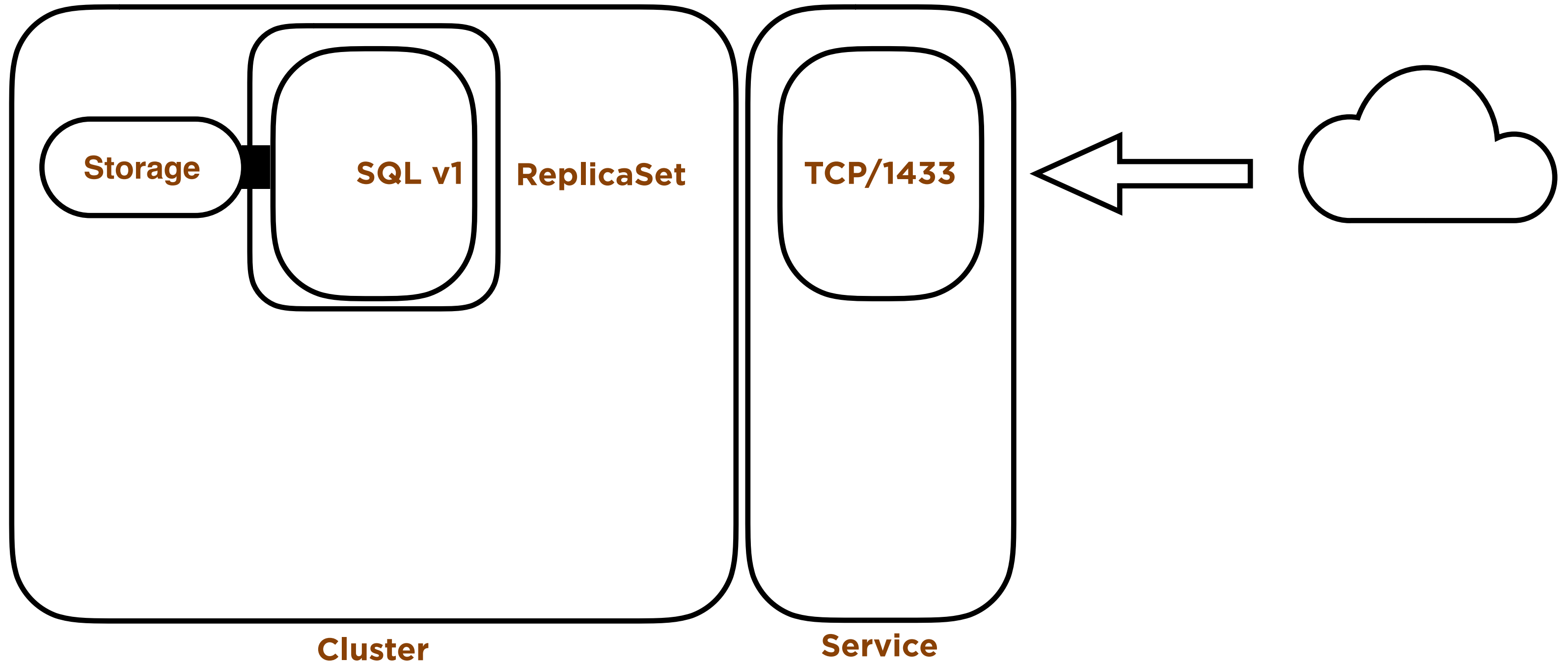
Demo!

- Imperatively deploying a web application
- Accessing Services within a Cluster

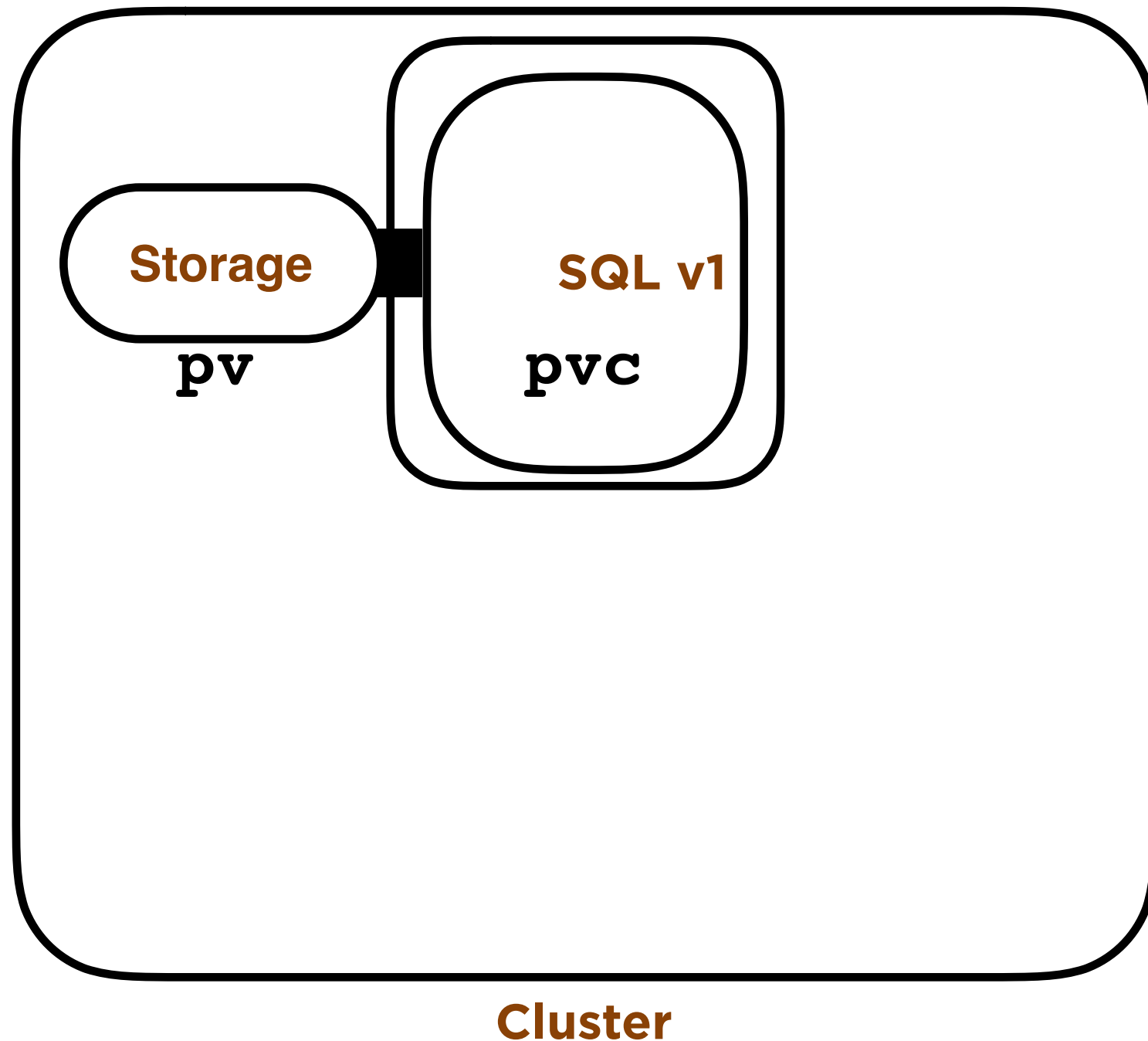
Running SQL Server in Kubernetes

- A Pod goes back to its initial state each time it's deployed
- **State** - where do we store data?
- **Configuration** - how do we configure SQL Server?

Decoupling Data and Computation



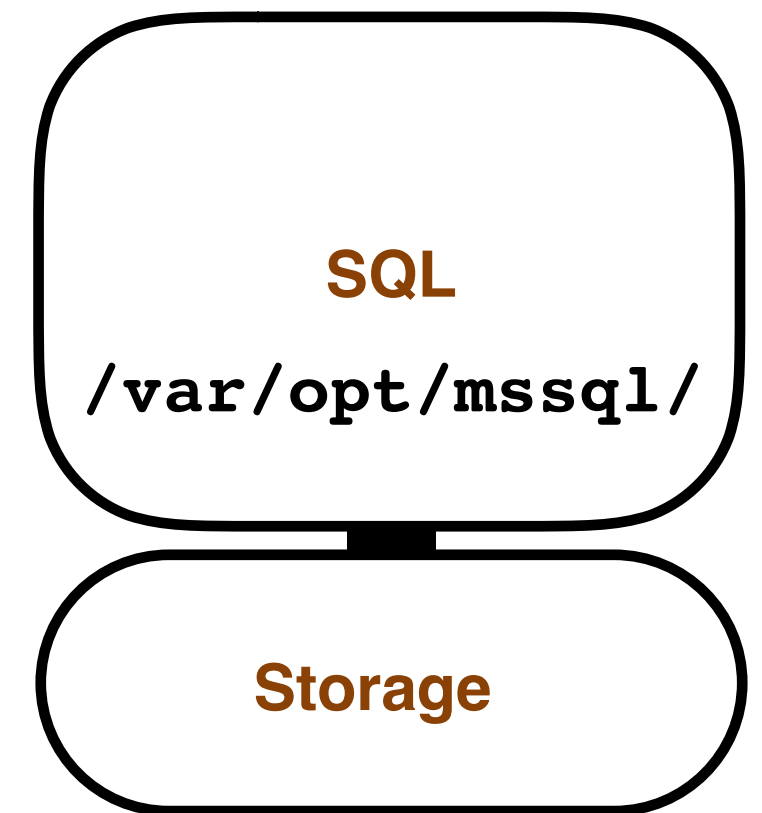
Storage in Kubernetes



- **Persistent Volume (pv)**
 - Administrator defined storage
 - iSCSI, NFS, FC, AzureDisk...many more
- **Persistent Volume Claim (pvc)**
 - The Pod “claims” the **pvc**
 - The **pvc** is mapped to the **pv** by k8s
 - Decouples the Pod and the storage

Data Persistency in SQL Server in K8S

- Define **Persistent Volumes/Persistent Volume Claims**
 - Instance directory (error log, default trace, etc..)
 - **`/var/opt/mssql/`**
 - User Database default directory
 - **`/var/opt/mssql/data`**

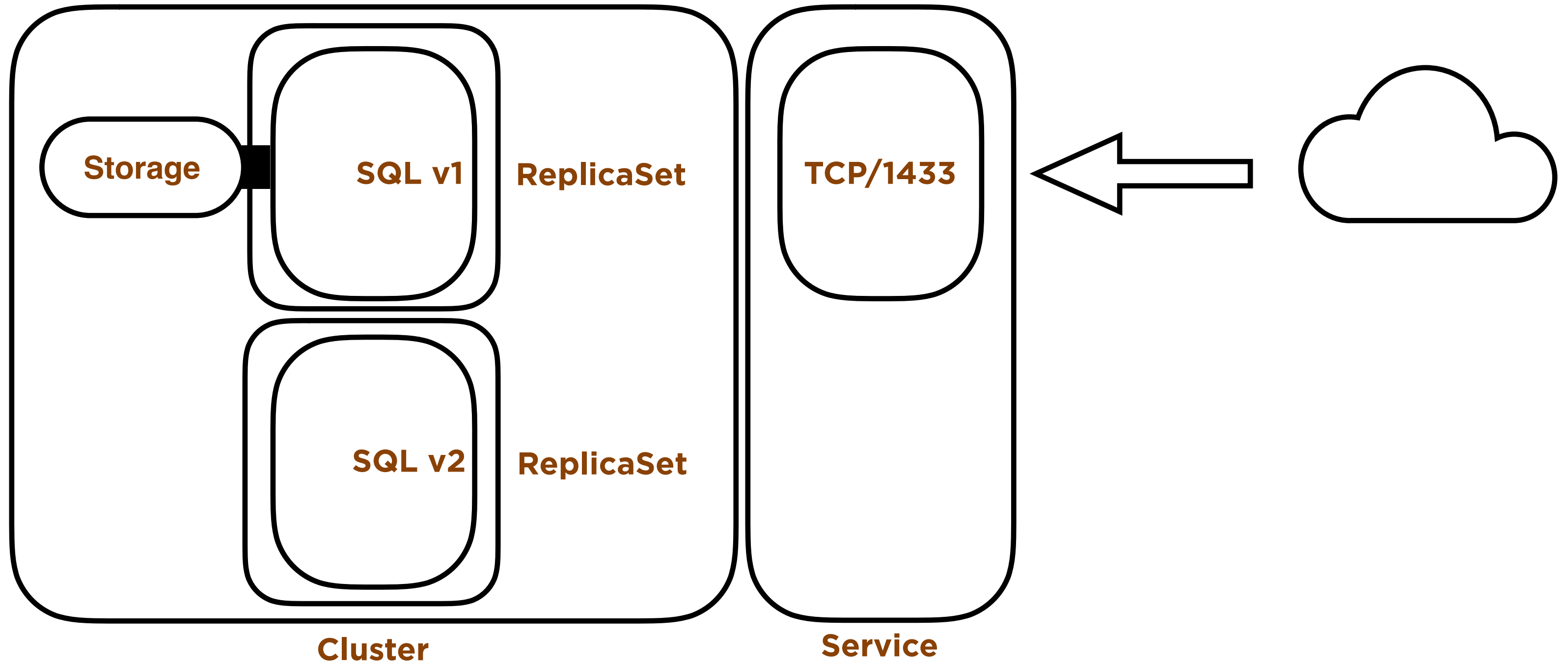


Configuring SQL Server in a Pod

- In our Pod configuration we define **Environment Variables**
 - Used at initial startup to configure the SQL Instance
 - **ACCEPT_EULA**
 - **MSSQL_SA_PASSWORD**
 - Stored in the cluster as a **Secret** (hashed, not encrypted)
 - Pods go back its initial state of the container image on creation

<https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-configure-environment-variables>

High Availability in SQL Server in Kubernetes



Demo

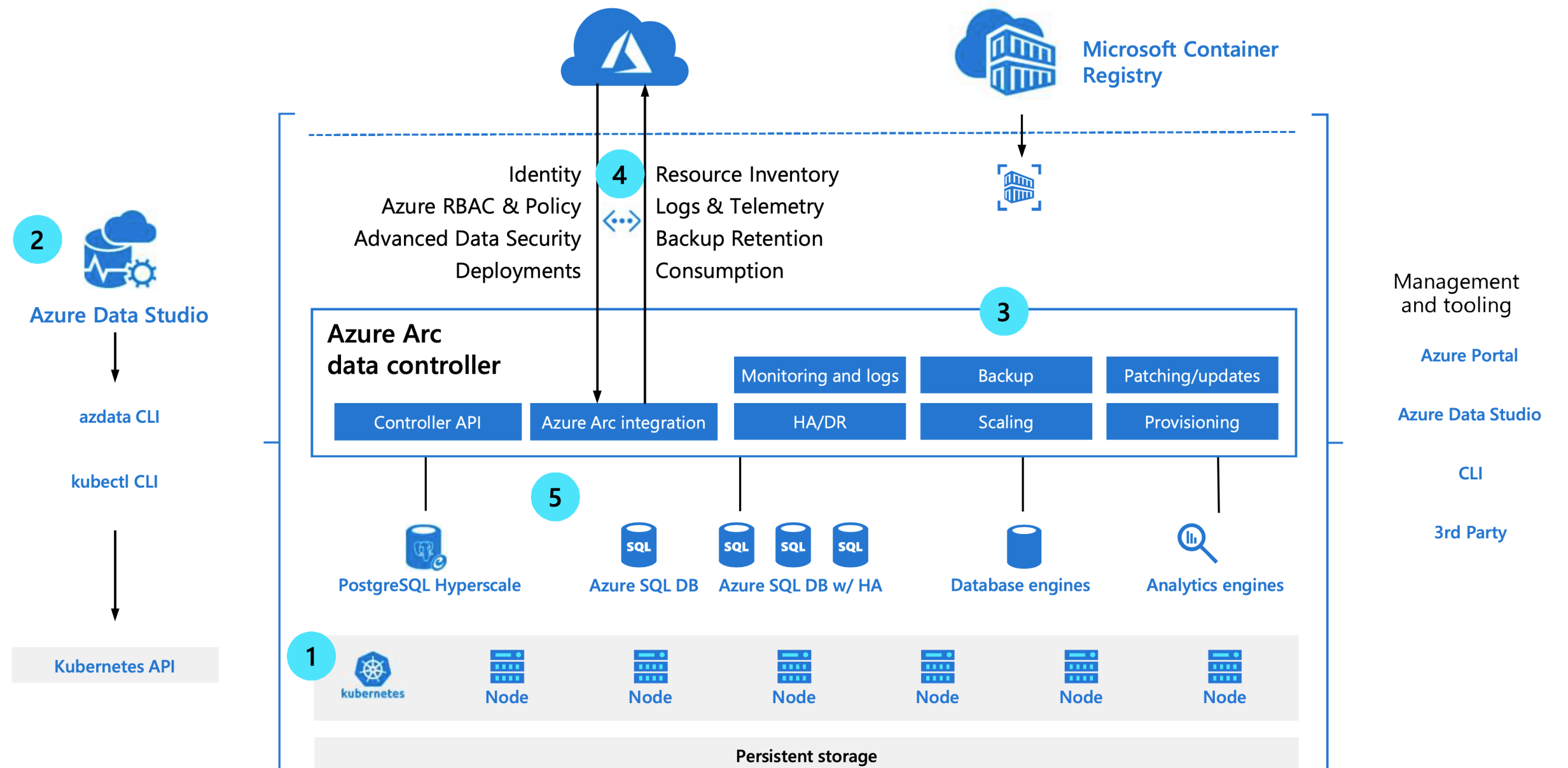
- Deploying SQL Server in a **Deployment** with Persistent Storage
- Recovery Scenario
- Upgrading SQL Server

Azure Arc - Data Services

How it works: architecture of Azure data services on customer infrastructure

A few steps to get Azure data services in your environment:

- 1 Have Kubernetes on your infrastructure
- 2 Prepare environment with APIs and CLIs
- 3 Install Azure Arc data controller
- 4 Connect to Azure
- 5 Deploy and run Azure data services for your workloads



Review

- What is Kubernetes
- Benefits of Using Kubernetes
- Kubernetes API Objects
- Exploring Kubernetes Architecture
- Deploying Applications
- Deploying SQL Server

More Resources

- **Docker for Windows/Mac**
- **Managed Service Providers**
 - Azure Kubernetes Service (**AKS**)
 - <https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough>
 - **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/>
- **Pluralsight**
 - <https://app.pluralsight.com/profile/author/anthony-nocentino>

Need more data or help?

<http://www.centinosystems.com/blog/talks/>

Links to resources

Demos

Presentation

Pluralsight

aen@centinosystems.com

@nocentino

www.centinosystems.com

Solving tough business challenges with technical innovation



Thank You!