

Inside Kubernetes Architecture Fundamentals

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?

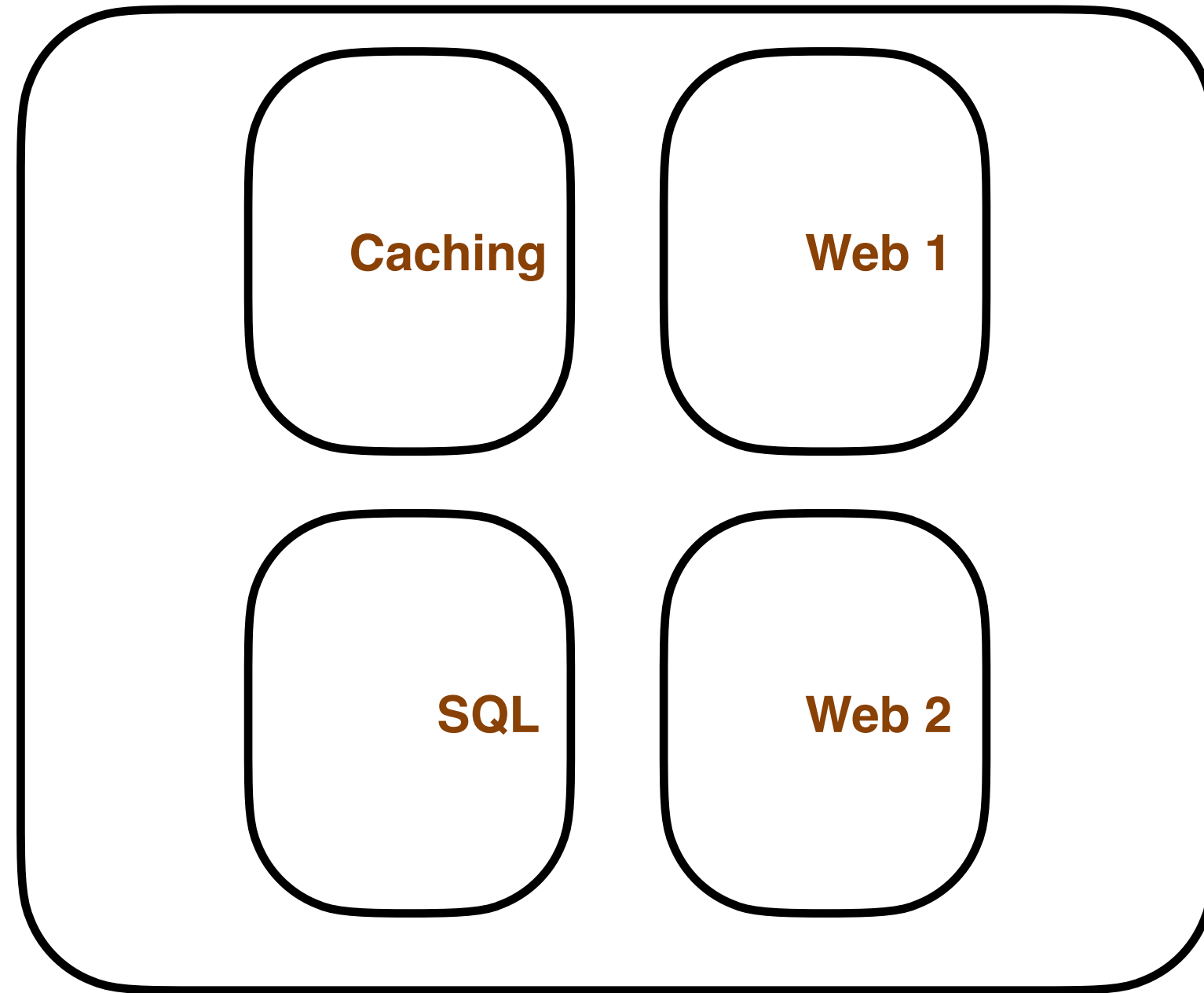
- Container Orchestrator
- Infrastructure Abstraction
- Desired State



Kubernetes Benefits

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

Kubernetes Cluster

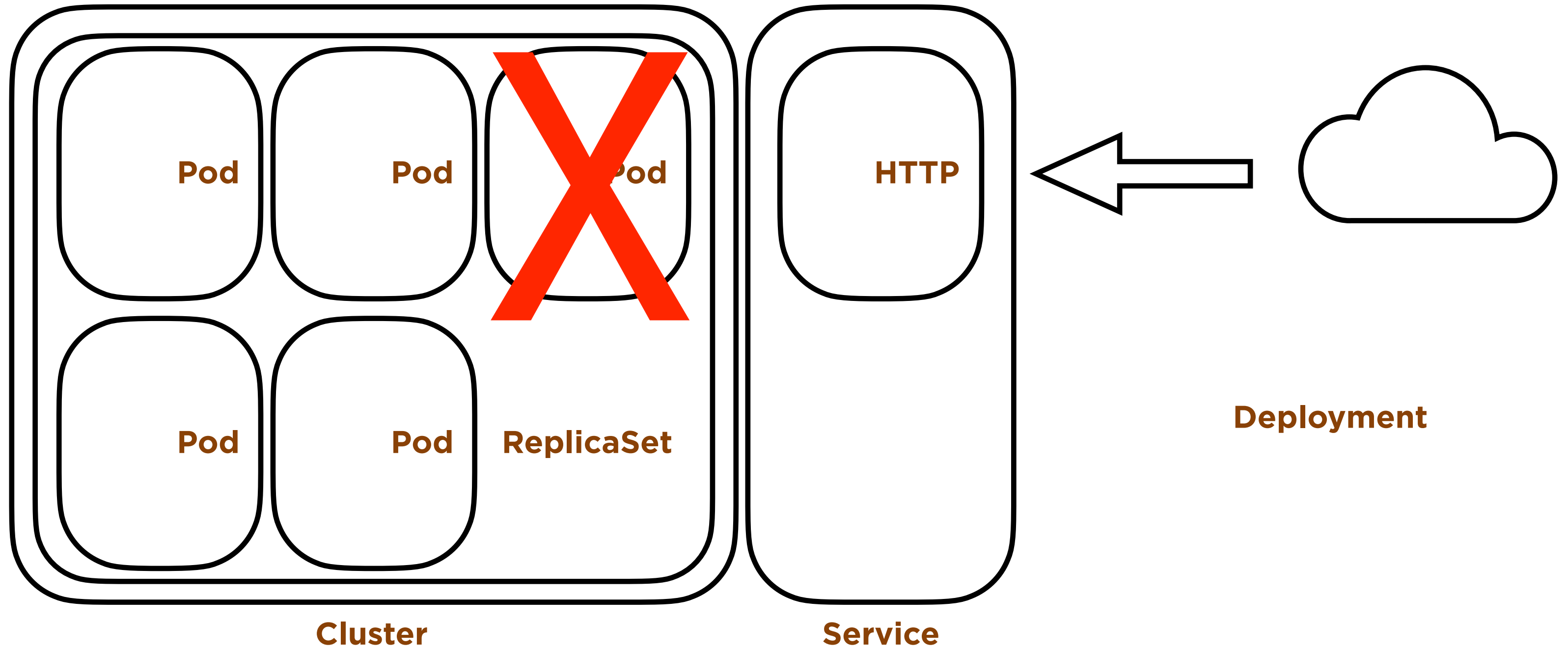


Cluster

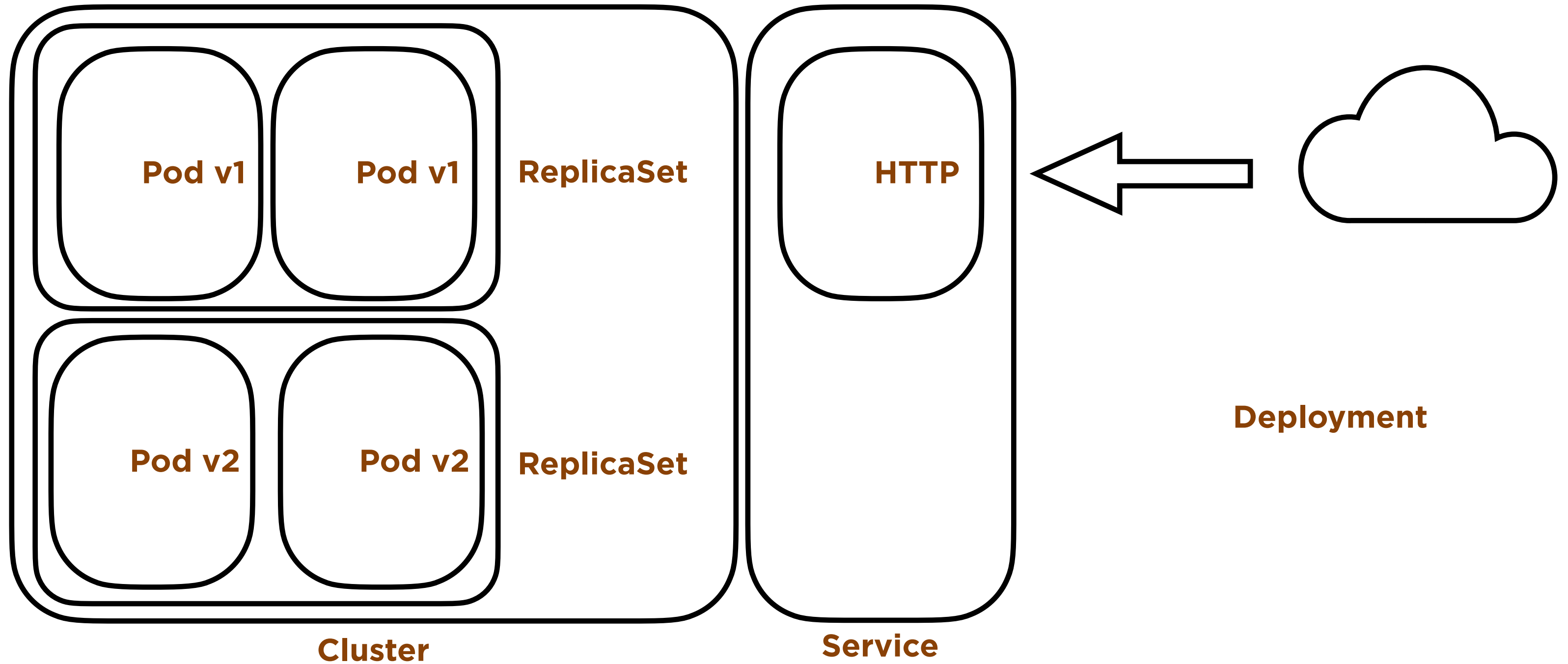
Kubernetes API

- **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

Services and ReplicaSets



Controller Operations - Deployment



Deploying Applications

- Imperative
- Declarative
- YAML and JSON

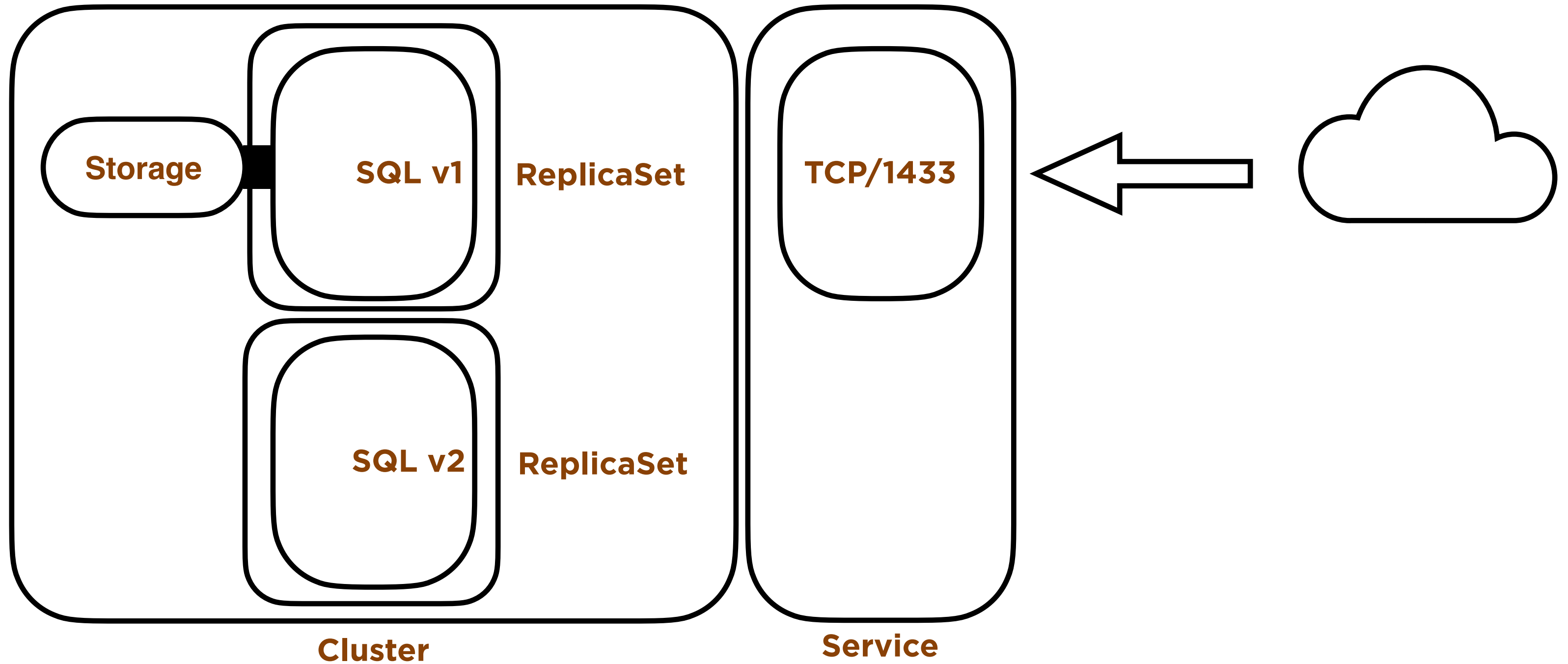
Demo!

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

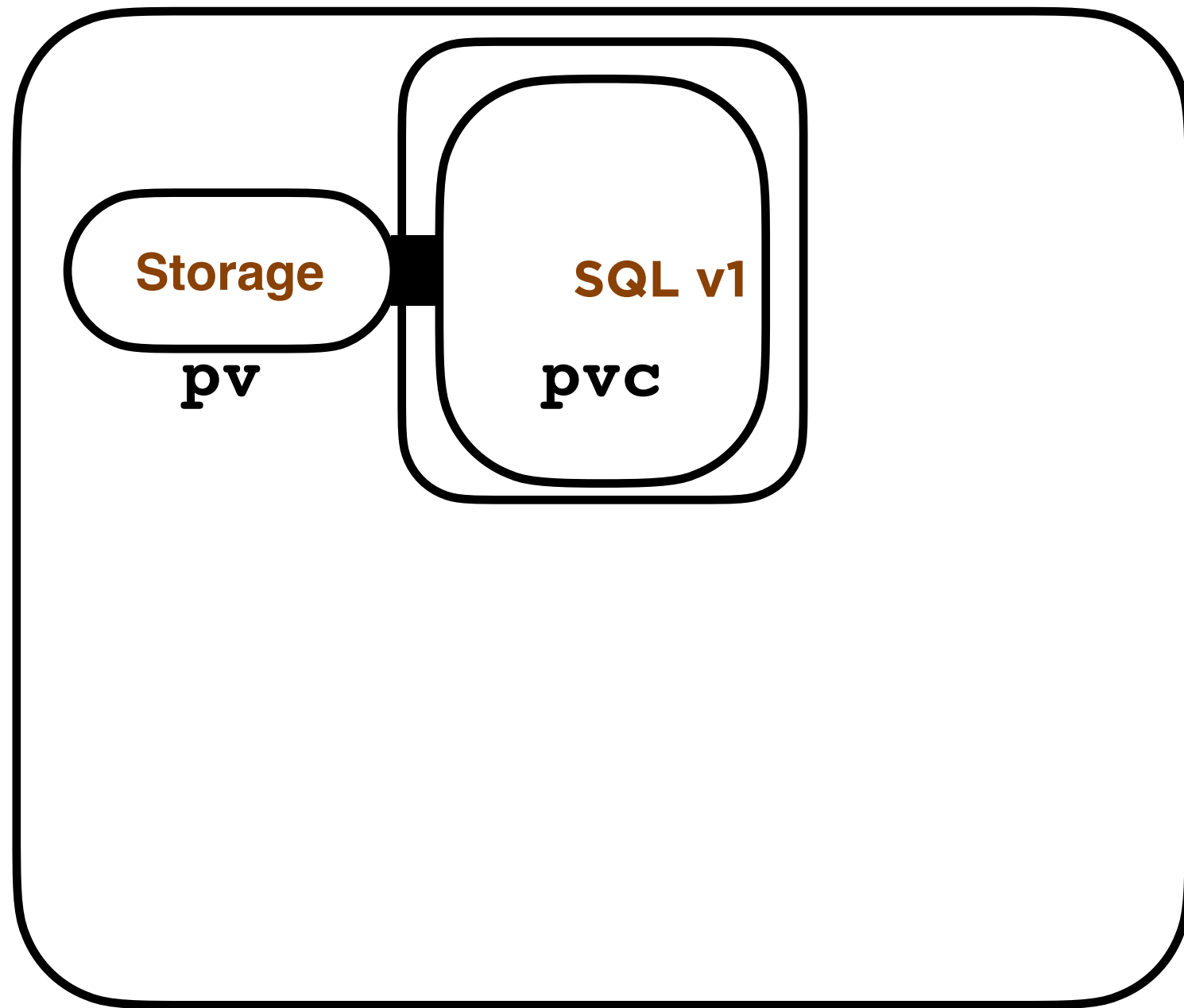
Running SQL Server in Kubernetes

- 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

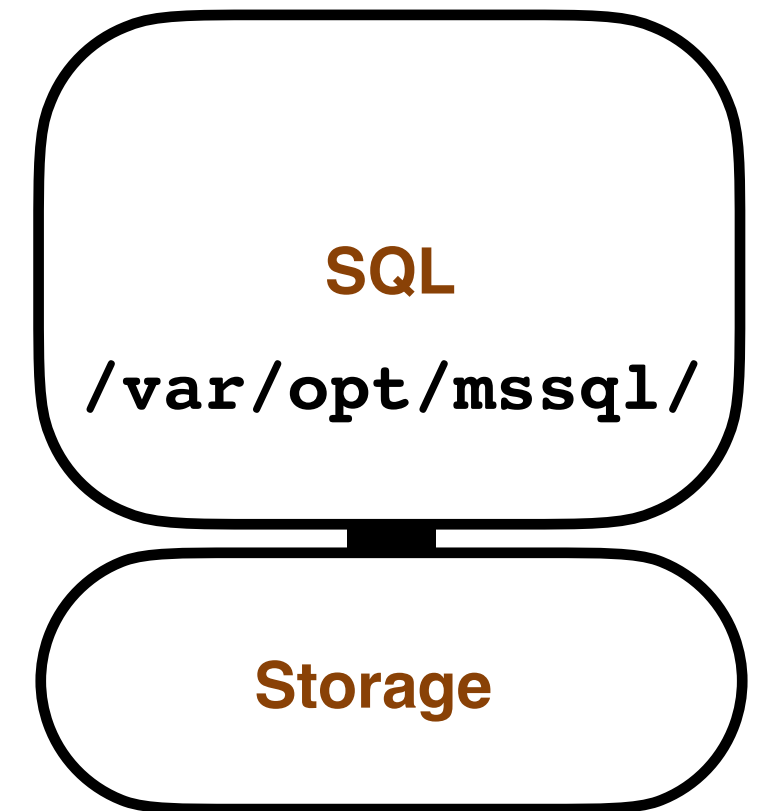


Cluster

- **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>

Define SQL Server in a Pod in YAML

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: mssql-deployment
```

```
spec:
```

```
  replicas: 1
```

```
  strategy:
```

```
    type: Recreate
```

```
  selector:
```

```
    matchLabels:
```

```
      app: mssql
```

```
spec:
```

```
  securityContext:
```

```
    fsGroup: 10001
```

```
  containers:
```

```
    - name: mssql
```

```
      image: '.../mssql/server:2019-CU5-ubuntu-18.04'
```

```
      ports:
```

```
        - containerPort: 1433
```

```
      env:
```

```
        - name: ACCEPT_EULA
```

```
          value: "Y"
```

```
        - name: SA_PASSWORD
```

```
          valueFrom:
```

```
            secretKeyRef:
```

```
              name: mssql
```

```
              key: SA_PASSWORD
```

```
  volumeMounts:
```

```
    - name: mssqldb
```

```
      mountPath: /var/opt/mssql
```

```
volumes:
```

```
  - name: mssqldb
```

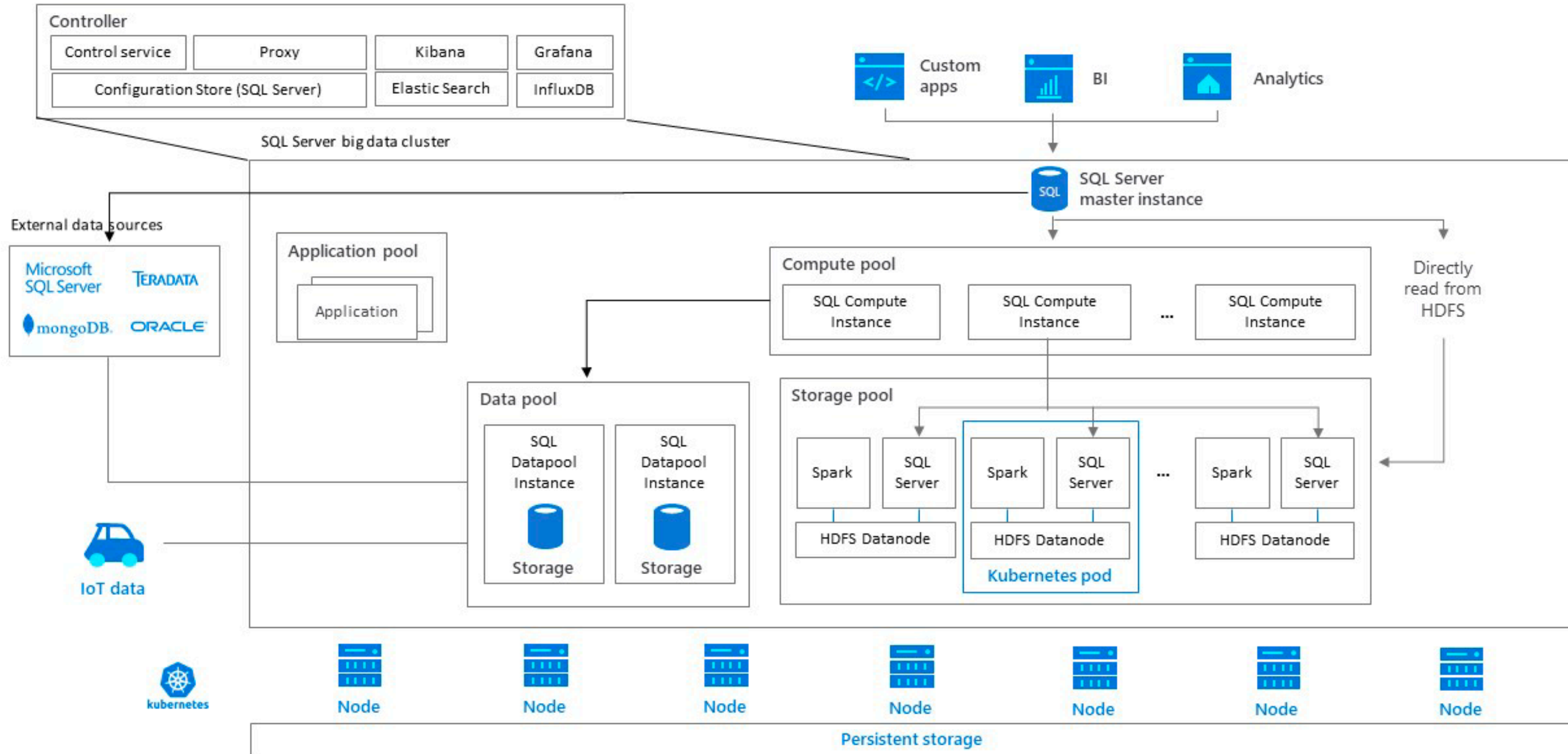
```
    persistentVolumeClaim:
```

```
      claimName: pvc-sql-data
```

Demo

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

Big Data Clusters



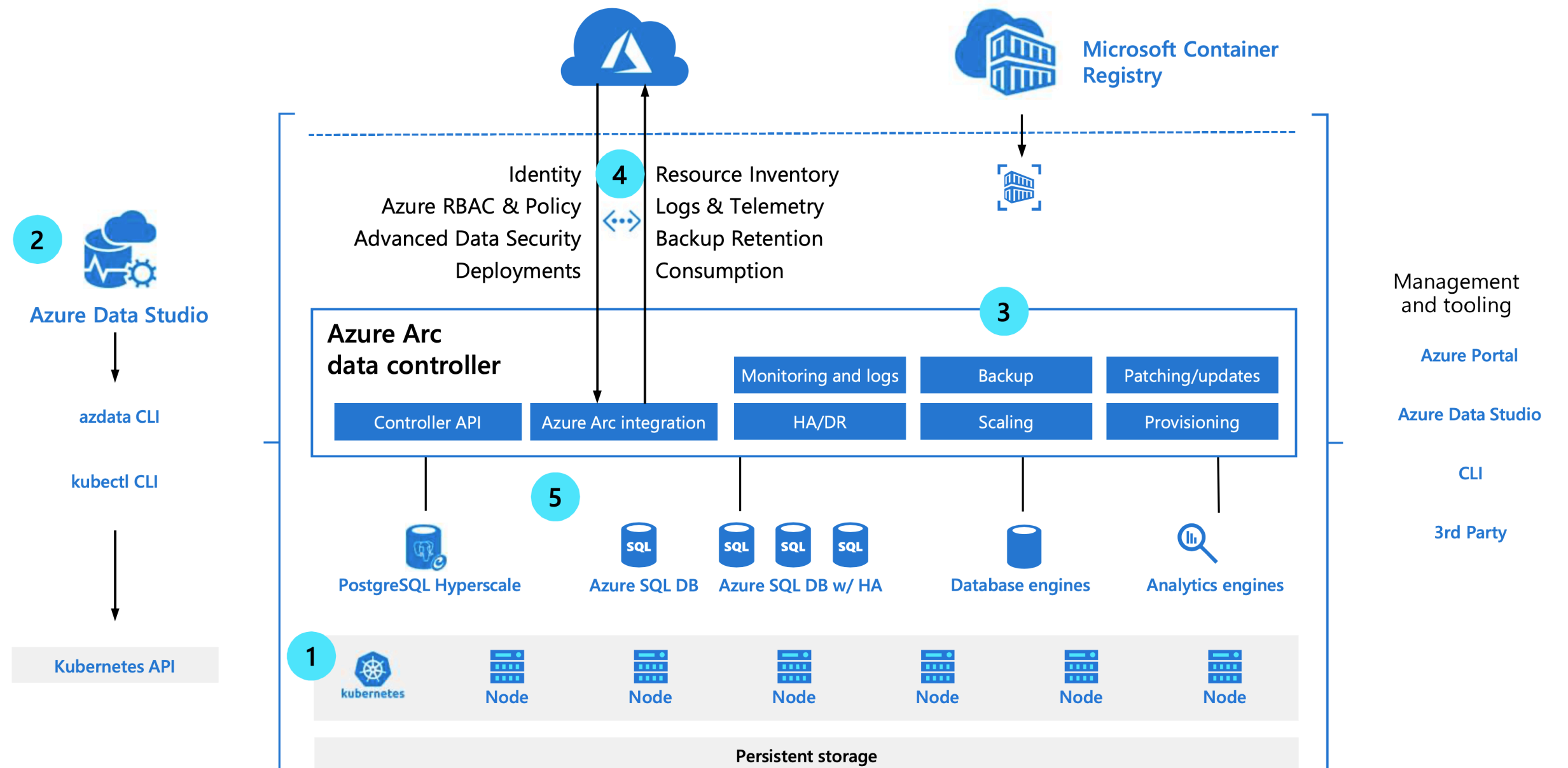
<https://docs.microsoft.com/en-us/sql/big-data-cluster/big-data-cluster-overview?view=sqlallproducts-allversions>

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/>
<http://www.github.com/nocentino/presentations>

Links to resources

Demos

Presentation

Pluralsight

aen@centinosystems.com

[@nocentino](#)

www.centinosystems.com

Solving tough business challenges with technical innovation



Thank You!