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


Centino
systems

Agenda

- **Deploying SQL Server in Kubernetes**

  - Data Persistency and Storage in Kubernetes

  - Pod Configuration and Running SQL Server in a Pod

  - Disk and Resource Configurations

  - Backups

  - The ~~Future~~ **Present** of SQL Server and Kubernetes
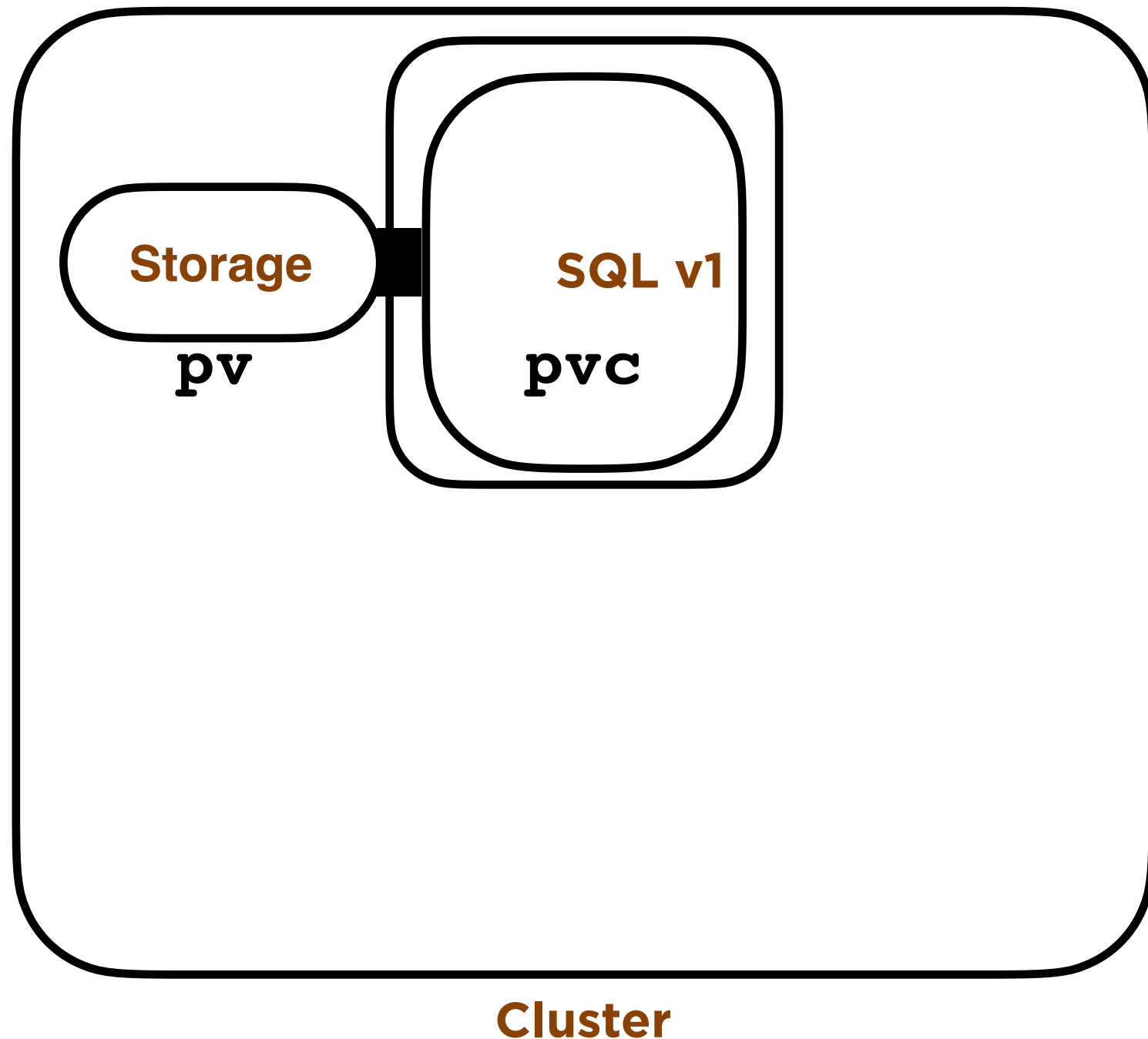
Running SQL Server in Kubernetes

- A Pod goes back 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

Storage **SQL v1**

pv **pvc**

Cluster
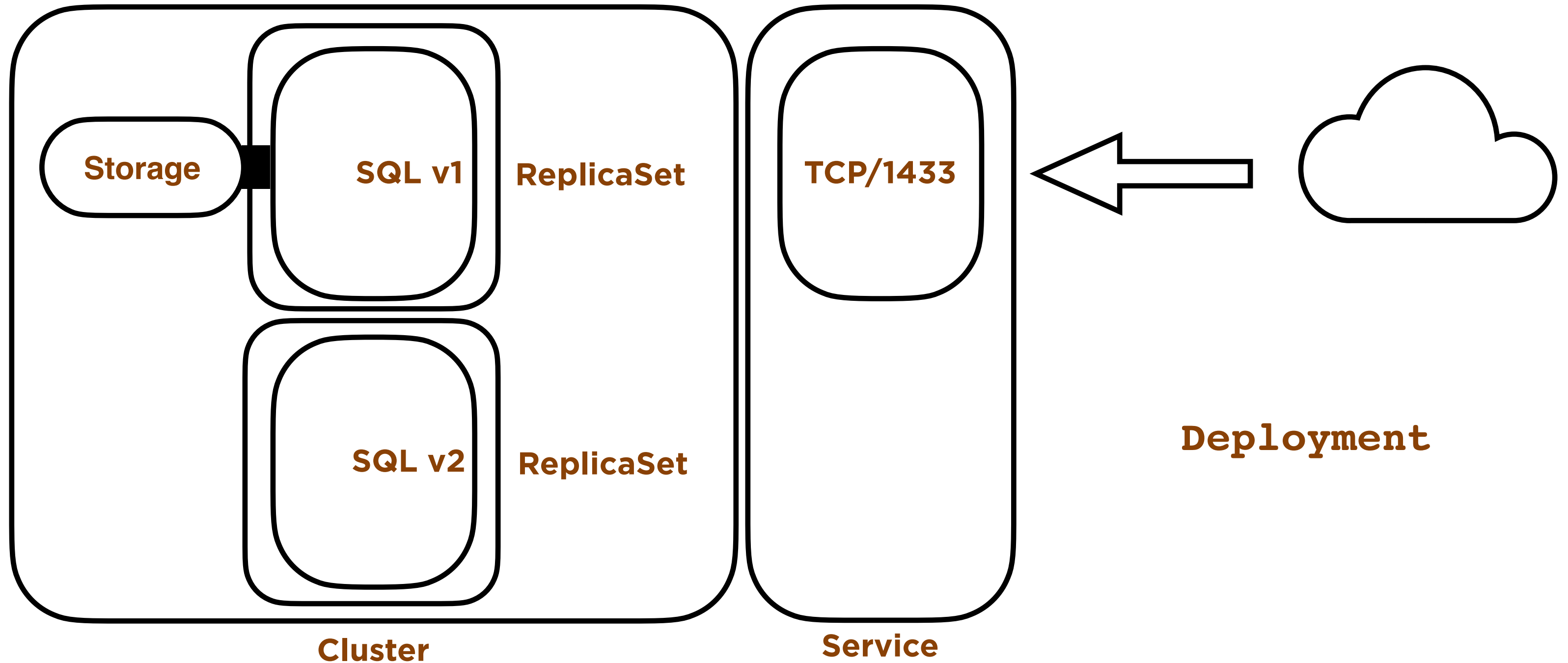
- **Persistent Volumes (pv)**
  - Administrator defined storage
  - iSCSI, NFS, FC, AzureDisk…many more
- **Persistent Volume Claims (pvc)**
  - The Pod "claims" the **pvc**
  - The **pvc** is mapped to the **pv** by k8s
  - Decouples the Pod and the storage

Centino
systems

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**

SQL

/var/opt/mssql/
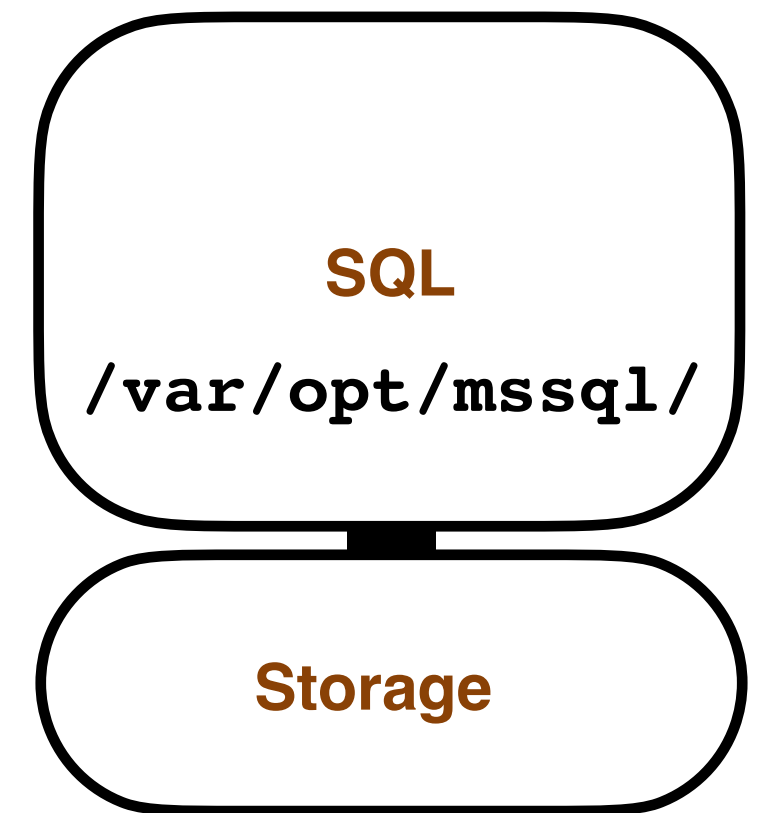
Storage

# Defining Persistent Volumes and Persistent Volume Claims

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nfs-data
  labels:
    disk: data
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: 172.16.94.5
    path: "/export/volumes/sql/data"
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nfs-data
spec:
  selector:
    matchLabels:
      disk: data
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Running SQL Server in a Pod

- In our Pod configuration define our storage configuration (**pvc**)

- Initial Pod deployment

  - If there's no system databases in the default data directory…

    - **/var/opt/mssql/data**

  - They're copied into the default data directory from the SFPs

- On subsequent Pod deployments the storage is attached into the 'new' Pod

  - Databases are already there

  - Master is read…contains our instance's configuration and state

  - Defined and accessible user databases are brought online

SQL

/var/opt/mssql/

Storage
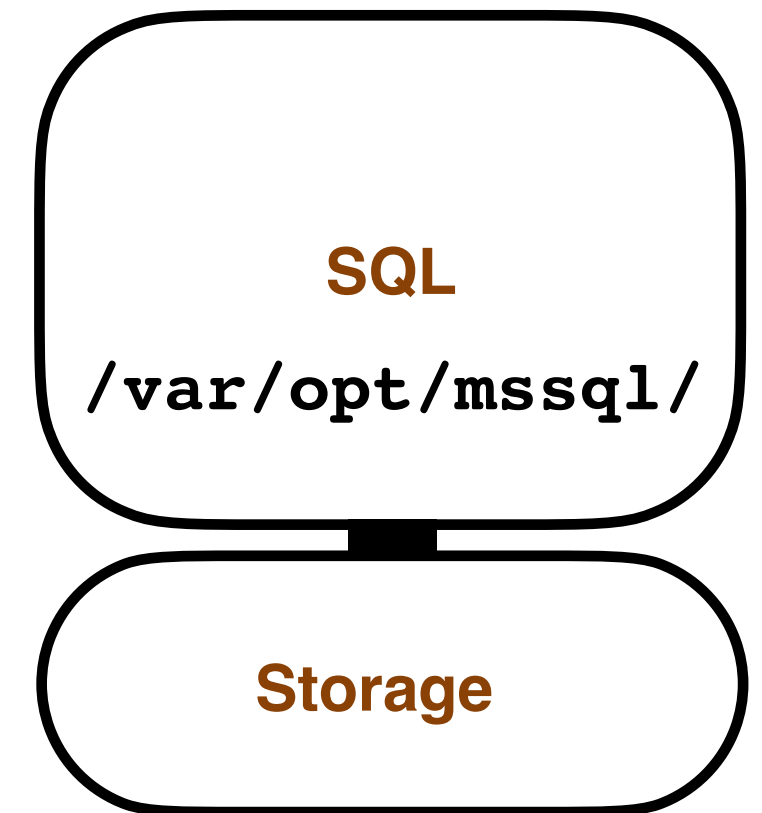
Configuring SQL Server in a Pod

- 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`**

  - Pods go back their initial state of the container image on creation

  - But some settings are persisted in master, right…yep!

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

Define SQL Server in a Pod in YAML

```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
```

Centino systems

Advanced Disk Topologies for SQL Server

- Define your **Persistent Volumes** and **Persistent Volume Claims**

- Use environment variables to specify default directories on Pod at startup

  - **MSSQL_DATA_DIR (/data)**

  - **MSSQL_LOG_DIR (/log)**

- New user databases will be created in these locations

- On Pod creation

  - All **PV/PVCs** will be mounted in the container at the defined locations

  - Master will online the databases

Resource Management

- Pod level resource management

  - CPU and Memory

    - **requests** - guaranteed

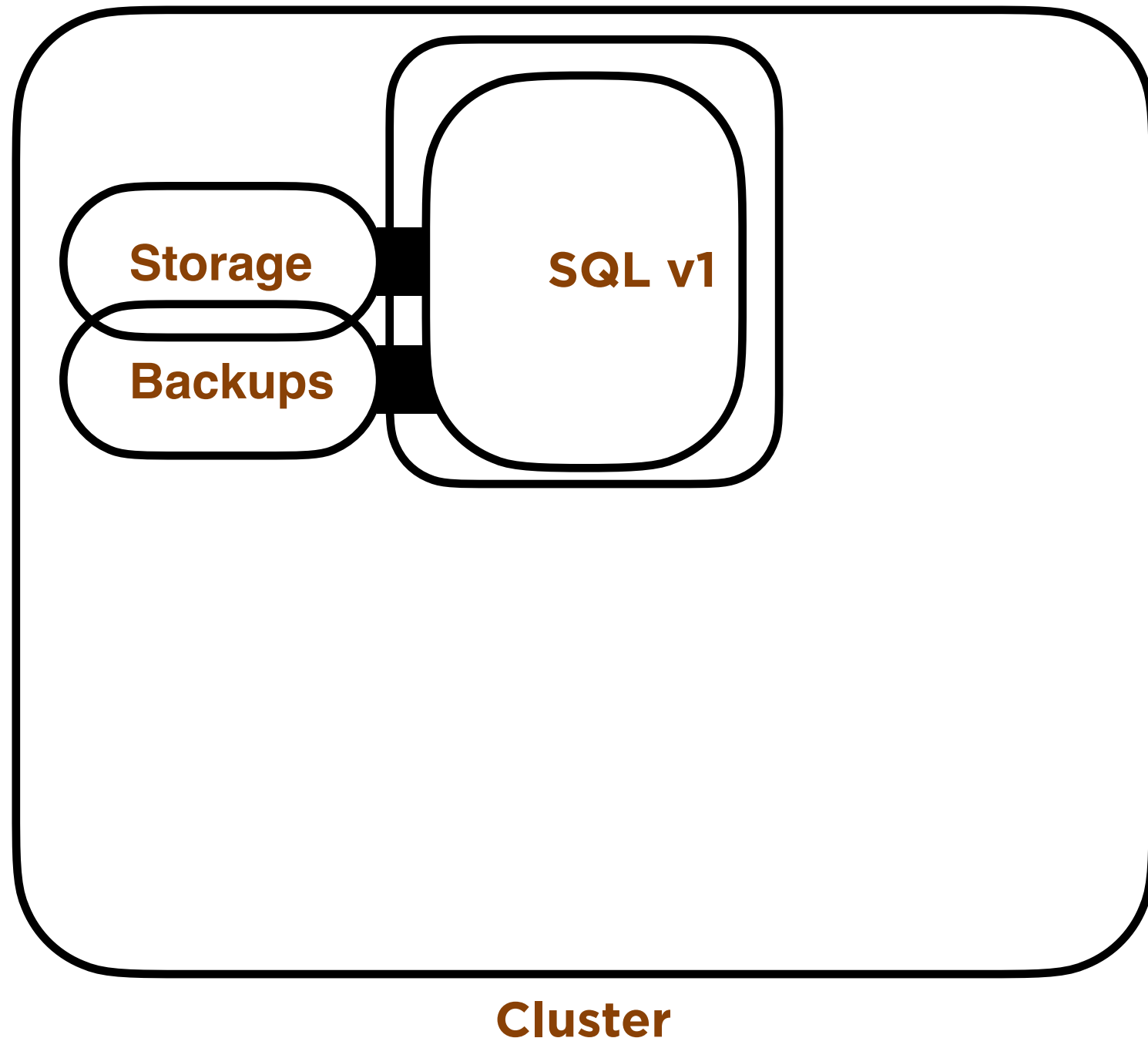   - **limits** - upper limit

  - No limits by default

- Server Instance settings still apply

- Kind of like multi-instance clusters

```
containers:
- name: mssql
  image: '.../mssql/server:2019-CU5-ubuntu-18.04'
  resources
    requests:
      cpu: 1
      memory: 1Gi
    limits:
      memory: 4Gi
```

Centino
systems

Backups!

- Persistent Volume (Shared or Dedicated)
  - AzureDisk
  - AzureFile
  - NFS/iSCSI/FC
- To URL
- Drive the backup jobs with normal techniques
  - Ola Hallengren's
  - Maintenance Plans
  - dbatools

Demo!

- Deploying SQL Server in a `Deployment` with Persistent Storage

  - Disk Topology

  - Setting Resource Limits

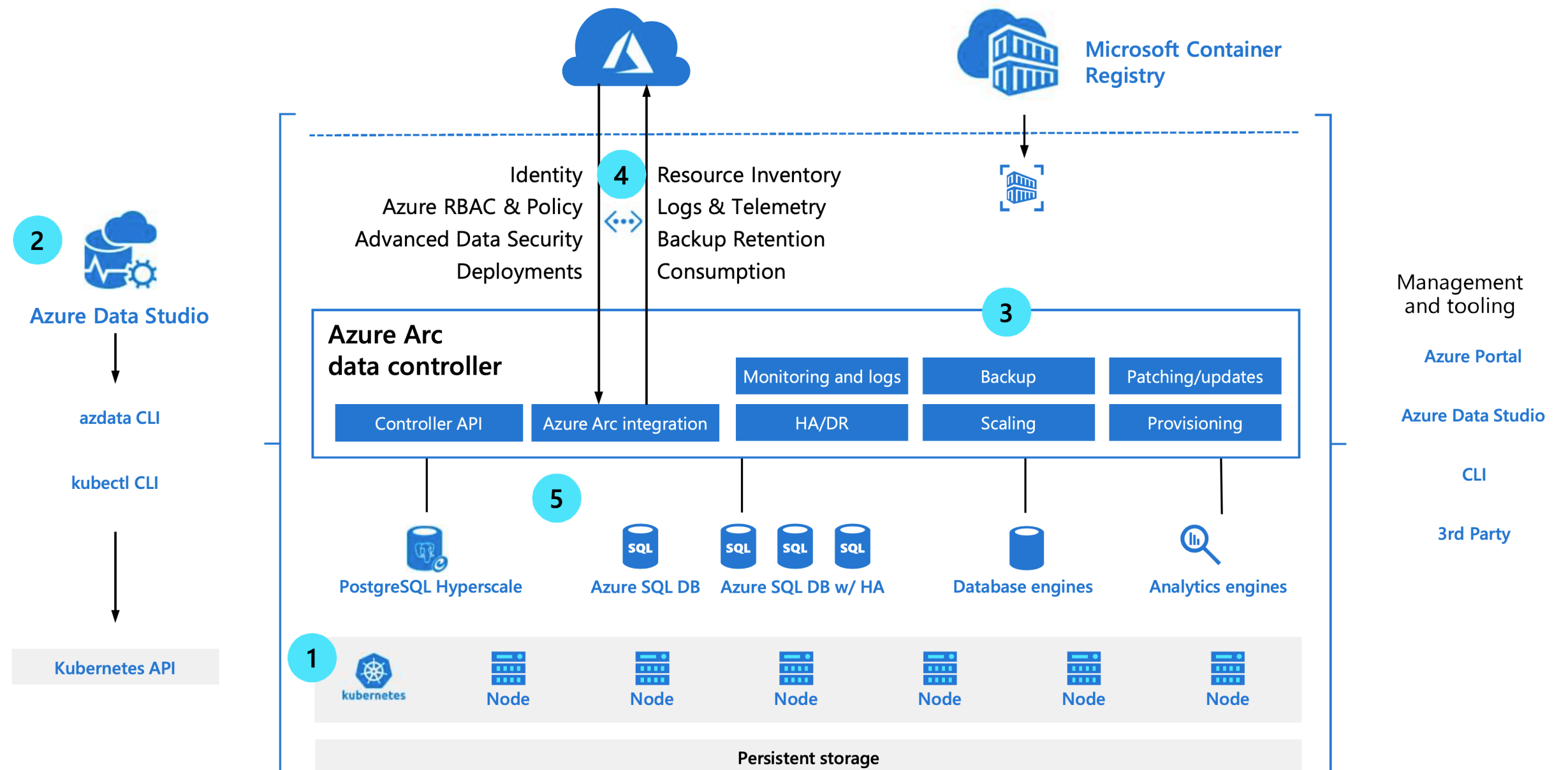 - High Availability

  - Backing up SQL Server in Kubernetes

Centino
systems

Big Data Clusters
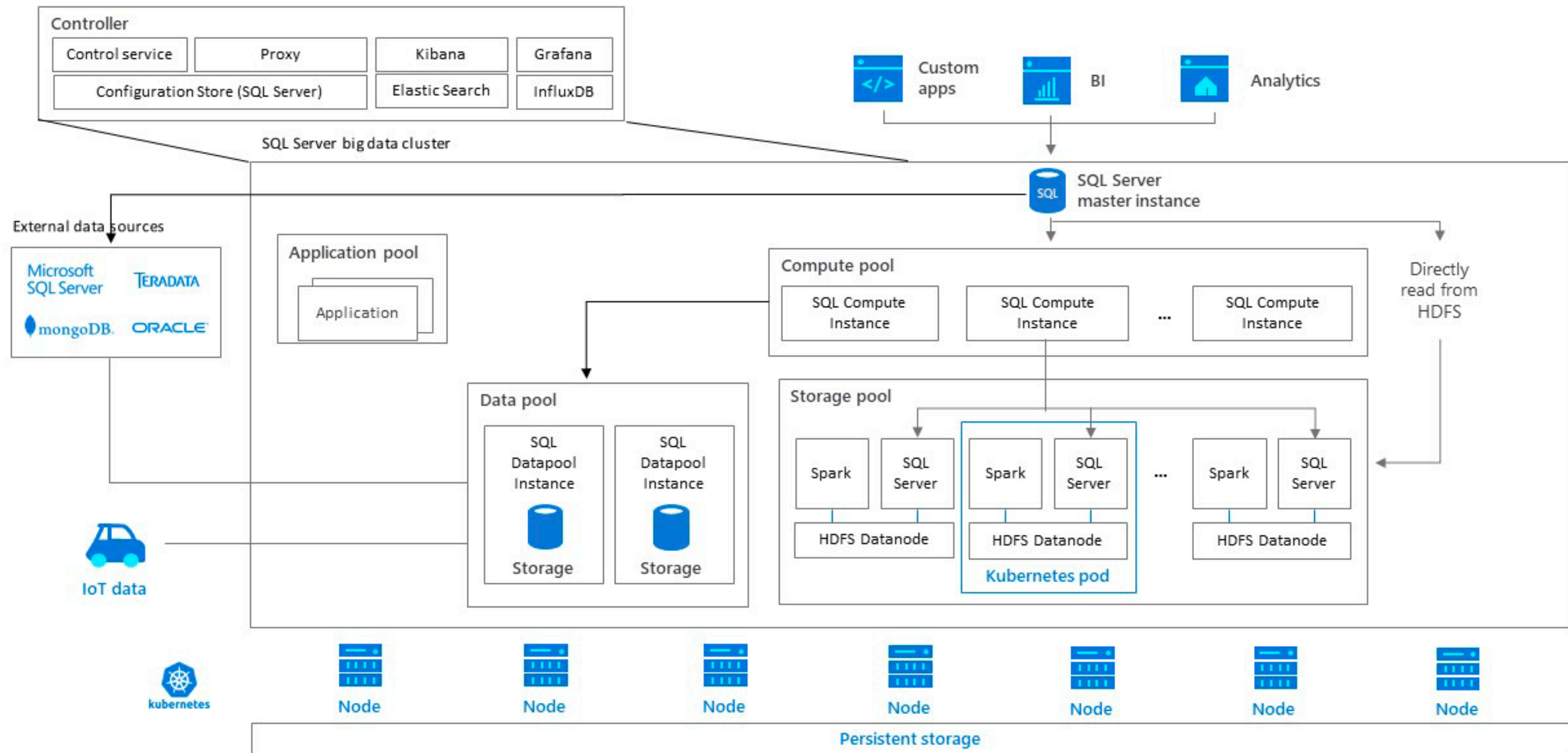
Azure Arc - Data Services

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

Azure Data Studio

↓

azdata CLI

↓

kubectl CLI

↓

Kubernetes API

Microsoft Container Registry

Identity **4** Resource Inventory
Azure RBAC & Policy · Logs & Telemetry
Advanced Data Security · Backup Retention
Deployments · Consumption

Azure Arc data controller

| Monitoring and logs | Backup | Patching/updates |
| Controller API | Azure Arc integration | HA/DR | Scaling | Provisioning |

3

5

PostgreSQL Hyperscale · Azure SQL DB · Azure SQL DB w/ HA · Database engines · Analytics engines

1 kubernetes · Node · Node · Node · Node · Node · Node

Persistent storage

Management and tooling

Azure Portal

Azure Data Studio

CLI

3rd Party

Review

- **Deploying SQL Server in Kubernetes**

  - Data Persistency and Storage in Kubernetes

  - Pod Configuration and Running SQL Server in a Pod

  - Disk and Resource Configurations

  - Backups

  - The ~~Future~~ **Present** of SQL Server and Kubernetes

Centino
systems

Thank you!

Pluralsight: www.pluralsight.com
email: aen@centinosystems.com
Twitter: @nocentino