Deploying SQL Server in Kubernetes

Anthony E. Nocentino
aen@centinosystems.com

Centino systems

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 - 2015-2019
- **email**: aen@centinosystems.com
- **Twitter**: @nocentino
- **Blog**: www.centinosystems.com/blog
- **Pluralsight Author**: www.pluralsight.com

Andrew Pruski

SQL Server DBA & Microsoft Data Platform MVP

@dbafromthecold

dbafromthecold@gmail.com

www.dbafromthecold.com

github.com/dbafromthecold

Agenda

- **Deploying SQL Server in Kubernetes**
  - Data Persistency and Storage in Kubernetes
  - Running SQL Server in a Pod
  - Disk and Resource Configurations
  - Backups
  - SQL Server Availability Groups in Kubernetes

Kubernetes 101

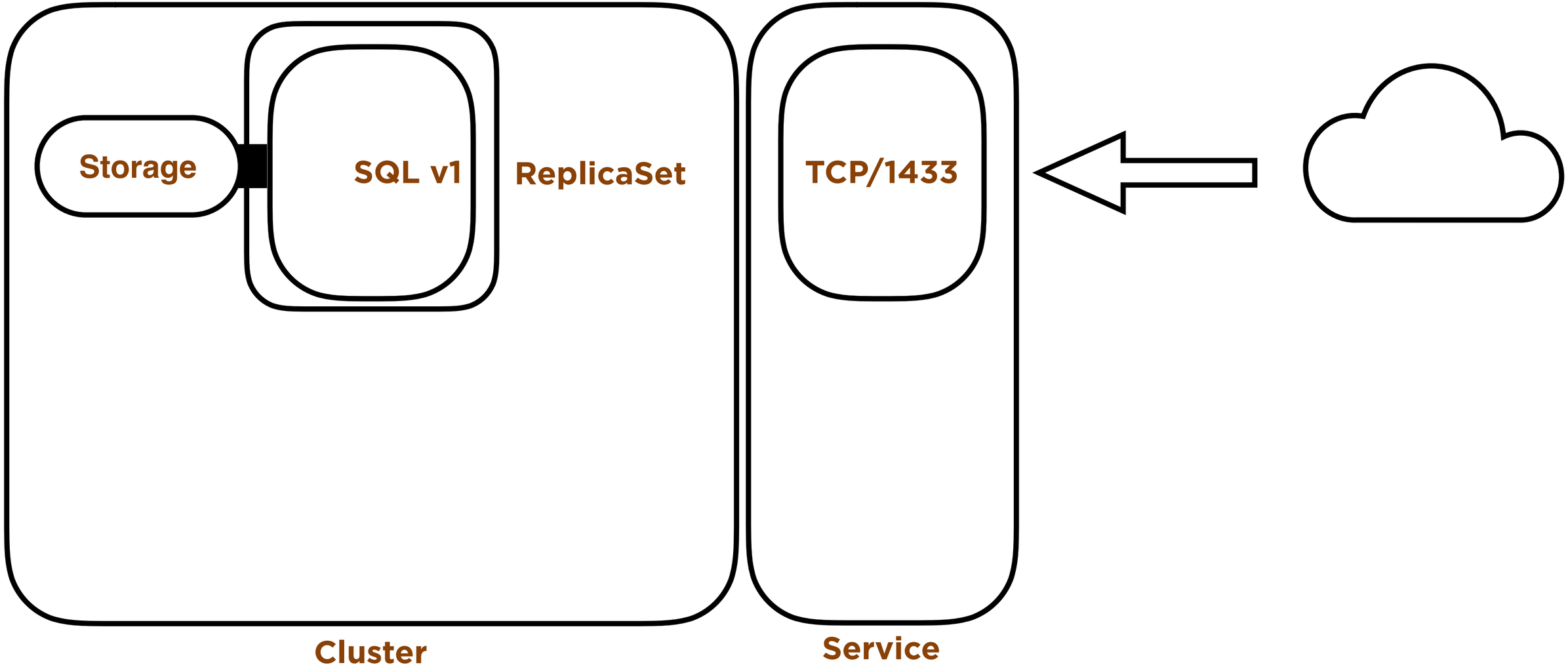
- Container Orchestrator
- Pods are Container Based Applications
- Infrastructure Abstraction
- Desired State
- Declarative Configuration in Code

Running SQL Server in Kubernetes

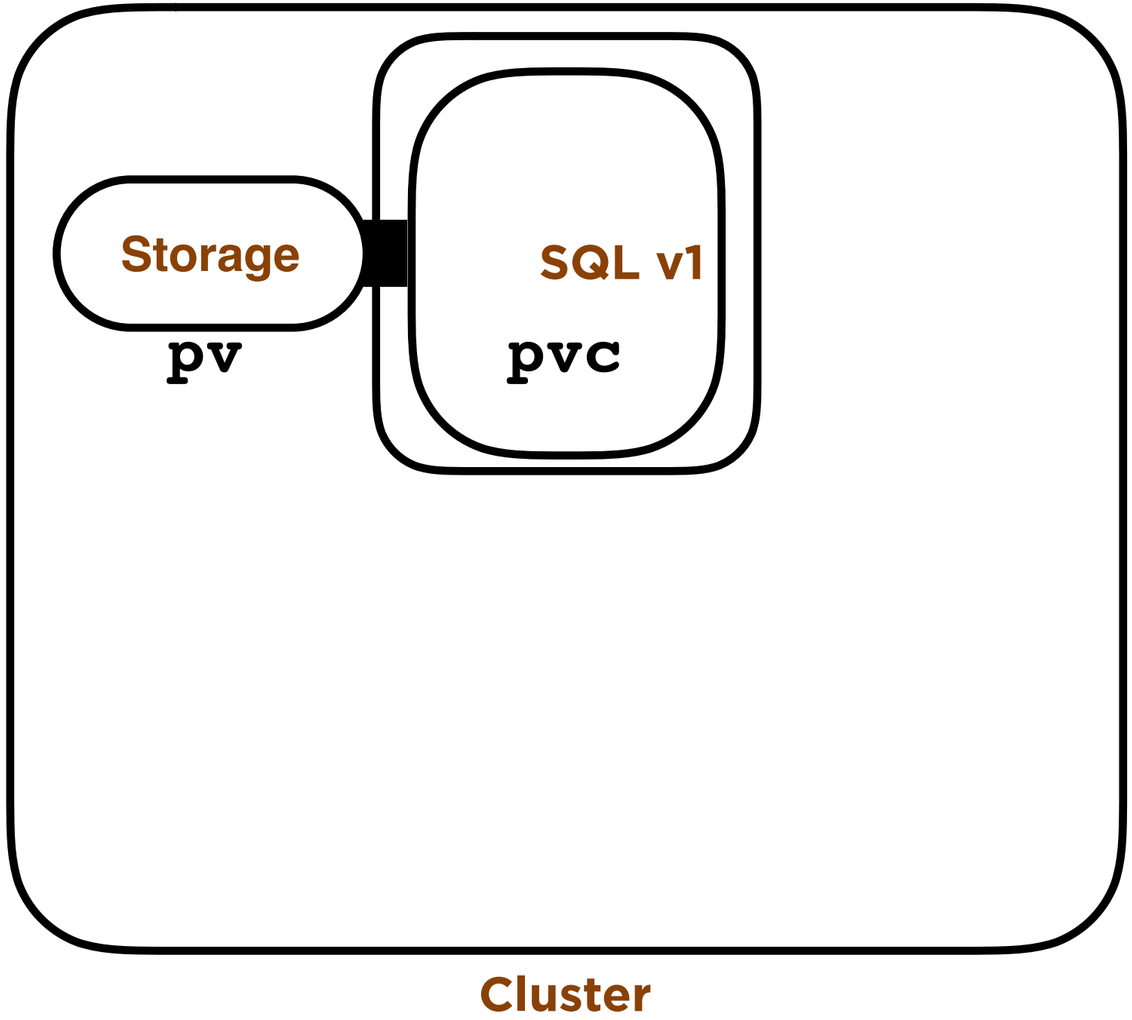
- A Pod goes back to the initial state each time it's deployed

- **State** - where do we store data?
- **Configuration** - how do we configure SQL Server?

Decoupling Data and Computation



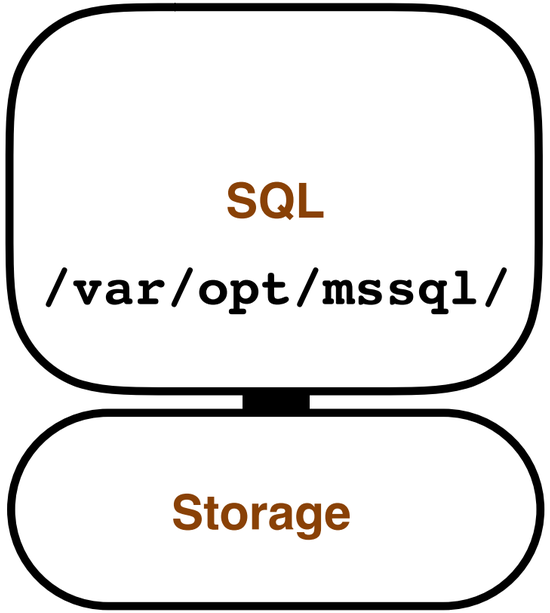
Storage in Kubernetes



- **`Persistent 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

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`**



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
```

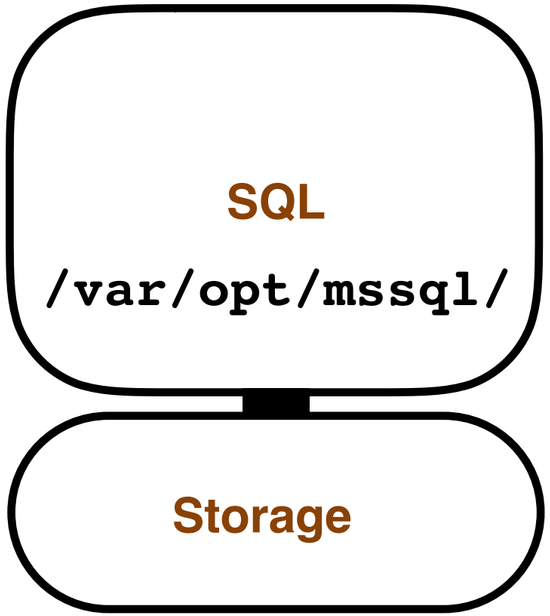
Configuring SQL Server in a Pod

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

https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-configure-environment-variables?view=sql-server-2017

Running SQL Server in a Pod (con't)

- 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



Definine 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
  template:
    metadata:
      labels:
        app: mssql
    spec:
      containers:
      - name: mssql
        image: 'mcr.microsoft.com/mssql/server:2017-CU15-ubuntu'
        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
```

Advanced Disk Topologies for SQL Server

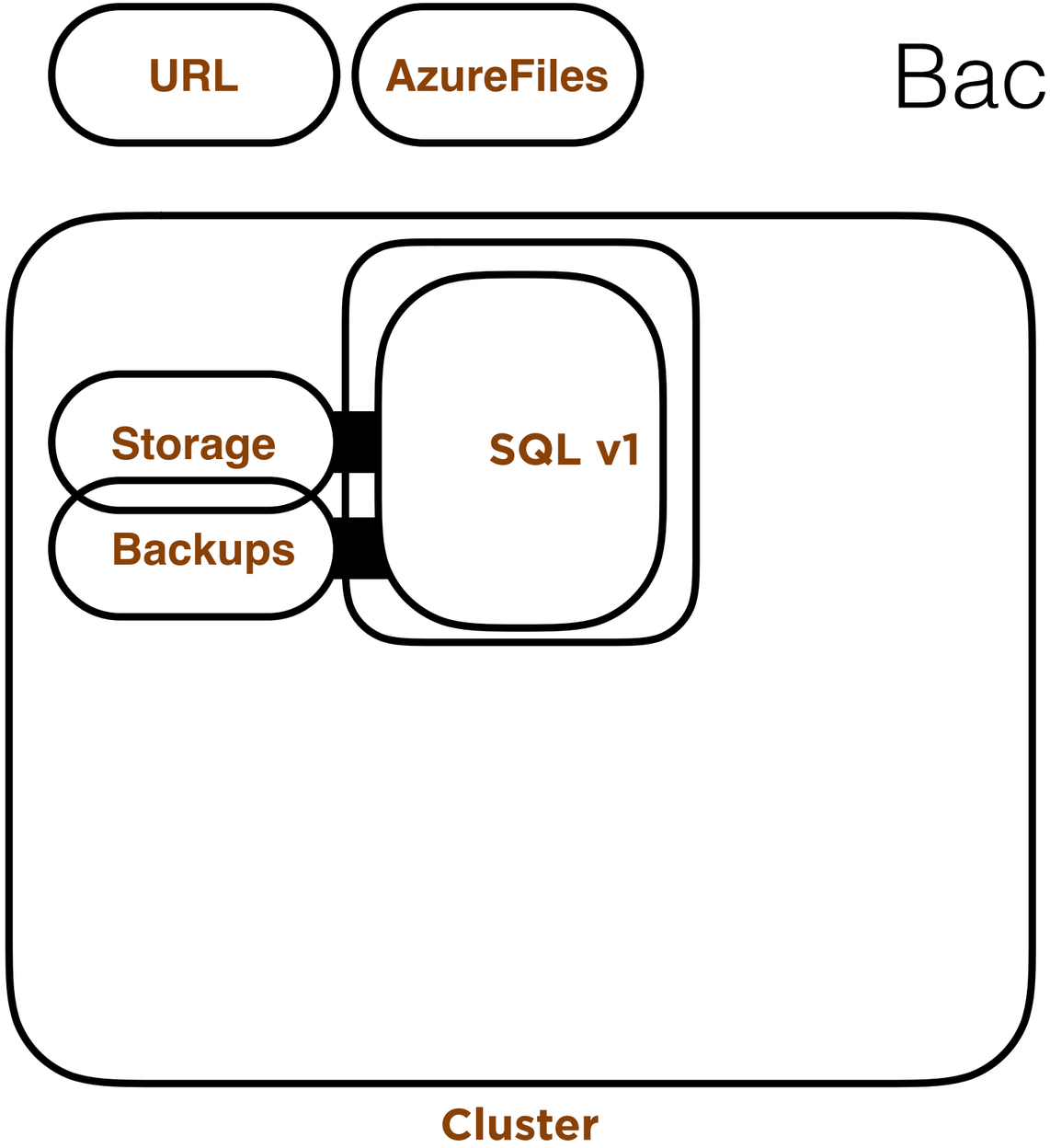
- Define your Persistent Volumes and Persistent Volume Claims
- Use environment variables to specify default directories on Pod at startup
  - **`MSSQL_DATA_DIR (/data)`**
  - **`MSSQL_LOG_DIR (/log)`**
- New user databases will be created in these locations
- On Pod creation
  - All PV/PVCs will be mounted in the container at the defined locations
  - Master will online the databases

Resource Management

- Resource management can happen at the Pod and Namespace levels
  - CPU and Memory
    - `requests` - guaranteed
    - `limits` - upper limit
  - No limits by default

```yaml
spec:
 containers:
  - name: mssql
    image: '…server:2017-CU15-ubuntu'
    resources:
      requests:
        cpu: 1
        memory: 4Gi
```

  - Server Instance settings still apply

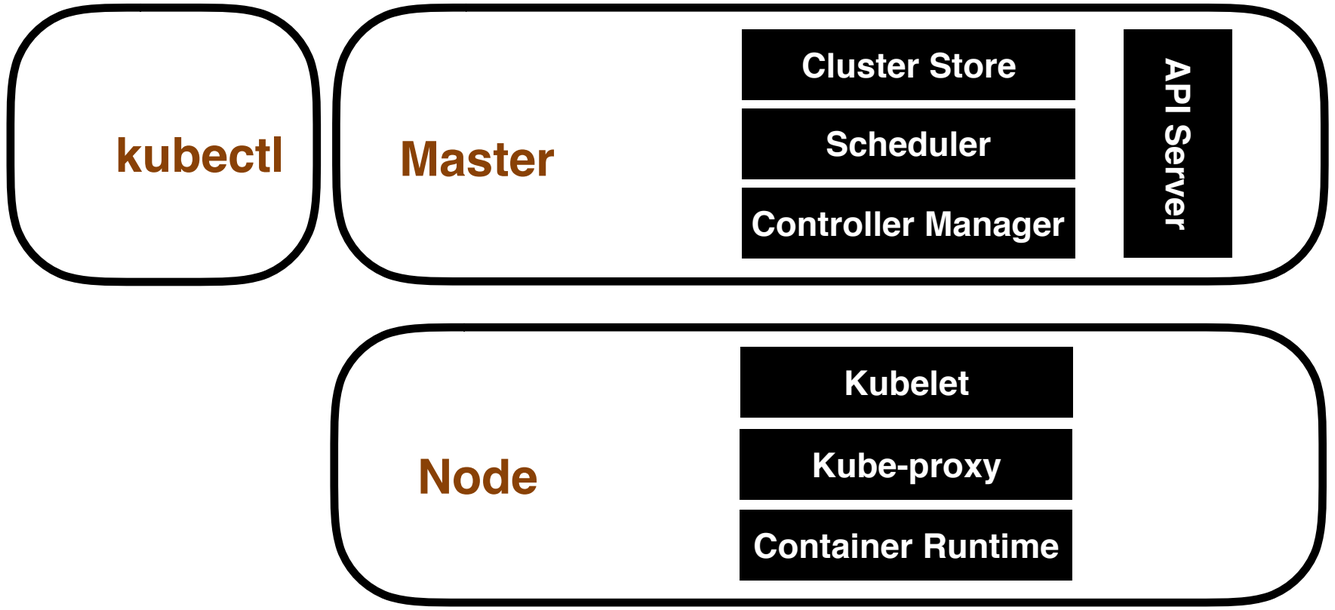


Backups!

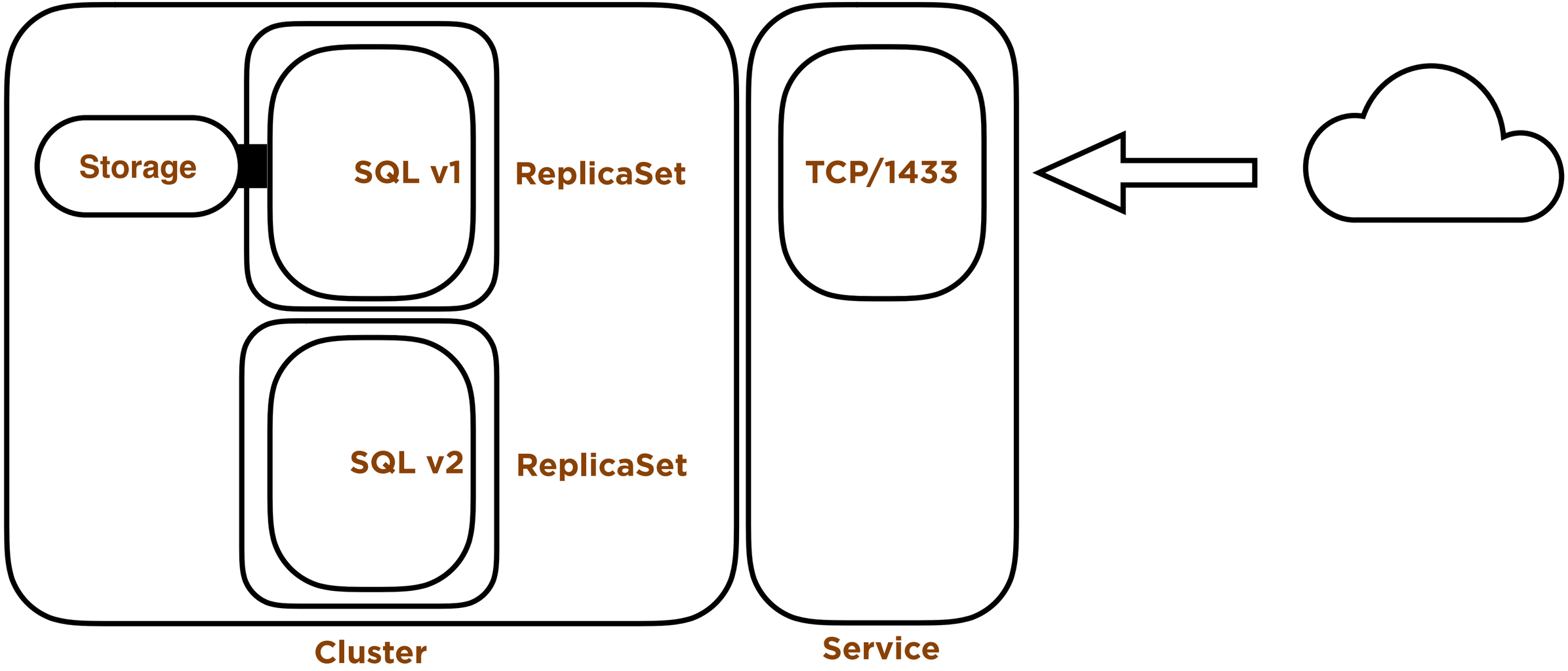
- Persistent Volume (Shared or Dedicated)
  - AzureDisk
  - AzureFile
  - NFS/iSCSI/FC
- To URL
- Drive the backup jobs with normal techniques
  - Ola Hallengren's
  - Maintenance Plans
  - dbatools

Demo!

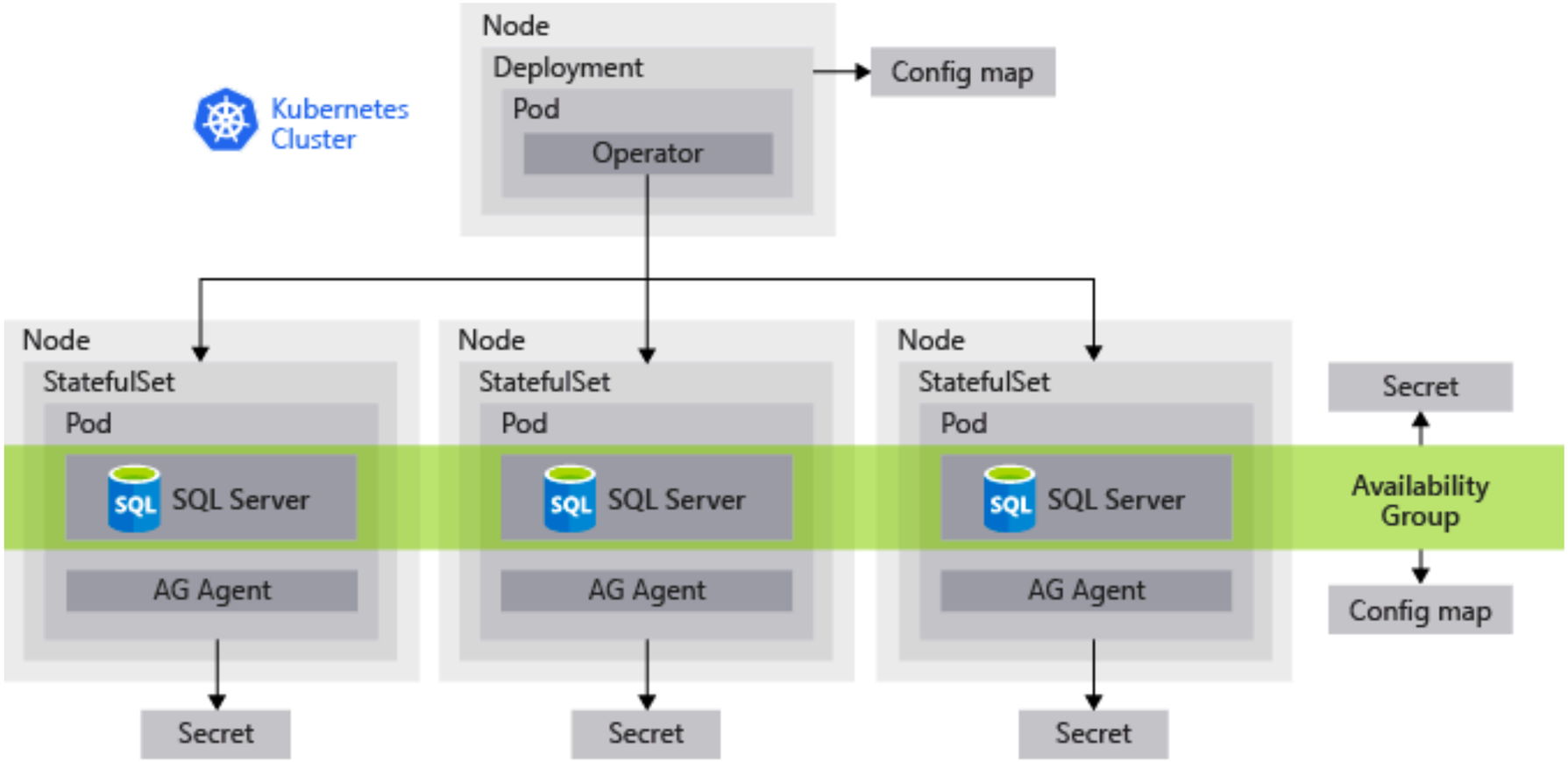
- Deploying SQL Server in a **`Deployment`** with Persistent Storage
  - Disk Topology
  - Setting Resource Limits
  - Backing up SQL Server in Kubernetes



High Availability in SQL Server in Kubernetes



Availability Groups in Kubernetes



From: https://cloudblogs.microsoft.com/sqlserver/2018/12/10/availability-groups-on-kubernetes-in-sql-server-2019-preview/

Review

- **Deploying SQL Server in Kubernetes**
  - Data Persistency and Storage in Kubernetes
  - Running SQL Server in a Pod
  - Disk and Resource configurations
  - Backups
  - SQL Server Availability Groups in Kubernetes

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**
  - **Kubernetes Installation and Configuration Fundamental and more!**
  - 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

Centino systems