# Lesson 4 : Manage Non-shared Storage with Stateful Sets

Deploy applications that scale without sharing storage.

© Copyright Jason Wong, Trainocate (M) 2020

# Application Clustering

- Clustering applications requires persistent storage
  - MySQL
  - Cassandra
- Multiple applications requires access to database system
- Need strong or eventual consistency read/write
- Common setup: Shared storage via NAS

© Copyright Jason Wong, Trainocate (M) 2020

# Network Attached Storage (NAS)

- File-based storage
- Good for data backup, archiving
- Has reliability, fault-tolerant, securirty, standard protocol, lock handler and caching
- No need for propietory hardware / software
- Common protocol: NFS and CIFS (SMB)
- Common case uses:
  - Web content
  - File share services
  - FTP storage
  - Backup archives
  - Video streaming, audio streaming
- Main disadvantage: not fit for structurable content

© Copyright Jason Wong, Trainocate (M) 2020

# Storage Area Network (SAN)

- Block-level storage
- Mostly required propietory hardware or conncetions
- Protocols:
  - FC, FCoE, iSCSI, NVMeoFC, NVMeoTCP
- Present raw device / LUN to hosts
- LUNs sharing; application handle locks and release
- Data protection: RAID configuration
- Common use:
  - SQL Databases (single node access), VM (multimode access)
  - High-performance data access, Server-side processing applications
  - Clustering applications and many more

© Copyright Jason Wong, Trainocate (M) 2020

# Stateless vs Statefull

**Stateless**

- Request sent to server and response relayed back without storing any information
- Or Data loss upon user exit/logouts
- Less complex setup
- Typically slower than Stateful
- Every transaction recreate, repeat, recalculate

**Statefull**

- Store data persistently
- Data can by used/tracked by servers, clients, other applications
- Simplifies data recovery (point in time)
- Requires design to complex setup
- Faster processing in subsequent transactions

© Copyright Jason Wong, Trainocate (M) 2020

# Stateful Sets

- Stateful application simplifies recovery from failures
- Consistent identities: Single stable DNS, hostname, storage
- Stateful set guarantees given network identity maps to same storage identity
  - Consistent permission to volumes, network resources
  - Facilitates graceful scaling operations and rolling updates
  - Pods added and removed in predictable order
- Best options for applications (databases) require consistent identities and non-shared persistent storage

© Copyright Jason Wong, Trainocate (M) 2020

# Differences

- Deployments provides stateless representation of set of pods
- StatefulSets provides stateful representation of set of pods
- ReplicaSets help manage traffic by scaling and load-balance multi-pods application

© Copyright Jason Wong, Trainocate (M) 2020

# Example application

Database requirements:

mysql-0 – First pod, primary role (read-write) → sit on higher end node

mysql-1 – Read-only replica → sits on lower end node

mysql-2 – Read-only replica → sits on lower end node

- Applications / user request connect o primary role for read-write access
- Event of failure, next replica is promoted to primary role, continue access by application
- However when first replica back online, failback can be automated
- If scaleDown occurs, only read-only replicas removed first

© Copyright Jason Wong, Trainocate (M) 2020

# Create StatefulSets using **YAML manifest** file

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: dbserver (1)
spec:
  selector:
    matchLabels:
      app: database (2)
  replicas: 3 (3)
  template:
    metadata:
      labels:
        app: database (4)
    spec:
      containers:
      - env: (5)
        - name: MYSQL_USER
          valueFrom:
            secretKeyRef:
              key: user
              name: sakila-cred
        image: registry.ocp4.example.com:8443/redhattraining/mysql-app:v1 (6)
        name: database (7)
        ports: (8)
        - containerPort: 3306
          name: database
        volumeMounts: (9)
        - mountPath: /var/lib/mysql
          name: data
      terminationGracePeriodSeconds: 10
```

1. Name of the stateful set.

2,4 Application labels.

3 Number of replicas.

5 Environment variables (can use secret object).

6 Image source.

7,8 Container name and ports.

9 Mount path information for the persistent volumes for each replica. Each persistent volume has the same configuration.

© Copyright Jason Wong, Trainocate (M) 2020

# Create StatefulSets using **YAML manifest** file

Create the stateful set by using the `create` command:

```
[user@host ~]$ oc create -f statefulset-dbserver.yml
```

Verify the creation of the stateful set named `dbserver`:

```
[user@host ~]$ kubectl get statefulset
NAME       READY   AGE
dbserver   3/3     6s
```

Verify the status of the instances:

```
[user@host ~]$ oc get pods
NAME         READY   STATUS    RESTARTS   AGE
dbserver-0   1/1     Running   0          85s
dbserver-1   1/1     Running   0          82s
dbserver-2   1/1     Running   0          79s
```

Verify the status of the persistent volumes:

```
[user@host ~]$ kubectl get pvc
NAME              STATUS   VOLUME             CAPACITY   ACCESS MODES
 STORAGECLASS ...
data-dbserver-0   Bound    pvc-c28f61ee-...   1Gi        RWO            nfs-
storage  ...
data-dbserver-1   Bound    pvc-ddbe6af1-...   1Gi        RWO            nfs-
storage  ...
data-dbserver-2   Bound    pvc-8302924a-...   1Gi        RWO            nfs-
storage  ...
```

Notice that three PVCs were created. Confirm that persistent volumes are attached to each instance:

```
[user@host ~]$ oc describe pod dbserver-0
...output omitted...
Volumes:
  data:
    Type:       PersistentVolumeClaim (a reference
 the same namespace)
    ClaimName:  data-dbserver-0
...output omitted...
```

```
[user@host ~]$ oc describe pod dbserver-1
...output omitted...
Volumes:
  data:
    Type:       PersistentVolumeClaim (a ref
 the same namespace)
    ClaimName:  data-dbserver-1
```

```
[user@host ~]$ oc describe pod dbserver-2
...output omitted...
Volumes:
  data:
    Type:       PersistentVolumeClaim (a re
 the same namespace)
    ClaimName:  data-dbserver-2
```

© Copyright Jason Wong, Trainocate (M) 2020

# StatefulSet lifecycles

You can update the number of replicas of the stateful set by using the `scale` command:

```
[user@host ~]$ oc scale statefulset/dbserver --replicas 1
NAME         READY   STATUS    RESTARTS  ...
dbserver-0   1/1     Running   0         ...
```

To delete the stateful set, use the `delete statefulset` command:

```
[user@host ~]$ kubectl delete statefulset dbserver
statefulset.apps "dbserver" deleted
```

Notice that the PVCs are not deleted after the execution of the `oc delete statefulset` command:

```
[user@host ~]$ oc get pvc
NAME              STATUS   VOLUME             CAPACITY   ACCESS MODES
 STORAGECLASS ...
data-dbserver-0   Bound    pvc-c28f61ee-...   1Gi        RWO           nfs-
storage  ...
data-dbserver-1   Bound    pvc-ddbe6af1-...   1Gi        RWO           nfs-
storage  ...
data-dbserver-2   Bound    pvc-8302924a-...   1Gi        RWO           nfs-
storage  ...
```

You can create a stateful set from the web console by clicking the **Workloads** > **StatefulSets** menu. Click **Create StatefulSet** and customize the YAML manifest.

© Copyright Jason Wong, Trainocate (M) 2020

# Guided Exercise: Manage Non-shared Storage with Stateful Sets

## You should be able to:

- Deploy a web server with persistent storage.
- Add data to the persistent storage.
- Scale the web server deployment and observe the data that is shared with the replicas.
- Create a database server with a stateful set by using a YAML manifest file.
- Verify that each instance from the stateful set has a persistent volume claim.

© Copyright Jason Wong, Trainocate (M) 2020

# Lab: Manage Storage for Application Configuration and Data

## You should be able to:

- Deploy a database server.
- Deploy a web application.
- Create a secret that contains the database server credentials.
- Create a configuration map that contains an SQL file.
- Add and remove a volume on the database server and the web application.
- Expose the database server and the web application.
- Scale up the web application.
- Mount the configuration map as a volume.

© Copyright Jason Wong, Trainocate (M) 2020