I get a lot of questions about Kubernetes (https://akomljen.com/tag/kubernetes/) and persistence. Of course, persistence is important for stateful apps. We often say that for stateful apps you need to use `StatefulSet` and for stateless apps a `Deployment`. This doesn't mean that you couldn't run stateful apps using deployments with persistent volumes. For example, the official MySQL Helm chart (https://github.com/helm/charts/tree/master/stable/mysql) is using deployment. So, it can be done, but users get confused about this. What is the deal? When should you use deployment and when stateful set?

Previous blog post

> *Just blogged: Monthly Update 1 https://t.co/2oBOGo032C (https://t.co/2oBOGo032C)*
>
> *— Alen Komljen (@alenkomljen) April 2, 2018 (https://twitter.com/alenkomljen/status/980786683196137473?ref_src=twsrc%5Etfw)*

Persistent Volume Claim

To have persistence in Kuberntes you need to create a Persistent Volume Claim or PVC (https://kubernetes.io/docs/concepts/storage/persistent-volumes/#lifecycle-of-a-volume-and-claim) which is later consumed by a pod. Also, you can get confused here because there is also a Persistent Volume or PV (https://kubernetes.io/docs/concepts/storage/persistent-volumes/). If you have a default Storage Class (https://kubernetes.io/docs/concepts/storage/storage-classes/) or you specify which storage class to use when creating a PVC, PV will be automatically created. PV holds information about physical storage. PVC is just a requests for PV. Another way and less desirable is to create a PV manually and attach PVC to it, skipping storage class altogether.

You can define a PVC and set the desired size, access modes, storage class name, etc. Let's create a `zookeeper-vol` PVC:

```
⚡ cat <<EOF | kubectl create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: zookeeper-vol
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: rbd
EOF

⚡ kubectl get pvc zookeeper-vol
NAME            STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
zookeeper-vol   Bound     pvc-693857a8-3a8b-11e8-a34e-0238efc27e9c   8Gi        RWO            rbd            10s
```

In my example, I have a storage class name `rbd` which is a Ceph cluster. When the new PVC gets created I will have the new 8GB volume which is ready to use. The important thing here are the access modes:

- **ReadWriteOnce** – Mount a volume as read-write by a single node
- **ReadOnlyMany** – Mount the volume as read-only by many nodes
- **ReadWriteMany** – Mount the volume as read-write by many nodes

Access mode will define how a pod will consume this volume. In most cases, you will set `ReadWriteOnce` so that only one node can do read-write. Please note that this means more pods on a single node can still use the same volume. In some cases for stateless apps you want to have read-only volumes and for that, you need to use `ReadOnlyMany`.

The rare case is `ReadWriteMany` because only a few storage providers have the support for it. Think of `ReadWriteMany` as NFS.

Define a Deployment with PVC

It is possible to create a PVC with `ReadWriteOnce` access mode, and then to create a deployment which will run a stateful application and use this PVC. It works perfectly fine, but only if you don't want to scale your deployment. If you try to do it you will probably get an error that volume is already in use when pod starts on another node. Even if that is not the case, and both pods end up on the same node, still they will write to the same volume. And you don't want this.

I created a Kubernetes ready Zookeeper Docker image (https://github.com/komljen/zookeeper-k8s-image) for this blog post. Let's use the `zookeeper-vol` PVC that I created before and create the new Zookeeper deployment which will mount this volume:

```
⚡ cat <<EOF | kubectl create -f -
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: zookeeper
spec:
  selector:
    matchLabels:
      app: zookeeper
  replicas: 1
  template:
    metadata:
      labels:
        app: zookeeper
    spec:
      containers:
      - env:
        - name: ZOOKEEPER_SERVERS
          value: "1"
        image: "komljen/zookeeper:3.4.10"
        imagePullPolicy: IfNotPresent
        name: zookeeper
        ports:
        - containerPort: 2181
          name: client
        - containerPort: 2888
          name: server
        - containerPort: 3888
          name: leader-election
        readinessProbe:
          exec:
            command:
            - /opt/zookeeper/bin/zkOK.sh
          initialDelaySeconds: 10
          timeoutSeconds: 2
          periodSeconds: 5
        livenessProbe:
          exec:
            command:
            - /opt/zookeeper/bin/zkOK.sh
          initialDelaySeconds: 120
          timeoutSeconds: 2
          periodSeconds: 5
        volumeMounts:
        - mountPath: /data
          name: zookeeper-data
      restartPolicy: Always
      volumes:
      - name: zookeeper-data
        persistentVolumeClaim:
          claimName: zookeeper-vol
---
apiVersion: v1
kind: Service
metadata:
  name: zookeeper
spec:
  ports:
  - name: client
    port: 2181
    targetPort: 2181
  selector:
    app: zookeeper
---
apiVersion: v1
kind: Service
metadata:
  name: zookeeper-server
spec:
  clusterIP: None
  ports:
  - name: server
    port: 2888
    targetPort: 2888
  - name: leader-election
    port: 3888
    targetPort: 3888
  selector:
    app: zookeeper
EOF
```

If you try to scale this deployment, other replicas will try to mount and use the same volume. This is fine if your volume is read-only. So, how to work around it for read-write volumes?

If you are new to Kubernetes read this book first Kubernetes in Action (https://amzn.to/2CJO1mb)

Define a Stateful Set with PVC

When you have an app which requires persistence you should create a stateful set instead of deployment. There are a lot of benefits. Also, you will not have to create a PVCs in advance, and you will be able to scale it easily. Of course, the scaling depends on the app you are deploying. With the stateful set, you can define a `volumeClaimTemplates` so that a new PVC is created for each replica automatically. And also you will end up with only one file which defines your app and also persistent volumes. Now let's try to deploy Zookeeper using stateful set:

```
⚡ cat <<EOF | kubectl create -f -
apiVersion: apps/v1beta2
kind: StatefulSet
metadata:
  name: zookeeper
spec:
  selector:
    matchLabels:
      app: zookeeper
  replicas: 1
  serviceName: zookeeper-server
  template:
    metadata:
      labels:
        app: zookeeper
    spec:
      containers:
      - env:
        - name: ZOOKEEPER_SERVERS
          value: "1"
        image: "komljen/zookeeper:3.4.10"
        imagePullPolicy: IfNotPresent
        name: zookeeper
        ports:
        - containerPort: 2181
          name: client
        - containerPort: 2888
          name: server
        - containerPort: 3888
          name: leader-election
        readinessProbe:
          exec:
            command:
            - /opt/zookeeper/bin/zkOK.sh
          initialDelaySeconds: 10
          timeoutSeconds: 2
          periodSeconds: 5
        livenessProbe:
          exec:
            command:
            - /opt/zookeeper/bin/zkOK.sh
          initialDelaySeconds: 120
          timeoutSeconds: 2
          periodSeconds: 5
        volumeMounts:
        - mountPath: /data
          name: zookeeper-vol
      restartPolicy: Always
  volumeClaimTemplates:
  - metadata:
      name: zookeeper-vol
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 8Gi
      storageClassName: rbd
---
apiVersion: v1
kind: Service
metadata:
  name: zookeeper
spec:
  ports:
  - name: client
    port: 2181
    targetPort: 2181
  selector:
    app: zookeeper
---
apiVersion: v1
kind: Service
metadata:
  name: zookeeper-server
spec:
  clusterIP: None
  ports:
  - name: server
    port: 2888
    targetPort: 2888
  - name: leader-election
```

```
    port: 3888
    targetPort: 3888
  selector:
    app: zookeeper
EOF
```

The major difference compared to deployment is in this part:

```
spec:
  volumeClaimTemplates:
  - metadata:
      name: zookeeper-vol
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 8Gi
      storageClassName: rbd
```

After you create this stateful set the new PVC is also created for a pod `zookeeper-0`:

```
⚡ kubectl get pvc | grep zookeeper-0
zookeeper-vol-zookeeper-0              Bound     pvc-68891ba1-3a94-11e8-a34e-0238efc27e9c   8Gi        RWO           rbd
```

For each extra replica, the stateful set will create a separate volume. Also, this way it is much easier to manage pods and PVCs at the same time.

Summary

Stateful sets are somehow left behind and most users don't even consider it. They are much better at managing stateful apps and persistent volumes. If you want to learn more about the stateful set, in general, check the blog post that I wrote a few months ago - Stateful Applications on Kubernetes (https://akomljen.com/stateful-applications-on-kubernetes/). Stay tuned for the next one.