**StatefulSet**

* Some apps need a stable environment, where objects are created in a **known order with fixed names**. Think of a replicated system like a message queue or a database - there's often a primary node and multiple secondaries. The secondaries depend on the primary starting first and they need to know how to find it so they can sync data. That's where you use a StatefulSet.
* StatefulSets are Pod controllers which can create multiple replicas in a stable environment. Replicas have known names, start consecutively and are individually addressable within the cluster.
* StatefulSet is used to manage stateful applications. It uses storage volumes to provide persistence for the workload.
* Example of Statefulsets: mysql, easticsearch, mongodb etc…
* Pods have persisted identity and not the random hash as in deployments. Eg: mysql-0, mysql-1, mysql-2, mysql-3 etc…
* Pods in statefulset are created/provisioned one by one in order (lowest to highest) and when scale down is required they are deprovisioned in reverse order (highest to lowest).
* For stateful applications with a StatefulSet controller, it is possible to set the first Pod as primary and other Pods as replicas—the first Pod will handle both read and write requests from the user, and other Pods always sync with the first Pod for data replication.

A picture containing diagram

Description automatically generated

* Each Pod in the spec gets its own dynamically-created PVC. Volumes are provisioned dynamically as new pods are created.
* If the Pod dies, a new Pod is created with the same name. The link between Pod and PVC remains in place even across Pod replacement.

Graphical user interface, application

Description automatically generated

* Deleting or scaling down a StatefulSet will not delete the volumes associated with the stateful application. This gives you your data safety. If you delete the MySQL Pod or if the MySQL Pod restarts, you can have access to the data in the same volume.

**StatefulSets provide the following advantages when compared to Deployment objects:**

1. Ordered numbers for each Pod.
2. The first Pod can be a primary, which makes it a good choice when creating a replicated database setup, which handles both reading and writing.
3. Other Pods act as replicas for reading purpose.
4. New Pods will only be created if the previous Pod is in running state and will clone the previous Pod’s data.
5. Deletion of Pods occurs in reverse order.

**Example: MySQL Workload.**Find the name of default storageclass and change "storageClassName" in the below YAML

kubectl get sc

**mysql.yaml**

apiVersion: apps/v1

kind: StatefulSet

metadata:

  name: mysql

spec:

  replicas: 2

  serviceName: mysql

  selector:

    matchLabels:

      app: mysql

  template:

    metadata:

      labels:

        app: mysql

    spec:

      terminationGracePeriodSeconds: 10

      containers:

        - name: mysql

          image: mysql:5.7

          args:

          - "--ignore-db-dir=lost+found"

          ports:

            - name: tcp

              protocol: TCP

              containerPort: 3306

          env:

            - name: MYSQL\_ROOT\_PASSWORD

              value: "tiger123"

          volumeMounts:

            - name: data

              mountPath: /var/lib/mysql

  volumeClaimTemplates:

  - metadata:

      name: data

    spec:

      accessModes: ["ReadWriteOnce"]

      storageClassName: "hostpath"

      resources:

        requests:

          storage: 1Gi

## What is a Headless Service?

When there is no need of load balancing or single-service IP addresses.We create a headless service which is used for creating a service grouping. That does not allocate an IP address or forward traffic.

**Note**: We do not use the load balancer service for a stateful application, but instead, create a headless service:

Now, create the headless service for the MySQL Pod.

# Headless service for stable DNS entries of StatefulSet members.

apiVersion: v1

kind: Service

metadata:

  name: mysql

  labels:

    app: mysql

spec:

  clusterIP: None

  selector:

    app: mysql

  ports:

    - name: tcp

      protocol: TCP

      port: 3306

* DNS entry for the service returns an IP address for each Pod in the StatefulSet.
* Each Pod gets its own DNS entry as well

**Create a Pod to test MySQL Server**

kubectl run -it --rm --image=mysql:5.6 **mysql-client** -- mysql -h **mysql-0.mysql.default.svc.cluster.local** -uroot -ptiger123

LoadBalancer Service for performing Read Operations.

apiVersion: v1

kind: Service

metadata:

  name: mysql-readonly

  labels:

    app: mysql

spec:

type: LoadBalancer

  selector:

    app: mysql

  ports:

    - name: tcp

      protocol: TCP

      port: 3306

Note: Remember that while Kubernetes helps you set up a stateful application, you will need to set up the data cloning and data sync by yourself. This cannot be done by the StatefulSets.