Kubernetes

StatefulSet

A **StatefulSet** is a Kubernetes resource designed to manage stateful applications. It is particularly useful when an application has a database attached and requires persistent storage, ensuring that the state of the application and its pods is maintained.

In scenarios where one of the database pods is deleted, a new pod is created to replace it. However, without a StatefulSet, the new pod may be assigned a different name and state, potentially causing issues for the application. This is where StatefulSets come into play.

A StatefulSet ensures:

- Pods have stable, unique identities (e.g., `mysql-statefulset-0`, `mysql-statefulset-1`).
- Each pod is associated with persistent storage volumes (PVs and PVCs) that retain their data across pod restarts.
- The configuration and state of each pod are preserved, enabling the application to continue functioning without disruption.

When you examine a StatefulSet manifest, it typically includes the following components:

1. **Pods**: Managed by the StatefulSet, each with a consistent identity and configuration.
2. **Persistent Volumes (PVs) and Persistent Volume Claims (PVCs)**: Ensure data persistence for each pod.
3. **Headless Service**: Used to provide stable network identities for the pods.

For example, if a pod in the StatefulSet is named `mysql-statefulset-0`, the number `0` indicates the pod's ordinal index, which reflects its stable state. If the pod is deleted, the StatefulSet ensures that a replacement pod is created with the same name and configuration, maintaining continuity for the application.

In summary, a StatefulSet is crucial for applications requiring persistent storage and stable pod identities, ensuring seamless recovery and consistency in the event of disruptions.

```
ubuntu@ip-172-31-46-229:~/Kubernetes-Manifest/mysql$ kubectl get pod -n mysql
NAME                  READY   STATUS    RESTARTS   AGE
mysql-statefulset-0   1/1     Running   0          107s
mysql-statefulset-1   1/1     Running   0          89s
mysql-statefulset-2   1/1     Running   0          65s
ubuntu@ip-172-31-46-229:~/Kubernetes-Manifest/mysql$
```

```
ubuntu@ip-172-31-46-229:~/Kubernetes-Manifest/mysql$ kubectl get pod -n mysql
NAME                  READY   STATUS              RESTARTS   AGE
mysql-statefulset-0   0/1     ContainerCreating   0          2s
mysql-statefulset-1   1/1     Running             0          16m
mysql-statefulset-2   1/1     Running             0          15m
ubuntu@ip-172-31-46-229:~/Kubernetes-Manifest/mysql$
```

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: mysql-statefulset
 namespace: mysql
spec:
 selector:
   matchLabels:
     app: myapp
 serviceName: mysql-service
 replicas: 3
 template:
   metadata:
     labels:
       app: myapp
   spec:
     containers:
     - name: mysql-container
       image: mysql:latest
       ports:
       - containerPort: 3306
```

Author- Amitt Ashok
Kubernetes-Manifest: -https://github.com/AmittAshok/Kubernetes-Manifest.git

```
      env:
      - name: MYSQL_ROOT_PASSWORD
        value: root
      - name: MYSQL_DATABASE
        value: devops
      volumeMounts:
      - name: mysql-data
        mountPath: /var/lib/mysql
volumeClaimTemplates:
- metadata:
    name: mysql-data
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 1Gi
```

ConfigMap:

In an application, there may be certain types of data that are **not highly sensitive** but are still crucial for the application's configuration. For example, data like database **names, user IDs, or configuration settings**. This is where a ConfigMap comes into play.

A ConfigMap is used to store non-sensitive configuration data in **key-value pairs**. It allows you to decouple configuration artifacts from application code. Unlike Secrets,

which are meant for sensitive data, ConfigMaps are used for **general-purpose configuration.**

You can create a ConfigMap and reference it in other Kubernetes resources such as StatefulSets, Deployments, or Pods using **configMapKeyRef**. This enables the application to access configuration data dynamically without hardcoding it into the application.

```
apiVersion: v1

kind: ConfigMap

metadata:

 name: mysql-configmap

 namespace: mysql

data:

 MYSQL_DATABASE: devops
```

Secret

Similar to a ConfigMap, which is used to store non-sensitive data, a **Secret** is a Kubernetes resource designed to store **sensitive information** securely. Secrets are used for data like passwords, API keys, TLS certificates, and other credentials that should not be exposed in plain text.

Secrets can be referenced in application manifest files using `secretKeyRef`, allowing sensitive data to be injected into containers without embedding it directly in the manifest files or application code.

```
apiVersion: v1
kind: Secret
metadata:
 name: mysql-secret
 namespace: mysql
type: Opaque
data:
 MYSQL_ROOT_PASSWORD: cm9vdAo=   #base64 encode password
```

StatefulSet with ConfigMap and Secret

In this StatefulSet, the database name is sourced from a **ConfigMap**, while the root password is taken from a **Secret**. This approach ensures that sensitive data, like passwords, is securely managed while configuration data, such as the database name, is kept dynamic and decoupled from the application code.

Below is how the updated StatefulSet manifest looks:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: mysql-statefulset
 namespace: mysql
spec:
 selector:
   matchLabels:
     app: myapp
 serviceName: mysql-service
 replicas: 3
 template:
   metadata:
     labels:
       app: myapp
```

```
  spec:
    containers:
    - name: mysql-container
      image: mysql:latest
      ports:
      - containerPort: 3306
      env:
      - name: MYSQL_ROOT_PASSWORD
        valueFrom:
          secretKeyRef:
            key: MYSQL_ROOT_PASSWORD
            name: mysql-secret    # secret name
      - name: MYSQL_DATABASE
        valueFrom:
          configMapKeyRef:
          name: mysql-configmap   # configmap name
          key: MYSQL_DATABASE
      volumeMounts:
      - name: mysql-data
        mountPath: /var/lib/mysql
volumeClaimTemplates:
- metadata:
    name: mysql-data
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 1Gi
```

Conclusion

Thank you for following along! In this article, we covered the concept of StatefulSets and explored how ConfigMaps and Secrets can be used to store non-sensitive and sensitive data, respectively, in a dynamic and secure manner. These features enable better configuration management and enhance the security of applications running in Kubernetes

```
ubuntu@ip-172-31-46-229:~/Kubernetes-Manifest/mysql$ kubectl get all -n mysql
NAME                      READY   STATUS    RESTARTS   AGE
pod/mysql-statefulset-0   1/1     Running   0          21m
pod/mysql-statefulset-1   1/1     Running   0          21m
pod/mysql-statefulset-2   1/1     Running   0          21m

NAME                    TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/mysql-service   ClusterIP   None         <none>        3306/TCP   84m

NAME                                 READY   AGE
statefulset.apps/mysql-statefulset   3/3     30m
ubuntu@ip-172-31-46-229:~/Kubernetes-Manifest/mysql$ kubectl get configmap -n mysql
NAME               DATA   AGE
kube-root-ca.crt   1      85m
mysql-configmap    2      44m
ubuntu@ip-172-31-46-229:~/Kubernetes-Manifest/mysql$ kubectl get secret -nmysql
NAME           TYPE     DATA   AGE
mysql-secret   Opaque   1      22m
```