Stateful sets

OW

Explain

- What is stateful sets?
- API
- Limitation

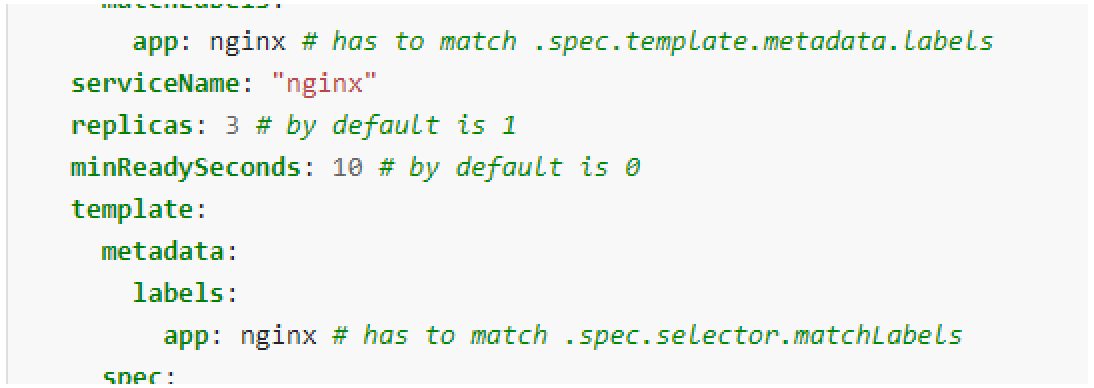
StatefulSets

- StatefulSet is the workload API object used to manage stateful application
- Like a Deployment, a StatefutSet manages Pods are based on an identical Spec
- StatefulSet maintains a sticky identity for each of their pods
- Each pod has persistent identifier that maintains across any rescheduling.
- When application need stable network identifier or stable persistent storage, StatefulSet are useful

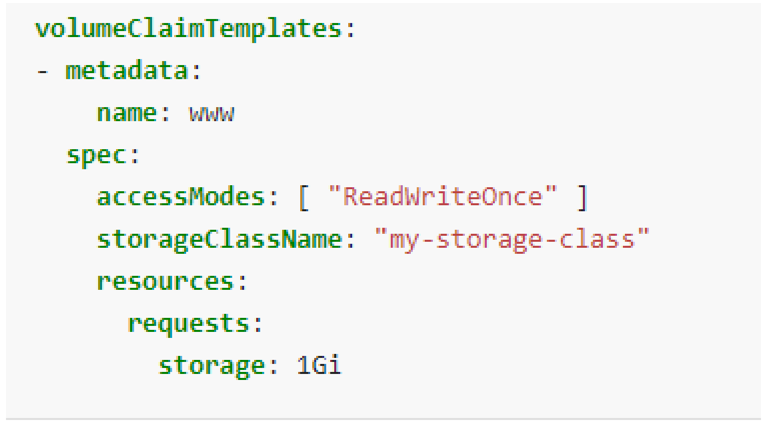
..

- Deleting or scaling down StatefulSet will not delete the volumes associated .
- StatefulSet currently require Headless Service to be responsible for network identity of the pods. Kubernetes admin has to create the service.
- StatefulSet persistent volume must be provided by storage class and storage provisioner

Api for stateful set



```yaml
    app: nginx # has to match .spec.template.metadata.labels
serviceName: "nginx"
replicas: 3 # by default is 1
minReadySeconds: 10 # by default is 0
template:
  metadata:
    labels:
      app: nginx # has to match .spec.selector.matchLabels
  spec:
```



```yaml
volumeClaimTemplates:
- metadata:
    name: www
  spec:
    accessModes: [ "ReadWriteOnce" ]
    storageClassName: "my-storage-class"
    resources:
      requests:
        storage: 1Gi
```

..

- A Headless Service, named nginx, is used to control the network domain.
- The StatefulSet, named web, has a Spec that indicates that 3 replicas of the nginx container will be launched in unique Pods.
- The volumeClaimTemplates will provide stable storage using PersistentVolumes provisioned by a PersistentVolume Provisioner.
- You must set the .spec.selector field of a StatefulSet to match the labels of its .spec.template.metadata.labels [Pod Selector]

Stable Network ID

- Each pod in StatefulSet drives its hostname from StatefulSet and the ordinal of the pod. The pattern constructed hostname is $(statefulsetname)-$(ordinal)
- The yaml in lab will create three pod: web-0,web-1,web-2.
- StatefulSet can use a Headless service to control the domain of the Pods.
- The domain managed by this service takes form: $(servicename).$(namespace).svc.cluster.local, where “cluster.local” is the cluster domain.
- Each pod is created, it gets matching DNS subdomain: $(podname).$(governing service domain).

• •

Cluster Domain	Service (ns/name)	StatefulSet (ns/name)	StatefulSet Domain	Pod DNS	Pod Hostname
cluster.local	default/nginx	default/web	nginx.default.svc.cluster.local	web-{0..N-1}.nginx.default.svc.cluster.local	web-{0..N-1}
cluster.local	foo/nginx	foo/web	nginx.foo.svc.cluster.local	web-{0..N-1}.nginx.foo.svc.cluster.local	web-{0..N-1}
kube.local	foo/nginx	foo/web	nginx.foo.svc.kube.local	web-{0..N-1}.nginx.foo.svc.kube.local	web-{0..N-1}

Deployment and Scaling

- For a StatefulSet with N replicas, when Pods are being deployed, they are created sequentially, in order from {0..N-1}.
- When Pods are being deleted, they are terminated in reverse order, from {N-1..0}.
- Before a scaling operation is applied to a Pod, all of its predecessors must be Running and Ready.
- Before a Pod is terminated, all of its successors must be completely shutdown

Update Strategies

- A StatefulSet’s .spec.updateStrategy field allows you to configure rolling update:
  - OnDelete: When a StatefulSet’s .spec.updateStrategy.type is set to OnDelete, the StatefulSet controller will not automatically update the pods. Users must manually delete Pods to cause the controller to create new pods that reflect modification made to .spec.template
  - RollingUpdate: When .spec.udpateStrategy.type is set to RollingUpdate, the controller will delete and recreate each pod. It will proceed from largest ordinal to the smallest. It will wait until the updated pod is Running and ready prior to update the next one.

Partitioned rolling updates

- The RollingUpdate update strategy can be partitioned, by specifying a .spec.updateStrategy.rollingUpdate.partition. If a partition is specified, all Pods with an ordinal that is greater than or equal to the partition will be updated when the StatefulSet's .spec.template is updated.
- All Pods with an ordinal that is less than the partition will not be updated, and, even if they are deleted, they will be recreated at the previous version.
- They are useful if you want to stage an update, roll out a canary, or perform a phased roll out.

PersistentVolumeClaim retention

- The optional .spec.persistentVolumeClaimRetentionPolicy field controls if and how PVCs are deleted during the lifecycle of a StatefulSet. You must enable the StatefulSetAutoDeletePVC feature gate to use this field. Once enabled, there are two policies you can configure for each StatefulSet:
- whenDeleted
  - configures the volume retention behavior that applies when the StatefulSet is deleted
- whenScaled
  - configures the volume retention behavior that applies when the replica count of the StatefulSet is reduced; for example, when scaling down the set.

• •

```yaml
apiVersion: apps/v1
kind: StatefulSet
...
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Delete
...
```

Replicas

- spec.replicas is an optional field that specifies the number of desired Pods. It defaults to 1.
- Should you manually scale a deployment, example via kubectl scale statefulset statefulset --replicas=X, and then you update that StatefulSet based on a manifest (for example: by running kubectl apply -f statefulset.yaml), then applying that manifest overwrites the manual scaling that you previously did.
- If a HorizontalPodAutoscaler (or any similar API for horizontal scaling) is managing scaling for a Statefulset, don't set .spec.replicas. Instead, allow the Kubernetes control plane to manage the .spec.replicas field automatically.