Déploiement d'une base de données MongoDB dans Kubernetes avec StatefulSets

1- Mise en place

Exécutez la commande suivante pour cloner le replica set MongoDB/Kubernetes à partir du repo Github:

```
git clone https://github.com/thesandlord/mongo-k8s-sidecar.git
```

Une fois cloné, accédez au répertoire `StatefulSet` avec la commande suivante:

```
cd ./mongo-k8s-sidecar/example/StatefulSet/
```

Une fois que vous avez vérifié que les fichiers ont été téléchargés et que vous êtes dans le bon répertoire, allons-y et créons un `StorageClass` Kubernetes.

Créer la StorageClass

La `StorageClass` indique à Kubernetes le type de stockage que vous souhaitez utiliser pour les nœuds de base de données. Sur Google Cloud, vous avez plusieurs choix de stockage : SSD et disques durs.

Si vous regardez dans le répertoire *`StatefulSet`* (vous pouvez le faire en exécutant la commande *`ls`*), vous verrez les fichiers de configuration SSD et HDD pour Azure et Google Cloud. Exécutez la commande suivante pour examiner le fichier *`googlecloud_ssd.yaml`*:

```
cat googlecloud_ssd.yaml
```

Cette configuration crée une nouvelle StorageClass appelée «fast» qui est sauvegardée par des volumes SSD. Exécutez la commande suivante pour déployer le StorageClass:

```
kubectl apply -f googlecloud_ssd.yaml
```

Maintenant que notre StorageClass est configuré, notre StatefulSet peut maintenant demander un volume qui sera automatiquement créé.

2- Déploiement du service Headless et StatefulSet

Trouvez et inspectez les fichiers

Avant de nous plonger dans ce que sont le service headless et les StatefulSets, ouvrons le fichier de configuration (mongo-statefulset.yaml) dans lequel ils sont tous deux hébergés.

```
nano mongo-statefulset.yaml
```

Vous devriez recevoir la sortie suivante (sans les pointeurs vers le service Headless et le contenu StatefulSet):

```
apiVersion: v1   <----------   Headless Service configuration
kind: Service
metadata:
  name: mongo
  labels:
    name: mongo
spec:
  ports:
  - port: 27017
    targetPort: 27017
  clusterIP: None
  selector:
    role: mongo
---
```

```
---
apiVersion: apps/v1beta1    <------- StatefulSet configuration
kind: StatefulSet
metadata:
  name: mongo
spec:
  serviceName: "mongo"
  replicas: 3
  template:
    metadata:
      labels:
        role: mongo
        environment: test
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: mongo
          image: mongo
          command:
            - mongod
            - "--replSet"
            - rs0
            - "--smallfiles"
            - "--noprealloc"
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongo-persistent-storage
              mountPath: /data/db
        - name: mongo-sidecar
          image: cvallance/mongo-k8s-sidecar
          env:
            - name: MONGO_SIDECAR_POD_LABELS
              value: "role=mongo,environment=test"
```

```
  volumeClaimTemplates:
  - metadata:
      name: mongo-persistent-storage
      annotations:
        volume.beta.kubernetes.io/storage-class: "fast"
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 100Gi
```

Supprimez les indicateurs suivants du fichier (lignes 49 et 50):

```
- "--smallfiles"
- "--noprealloc"
```

Assurez-vous que cette section de votre fichier ressemble à ce qui suit:

```
containers:
        - name: mongo
          image: mongo
          command:
            - mongod
            - "--replSet"
            - rs0
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongo-persistent-storage
              mountPath: /data/db
```

Quittez l'éditeur nano avec **CTRL** + **X** > **Y** > **ENTER**.

Headless Service: aperçu

La première section de `mongo-statefulset.yaml`fait référence à un `service headless`. Dans les termes de Kubernetes, un service décrit des stratégies ou des règles pour accéder à des pods spécifiques. En bref, un service headless est un service qui ne prescrit pas d'équilibrage de charge. Lorsqu'il est combiné avec StatefulSets, cela nous donnera des DNS individuels pour accéder à nos pods, et à son tour un moyen de se connecter individuellement à tous nos nœuds MongoDB. Dans le fichier `yaml`, vous pouvez vous assurer que le service est sans tête en vérifiant que le champ `clusterIP` est défini sur `None`.

StatefulSet: présentation

La configuration StatefulSet est la deuxième section de `mongo-statefulset.yaml`.

C'est la charge de travail qui exécute MongoDB et qui orchestre vos ressources Kubernetes.

En référençant le fichier `yaml`, nous voyons que la première section décrit l'objet StatefulSet.

Ensuite, nous passons à la section Métadonnées, où les étiquettes et le nombre de répliques sont spécifiés.

Vient ensuite la spécification du pod. Le `terminationGracePeriodSeconds` est utilisé pour arrêter correctement le pod lorsque vous réduisez le nombre de répliques. Ensuite, les configurations des deux conteneurs sont affichées.

Le premier exécute MongoDB avec des indicateurs de ligne de commande (command line flags) qui configurent le nom du jeu de réplicas. Il monte également le volume de stockage persistant sur `/data/db`: l'emplacement où MongoDB enregistre ses données. Le deuxième conteneur exécute le side-car.
Ce conteneur side-car configurera automatiquement le jeu de réplicas MongoDB. Comme mentionné précédemment, un "side-car" est un conteneur d'assistance qui aide le conteneur principal à exécuter ses travaux et ses tâches.
Enfin, il y a le `volumeClaimTemplates`. C'est ce qui parle à la StorageClass que nous avons créée avant de provisionner le volume. Il provisionne un disque de 100 Go pour chaque réplique MongoDB.

Déployer le service Headless et le StatefulSet

Maintenant que nous avons une compréhension de base de ce que sont un headless service et un StatefulSet, allons-y et déployez-les. Étant donné que les deux sont intégrés `mongo-statefulset.yaml`, nous pouvons exécuter la commande suivante pour les exécuter tous les deux: `kubectl apply -f mongo-statefulset.yaml`
Vous devriez recevoir la sortie suivante:

```
service "mongo" created
statefulset "mongo" created
```

3- Connectez-vous à l'ensemble de réplicas MongoDB

Maintenant que nous avons un cluster en cours d'exécution et que notre replica set est déployé, allons-y et connectez-vous à lui.

Lancement et affichage du replica set MongoDB

À ce stade, vous devriez avoir trois pods créés dans votre cluster. Ceux-ci correspondent aux trois nœuds de votre replica set MongoDB. Exécutez cette commande pour afficher: `kubectl get pods`

Attendez que les trois membres soient créés avant de continuer. Connectez-vous au premier membre du jeu de réplicas: `kubectl exec -ti mongo-0 mongo`
Vous disposez maintenant d'un environnement *REPL* connecté à MongoDB.
Instancions l'ensemble de réplicas avec une configuration par défaut en exécutant la commande **`rs.initiate()`**. Imprimez la configuration du jeu de réplicas; exécutez la **`rs.conf()`**commande:

Cela génère les détails du membre actuel du replica set `rs0`. Dans ce lab, vous ne voyez qu'un seul membre. Pour obtenir des détails sur tous les membres, vous devez exposer l'ensemble de réplicas via des services supplémentaires tels que nodeport ou load balancer.

```
rs0:OTHER> rs.conf()
{
        "_id" : "rs0",
        "version" : 1,
        "protocolVersion" : NumberLong(1),
        "writeConcernMajorityJournalDefault" : true,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27017",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {
                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                }
        ],
        "settings" : {
                "chainingAllowed" : true,
                "heartbeatIntervalMillis" : 2000,
                "heartbeatTimeoutSecs" : 10,
                "electionTimeoutMillis" : 10000,
                "catchUpTimeoutMillis" : -1,
```

```
                "catchUpTakeoverDelayMillis" : 30000,
                "getLastErrorModes" : {
                },
                "getLastErrorDefaults" : {
                        "w" : 1,
                        "wtimeout" : 0
                },
                "replicaSetId" : ObjectId("5c526b6501fa2d29fc65c48c")
        }
}
```

Tapez "exit" et appuyez sur Entrée pour quitter REPL.

4- Mise à l'échelle du replica set MongoDB

Un gros avantage de Kubernetes et des StatefulSets est que vous pouvez augmenter et diminuer le nombre de réplicas MongoDB avec une seule commande! Pour augmenter le nombre de membres du jeu de réplicas de 3 à 5, exécutez cette commande:

```
kubectl scale --replicas=5 statefulset mongo
```

Dans quelques minutes, il y aura 5 pods MongoDB. Exécutez cette commande pour les afficher: `kubectl get pods`

Pour réduire le nombre de membres du jeu de réplicas de 5 à 3, exécutez cette commande:

```
kubectl scale --replicas=3 statefulset mongo
```

En quelques secondes, il y a 3 pods MongoDB. Exécutez cette commande pour afficher:

```
kubectl get pods
```

5- Utilisation du replica set MongoDB

Chaque pod d'un StatefulSet soutenu par un Headless service aura un nom DNS stable. Le modèle suit ce format:**<pod-name>.<service-name>.** Cela signifie que les noms DNS du jeu de réplicas MongoDB sont:

Vous pouvez utiliser ces noms directement dans l' URI de la chaîne de connexion de votre application. L'utilisation d'une base de données n'entre pas dans le cadre de ce lab, mais dans ce cas, l'URI de la chaîne de connexion serait:

`"mongodb://mongo-0.mongo,mongo-1.mongo,mongo-2.mongo:27017/dbname_?"`

6- Nettoyer

Pour nettoyer les ressources déployées, exécutez les commandes suivantes pour supprimer le StatefulSet, le service Headless et les volumes provisionnés.

- Supprimez le StatefulSet: `kubectl delete statefulset mongo`
- Supprimer le service: `kubectl delete svc mongo`
- Supprimer les volumes: `kubectl delete pvc -l role=mongo`
- Enfin, vous pouvez supprimer le cluster, si vous n’en avez plus besoin: `gcloud container clusters delete "<cluster-name>”`