# Déploiement de Cassandra avec des Stateful Sets

Cet exercice explique comment développer un déploiement Cassandra en cloud natif sur Kubernetes. Ici, **Cassandra SeedProvider** personnalisé (custom) permet à Cassandra de découvrir de nouveaux nœuds Cassandra lorsqu'ils rejoignent le cluster.

**StatefulSet** facilite le déploiement d'applications avec état (**stateful**) dans un environnement en cluster.

## Cassandra sur Docker

Les pods de cet exercice utilisent l'image **gcr.io/google-samples/cassandra:v13** du registre de conteneurs de Google. L'image Docker ci-dessus est sur **debian-base** et inclut **OpenJDK 8**.

Cette image comprend une installation Cassandra standard à partir du dépôt Apache Debian. En utilisant des variables d'environnement, vous pouvez modifier les valeurs insérées dans **cassandra.yaml.**

| ENV VAR | DEFAULT VALUE |
|---|---|
| CASSANDRA_CLUSTER_NAME | 'Test Cluster' |
| CASSANDRA_NUM_TOKENS | 32 |
| CASSANDRA_RPC_ADDRESS | 0.0.0.0 |

## Objectifs

- Créer et valider un service **headless** Cassandra.
- Utilisez un **StatefulSet** pour créer un anneau Cassandra.
- Valider le **StatefulSet**.
- Modifier le **StatefulSet**.
- Supprimez le **StatefulSet** et ses pods.

## Avant de commencer

Pour terminer cet exercice, vous devriez déjà avoir une connaissance de base des pods, des services et des StatefulSets. De plus, vous devriez :

- Installer et configurer l'outil de ligne de commande **kubectl.**
- Télécharger **cassandra-service.yaml** et **cassandra-statefulset.yaml.**
- Exécuter un cluster Kubernetes pris en charge.

## Instructions supplémentaires pour minikube

Attention : **Minikube** utilise par défaut 1024 Mo de mémoire et 1 CPU. L'exécution de Minikube avec la configuration de ressources par défaut entraîne des erreurs de ressources

insuffisantes pendant ce tutoriel. Pour éviter ces erreurs, démarrez **Minikube** avec les paramètres suivants :

**sudo minikube start --memory 5120 --cpus=4**

## Création d'un service headless Cassandra

Un service Kubernetes décrit un ensemble de pods effectuant la même tâche.

Le service suivant est utilisé pour les recherches DNS entre les pods Cassandra et les clients au sein du cluster Kubernetes.

**cassandra-service.yaml**

```
apiVersion: v1
kind: Service
metadata:
labels:
app: cassandra
name: cassandra
spec:
clusterIP: None
ports:
- port: 9042
selector:
app: cassandra
```

1. Lancez une fenêtre de terminal dans le répertoire dans lequel vous avez téléchargé les fichiers de manifeste (manifest).

2. Créez un service pour suivre tous les nœuds StatefulSet Cassandra à partir du fichier **cassandra-service.yaml** :

**sudo kubectl create -f  cassandra-service.yaml**

## Validation (optionnel)

Obtenez le service Cassandra.

**sudo kubectl get svc cassandra**

**La réponse est :**

```
NAME      TYPE       CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
cassandra   ClusterIP   None         <none>        9042/TCP   45s
```

Service creation failed if anything else is returned. Read Debug Services for common issues.

**Utiliser un StatefulSet pour créer un anneau de Cassandra (Cassandra ring)**

Le manifeste StatefulSet, inclus ci-dessous, crée un anneau (ring) Cassandra composé de trois pods.

Remarque : Cet exercice utilise le **provisioner par défaut** pour Minikube. Veuillez mettre à jour le StatefulSet suivant pour le cloud avec lequel vous travaillez.

**cassandra-statefulset.yaml**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: cassandra
  labels:
    app: cassandra
spec:
  serviceName: cassandra
  replicas: 3
  selector:
    matchLabels:
      app: cassandra
  template:
    metadata:
      labels:
        app: cassandra
    spec:
      terminationGracePeriodSeconds: 1800
      containers:
      - name: cassandra
        image: gcr.io/google-samples/cassandra:v13
        imagePullPolicy: Always
        ports:
        - containerPort: 7000
          name: intra-node
        - containerPort: 7001
          name: tls-intra-node
        - containerPort: 7199
          name: jmx
        - containerPort: 9042
          name: cql
        resources:
          limits:
            cpu: "500m"
            memory: 1Gi
          requests:
```

```
    cpu: "500m"
    memory: 1Gi
securityContext:
  capabilities:
    add:
      - IPC_LOCK
lifecycle:
  preStop:
    exec:
      command:
      - /bin/sh
      - -c
      - nodetool drain
env:
  - name: MAX_HEAP_SIZE
    value: 512M
  - name: HEAP_NEWSIZE
    value: 100M
  - name: CASSANDRA_SEEDS
    value: "cassandra-0.cassandra.default.svc.cluster.local"
  - name: CASSANDRA_CLUSTER_NAME
    value: "K8Demo"
  - name: CASSANDRA_DC
    value: "DC1-K8Demo"
  - name: CASSANDRA_RACK
    value: "Rack1-K8Demo"
  - name: POD_IP
    valueFrom:
      fieldRef:
        fieldPath: status.podIP
readinessProbe:
  exec:
    command:
    - /bin/bash
    - -c
    - /ready-probe.sh
  initialDelaySeconds: 15
  timeoutSeconds: 5
# These volume mounts are persistent. They are like inline claims,
# but not exactly because the names need to match exactly one of
# the stateful pod volumes.
volumeMounts:
```

**cassandra-statefulset.yaml**

```
        - name: cassandra-data
          mountPath: /cassandra_data
  # These are converted to volume claims by the controller
  # and mounted at the paths mentioned above.
  # do not use these in production until ssd GCEPersistentDisk or other ssd pd
  volumeClaimTemplates:
  - metadata:
      name: cassandra-data
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: fast
      resources:
        requests:
          storage: 1Gi
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
provisioner: k8s.io/minikube-hostpath
parameters:
  type: pd-ssd
```

1. Mettez à jour le **StatefulSet** si nécessaire.

2. Créez le Cassandra **StatefulSet** à partir du fichier **cassandra-statefulset.yaml** :

```
sudo kubectl create -f  cassandra-statefulset.yaml
```

## Valider le StatefulSet Cassandra

1. Obtenez le StatefulSet Cassandra :

```
sudo kubectl get statefulset cassandra
```

La réponse devrait être :

```
NAME        DESIRED   CURRENT   AGE
cassandra   3         0         13s
```

La ressource StatefulSet déploie les pods de manière séquentielle.

2. Obtenez les pods pour voir le statut de création commandé :

```
sudo kubectl get pods -l="app=cassandra"
```

**La réponse devrait être :**

```
NAME        READY    STATUS              RESTARTS   AGE
cassandra-0 1/1      Running             0          1m
cassandra-1 0/1      ContainerCreating   0          8s
```

Le déploiement des trois pods peut prendre plusieurs minutes. Une fois qu'ils sont déployés, la même commande retourne :

```
NAME        READY    STATUS    RESTARTS   AGE
cassandra-0 1/1      Running   0          10m
cassandra-1 1/1      Running   0          9m
cassandra-2 1/1      Running   0          8m
```

Exécutez l'outil **nodetool** Cassandra pour afficher l'état de l'anneau.

**sudo kubectl exec -it cassandra-0 -- nodetool status**

La réponse devrait être comme ceci :

```
Datacenter: DC1-K8Demo
======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens       Owns (effective)  Host ID                               Rack
UN  172.17.0.5  83.57 KiB  32           74.0%             e2dd09e6-d9d3-477e-96c5-45094c08db0f
Rack1-K8Demo
UN  172.17.0.4  101.04 KiB  32          58.8%             f89d6835-3a42-4419-92b3-0e62cae1479c
Rack1-K8Demo
UN  172.17.0.6  84.74 KiB  32           67.1%             a6a1e8c2-3dc5-4417-b1a0-26507af2aaad
Rack1-K8Demo
```

**Modification du Cassandra StatefulSet**

Utilisez **kubectl edit** pour modifier la taille d'un **StatefulSet Cassandra**.

Exécutez la commande suivante :

**sudo kubectl edit statefulset cassandra**

Cette commande ouvre un éditeur dans votre terminal. La ligne à modifier est le champ réplicas. L'exemple suivant est un extrait du fichier StatefulSet :

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: StatefulSet
```

```
metadata:
  creationTimestamp: 2016-08-13T18:40:58Z
  generation: 1
  labels:
  app: cassandra
  name: cassandra
  namespace: default
  resourceVersion: "323"
  selfLink: /apis/apps/v1/namespaces/default/statefulsets/cassandra
  uid: 7a219483-6185-11e6-a910-42010a8a0fc0
spec:
  replicas: 3
```

1. Définissez le nombre de répliques à 4, puis enregistrez le manifeste.

Le StatefulSet contient maintenant 4 pods.

2. Demandez au **StatefulSet** de Cassandra de vérifier :

```
sudo kubectl get statefulset cassandra
```

La réponse devrait être :

```
NAME       DESIRED   CURRENT   AGE
cassandra  4         4         36m
```

## Nettoyage

**La** suppression **ou le** scale down (réduction) d'un StatefulSet **ne supprime pas les volumes associés à StatefulSet. Ce paramètre est destiné à votre sécurité, car vos données ont plus de valeur que la purge automatique de toutes les ressources StatefulSet associées.**

Avertissement : En fonction de la storage class et de la reclaim policy, la suppression de **PersistentVolumeClaims** peut entraîner la suppression des volumes associés. Ne supposez jamais que vous pourrez accéder aux données si leurs volume claims sont supprimés.

**1**. Exécutez les commandes suivantes (chaînées ensemble en une seule commande) pour supprimer tout ce qui se trouve dans le StatefulSet Cassandra :

```
grace=$(kubectl get po cassandra-0 -o=jsonpath='{.spec.terminationGracePeriodSeconds}') \
  && sudo kubectl delete statefulset -l app=cassandra \
  && echo "Sleeping $grace" \
  && sleep $grace \
  && sudo kubectl delete pvc -l app=cassandra
```

**2.** Exécutez la commande suivante pour supprimer le service Cassandra.

```
sudo kubectl delete service -l app=cassandra
```