Desafio Técnico - MongoDB no AKS (Kubernetes)

Estrutura do projeto

mongodb-minikube/
├── Dockerfile
├── entrypoint.sh
├── backup_script.sh
├── kubernetes/
│ ├── 01-mongodb-deployment.yaml
│ ├── 02-mongodb-service.yaml
│ └── 03-mongodb-pvc.yaml
└── README.md, docs, etc.

Os detalhes e comentários estão nos próprios arquivos. O deployment e manutenção está descrito aqui, com comentários adicionais.

Passos para o funcionamento do cluster

- Instalação do docker
- Instalação do minikube
- Instalação do kubectl
- Instalação do mongodb
- Criação do Dockerfile, com descrições no próprio arquivo.
- Criação do entrypoint.sh, referenciado no Dockerfile, com descrições no arquivo.
- Criação do backup_script.sh, referenciado no próprio arquivo.

- Inicialização do mongodb
  systemctl start mongodb

- Construção da imagem docker:
  docker build -t mongo-custom:v1 .
  docker images | grep mongo-custom # para ver se foi criada

- Inicialização do Minikube:

  minikube start --driver=docker
  minikube image load mongo-custom:v1 # usando a imagem criada
  minikube start
  minikube addons enable metrics-server
  minikube dashboard

Deployment

Uma vez criados os arquivos conforme a estrutura apresentada, na pasta do projeto (mongodb-minikube), inicializar o deployment:

- kubectl apply -f kubernetes/

Uma vez inicializado, os comandos abaixo apresentarão status específicos:

- kubectl get deployment mongodb-deployment #deployment
- kubectl get pods -l app=mongodb #pods
- kubectl get pvc mongodb-pvc #persistência
- kubectl get service mongodb-service #serviço do mongo no kube

Aguarde até que o Pod do MongoDB esteja no estado `Running` e `Ready`. Vai variar conforme a máquina, número de processadores, etc., ou esteja sem erros.

- kubectl get pods -w -l app=mongodb # um tail -f dos pods

Persistência e testes

PODs

No bash, pegar o nome do POD:

- MONGODB_POD_NAME=$(kubectl get pods -l app=mongodb -o jsonpath='{.items[0].metadata.name}')

echo "Nome do Pod MongoDB: $MONGODB_POD_NAME"

Inserir dados (senhas estão padrão) no pod:

- kubectl exec -it $MONGODB_POD_NAME -- mongosh admin -u admin -p password --authenticationDatabase admin

No Mongosh:

- use mydatabase
- db.mycollection.insertOne({ "name": "Teste", "value": 123 });
- db.mycollection.find();
- exit

Derrubar o pod:

- kubectl delete pod $MONGODB_POD_NAME

Procurar por um novo pod, que esteja com status running e ready:

- kubectl get pods -w -l app=mongodb

Dados

Se não surgir um pod ou o status estiver diferente, a persistência falhou.
A persistência do dados se dá aplicando, no mongosh do novo pod:

- use mydatabase
- db.mycollection.find();
- exit

O documento “Teste, 123” deve ser visualizado aí. Caso não seja, houve falha na persistência do dados, e deve haver revisão no deployment.

Backup

Execução do backup:

- kubectl exec -it $MONGODB_POD_NAME /usr/local/bin/backup_script.sh

Importante entender aqui que o backup foi copiado de /usr/local/bin na inicialização do POD, e é a imagem do arquivo existente no deployment do cluster.

Itens de manutenção e gerenciamento:

Desligar o minikube e eliminar o deployment:

- kubectl delete -f kubernetes/
- minikube stop
- minikube delete

Parar o cluster, sem desligar, para outras manutenções (mais rápido):

- kubectl scale deployment mongodb --replicas=0 -n mongodb

Lembrando que parar escalar o cluster, basta aumentar o número de réplicas, que serão mais PODs para responder às requisições.

Na maioria das nuvens, no geral, há 3 possibilidades de scalling, que dependem do custo para aquisição / disponibilidade / necessidade: Preemptive, preallocated (mais caro), shared.