Compte rendu TP3

Romain LECOUVREUR & Victorien GONTIER–DURAND

- ***Quelle est la différence entre un scaling horizontal et vertical***

  Le scaling horizontal :

  - Ajoute ou retire des pods sans changer la taille de chaque pod. C'est comme ajuster le nombre d'équipes pour répondre à la demande croissante ou décroissante.

  Le scaling vertical :

  - Modifie la taille de chaque pod sans forcément changer leur nombre. Cela revient à augmenter ou diminuer les ressources (comme la puissance CPU ou la mémoire) de chaque membre de l'équipe plutôt que d'ajouter de nouveaux membres.

- ***Quels objets Kubernetes peut-on scale avec le HPA***

  Le HPA de Kubernetes peut scaler ces objets :

  - Déploiements (Deployments) : Utilisés pour déployer des applications. Le HPA ajuste le nombre de pods en fonction de la charge.
  - StatefulSets : Moins fréquents, utilisés pour des applications nécessitant un stockage persistant. Le HPA peut aussi ajuster le nombre de pods ici.
  - Répliques de Contrôleurs (ReplicaSets) : Parfois directement scalés par le HPA, mais généralement gérés via des déploiements dans la pratique.

- ***Pourquoi scale une application***
  Scaler une application permet de :

    - S'adapter à la demande : Ajuster automatiquement les ressources pour répondre aux pics ou baisses d'utilisation.
    - Économiser les ressources : Utiliser juste ce qu'il faut pour éviter le gaspillage ou les ralentissements.
    - Maintenir la disponibilité : Assurer que l'application reste disponible même en cas d'augmentation soudaine de la demande.

  En bref, le scaling aide à s'ajuster à la demande tout en économisant les ressources et en maintenant l'application disponible et réactive.

- ***Quel est le résulat de la commande :***

  kubectl get --raw "/apis/metrics.k8s.io/v1beta1/nodes" | jq .

- ***Expliquer simplement le résulat de cette commande***

  Cette commande interroge l'API de métriques de Kubernetes pour récupérer les métriques des nœuds du cluster. Voici ce que chaque partie du résultat signifie :

  - kind: Indique le type de données retournées, ici une liste de métriques de nœuds.
  - apiVersion: La version de l'API utilisée pour récupérer ces métriques.
  - items: Une liste contenant les détails des métriques pour chaque nœud du cluster.

  Pour chaque nœud (dans ce cas, un nœud nommé "zebi-control-plane"), voici ce que vous avez :

  - metadata: Contient des informations sur le nœud, comme son nom, sa configuration, etc.
  - timestamp: Horodatage indiquant quand ces métriques ont été collectées.
  - window: La période de temps sur laquelle les métriques ont été collectées.
  - usage: Les métriques elles-mêmes, avec les valeurs de consommation de CPU et de mémoire du nœud à ce moment-là. Dans cet exemple :
    - cpu: 330304581 nanocores (unité de mesure pour la consommation de CPU)
    - memory: 740520Ki (Kibioctets de mémoire utilisée)

  Ces données fournissent des informations sur l'utilisation actuelle des ressources du nœud, ce qui est utile pour comprendre les tendances de consommation et prendre des décisions en matière de scaling ou d'allocation de ressources dans le cluster Kubernetes.

- ***Builder votre image docker avec le nom mon-app et le tag v0.1***

  Quelle est la commande docker pour créer son image

docker build --build-arg http_proxy=http://c3-lecouvr211:mdp@192.168.0.2:3128 --build-arg https_proxy=http://c3-lecouvr211:mdp@192.168.0.2:3128 -t mon-app:v0.1 /home/user/kubernetes/express/

Ouvrer 3 terminals avec, respectivement, comme commandes :

watch -n 1 -t kubectl get pods

watch -n 1 -t -w kubectl get hpa

curl mon-app.local/cpu

- ***Que remarquez-vous dans les 2 premiers shells, expliquer pourquoi***

  Shell 1 :

  2 pods en cours : code-server qui est en cours d'exécution (Running) depuis 6 jours et express qui a rencontré une erreur lors du pull de l'image (ErrImageNeverPull) et n'est pas en cours d'exécution.

  Shell 2 :

  Le HPA (Horizontal Pod Autoscaler) surveille le déploiement nommé express.
  Les cibles de consommation de ressources (CPU et mémoire) pour le scaling automatique sont définies à 80% pour le CPU et 70% pour la mémoire.
  Cependant, les métriques réelles ne sont pas disponibles (<unknown>) pour les deux critères, ce qui signifie que le HPA ne peut pas actuellement évaluer la situation et prendre des décisions de scaling en fonction de ces critères.

- ***Combien de temps le HPA met-il a scale-up un nouveau pod***

  Dans cette configuration, lorsqu'une augmentation est nécessaire, le HPA ajoutera 1 pod toutes les 60 secondes (periodSeconds: 60).

  Pour le scaling up, le HPA ajoutera un nouveau pod toutes les 60 secondes s'il détermine qu'il est nécessaire d'augmenter la capacité en fonction des métriques de CPU et de mémoire.

- ***Combien de temps le HPA met-il a scale-down un pod***

Pour la réduction, le HPA réduira jusqu'à 10% des pods actuels toutes les 60 secondes (periodSeconds: 60).

Pour le scaling down, le HPA réduira jusqu'à 10% des pods actuels toutes les 60 secondes s'il détermine qu'il y a une sous-utilisation des ressources en fonction des métriques.