Projet Kubernetes
Déploiement et orchestration de l'application Dockercoins

Brahim Hamdi

Générer les bitcoins

> Le minage c'est le procédé par lequel les transactions Bitcoin sont sécurisées. A cette fin les mineurs effectuent avec leur matériel informatique des calculs mathématiques pour le réseau Bitcoin. Comme récompense pour leurs services, ils collectent les bitcoins nouvellement créés ainsi que les frais des transactions qu'ils confirment.
>
> Les mineurs (ou les coopératives de mineurs) sont en concurrence et leurs revenus sont proportionnels à la puissance de calcul déployée.
>
> — minage de bitcoins, https://bitcoin.fr/minage/

Dockercoins

Notre application exemple est **Dockercoins**, une application de minage de Dockercoins! (simulation de bitcoins).
Toutes les images de cette application (fictive et complètement inutile) sont disponibles sur DockerHub.

Elle est composée de 5 composants :

- **rng** = un service web (Python) générant en sortie des nombres aléatoires
  image docker : ***brahimhamdi/rng:2.0***

- **hasher** = un service web(Ruby) générant en sortie un hash des données qui lui sont envoyées par HTTP POST
  image docker : ***brahimhamdi/hasher:2.0***

- **worker** = processus(Python) utilisant rng et hasher
  image docker : ***brahimhamdi/worker:2.0***

- **webui** = web interface (JS)
  image docker : ***brahimhamdi/webui:2.0***

- **redis** = base de données NoSQL
  image docker : ***redis:latest***

Principe

- worker demande à rng de lui fournir des données aléatoires
- worker injecte ces données dans hasher , hasher génére un hash, récupéré par worker,
- Pour chaque hash commençant par 0, worker génére un DockerCoin
- Le worker stocke les DockerCoins générés dans une base de données Redis,
- La webui affiche les données ajoutées à la base Redis en temps réel.

Chaque service est appelé par un nom DNS simple (hasher, rng, etc..). Ce sera notre rôle de démarrer ces différents services avec ce nom pour que le code soit valide.

Travail demandé

- Créez un nouveau Namespace nommé « dockercoins », vous allez appliquer tout le travail dans ce namspace.

- En se basant sur le principe de fonctionnement de l'application décrit ci-dessus, créez le manifest yaml permettant de déployer cette application.
  - Vérifiez le fonctionnement de l'application via son interface web (en utilisant le navigateur web). Quel est le taux de hachage .

- Pour la base de données redis, attachez un volume au dossier « /data ».

- Réservez les ressources suivantes à hasher :
  - CPU : 0.1
  - Mémoire : 300 Mi
  La limite de chaque ressource est deux fois la valeur réservée.

- Installez le serveur des métrique en appliquant le fichier `components.yaml` ci-joint
  - Le serveur a t-il bien démarré ? Sur quel node ?
  - Configurer et appliquez l'autoscaling du hasher en se basant sur les paramètres suivants :
    - scale out si la charge moyenne du CPU dépasse 50%
    - Nombre min de replicas : 1
    - Nombre max de replicas: 5

- Augmentez le nombre de workers à 20, en tolérant la création des replicas sur le node Control Plane.
  - En utilisant la commande 'watch', observez l'autoscaling de hasher au fur et au mesure de création des réplicas worker
  - Y a t-il des workers crées sur le node de Control Plane
  - Quel est le taux de hachage ?

- Créez une Network Policy avec les règles suivantes :
  - Autoriser uniquement les pods worker à accèder à chacun des pods rng, hasher et redis.
  - Autoriser les pods worker et webui à accéder à la base de données redis.