Formation Kubernetes

Ihab ABADI / UTOPIOS

SOMMAIRE

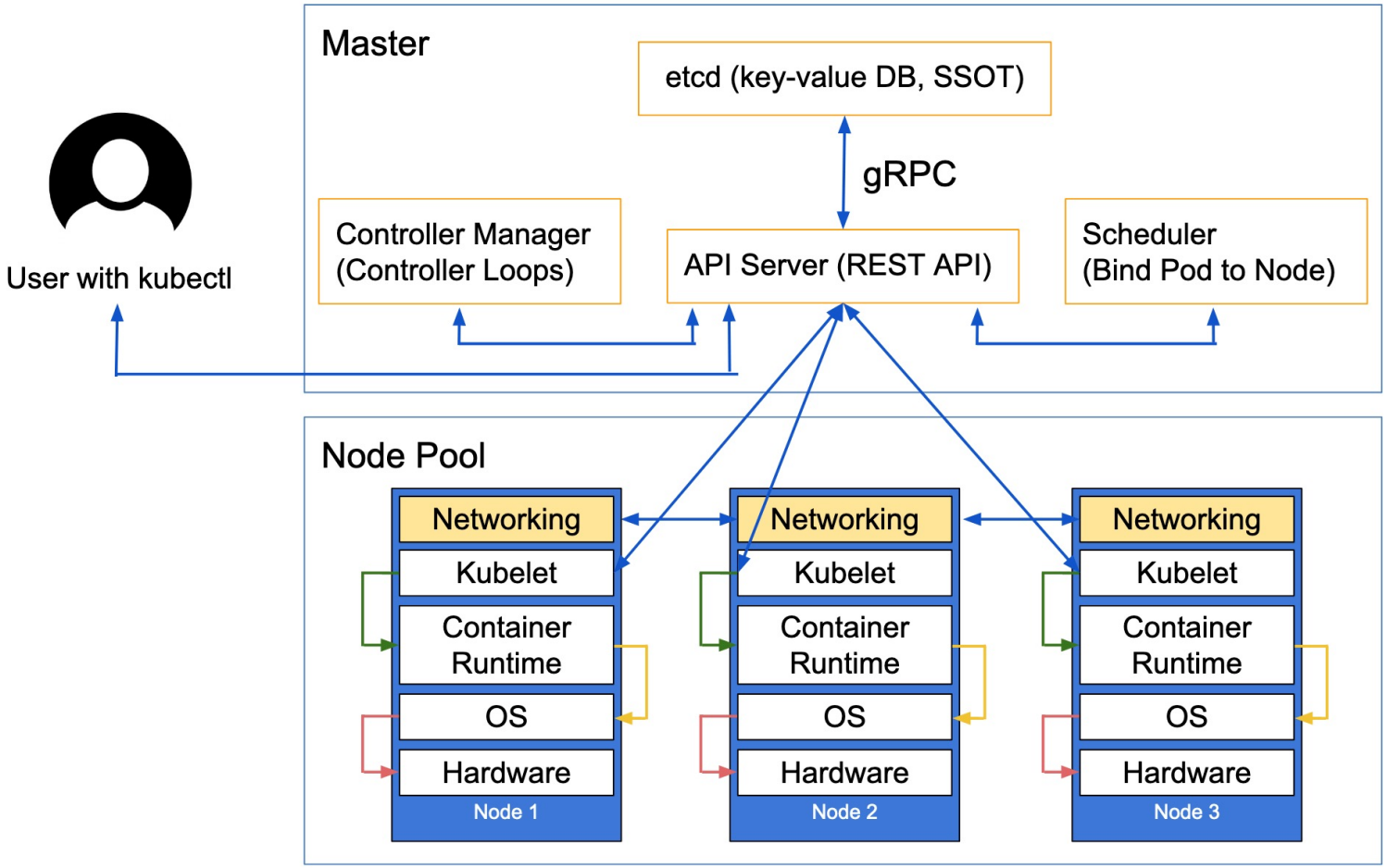
Introduction Kubernetes

- Qu’est-ce que Kubernetes?
- Un système mature et robuste d’orchestration de conteneur Créé par Google, basé sur Borg et Omega, les systèmes qui fonctionnent aujourd’hui chez Google et dont la robustesse est éprouvée depuis plus de 10 ans.
- Google exécute 2 milliards de conteneurs par semaine avec ces systèmes.
- Créé par trois employés de Google initialement pendant l'été 2014.

Fonctionnement de Kubernetes

- Abstraction de l'infrastructure matérielle grâce au concept de "Node"
- Principe: Gérez vos applications comme un troupeau (généricité, opérations de masse) plutôt que comme des animaux de compagnie (chaque opération est personnalisée avec soin et amour pour l'individu).
- Kubernetes est le Linux pour les systèmes distribués:
  - Linux (OS) résout les différences matérielles (avec différents types de CPU, etc.)
  - Kubernetes abstrait les milliers de nœuds d'un cluster et fournit des méthodes industrielles pour gérer les applications.
- L'administrateur décrit et déclare l'"état souhaité", et Kubernetes convertit l'"état courant" en l'"état souhaité".

Architecture de Kubernetes



Les concepts de base de Kubernetes

Pod: Le bloc de base de Kubernetes, atomiquement planifiable, représente une instance unique d'uneapplication dans Kubernetes. Chaque Pod a sa propre IP interne et unique. Les Pods sont mortels.

Deployment: Inclut un modèle de Pod et un ensemble de réplicas. Kubernetes s'assurera que l'étatcourant (nombre de réplicas, modèle de Pod) correspond toujours à l'état souhaité. La mise à jour d'undéploiement déclenche une "rolling update".

Service: Sélectionne les Pods à l'aide d'étiquettes (labels) et fournit un moyen stable et immortel decommuniquer avec votre application en utilisant une IP interne ou un nom DNS.

Namespace: Méthode d'isolation logique, la plupart des objets kubernetes ont une portée relative à unseul Namespace. Vous pouvez y regrouper des applications et leur appliquer vos différentes stratégiesde gestion.

Quelque fonctionnalités de Kubernetes

Self-healing: redémarre les conteneurs qui échouent, remplace et re-planifie les conteneurs lorsque les nœuds meurent, tue les conteneurs qui ne répondent pas au contrôle d'intégrité défini par l'utilisateur et les publie auprès des clients seulement lorsqu'il sont prêts

Automatic binpacking: place automatiquement les conteneurs en fonction de leurs besoins en ressources et d'autres contraintes, sans sacrifier la disponibilité. Co-localisez les charges de travail critiques et "best-effort" afin de maximiser l'utilisation des ressources

Horizontal scaling and autoscaling: Faites passez votre application à l'échelle à l'aide d'une simple commande, d'une interface graphique, ou automatiquement en fonction de l'utilisation du processeur ou de métriques personnalisées.

Automated rollouts and rollbacks: Kubernetes déploie progressivement les modifications apportées à votre application ou à sa configuration, tout en surveillant l'intégrité de l'application afin de s'assurer qu'elle ne tue pas toutes vos instances en même temps. En cas de problème, Kubernetes annulera les changements pour vous.

Quelque fonctionnalités de Kubernetes

Service Discovery and Load Balancing: inutile de modifier votre application pour utiliser un mécanisme de découverte de service tiers. Kubernetes donne aux conteneurs leurs propres adresses IP et un seul nom DNS pour un ensemble deconteneurs, et peut équilibrer la charge entre eux.

Secret and configuration management: Déployez et mettez à jour les secrets et la configuration de l'application sans reconstruire votre image et sans exposer les secrets les fichiers de configuration de votre déploiement.

Storage Orchestration: monte automatiquement le système de stockage de votre choix, qu'il s'agisse du stockage local, d'un fournisseur de cloud public tel que GCP ou AWS, ou d'un système de stockage réseau tel que NFS, iSCSI, Gluster,Ceph, Cinder ou Flocker

Batch Execution: En plus des services, Kubernetes peut gérer vos batch et votre CI, en remplaçant les conteneurs quiéchouent, si vous le souhaitez.

Démarrage Avec Kubernetes

Play with Kubernetes: fonctionne instantanément dans votre navigateur
Créez un cluster sur votre portable ou votre poste de travail avec minikube
Créez un cluster en 2 lignes de commande avec kubeadm
Créer un cluster de production sur AWS avec kops
Créer un cluster de production sur GCE avec GKE (Google Container Engine)
kubicorn, Juju: d'autres méthodes visant à simplifier l'installation

Gestion des pods dans Kubernetes

Un Pod est un groupe d'un ou plusieurs conteneurs, avec un stockage et un réseau partagé.
Par conséquence, ces conteneurs communiquent plus efficacement entre eux et assurent une localisation de la donnée.
Il existe deux types de Pods :
Pod à conteneur simple.
Pod à conteneur multiple.

Gestion des pods dans Kubernetes

Création d’un pod:
Kubernetes utilise un mécanisme de templates en yaml pour décrire les différents types objets
Chaque objet kubernetes doit définir les paramètre suivants :
apiVersion : la version de l'API Kubernetes que vous utilisez pour créer cet objet .
kind : le type d'objet k8s que vous comptez créer.
metadata : données de type clé valeur permettant d’identifier de manière unique l’objet.
spec : contient la spécification avec des champs imbriqués propres à chaque objet k8S.
Le format est donc différent pour chaque objet Kubernetes

Gestion des pods dans Kubernetes

Création d’un pod à conteneur unique:

Pour créer notre premier pod, on peut utiliser la description suivante

On peut utiliser la commande

kubectl create -f nginx-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    type: web
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
```

Gestion des pods dans Kubernetes

Pour tester notre pod il faut se connecter au nœud sur le quel est hébergé notre pod.
l'ip du Pod n'est accessible qu'à partir du notre nœud sur lequel il est hébergé

Gestion des pods dans Kubernetes

Création d'un pod à conteneur multiple:

Pour créer notre premier pod à conteneur multiple, on peut utiliser la description suivante

On peut utiliser la commande

kubectl create -f nginx-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: multic
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
  - name: alpine
    image: alpine
    volumeMounts:
    - name: html
      mountPath: /html
    command: ["/bin/sh", "-c"]
    args:
      - date >> /html/index.html;
        while true; do
          sleep 1;
        done

  volumes:
  - name: html
    hostPath:
      path: /data
      type: DirectoryOrCreate
```

Exercice Pod

Créer un pod avec deux conteneurs qui peuvent communiquer entre eux.
1er conteneur : Application web
2ème conteneur : Base de données relationnelles.

Gestion des ReplicaSets

Les ReplicaSets permettent de vérifier que le nombre de Pods souhaités est bien disponible.
La création et manipulation des ReplicaSets se fait à l'aide de description en yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx
  labels:
    app: web-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web-pod
  template:
    metadata:
      labels:
        app: web-pod
    spec:
      containers:
      - name: nginx
        image: nginx
```

Gestion des ReplicaSets

On peut scaler les pods à l'aide des ReplicaSets soit:

- A l'aide de la commande scale
- A l'aide de la description du ReplicaSet et une mise à à l'aide de la commande apply

Exercice ReplicaSets

- Créer un replicaSet de 3 pods du Pod de l'exercice 1
- Scaler le nombre de pod en augmentant le nombre de pods à 5
- Diminuer le nombre de pod à 2

Gestion des deployments

- Un deployment est une abstraction qui permet à la fois de définir les replicaSets et les Pods
- Un objet de type deployment peut être créer, à la fois, à l'aide d'un document de description ou en cli

Gestion des deployments

- Pour créer un deployment à l'aide du cli, on utilise la commande:
- Kubectl run
- Pour créer un deployment à l'aide d'un document de description on utilise :
- La version de kubernetes apps/v1
- L'attribut kind Deployment
- Exemple

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: node-deployment
  labels:
    app: node
spec:
  replicas: 3
  selector:
    matchLabels:
      app: node
  template:
    metadata:
      labels:
        app: node
    spec:
      containers:
      - name: node
        image: node
        ports:
        - containerPort: 80
```

Gestion des deployments

- Le mécanisme des deployments nous offre la possibilité de modifier la configuration de notre spécification à la volée à l'aide de la commande :
- Kubectl set
- Le mécanisme nous permet également de naviguer dans l'historique des versions et de revenir en arrière à l'aide des commandes
- Kubectl rollout
- Le mécanisme nous permet également de procéder à une mise à l'échelle soit manuelle ou automatique à l'aide de la commande :
- Kubectl scale

Gestion des services

- Kubernetes nous permets de gérer l'accès au différents pods à leurs propres adresses IP
- Il permet également d'équilibrer les charges entre eux.
- Il existe 4 types de services :
- ClusterIP
- NodePort
- LoadBalancer
- ExternalName

Gestion des services

- Pour créer un objet de type service, on peut utiliser :
- Un document de description
- La commande kubectl expose
- Pour créer un un service à l'aide d'un document de description on utilise:
- La version de L'api Kubernetes v1
- Un kind service
- Exemple de service NodePort
- Exemple de service ClusterIP