Engineer

Kubernetes

Système qui maintient un parc de conteneurs en marche

Ce que c’est

Kubernetes, souvent écrit K8s, est un logiciel libre qui exécute des applications conteneurisées sur un groupe de machines, appelé cluster, et les maintient conformes à une description que vous fournissez : l’image utilisée par chaque application, le nombre de copies, les ressources mémoire et processeur que chacune peut utiliser, et la manière dont les autres programmes y accèdent. Sa documentation le définit comme « une plateforme portable, extensible et open source pour gérer les charges de travail et les services conteneurisés ». Le nom signifie timonier ou pilote en grec ; K8s garde le K et le s et remplace les huit lettres intermédiaires par leur nombre ; Google a ouvert le code du projet en 2014.

Un cluster comprend deux parties. Le plan de contrôle conserve la description : le serveur d’API est l’unique interface HTTP par laquelle passe toute modification, etcd stocke ce qu’on lui indique, l’ordonnanceur affecte chaque nouveau Pod à un nœud adapté et le gestionnaire de contrôleurs exécute les boucles qui agissent sur la description. Sur chaque nœud de travail, kubelet s’assure que les Pods affectés fonctionnent, kube-proxy maintient les règles réseau derrière les Services et un moteur d’exécution de conteneurs comme containerd ou CRI-O lance les conteneurs. Docker Engine a cessé d’être un moteur d’exécution intégré dans Kubernetes 1.24, et les images construites avec Docker fonctionnent sans modification, car ce sont des images OCI standard.

La description est un ensemble d’objets, généralement écrits en YAML. Un Pod est la plus petite unité ordonnancée par Kubernetes : un ou plusieurs conteneurs partageant une adresse réseau et du stockage sur une même machine. Un Deployment indique le nombre de Pods identiques qui doivent exister et les remplace progressivement lorsque leur modèle change. Un Service donne un nom et une adresse stables à un ensemble changeant de Pods, dont il choisit les membres par étiquette. Une petite application web ressemble à ceci :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: registry.example.com/web:1.4.2
          resources:
            requests:
              cpu: 250m
              memory: 256Mi
            limits:
              memory: 256Mi
          readinessProbe:
            httpGet:
              path: /healthz
              port: 8080
```

On le compare généralement à trois solutions plus légères. Docker Compose décrit les conteneurs d'une machine ; quand celle-ci disparaît, eux aussi. Un gestionnaire de services comme systemd peut relancer un conteneur planté et le démarrer au lancement de la machine, ce qui couvre de nombreux besoins avec bien moins de mécanismes, mais il ne sait pas déplacer le travail sur une autre machine. Les plateformes hébergées comme Google Cloud Run ou Azure Container Apps exécutent les conteneurs sans vous confier un plan de contrôle à exploiter, en échange de réglages moins nombreux. La documentation est aussi claire sur ce que Kubernetes n'est pas : une plateforme en tant que service traditionnelle et tout compris. Il ne construit pas votre application et ne fournit ni base de données ni file de messages en propre.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Kubernetes ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Un logiciel conditionné pour se comporter à l'identique sur un portable, un serveur de test et en production, et monter en charge lors des semaines chargées.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 20 applications Docker sur Google Cloud et AWS — 20 applications conteneurisées déployées et surveillées

20 applications Docker sur Google Cloud et AWS Contactez-nous →

Pourquoi c’est important

L’idée centrale est que le cluster agit continuellement pour se conformer à une description écrite plutôt que d’exécuter une liste de commandes une seule fois. Appliquer le fichier ci-dessus ne lance rien directement : cela enregistre un souhait. Les contrôleurs sont des boucles de contrôle qui observent le cluster et agissent pour rapprocher son état courant de l’état voulu. Si une machine tombe et que trois copies deviennent deux, un contrôleur voit l’écart, un nouveau Pod est créé et l’ordonnanceur le place sur une machine ayant de la place. Puisque la description est un ensemble de fichiers, elle peut vivre dans un dépôt et subir la même revue que le code exécuté.

Les déploiements de versions font partie de la plateforme. La stratégie prédéfinie est une mise à jour progressive : de nouveaux Pods démarrent et les anciens s’arrêtent à mesure que les nouveaux sont prêts, dans des limites fixées, sans autre réglage, à 25 % de Pods supplémentaires et 25 % indisponibles à tout instant. Chaque modification du modèle de Pod est conservée comme révision — dix sans autre réglage —, et `kubectl rollout undo` revient à la précédente. Le script qui retirait le trafic d’un serveur, remplaçait un binaire et remettait le serveur en service devient un comportement intégré.

L’ordonnanceur change aussi l’économie du parc. Il place chaque Pod selon le processeur et la mémoire demandés : de nombreux petits services partagent donc un groupe de machines au lieu de recevoir chacun un serveur dimensionné pour son pire jour.

Le prix est réel. Un cluster est lui-même un système distribué, avec des versions à mettre à jour selon un calendrier, une couche réseau, une couche stockage, des certificats et du contrôle d’accès : quelqu’un doit comprendre tout cela en cas de dysfonctionnement. Les offres gérées comme Google Kubernetes Engine, Amazon EKS et Azure Kubernetes Service exploitent le plan de contrôle pour vous, mais chaque manifeste et généralement les nœuds de travail restent votre responsabilité. Pour quelques services sur une ou deux machines, un simple gestionnaire de services fait souvent l’affaire avec moins de composants à gérer.

À quoi cela ressemble en pratique

La page des technologies de l'entreprise énonce clairement la règle : Docker et Kubernetes pour les charges qui les justifient, et de simples unités systemd pour les autres. Kubernetes est choisi par charge, selon la taille et la forme du travail, pas systématiquement.

Une mission montre des charges qui le justifiaient. Vingt applications Docker ont été déployées et maintenues sur Google Cloud et AWS, dont des applications d'analyse en R, avec une configuration et une supervision cohérentes pour qu'aucune ne devienne un cas particulier. Le dépannage passait par Podman et le débogage de l'orchestration par Kubernetes. La page rapporte des diagnostics plus rapides et des déploiements restés stables et reproductibles.

L'autre moitié de la règle se voit sur la plateforme propre à l'entreprise. Une couche de conteneurs construite sur Podman et Quadlet exécute chaque conteneur comme un service du superviseur du système d'exploitation lui-même, décrit sous la même forme déclarative que le reste de l'hôte. Les conteneurs utilisent le réseau de l'hôte et ne publient aucun port : les règles de pare-feu installées par le rôle de durcissement continuent donc de régir le trafic. Là où Kubernetes déploierait une image par son API, un utilisateur de déploiement sans privilèges root exécute un unique script fixe qui construit l'image et redémarre le service. Sur un hôte unique, cela donne à chaque conteneur une description écrite et le superviseur auquel l'hôte fait déjà confiance, sans second gestionnaire de processus. Ce qui manque est le placement entre machines et le remplacement d'un nœud mort : c'est là que Kubernetes commence à justifier son poids.

Où les problèmes surviennent

La plupart des problèmes de Kubernetes viennent de l'écart entre ce qu'un manifeste demande et ce que le cluster rapporte.

Un déploiement reste à moitié terminé alors que la tâche de déploiement annonce un succès. Quand les nouveaux Pods ne deviennent jamais prêts, à cause d'un mauvais tag d'image, d'un plantage au démarrage, d'une sonde de disponibilité en échec ou d'un manque de quota, le Deployment cesse de progresser. Une fois le délai de progression écoulé — 600 secondes sans autre configuration —, il enregistre la raison `ProgressDeadlineExceeded`, et la documentation indique que Kubernetes ne fait rien d'autre : les anciens Pods

encore en exécution continuent de servir, et aucun retour arrière ne se produit seul. `kubectl rollout status` se termine avec un code non nul après le délai : une étape de déploiement peut donc l'attendre et annuler le déploiement :

```
kubectl apply -f web.yaml
kubectl rollout status deployment/web --timeout=10m \
  || kubectl rollout undo deployment/web
```

Un conteneur redémarre avec `OOMKilled`, ou ralentit sur un nœud où il reste de la capacité processeur. Le noyau impose les limites, et les deux ressources se comportent différemment. Un conteneur au-delà de sa limite mémoire peut être arrêté puis redémarré ; à sa limite processeur, il est bridé et ralentit sans mourir, ce qui ressemble à un problème d'application. L'ordonnanceur place les Pods selon leurs demandes, pas selon leur consommation : des demandes faibles entassent les charges sur un nœud, des demandes élevées laissent des machines inactives. Une limite donnée sans demande devient la demande. Réglez la mémoire selon l'usage mesuré et considérez les limites processeur comme un choix délibéré.

Une application au démarrage lent est arrêtée avant de finir de démarrer. Une sonde de vie en échec conduit kubelet à arrêter puis relancer le conteneur : une application qui nécessite une minute de chargement, mais est contrôlée après dix secondes, redémarre sans fin. Une sonde de démarrage retarde les autres contrôles jusqu'à ce que l'application fonctionne. Une sonde de disponibilité retire un Pod de son Service sans le redémarrer, ce qui convient à un Pod occupé ou qui réchauffe un cache. Une sonde de vie qui appelle la base de données redémarre toutes les répliques à la fois quand la base ralentit, transformant une dépendance lente en panne.

Un Pod lancé à la main disparaît, et une correction manuelle est annulée. Un Pod créé sans contrôleur reste sur son nœud jusqu'à sa fin, sa suppression, son éviction ou la défaillance du nœud, et aucun contrôleur ne le recrée ; la documentation conseille un Deployment ou un Job même pour une seule copie. Une modification faite avec `kubectl edit` ne vit que dans le cluster, et l'application suivante depuis le dépôt rétablit l'ancienne valeur : une correction urgente doit donc rejoindre les fichiers dès la fin de l'urgence.

Pour aller plus loin

- Vue d’ensemble de Kubernetes — la présentation du projet lui-même : ce qu’est Kubernetes, ce qu’il offre et ce qu’il laisse de côté.
- Pods — la plus petite unité ordonnancée par Kubernetes, son cycle de vie et les raisons pour lesquelles on crée rarement les Pods à la main.
- Deployments — mises à jour progressives, retours arrière, historique des révisions et délai de progression.
- Service — noms et adresses stables devant un ensemble de Pods changeant.
- Gestion des ressources des Pods et conteneurs — demandes, limites et application de chacune par le noyau.
- Configurer les sondes de vie, de disponibilité et de démarrage — les trois sondes et la protection d’un conteneur au démarrage lent.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que Kubernetes ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/kubernetes/