Chapitre 3
Tableau de bord et ligne de commande

1. Objectifs du chapitre et prérequis

Ce chapitre fera tout d'abord un petit historique sur l'origine du projet et sur les notions autour des conteneurs.

La suite sera consacrée à une introduction sur l'utilisation de Kubernetes et la découverte de quelques éléments :

– le tableau de bord (dashboard) Kubernetes ;

– la commande kubectl ;

– le moteur Docker de Minikube ;

– la consultation de quelques objets de base (pods et nodes).

À la fin de ce chapitre, vous disposerez des éléments suivants :

– un cluster Kubernetes fonctionnel ;

– un contexte kubectl cohérent.

Afin de suivre l'ensemble des exercices, il est indispensable de disposer d'une connexion Internet haut débit (ADSL ou supérieure).

2. Préambule

2.1 Origine du nom et du logo

Kubernetes est un mot d'origine grec (κυβερνήτης) qui signifie timonier (ou pilote). En navigation, c'est la personne qui est en charge de tenir la barre du gouvernail.

Chez Google, le nom interne de ce projet était Project Seven, en référence à un personnage de l'univers Star Trek (Seven of Nine ou de son vrai nom Annika Hansen). Les sept rayons du logo en forme de barre de gouvernail sont un clin d'œil à cet ancien nom.

2.2 Pourquoi utiliser Kubernetes ?

La sortie de Docker ces dernières années a constitué une rupture importante dans la manière de gérer les applications au sein d'un système d'information. Il devenait ainsi possible de réaliser des opérations auparavant compliquées de façon plus rapide et simple, comme par exemple :

– démarrer des environnements de tests à la demande ;
– s'affranchir des problèmes de dépendances de librairies ;
– uniformiser les livrables de tous les environnements ;
– isoler de manière très poussée les ressources sur un même serveur.

Comme tout nouvel outil, certaines questions n'avaient pas forcément de solution simple et introduisaient de nouveaux problèmes. Parmi ces problèmes, on retrouve les points suivants :

– persistance des données ;
– surveillance des applications dans les conteneurs ;
– mise à jour automatique des applications ;
– scalabilité du moteur Docker ;

– scalabilité de la charge applicative ;
– publication des environnements sur l'extérieur.

C'est entre autres pour ces raisons que Kubernetes a été introduit en 2014 afin de répondre aux problèmes de passage à l'échelle de Docker.

2.3 Origine de Kubernetes

Les premières versions de Kubernetes proviennent de Google. La société derrière le moteur de recherche travaille depuis plus de dix ans sur les notions de conteneurisation. Ces travaux ont amené Google à reverser à la communauté de très nombreuses contributions sur le noyau Linux (Google fait partie des cinq plus gros contributeurs).

Une partie de ces travaux a d'ailleurs aidé à l'émergence de Docker, comme par exemple les Cgroups. Ce mécanisme – à l'origine développé par des ingénieurs de chez Google – permet de mettre en place une limitation de la consommation CPU et/ou mémoire et également de l'isolation de process.

Chez Google, ces ancêtres des conteneurs contemporains étaient pilotés en interne par un produit maison : Borg. Ce produit est toujours en activité chez Google et gère quotidiennement des milliers de machines dans le monde.

À noter qu'une partie des développeurs de Kubernetes sont d'anciens contributeurs sur le projet Borg. Ces derniers ont repris une grande partie des principes qui étaient présents dans Borg pour les appliquer dans Kubernetes.

2.4 Fondation CNCF

Le code du produit a été donné par Google à la Linux Foundation, ce qui a servi à la création de la CNCF (*Cloud Native Computing Foundation*). Cette organisation est derrière la promotion des produits open source dans le cloud comme Prometheus, OpenTracing ou Containerd.

L'idée de Kubernetes est de garder ce qui fonctionne bien dans Docker (le principe des conteneurs Dockers) et d'étendre le reste avec une couche d'API ainsi qu'un mécanisme d'extension et d'orchestration.

2.5 Les orchestrateurs du marché

La société à l'origine de Docker a développé son propre moteur d'orchestration : Docker Swarm. Ce livre n'en parlera pas, mais ce produit répond lui aussi à certains problèmes évoqués plus haut. À noter que les dernières évolutions de cet outil permettent de communiquer avec Kubernetes.

En plus de Docker Swarm, il existe d'autres orchestrateurs comme par exemple Nomad ou Mesos qui utilisent des mécanismes totalement différents de ceux de Kubernetes.

Remarque

L'intégration Docker Shim qui permet de faire dialoguer Kubernetes avec Docker est difficile à maintenir alors qu'il est possible de communiquer directement avec le moteur de conteneur sous-jacent (Containerd). La dépréciation vient directement de cette raison.

Il existe également des surcouches à Kubernetes apportant des fonctionnalités ou un support commercial. On retrouve notamment les produits suivants :

– Openshift chez RedHat ;
– Tectonic chez CoreOS ;
– Rancher chez Rancher Labs.

Ces produits sont tous basés sur une version de Kubernetes sur lequel l'éditeur va apporter les services suivants :

– préconfigurer des services prêts à l'emploi ;
– figer un ensemble de versions afin d'offrir un support longue durée.

Heureusement, ces différences sont généralement marginales. Elles ne devraient pas vous poser de problème vis-à-vis des instructions qui se trouvent dans ce livre.

3. Le tableau de bord de Kubernetes (dashboard)

3.1 Présentation

Le dashboard de Kubernetes est un moyen de consulter l'état des différents éléments d'un cluster. Pour l'utilisateur débutant sur cette technologie, il s'agit d'une bonne porte d'entrée puisque cette application permet de consulter graphiquement les différents constituants d'un cluster Kubernetes.

Néanmoins, si ce dernier peut être toléré sur une machine Minikube, il n'est pas forcément souhaitable de le déployer sur les plateformes de production pour des raisons de sécurité.

3.2 Tableau de bord Kubernetes sur service managé

Dans le cas où vous feriez appel à un service managé, il sera peut-être nécessaire de procéder à l'installation du dashboard avant de continuer.

Consultez la section Déploiement du tableau de bord Kubernetes dans les annexes afin de réaliser l'installation de ce dernier.

3.3 Déploiement du dashboard sur Minikube

Sous Minikube, l'activation du dashboard se fait à l'aide de l'instruction `minikube` suivie des options suivantes :

– l'option `addons` suivie du mot-clé `enable` ;

– le nom du plugin à activer (ici `dashboard`).

Ci-dessous la commande à lancer :

```
$ minikube addons enable dashboard
```

Ci-après le résultat de cette commande :

```
    Using image kubernetesui/dashboard:v2.3.1
    Using image kubernetesui/metrics-scraper:v1.0.7
    Some dashboard features require the metrics-server addon. To
enable all features please run:
        minikube addons enable metrics-server


    The 'dashboard' addon is enabled
```

Comme indiqué par la commande précédente, activez le serveur de métriques (`metrics-server`) pour bénéficier de l'ensemble des fonctionnalités du tableau de bord :

```
$ minikube addons enable metrics-server
```

Ci-dessous les messages renvoyés par cette opération :

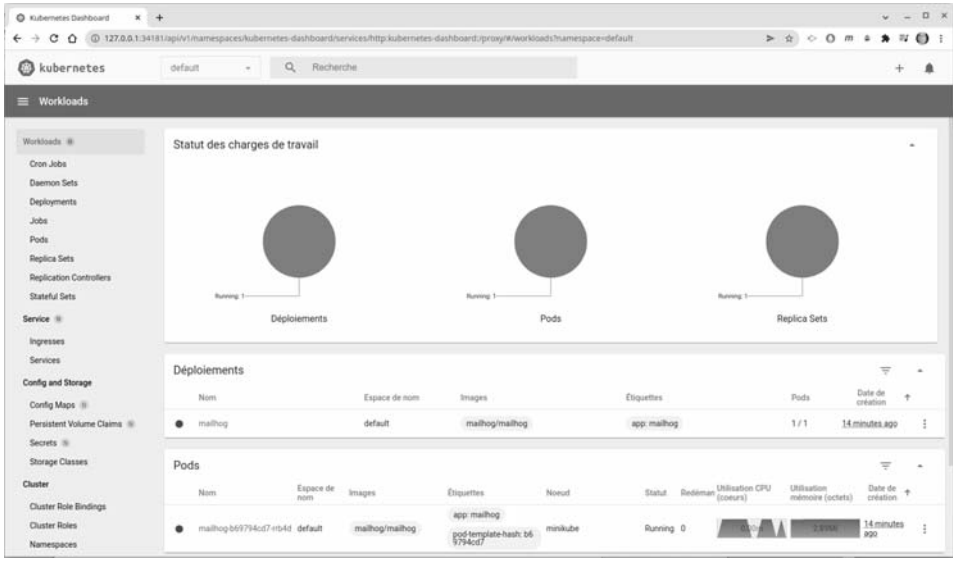
```
    Using image k8s.gcr.io/metrics-server/metrics-server:v0.4.2
    The 'metrics-server' addon is enabled
```

3.4 Accès au dashboard sur Minikube

L'accès au tableau de bord peut se faire avec la commande `minikube` suivie de l'option `dashboard`.

Ci-dessous la commande correspondante :

```
$ minikube dashboard
```



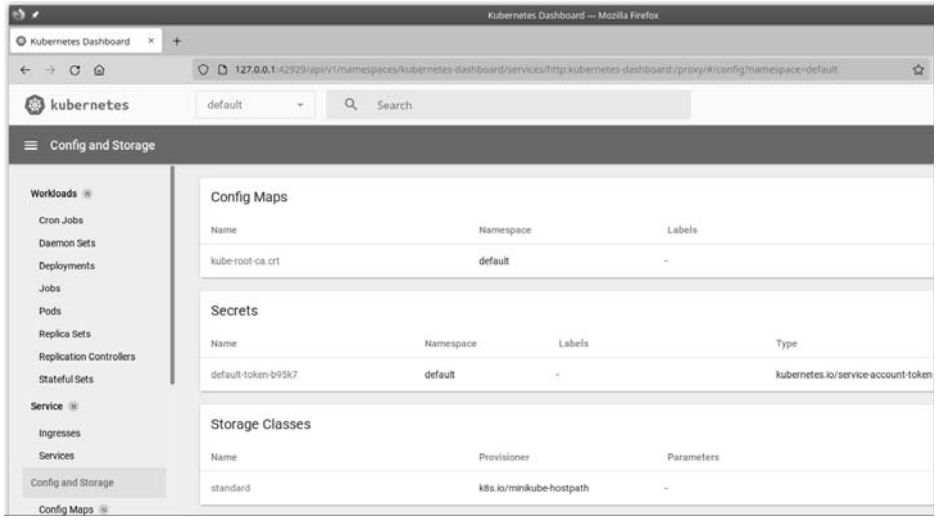
Page d'accueil du tableau de bord de Kubernetes

La commande s'occupera alors de publier le dashboard dans un tunnel sécurisé et de lancer le navigateur par défaut.

3.5 Structure du tableau de bord

La partie de gauche contient des liens vers les éléments suivants :

- les éléments déployés (**Workloads**) ;
- les entrées de service interne et externe (**Services**, **Ingresses**) ;
- les éléments de configuration (**Config and Storage**) ;
- les éléments du cluster (**Namespaces**, **Nodes**, **Persistent Volumes**, etc.) ;
- les définitions d'objets personnalisés (**Custom Resource Definitions**).



Vue d'ensemble de l'écran ***Config and Storage***

Remarque

À noter la présence du secret default-token. Combiné avec le service kubernetes de l'espace de noms par défaut, ceux-ci permettront de référencer en interne le point d'entrée de l'API de Kubernetes avec un compte de service par défaut.

3.6 Création d'un déploiement

3.6.1 Un petit mot sur Mailpit

Depuis l'interface de déploiement, il est possible de déployer une première application. Pour les besoins du test, vous déploierez l'application Mailpit. Cette application embarque un serveur de mail SMTP. Une interface web est également présente et permet de consulter les mails reçus au travers du serveur SMTP.

Cette application sert à simuler l'envoi de mails. Du fait de sa simplicité et de sa légèreté, cette application fera une excellente candidate pour aborder les déploiements sous Kubernetes.

Sur Docker Hub, l'image de Mailpit porte le nom de axllent/mailpit.

Remarque

Mailpit trouve son inspiration originale dans un produit similaire : MailHog. Malheureusement, ce dernier n'est plus maintenu : il est préférable de ne plus y faire appel. Pour en apprendre davantage sur les avantages de Mailpit par rapport à MailHog, n'hésitez pas à consulter la page GitHub du projet : https://github.com/axllent/mailpit

3.6.2 Lancement du déploiement

Le dashboard permet le déploiement d'applications. Vous devez pour cela suivre le lien **Deployments** sous le titre **Workloads**.

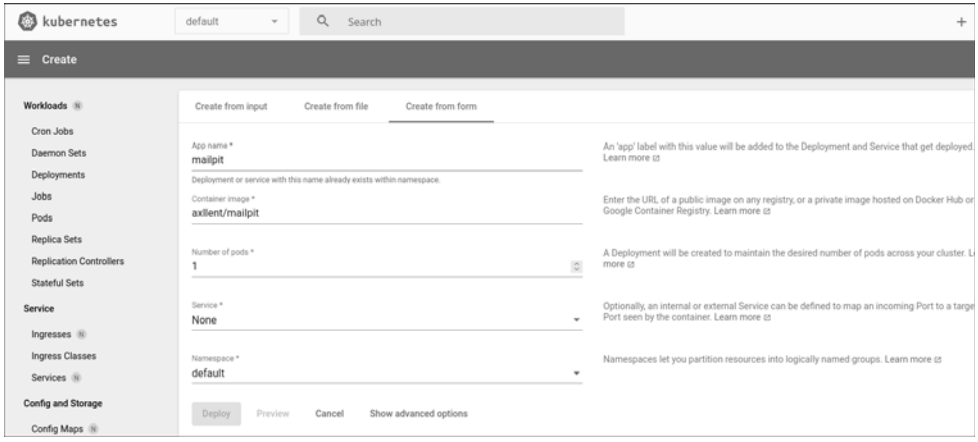
Sur l'écran qui apparaît, cliquez sur le bouton **+** en haut à droite pour créer un nouvel objet. Sur cet écran, cliquez sur le lien **Create from form** puis entrez les informations suivantes :

– le nom du déploiement **(App name)** : mailpit ;

– le nom de l'image (**Container image**) : axllent/mailpit.

Les autres champs seront laissés aux valeurs par défaut.

Vous devez ensuite cliquer sur le bouton **Deploy** pour lancer le déploiement de l'application Mailpit.



L'assistant de création d'applications dans Kubernetes avec les champs préremplis pour la création de l'application Mailpit