sopra steria

S'Breaking News

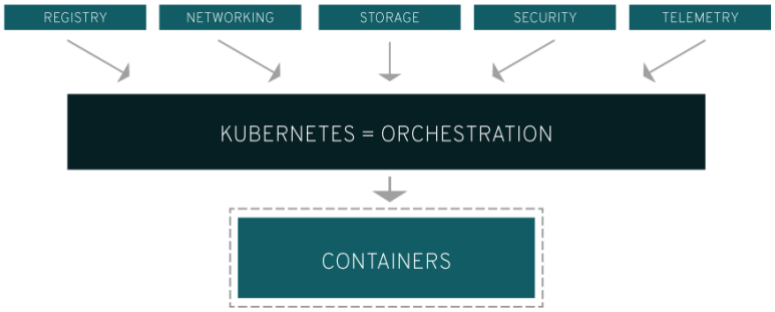
La lettre d'information des bâtisseurs de la défense et de la sécurité de demain

#BuilderException

#112 – 15 mai 2020

Kubernetes, le grand gagnant de la guerre des orchestrateurs

Vous connaissez sans doute déjà, au moins de nom, *Kubernetes* (ou sa version raccourcie *K8s*).
Si toutefois ce n'est pas le cas, on peut définir assez basiquement *Kubernetes* comme étant une **plateforme open source d'orchestration de containers**, qui permet de **supprimer de nombreux processus manuels** associés au déploiement et à la mise à l'échelle d'applications containerisées.



Un peu d'histoire…

Avant d'être confié à la *CNCF* (*Cloud Native Computing Foundation*), il faut rappeler que *Kubernetes* est un **projet à l'initiative de *Google*** et de son savoir-faire en matière de containers (*Google* développe en interne des systèmes de gestions de clusters à grande échelle depuis 2003, avec notamment *Borg*, puis *Omega* quelques années plus tard).
En **2014**, à l'heure où les containers Linux se démocratisent dans le monde professionnel, *Google* lance *Kubernetes*, une **version open-source de *Borg***, plus ouverte et flexible.
L'idée derrière *Kubernetes* est de **faciliter le déploiement et la gestion de systèmes distribués** complexes, tout en bénéficiant des **avantages de la containerisation**, le tout sur une **plateforme qui fait totalement abstraction de son infrastructure**.

Comment ça marche ?

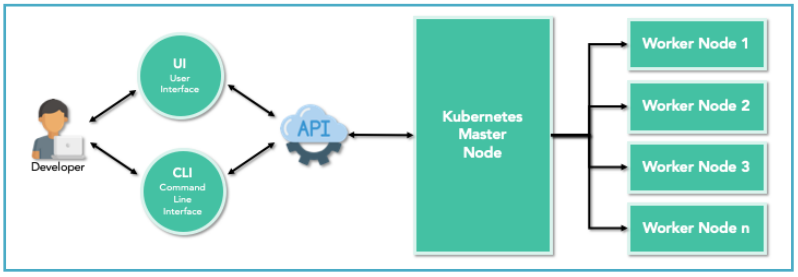
Prenons pour exemple une application simple : un **serveur web containerisé**, qui lorsqu'il reçoit une requête, lit une valeur dans une base de données, puis réponds au client HTTP.
En cas de problème avec ce serveur (par ex. une panne ou une charge élevée), on peut imaginer qu'un système de monitoring alertera l'équipe d'exploitation, qui analysera puis corrigera l'erreur.

Kubernetes propose une approche différente : **l'approche déclarative** (en opposition à l'approche impérative). **L'idée est de décrire l'état dans lequel doit se trouver la plateforme, et le système est programmé pour trouver une solution afin de répondre à ces exigences.**
Dans notre exemple, ceci est notamment rendu possible grâce à deux fonctionnalités intégrées :

- Le **self-healing** : le cluster *Kubernetes* doit trouver le moyen de remettre l'application dans son état nominal, en démarrant le container sur une autre machine par exemple.
- L'**autoscaling** : le cluster *Kubernetes* s'adapte à la charge, en augmentant le nombre de nœuds lorsque le serveur est fortement sollicité, et en le réduisant la nuit par exemple.

Composantes de Kubernetes

A l'instar d'autres systèmes distribués, *Kubernetes* est composé d'un nœud maître, le *Master Node*, et d'un ou plusieurs nœuds esclaves, les *Worker Nodes*.



Macroarchitecture

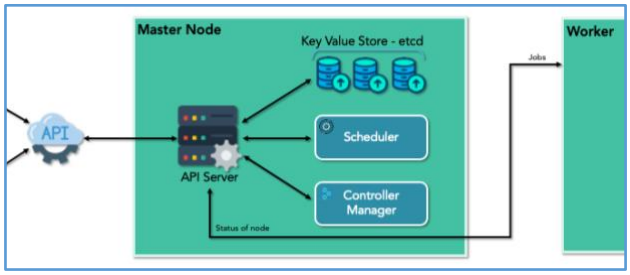
L'utilisateur peut interagir avec *Kubernetes* de deux manières :

- Graphiquement, via une interface utilisateur (UI),
- Depuis un terminal, via une interface en ligne de commande (CLI).

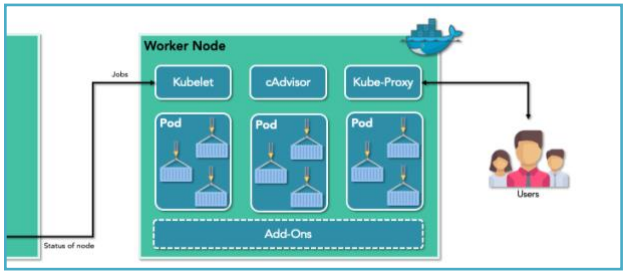
Ces deux entités communiquent ensuite avec *Kubernetes* via une API *REST*, exposée par le *Master Node*.

Le *Master Node* est responsable de l'exposition de l'API avec laquelle interagît l'utilisateur, mais également de la planification des déploiements, et de la gestion du cluster dans son ensemble. Ainsi, il **identifie** et **surveille** l'état des *Worker Nodes*.

Les *Worker Nodes* **exposent** des **ressources de calcul**, **de stockage**, et **de réseau** aux applications. Ils possèdent chacun un environnement d'exécution containerisé (par exemple *Docker* ou *RKT*), et communiquent avec le *Master Node* par le biais d'un agent. Ce sont ces nœuds qui seront utilisés pour faire fonctionner notre application de manière totalement contrôlée.



Master Node



Worker Node

Pourquoi c'est si populaire ?

Face à *Kubernetes*, d'autres solutions d'orchestration de containers, telles que *Docker Swarm,* ou encore *Apache Mesos* ont émergé.
Pourtant, *Kubernetes* domine outrageusement ce marché. D'après une étude menée par la *CNCF*, 78% des entreprises utilisant des containers en production ont adopté *Kubernetes* pour les gérer.

De nombreuses raisons expliquent ce succès, parmi lesquelles :

1. ***Kubernetes* est plus stable que ses concurrents**. Bien qu'encore relativement récent, le projet bénéficie de l'expérience de *Google* en matière d'orchestration de containers.
2. ***Kubernetes* est portable**. Il est possible de l'utiliser dans différents environnements, et de le déplacer d'un environnement à un autre.
3. ***Kubernetes* possède une communauté active et collaborative**, composée aussi bien de développeurs indépendants que de cadors de l'IT à l'image de *Google*, *RedHat* (*IBM*), *Microsoft*, etc. *Kubernetes* compte aujourd'hui parmi les projets open source les plus importants (juste derrière *Linux* et *React*), avec près de **45 000 contributeurs**, pour plus de **226 000 commits**, et **74 000 forks**.

Fun fact

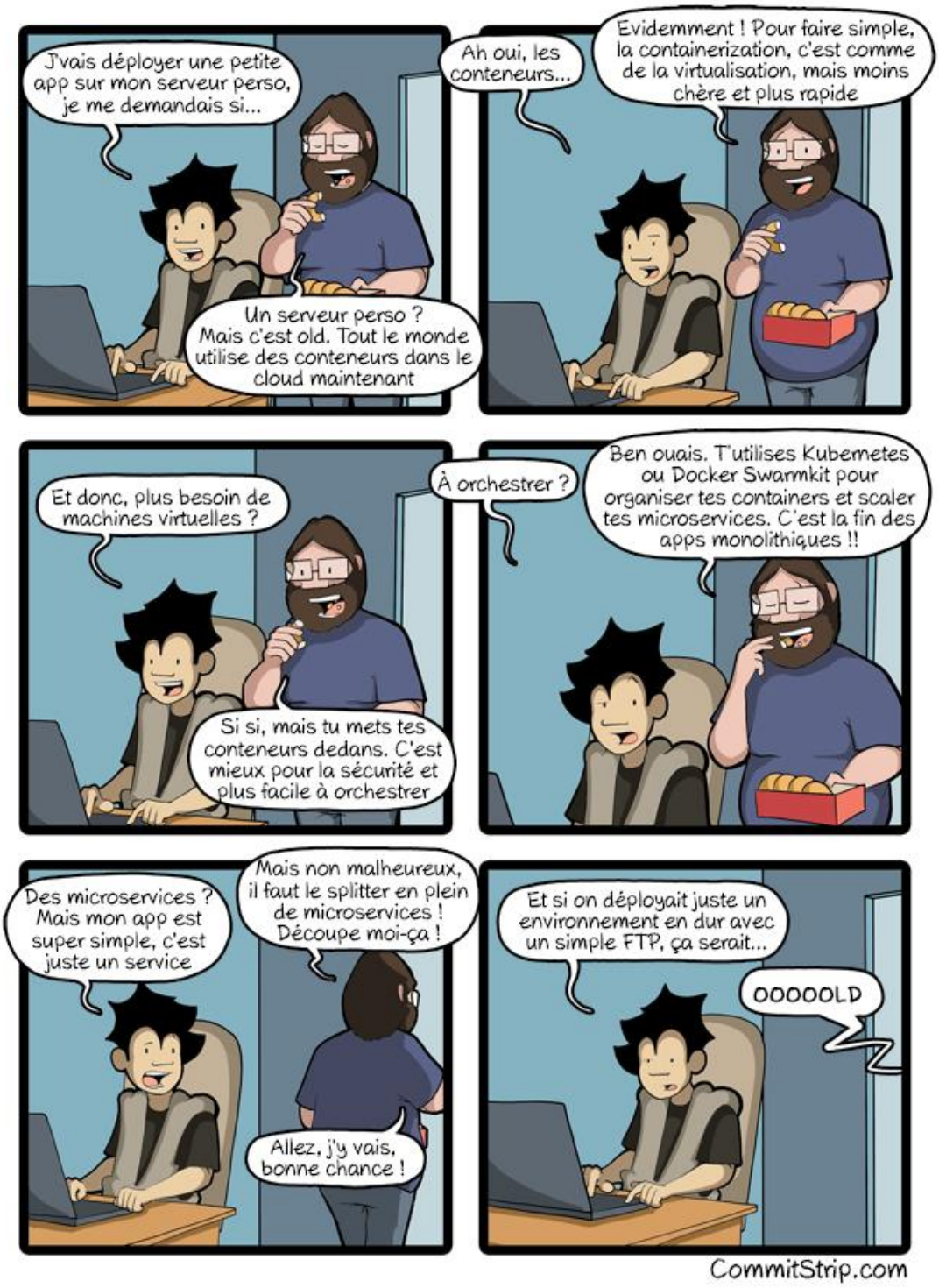
Kubernetes et sa capacité de mise à l'échelle ont rendu possible le succès de l'application *Pokemon Go*. En effet, l'application étant rapidement devenue virale, le trafic constaté s'est révélé être 50 fois supérieur aux estimations effectuées par les équipes de développement.

Quelques liens

Success stories
L'essayer, c'est l'adopter

Clément LEFEVRE

A l'ancienne


J'vais déployer une petite app sur mon serveur perso, je me demandais si...
Un serveur perso ? Mais c'est old. Tout le monde utilise des conteneurs dans le cloud maintenant
Ah oui, les conteneurs...
Evidemment ! Pour faire simple, la containerization, c'est comme de la virtualisation, mais moins chère et plus rapide
Et donc, plus besoin de machines virtuelles ?
Si si, mais tu mets tes conteneurs dedans. C'est mieux pour la sécurité et plus facile à orchestrer
À orchestrer ?
Ben ouais. T'utilises Kubernetes ou Docker Swarmkit pour organiser tes containers et scaler tes microservices. C'est la fin des apps monolithiques !!
Des microservices ? Mais mon app est super simple, c'est juste un service
Mais non malheureux, il faut le splitter en plein de microservices ! Découpe moi-ça !
Allez, j'y vais, bonne chance !
Et si on déployait juste un environnement en dur avec un simple FTP, ça serait...
OOOOOLD
CommitStrip.com

Agenda

Special Event – Dessiné c'est gagné
Tous les vendredis – RDV 13h30 sur le Teams de la Communauté SB

Inspiring Talks – Gitlab CI / CD (partie 2) par Matthieu GUTIERREZ
19 mai 2020

Retrouvez tous vos articles préférés sur le site de la communauté ici.