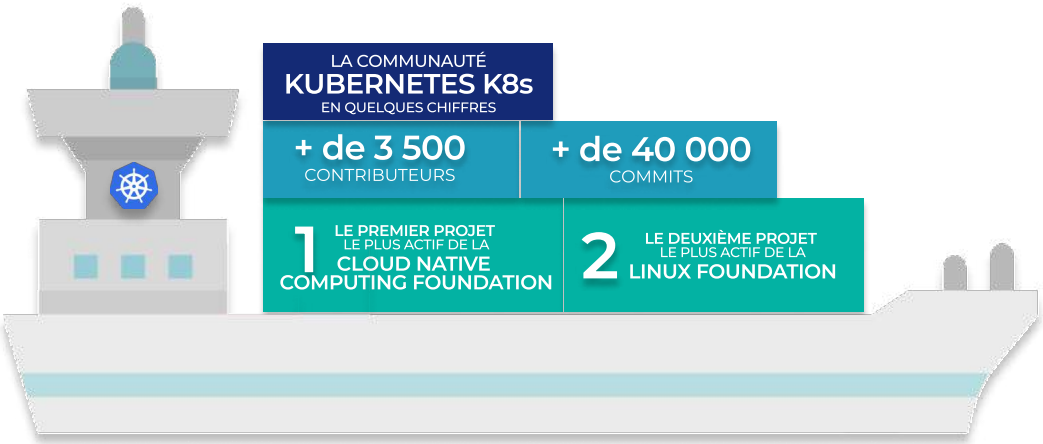
Cloud FinOps

NEWSLETTER
MARS 2024

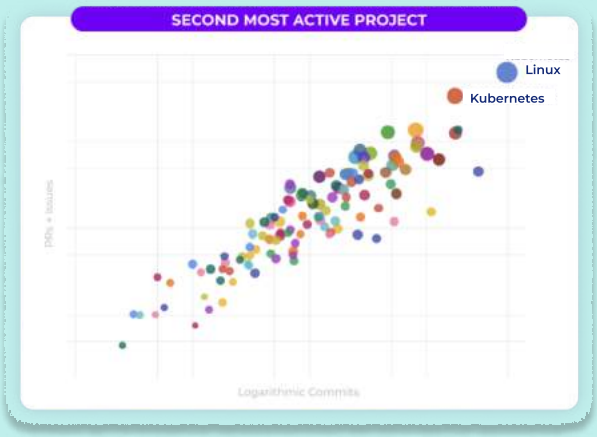
KUBERNETES
En tirer pleinement parti sans se ruiner



Qu'est-ce qui explique un tel succès ?
Quels sont les enjeux auxquels Kubernetes répond ?
Quels sont les défis soulevés ? *en particulier sous l'angle FinOps*
Quels sont les points de vigilance à avoir en tête ?

Voici les questions auxquelles nous allons tenter de répondre à travers cette newsletter.

kubernetes est le **deuxième projet** le plus actif et répandu, **talonnant de peu l'historique Linux.** *(1) et (2)*



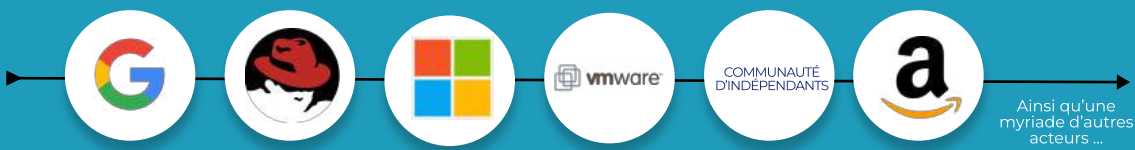
L'évolution de Kubernetes
Son histoire et ses principaux contributeurs.

Annonce de la création de Kubernetes

Lancement officiel de la V1

Succès immédiat avec des contributeurs majeurs qui n'ont cessé d'investir en temps, ressources et améliorations continues.

Les 6 principaux contributeurs sont dans l'ordre :



Google, le concepteur d'origine

Red Hat, IBM

Microsoft

VMware Broadcom

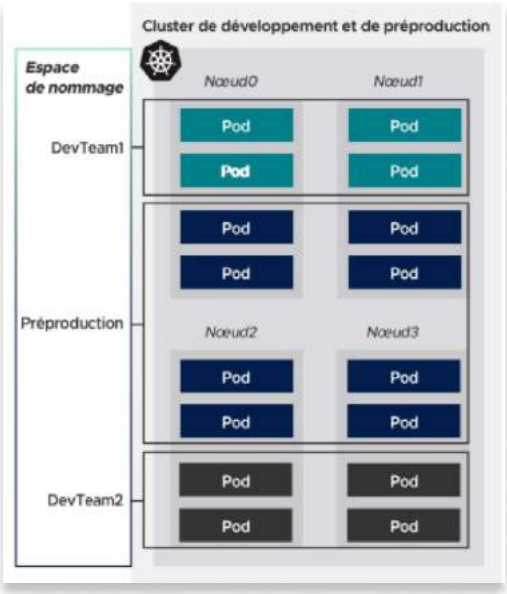
Le cinquième contributeur

Amazon

Qu'est ce que Kubernetes ?
Principe de base et clés du succès

La clé de succès initiale du projet Kubernetes est intimement liée au **succès des conteneurs (containers)** eux-mêmes. Cela tient d'abord au fait que les conteneurs sont un **moyen simple de packager des logiciels.** Les prérequis et paramètres sont intégrés dans une image déployable, permettant **gain de temps, automatisation et très grande agilité (« Scale Up » / « Scale Down »).**

Le challenge est qu'Il est alors nécessaire d'utiliser un **outil d'orchestration** de conteneurs et c'est là que Kubernetes intervient pour permettre aux ingénieurs de **déployer et gérer de manière automatisée et à l'échelle.**



L'image est le modèle d'un **conteneur** avec le programme qui doit être exécuté, le conteneur est l'instance d'une image, **plusieurs copies pouvant être exécutées en même temps,** on parle ensuite **d'instance de serveur (nœud)** pour un serveur cloud, puis **un pod** qui constitue un groupe de conteneurs qu'il considère comme un bloc de ressources unique.

Le cluster est un groupe d'instances de serveurs gérés par l'orchestration de conteneurs, qui gère le cluster d'instances de serveurs et le cycle de vie conteneurs/pods.

Citons les **plus répandus** *(3)* **:**

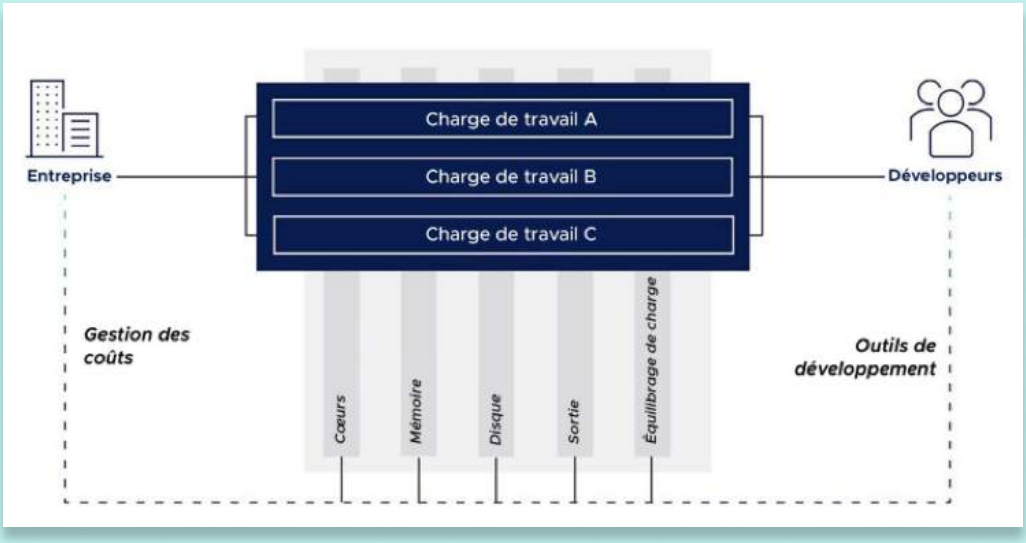
- **Amazon Elastic Kubernetes Service** (EKS)
- **Google Kubernetes Engine** (GKE)
- **Microsoft Azure Kubernetes Service** (AKS)
- **IBM Cloud Kubernetes Service** (IKS)
- **Oracle Container Engine for Kubernetes** (OKE)
- **Alibaba Cloud Container Service for Kubernetes** (ACK)
- ...

Le modèle de facturation des Cloud Providers

Principe de base et challenges FinOps

Le modèle de tarification des Cloud Providers consiste à facturer pour chaque instance de serveur constituant **un cluster**, **lorsque les conteneurs sont déployés en cluster** puisqu'ils consomment des capacités en ressources. **La facturation** est faite **dès qu'un processus est exécuté**, donc le paiement est dû **que la ressource soit utilisée ou non.**

C'est notamment là où intervient vite le sujet de **l'allocation des coûts** (équipes, projets, individus…) selon le conteneur réellement exécuté dans le cluster. De plus, les coûts mutualisés (shared costs) sous-jacents incluant les nœuds de gestion, les données utilisées, les licences software, les sauvegardes et potentiels PRA…qui **doivent être pris en compte dans l'allocation des coûts.**



Par ailleurs ...
Les méthodes FinOps traditionnelles présentent des limites

Si le principe de déclenchement de facturation n'est qu'à partir du moment où les déploiements sont exécutés. Néanmoins pas mal de pièges à éviter...

Applications with Shared Infrastructure and Services (Kubernetes)
Source: Casey Doran, "FinOps Summit: Cost Visibility and Optimization in Kubernetes" (4)

Ainsi, comment repousser ces limites, pour conserver les qualités de nos outils ?

Plus que tout autre discipline dans le FinOps, appliquer les pratiques FinOps au monde des architectures à base de microservices a d'autant plus de sens de par la volumétrie, la complexité et le principe même d'environnements partagés.

INFORMER

De la qualité de la structuration de l'approche va découler **la qualité des informations exploitables** pour optimiser et ce en particulier pour :

- **Allocation des coûts** (hiérarchisation de facturation, ressources, espaces de nommage, libellés)
- **Proportions des containers** (remontées par les équipes du niveau d'utilisation par équipe sur base de déclaratif)
- **Suivi simultané des coûts des ressources sur clusters mutualisés et hors cluster** (Tags, Labels, Namespaces)
- **Coûts statiques** avec persistance (serveurs d'appli, bases de données, CMS..)
- **Coûts d'exécution** (hidden resources, bande passante, coûts de calcul, requête de données...) sont des exemples à bien surveiller
- **Classes de conteneurs** (selon la QoS)
- **Coûts annexes** (coûts opérationnels de gestion des clusters, coûts de stockage, licences logicielles – associées ou en mode BYOL – observabilité, sécurité)

OPTIMISER

Par principe, **la mutualisation des ressources** via **la containerisation** est de nature à aller dans le sens des bonnes pratiques FinOps. Néanmoins, **les axes d'optimisation** sont nombreux et comprennent notamment :

- **L'emplacement du Cluster** (instances Spot, choix de la région....)

- **Optimisation des usages :**
  - **Rightsizing horizontal** (packing des conteneurs au niveau des instances serveurs) : le projet open source HPA (Horizontal Pod Autoscaler) facilite la scalibilité horizontale,
  - **Rightsizing vertical** (redimensionnement des instances serveurs) le projet VPA (Vertical Pod Autoscaler) permet de vous aider à ajuster automatiquement la configuration à la demande,
  - **Workload matching** (vCPU-to-memory ratio)

- **Dimensionnement adapté des noeux**
- **Optimisation du coût du serveur d'instance** en s'appuyant sur les Reserved Instances (RIs) Savings Plans (SPs) et les remises globales

OPERER

A présent qu'allocation plus précise et méthodes d'optimisation ont été mis en œuvre, **comment inscrire cette pratique dans la durée ?** Il est important de comprendre que planifier les arrêts (turn on/off) des conteneurs, rechercher et supprimer les conteneurs inactifs et mieux gérer les tags/labels sont **les bonnes pratiques FinOps à diffuser au sein des équipes.**

Pour aller plus loin :

- **Utiliser les outils cloud natifs** et en particulier les offres Serverless containers des Cloud Providers où une partie de l'allocation des coûts peut être prise en charge, néanmoins ceci est à mettre au regard du coût additionnel du service lui-même.

On parle de :
- Fargate chez AWS
- Azure Container Instances (ACI) chez Azure
- Google Cloud Run chez GCP *(5)*

- **Utiliser des plates-formes FinOps** telles que **CAST AI** : https://cast.ai/ pour le monitoring et l'optimisation 100% dédié K8s ou **Finout** qui permet de détecter le gaspillage Kubernetes dans un écosystème Cloud plus vaste : https://www.finout.io/kubernetes-waste-detection
- **Rendre autonomes et encourager les développeurs** à suivre leur utlilisation et consommation de Kubernetes

Cloud FinOps

Cloud FinOps

cloud-finops.io

SOURCES

- *(1) CNCF Projects (2023-2024)* **(Lien)**
- (2) Linux Foundation Projects (2023-2024) **(Lien)**
- (3) VMware FinOps pour Kubernetes (2021) **(Lien)**
- (4) Calculer les coûts des Conteneurs (FinOps Foundation) **(Lien)**
- (5) (Serverless Container Showdown (21 Juillet 2023) **(Lien)**