Guillaume Chervet  
Passionné par l’informatique, tout simplement !   
Principal ML Engineer at AXA France - Microsoft MVP

# image profil

# J'ai perdu du poids sur Kubernetes avec SlimFaas

## Introduction

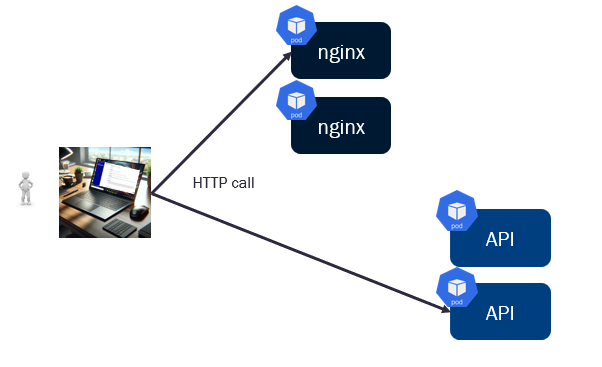
Savez-vous que dans les clusters Kubernetes il y a entre 30% à 70% des CPU non utilisé et 20% à 60% RAM non utilisée en environnement de production. Et encore plus, avec entre 60% à 90% CPU non utilisé et 50 à 80% à RAM non utilisée en environnements de développement, recette et préproduction.

Autant de ressources qui consomment de l’électricité et augmentent les coûts pour rien. Heureusement, SlimFaas est là pour vous aider, c’est un tout petit proxy HTTP qui se plug sans couplage à votre architecture existante et permet d’éteindre vos instances de PODS non sollicité. L’impacte **GreenIT** que vous pouvez obtenir en quelques minutes est tout simplement incroyable ! Cela nécésite tout de même de que vous compreniez comment cela fonctionne « sous le capot » et c’est tout l’objectif de cet article.

## Pourquoi SlimFaas ?

Prenons un exemple et imaginez que :

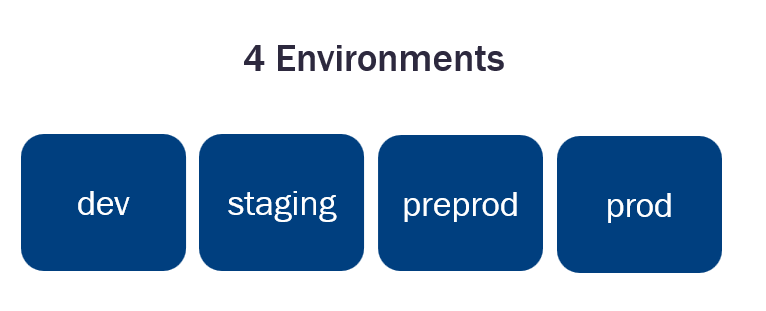
* Vous avez en production sur un namespace Kubernetes une application statique HTML + Javascript servie par un serveur nginx, et que cette application web appelle une API via le protocole HTTP.
* Votre application est utilisée principalement du lundi au vendredi de 8h à 19h uniquement.
* Vous payez 92 € par mois par gigaoctet.
* Pour être résilient aux pannes vous avez doublé le nombre d’instance de pods.
* Un pod nginx nécessite 40 Mo de RAM.
* Un pod de l’API consomme 1 Go de RAM.



**Architecture Avant SlimFaas : API + NGINX**

Votre infrastructure vous coute : (1 Go \* 2 Pods + 0,04 Go \* 2 Pods) \* 92 € = **191 € / mois**

Sachant que l’on a 4 environnements (Développement, recette, préproduction, production) le coût réel est : 191 \* 4 = **764 € / mois** soit **9 168 € / an** !



Imaginez maintenant que l’on éteint votre infrastructure et que vous ne la payez pas entre 19h et 8h du matin ainsi que le weekend. Théoriquement, cela représente une économie de 13h par jour ouvré et 48h le week-end. Votre facture passe de **764 € / mois** à **255 € / mois** soit environ **66%** d’économie !

Et vous allez voir qu’avec SlimFaas, vous pouvez économiser encore beaucoup plus !

## Comment fonctionne SlimFaas ?

SlimFaas se déploie comme vos propres images Docker, dans le même namespace que votre API. Il agit comme un proxy qui se met en façade de vos API.

Si votre API Kubernetes s’appelle « fibonacci » et possède une route HTTP POST /compute, vous pourrez l’appeler en passant par Slimfaas, par exemple :

HTTP POST http://slimfaas /function/fibonacci/compute {"Input":29}

Slimfaas re-routera alors l’appel à votre pod fibonacci :

HTTP POST http://fibinacci /compute {"Input":29}

SlimFaas va donc voir passer toutes vos requêtes, et ainsi être capable de savoir lorsqu’un pod n’a pas été appelé depuis une période définie (par exemple 5 minutes) et pourra alors décider de l’éteindre afin d’économiser les ressources inutilisées (scale to 0).

En cas de nouvelle requête, si aucune instance du pod fibonacci n’est allumé, SlimFaas va pouvoir garder la requête en mémoire, lancer le démarrage d’une instance du pod fibonacci, et lorsque celui-ci sera démarré et en état « Ready » (c’est-à-dire répondant à un ping sur une route de type /heath utilisé pour le heath check), SlimFaas enverra la requête a « fibonacci ». Dans le monde Kubernetes, une instance d’un pod s’appelle un « replica ».

### Son architecture

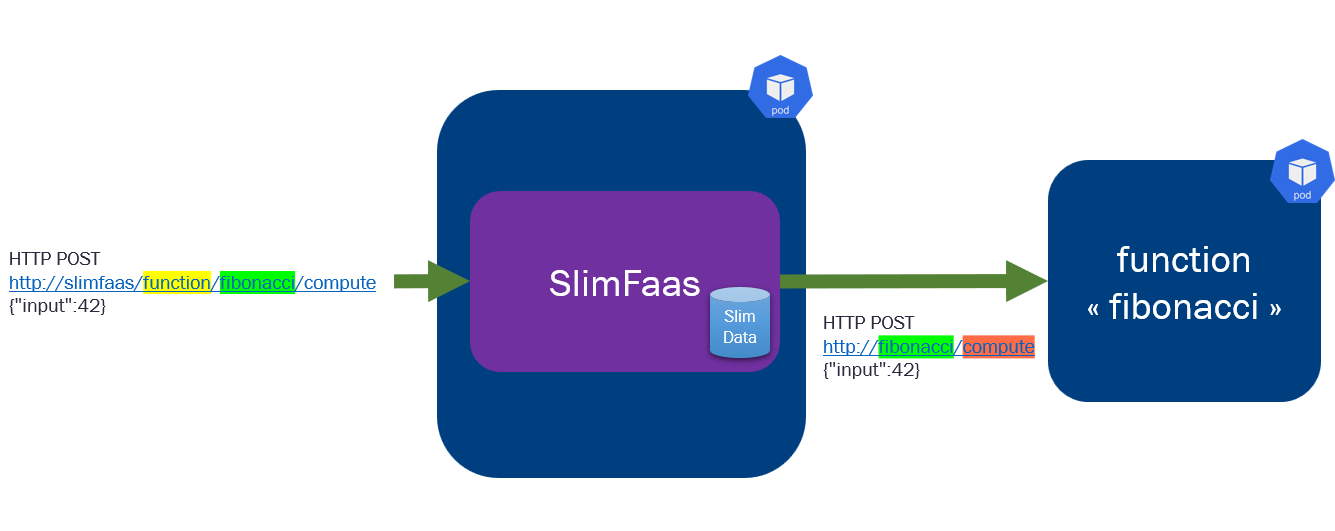
En production, pour être résiliant aux pannes, il vous faut au minimum 2 replicas de chaque pod. Le problème avec cela, c’est que chaque replica de SlimFass a besoin d’avoir une vision complète des requêtes qui passent et d’être parfaitement synchronisé, car une requête peut passer par n’importe quelle instance du pod de SlimFaas.

La Solution ? SlimFaas est à la fois un proxy ainsi qu’une vraie base donnée afin d’atteindre un seul et même objectif : être plug and play !

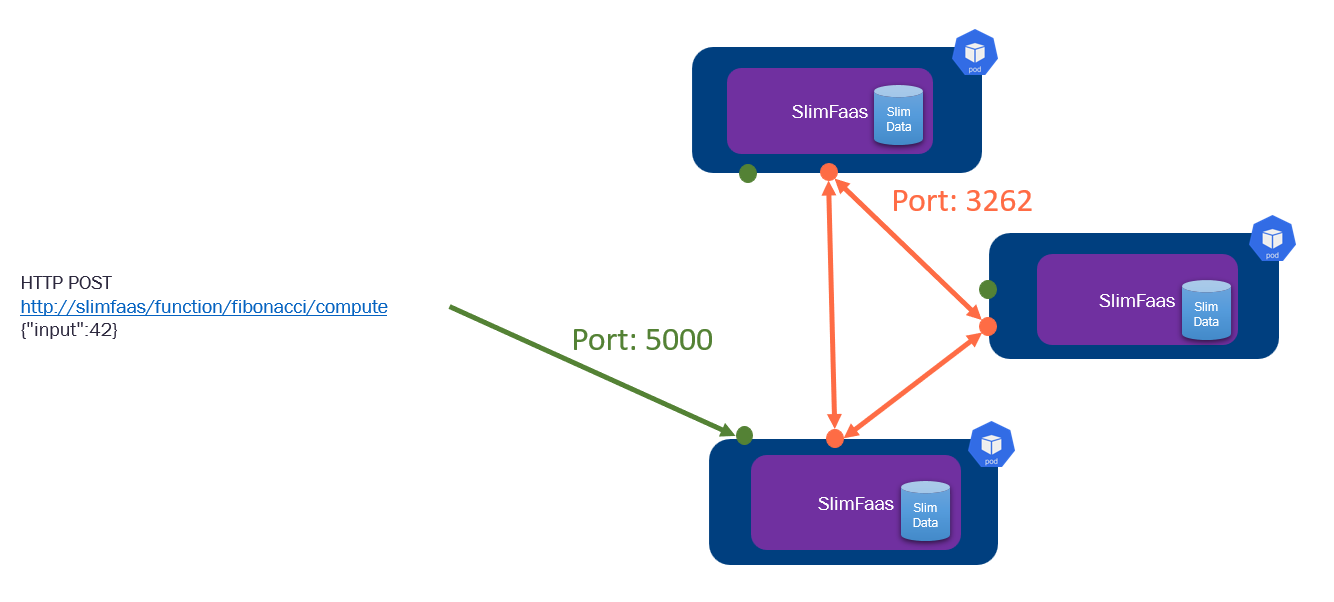
Vous avez une infrastructure existante sur Kubernetes ?

Déployez SlimFass, ajoutez **une** seule annotation à vos pods existants et c’est bon : vous pouvez jouer avec !

|  |
| --- |
| apiVersion**:** apps/v1 kind**:** Deployment metadata**:** name**:** fibonacci  namespace**:** slimfaas-demo spec**:** selector**:** matchLabels**:** app**:** fibonacci  template**:** metadata**:** labels**:** app**:** fibonacci  annotations**:** SlimFaas/Function**:** "true" # Declare votre pod a SlimFaas  SlimFaas/TimeoutSecondBeforeSetReplicasMin: "5" # Eteindre après 5 secondes d’inutilisation  spec**:** serviceAccountName**:** default  automountServiceAccountToken**:** false  containers**:  -** name**:** fibonacci1  image**:** axaguildev/fibonacci-api:latest  resources**:** limits**:** memory**:** "96Mi"  cpu**:** "50m"  requests**:** ephemeral-storage**:** "200Mi"  memory**:** "96Mi"  cpu**:** "10m"  ports**:  -** containerPort**:** 5000 |

**Configuration Kubernetes d’un Pod Fibonacci avec l’annotation SlimFaas/Function**

**Schéma du fonctionnement du Proxy SlimFaas**

SlimData est une base de données simple de type Redis incluse dans l’exécutable SlimFaas. Il est basé sur le protocole RAFT. C’est le même protocole qui est aussi utilisé pour les bases de type SQL (ainsi que Redis), avec un nœud principal qui prend les écritures et des nœuds qui réplique la donnée. Par défaut, SlimData utilise un second port HTTP 3262 que les nœuds utilisent pour communiquer entre eux. Pour des raisons de sécurité, ne l’exposez pas en dehors de votre namespace. SlimFaas nécessite au moins 3 nœuds en production pour être résiliant aux pannes. 2 nœuds minimum sont nécessaires pour maintenir la base de données dans un état cohérent. Bien sûr, en dévelopement, un seul nœud suffit.

**Schéma de la communication SlimData entre les replicas SlimFaas**

Pour être utile, SlimFaas doit consommer le moins de RAM et CPU possible. Aujourd’hui, je configure en production SlimFaas à **80 Mo** de mémoire et **200 m** en CPU (0,2 cœur d’un CPU). Avec notre exemple précédent, SlimFaas coute : 3 \* 0,08 Go \* 92 € = **22,08 € / mois**.

SlimFaas a été codé en .NET 9 et est compilé en mode natif, c’est-à-dire en exécutable indépendant de la Machine Virtuel .NET 9. Il est actuellement compilé pour les OS de type x64 et aussi ARM64 (pratique sur un Raspberry).

## Problème du démarrage via un trigger http : Planet Saver

Reprenons notre exemple du tout début et ajoutons SlimFaas à notre architecture. L’avantage, c’est que l’on va pouvoir éteindre notre API lorsqu’elle n’est plus utilisée. Afin que SlimFaas ait un véritable intérêt, il faut que votre instance de Kubernetes soit configurée pour pouvoir provisionner des nœuds dynamiquement (c’est-à-dire des machines virtuelles dans lesquelles seront déployés vos pods).

Le démarrage de 0 à 1 replica n’est en réalité pas si simple que cela ! Il peut y avoir un problème. Si une Machine Virtuelle est disponible et que votre API est rapide à démarrer, il n’y a pas de souci. Si, au contraire, l’API est lente à démarrer (par exemple une minute) ou qu’aucune Machine Virtuelle n’est disponible et qu'il faut en provisionner une nouvelle, vos requêtes risquent de tomber en timeout ! Cela entraînera un dysfonctionnement de votre application. Pour information, le provisionning dynamique par une infrastructure Kubernetes d’une machine virtuelle sur Azure nécessite environ 7 minutes !

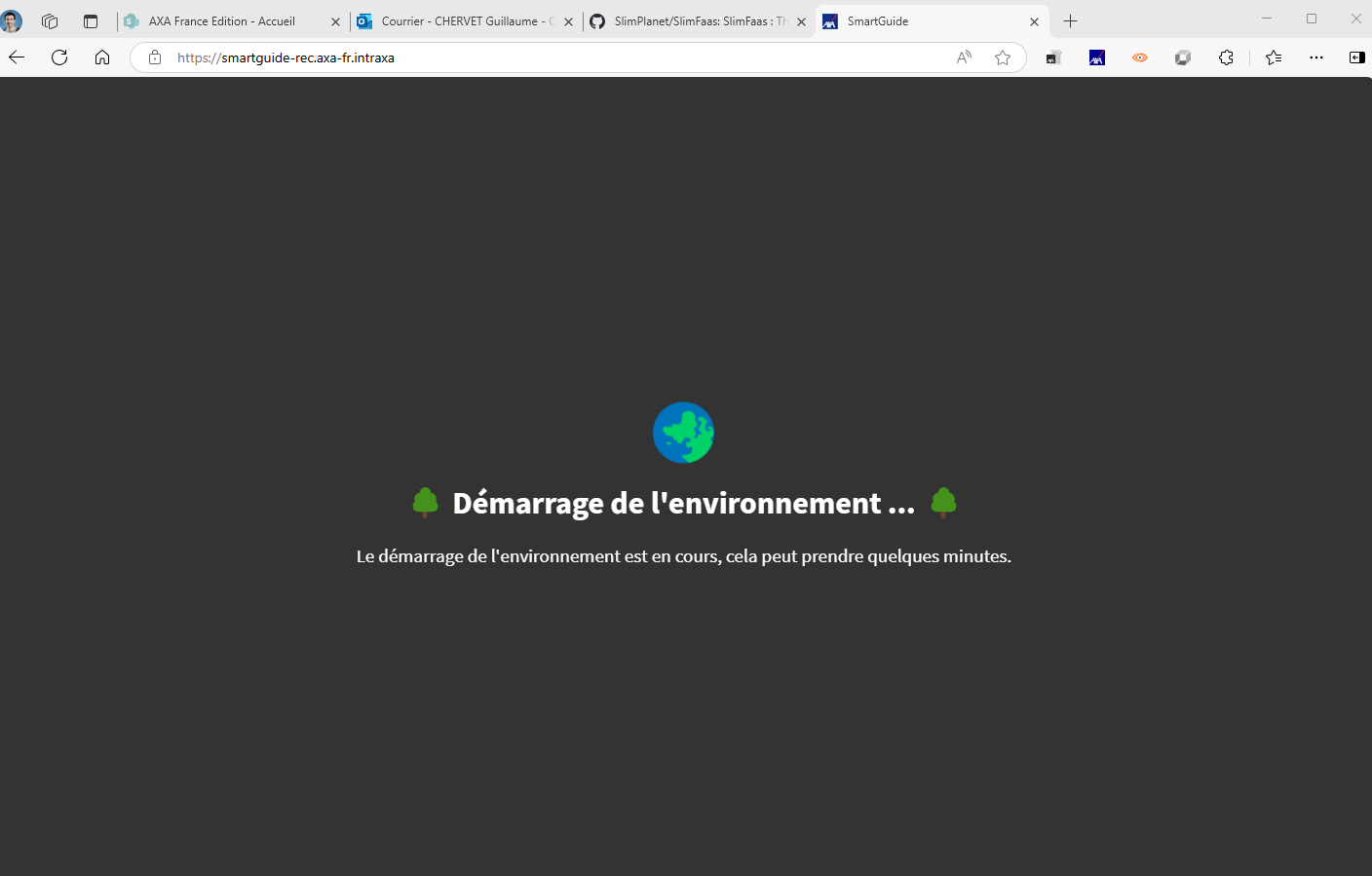
C’est là où intervient la librairie Javascript @axa-fr/SlimFaasPlanetSaver, qui utilise des routes spécifiques à SlimFaas pour communiquer des informations au front Javascript sur l’état de l’infrastructure. L’idée de ce module est de changer les mentalités, un « **Mind Changer** ». Avant SlimFaas, on utilisait minimum deux replicas du pod API afin d’être résiliant aux pannes. Avec SlimFaas et PlanetSaver, l’idée est de transférer cette résilience d’architecture « **Physique »** qui coute cher à la planète par une résilience « **UX » (User eXpérience)** côté navigateur. Le petit module Javascript est écrit en pur javascript. Il est compatible avec tous les Frameworks du marché : react, angular, etc. Il appelle une route spécifique de SlimFaas qui communique des informations très sommaire sur l’infrastructure. Exemple :

HTTP POST <http://slimfaas/status-functions>

qui retourne un tableau JSON du type: [{"NumberReady":0,"NumberRequested":0,PodType":"Deployment","Visibility":"Public","Name":"fibonacci"}]

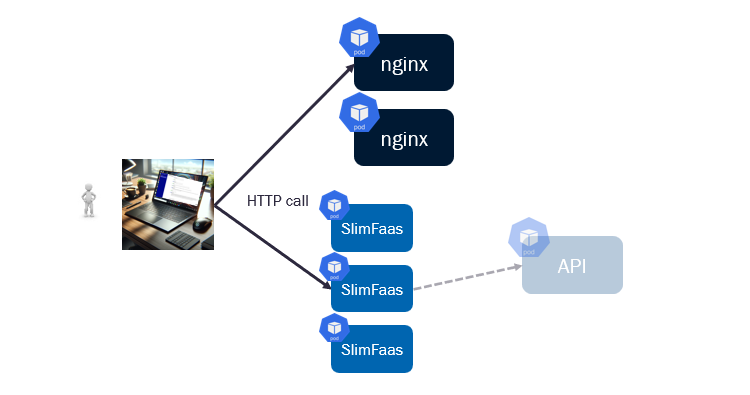
Si l’environnement est éteint, le composant Javascript réalise l’appel à une route spécifique afin de démarrer un replica de l’api et il affiche un loader qui informe l’utilisateur que l’infrastructure démarre.

HTTP POST [http://slimfaas/wake-function/fibonacci](http://slimfaas/wake-function/fibonacci1)



**Loader du composant Javascript SlimFaasPlanetSaver**

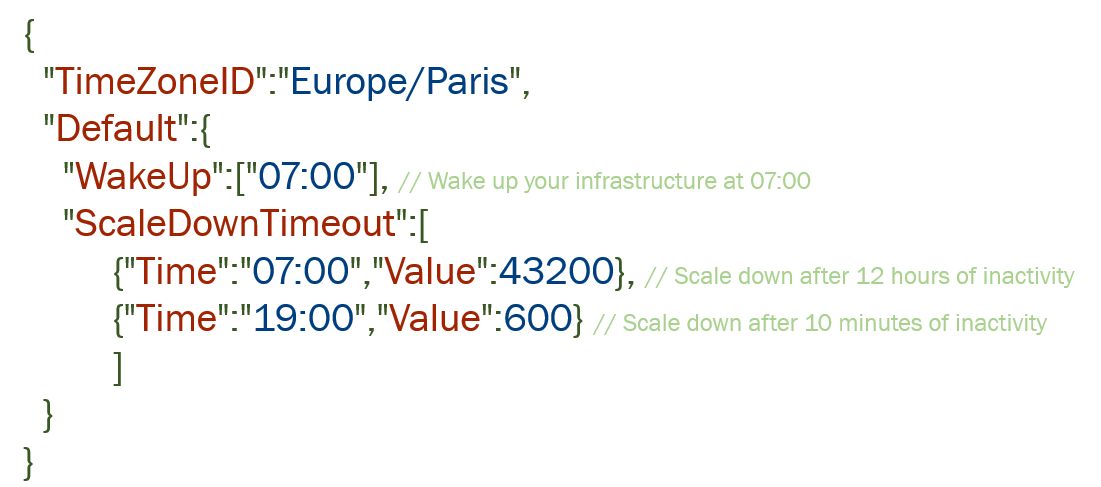
L’avantage de ce composant est que si l’API s’éteint pendant la saisie d’un formulaire par exemple, le « loader » SlimPlanet va s’afficher mais vous ne perdrez pas votre contexte Javascript ; vous retrouverez votre formulaire dans l’état ou il était auparavant.



**Architecture Avec SlimFaas**

Si l'on refait les calculs, en éteignant votre environnement entre 8 h et 19 h ainsi que tout le week-end, vous économisez environ 66 % des ressources, mais maintenant avec SlimFaas + PlanetSaver, vous pouvez en production utilisez seulement qu’un replica de votre API au lieu de deux à chaque fois. Vous réalisez maintenant **minimum 80%** d’économie ! Votre infrastructure vous coûtera **1 834 €** par an au lieu de **9 168 €**, soit **7 334 €** d’économies !

Si vos utilisateurs ne veulent vraiment jamais attendre, vous pouvez démarrer les replicas de vos pods en avance de phase en utilisant l’annotation « SlimFaas/Schedule » qui s’ajoute sur les scripts Kubernetes de déploiement de votre API. Dans l’exemple ci-dessous, votre API démarre à 7 heure du matin et ne s’éteindra pas de la journée. Cela laisse 1h à votre machine Virtuel pour démarrer si besoin ! Les utilisateurs qui commencent à travailler à 8 h du matin n’auront alors jamais de temps d’attente dans la journée. L’API s’éteindra ensuite à partir de 19h en cas de non-sollicitation.



**SlimFaas/Schedule**

**Tableau récapitulatif pour notre exemple :**

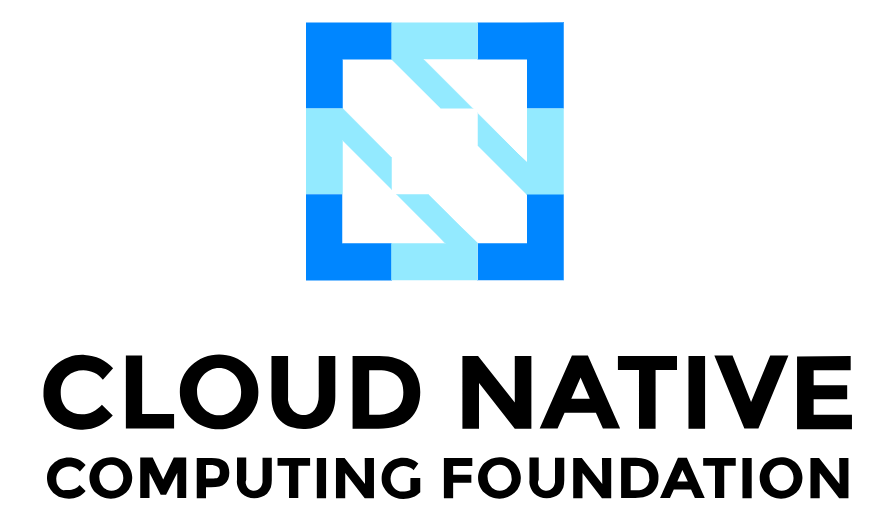
|  |  |  |
| --- | --- | --- |
|  | Avant SlimFaas | Avec SlimFaas |
| Architecture |  |  |
| Résilience | Physique : 2 replicas allumés en permanence. | UX : via  @axa-fr/SlimFaasPlanetSaver |
| Temp de latence moyen  (estimé via benchmark 33 request/seconds) sur route http GET) | 7 ms | 13 ms  (+ 6 ms) |
| Cout | 9 168 € / an | 1 834 € /an soit 80% d’économie |

Conclusion : Soyez les prochains à témoigner !

Perdre du poids sur Kubernetes est à la portée de tous. Cela demande bien sûr un investissement ; cependant, SlimFaas met en place les recettes qui vous aideront ! Cet article ne parle pas de toutes les ingrédients, fonctionnalités offertes par SlimFaas, il y a aussi par exemple :

* Les fonctions asynchrones, un vrai système de « queue » grâce à SlimData.
* Le système de publication d’événements via HTTP (de type publish-subscribe), sans nécessité de driver qui couple votre code.
* Le démarrage de « Jobs » via des appels API et de manière sécurisée,
* Etc.

SlimFaas a rejoint début 2025 la Sandbox de la Cloud Native Computing Foundation (CNCF). C’est une fondation sous l'égide de la **Linux Foundation** qui vise à promouvoir l'adoption et le développement des technologies **cloud-native**. Elle héberge et soutient des projets open source essentiels à la construction, au déploiement et à la gestion d'applications modernes.



Si vous voulez en savoir plus sur SlimFaas ou bien même contribuer, rendez-vous sur la page GitHub : <https://github.com/SlimPlanet/SlimFaas>Soyez les prochains à témoigner de votre perte de poids dans votre cluster Kubernetes ! Un grand merci à Fares Ahmed et Antoine Lernould pour leur relecture !