Rapport du projet de cloud

https://git.unistra.fr/nderousseaux/projet-cloud-virt

I - Contexte

Dans le cadre de ce projet, nous avons une application fournissant un service de redimensionnement d'images de manière hautement automatisée et résiliente. L'objectif principal était de permettre aux utilisateurs de déposer leurs images tout en les redimensionnant simultanément dans différentes tailles.

L'intérêt de ce projet était de rendre l'application résiliente et automatisée. Ainsi, à l'aide des technologies nomad, consul, keepalived et docker notre application peut facilement se scaler horizontalement pour gérer la montée en charge, et est résiliente à la perte d'un des deux serveurs.

Nous avons aussi créé un docker compos, permettant aux développeurs de tester leur application dans un environnement proche de la production avant de déployer l'application. Toute la démarche est expliquée dans le readme.

II - Déploiement d'une nouvelle version de l'application

Déployer une nouvelle version de l'application est plutôt simple, du moment que l'on est un utilisateur autorisé à déployer, on possède les secrets qui sont:

- Les différentes variables d'environnement (accès au bucket AWS, et à la file de messages Rabitmq).
- L'accès à la machine krimmeri et les clés privées correspondantes.
- L'accès au repos quay.io d'image docker.

Un développeur possédant ces secrets est donc autorisé à déployer. Il lui suffit de mettre les variables d'environnement dans un fichier env-file à la racine du projet, puis de lancer le script deploy.sh. Ce script va build les images docker, les envoyer sur les dépôt quay.io, puis lancer les jobs nomad sur krimmeri avec les bonnes variables d'environnement.

Il faut que le développeur indique un nouveau numéro de version pour que nomad redéploie.

Évidemment dans notre cas, il faut aussi être connecté au vpn de l'université.

III - Procédure pour effectuer une maintenance planifiée d'un nœud

Nomad gère le fait qu'un des deux nœuds de notre système puisse planter. Pour une maintenance planifiée, et donc rediriger tous les services sur le noeud restant il faudrait, sur le noeud à éteindre :

- Éteindre keepalived (systemctl stop keepalived). Ainsi l'ip flottante sera attribuée au bon noeud.
- Éteindre nomad et consul (systemctl stop nomad.service && systemctl stop consul.service) Ainsi tous les services seront redirigés sur le nœud restant.

IV - Ajouter ou supprimer un noeud de notre infrastructure

Pour supprimer un nœud de l'infrastructure, il suffira de faire comme précédemment : éteindre keepalived, nomad et consul. Avant ça, consul devra se désinscrire du cluster (consul leave sur la machine à désinscrire)

Pour ajouter un nœud il faudra une nouvelle machine. On supposera que cette nouvelle machine est connectée de la même manière que les deux premières, c'est-à-dire avec une ip et un nom d'hôte interne, et pouvant avoir l'ip flottante (les deux sur vxlan100).

- Installer consul, nomad, docker et keepalived.
- Configurer consul, nomad et keepalived avec les fichier de config (dans le dossier deploy/config)
- Se connecter au dépot docker Quay.io (docker login quay.io)
- Redémarrer consul et nomad (systemctl restart nomad.service && systemctl restart consul.service)
- Ajouter le nouveau serveur consul au cluster (consul join <nom-hote-du-nouveau-noeud> sur le serveur principal de nomad, soit krimmeri)

V - Impact des différents scénario de pannes

Si le réseau tombe en panne, comme ce n'est pas nous qui le gérons, on ne peut rien faire. Si l'un des deux nœuds et le serveur krimmeri sont toujours connectés (entre eux et à l'extérieur), l'application marchera toujours car nomad transférera les services sur le nœud encore fonctionnel, et keepalived mettra l'ip flottante sur celui-çi.

De la même manière, si l'un des deux nœuds tombe, aucun problème, keepalived et nomad se chargeront de fonctionner sur un seul serveur.

En revanche, si le serveur nomad tombe en panne, toute l'application ne fonctionnera plus. En effet, comme c'est lui qui orchestre le tout, il est primordial.

On part aussi du principe que le bucket S3 et la fille de message rabitmq ne risque pas de tomber. Dans le cas contraire, toute l'application cesserait de fonctionner.

VI - Explications pour un nouvel arrivant

Toute la démarche explicative pour un nouvel arrivant qui devrait maintenir cette infrastructure à notre place est déjà expliquée dans le wiki du gitlab.

VII - Piste d'amélioration

Pour beaucoup plus de sécurité, il aurait fallu mettre les variables d'environnement dans un coffre fort vault, surtout que cela s'intègre très bien avec nomad.

Nous avons décidé d'utiliser un script deploy.sh pour déployer une nouvelle version de l'application. Pour simplifier les choses, on aurait pu mettre en place un auto déploiement pour chaque commit sur master avec les ci/cd gitlab. Mais cela aurait demandé de partager

notre clé privée avec le gitlab-runner. Dans un cas réel, le mieux aurait été de créer notre propre instance gitlab afin de ne pas partager les informations sensibles. Sans instance gitlab personnelle, nous avons considéré que le plus sécurisé était d'utiliser un script deploy.sh.

Nos jobs nomad n'intègrent pas de scale up horizontal automatique. Il aurait fallu pour cela ajouter une extension nomad qui ferait le monitoring pour nous (comme datadog par exemple)

Une autre bonne idée aurait été d'ajouter des alertes afin de nous mettre au courant en cas de problème (par mail par exemple). Un service de monitoring datadog aurait pu faire l'affaire.

Pour finir, nous aurions pu mettre en place des scripts pour copier les images des vms avec kvm, afin de permettre un redéploiement rapide en cas de perte totale des données d'une vm.