Télécom Physique Strasbourg

Informatique & Réseaux

Semestre 8

Cours : Cloud et Virtualisation

Rapport de projet : Déploiement d'une application résiliente et scalable avec Consul, Nomad, HAProxy et MQRabbit

Sofian Mareghni

Nicolas cipolla

Lien vers le git : [Schachouflash/projet-cloud-virt (github.com)]

Table des matières

I. Contexte

Objectif du projet :

Ce projet a pour but de déployer une application fournie de la manière la plus automatisée et résiliente possible. L'application permet à ses utilisateurs d'héberger des images, en les redimensionnant au passage dans différentes tailles.

Détails de l'infrastructure :

L'infrastructure est composée de :

- D'un stockage type S3
- Une file de message RabbitMQ
- Un serveur Consul
- Un serveur Nomad
- Trois machines virtuelles (dont une hébergeant le serveur Consul et Nomad)
- Une IP flottante
- Des tunnels HTTP vers vos machines virtuelles et votre IP flottante

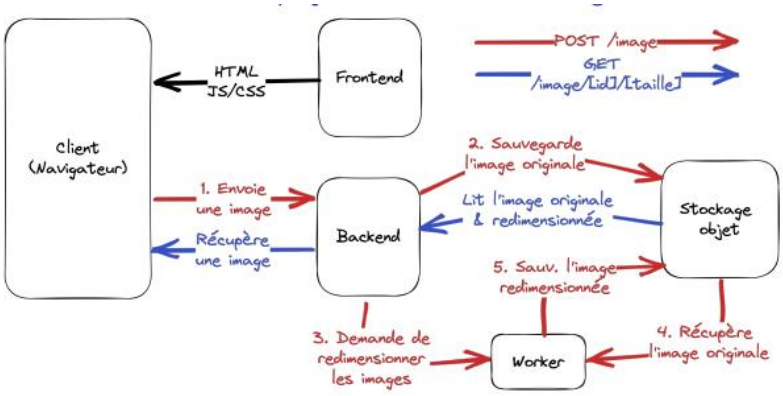


Figure 2: Architecture du projet

Approche pour atteindre nos objectifs :

Pour répondre aux objectifs du projet, nous avions opté pour le mode d'emploi suivant :

- Clonage du dépôt et configuration de l'application
- Configuration de Nomad et Consul
- Conteneurisation sur chaque VM des différentes instances via des Dockerfile
- Configuration du registre d'images de conteneurs
- Mise en place d'une IP flottante
- Configuration du Load balancer par HAProxy
- Configuration de l'infrastructure via le playbook Ansible

Cette façon de procédé avait pour but d'avoir un déploiement rapide de l'application dont l'automatisation sera faite peu à peu par la suite via le playbook Ansible et autres scripts. En effet, nous pensions que dont le cas réel, il aurait été préférable de déployer l'application rapidement pour les clients afin qu'ils puissent l'utiliser, puis rendre le travail totalement propre par une automatisation complète petit à petit.

Note :

Chaque phase a été accompagnée de tests pour vérifier le bon fonctionnement de l'application et de son infrastructure. Ces tests nous ont permis de nous assurer que chaque composant était configuré correctement et que l'ensemble du système fonctionnait de manière cohérente.

II. Notre travail

Cette partie explique nos différentes mises en œuvre pour répondre à nos besoins. Ce qui n'a pas été fait sera expliqué dans la section "Améliorations" du rapport, où nous expliquerons comment y parvenir, ainsi que dans la section "Difficultés", où nous expliquerons les problèmes rencontrés.

a. Consul et Nomad

Dans un premier temps, nous avons configuré Consul pour répertorier les différentes machines virtuelles (VM). Cela permet aux VM de découvrir les différents services et les changements au niveau de la structure globale. Ensuite, nous avons déployé les différents jobs Nomad. Ainsi, notre infrastructure Docker pourra être déployée sur n'importe quelle infrastructure de conteneurs.

Pour cela, nous avons tout d'abord créé les fichiers de configurations nomad.hcl et consul.hcl en s'inspirant du fichier de configuration de la VM principale. Cependant, contrairement à cette dernière, les VMs vont être clientes et non serveur. Puis, avec les commandes consul join montagne-verte et nomad server join montagne-verte.

b. Containerisation via Dockerfile

En parallèle, nous avons également mis en place les Dockerfiles pour le frontend et le backend. La création de ces conteneurs permet de maintenir les services indépendamment les uns des autres, tout en leur permettant de communiquer entre eux via le réseau pour former l'application complète.

Nous avons créé deux Dockerfiles :

- Frontend : Nous avons utilisé l'image node.js@latest (FROM) et configuré les différentes dépendances en utilisant les commandes "run" pour construire le projet.
- Back-end / worker : le dockerfile se trouve cette fois ci dans le dossier API : nous avons ici utilisé l'image python :latest. Nous avons ensuite configuré les différentes variables d'environnement nécessaire pour faire fonctionner le worker : la file de message RabbitMQ pour le broker celery, L'adresse du stockage S3, et les access ID et key

Nous avions ensuite utilisé ghcr.io pour stocker nos images et ainsi les réutiliser dans le fichier de configuration utilisé pour instancier le job nomad

En utilisant la commande nomad job run nous créons ainsi un job sur nomad

c. Configurations du registre d'images de conteneur

Comme mentionné précédemment, nous avons utiliser ghcr.io nos images et ainsi les réutiliser dans le fichier de configuration utilisé pour instancier le job nomad

d. IP Flottante :

Cette partie vise à mettre en place la résilience de notre infrastructure. L'utilisation d'une adresse IP flottante permet de rediriger le trafic vers un nœud actif en cas de défaillance d'un autre nœud.

Pour cela, nous avons choisi d'utiliser Keepalived et HAProxy, comme suggéré dans l'énoncé. Après l'installation de Keepalived, des fichiers /etc/keepalived/keepalived.conf ont été générés sur chaque

VM afin de définir l'adresse IP flottante et les rôles respectifs de chaque VM (MASTER ou BACKUP). Selon nos recherches, il était également recommandé d'ajouter un script /etc/keepalived/chk_haproxy.sh permettant à l'adresse IP de basculer automatiquement d'une VM à une autre en fonction de l'état du serveur. Cependant, il s'est avéré que ce script n'était pas nécessaire pour la réussite de cette opération grâce à consul qui va vérifier que l'état des VMs.

Nous avons rencontré de nombreuses difficultés pour cette partie qui seront plus détaillé dans la parties difficulté.

e. Load Balancer

Dans notre projet, nous avons utilisé HAProxy comme Load Balancer pour répartir le trafic entre les instances de notre application backend. Cela améliore la disponibilité, la résilience et les performances de l'application en redirigeant les demandes vers des nœuds actifs et en équilibrant la charge.

Nous avons configuré HAProxy en spécifiant les adresses IP, les ports et les vérifications de santé des instances backend (http-in et backend). Nous avons également défini les services de backup et intégré dans ce même fichier haproxy.cfg. Pour le déploiement des applications backend, nous avons utilisé des fichiers de tâches Nomad qui définissent les ressources, les paramètres réseau et les configurations de service nécessaires via leurs images. Cette solution est censée garantir la scalabilité et la fiabilité de notre infrastructure, tout en optimisant l'expérience utilisateur en répartissant efficacement le trafic.

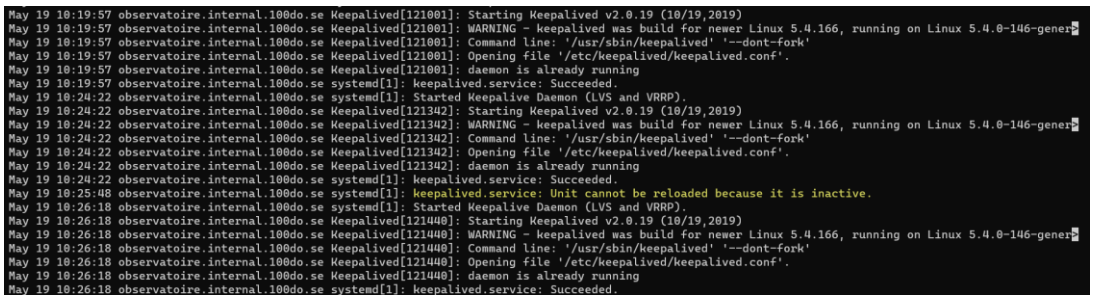
Note : c'est partie n'as pas eu le temps d'être totalement vérifié.

f. Difficultés

La première difficulté a été de faire les différents dockerfile. N'étant pas habitué à en faire, comprendre comment bien les utiliser, les positionner et les instancier ne fut pas une mince affaire. Mais au bout de quelques heures et de recherches internet, nous avons réussi à créer des dockerfiles créant des images fonctionnant comme voulu.

La 2ème grosse difficulté a été de savoir ou stocker ses images pour que Nomad puisse créer ses jobs facilement. Nous avons décidé d'utiliser une solution fournie par git : ghcr.io qui permet de stocker les images docker pour pouvoir les utiliser dans le fichier utilisé pour configurer le job Nomad.

La 3ème grosse difficultés concerne l'IP flottante. Après configuration, celle-ci devait apparaître et être active sur toutes les VMs tout en ayant qu'une seul VM qui répond aux requêtes. Cependant, l'IP n'apparaît ce n'est pas simultanément sur les VMs. Force de tentative, une erreur a été commise au niveau de l'IP ce qui a causé un problème de fusion au niveau de consul. Ensuite, certaines VM mettant fait à keepalived pour des raisons inconnues. Nous étions bloqués avec ces log étranges :



Le problème a été réglé par la suite.

III. Fonctionnement

a. Déploiement d'une nouvelle version

Nous recommandons d'adopter une approche progressive lors du déploiement d'une nouvelle version de l'application. Voici les étapes à suivre :

1. Effectuer une mise à jour sur une seule VM : Nous vous conseillons de commencer par déployer la nouvelle version de l'application sur une seule VM. Cela nous permettra de tester la nouvelle version dans un environnement contrôlé tout en maintenant les autres instances en cours d'exécution avec l'ancienne version.
2. Effectuer des tests et vérifications : Après avoir déployé la nouvelle version sur une VM, il est essentiel de réaliser des tests approfondis pour s'assurer que l'application fonctionne correctement. Nous devons vérifier son intégration avec les autres composants du système tels que Consul, Nomad et le load balancer HAProxy.
3. Valider les résultats des tests : Une fois que les tests sont concluants et que la nouvelle version répond à nos attentes, nous pouvons procéder à la mise à jour des autres VMs.
4. Déploiement progressif : Nous recommandons d'effectuer le déploiement progressif de la nouvelle version sur les autres VMs une par une, en suivant la même approche que celle utilisée lors de la première mise à jour. Cela nous permettra de surveiller l'impact de la nouvelle version sur chaque instance et de réagir rapidement en cas de problèmes.

Cette approche progressive nous permet de limiter les risques en cas de problème avec la nouvelle version. Si un problème est détecté, il est plus facile de revenir à l'ancienne version en isolant la VM affectée plutôt que de devoir effectuer un rollback sur l'ensemble de l'infrastructure.

Si plus de temps et de moyens sont disponible, cela peut être fait sur une VM tests privée. Si validé déploiement sur les ¾ puis ¼

b. Procédure de maintenance

Ici, en plus de l'aspect technique, la communication, la rigueur et la régularité sont primordiale. Voici les étapes clés à suivre :

1. Planification de la maintenance : Il est important de planifier à l'avance les périodes de maintenance afin de minimiser les perturbations pour les utilisateurs. Idéalement, choisissez des moments où le trafic est moins important ou prévoyez une fenêtre de maintenance prévue.
2. Communication avec les utilisateurs : Avant la période de maintenance, informez les utilisateurs des éventuelles perturbations à prévoir. Expliquez la raison de la maintenance et communiquez toute information pertinente, telle que la durée estimée de l'indisponibilité des services.
3. Sauvegarde des données : Avant de commencer la maintenance, effectuez une sauvegarde complète des données et des configurations critiques. Cela permet de restaurer rapidement les services en cas de problème pendant la maintenance.
4. Mise en mode hors service : Mettez le load balancer HAProxy en mode hors service pour rediriger le trafic vers une page de maintenance ou une autre instance fonctionnelle. Cela garantit une expérience transparente pour les utilisateurs pendant la maintenance.

5. Séquences logiques : Procédez à l'arrêt contrôlé des services sur chaque VM, en commençant par les instances non critiques. Assurez-vous de suivre une séquence logique pour éviter les dépendances non satisfaites entre les services. Debugger un code est compliqué, alors image tout un déploiement d'application à une échelle mondiale…
6. Maintenance des VMs : Profitez de la période de maintenance pour effectuer des opérations de maintenance sur les VMs, telles que les mises à jour du système d'exploitation, les correctifs de sécurité, les mises à jour des logiciels, etc.
7. Vérification et test : Après avoir terminé les opérations de maintenance sur chaque VM, effectuez des vérifications et des tests approfondis pour vous assurer que tous les services sont revenus en ligne et fonctionnent correctement.
8. Rétablissement du mode de service : Une fois que toutes les vérifications sont satisfaisantes, rétablissez le load balancer HAProxy en mode de service normal pour réacheminer le trafic vers toutes les instances.
9. Communication de fin de maintenance : Informez les utilisateurs et les parties prenantes de la fin de la maintenance et de la disponibilité normale des services. Fournissez également des informations sur les améliorations ou les correctifs appliqués pendant la maintenance.

En suivant cette procédure de maintenance, nous nous assurons que notre infrastructure reste robuste, sécurisée et à jour. De plus, une communication transparente avec les utilisateurs et les parties prenantes contribue à instaurer la confiance et à minimiser les perturbations pendant les périodes de maintenance. Perdre des clients alors que vont les satisfaire serait dommage….

c. Ajouter ou supprimer un nœud

Ajout d'un nœud :

Configuration du nœud : Préparez une nouvelle VM avec la configuration requise pour le nœud que vous souhaitez ajouter. Assurez-vous que la VM est accessible et qu'elle répond aux exigences de notre infrastructure.

Enregistrement dans Consul : Ajoutez la nouvelle VM en tant que nœud dans Consul en utilisant sa configuration. Cela permettra aux autres nœuds de découvrir et de communiquer avec le nouveau nœud.

Configuration de Nomad : Ajoutez les tâches Nomad correspondantes pour le nouveau nœud, en spécifiant les ressources nécessaires, les paramètres réseau et les configurations de service. Veillez à équilibrer la charge de manière appropriée en fonction des besoins de l'application.

Configuration de l'adresse IP flottante : Configurez l'adresse IP flottante pour inclure le nouveau nœud. Cela garantira que le trafic est réparti de manière équilibrée et redirigé vers le nouveau nœud lorsque nécessaire.

Configuration de Keepalived et HAProxy : Si nécessaire, ajustez la configuration de Keepalived et HAProxy pour prendre en compte le nouveau nœud. Vérifier que le load balancer est configuré pour inclure le nouveau nœud dans la répartition de charge.

Suppression d'un nœud :

Mise hors service du nœud : Avant de supprimer un nœud, mettez-le hors service en ajustant la configuration du load balancer pour exclure le nœud de la répartition de charge. Cela garantit que le trafic n'est plus envoyé vers le nœud en cours de suppression.

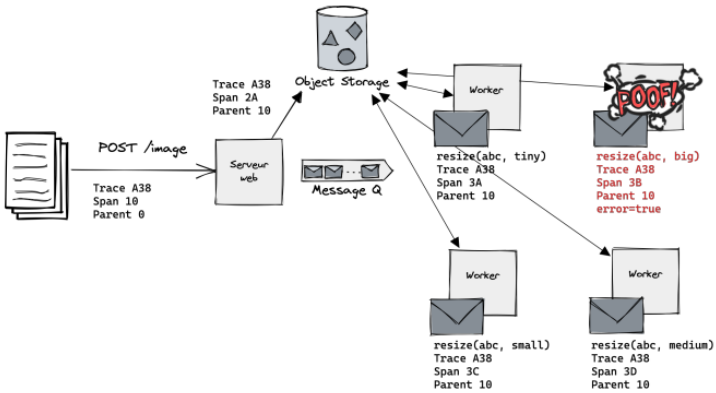
Retrait du nœud de Consul et Nomad : Supprimez le nœud de Consul en le désenregistrant. Assurez-vous également de retirer les tâches Nomad associées au nœud que vous souhaitez supprimer.

Mise à jour de l'adresse IP flottante : Si le nœud supprimé était associé à l'adresse IP flottante, mettez à jour la configuration pour exclure l'adresse IP du nœud supprimé.

Vérification et ajustement de la configuration : Effectuez des vérifications pour vous assurer que le nœud a été correctement supprimé de tous les composants de l'infrastructure. Ajustez si nécessaire la configuration de Consul, Nomad, l'adresse IP flottante, Keepalived et HAProxy pour refléter les changements.

d. Impact de différents scenarios de panne

Dans la plupart des cas, les moyens de remédier aux pannes sont similaires. Nous devrons mettre en place des systèmes de relance et de détection en observant différents schémas grâce à l'analyse des journaux, des métriques et des traces, comme nous l'avons étudié lors du chapitre 8. Pour cela, il sera nécessaire de détecter les problèmes (logs et métriques), de les catégoriser (seuils d'alerte, etc.) puis de trouver leur cause. La technique relativement nouvelle de traces distribuées peut également être envisagée.



Exemple de situations :

1. Panne d'un nœud Consul : Mettre en place un mécanisme de détection automatique des pannes à l'aide d'un outil de surveillance « health check ». Configurer Consul pour qu'il rééquilibre automatiquement les services sur les nœuds disponibles en cas de défaillance. Effectuer des sauvegardes régulières de l'état partagé de Consul pour faciliter la récupération en cas de panne.
2. Panne d'un nœud Nomad : Configurer Nomad pour détecter les pannes de nœuds et rééquilibrer automatiquement les tâches sur les nœuds fonctionnels « health check ». Surveiller l'état des nœuds Nomad pour détecter rapidement les pannes et prendre des mesures correctives. Mettre en place des mécanismes de reprise automatique des tâches sur d'autres nœuds en cas de panne.
3. Panne d'un nœud Consul et d'un nœud Nomad : Avoir des procédures manuelles pour rétablir la coordination entre les services et les tâches en cas de panne simultanée. Documenter les étapes nécessaires pour rééquilibrer manuellement les services et les tâches sur d'autres nœuds opérationnels. Mettre en place des mécanismes de notification pour alerter l'équipe en cas de panne simultanée des nœuds Consul et Nomad.
4. Panne de l'adresse IP flottante : Configurer un mécanisme de basculement automatique vers une autre adresse IP disponible en cas de panne de l'adresse IP flottante. Avoir des procédures

documentées pour résoudre rapidement les problèmes liés à l'adresse IP flottante, par exemple en vérifiant la configuration réseau et les paramètres de Keepalived.

5. Panne de Keepalived : Surveiller en permanence l'état de Keepalived pour détecter les pannes et prendre des mesures correctives. Configurer Keepalived pour redémarrer automatiquement en cas de panne. Mettre en place des mécanismes de basculement automatique vers un autre nœud responsable de la gestion de l'adresse IP flottante en cas de panne de Keepalived.
6. Panne de HAProxy : Configurer HAProxy en mode haute disponibilité avec plusieurs instances fonctionnelles. Utiliser un mécanisme de surveillance pour détecter les pannes de HAProxy et effectuer un basculement automatique vers une autre instance fonctionnelle. Avoir des procédures documentées pour réparer rapidement les problèmes de HAProxy, par exemple en redémarrant le service ou en vérifiant la configuration.
7. Panne d'un conteneur : Configurer les conteneurs pour qu'ils redémarrent automatiquement en cas de panne. Mettre en place des mécanismes de surveillance des conteneurs pour détecter les pannes et prendre des mesures correctives. Configurer Consul et Nomad pour qu'ils rééquilibrent automatiquement les services et les tâches sur d'autres conteneurs fonctionnels.

IV. Amélioration et comment y arriver

Dans le cadre de notre projet, nous avons identifié plusieurs améliorations potentielles pour renforcer notre infrastructure et optimiser son fonctionnement. Voici quelques idées que nous recommandons d'explorer :

Plus d'automatisation avec Ansible et Terraform : Nous pouvons utiliser des outils tels qu'Ansible et Terraform pour automatiser le déploiement et la configuration de notre infrastructure. Cela nous permettra de gagner du temps et d'assurer une mise en place cohérente et reproductible de nos composants, tels que les VM, les conteneurs Docker, les services Consul et Nomad, et le load balancer.

Renforcement des tests : Nous devons mettre en place des tests approfondis pour vérifier la disponibilité, la performance et la fiabilité de notre application. Cela comprend des tests de charge, des tests de résilience, des tests de sécurité et des tests fonctionnels. Les tests automatisés nous aideront à identifier et à résoudre les problèmes potentiels avant qu'ils ne se manifestent dans un environnement de production.

Système de sauvegarde automatique : Il est important de mettre en place un système de sauvegarde régulier et automatique pour protéger nos données critiques. Nous pouvons planifier des sauvegardes régulières des bases de données, des fichiers de configuration et d'autres éléments importants, et les stocker dans un emplacement sécurisé. Nous devons également effectuer des tests de restauration périodiques pour nous assurer que nos sauvegardes sont valides et prêtes à être utilisées en cas de besoin.

Détection des problèmes et relance automatique : Nous devons mettre en place un système de surveillance en temps réel pour détecter les pannes et les problèmes potentiels dans notre infrastructure. Cela peut inclure la surveillance de l'état des VM, des conteneurs, du load balancer et des services Consul et Nomad. En cas de défaillance, nous pouvons configurer des mécanismes de relance automatique pour restaurer rapidement les services et minimiser les temps d'arrêt.

Health checks : Il est essentiel de configurer des vérifications régulières de l'état de nos instances backend. Cela nous permettra de détecter rapidement les problèmes et les défaillances, et de prendre des mesures appropriées, comme retirer automatiquement les instances défaillantes de la rotation du

load balancer. Les vérifications de santé peuvent être basées sur des sondes HTTP, des pings de réseau ou d'autres mécanismes adaptés à notre application.

Traçabilité et journalisation : Nous devons mettre en place un système de traçabilité et de journalisation pour collecter et analyser les journaux de notre infrastructure et de notre application. Cela nous aidera à diagnostiquer les problèmes, à suivre les performances et à répondre aux exigences de conformité. Nous pouvons utiliser des outils tels que Prometheus, Elasticsearch, Grafana pour gérer efficacement nos journaux.

Scalabilité et gestion de la charge : Analyse des métriques pour voir les goulots d'étranglement.

Remerciement :

Nous vous remercions pour votre forte disponibilité pour réparer certaines de nos erreurs à l'approche de la date butoir qui nous a permis de produire ce travail de recherche. Ce fut très instructif !