[redacted]

OLIVEIRA COSTA Ricardo

Rapport de projet – Secured Architecture

Table des matières

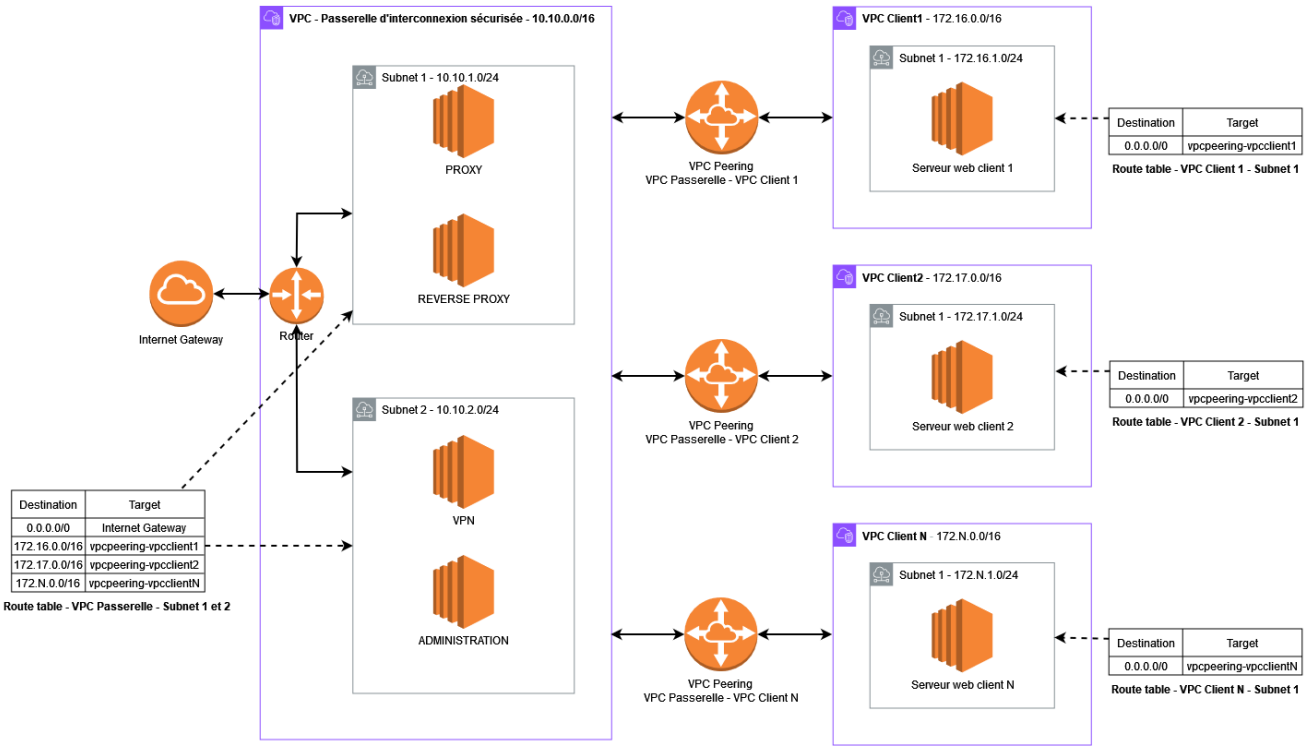
I. Cahier des charges

L'infrastructure qui sera déployée dans le cloud AWS doit répondre à certains critères afin de répondre aux besoins et contraintes de l'entreprise qui va l'utiliser. Une liste de services et contraintes à mettre en place a été définie :

- Créer un VPC qui héberger le VPN et les proxy de l'entreprise ;
- Créer une instance EC2 pour héberger le site web de chaque entreprises ;
- Déploiement via Terraform et Ansible ;
- Déployer une matrice de flux la plus sécuriser possible ;
- Déployer l'infrastructure côté client en moins de 3 minutes ;
- Les zones clients ne doivent pas communiquer entre-elles ;
- Une seule passerelle sécurisée pour communiquer avec internet ;

II. Infrastructure

Schéma d'architecture de l'infrastructure finale



Vous pouvez voir au-dessus le schéma d'architecture globale de notre infrastructure AWS.

Ce schéma est divisé en deux grandes parties, on retrouve à gauche la zone passerelle, elle accueillera notre infrastructure principal permettant de servir d'intermédiaire entre internet et la deuxième partie de l'architecture à droite qui correspond aux zones clients.

Dans cette première zone on va tout d'abord retrouver un VPC, dans lequel nous avons décidé d'installer deux sous-réseaux pour séparer les tâches d'administration et les services dit "web" qui sont des éléments actifs qui communiquent avec les zones clients.

On retrouve 4 instances EC2 qui hébergeront les différents services. Ce VPC dispose d'une passerelle internet qui permet l'exposition de certains de ces services.

Chaque VPC présent ici répond à un objectif différent, le VPC passerelle doit répondre à un besoin de sécurité, en protégeant les VPC client tout en permettant l'administration des différentes instances de l'infrastructure.

A droite on peut retrouver les zones clientes chacune constituées d'un VPC, d'un sous réseau et d'une instance EC2, l'interconnexion entre un client et la zone passerelle s'effectue via un VPC Peering.

Un élément qui n'apparait pas sur ce schéma est le groupe de sécurité, ils sont mis en place par instance EC2 afin de garantir uniquement le passage de certains flux réseaux.

III. Choix des solutions

1. VPC

Suite au schéma montrée précédemment, on peut voir que 3 VPC ont été créé :

- Le main-vpc ;
- Le client1-vpc ;
- Le client2-vpc ;

Chaque VPC offre une isolation logique des ressources. En utilisant plusieurs VPC, on isole chaque système les uns des autres. Avec plusieurs VPC, il est possible d’appliquer des politiques de sécurité spécifiques à chaque VPC, et les ressources dans un VPC ne peuvent pas communiquer directement avec celles d’une autre VPC sans configuration appropriée.

2. EC2

Dans cet exemple on a créé 6 instances :

- Une EC2 main-proxy, qui sert à gérer le proxy de l'infrastructure, pour ça dans cette instance on à installer Squid ;
- Une EC2 main-reverseproxy, qui desservira l'infrastructure comme reverse proxy, l'outil utilisé dans cette instance est Nginx ;
- Une EC2 main-bastion, qui sert comme machine d'administration qu'on utilise pour se connecter aux autres machines ;
- Une EC2 main-VPN, on utilise cette instance comme moyen de se connecter à l'instance bastion, elle géré le VPN pour notre infrastructure avec l'outil OpenVPN ;
- Deux EC2 client-web (client1-web et client2-web) qui sont des serveurs web normaux ;

Chaque EC2 créée permet de gérer les serveurs WEB des différents clients, et d'avoir la machine d'administration.

Nous avons décidé d'utiliser des images Debian pour ces instances car c'est une distribution que nous maitrisons et c'est une solution compatible pour de la production

3. OpenVPN

Nous avons décidé d'utiliser OpenVPN car cette solution en tant que VPN est de facile utilisation pour les clients et la mise en place de cette dernière se fait rapidement avec un script ansible.

OpenVPN nous permet d'utiliser du SSL/TLS ainsi que d'autres algorithmes de chiffrements reconnus pour leur puissance. Il est aussi un outil flexible qui peut être installer sur n'importe quel Operating System. Cet outil permet de crée des comptes simples mais bien sécurisés avec de la MFA.

4. Proxy/ReverseProxy

Dans ce qui concerne le choix du Proxy et le Reverse Proxy on a choisi deux outils très connus mais au même temps ce sont des outils très puissants :

- Pour le proxy on a choisi **Squid :** c'est un outil qui est connue par sa rapidité de sa mise en cache des requêtes afin de réduire le temps de réponse et économiser la

bande passante. Il permet aussi d'ajouter une couche de sécurité supplémentaire et de limiter la surface d'attaque exposée vu qu'on peut configurer le contrôle d'accès à certaines ressources web ;

- Pour le reverse proxy on a choisi **Nginx :** Ceci est un logiciel qui est reconnu pour sa performance et stabilité en tant que web server et reverse proxy. C'est un outil qui gère bien les charges de trafic élevées et offre aussi des nombreuses fonctionnalités de sécurité comme la prévention contres des attaques DDOS et il à aussi l'intégration native SSL/TLS pour le chiffrement.

5. Serveur WEB

AWS offre une vaste gamme d'options de serveurs web parmi lesquelles les clients peuvent sélectionner celui qui répond le mieux à leurs exigences spécifiques en matière de performances, de fonctionnalités et de compatibilité.

Tant que le serveur web souhaité est disponible sur AWS, les clients ont la liberté de faire leur choix en fonction de critères tels que la popularité, la compatibilité avec leur application, ou d'autres préférences techniques.

Cette approche permet aux clients de personnaliser leur environnement d'hébergement de manière à tirer pleinement parti des fonctionnalités offertes par le serveur web de leur choix, tout en profitant des avantages et de l'infrastructure robuste fournis par AWS.

IV. Mise en place

1. Présentation des outils utilisés

La mise en place de cette infrastructure se fait dans AWS avec l'aide de Terraform et Ansible qui permettent de déployer automatiquement cette infrastructure.

Terraform est un outil qui est spécialisé pour faire de l'Infrastructure as Code, avec cet outil on peut créer, modifier et aussi versionner de façon sécurisée et assez efficacement des infrastructures cloud. C'est un outil crucial pour la mise à l'échelle et l'automatisation des déploiements sur des plateformes cloud tels que AWS.

Pour Ansible, c'est un outil qui est utilisé pour la configuration automatisée. C'est un outil idéal pour faire de la gestion de configurations à travers les systèmes et les applications, il permet de les déployer de façon cohérente et répétables.

L'utilisation de ces deux outils ensemble permet de déployer l'infrastructure initiale et d'assurer que les configurations de logiciels sur ce style d'infrastructure sont correctement mises en place et maintenues au fil du temps. Pour conclure ce point cette combinaison apporte une automatisation puissante, une plus grande reproductibilité des environnements et une réduction significative des erreurs types humain.

2. Processus de déploiement de l'infrastructure

Le processus de déploiement à l'aide de Terraform est réalisé depuis un machine virtuelle Linux sous Debian 12. La première étape consiste à installer les outils Terraform, aws-cli et Ansible :

Liens vers les documentations d'installation
Terraform
aws-cli
Ansible

Il est ensuite nécessaire de récupérer les identifiants aws pour permettre à Terraform d'envoyer des requêtes à l'API d'AWS afin de réaliser les demandes de création de ressources. Il faut déposer ces identifiants dans un fichier qui se situe dans le répertoire utilisateur qui exécute les commandes Terraform. Le contenu de ce fichier /home/user/.aws/credentials ressemble à ceci :

```
[default]
aws_access_key_id=ASIAYS2NTEPFIUHY7CFI
aws_secret_access_key=lynAtYm6Qpu9yBy8
aws_session_token=FwoGZXIvYXdzEPH/////
```

Une fois ces deux prérequis remplis, il est possible de passer à la préparation du projet Terraform.

- Récupérer le projet Terraform sur la machine qui possède le paquet Terraform.

- Exécuter la commande suivante pour télécharger les dépendances Terraform :

```
terraform init
```

- Exécuter la commande suivante pour préparer le déploiement de l'infrastructure et vérifier l'état actuel de l'environnement :

```
terraform plan
```

```
Plan: 52 to add, 0 to change, 0 to destroy.
```

- Si la commande précédente s'est déroulée sans erreur et que les identifiants AWS sont bons, exécuter la commande suivante pour lancer la création de l'infrastructure (saisir 'yes' lorsque cela est demandé) :

```
terraform apply
```

```
Apply complete! Resources: 52 added, 0 changed, 0 destroyed.
```

Suite à la création de l'infrastructure, la création de plusieurs fichiers va avoir lieu sur la machine hôte :

- Clé privée SSH : 'aws-sshkey.pem' qui est à utiliser pour toutes les connexions aux différentes instances via le SSH
- Fichier inventaire Ansible : 'inventory' qui est généré de façon dynamique avec les adresses IP privées des instances cibles d'Ansible
- Fichier de configuration de Nginx : 'reverse-config' qui contient la configuration de Nginx pour rediriger vers les bonnes entreprises avec les adresses IP des instances de façon dynamique

Une fois la construction de l'infrastructure AWS faite, il faut installer manuellement le serveur OpenVPN :

- L'administrateur se connecte sur l'instance du serveur VPN et utilisera un script afin d'installer OpenVPN :

```
wget https://git.io/vpn -O openvpn-install.sh
sudo bash openvpn-install.sh
```

- Répondre aux différentes questions pour configurer le serveur

- Il faut ensuite copier le fichier de configuration client « client.ovpn » dans le home directory de l'utilisateur (dans cet exemple, on utilise admin)

```
sudo cp /root/client.ovpn /home/admin
```

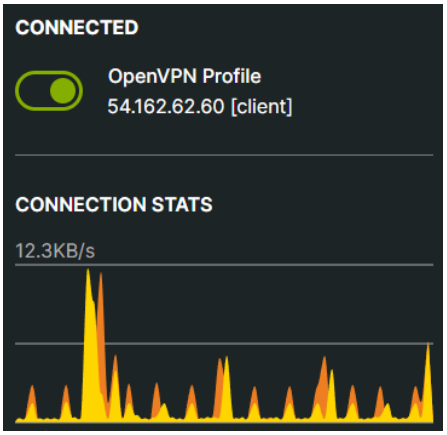
- Modifier les droits d'accès au fichier pour pouvoir le récupérer ensuite sur notre machine d'administration :

```
sudo chown admin:admin /home/admin/client.ovpn
```

- Récupérer ensuite le fichier de configuration (par exemple avec scp) :

```
scp -i aws-sshkey.pem admin@54.162.62.60:/home/admin/client.ovpn .
```

- Il faut ensuite se connecter à l'aide d'un client, par exemple celui de Windows accessible à partir de ce lien



Une fois connecté au VPN, il est possible de se connecter en SSH à la machine de rebond et ensuite de se connecter aux autres instances :

```
C:\Users\qllet\Documents>ssh -i aws-sshkey.pem admin@10.10.2.240
Linux ip-10-10-2-240 6.1.0-13-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Mar  7 08:47:06 2024 from 10.10.2.106
admin@ip-10-10-2-240:~$
```

La prochaine étape est l'utilisation d'Ansible pour déployer les différents services de la passerelle d'interconnexion et des instances des clients. Cette partie n'est pas opérationnelle en l'état car une procédure de déploiement n'a pas été réalisée. Il faudrait déposer le code Ansible avec les fichiers générés grâce à Terraform et les déposer sur la VM de Rebond puis installer ansible dessus et lancer les déploiements depuis celle-ci.

Cela n'a pas été testé en l'état et la configuration du proxy n'étant pas au point, il n'est pas possible d'installer des paquets depuis les instances des clients.

Pour lancer le déploiement via ansible, les commandes suivantes seraient à exécuter :

```
cd ansible/
ansible-playbook infra.yml
```

V. Utilisation

1. Le VPN

Il faut télécharger le fichier de configuration sur la machine client, la glisser-déposer dans le client OpenVPN du poste.

2. La machine d’administration

La machine d’administration est une machine sur laquelle nous allons nous connecter en SSH depuis le VPN afin de se connecter aux autres instances en SSH.

3. Les serveurs clients

Le client doit fournir une archive contenant les fichiers et le cahier des charges requis pour les services qu’il souhaite mettre en place.

VI. Stratégies de sécurité mises en place

Une passerelle d'interconnexion a été mise en place, cette dernière représente une zone stratégique qui agit en tant qu'interface entre les utilisateurs et les sites clients au sein de l'infrastructure. Cette passerelle offre une solution efficace pour administrer les serveurs de manière sécurisée, garantissant un contrôle rigoureux sur les flux de données. La segmentation des machines rebonds en deux subnets distincts renforce la sécurité en créant une séparation nette entre les instances destinées aux opérations d'administration et celles dédiées au filtrage du traffic des instances clients.

Cette architecture contribue à limiter l'accès aux ressources sensibles, améliorant ainsi la résilience du système. De manière stratégique, la passerelle expose sur Internet uniquement les instances Reverse Proxy et VPN. Cela permet de concentrer les points d'accès exposés, réduisant ainsi la surface d'attaque potentielle et augmentant la robustesse du système face aux menaces en ligne. En optant pour cette approche, la passerelle d'interconnexion offre un équilibre optimal entre sécurité, performance et gestion efficace des ressources réseau au sein de l'environnement informatique.

De plus, les instances clients sur notre plateforme bénéficient de l'étanchéité de son propre Virtual Private Cloud (VPC), assurant une isolation complète de ses ressources informatiques. Dans cette configuration, aucune instance client ne maintient une connexion directe à Internet ou à une autre instance client. La communication entre les instances clients et la zone de rebond s'effectue de manière sécurisée via un VPC Peering dédié pour chaque zone client.

Cette méthodologie offre une connectivité fiable et maîtrisée entre les environnements, tout en préservant l'intégrité des données. De plus, l'architecture mise en place est conçue pour évoluer en fonction des besoins spécifiques de chaque client. Cette flexibilité permet d'ajuster la configuration en réponse aux évolutions de la demande, garantissant ainsi une infrastructure informatique parfaitement adaptée aux exigences changeantes de nos utilisateurs.

VII. Stratégies de sécurité qui auraient pu être mises en place

Au sein de notre infrastructure, nous aurions pu mettre en place des mécanismes avancés visant à renforcer la sécurité et à assurer une surveillance proactive. Afin de prévenir toute modification non prévue, des actions automatisées pourraient être instaurées, notamment pour assurer la protection contre les modifications indésirables des groupes de sécurité. Ce dispositif utiliserait des services tels que AWS Config et Lambda pour mettre en œuvre des règles strictes qui empêcherait toute altération non autorisée.

Parallèlement, une surveillance étroite de la zone passerelle et des zones clients pourrait être configurée pour détecter toute activité suspecte. En cas de multiples tentatives de connexion infructueuses sur un compte d'administration, des alertes seraient alors déclenchées et une notification par mail serait envoyée pour prévenir les administrateurs de la plateforme, assurant une réactivité immédiate face à toute tentative d'intrusion. La mise en place de ces alertes est possible grâce aux services intégrés à AWS tels que CloudTrail, CloudWatch, et SNS pour garantir une gestion proactive des événements de sécurité.

De plus l'envoi automatique de mails pourrait également être utilisé dans les cas où une charge trop importante serait à l'origine de lenteurs ou de coupure de services, l'utilisation des solutions telles que CloudWatch et SNS, permettrait une gestion proactive des ressources en anticipant tout pic inhabituel d'activité.

VIII. Stratégie d'évolutivité

Comme toute infrastructure, en fonction des besoins des évolutions constantes des technologies, il est nécessaire de prendre du recul sur l'évolution de son infrastructure.

Nous avons réfléchi à la mise en place de plusieurs solutions, voici une liste les regroupant :

Mise en place d'un système de backup :

Configurer la mise en place de backup pour sauvegarder les instances EC2 mises en place et des potentielles bases de données pour éviter toute perte de données.

Optimisation du stockage :

Optimisation du stockage avec l'utilisation de buckets S3 pour stocker des objets statiques pour les clients ou concernant l'infrastructure.

Mise en place d'un plan de récupération :

Développer un plan de récupération après un sinistre, ce plan inclura des stratégies de basculement vers d'autres zones de disponibilités et de reprise d'activité.

Automatiser des audits :

Automatiser des audits avec AWS Config afin de s'assurer que les règles de sécurité son appliquées comme on le souhaite et que des changements imprévues n'aient pas lieux. Utilisation d'Amazon Inspector pour détecter des vulnérabilités sur des instances de production

Mise en place d'un WAF :

La mise en place d'un WAF aurait pour objectif de rajouter une couche de sécurité à notre passerelle d'interconnexion, notamment au niveau du reverse proxy qui serait d'avantage renforcé contre les potentiels attaques comme le XSS, DDOS etc.

Mise en place d'un système d'équilibrage de charge :

Une étude doit être réalisée concernant la répartition de charge sur l'infrastructure car elle va très probablement être amenée à évoluer et les entreprises clientes demanderont de plus en plus de ressources au fur et à mesure qu'ils grandissent.

Il est important de bien gérer ces ressources dans le cloud car cela peut générer une augmentation des coûts de l'infrastructure, si nous ne faisons pas suffisamment attention, la quantité de ressource peut être insuffisante et cela peut amener à créer de la latence ou pire encore, des coupures des accès vers les instances des clients.

Aujourd'hui, nous avons décidé de ne pas mettre en place de redondance mais cela va devenir un point bloquant assez rapidement. C'est pourquoi nous avons réfléchi à la mise en place de solutions AWS comme les suivantes :

- Mise en place d'AWS Auto Scaling pour allouer la quantité de ressources nécessaires en fonction de la charge utilisateur (par exemple allouer plus de ressources la journée mais moins la nuit en fonction de la charge habituelle en fonction des horaires).
- Utilisation d'Elastic Load Balancing pour répartir la charge sur les différentes instances EC2, en combinaison avec AWS Auto Scaling pour s'assurer du bon fonctionnement de l'auto scaling avec un répartiteur de charge.
- Etudier une utilisation futur de conteneurs / orchestration de conteneurs avec Amazon ECS. Les conteneurs sont plus flexibles et permettent des déploiements plus rapide, ce qui est en lien avec les exigences des clients qui souhaite que les déploiements d'applications soient rapides.

Versioning du code de l’infrastructure :

Parce que nous utilisons des solutions comme Terraform et Ansible qui fonctionnent sous le format d’Infrastructure As Code, il est possible de versionner les différents fichiers afin de pouvoir suivre les évolutions d’architecture grâce à des outils comme Github afin d’amener des améliorations ou avoir la possibilité de faire des rollbacks si l’on rencontre des problèmes lors des déploiements.

Annexe

Fichiers de déploiements :

Liste des fichiers présents dans le répertoire aws-infra/ du livrable :

ansible/

| **roles/**

| | **nginx/**

| | | **files/**

| | | | *reverse-config (généré avec Terraform)*

| | **squid/**

| | **clients/**

| | | **tasks/**

| | | | main.yml

| | | **templates/**

| | | | index.html.j2

| ansible.cfg

| infra.yml

| *inventory (généré avec Terraform)*

ansible.tf

aws-sshkey.pem (généré avec Terraform)

gateways.tf

instances.tf

providers.tf

routetables.tf

securitygroups.tf

sshkey.tf

vars.tf

vpc.tf