Elodie LEFEVRE

RAPPORT DE STAGE

Stage Cloud (AWS)

Du 02/01/2023 au 03/02/2023

Étudiante en 2ème année de BTS Sio option SISR au sein du
CNED de Poitiers

believe®

Table des matières

Introduction 3

Présentation de l'entreprise Believe 5

AWS (Amazon Web Services) 7

Terraform 13

Conclusion 18

Compétences visées pendant le stage de deuxième année

Gérer le patrimoine informatique.....

Répondre aux incidents et aux demandes d'assistance et d'évolution.....

Travailler en mode projet.....

Mettre à disposition des utilisateurs un service informatique

Organiser son développement professionnel.....

Introduction

J'ai fait mon stage dans la société Believe qui est une entreprise musicale située au 24 rue Toulouse Lautrec à Paris 75017. Je faisais partie du service cloud pour découvrir AWS, Kubernetes, Docker, Terraform. Au cours de mon stage, j'ai pu découvrir les divers services AWS et mettre en place une petite architecture réseau grâce aux outils mis à ma disposition.

Mon sujet était de mettre en œuvre une démonstration de plateforme télémétrie dans un environnement Kubernetes.

Mes missions étaient :

- Constitution d'un environnement de « Proof-of-concept » dans un environnement de « playground ».
- Déploiement d'un cluster EKS à partir des automatisations développées par l'équipe Cloud l'Ingénierie Infrastructure (Terraform).
- Déploiement et/ou intégration avec les services périphériques du modèle standard.
- Déploiement des agents et services AWS pour OpenTelemetry dans le cluster de test.

Le déploiement sera validé par la collecte des métriques/traces des nœuds du cluster et des composants déployés.

Validation du modèle :

- Choix d'une application « témoin » permettant de valider le déploiement (contraintes à définir).
- Création d'un jeu de Dashboard pour démontrer la corrélation traces-métriques.
- Démonstration de l'intérêt du modèle dans un contexte multi-instance de l'application.

Mon équipe était constitué d'un manager, d'un Principal Infrastructure Engineering, Senior DevOPS Architect, Lead DevOPS Engineer.

Believe est une multinational qui se situe au 24 rue Toulouse Lautrec, 75017 Paris.

Believe est implanté dans plusieurs pays comme Paris, Inde, Canada ainsi que pleins d'autres. Believe utilise un réseau du type hybride car ils utilisent des datacenters pour stocker les données de Believe puis de l'AWS pour stocker une autre partie des données. Il y a une équipe qui s'occupe uniquement des datacenters qui s'appelle l'équipe On-premise qui installent les logiciels/matériels nécessaires dans une salle serveurs.

VPN / Firewall
NAS
SAN

_____
_____
_____
_____
_____

Services

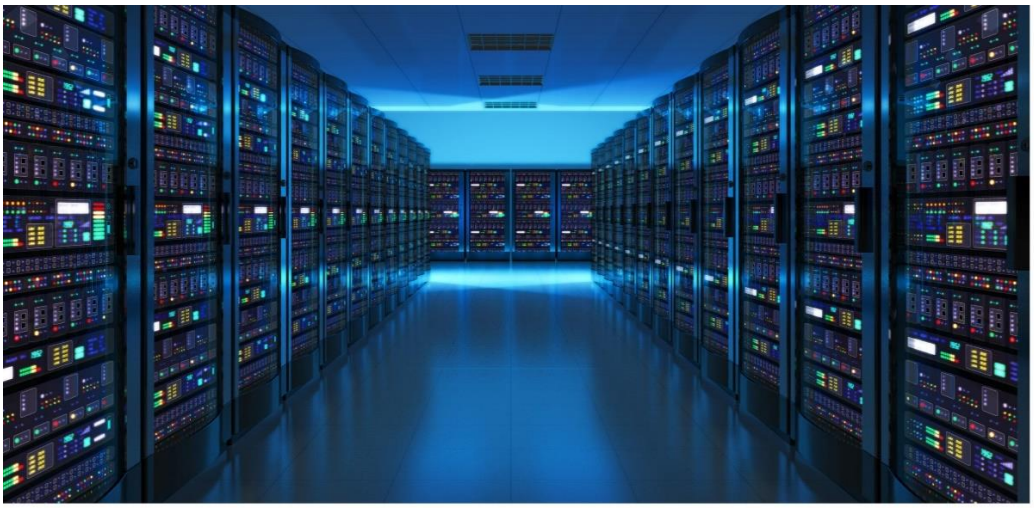
Network
Physique
ESX

Service

DNS, NTP, Métrique,
Proxy, Virtualisation, Log
Base de données, Supervision, CI/CD
Et encore pleins d'autres…

<u>Comment l'entreprise choisie où les données doivent être stockées ?</u>

Pour choisir où doit être stocké les données entre AWS et les datacenters ce n'est pas qu'une question de coût mais de rentabilité au fil des années. Il faut prendre en compte plusieurs critères sur les données si ce sont des données confidentielles ou non. Il y a une expertise à faire en fonction des besoins de l'entreprise afin de choisir l'emplacement des données.



Présentation de l'entreprise Believe

Believe est une entreprise française spécialisée dans l'accompagnement des artistes et labels. Elle possède plusieurs marques de distribution et des labels dont TuneCore, Groove Attack, Believe Distribution, AllPoints, Naïve, et Nuclear Blast.

La société est fondée en 2005 par Denis Ladegaillerie, Arnaud Chiaramonti et Nicolas Laclias.

En 2010, Believe Recordings signe son premier artiste.

Believe acquiert en 2015 TuneCore, service de distribution pour les artistes indépendants. L'extension de Believe a été alimentée par un investissement en capital de 60 millions de dollars, par Ventech, Technology Crossover Ventures (TCV) et XAnge.

En août 2016, la société rachète le label indépendant français Naïve Records pour un montant de 10 millions d'euros, cherchant ainsi à améliorer son catalogue, et relance en 2017 l'édition de nouveaux enregistrements sous forme de CD physiques par le label.

En 2017, Believe Recordings devient AllPoints, et développe sa branche production avec le lancement de trois labels : All Points, Naïve et Animal 63.

En septembre 2018, Believe acquièrent 49 % des parts du label indépendant français tôt ou tard auprès de Wagram Music.

En octobre 2018, Believe devient actionnaire majoritaire du label allemand Nuclear Blast.

L'entreprise s'est lancée dans une dynamique d'expansion en développant des marchés spécialisés en musique digitale telle que la Russie et l'Inde.

En 2019, Believe rachète Entco, spécialiste de la production d'événements en direct à Mumbai, et rebaptise la société « Believe Entertainment ».

Interviewé début octobre 2020, le PDG Denis Ladegaillerie annonce que Believe table sur une augmentation de 25 % de son chiffre d'affaires sur l'année. Pas affectée par la crise, l'entreprise Believe doit sa bonne santé à l'arrêt des concerts qui a ramené les fans de musique vers ses plateformes.

La société a été introduite en bourse le 10 juin 2021 sur Euronext Paris.

En 2021, Believe a pris une participation de 25% dans Play Two, premier label de musique indépendant en France et filiale du groupe TF1

En 2022, alors que les autres majors ont quitté le marché russe à la suite de l'invasion de l'Ukraine, elle y maintient ses activités.

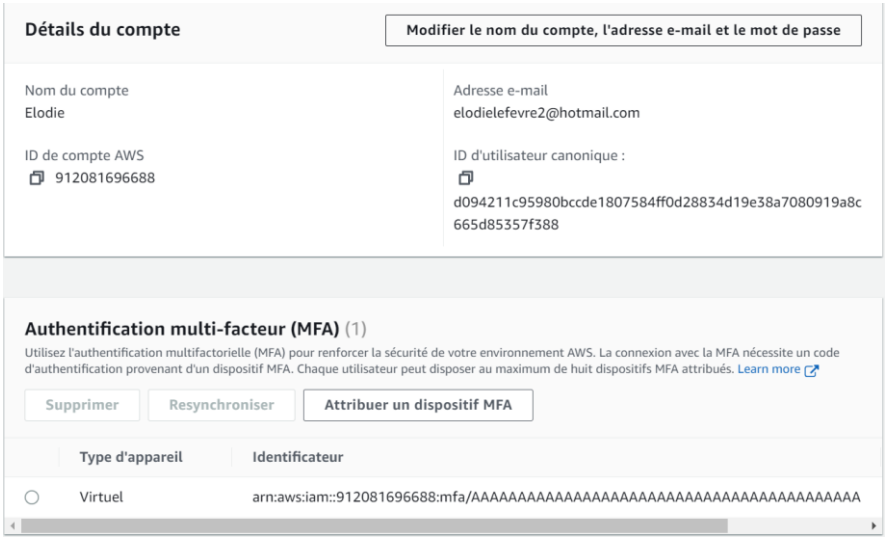
AWS (Amazon Web Services)

Pour comprendre le principe d'AWS je me suis beaucoup renseigné dessus et j'ai regardé des vidéos pour comprendre le principe afin de le manipuler et ensuite je me suis créé un compte dessus pour commencer à manipuler doucement avec les services gratuits qui mettent à disposition.

J'ai commencé par mettre en place l'authentification à multi-facteur (MFA) sur mon compte personnel AWS.

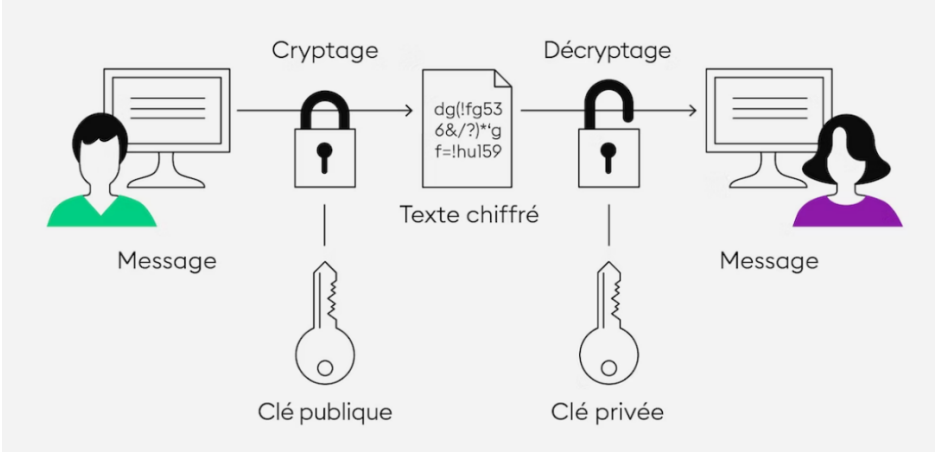
- J'ai choisi l'authentification par application.

L'authentification à multi-facteur sert à sécuriser notre compte AWS et nos données qu'on stocke à l'intérieur.



Le chiffrement utilisé est RSA qui est un cryptage asymétrique.

- Le chiffrement asymétrique utilise un ensemble de deux clés : une clé publique pour le chiffrement et une clé privée pour le déchiffrement, que seule une partie connaît.

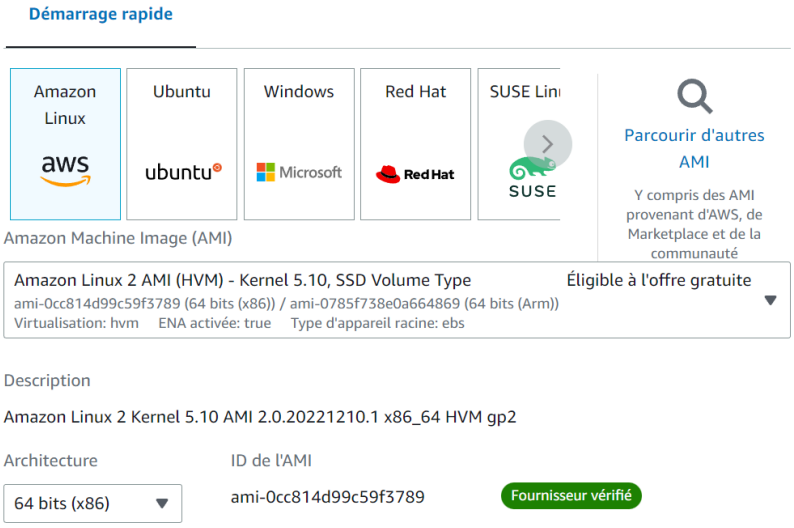


- La clé est chiffrée (par conséquent protégée) au moyen de la clé publique du destinataire. Celle-ci est

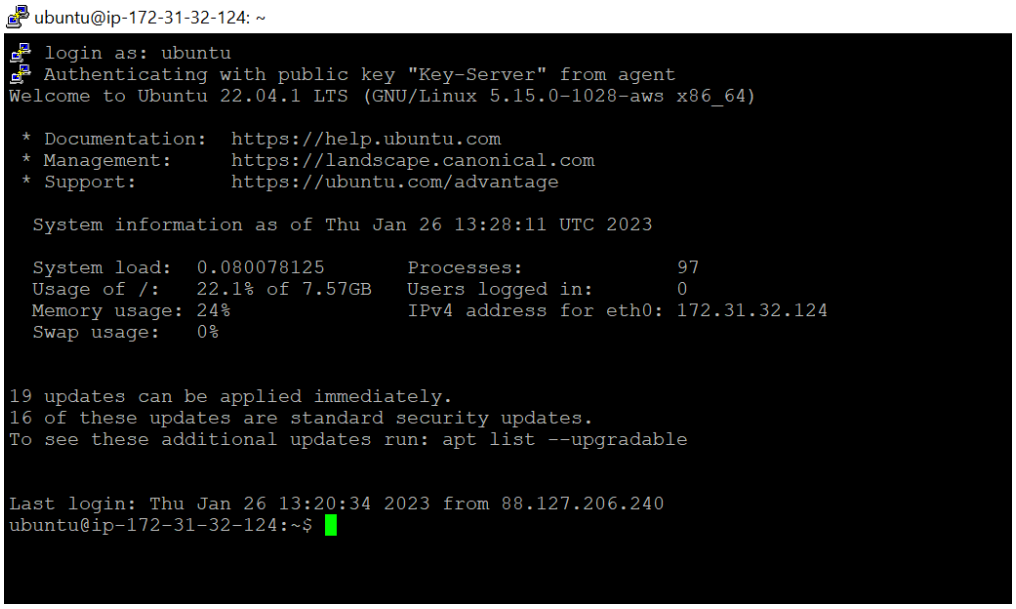
publique, donc il n'est pas nécessaire de la transmettre, l'expéditeur peut la récupérer facilement. Le destinataire va pouvoir déchiffrer la clé à l'aide de sa clé privée.

Démarrage d'une machine virtuelle ou instances qui s'exécutent sur le Cloud.

- Il existe différentes instances de plusieurs tailles chacune qui ont différentes capacités à offrir. Dans mon cas, j'ai choisi un serveur t2.micro parce qu'il est éligible à l'offre gratuite mais aussi suffisante pour mon usage et j'ai choisi l'OS Ubuntu.
- Il faut choisir une paire de clés avec un chiffrement dessus afin de se connecter à notre serveur en SSH avec un format de clés pour se connecter via PuTTY.



Pour me connecter à mon instance j'utilise PuTTY et j'entre mon nom d'utilisateur et il procède à un échange de clés pour établir une connexion sécurisée.

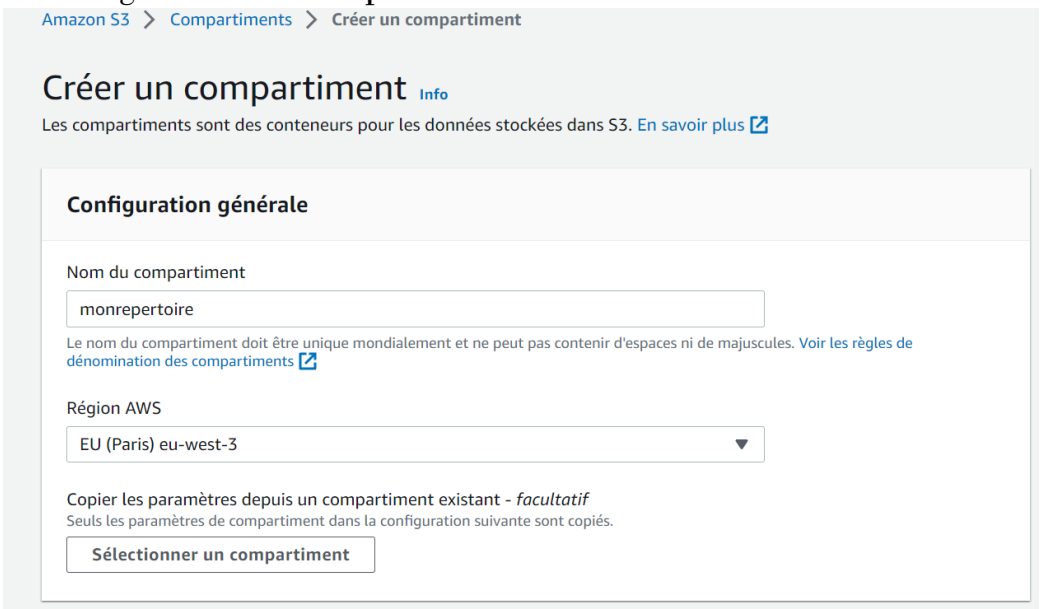


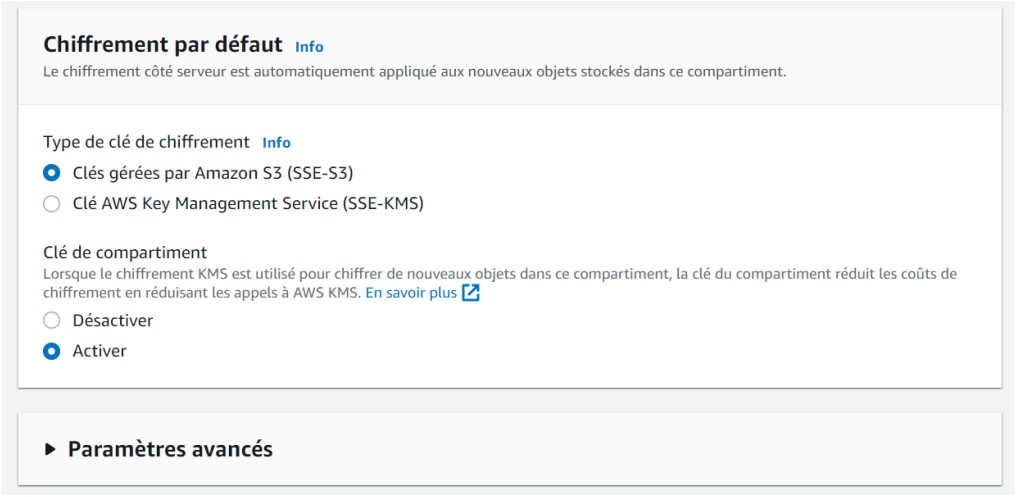
Pour stocker mes données j'utilise un bucket Amazon S3 qui est un service de stockage à travers des services web.

- J'ai choisi un nom de compartiment, la région que j'ai définie au début ainsi que le type de clé de chiffrement.
- Ce répertoire me permettra d'y mettre des fichiers et de les consulter

à partir des EC2 si nécessaire.

- Dans mon bucket S3 après sa création, il est possible de transférer des fichiers/dossiers à l'intérieur. Je peux les afficher, les télécharger voir même copier l'URL S3.





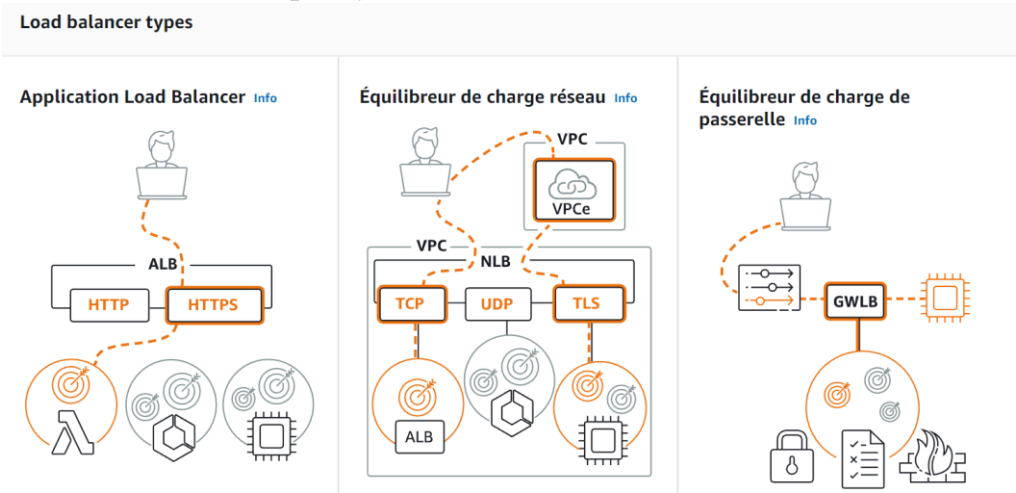
Fichiers et dossiers (7 Total, 462.9 Ko)

Rechercher par nom

1

Nom	Dossier	Type	Taille	Statut	Erreur
Amazon S3.PNG	-	image/png	66.0 Ko	Opération réussie	-
Clé de chiffrement S3.PNG	-	image/png	52.2 Ko	Opération réussie	-
Connexion sur l'instance EC2.PNG	-	image/png	38.8 Ko	Opération réussie	-
Démarrage d'un EC2.PNG	-	image/png	90.9 Ko	Opération réussie	-
MFA modifié.png	-	image/png	79.0 Ko	Opération réussie	-
MFA.PNG	-	image/png	84.5 Ko	Opération réussie	-
Mon instance en cours.PNG	-	image/png	51.4 Ko	Opération réussie	-

J'ai créé un Load balancer « Répartition des charges » en français qui permet d'adapter automatiquement la capacité de l'équilibreur de charge en fonction du trafic entrant qu'il y a.



<u>Les différents Load balancer proposés :</u>

<u>*Application Load Balancer*</u>

- Choisissez un équilibreur de charge d'application lorsque vous avez besoin d'un ensemble de fonctionnalités flexibles pour vos applications avec un trafic HTTP et HTTPS. Fonctionnant au niveau de la demande, les équilibreurs de charge d'application fournissent des fonctionnalités avancées

de routage et de visibilité ciblées sur les architectures d'application, y compris les micro-services et les conteneurs.

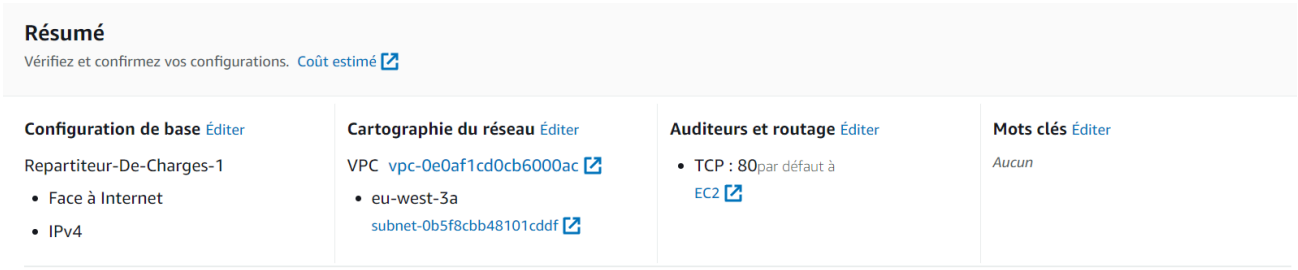
Équilibreur de charges réseau

- Choisissez un Network Load balancer lorsque vous avez besoin de performances ultra-élevées, d'un déchargement TLS à grande échelle, d'un déploiement centralisé de certificats, de la prise en charge d'UDP et d'adresses IP statiques pour vos applications. Fonctionnant au niveau de la connexion, les Network Load Balancers sont capables de gérer des millions de requêtes par seconde en toute sécurité tout en maintenant des latences ultra-faibles.

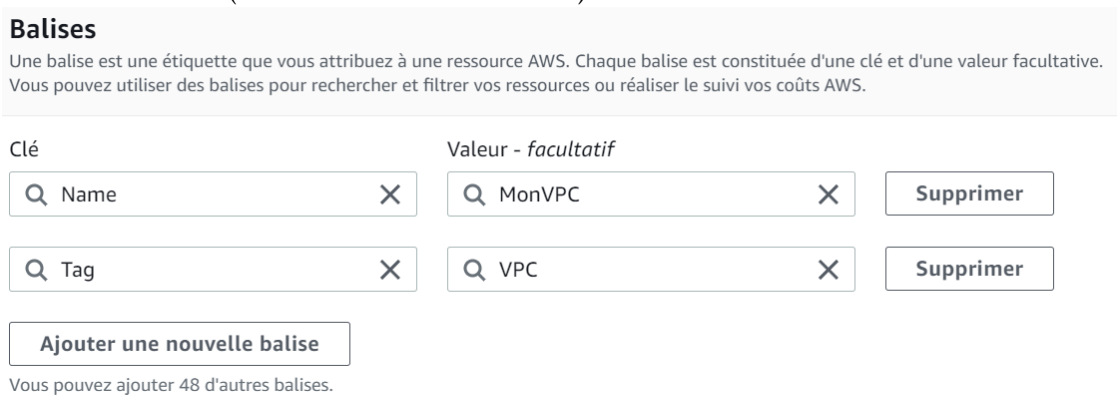
Équilibreur de charge de passerelle

- Choisissez un équilibreur de charge de passerelle lorsque vous devez déployer et gérer une flotte d'appliances virtuelles tierces qui prend en charge GENEVE. Ces appliances vous permettent d'améliorer la sécurité, la conformité et les contrôles des politiques.

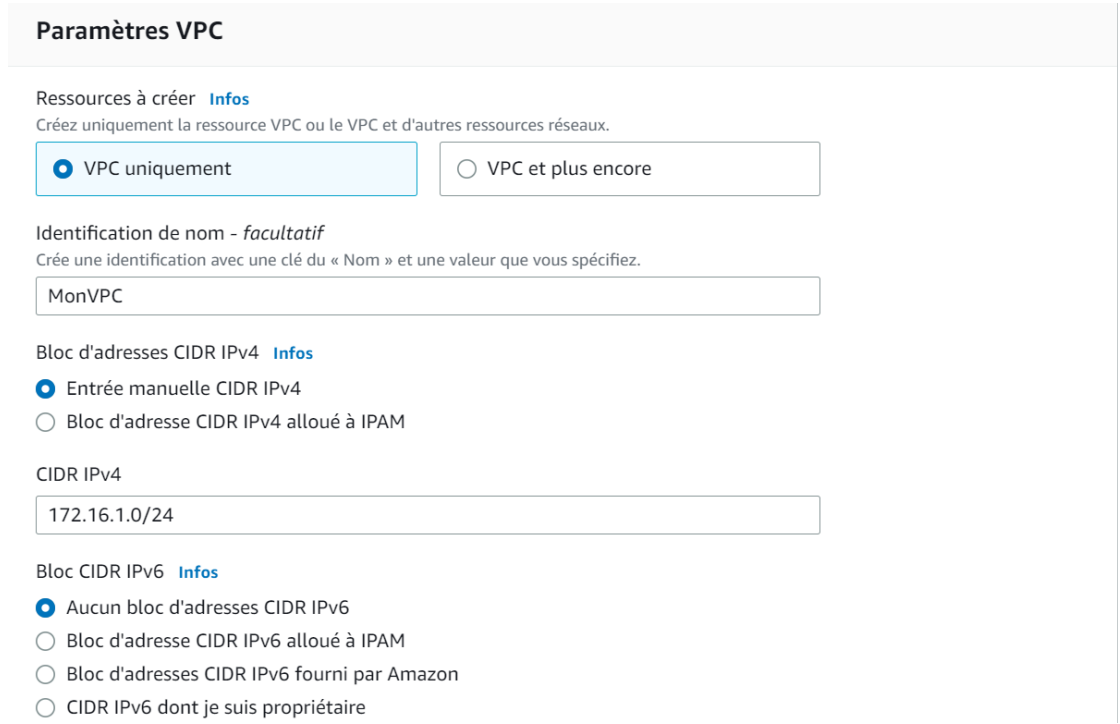
J'ai choisi l'équilibreur de charge du réseau afin de distribuer le trafic TCP et UDP entrantes sur les instances EC2.

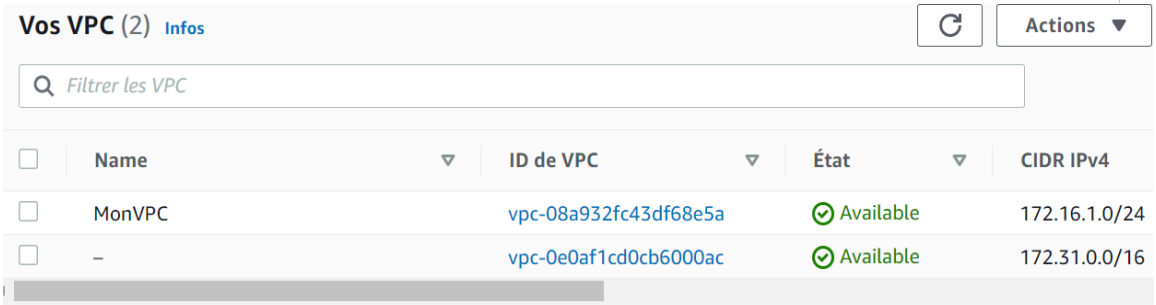


Création VPC (Virtual Private Cloud)



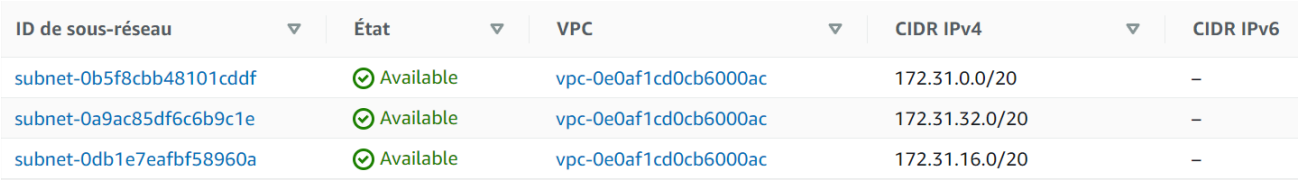
- Une balise est une étiquette qu'on attribue à une ressource AWS. Chaque balise est constituée d'une clé et d'une valeur facultatives. Les balises sont utilisables aussi pour rechercher et filtrer les ressources.





- Une VPC (Virtual Private Cloud) est une partie du réseau au sein d'Amazon Web Service isolée du réseau traditionnel et des réseaux des autres clients d'AWS. Il est possible aussi de créer notre propre plage d'adresses IP et sous
Réseaux.

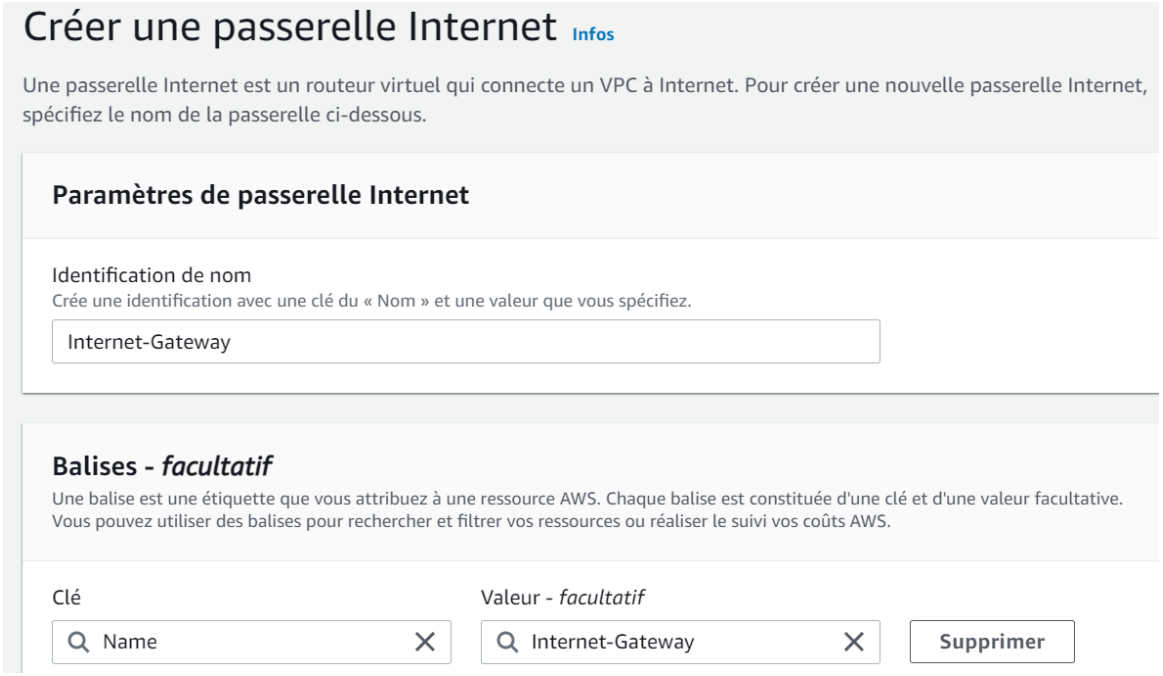
Subnet (Sous-réseaux)



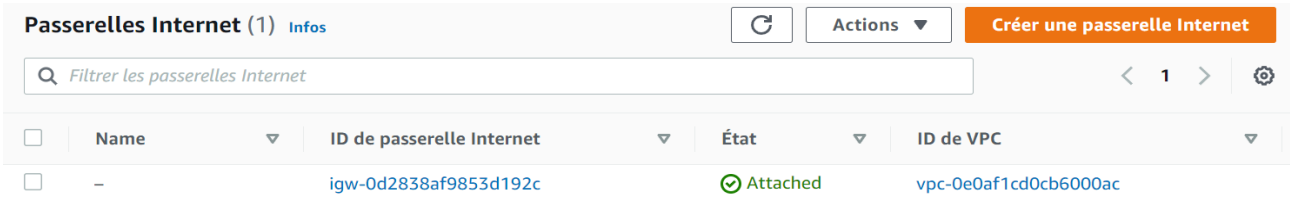
- Il est possible de choisir le nombre d'adresses IPv4 qui sont disponibles en fonction de la plage d'adresses IP choisi.

Adresses IPv4 disponibles	Zone de disponibilité	ID de zone de disponi...	Table de routage
4090	eu-west-3a	euw3-az1	rtb-012a88fd002b3065d
4091	eu-west-3c	euw3-az3	rtb-012a88fd002b3065d
4091	eu-west-3b	euw3-az2	rtb-012a88fd002b3065d

Création d'une Internet Gateway



- Une passerelle Internet est un routeur virtuel qui connecte une VPC à Internet.



- Ma passerelle internet sert à fournir un accès internet à mon VPC.

Terraform

Pour comprendre le principe de Terraform, le principe est identique à celui d'AWS. Il était beaucoup plus simple pour moi d'avoir déjà une vision avec des vidéos au sujet du travail qui m'attendait que de commencer sans aucune compétence.

Terraform est un environnement logiciel « d'infrastructure as code » (IaaC) publié en open source par la société Hashi Corp. Cet outil permet d'automatiser la construction des ressources d'une infrastructure de centre de données comme un réseau, des machines virtuelles, un groupe de sécurité ou une base de données.

La première partie je voulais simplement afficher un « Hello world » donc j'utilise le module output de Terraform avec le nom d'une variable que je choisis et une valeur qui sera affichée quand je vais appliquer.

```
  projet_Terraform ls
hosts.txt  main.tf  projetTerraform2  terraform.tfstate  terraform.tfstate.backup
  projet_Terraform cd projetTerraform2
  projetTerraform2 terraform apply

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

Outputs:

mavariable = Hello world !
  projetTerraform2
```

Les différentes variables :

String

```
variable "str" {
  type = string
  default = "127.0.0.1 gitlab.test"
}
resource "null_resource" "premiereVariable" {
 provisioner "local-exec" {
  command = "echo '${var.str}' > hosts.txt"
 }
}

output "str" {
 value = var.str
}
```

```
Outputs:

str = 127.0.0.1 gitlab.test
```

La variable Map possède des paramètres à rentrer avec un système de clés et de valeur. Pour parcourir une map on utilise la boucle foreach en lui spécifiant la variable qu'on a saisie et ça va lui permettre de savoir qu'il va avoir une map à parcourir et cette map est composée de clé et de valeur. Le foreach va être appliqué au provisioner qui est local-exec qui va ensuite récupérer les clés et les valeurs et qui vont être ajouter au fichier hosts.txt .

```
variable "map" {
  default = {
    "127.0.0.1" = "localhost gitlab.local"
    "192.168.1.168" = gitlab.test
    "192.169.1.169" = essai.test"
  }
}
ressource "null_resource" "map" {
 for_each = var.hosts
 provisioner "local-exec" {
  command = "echo '${each.key} ${each.value}' >> hosts.txt"
 }
}
```

J'ai fait un Terraform plan pour planifier mes changements, je peux voir mes 3 ressources qui vont être ajoutées. Il suffit de taper Terraform apply pour appliquer les changements.

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # null_resource.map["151.101.152.85"] will be created
  + resource "null_resource" "map" {
      + id = (known after apply)
    }

  # null_resource.map["51.70.126.155"] will be created
  + resource "null_resource" "map" {
      + id = (known after apply)
    }

  # null_resource.map["54.239.28.85"] will be created
  + resource "null_resource" "map" {
      + id = (known after apply)
    }

Plan: 3 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

  projet_Terraform
```

J'ai ajouté un trigger qui permet de voir les modifications de valeur si le foreach évolue. C'est en partie grâce au terraform.tfstate que c'est possible car il gère les états de nos ressources globales.

```
variable "map" {
  default = {
    "151.101.128.84" = "pinterest.fr gitlab.me gitlab.elodie"
    "52.94.236.248" = "amazon.com"
    "142.250.179.206" = "google.com"
  }
}
resource "null_resource" "map" {
 for_each = var.map
 triggers = {
   foo = each.value
}
 provisioner "local-exec" {
  command = "echo '${each.key} ${each.value}' >> hosts.txt"
 }
}
```

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # null_resource.map["142.250.179.206"] must be replaced
-/+ resource "null_resource" "map" {
      ~ id       = "2263830543942430963" -> (known after apply)
      + triggers = {
          + "foo" = "google.com"
        } # forces replacement
    }

  # null_resource.map["151.101.128.84"] must be replaced
-/+ resource "null_resource" "map" {
      ~ id       = "830045773079457038" -> (known after apply)
      + triggers = {
          + "foo" = "pinterest.fr gitlab.me gitlab.elodie"
        } # forces replacement
    }

  # null_resource.map["52.94.236.248"] must be replaced
-/+ resource "null_resource" "map" {
      ~ id       = "6381386707278938619" -> (known after apply)
      + triggers = {
          + "foo" = "amazon.com"
        } # forces replacement
    }

Plan: 3 to add, 0 to change, 3 to destroy.
```

La variable List se déclare comme un élément string, il faut déclarer le default entre crochets avec chacun des éléments. Le count doit être pris en compte car il sert d'itération qui est liée au nombre d'éléments qui se trouvent dans le default.

C'est-à-dire, au count on va lui passer la valeur de la longueur de la liste (variable « list »).

Pour chaque var.list, on récupère la valeur à l'index correspondant (index 0 et index 1) donc on utilise la fonction élément qui prend la valeur de la liste.

```
variable "list" {
   default = ["127.0.0.1 localhost","192.168.1.133 gitlab.test"]
}
resource "null_resource" "hosts" {
 count = "${length(var.list)}"
 provisioner "local-exec" {
  command = "echo '${element(var.list, count.index)}' >> hosts.txt"
 }
}
```

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # null_resource.hosts[0] will be created
  + resource "null_resource" "hosts" {
      + id = (known after apply)
    }

  # null_resource.hosts[1] will be created
  + resource "null_resource" "hosts" {
      + id = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.
```

Conclusion

Au cours de mon stage, j'ai pu découvrir AWS avec ses différents services que j'ai pu mettre en place pour rédiger mon rapport de stage. J'ai entre autres pus m'initier à l'utilisation de k8s (Kubernetes) et Docker pour manipuler des containers.

Pendant mon stage chez Believe, j'ai pu apprendre AWS avec les différents outils qui était à ma disposition comme : S3, Bucket S3, EC2, CIDR_BLOCK, Nat Gateway, Elastic Load Balancer.

Le stage m'a permis d'acquérir de nouvelles connaissances et d'obtenir une alternance dans le service réseau. Les points forts de mon stage sont l'esprit d'équipe chez Believe, j'ai su m'adapter très rapidement à l'environnement d'AWS ainsi qu'à mon équipe Cloud, j'ai su travailler en autonomie totale. Pendant mon stage il n'y a pas eu de point faible car l'équipe Cloud était constamment présente et je pouvais compter sur mes collègues de travail pour m'apporter leur aide quand j'en avais besoin.

Remerciements

Je tiens à remercier Monsieur Paco FALL pour avoir fait les premières démarches pour mon stage.

Merci aussi à Monsieur Emmanuel Paccoud de m'avoir accueilli à son service et suivi mon stage.

Ensuite, je veux remercier l'équipe du Cloud avec qui j'étais pour m'accompagner tout au long de mon stage et m'aider quand j'en avais besoin.