Rapport sur le Déploiement d'une Architecture Cloud dans Amazon AWS

Table des matières

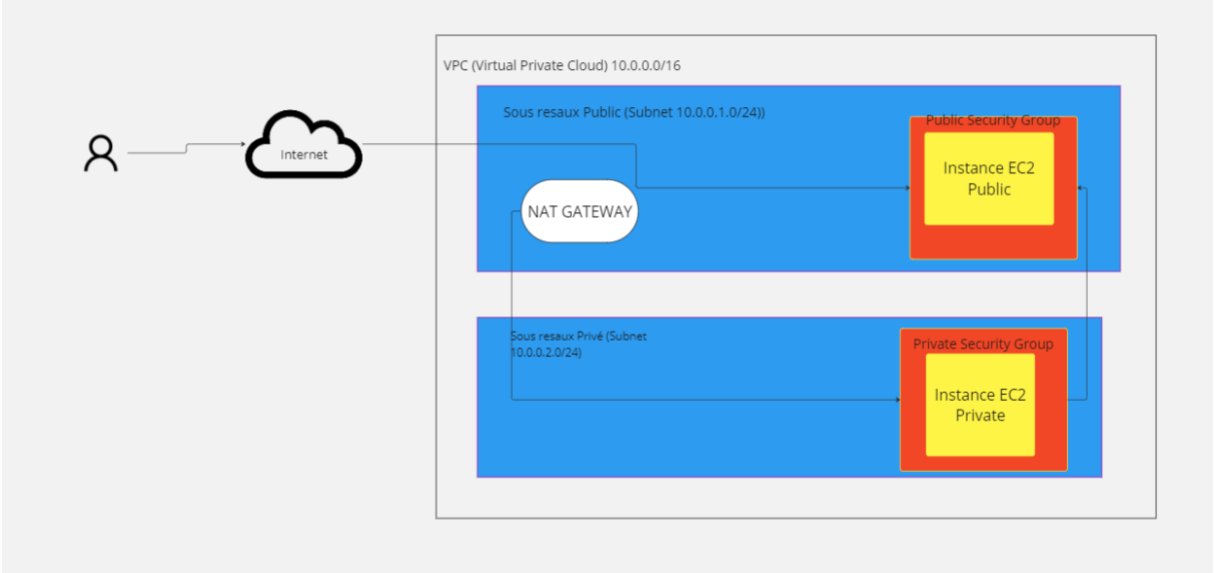
1. Objectifs

L'objectif principal du projet était de déployer une infrastructure cloud dans un environnement de développement pour le nouveau client, en utilisant les services d'Amazon AWS.

Les exigences spécifiques du client étaient les suivantes :

- Configuration d'1 VPC avec 2 subnets : un public et un privé.
- Installation d'1 instance dans chaque subnet.
- Autorisation de la communication entre les instances via le port 22 uniquement.
- Accessibilité de l'instance du VPC public depuis Internet sur le port 22 et un autre port pour l'application de test.
- Sécurisation de l'instance dans le subnet privé sans adresse IP publique.
- Utilisation des ACLs et des groupes de sécurité pour assurer la sécurité de l'architecture

Schéma de l'infrastructure :



2. Démarche du déploiement de notre infrastructure AWS

2.1. Création de l'infrastructure

A noter que tout les textes en vert correspondent a des ID propre a ce TP. Il faudra prendre en compte la modification de celles-ci.

Création VPC

Nous allons commencer par créer un VPC qui est un espace réseau virtuel personnel et privé hébergé dans un environnement de cloud public., pour cela et pour la suite des démarches nous allons utiliser le CloudShell proposé par AWS :

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --query Vpc.VpcId --output text
```

Un ID de VPC va nous être fourni des lors ou le VPC est créé, l'ID est propre a ce VPC, il faut le garder précieusement afin de mieux identifier le VPC, exemple lorsque que l'on veut renommer celui-ci :

```
vpc-0fb3f11ec680d7e02
aws ec2 create-tags --resources vpc-0fb3f11ec680d7e02 --tags
Key=Name,Value=VPC_Djame_EVAL
```

Création SubNet

Ensuite, nous avons créé deux subnets, un public et un privé, avec ces plages respectifs 10.0.1.0/24 et 10.0.2.0/24.

Creation subnet public:

```
aws ec2 create-subnet --vpc-id vpc-0fb3f11ec680d7e02 --cidr-block 10.0.1.0/24
--availability-zone us-east-2a --query Subnet.SubnetId --output text

aws ec2 create-tags --resources subnet-06e26c05a5952cfdd --tags
Key=Name,Value=Public_Subnet_eval
```

Creation subnet private:

```
aws ec2 create-subnet --vpc-id vpc-0fb3f11ec680d7e02 --cidr-block 10.0.2.0/24
--availability-zone us-east-2a --query Subnet.SubnetId --output text

aws ec2 create-tags --resources subnet-0260b50f6482917aa --tags
Key=Name,Value=Private_Subnet
```

Création et attachement au VPC de la Gateway

Nous allons créer une passerelle Internet dans un VPC, cette passerelle permet aux ressources hébergées dans le VPC d'accéder à Internet et vice versa

```
aws ec2 create-internet-gateway --query InternetGateway.InternetGatewayId --
output text
aws ec2 create-tags --resources igw-0e23c6436d9fb355e --tags
Key=Name,Value=djam-gateway-cli
```

Nous allons ensuire rattaché cette passere Internten a notre VPC

```
aws ec2 attach-internet-gateway --vpc-id vpc-0fb3f11ec680d7e02 --internet-
gateway-id igw-0e23c6436d9fb355e
```

Création et configurer la table de routage pour Internet Gateway et NAT Gateway :

Nous allons créer 2 tables de routage : Table de routage par defaut et Table de Routage NAT

Les tables de routage sur AWS sont des composants essentiels pour gérer le trafic réseau au sein de notre VPC. Elles déterminent comment le trafic réseau est dirigé entre les différentes ressources au sein du VPC et vers l'extérieur du VPC

```
aws ec2 create-route-table --vpc-id vpc-0fb3f11ec680d7e02 --query
RouteTable.RouteTableId --output text

aws ec2 create-tags --resources rtb-070164651c2ce0384 --tags
Key=Name,Value=Routable_djam
```

```
aws ec2 create-route-table --vpc-id vpc-0fb3f11ec680d7e02 --query
RouteTable.RouteTableId --output text
```

```
aws ec2 create-tags --resources rtb-04db62a5adc10727e --tags
Key=Name,Value=Routable_djam_nat
```

```
aws ec2 create-route --route-table-id rtb-070164651c2ce0384 --destination-
cidr-block 0.0.0.0/0 --gateway-id igw-0e23c6436d9fb355e
aws ec2 create-route --route-table-id rtb-04db62a5adc10727e--destination-cidr-
block 0.0.0.0/0 --nat-gateway-id nat-08676ff584c3a39a6
```

Associer les tables de routage au Subnet :

```
aws ec2 associate-route-table --route-table-id rtb-070164651c2ce0384 --subnet-
id subnet-06e26c05a5952cfdd

aws ec2 associate-route-table --route-table-id rtb-04db62a5adc10727e --subnet-
id subnet-0260b50f6482917aa
```

Création de groupe de sécurité

Les groupes de sécurité sont un mécanisme de sécurité essentiel dans Amazon Web Services (AWS). Ils servent à contrôler le trafic réseau entrant et sortant des instances EC2 ,ainsi que d'autres ressources réseau comme les instances RDS les load balancers.

```
aws ec2 create-security-group --group-name SG-Eval --description "Groupe de
securite cli" --vpc-id vpc-0fb3f11ec680d7e02
```

Creation Nat Gateway et IP Elastic

```
aws ec2 allocate-address --domain vpc --query 'AllocationId' --output text --
region us-east-2a --tag-specifications 'ResourceType=elastic-
ip,Tags=[{Key=Name,Value=IPElastic_Djam}]'
aws ec2 create-nat-gateway --subnet-id subnet-0260b50f6482917aa --allocation-
id eipalloc-04790fbec2febb298
```

Creation Instance Privé et publique

Public:

```
aws ec2 run-instances --image-id ami-0ec3d9efceafb89e0 --count 1 --instance-
type t2.micro --key-name VM_Docker_AWS_1 --security-group-ids sg-
016ad7ba11da09165 --subnet-id subnet-06e26c05a5952cfdd --associate-public-ip-
address --tag-specifications
'ResourceType=instance,Tags=[{Key=DebianServer,Value=Beta}]' --region us-east-
2
```

Privé:

```
aws ec2 run-instances --image-id ami-0ec3d9efceafb89e0 --count 1 --instance-
type t2.micro --key-name VM_Docker_AWS_1 --security-group-ids sg-
016ad7ba11da09165 --subnet-id subnet-0260b50f6482917aa --tag-specifications
```

```
'ResourceType=instance,Tags=[{Key=DebianServer,Value=Beta}]' --region us-east-
2
```

3. Connexion Aux Instances

Afin de vous Connecter Aux instances il vous faut vous munir de l'adresse Publique et de saisir la commande de connexon SSH :

ssh -i VM_Docker_AWS_1.pem admin@3.145.37.107

```
PS D:\Users\Admin\Downloads> ssh -i VM_Docker_AWS_1.pem admin@3.145.37.107
The authenticity of host '3.145.37.107 (3.145.37.107)' can't be established.
ECDSA key fingerprint is SHA256:dnDgtN8jff+RXHsfR0HY5HvS6/O8/0jvZvT1yKujaC4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.145.37.107' (ECDSA) to the list of known hosts.
Linux ip-10-0-1-215 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.
admin@ip-10-0-1-215:~$ cd .ssh\
```

Ensuite pour pouvoir se connecter à l'instance privé depuis l'instance publique :

Il va falloir vous placer dans le dossier

```
admin@ip-10-0-1-215:~$ cd .ssh\
```

Ajouter l'intégralité de votre private Key

```
admin@ip-10-0-1-215:~/.ssh$ nano VM_Docker_AWS_1
```

Un chmod 400 est necessaire pour pouvoir donner les droits a la clef

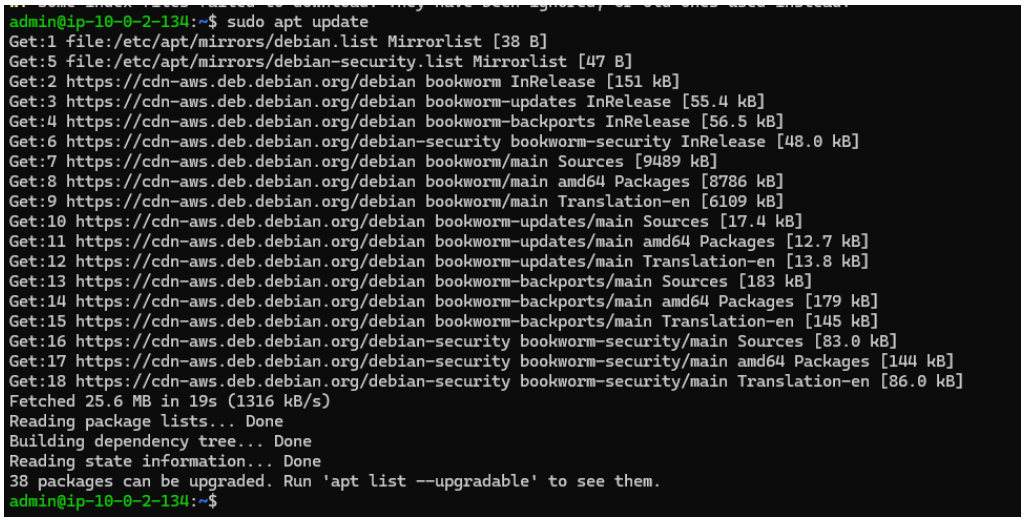
```
admin@ip-10-0-1-215:~/.ssh$ chmod 400 VM_Docker_AWS_1
```

```
admin@ip-10-0-1-215:~/.ssh$ ssh -i VM_Docker_AWS_1 admin@10.0.2.134
Linux ip-10-0-2-134 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.
admin@ip-10-0-2-134:~$ sudo apt update
```

Vous pouvez a présent vous connecter a votre VM et lancer des mis a jour


admin@ip-10-0-2-134:~$ sudo apt update
Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [38 B]
Get:5 file:/etc/apt/mirrors/debian-security.list Mirrorlist [47 B]
Get:2 https://cdn-aws.deb.debian.org/debian bookworm InRelease [151 kB]
Get:3 https://cdn-aws.deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:4 https://cdn-aws.deb.debian.org/debian bookworm-backports InRelease [56.5 kB]
Get:6 https://cdn-aws.deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:7 https://cdn-aws.deb.debian.org/debian bookworm/main Sources [9489 kB]
Get:8 https://cdn-aws.deb.debian.org/debian bookworm/main amd64 Packages [8786 kB]
Get:9 https://cdn-aws.deb.debian.org/debian bookworm/main Translation-en [6109 kB]
Get:10 https://cdn-aws.deb.debian.org/debian bookworm-updates/main Sources [17.4 kB]
Get:11 https://cdn-aws.deb.debian.org/debian bookworm-updates/main amd64 Packages [12.7 kB]
Get:12 https://cdn-aws.deb.debian.org/debian bookworm-updates/main Translation-en [13.8 kB]
Get:13 https://cdn-aws.deb.debian.org/debian bookworm-backports/main Sources [183 kB]
Get:14 https://cdn-aws.deb.debian.org/debian bookworm-backports/main amd64 Packages [179 kB]
Get:15 https://cdn-aws.deb.debian.org/debian bookworm-backports/main Translation-en [145 kB]
Get:16 https://cdn-aws.deb.debian.org/debian-security bookworm-security/main Sources [83.0 kB]
Get:17 https://cdn-aws.deb.debian.org/debian-security bookworm-security/main amd64 Packages [144 kB]
Get:18 https://cdn-aws.deb.debian.org/debian-security bookworm-security/main Translation-en [86.0 kB]
Fetched 25.6 MB in 19s (1316 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
38 packages can be upgraded. Run 'apt list --upgradable' to see them.
admin@ip-10-0-2-134:~$