

Lab 3: Réseau cloud, topologie VPC et filtrage du trafic

Cloud Computing

Université Claude Bernard Lyon 1 May 28, 2026

Assignment Overview

Document Contents:

1. Objectif	2
2. Pré-requis	2
2.1. Conseils	2
3. Contexte	2
4. Architecture cible	4
5. Étapes	5
5.1. Étape 1: Lire les fichiers Terraform	5
5.2. Étape 2: Initialiser le projet et déployer ..	5
5.3. Étape 3: Vérifier la topologie déployée ..	5
5.4. Étape 4: Analyser les security groups ...	6
5.5. Étape 5: Analyser les NACLs	6
5.6. Étape 6: Destruction des ressources	6
6. Questions	7
7. Aller plus loin	7

Assignment Details:

- **Course:** Cloud Computing
- **Instructor:** Hugo Blanc
- **Software:** Terraform, AWS, LocalStack
- **Duration:** ~ 1 heure
- **Lab Number:** Lab 3

Objectif

Construire une topologie réseau complète dans un VPC AWS : subnets publics et privés répartis sur deux zones de disponibilité, Internet Gateway, NAT Gateway, tables de routage, security groups chaînés et Network ACLs. À la fin du TP, vous comprendrez en profondeur comment le routage et le filtrage du trafic fonctionnent dans le cloud.

Pré-requis

- Avoir Terraform (>= 1.5.0);
- avoir tflocal (`pip3 install terraform-local`);
- avoir LocalStack qui tourne;
- avoir la CLI AWS (voir la documentation officielle).

Conseils

Je vous recommande de faire les alias suivants:

```
alias tf=tflocal
alias aws="aws --endpoint-url=http://localhost:4566"
```

Également, pour simplifier l'usage de la CLI AWS, je vous recommande de définir des faux credentials pour éviter les erreurs:

```
export AWS_ACCESS_KEY_ID=test
export AWS_SECRET_ACCESS_KEY=test
export AWS_DEFAULT_REGION=eu-west-1
```

Contexte

LocalStack émule des services AWS en local sur la machine, sur le port 4566. La commande `tflocal` est un wrapper autour de Terraform qui permet de rediriger les appels d'API vers LocalStack.

Dans une architecture cloud bien conçue, les ressources sont séparées en deux catégories :

- **Subnet public** : ressources qui doivent être joignables depuis Internet (load balancers, bastions). Ces subnets possèdent une route vers une Internet Gateway (IGW).
- **Subnet privé** : ressources qui ne doivent pas être directement accessibles depuis Internet (serveurs d'application, bases de données). Ces subnets n'ont pas de route directe vers l'IGW.

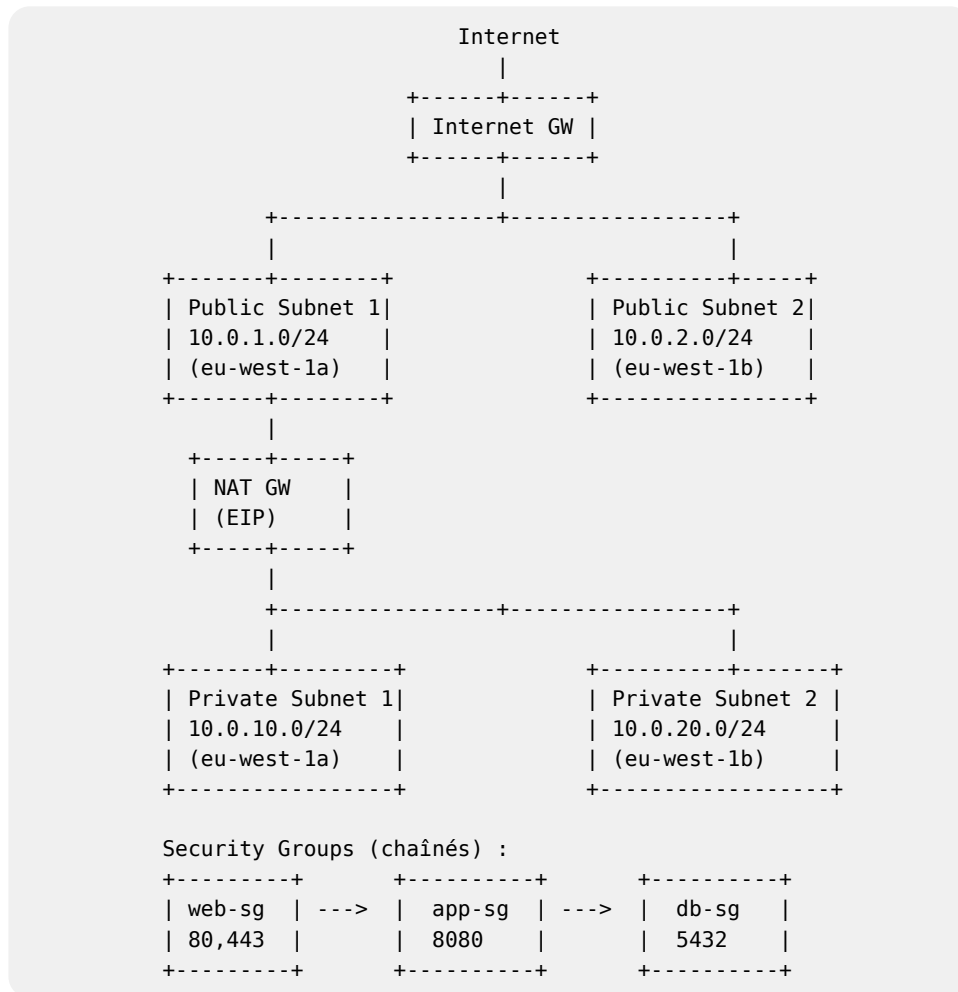
Les instances d'un subnet privé ont souvent besoin de joindre Internet (mises à jour, appels d'API externes) sans pour autant être joignables de l'extérieur. La **NAT Gateway** permet cela : elle traduit les adresses privées en une adresse publique pour le trafic sortant, et bloque le trafic entrant non sollicité. Elle est placée dans un subnet **public** (elle a besoin d'accéder à l'IGW), et les subnets privés routent leur trafic sortant à travers elle.

Enfin, deux mécanismes de filtrage coexistent :

Critère	Security Group	Network ACL
---------	----------------	-------------

Niveau	Instance (ENI)	Subnet
État	Stateful (retour automatique)	Stateless (règles explicites requises)
Règles	Allow uniquement	Allow et deny
Évaluation	Toutes les règles évaluées	Ordre de priorité
Défaut	Deny tout l'entrant	Allow tout

Architecture cible



Étapes

Étape 1: Lire les fichiers Terraform

Avant d'entrer des commandes, il vous faut lire attentivement chaque fichier .tf présent dans ce dossier.

Question

Identifiez :

- Comment le VPC est créé et quel bloc CIDR il utilise;
- comment les subnets sont distribués entre les zones de disponibilité;
- comment le routage dirige différemment le trafic des subnets publics et privés;
- comment les security groups forment une chaîne de confiance;
- comment les NACLs ajoutent une couche de filtrage supplémentaire.

Étape 2: Initialiser le projet et déployer

```
tflocal init
tflocal plan -var="student_name=your-name"
tflocal apply -var="student_name=your-name"
```

Question

Observez l'ordre dans lequel Terraform crée les ressources. Pourquoi la NAT Gateway est-elle créée après l'Internet Gateway et l'Elastic IP ?

Étape 3: Vérifier la topologie déployée

```
# Lister les VPC
aws ec2 describe-vpcs \
  --query 'Vpcs[*].{ID:VpcId,CIDR:CidrBlock}'

# Lister les subnets
aws ec2 describe-subnets \
  --query 'Subnets[*].{ID:SubnetId,CIDR:CidrBlock,AZ:AvailabilityZone}'

# Visualiser les tables de routage
aws ec2 describe-route-tables \
  --query 'RouteTables[*].{ID:RouteTableId,Routes:Routes}'
```

Question

Combien de tables de routage existent dans le VPC ? Quelle table est associée à quels subnets, et quelles routes par défaut chacune contient-elle ?

Étape 4: Analyser les security groups

```
# Règles du security group web
aws ec2 describe-security-groups \
  --group-names "your-name-web-sg" \
  --query 'SecurityGroups[*].IpPermissions'

# Règles du security group app
aws ec2 describe-security-groups \
  --group-names "your-name-app-sg" \
  --query 'SecurityGroups[*].IpPermissions'

# Règles du security group db
aws ec2 describe-security-groups \
  --group-names "your-name-db-sg" \
  --query 'SecurityGroups[*].IpPermissions'
```

Question

Le security group app n'autorise le port 8080 que depuis web. Comment cette référence est-elle exprimée dans la règle ? En quoi est-ce plus robuste que d'autoriser une plage d'IP du subnet public ?

Étape 5: Analyser les NACLs

```
aws ec2 describe-network-acls \
  --query 'NetworkAcls[*].{ID:NetworkAclId,Entries:Entries}'
```

Question

Pour la NACL privée, identifiez les règles d'entrée et de sortie. Pourquoi une règle d'entrée sur les ports éphémères (1024-65535) est-elle nécessaire alors qu'aucun service n'écoute sur ces ports ?

Étape 6: Destruction des ressources

```
tflocal destroy -var="student_name=your-name"
```

Question

Observez l'ordre de destruction. Quelles ressources sont détruites en premier, et pourquoi cet ordre est-il l'inverse exact de l'ordre de création ?

Questions

1. **Pourquoi les instances d'un subnet privé ne peuvent-elles pas joindre Internet directement ?** Que se passe-t-il concrètement quand un paquet sort d'un subnet privé sans NAT Gateway ?
2. **Pourquoi la NAT Gateway est-elle placée dans un subnet public plutôt que privé ?** De quoi a-t-elle besoin pour fonctionner ?
3. **Quelle est la différence fondamentale entre "stateful" et "stateless" dans le contexte du filtrage réseau ?** Pourquoi les NACLs requièrent-elles des règles explicites pour le trafic de retour sur les ports éphémères (1024-65535) ?
4. **Le security group app-sg n'autorise le port 8080 que depuis web-sg. Que se passe-t-il si un attaquant compromet une instance dans le subnet public qui ne fait pas partie de web-sg ?** L'attaquant peut-il atteindre le port 8080 des instances applicatives ?
5. **Pourquoi les subnets sont-ils déployés sur plusieurs zones de disponibilité ?** Quel type de panne cela permet-il de tolérer ?
6. **Si vous ajoutez une règle NACL numérotée 50 qui refuse le trafic HTTP, alors qu'une règle 100 l'autorise déjà, quel sera le comportement ?** Pourquoi l'ordre des règles compte-t-il dans une NACL, contrairement à un security group ?
7. **D'un point de vue sécurité, quel avantage une NACL apporte-t-elle par rapport à un security group seul ?** Dans quel scénario précis voudriez-vous utiliser les deux ?
8. **Observez les dépendances entre ressources.** Dessinez le graphe de dépendances du VPC jusqu'aux NACLs : qui dépend de quoi, et dans quel ordre Terraform peut paralléliser la création ?

Aller plus loin

- Ajoutez un **VPC Peering** avec un second VPC dans un autre bloc CIDR.
- Configurez un **VPN Gateway** (Site-to-Site) pour simuler une connexion vers un réseau on-premise.
- Activez les **VPC Flow Logs** sur la VPC et inspectez les enregistrements générés.
- Implémentez un **VPC Endpoint** (Gateway pour S3 ou Interface pour SSM) pour accéder à un service AWS sans passer par Internet.