TD2 – Sécurisation réseau AWS (VPC, Security Groups, ALB)

Objectifs

- Comprendre la segmentation réseau dans AWS avec VPC et subnets
- Identifier les mauvaises pratiques de sécurité réseau
- Mettre en place un point d'entrée unique avec un ALB
- Contrôler les flux réseau avec les Security Groups
- Appliquer le principe du moindre privilège sur chaque couche

Contexte

Vous intervenez en tant qu'ingénieur DevOps dans une startup.

L'équipe de développement a déployé une application backend sur AWS via Terraform.

Le déploiement a été fait rapidement pour respecter un délai.

Résultat :

- l'application fonctionne
- mais toute l'infrastructure est directement exposée sur Internet

Un audit de sécurité interne a identifié trois problèmes critiques :

- le port applicatif du backend est ouvert à Internet
- le port de la base de données est ouvert à Internet
- aucun point d'entrée centralisé ne permet de contrôler le trafic

Votre mission : transformer l'architecture sans interrompre le service.

Architecture cible

L'architecture actuelle :

```
Internet → Backend (port 80 ouvert partout)
Internet → DB (port 5432 ouvert partout)
```

L'architecture cible :

```
Internet → ALB (port 80) → Backend (port 80, privé) → DB (port 5432, isolée)
```

Contraintes

- L'application doit rester accessible depuis Internet
- Le backend ne doit plus être accessible directement depuis Internet
- La base de données ne doit être accessible que depuis le backend

- Tout le code est en Terraform

Pré-requis

- Terraform installé et configuré
- Credentials AWS valides
- Notions de base : VPC, subnet, Security Group

Infrastructure de départ

L'infrastructure existante présente plusieurs problèmes volontaires.

Lire le code attentivement avant de l'appliquer.

Fichier : `td2/network/main.tf`

```
provider "aws" {
  region = "eu-west-3"
}

# VPC principal
resource "aws_vpc" "main" {
  cidr_block           = "10.0.0.0/16"
  enable_dns_hostnames = true

  tags = {
    Name = "td2-vpc"
  }
}

# Passerelle Internet
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "td2-igw"
  }
}

# Subnet public — AZ a
resource "aws_subnet" "public_a" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "eu-west-3a"
  map_public_ip_on_launch = true

  tags = {
    Name = "td2-public-a"
  }
}

# Subnet public — AZ b (requis par l'ALB)
```

```
resource "aws_subnet" "public_b" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.2.0/24"
  availability_zone       = "eu-west-3b"
  map_public_ip_on_launch = true

  tags = {
    Name = "td2-public-b"
  }
}

# Subnet privé — DB
resource "aws_subnet" "private" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.3.0/24"
  availability_zone = "eu-west-3a"

  tags = {
    Name = "td2-private"
  }
}

# Table de routage publique
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }

  tags = {
    Name = "td2-rt-public"
  }
}

resource "aws_route_table_association" "public_a" {
  subnet_id      = aws_subnet.public_a.id
  route_table_id = aws_route_table.public.id
}

resource "aws_route_table_association" "public_b" {
  subnet_id      = aws_subnet.public_b.id
  route_table_id = aws_route_table.public.id
}
```

Fichier : `td2/network/security_groups.tf`

```
# Security Group backend — PROBLEME VOLONTAIRE
# Ce fichier sera modifié au cours du TD
resource "aws_security_group" "backend_sg" {
  name   = "backend-sg"
  vpc_id = aws_vpc.main.id

  ingress {
```

```
    description = "HTTP depuis Internet"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "backend-sg"
  }
}

# Security Group DB — PROBLEME VOLONTAIRE
resource "aws_security_group" "db_sg" {
  name   = "db-sg"
  vpc_id = aws_vpc.main.id

  ingress {
    description = "PostgreSQL depuis Internet"
    from_port   = 5432
    to_port     = 5432
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "db-sg"
  }
}
```

Fichier : `td2/network/instances.tf`

```
resource "aws_instance" "backend" {
  ami                    = "ami-0f61de2873e29e866"
  instance_type          = "t2.micro"
  subnet_id              = aws_subnet.public_a.id
  vpc_security_group_ids = [aws_security_group.backend_sg.id]

  user_data = <<-EOF
    #!/bin/bash
    yum install -y python3
```

```
    python3 -m http.server 80 &
  EOF

  tags = {
    Name = "td2-backend"
  }
}

resource "aws_db_instance" "db" {
  identifier             = "td2-db"
  engine                 = "postgres"
  engine_version         = "15"
  instance_class         = "db.t3.micro"
  allocated_storage      = 20
  username               = "admin"
  password               = "changeme123"
  db_subnet_group_name   = aws_db_subnet_group.main.name
  vpc_security_group_ids = [aws_security_group.db_sg.id]
  skip_final_snapshot    = true

  tags = {
    Name = "td2-db"
  }
}

resource "aws_db_subnet_group" "main" {
  name       = "td2-db-subnet-group"
  subnet_ids = [aws_subnet.private.id, aws_subnet.public_b.id]

  tags = {
    Name = "td2-db-subnet-group"
  }
}
```

Commandes Terraform

Fichier : `td2/commandes/terraform.txt`

```
# Initialiser le projet
terraform init

# Vérifier le plan sans appliquer
terraform plan

# Appliquer la configuration
terraform apply

# Récupérer les outputs
terraform output

# Détruire l'infrastructure
terraform destroy
```

1. Lecture du code

Lire les trois fichiers avant toute manipulation.

Lister les ressources présentes dans le code.

Pour chaque ressource, indiquer :

- son rôle
- le subnet dans lequel elle est placée
- les ports ouverts en entrée

Pourquoi l'ALB nécessite-t-il deux subnets dans deux zones de disponibilité différentes ?

Quelle contrainte AWS cela reflète-t-il ?

La base de données est une instance RDS et non une instance EC2.

Quelle est la différence en termes de gestion et de sécurité ?

2. Identification des problèmes

Identifier dans `security_groups.tf` les deux règles les plus dangereuses.

Pour chaque règle :

- expliquer ce qu'elle autorise concrètement
- expliquer ce qu'un attaquant pourrait faire avec cet accès

Le mot de passe de la base de données est en clair dans le fichier Terraform.

Quel problème cela pose-t-il ?

Ce problème sera traité dans un TD ultérieur sur la gestion des secrets.

La base de données RDS est placée dans un subnet group qui inclut le subnet `public_b`.

Est-ce un problème ? Pourquoi ?

3. Déploiement de l'infrastructure initiale

Appliquer la configuration telle quelle :

```
terraform apply
```

Relever les outputs suivants :

- IP publique de l'instance backend
- endpoint de la base RDS

Fichier : `td2/network/outputs.tf`

```
output "backend_public_ip" {
  value       = aws_instance.backend.public_ip
  description = "IP publique du backend"
}

output "db_endpoint" {
  value       = aws_db_instance.db.endpoint
  description = "Endpoint de la base de données"
}
```

Tester l'accès au backend depuis un navigateur ou curl :

```
curl http://<backend_public_ip>
```

Tenter une connexion à la base de données depuis votre poste :

```
psql -h <db_endpoint> -U admin -d postgres
```

Noter le résultat de chaque test.

Le backend répond-il ?

La connexion à la base de données est-elle possible depuis votre poste ?

Pourquoi ce comportement est-il problématique dans un contexte de production ?

4. Mise en place de l'ALB

Créer le fichier suivant sans modifier les Security Groups existants.

Fichier : `td2/network/alb.tf`

```
# Security Group de l'ALB
resource "aws_security_group" "alb_sg" {
  name   = "alb-sg"
  vpc_id = aws_vpc.main.id

  ingress {
    description = "HTTP depuis Internet"
    from_port   = 80
    to_port     = 80
```

```
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "alb-sg"
  }
}

# Application Load Balancer
resource "aws_lb" "main" {
  name               = "td2-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.alb_sg.id]
  subnets            = [aws_subnet.public_a.id, aws_subnet.public_b.id]

  tags = {
    Name = "td2-alb"
  }
}

# Target Group
resource "aws_lb_target_group" "backend" {
  name     = "td2-backend-tg"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main.id

  health_check {
    path                = "/"
    healthy_threshold   = 2
    unhealthy_threshold = 2
    interval            = 30
  }

  tags = {
    Name = "td2-backend-tg"
  }
}

# Attachement du backend au Target Group
resource "aws_lb_target_group_attachment" "backend" {
  target_group_arn = aws_lb_target_group.backend.arn
  target_id        = aws_instance.backend.id
  port             = 80
}

# Listener HTTP
```

```
resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.main.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.backend.arn
  }
}
```

Ajouter l'output du DNS de l'ALB dans `outputs.tf` :

```
output "alb_dns" {
  value       = aws_lb.main.dns_name
  description = "DNS public de l'ALB"
}
```

Appliquer :

```
terraform apply
```

Tester l'accès via l'ALB :

```
curl http://<alb_dns>
```

Tester également l'accès direct au backend :

```
curl http://<backend_public_ip>
```

Les deux accès fonctionnent-ils ?

Qu'est-ce que cela démontre sur l'état actuel de la sécurisation ?

L'ALB est-il suffisant seul pour sécuriser le backend ?

5. Erreur courante – le backend reste exposé

L'ALB est en place mais le Security Group du backend n'a pas été modifié.

Le port 80 du backend est toujours ouvert sur `0.0.0.0/0`.

Pourquoi ajouter un ALB ne suffit-il pas à protéger le backend ?

Quel composant contrôle réellement les accès réseau sur une instance AWS ?

6. Correction – restriction du backend

Modifier `security_groups.tf`.

Remplacer la règle ingress du Security Group `backend_sg` :

- supprimer l'accès depuis `0.0.0.0/0`
- autoriser uniquement le trafic depuis le Security Group de l'ALB

Fichier : `td2/network/security_groups.tf`

```
resource "aws_security_group" "backend_sg" {
  name   = "backend-sg"
  vpc_id = aws_vpc.main.id

  ingress {
    description     = "HTTP depuis l'ALB uniquement"
    from_port       = 80
    to_port         = 80
    protocol        = "tcp"
    security_groups = [aws_security_group.alb_sg.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "backend-sg"
  }
}

resource "aws_security_group" "db_sg" {
  name   = "db-sg"
  vpc_id = aws_vpc.main.id

  ingress {
    description     = "PostgreSQL depuis le backend uniquement"
    from_port       = 5432
    to_port         = 5432
    protocol        = "tcp"
    security_groups = [aws_security_group.backend_sg.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
```

```
  }

  tags = {
    Name = "db-sg"
  }
}
```

Appliquer les modifications :

```
terraform apply
```

Tester les deux accès :

```
# Via l'ALB — doit fonctionner
curl http://<alb_dns>
```

```
# Direct — doit échouer
curl http://<backend_public_ip>
```

L'accès direct au backend est-il maintenant bloqué ?

L'accès via l'ALB fonctionne-t-il toujours ?

Que se passe-t-il si le Health Check de l'ALB échoue ? Comment vérifier l'état du Target Group dans la console AWS ?

7. Problème potentiel – Health Check

Après modification des Security Groups, l'ALB peut afficher le backend comme “unhealthy”.

Expliquer ce qu'est un Health Check d'ALB.

Quel flux réseau le Health Check génère-t-il ?

Depuis quelle source arrive ce flux sur le backend ?

Si le backend est “unhealthy”, l'ALB cesse de lui envoyer du trafic.

Le service devient indisponible même si le backend fonctionne.

Quelle règle dans le Security Group du backend pourrait causer ce blocage ?

Vérifier dans la console AWS :

- aller dans EC2 > Target Groups
- vérifier le statut de l'instance dans le Target Group
- lire le message d'erreur du Health Check si présent

8. Diagnostic et correction

Si le Health Check échoue, identifier la cause :

- le port 80 est-il bien ouvert depuis le Security Group de l'ALB ?
- le service python3 est-il démarré sur l'instance ?
- la règle egress de l'ALB permet-elle les réponses ?

Corriger la règle manquante et réappliquer.

9. Vérification finale

Réaliser les tests suivants et noter les résultats dans un tableau :

- accès HTTP via l'ALB → attendu : 200
- accès HTTP direct au backend → attendu : timeout ou refus
- connexion PostgreSQL depuis votre poste → attendu : timeout ou refus
- connexion PostgreSQL depuis le backend → attendu : possible

Comment vérifier qu'un flux est bien bloqué et non juste lent ?

Quelle différence entre un timeout et un Connection refused en termes de Security Group ?

10. Extension – déplacement du backend en subnet privé

Le backend a une IP publique car il est dans un subnet public.

Ce n'est pas nécessaire si l'ALB gère tout le trafic entrant.

Modifier `instances.tf` pour déplacer le backend dans le subnet privé :

```
subnet_id = aws_subnet.private.id
```

Supprimer également le `map_public_ip_on_launch` implicite.

Appliquer et vérifier que l'ALB continue de fonctionner.

Le backend n'a plus d'IP publique.

Est-il encore accessible depuis Internet ?

Comment l'ALB peut-il atteindre une instance dans un subnet privé ?

Quel composant réseau manque pour que le backend puisse faire des appels sortants (mises à jour, appels API

externes) ?

Challenge final

Objectif : valider l'architecture complète Internet → ALB → Backend → DB.

Dessiner l'architecture finale avec :

- les subnets (public_a, public_b, private)
- les composants dans chaque subnet
- les Security Groups et leurs règles
- les flux autorisés avec les ports

Répondre aux questions suivantes par écrit :

- Quel composant est le seul point d'entrée depuis Internet ?
- Quelles ressources ne sont plus accessibles directement depuis Internet ?
- Quel principe de sécurité est appliqué sur chaque Security Group ?
- Que se passe-t-il si l'ALB est supprimé ? L'application reste-t-elle accessible ?

Bonus

Ajouter une seconde instance backend dans `public_b` :

```
resource "aws_instance" "backend_b" {
  ami                    = "ami-0f61de2873e29e866"
  instance_type          = "t2.micro"
  subnet_id              = aws_subnet.public_b.id
  vpc_security_group_ids = [aws_security_group.backend_sg.id]
  ...
}
```

Attacher cette instance au même Target Group.

Quels sont les avantages d'avoir deux instances backend derrière un ALB ?

Répondre selon deux axes :

- disponibilité : que se passe-t-il si une instance tombe ?
- sécurité : l'exposition du système augmente-t-elle ou diminue-t-elle ?

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/eadl/bloc4/fm4/td2

Last update: **2026/06/14 17:07**