04/02/2026

TP AWS04

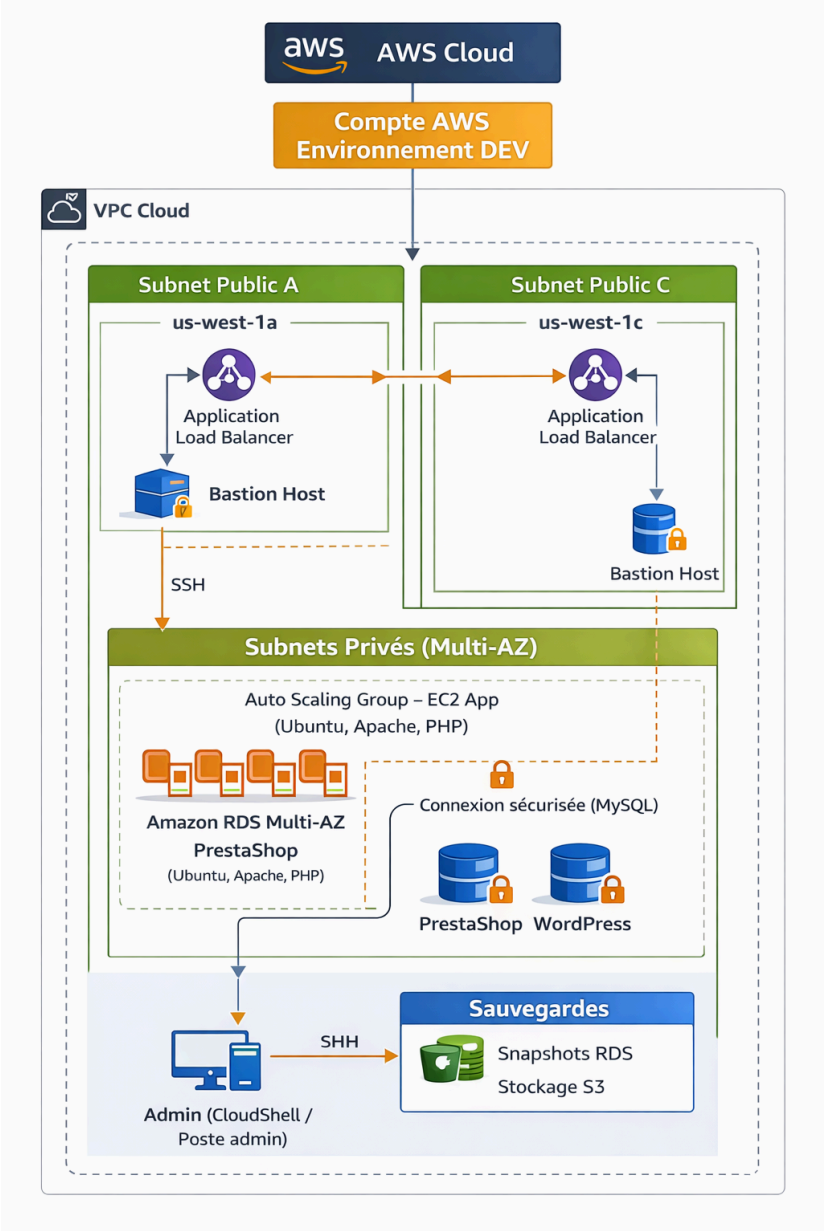
Date	Utilisateur	Observation
04/02/2026	Bryan MAHAUT / Axel PEIFFER / Axel IRSUTTI	Réalisation du document technique

Sommaire :

04/02/2026

Partie 1 :

Schéma



Création du VPC

Un Virtual Private Cloud (VPC) unique a été créé afin d'héberger l'ensemble de l'infrastructure du MVP, conformément aux exigences du client visant à limiter les coûts. Le VPC utilise le plan d'adressage 10.0.0.0/16, permettant une segmentation future en sous-réseaux publics et privés tout en conservant une grande flexibilité pour l'évolution de l'infrastructure.

```
~ $ aws ec2 create-vpc \
>   --cidr-block 10.0.0.0/16 \
~ $ aws ec2 create-vpc \
>   --cidr-block 10.0.0.0/16 \
>   --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=dev-vpc}]'
{
    "Vpc": {
        "OwnerId": "336749236319",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0f5c24d36e0fc71a1",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Name",
                "Value": "dev-vpc"
            }
        ],
        "VpcId": "vpc-00ddb0031795e24fd",
        "State": "pending",
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-095f8d489f0fac277"
    }
}
```

Les options DNS ont été activées afin de permettre la résolution de noms nécessaire au fonctionnement des services managés AWS.

```
}
~ $ aws ec2 modify-vpc-attribute \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --enable-dns-support
~ $ aws ec2 modify-vpc-attribute \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --enable-dns-hostnames
~ $
```

Création des sous réseaux

Afin de structurer le réseau et de respecter les bonnes pratiques de sécurité, le VPC a été découpé en plusieurs sous-réseaux publics et privés.

Les subnets publics sont destinés aux ressources nécessitant un accès direct à Internet, comme le Load Balancer ou la passerelle NAT.
Les subnets privés hébergent les applications et les bases de données afin de limiter leur exposition et renforcer la sécurité.

Les sous-réseaux ont été répartis sur deux zones de disponibilité (us-west-1a et us-west-1c) afin d'assurer une meilleure tolérance aux pannes et de préparer la haute disponibilité de l'infrastructure.

Subnet public a: 10.0.1.0/24

```
~ $ aws ec2 create-subnet \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --cidr-block 10.0.1.0/24 \
>   --availability-zone us-west-1a \
>   --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=dev-public-a}]'
{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az1",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "dev-public-a"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-075b27dd05884f660",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-075b27dd05884f660",
        "State": "available",
        "VpcId": "vpc-00ddb0031795e24fd",
        "CidrBlock": "10.0.1.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1a",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
```

Subnet public c: 10.0.2.0/24

```
~ $ aws ec2 create-subnet \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --cidr-block 10.0.2.0/24 \
>   --availability-zone us-west-1c \
>   --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=dev-public-c}]'
{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az3",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "dev-public-c"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-02ad6b59c5465f37c",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-02ad6b59c5465f37c",
        "State": "available",
        "VpcId": "vpc-00ddb0031795e24fd",
        "CidrBlock": "10.0.2.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1c",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
```

Subnet private a: 10.0.11.0/24

```
~ $ aws ec2 create-subnet \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --cidr-block 10.0.11.0/24 \
>   --availability-zone us-west-1a \
>   --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=dev-private-a}]'
{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az1",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "dev-private-a"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-0470418ab76223757",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-0470418ab76223757",
        "State": "available",
        "VpcId": "vpc-00ddb0031795e24fd",
        "CidrBlock": "10.0.11.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1a",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
```

Subnet private c: 10.0.12.0/24

```
~ $ aws ec2 create-subnet \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --cidr-block 10.0.12.0/24 \
>   --availability-zone us-west-1c \
>   --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=dev-private-c}]'
{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az3",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "dev-private-c"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-05ee0ecb62172543e",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-05ee0ecb62172543e",
        "State": "available",
        "VpcId": "vpc-00ddb0031795e24fd",
        "CidrBlock": "10.0.12.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1c",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
```

Création + attachement de l’internet Gateway (IGW) au VPC

Une Internet Gateway a été déployée et attachée au VPC afin de permettre aux subnets publics d’accéder à Internet.

```
        }
    }
    ~ $ aws ec2 create-internet-gateway \
    >   --tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name,Value=dev-igw}]'
    {
        "InternetGateway": {
            "Attachments": [],
            "InternetGatewayId": "igw-086fa960aa64edfd4",
            "OwnerId": "336749236319",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "dev-igw"
                }
            ]
        }
    }
    ~ $ aws ec2 attach-internet-gateway \
    >   --internet-gateway-id igw-086fa960aa64edfd4 \
    >   --vpc-id vpc-00ddb0031795e24fd
```

Table de routage pour les subnets publics

Une table de routage dédiée a été créée et associée aux subnets publics, avec une route par défaut vers l’Internet Gateway.

Une route par défaut vers l’Internet Gateway permet aux ressources exposées d’accéder à Internet, tandis que les subnets privés restent isolés.

```
~ $ aws ec2 create-route-table \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=dev-public-rt}]'
{
    "RouteTable": {
        "Associations": [],
        "PropagatingVgws": [],
        "RouteTableId": "rtb-0b308a2a4d42462a6",
        "Routes": [
            {
                "DestinationCidrBlock": "10.0.0.0/16",
                "GatewayId": "local",
                "Origin": "CreateRouteTable",
                "State": "active"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "dev-public-rt"
            }
        ],
        "VpcId": "vpc-00ddb0031795e24fd",
        "OwnerId": "336749236319"
    },
    "ClientToken": "52a677b2-054d-48ab-96c8-8d58d9f656dc"
}
~ $
```

Route vers internet

```
}
~ $ aws ec2 create-route \
>   --route-table-id rtb-0b308a2a4d42462a6 \
>   --destination-cidr-block 0.0.0.0/0 \
>   --gateway-id igw-086fa960aa64edfd4
{
    "Return": true
}
~ $
```

Association des subnets publics

```
~ $ aws ec2 associate-route-table \
>   --route-table-id rtb-0b308a2a4d42462a6 \
>   --subnet-id subnet-075b27dd05884f660
{
    "AssociationId": "rtbassoc-0abfb8353441c452a",
    "AssociationState": {
        "State": "associated"
    }
}
~ $ aws ec2 associate-route-table \
>   --route-table-id rtb-0b308a2a4d42462a6 \
>   --subnet-id subnet-02ad6b59c5465f37c
{
    "AssociationId": "rtbassoc-00054c3a081b53a56",
    "AssociationState": {
        "State": "associated"
    }
}
```

Création d’une Elastic IP (EIP)

Le NAT Gateway a besoin d’une IP publique dédiée.

```
~ $ aws ec2 allocate-address --domain vpc
{
    "AllocationId": "eipalloc-0ef00dc0c727181fb",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-west-1",
    "Domain": "vpc",
    "PublicIp": "54.241.182.75"
}
~ $
```

Création du NAT Gateway

Une passerelle NAT a été déployée dans un subnet public afin de permettre aux instances situées dans les subnets privés d’accéder à Internet sans exposer d’adresses IP publiques.

Création de la table de routage privée

Une table de routage privée dédiée a été créée et associée aux subnets privés, avec une route par défaut pointant vers la passerelle NAT.

```
~ $ aws ec2 create-route-table \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=dev-private-rt}]'
{
    "RouteTable": {
        "Associations": [],
        "PropagatingVgws": [],
        "RouteTableId": "rtb-00bbc2d1bbb7fdbdb",
        "Routes": [
            {
                "DestinationCidrBlock": "10.0.0.0/16",
                "GatewayId": "local",
                "Origin": "CreateRouteTable",
                "State": "active"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "dev-private-rt"
            }
        ],
        "VpcId": "vpc-00ddb0031795e24fd",
        "OwnerId": "336749236319"
    },
    "ClientToken": "7f5f7de6-3ce9-4dc6-98b1-7ac5351c0019"
}
~ $
```

Route vers internet via le NAT

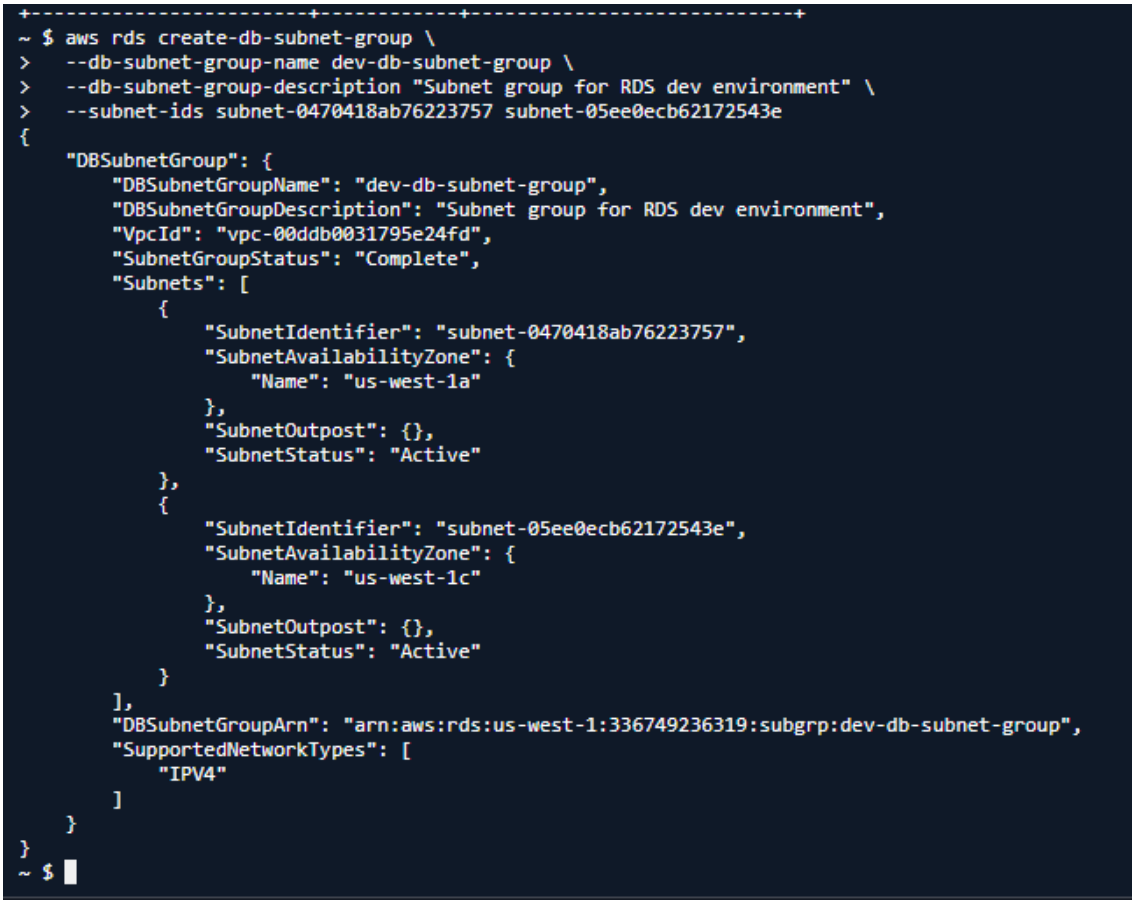
```
}
~ $ aws ec2 create-route \
>   --route-table-id rtb-00bbc2d1bbb7fdbdb \
>   --destination-cidr-block 0.0.0.0/0 \
>   --nat-gateway-id nat-0bbfec6fee7c3c87e
{
    "Return": true
}
~ $
```

Association des subnets privés

```
}
~ $ aws ec2 associate-route-table \
>   --route-table-id rtb-00bbc2d1bbb7fdbdb \
>   --subnet-id subnet-0470418ab76223757
{
    "AssociationId": "rtbassoc-0a7d4f5004f74df33",
    "AssociationState": {
        "State": "associated"
    }
}
~ $ aws ec2 associate-route-table \
>   --route-table-id rtb-00bbc2d1bbb7fdbdb \
>   --subnet-id subnet-05ee0ecb62172543e
{
    "AssociationId": "rtbassoc-04e5c0d41b8b6546d",
    "AssociationState": {
        "State": "associated"
    }
}
~ $
```

Création du DB Subnet Group

Un DB Subnet Group a été créé afin de définir les sous-réseaux dans lesquels la base de données RDS peut être déployée.
Ce groupe regroupe des subnets privés répartis sur plusieurs zones de disponibilité, permettant à la base de données d'être hébergée de manière sécurisée et hautement disponible.
Cette configuration est indispensable au déploiement d'une base RDS en mode Multi-AZ et garantit l'isolation réseau de la base de données.



```
+-----------------------+------------+---------------------------+
~ $ aws rds create-db-subnet-group \
>   --db-subnet-group-name dev-db-subnet-group \
>   --db-subnet-group-description "Subnet group for RDS dev environment" \
>   --subnet-ids subnet-0470418ab76223757 subnet-05ee0ecb62172543e
{
    "DBSubnetGroup": {
        "DBSubnetGroupName": "dev-db-subnet-group",
        "DBSubnetGroupDescription": "Subnet group for RDS dev environment",
        "VpcId": "vpc-00ddb0031795e24fd",
        "SubnetGroupStatus": "Complete",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-0470418ab76223757",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-1a"
                },
                "SubnetOutpost": {},
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-05ee0ecb62172543e",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-1c"
                },
                "SubnetOutpost": {},
                "SubnetStatus": "Active"
            }
        ],
        "DBSubnetGroupArn": "arn:aws:rds:us-west-1:336749236319:subgrp:dev-db-subnet-group",
        "SupportedNetworkTypes": [
            "IPV4"
        ]
    }
}
~ $
```

Création de la base de données RDS

Une base de données mysql managée Amazon RDS a été déployée afin de limiter la gestion opérationnelle tout en garantissant la disponibilité du service.
Le mode Multi-AZ a été activé pour assurer la haute disponibilité, et la base de données a été placée dans des subnets privés afin de renforcer la sécurité.

```
}
~ $ aws rds create-db-instance \
>   --db-instance-identifier dev-mysql-db \
>   --db-instance-class db.t3.micro \
>   --engine mysql \
>   --allocated-storage 20 \
>   --master-username admin \
>   --master-user-password Admin123! \
>   --multi-az \
>   --no-publicly-accessible \
>   --db-subnet-group-name dev-db-subnet-group \
>   --backup-retention-period 1 \
>   --tags Key=Name,Value=dev-mysql-db
{
    "DBInstance": {
        "DBInstanceIdentifier": "dev-mysql-db",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 20,
        "PreferredBackupWindow": "06:05-06:35",
        "BackupRetentionPeriod": 1,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-08208ebde59d74bc3",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.mysql8.4",
                "ParameterApplyStatus": "in-sync"
            }
        ],
```

Status

```
~ $ aws rds describe-db-instances \
>   --db-instance-identifier dev-mysql-db \
>   --query "DBInstances[0].DBInstanceStatus"
"creating"
~ $ aws rds describe-db-instances \
>   --db-instance-identifier dev-mysql-db \
>   --query "DBInstances[0].DBInstanceStatus"
"modifying"
~ $ aws rds describe-db-instances   --db-instance-identifier dev-mysql-db   --query "DBInstances[0].DBInstanceStatus"
"available"
~ $
```

Création du security group de l’application

Un groupe de sécurité dédié au serveur applicatif a été créé afin de contrôler les flux entrants vers les applications.
Les ports HTTP (80) et HTTPS (443) ont été autorisés afin de permettre l’accès aux utilisateurs via le Load Balancer, tout en laissant à AWS la gestion des flux sortants.

```
~ $ aws ec2 create-security-group \
>   --group-name dev-app-sg \
>   --description "Security group for application servers" \
>   --vpc-id vpc-00ddb0031795e24fd
{
    "GroupId": "sg-033238da793540f1b",
    "SecurityGroupArn": "arn:aws:ec2:us-west-1:336749236319:security-group/sg-033238da793540f1b"
}
```

Ports autorisés

```
~ $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-033238da793540f1b \
>   --protocol tcp \
>   --port 80 \
>   --cidr 0.0.0.0/0
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-032bab7170f158044",
            "GroupId": "sg-033238da793540f1b",
            "GroupOwnerId": "336749236319",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "0.0.0.0/0",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-west-1:336749236319:security-group-rule/sgr-032bab7170f158044"
        }
    ]
}
~ $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-033238da793540f1b \
>   --protocol tcp \
>   --port 443 \
>   --cidr 0.0.0.0/0
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-09906810ffa301510",
            "GroupId": "sg-033238da793540f1b",
            "GroupOwnerId": "336749236319",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 443,
            "ToPort": 443,
            "CidrIpv4": "0.0.0.0/0",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-west-1:336749236319:security-group-rule/sgr-09906810ffa301510"
        }
    ]
```

Création du security group de la DB

Un groupe de sécurité spécifique à la base de données a été mis en place afin de garantir l'isolation du service.
Seules les connexions MySQL (port 3306) provenant du groupe de sécurité du serveur applicatif sont autorisées, empêchant tout accès direct depuis Internet et renforçant la sécurité de la base de données.

```
}
~ $ aws ec2 create-security-group \
>   --group-name dev-db-sg \
>   --description "Security group for RDS MySQL" \
>   --vpc-id vpc-00ddb0031795e24fd
{
    "GroupId": "sg-024f0815ad47b5393",
    "SecurityGroupArn": "arn:aws:ec2:us-west-1:336749236319:security-group/sg-024f0815ad47b5393"
}
~ $
```

Port autorisé depuis l'APP

```
~ $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-024f0815ad47b5393 \
>   --protocol tcp \
>   --port 3306 \
>   --source-group sg-033238da793540f1b
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0bb3680d350060245",
            "GroupId": "sg-024f0815ad47b5393",
            "GroupOwnerId": "336749236319",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 3306,
            "ToPort": 3306,
            "ReferencedGroupInfo": {
                "GroupId": "sg-033238da793540f1b",
                "UserId": "336749236319"
            },
            "SecurityGroupRuleArn": "arn:aws:ec2:us-west-1:336749236319:security-group-rule/sgr-0bb3680d350060245"
        }
    ]
}
~ $
```

Le groupe de sécurité de la base de données a été attaché à l'instance RDS afin d'appliquer les règles de filtrage réseau définies et de garantir que seules les applications autorisées puissent s'y connecter.

```
}
~ $ aws rds modify-db-instance \
>   --db-instance-identifier dev-mysql-db \
>   --vpc-security-group-ids sg-024f0815ad47b5393 \
>   --apply-immediately
{
    "DBInstance": {
        "DBInstanceIdentifier": "dev-mysql-db",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "available",
        "MasterUsername": "admin",
        "Endpoint": {
            "Address": "dev-mysql-db.czcc4yewi3vy.us-west-1.rds.amazonaws.com",
            "Port": 3306,
            "HostedZoneId": "Z10WI91S59XXQN"
        },
        "AllocatedStorage": 20,
        "InstanceCreateTime": "2026-02-04T10:40:25.334000+00:00",
        "PreferredBackupWindow": "06:05-06:35",
        "BackupRetentionPeriod": 1,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-024f0815ad47b5393",
                "Status": "adding"
            },
            {
                "VpcSecurityGroupId": "sg-08208ebde59d74bc3",
                "Status": "removing"
            }
        ],
```

Pré-requis de l’application

Dans cette étape, une architecture applicative scalable a été mise en place afin d’héberger l’application e-commerce du MVP.
L’application est exposée via un Application Load Balancer situé dans les subnets publics, tandis que le serveur applicatif est déployé dans des subnets privés afin de renforcer la sécurité.
Cette architecture permet de répartir la charge, d’améliorer la disponibilité du service et de préparer l’infrastructure à une montée en charge automatique.

Création du Security Group du Load Balancer

Un groupe de sécurité dédié au Load Balancer a été créé afin d’autoriser l’accès HTTP depuis Internet.
Ce groupe de sécurité permet de contrôler les flux entrants tout en limitant l’exposition du serveur applicatif.

```
~ $ aws ec2 create-security-group \
>   --group-name dev-alb-sg \
>   --description "Security group for Application Load Balancer" \
>   --vpc-id vpc-00ddb0031795e24fd
{
    "GroupId": "sg-0e2359ede70af6a35",
    "SecurityGroupArn": "arn:aws:ec2:us-west-1:336749236319:security-group/sg-0e2359ede70af6a35"
}
~ $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-0e2359ede70af6a35 \
>   --protocol tcp \
>   --port 80 \
>   --cidr 0.0.0.0/0
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0539b7b06faf50435",
            "GroupId": "sg-0e2359ede70af6a35",
            "GroupOwnerId": "336749236319",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "0.0.0.0/0",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-west-1:336749236319:security-group-rule/sgr-0539b7b06faf50435"
        }
    ]
}
```

Autorisation du trafic ALB to APP

Une règle de sécurité a été ajoutée afin d'autoriser le trafic HTTP provenant exclusivement du Load Balancer vers le serveur applicatif.
Cette configuration permet d'empêcher tout accès direct aux instances applicatives depuis Internet.

```
    ]
}
~ $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-033238da793540f1b \
>   --protocol tcp \
>   --port 80 \
>   --source-group sg-0e2359ede70af6a35
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0928268ea89a27973",
            "GroupId": "sg-033238da793540f1b",
            "GroupOwnerId": "336749236319",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "ReferencedGroupInfo": {
                "GroupId": "sg-0e2359ede70af6a35",
                "UserId": "336749236319"
            },
            "SecurityGroupRuleArn": "arn:aws:ec2:us-west-1:336749236319:security-group-rule/sgr-0928268ea89a27973"
        }
    ]
}
~ $
```

Création du Load Balancer (ALB)

Un Application Load Balancer a été déployé dans les subnets publics afin de répartir le trafic entrant sur plusieurs zones de disponibilité.
Le Load Balancer est exposé publiquement, tandis que les instances applicatives restent hébergées dans des subnets privés.

```
~ $ aws elbv2 create-load-balancer \
>   --name dev-alb \
>   --subnets subnet-075b27dd05884f660 subnet-02ad6b59c5465f37c \
>   --security-groups sg-0e2359ede70af6a35 \
>   --scheme internet-facing \
>   --type application
{
    "LoadBalancers": [
        {
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-west-1:336749236319:loadbalancer/app/dev-alb/5d925d7279603d6d",
            "DNSName": "dev-alb-784941924.us-west-1.elb.amazonaws.com",
            "CanonicalHostedZoneId": "Z368ELLRRE2KJ0",
            "CreatedTime": "2026-02-04T11:27:04.396000+00:00",
            "LoadBalancerName": "dev-alb",
            "Scheme": "internet-facing",
            "VpcId": "vpc-00ddb0031795e24fd",
            "State": {
                "Code": "provisioning"
            },
            "Type": "application",
            "AvailabilityZones": [
                {
                    "ZoneName": "us-west-1a",
                    "SubnetId": "subnet-075b27dd05884f660",
                    "LoadBalancerAddresses": []
                },
                {
                    "ZoneName": "us-west-1c",
                    "SubnetId": "subnet-02ad6b59c5465f37c",
                    "LoadBalancerAddresses": []
                }
            ],
            "SecurityGroups": [
                "sg-0e2359ede70af6a35"
            ],
            "IpAddressType": "ipv4"
        }
    ]
}
~ $
```

Création du Target Group

Un groupe de cibles (Target Group) a été créé afin de définir les instances applicatives qui recevront le trafic du Load Balancer.
Un mécanisme de contrôle de l'état de santé a été configuré pour garantir que seules les instances fonctionnelles reçoivent du trafic.

```
~ $ aws elbv2 create-target-group \
>   --name dev-app-tg \
>   --protocol HTTP \
>   --port 80 \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --target-type instance \
>   --health-check-path /
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-west-1:336749236319:targetgroup/dev-app-tg/6f5bb0c416c91568",
            "TargetGroupName": "dev-app-tg",
            "Protocol": "HTTP",
            "Port": 80,
            "VpcId": "vpc-00ddb0031795e24fd",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
~ $
```

Création listener HTTP:80

```
arn:aws:elasticloadbalancing:us-west-1:336749236319:loadbalancer/app/dev-alb/5d925d7279603d6d
~ $ aws elbv2 create-listener \
>   --load-balancer-arn arn:aws:elasticloadbalancing:us-west-1:336749236319:loadbalancer/app/dev-alb/5d925d7279603d6d \
>   --protocol HTTP \
>   --port 80 \
>   --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-west-1:336749236319:targetgroup/dev-app-tg/6f5bb0c416c91568
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-west-1:336749236319:listener/app/dev-alb/5d925d7279603d6d/ae207b3ae042673e",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-west-1:336749236319:loadbalancer/app/dev-alb/5d925d7279603d6d",
            "Port": 80,
            "Protocol": "HTTP",
            "DefaultActions": [
                {
                    "Type": "forward",
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-west-1:336749236319:targetgroup/dev-app-tg/6f5bb0c416c91568",
                    "ForwardConfig": {
                        "TargetGroups": [
                            {
                                "TargetGroupArn": "arn:aws:elasticloadbalancing:us-west-1:336749236319:targetgroup/dev-app-tg/6f5bb0c416c91568",
                                "Weight": 1
                            }
                        ],
                        "TargetGroupStickinessConfig": {
                            "Enabled": false
                        }
                    }
                }
            ]
        }
    ]
}
~ $
```

Relation Listener HTTP to Target Group

Un listener HTTP a été configuré sur le Load Balancer afin de rediriger automatiquement les requêtes entrantes vers le Target Group associé.
Cette étape permet d'assurer le lien entre le Load Balancer et le serveur applicatif.

```
~ $ aws elbv2 create-target-group \
>   --name dev-app-tg \
>   --protocol HTTP \
>   --port 80 \
>   --vpc-id vpc-00ddb0031795e24fd \
>   --target-type instance \
>   --health-check-path /
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-west-1:336749236319:targetgroup/dev-app-tg/6f5bb0c416c91568",
            "TargetGroupName": "dev-app-tg",
            "Protocol": "HTTP",
            "Port": 80,
            "VpcId": "vpc-00ddb0031795e24fd",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
~ $
```

Déploiement de l'instance EC2 App + Auto Scaling

Cette étape consiste à déployer un serveur applicatif de l'application e-commerce à l'aide d'un Auto Scaling Group.
Cette approche permet d'ajuster automatiquement le nombre d'instances en fonction de la charge tout en garantissant une haute disponibilité et une meilleure résilience.

Création de l'instance app via launch template

L'instance applicative est basée sur une image Ubuntu Server LTS afin de bénéficier d'un environnement stable et largement utilisé.
Un script d'initialisation permet l'installation automatique (launch template) du serveur web Apache et le déploiement de l'application dès le démarrage de l'instance.

```
~ $ aws ec2 create-launch-template-version \
>   --launch-template-name dev-app-lt \
>   --source-version 1 \
>   --launch-template-data "{ \"ImageId\":\"ami-072028e29f8a73b88\" }"
{
    "LaunchTemplateVersion": {
        "LaunchTemplateId": "lt-027c175701dbfc3b0",
        "LaunchTemplateName": "dev-app-lt",
        "VersionNumber": 2,
        "CreateTime": "2026-02-04T11:42:49+00:00",
        "CreatedBy": "arn:aws:iam::336749236319:user/MAHAUT-Bryan",
        "DefaultVersion": false,
        "LaunchTemplateData": {
            "ImageId": "ami-072028e29f8a73b88",
            "InstanceType": "t3.micro",
            "UserData": "IyEvYmluL2Jhc2gKYXB0IHVwZGF0ZSAtYXB0IHVwZ3JhZGUgLXkKYXB0IGluc3RhbGwgLXkgYXBhY2hlMgplbmFibGUgYXBhY2hlMgpzeXN0
5odG1s",
            "SecurityGroupIds": [
                "sg-033238da793540f1b"
            ]
        },
        "Operator": {
            "Managed": false
        }
    }
}
~ $
```

Ajout d’une clé SSH dev-key.pem pour pouvoir se connecter sur l’instance app depuis le bastion

```
~ $ aws ec2 create-launch-template-version \
>   --launch-template-name dev-app-lt \
>   --source-version 2 \
>   --version-description "Add SSH key for bastion access" \
>   --launch-template-data "{
>     \"KeyName\":\"dev-key\"
>   }"
{
    "LaunchTemplateVersion": {
        "LaunchTemplateId": "lt-027c175701dbfc3b0",
        "LaunchTemplateName": "dev-app-lt",
        "VersionNumber": 3,
        "VersionDescription": "Add SSH key for bastion access",
        "CreateTime": "2026-02-04T12:21:17+00:00",
        "CreatedBy": "arn:aws:iam::336749236319:user/MAHAUT-Bryan",
        "DefaultVersion": false,
        "LaunchTemplateData": {
            "ImageId": "ami-072028e29f8a73b88",
            "InstanceType": "t3.micro",
            "KeyName": "dev-key",
            "UserData": "IyEvYmluL2Jhc2gKYXB0IHVwZGF0ZSAtYXB0IHVwZ3JhZGUgLXkKYXB0IGluc3RhbGwgLXkgYXBhY
dG1s",
            "SecurityGroupIds": [
                "sg-033238da793540f1b"
            ]
        },
        "Operator": {
            "Managed": false
        }
    }
}
```

IP Privée de l’instance App

```
~ $ aws ec2 describe-instances \
>   --instance-ids i-084c915c47c7e7056 \
>   --query "Reservations[0].Instances[0].PrivateIpAddress" \
>   --output text
10.0.12.212
~ $
```

Création de l’Auto Scaling Group

Un Auto Scaling Group a été configuré afin de déployer automatiquement les instances applicatives dans plusieurs zones de disponibilité.
Les instances sont enregistrées dans un Target Group, permettant au Load Balancer de répartir le trafic de manière dynamique.

```
~ $ aws autoscaling create-auto-scaling-group \
>   --auto-scaling-group-name dev-app-asg \
>   --launch-template LaunchTemplateName=dev-app-lt,Version=2 \
>   --min-size 1 \
>   --max-size 2 \
>   --desired-capacity 1 \
>   --vpc-zone-identifier "subnet-0470418ab76223757,subnet-05ee0ecb62172543e" \
>   --target-group-arns arn:aws:elasticloadbalancing:us-west-1:336749236319:targetgroup/dev-app-tg/6f5bb0c416c91568
~ $
```

Mise à jour du auto scaling group avec la nouvelle instance (avec clé)

```
}
~ $ aws autoscaling update-auto-scaling-group \
>   --auto-scaling-group-name dev-app-asg \
>   --launch-template LaunchTemplateName=dev-app-lt,Version=3
```

Recyclage de l’instance applicative

```
Unknown options: false

~ $ aws autoscaling terminate-instance-in-auto-scaling-group \
>   --instance-id i-03ec5c51c93cca8e5 \
>   --no-should-decrement-desired-capacity
{
    "Activity": {
        "ActivityId": "6c3670a2-6e79-c05c-a25a-d43c88e35be9",
        "AutoScalingGroupName": "dev-app-asg",
        "Description": "Terminating EC2 instance: i-03ec5c51c93cca8e5",
        "Cause": "At 2026-02-04T12:23:33Z instance i-03ec5c51c93cca8e5 was taken out of service in response to a user request.",
        "StartTime": "2026-02-04T12:23:33.232000+00:00",
        "StatusCode": "InProgress",
        "Progress": 0,
        "Details": "{\"Availability Zone ID\":\"usw1-az1\",\"Subnet ID\":\"subnet-0470418ab76223757\",\"Availability Zone\":\"us-west-1a\"}"
    }
}
~ $ aws autoscaling describe-auto-scaling-groups \
```

Suite à un besoin d’accès sécurisé aux instances applicatives, le Launch Template a été mis à jour afin d’y intégrer une clé SSH.
Une nouvelle version du template a ensuite été appliquée à l’Auto Scaling Group, permettant aux nouvelles instances de bénéficier automatiquement de cette configuration.
Les instances existantes ont été recyclées par l’Auto Scaling Group afin d’appliquer les modifications sans interruption de service, illustrant le fonctionnement dynamique et automatisé de la mise à l’échelle sur AWS.

Création Bastion

Un bastion host a été déployé dans un subnet public afin de servir de point d’accès sécurisé vers les instances applicatives hébergées dans des subnets privés.

Création du Security Group du bastion

```
~ $ aws ec2 create-security-group \
>   --group-name dev-bastion-sg \
>   --description "Security group for bastion host" \
>   --vpc-id vpc-00ddb0031795e24fd
{
    "GroupId": "sg-04ae207dbfd13cea8",
    "SecurityGroupArn": "arn:aws:ec2:us-west-1:336749236319:security-group/sg-04ae207dbfd13cea8"
}
```

Autorisation du SSH depuis le cloudshell

```
~ $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-04ae207dbfd13cea8 \
>   --protocol tcp \
>   --port 22 \
>   --cidr 13.52.247.204/32
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-09f08ff23d2571939",
            "GroupId": "sg-04ae207dbfd13cea8",
            "GroupOwnerId": "336749236319",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "13.52.247.204/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-west-1:336749236319:security-group-rule/sgr-09f08ff23d2571939"
        }
    ]
```

Autorisation connexion SSH du bastion vers l’instance APP

```
~ $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-033238da793540f1b \
>   --protocol tcp \
>   --port 22 \
>   --source-group sg-04ae207dbfd13cea8
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-02bf3d02faacef4b9",
            "GroupId": "sg-033238da793540f1b",
            "GroupOwnerId": "336749236319",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "ReferencedGroupInfo": {
                "GroupId": "sg-04ae207dbfd13cea8",
                "UserId": "336749236319"
            },
            "SecurityGroupRuleArn": "arn:aws:ec2:us-west-1:336749236319:security-group-rule/sgr-02bf3d02faacef4b9"
        }
    ]
}
```

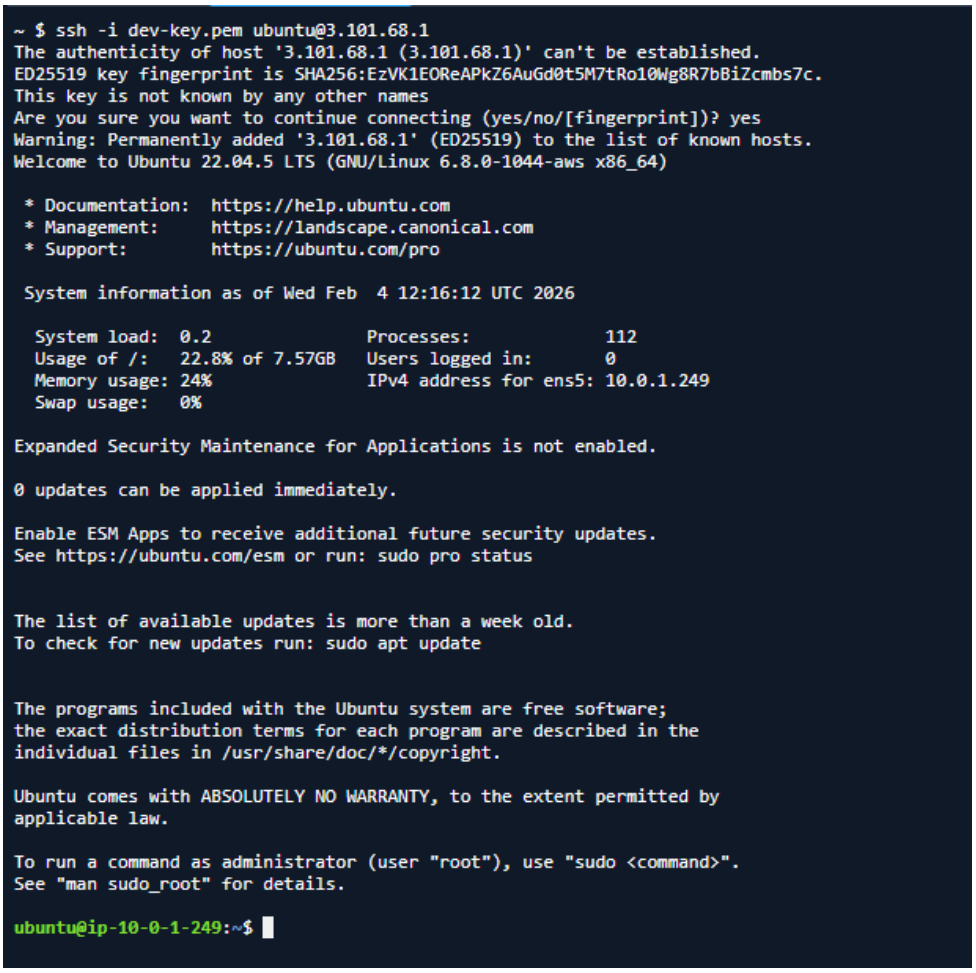
Création d’une clé SSH .pem

```
}
~ $ aws ec2 create-key-pair \
>   --key-name dev-key \
>   --query "KeyMaterial" \
>   --output text > dev-key.pem
~ $
```

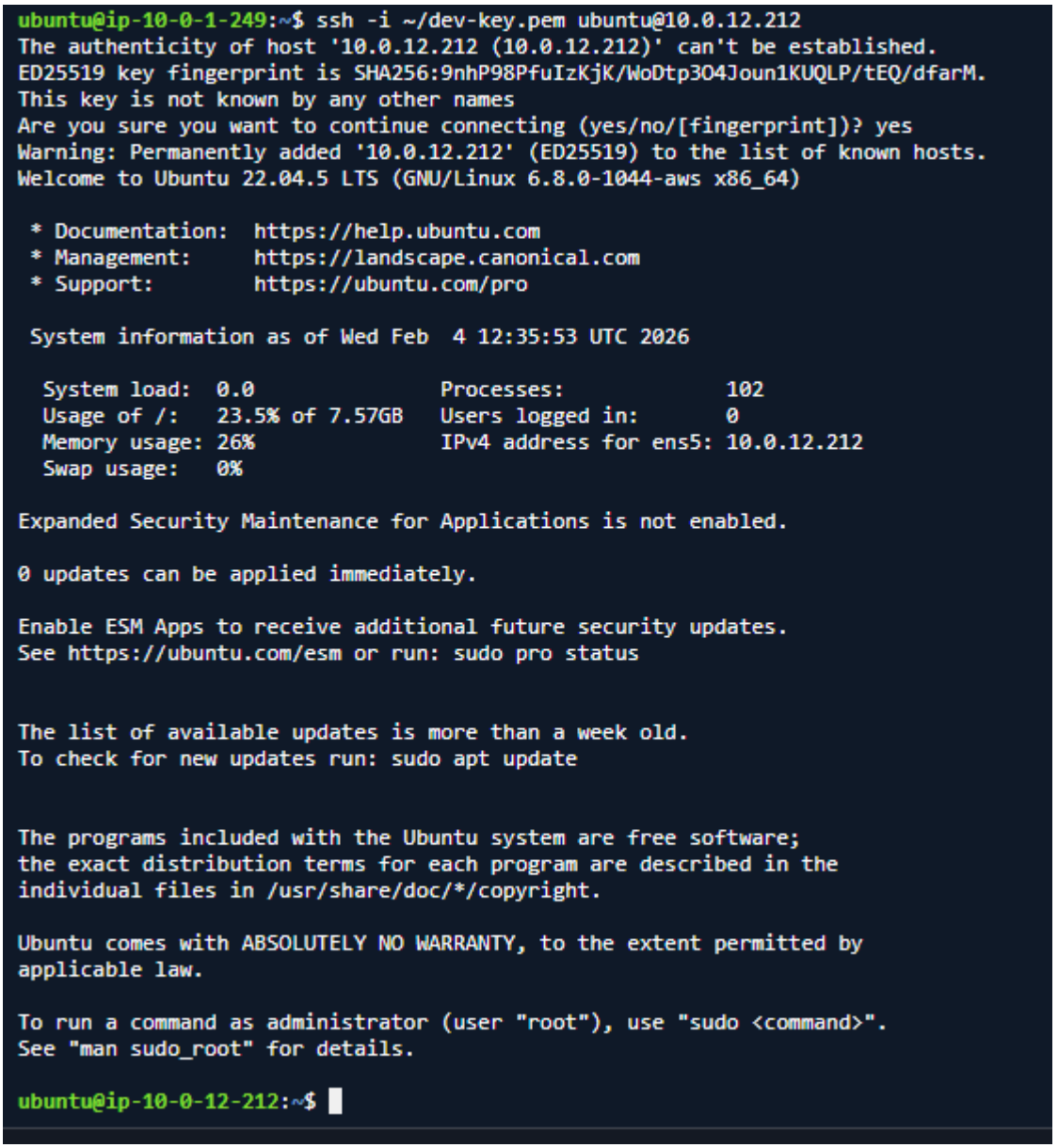
Création EC2 Bastion

```
~ $ aws ec2 run-instances \
>   --image-id ami-072028e29f8a73b88 \
>   --instance-type t3.micro \
>   --key-name dev-key \
>   --subnet-id subnet-075b27dd05884f660 \
>   --security-group-ids sg-04ae207dbfd13cea8 \
>   --associate-public-ip-address \
>   --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=dev-bastion}]'
{
    "ReservationId": "r-0ca72a053764ed603",
    "OwnerId": "336749236319",
    "Groups": [],
    "Instances": [
        {
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "ba26489f-16a1-46b9-974a-683135b04312",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2026-02-04T12:14:28+00:00",
                        "AttachmentId": "eni-attach-09585c55a96c3e818",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "Status": "attaching",
                        "NetworkCardIndex": 0
                    },
                    "Description": "",
                    "Groups": [
                        {
                            "GroupId": "sg-04ae207dbfd13cea8",
                            "GroupName": "dev-bastion-sg"
                        }
                    ],
                    "Ipv6Addresses": [],
                    "MacAddress": "02:bd:ff:f4:1a:bb",
                    "NetworkInterfaceId": "eni-082e674736ef418ba",
                    "OwnerId": "336749236319",
                    "PrivateDnsName": "ip-10-0-1-249.us-west-1.compute.internal",
```

Connexion SSH sur le Bastion



Connexion à l’instance App depuis le Bastion



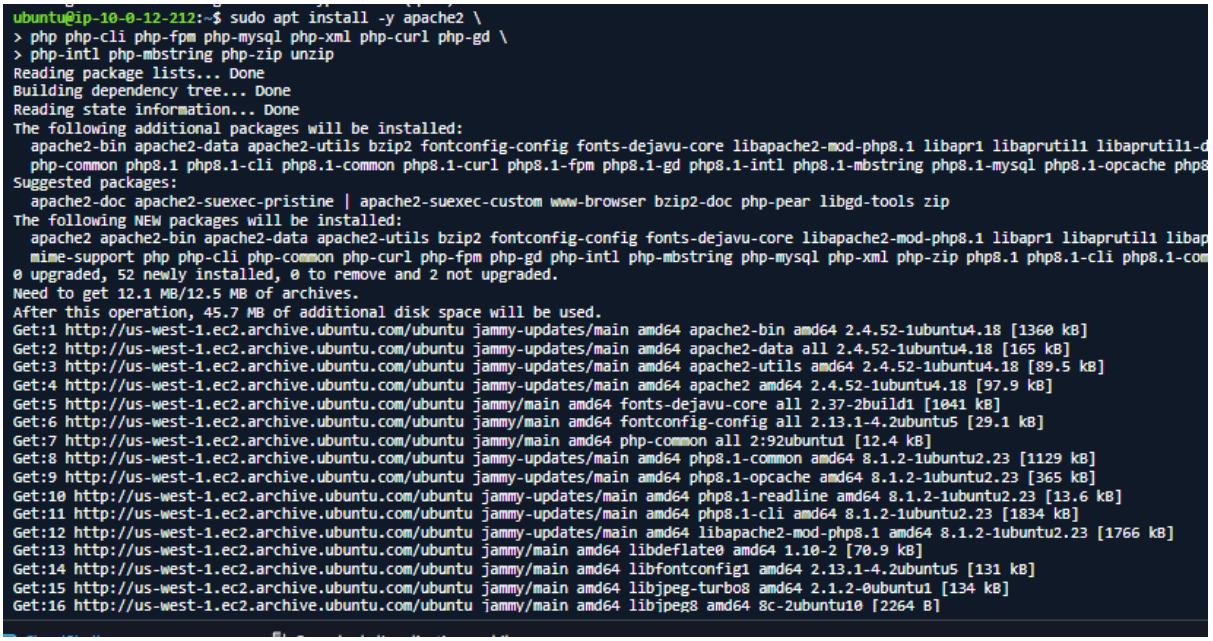
L’accès au serveur applicatif a été réalisé via un bastion host déployé dans un subnet public.
Cette méthode permet de se connecter aux instances situées dans des subnets privés sans leur attribuer d’adresse IP publique, renforçant ainsi la sécurité de l’infrastructure.
Le bastion joue le rôle de point d’entrée contrôlé, conformément aux bonnes pratiques AWS.

Déploiement de l’application e-commerce PrestaShop sur l’instance application

Cette partie a pour objectif de déployer une application e-commerce de type PrestaShop dans l’environnement AWS mis en place précédemment. L’application est installée sur une instance Ubuntu située dans des subnets privés et gérées par un Auto Scaling Group. La base de données est hébergée sur Amazon RDS MySQL afin de limiter la gestion de l’infrastructure tout en garantissant la disponibilité du service. L’accès à l’application se fait via un Application Load Balancer, conformément aux bonnes pratiques de sécurité et aux consignes du projet.

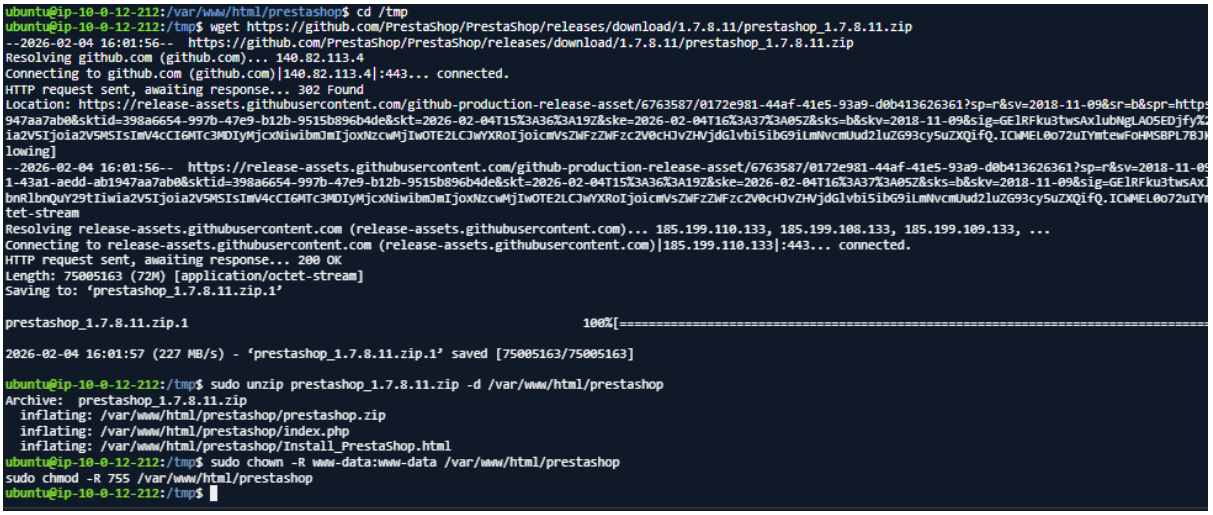
Installation serveur apache et de php

Apache est utilisé comme serveur web pour héberger PrestaShop.
PHP et ses extensions sont nécessaires au bon fonctionnement de l’application.
La base de données n’est pas installée localement car elle est externalisée sur Amazon RDS.

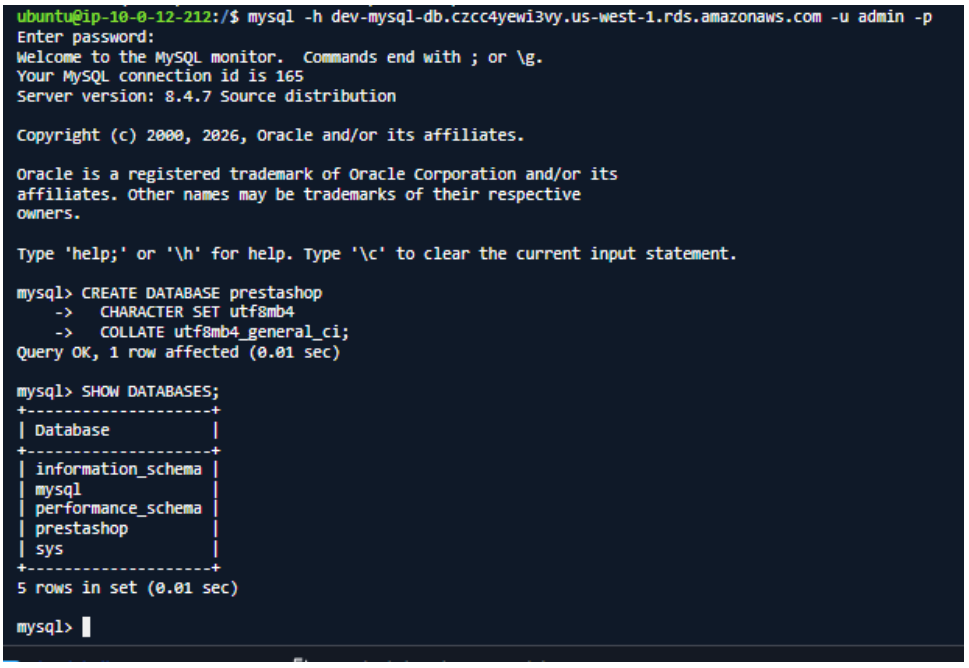


Téléchargement de PrestaShop

L'archive officielle de PrestaShop (version 1.7.8.11) a été téléchargée depuis le dépôt GitHub du projet puis décompressée dans le répertoire web /var/www/html/prestashop. Les droits d'accès ont ensuite été configurés afin de permettre au serveur web Apache d'accéder aux fichiers nécessaires à l'installation et au fonctionnement de l'application e-commerce.



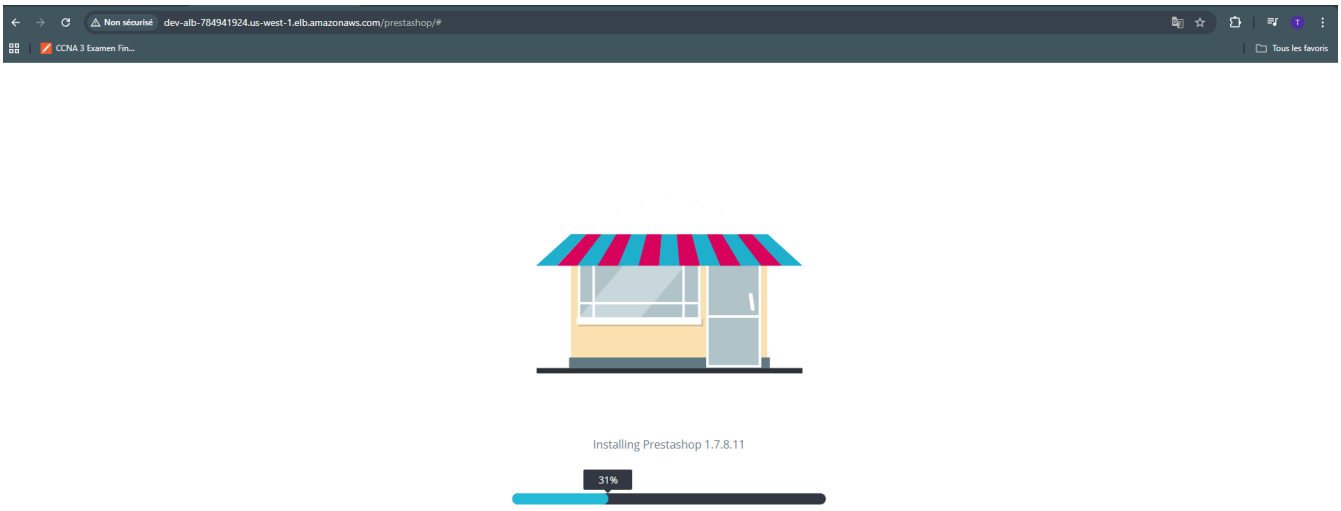
Connexion à la BDD d'Amazon RDS + Création de la table prestashop dans l'instance application



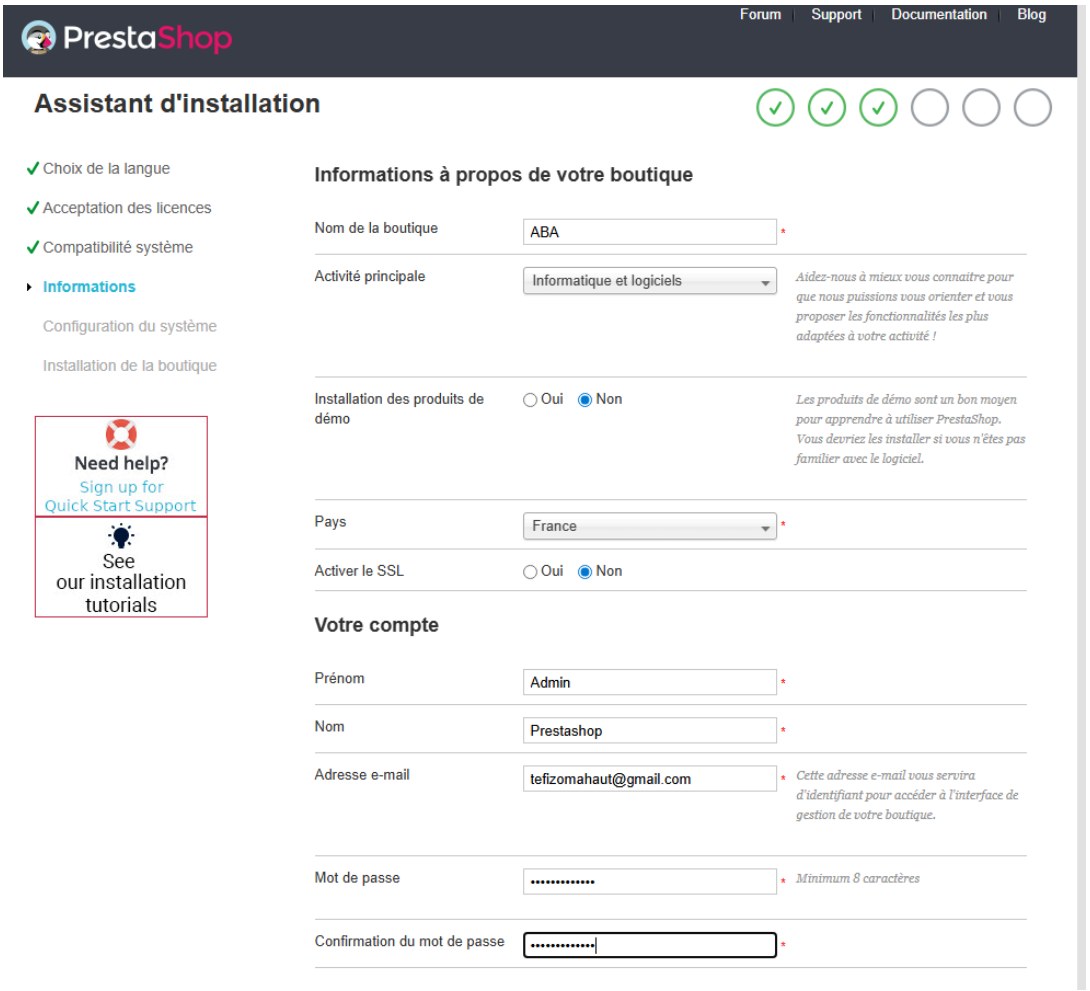
Accès à l’interface de l’application via le nom DNS notre ALB (application load balancer)

Après le déploiement des fichiers applicatifs, l’assistant d’installation de PrestaShop est lancé via le navigateur. Cette étape permet d’initialiser l’application e-commerce et de procéder à la configuration de l’environnement avant la connexion à la base de données.

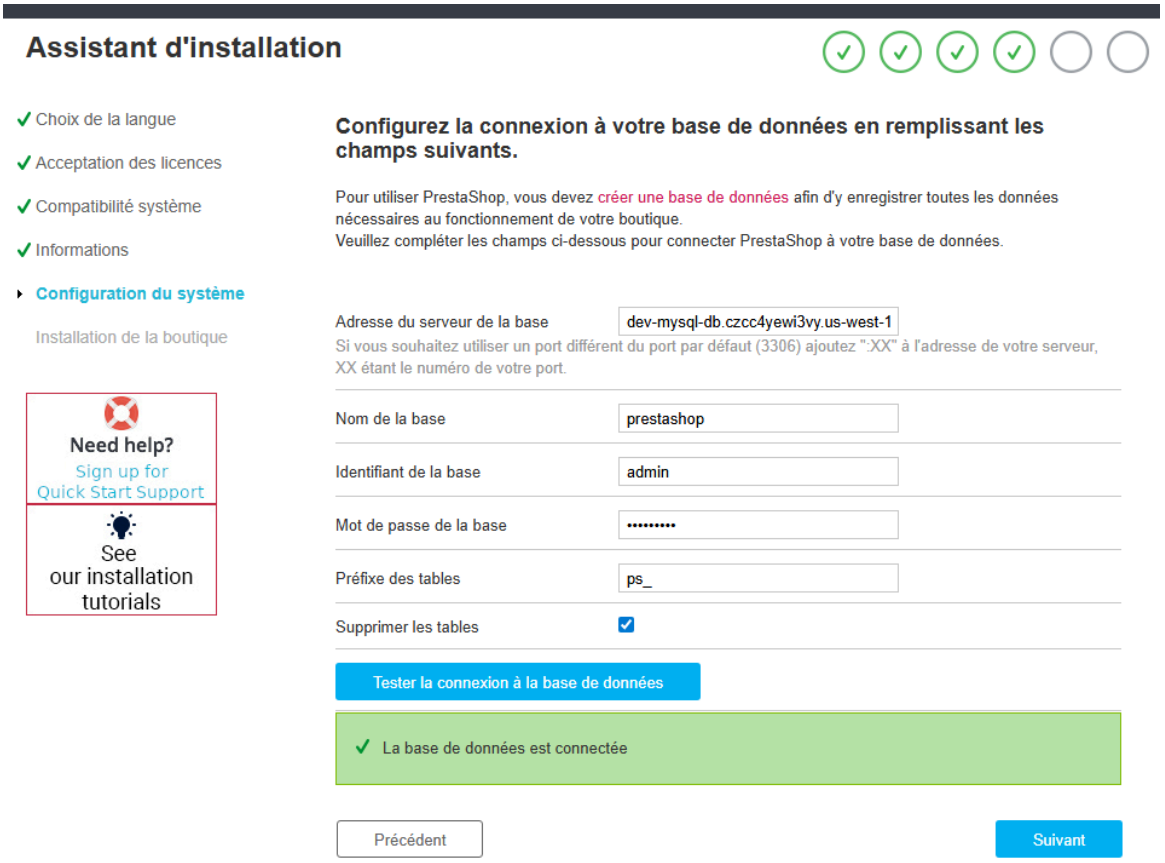
L’accès à l’application e-commerce s’effectue via le nom DNS de l’Application Load Balancer, garantissant une exposition sécurisée et centralisée du service sans accès direct aux instances

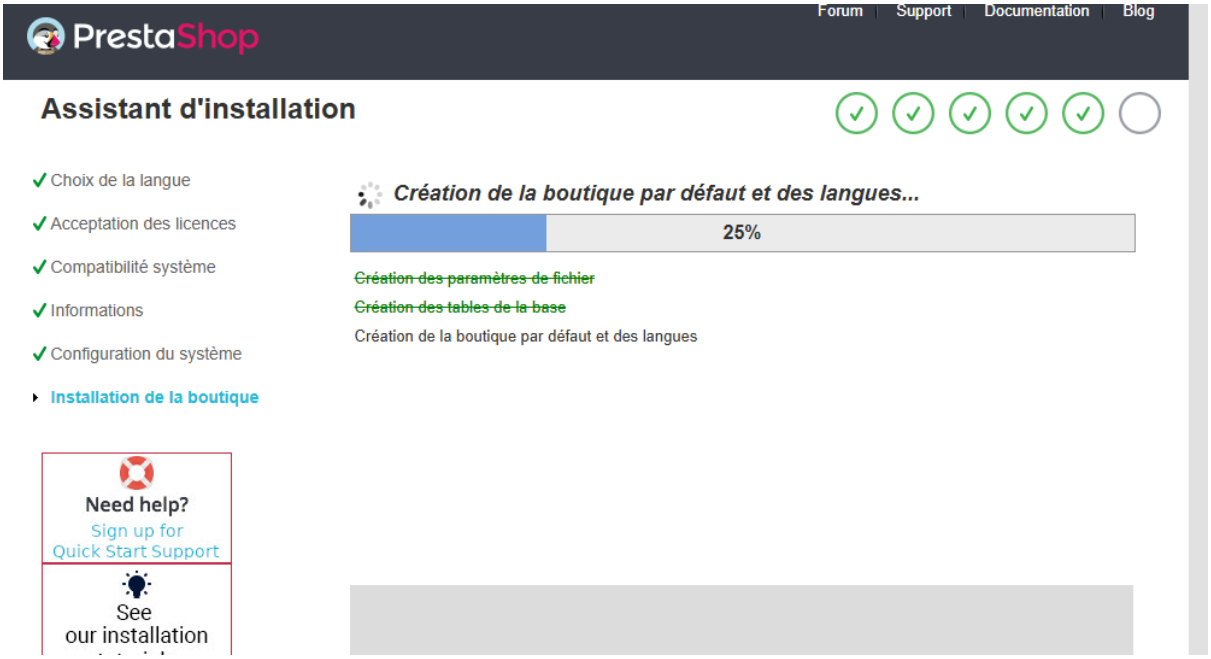


Configuration de l'application sur son interface



Configuration de l’accès à la BDD Amazon RDS depuis l’application





Partie 2 :

Contexte

Le client souhaite déployer deux nouvelles équipes :

- **Équipe IA** : développement et expérimentation de l'Intelligence Artificielle.
- **Équipe Cybersécurité** : tests et déploiement d'outils de sécurité.

Les objectifs principaux sont :

- Isoler les environnements des deux équipes tout en leur permettant certaines communications selon les besoins.
- Créer pour chaque équipe un **subnet public et un subnet privé**.
- Déployer des **instances EC2 vierges** dans chaque subnet.
- Garantir **l'accès Internet même depuis les subnets privés**.
- Permettre à l'équipe Cybersécurité de **se connecter à tous les autres environnements** pour supervision.
- Déployer une **stack GitLab** uniquement dans l'environnement de cybersécurité.
- Mettre en place un **framework de monitoring** pour vérifier si les applications sont en ligne.
- Assurer une **connexion sécurisée aux instances** sans utiliser directement les IP publiques.

Création des 2 VPC :

Objectif

Créer deux VPC distincts pour isoler les environnements IA et Cybersécurité.

Décisions prises

- **VPC IA** : CIDR 10.10.0.0/16
- **VPC Cybersécurité** : CIDR 10.20.0.0/16
- Le choix de plages /16 permet de réserver **beaucoup d'adresses IP** pour des subnets publics et privés sans risque de pénurie pour le POC.

VPC-IA

```
~ $ aws ec2 create-tags \
>   --resources $VPC_IA_ID \
>   --tags Key=Name,Value=VPC-IA

An error occurred (MissingParameter) when calling the CreateTags operation: The request must contain the parameter resourceIdSet
~ $ aws ec2 create-vpc \
>   --cidr-block 10.10.0.0/16 \
>   --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC-IA}]'
{
    "Vpc": {
        "OwnerId": "336749236319",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0473a9e0179ab1ea2",
                "CidrBlock": "10.10.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
:...skipping...
{
    "Vpc": {
        "OwnerId": "336749236319",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0473a9e0179ab1ea2",
                "CidrBlock": "10.10.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Name",
                "Value": "VPC-IA"
            }
        ],
        "VpcId": "vpc-0253d839e4d371c18",
        "State": "pending",
        "CidrBlock": "10.10.0.0/16",
        "DhcpOptionsId": "dopt-095f8d489f0fac277"
    }
```

VPC-CYBER

```
~ $ aws ec2 create-vpc \
>   --cidr-block 10.20.0.0/16 \
>   --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC-CYBER}]' \
>
{
    "Vpc": {
        "OwnerId": "336749236319",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0f42d9668b341314d",
                "CidrBlock": "10.20.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Name",
                "Value": "VPC-CYBER"
            }
        ],
        "VpcId": "vpc-055154975d24a7135",
        "State": "pending",
        "CidrBlock": "10.20.0.0/16",
        "DhcpOptionsId": "dopt-095f8d489f0fac277"
    }
}
~ $
```

Création des Subnets Privée et Publique

Objectif

Créer pour chaque équipe :

- **1 subnet public** pour le serveur accessible depuis Internet.
- **1 subnet privé** pour le serveur interne ou critique.

Décisions prises

- Subnets publics pour le trafic entrant/entrant Internet.
- Subnets privés pour les instances internes, avec accès Internet via NAT.
- Exemple de plages IP :
    - IA public : 10.10.1.0/24
    - IA privé : 10.10.2.0/24
    - Cyber public : 10.20.1.0/24
    - Cyber privé : 10.20.2.0/24

Subnet Cyber

Subnet Cyber-Public

```
~ $ aws ec2 create-subnet \
>     --vpc-id vpc-04c582adbd6b5c273 \
>     --cidr-block 10.20.1.0/24 \
>     --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Cyber-Public}]'
{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az3",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "Cyber-Public"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-0e408ae5242779ee8",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-0e408ae5242779ee8",
        "State": "available",
        "VpcId": "vpc-04c582adbd6b5c273",
        "CidrBlock": "10.20.1.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1c",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
```

Subnet Cyber-Private

```
~ $ aws ec2 create-subnet \
>     --vpc-id vpc-04c582adbd6b5c273 \
>     --cidr-block 10.20.2.0/24 \
>     --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Cyber-Private}]'

{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az3",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "Cyber-Private"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-02bb3016a7e89f008",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-02bb3016a7e89f008",
        "State": "available",
        "VpcId": "vpc-04c582adbd6b5c273",
        "CidrBlock": "10.20.2.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1c",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
~ $
```

Subnet IA

Subnet IA-Public

```
~ $ aws ec2 create-subnet \
>     --vpc-id vpc-054749ff7be656e81 \
>     --cidr-block 10.10.1.0/24 \
>     --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=IA-Public}]'
{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az3",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "IA-Public"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-0adf8d6cd1a1577c6",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-0adf8d6cd1a1577c6",
        "State": "available",
        "VpcId": "vpc-054749ff7be656e81",
        "CidrBlock": "10.10.1.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1c",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
~ $
```

Subnet IA-Private

```
~ $ aws ec2 create-subnet \
>     --vpc-id vpc-054749ff7be656e81 \
>     --cidr-block 10.10.2.0/24 \
>     --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=IA-Private}]'
{
    "Subnet": {
        "AvailabilityZoneId": "usw1-az3",
        "MapCustomerOwnedIpOnLaunch": false,
        "OwnerId": "336749236319",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "IA-Private"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-1:336749236319:subnet/subnet-0e900948db12ad81b",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-0e900948db12ad81b",
        "State": "available",
        "VpcId": "vpc-054749ff7be656e81",
        "CidrBlock": "10.10.2.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-west-1c",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
~ $
```

Création de la Gateway pour internet :

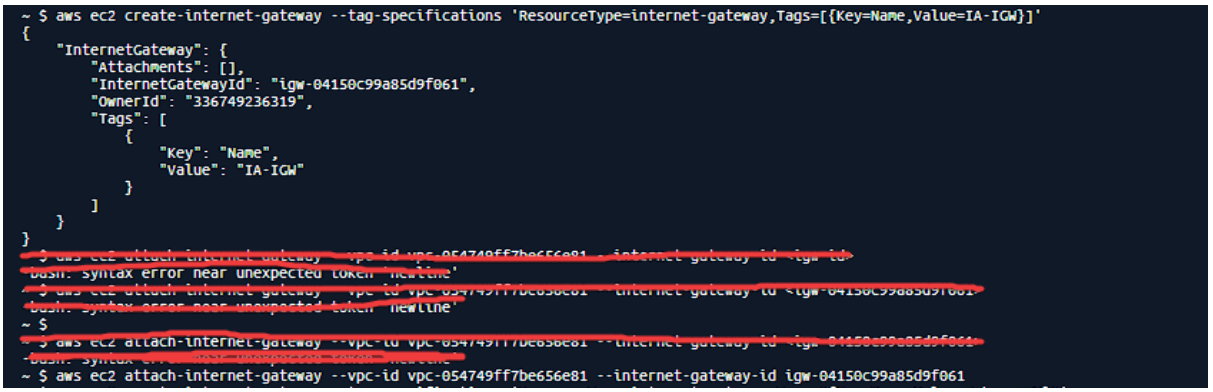
Objectif

Permettre l’accès Internet pour les subnets publics et privés.

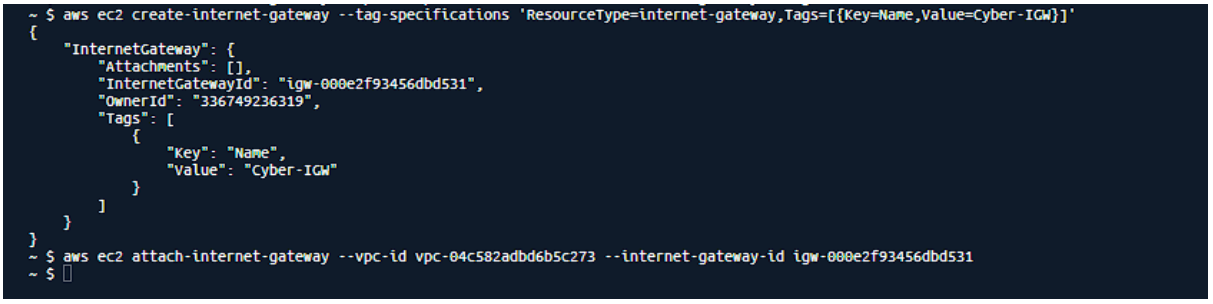
Décisions prises

- Chaque VPC a son propre **Internet Gateway (IGW)** pour les subnets publics.
- Les subnets privés utilisent une **NAT Gateway** pour sortir sur Internet tout en restant privés.

Gateway VPC-IA



Gateway VPC-Cyber



Création de la table de routage pour les subnets publiques

Objectif

Définir comment le trafic circule entre les subnets et Internet.

Décisions prises

- Chaque subnet public a une route vers l’IGW.
- Chaque subnet privé a une route vers la NAT Gateway.
- L’équipe Cybersécurité pourra aussi accéder aux autres VPC via peering ou routes inter-VPC (pour supervision).

Table de routage IA

```
~ $ aws ec2 create-route-table --vpc-id vpc-054749ff7be656e81 --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=IA-Public-RT}]'
{
    "RouteTable": {
        "Associations": [],
        "PropagatingVgws": [],
        "RouteTableId": "rtb-0f1a0ca07c5416a2f",
        "Routes": [
            {
                "DestinationCidrBlock": "10.10.0.0/16",
                "GatewayId": "local",
                "Origin": "CreateRouteTable",
                "State": "active"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "IA-Public-RT"
            }
        ],
        "VpcId": "vpc-054749ff7be656e81",
        "OwnerId": "336749236319"
    },
    "ClientToken": "ace0035d-74f2-44c7-83b7-236a17bc9e04"
}
~ $ aws ec2 create-route --route-table-id rtb-0f1a0ca07c5416a2f --destination-cidr-block 0.0.0.0/0 --gateway-id igw-04150c99a85d9f061
{
    "Return": true
}
```

```
~ $ aws ec2 associate-route-table --subnet-id subnet-0adf8d6cd1a1577c6 --route-table-id rtb-0f1a0ca07c5416a2f

{
    "AssociationId": "rtbassoc-0b204eb1ac4c4a648",
    "AssociationState": {
        "State": "associated"
    }
}
~ $
~ $
```

Table de routage Cyber

```
~ $ aws ec2 create-route-table --vpc-id vpc-04c582adbd6b5c273 --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Cyber-Public-RT}]'
{
    "RouteTable": {
        "Associations": [],
        "PropagatingVgws": [],
        "RouteTableId": "rtb-015bcc93ed17ea117",
        "Routes": [
            {
                "DestinationCidrBlock": "10.20.0.0/16",
                "GatewayId": "local",
                "Origin": "CreateRouteTable",
                "State": "active"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "Cyber-Public-RT"
            }
        ],
        "VpcId": "vpc-04c582adbd6b5c273",
        "OwnerId": "336749236319"
    },
    "ClientToken": "9bd85066-f632-4aaa-ab38-50b9f70d1b9f"
```

```
~ $ aws ec2 create-route --route-table-id rtb-015bcc93ed17ea117 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-000e2f93456dbd531
{
    "Return": true
}
```

```
~ $ aws ec2 associate-route-table --subnet-id subnet-0e408ae5242779ee8 --route-table-id rtb-015bcc93ed17ea117
{
    "AssociationId": "rtbassoc-0d4add671e2ffbf71",
    "AssociationState": {
        "State": "associated"
    }
}
~ $
```

Partie 3:

Refonte de l’infrastructure

Contexte :

Le soucis avec l’infrastructure des 2 premières partie est que :

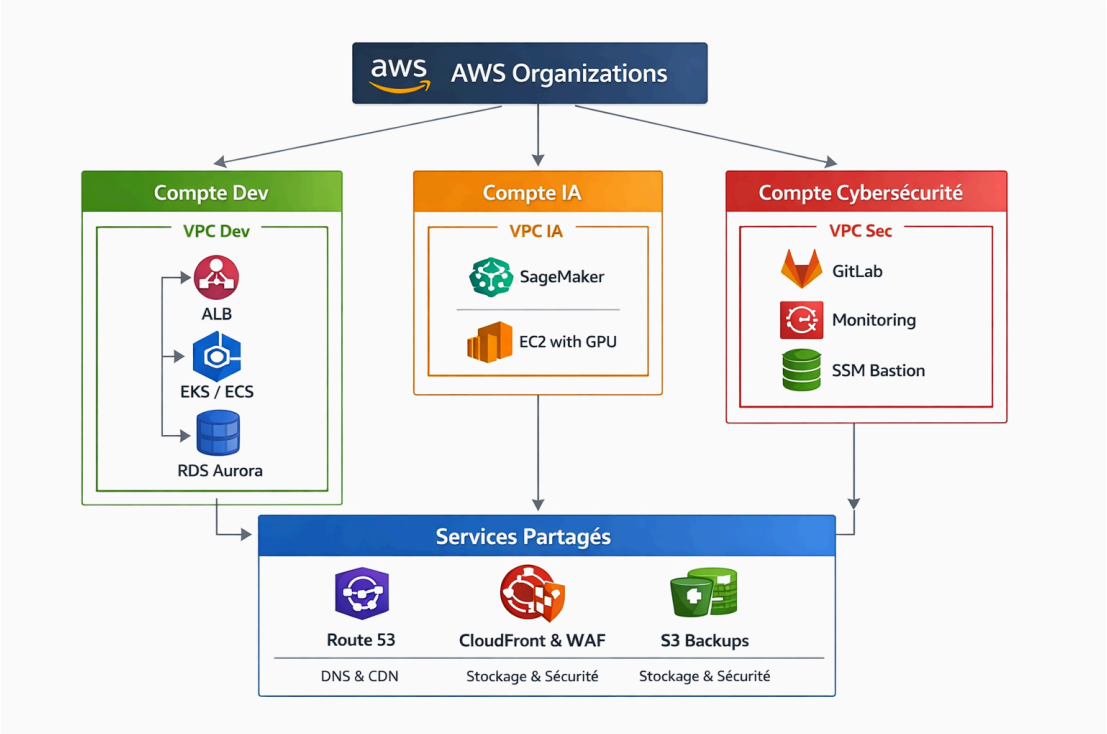
- scalabilité limitée
- pas de gouvernance multi-comptes
- gestion manuelle de certains services
- peu de tolérance aux pannes

Le client souhaite une infrastructure:

- hautement disponible
- scalable automatiquement
- sécurisée

Tout ca pour un budget moyen

Schéma réseau :



Simulation de connexion type :

L’utilisateur accède au site via Route 53
Le trafic passe par CloudFront + WAF
L’ALB distribue vers EKS/ECS
L’application accède à Aurora
Les logs vont vers CloudWatch
Les backups sont stockés sur S3
Les admins se connectent via SSM

Pour la partie dev :

Amazon EKS:

Amazon EKS permet de gérer des applications conteneurisées avec une haute disponibilité, des déploiements automatisés et une scalabilité automatique.

Points forts :

- La haute disponibilité
- Le déploiement automatisé
- La scalabilité automatique

ou

ECS Fargate :

ECS Fargate propose une solution équivalente mais plus simple à mettre en place, avec moins de gestion d’infrastructure.

Application Load Balancer (ALB) :

L’Application Load Balancer permet de gérer les connexions vers l’application en HTTPS et de répartir automatiquement la charge entre les services.
Il améliore la disponibilité et la sécurité de l’application.

Ajout d’un Amazon RDS Aurora :

Amazon RDS Aurora est une base de données managée offrant :

- des sauvegardes automatiques
- une réplication automatique
- de très bonnes performances

Il permet de réduire fortement la gestion de la base de données tout en assurant une haute disponibilité.

Pour les services partagés :

Amazon S3 :

Amazon S3 est utilisé pour stocker les sauvegardes.
Les données sont chiffrées et le coût de stockage est faible.

Points forts :
- Backups chiffrés
- Coût faible

Amazon CloudFront :

Gestionnaire du réseau de diffusion de contenu

Amazon CloudFront est un CDN qui permet :

- de réduire la charge sur le backend
- d'améliorer les performances
- de protéger contre les attaques DDoS

Et

AWS WAF:

AWS WAF permet d'appliquer un firewall applicatif afin de protéger les applications contre les attaques courantes comme les injections SQL ou le XSS.

Amazon Route 53:

Amazon Route 53 est un service DNS managé permettant :

- la gestion des noms de domaine
- le routage intelligent
- la redirection du trafic vers les services AWS

Pour la partie IA :

Amazon SageMaker :

Amazon SageMaker permet de gérer les modèles d'intelligence artificielle, depuis l'entraînement des modèles de Machine Learning jusqu'à leur déploiement en production.

Points forts :

- Entraînement des ML
- Déploiement des modèles

EC2 avec GPU :

Les instances EC2 avec GPU offrent une forte puissance de calcul pour les traitements IA. Elles sont compatibles avec les frameworks de Machine Learning et offrent une grande flexibilité.

Points forts :

- Compatibilité avec les ML
- Flexibilité
- Puissance de calcul

Pour la partie cybersécurité :

SMM Bastion :

AWS Systems Manager permet un accès sécurisé aux serveurs sans utiliser de SSH ni d'adresse IP publique, directement depuis les subnets privés.
Toutes les connexions sont tracées pour des raisons de sécurité.

Points forts :

- Accès depuis subnet privé
- Sécurité maximale
- Audit des connexions

Monitoring (CloudWatch) :

Amazon CloudWatch permet de surveiller l'infrastructure et les applications :

- CPU et RAM
- logs applicatifs
- disponibilité des services
- alertes en cas de problème

GitLab :

GitLab permet la gestion du code source, des utilisateurs et l’automatisation des déploiements via des pipelines CI/CD.

Conclusion :

Cette infrastructure repose sur des services managés AWS afin d’offrir une haute disponibilité, une forte scalabilité et une sécurité renforcée, tout en réduisant la charge opérationnelle pour les équipes techniques.