



INF8102 – Sécurité dans les environnements infonuagiques

TP4 : Infrastructure as Code Security

Soumis à :

Steph Yann TSAPI II

Soumis par :

Anis Ait-Kaci Ali – 2132752

Michlove Pierre - 2144722

Groupe : 02

Session Automne 2025

11 décembre 2025

1.

```
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4$ nano vpc.yaml
```

Figure 1: Création du fichier vpc.yaml



```
GNU nano 7.2
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4$ nano vpc.yaml
vpc.yaml

# Create a VPC with:
#   2 Public Subnets
#   2 Private Subnets
#   An Internet Gateway (with routes to it for Public Subnets)
#   A NAT Gateway for outbound access (with routes from Private Subnets set to use it)

Description: This deploys a VPC, with a pair of public and private subnets spread across two Availability Zones. It deploys an internet gateway, with a default route on the public subnets. It deploys a pair of NAT gateways (one in each AZ), and default routes for them in the private subnets.

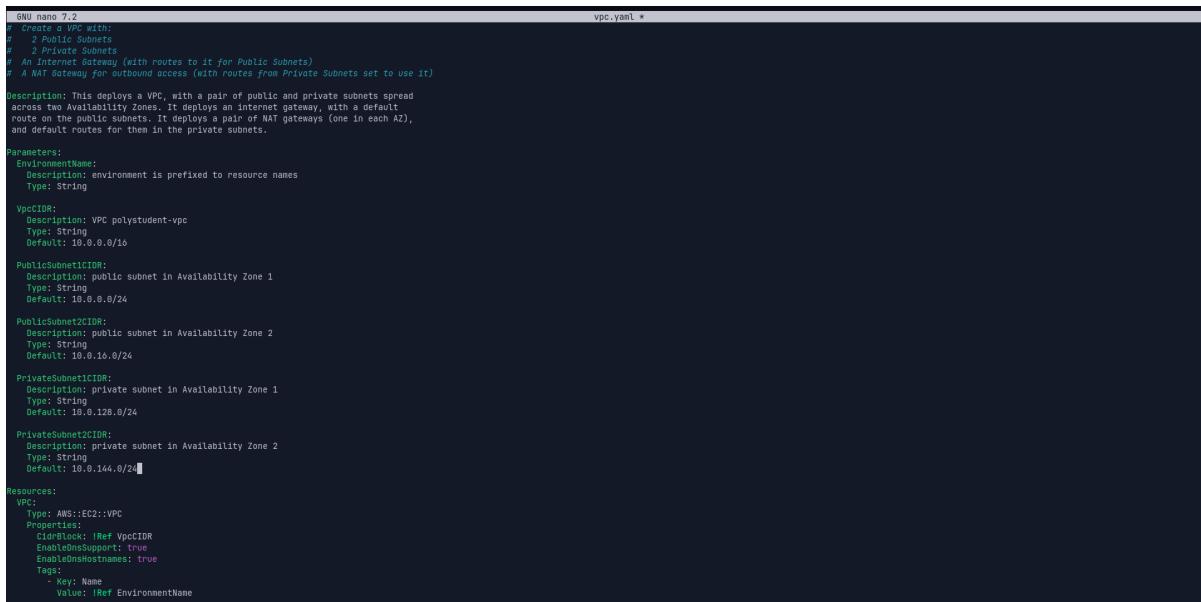
Parameters:
  EnvironmentName:
    Description: environment is prefixed to resource names
    Type: String

  VpcCIDR:
    Description: VPC polystudent-vpc
    Type: String
    Default: 10.0.0.0/16

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
```

Figure 2: Ajout de la description, des paramètres et ressources dans le fichier vpc.yaml



```
GNU nano 7.2
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4$ nano vpc.yaml
vpc.yaml *

# Create a VPC with:
#   2 Public Subnets
#   2 Private Subnets
#   An Internet Gateway (with routes to it for Public Subnets)
#   A NAT Gateway for outbound access (with routes from Private Subnets set to use it)

Description: This deploys a VPC, with a pair of public and private subnets spread across two Availability Zones. It deploys an internet gateway, with a default route on the public subnets. It deploys a pair of NAT gateways (one in each AZ), and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: environment is prefixed to resource names
    Type: String

  VpcCIDR:
    Description: VPC polystudent-vpc
    Type: String
    Default: 10.0.0.0/16

  PublicSubnet1CIDR:
    Description: public subnet in Availability Zone 1
    Type: String
    Default: 10.0.0.0/24

  PublicSubnet2CIDR:
    Description: public subnet in Availability Zone 2
    Type: String
    Default: 10.0.10.0/24

  PrivateSubnet1CIDR:
    Description: private subnet in Availability Zone 1
    Type: String
    Default: 10.0.128.0/24

  PrivateSubnet2CIDR:
    Description: private subnet in Availability Zone 2
    Type: String
    Default: 10.0.144.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName
```

Figure 3: Ajout des paramètres supplémentaire pour la création de AZ1 et AZ2 dans le fichier vpc.yaml

```

Nov 21 13:31 •
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TPA
vpc.yaml *

GNU nano 7.2
  CidrBlock: !Ref PublicSubnet1CIDR
  MapPublicIpOnLaunch: true
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Public Subnet (AZ1)
  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs "" ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)
  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs "" ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)
  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs "" ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes
  Outputs:
    VPC:
      Description: A reference to the created VPC
      Value: !Ref VPC

```

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line M-U Undo M-D Redo Set Mark To Bracket Where Was Previous Next Back Forward Prev Word Next Word Home End

Figure 4: Ajout des ressources supplémentaires pour la création de AZ1 et AZ2 dans le fichier vpc.yaml

```

Nov 21 13:33 •
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TPA
vpc.yaml *

GNU nano 7.2
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Private Subnet (AZ1)
  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs "" ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)
  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs "" ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes
  Outputs:
    VPC:
      Description: A reference to the created VPC
      Value: !Ref VPC
    PublicSubnets:
      Description: A list of the public subnets
      Value: !Join [ "", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ] ]
    PrivateSubnets:
      Description: A list of the private subnets
      Value: !Join [ "", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ] ]
    PublicSubnet1:
      Description: A reference to the public subnet in Availability Zone 1
      Value: !Ref PublicSubnet1
    PublicSubnet2:
      Description: A reference to the public subnet in Availability Zone 2
      Value: !Ref PublicSubnet2
    PrivateSubnet1:
      Description: A reference to the private subnet in Availability Zone 1
      Value: !Ref PrivateSubnet1
    PrivateSubnet2:
      Description: A reference to the private subnet in Availability Zone 2
      Value: !Ref PrivateSubnet2

```

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line M-U Undo M-D Redo Set Mark To Bracket Where Was Previous Next Back Forward Prev Word Next Word Home End

Figure 5: Ajout des configurations dans le outputs pour afficher les contenue dans le CloudFormation

```

Nov 21 13:34 •
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4
vpc.yaml *

GNU nano 7.2
VpcId: !Ref VPC
  AvailabilityZone: !Select [ 0, !GetAZs '' ]
  CidrBlock: !Ref PrivateSubnet1CIDR
  MaxPublicIpOnLaunch: False
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Private Subnet (AZ1)
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MaxPublicIpOnLaunch: False
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Routes
InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
      - Key: Name
        Value: !Ref EnvironmentName
InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ "", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ] ]
  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ "", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ] ]

```

Figure 6: Ajout de l'internet gateway dans le fichier vpc.yaml

```

Nov 21 13:35 •
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4
vpc.yaml *

GNU nano 7.2
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Routes
InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
      - Key: Name
        Value: !Ref EnvironmentName
InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC
NetGateway1IP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
NetGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NetGateway1IP.AllocationId
    SubnetId: !Ref PublicSubnet1
NetGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NetGateway2IP.AllocationId
    SubnetId: !Ref PublicSubnet2
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

```

Figure 7: Ajout du NAT gateway dans le fichier vpc.yaml

```

Nov 21 13:37 •
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4
vpc.yaml *

GNU nano 7.2
Type: AWS::EC2::VPCCAttachment
Properties:
  InternetGatewayId: !Ref InternetGateway
  VpcId: !Ref VPC

NetGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NetGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NetGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NetGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1

NetGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NetGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2

DefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2

Outputs:

```

Figure 8: Créeation de la table de routage pour le subnet public dans le fichier vpc.yaml

```

Nov 21 13:39 •
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4
vpc.yaml *

GNU nano 7.2
PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2

PrivateRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ2)

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

Outputs:

```

Figure 9: Créeation de la table de routage pour le subnet privé dans le fichier vpc.yaml

```

Nov 21 13:42 •
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/TP4$ nano vpc.yaml
GNU nano 7.2
vpc.yaml *

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ2)
DefaultPrivateRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2
PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2
IngressSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "polystudent-sp"
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - CidrIp: 0.0.0.0/0
        FromPort: 22
        ToPort: 22
        Protocol: tcp
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ] ]
  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ] ]
  PublicSubnet1:
    Description: A reference to the public subnet in Availability Zone 1
    Value: !Ref PublicSubnet1

```

Figure 10: Création du groupe de sécurité dans le fichier vpc.yaml

```

michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP4$ python3 -m venv venv
The virtual environment was not created successfully because ensurepip is not
available. On Debian/Ubuntu systems, you need to install the python3-venv
package using the following command.

apt install python3.12-venv

You may need to use sudo with that command. After installing the python3-venv
package, recreate your virtual environment.

Failing command: /home/michlove/Documents/École/INF8102/TP4/venv/bin/python3

michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP4$ sudo
  apt install python3.12-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-pip-whl python3-setuptools-whl
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.12-venv
0 upgraded, 3 newly installed, 0 to remove and 87 not upgraded.

```

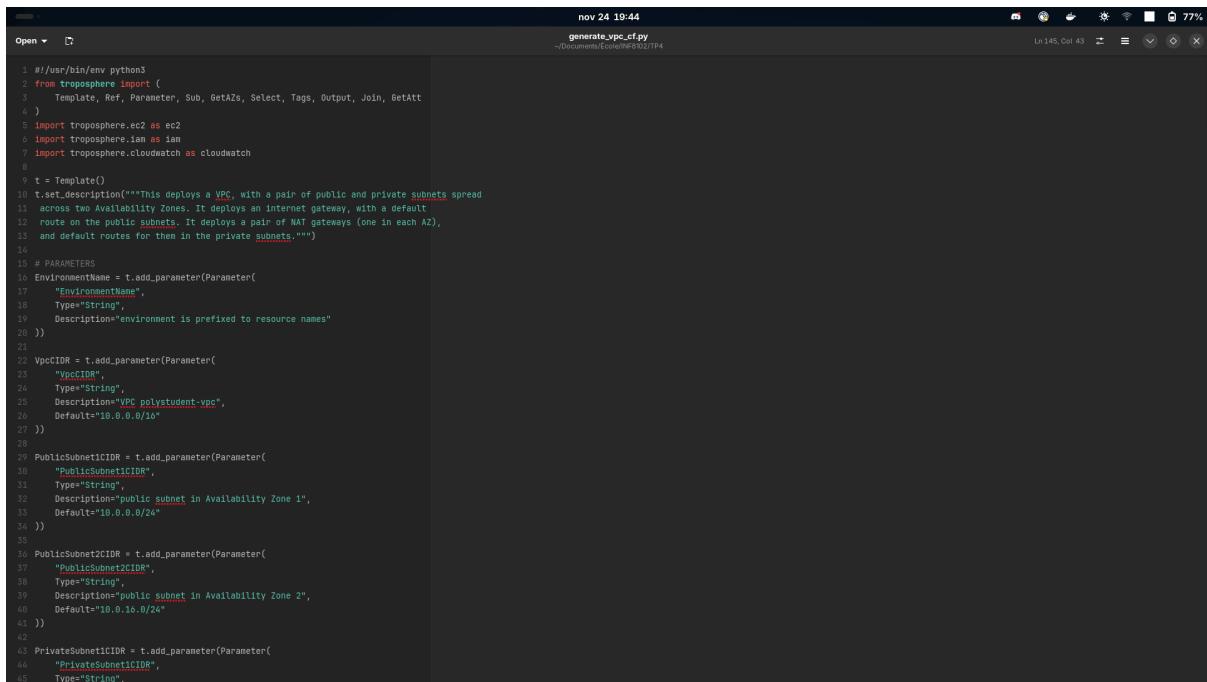
Figure 11: Installation de python3.12-venv

```

michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP4$ python3 -m venv venv
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP4$ source venv/bin/activate
(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP4$ pip install troposphere boto3 pyyaml
Collecting troposphere
  Downloading troposphere-4.9.4-py3-none-any.whl.metadata (9.2 kB)
Collecting boto3
  Downloading boto3-1.41.1-py3-none-any.whl.metadata (6.8 kB)
Collecting pyyaml
  Downloading pyyaml-6.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (2.4 kB)
Collecting cfn_flip>=1.0.2 (from troposphere)
  Downloading cfn_flip-1.3.0-py3-none-any.whl.metadata (6.0 kB)
Collecting botocore<1.42.0,>=1.41.1 (from boto3)
  Downloading botocore-1.41.1-py3-none-any.whl.metadata (5.9 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)

```

Figure 12: Activation de venv et installation de troposphere, boto3 et pyyaml

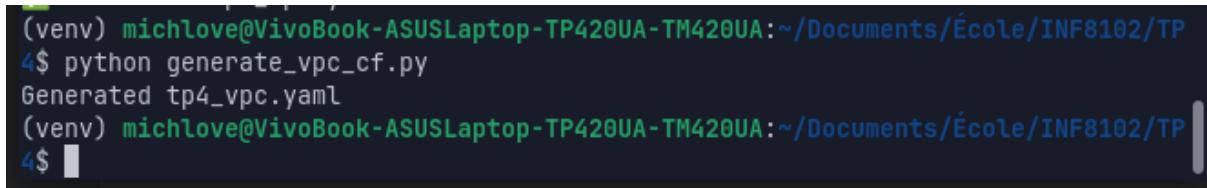


```

1 #!/usr/bin/env python3
2 from troposphere import (
3     Template, Ref, Parameter, Sub, GetAZs, Select, Tags, Output, Join, GetAtt
4 )
5 import troposphere.ec2 as ec2
6 import troposphere.iam as iam
7 import troposphere.cloudwatch as cloudwatch
8
9 t = Template()
10 t.set_description("""This deploys a VPC, with a pair of public and private subnets spread
11 across two Availability Zones. It deploys an internet gateway, with a default
12 route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
13 and default routes for them in the private subnets.""")
14
15 # PARAMETERS
16 EnvironmentName = t.add_parameter(Parameter(
17     "EnvironmentName",
18     Type="String",
19     Description="environment is prefixed to resource names"
20 ))
21
22 VpcCIDR = t.add_parameter(Parameter(
23     "VpcCIDR",
24     Type="String",
25     Description="VPC polystudent-vpc",
26     Default="10.0.0.0/16"
27 ))
28
29 PublicSubnet1CIDR = t.add_parameter(Parameter(
30     "PublicSubnet1CIDR",
31     Type="String",
32     Description="public subnet in Availability Zone 1",
33     Default="10.0.0.0/24"
34 ))
35
36 PublicSubnet2CIDR = t.add_parameter(Parameter(
37     "PublicSubnet2CIDR",
38     Type="String",
39     Description="public subnet in Availability Zone 2",
40     Default="10.0.16.0/24"
41 ))
42
43 PrivateSubnet1CIDR = t.add_parameter(Parameter(
44     "PrivateSubnet1CIDR",
45     Type="String",

```

Figure 13: Crédit du fichier generate_vpc_cf.py pour reproduire le VPC



```

(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP4$ python generate_vpc_cf.py
Generated tp4_vpc.yaml
(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP4$ 

```

Figure 14: Exécution du fichier generate_vpc_cf.py

```

1 Description: "This deploys a VPC, with a pair of public and private subnets spread\n across two Availability Zones. It deploys an internet gateway, with a default\n route on the public subnets. It deploys\n
2 \ a pair of NAT gateways (one in each AZ),\n and default routes for them in the private subnets."
3 Outputs:
4   PrivateSubnet1:
5     Description: A reference to the private subnet in Availability Zone 1
6     Value: !Ref 'PrivateSubnet1'
7   PrivateSubnet2:
8     Description: A reference to the private subnet in Availability Zone 2
9     Value: !Ref 'PrivateSubnet2'
10  PrivateSubnets:
11    Description: A list of the private subnets
12    Value: !Join
13      - [
14        - !Ref 'PrivateSubnet1'
15        - !Ref 'PrivateSubnet2'
16      ]
17  PublicSubnet1:
18    Description: A reference to the public subnet in Availability Zone 1
19    Value: !Ref 'PublicSubnet1'
20  PublicSubnet2:
21    Description: A reference to the public subnet in Availability Zone 2
22    Value: !Ref 'PublicSubnet2'
23  PublicSubnets:
24    Description: A list of the public subnets
25    Value: !Join
26      - [
27        - !Ref 'PublicSubnet1'
28        - !Ref 'PublicSubnet2'
29      ]
30  VPC:
31    Description: A reference to the created VPC
32    Value: !Ref 'VPC'
33 Parameters:
34   EnvironmentName:
35     Description: environment is prefixed to resource names
36     Type: String
37   PrivateSubnet1CIDR:
38     Default: 10.0.128.0/24
39     Description: private subnet in Availability Zone 1
40     Type: String
41   PrivateSubnet2CIDR:
42     Default: 10.0.164.0/24
43     Description: private subnet in Availability Zone 2
44     Type: String
45   PublicSubnetCIDR:

```

Figure 15: Résultat suite à l'exécution du script python

```

Resources:
40  DefaultPrivateRoute1:
Properties:
41    DestinationCidrBlock: '0.0.0.0/0'
42    NatGatewayId: !Ref 'NatGateway1'
43    RouteTableId: !Ref 'PrivateRouteTable1'
44    Type: AWS::EC2::Route
45  DefaultPrivateRoute2:
Properties:
46    DestinationCidrBlock: '0.0.0.0/0'
47    NatGatewayId: !Ref 'NatGateway2'
48    RouteTableId: !Ref 'PrivateRouteTable2'
49    Type: AWS::EC2::Route
50  DefaultPublicRoute:
51    DependsOn: InternetGatewayAttachment
Properties:
52    DestinationCidrBlock: '0.0.0.0/0'
53    GatewayId: !Ref 'InternetGateway'
54    RouteTableId: !Ref 'PublicRouteTable'
55    Type: AWS::EC2::Route
56  IngressSecurityGroup:
Properties:
57    GroupDescription: Security group allows SSH, HTTP, HTTPS, MSSQL, PostgreSQL, MySQL, RDP, DSSEC, ElasticSearch, DNS...
58    GroupName: polystudent_sg
59    SecurityGroupIngress:
60      - CidrIp: '0.0.0.0/0'
61        FromPort: 22
62        IpProtocol: tcp
63        ToPort: 22
64      - CidrIp: '0.0.0.0/0'
65        FromPort: 80
66        IpProtocol: tcp
67        ToPort: 80
68      - CidrIp: '0.0.0.0/0'
69        FromPort: 443
70        IpProtocol: tcp
71        ToPort: 443
72      - CidrIp: '0.0.0.0/0'
73        FromPort: 53
74        IpProtocol: udp
75        ToPort: 53
76      - CidrIp: '0.0.0.0/0'
77        FromPort: 53
78        IpProtocol: udp
79        ToPort: 53
80

```

Figure 16: Résultat suite à l'exécution du script python

```

100     IpProtocol: udp
101     ToPort: 53
102     - CidrIp: '0.0.0.0/0'
103     FromPort: 1433
104     IpProtocol: tcp
105     ToPort: 1433
106     - CidrIp: '0.0.0.0/0'
107     FromPort: 5432
108     IpProtocol: tcp
109     ToPort: 5432
110     - CidrIp: '0.0.0.0/0'
111     FromPort: 3386
112     IpProtocol: tcp
113     ToPort: 3386
114     - CidrIp: '0.0.0.0/0'
115     FromPort: 3389
116     IpProtocol: tcp
117     ToPort: 3389
118     - CidrIp: '0.0.0.0/0'
119     FromPort: 9200
120     IpProtocol: tcp
121     ToPort: 9200
122     - CidrIp: '0.0.0.0/0'
123     FromPort: 9300
124     IpProtocol: tcp
125     ToPort: 9300
126     VpcId: !Ref 'VPC'
127     Type: AWS::EC2::SecurityGroup
128   InternetGateway:
129     Properties:
130       Tags:
131         - Key: Name
132           Value: !Ref 'EnvironmentName'
133       Type: AWS::EC2::InternetGateway
134   InternetGatewayAttachment:
135     Properties:
136       InternetGatewayId: !Ref 'InternetGateway'
137       VpcId: !Ref 'VPC'
138     Type: AWS::EC2::VPCGatewayAttachment
139   NatGateway:

```

Figure 17: Résultat suite à l'exécution du script python

```

133   NatGateway1:
134     Properties:
135       AllocationId: !GetAtt 'NatGateway1EIP.AllocationId'
136       SubnetId: !Ref 'PublicSubnet1'
137       Type: AWS::EC2::NatGateway
138   NatGateway1EIP:
139     DependsOn: InternetGatewayAttachment
140     Properties:
141       Domain: vpc
142     Type: AWS::EC2::EIP
143   NatGateway2:
144     Properties:
145       AllocationId: !GetAtt 'NatGateway2EIP.AllocationId'
146       SubnetId: !Ref 'PublicSubnet2'
147       Type: AWS::EC2::NatGateway
148   NatGateway2EIP:
149     DependsOn: InternetGatewayAttachment
150     Properties:
151       Domain: vpc
152     Type: AWS::EC2::EIP
153   PrivateRouteTable1:
154     Properties:
155       Tags:
156         - Key: Name
157           Value: !Sub '${EnvironmentName} Private Routes (AZ1)'
158       VpcId: !Ref 'VPC'
159     Type: AWS::EC2::RouteTable
160   PrivateRouteTable2:
161     Properties:
162       Tags:
163         - Key: Name
164           Value: !Sub '${EnvironmentName} Private Routes (AZ2)'
165       VpcId: !Ref 'VPC'
166     Type: AWS::EC2::RouteTable
167   PrivateSubnet1:
168     Properties:
169       AvailabilityZone: !Select
170         - 0
171         - !GetAZs ''
172       CidrBlock: !Ref 'PrivateSubnet1CIDR'
173       MapPublicIpOnLaunch: false
174     Tags:
175       - Key: Name
176         Value: !Sub '${EnvironmentName} Private Subnet (AZ1)'

```

Figure 18: Résultat suite à l'exécution du script python

```

175     Value: !Sub "${EnvironmentName} Private Subnet (AZ1)"
176     VpcId: !Ref 'VPC'
177     Type: AWS::EC2::Subnet
178     PrivateSubnetRouteTableAssociation:
179       Properties:
180         RouteTableId: !Ref 'PrivateRouteTable1'
181         SubnetId: !Ref 'PrivateSubnet1'
182         Type: AWS::EC2::SubnetRouteTableAssociation
183     PrivateSubnet2:
184       Properties:
185         AvailabilityZone: !Select
186           - 0
187           - !GetAZs ''
188         CidrBlock: !Ref 'PrivateSubnet2CIDR'
189         MapPublicIpOnLaunch: false
190       Tags:
191         - Key: Name
192           Value: !Sub "${EnvironmentName} Private Subnet (AZ2)"
193         VpcId: !Ref 'VPC'
194         Type: AWS::EC2::Subnet
195     PrivateSubnet2RouteTableAssociation:
196       Properties:
197         RouteTableId: !Ref 'PrivateRouteTable2'
198         SubnetId: !Ref 'PrivateSubnet2'
199         Type: AWS::EC2::SubnetRouteTableAssociation
200   PublicRouteTable:
201     Properties:
202       Tags:
203         - Key: Name
204           Value: !Sub "${EnvironmentName} Public Routes"
205       VpcId: !Ref 'VPC'
206       Type: AWS::EC2::RouteTable
207   PublicSubnet1:
208     Properties:
209       AvailabilityZone: !Select
210         - 0
211         - !GetAZs ''
212       CidrBlock: !Ref 'PublicSubnet1CIDR'
213       MapPublicIpOnLaunch: true
214     Tags:
215       - Key: Name
216         Value: !Sub "${EnvironmentName} Public Subnet (AZ1)"
217         VpcId: !Ref 'VPC'
218       Type: AWS::EC2::Subnet

```

Figure 19: Résultat suite à l'exécution du script python

```

219   Type: AWS::EC2::Subnet
220   PublicSubnet1RouteTableAssociation:
221     Properties:
222       RouteTableId: !Ref 'PublicRouteTable'
223       SubnetId: !Ref 'PublicSubnet1'
224       Type: AWS::EC2::SubnetRouteTableAssociation
225   PublicSubnet2:
226     Properties:
227       AvailabilityZone: !Select
228         - 1
229         - !GetAZs ''
230       CidrBlock: !Ref 'PublicSubnet2CIDR'
231       MapPublicIpOnLaunch: true
232     Tags:
233       - Key: Name
234         Value: !Sub "${EnvironmentName} Public Subnet (AZ2)"
235         VpcId: !Ref 'VPC'
236       Type: AWS::EC2::Subnet
237   PublicSubnet2RouteTableAssociation:
238     Properties:
239       RouteTableId: !Ref 'PublicRouteTable'
240       SubnetId: !Ref 'PublicSubnet2'
241       Type: AWS::EC2::SubnetRouteTableAssociation
242   VPC:
243     Properties:
244       CidrBlock: !Ref 'VpcCIDR'
245       EnableDnsHostnames: true
246       EnableDnsSupport: true
247     Tags:
248       - Key: Name
249         Value: !Ref 'EnvironmentName'
250       Type: AWS::EC2::VPC

```

Figure 20: Résultat suite à l'exécution du script python

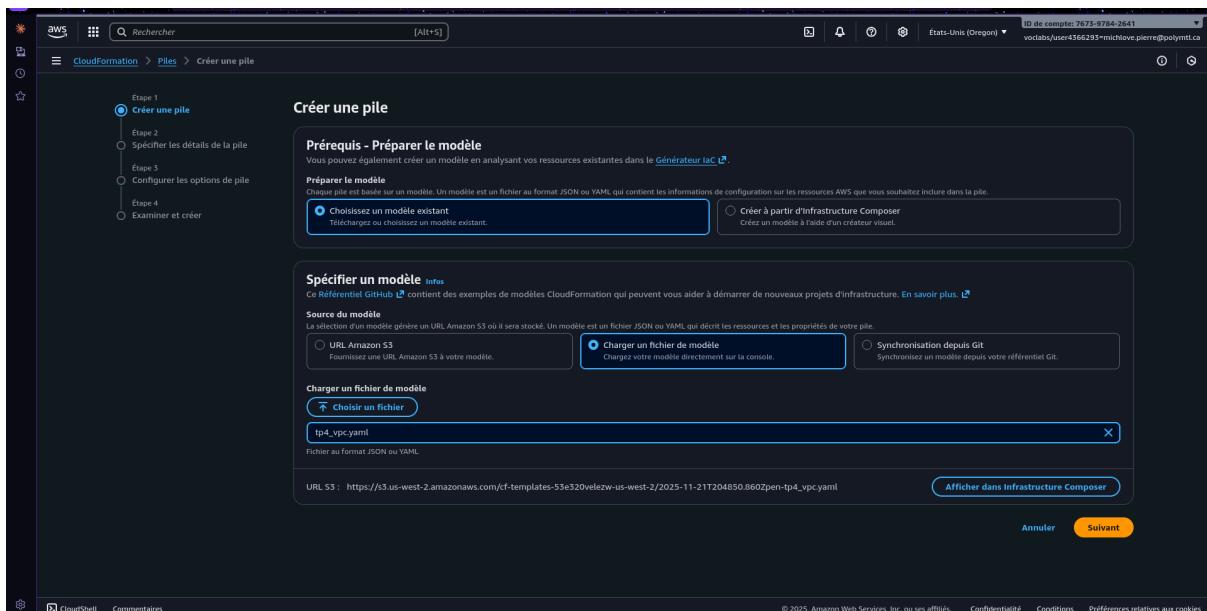


Figure 21: Crédit de la pile et chargement du fichier yaml obtenu à l'aide du script

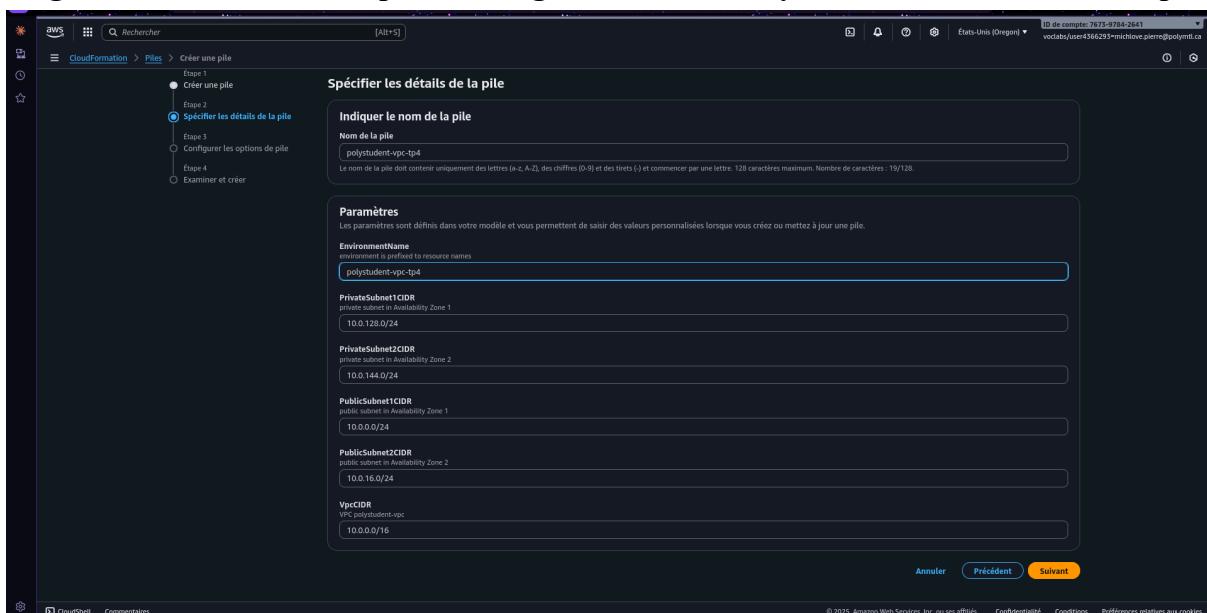


Figure 22: Inscription du nom de la pile

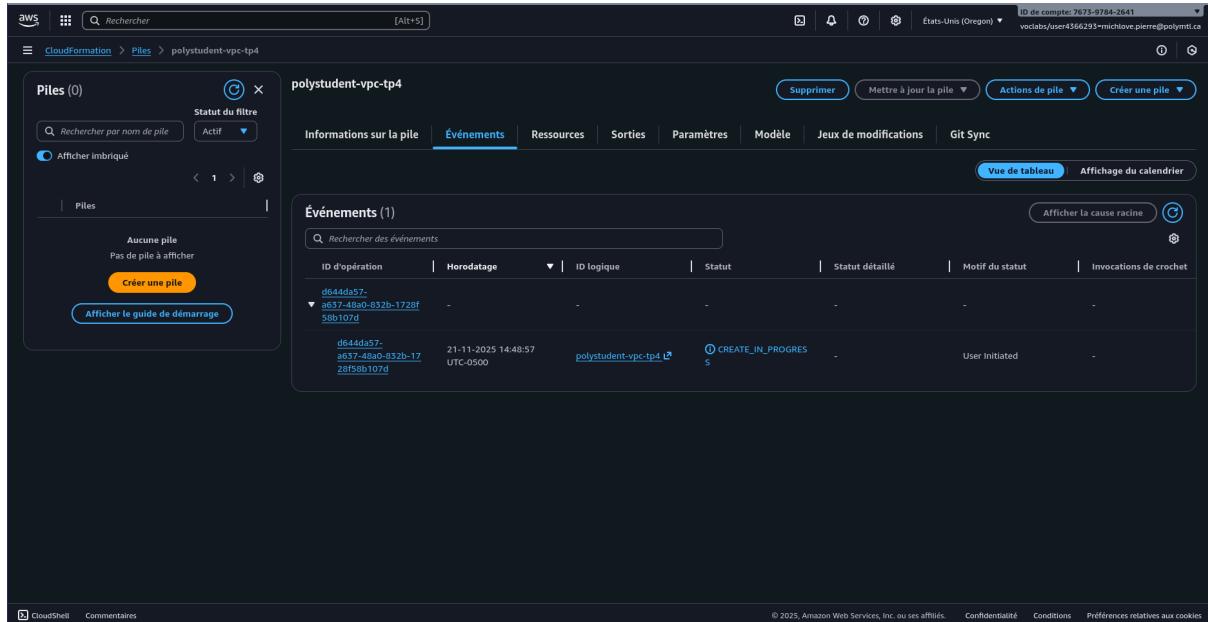


Figure 23: Démontrer que la pile est en cours d'exécution

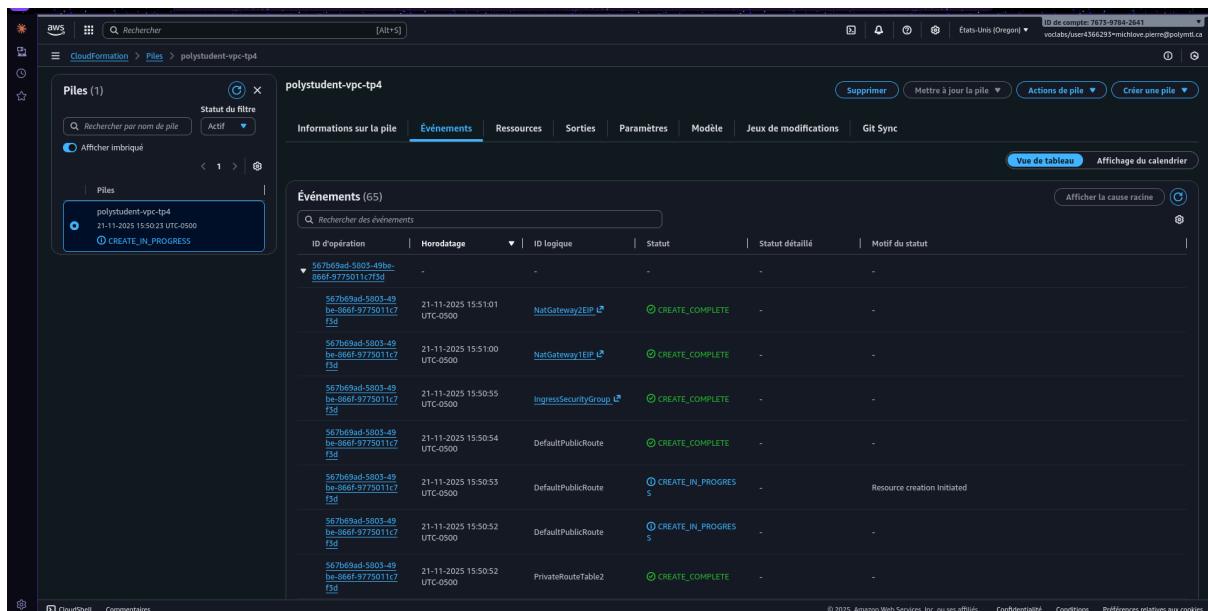


Figure 24: Démontrer que le NAT Gateway, le groupe de sécurité et les tables de routage sont bien créé à l'aide du YAML

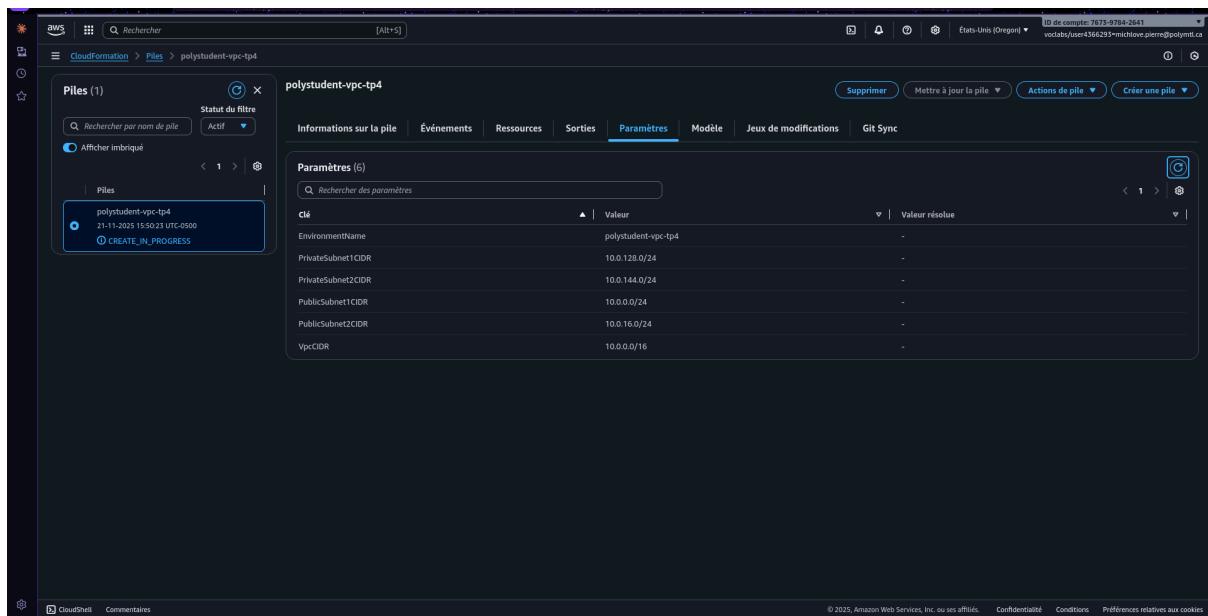


Figure 25: Afficher les paramètres créés grâce au fichier YAML

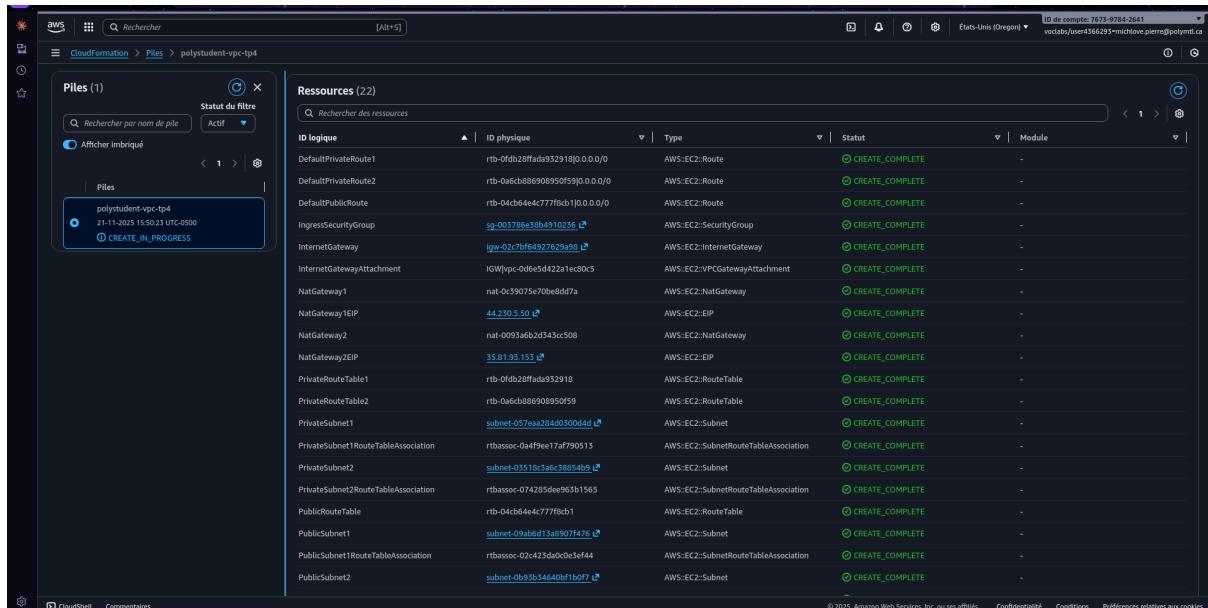


Figure 26: Afficher les ressources créées grâce au fichier YAML

Figure 27: Afficher les ressources créés grâce au fichier YAML(2)

Figure 28: Afficher les sorties créés grâce au fichier YAML

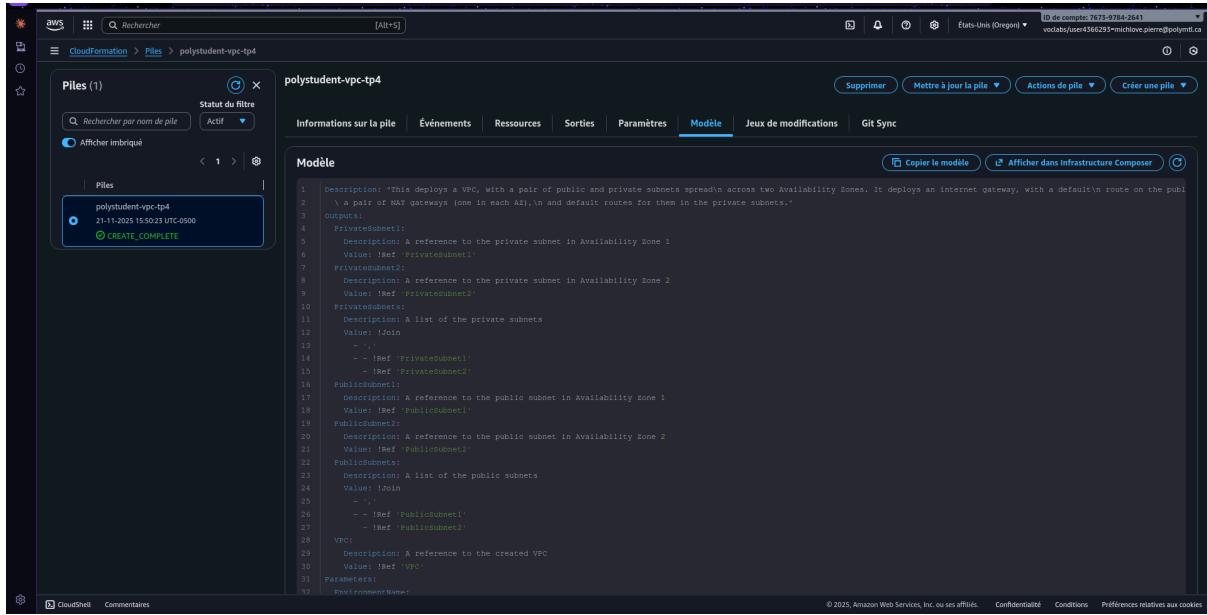


Figure 29: Afficher le modèle extrait du fichier YAML

2.

```

1 import boto3
2 from botocore.exceptions import ClientError
3
4 KMS_KEY_ARN = "arn:aws:kms:us-west-2:767597842641:key//66fd78a9-01ad-4323-8cc0-2ae608de0197"
5 BUCKET_NAME = "polystudent3-amis-nichlove-unique"
6 REGION = "us-west-2"
7
8 s3_client = boto3.client("s3", region_name=REGION)
9
10
11 def create_bucket(bucket):
12     try:
13         s3_client.create_bucket(
14             Bucket=bucket,
15             CreateBucketConfiguration={"LocationConstraint": REGION}
16         )
17         print(f"Bucket {BUCKET_NAME} créé avec succès dans la région {REGION}.")
18     except ClientError as e:
19         if e.response["Error"]["Code"] in ["BucketAlreadyOwnedByYou", "BucketAlreadyExists"]:
20             print(f"Le Bucket {bucket} existe déjà.")
21         else:
22             raise e
23
24 def secure_bucket(bucket):
25     s3_client.put_bucket_versioning(
26         Bucket=bucket,
27         VersioningConfiguration={"Status": "Enabled"}
28     )
29     print(f"Versioning activé pour {bucket}.")
30
31 if bucket == BUCKET_NAME:
32     s3_client.put_bucket_encryption(
33         Bucket=bucket,
34         ServerSideEncryptionConfiguration={
35             "Rules": [
36                 {"ApplyServerSideEncryptionByDefault": {
37                     "SSEAlgorithm": "aws:kms",
38                     "KMSMasterKeyID": KMS_KEY_ARN
39                 }
40             ]
41         }
42     )
43     print(f"chiffrement SSE-KMS actif pour {BUCKET_NAME} avec la clé {KMS_KEY_ARN}.")

```

Figure 30: Création du fichier generate_s3_bucket.py

```

1     Bucket=bucket,
2     CreateBucketConfiguration={"LocationConstraint": REGION}
3   )
4   print(f"Bucket {BUCKET_NAME} créé avec succès dans la région {REGION}.")
5   except ClientError as e:
6     if e.response["Error"]["Code"] in ["BucketAlreadyOwnedByYou", "BucketAlreadyExists"]:
7       print(f"Bucket {bucket} existe déjà.")
8     else:
9       raise e
10
11 def secure_bucket(bucket):
12   s3_client.put_bucket_versioning(
13     Bucket=bucket,
14     VersioningConfiguration={"Status": "Enabled"}
15   )
16   print(f"Versioning activé pour {bucket}.")
17
18   if bucket == BUCKET_NAME:
19     s3_client.put_bucket_encryption(
20       Bucket=bucket,
21       ServerSideEncryptionConfiguration={
22         "Rules": [
23           {"ApplyServerSideEncryptionByDefault": {
24             "AES256Algorithm": "aws:kms",
25             "KMSMasterKeyID": KMS_KEY_ARN
26           }
27         ]
28       }
29     )
30   print(f"Chiffrement SSE-KMS activé pour {BUCKET_NAME} avec la clé {KMS_KEY_ARN}.")
31
32   s3_client.put_public_access_block(
33     Bucket=bucket,
34     PublicAccessBlockConfiguration={
35       "BlockPublicAccess": True,
36       "IgnorePublicACLE": True,
37       "BlockPublicPolicy": True,
38       "RestrictPublicBuckets": True
39     }
40   )
41   print(f"Accès public bloqué pour {bucket}.")
42   print(f"Bucket {BUCKET_NAME} configuré avec succès selon les exigences de sécurité.")
43   print(f"Politiques appliquées : Versioning, SSE-KMS, Contrôle d'accès privé, Blocage d'accès public.")
44

```

Figure 31: Création du fichier generate_s3_bucket.py (2)

```

(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP
$ python generate_s3_bucket.py
Bucket polystudents3-anis-michlove-unique créé avec succès dans la région us-west-2.
Versioning activé pour polystudents3-anis-michlove-unique.
Chiffrement SSE-KMS activé pour polystudents3-anis-michlove-unique avec la clé arn:aws:kms:us-west-2:767397842641:key/44fd78a9-01ad-4323-8ccd-24e608de0197.
Accès public bloqué pour polystudents3-anis-michlove-unique.
Bucket polystudents3-anis-michlove-unique configuré avec succès selon les exigences de sécurité.
Politiques appliquées : Versioning, SSE-KMS, Contrôle d'accès privé, Blocage d'accès public.
(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP

```

Figure 32: Exécution du fichier generate_s3_bucket.py

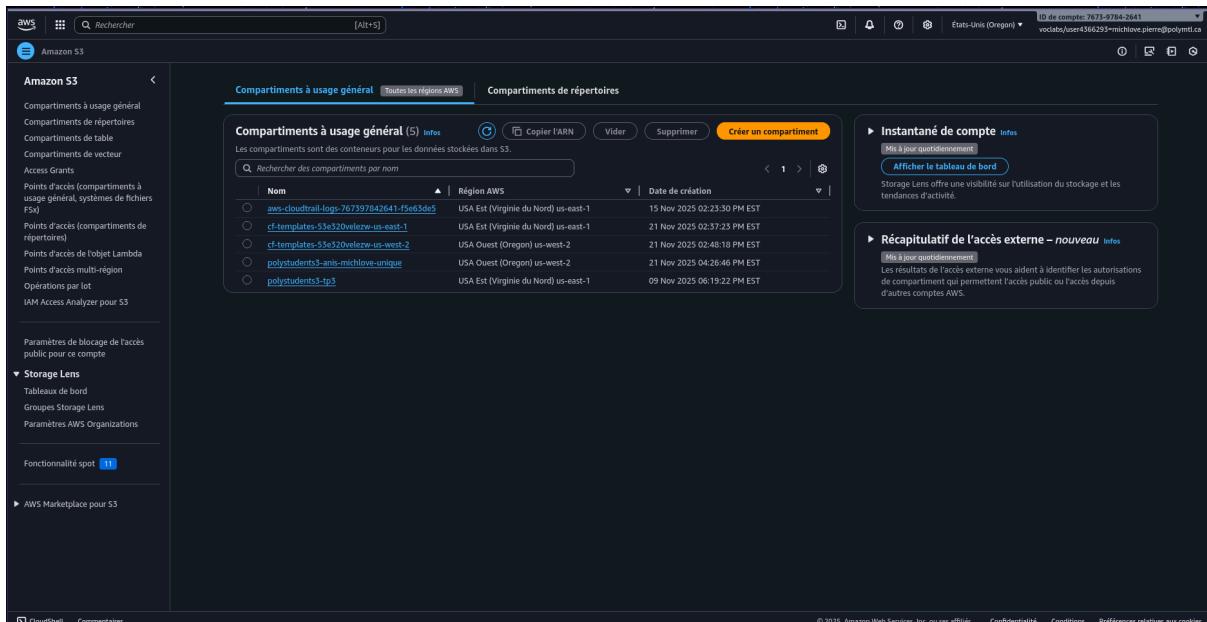


Figure 33: Démontrer que le bucket S3 polystudents3-anis-michlove-unique est créé

3.1

```

87     MapPublicIpOnLaunch=False,
88     Tags=Tags(Name=Sub("${EnvironmentName} Private Subnet (AZ1)"))
89   )
90
91 PrivateSubnet2 = t.add_resource(ec2.Subnet(
92   "PrivateSubnet2",
93   VpcId=Ref(VPC),
94   AvailabilityZone=Select(1, GetAZs()),
95   CidrBlock=Ref("PrivateSubnet2CIDR"),
96   MapPublicIpOnLaunch=False,
97   Tags=Tags(Name=Sub("${EnvironmentName} Private Subnet (AZ2)"))
98 ))
99
100 VPCFlowLogToS3 = t.add_resource(ec2.FlowLog(
101   "VPCFlowLogToS3",
102   ResourceType="VPC",
103   ResourceId=Ref(VPC),
104   TrafficType="REJECT",
105   LogDestination="arn:aws:s3:::polystudents3-anis-michlove-unique",
106   LogDestinationType="s3"
107 ))
108
109 # Internet Gateway
110 InternetGateway = t.add_resource(ec2.InternetGateway(
111   "InternetGateway",
112   Tags=Tags(Name=Ref(EnvironmentName))
113 ))
114
115 InternetGatewayAttachment = t.add_resource(ec2.VPCGatewayAttachment(
116   "InternetGatewayAttachment",
117   InternetGatewayId=Ref(InternetGateway),
118   VpcId=Ref(VPC)
119 ))
120
121 # Route Tables
122 PublicRouteTable = t.add_resource(ec2.RouteTable(
123   "PublicRouteTable",
124   VpcId=Ref(VPC),
125   Tags=Tags(Name=Sub("${EnvironmentName} Public Routes"))
126 ))
127
128 t.add_resource(ec2.Route(
129   "DefaultPublicRoute",
130   DestinationCidrBlock="0.0.0.0/0",
131   GatewayId=Ref(InternetGateway),
132   RouteTableId=Ref(PublicRouteTable)
133 ))

```

Figure 34: Modification du fichier generate_vpc_cf.py pour ajouter le VPC flow logs

```

1 REGION = "us-west-2"
2 BUCKET_NAME = "polystudents3-anis-michlove-unique"
3 KMS_KEY_ARN = "arn:aws:kms:us-west-2:767397842641:key/44fd78a9-01ad-4323-8cc0-24e088de0197"
4
5 s3_client = boto3.client("s3", region_name=REGION)
6
7 flow_log_policy = {
8   "Version": "2012-10-17",
9   "Statement": [
10     {
11       "Effect": "Allow",
12       "Principal": {"Service": "delivery.logs.amazonaws.com"},
13       "Action": ["s3:PutObject", "s3:PutObjectAcl"],
14       "Resource": f"arn:aws:s3:::{BUCKET_NAME}/*",
15       "Condition": {"StringEquals": {"s3:amz-acl": "Bucket-owner-full-control"}}
16     },
17     {
18       "Effect": "Allow",
19       "Principal": {"Service": "delivery.logs.amazonaws.com"},
20       "Action": ["s3:GetBucketAcl"],
21       "Resource": f"arn:aws:s3:::{BUCKET_NAME}"
22     }
23   ]
24 }
25
26 def create_bucket(bucket):
27   try:
28     s3_client.create_bucket(
29       Bucket=bucket,
30       CreateBucketConfiguration={"LocationConstraint": REGION}
31     )
32     print(f"Bucket {BUCKET_NAME} créé avec succès dans la région {REGION}.")
33   except ClientError as e:
34     if e.response["Error"]["Code"] in ["BucketAlreadyOwnedByYou", "BucketAlreadyExists"]:
35       print(f"Bucket {BUCKET_NAME} existe déjà. Ajustez de la politique S3 pour les Flow Logs...")
36       s3_client.put_bucket_policy(
37         Bucket=BUCKET_NAME,
38         Policy=json.dumps(flow_log_policy)
39       )
40     print(f"Politique S3 pour les Flow Logs ajoutée au bucket {BUCKET_NAME}.")
41   else:
42     raise e
43
44 def service_bucket(bucket):
45

```

Figure 35: Modification du fichier generate_s3_bucket.py pour ajouter une politique dans S3 pour faire fonctionner flow logs

```

(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP
$ python generate_vpc_cf.py
Generated tp4_vpc.yaml

```

Figure 35: Exécution du fichier generate_vpc_cf.py

```
(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/Ecole/INF8102/TP
$ python generate_s3_bucket.py
Bucket polystudents3-anis-michlove-unique existe déjà. Ajout de la politique S3
pour les Flow Logs...
Politique S3 pour les Flow Logs ajoutée au bucket polystudents3-anis-michlove-un
ique.
```

Figure 35: Exécution du fichier generate_s3_bucket.py

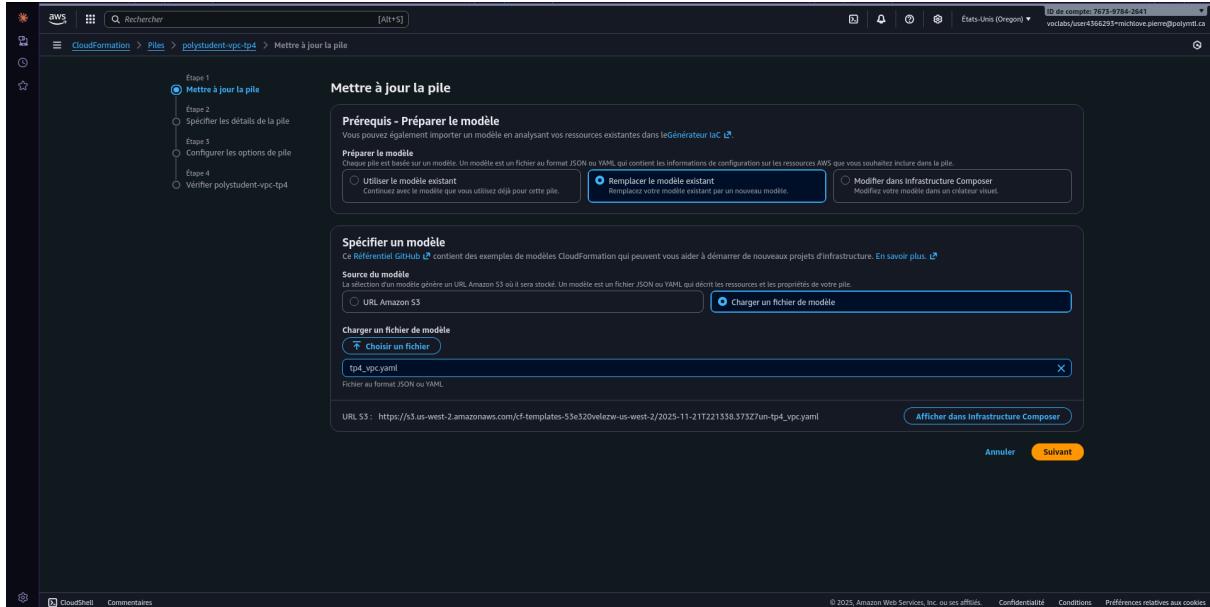


Figure 36: Mise à jour de la pile et chargement du fichier yaml obtenu à l'aide du script

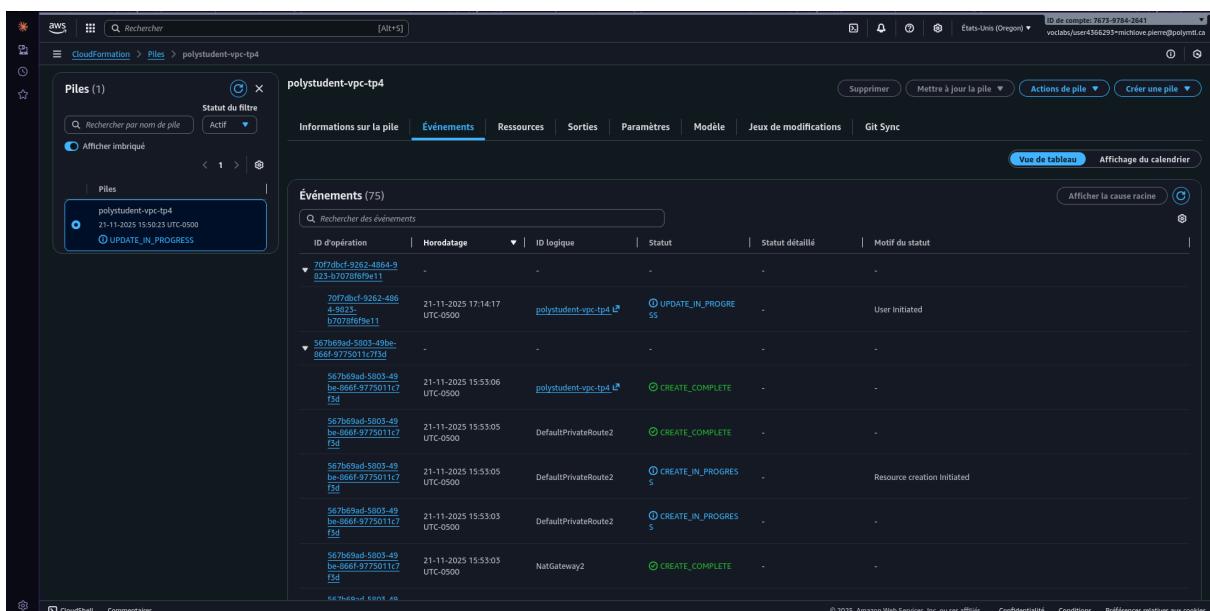


Figure 37: Démontrer que la pile est en cours de mise à jour

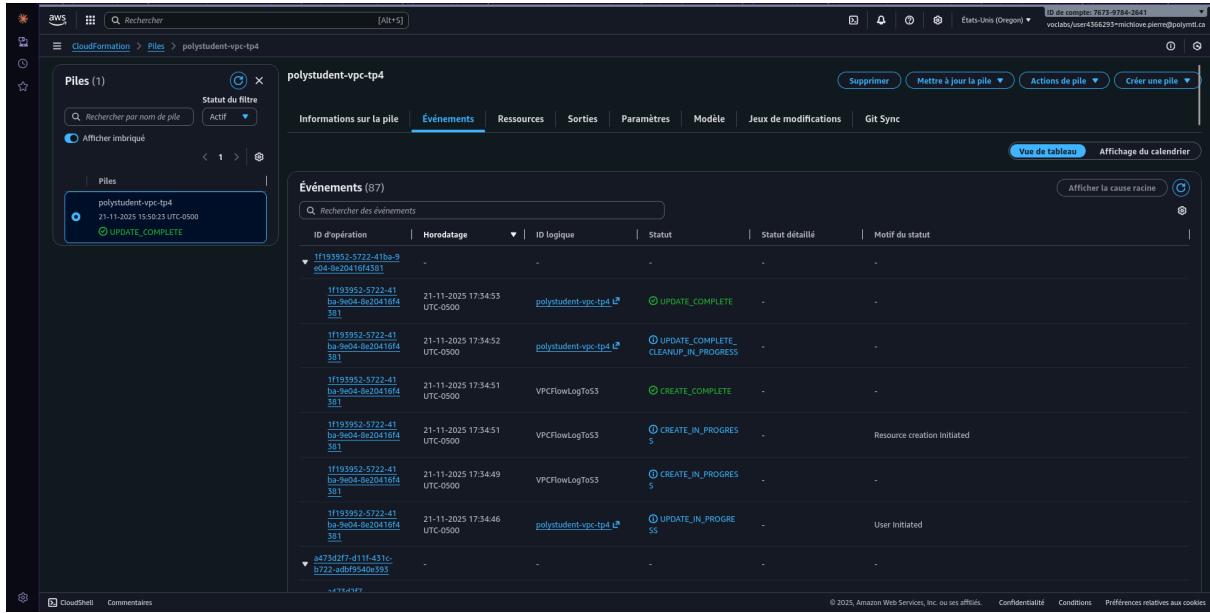


Figure 38: Démontrer que la pile à été mise à jour avec succès

3.2

```
# Public Instances
PublicInstance1 = t.add_resource(ec2.Instance(
    "PublicInstance1",
    ImageId="ami-0ff44cc01cbfb363e",
    InstanceType="t3.micro",
    SubnetId=Ref(PublicSubnet1),
    IamInstanceProfile="LambdaInstanceProfile",
    Tags=Tags(Name=Sub("${!EnvironmentName} Public Instance 1"))
))

PublicInstance2 = t.add_resource(ec2.Instance(
    "PublicInstance2",
    ImageId="ami-0ff44cc01cbfb363e",
    InstanceType="t3.micro",
    SubnetId=Ref(PublicSubnet2),
    IamInstanceProfile="LambdaInstanceProfile",
    Tags=Tags(Name=Sub("${!EnvironmentName} Public Instance 2"))
))

# Private Instances
PrivateInstance1 = t.add_resource(ec2.Instance(
    "PrivateInstance1",
    ImageId="ami-0ff44cc01cbfb363e",
    InstanceType="t3.micro",
    SubnetId=Ref(PrivateSubnet1),
    IamInstanceProfile="LambdaInstanceProfile",
    Tags=Tags(Name=Sub("${!EnvironmentName} Private Instance 1"))
))

PrivateInstance2 = t.add_resource(ec2.Instance(
    "PrivateInstance2",
    ImageId="ami-0ff44cc01cbfb363e",
    InstanceType="t3.micro",
    SubnetId=Ref(PrivateSubnet2),
    IamInstanceProfile="LambdaInstanceProfile",
    Tags=Tags(Name=Sub("${!EnvironmentName} Private Instance 2"))
))
```

... (continuation of the code block)

Figure 39: Mise à jour du script python generate_vpc_cf.py pour mettre en place des instances EC2 public et privé

```
for i, instance in enumerate([PublicInstance1, PublicInstance2, PrivateInstance1, PrivateInstance2], 1):
    t.add_resource(cloudwatch.Alarm(
        f"IngressPacketsAlarm{i}",
        AlarmDescription=f"Alarm if ingress packets exceed 1000 pps/sec on instance {i}",
        Namespace="AWS/ECS",
        MetricName="NetworkPacketsIn",
        Dimensions=[cloudwatch.MetricDimension(
            Name="InstanceId",
            Value=Ref(instance)
        )],
        Statistic="Average",
        Period=60,
        EvaluationPeriods=1,
        Threshold=1000,
        ComparisonOperator="GreaterThanOrEqualToThreshold",
        AlarmActions=[],
        OKActions=[]
    ))
```

Figure 40: Mise à jour du script python generate_vpc_cf.py pour mettre en place des alarmes aux instances EC2 public et privé

```

(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP
$ python generate_vpc_cf.py
Generated tp4_vpc.yaml
(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/École/INF8102/TP
$ 

```

Figure 41: Exécution du script afin de mettre à jour le code du script generate_vpc_cf.py

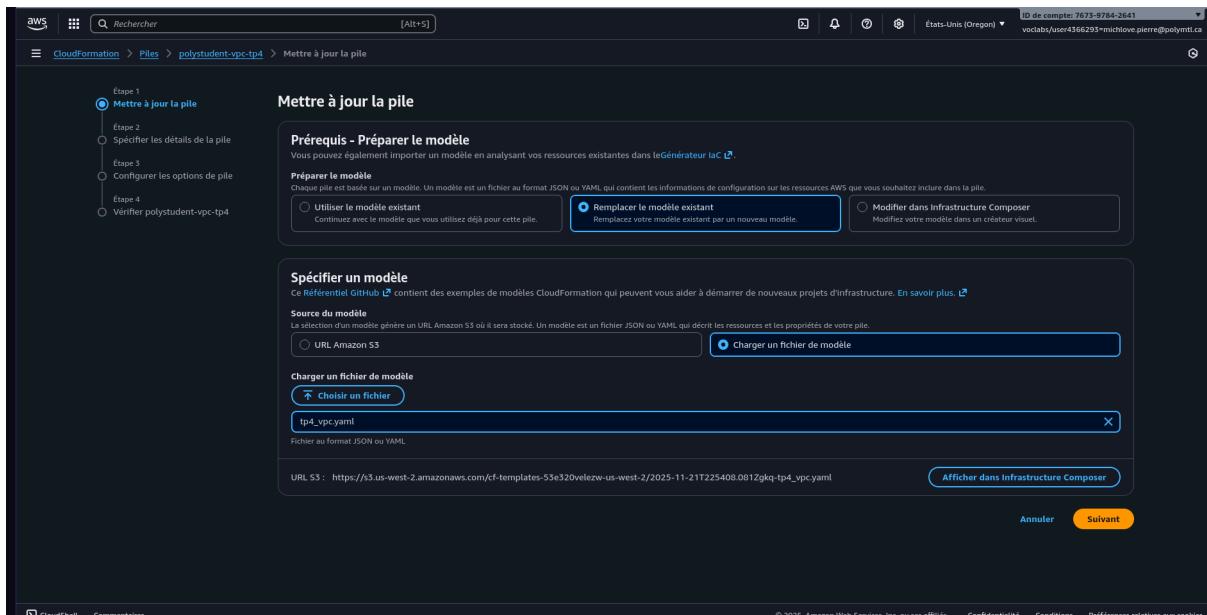


Figure 42: Mise à jour de la pile et chargement du fichier yaml obtenu à l'aide du script

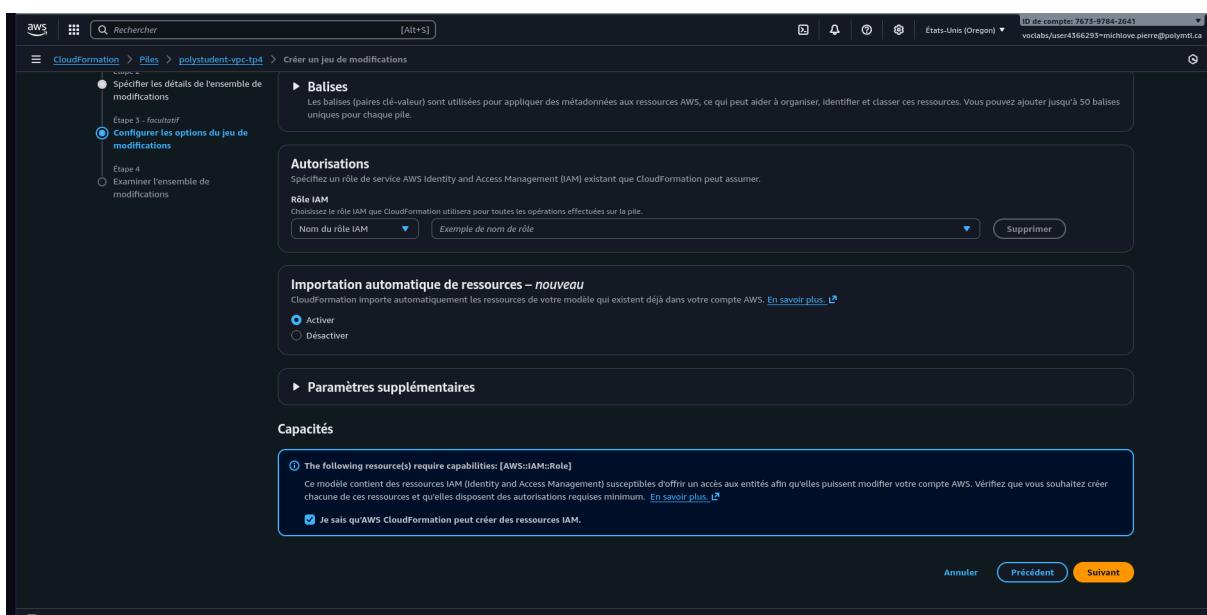


Figure 43: Démontrer que nous avons coché la section capacités

Action	ID logique	ID physique	Type de ressource	Remplacement	Module	Invocation
Add	IngressPacketsAlarm1	-	AWS::CloudWatch::Alarm	-	-	-
Add	IngressPacketsAlarm2	-	AWS::CloudWatch::Alarm	-	-	-
Add	IngressPacketsAlarm3	-	AWS::CloudWatch::Alarm	-	-	-
Add	IngressPacketsAlarm4	-	AWS::CloudWatch::Alarm	-	-	-
Add	PrivateInstance1	-	AWS::EC2::Instance	-	-	-
Add	PrivateInstance2	-	AWS::EC2::Instance	-	-	-
Add	PublicInstance1	-	AWS::EC2::Instance	-	-	-
Add	PublicInstance2	-	AWS::EC2::Instance	-	-	-

Figure 44: Démontrer que les instances EC2 et les alarmes font partie de la section mis à jour

ID d'opération	Horodatage	ID logique	Statut	Statut détaillé	Motif du statut
e0a852c8- c133-4ec5-9e7e- e53228515e05	21-11-2025 18:32:04 UTC-0500	polystudent-vpc-tp4	✓ UPDATE_COMPLETE	-	-
e0a852c8- c133-4ec5-9e7e- e53228515e05	21-11-2025 18:32:03 UTC-0500	polystudent-vpc-tp4	○ UPDATE_COMPLETE_, CLEANUP_IN_PROGRESS	-	-
e0a852c8- c133-4ec5-9e7e- e53228515e05	21-11-2025 18:32:02 UTC-0500	IngressPacketsAlarm3	✓ CREATE_COMPLETE	-	-
e0a852c8- c133-4ec5-9e7e- e53228515e05	21-11-2025 18:32:01 UTC-0500	IngressPacketsAlarm1	✓ CREATE_COMPLETE	-	-

Figure 45: Démontrer que la pile a bien été mise à jour.

3.3

```

import boto3
import json
from botocore.exceptions import ClientError

REGION = "us-west-2"
BUCKET_NAME = "polystudents3-anis-michlove-unique"
BUCKET_BACK = "polystudents3-back-anis-michlove-unique"
KMS_KEY_ARN = "arn:aws:kms:us-west-2:76797842641:key/44fd78a9-0lad-4323-8cc0-24e608de0197"

s3_client = boto3.client("s3", region_name=REGION)
s3_resource = boto3.resource("s3", region_name=REGION)
cloudtrail = boto3.client("cloudtrail", region_name=REGION)

flow_log_policy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": ["s3:PutObject", "s3:PutObjectAcl"],
            "Resource": f"arn:aws:s3:::{BUCKET_NAME}/",
            "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-control"}}
        },
        {
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": "s3:GetBucketAcl",
            "Resource": f"arn:aws:s3:::{BUCKET_NAME}"
        }
    ]
}

def create_bucket(bucket):
    try:
        s3_client.create_bucket(
            Bucket=bucket,
            CreateBucketConfiguration={"LocationConstraint": REGION}
        )
        print(f"Bucket {bucket} créé.")
    except ClientError as e:
        if e.response["Error"]["Code"] in ["BucketAlreadyOwnedByYou", "BucketAlreadyExists"]:
            print(f"Bucket {bucket} existe déjà.")
        else:
            raise e

def create_bucket_with_back(bucket):
    try:
        s3_client.create_bucket(
            Bucket=bucket,
            CreateBucketConfiguration={"LocationConstraint": REGION}
        )
        print(f"Bucket {bucket} créé.")
    except ClientError as e:
        if e.response["Error"]["Code"] in ["BucketAlreadyOwnedByYou", "BucketAlreadyExists"]:
            print(f"Bucket {bucket} existe déjà.")
        else:
            raise e

```

Figure 46: Mise à jour du script generate_s3_bucket_with_bucket_back.py pour créer le nouveau bucket

```

def create_bucket(bucket):
    try:
        s3_client.create_bucket(
            Bucket=bucket,
            CreateBucketConfiguration={"LocationConstraint": REGION}
        )
        print(f"Bucket {bucket} créé.")
    except ClientError as e:
        if e.response["Error"]["Code"] in ["BucketAlreadyOwnedByYou", "BucketAlreadyExists"]:
            print(f"Bucket {bucket} existe déjà.")
        else:
            raise e

def secure_bucket(bucket):
    s3_client.put_bucket_versioning(
        Bucket=bucket,
        VersioningConfiguration={"Status": "Enabled"}
    )
    print(f"Versioning activé pour {bucket}.")

    if bucket == BUCKET_NAME:
        s3_client.put_bucket_encryption(
            Bucket=bucket,
            ServerSideEncryptionConfiguration={
                "Rules": [
                    {
                        "ApplyServerSideEncryptionByDefault": {
                            "AWSKMSKeyId": "arn:aws:kms:us-west-2:76797842641:key/44fd78a9-0lad-4323-8cc0-24e608de0197",
                            "NoEncryptionKey": False
                        }
                    }
                ]
            }
        )
        print(f"Chiffrement SSE-KMS activé pour {bucket}.")  

    s3_client.put_public_access_block(
        Bucket=bucket,
        PublicAccessBlockConfiguration={
            "BlockPublicRead": True,
            "BlockPublicWrite": True,
            "IgnorePublicPolicy": False,
            "RestrictPublicBuckets": True
        }
    )
    print(f"Accès public bloqué pour {bucket}.")

```

Figure 47: Mise à jour du script generate_s3_bucket_with_bucket_back.py pour ajuster les accès accès au bucket

```

def replicate_objects():
    print("Replication commence")
    src_bucket = s3.resource.Bucket(BUCKET_NAME)
    dest_bucket = s3.resource.Bucket(BUCKET_BACK)

    for obj in src_bucket.objects.all():
        copy_source = {"Bucket": BUCKET_NAME, "Key": obj.key}
        dest_bucket.copy(copy_source, obj.key)
        print("Copié : " + obj.key)

    print("Replication terminée.\n")

def apply_cloudtrail_bucket_policy():
    policy = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AWSCloudTrailAccessCheck",
                "Effect": "Allow",
                "Principal": {"Service": "cloudtrail.amazonaws.com"},
                "Action": "S3:GetBucketAcl",
                "Resource": f"arn:aws:s3:::{BUCKET_BACK}"
            },
            {
                "Sid": "AWSCloudTrailWrite",
                "Effect": "Allow",
                "Principal": {"Service": "cloudtrail.amazonaws.com"},
                "Action": "S3:PutObject",
                "Resource": f"arn:aws:s3:::{BUCKET_BACK}/AWSLogs/767397842641/*",
                "Condition": {
                    "StringEquals": {
                        "s3:x-amz-acl": "bucket-owner-full-control"
                    }
                }
            }
        ]
    }

    s3_client.put_bucket_policy(
        Bucket=BUCKET_BACK,
        Policy=json.dumps(policy)
    )
    print(f"Politique CloudTrail appliquée au bucket {BUCKET_BACK}")

```

Figure 48 : Mise à jour du script generate_s3_bucket_with_bucket_back.py pour répliquer le bucket S3 vers le bucket S3 polystudents3-back et pour activer CloudTrail

```

def setup_cloudtrail():
    print("Activation CloudTrail pour audit des objets S3...")
    trail_name = "S3ObjectTrail"

    try:
        cloudtrail.create_trail(
            Name=trail_name,
            S3BucketName=BUCKET_BACK,
            IsMultiRegionTrail=False
        )
        print("Trail créé.")
    except ClientError as e:
        if "TrailAlreadyExistsException" in str(e):
            print("Le Trail existe déjà.")
        else:
            raise e

    cloudtrail.put_event_selectors(
        TrailName=trail_name,
        EventSelectors=[
            {
                "ReadWriteType": "WriteOnly",
                "IncludeManagementEvents": False,
                "DataResources": [
                    {
                        "Type": "AWS::S3::Object",
                        "Values": [f"arn:aws:s3:::{BUCKET_NAME}/*"]
                    }
                ]
            }
        ]
    )

    cloudtrail.start_logging(Name=trail_name)
    print("CloudTrail activé pour surveiller modifications/suppressions.\n")

```

Figure 49: Setup du clouptrail afin de l'activer

```

Nov 23 13:49
michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA: ~/Documents/Ecole/INF8102/TP
$ python3 generate_s3_bucket_with_bucket_back.py

# CONFIGURATION DES BUCKETS S3 ==
Bucket polystudents3-anis-michlove-unique existe déjà.
Versioning activé pour polystudents3-anis-michlove-unique.
Chiffrement SSE-KMS activé pour polystudents3-anis-michlove-unique.
Accès public bloqué pour polystudents3-anis-michlove-unique.
Politique Flow Logs appliquée à polystudents3-anis-michlove-unique.
Bucket polystudents3-back-anis-michlove-unique existe déjà.
Versioning active pour polystudents3-back-anis-michlove-unique.
Accès public bloqué pour polystudents3-back-anis-michlove-unique.

Réplication manuelle polystudents3-anis-michlove-unique > polystudents3-back-anis-michlove-unique...
Réplication terminée.

Politique CloudTrail appliquée au bucket polystudents3-back-anis-michlove-unique
Activation CloudTrail pour audit des objets S3...
✓ Trail créé.
✓ CloudWatch activé pour surveiller modifications/suppressions.

Tous les éléments S3 ont été configurés avec succès !
(venv) michlove@VivoBook-ASUSLaptop-TP420UA-TM420UA: ~/Documents/Ecole/INF8102/TP>

```

Figure 50: Exécution du script generate_s3_bucket_with_bucket_back.py

Nom	Région AWS	Date de création
aws-cloudtrail-logs-767597842641-fse63des	USA Est (Virginie du Nord) us-east-1	15 Nov 2025 02:23:30 PM EST
ct-templates-5Se320elezw-us-east-1	USA Est (Virginie du Nord) us-east-1	21 Nov 2025 02:37:23 PM EST
ct-templates-5Se320elezw-us-west-2	USA Ouest (Oregon) us-west-2	21 Nov 2025 02:48:18 PM EST
polystudents3-anis-michlove-unique	USA Ouest (Oregon) us-west-2	21 Nov 2025 04:26:46 PM EST
polystudents3-back-anis-michlove-unique	USA Ouest (Oregon) us-west-2	23 Nov 2025 01:36:46 PM EST
polystudents3-tp3	USA Est (Virginie du Nord) us-east-1	09 Nov 2025 06:19:22 PM EST

Figure 51: Démontrer que le bucket S3 back à bien été créer

Figure 52: Démontrer que le clouptrail est bien activé dans le bucket S3 back

Figure 52: Preuve que le clouptrail est bien activé dans le bucket S3 back

4.

```
(venv) michlove@VivoBook-ASUSLaptop-TPA20UA-TM420UA:~/Documents/Ecole/INFS102/TD$ curl -sfl https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh | sudo sh -s -- -b /usr/local/bin
aquasecurity/trivy info checking GitHub for latest tag
aquasecurity/trivy info found version: 0.67.2 for v0.67.2/Linux/x64bit
aquasecurity/trivy info installed /usr/local/bin/trivy
```

Figure 53: Installation de trivy

```
(venv) michlove@VivoBook-ASUSLaptop-TPA20UA-TM420UA:~/Documents/Ecole/INFS102/TD$ trivy --version
Version: 0.67.2
```

Figure 54: Vérification de la version de trivy

4.1

```

nichlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/Ecole/INF8102/TP4/TP4_1
$ trivy fs --scanners vuln,misconfig,secret --severity MEDIUM,HIGH,CRITICAL --format json --output scan_report.json
2025-12-11T00:36:34-05:00      INFO  [vuln] Vulnerability scanning is enabled
2025-12-11T00:36:34-05:00      INFO  [misconfig] Misconfiguration scanning is enabled
2025-12-11T00:36:35-05:00      INFO  [secret] Secret scanning is enabled
2025-12-11T00:36:35-05:00      INFO  [secret] If your scanning is slow, please try '--scanners vuln,misconfig' to disable secret scanning
2025-12-11T00:36:35-05:00      INFO  [secret] Please see https://trivy.dev/v0.67/docs/scanner/secret#recommendation for faster secret detection
2025-12-11T00:36:35-05:00      WARN   [cloudformation parser] Missing parameter values      file_path="tp4_vpc.yaml" parameters="EnvironmentName"
2025-12-11T00:36:35-05:00      INFO  Number of language-specific files      num=0
2025-12-11T00:36:35-05:00      INFO  Detected config files      num=1

```

Figure 55: Exécution de trivy sur le dossier du tp afin de générer le rapport

```

{
  "SchemaVersion": 2,
  "CreatedAt": "2025-12-11T00:36:35.437349426-05:00",
  "ArtifactName": "",
  "ArtifactType": "filesystem",
  "Metadata": {
    "RepoURL": "git@github.com:mipie/TP4_INF8102.git",
    "Branch": "main",
    "Commit": "c5d9d5d0bf6f155283f8e26ecb4ff144ad9db6",
    "CommitMsg": "Création du nouveau script pour créer le nouveau bucket et activer le CloudTrail",
    "Author": "Mchi <nichlove.pierre@polymtl.ca>",
    "Committer": "Mchi <nichlove.pierre@polymtl.ca>"
  },
  "Results": [
    {
      "Target": "tp4_vpc.yaml",
      "Class": "config",
      "Type": "CloudFormation"
    }
  ],
  "MisconSummary": {
    "Successes": 34,
    "Failures": 12
  },
  "Misconfigurations": [
    {
      "Type": "CloudFormation Security Check",
      "ID": "AVD-AWS-0028",
      "AVDID": "AVD-AWS-0028",
      "Title": "AWS CloudFormation should activate session tokens for Instance Metadata Service",
      "Description": "IMDS v2 (Instance Metadata Service) introduced session authentication tokens which improve security when talking to IMDS.\\nBy default, AWS CloudFormation sets IMDS session auth tokens to be optional.\\n\\nTo fully protect IMDS you need to enable session tokens by using <code><aws:instance>/<code> resource sets <code>IMDS session auth tokens</code> variable set to <code>required</code>.\\n\\n",
      "Message": "IMDS does not require IMDS access to require a token.",
      "Namespace": "builtin.aws.ec2.aws0028",
      "Query": "data.builtin.aws.ec2.aws0028.deny",
      "Resolution": "Enable HTTP token requirement for IMDS",
      "Severity": "HIGH",
      "PrimaryURL": "https://avd.aquasec.com/misconfig/avd-aws-0028",
      "References": [
        {
          "ID": "https://aws.amazon.com/blogs/security/defending-in-depth-open-firewalls-reverse-previews-ssrf-vulnerabilities-ec2-instance-metadata-service"
        },
        {
          "ID": "https://avd.aquasec.com/misconfig/avd-aws-0028"
        }
      ],
      "Status": "FAIL"
    }
  ],
  "CauseOfMetadata": {
    "Resource": "tp4_vpc.yaml:225-234",
    "Provider": "AWS",
    "Service": "ec2",
    "StartTime": 225,
    "EndLine": 234
  },
  "Code": [
    {
      "Lines": [
        {
          "Number": 225,
          "Content": "  PrivateInstanceProfile: ", "IsCause": true, ...
        },
        {
          "Number": 226,
          "Content": "    Properties": "", "IsCause": true, ...
        },
        {
          "Number": 227,
          "Content": "    IAMInstanceProfile: IAMInstanceProfile", "IsCause": true, ...
        }
      ]
    }
  ]
}

```

Figure 56: Preuve que le rapport à bien été générée

4.2

```

nichlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/Ecole/INF8102/TP4/TP4_1$ jq '.Results[].Vulnerabilities // [] | .[] | objects | select(.Severity == "HIGH") | {Description: (.Description // ".Message // \"N/A\""), Severity, CVSS3: (.CVSS3.nvgt,V3Score // "N/A")}' scan_report.json > v3vuln.json
nichlove@VivoBook-ASUSLaptop-TP420UA-TM420UA:~/Documents/Ecole/INF8102/TP4/TP4_1$ 

```

Figure 57: Extraction des cve du rapport json

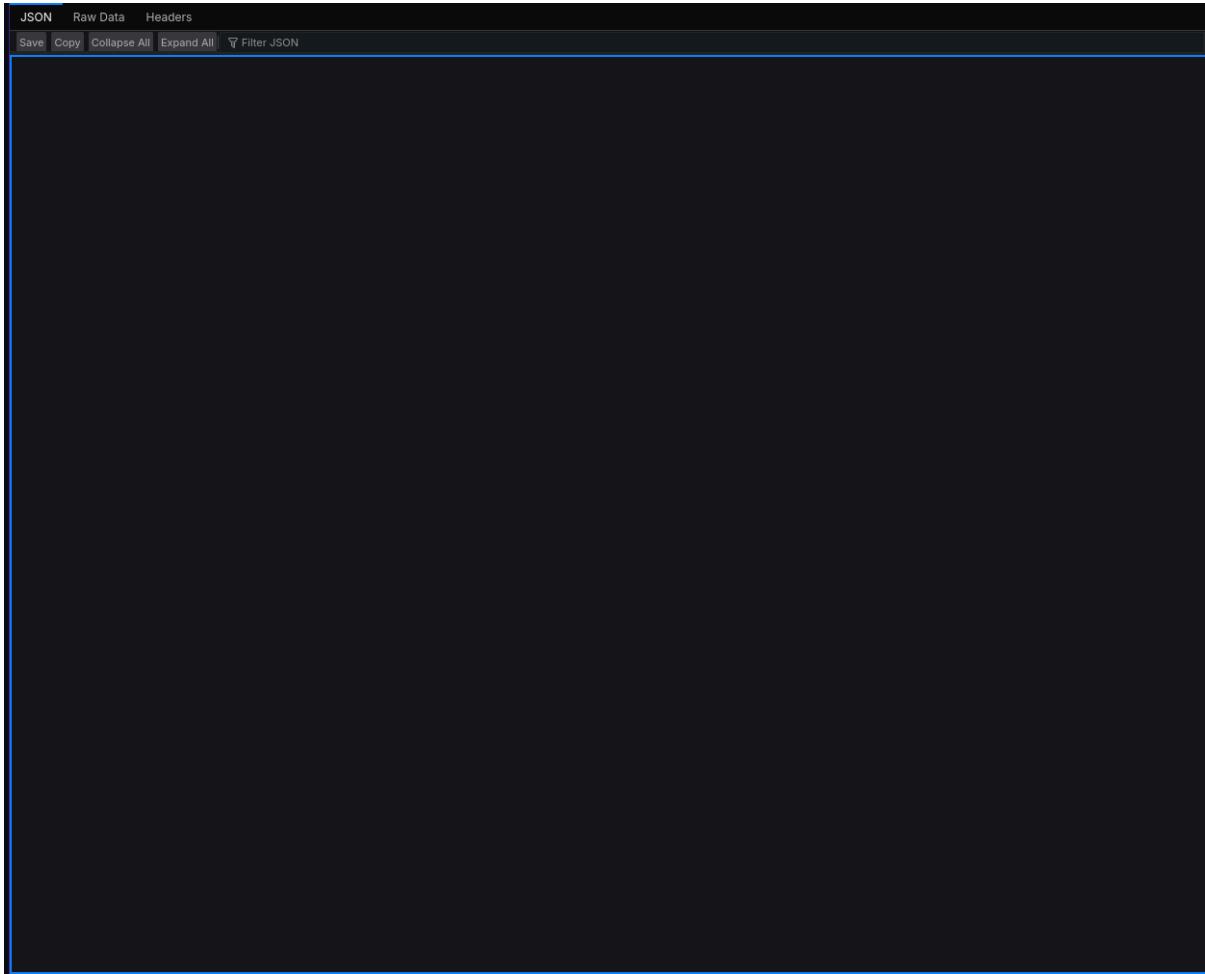


Figure 58: Preuve que le fichier cve.json a bien été créer

4.3

Selon les résultats obtenus, quatre vulnérabilités ont été détectées dans notre code IaC : AVD-AWS-0028, AVD-AWS-0107, AVD-AWS-0131 et AVD-AWS-0164. Les paragraphes suivants présentent, pour chacune de ces vulnérabilités, une mesure de sécurité appropriée permettant d'en assurer la mitigation.

En premier lieu, au niveau de la vulnérabilité AVD-AWS-0028, la solution serait d'activer les tokens de session en utilisant le paramètre *metadata_options* et en modifiant sa variable *http_tokens* à l'état *required*.

En deuxième lieu, au niveau de la vulnérabilité AVD-AWS-0107, la solution serait d'utiliser une unique adresse IP autorisée *x.x.x.x/x* sur la variable *CidrIp* et d'éviter l'utilisation de *0.0.0.0/0*. Cette limitation assure qu'uniquement un utilisateur de confiance puisse atteindre les ports critiques tels que SSH (22) et RDP (3389).

En troisième lieu, au niveau de la vulnérabilité AVD-AWS-0131, la solution serait d'activer le chiffrement des volumes EBS pour chaque instance EC2. Afin d'y arriver, il faut ajouter le

paramètre *BlockDeviceMappings* en modifiant sa variable *Encrypted* à l'état *true*. Ainsi, au repos, les données des instances seraient encryptées et sécurisées.

En quatrième lieu, au niveau de la vulnérabilité AVD-AWS-0164, la solution serait de modifier le paramètre *MapPublicIpOnLaunch* à l'état *false*. Ainsi, cela permettrait d'éviter que le VPC attribue automatiquement aux instances des adresses IP publiques par défaut.

Voici le lien vers notre répo: https://github.com/mipie/TP4_INF8102.git

Références

<https://avd.aquasec.com/misconfig/aws/ec2/avd-aws-0028/>

<https://avd.aquasec.com/misconfig/aws/ec2/avd-aws-0107/>

<https://avd.aquasec.com/misconfig/aws/ec2/avd-aws-0131/>

<https://avd.aquasec.com/misconfig/aws/ec2/avd-aws-0164/>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIEncryption.html>

<https://support.iompaas.com/support/solutions/articles/62000233593-ensure-vpc-subnets-do-not-assign-public-ip-by-default/>