TERRAFORM

Version 2.1

07 juin 2024

Contents:

Note : This project is under active development.

CHAPITRE 1

Installer Terraform

Nous allons installer terraform sur un poste de travail (Desktop Windows/Mac/Linux)

1.1 Linux

1.1.1 Centos 7

Installer avec yum

```
$ sudo yum install -y yum-utils
$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.
↪repo
$ sudo yum -y install terraform
```

1.2 Windows

1.2.1 Télécharger le binaire

Télécharger le fichier

```
$ wget -LO https://releases.hashicorp.com/terraform/1.5.3/terraform_1.5.3_windows_386.zip
```

Puis ajouter le binaire dans le PATH

1.3 MAC OS

1.3.1 Avec wget

Télécharger le fichier

```
$ wget https://releases.hashicorp.com/terraform/1.5.3/terraform_1.5.3_darwin_amd64.zip
```

1.3.2 Avec homebrew

```
$ brew tap hashicorp/tap
$ brew install hashicorp/tap/terraform
```

1.4 Vérifier l'installation

Afficher la version

```
$ terraform version
```

CHAPITRE 2

Initialisation du Provider

2.1 Provider AWS

2.1.1 créer le fichier credentials

```
$ mkdir .aws
$ vi .aws/credentials
```

```
[default]
aws_access_key_id = AKIA2UC2746GI6ZWN6EG
aws_secret_access_key = 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
```

2.1.2 Créer un fichier principal

créer le fichier main.tf

```
$ vi provider.tf
```

```
provider "aws" {
    region = "eu-west-1"
}
```

2.1.3 Initialiser le projet

Exécuter la commande **terraform init**

```
$ terraform init
```

```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.8.0...
- Installed hashicorp/aws v5.8.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2.2 Provider Oracle

2.2.1 Modifier le fichier principal

modifier le fichier main.tf

```
$ vi main.tf
```

```
provider "aws" {
    region = "eu-central-1"
}
provider "oci" {
  # Configuration options
}
```

2.2.2 Initialiser le projet

Exécuter la commande **terraform init**

```
$ terraform init
```

```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Finding latest version of hashicorp/oci...
- Installing hashicorp/oci v5.4.0...
- Installed hashicorp/oci v5.4.0 (signed by HashiCorp)
- Using previously-installed hashicorp/aws v5.8.0

Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.


 Warning: Additional provider information from registry

 The remote registry returned warnings for registry.terraform.io/hashicorp/oci:
 - For users on Terraform 0.13 or greater, this provider has moved to oracle/oci. Please␣
↪update your source in
 required_providers.


Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2.3 Pré-requis des Providers

2.3.1 Créer un fichier providers

créer le fichier providers.tf

```
$ vi providers.tf
```

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
```

(suite sur la page suivante)

(Suite de la page précédente)

```
      version = "5.8.0"
    }
    oci = {
      source = "oracle/oci"
      version = "5.4.0"
    }
  }
}
```

2.3.2 Ré-Initialiser le projet

Exécuter la commande **terraform init**

```
$ terraform init
```

```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Finding oracle/oci versions matching "5.4.0"...
- Using previously-installed hashicorp/aws v5.8.0
- Installing oracle/oci v5.4.0...
- Installed oracle/oci v5.4.0 (signed by a HashiCorp partner, key ID 1533A49284137CEB)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
re run this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2.3.3 Finalier le projet

Aller sur le Terraform registry, puis sur le : provider AWS.

Aller sur l'onglet **Use Provider**

```
$ vi providers.tf
```

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.9.0"
    }
  }
}

provider "aws" {
  region = "eu-west-1"
}
```

CHAPITRE 3

Créer première ressource

3.1 Ressource VPC (Virtual Private Cloud)

3.1.1 Créer un fichier principal

créer le fichier main.tf et remplacer le X par le numéro de l'utilisateur

```
$ vi main.tf
```

```
resource "aws_vpc" "vpc-dev" {
  cidr_block = "10.X.0.0/16"
  tags = {
    Name = "vpc-X"
  }
}
resource "aws_subnet" "subnet-dev-1" {
  vpc_id     = aws_vpc.vpc-dev.id
  cidr_block = "10.X.1.0/24"
}
resource "aws_subnet" "subnet-dev-2" {
  vpc_id     = "${aws_vpc.vpc-dev.id}"
  cidr_block = "10.X.2.0/24"
}
```

3.1.2 Créer un plan

Exécuter la commande **terraform plan**

```
$ terraform plan
```

```
Terraform used the selected providers to generate the following execution plan. Resource␣
↪actions are indicated
with the following symbols:
+ create

Terraform will perform the following actions:

+ resource "aws_vpc" "vpc-dev" {
    + arn                                  = (known after apply)
    + cidr_block                           = "10.0.0.0/16"
    + default_network_acl_id               = (known after apply)
    + default_route_table_id               = (known after apply)
    + default_security_group_id            = (known after apply)
    + dhcp_options_id                      = (known after apply)
    + enable_dns_hostnames                 = (known after apply)
    + enable_dns_support                   = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                                   = (known after apply)
    + instance_tenancy                     = "default"
    + ipv6_association_id                  = (known after apply)
    + ipv6_cidr_block                      = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id                  = (known after apply)
    + owner_id                             = (known after apply)
    + tags_all                             = (known after apply)
    }

Plan: 3 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to␣
↪take exactly these
actions if you run "terraform apply" now.
```

3.1.3 Appliquer le plan

Exécuter la commande **terraform apply**

```
$ terraform apply
```

```
Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes
```

(suite sur la page suivante)

(Suite de la page précédente)

```
aws_vpc.vpc-dev: Creating...
aws_vpc.vpc-dev: Creation complete after 1s [id=vpc-0e48aff53f634145a]
aws_subnet.subnet-dev-2: Creating...
aws_subnet.subnet-dev-1: Creating...
aws_subnet.subnet-dev-2: Creation complete after 0s [id=subnet-08a1804a0eabb87c3]
aws_subnet.subnet-dev-1: Creation complete after 0s [id=subnet-0805263ae47c9ad3f]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

3.1.4 Supprimer les ressources

Exécuter la commande **terraform destroy**

```
$ terraform destroy
```

```
# aws_vpc.vpc-dev will be destroyed
- resource "aws_vpc" "vpc-dev" {
    - arn                                  = "arn:aws:ec2:eu-central-1:754448004632:vpc/
↪vpc-0e48aff53f634145a" -> null
    - assign_generated_ipv6_cidr_block     = false -> null
    - cidr_block                           = "10.0.0.0/16" -> null
    - default_network_acl_id               = "acl-061b5d2df73ec46e5" -> null
    - default_route_table_id               = "rtb-0b259066975cb8c9f" -> null
    - default_security_group_id            = "sg-030317041393a456d" -> null
    - dhcp_options_id                      = "dopt-0777af6e9ee33566e" -> null
    - enable_dns_hostnames                 = false -> null
    - enable_dns_support                   = true -> null
    - enable_network_address_usage_metrics = false -> null
    - id                                   = "vpc-0e48aff53f634145a" -> null
    - instance_tenancy                     = "default" -> null
    - ipv6_netmask_length                  = 0 -> null
    - main_route_table_id                  = "rtb-0b259066975cb8c9f" -> null
    - owner_id                             = "754448004632" -> null
    - tags                                 = {} -> null
    - tags_all                             = {} -> null
    }

Plan: 0 to add, 0 to change, 3 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_subnet.subnet-dev-2: Destroying... [id=subnet-08a1804a0eabb87c3]
aws_subnet.subnet-dev-1: Destroying... [id=subnet-0805263ae47c9ad3f]
aws_subnet.subnet-dev-2: Destruction complete after 0s
aws_subnet.subnet-dev-1: Destruction complete after 0s
```

(suite sur la page suivante)

(Suite de la page précédente)

```
aws_vpc.vpc-dev: Destroying... [id=vpc-0e48aff53f634145a]
aws_vpc.vpc-dev: Destruction complete after 1s

Destroy complete! Resources: 3 destroyed.
```

3.2 Data VPC

3.2.1 Ajouter un subnet au default vpc

```
$ vi main.tf
```

```
data "aws_vpc" "default-pvc" {
  default = true
}

resource "aws_subnet" "subnet-dev-3" {
  vpc_id     = data.aws_vpc.default-pvc.id
  cidr_block = "172.31.48.0/20"

}
```

Exécuter la commande **terraform apply**

```
$ terraform apply
```

```
data.aws_vpc.default-pvc: Reading...
data.aws_vpc.default-pvc: Read complete after 0s [id=vpc-0f832cd1239745391]

Terraform used the selected providers to generate the following execution plan. Resource␣
↪actions are
indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_subnet.subnet-dev-3 will be created
+ resource "aws_subnet" "subnet-dev-3" {
    + arn                                            = (known after apply)
    + assign_ipv6_address_on_creation                = false
    + availability_zone                              = (known after apply)
    + availability_zone_id                           = (known after apply)
    + cidr_block                                     = "172.31.48.0/20"
    + enable_dns64                                   = false
    + enable_resource_name_dns_a_record_on_launch    = false
    + enable_resource_name_dns_aaaa_record_on_launch = false
    + id                                             = (known after apply)
    + ipv6_cidr_block_association_id                 = (known after apply)
    + ipv6_native                                    = false
    + map_public_ip_on_launch                        = false
```

(suite sur la page suivante)

(Suite de la page précédente)

```
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags_all                                       = (known after apply)
      + vpc_id                                         = "vpc-0f832cd1239745391"
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_subnet.subnet-dev-3: Creating...
aws_subnet.subnet-dev-3: Creation complete after 0s [id=subnet-01a667d890cdfc356]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Console

$ terraform apply

CHAPITRE 4

Ajout/Supp/Modification

4.1 Modifier une ressource

Modifier le Tag Name en user-X (X numéro user)

```
$ vi main.tf
```

```
provider "aws" {
    region = "eu-central-1"
}
resource "aws_vpc" "vpc-dev" {
  cidr_block = "10.X.0.0/16"
  tags = {
    Name = "user-X"
  }
}
resource "aws_subnet" "subnet-dev-1" {
  vpc_id     = aws_vpc.vpc-dev.id
  cidr_block = "10.X.1.0/24"
}
resource "aws_subnet" "subnet-dev-2" {
  vpc_id     = "${aws_vpc.vpc-dev.id}"
  cidr_block = "10.X.2.0/24"
}
```

4.1.1 Faire un apply

Et éxaminer le résultat (~ pour modification)

```
$ terraform apply
```

```
aws_vpc.vpc-dev: Refreshing state... [id=vpc-0551c53e342aabad5]
aws_subnet.subnet-dev-1: Refreshing state... [id=subnet-02b82c4c9e75c5636]
aws_subnet.subnet-dev-2: Refreshing state... [id=subnet-00861b70086f8c65d]

Terraform used the selected providers to generate the following execution plan. Resource␣
↪actions are
indicated with the following symbols:
~ update in-place

Terraform will perform the following actions:

# aws_vpc.vpc-dev will be updated in-place
~ resource "aws_vpc" "vpc-dev" {
        id                                   = "vpc-0551c53e342aabad5"
    ~ tags                                 = {
        ~ "Name" = "vpc-1" -> "user-1"
        }
    ~ tags_all                             = {
        ~ "Name" = "vpc-1" -> "user-1"
        }
        # (14 unchanged attributes hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.
```

4.2 Supprimer une ressource

Supprimer le subnet-2

```
$ vi main.tf
```

```
provider "aws" {
    region = "eu-central-1"
}
resource "aws_vpc" "vpc-dev" {
cidr_block = "10.X.0.0/16"
tags = {
    Name = "user-X"
}
}
resource "aws_subnet" "subnet-dev-1" {
vpc_id     = aws_vpc.vpc-dev.id
cidr_block = "10.X.1.0/24"
}
```

4.2.1 Faire un apply

Et éxaminer le résultat (- pour suppression)

```
$ terraform apply
```

```
aws_vpc.vpc-dev: Refreshing state... [id=vpc-0551c53e342aabad5]
aws_subnet.subnet-dev-2: Refreshing state... [id=subnet-00861b70086f8c65d]
aws_subnet.subnet-dev-1: Refreshing state... [id=subnet-02b82c4c9e75c5636]

Terraform used the selected providers to generate the following execution plan. Resource␣
↪actions are
indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_subnet.subnet-dev-2 will be destroyed
# (because aws_subnet.subnet-dev-2 is not in configuration)
- resource "aws_subnet" "subnet-dev-2" {
    - arn                                            = "arn:aws:ec2:eu-central-
↪1:754448004632:subnet/subnet-00861b70086f8c65d" -> null
    - assign_ipv6_address_on_creation                = false -> null
    - availability_zone                              = "eu-central-1c" -> null
    - availability_zone_id                           = "euc1-az1" -> null
    - cidr_block                                     = "10.1.2.0/24" -> null
    - enable_dns64                                   = false -> null
    - enable_lni_at_device_index                     = 0 -> null
    - enable_resource_name_dns_a_record_on_launch    = false -> null
    - enable_resource_name_dns_aaaa_record_on_launch = false -> null
    - id                                             = "subnet-00861b70086f8c65d" -> null
    - ipv6_native                                    = false -> null
    - map_customer_owned_ip_on_launch                = false -> null
    - map_public_ip_on_launch                        = false -> null
    - owner_id                                       = "754448004632" -> null
    - private_dns_hostname_type_on_launch            = "ip-name" -> null
    - tags                                           = {} -> null
    - tags_all                                       = {} -> null
    - vpc_id                                         = "vpc-0551c53e342aabad5" -> null
    }

Plan: 0 to add, 0 to change, 1 to destroy.
```

4.3 Ajouter une ressource

Ajouter un subnet-dev-3

```
$ vi main.tf
```

```
provider "aws" {
    region = "eu-central-1"
}
resource "aws_vpc" "vpc-dev" {
  cidr_block = "10.X.0.0/16"
  tags = {
    Name = "user-X"
  }
}
resource "aws_subnet" "subnet-dev-1" {
  vpc_id     = aws_vpc.vpc-dev.id
  cidr_block = "10.X.1.0/24"
}
resource "aws_subnet" "subnet-dev-3" {
  vpc_id     = "${aws_vpc.vpc-dev.id}"
  cidr_block = "10.X.3.0/24"
}
```

4.3.1 Faire un apply

Et éxaminer le résultat (+ pour création)

```
$ terraform apply
```

```
aws_vpc.vpc-dev: Refreshing state... [id=vpc-0551c53e342aabad5]
aws_subnet.subnet-dev-1: Refreshing state... [id=subnet-02b82c4c9e75c5636]

Terraform used the selected providers to generate the following execution plan. Resource␣
↪actions are
indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_subnet.subnet-dev-3 will be created
+ resource "aws_subnet" "subnet-dev-3" {
    + arn                                            = (known after apply)
    + assign_ipv6_address_on_creation                = false
    + availability_zone                              = (known after apply)
    + availability_zone_id                           = (known after apply)
    + cidr_block                                     = "10.1.3.0/24"
    + enable_dns64                                   = false
    + enable_resource_name_dns_a_record_on_launch    = false
    + enable_resource_name_dns_aaaa_record_on_launch = false
    + id                                             = (known after apply)
```

(suite sur la page suivante)

(Suite de la page précédente)

```
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags_all                                       = (known after apply)
      + vpc_id                                         = "vpc-0551c53e342aabad5"
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

CHAPITRE 5

Utilisation des variables

5.1 Créer une ressource ec2_instance

créer un nouveau dossier lab1_variables

créer le fichier main.tf avec la définition de la ressource.

ajouter un tag name égal à <userX>-<ressource-name>

créer un fichier variables.tf qui contient la définition des variables

créer un fichier providers.tf qui contient les pré-requis du provider aws

créer un fichier terraform.tfvars qui contient les valeurs des variables

CHAPITRE 6

Solution Utilisation des variables

6.1 Créer une ressource ec2_instance

```
more main.tf
```

```
resource "aws_instance" "my_ec2" {
    tags = {
        Name = "${var.tag_name}"
    }
    ami  = var.amis[var.region]
    instance_type = var.instance_type
}
```

```
more providers.tf
```

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.8.0"
    }
  }
}

provider "aws" {
  region = var.region
}
```

```
more variables.tf
```

```
variable "region" {}
variable "instance_type" {}
variable "amis" {}
variable "tag_name" {}
```

```
more terraform.tfvars
```

```
region                = "eu-west-1"
instance_type         = "t2.micro"
tag_name              = "user-11"
amis                   = {
  "eu-west-1" = "ami-0d31449d0dd5f363f"
  "eu-west-2" = "ami-0e603d96bf395bc01"
  "eu-west-3" = "ami-0eeeb6788f77d3616"
}
```

```
more output.tf
```

```
output "public_ip" {
    value = aws_instance.my_ec2.public_ip
}
```

CHAPITRE 7

Utilisation des provisioners

7.1 Méthode 1

créer un nouveau dossier lab2_user_data

créer le fichier main.tf avec la définition de la ressource.

installer le serveur web apache aver le provisioner user_data

vérifier l'accès au serveur

supprimer la ressource

7.2 Méthode 2

utiliser un script pour installer apache avec le provisioner user_data

7.3 Méthode 3

utiliser le provisioner remote_exec pour installer apache

utiliser le provisioner local_exec pour créer un fichier local contenant l'IP publique de l'instance

```
resource "aws_instance" "my_ec2_2" {
    vpc_security_group_ids = [aws_security_group.instance_sg.id]
}

resource "aws_security_group" "instance_sg" {
    name = "terraform-test-sg-userX"

    egress {
```

(suite sur la page suivante)

(Suite de la page précédente)

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

    ingress {
        from_port   = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
}
```

```
resource "aws_key_pair" "server-key" {
  key_name   = "server-key-userX"
  public_key = file(var.public_key)
}

resource "aws_instance" "my_ec2" {

  key_name = aws_key_pair.server-key.key_name


  connection {
      type        = "ssh"
      user        = "admin"
      private_key = file(var.private_key)
      host        = self.public_ip
  }
```

CHAPITRE 8

Solution provisioners

8.1 Méthode 1

```
more main.tf
```

```
resource "aws_instance" "my_ec2" {
    ami = var.my_ami[var.region]
    instance_type = "t2.micro"
    vpc_security_group_ids = [aws_security_group.instance_sg.id]
    tags = {
        Name = "terraform-debian"
    }
    user_data = <<-EOF
  #!/bin/bash
        sudo apt-get update
  sudo apt-get install -y apache2
  sudo systemctl start apache2
  sudo systemctl enable apache2
  sudo echo "<h1>Hello From Terraform</h1>" > /var/www/html/index.html
    EOF
}
resource "aws_security_group" "instance_sg" {
    name = "terraform-user10-sg"

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

(suite sur la page suivante)

(Suite de la page précédente)

```
    ingress {
        from_port   = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
}

output "public_ip" {
    value = [aws_instance.my_ec2.public_ip]
}
```

8.2 Méthode 2

```
more main.tf
```

```
resource "aws_instance" "my_ec2" {
    ami = "ami-0b0c5a84b89c4bf99"
    instance_type = "t2.micro"
    vpc_security_group_ids = [aws_security_group.instance_sg.id]
    tags = {
        Name = "terraform-debian"
    }
    user_data = "${file("install_apache.sh")}"
}
resource "aws_security_group" "instance_sg" {
    name = "terraform-user10-sg"

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

    ingress {
        from_port   = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
}

output "public_ip" {
    value = [aws_instance.my_ec2.public_ip,aws_instance.my_ec2_2.public_ip]
}
```

8.3 Méthode 3

```
more main.tf
```

```
resource "aws_key_pair" "server-key" {
key_name   = "server-key"
public_key = file(var.public_key)
}

resource "aws_instance" "my_ec2" {
  ami = "ami-0d31449d0dd5f363f"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.instance_sg.id]
  tags = {
      Name = "terraform-debian"
  }
  key_name = aws_key_pair.server-key.key_name
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get -y update",
      "sudo apt-get install -y apache2",
      "sudo systemctl start apache2",
      "sudo systemctl enable apache2",
      "sudo sh -c 'echo \"Hello From Terraform ....\" > /var/www/html/index.html'",
    ]
  connection {
      type        = "ssh"
      user        = "admin"
      private_key = file(var.private_key)
      host        = self.public_ip
  }

  }
  provisioner "local-exec" {
    when        = destroy
    on_failure  = continue
    command = "echo la fin ${self.private_ip} >> private_ips.txt"
  }
}

resource "aws_security_group" "instance_sg" {
    name = "terraform-test-sg"

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

    ingress {
        from_port   = 80
```

(suite sur la page suivante)

(Suite de la page précédente)

```
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

}

output "public_ip" {
    value = aws_instance.my_ec2.public_ip
}
```

```
more terraforms.tfvars
```

```
region                  = "eu-west-1"
public_key  = "/home/sadmin/.ssh/id_rsa.pub"
private_key         = "/home/sadmin/.ssh/id_rsa"
```

CHAPITRE 9

Utilisation des modules

9.1 Reprendre la création d'instance avec un module

créer un nouveau dossier lab3_module (module root)

créer un nouveau dossier lab3_module/modules/ec2_instance (module child)

créer deux instances avec le module ec2_instance

Ajouter à l'instance un security group avec le module proposé sur le registry

CHAPITRE 10

Solution Modules

10.1 Module local

```
more main.tf
```

```
module "ec2_instance_1" {
  source = "./modules/ec2_instance"
  instance_type = "t2.micro"
  tags = {
      Name = "vm1"
  }
}
module "ec2_instance_2" {
  source = "./modules/ec2_instance"
}
```

```
more terraform.tfvars
```

```
region = "eu-west-1"
tags = {
    Name = "dev"
}
```

```
more variables.tf
```

```
variable "region" {}
variable "tags" {}
```

```
more output.tf
```

```
output "public_ip_1" {
    value       = module.ec2_instance_1.public_ip
}
output "ec2_tags_1" {
    value       = module.ec2_instance_1.ec2_tags
}
output "public_ip_2" {
    value       = module.ec2_instance_2.public_ip
}
output "ec2_tags_2" {
    value       = module.ec2_instance_2.ec2_tags
}
```

```
more modules/ec2_instance/main.tf
```

```
resource "aws_instance" "web_server" {
  ami = var.ami
  instance_type = var.instance_type
  tags = var.tags
  security_groups = [module.security-group.security_group_name]
}

module "security-group" {
  source  = "terraform-aws-modules/security-group/aws"
  name = "sg_web"
  ingress_with_cidr_blocks = [
    {
      from_port   = 80
      to_port     = 80
      protocol    = "tcp"
      description = "HTTP"
      cidr_blocks = "0.0.0.0/0"
    }
  ]
}
```

```
more modules/ec2_instance/variables.tf
```

```
variable "ami" {
  type = string
  default = "ami-0d31449d0dd5f363f"
}
variable "instance_type" {
  type = string
  default = "t2.micro"
}
variable "region" {
  type = string
  default = "eu-west-1"
}
variable "tags" {
  description = "Tags to set for all resources"
```

(suite sur la page suivante)

(Suite de la page précédente)

```
  type        = map(string)
  default     = {
    environment = "dev"
  }
}
```

```
more modules/ec2_instance/output.tf
```

```
output "public_ip" {
  value = aws_instance.web_server[*].public_ip
}
output "ec2_tags" {
  value = aws_instance.web_server[*].tags_all
}
```

10.2 Module registry

```
more modules/ec2_instance/main.tf
```

```
resource "aws_instance" "web_server" {
  ami = var.ami
  instance_type = var.instance_type
  tags = var.tags
  security_groups = [module.security-group.security_group_name]
}

module "security-group" {
  source   = "terraform-aws-modules/security-group/aws"
  name = "sg_web"
  ingress_with_cidr_blocks = [
    {
      from_port   = 80
      to_port     = 80
      protocol    = "tcp"
      description = "HTTP"
      cidr_blocks = "0.0.0.0/0"
    }
  ]
}
```

CHAPITRE 11

Autre Solution

```
[sadmin@poste7 lab6]$ cat main.tf
resource "aws_key_pair" "server-key" {
key_name   = "server-key"
public_key = file(var.public_key)
}

module "security-group" {
  source = "terraform-aws-modules/security-group/aws"
  name   = "sg_web"
  egress_with_cidr_blocks = [
    {
      from_port   = "0"
      to_port     = "0"
      protocol    = "-1"
      cidr_blocks = "0.0.0.0/0"
    }
  ]

  ingress_with_cidr_blocks = [
    {
      from_port   = 80
      to_port     = 80
      protocol    = "tcp"
      description = "HTTP"
      cidr_blocks = "0.0.0.0/0"
    },
    {
      from_port   = "22"
      to_port     = "22"
      protocol    = "tcp"
      description = "HTTP"
```

(suite sur la page suivante)

(Suite de la page précédente)

```
      cidr_blocks = "0.0.0.0/0"
    },

  ]
}
module "ec2_instance_1" {
  source        = "./modules/ec2_instance"
  instance_type = "t2.micro"
  tags = {
    Name        = "vm1"
    Evironment = "prod"
  }
  security_groups = [module.security-group.security_group_name]
  key_name = aws_key_pair.server-key.key_name

}
#module "ec2_instance_2" {
  #source          = "./modules/ec2_instance"
  #security_groups = [module.security-group.security_group_name]
#}
```

```
[sadmin@poste7 lab6]$ cat vars.tf
variable "public_key" {}
variable "private_key" {}

variable "region" {}
variable "tags" {}
[sadmin@poste7 lab6]$ cat terraform.tfvars
public_key  = "terraform.pub"
private_key = "terraform"
region = "eu-west-1"
tags = {
  Name          = "db"
  Environement = "test"
}
```

```
[sadmin@poste7 lab6]$ cat modules/ec2_instance/main.tf
resource "aws_instance" "web_server" {
  key_name = var.key_name
  ami  = var.ami
  instance_type = var.instance_type
  tags = var.tags
  security_groups = var.security_groups
  user_data = <<-EOF
  #!/bin/bash
  sudo apt-get update
  sudo apt-get install -y apache2
  sudo systemctl start apache2
  sudo systemctl enable apache2
  sudo echo "<h1>Hello From Terraform</h1>" > /var/www/html/index.html
  EOF
}
```

```
[sadmin@poste7 lab6]$ cat modules/ec2_instance/vars.tf
variable key_name {}

variable "ami" {
  type = string
  default = "ami-0d31449d0dd5f363f"
}
variable "instance_type" {
  type = string
  default = "t2.micro"
}
variable "region" {
  type = string
  default = "eu-west-1"
}
variable "tags" {
  description = "Tags to set for all resources"
  type        = map(string)
  default     = {
    Name = "web_server"
    Environment = "dev"
  }
}
variable security_groups {}
```

```
[sadmin@poste7 lab6]$ cat modules/ec2_instance/outputs.tf
output "public_ip" {
  value = aws_instance.web_server[*].public_ip
}
```

CHAPITRE 12

Instruction for_each

Créer une variable **env** de type map avec comme clé les nom des serveurs et comme valeur les environnements des serveurs

créer des instances aws avec les serveurs de la variable **env** et ajouter les tags Name et ENV pour chaque instance

CHAPITRE 13

Solution for_each

Créer une variable **env** de type map avec comme clé les nom des serveurs et comme valeur les environnements des serveurs

```
vi variables.tf
```

```
variable "env" {
  type = map
  default = {
    "server-1" = "test"
    "server-2" = "dev"
  }
}
```

créer des instances aws avec les serveurs de la variable **env** et ajouter les tags Name et ENV pour chaque instance

```
vi main.tf
```

```
resource "aws_instance" "ec2" {
    ami = "ami-0d31449d0dd5f363f"
    instance_type = "t2.micro"
    for_each = var.env
    tags = {
        Name = "${each.key}"
        Env = "${each.value}"
    }
}
```

CHAPITRE 14

Solution Workspace

14.1 créer le projet

```
vi main.tf
```

```
resource "aws_instance" "ec2" {
    ami = var.region == "eu-west-1" ? "ami-0d31449d0dd5f363f" :  "ami-0e603d96bf395bc01"
    instance_type = "t2.micro"
    tags = {
        Name = "app-${var.region}"
    }
}
```

14.2 Créer le workspace

```
terraform workspace new west1
```

```
terraform apply -var region="eu-west-1"
```

CHAPITRE 15

Solution BackEnd

15.1 créer le S3

```
more s3.tf
```

```
resource "aws_s3_bucket" "example" {
  bucket = "terraform-user1-s3-bucket-19-07-2023"

  tags = {
    Name        = "My bucket"
    Environment = "Dev"
  }
}
```

15.2 créer le projet

```
more main.tf
```

```
provider "aws" {
  region = "eu-west-1"
}

resource "aws_instance" "my_ec2" {
  ami  = "ami-0d31449d0dd5f363f"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.instance_sg.id]
  tags = {
      Name = "${terraform.workspace == "prod" ? "prod-ec2" : "default-ec2"}"
```

(suite sur la page suivante)

(Suite de la page précédente)

```
  }
}

terraform {
  backend "s3" {
    bucket = "terraform-user1-s3-bucket-19-07-2023"
    key    = "states/terraform.state"
    region = "eu-west-1"
  }
}

resource "aws_security_group" "instance_sg" {
    name = "${terraform.workspace == "prod" ? "prod-ec2" : "default-ec2"}"

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

    ingress {
        from_port    = 80
        to_port      = 80
        protocol     = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

}

output "public_ip" {
    value = aws_instance.my_ec2.public_ip
}

region = "eu-west-1"
tags = {
    Name = "dev"
}
```

CHAPITRE 16

Dynamic block

```
more main.tf
```

```
locals {
  ports_in = [
    443,
    80,
    22
  ]
  ports_out = [
    0
  ]
}

resource "aws_vpc" "test" {
  cidr_block = "10.10.0.0/16"
}

resource "aws_security_group" "test" {
  name        = "test"
  description = "test"
  vpc_id      = aws_vpc.test.id

  dynamic "ingress" {
    for_each = toset(local.ports_in)
    content {
      from_port         = ingress.value
      to_port           = ingress.value
      protocol          = "tcp"
      cidr_blocks       = ["0.0.0.0/0"]
    }
  }
```

(suite sur la page suivante)

(Suite de la page précédente)

```
  dynamic "egress" {
    for_each = toset(local.ports_out)
    content {
      from_port        = egress.value
      to_port          = egress.value
      protocol         = "-1"
      cidr_blocks      = ["0.0.0.0/0"]
    }
  }
}
```

CHAPITRE 17

Tools

17.1 TFLint

TFLint est un outil d'analyse statique (linter) pour le code Terraform qui se concentre sur la recherche d'erreurs potentielles et de violations des meilleures pratiques dans vos configurations Terraform avant de les appliquer.

17.1.1 Install

```
$ curl -s https://raw.githubusercontent.com/terraform-linters/tflint/master/install_
↪linux.sh | bash
```

17.1.2 Utilisation

```
$ tflint --init
```

```
$ tflint

2 issue(s) found:

Warning: `region` variable has no type (terraform_typed_variables)

  on variables.tf line 1:
  1: variable "region" {}

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.7.0/
↪docs/rules/terraform_typed_variables.md

Warning: `instance_type` variable has no type (terraform_typed_variables)
```

(suite sur la page suivante)

(Suite de la page précédente)

```
  on variables.tf line 2:
  2: variable "instance_type" {}

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.7.0/
↪docs/rules/terraform_typed_variables.md

Warning: `amis` variable has no type (terraform_typed_variables)

  on variables.tf line 3:
  3: variable "amis" {}

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.7.0/
↪docs/rules/terraform_typed_variables.md

Warning: `tag_name` variable has no type (terraform_typed_variables)

  on variables.tf line 4:
  4: variable "tag_name" {}

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.7.0/
↪docs/rules/terraform_typed_variables.md
```

17.1.3 Ajout de plugins

```
$ vi .tflint.hcl
```

```
plugin "aws" {
    enabled = true
    version = "0.31.0"
    source  = "github.com/terraform-linters/tflint-ruleset-aws"
}
```

```
$ tflint --init
$ tflint

Error: "t2.xmicro" is an invalid value as instance_type (aws_instance_invalid_type)
```

17.2 Terrascan

17.2.1 Install

```
$ curl -L "$(curl -s https://api.github.com/repos/tenable/terrascan/releases/latest |␣
↪grep -o -E "https://.+?_Darwin_x86_64.tar.gz")" > terrascan.tar.gz
$ tar -xf terrascan.tar.gz terrascan && rm terrascan.tar.gz
$ sudo install terrascan /usr/local/bin && rm terrascan
```

17.2.2 Utilisation

```
$ terrascan init
```

```
$ terrascan scan

    Violation Details -

            Description    :        EC2 instances should disable IMDS or require IMDSv2␣
↪as this can be related to the weaponization phase of kill chain
            File           :        main.tf
            Module Name    :        root
            Plan Root      :        ./
            Line           :        1
            Severity       :        MEDIUM
            -----------------------------------------------------------------------

            Description    :        Ensure that detailed monitoring is enabled for EC2␣
↪instances.
            File           :        main.tf
            Module Name    :        root
            Plan Root      :        ./
            Line           :        1
            Severity       :        HIGH
            -----------------------------------------------------------------------

            Description    :        Ensure that your AWS application is not deployed␣
↪within the default Virtual Private Cloud in order to follow security best practices
            File           :        main.tf
            Module Name    :        root
            Plan Root      :        ./
            Line           :        1
            Severity       :        MEDIUM
            -----------------------------------------------------------------------


    Scan Summary -

            File/Folder         :   /home/sadmin/lab2
            IaC Type            :   terraform
            Scanned At          :   2024-05-24 08:29:23.10376681 +0000 UTC
            Policies Validated  :   5
            Violated Policies   :   3
            Low                 :   0
            Medium              :   2
            High                :   1
```

CHAPITRE 18

Downloads

Support de cours pdf

Labs du cours pdf

Aide-mémoire Terraform png

```
"eu-west-1" = "ami-0d31449d0dd5f363f"
"eu-west-2" = "ami-0e603d96bf395bc01"
"eu-west-3"  = "ami-016541e1c72b73883"
```

CHAPITRE 19

Emargement

Lien Bienvenue Formation

user : terraform

CHAPITRE 20

Indices and tables

- genindex
- modindex
- search