

TERRAFORM

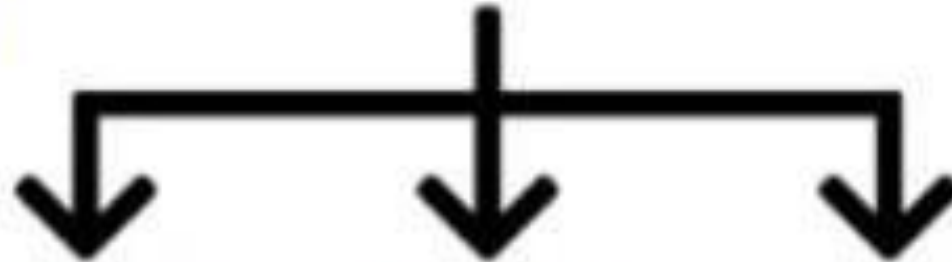
:

Les Bases
pour DEVOPS



HashiCorp

Terraform



Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

Remote management

Module

Mini-projet

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

Remote management

Module

Mini-projet

Ulrich MONJI - Ingénieur en
Systèmes, Réseaux et
Telecommunications - UTT

Atos-Worldline - Ingénieur Système

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

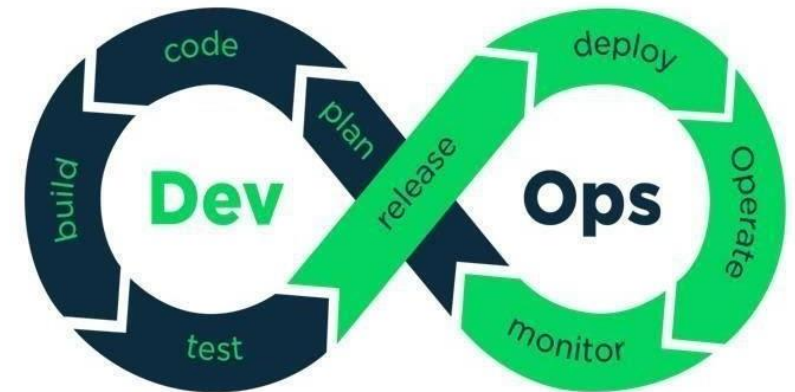
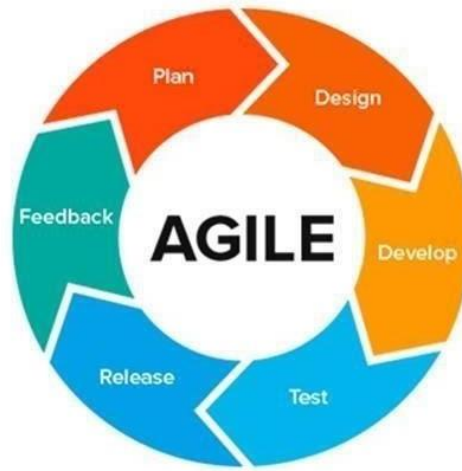
Remote management

Module

Mini-projet

- Agile: méthode de développement
- DevOps: agilité dans le Dev et l'Ops = CI + CD

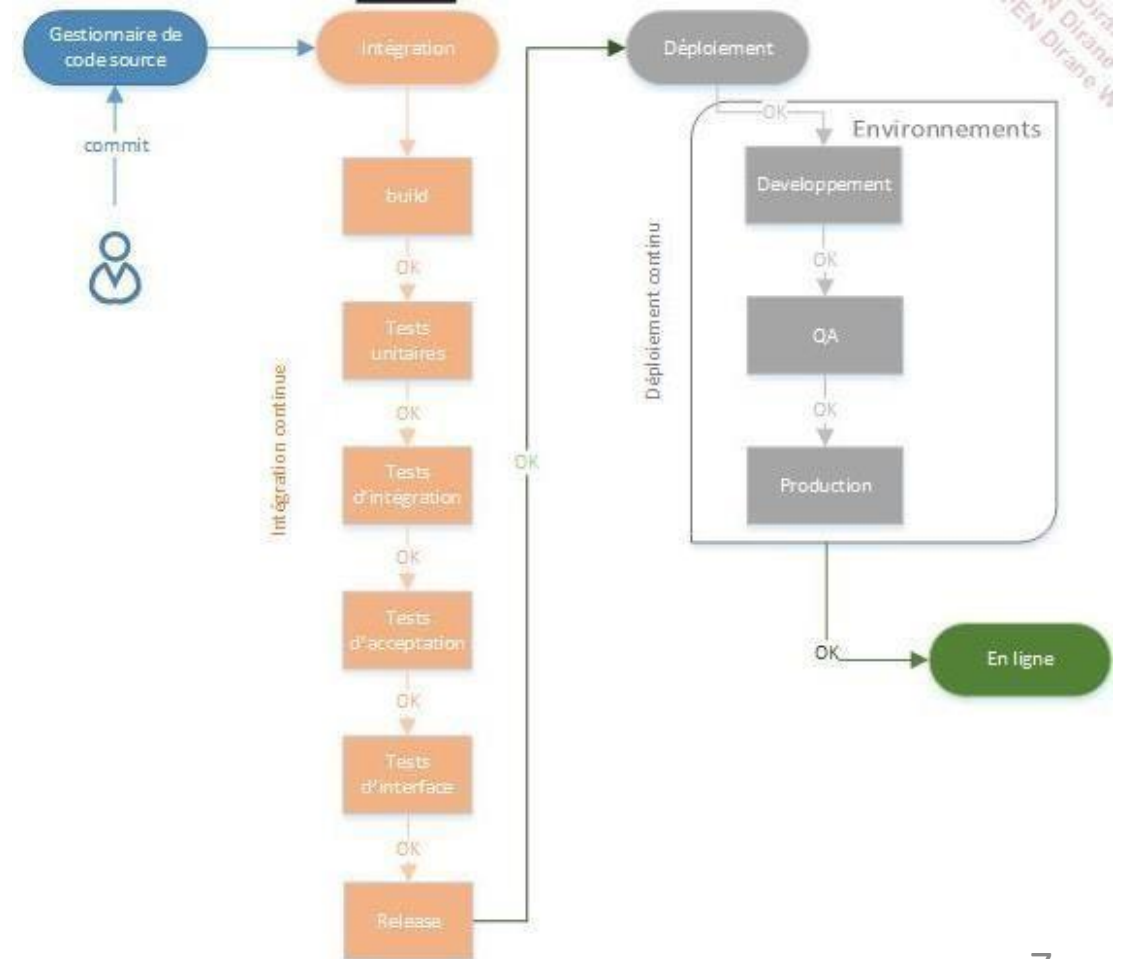
Agile vs. DevOps



TERRAFORM

Introduction au CI/CD et à L'IAC

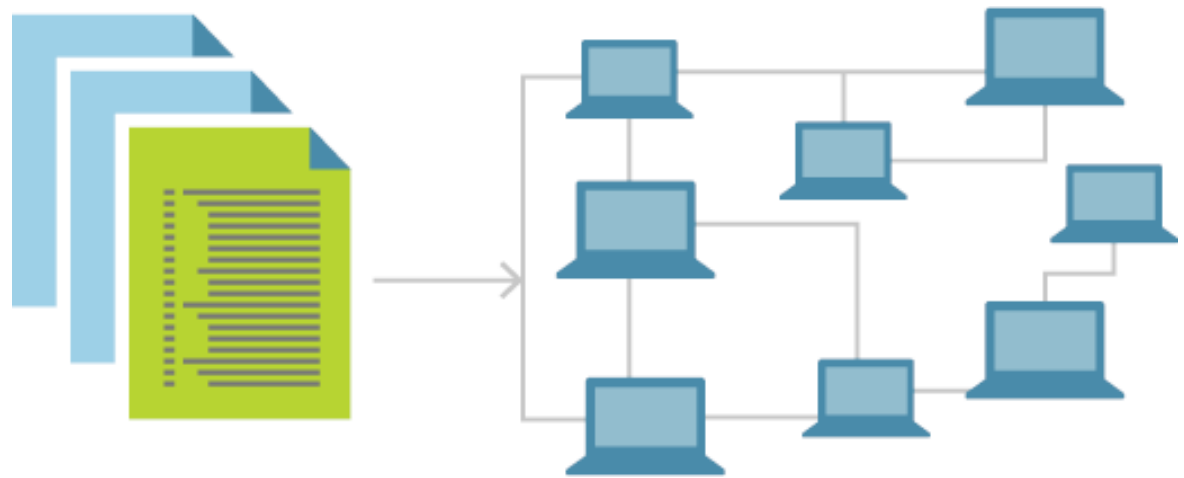
- Intégration en continu
- Test en continu
- Déploiement en continu



TERRAFORM

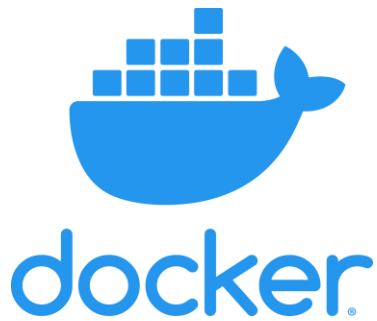
Introduction au CI/CD et à L'IAC

- Réutilisation
- Evolutivité
- Collaboration



TERRAFORM

Infrastructure As Code



SALTSTACK



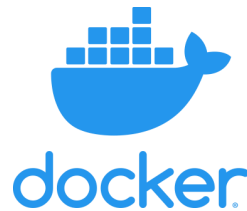
TERRAFORM

Types d'Outils d'Infrastructure As Code

Configuration Management



Server Templating



Provisioning Tools



Infrastructure as Code

ec2.sh

```
#!/bin/bash
```

```
IP_ADDRESS="10.2.2.1"
```

```
EC2_INSTANCE=$(ec2-run-instances --instance-type  
t2.micro ami-0edab43b6fa892279)
```

```
INSTANCE=$(echo ${EC2_INSTANCE} | sed 's/*INSTANCE //' |  
| sed 's/ .*//')
```

```
# Wait for instance to be ready  
while ! ec2-describe-instances $INSTANCE | grep -q  
"running"  
do  
    echo Waiting for $INSTANCE is to be ready...  
done
```

```
# Check if instance is not provisioned and exit  
if [ ! $(ec2-describe-instances $INSTANCE | grep -q  
"running") ]; then  
    echo Instance $INSTANCE is stopped.  
    exit  
fi
```

```
ec2-associate-address $IP_ADDRESS -i $INSTANCE
```

```
echo Instance $INSTANCE was created successfully!!!
```



Services

Resource Groups



1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Groups

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign your instance.

AMI Details



Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0b1e2eeb33ce3d66f

Free tier
eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance.

Root Device Type: ebs

Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)
t2.micro	Variable	1	1	EBS only

Security Groups

Security group name

launch-wizard-1

Description

launch-wizard-1 created 2020-07-09T15:48:36.426-04:00

Type ⓘ

Protocol ⓘ

Port Range ⓘ

This security group has

Instance Details

Number of instances 1

Network vpc-fe3baa86

Infrastructure as Code

ec2.sh

```
#!/bin/bash

IP_ADDRESS="10.2.2.1"

EC2_INSTANCE=$(ec2-run-instances --instance-type
t2.micro ami-0edab43b6fa892279)

INSTANCE=$(echo ${EC2_INSTANCE} | sed 's/*INSTANCE //'
| sed 's/ .*//')

# Wait for instance to be ready
while ! ec2-describe-instances $INSTANCE | grep -q
"running"
do
    echo Waiting for $INSTANCE is to be ready...
done

# Check if instance is not provisioned and exit
if [ ! $(ec2-describe-instances $INSTANCE | grep -q
"running") ]; then
    echo Instance $INSTANCE is stopped.
    exit
fi

ec2-associate-address $IP_ADDRESS -i $INSTANCE

echo Instance $INSTANCE was created successfully!!!
```

main.tf

```
resource "aws_instance" "webserver" {
    ami
        = "ami-0edab43b6fa892279"
    instance_type = "t2.micro"
}
```

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

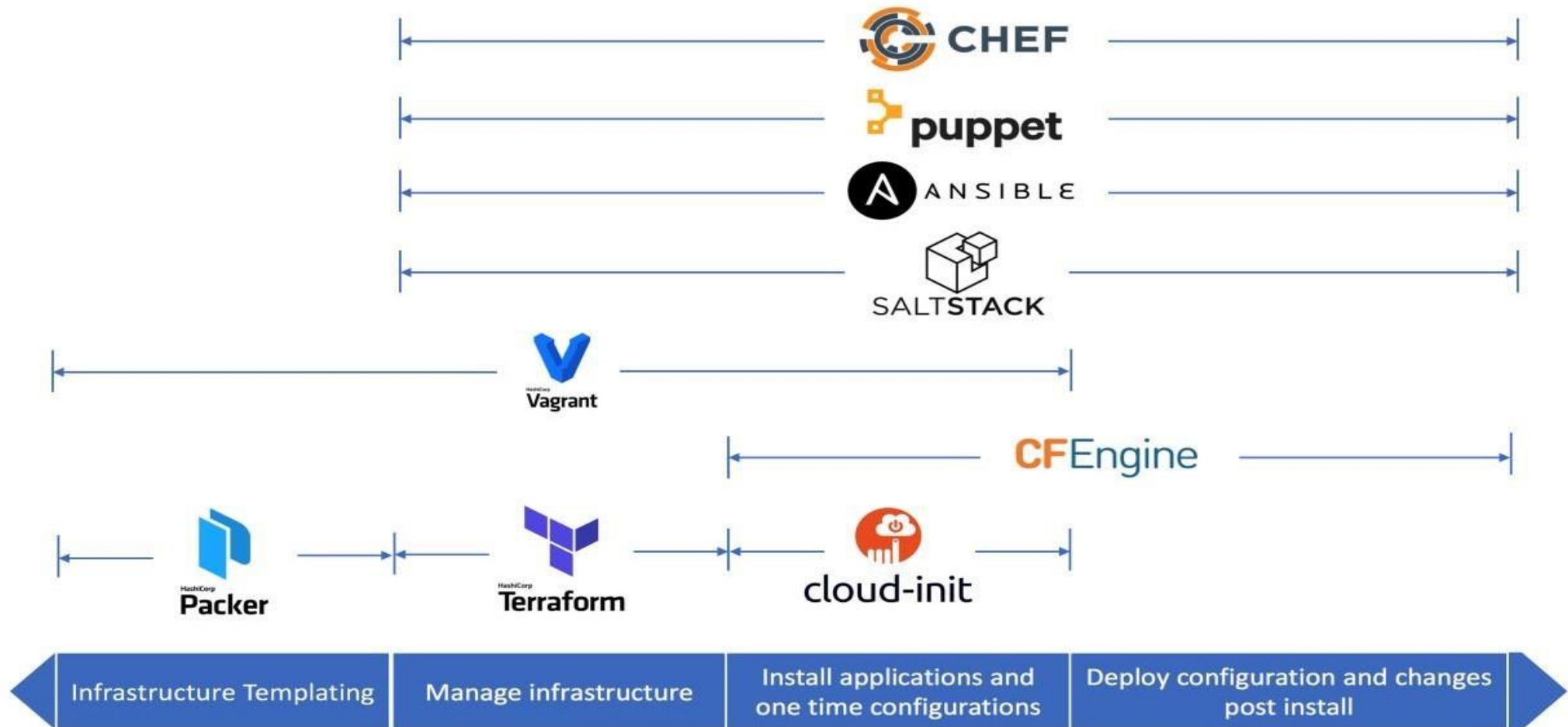
Remote management

Module

Mini-projet

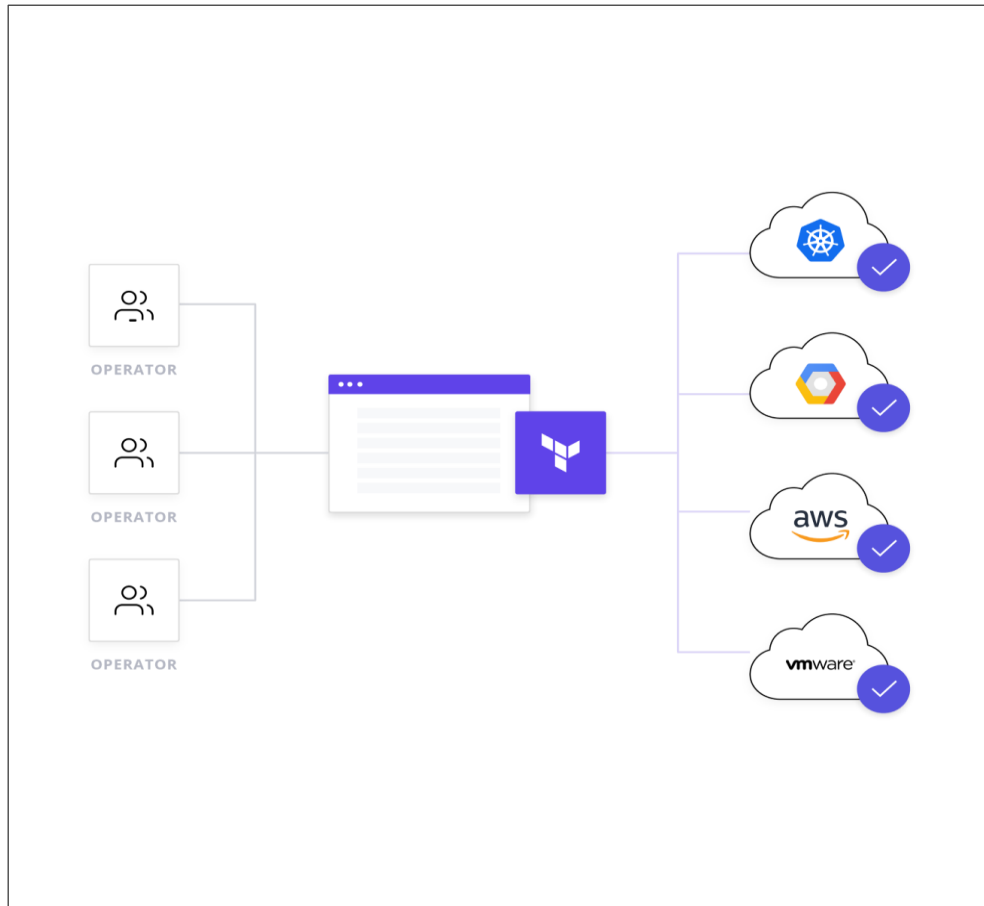
TERRAFORM

ALTERNATIVES A TERRAFORM



TERRAFORM

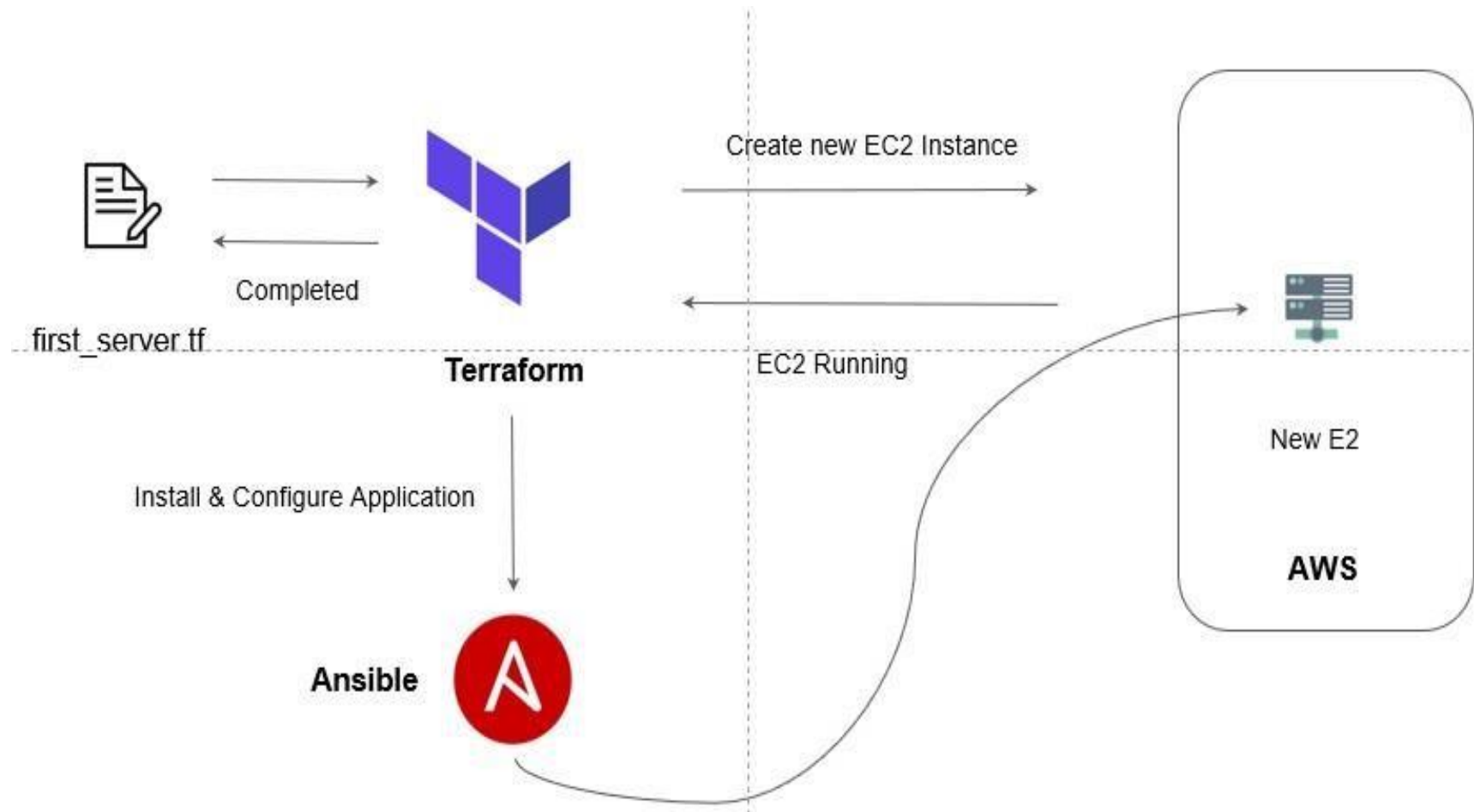
POURQUOI TERRAFORM (AVANTAGES)



- Multi-Cloud (de multiple providers)
- Gratuit
- Langage facile à lire
- Extensible
- S'intègre avec les outils de Configuration Management

TERRAFORM

IAC et CM (Complémentaires)



TERRAFORM

Installation de Terraform

```
>_  
  
$ wget https://releases.hashicorp.com/terraform/0.13.0/terraform_0.13.0_linux_amd64.zip  
$ unzip terraform_0.13.0_linux_amd64.zip  
$ mvterraform /usr/local/bin  
$ terraform version  
Terraform v0.13.0
```



macOS
64-bit



FreeBSD
32-bit | 64-bit | Arm



Linux
32-bit | 64-bit | Arm



OpenBSD
32-bit | 64-bit



Solaris
64-bit



Windows
32-bit | 64-bit

- Créez un compte AWS gratuit
- Protégez votre compte root avec un mot de passe fort
- Créez un compte nominatif avec les droits full admin et qui vous permettra de déployer les ressources
- Installez un IDE, par exemple Visual Studio Code et installez un plugin terraform pour vous faciliter la correction syntaxique
- Vous êtes prêt à installer terraform !

- Installez Terraform en récupérant le binaire via le lien suivant: <https://www.terraform.io/downloads.html>
- Si vous êtes sous windows, copiez le binaire dans un dossier de votre disque dur system par exemple C:\terraform\ , ensuite il vous faudra rajouter le repertoire precedent dans le PATH de votre système d'exploitation
- Si vous êtes sous linux, vous pouvez le déplacer dans /usr/bin/après l'avoir rendu executable
- Pour verifiez l'instalation, veuillez juste entrez la commande terraform et l'aide apparaitra !

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

Remote management

Module

Mini-projet

DÉPLOYEZ VOS PREMIÈRES RESSOURCES : Le langage HCL

HCL – Declarative Language

```
<block> <parameters> {  
  key1 = value1  
  key2 = value2  
}
```

```
aws.tf  
  
resource "aws_instance" "webserver" {  
  ami = "ami-0c2f25c1f66a1ff4d"  
  instance_type = "t2.micro"  
}
```

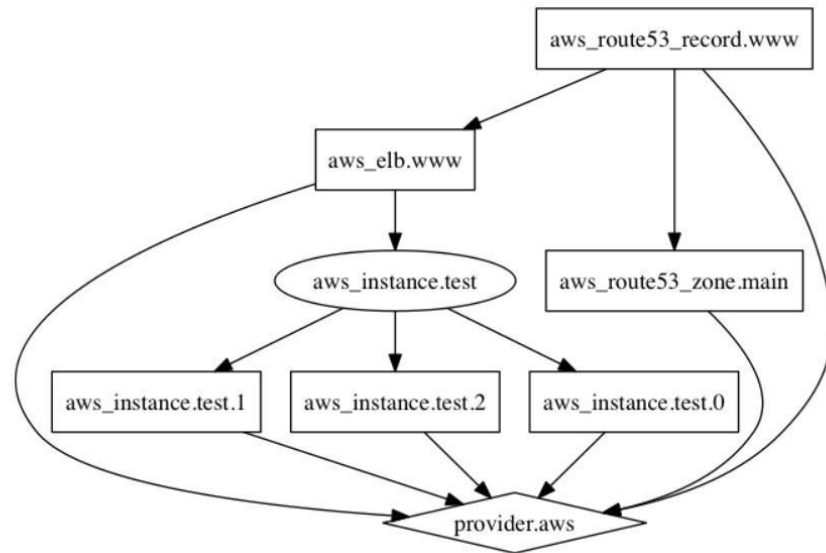
DÉPLOYEZ VOS PREMIÈRES RESSOURCES : Ressources et Providers

Ressources et Providers

Providers



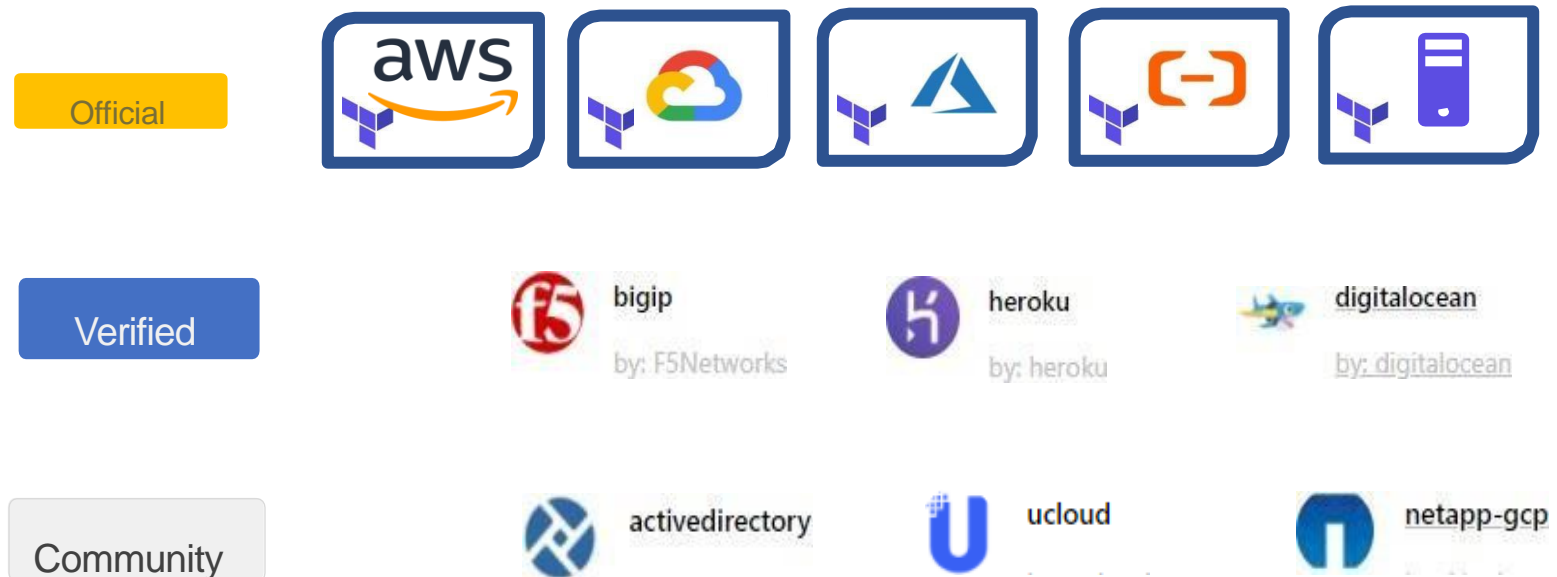
Ressources



- **Providers:** fourni le plugin permettant de communiquer avec la plateforme à provisionner (AWS, GCP, AZURE, Digital Ocean, OVH ...), ils sont versionnés et mis à jour constamment (attention aux mises à jour)
- **Ressource:** Représente le type d'objet à créer sur la plateforme à provisionner

DÉPLOYEZ VOS PREMIÈRES RESSOURCES : Les Providers

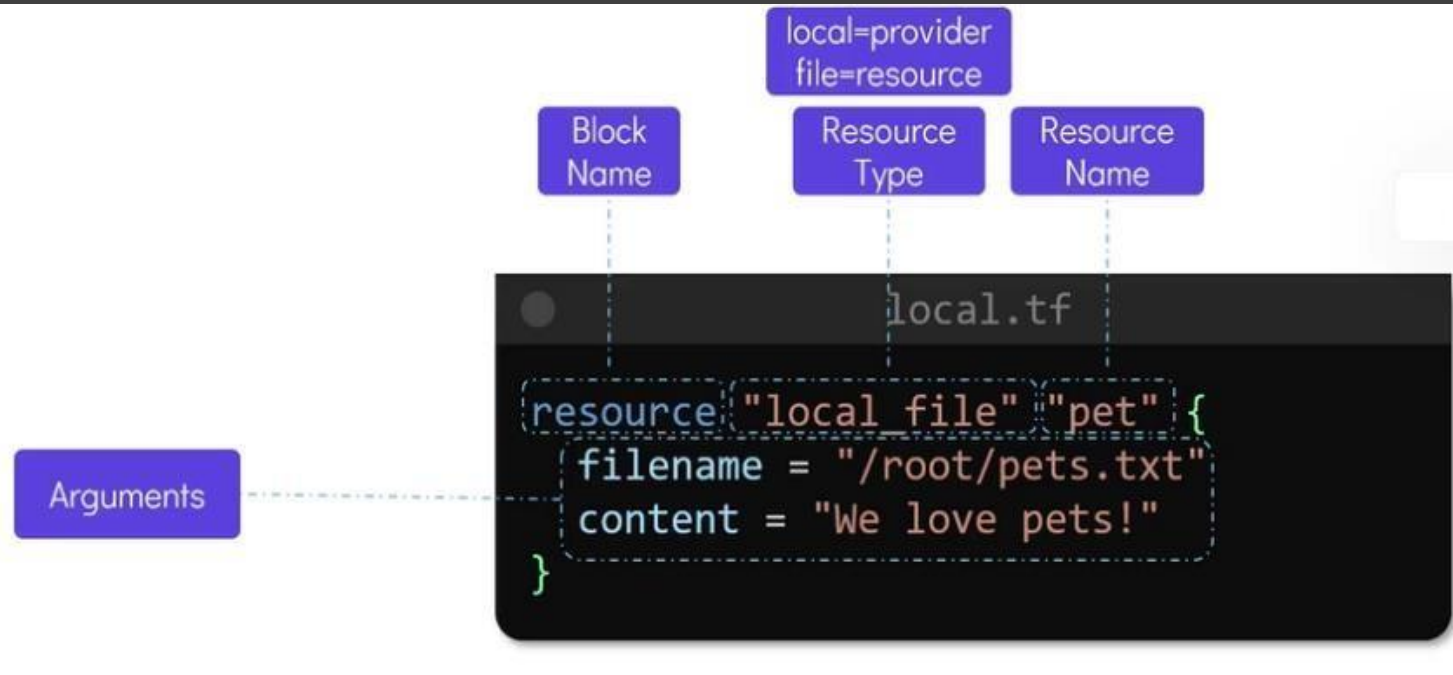
Types de Providers



registry.terraform.io

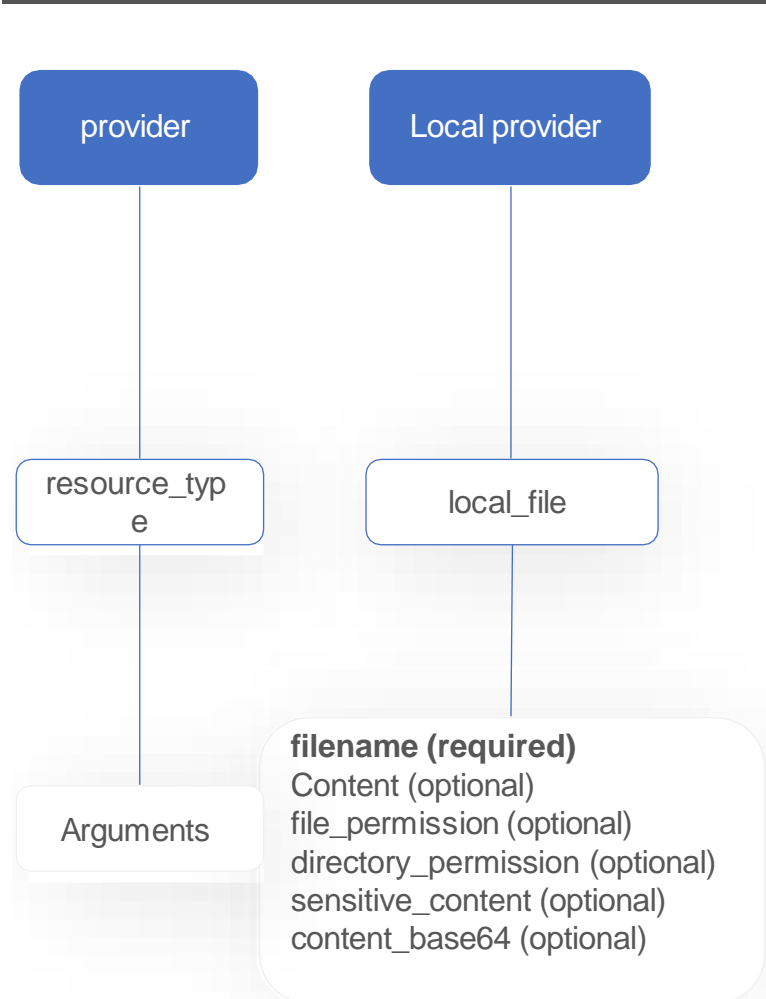
DÉPLOYEZ VOS PREMIÈRES RESSOURCES : Les ressources

HCL – Declarative Language



TERRAFORM

LES RESSOURCES



local provider

▼ Resources

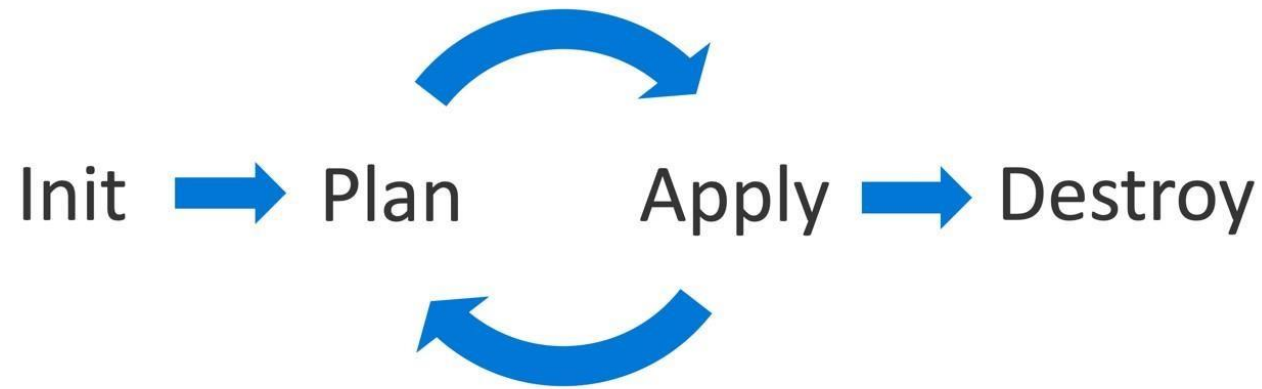
• [local_file](#)

› Data Sources

Argument Reference

The following arguments are supported:

- `content` - (Optional) The content of file to create. Conflicts with `sensitive_content` and `content_base64`.
- `sensitive_content` - (Optional) The content of file to create. Will not be stored in the state file. Conflicts with `content` and `content_base64`.
- `content_base64` - (Optional) The base64 encoded content of the file to create. Useful when dealing with binary data. Conflicts with `content` and `sensitive_content`.
- `filename` - (Required) The path of the file to create.
- `file_permission` - (Optional) The permission to set for the created file. Expected as a string. The default value is `"0777"`.
- `directory_permission` - (Optional) The permission to set for any directories created. Expected as a string. The default value is `"0777"`.



**DÉPLOYEZ VOS PREMIÈRES RESSOURCES
: COMMANDES DE BASE**

TERRAFORM

DEPLOYEZ VOS PREMIERES RESSOURCES : FICHER (local provider)

```
> _  
  
$ terraform init  
Initializing the backend...  
  
Initializing provider plugins...  
- Finding latest version of hashicorp/local...  
- Installing hashicorp/local v2.0.0...  
- Installed hashicorp/local v2.0.0 (signed by HashiCorp)  
  
The following providers do not have any version constraints in  
configuration,  
so the latest version was installed.  
  
To prevent automatic upgrades to new major versions that may  
contain breaking  
changes, we recommend adding version constraints in a  
required_providers block  
in your configuration, with the constraint strings suggested  
below.  
  
* hashicorp/local: version = "~> 2.0.0"  
  
Terraform has been successfully initialized!
```

```
> _  
  
$ ls /root/terraform-local-file/.terraform  
plugins
```

TERRAFORM

DEPLOYEZ VOS PREMIERES RESSOURCES : FICHER (local provider)

```
local.tf

resource "local_file" "pet" {
  filename = "/root/pets.txt"
  content = "We love pets!"
  file_permission = "0700"
}
```



```
> _

$ terraform plan

local_file.pet: Refreshing state...
[id=5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf]

-----
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

# local_file.pet must be replaced
-/+ resource "local_file" "pet" {
  content           = "We love pets!"
  directory_permission = "0777"
  ~ file_permission = "0777" -> "0700" # forces replacement
  filename          = "/root/pet.txt"
  ~ id              =
  "5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf" -> (known after apply)
}

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

-----
Note: You didn't specify an "-out" parameter to save this plan, so
Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.
```

TERRAFORM

DEPLOYEZ VOS PREMIERES RESSOURCES : FICHER (local provider)

```
> _  
  
$ls -ltr /root/pets.txt  
-rwx----- 1 root root 30 Aug 17 23:20 pet.txt
```



```
> _  
  
$terraform apply  
  
#local_file.pet must be replaced  
-/+ resource "local_file" "pet" {  
    content          = "We love pets!"  
    directory_permission = "0777"  
    ~ file_permission = "0777" -> "0700" # forces replacement  
    filename         = "/root/pet.txt"  
    ~ id              =  
    "5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf" -> (known after apply)  
}  
  
Plan: 1 to add, 0 to change, 1 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  
  
local_file.pet: Destroying...  
[id=5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf]  
local_file.pet: Destruction complete after 0s  
local_file.pet: Creating...  
local_file.pet: Creation complete after 0s  
[id=5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf]  
  
Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
```

TERRAFORM

DEPLOYEZ VOS PREMIERES RESSOURCES : FICHER (local provider)



```
> _
$ terraform destroy
local_file.pet: Refreshing state...
[id=5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

#local_file.pet will be destroyed
- resource "local_file" "pet" {
  - content          = "My favorite pet is a gold fish" -> null
  - directory_permission = "0777" -> null
  - file_permission   = "0700" -> null
  - filename         = "/root/pet.txt" -> null
  - id               = "5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf" -> null
}

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

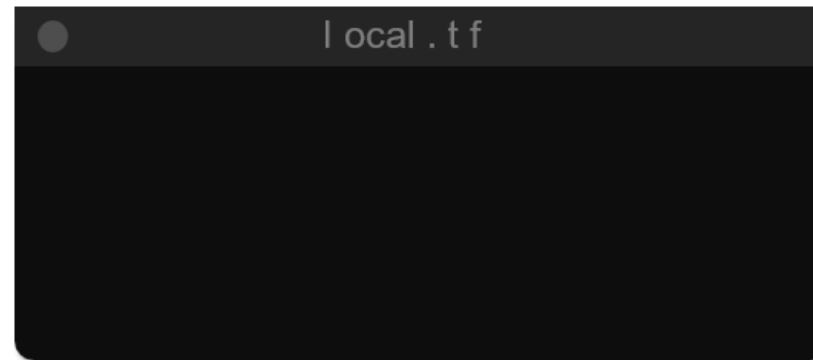
Enter a value: yes

local_file.pet: Destroying... [id=5f8fb950ac60f7f23ef968097cda0a1fd3c11bdf]
local_file.pet: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed
```


TERRAFORM

REPERTOIRE PROJET OU DE CONFIGURATION ET ORGANISATION DES FICHIERS



```
main.tf

resource "local_file" "pet" {
  filename = "/root/pets.txt"
  content = "We love pets!"
}

resource "random_pet" "my-pet" {
  prefix = "Mrs"
  separator = "."
  length = "1"
}
```

File Name	Purpose
main.tf	Main configuration file containing resource definition
variables.tf	Contains variable declarations
outputs.tf	Contains outputs from resources
provider.tf	Contains Provider definition

terraform.state



terraform.tfvars



terraform-provider.tf



terraform- instances.tf



aws




vm



DÉPLOYEZ VOS PREMIÈRES RESSOURCES : TFSTATE

- État de l'infra
- Sécurité (pas dans GIT)
- Remote state



DÉPLOYEZ VOS PREMIÈRES RESSOURCES :EC2

```
provider "aws" {  
    region      = "us-west-2"  
    access_key  = "PUT-YOUR-ACCESS-KEY-HERE"  
    secret_key  = "PUT-YOUR-SECRET-KEY-HERE"  
}  
  
resource "aws_instance" "myec2" {  
    ami = "ami-082b5a644766e0e6f"  
    instance_type = "t2.micro"  
}
```

DÉPLOYEZ VOS PREMIÈRES RESSOURCES :IAM USER

```
main.tf

provider "aws" {
  region = "us-west-2"
}

resource "aws_iam_user" "admin-user" {
  name = "lucy"
  tags = {
    Description = "Technical Team Leader"
  }
}
```

.aws/config/credentials

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

> _

```
$ export AWS_ACCESS_KEY_ID=AKIA44QH8DHBEXAMPLE
$ export AWS_SECRET_ACCESS_KEY_ID=je7MtGbClwBF/2tk/h3yCo8n...
$ export AWS_REGION=us-west-2
```

TP-2: Déployez votre Ressource AWS avec terraform

- Récupérez le secret et access key de votre compte (dans les paramètres sécurité de votre compte dans IAM)
- Créez une paire de clés dans EC2 et nommez cette clé devops-<votre prenom>, un fichier devops-<votre prenom>.pem sera téléchargé (conservez la jalousement)
- Créez un fichier ec2.tf dans un répertoire nommé tp-2
- Renseignez les informations permettant de créer une VM avec l'image centos suivante: centos7-minimal-v20190919.0.0 (ami-0083662ba17882949)
- ATTENTION nous travaillerons uniquement dans la région US East (N. Virginia) us-east-1 dans **toute cette formation**
- Vérifiez que votre instance est bien créée et observez le contenu de fichier tfstate
- Modifiez le fichier ec2.tf afin d'y inclure le tag de votre instance: "Name: ec2-<votre prenom>"
- Appliquez la modification et constatez les changements apportés ainsi que dans le fichier tfstate
- Supprimez votre ec2

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

Remote management

Module

Mini-projet

TERRAFORM

DEPLOIEMENTS DYNAMIQUES : Les variables (Définition)

Type	Example
string	"/root/pets.txt"
number	1
bool	true/false
any	Default Value
list	["cat", "dog"]
map	pet1 = cat pet2 = dog
object	Complex Data Structure
tuple	Complex Data Structure

```
variable "filename" {  
  default = "/root/pets.txt"  
  type = string  
  description = "the path of local file"  
}
```

```
variables.tf  
  
variable "prefix" {  
  default = ["Mr", "Mrs", "Sir"]  
  type = list 0 1 2  
}
```

```
maint.tf  
  
resource "random_pet" "my-pet" {  
  prefix = var.prefix[0]  
}
```

```
variables.tf  
  
variable file-content {  
  type = map  
  default = {  
    "statement1" = "We love pets!"  
    "statement2" = "We love animals!"  
  }  
}
```

```
maint.tf  
  
resource local_file my-pet {  
  filename = "/root/pets.txt"  
  content = var.file-content["statement2"]  
}
```

TERRAFORM

DEPLOIEMENTS DYNAMIQUES : Les variables (Définition)

Type	Example
string	"/root/pets.txt"
number	1
bool	true/false
any	Default Value
list	["cat", "dog"]
map	pet1 = cat pet2 = dog
object	Complex Data Structure
tuple	Complex Data Structure

```
variable "cats" {  
  default = {  
    "color" = "brown"  
    "name"  = "bella"  
  }  
  type = map(string)  
}
```

```
variable "pet_count" {  
  default = {  
    "dogs" = "3"  
    "cats" = "1"  
    "goldfish" = "2"  
  }  
  type = map(number)  
}
```

Set

Les variables de type **set** sont identiques aux variables « list », à la seule différence que dans un set il n'est pas possible d'avoir plus d'une fois le même élément

```
variables.tf  
  
variable "prefix" {  
  default = ["Mr", "Mrs", "Sir"]  
  type = set(string)  
}
```

```
variables.tf  
  
variable "prefix" {  
  default = ["Mr", "Mrs", "Sir", "Sir"]  
  type = set(string)  
}
```

```
variables.tf  
  
variable "fruit" {  
  default = ["apple", "banana"]  
  type = set(string)  
}
```

```
variables.tf  
  
variable "fruit" {  
  default = ["apple", "banana", "banana"]  
  type = set(string)  
}
```

```
variables.tf  
  
variable "age" {  
  default = ["10", "12", "15"]  
  type = set(number)  
}
```

```
variables.tf  
  
variable "age" {  
  default = ["10", "12", "15", "10"]  
  type = set(number)  
}
```

TERRAFORM

DEPLOIEMENTS DYNAMIQUES : Les variables (Définition)

Type	Example
string	"/root/pets.txt"
number	1
bool	true/false
any	Default Value
list	["cat", "dog"]
map	pet1 = cat pet2 = dog
object	Complex Data Structure
tuple	Complex Data Structure

Les Objets

```
variables.tf

variable "bella" {
  type = object({
    name = string
    color = string
    age = number
    food = list(string)
    favorite_pet = bool
  })

  default = {
    name = "bella"
    color = "brown"
    age = 7
    food = ["fish", "chicken", "turkey"]
    favorite_pet = true
  }
}
```

Les Tuples

```
variables.tf

variable kitty {
  type      = tuple([string, number, bool])
  default   = ["cat", 7, true]
}
```

```
variables.tf

variable kitty {
  type      = tuple([string, number, bool])
  default   = ["cat", 7, true, "dog"]
}
```

TERRAFORM

DEPLOIEMENTS DYNAMIQUES : Les variables (Définition)

Type	Example
string	"/root/pets.txt"
number	1
bool	true/false
any	Default Value
list	["cat", "dog"]
map	pet1 = cat pet2 = dog
object	Complex Data Structure
tuple	Complex Data Structure

Les Objets

```
variables.tf

variable "bella" {
  type = object({
    name = string
    color = string
    age = number
    food = list(string)
    favorite_pet = bool
  })

  default = {
    name = "bella"
    color = "brown"
    age = 7
    food = ["fish", "chicken", "turkey"]
    favorite_pet = true
  }
}
```

Les Tuples

```
variables.tf

variable kitty {
  type      = tuple([string, number, bool])
  default   = ["cat", 7, true]
}

variables.tf

variable kitty {
  type      = tuple([string, number, bool])
  default   = ["cat", 7, true, "dog"]
}
```

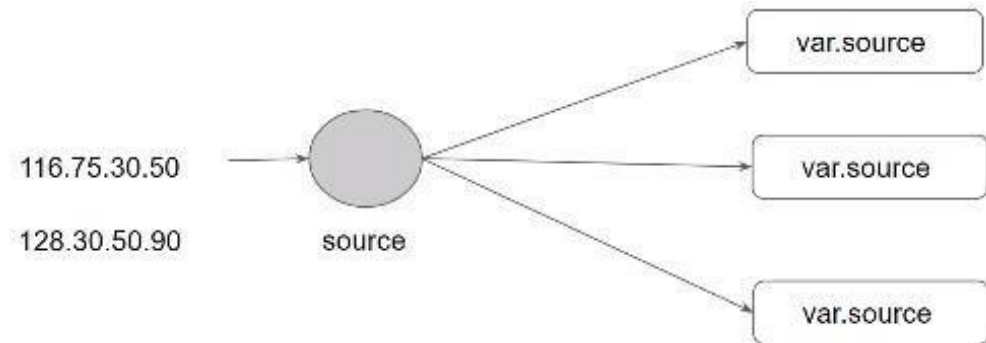

TERRAFORM

DEPLOIEMENTS DYNAMIQUES : Les variables (Utilisation)

```
provider "aws" {  
  region      = "us-west-2"  
  access_key  = "YOUR-ACCESS-KEY"  
  secret_key  = "YOUR-SECRET-KEY"  
}  
  
resource "aws_instance" "myec2" {  
  ami         = "ami-082b5a644766e0e6f"  
  instance_type = var.instance_type  
}
```

variables.tf

```
variable "instancetype" {  
  default = "t2.micro"  
}
```



TERRAFORM

DEPLOIEMENTS DYNAMIQUES : Les surcharges de variables

Loading Variable Values from CLI

```
terraform plan -var="instancetype=t2.small"
```

Loading from custom tfvars file

```
terraform plan -var-file="custom.tfvars"
```

Windows Specific Commands

```
setx TF_VAR_instancetype m5.large  
echo %TF_VAR_instancetype
```

Linux / MAC specific commands

```
export TF_VAR_instancetype t2.nano  
echo TF_VAR_instancetype
```

variables.tf

```
variable "username" {  
  type = number  
}  
  
variable "elb_name" {  
  type = string  
}  
  
variable "az" {  
  type = list  
}  
  
variable "timeout" {  
  type = number  
}
```

terraform.tfvars

```
elb_name="myelb"  
timeout="400"  
az=["us-west-1a","us-west-1b"]
```

terraform.tfvars

```
filename = "/root/pets.txt"  
content = "We love pets!"  
prefix = "Mrs"  
separator = "."  
length = "2"
```

```
$ export TF_VAR_filename="/root/pets.txt"  
$ export TF_VAR_content="We love pets!"  
$ export TF_VAR_prefix="Mrs"  
$ export TF_VAR_separator="."  
$ export TF_VAR_length="2"  
$ terraform apply
```

```
$ terraform apply -var "filename=/root/pets.txt" -var "content=We love  
Pets!" -var "prefix=Mrs" -var "separator=." -var "length=2"
```

TERRAFORM

DEPLOIEMENTS DYNAMIQUES : Ordre de priorité des variables

Variable Definition Precedence

Order	Option
1	Environment Variables
2	terraform.tfvars
3	*.auto.tfvars (alphabetical order)
4	-var or -var-file (command-line flags)



>_

```
$ export TF_VAR_filename="/root/cats.txt" 1
```

● terraform.tfvars


```
filename = "/root/pets.txt" 2
```

● variable.auto.tfvars

```
filename = "/root/mypet.txt" 3
```

>_

```
$ terraform apply -var "filename=/root/best-pet.txt" 4
```



RENDEZVOS DÉPLOIEMENTS DYNAMIQUE: EXEMPLE

```
provider "aws" {  
  region      = "us-west-2"  
  access_key  = "YOUR-ACCESS-KEY"  
  secret_key  = "YOUR-SECRET-KEY"  
}  
  
resource "aws_instance" "myec2" {  
  ami = "ami-082b5a644766e0e6f"  
  instance_type = var.instance_type  
}
```

variables.tf

```
variable "instancetype" {  
  default = "t2.micro"  
}
```

terraform.tfvars

```
instancetype="t2.large"
```

TERRAFORM

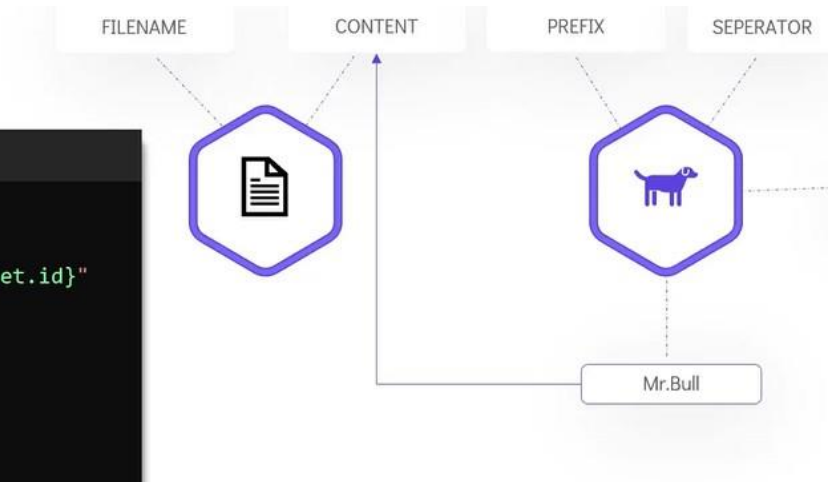
DEPLOIEMENTS DYNAMIQUES : ATTRIBUTS

- Customization de votre déploiement
- Référencement des ressources pour la création de nouvelles

```
provider "aws" {  
  region      = "us-west-2"  
  access_key  = "PUT-YOUR-ACCESS-KEY-HERE"  
  secret_key  = "PUT-YOUR-SECRET-KEY-HERE"  
}  
  
resource "aws_eip" "lb" {  
  vpc      = true  
}  
  
output "eip" {  
  value = aws_eip.lb  
}  
  
resource "aws_s3_bucket" "mys3" {  
  bucket = "kplabs-attribute-demo-001"  
}  
  
output "mys3bucket" {  
  value = aws_s3_bucket.mys3  
}
```

```
main.tf  
  
resource "local_file" "pet" {  
  filename = var.filename  
  content  = "My favorite pet is ${random_pet.my-pet.id}"  
}  
  
resource "random_pet" "my-pet" {  
  prefix      = var.prefix  
  separator   = var.separator  
  length      = var.length  
}
```

```
>_  
  
random_pet.my-pet: Creating...  
local_file.pet: Creating...  
random_pet.my-pet: Creation complete after 0s [id=Mr.bull]  
local_file.pet: Creation complete after 0s  
[id=059090e865809f9b6debfd7aebf48fdce2220a6]  
  
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```



Attribute Reference

The following attributes are supported:

- `id` - (string) The random pet name

TERRAFORM

DEPLOIEMENTS DYNAMIQUES : OUTPUTS

- Récupération d'informations dynamiques
- Référencement des ressources entre plusieurs répertoires projets

```
provider "aws" {  
  region      = "us-west-2"  
  access_key  = "PUT-YOUR-ACCESS-KEY-HERE"  
  secret_key  = "PUT-YOUR-SECRET-KEY-HERE"  
}  
  
resource "aws_eip" "lb" {  
  vpc      = true  
}  
  
output "eip" {  
  value = aws_eip.lb  
}  
  
resource "aws_s3_bucket" "mys3" {  
  bucket = "kplabs-attribute-demo-001"  
}  
  
output "mys3bucket" {  
  value = aws_s3_bucket.mys3  
}
```

```
resource "local_file" "pet" {  
  filename = var.filename  
  content  = "My favorite pet is ${random_pet.my-pet.id}"  
}  
  
resource "random_pet" "my-pet" {  
  prefix = var.prefix  
  separator = var.separator  
  length = var.length  
}  
  
output pet-name {  
  value          = random_pet.my-pet.id  
  description    = "Record the value of pet ID generated by  
random_pet resource"  
}
```

```
>_  
$ terraform output  
pet-name = Mrs.gibbon
```

```
>_  
$ terraform output pet-name  
Mrs.gibbon
```

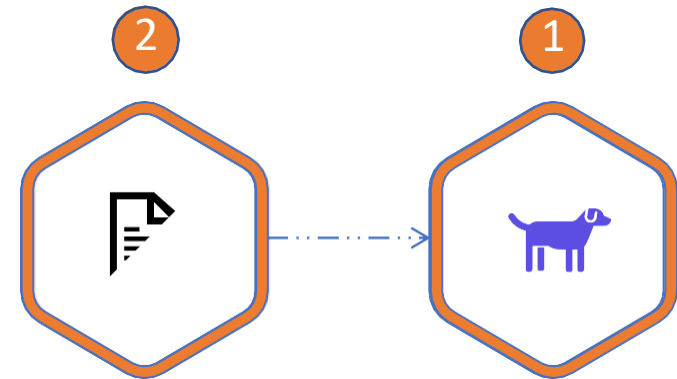
Dépendances explicites

```
main.tf

resource "local_file" "pet" {
  filename = var.filename
  content = "My favorite pet is Mr.Cat"

  depends_on = [
    random_pet.my-pet
  ]
}

resource "random_pet" "my-pet" {
  prefix = var.prefix
  separator = var.separator
  length = var.length
}
```

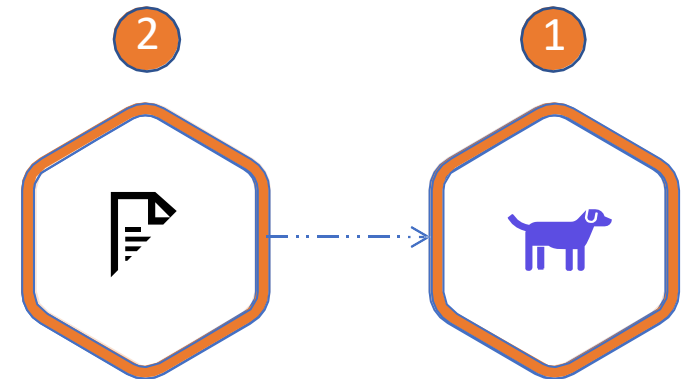


Dépendances implicites

```
main.tf

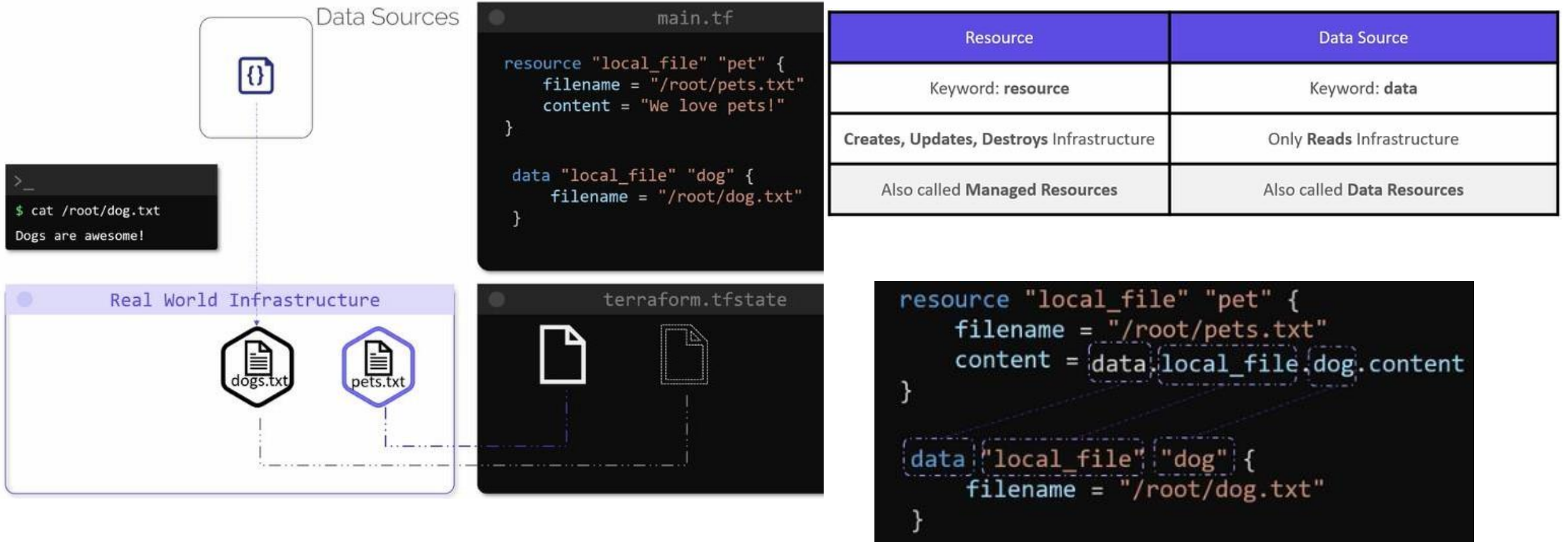
resource "local_file" "pet" {
  filename = var.filename
  content = "My favorite pet is ${random_pet.my-pet.id}"
}

resource "random_pet" "my-pet" {
  prefix = var.prefix
  separator = var.separator
  length = var.length
}
```



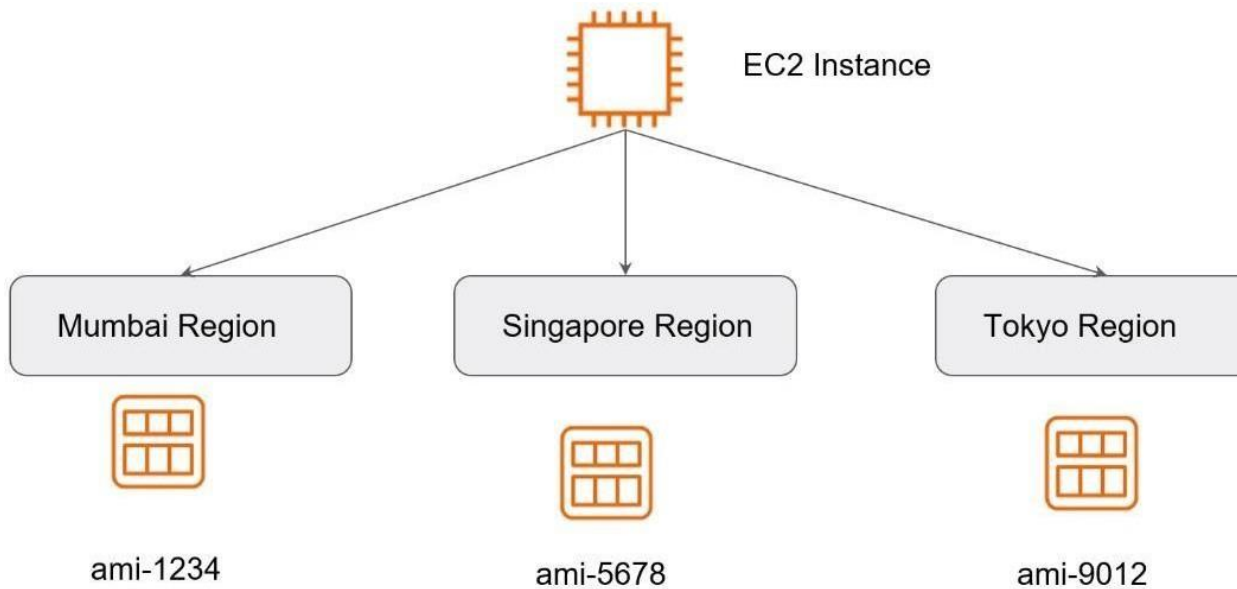
TERRAFORM

DEPLOIEMENTS DYNAMIQUES : DATA SOURCE



TERRAFORM

DEPLOIEMENTS DYNAMIQUES : DATA SOURCE (EXEMPLE)



data-source.tf

```
provider "aws" {  
  region      = "ap-southeast-1"  
  access_key  = "YOUR-ACCESS-KEY"  
  secret_key  = "YOUR-SECRET-KEY"  
}  
  
data "aws_ami" "app_ami" {  
  most_recent = true  
  owners     = ["amazon"]  
  
  filter {  
    name   = "name"  
    values = ["amzn2-ami-hvm*"]  
  }  
}  
  
resource "aws_instance" "instance-1" {  
  ami           = data.aws_ami.app_ami.id  
  instance_type = "t2.micro"  
}
```

TP-3: Déployez une infrastructure dynamique

- L'objectif est de déployer une instance ec2 avec une ip publique et un security group
- IP publique: vous allez créer une ip publique pour votre EC2
- Security Group: créez une security group pour ouvrir le port 80 et 443, attachez cette security group à votre IP publique
- EC2
 - Votre ec2 doit avoir une taille variabilisée, la valeur par défaut devrait être t2.nano et la valeur à surcharger sera t2.micro
 - L'image AMI à utiliser sera l'image la plus à jour de AMAZON LINUX
 - Spécifiez la key pair à utiliser (devops-<votre prénom>)
 - Attachez l'ip publique à votre instance
 - Variabilisez le Tag afin qu'il contienne au moins le tag: « Name: ec2-<votre prénom> » le N est bien en majuscule
- Supprimez vos ressources avec terraform destroy
- Créez un dossier tp-3 comme vous l'avez fait au tp-2 pour conserver votre code

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

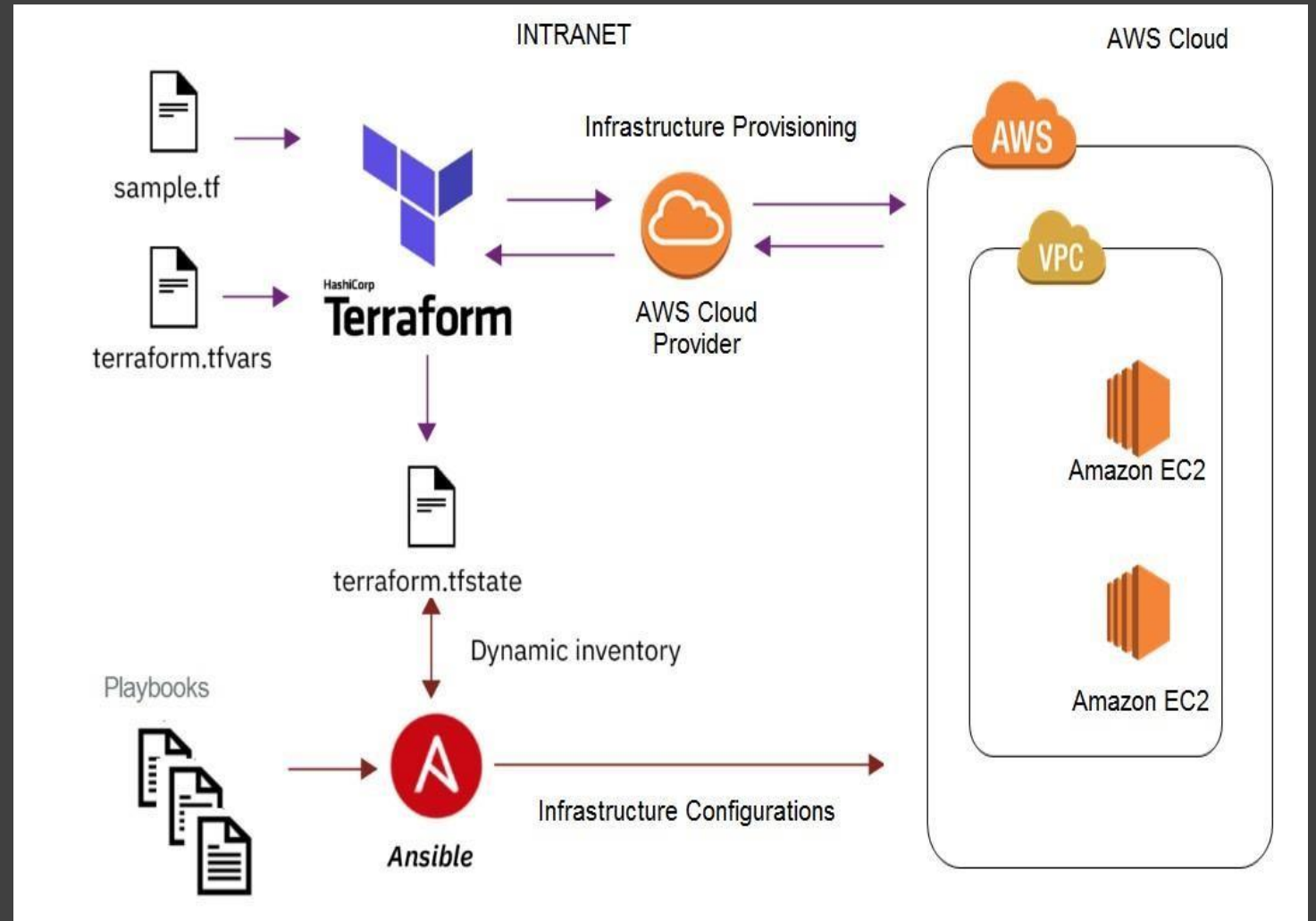
Terraform Provisioners

Remote management

Module

Mini-projet

TERRAFORM PROVISIONERS (1/3): PROBLÉMATIQUE



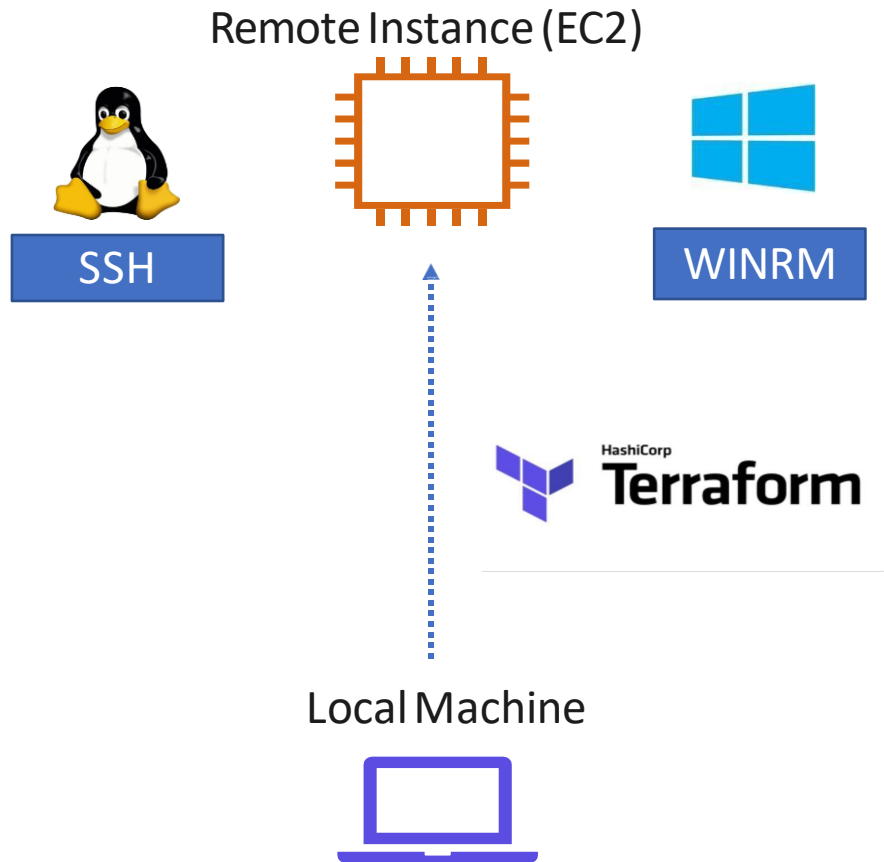
TERRAFORM

PROVISIONERS: LOCAL PROVISIONERS

```
resource "aws_instance" "myec2" {  
    ami = "ami-082b5a644766e0e6f"  
    instance_type = "t2.micro"  
  
    provisioner "local-exec" {  
        command = "echo ${aws_instance.myec2.private_ip} >> private_ips.txt"  
    }  
}
```

TERRAFORM

PROVISIONERS: REMOTE PROVISIONERS



- ✓ Network Connectivity (Security Group)
- ✓ Authentication (SSH Key Pair)

```
resource "aws_instance" "myec2" {  
  ami = "ami-082b5a644766e0e6f"  
  instance_type = "t2.micro"  
  key_name = "kplabs-terraform"  
  
  provisioner "remote-exec" {  
    inline = [  
      "sudo amazon-linux-extras install -y nginx1.12",  
      "sudo systemctl start nginx"  
    ]  
  }  
  
  connection {  
    type = "ssh"  
    user = "ec2-user"  
    private_key = file("../kplabs-terraform.pem")  
    host = self.public_ip  
  }  
}
```

TERRAFORM

PROVISIONERS: REMOTE PROVISIONERS

```
main.tf

resource "aws_instance" "webserver" {
  ami           = "ami-0edab43b6fa892279"
  instance_type = "t2.micro"
  user_data = <<EOF
    #!/bin/bash
    sudo apt update
    sudo apt install nginx -y
    systemctl enable nginx
    systemctl start nginx
  EOF
  key_name     = aws_key_pair.web.id
  vpc_security_group_ids = [ aws_security_group.ssh-access.id ]
}

resource "aws_key_pair" "web" {
  <<code hidden >>
}

resource "aws_security_group" "ssh-access" {
  <<code hidden >>
}
```

▼ Advanced Details

Metadata accessible	<input checked="" type="checkbox"/> Enabled
Metadata version	V1 and V2 (token optional)
Metadata token response hop limit	1
User data	<input checked="" type="radio"/> As text <input type="radio"/> As file <input type="checkbox"/> Input is already base64 encoded

```
#!/bin/bash
sudo apt update
sudo apt install nginx -y
systemctl enable nginx
systemctl start nginx
```


TP-4: Déployez nginx et enregistrez l'ip

- A partir du code du tp-3, vous allez le modifier pour installer nginx sur votre VM
- Vous allez récupérer l'ip, id et la zone de disponibilité de la vm et vous les mettrez dans un fichier nommé `infos_ec2.txt`
- Supprimez vos ressources avec terraform destroy
- Créez un dossier tp-4 comme vous l'avez fait au tp-3 pour conserver votre code

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

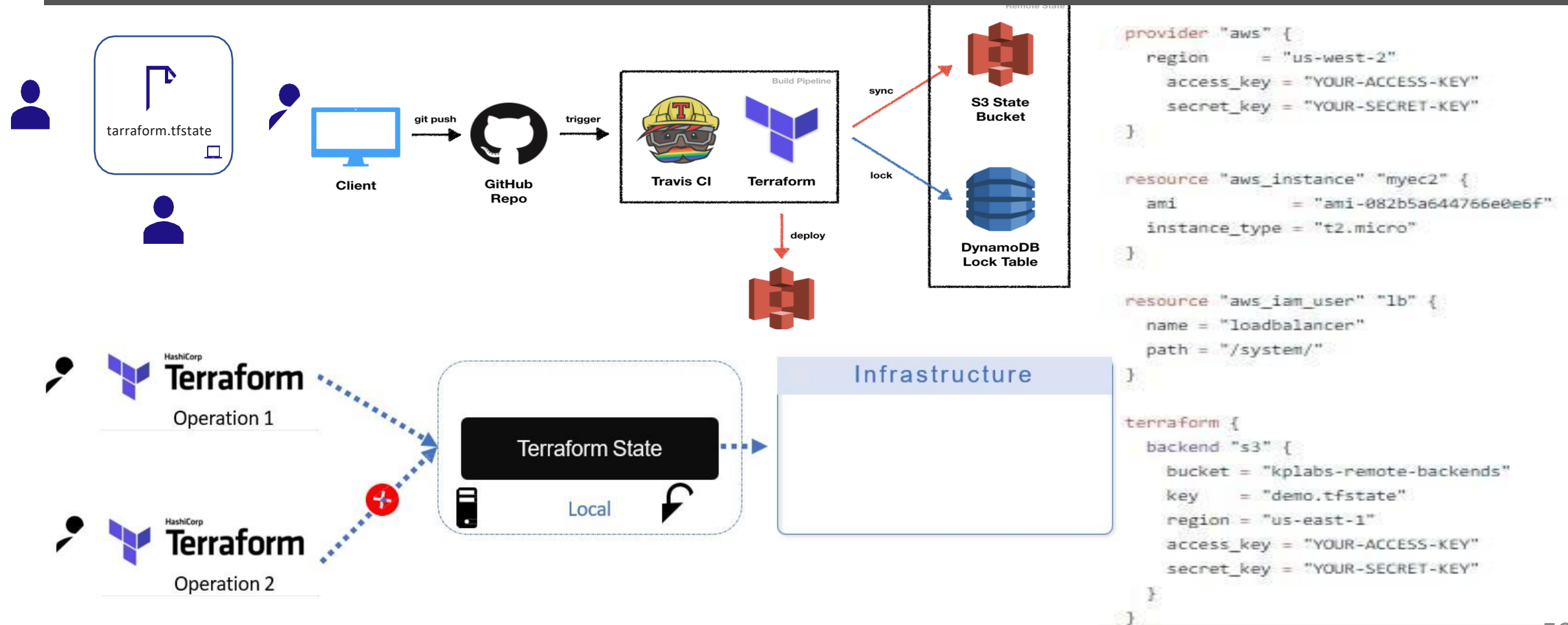
Remote management

Module

Mini-projet

TERRAFORM

REMOTE MANAGEMENT: LOCAL PROVISIONERS (Remote Backend)



REMOTE MANAGEMENT: SECURITE

ages	
Notebook	368
ript	163
	153
	145
	132
own	125
	101
	100
	71
	59

Sort: **Best ma**

```
1 slack:
2   api_token: "xoxp-????????????????????????????????????????"
```

1 xoxp- [REDACTED]

```
1 SLACK_API_TOKEN="xoxp-hogehoghog"
```

```
1 {
2   "SLACK_TOKEN": "xoxp-[REDACTED]",
3 }
```

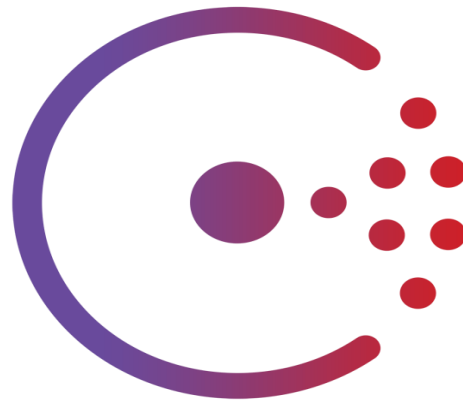
```
provider "aws" {
  region = "us-west-1"
  access_key = "YOUR-ACCESS-KEY"
  secret_key = "YOUR-SECRET-KEY"
}
```

```
resource "aws_db_instance" "default" {
    allocated_storage      = 5
    storage_type           = "gp2"
    engine                 = "mysql"
    engine_version         = "5.7"
    instance_class         = "db.t2.micro"
    name                   = "mydb"
    username                = "foo"
    password               = file("../rds_pass.txt")
    parameter_group_name   = "default.mysql5.7"
    skip_final_snapshot    = "true"
}
```

mysecretpassword505

TERRAFORM

REMOTE MANAGEMENT: SUPPORTED BACKEND



TP-5: Remote Backend

- Créez un s3 nommé terraform-backend-<votre prénom>
- Modifiez votre rendu du tp-4 afin d'y intégrer le stockage du tfstate sur votre s3
- Vérifiez après avoir lancé un déploiement que le fichier sur le drive est bien créé et contient bien les infos à jour
- Supprimez vos ressources avec terraform destroy
- Créez un dossier tp-5 comme vous l'avez fait au tp-4 pour conserver votre code

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

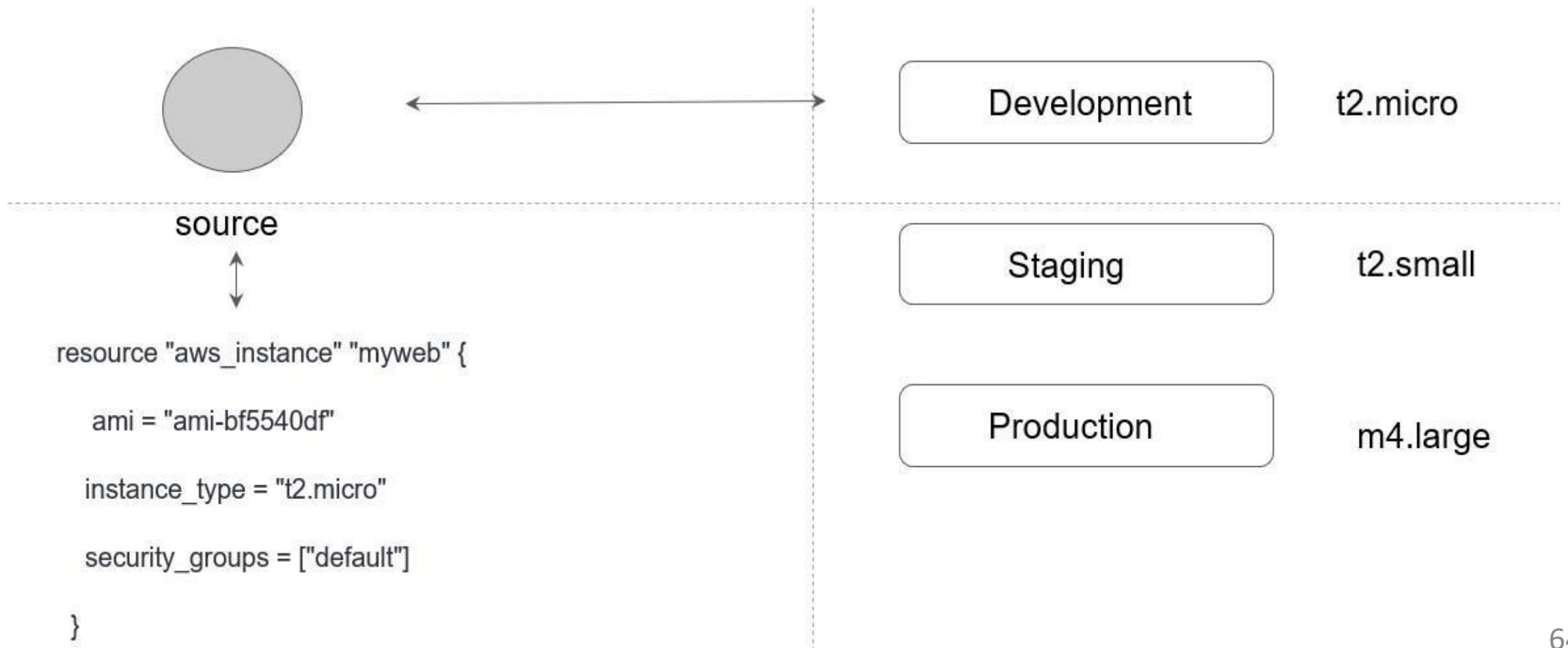
Remote management

Module

Mini-projet

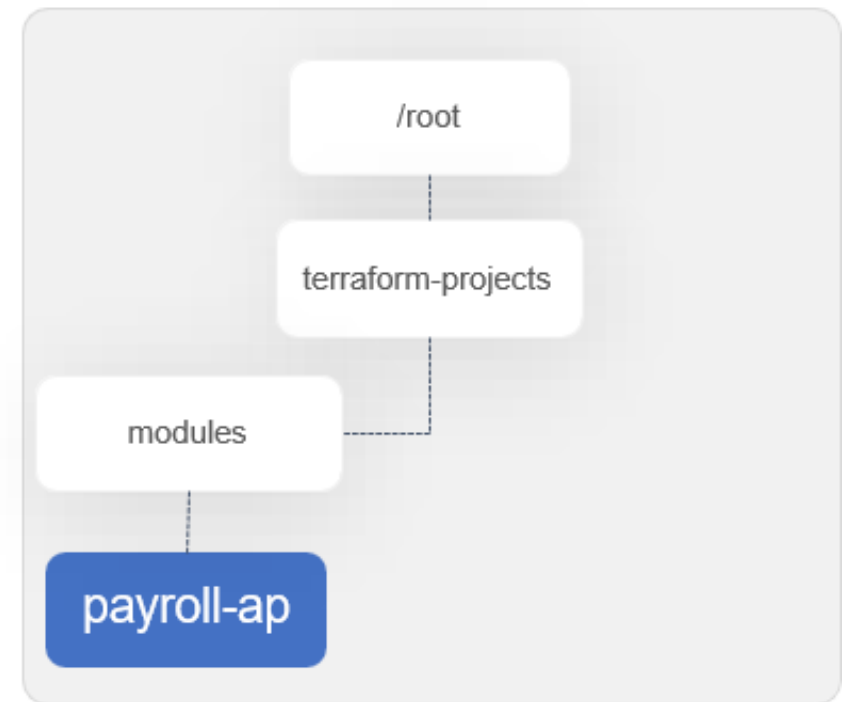
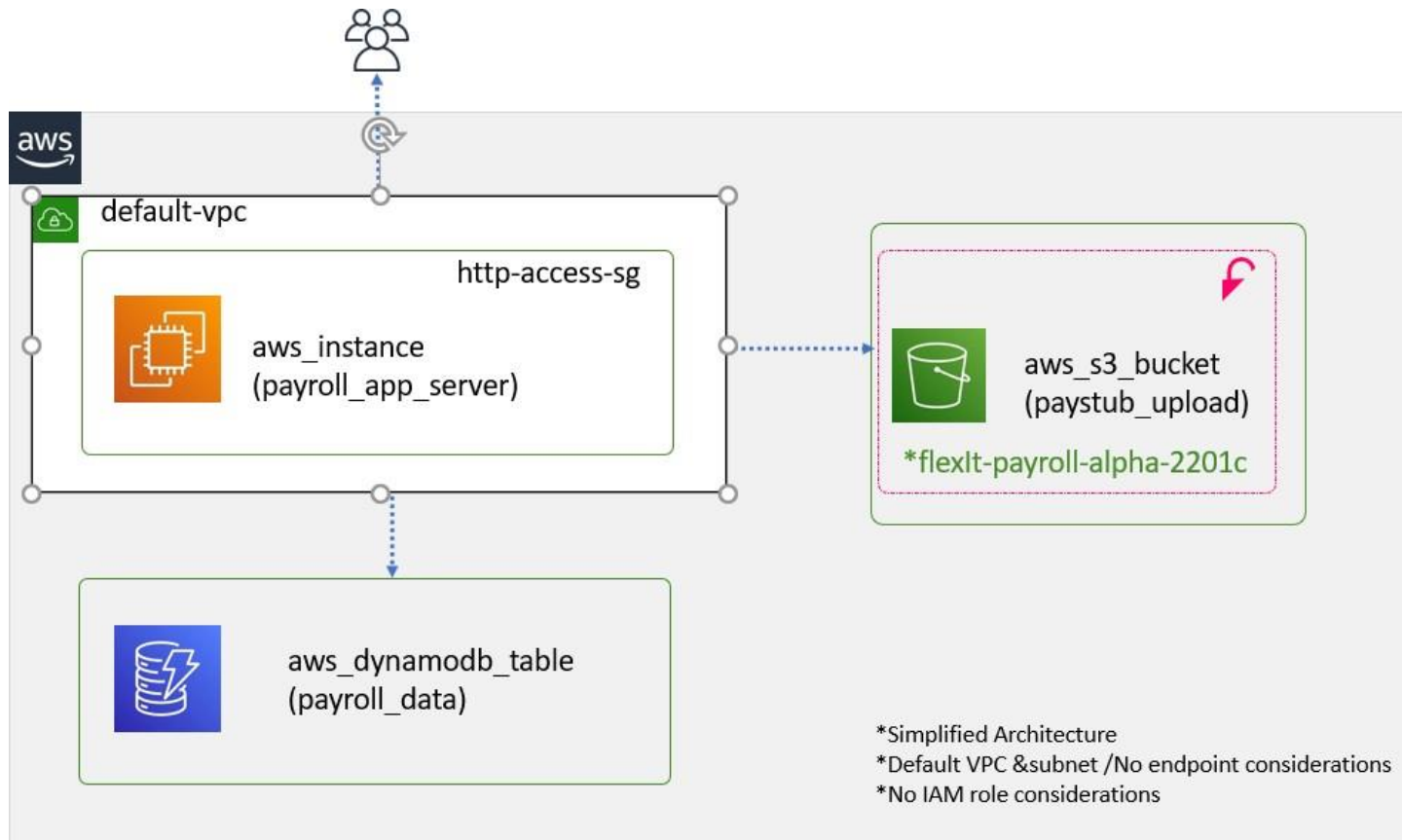
TERRAFORM

MODULE: PROBLEMATIQUE



TERRAFORM

MODULE: EXAMPLE



TERRAFORM

MODULE: EXAMPLE

```
> _
```

```
$ mkdir /root/terraform-projects/modules/payroll-app  
app_server.tf dynamodb_table.tf s3_bucket.tf variables.tf
```

s3_bucket.tf

```
resource "aws_s3_bucket" "payroll_data" {  
  bucket = "${var.app_region}-${var.bucket}"  
}
```

app_server.tf

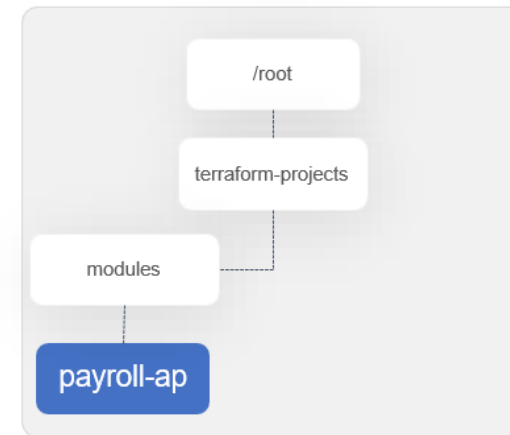
```
resource "aws_instance" "app_server" {  
  ami = var.ami  
  instance_type = "t2.medium"  
  tags = {  
    Name = "${var.app_region}-app-server"  
  }  
  depends_on = [  
    aws_dynamodb_table.payroll_db,  
    aws_s3_bucket.payroll_data  
  ]  
}
```

variables.tf

```
variable "app_region" {  
  type = string  
}  
variable "bucket" {  
  default = "flexit-payroll-alpha-22001c"  
}  
variable "ami" {  
  type = string  
}
```

dynamodb_table.tf

```
resource "aws_dynamodb_table" "payroll_db" {  
  name = "user_data"  
  billing_mode = "PAY_PER_REQUEST"  
  hash_key = "EmployeeID"  
  
  attribute {  
    name = "EmployeeID"  
    type = "N"  
  }  
}
```



TERRAFORM

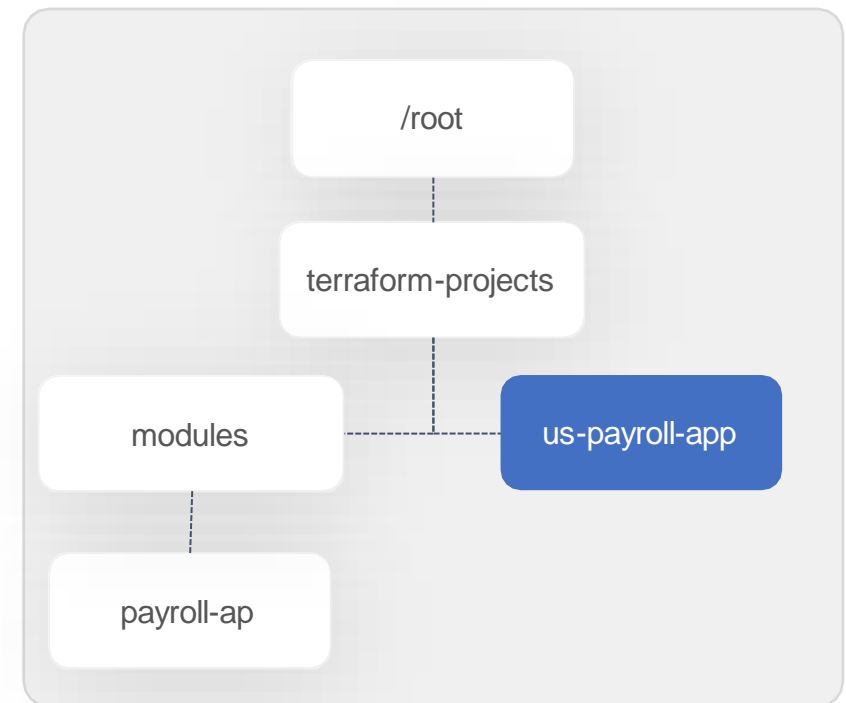
MODULE: EXAMPLE

```
>_
```

```
$ mkdir /root/terraform-projects/us-payroll-app  
main.tf provider.tf
```

main.tf

```
module "us_payroll" {  
  source = "../modules/payroll-app"  
  app_region = "us-east-1"  
  ami        = "ami-24e140119877avm"  
}
```



TERRAFORM

MODULE: EXAMPLE

```
> _
```

```
$ mkdir /root/terraform-projects/uk-payroll-app  
main.tf provider.tf
```

main.tf

```
module "uk_payroll" {  
  source = "../modules/payroll-app"  
  app_region = "eu-west-2"  
  ami       = "ami-35e140119877avm"  
}
```

provider.tf

```
provider "aws" {  
  region = "eu-west-2"  
}
```



Simpler Configuration Files

Lower Risk

Re-Usability

Standardized Configuration

MODULE: STRUCTURE

Dev

- jenkins-using-packer.tfvars
- main.tf
- variable.tf

Module

Ec2andLoadBalancer

- main.tf
- variable.tf

ECS

- main.tf
- output.tf
- variable.tf

Networking

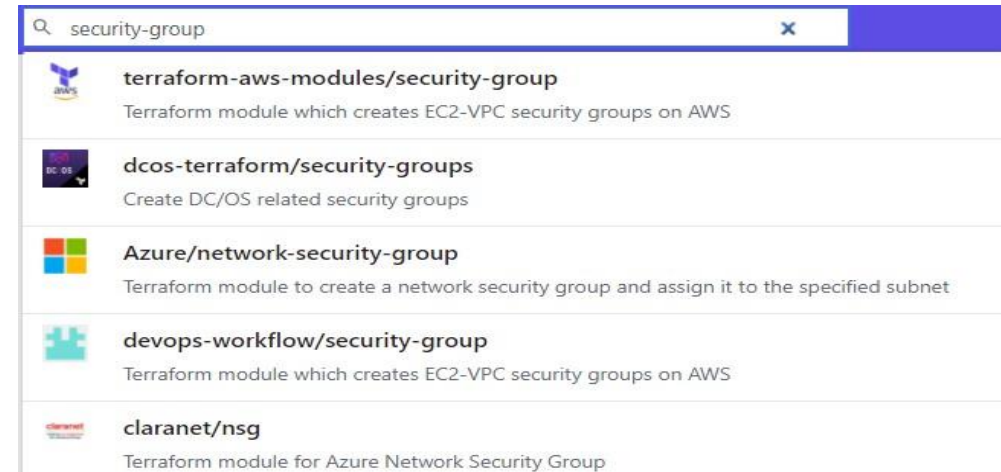
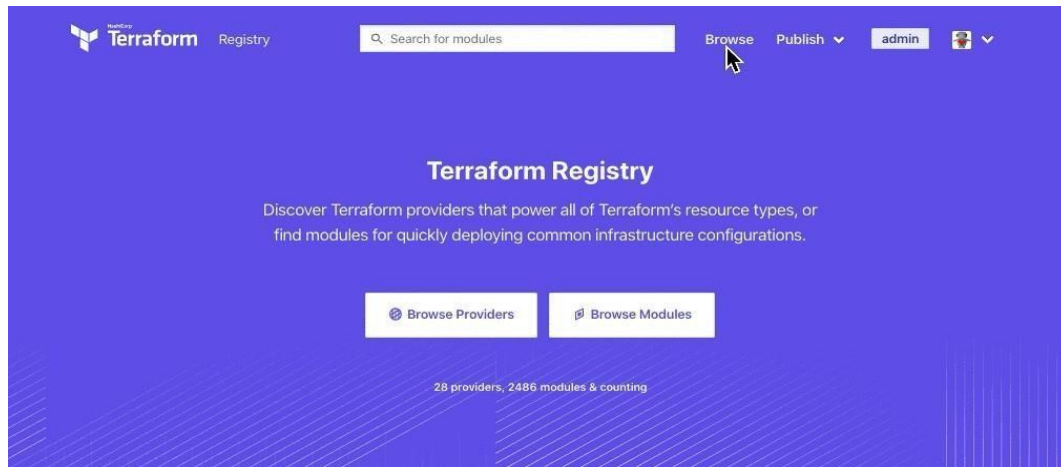
- main.tf
- output.tf
- variable.tf

SecurityGroup

- main.tf
- output.tf
- variable.tf

TERRAFORM

MODULE: REGISTRY



security-group

AWS

Terraform module which creates EC2-VPC security groups on AWS

Published August 20, 2020 by terraform-aws-modules

Module managed by antonbabenko

Total provisions: 5.4M

Source Code: github.com/terraform-aws-modules/terraform-aws-security-group (report an issue)

Submodules

Examples

Version 3.16.0 (latest)

Provision Instructions

Copy and paste into your Terraform configuration, insert the variables, and run terraform init :

```
module "security-group" {  
  source = "terraform-aws-modules/security-group,  
  version = "3.16.0"  
  # insert the 2 required variables here  
}
```

TERRAFORM

MODULE: EXAMPLE

main.tf

```
module "security-group_ssh" {  
  source = "terraform-aws-modules/security-group/aws/modules/ssh"  
  version = "3.16.0"  
  #insert the 2 required variables here  
  vpc_id = "vpc-7d8d215"  
  ingress_cidr_blocks = [ "10.10.0.0/16"]  
  name = "ssh-access"  
}
```

Provision Instructions

Copy and paste into your Terraform configuration, insert the variables, and run terraform init :

```
module "security-group" {  
  source = "terraform-aws-modules/security-group/  
  version = "3.16.0"  
  # insert the 2 required variables here  
}
```

> _

\$ terraform get

Downloading terraform-aws-modules/security-group/aws 3.16.0 for security-group_ssh...
- security-group_ssh in .terraform\modules\security-group_ssh\modules\ssh

TP-6: Module ec2

- Créez un module `ec2module` afin de déployer l'instance de la façon que vous l'avez fait aux tps précédents (`ec2` + `security group` + `ip publique`)
- Créez ensuite deux dossiers, `prod` et `dev`, chacun avec un `terraform (main.tf)` utilisant le module `ec2module` créé pour déployer une instance avec respectivement pour taille `t2.micro` pour la `prod` et `t2.nano` pour la `dev`
- Veuillez également à surcharger le `tag` pour qu'il ait cette forme : « `Name: ec2-prod-<votre prenom>` » pour la `prod` et « `Name: ec2-dev-<votre prenom>` » pour la `Dev`
- Lancez ensuite la création de votre `ec2` de `prod` et de `dev`
- Vérifiez que les `ec2` portent bien le bon nom (`Tag`) et ont la bonne taille correspondant à l'environnement
- Supprimez vos ressources avec `terraform destroy`
- Créez un dossier `tp-6` comme vous l'avez fait au `tp-5` pour conserver votre code

Plan

Présentation du formateur

Introduction au DevOps et IaC

Terraform

Déployez vos premières ressources

Rendez vos déploiements dynamiques

Terraform Provisioners

Remote management

Module

Mini-projet

Mini-projet: Déployez une infra complète

- Ecrivez un module pour créer une instance ec2 utilisant la dernière version de ubuntu bionic (qui s'at acher al'ebs et l'ip publique) dont la taille et le tag seront variabilisés
- Ecrivez un module pour créer un volume ebs dont la taille sera variabilisée
- Ecrivez un module pour une ip publique (qui s'at acher alasecurity group)
- Ecrivez un module pour créer une security qui ouvrira le 80 et 443
- Créez un dossier app qui va utiliser les 4 modules pour déployer une ec2, bien-sûr vous allez surcharger les variables afin de rendre votre application plus dynamique
- A la fin du déploiement, installez nginx et enregistrez l'ip publique dans un fichier nommé ip_ec2.txt (ces éléments sont à intégrer dans le module ec2)
- A la fin de votre travail, poussez votre rôle sur github, rédigez un rapport détaillé de vos travaux et envoyez nous ce rapport à easytrainingfr@gmail.com et nous vous dirons si votre solution respecte les bonnes pratiques



**MERCI POUR VOTRE ATTENTION ET À
TRÈS BIENTÔT SUR EAZY TRAINING**