

Terraform

Christophe DUFOUR



HashiCorp

Terraform

Prérequis

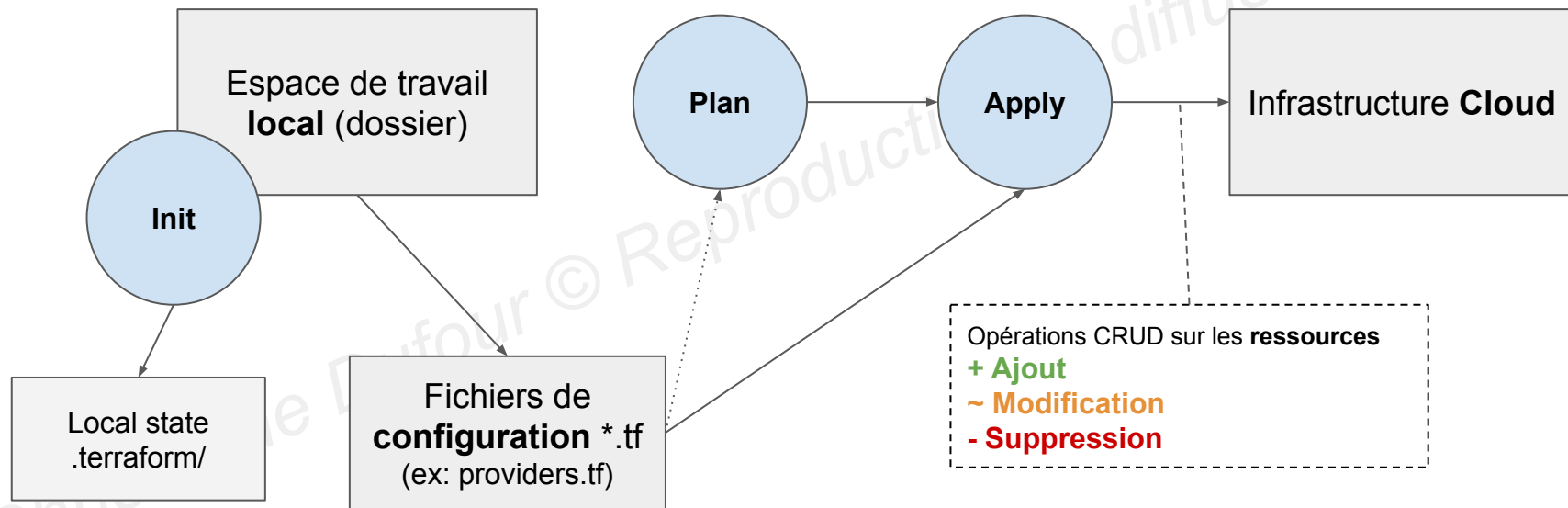
- Expérience en infrastructure (réseau, serveurs et base de données)
- A l'aise avec les lignes de commande (Windows or *nix)
- Compréhension élémentaire d'AWS est un plus
- Compte AWS
- AWS CLI avec configuration des identifiants
- Git
- Shell (bash, PowerShell, cmd, git-bash)
- Éditeur de texte (Visual Studio Code)

Terraform, c'est quoi ?



- Un outil de gestion et d'automatisation d'infrastructure
- IaC : Infrastructure As Code
- Configurations répétables et réutilisables
- Utilisation des meilleures pratiques de développement logiciel, telles que le code versioning ou les tests et déploiements automatisés
- Documentation vivante
- Itère avec sécurité sur l'infrastructure
- HCL = Hashicorp Configuration Language
- Outils similaires (en partie): Ansible, Puppet, Chef, AWS CloudFormation

Schéma général



Installation

```
https://www.terraform.io/downloads
# installation sur Debian/Ubuntu
## téléchargement de l'archive
curl https://releases.hashicorp.com/terraform/1.2.0/terraform_1.2.0_linux_386.zip -o tf.zip

## désarchivage
unzip tf.zip

## déplacement de l'exécutable dans le dossier cible
sudo mv terraform /usr/bin

## affichage de la version
terraform version => Terraform v1.2.0
```

Commandes

- terraform **init**
- terraform refresh
- terraform validate
- terraform **plan** (-destroy)
- terraform **apply** [-target *ressource*]
- terraform **destroy** [-target *ressource*]
- terraform fmt

<https://acloudguru.com/blog/engineering/the-ultimate-terraform-cheatsheet>

Le langage HCL

- HashiCorp Configuration Language
- Ressemble à JSON, avec des structures de données supplémentaires
- HCL2 = HCL + HIL (HashiCorp Interpolation Language), `${ }`
- Parseurs en Go, Java, Python, etc.
- Fonctionnalités:
 - commentaires simple ligne (`#` ou `//`) ou multi-lignes (`/* */`)
 - assignation de variable **clé = valeur**. Les espaces n'importent pas, valeur peut être une primitive (string, number, boolean, object, list)
 - chaînes entre guillemets. Peuvent contenir tout caractères utf8
 - nombres peuvent être écrits et parsés de différentes façons: Base 10 (par défaut), Hexa (préfixe `0x`), Octal (préfixe `0`), Scientifique (ex: `1e10`)
 - tableaux `[]`, listes `{ key = value }`, objets `{ key: value }` faciles à créer

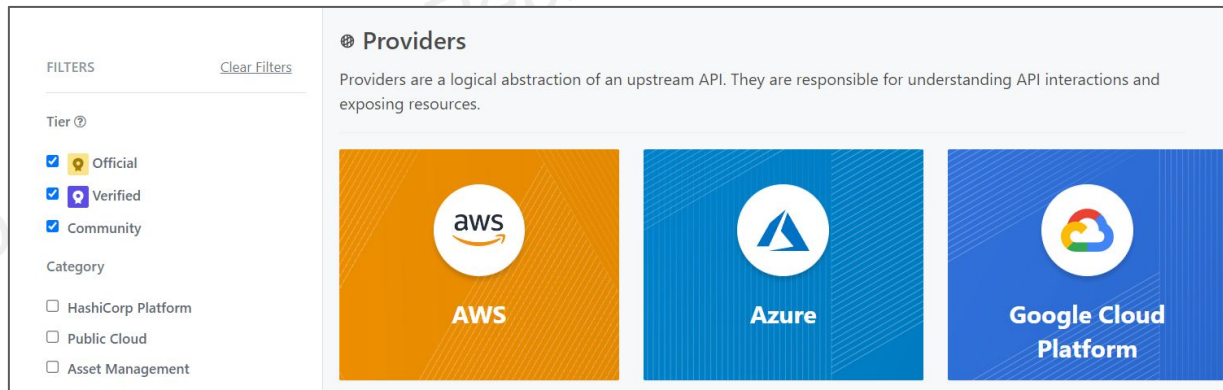
Le langage HCL - exemple

```
/*
Define the default configuration values here
*/
default_address = "127.0.0.1"
default_message = upper("Incident: ${incident}")
default_options = {
  priority: "High",
  color: "Red"
}

incident_rules {
  # Rule number 1
  rule "down_server" "infrastructure" {
    incident = 100
    options  = var.override_options ? var.override_options : var.default_options
    server   = default_address
    message  = default_message
  }
}
```


Provider


- Terraform requiert un **provider** (fournisseur) avec qui interagir via des requêtes API afin d'administrer des **ressources**




Provider



- AWS est un provider officiel. L'onglet documentation permet d'explorer l'ensemble des ressources exposées, détails et exemples.

Providers / hashicorp / aws / Version 4.19.0 ▾ Latest Version

aws  [Overview](#) [Documentation](#) [USE PROVIDER ▾](#)





aws

 Official by:  [HashiCorp](#)

Public Cloud

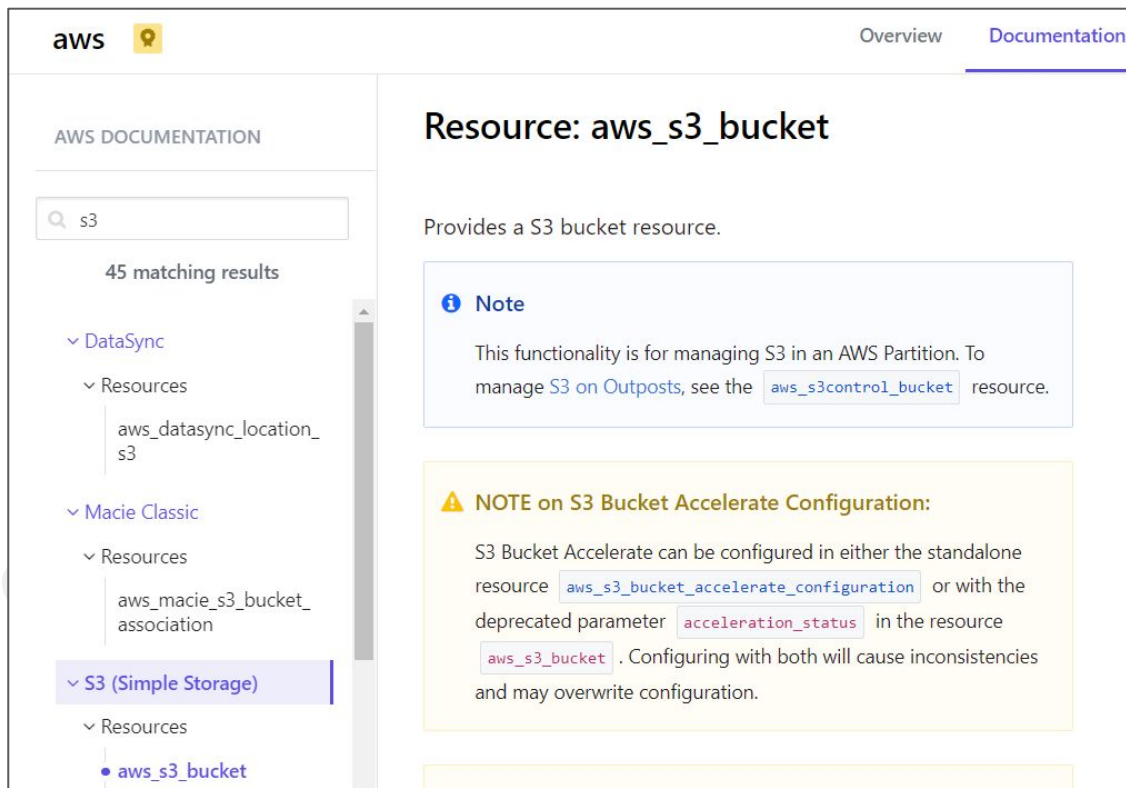
Lifecycle management of AWS resources, including EC2, Lambda, EKS, ECS, VPC, S3, RDS, DynamoDB, and more. This provider is maintained internally by the HashiCorp AWS Provider team.

VERSION **4.19.0**  PUBLISHED **3 days ago**  SOURCE CODE [hashicorp/terraform-provider-aws](#)


Provider Downloads All versions ▾

Downloads this week	16.0M
Downloads this month	41.2M
Downloads this year	296.0M
Downloads over all time	903.3M

Provider



The screenshot shows the AWS documentation page for the `aws_s3_bucket` resource. The left sidebar contains a search bar with the text "s3" and a list of 45 matching results. The results are categorized under "DataSync", "Macie Classic", and "S3 (Simple Storage)". The "S3 (Simple Storage)" category is selected, and the "aws_s3_bucket" resource is highlighted. The main content area displays the title "Resource: aws_s3_bucket" and a description: "Provides a S3 bucket resource." Below this, there is a "Note" section with an information icon, stating that the functionality is for managing S3 in an AWS Partition and directing users to the `aws_s3control_bucket` resource for managing S3 on Outposts. A yellow "NOTE on S3 Bucket Accelerate Configuration" section follows, explaining that S3 Bucket Accelerate can be configured in either the standalone resource `aws_s3_bucket_accelerate_configuration` or with the deprecated parameter `acceleration_status` in the `aws_s3_bucket` resource, and that configuring with both will cause inconsistencies and may overwrite configuration.

aws 

Overview Documentation

AWS DOCUMENTATION

Q s3

45 matching results

- ▼ DataSync
 - ▼ Resources
 - aws_datasync_location_s3
- ▼ Macie Classic
 - ▼ Resources
 - aws_macie_s3_bucket_association
- ▼ S3 (Simple Storage)
 - ▼ Resources
 - aws_s3_bucket

Resource: aws_s3_bucket

Provides a S3 bucket resource.

Note

This functionality is for managing S3 in an AWS Partition. To manage S3 on Outposts, see the `aws_s3control_bucket` resource.

NOTE on S3 Bucket Accelerate Configuration:

S3 Bucket Accelerate can be configured in either the standalone resource `aws_s3_bucket_accelerate_configuration` or with the deprecated parameter `acceleration_status` in the resource `aws_s3_bucket`. Configuring with both will cause inconsistencies and may overwrite configuration.

Installer un provider

```
terraform {  
  required_providers {  
    aws = {  
      source  = "hashicorp/aws"  
      version = "~> 3.0"  
    }  
  }  
}  
  
provider "aws" {  
  region                  = "eu-central-1"  
  shared_credentials_file = "~/.aws/credentials"  
}
```

terraform init

requête

Dépôt distant du provider
(source, ex: hashicorp/aws)

téléchargement

Copie locale
(dans le dossier projet)

Installer un provider

```
demo > main.tf
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
}
```

```
$ terraform init
```

Initializing the backend...

Initializing provider plugins...

- Finding hashicorp/aws versions matching "~> 3.0"...
- Installing hashicorp/aws v3.75.2...
- Installed hashicorp/aws v3.75.2 (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!

La commande **terraform init** installe le provider dans le dossier **.terraform**

```
demo
├── .terraform
├── .terraform.lock.hcl
└── main.tf
```

Bloc Terraform

- Bloc permettant de configurer Terraform
 - version requise de Terraform
 - providers requis (source, version, etc.)

```
terraform {  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 3.0"  
    }  
  }  
}
```

Bloc Provider

- Bloc permettant de déclarer un provider et de configurer sa connexion (ids, region, etc.)

```
provider "aws" {  
  region          = "eu-central-1"  
  shared_credentials_file = "~/.aws/credentials"  
  profile          = "default"  
}
```

```
provider "azurerm" {  
  version = "~> 1.21"  
  subscription_id = "${var.azure_sub_id}"  
  client_id = "${var.azure_client_id}"  
  client_secret = "${var.azure_client_secret}"  
  tenant_id = "${var.azure_tenant_id}"  
}
```

Bloc Resource

- “brique” de base d’une infrastructure
- Attribut = valeur
- Un fichier .tf peut définir plusieurs ressources
- Les ressources peuvent être organisées en différents fichiers .tf

type de ressource

Identifiant de ressource

```
resource "aws_s3_bucket" "bucket" {  
  bucket = "bucket-terraform-demo"  
  tags = {  
    Name = "Demo"  
  }  
}
```

```
resource "azurerm_subnet" "subnet001" {  
  name = "mySubnet"  
  virtual_network_name = "myVnet"  
  resource_group_name = "myRG"  
  address_prefix = "10.0.0.0/27"  
}
```


Bloc Output

- Bloc permettant de déclarer/exposer une valeur de sortie
- Similaire à une valeur de sortie d'une fonction en programmation
- Valeur listée dans le fichier terraform.tfstate
- Utile pour récupérer une valeur connue uniquement après exécution

Bloc Output

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

Changes to Outputs:

+ vm_ip = (known after apply)

```
resource "aws_instance" "vm" {  
  instance_type = "t2.micro"  
  ami = "ami-057fb6ca30447d5c7" # eu-central-1, focal 20.04 LTS, amd64  
}
```

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

Outputs:

vm_ip = "3.72.41.221"

```
output "vm_ip" {  
  value = aws_instance.vm.public_ip  
}
```

Bloc Output

extrait du fichier terraform.tfstate
après exécution de la commande
terraform apply

```
$ terraform output  
vm_ip = "3.72.41.221"
```

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

Changes to Outputs:

```
- vm_ip = "3.72.41.221" -> null
```

```
$ terraform output
```

Warning: No outputs found

The state file either has no outputs defined, or all the defined outputs are empty. Please define an output in your configuration with the `output` (...)

```
"outputs": {  
  "vm_ip": {  
    "value": "3.72.41.221",  
    "type": "string"  
  }  
},  
"resources": [  
  {  
    "type": "aws_instance",  
    "name": "vm",  
    "instances": [  
      {  
        "attributes": {  
          "ami": "ami-057fb6ca30447d5c7",  
          "instance_type": "t2.micro",  
          "public_ip": "3.72.41.221",
```

Bloc Module

- code terraform destiné à être réutilisé
- doit être appelé par un module racine (root module)
- seul la clé “**source**” est requise
- en option, des variables peuvent être données

```
module "buckets_momo" {  
    source = "./buckets"  
    student = "maurice-ravel"  
}  
  
module "buckets_claudio" {  
    source = "./buckets"  
    student = "claudio-debussy"  
}
```

module racine (dossier)

module **buckets** (dossier)

ensemble de fichiers
terraform

```
resource "aws_s3_bucket" "intro" {  
    bucket = "bucket-terraform-intro-${var.student}"  
}
```

Bloc Data

- permet de requérir une ressource (source de données) déjà présente chez le provider mais non contrôlée par terraform
- doit correspondre au type de ressource telle que définie par le provider
- requête par critères, filtres
- rubrique **Data Sources** de la documentation des providers
- figure dans le state ("mode": "data")
- exemple: Resource "aws_s3_bucket" pour créer un bucket, Data Source "aws_s3_bucket" pour utiliser un bucket déjà existant

Data Sources

provider: Google Cloud Platform

▼ Cloud SQL

▼ Resources

- google_sql_database
- google_sql_database_instance
- google_sql_source_representation_instance
- google_sql_ssl_cert
- google_sql_user

▼ Data Sources

- google_sql_backup_run
- google_sql_ca_certs
- google_sql_database_instance

Resources

Data Sources

provider: AWS

▼ S3 (Simple Storage)

> Resources

▼ Data Sources

- aws_canonical_user_id
- aws_s3_bucket
- aws_s3_bucket_object
- aws_s3_bucket_objects
- aws_s3_bucket_policy
- aws_s3_object
- aws_s3_objects

Data Sources

```
data "aws_availability_zones" "available" {  
  state = "available"  
}
```

demande à AWS les zone de disponibilité de la région
renseignée dans la config (eu-central-1)

terraform apply

state

```
"resources": [  
  {  
    "mode": "data",  
    "type": "aws_availability_zones",  
    "name": "available",
```

Variables

- Types de variable

- string
- number
- bool
- list (ou tuple)
- map (ou object)

```
variable "mybool" {  
  type = "bool"  
  default = true  
}
```

```
variable "host" {  
  type = string  
  default = "127.0.0.1 gitlab.test"  
}
```

```
variable "musketters" {  
  type = list(string)  
  default = ["Athos", "Porthos", "Aramis"]  
}
```

```
variable "dev1" {  
  type = map  
  default = {name = "Renato", age = 70}  
}
```


Accès aux variables

```
variable "dev1" {  
  type = map  
  default = {name = "Renato", age = 70}  
}
```

lecture de la propriété **name** du
map (object) dev1

```
output "dev1_name" {  
  value = var.dev1.name  
}
```

```
output "dev1_name" {  
  value = "Nom: ${var.dev1.name}"  
}
```

Avec interpolation `${ }`, favorise
la concaténation

Niveaux de définition des variables

1. environnement
2. fichier: terraform.tfvars
3. fichier json: terraform.tfvars.json
4. fichier *.auto.tfvars ou *.auto.tfvars.json
5. CLI: -var ou -var-file



priorité faible

priorité forte

```
export TF_VAR_str="niv1"
```

fichier **terraform.tfvars**

```
str="niv2"
```

fichier **prod.auto.tfvars**

```
str="niv4"
```

```
terraform apply -var 'str=niv5'
```

```
variable "str" {}  
output "inspect" {  
  value = "${var.str}"  
}
```

$niv1 < niv2 < niv4 < niv5$

Local State (état local)

- Principe
 - chaque action sur une ressource (création, modification, etc.) engendre un potentiel changement d'état du système (infrastructure)
 - l'ensemble des états connus de chaque ressource "contrôlée" correspond à l'état connu du système
- Fichier JSON décrivant l'état de l'infrastructure
 - fichier **terraform.tfstate**
 - créé/modifié à chaque exécution des commandes **plan/apply**
- Idempotence
 - terraform compare l'état actuel des ressources et leur état tel que décrit dans le fichier d'état. S'il détecte une différence, il **modifie** ~ ou **recrée** + (en la **supprimant** - d'abord) la ressource

terraform.tfstate

```
{
  "version": 4,
  "terraform_version": "0.15.4",
  "serial": 1,
  "lineage": "5a5fe070-d17a-423c-a5d3-ef708a669001",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_s3_bucket",
      "name": "bucket",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [...]
    }
  ]
}
```

Plan: 1 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

aws_s3_bucket.bucket: Creating...

aws_s3_bucket.bucket: Creation complete after 3s [id=terraform-...

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

terraform apply

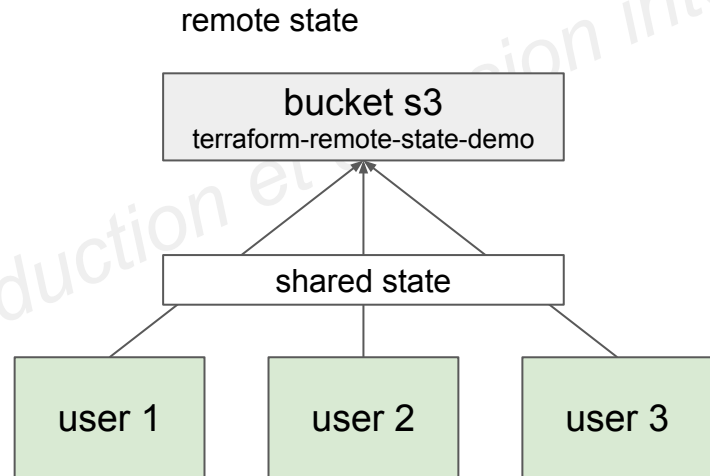
```
resource "aws_s3_bucket" "bucket" {
  bucket = "terraform-tmp-example-ressource-1"
}
```

Remote State (état distant)

- bonne pratique
- moins “fragile” qu’un état local
- permet la centralisation
- peut être encrypté (sécurité)
- un “backend” doit être choisi pour persister le state
 - ex: http, s3, consul, etc.

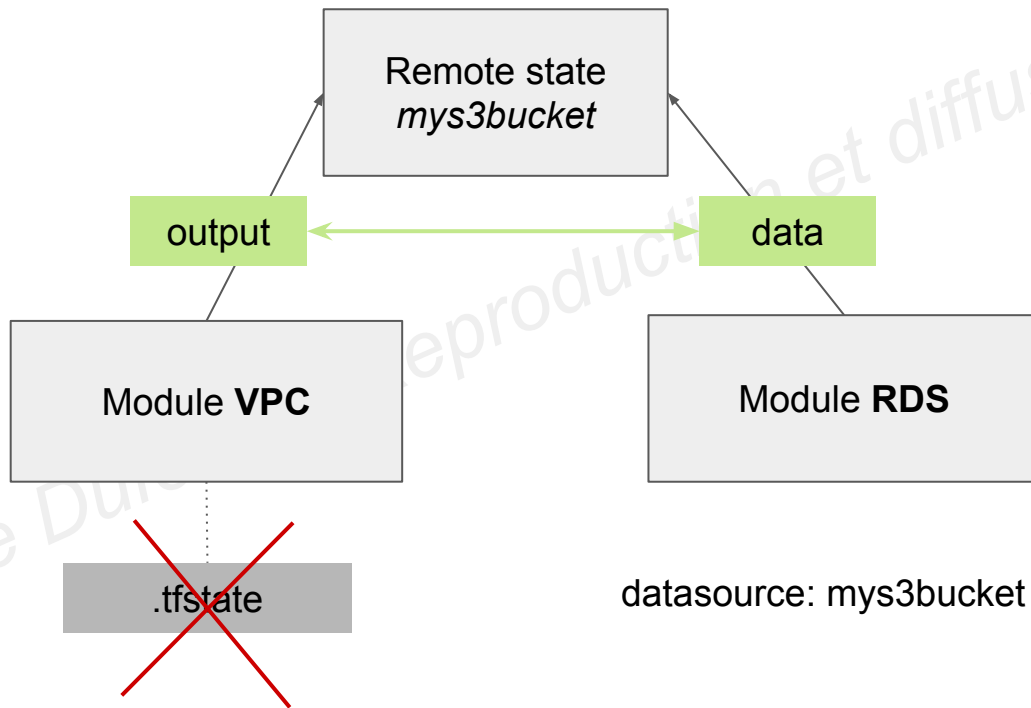
Remote State (état distant)

```
# Remote State
terraform {
  backend "s3" {
    bucket      = "terraform-remote-state-demo"
    key         = "formation/terraform.tfstate"
    region      = "eu-central-1"
    shared_credentials_file = "xxx"
    profile      = "demo"
  }
}
```



Le bloc **backend** définit **s3** comme
"hébergeur" du state

Remote State (état distant)



TP - infrastructure sur AWS

VPC

