# Terraform

# Ce este terraform ?

**Terraform** este un tool cloud agnostic si open source de IaC ( infrastructure as code ). Cu el poti crea, modifica, sterge infrastructura intr-un mod automatizat

HashiCorp
Terraform

IT School
FROM ZERO TO HERO!

# Instalare

**Pentru a instala terraform este nevoie de a urmarii pasii urmatori**

Dupa instalare putem verifica cu **terraform --version** ce versiune de terraform avem si daca instalarea a avut succes

In terraform toate fisierele trebuie sa aiba extensia **.tf .**
Ele vor fi proceaste exact ca si cum ar fi concatenate intr-un singur fisier mai mare

# Primii pasi cu terraform

Terraform foloseste un limbaj propriu si anume HashiCorp Configuration Language

```
resource "aws_instance" "example" {
  ami = "abc123"

  network_interface {
    # ...
  }
}
```

# Terraform provider

Este un plugin prin intermediul carora ne conectam la un API si executam comenzi ( creare , editare, stergere de infrastructura )

Providerii pot fi**: AWS, Azure, Kuberenetes, Oracle, Google Cloud**

Daca folositi VSCode este recomandat plugin-ul **HashiCorp Terraform**

# Conexiune cu AWS

Inainte de a rula comenzi cu providerul de aws este necesar sa avem access la un cont de aws si un set de credentiale. De asemenea, setul acela de credentiale trebuie sa aiba acces la ce resurse dorim noi sa cream

**https://registry.terraform.io/providers/hashicorp/aws/latest/docs**

# Terraform provider aws

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "12.0.0.0/16"
}
```

# Terraform init

Comanda **terraform init** inițializează un director de lucru care conține fișiere de configurare Terraform. Aceasta este prima comandă care ar trebui să fie rulată după ce ați scris o nouă configurație Terraform.

IT School®
FROM ZERO TO HERO!

# Terraform validate

Folosim terraform validate pentru a new verifica automat codul de terraform pentru eventuale erori inainte de a rula. Aceste erori sunt fix din punct de vedere al scrierii codului nu poate vedea daca este corect si din punct de vedere al providerului de Cloud.

De asemenea, cu **terraform fmt** putem aseza in pagina automat tot codul nostru

# Working Directory Contents

Dupa ce vom rula **terraform init** se vor crea niste fisiere si foldere pe care terraform le va folosi pentru a stoca anumite setari, plugin-uri , module si uneori **state data**

# Terraform plan

Comanda terraform plan creează un plan de execuție, care vă permite să previzualizați modificările pe care Terraform intenționează să le facă infrastructurii dumneavoastră.

terraform plan
terraform plan -out=plan.txt -> daca vrem sa salvam planul intr-un fisier pentru o analiza ulterioara

# Terraform apply

Comanda terraform apply execută acțiunile propuse într-un plan Terraform.

**State data**
Prin acest fisier terraform poate compara starea resurselor de aici cu cele din cloud si pe baza acestei comparatii se vor face modificari, stergeri sau se vor crea resurse

Opusul lui terraform apply este: **terraform destroy**

# Variabile

Variabilele de intrare vă permit să personalizați codul terraform fără a modifica propriul cod sursă .

```
variable "image_id" {
  type        = string
  description = "The id of the machine image (AMI) to use for the
server."
}
```

Putem de asemenea da si variabile individuale cu flag-ul “-var”
terraform apply -var="image_id=ami-abc123"

# Variabile

Terraform va utiliza toate variabilele prezente in fisierele numele terraform.tfvars, terraform.tfvars.json , .auto.tfvars , .auto.tfvars.json

# Workspaces

Terraform Cloud gestionează colecțiile de infrastructură cu workspace-uri în loc de directoare. Un workspace conține tot ce are nevoie Terraform pentru a gestiona o anumită colecție de infrastructură, iar workspace-urile separate funcționează ca un director de lucru complet separat.

Acestea pot fi foarte utile deoarece puteti avea un workspace de test, unul de dev si unul de prod.

**Comenzi utilizate:**
terraform workspace list
terraform workspace new myworkspace
terraform workspace select myworkspace

# tfenv

La o anumita perioada de timp apar update-uri pentru terraform si uneori sufera si modificari la nivel de cod. Astfel unele versiuni mai vechi de terraform e posibil sa nu mai fie compatibile cu un cod a unei versiuni recente. De accea putem folosi tool-uri precum tfenv si sa facem switch-ul rapid intre versiuni

**https://github.com/tfutils/tfenv**

# Terraform state

**Terraform state list** -> listam state-ul current de terraform

Terraform state show resource_name.resource_id **->** vedem cum arata acea resursa in state-ul terraform

# Terraform import

Cu terraform import putem importa resurse existente in aws. Fiecare resursa se poate importa in felul ei, pe fiecare pagina de documentatie a unei resurse se poate vedea in ce mod

Pentru a importa o resursa:
**terraform import aws_instance.myvm <Instance ID>**

Pentru a sterge o resursa din state
**terraform state rm aws_instance.myvm**

IT School
FROM ZERO TO HERO!

# Backend S3

Prin acest mod ne asiguram ca state-ul nostru este public intr-un bucket de aws si astfel mai multi oameni pot lucra in paralel pe acel cod de terraform

Mai multe informatii se pot gasi aici

# Terraform output

Output-ul are rolul de a afisa informatii despre infrastructura direct in cli pentru a le putea vedea mai bine, dar si ca alte configuratii de terraform sa le poate prelua si utiliza mai departe

https://spacelift.io/blog/terraform-output

**Userdata encrypted in tf**
**Pt asg**
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/autoscaling_group

# Terraform taint

Prin comanda terraform taint putem informa terraform de faptul ca o resursa de-a noastra are anumite probleme si chiar daca el nu le detecteaza, cerem distrugerea si refacerea de la zero al acelei resurse.

# depends_on

Prin depends_on puteti specifica o anumita resursa ( Resursa A )  sa depinda de alta ( Resursa B). Astel, Resursa A nu va fi creata inainte ca Resursa B din codul terraform sa fie creata complet.

Mai multe detalii se pot gasi aici

# lifecycle

Prin lifecycle putem specifica faptul ca terraform sa ignore anumite proprietati ale resursei chiar daca exita anumite diferente intre infrastructura din cloud si cea din state.

```
lifecycle {
create_before_destroy = true
 }
```

# Module de terraform

Modulele sunt similare functiilor din programare. Folosind niste template-uri cu arugmentele corespunzatoare, noi putem configura cum dorim intrastructura noastra fara sa scriem de fiecare data codul de la zero.

Exemplu putem avea modul pentru vpc si sa avem un vpc configurat cu terraform in cateva minute.

IT School
FROM ZERO TO HERO!

# Sfârșit