HASHICORP TERRAFORM

DevOps Engineer

AllOps Solutions

DRAGAN PAVLOVIC

SADRZAJ

STA JE TO TERRAFORM

Terraform je open-source alat za upravljanje infrastrukturom kao kodom. Omogucava nam da definisemo infrastrukturu za svoje aplikacije koristeci jednostavnu sintaxu HCL. Terraform za razliku od Cloudformation podrzava razne provajdere infrastrukture, kao sto su AWS, Microsoft Azure, Google Cloud, OpenStack I druge.

Definisanje strukture se izvrsava u konfiguracijskom fajlu, koji opisuje stanje sistema. Terraform onda koristi ovu konfiguraciju da bi upravljao stvarnjem, azuriranjem I brisanjem infrastrukture.

Prednost Terraforma je u tome sto omogucava ponovljivost I doslijednost infrastrukture. Mozete kreirati infrasturkturu u jednom okruzenju recimo development, I onda je lako reprodukovati infrastrukturu u drugom okruzenju kao sto je produkcija Terraform takodje omogucava pracenje promjena u konfiguraciji I automatiozavnu primjenu tih promjena na infrastrutkuru.

OSNOVNI FAJLOVI TERRAFORMA

providers.tf

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


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

main.tf

```
resource "aws_s3_bucket" "example" {
    bucket = "my-tf-test-bucket"
    tags = {
      Name = "My bucket"
      Environment = "Dev"
    }
  }

resource "aws_sns_topic" "user_updates" {
    name = "user-updates-topic"
}
```

TERRAFORM WORKSPACES

- U Terraformu, workspaces (radni prostori) su način organizovanja i upravljanja različitim instancama iste infrastrukture. Svaki workspace predstavlja odvojeno okruženje u kojem se primenjuju Terraform konfiguracije. Workspaces nam omogućava da radimo sa više kopija infrastrukture na istom kodu, a svaki workspace čuva svoje stanje (state) i konfiguraciju.
- Workspaces su korisni kada želite da imate više kopija iste infrastrukture, na primer za development, test i produkciju. Možete imati odvojen workspace za svaku fazu razvojnog ciklusa, gde možete primeniti promene i testirati ih pre nego što se primene na stvarnu produkciju.
- Kada radite sa workspaces u Terraformu, svaki workspace ima svoj zaseban direktorijum u kojem se čuva konfiguracija i stanje. Terraform koristi direktorijum **.terraform** u workspace-u da bi čuvao stanje (state) infrastrukture za svaki workspace.
- Možete koristiti komandu **terraform workspace** za rad sa workspaces-ima. Evo nekoliko osnovnih komandi koje možete koristiti:
  - **terraform workspace new <ime>**: Kreira novi workspace sa zadatim imenom.
  - **terraform workspace select <ime>**: Menja trenutni workspace na zadati.
  - **terraform workspace list**: Prikazuje listu svih dostupnih workspace-ova.
  - **terraform workspace delete <ime>**: Briše zadati workspace.
- Kada radite sa workspaces-ima, svaki workspace ima svoju kopiju infrastrukture, tako da možete primenjivati promene samo na odabranom workspace-u, a ostali workspace-ovi ostaju nepromenjeni. Ovo je korisno za izolaciju promena i testiranje novih konfiguracija pre nego što se primene na produkciju.
- Workspaces u Terraformu su korisni alati za organizaciju i upravljanje infrastrukturom u različitim okruženjima. Omogućavaju vam jednostavno upravljanje i praćenje promena na infrastrukturi za različite faze razvojnog ciklusa.

TERRAFORM STATE FAJL

Terraform state file je datoteka koju Terraform koristi za praćenje stanja vaše infrastrukture koju upravlja Terraform. Terraform je alat za upravljanje infrastrukturom kao kôdom (Infrastructure as Code - IaC) koji omogućuje automatsko upravljanje i održavanje infrastrukture u oblaku.

Terraform koristi state file kako bi pratio trenutno stanje vaše infrastrukture i razumio promene koje treba napraviti kako bi se infrastruktura dovela u željeno stanje definisano Terraform konfiguracijom. State file sadrži informacije o svim resursima koje Terraform upravlja, kao što su njihove trenutne vrednosti, ažuriranja, zavisnosti između resursa i druge metapodatke.

State file je obično u JSON ili binarnom formatu i može biti smešten lokalno na vašem računaru ili na udaljenom uređaju, kao što je sistem za skladištenje u oblaku (npr. Amazon S3). Terraform preporučuje upotrebu udaljenog skladišta kako bi se izbegle probleme sa gubitkom ili oštećenjem lokalnog state file-a.

Kada pokrenete Terraform komande, on proverava state file kako bi razumeo trenutno stanje infrastrukture i primenio promene na osnovu definicija u Terraform konfiguraciji. Terraform automatski ažurira state file kako bi odražavao nove promene u infrastrukturi.

Terraform state file je važan deo Terraform ekosistema jer omogućuje Terraformu da precizno prati i upravlja vašom infrastrukturom. Važno je voditi računa o sigurnosti i zaštiti state file-a, jer sadrži osetljive informacije o infrastrukturi i resursima koje Terraform upravlja.

PRIMJER TF.STATE

Main.tf

```
resource "aws_instance" "example" {
 ami = "ami-0fb653ca2d3203ac1"
 instance_type = "t2.micro"
}
```

Kako omoguciti kolaboraciju izmedju vise korisnika?


```
{
  "version": 4,
  "terraform_version": "1.2.3",
  "serial": 1,
  "lineage": "86545604-7463-4aa5-e9e8-a2a221de98d2",
  "outputs": {},
  "resources": [ {
    "mode": "managed",
    "type": "aws_instance",
    "name": "example",
    "provider": "provider[\"registry.terraform.io/...\"]",
    "instances": [ {
      "schema_version": 1,
      "attributes": {
       "ami": "ami-0fb653ca2d3203ac1",
       "availability_zone": "us-east-2b",
       "id": "i-0bc4bbe5b84387543",
       "instance_state": "running",
       "instance_type": "t2.micro",
       "(...)": "(truncated)"
      }
     }
    ]
   }
  ]
}
```

TERRAFORM CLOUD

- Terraform Cloud je usluga koju pruža HashiCorp, kompanija koja je razvila Terraform. Terraform Cloud je platforma za upravljanje infrastrukturom kao kodom (IaC) u oblaku. Omogućava centralizovano upravljanje Terraform konfiguracijama, stanjem infrastrukture i saradnju timova.
- Jedna od glavnih prednosti Terraform Clouda je mogućnost čuvanja stanja infrastrukture u oblaku. Umesto da se stanje infrastrukture čuva lokalno na računaru, Terraform Cloud pruža sigurno skladište za stanje infrastrukture. To je važno jer Terraform state čuva informacije o trenutnom stanju infrastrukture, tako da je neophodno da je očuvano i dostupno timu koji upravlja infrastrukturom.
    - Kreirajte nalog na Terraform Cloudu: Prvo treba da kreirate nalog na Terraform Cloudu. Možete se registrovati na njihovoj web stranici i kreirati organizaciju za vaš projekat.
    - Konfigurišite Terraform Cloud backend: U konfiguracionim fajlovima Terraforma definišite backend blok koji upućuje na Terraform Cloud. Na primer:
    - Gde **<ORGANIZATION_NAME>** predstavlja ime vaše organizacije na Terraform Cloudu, a **<WORKSPACE_NAME>** je ime radnog prostora (workspace) koji želite da koristite.
    - Pokrenite **terraform init**: Pokrenite komandu **terraform init** u lokalnom direktorijumu Terraforma. Ova komanda će inicijalizovati konfiguraciju Terraforma i postaviti backend za čuvanje stanja infrastrukture u Terraform Cloudu.
    - Konfigurišite autorizaciju za Terraform Cloud: Podesite odgovarajuće autentifikacione podatke kako biste omogućili Terraformu pristup Terraform Cloudu i čuvanje stanja infrastrukture.
    - Pokrenite **terraform apply**: Nakon što je konfiguracija inicijalizovana i backend je konfigurisan za Terraform Cloud, možete pokrenuti **terraform apply** komandu za primenu konfiguracije i ažuriranje stanja infrastrukture na Terraform Cloudu.
- Stanje infrastrukture biće automatski sinhronizovano sa Terraform Cloudom prilikom izvršavanja **terraform apply** komande. Takođe, Terraform Cloud pruža dodatne mogućnosti kao što su praćenje istorije promena, upravljanje timskim pristupom, i mogućnost izvršavanja Terraform planova i primene sa interfejsa u oblaku.

```
terraform {
  backend "remote" {
  hostname =  "app.terraform.io"
  organization = "
<ORGANIZATION_NAME>"

workspaces {
  name = "<WORKSPACE_NAME>"
  }
 }
}
```

VARIJABLE

- Varijable u Terraformu omogućavaju vam da definirate vrijednosti koje se mogu koristiti u konfiguracijskim datotekama. Varijable vam omogućavaju da parametrizirate konfiguraciju i pružite fleksibilnost prilikom izvršavanja Terraform planova i primjene promjena.
- Postoje različiti tipovi varijabli koje možete koristiti u Terraformu:
  - string (default): Ovaj tip varijable predstavlja tekstualni string. Možete definirati zadani string za varijablu ili je ostaviti praznu.
  - number: Ovaj tip varijable predstavlja numeričku vrijednost. Terraform automatski provjerava je li predana vrijednost broj i može izvršiti konverziju između stringa i broja ako je potrebno. Primjer:

```
# Primjer string varijable
variable "region" {
  type = string
  default = "us-west-2"
}

# Primjer number varijable
variable "instance_count" {
  type = number
  default = 3
}
```

VARIJABLE

- ***bool:*** Ovaj tip varijable predstavlja boolean (istinito/nistinito) vrijednost. Možete koristiti **true** ili **false** za definiranje varijable.
- ***list:*** Ovaj tip varijable predstavlja listu vrijednosti određenog tipa. Možete specificirati tip elemenata liste, kao što su **string**, **number** ili **bool**. Primjer:
- ***map:*** Ovaj tip varijable predstavlja mapu ključ-vrijednost parova. Možete specificirati tip ključeva i vrijednosti u mapi. Primjer:

- Osim ovih osnovnih tipova, Terraform također podržava i kompleksne tipove poput ***object i tuple***, koji vam omogućavaju da definirate strukturirane podatke za vaše varijable.
- Varijable možete definirati u zasebnoj datoteci (npr. ***variables.tf)*** ili kao argumente pri pokretanju Terraform naredbi. Omogućavaju vam dinamičko prilagođavanje konfiguracije prema potrebama i pružaju fleksibilnost u radu s Terraformom.

```
# Primjer bool varijable

variable "enable_monitoring" {
  type = bool
  default = true
}


# Primjer list varijable
variable "availability_zones" {
  type = list(string)
  default = ["us-west-2a", "us-west-2b", "us-west-2c"]
}


# primjer map varijable
variable "tags" {
  type = map(string)
  default = {
  Name = "my-instance"
  Environment = "production"
}
```

FUNKCIJE U TERRAFORMU

count: *Funkcija count omogućava vam definiranje broja instanci resursa koje želite stvoriti. Na primjer, možete koristiti count funkciju za stvaranje određenog broja virtualnih mašina u cloud provajderu*

concat: *Funkcija concat kombinira dvije ili više liste ili nizova u jedan. Na primjer, možete koristiti concat funkciju za spajanje dvije liste IP adresa*

```
# Primjer za count funkciju
resource "aws_instance" "example" {
  count = 3
  # Ostatak konfiguracije za AWS instancu
}
```

```
# Primjer za concat funkciju
variable "ip_addresses" {
  type = list(string)
  default = ["192.168.1.10", "192.168.1.20"]
}

locals {
  all_ips = concat(var.ip_addresses, ["192.168.1.30"])
}

output "all_ips" {
  value = local.all_ips
}
```

FUNKCIJE U TERRAFORMU

join: *Funkcija join spaja elemente niza u jedan string koristeći određeni separator. Na primjer, možete koristiti join funkciju za stvaranje formata za konfiguracijske skripte*

element: *Funkcija element vraća element niza na određenom indeksu. Može se koristiti za pristup određenom elementu u listi ili nizu. Na primjer, možete koristiti element funkciju za dobivanje vrijednosti iz liste IP adresa na određenom indeksu*

```
# Primjer  join funkcije
variable "users" {
  type = list(string)
  default = ["alice", "bob", "charlie"]
}
locals {
  formatted_users = join(",", var.users)
}
output "formatted_users" {
  value = local.formatted_users
}
```

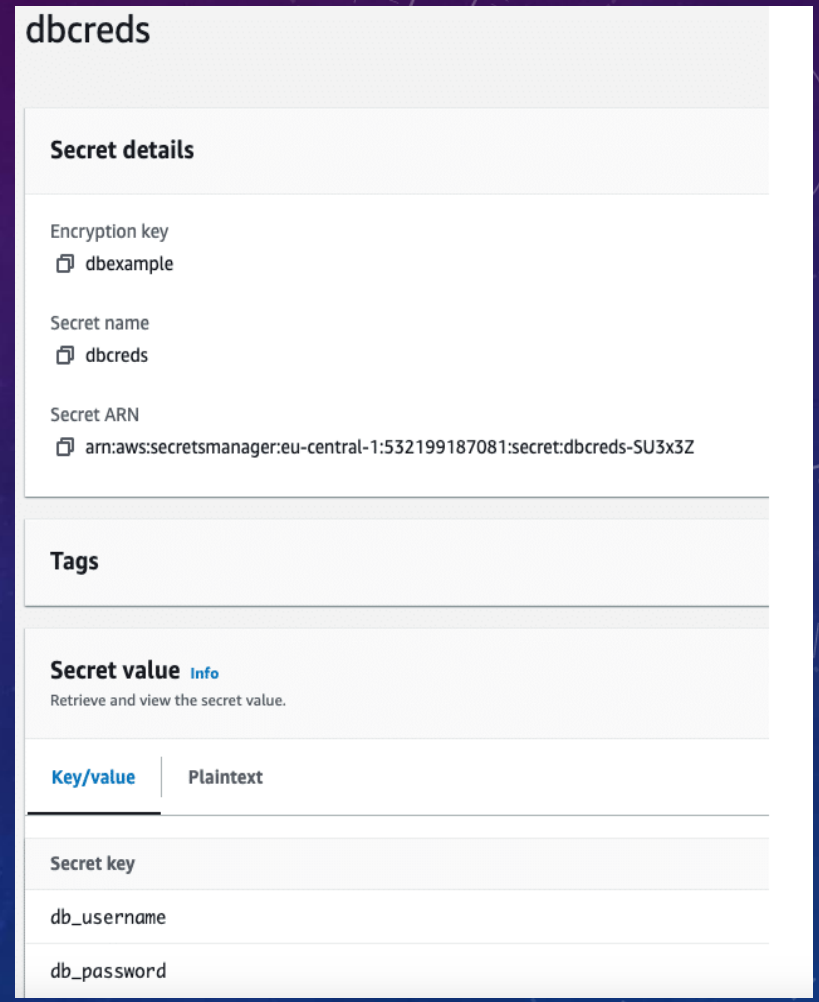
```
# Primjer  element funkcije
variable "ip_addresses" {
  type = list(string)
  default = ["192.168.1.10", "192.168.1.20", "192.168.1.30"]
}
locals {
  first_ip = element(var.ip_addresses, 0)
}
output "first_ip" {
  value = local.first_ip
}
```

DATA BLOCK

```
data "aws_secretsmanager_secret" "dbcreds" {
  name = "dbcreds"
}

data "aws_secretsmanager_secret_version" "secret_credentials" {
  secret_id = data.aws_secretsmanager_secret.dbcreds.id
}

# Create an AWS DB instance resource that requires secrets
resource "aws_db_instance" "mydb" {
  allocated_storage    = 10
  db_name              = "mydb"
  engine               = "mysql"
  engine_version       = "5.7"
  instance_class       = "db.t3.micro"
  username             = jsondecode(data.aws_secretsmanager_secret_version.secret_credentials.secret_string). ["db_username"]
  password             = jsondecode(data.aws_secretsmanager_secret_version.secret_credentials.secret_string)["db_password"]
```



EXPLICIT VS IMPLICIT DEPENDENCY

Implicit

```
resource "aws_instance"  "example_a" {
  ami = data.aws_ami.amazon_linux.id
  instance_type  = "t2.micro"
}
resource "aws_instance"  "example_b" {
  ami = data.aws_ami.amazon_linux.id
  instance_type  = "t2.micro"
}
resource "aws_eip"  "ip" {
  vpc = true
  instance = aws_instance.example_a.id
}
```

Explicit

```
resource "aws_s3_bucket" "example" { }
resource "aws_instance" "example_c" {
  ami = data.aws_ami.amazon_linux.id
  instance_type = "t2.micro"
  depends_on = [aws_s3_bucket.example]
}
module "example_sqs_queue" {
  source = "terraform-aws-modules/sqs/aws"
  version = "3.3.0"
  depends_on = [aws_s3_bucket.example,
aws_instance.example_c]
}
```

MODULI

- Moduli u Terraformu su ponovno iskoristive komponente koje se koriste za definiciju infrastrukture i resursa u cloudu.
- Moduli u Terraformu omogućuju organiziranje i ponovno korištenje konfiguracija infrastrukture. Oni omogućuju razdvajanje logike i funkcionalnosti infrastrukture na manje dijelove koji se mogu ponovno koristiti u različitim projektima ili okruženjima.
- Moduli u Terraformu mogu se koristiti za definiranje različitih vrsta resursa, poput virtualnih strojeva, mrežnih postavki, usluga baze podataka itd. Svaki modul ima svoje inpute("input variables") koja omogućuju fleksibilnost konfiguracije I outpute (eng. "output values") koji se mogu koristiti u drugim dijelovima infrastrukture.
- Prednosti korištenja modula u Terraformu uključuju:
  - Ponovno iskoristivost: Moduli omogućuju definiranje konfiguracija infrastrukture koje se mogu ponovno koristiti u različitim projektima ili okruženjima, što smanjuje potrebu za pisanjem istog koda iznova.
  - Modularnost: Moduli omogućuju organiziranje infrastrukture u manje, samodostatne dijelove, što olakšava upravljanje i održavanje kompleksnih konfiguracija.
  - Čitljivost i skalabilnost: Korištenje modula može poboljšati čitljivost koda i omogućiti skalabilnost infrastrukture putem konfiguracija koje se mogu lako proširiti i prilagoditi.
  - Timski rad: Moduli olakšavaju timski rad jer omogućuju razdvajanje odgovornosti i suradnju na konkretnim dijelovima infrastrukture.
  - Ukratko, moduli u Terraformu su modularne komponente koje olakšavaju definiranje, upravljanje i ponovno korištenje konfiguracija infrastrukture u oblaku.

TERRAFORM IMMPORT

- Komanda ***`terraform import`*** u alatu Terraform koristi se za unošenje postojećih resursa u stanje infrastrukture Terraforma.
- 1. Definišite Terraform konfiguraciju: Pre nego što možete koristiti `terraform import`, morate imati definisanu Terraform konfiguraciju u direktorijumu. To uključuje datoteke `.tf` koje opisuju resurse koje želite upravljati.
- 2. Identifikujte resurs za unošenje: Identifikujte postojeći resurs u vašoj infrastrukturi koji želite da unesete u Terraform. Ovo uključuje određivanje tipa resursa i identifikatora resursa u okviru odgovarajuće infrastrukture.
- 3. Izvršite komandu `terraform import`: Pokrenite komandu `terraform import` u terminalu ili komandnoj liniji, sa sledećom sintaksom:
- terraform import <RESOURCE_TYPE>.<RESOURCE_NAME> <RESOURCE_ID>
  - - `<RESOURCE_TYPE>` je tip resursa koji želite da unesete (na primer, `aws_instance`, `google_compute_instance`, `azurerm_virtual_machine`, itd.)
  - - `<RESOURCE_NAME>` je ime resursa u konfiguraciji Terraforma
  - - `<RESOURCE_ID>` je identifikator resursa u okviru odgovarajuće infrastrukture
- Na primer, za import AWS EC2 instance resursa, komanda bi mogla izgledati ovako:
- ***terraform import aws_instance.my_instance i-1234567890abcdef0***
- 4. Proverite stanje infrastrukture: Nakon izvršavanja `terraform import`, Terraform će uneti informacije o resursu u stanje infrastrukture. Možete izvršiti `terraform show` komandu da biste videli ažurirano stanje infrastrukture i potvrdili da je resurs uspešno unesen.
- Važno je napomenuti da `terraform import` samo unosi informacije o resursu u stanje infrastrukture, ali ne kreira odgovarajući konfiguracijski kod za taj resurs. Nakon unošenja resursa, trebali biste ručno prilagoditi konfiguracione fajlove kako biste opisali željeno stanje resursa i omogućili Terraformu da upravlja njima.

CLOUDFORMATION VS TERRAFORM

- *Deklarativni vs. imperativni jezik: CloudFormation je alat koji koristi deklarativni jezik. Definišete željeno stanje infrastrukture i CloudFormation upravlja procesom kreiranja i konfigurisanja resursa da bi se postiglo to stanje. Terraform takodjer kao I CloudFormation koristi deklerativni pristup*
- *Više platformi vs. više cloud provajdera: CloudFormation je specifičan za Amazon Web Services (AWS) i pruža podršku samo za AWS resurse. Terraform je platforma-agnostičan i može se koristiti za upravljanje resursima na različitim cloud provajderima kao što su AWS, Azure, Google Cloud, ali i za lokalne resurse, kao i za druge platforme i alate.*
- *Resursi i ekstenzija: CloudFormation ima bogat set resursa i usluga koji su dostupni direktno iz AWS ekosistema. Terraform takođe ima podršku za veliki broj cloud resursa, ali može i da se proširi putem modula i provajdera koje su zajednica razvila za podršku drugih usluga i platformi.*
- *Držanje stanja: CloudFormation automatski prati stanje vaše infrastrukture u AWS-ovom sistemu i obezbeđuje doslednost između trenutnog i željenog stanja. Terraform koristi lokalni fajl stanja (state file) koji prati trenutno stanje infrastrukture. Taj fajl stanja može biti skladišten na lokalnom računaru ili u udaljenom skladištu, poput AWS S3. Fajl stanja obezbeđuje Terraform da održava stanje infrastrukture i pravilno ažurira resurse.*
- *Ekosistem i zajednica: CloudFormation ima bogat ekosistem AWS CloudFormation predložaka koji vam omogućavaju da delite i ponovo koristite predloške infrastrukture. Terraform takođe ima bogatu zajednicu koja deli module, provajdere i resurse koji su kompatibilni sa Terraformom. Terraform ima svoj registar modula (Terraform Registry) koji omogućava brzo pronalaženje i korišćenje modula iz zajednice.*
- *Složenost i fleksibilnost: Terraform pruža veću fleksibilnost u izgradnji kompleksnih infrastrukturnih konfiguracija. Omogućava bolju kontrolu nad redosledom kreiranja i ažuriranja resursa, interakciju sa spoljnim API-ima i više naprednih mogućnosti. CloudFormation je lakši za upotrebu i može biti pogodan za jednostavnije scenarije, ali može biti manje fleksibilan u nekim situacijama.*
- *Odluka između CloudFormationa i Terraforma zavisi od vaših potreba, preferencija, ekosistema i okruženja u kojem radite. Takođe, možete koristiti i kombinaciju oba alata u zavisnosti od specifičnih zahteva projekta.*

CERTIFIKACIJA

- ***HashiCorp Certified: Terraform Associate (003)***
- ***Ispit se sastoji od 57 pitanja***
  - ***70% je potrebno za prolaz***
- ***Vrijeme za polaganje certifikata je 60 minuta***
- ***Pitanja su koncepirana na nacin da imate ponudjene odgovore***

- ***NOTE: HashiCorp posjeduje dosta dodatnih certifikata***

DRAGAN PAVLOVIC

PIPELINE I DEMO