Terraform este un utilitar de infrastructură ca cod, care vă permite să configurați și să gestionați infrastructura IT în cloud sau în afara cloud-ului. Acesta folosește un limbaj declarativ pentru a descrie infrastructura dorită și apoi utilizează acest limbaj pentru a crea și actualiza infrastructura în mod automat.

Terraform poate fi utilizat pentru a configura și gestiona resurse precum servere, baze de date, rețele, stocare și altele, în mai multe platforme de cloud, cum ar fi Amazon Web Services (AWS), Microsoft Azure sau Google Cloud Platform (GCP).

Terraform poate fi utilizat pentru a automatiza procesele de configurare și gestionare a infrastructurii, cum ar fi crearea de servere sau setarea de reglementări de securitate. De asemenea, poate fi utilizat pentru a crea și a distribui infrastructura în mai multe medii, cum ar fi dezvoltare, testare și producție.

Terraform este un utilitar puternic care poate face ca procesul de gestionare a infrastructurii să fie mai eficient și mai fiabil, oferind posibilitatea de a automatiza procesele și de a face ușor de partajat și de gestionat configurațiile infrastructurii.

I. Prima chestie --- Autoportret (Autobronzant)

-----------

1. main.tf – ceea ce vrem sa se creeze

2. variables.tf – definim variabilele pe care le vom folosi in main

3. output.tf – afisam ARN uri sau ID uri, coduri de identificare

INTREBARE Interviu Cloud AWS – ARN u – codul de identificare al unei resurse (nu toate resursele au ARN) – ARN ul si ID ul sunt diferite

ID ul e strict un mod de identificare al resursei

ARN ul contine si anumite informatii – ce tip de data e, proiect, zona si ID

II. A doua chestie --- Comenzi de baza

-----------

1. terraform init (pentru initializarea configuratia)

2. terraform plan (definim o configuratie si facem un dry run)

3. terraform apply

4. terraform taint / untaint --- !!! (e cam deprecated, se mai poate folosi dar nu e de actualitate) -> terraform apply -replace=<nume\_resursa>

5. terraform fmt – formateaza textul iti face identarea

6. terraform workspace – in terraform putem avea mai multe environmenturi (dev, prod, test etc) – ca sa cream diferite configuratii(ex: parole diferite) intre aceste environmenturi putem folosi workspace uri -> terraform workspace create [name] unde definim variabile pe care le putem in workspace u ala afferent unui env;

7. terraform state list – putem sa vedem ce avem noi creat in terraform pe baza de fisierul pe care el il creaza automat – terraform state

8. terraform state mv

Terraform oferă o serie de comenzi pentru a gestiona infrastructura definită în fișierele Terraform. Iată câteva dintre cele mai comune comenzi:

- terraform init -- Aceasta este prima comandă pe care trebuie să o executați după ce ați clonat un depozit Terraform. Aceasta va descărca și va instala modulele necesare pentru a gestiona infrastructura.

- terraform plan -- Aceasta va genera un plan de acțiune pentru a aplica schimbările definite în fișierele Terraform. Acest plan vă va arăta care resurse vor fi create, modificate sau șterse.

- terraform apply -- Aceasta va aplica planul generat de comanda terraform plan și va crea sau modifica resursele definite în fișierele Terraform.

- terraform apply –auto-approve -– sari peste “yes” step

- terraform show -- Aceasta va afișa informații despre starea actuală a infrastructurii gestionate de Terraform.

- terraform destroy -- Aceasta va șterge toate resursele create de Terraform.

- terraform state -- Aceasta oferă comenzi pentru a gestiona starea Terraform. Exemple: terraform state list, terraform state mv, terraform state rm

- terraform output -- Aceasta va afișa valorile output-urilor definite în fișierele Terraform.

API – application programming interface – este un mod prin care un software/un tool comunica cu o aplicatie. Noi cand ne facem infrastructura in AWS putem face prin UI sau prin CLI. Cand facem prin CLI, dupa ce dam comanda la capatul celalalt pe internet e un API care proceseaza comanda si ne returneaza un rezultat.

HCL = HashiCorp Configuration Language

.terraform.lock.hcl – in mom in care tf u ruleaza el creaza un state si face un lock pe state u lui in asa fel incat daca pe acelasi tf script se va aplica simultan tf apply iti va da eroare ca nu se poate pt ca deja incearca sa aplice lucruri si trebuei sa astepti sa se termine si ulterior poti aplica sit u, altfel ii poate compromite starea lui si devine corupta. Dupa ce apply u va lua sfarsi lock u va fi released si vom putea da apply.

Terraform.tfstate e state u pe baza caruia Terraformu stie ce resurse sunt in cloud provider -> nu e recomandat sa l tinem local, ci e recomandat sa stea intr un bucket astfel incat tf u cand facem un plan/apply il ia, il modifica si il impinge inapoi in bucket, iar toata lumea cand va lucra va fi conectat la acel bucket

Daca schimbam numele resursei terraformul o distruge si o recreaza -> daca avem create fisiere pe bucket nu o sa putem face asta pentru ca initial trebuie sterse fisierele si ulterior sa stergem bucketul, se poate manual, prin scripturi sau prin terraform state mv

Terraform state mv tipresursa.numeresursaveche tipresursa.numeresursanoua

HashiCorp Terraform -> extensie pt VSC; din command pallete > settings preferences: open user settings (JSON) si dam paste la asta:

{

"workbench.colorTheme": "Tokyo Night Storm",

"files.autoSave": "afterDelay",

"git.autofetch": true,

"explorer.confirmDelete": false,

"editor.formatOnSave": true,

"window.zoomLevel": 2

}

CURSUL 2

In amazon daca nu ti creezi servere custom de DNS va crea el un server automat de DNS -> care este IP ul CIDR block ului +2; 10.42.0.0 de exemplu, server de DNS va avea IPu 10.42.0.2.

Idee de proiect de la 0 pana la 1h

https://github.com/alexandrub88/ITSchool/tree/master/Materiale/Terraform