Terraform
– Infrastructure As a Code -

U ovom članku postavićemo funkcionalnu virtuelnu Linux instancu sa veb serverom na AWS (*Amazon Web Services*) i to potpuno programskim putem, na deklarativni način alatom **Terraform**. Od rekvizita je potreban isključivo AWS nalog. Na početku mali uvod u zahteve i tehnologije.
Uspon virtuelizacije hardvera sredinom 2000-tih doveo je do novih mogućnosti uspostavljanja infrastrukture u *cloud*-u i korišćenje resursa kao usluge preko Interneta (*Infrastructure as a Service*). Kako su ove platforme rasle i postajale složenije, rasla je i složenost sistemske administracije. Kao odgovor na ovaj izazov pojavila se ideja o modelovanju infrastrukture na deklarativan način, tj. pisanjem koda.

Sadržaj

Infractructure as a Code

Infrastructure as a code (IaC) je termin koji se koristi za alate koji nam omogućavaju da upravljamo IT infrastrukturom pomoću određenih **konfiguracionih fajlova**. Oni predstavljaju specifikaciju (deskriptivni model) preko koje se može kreirati željeno okruženje.
Infrastruktura se odnosi na sve komponente IT okruženja (npr. hardver, softver, mreža, *storage*…) i ona se može koristiti kao deo nekog *cloud* servisa. Koristi od ovakvih alata su:

- **Smanjenje troškova**
- **Povećanje brzine deployment-a**
- **Smanjivanje grešaka**
- **Poboljšanje konzistentnosti infrastrukture**
- **Lakša izmena i distrubucija infrastructure**

IaaC je danas važan deo **DevOps posla** i poboljšava ceo proces upravljanja okruženjima u kojima se hostuju IT servisi tako što smanjuje broj ljudskih interakcija sa samim sistemom.

Terraform

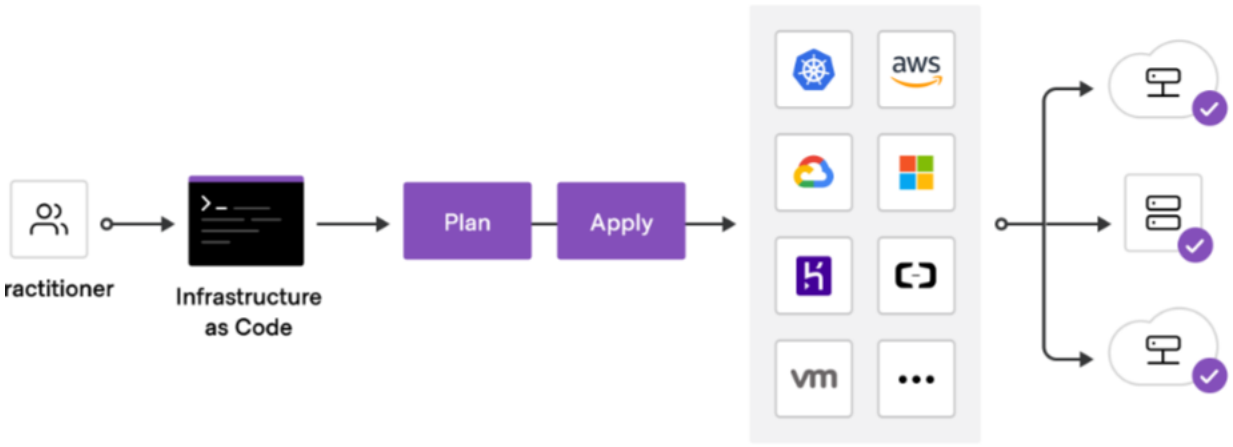
Terraform je jedan od vodećih alata za upravljanje infrastrukturom i potiče od softverske kompanije *HashiCorp*. Pomoću njega možemo koristiti jezik za konfiguraciju **HashiCorp Configuration Language (HCL)** kako bi definisali infrastrukturu u deklarativnim konfiguracionim fajlovima. Ono što Terraform olakšava u odnosu na ručno upravljanje infrastrukturom je:

1. **Upravljanje infrastrukturom na više različitih cloud platformi.**
   Za komunikaciju sa cloud platformama preko API-ja, Terraform koristi takozvane **provajdere**. Zajednica Terraform-a i HashiCorp-a podržava više od 1000 provajdera za upravljanje resursima na Amazon Web Services (AWS), Azure, Google Cloud Pratform (GCP), Kubernetes, GitHub… Oni zapravo definišu pojedinačne jedinice infrastrukture koje se kasnije mogu sastaviti u konfiguraciju pomoću **jezika HCL**.
2. **Jezik za konfiguraciju je jednostavan za učenje.**

Konfiguracioni jezik Terraform-a je deklarativni, što znaci da ga koristimo da opišemo željeno stanje infrastrukture, a ne da dajemo upustva korak po korak kao kod proceduralnog programiranja. Posao provajdera je da automatski reše zavisnosti između komponenti, kako bi ih kreirali ili uništili resurse u ispravnom redosledu.

3. **Možemo pratiti promene resursa.**
   Terraform prati našu infrastrukturu u **fajlu stanja** (*state file*). Ovaj fajl se koristi da bi mogle da se odrede optimalne izmene koje treba izvršiti na infrastrukturi tako da ona odgovara konfiguraciji.

4. **Korišćenje sistema za kontrolu verzija za razvoj konfiguracije.**
   Možemo sarađivati sa drugim kolegama na upravljanju infrastrukturom i iskoristiti sve pogodnosti koje *version control sistemi* (VSC) omogućavaju. Takođe, Terraform Cloud se može direktno povezati sa VSC-om kao što su GitHub ili GitLab, i omogućiti automatsku izmenu infrastrukture kada se promene *commit*-uju na VSC.



Instalacija

Terraform podržava mnoge operativne sistemime. Ovde ćemo koristiti najzastupljeniji Ubuntu Linux.

Najpre se moramo osigurati da je sistem ažuriran i da imamo instalirane pakete neophodne za proces instalacije – ***gnupg***, ***software-properties-common*** **i** ***curl***:

```
$ sudo apt-get update && sudo apt-get install -y gnupg software-
properties-common
```

Nakon toga instaliramo HashiCorp GPG ključ**.**
GPG je skraćenica od *Gnu Privacy Guard* i to je tehnika šifrovanja koja se koristi u brojnim aplikacijama. Funkcioniše po principu asimetričnog šifrovanja , tj. **javnog i privatnog ključa** i predstavlja neku vrstu digitalnog potpisa.

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | \
    gpg --dearmor | \
    sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

Javni ključ je dostupan svima i njegova autentičnost se može proveriti uz pomoć takozvanog otiska (*fingerprint*). Ispisujemo otisak ključa komandom `gpg`:

```
$ gpg --no-default-keyring \
     --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
     --fingerprint

/usr/share/keyrings/hashicorp-archive-keyring.gpg
-------------------------------------------------
pub   rsa4096 2020-05-07 [SC]
      E8A0 32E0 94D8 EB4E A189  D270 DA41 8C88 A321 9F7B
uid           [ unknown] HashiCorp Security (HashiCorp Package Signing)
sub   rsa4096 2020-05-07 [E]
```

Dodajemo zvaničan HashiCorp repozitorijum:

```
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
\
    https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
    sudo tee /etc/apt/sources.list.d/hashicorp.list
```

Preuzimamo HashiCorp pakete i instaliramo Terraform:

```
$ sudo apt update
$ sudo apt-get install terraform
```

Možemo proveriti da li je instalacija uspešna tako što ćemo otvoriti novi terminal i izlistati sve Terraform-ove podkomande.

```
$ terraform -help

Usage: terraform [-version] [-help]  [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.
...
```

Izgradnja infrastructure

U ovom delu biće obezbeđena jedna EC2 instanca (virtuelna mašina) na Amazon Web Services (AWS).

Za dalji rad potrebno je imati nalog za Amazon Web Services. AWS nudi ***free tier*** opciju koja je dostupna svim novim korisnicima i pomoću nje je moguće koristiti servise bez plaćanja u trajanju od 12 meseci. Svi koraci u ovom članku se mogu izvesti pod ovom licencom. Da bismo koristili IAM kredencijale za autentifikaciju, moramo definisati promenljive okruženja (*environment variable*) za privatni i javni ključ koji su povezani sa našim AWS nalogom.

```
$ export AWS_ACCESS_KEY_ID=
$ export AWS_SECRET_ACCESS_KEY=
```

Ovi kredencijali se mogu kreirati i naći u odeljku *Security credentials* na AWS web portalu. Koristimo ih za automatsku autentikaciju i autorizaciju prilikom poziva AWS-u.

Kreiranje konfiguracije

Fajl ili skup fajlova koji se koriste da opišu željenu infrastrukturu se naziva **Terraform configuration**.
Svaka terraform konfiguracija mora biti smeštena u određenom radnom direktorijumu:

```
$ mkdir learn-terraform-aws-instance
$ cd learn-terraform-aws-instance
```

Kreiramo fajl koji će definisati našu infrastrukturu:

```
$ touch main.tf
```

U *main.tf* unosimo sledeću konfiguraciju:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region  = "eu-central-1"
}

resource "aws_instance" "app_server" {
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInstance"
  }
}
```

Ovo je kompletna konfiguracija koja se može iskoristiti za izgradnju elementarne infrastrukture.

- **Terraform blok**
  Blok *terraform {}* sadrži sva podešavanja, uključujući provajdere koje će Terraform koristiti za izgradnju infrastrukture. Za svakog provajdera postoji atribut *source* koji definiše opciono ime host-a, *namespace* i tip provajdera. Atribut *version* je opcion, ali je preporučljivo definisati ga, jer može da se desi da Terraform instalira verziju provajdera koja ne radi sa našom konfiguracijom.
- **Provider blok**
  Blok *provider {}* konfiguriše navedenog provajdera. Kao što je već rečeno, provajder je dodatak (plugin) koji Terraform koristi za kreiranje i upravljanje našim resursima. Možemo koristiti više ovakvih blokova za upravljanje resursima kod različitih *cloud* provajdera. U ovom slučaju imamo *aws* provajder kome specificiramo region na *eu-central-1*.
- **Resource blok**
  Blok *resource {}* se koristi za specifikaciju komponenati infrastrukture. Resurs može biti fizička ili virtuelna komponenta kao što je EC2 instanca, a može biti i logičkog tipa. Ovi blokovi imaju dva argumenta ispred samog bloka: tip resursa i ime resursa. U ovom slučaju tip resursa je *aws_instance*, a ime je *app_server* i oni zajedno formiraju jedinstveni ID za navedeni resurs (*aws_instance.app_server*). Ovaj blok sadrži polja koja

konfigurišu resurs. U ovom primeru imamo postavljanje *AMI (Amazon Machine Instance)* ID-a koji ukazuje da se radi o Ubuntu slici (*image*) i tip instance na *t2.micro*.

Inicijalizacija direktorijuma

Za svaku konfiguraciju potrebno je inicijalizovati direktorijum komandom **`terraform init`** koja će preuzeti i instalirati sve provajdere definisane u konfiguraciji, što je u ovom slučaju *aws* provajder.

```
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 4.16"...
- Installing hashicorp/aws v4.17.0...
- Installed hashicorp/aws v4.17.0 (signed by HashiCorp)
...
```

Takođe, možemo se uveriti da je konfiguracija sintaksički validna korišćenjem komande `terraform validate`.

```
$ terraform validate

Success! The configuration is valid.
```

Kreiranje infrastrukture

Primena konfiguracije, odnosno kreiranje infrastrukture, se radi pomoću komande `terraform apply`.

```
$ terraform apply

Terraform used the selected providers to generate the following execution
plan. Resource actions
are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.app_server will be created
  + resource "aws_instance" "app_server" {
      + ami                                  = "ami-06ce824c157700cd2"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
```

```
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "ExampleAppServerInstance"
        }
      + tags_all                             = {
          + "Name" = "ExampleAppServerInstance"
        }
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)
...
```

Pre samog izvršenja, biće prikazan ceo plan primene konfiguracije koji je potrebno odobriti u komandnoj liniji.

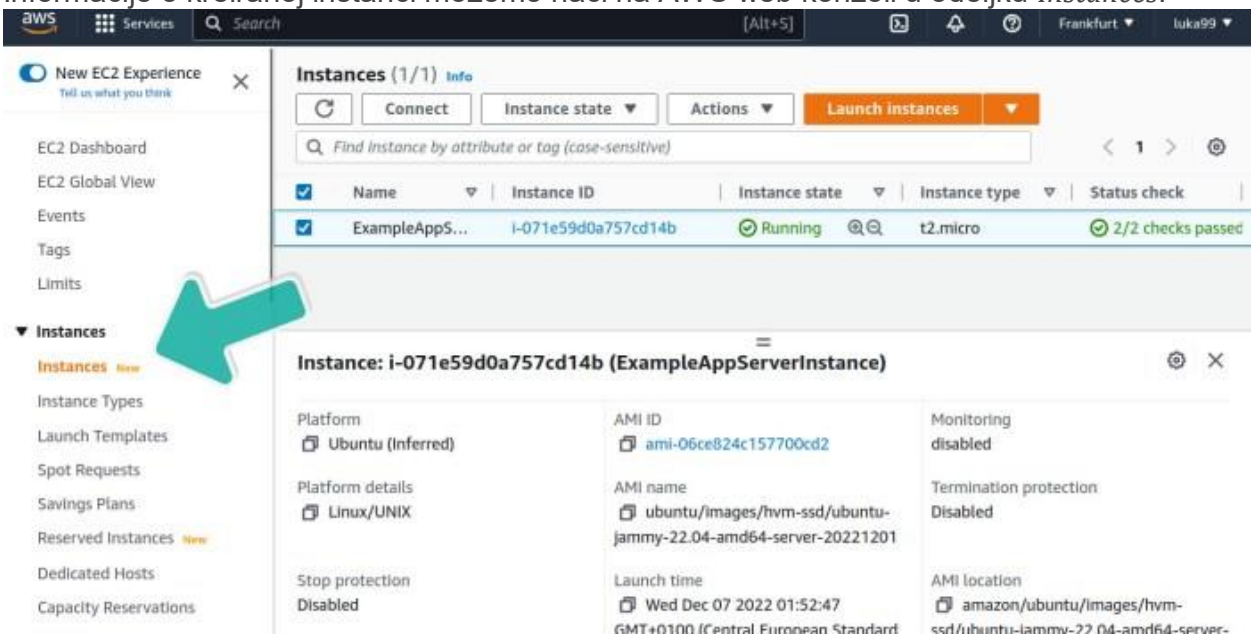
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.app_server: Creating...
aws_instance.app_server: Still creating... [10s elapsed]
aws_instance.app_server: Still creating... [20s elapsed]
aws_instance.app_server: Still creating... [30s elapsed]
```

```
aws_instance.app_server: Creation complete after 33s [id=i-
02a0e5ea9876b6881]
```

Ukoliko se komanda uspešno izvrši, naša instanca virtuelne mašine EC2 je kreirana. Informacije o kreiranoj instanci možemo naći na AWS web konzoli u odeljku *Instances*.



Pregled stanja infrastrukture

Pri primeni konfiguracije, terraform upisuje metapodatke (podatke o resursima koje koristi) u fajl koji se zove ***terraform.tfstate***. Ovaj fajl predstavlja već pomenuti **fajl stanja (state file)** i to je način na koji Terraform može da prati kojim resursima upravlja. Često sadrži osetljive informacije, pa se preporučuje da se u produkciji skladišti na nekom drugom mestu, npr. Terraform cloud-u.
Pregled trenutnog stanja se vrši komandom **`terraform show`**

```
$ terraform show

# aws_instance.app_server:
resource "aws_instance" "app_server" {
    ami                                  = "ami-06ce824c157700cd2"
    arn                                  = "arn:aws:ec2:eu-central-
1:697167208652:instance/i-07b5dc607a953252b"
    associate_public_ip_address          = true
    availability_zone                    = "eu-central-1b"
    cpu_core_count                       = 1
    cpu_threads_per_core                 = 1
    disable_api_stop                     = false
    disable_api_termination              = false
    ebs_optimized                        = false
    get_password_data                    = false
    hibernation                          = false
```

```
    id                                   = "i-07b5dc607a953252b"
    instance_initiated_shutdown_behavior = "stop"
    instance_state                       = "running"
    instance_type                        = "t2.micro"
    ipv6_address_count                   = 0
    ipv6_addresses                       = []
    monitoring                           = false
    primary_network_interface_id         = "eni-09b7363d291f9edd9"
    private_dns                          = "ip-172-31-38-77.eu-central-
1.compute.internal"
    private_ip                           = "172.31.38.77"
    public_dns                           = "ec2-18-197-115-183.eu-central-
1.compute.amazonaws.com"
    public_ip                            = "18.197.115.183"
    secondary_private_ips                = []
    security_groups                      = [
        "default",
    ]
    source_dest_check                    = true
    subnet_id                            = "subnet-007498823f13c2d01"
    tags                                 = {
        "Name" = "ExampleAppServerInstance"
    }
    tags_all                             = {
        "Name" = "ExampleAppServerInstance"
    }
```

Izmena infrastrukture

Infrastrukture se kontinualno menjaju, pa je obezbeđeno da promenom konfiguracije Terraform izgradi **plan izvršenja (execution plan)** koji menja samo ono što je neophodno da bi se dostiglo željeno stanje.
Izmena se vrši tako što promenimo fajl *main.tf* (npr. promenimo *AMI* na drugu verziju Ubuntu), nakon čega treba ponoviti komandu `terraform apply` da bi realizovali optimalni plan izvršenja koji će terraform kreirati.Nakon našeg odobrenja plana izvršenja, stanje će biti izmenjeno.

Uklanjanje infrastrukture

Komanda **`terraform destroy`** se ponaša suprotno od komande `terraform apply`, tj. ona uklanja sve resurse koji su definisani u Terraform projektu.

```
$ terraform destroy

aws_instance.app_server: Refreshing state... [id=i-07b5dc607a953252b]

Terraform used the selected providers to generate the following execution
plan. Resource actions
are indicated with the following symbols:
  - destroy
```

```
Terraform will perform the following actions:

  # aws_instance.app_server will be destroyed
  - resource "aws_instance" "app_server" {
      - ami                                  = "ami-06ce824c157700cd2" ->
null
      - arn                                  = "arn:aws:ec2:eu-central-
1:697167208652:instance/i-07b5dc607a953252b" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "eu-central-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-07b5dc607a953252b" ->
null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
...
```

Ukoliko potvrdimo plan izvršenja, infrastruktura će biti uklonjena.

```
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

aws_instance.app_server: Destroying... [id=i-07b5dc607a953252b]
aws_instance.app_server: Still destroying... [id=i-07b5dc607a953252b, 10s
elapsed]
aws_instance.app_server: Still destroying... [id=i-07b5dc607a953252b, 20s
elapsed]
aws_instance.app_server: Still destroying... [id=i-07b5dc607a953252b, 30s
elapsed]
aws_instance.app_server: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
```

Definisanje izlaza

Moguće je definisati izlazne vrednosti (*output values*) koje će se generisati nakon što se infrastruktura kreira. Na primer, izlaz koji bi nam bio od koristi u ovom slučaju je IP adresa kreirane instance. Ove vrednosti možemo koristiti da povežemo projekte sa drugim delovima infrastrukture ili drugim projektima.
Ovo možemo izvesti dodavanjem *output* bloka direktno u *main.tf* fajl ili da posebno definišemo izlaze u novom fajlu.

```
output "instance_public_ip" {
  description = "Public IP address of the EC2 instance"
  value       = aws_instance.app_server.public_ip
}
```

Pre nego što možemo koristiti izlazne vrednosti, moramo primeniti promene.

```
$ terraform apply

aws_instance.app_server: Refreshing state... [id=i-071e59d0a757cd14b]

Changes to Outputs:
  + instance_public_ip = "18.197.129.195"

You can apply this plan to save these new output values to the Terraform
state, without changing any real infrastructure.

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


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

Outputs:

instance_public_ip = "18.197.129.195"
```

Terraform štampa izlazne vrednosti u konzoli, a možemo ih i ručno pregledati komandom **`terraform output`**.

```
$ terraform output

instance_public_ip = "18.197.129.195"
```

Postavljanje web sajta

Do sada je demonstrirana izgradnja, promena i uklanjanje infrastrukture sa lokalne mašine. U produkciji se preporučuje da se stanje infrastrukture šifruje i čuva na nekom bezbednom mestu.

Dosadašnja konfiguracija je sledeća:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region  = "eu-central-1"
}

resource "aws_instance" "app_server" {
  ami           = "ami-06ce824c157700cd2"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInstance"
  }
}
output "instance_public_ip" {
  description = "Public IP address of the EC2 instance"
  value       = aws_instance.app_server.public_ip
}
```

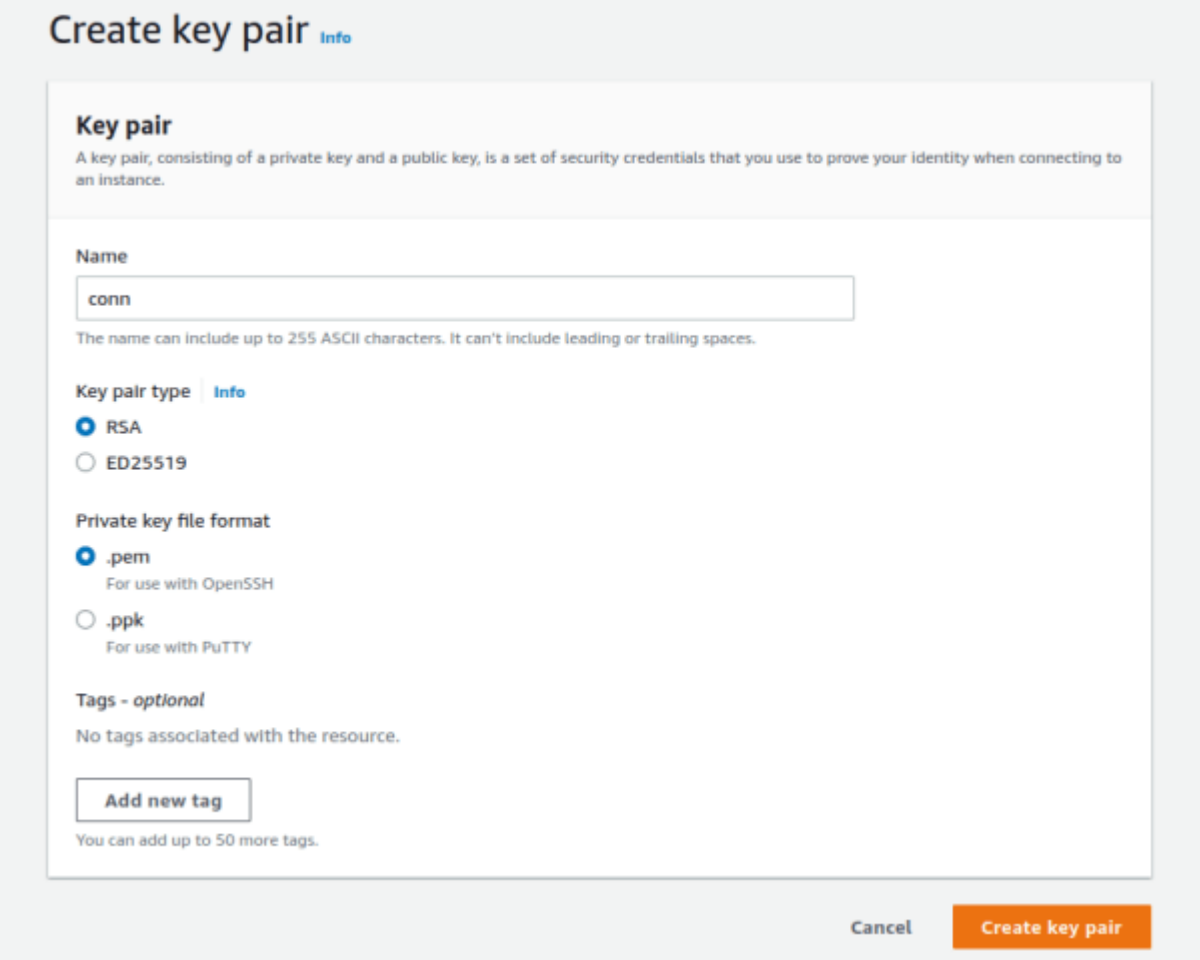
U ovom delu izmenićemo konfiguraciju tako da omogućimo automatizovanje postavljanja jednostavne veb stranice na našu EC2 instancu i pristupimo joj preko veb pretraživača.

Kreiranje ključa za server

Pre nego što možemo pristupiti instanci, moramo kreirati par ključ-vrednosti (key-value).

Kao i za AWS nalog, par ključeva (privatni i javni) predstavlja bezbednosne kredencijale koji se koriste da dokažemo svoj indentitet kada se povezujemo na EC2 instancu. Na toj instanci će se čuvati javni ključ, a mi čuvamo privatni ključ.

Kreiranje ključa se izvodi u AWS konzoli u odeljku *Key pairs* pri čemu je potrebno uneti ime ključa. Nakon kreiranja, fajl koji predstavlja privatni ključ će se automatski preuzeti na kompjuter.

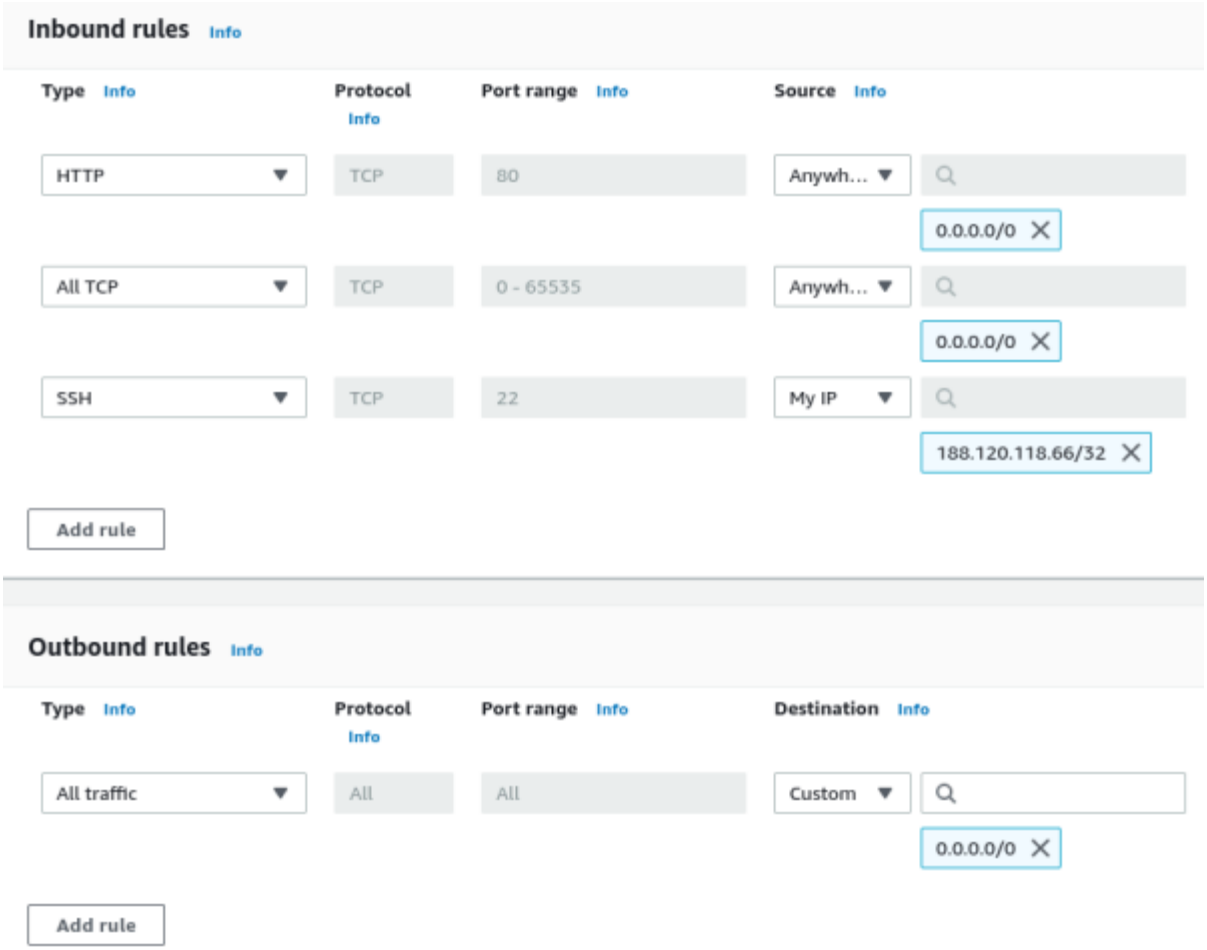


S obzirom da ćemo koristiti SSH klijent da bismo se konektovali na instancu, moramo podesiti dozvole za fajl privatnog ključa tako da ga samo mi smemo čitati.

```
$ chmod 400 conn.pem
```

Kreiranje bezbednosne grupe

Kreiranje bezbednosne grupe (security group) za našu instancu se radi u odeljku *Security groups* AWS konzole.
Pored imena i opisa, pri definisanju grupe navodimo ulazna (inbound) i izlazna (outbound) pravila. Ulazna pravila štite mrežu od dolaznog saobraćaja, npr. od nedozvoljenih konekcija, malvera i denial-of-service (DoS) napada, a izlazna pravila štite od odlaznog saobraćaja.



Od ulaznih pravila (*Type – Source*) definišemo:

- ***HTTP – Anywhere for Source***
- ***All TCP – Anywhere for Source***
- ***SSH – My IP***

Za odlazno pravilo imamo: *All traffic – 0.0.0.0/0*.

SSH povezivanje i hostovanje

Da bi hostovali sajt, potrebno je da se povežemo preko SSH i da instaliramo sve potrebne pakete korišćenjem *Terraform provisioner*-a.
Pre nego što probamo da se konektujemo na server, moramo u konfiguraciji dodati naziv kreiranog ključa i bezbednosne grupe za našu instancu.

```
resource "aws_instance" "app_server" {
  ami           = "ami-06ce824c157700cd2"
  instance_type = "t2.micro"
+  security_groups = ["LukaWebServerGroup"]
+  key_name = "conn"
  tags = {
    Name = "ExampleAppServerInstance"
  }
```

```
}
...
```

Nakon izmene infrastrukture, vezu možemo testirati preko terminala komandom **ssh -i "~/.ssh/your_KeyPair.pem" ec2-user@public_IPv4_DNS.**
Kod koji sledi dodajemo u *main.tf* konfiguraciju i koristimo da bi izvršili skriptu na našoj instanci. Skripta preuzima paket *apache2* koji od naše instance pravi HTTP server i paket *git* preko koga kloniramo repozitorijum web sajta.

```
...
resource "null_resource" "remote"{
  connection {
    type = "ssh"
    user = "ubuntu"
    private_key = file("~/.ssh/conn.pem")
    host  = aws_instance.app_server.public_ip
  }
  provisioner "remote-exec" {
    inline = [
      "sudo apt install apache2 -y",
      "sudo apt install git -y",
      "sudo git clone
https://github.com/lukamaletkovic01/webTerraformExample.git
/var/www/html/web/",
      "sudo service apache2 start"
    ]
  }
}
$ terraform apply

aws_instance.app_server: Refreshing state... [id=i-0356b951fc4456e5a]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # null_resource.remote will be created
  + resource "null_resource" "remote" {
      + id = (known after apply)
    }

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
```

```
null_resource.remote: Creating...
null_resource.remote: Provisioning with 'remote-exec'...
null_resource.remote (remote-exec): Connecting to remote host via SSH...
null_resource.remote (remote-exec):   Host: 3.72.37.24
null_resource.remote (remote-exec):   User: ubuntu
null_resource.remote (remote-exec):   Password: false
null_resource.remote (remote-exec):   Private key: true
null_resource.remote (remote-exec):   Certificate: false
null_resource.remote (remote-exec):   SSH Agent: true
null_resource.remote (remote-exec):   Checking Host Key: false
null_resource.remote (remote-exec):   Target Platform: unix
null_resource.remote (remote-exec): Connected!
null_resource.remote (remote-exec): Reading package lists... 0%
null_resource.remote (remote-exec): Reading package lists... 100%
null_resource.remote (remote-exec): Reading package lists... Done
null_resource.remote (remote-exec): Building dependency tree... 0%
null_resource.remote (remote-exec): Building dependency tree... 0%
null_resource.remote (remote-exec): Building dependency tree... 50%
null_resource.remote (remote-exec): Building dependency tree... 50%
null_resource.remote (remote-exec): Building dependency tree... Done
null_resource.remote (remote-exec): Reading state information... 0%
null_resource.remote (remote-exec): Reading state information... 0%
null_resource.remote (remote-exec): Reading state information... Done
null_resource.remote (remote-exec): The following additional packages will
be installed:
null_resource.remote (remote-exec):   apache2-bin apache2-data

...

null_resource.remote (remote-exec): git set to manually installed.
null_resource.remote (remote-exec): 0 upgraded, 0 newly installed, 0 to
remove and 0 not upgraded.
null_resource.remote (remote-exec): Cloning into '/var/www/html/web'...
null_resource.remote (remote-exec): remote: Enumerating objects: 3, done.
null_resource.remote (remote-exec): remote: Counting objects:  33% (1/3)
null_resource.remote (remote-exec): remote: Counting objects:  66% (2/3)
null_resource.remote (remote-exec): remote: Counting objects: 100% (3/3)
null_resource.remote (remote-exec): remote: Counting objects: 100% (3/3),
done.
null_resource.remote (remote-exec): remote: Compressing objects:  50%
(1/2)
null_resource.remote (remote-exec): remote: Compressing objects: 100%
(2/2)
null_resource.remote (remote-exec): remote: Compressing objects: 100%
(2/2), done.
null_resource.remote (remote-exec): remote: Total 3 (delta 0), reused 0
(delta 0), pack-reused 0
null_resource.remote (remote-exec): Receiving objects:  33% (1/3)
null_resource.remote (remote-exec): Receiving objects:  66% (2/3)
null_resource.remote (remote-exec): Receiving objects: 100% (3/3)
null_resource.remote (remote-exec): Receiving objects: 100% (3/3), done.
null_resource.remote: Creation complete after 15s [id=1445036431751750334]
```

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

Outputs:

instance_public_ip = "3.72.37.24"
```

Kada se konfiguracija izvrši, možemo testirati web server tako što u pretraživaču pristupimo dobijenoj IP adresi *http://3.72.37.24/web*:

Zaključak

Uspostavljanje infrastrukture bez IaaC automatizovanja podrazumeva dugotrajan i skup proces manuelnog upravljanja serverima, operativnim sistemima, podacima i ostalim komponentama. Ako uzmemo u obzir rastući trend računarstva u oblaku, bez IaaC bi bilo izuzetno teško upravljati obimom savremenih infrastruktura. Umesto otvaranja AWS web konzole i posebnog kreiranja ili menjanja svake komponente, možemo to uraditi preko konfiguracionog koda.

Terraform zajednica je danas najaktivnija IaaC zajednica, što ima ogroman uticaj na iskustvo sa samom tehnologijom, jer ukazuje na to koliko ljudi doprinosi projektu i koliko je jednostavno pronaći pomoć na Internetu.

Korisni linkovi

- **AWS | Terraform | HashiCorp Developer**
- **What is Infrastructure as Code (IaC)?**
- **GPG — Wikipedia**
- **Amazon EC2 key pairs and Linux instances – Amazon Elastic Compute Cloud**
- **What are Firewall Rules | Components of a Good Firewall Policy | AlgoSec**