Automatizácia konfigurácie sieťových zariadení

Obsahom tohto dokumentu je demonštrácia práce s nástrojmi Ansible, ktorý umožňuje automatizované nastavovanie a konfiguráciu cieľových zariadení (sieťové zariadenia, servery) a nástrojom Terraform, ktorý patrí do kategórie IaaC nástrojov a umožňuje automatizované vytváranie zdrojov, či už na úrovni serverovej virtualizácie, kontajnerov, no najmä cloud-ových služieb (AWS, Azure, Google Cloud, …).

Ansible

Ansible je nástroj s otvoreným kódom, ktorý umožňuje nasadzovanie softvéru, konfiguráciu a nasadzovanie aplikácií. V rámci tohto dokumentu bude ukázané, ako je tento nástroj možné používať na správu konfigurácie sieťových zariadení. Ansible funguje tak, že sa pripojí na cieľové zariadenie protokolom SSH, odošle naň ansible module, ktorý vykoná python interpreter na cieľovom zariadení (z tohto dôvodu je python povinnou súčasťou cieľového zariadenia). Inštalácia nástroja je jednoduchá a vyžaduje zadanie jedného príkazu:

```
sudo apt update
sudo apt install ansible
ansible --version
```

```
root@monit-VirtualBox:~# ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0]
root@monit-VirtualBox:~#
```

Nie je to nutnosťou, no pre lepšiu orientáciu vytvorme v domovskom adresári priečinok s názvom ansible, do ktorého budú ukladané všetky konfiguračné súbory.

```
mkdir ansible
cd ansible
```

V tomto adresári vytvoríme dvojicu súborov a to konfiguračný súbor config_csr.yml a inventory súbor s názvom inventory.net. Konfiguračný súbor bude obsahovať definíciu toho, čo sa má vykonať a inventory súbor bude obsahovať informácie identifikujúce cieľové zariadenie spolu s údajmi potrebnými pre úspešné pripojenie a autentifikáciu. V našom prípade je cieľovým zariadením virtuálny smerovač CSR1kv:

nano config_csr.yml

```
---
- name: Basic Example
  hosts: csr
  gather_facts: no
```

```
  tasks:
      - name: "Create interface"
        ios_config:
          lines:
            - "interface loopback 22"
            - "ip add 22.22.22.22 255.255.255.255"
```

Konfiguračný súbor začína troma pomlčkami, ktoré označujú začiatok YAML súboru. Ďalej je označenie názvu daného konfiguračného playbooku (má len lokálny význam). Parameter hosts definuje, na aké zariadenie sa budeme pripájať. V uvedenej konfigurácii je názov premennej uloženej v súbore inventory.net. gather_facts umožňuje prijať fakty o cieľovom sieťovom zariadení v štandardnom key:value formáte. Druhou časťou konfiguračného súboru je definovanie vykonávaných úloh (tasks). Táto sekcia môže obsahovať definovanie viacerých úloh. Každá úloha je definovaná názvom, za ktorým nasleduje modul, ktorý obsahuje nástroje pre interakciu s cieľovým zariadením. V našom prípade je použitý modul **ios_config**, ktorý obsahuje funkciu **lines**, ktorou je možné na cieľové zariadenie poslať sériu vymenovaných príkazov. V našom prípade sa jedna o príkazy potrebné pre vytvorenie loopback rozhrania a nastavenie IPv4 adresy a masky.

nano inventory.net

```
[servers]
csr ansible_host=192.168.2.5

[servers:vars]
ansible_become=yes
ansible_become_method=enable
#ansible_become_password=test
ansible_network_os=ios
ansible_connection=network_cli
ansible_user=cisco
ansible_password=cisco123!

[all:vars]
ansible_python_interpreter=/usr/bin/python
```

Inventory súbor obsahuje skupinu **servers**, ktorá v sebe obsahuje jediného používateľa s názvom **csr**, ktorého IP adresa je 192.168.2.5. Skupina servers tiež obsahuje premenné, ktorými je možné definovať správanie sa a autentifikačné údaje cieľového zariadenia. **ansible_become** hovorí, že po pripojení sa staneme privilegovaným používateľom využitím metódy enable. Vzhľadom k tomu, že sa pripájame na používateľa, ktorý má root práva, tak netreba zadávať heslo do privilegovaného režimu parametrom **ansible_become_password**. Ďalej je potrebné nastaviť typ operačného systému a spôsob pripojenia, ktorý je v našom prípade **network_cli**. Poslednými parametrami je prihlasovacie meno a prihlasovacie heslo. Posledným parametrom je python interpreter, ktorý bude použitý na cieľovom zariadení na vykonanie ansible modulu.

Konfiguračný súbor sa zvykne nazývať tiež ansible playbook a je ho možné spustiť použitím nasledujúceho príkazu:

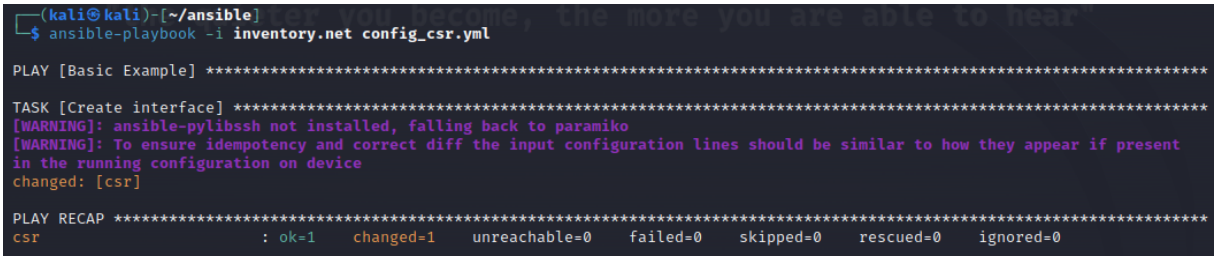
```
ansible-playbook -i inventory.net config_csr.yml
```

Pri prvotnom spustení vytvoreného playbook-u dôjde k chybe kvôli tomu, že cieľový virtuálny smerovač používa starší typ ssh zabezpečenia, čo znemožní ssh pripojenie:

Riešením je nastavenie ssh klienta a definovanie algoritmov pre šifrovanie a hash. Nasledujúce nastavenie umožní zmeniť defaultné algoritmy pre konkrétne zariadenie:

```
sudo nano /etc/ssh/ssh_config
Host 192.168.2.5
  HostKeyAlgorithms ssh-rsa
  KexAlgorithms diffie-hellman-group-exchange-sha1
```

Po opätovnom spustení vytvoreného ansible-playbook-u dôjde k úspešnej zmene konfigurácie, čo potvrdzuje aj kontrolný výpis:



Na cieľovom zariadení je možné skontrolovať úspešnosť vytvorenia rozhrania a nastavenia jeho IP adresy:

```
fromKali#sh ip int br
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       192.168.2.5     YES DHCP   up                    up
Loopback22             22.22.22.22     YES manual up                    up
```

Obohaťte vytvorený konfiguračný súbor o niekoľko ďalších úloh na zmenu konfigurácie:

```
---
- name: change config example
  hosts: csr
  gather_facts: no

  tasks:
      - name: "Create interface"
        ios_config:
          lines:
            - "interface loopback 22"
            - "ip add 22.22.22.22 255.255.255.255"

      - name: "Configure router ospf"
```

```
      ios_config:
        lines:
          - "network 22.22.22.22 0.0.0.0 area 0"
          - "network 2.22.222.0 0.0.0.255 area 0"
        parents:
          - "router ospf 1"

    - name: "Create ACL"
      ios_config:
        lines:
          - "10 permit ip host 10.20.30.22 any"
          - "20 permit ip 192.168.22.0 0.0.0.255 host 22.22.22.22"
        parents:
          - "ip access-list extended from_ansible"
```

```
┌──(kali㉿kali)-[~/ansible]
└─$ ansible-playbook -i inventory.net config_csr.yml

PLAY [change config example] ***************************************************************************************************

TASK [Create interface] ********************************************************************************************************
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar to how they appear if present
in the running configuration on device
changed: [csr]

TASK [Configure router ospf] ***************************************************************************************************
changed: [csr]

TASK [Create ACL] **************************************************************************************************************
changed: [csr]

PLAY RECAP *********************************************************************************************************************
csr                        : ok=3    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Pridajte úlohy umožňujúce zobraziť informácie zo zariadenia a uložte konfiguráciu do súborového systému Vášho zariadenia:

```
---
- name: change config example
  hosts: csr
  gather_facts: no

  vars:
     dir: ./dir/file

  tasks:
     - name: "Show Interfaces"
       ios_command:
         commands:
                - "show ip int br"
         register: cli_result

     - name: "Show interfaces in user friendly form"
       debug:
         msg: "{{ cli_result.stdout_lines[0] }}"
     - name: "Show ip route"
       ios_command:
         commands:
           - "show ip route"
         register: route_result

     - name: "Save route table to file"
```

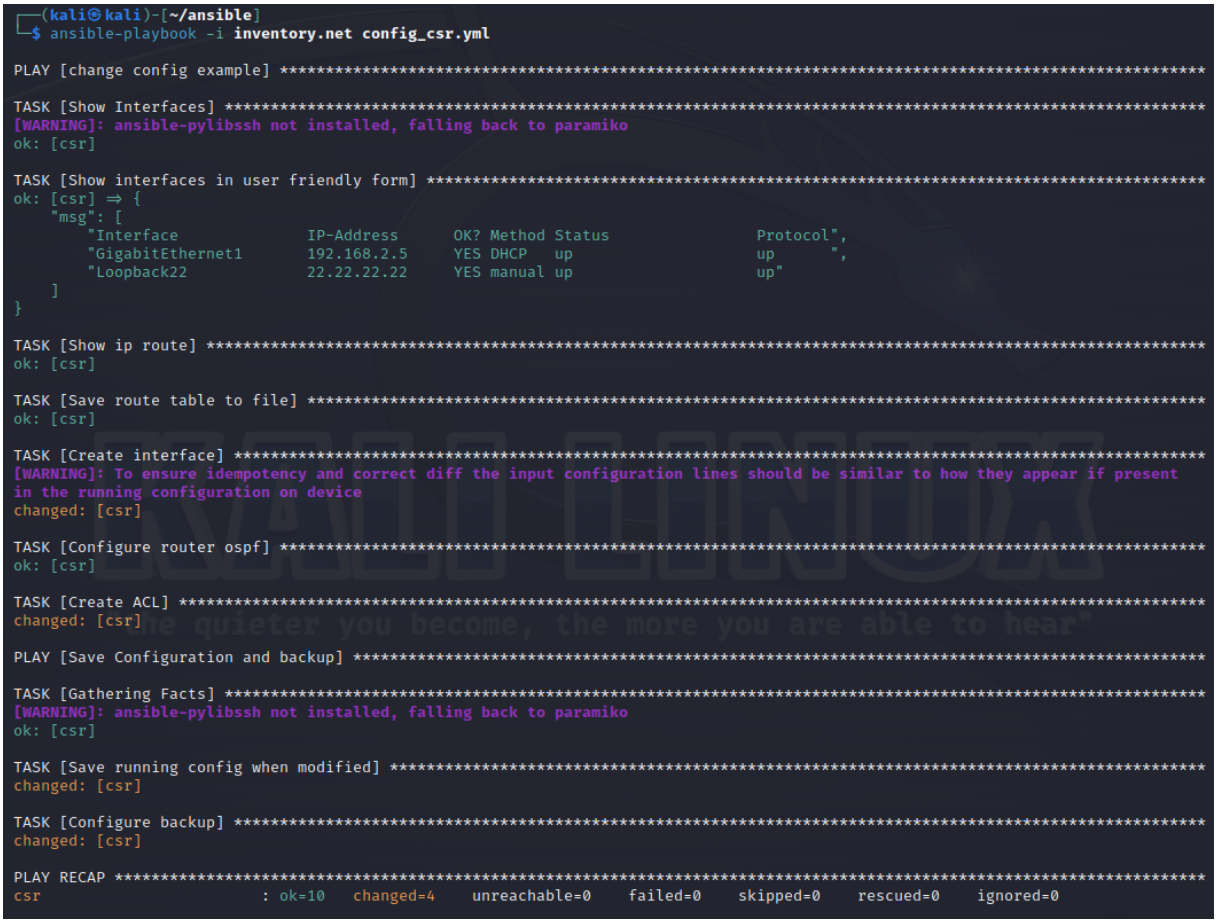
```
        copy:
          content: "{{ route_result.stdout[0]  }}"
          #dest: "./backup.txt"
          dest: "{{ dir  }}.txt"

    - name: "Create interface"
      ios_config:
        lines:
          - "interface loopback 22"
          - "ip add 22.22.22.22 255.255.255.255"

    - name: "Configure router ospf"
      ios_config:
        lines:
          - "network 22.22.22.22 0.0.0.0 area 0"
          - "network 2.22.222.0 0.0.0.255 area 0"
        parents:
          - "router ospf 1"

    - name: "Create ACL"
      ios_config:
        lines:
          - "10 permit ip host 10.20.30.22 any"
          - "20 permit ip 192.168.22.0 0.0.0.255 host 22.22.22.22"
        parents:
          - "ip access-list extended from_ansible"
```

```
┌──(kali㉿kali)-[~/ansible]
└─$ ansible-playbook -i inventory.net config_csr.yml

PLAY [change config example] ***************************************************************************************************

TASK [Show Interfaces] *********************************************************************************************************
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
ok: [csr]

TASK [Show interfaces in user friendly form] ***********************************************************************************
ok: [csr] => {
    "msg": [
        "Interface              IP-Address      OK? Method Status                Protocol",
        "GigabitEthernet1       192.168.2.5     YES DHCP   up                    up      ",
        "Loopback22             22.22.22.22     YES manual up                    up      "
    ]
}

TASK [Show ip route] ***********************************************************************************************************
ok: [csr]

TASK [Save route table to file] ************************************************************************************************
changed: [csr]

TASK [Create interface] ********************************************************************************************************
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar to how they appear if present
in the running configuration on device
changed: [csr]

TASK [Configure router ospf] ***************************************************************************************************
ok: [csr]

TASK [Create ACL] **************************************************************************************************************
changed: [csr]

PLAY RECAP *********************************************************************************************************************
csr                        : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Nakoniec pridajte na koniec konfiguračného súboru úplne novú sekciu, ktorá bude slúžiť na uloženie celej konfigurácie po vykonaní zmeny na sieťovom zariadení. Zároveň sa vykoná uloženie celej konfigurácie aj na lokálne zariadenie:

```
- name: Save Configuration and backup
  hosts: csr

  tasks:
      - name: "Save running config when modified"
        ios_config:
          save_when: modified
```

```
      - name: "Configure backup"
        ios_config:
          backup: yes
          backup_options:
              filename: "backup_config.txt"
              dir_path: "./"
...
```



Ako je možné vidieť, tak v rámci konfiguračného ansible súboru je možné tiež využívať prácu s lokálnym súborovým systémom a ukladať na neho získané dáta. Ďalej je možné výstup z jednej úlohy použiť v rámci druhej úlohy. Možné je tiež používať premenné a vypisovať výstup priamo na terminál.

Vytvorte ansible playbook, ktorým na cieľovom zariadení spustite nginx docker kontajner.

Najskôr je na cieľovom zariadení potrebné spustiť ssh server:

```
sudo apt-get install openssh-server
sudo systemctl enable ssh
sudo systemctl start ssh
```

Na lokálnom zariadení nainštalujte ssh pass a vytvorte inventory a konfiguračný súbor:

sudo apt install sshpass

nano ansible_docker.yml

```
---
- hosts: vm
  become: true
  tasks:

   - name: Pull default Docker image
          docker_image:
          name: "nginx"
          source: pull

   - name: Create default containers
          docker_container:
          name: "ansiblecontanier{{item}}"
          image: "nginx"
          state: started
          with_sequence: count=2
```

nano rvm.net

```
[remotevm]
vm ansible_host=192.168.2.6
[remotevm:vars]
ansible_ssh_user=testing
ansible_ssh_pass=testing
ansible_sudo_pass=testing
```

ansible-playbook -i rvm.net ansible_docker.yml

Uvedený konfiguračný súbor spustí na cieľovom serveri 2 nginx kontajneri. Uvedená úloha predpokladá, na cieľovom serveri, nainštalovaný docker engine.

Terraform

Terraform je IaaC nástroj umožňujúci vytvárať zdroje a prostredia využitím deklaratívnych konfiguračných súborov. Konfiguračné súbory definujú výsledný stav prostredia, voči ktorému sa porovnáva ten aktuálny a v prípade rozdielov sa vykonajú konfiguračné zmeny. Štandardne sa používa na vytváranie prostredí, zdrojov a virtuálnych serverov v cloud-ovom prostredí, no je ním možné riadiť aj lokálne, virtualizované a kontajnerizované prostredie.

Vďaka konfiguračnej povahe je možné na správu prostredia používať rôzne verzionovacie systémy ako napr. Git, čo umožňuje správu prostredí v tímoch. Terraform je univerzálny nástroj, ktorý vďaka doplnkom (angl. plugin) a poskytovateľom (angl. provider), ktoré poskytujú všetky potrebné nástroje na pripojenie k aplikačným rozhraniam (API) cieľového systému, dokáže spravovať širokú škálu prostredí. Úlohou poskytovateľov je pri interakcii so zdrojmi identifikovať všetky závislosti, či už pri vytváraní, ale aj mazaní zdrojov prostredia.

Vykonávanie v kontexte Terraform-u pozostáva z nasledujúcich krokov:

- **inicializácia**: nainštaluje všetky doplnky, ktoré sú potrebné pre správu infraštruktúry
- **plán**: ukáže plánované zmeny, ktoré budú vykonané pre dosiahnutie cieľového stavu
- **aplikovanie**: vykonanie plánovaných zmien

Terraform uchováva tzv. stav, ktorý umožňuje sledovať zmeny zdrojov a reprezentuje aktuálny stav, ktorý je pri potrebe zmien porovnávaný s novým očakávaným stavom vzhľadom k čomu sa vytvorí plán zmien, ktoré je potrebné vykonať tak, aby nový aktuálny stav zodpovedal definovanej konfigurácii.

K najznámejším Terraform poskytovateľom (definujú nástroje pre pripojenie a zdroje, s ktorými je možné interagovať) patria:

- vsphere/latest
- local/latest
- aws/latest
- azurerm/latest
- google/latest
- kubernetes/latest

Najskôr si ukážeme, ako je možné využitím nástroja Terraform čítať a meniť konfiguráciu sieťového zariadenia využitím protokolu RESTCONF. Dôležité je poznamenať, že pre lepšiu čitateľnosť je vhodné definovať terraform konfiguráciu viacerými súbormi, napr.:

- providers.tf
- resource.tf
- variables.tf

Terraform príkazy sú aplikované na všetky súbory ako celok. Po vykonaní konfiguračných zmien je pri prvom spustení vygenerovaný súbor s názvom terraform.tfstate, ktorý v sebe obsahuje reprezentáciu vytvoreného prostredia.

V našom prípade vytvorme súbor pre definovanie poskytovateľov, premenné, ktoré budú uchovávať prihlasovacie údaje, zdroje pre čítanie a zdroje pre zápis, konkrétne vytvoríme nasledujúcu štruktúru súborov:

- providers.tf
- ios_get.tf
- ios_change.tf
- ios_variables.tf

Obsah uvedených súborov znázorňujú nasledujúce výpisy:
ios_variables.tf

```
variable "host" {
  type     = string
  default  = "https://192.168.2.5"
  description = "Destination host address"
}

variable "username" {
  type     = string
  default  = "cisco"
  description = "Username"
}


variable "password" {
  type     = string
  default  = "cisco123!"
  description = "Password"
  sensitive   = true
}
```

providers.tf

```
terraform {
  required_providers {
      iosxe = {
      version = "0.1.1"
      source  = "CiscoDevNet/iosxe"
      }
  }
}

provider "iosxe" {
  host           = "${var.host}"
  device_username = "${var.username}"
  device_password = "${var.password}"
  request_timeout = 30
  insecure       = true
}
```

ios_get.tf

```
resource "iosxe_rest" "CONFIG_GET" {
  method  = "GET"
  path     = "/data/Cisco-IOS-XE-native:native"
  payload = ""
}

resource "iosxe_rest" "HOSTNAME_GET" {
  method  = "GET"
  path     = "/data/Cisco-IOS-XE-native:native/hostname"
  payload = ""
}

resource "iosxe_rest" "ACL_GET" {
```

```
  method = "GET"
  path   = "/data/Cisco-IOS-XE-native:native/ip/access-list"
}

resource "iosxe_rest" "NAT_GET" {
  method = "GET"
  path   = "/data/Cisco-IOS-XE-native:native/ip/nat"
}


resource "iosxe_rest" "OSPF_GET" {
  method = "GET"
  path   = "/data/Cisco-IOS-XE-native:native/router/ospf"
}
```

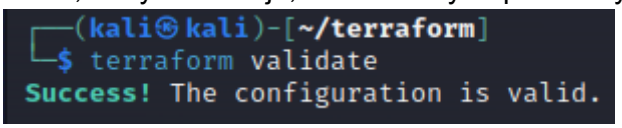
ios_change.tf

```
resource "iosxe_rest" "HOSTNAME_PUT" {
  method = "PUT"
  path   = "/data/Cisco-IOS-XE-native:native/hostname"
  payload = jsonencode(
      {
      "Cisco-IOS-XE-native:hostname" : "from_terraform"
      }
  )
}

resource "iosxe_rest" "ACL_POST" {
  method = "POST"
  path   = "/data/Cisco-IOS-XE-native:native/ip/access-list"
  payload = jsonencode(
      {
      "Cisco-IOS-XE-acl:extended" : [
      {
            "name" : "ACL",
            "access-list-seq-rule" : [
            {
            "sequence" : "10",
            "ace-rule" : {
                  "action" : "permit",
                  "protocol" : "ip",
                  "host" : "10.1.1.4",
                  "dst-any" : [
                  null
                  ]
            }
            },
            {
            "sequence" : "20",
            "ace-rule" : {
                  "action" : "permit",
                  "protocol" : "tcp",
                  "any" : [
                  null
```

```
                ],
                "dst-any" : [
                null
                ],
                "dst-eq" : 80
          }
          }
          ]
     }
     ]
     }
  )
}

resource "iosxe_rest" "NAT_PUT" {
  method = "PATCH"
  path   = "/data/Cisco-IOS-XE-native:native/ip/nat"
  payload = jsonencode(
     {
     "Cisco-IOS-XE-nat:nat" : {
     "pool" : [
          {
          "id" : "natpool",
          "prefix-length" : 27
          }
     ],
     "inside" : {
          "source" : {
          "list" : [
          {
                "id" : 1,
                "pool" : "natpool"
          }
          ]
          }
     }
     }
     }
  )
}


resource "iosxe_rest" "OSPF_POST" {
  method = "POST"
  path   = "/data/Cisco-IOS-XE-native:native/router"
  payload = jsonencode(
     {
     "Cisco-IOS-XE-ospf:ospf" : [
     {
          "id" : 123,
          "network" : [
          {
          "ip" : "10.1.1.0",
          "mask" : "0.0.0.15",
          "area" : 20
```

```
            },
            {
            "ip" : "10.22.22.0",
            "mask" : "0.0.0.255",
            "area" : 22
            }
            ]
        }
        ]
        }
    )
}
```

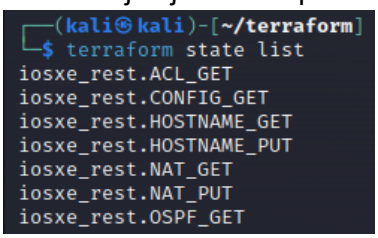
Po vytvorení súborov je vhodné použiť príkaz **terraform fmt**, ktorý formátuje vytvorené súbory do čitateľnej podoby tým, že preusporiada text s vhodným odsadením. Vhodným príkazom je tiež príkaz terraform validate, ktorý kontroluje, či sú súbory napísané syntakticky správne.



V tejto chvíli je možné vykonať inicializáciu, skontrolovať plán zmien a vykonať požadované zmeny:

```
terraform init
terraform plan
terraform apply –auto-approve
```

Pre zobrazenie aktuálne existujúcich zdrojov je možné použiť príkaz **terraform state list**:



Na zmazanie vytvorených zdrojov je možné použiť príkaz **terraform destroy**. V našom prípade ale tento príkaz nezmaže konfiguráciu, pretože nepracujeme so štandardným zdrojom, ale na zariadenie sme posielali HTTP GET a PUT požiadavky. Výstupy GET zdrojov je možné nájsť v stavovom súbore. Znova, takto sa pri získavaní dát nepracuje, ale je potrebné pracovať s **output** objektom.

Vytvorme a spusťme nginx kontajner využitím nástroja terraform:
main.tf

```
terraform {
  required_providers {
      docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
      }
  }
}
```

```
provider "docker" {}

resource "docker_image" "nginx" {
  name          = "nginx"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "testContanierTerraform"

  ports {
      internal = 80
      external = 8000
  }

  volumes {
      container_path = "/data"
      host_path       = "${path.cwd}/data"
  }
}
```

terraform init
sudo terraform plan
sudo terraform apply

```
┌──(kali㉿kali)-[~/terraform/docker]
└─$ sudo docker container ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                  NAMES
441cc6128a56   a6bd71f48f68   "/docker-entrypoint.…"   57 seconds ago   Up 55 seconds   0.0.0.0:8000→80/tcp   testContanierTerraf
orm
```

sudo terraform destroy

```
┌──(kali㉿kali)-[~/terraform/docker]
└─$ sudo docker container ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Správa infraštruktúry v AWS cloud-e:
aws.tf

```
resource "aws_vpc" "my_vpc" {
  cidr_block       = "10.10.0.0/16"
  enable_dns_hostnames = true
  enable_dns_support = true

  tags = {
    Name = "myVPC"
  }
```

```
}

resource "aws_subnet" "my_public_subnet" {
  vpc_id     = aws_vpc.my_vpc.id
  cidr_block = "10.10.10.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-2a"

  tags = {
    Name = "my-public"
  }
}

resource "aws_internet_gateway" "my_internet_gateway" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "my-igw"
  }
}

resource "aws_route_table" "my_public_rt" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "my-public-RT"
  }
}

resource "aws_route" "my_default_route" {
  route_table_id= aws_route_table.my_public_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.my_internet_gateway.id
}

resource "aws_route_table_association" "my_public_assoc" {
  subnet_id      = aws_subnet.my_public_subnet.id
  route_table_id = aws_route_table.my_public_rt.id
}

resource "aws_security_group" "my_sg" {
  name        = "my_sg"
  description = "my security group"
  vpc_id      = aws_vpc.my_vpc.id


  ingress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["myIP"]
  }

  egress {
    from_port        = 0
```

```
    to_port           = 0
    protocol          = "-1"
    cidr_blocks       = ["0.0.0.0/0"]
  }
}

data "aws_ami" "server_ami"{
  most_recent = true
  owners = ["cislo"]

  filter = {
    name = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-...-amd64-server-*"]
  }
}


ssh-keygen -t ed25519
[nastaviť kľúč ~/.ssh/mykey]
Zadajte heslo
   -  vygeneruje dvojicu súborov - kľúče mykey and mykey.pub

resource "aws_key_pair" "my_auth" {
  key_name   = "mykey"
  public_key = file("~/ssh/mykey.pub")
}

vytvoriť template definujúci, čo sa na zariadení vykoná:
#!/bin/bash
sudo apt-get update -y &&
sudo apt-get install -y \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common &&
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-
key add - &&
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
&&
sudo apt-get update -y &&
sudo sudo apt-get install docker-ce docker-ce-cli containerd.io -y
&&
sudo usermod -aG docker ubuntu

resource "aws_instance" "my_node" {
  instance_type = "t2.micro"
  ami           = data.aws_ami.server_ami.id
  key_name = aws_key_pair.my_auth.id
  vpc_security_group_ids = [aws_security_group.my_sg.id]
  subnet_id = aws_subnet.my_public_subnet.id
  User_data = file("userdata.tpl")

  root_block_device {
```

```
    volume_size = 10
  }

  tags = {
    Name = "dev-node"
  }
}
```

Poslednou témou, ktorá je ukázaná v rámci tohto manuálu je práca s provisioner objektom. Jedná sa o nástroj, ktorým je možné vykonať konfiguračné zmeny priamo na lokálnom alebo vzdialenom termináli. Tento prístup sa ale neodporúča používať, na takéto úlohy sa skôr používa Ansible.

```
resource "null_resource" "priklad1" {
  provisioner "local-exec" {
      command = "ping -c 5 www.cnl.sk"
  }
}

resource "null_resource" "priklad2" {
  provisioner "local-exec" {
      interpreter = ["python", "-c"]
      command    = "print('Hello World from Terraform-python')"
  }
}

resource "null_resource" "priklad3" {
  provisioner "local-exec" {
      command = "echo $VAR1 $VAR2 $VAR3 >> file.txt"
      environment = {
      VAR1 = "CNL"
      VAR2 = "KPI"
      VAR3 = "FEI"
      }
  }
}
```