Information

I have been lurking at Infrastructure as Code (IaC) for a while. I am using a proxmox server at work to setup a small network that i use to teach network security.
At the moment everything is very manual with a lot of clicking around in the proxmox gui and the VM command lines.

My goal is to be able to spin up an entire lab environment in a few minutes using terraform to provision infrastructure and Ansible to configure hosts and network devices.
My journey towards this begins with a writeup of the video by Learn Linux TV:
Provisioning Virtual Machines in Proxmox with Terraform – Full Walkthrough
I did tweak the `main.tf` file to comply with recent changes to the provider plugin `telmate/proxmox` and no guarantees are made that this will work in your environment.

Consider this guide as a crude working example of using terraform to provision a vm in proxmox, and beware that you probably need to tweak it to your needs, before using it.

There are a few prerequisites:

- proxmox server
- proxmox VM template

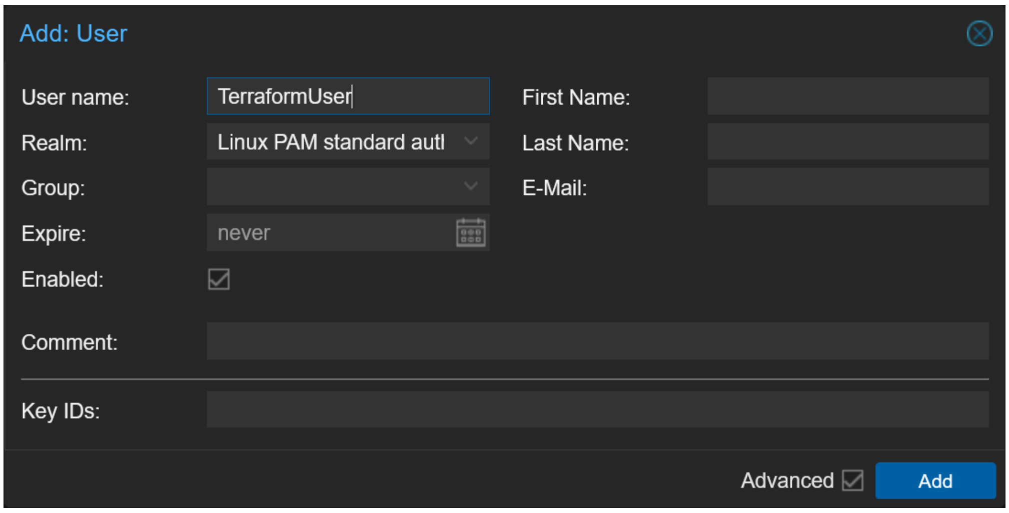
Proxmox permissions and access token

1. Copy the URL of your proxmox server ie. `https://10.10.10.10:8006`
2. Copy name of the VM template ie. `debian-12`

Setup proxmox ressources

1. click datacenter
2. go to users

3. add user - do not configure anything other than the name `TerraformUser`



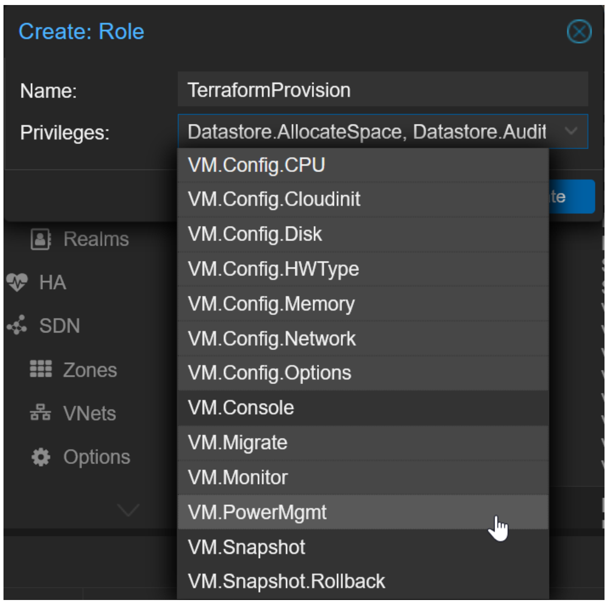
Add: User

User name:	TerraformUser	First Name:	
Realm:	Linux PAM standard autl	Last Name:	
Group:		E-Mail:	
Expire:	never		
Enabled:	✓		
Comment:			
Key IDs:			

Advanced ✓ Add

4. go to roles
5. create new role call it `TerraformProvision` and add these permissions:

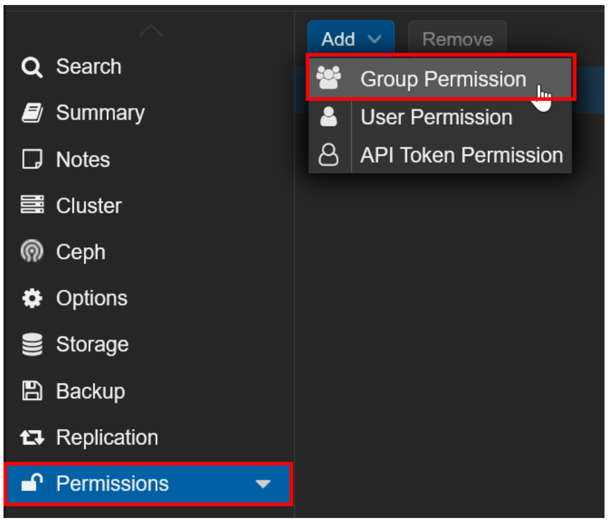
```
Datastore.AllocateSpace
Datastore.Audit
Pool.Allocate
SDN.Use
Sys.Audit
Sys.Console
Sys.Modify
Sys.PowerMgmt
VM.Allocate
VM.Audit
VM.Clone
VM.Config.CDROM
VM.Config.CPU
VM.Config.Cloudinit
VM.Config.Disk
VM.Config.HWType
VM.Config.Memory
VM.Config.Network
VM.Config.Options
VM.Migrate
VM.Monitor
VM.PowerMgmt
```


Create: Role

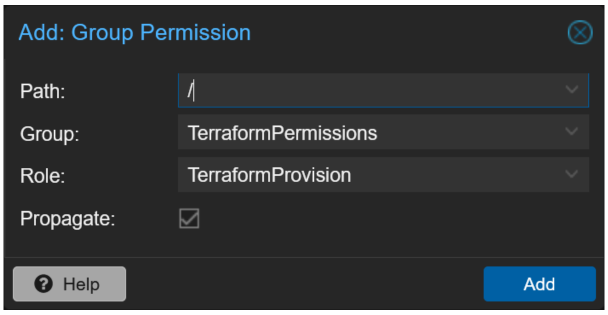
Name: TerraformProvision

Privileges: Datastore.AllocateSpace, Datastore.Audit

6. click groups
7. create a group called `TerraformPermissions`
8. click permissions -> add group permissions



9. add the following:


Add: Group Permission

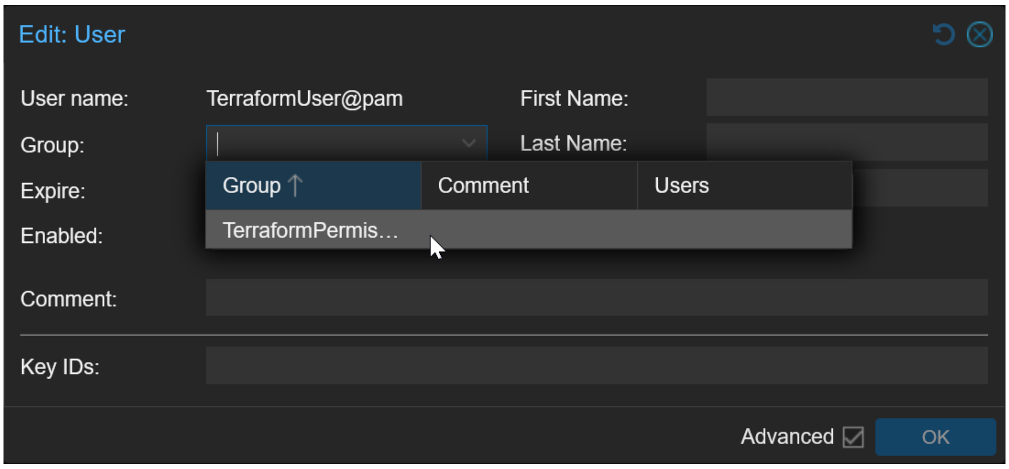
Path: /

Group: TerraformPermissions

Role: TerraformProvision

Propagate: ☑

10. edit `TerraformUser` and add group `TerraformPermissions` to it



Edit: User

User name: TerraformUser@pam

First Name:

Group:

Last Name:

Expire:

Enabled:

Group ↑	Comment	Users
TerraformPermis...		

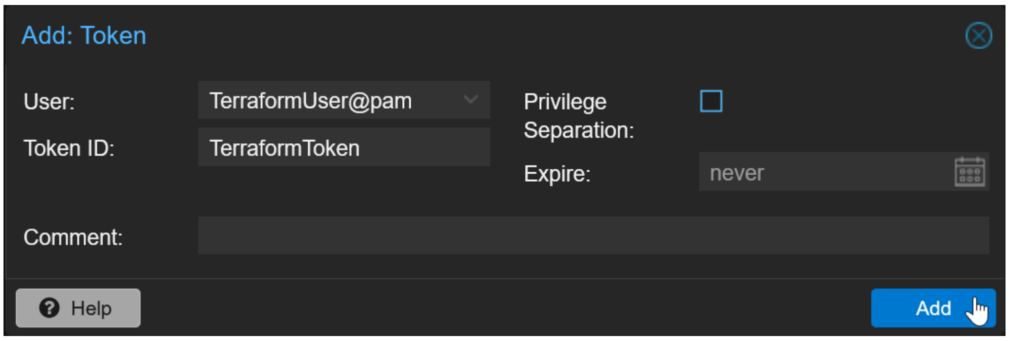
Comment:

Key IDs:

Advanced ☑ OK

Create proxmox API token

1. Click `API tokens`
2. Select the `TerraformUser`
3. Give the Token a name ie. `TerraformToken`
4. Uncheck `Privelege separation`
5. Click `Add`



Add: Token

User: TerraformUser@pam

Token ID: TerraformToken

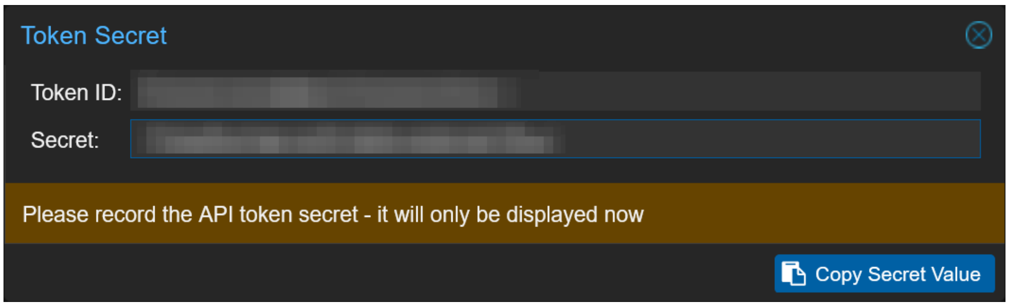
Privilege Separation: ☐

Expire: never

Comment:

Help Add

6. Save the `Token ID` and the `Secret` in a safe place like your vault or password manager
DO NOT EXPOSE THIS IN A GIT REPO OR OTHER PUBLIC PLACE



Token Secret

Token ID: [redacted]

Secret: [redacted]

Please record the API token secret - it will only be displayed now

Copy Secret Value

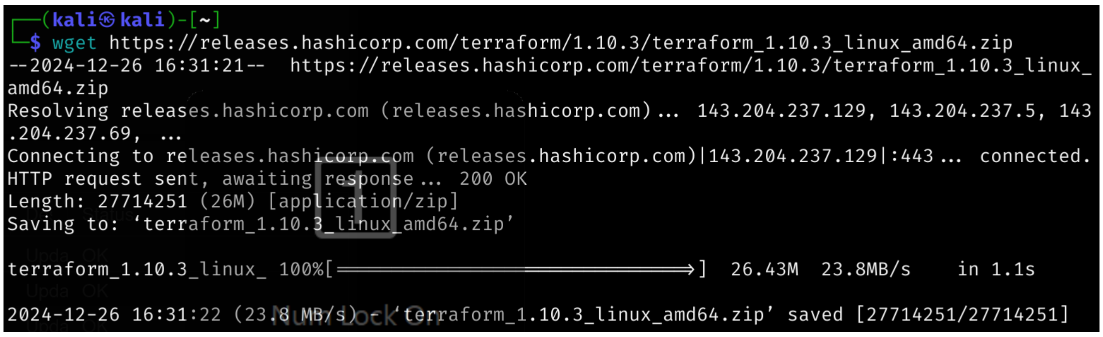
Terraform setup and VM configuration

These steps sets up terraform on a local Linux machine, this can be a VM or physical machine.
After setup of terraform the creation of a VM using terraform is explained.

Prepare Terraform on local machine

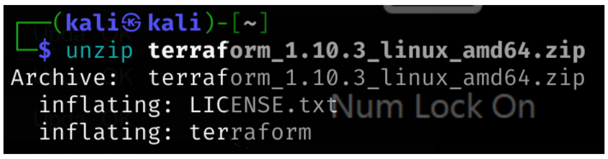
TODO: Add section about protecting secrets from version control

1. go to https://developer.hashicorp.com/terraform/install
2. copy the download link that fits your processor, ie. AMD64
3. switch to a linux terminal and `WGET` the link



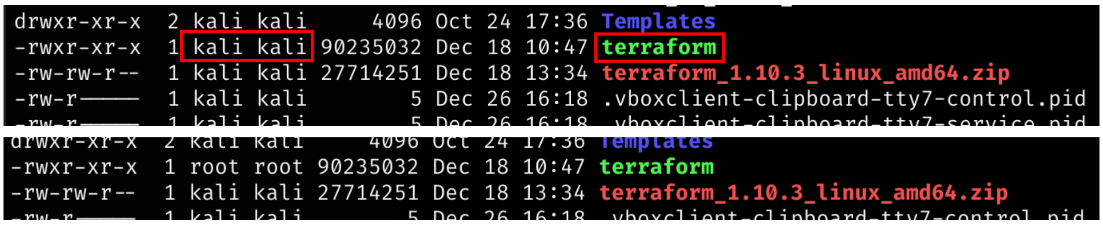
```
┌──(kali㉿kali)-[~]
└─$ wget https://releases.hashicorp.com/terraform/1.10.3/terraform_1.10.3_linux_amd64.zip
--2024-12-26 16:31:21--  https://releases.hashicorp.com/terraform/1.10.3/terraform_1.10.3_linux_
amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 143.204.237.129, 143.204.237.5, 143
.204.237.69, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|143.204.237.129|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27714251 (26M) [application/zip]
Saving to: 'terraform_1.10.3_linux_amd64.zip'

terraform_1.10.3_linux_ 100%[===================================>]  26.43M  23.8MB/s    in 1.1s

2024-12-26 16:31:22 (23.8 MB/s) - 'terraform_1.10.3_linux_amd64.zip' saved [27714251/27714251]
```

4. use the terminal to unzip the downloaded file



```
┌──(kali㉿kali)-[~]
└─$ unzip terraform_1.10.3_linux_amd64.zip
Archive:  terraform_1.10.3_linux_amd64.zip
  inflating: LICENSE.txt
  inflating: terraform
```

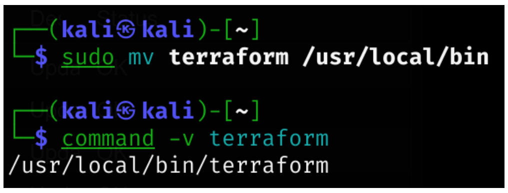
5. if on a multi user system you can change ownership to ie. `root` with `chown root:root terraform`



```
drwxr-xr-x  2 kali kali     4096 Oct 24 17:36 Templates
-rwxr-xr-x  1 kali kali 90235032 Dec 18 10:47 terraform
-rw-rw-r--  1 kali kali 27714251 Dec 18 13:34 terraform_1.10.3_linux_amd64.zip
-rw-r-----  1 kali kali        5 Dec 26 16:18 .vboxclient-clipboard-tty7-control.pid
```

```
-rwxr-xr-x  1 root root 90235032 Dec 18 10:47 terraform
-rw-rw-r--  1 kali kali 27714251 Dec 18 13:34 terraform_1.10.3_linux_amd64.zip
```

6. To use the `terraform` command you need to move the file with the command `sudo mv terraform /usr/local/bin/`
7. Check that the path is recognized by typing `command -v terraform` and confirm that the output is `/usr/local/bin/terraform` (this confirms that the `terraform` command is available)



```
┌──(kali㉿kali)-[~]
└─$ sudo mv terraform /usr/local/bin

┌──(kali㉿kali)-[~]
└─$ command -v terraform
/usr/local/bin/terraform
```

Create terraform files

1. create a directory for terraform in the home directory `mkdir ~/terraform`
2. create a file called `main.tf` in the `~/terraform` directory
3. open the `main.tf` file in your favorite editor and add the following:

main.tf

```
terraform {
    required_providers {
        proxmox = {
            source = "telmate/proxmox"
            version = "3.0.1-rc6" // https://registry.terraform.io/telmate/proxmox
        }
    }
}

provider "proxmox" {
    pm_api_url          = "https://url-to-proxmox-server:8006/api2/json"
    pm_api_token_id     = "your-token-id"
    pm_api_token_secret = "your-secret"
    pm_tls_insecure     = true
}

resource "proxmox_vm_qemu" "vm-instance" {
    name                = "vm-instance"
    target_node         = "your-proxmox-node-name"
    clone               = "your-template-name"
    full_clone          = true
    cores               = 2
    memory              = 2048

    disk {
        slot            = "scsi0"
        size            = "32G"
        type            = "disk"
        storage         = "your-storage-volume"
        discard         = "true"
    }

    network {
        model     = "virtio"
        bridge    = "vmbr1"
        firewall  = false
        link_down = false
        id        = 0
    }

}
```

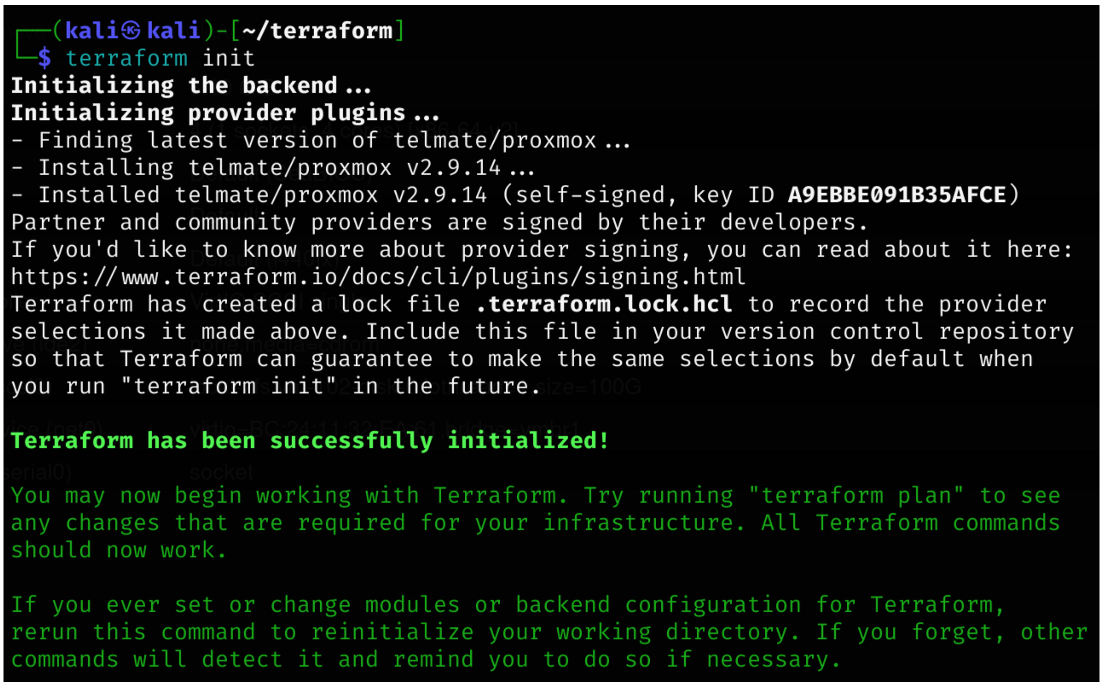
4. replace the `your-token-id` and `your-secret` with the token id and secret from the previous steps

DISCLAIMER: DO NOT EXPOSE TOKEN ID AND SECRET IN A GIT REPO OR OTHER PUBLIC PLACE
PLEASE USE A VARIABLES FILE, ENVIRONMENT VARIABLES OR HASHICORP VAULT TO MANAGE THIS INFORMATION OUTSIDE OF VERSION CONTROL
FOR ADDTIONAL INFORMATION SEE:
https://developer.hashicorp.com/terraform/tutorials/configuration-language/sensitive-variables

5. replace the `your-proxmox-node-name` with the name of your proxmox node
6. replace the `your-template-name` with the name of your VM template
7. check that the `vmbr1` corresponds to your desired network bridge in proxmox

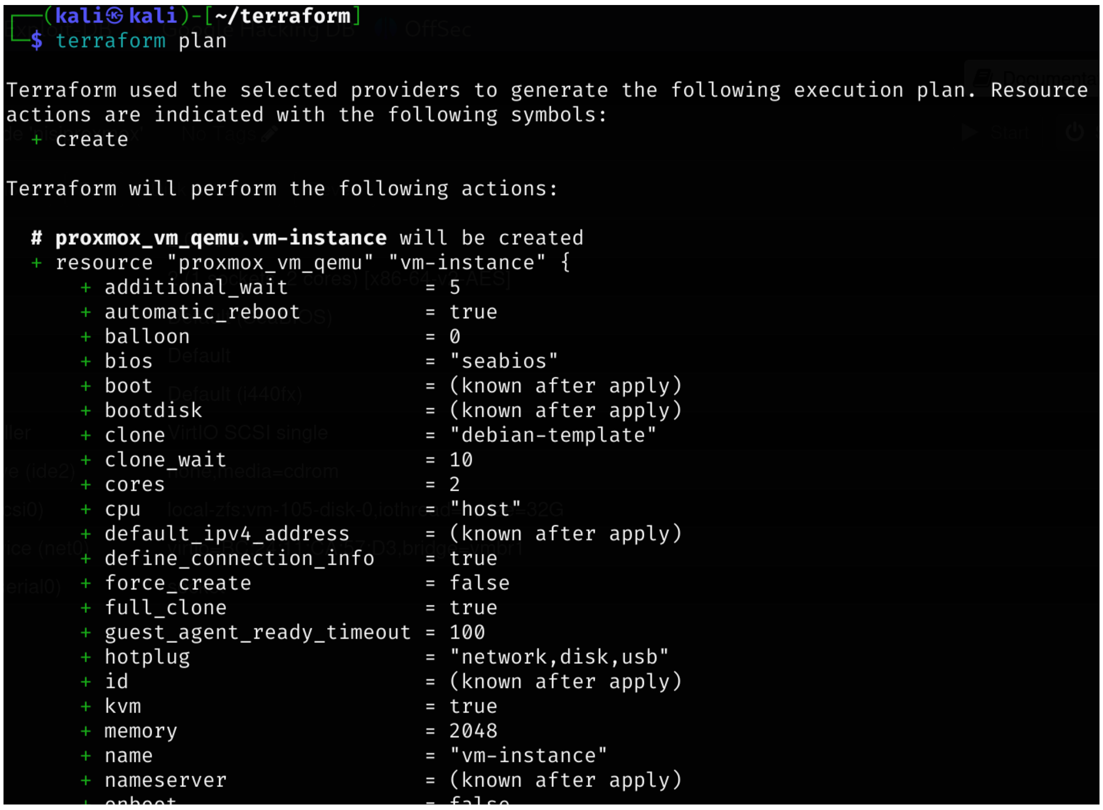
Run terraform

1. go to the `~/terraform` directory
2. run `terraform init`



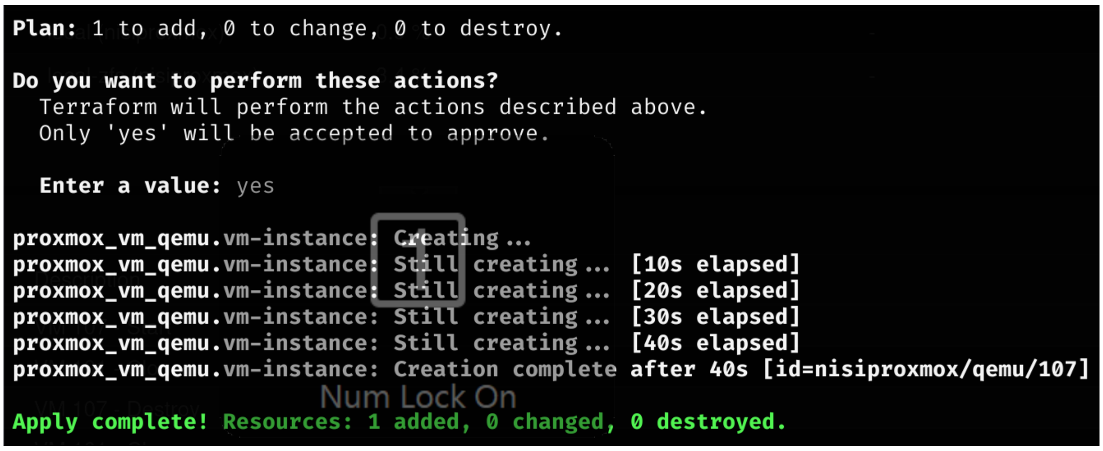
```
┌──(kali㉿kali)-[~/terraform]
└─$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of telmate/proxmox...
- Installing telmate/proxmox v2.9.14...
- Installed telmate/proxmox v2.9.14 (self-signed, key ID A9EBBE091B35AFCE)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

3. run `terraform plan` to check what will be changed (this does not cjange anything in your proxmox node)



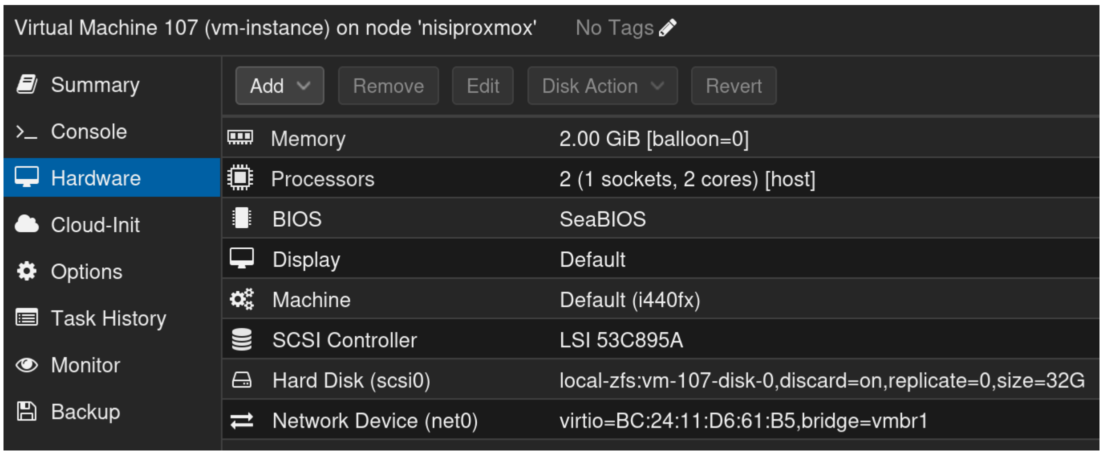
```
┌──(kali㉿kali)-[~/terraform]
└─$ terraform plan

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:

  # proxmox_vm_qemu.vm-instance will be created
  + resource "proxmox_vm_qemu" "vm-instance" {
      + additional_wait           = 5
      + automatic_reboot          = true
      + balloon                   = 0
      + bios                      = "seabios"
      + boot                      = (known after apply)
      + bootdisk                  = (known after apply)
      + clone                     = "debian-template"
      + clone_wait                = 10
      + cores                     = 2
      + cpu                       = "host"
      + default_ipv4_address      = (known after apply)
      + define_connection_info    = true
      + force_create              = false
      + full_clone                = true
      + guest_agent_ready_timeout = 100
      + hotplug                   = "network,disk,usb"
      + id                        = (known after apply)
      + kvm                       = true
      + memory                    = 2048
      + name                      = "vm-instance"
      + nameserver                = (known after apply)
      + onboot                    = false
```

4. run `terraform apply` this will attempt to create the VM in proxmox



```
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

proxmox_vm_qemu.vm-instance: Creating...
proxmox_vm_qemu.vm-instance: Still creating... [10s elapsed]
proxmox_vm_qemu.vm-instance: Still creating... [20s elapsed]
proxmox_vm_qemu.vm-instance: Still creating... [30s elapsed]
proxmox_vm_qemu.vm-instance: Still creating... [40s elapsed]
proxmox_vm_qemu.vm-instance: Creation complete after 40s [id=nisiproxmox/qemu/107]

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

5. check that the VM is created in the proxmox GUI



Virtual Machine 107 (vm-instance) on node 'nisiproxmox'

Memory	2.00 GiB [balloon=0]
Processors	2 (1 sockets, 2 cores) [host]
BIOS	SeaBIOS
Display	Default
Machine	Default (i440fx)
SCSI Controller	LSI 53C895A
Hard Disk (scsi0)	local-zfs:vm-107-disk-0,discard=on,replicate=0,size=32G
Network Device (net0)	virtio=BC:24:11:D6:61:B5,bridge=vmbr1

2024-12-27 08:38:53