Terraform Digital Ocean LUG

This is a simple documentation about my first tests using Terraform by HashiCorp with the cloud provider digitalocean.

If you use **this link** to register with digitalocean you will receive a **$50 in credit** that is **valid for 30 days** meaning you can spin up 6 of the cheapest droplets for 1 month for free!

Install Terraform

I used Arch Linux for my tests, so installing terraform was as simple as

```
sudo pacman -S terraform
```

If you use a different distribution search the web how to install it there.

If your installation is working you should be able to execute the following command without any errors:

```
[user@arch]$ terraform version
Terraform v0.12.15
```

Verify / Create SSH Keys

The user you are logged in must have valid SSH keys setup. If not done already create the necessary public/private keypair with

```
[user@arch]$ ssh-keygen
```

Get Digital Ocean API Token

Generate a Personal Access Token via the DigitalOcean control panel. Instructions to do that can be found in this link: How to Generate a Personal Access Token.

Store the token in a save place for later use.

Create a work folder

A terraform project lives within a folder. So let's create the folder `~/Dokumente/terraform/do` (do=digitalocean):

```
[user@arch]$ mkdir -p "~/Dokumente/terraform/do"
```

and go to this folder:

```
[user@arch]$ cd "~/Dokumente/terraform/do"
[user@arch do]$
```

Create terraform.tfvars

The "Digital Ocean provider" within Terraform needs the 4 variables `do_token`, `pub_key`, `pvt_key` and `ssh_fingerprint`.
The `do_token` is your Digital Ocean API key.
`pub_key` is the path to your SSH public key eg. `/home/user/.ssh/id_rsa.pub`.
`pvt_key` is the path to your SSH private key eg. `/home/pcfreak/.ssh/id_rsa`
`ssh_fingerprint` is the MD5 hash of your public key without the `MD5:` at the beginning. You can output it in the correct format with:

```
ssh-keygen -E md5 -lf ~/.ssh/id_rsa.pub | awk '{print $2}' | sed 's@^MD5:@@g'
```

We now will put these 4 variables into the file `~/Dokumente/terraform/do/terraform.tfvars`.
The file looks like this (use your values here):

```
do_token = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
pub_key = "/home/user/.ssh/id_rsa.pub"
pvt_key = "/home/user/.ssh/id_rsa"
ssh_fingerprint = "b6:b6:3e:00:b3:80:bc:5e:64:44:bd:73:b1:83:11:80"
```

terraform provider file

We will now create a so called provider file for Digital Ocean. The name does not really matter but the extension must be `.tf`. Let's just create `provider_do.tf` with the following content:

```
variable "do_token" {}
variable "pub_key" {}
variable "pvt_key" {}
variable "ssh_fingerprint" {}



provider "digitalocean" {
  token = var.do_token
```

```
  version = "~> 1.11"
}
```

Initialize Terraform

The first run of `terraform init` will now retrieve the necessary files for the provider `digitalocean` and store them in the hidden folder `./.terraform`

```
[user@arch do]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "digitalocean" (terraform-providers/digitalocean)
1.11.0...

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.



[user@arch do]$ ls -al
drwxr-xr-x 3 user users 4096 18. Nov 15:05 ./
drwxr-xr-x 4 user users 4096 18. Nov 13:57 ../
drwxr-xr-x 3 user users 4096 18. Nov 15:05 .terraform/
-rw-r--r-- 1 user users  201 18. Nov 14:25 provider_do.tf
-rw-r--r-- 1 user users  230 18. Nov 14:04 terraform.tfvars
```

Create a plan

We will now create a plan to create a droplet named `ubuntu-terraform` with the image `ubuntu-19-10-x64` in the region `fra1` with the size `s-1vcpu-1gb` and private networking `true` by creating the file `ubuntu-19-10-x64.tf` with the following content:

```
# get a list of images with curl -X GET -H "Authorization: Bearer $DO_PAT"
"https://api.digitalocean.com/v2/images"
```

```
# get a list of all regions with curl -X GET -H "Authorization: Bearer $DO_PAT" "https://
api.digitalocean.com/v2/regions"
# get a list of all sizes with curl -X GET -H "Authorization: Bearer $DO_PAT"
"https://api.digitalocean.com/v2/sizes"

resource "digitalocean_droplet" "test" {
  name = "ubuntu-terraform"

  image              = "ubuntu-19-10-x64"
  region             = "fra1"
  size               = "s-1vcpu-1gb"
  private_networking = true
  ssh_keys = [
    "${var.ssh_fingerprint}"
  ]

  connection {
    host        = self.ipv4_address
    user        = "root"
    type        = "ssh"
    private_key = file(var.pvt_key)
    timeout     = "2m"
  }


  provisioner "remote-exec" {
    inline = [
      "export PATH=$PATH:/usr/bin",
      # update system
      "sudo apt-get -y update",
      "sudo apt-get -y upgrade",
      # install nginx
      "sudo apt-get -y install nginx"
    ]
  }
}
```

The above size `s-1vcpu-1gb` is the 5$/month plan at digital ocean, the cheapest droplet.

How did I find out the values? You can use the digital ocean API to do so:

```
export DO_PAT='<yourdigitaloceanapikey>'
# images
```

```
curl -X GET -H "Authorization: Bearer $DO_PAT" "https://api.digitalocean.com/v2/images" |
jq
# regions
curl -X GET -H "Authorization: Bearer $DO_PAT" "https://api.digitalocean.com/v2/regions"
| jq
# sizes
curl -X GET -H "Authorization: Bearer $DO_PAT" "https://api.digitalocean.com/v2/sizes" |
jq
```

Our folder looks now like this:

```
[user@arch do]$ ls -al
drwxr-xr-x 3 user users 4096 18. Nov 15:14 ./
drwxr-xr-x 5 user users 4096 18. Nov 15:08 ../
drwxr-xr-x 3 user users 4096 18. Nov 15:09 .terraform/
-rw-r--r-- 1 user users  201 18. Nov 14:25 provider_do.tf
-rw-r--r-- 1 user users  230 18. Nov 14:04 terraform.tfvars
-rw-r--r-- 1 user users  524 18. Nov 15:14 ubuntu-19-10-x64.tf
```

Validate configuration

The command `terraform validate` verifies all of our configuration files and checks their syntax:

```
[user@arch do]$ terraform validate
Success! The configuration is valid.
```

Create a terraform plan

The following command will plan what we want to do and stores the plan in the binary file `ubuntu-19-10-x64.tfplan`

```
[user@arch do]$ terraform plan -out ubuntu-19-10-x64.tfplan

Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.



------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
```

```
Terraform will perform the following actions:

  # digitalocean_droplet.test will be created
  + resource "digitalocean_droplet" "test" {
      + backups              = false
      + created_at           = (known after apply)
      + disk                 = (known after apply)
      + id                   = (known after apply)
      + image                = "ubuntu-19-10-x64"
      + ipv4_address         = (known after apply)
      + ipv4_address_private = (known after apply)
      + ipv6                 = false
      + ipv6_address         = (known after apply)
      + ipv6_address_private = (known after apply)
      + locked               = (known after apply)
      + memory               = (known after apply)
      + monitoring           = false
      + name                 = "ubuntu-terraform"
      + price_hourly         = (known after apply)
      + price_monthly        = (known after apply)
      + private_networking   = true
      + region               = "fra1"
      + resize_disk          = true
      + size                 = "s-1vcpu-1gb"
      + ssh_keys             = [
          + "5c:f2:49:e9:d4:65:55:ba:a3:24:bc:20:cc:34:ae:1b",
        ]
      + status               = (known after apply)
      + urn                  = (known after apply)
      + vcpus                = (known after apply)
      + volume_ids           = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

This plan was saved to: ubuntu-19-10-x64.tfplan

To perform exactly these actions, run the following command to apply:
    terraform apply "ubuntu-19-10-x64.tfplan"
```

Apply the plan

After checking the ouput of the previous command we can now apply the plan. **(This will setup the droplet)**

```
[user@arch do]$ terraform apply "ubuntu-19-10-x64.tfplan"
digitalocean_droplet.test: Creating...
digitalocean_droplet.test: Still creating... [10s elapsed]
digitalocean_droplet.test: Still creating... [20s elapsed]
digitalocean_droplet.test: Still creating... [30s elapsed]
digitalocean_droplet.test: Still creating... [40s elapsed]
digitalocean_droplet.test: Creation complete after 43s [id=167628939]

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

The state of your infrastructure has been saved to the path
below. This state is required to modify and destroy your
infrastructure, so keep it safe. To inspect the complete state
use the `terraform show` command.

State path: terraform.tfstate
```

The droplet has been created. If want, you can verify it via your digitalocean web console.

get ip of droplet

Terraform keeps the last known state of in the file `terraform.tfstate`. It is in binary format (actually just a zip file) but the `terraform` command can read it directly and output its values:

terraform state pull

```
[user@arch do]$ terraform state pull
{
  "version": 4,
  "terraform_version": "0.12.15",
  "serial": 1,
  "lineage": "a5050872-d579-b0b0-2193-7cf38b661a1b",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "digitalocean_droplet",
```

```
    "name": "test",
    "provider": "provider.digitalocean",
    "instances": [
      {
        "schema_version": 1,
        "attributes": {
          "backups": false,
          "created_at": "2019-11-18T14:20:55Z",
          "disk": 25,
          "id": "167628939",
          "image": "ubuntu-19-10-x64",
          "ipv4_address": "165.22.89.5",
          "ipv4_address_private": "10.135.132.247",
          "ipv6": false,
          "ipv6_address": "",
          "ipv6_address_private": null,
          "locked": false,
          "memory": 1024,
          "monitoring": false,
          "name": "ubuntu-terraform",
          "price_hourly": 0.00744,
          "price_monthly": 5,
          "private_networking": true,
          "region": "fra1",
          "resize_disk": true,
          "size": "s-1vcpu-1gb",
          "ssh_keys": [
              "b6:b6:3e:00:b3:80:bc:5e:64:44:bd:73:b1:83:11:80"
          ],
          "ssh_keys": null,
          "status": "active",
          "tags": null,
          "urn": "do:droplet:167628939",
          "user_data": null,
          "vcpus": 1,
          "volume_ids": []
        },
        "private": "eyJas5hla4dVyc2lvbiI6IjEifQ=="
      }
    ]
  }
]
```

```
}
```

terraform show

```
[user@arch do]$ terraform show
# digitalocean_droplet.test:
resource "digitalocean_droplet" "test" {
    backups              = false
    created_at           = "2019-11-18T14:36:01Z"
    disk                 = 25
    id                   = "167631477"
    image                = "ubuntu-19-10-x64"
    ipv4_address         = "165.22.89.5"
    ipv4_address_private = "10.135.132.247"
    ipv6                 = false
    locked               = false
    memory               = 1024
    monitoring           = false
    name                 = "ubuntu-terraform"
    price_hourly         = 0.00744
    price_monthly        = 5
    private_networking   = true
    region               = "fra1"
    resize_disk          = true
    size                 = "s-1vcpu-1gb"
    ssh_keys             = [
        "b6:b6:3e:00:b3:80:bc:5e:64:44:bd:73:b1:83:11:80",
    ]
    status               = "active"
    urn                  = "do:droplet:167631477"
    vcpus                = 1
    volume_ids           = []
}
```

To just retrieve the `ipv4_address` you could use something like:

```
[user@arch do]$ terraform state pull | jq -
r .resources[].instances[].attributes.ipv4_address
165.22.89.5
```

Connect to droplet via SSH

Because we already added our SSH keys to the droplet we should be able to connect directly using SSH:

```
[user@arch do]$ ssh root@165.22.89.5
The authenticity of host '165.22.89.5 (165.22.89.5)' can't be established.
ECDSA key fingerprint is SHA256:BgWZknjXWB6TZhy/Lj59oNo7dfvVYZU5uuV2vDKhFKc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '165.22.89.5' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-18-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Nov 18 14:49:17 UTC 2019

  System load:  0.08              Processes:           100
  Usage of /:   4.8% of 24.06GB   Users logged in:     0
  Memory usage: 17%               IP address for eth0: 104.248.248.159
  Swap usage:   0%                IP address for eth1: 10.135.2.69

0 updates can be installed immediately.
0 of these updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ubuntu-terraform:~#
```

Destroy the droplet

To destroy everything we created we execute:

```
[user@arch do]$  terraform destroy
digitalocean_droplet.test: Refreshing state... [id=167628939]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy
```

```
Terraform will perform the following actions:

  # digitalocean_droplet.test will be destroyed
  - resource "digitalocean_droplet" "test" {
      - backups              = false -> null
      - created_at           = "2019-11-18T14:20:55Z" -> null
      - disk                 = 25 -> null
      - id                   = "167628939" -> null
      - image                = "ubuntu-19-10-x64" -> null
      - ipv4_address         = "46.101.217.157" -> null
      - ipv4_address_private = "10.135.30.242" -> null
      - ipv6                 = false -> null
      - locked               = false -> null
      - memory               = 1024 -> null
      - monitoring           = false -> null
      - name                 = "ubuntu-terraform" -> null
      - price_hourly         = 0.00744 -> null
      - price_monthly        = 5 -> null
      - private_networking   = true -> null
      - region               = "fra1" -> null
      - resize_disk          = true -> null
      - size                 = "s-1vcpu-1gb" -> null
      - status               = "active" -> null
      - tags                 = [] -> null
      - urn                  = "do:droplet:167628939" -> null
      - vcpus                = 1 -> null
      - volume_ids           = [] -> null
    }

Plan: 0 to add, 0 to change, 1 to destroy.

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

digitalocean_droplet.test: Destroying... [id=167628939]
digitalocean_droplet.test: Still destroying... [id=167628939, 10s elapsed]
digitalocean_droplet.test: Still destroying... [id=167628939, 20s elapsed]
digitalocean_droplet.test: Destruction complete after 22s

Destroy complete! Resources: 1 destroyed.
```

Our droplet has been destroyed!

This was a very quick introduction to **Terraform**. Feel free to read the online docs and continue to improve your knowledge.