UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Compiladores 2 – Gabriel Brolo

Laboratorio 3

Daniel Alfredo Rayo Roldán, 22933

Flavio André Galán Donis, 22386

José Alejandro Prince Martinez,z 22087

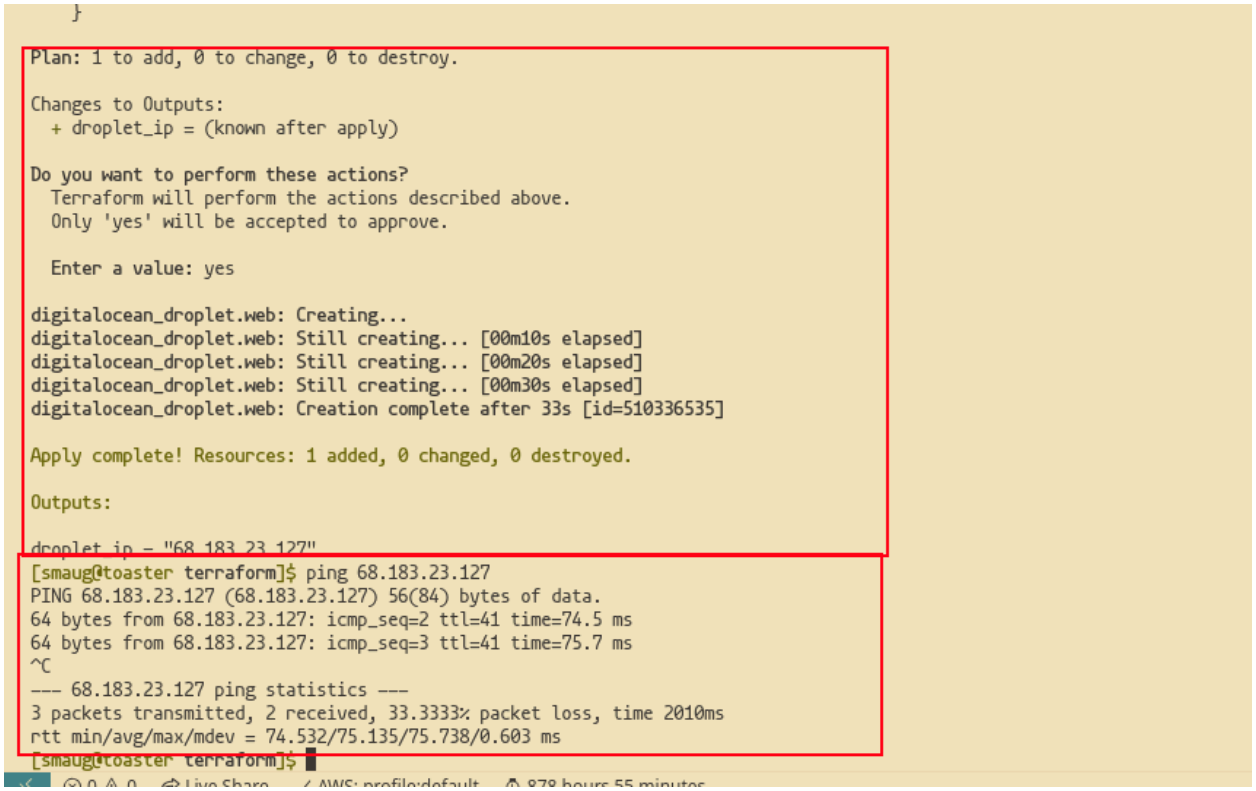
Repositorio: https://github.com/DanielRasho/Compiladore-lab3

Video:https://youtu.be/IEjQvq9lyNQ

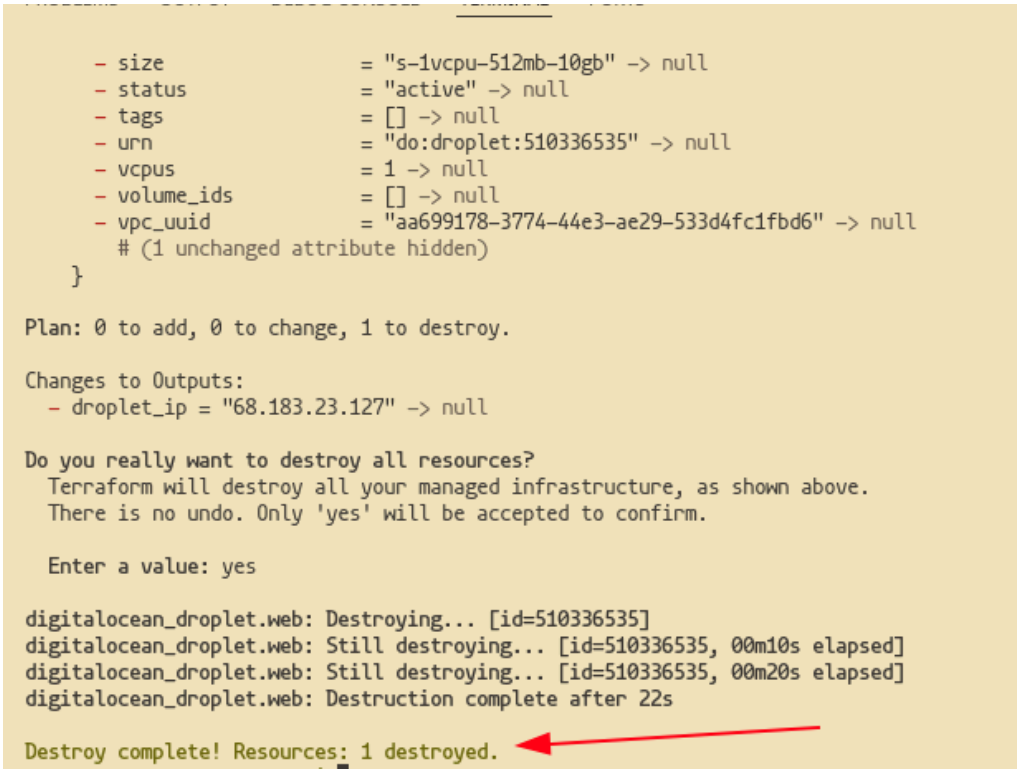
Terraform

```
Terraform initialized in an empty directory!

The directory has no Terraform configuration files. You may begin working
with Terraform immediately by creating Terraform configuration files.
[smaug@toaster lab3]$ cd terraform/
[smaug@toaster terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding digitalocean/digitalocean versions matching "~> 2.0"...
- Installing digitalocean/digitalocean v2.61.0...
- Installed digitalocean/digitalocean v2.61.0 (signed by a HashiCorp partner, key ID F82037E524B9C0E8)
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://developer.hashicorp.com/terraform/cli/plugins/signing
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.
[smaug@toaster 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:

  # digitalocean_droplet.web will be created
  + resource "digitalocean_droplet" "web" {
      + backups              = false
```

```
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.
[smaug@toaster 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:

  # digitalocean_droplet.web will be created
  + resource "digitalocean_droplet" "web" {
      + backups              = false
      + created_at           = (known after apply)
      + disk                 = (known after apply)
      + graceful_shutdown    = false
      + id                   = (known after apply)
      + image                = "ubuntu-24-10-x64"
      + ipv4_address         = (known after apply)
      + ipv4_address_private = (known after apply)
      + ipv6                 = false
      + ipv6_address         = (known after apply)
      + locked               = (known after apply)
      + memory               = (known after apply)
      + monitoring           = false
      + name                 = "example-droplet"
      + price_hourly         = (known after apply)
      + price_monthly        = (known after apply)
      + private_networking   = (known after apply)
      + region               = "nyc1"
      + resize_disk          = true
      + size                 = "s-1vcpu-512mb-10gb"
      + status               = (known after apply)
      + urn                  = (known after apply)
      + vcpus                = (known after apply)
      + volume_ids           = (known after apply)
      + vpc_uuid             = (known after apply)
    }

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

Changes to Outputs:
```



En el último recuadro marcado de rojo se puede observar que el droplet ya puede ser “pingueado”.



Bash

Para la gestión de dependencias se utilizó Nix en vez de Docker, a continuación se encuentra el archivo de configuración.

flake.nix

```nix
{
  description = "A basic multiplatform flake";

  inputs = {
    nixpkgs.url = "github:nixos/nixpkgs?ref=nixos-unstable";
  };

  outputs = {
    self,
    nixpkgs,
  }: let
    # System types to support.
    supportedSystems = ["x86_64-linux" "x86_64-darwin" "aarch64-linux" "aarch64-darwin"];

    # Helper function to generate an attrset '{ x86_64-linux = f "x86_64-linux"; ... }'.
    forAllSystems = nixpkgs.lib.genAttrs supportedSystems;

    # Nixpkgs instantiated for supported system types.
    nixpkgsFor = forAllSystems (system: import nixpkgs {inherit system; config.allowUnfree = true;});
  in {
    devShells = forAllSystems (system: let
      pkgs = nixpkgsFor.${system};
    in {
      default = pkgs.mkShell {
        packages = [
          pkgs.terraform
          pkgs.jq
        ];
      };
    });
  };
}
```

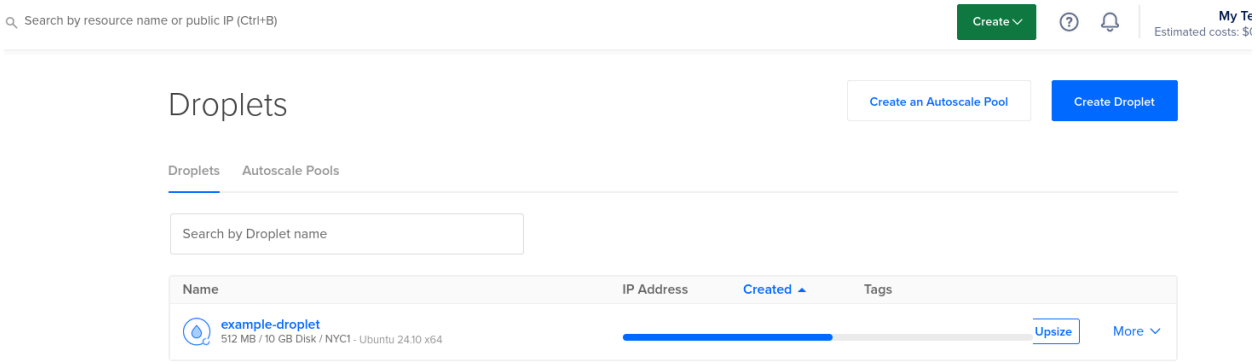
A continuación se muestran los 2 comandos para la creación y destrucción de un droplet ejecutados con éxito.

```
[smaug@toaster bash]$ ./create_droplet.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3975  100  3878  100    97   3539     88  0:00:01  0:00:01 --:--:--  3626
Droplet created with ID: 510337886
[smaug@toaster bash]$ ./destroy_droplet.sh
Droplet with ID 510337886 has been destroyed
[smaug@toaster bash]$
```

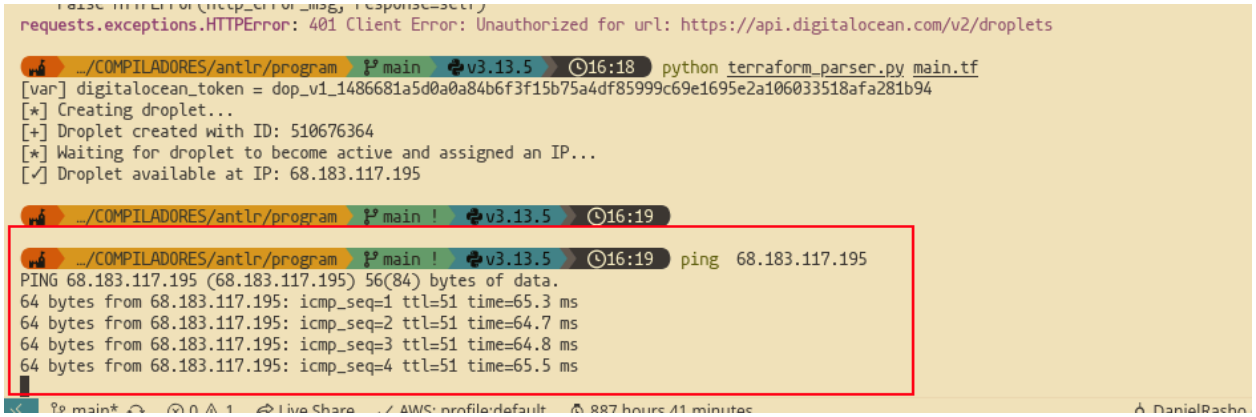
ANTLR

Creacion del droplet

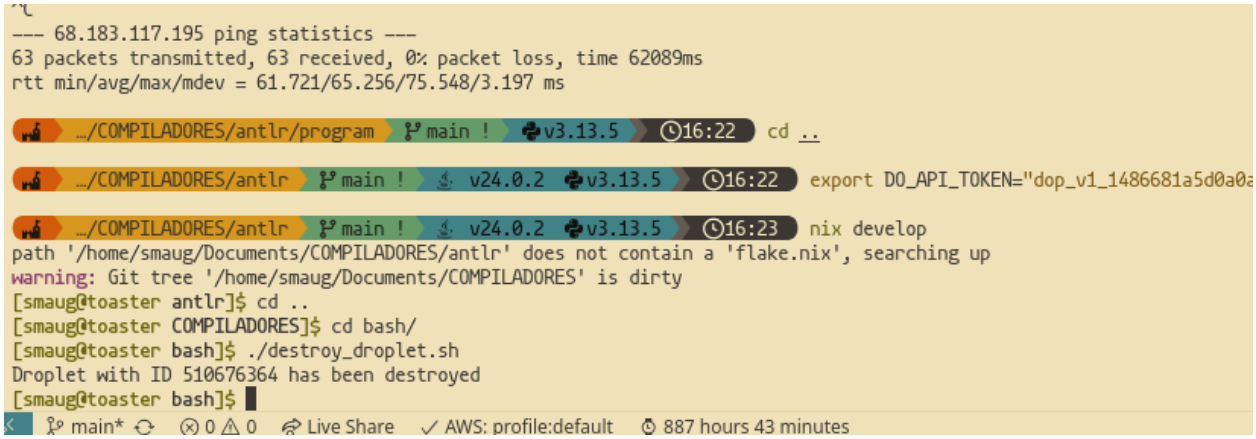
El droplet creado ya se muestra en el dashboard



Ping al droplet creado



Eliminación del droplet manualmente



Destrucción de Recursos en la nube

Configuración SSH Key

La configuración de la ssh key se realiza durante la creación del droplet, se verifica si la key usada ya existe o no. Caso de ssh key existente:

```
(venv) [akice@Akice-PC program]$ python terraform_parser.py main.tf
[var] digitalocean_token = dop_v1_6342c964d1100c1e96f09fccc575574c314588939136efc413d8c874a2587654
[i] SSH key already exists in DigitalOcean: prince (1a:87:ab:a0:2a:46:7c:db:08:fe:c1:d1:93:6b:82:79)
[*] Creating droplet...
[+] Droplet created with ID: 510819538
[*] Waiting for droplet to become active and assigned an IP...
[✓] Droplet available at IP (134.209.117.224) with ID (510819538)
```

En el caso de querer usar un ssh key nueva, primero se debe subir a digital ocean porque de no subirla, da error 403 indicando que el token no tiene los permisos para realizar la acción de subir la ssh key a digital ocean:

```
[var] digitalocean_token = dop_v1_6342c964d1100c1e96f09fccc575574c314588939136efc413d8c874a2587654
Traceback (most recent call last):
  File "/home/akice/Documents/Repositories/Compiladore-lab3/antlr/program/terraform_parser.py", line 220,
in <module>
    main()
    ~~~~^^
  File "/home/akice/Documents/Repositories/Compiladore-lab3/antlr/program/terraform_parser.py", line 213,
in main
    (ip, id) = create_droplet(token, listener.droplet_config)
               ~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/akice/Documents/Repositories/Compiladore-lab3/antlr/program/terraform_parser.py", line 137,
in create_droplet
    add_resp.raise_for_status()
    ~~~~~~~~~~~~~~~~~~~~~~~~~^^
  File "/home/akice/Documents/Repositories/Compiladore-lab3/antlr/venv/lib/python3.13/site-packages/reques
ts/models.py", line 1026, in raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 403 Client Error: Forbidden for url: https://api.digitalocean.com/v2/accoun
t/keys
```