

Laboratorio 3 - Droplet en DigitalOcean con Terraform

Se simuló un flujo tipo terraform apply y terraform destroy, pero implementado desde cero, usando un subconjunto del lenguaje Terraform. Se aplicó los principios de análisis léxico, sintáctico y semántico aprendidos en el curso de Construcción de Compiladores.

| Componente | Aplicación en el lab |
|----------------------------|--|
| Análisis léxico | ANTLR genera el lexer desde TerraformSubset.g4. |
| Análisis sintáctico | Se usa el parser generado para validar estructuras. |
| Análisis semántico | El listener interpreta lo que el parser reconoce y ejecuta lógica como create_droplet. |

Integrantes

- Josué Say

Enlaces

- Enlace a repositorio
- Enlace a ejecución del código

Guia 1 (Construcción de compiladores laboraotrio guiado no 4 bash)

Se creó y destruyó un Droplet en DigitalOcean manualmente usando Bash + Docker.

Respuesta de la ejecución de los comandos:

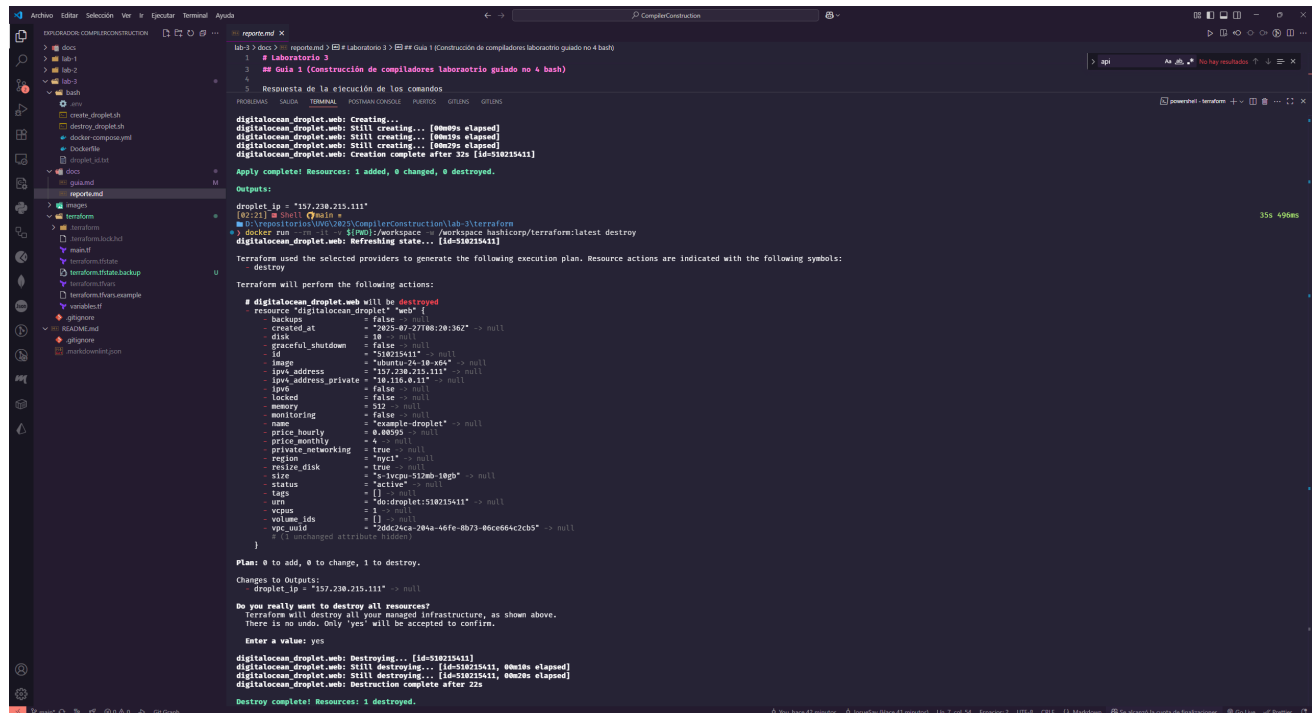
[illegible]

Figure 1: Ejecución de comandos docker

Guia 2 (Construcción de compiladores laboraotrio guiado no 4 terraform)

Se utilizó la herramienta Terraform oficial para crear un droplet. Se usaron las versiones de docker porque local en windows no me funcionó (xd). Para esto se puede revisar la [guia](#) realizada.

Respuesta de la ejecución de los comandos:



```
lab-3 > reportand > # Laboratorio 3 > # Guia 1 (Construcción de compiladores laboraotrio guiado no 4 bash)
1
2 # Laboratorio 2
3 # Guia 1 (Construcción de compiladores laboraotrio guiado no 4 bash)
4
5 Respuesta de la ejecución de los comandos
PROBLEMAS GUÍA TERMINAL PROBLEMAS GUÍA PROBLEMAS GUÍA PROBLEMAS GUÍA PROBLEMAS GUÍA
digitalocean_droplet.web: Creating...
digitalocean_droplet.web: Still creating... [0m00s elapsed]
digitalocean_droplet.web: Still creating... [0m01s elapsed]
digitalocean_droplet.web: Still creating... [0m02s elapsed]
digitalocean_droplet.web: Creation complete after 32s [id=518215411]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
Outputs:
droplet_ip = "157.230.215.111"
[4121] # Shell: Q/main = $[0m02s elapsed]
# P: Repositorio: W012025/CompilerConstruction/lab-3/terraform
# docker run --rm --shm-size 1g --env DO_API_TOKEN=$DO_API_TOKEN --env DO_REPO_PATH=/workspaces/CompilerConstruction/lab-3/terraform:latest destroy
digitalocean_droplet.web: Refreshing state... [id=518215411]
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:
# digitalocean_droplet.web will be destroyed
resource "digitalocean_droplet" "web" {
  backups      = false
  created_at   = "2025-09-27T08:20:36Z"
  disk        = 10
  graceful_shutdown = false
  id          = "518215411"
  image       = "ubuntu-20-04"
  ipxe_address = "157.230.215.111"
  ipvs_address = "10.156.0.11"
  ipvs        = false
  locked      = false
  memory      = 512
  monitoring  = false
  name        = "example-droplet"
  price_hourly = 0.00595
  price_monthly = 0
  private_networking = true
  region      = "nyc1"
  resize_disk = true
  size        = "4gb-512mb-10gb"
  status      = "active"
  tags        = []
  urn         = "do:digitalocean:droplet:518215411"
  vcpus       = 1
  volume_ids  = []
  vpc_uuid    = "206c24ca-204a-46fe-b073-06c064c2c0b"
}
Plan: 0 to add, 0 to change, 1 to destroy.
Changes to Outputs:
droplet_ip = "157.230.215.111"
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.web: Destroying... [id=518215411]
digitalocean_droplet.web: Still destroying... [id=518215411, 0m00s elapsed]
digitalocean_droplet.web: Still destroying... [id=518215411, 0m00s elapsed]
digitalocean_droplet.web: Destruction complete after 22s
Destroy complete! Resources: 1 destroyed.
```

Figure 2: Ejecución de comandos docker

Guia principal

Se diseñó, parseó y ejecutó un subconjunto del lenguaje Terraform usando ANTLR.

Se creó un archivo adicional main.tf.example.

Para comenzar, copia su contenido a un nuevo archivo main.tf con el siguiente comando:

```
cp main.tf.example main.tf
```

Luego, edita el archivo main.tf y reemplaza en la **línea 20** el valor default por el API Key real (usada en los pasos previos para la carpeta bash y transform) para más información se puede consultar la [guia](#) hecha para este repositorio:

```
default = "DO_API_TOKEN"
```

Además, se creó un archivo run.sh que automatiza el proceso de ejecución posterior.

Para ejecutarlo, utiliza:

`./run.sh`

Esto ejecutará todo el proceso solicitado.

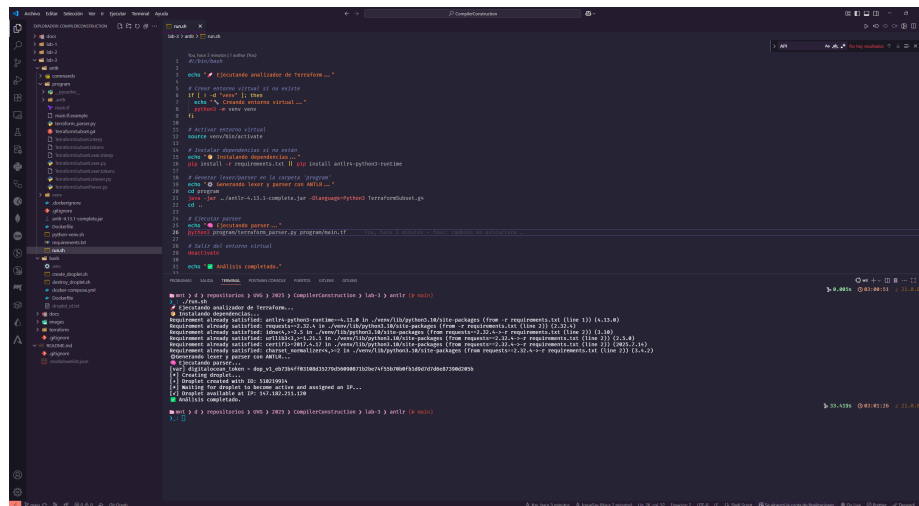


Figure 3: Primera ejecución del archivo principal

Al hacer un ping y ejecutar el destroy al droplet dado el nuevo id:

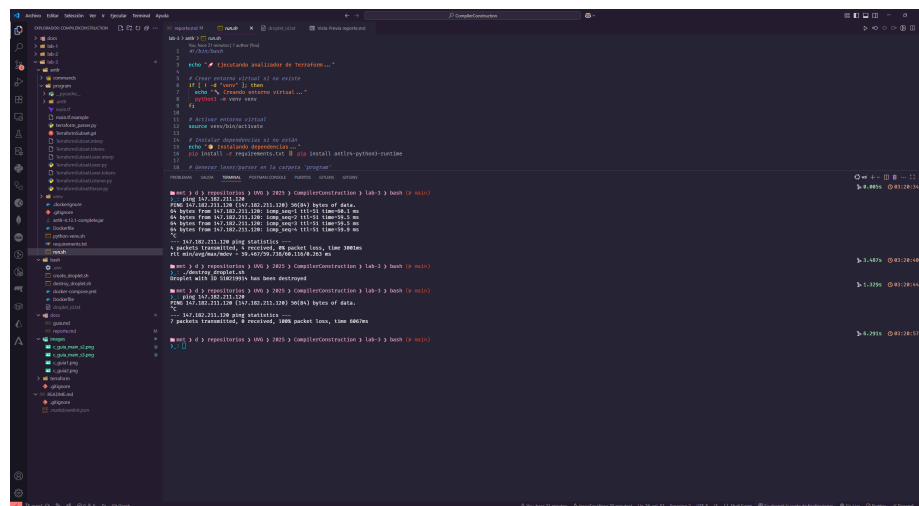


Figure 4: Segunda ejecución del archivo principal

¿Qué se implementó?

- Una gramática TerraformSubset.g4 que define sintaxis válida: provider, variable, resource, output.
- Se generaron clases en Python con ANTLR para lexer/parser.
- Se usó un listener (TerraformApplyListener) para construir un analizador semántico que:
 - Extrae variables.

- Valida el provider.
- Interpreta recursos.
- Aplica (crea) o destruye droplets dependiendo de la bandera CLI `--apply` o `--destroy`.
- Se agregó soporte para parámetros desde la línea de comandos (`--apply` y `--destroy`) para simular `terraform apply` y `terraform destroy`.
- Se implementó la función `saveState(...)` para guardar un archivo `terraform.tfstate` en formato JSON, que almacena el ID, nombre e IP pública del droplet creado.
- La función `create_droplet(...)` fue modificada para devolver tanto la IP como el ID del droplet, permitiendo guardar su estado.
- Se implementó la función `destroyDroplet(...)` que lee el archivo `.tfstate`, extrae el ID del droplet y lo elimina utilizando la API de DigitalOcean.
- El archivo de entrada `.tf` ahora es interpretado con una lógica condicional que, según el parámetro proporcionado, ejecuta la creación (`apply`) o eliminación (`destroy`) del recurso definido.

Archivo `.tfstate`

- Implementaste lógica para guardar un state como Terraform:
 - Con ID, nombre e IP del droplet creado.
 - Permitted luego eliminarlo con `--destroy`.

Comandos

```
cd antlr/
python3 program/terraform_parser.py program/main.tf --apply
python3 program/terraform_parser.py program/main.tf --destroy
```

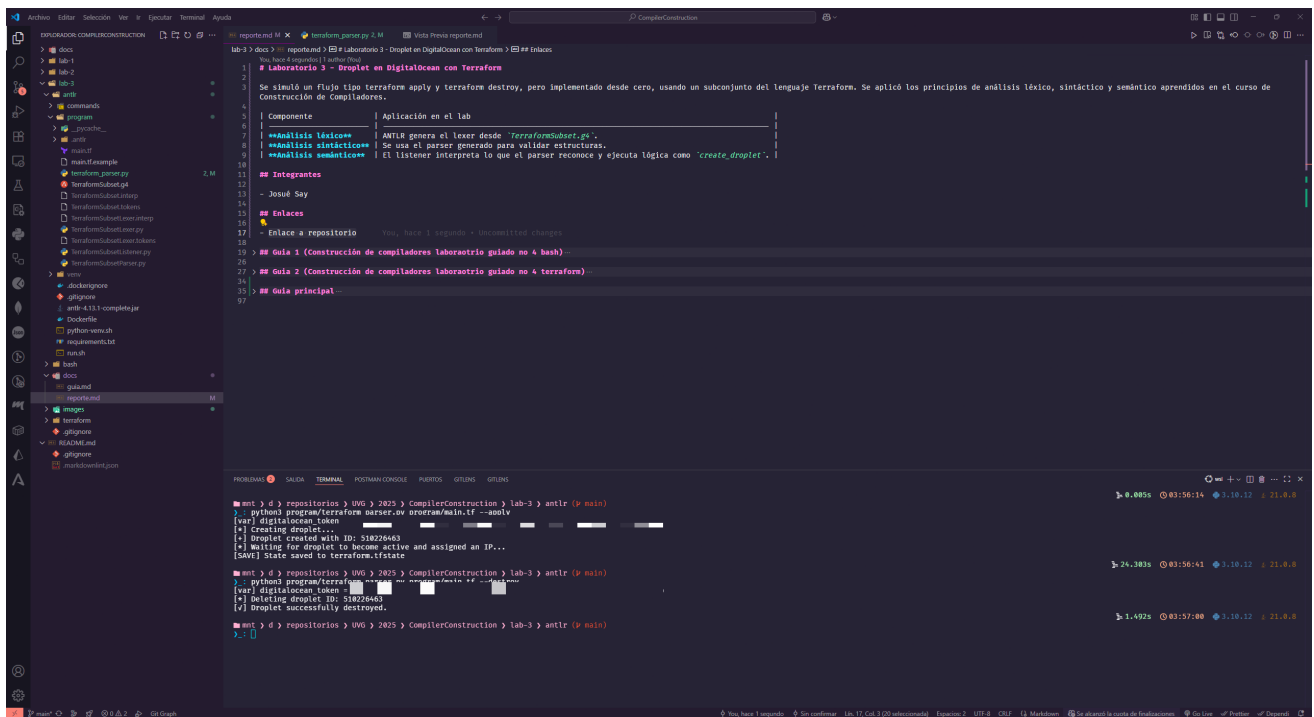


Figure 5: Ejecución final