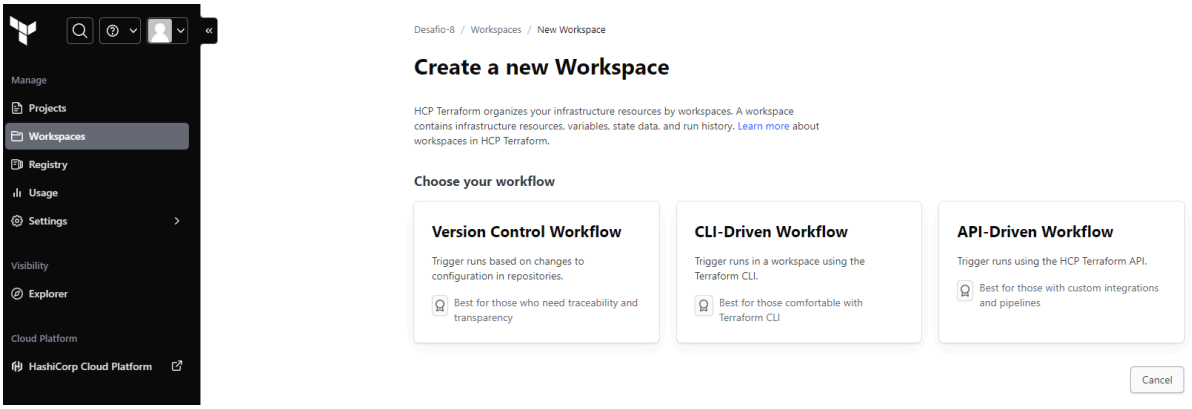
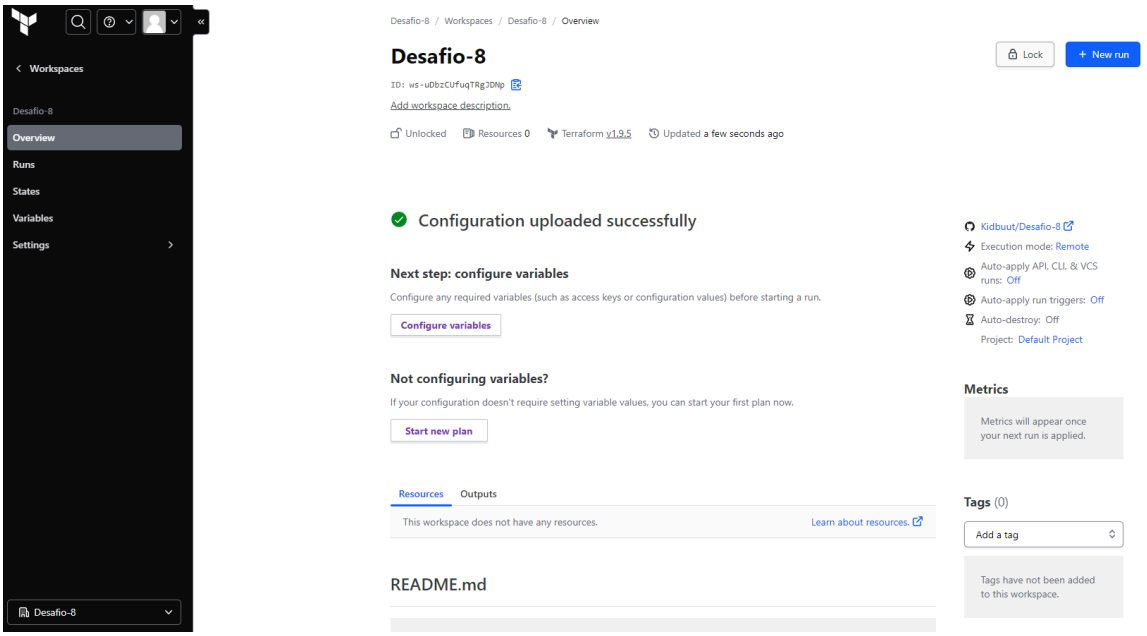
Desafio 8

Para iniciar este desafío tenemos que tener cuentas e instalación en nuestro entorno local las herramientas:

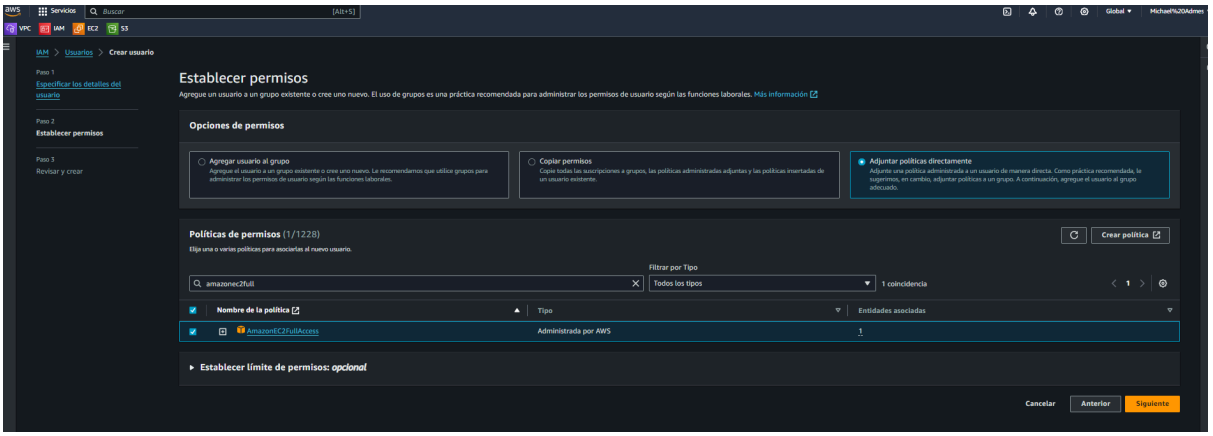
- Terraform
- Github
- AWS

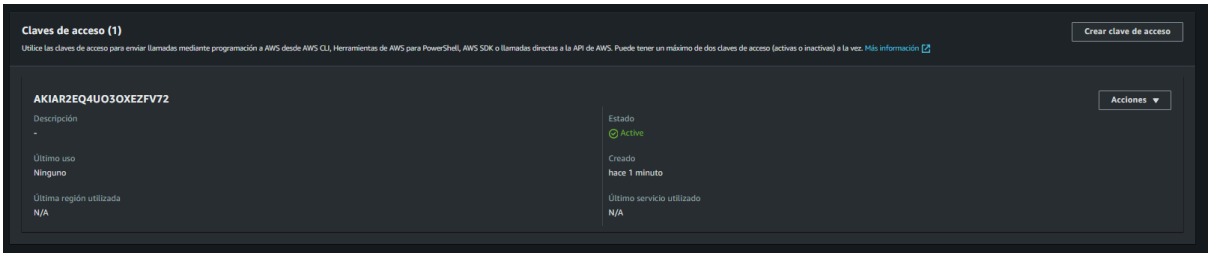


Una vez teniendo todo seteado pasamos a la dirección https://app.terraform.io/ a la creación de nuestra organización en mi caso **Desafio-8** seguidamente del Workspaces con la configuración para que tenga acceso a nuestro Github.

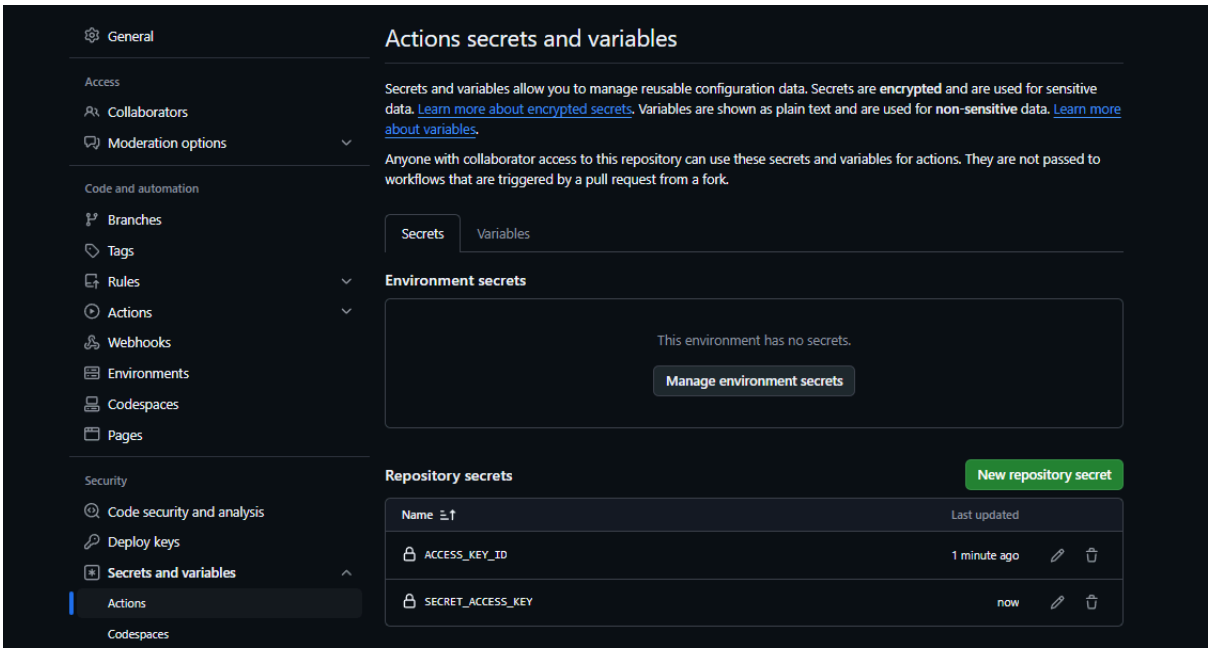


Luego seguimos con la creación de nuestro usuario en AWS, y le damos los permisos correspondientes y seguidamente generamos la Clave de acceso.



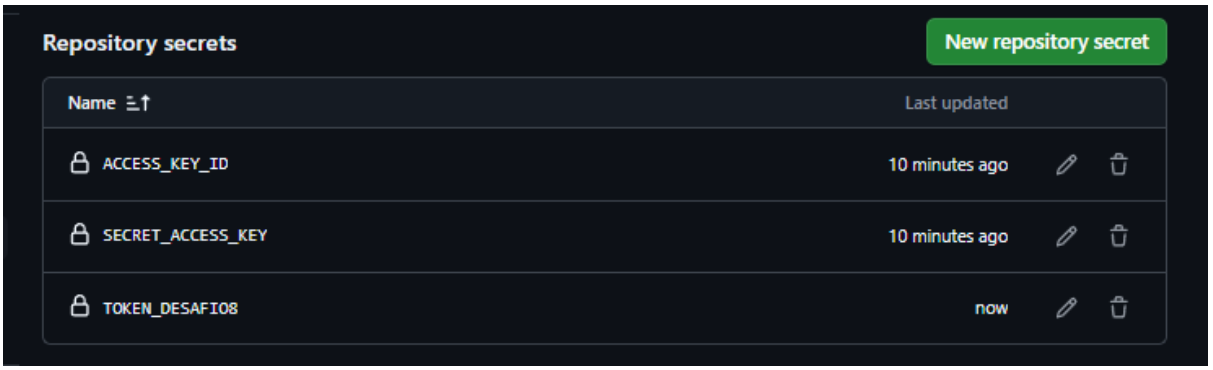


Una vez creado el usuario nos dirigimos a Github a la ventana de Settings-Credential and variables-action y agregamos las credenciales del usuario de AWS.



Luego de esto en AWS creamos una Api Token y lo hacemos desde EC2-Setting-Token

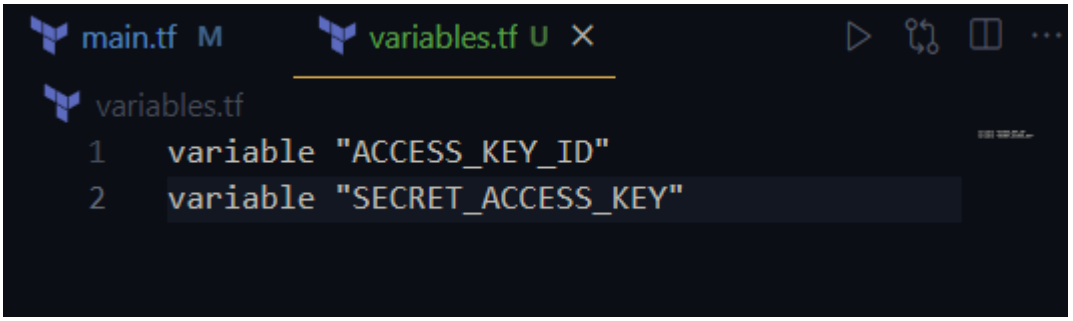
Esa clave la pegamos en las credenciales de Github junto con las que agregamos anteriormente.



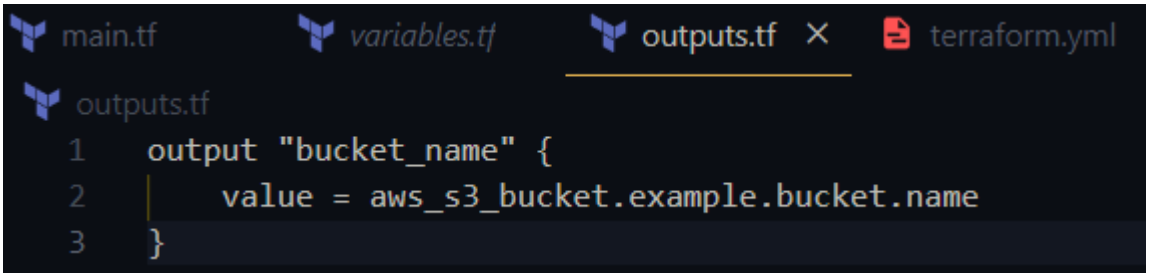
Seguimos con la creación del código de Terraform

1 - archivo main.tf

2 - archivo variables.tf



3 - outputs.tf



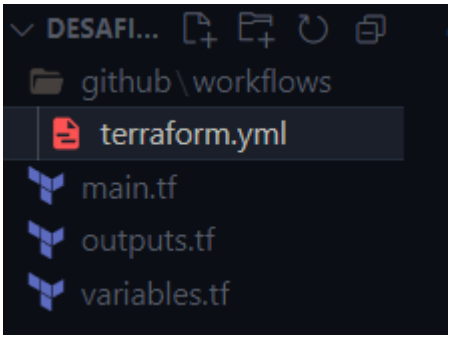
4 - terraform.yml

main.tf variables.tf outputs.tf terraform.yml

github > workflows > terraform.yml

```yaml
name: Provision t3.micro EC2

on:
  push:
    branches:
      - '**'  # Se ejecutará en cualquier rama con cualquier commit
  workflow_dispatch:
    inputs:
      ec2-name:
        description: EC2 name
        required: true
        default: 'App Server'
        type: string

jobs:
  provision-ec2:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - uses: actions/setup-node@v3
        with:
          node-version: '14'

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: '${{ secrets.AWS_ACCESS_KEY_ID }}'
          aws-secret-access-key: '${{ secrets.AWS_SECRET_ACCESS_KEY }}'
          aws-region: us-east-2

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2
        with:
          terraform_wrapper: false

      - name: Terraform Apply
        id: apply
        env:
```

nuestro codigo quedaria asi



una vez terminado de configurar todo y crear nuestro código en nuestra terminal corremos los siguientes codigos:

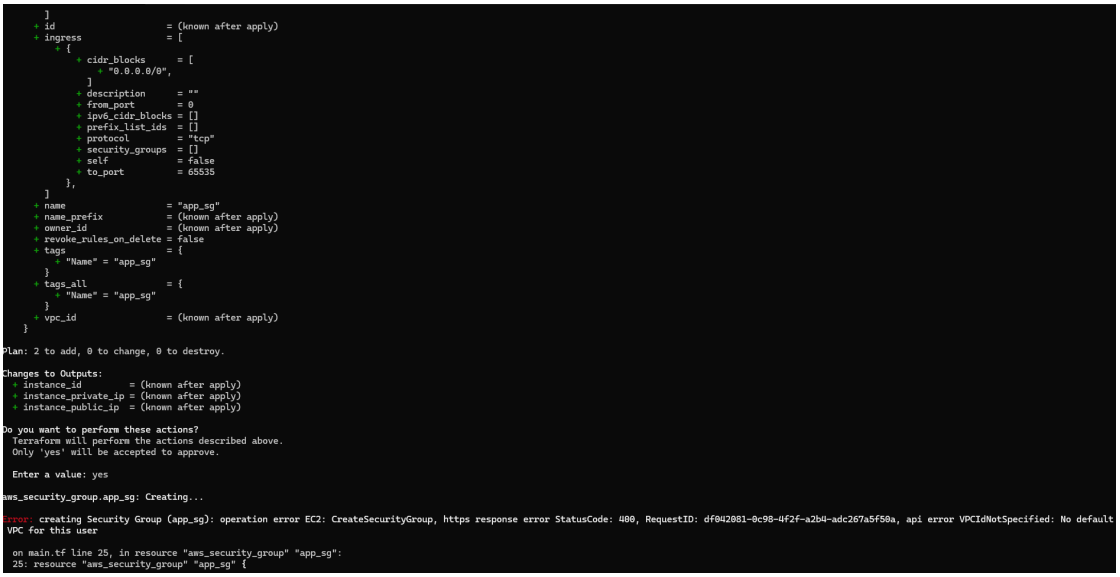
Terraform init

```
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.67.0...
- Installed hashicorp/aws v5.67.0 (self-signed, key ID 34365D9472D7468F)

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/plugins/signing.html

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* hashicorp/aws: version = "~> 5.67.0"

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.
```

Terraform validate

```
C:\Users\Michael\Desktop\Desafio-8>terraform validate
Success! The configuration is valid.
```

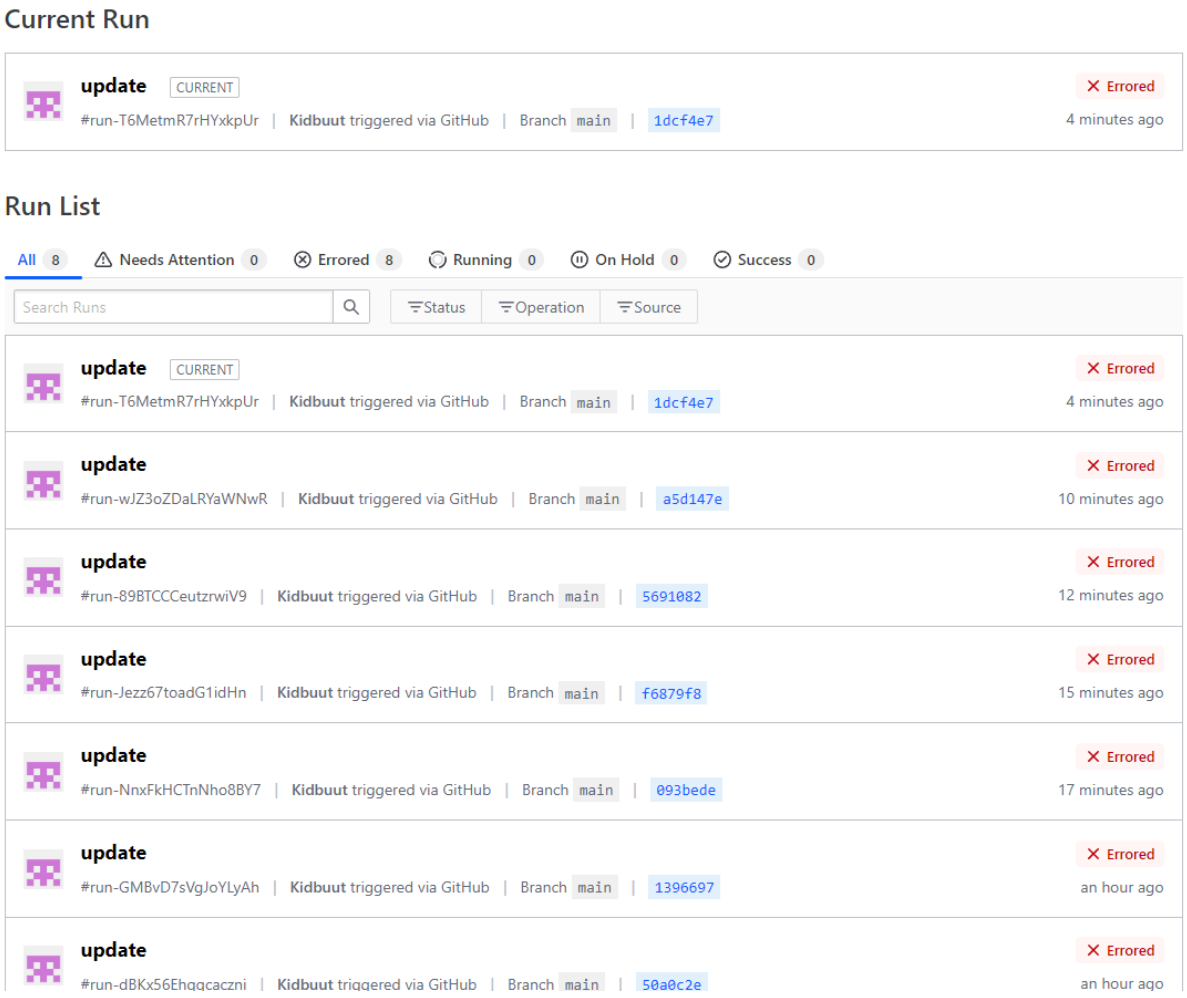
Despues de muchos intentos me dio infinidades de errores el cual no me corrio porque me decia que tenia un error con la VPC



```
    ]
  + id                     = (known after apply)
  + ingress                = [
      + {
          + cidr_blocks      = [
              + "0.0.0.0/0",
            ]
          + description      = ""
          + from_port        = 0
          + ipv6_cidr_blocks = []
          + prefix_list_ids  = []
          + protocol         = "tcp"
          + security_groups  = []
          + self             = false
          + to_port          = 65535
        },
    ]
  + name                   = "app_sg"
  + name_prefix            = (known after apply)
  + owner_id               = (known after apply)
  + revoke_rules_on_delete = false
  + tags                   = {
      + "Name" = "app_sg"
    }
  + tags_all               = {
      + "Name" = "app_sg"
    }
  + vpc_id                 = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + instance_id         = (known after apply)
  + instance_private_ip = (known after apply)
  + instance_public_ip  = (known after apply)

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

aws_security_group.app_sg: Creating...

Error: creating Security Group (app_sg): operation error EC2: CreateSecurityGroup, https response error StatusCode: 400, RequestID: df042081-0c98-4f2f-a2b4-adc267a5f50a, api error VPCIdNotSpecified: No default
VPC for this user

  on main.tf line 25, in resource "aws_security_group" "app_sg":
  25: resource "aws_security_group" "app_sg" {
```

busque informacion y ayuda por todos lados y no encontre nada, dejo capturas de cómo todo estaba bien configurado y me seguia dando el error.

```
17
18  resource "aws_security_group" "app_sg" {
19    vpc_id = "vpc-01d9d403a6d7a6b84"  # Reemplaza con tu VPC ID
20
```



el próximo desafío lo haré mejor.