

## TERRAFORM

### Comandos Básicos de Terraform

- **terraform init**: Prepara tu directorio de trabajo. ¡Siempre al iniciar!
- **terraform plan**: Muestra qué cambios hará Terraform. ¡Siempre antes de aplicar!
- **terraform apply**: Ejecuta los cambios propuestos. ¡Crea o modifica recursos!
- **terraform destroy**: Elimina los recursos gestionados. ¡Cuidado, borra todo!
- **terraform validate**: Comprueba la sintaxis de tus archivos. ¡Para errores rápidos!

### Documentacion Azure

<https://registry.terraform.io/>

### Grupo de recursos

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "4.16.0"
    }
  }
}

provider "azurerm" {
  features {}
  subscription_id = "30a83aff-7a8b-4ca3-aa48-ab93268b5a8b"
}

resource "azurerm_resource_group" "rg" {
  location = "mexicocentral"
  name    = "miPrimerGrupoPlatzi"
}
```

## Terraform Plan

terraform plan -out plan.out

Este comando **calcula qué cambios hará Terraform y guarda ese plan exacto en un archivo (plan.out)**.

Sirve para **revisar y compartir** los cambios antes de aplicarlos

Aplicar plan anterior:

terraform apply "plan.out"

## Variables

En el mismo archivo [main.tf](#)

Usa default para asignar un valor por defecto a la variable

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "4.16.0"
    }
  }
}

provider "azurerm" {
  features {}
  subscription_id = "30a83aff-7a8b-4ca3-aa48-ab93268b5a8b"
}

variable "rg_nombre" {
  type = string
  default = "miPrimerGrupoAmin"
}

resource "azurerm_resource_group" "rg" {
  location = "mexicocentral"
  name     = var.rg_nombre
}
```

## Configurar variable en otro archivo:

El archivo debe llamarse si o si [variables.tf](#)



```
variable "rg_nombre" {
  type = string
}
```

## Asignar valores a las variables

El archivo debe llamarse terraform.tfvars

Este archivo debe agregarse el gitignore porque puede tener información sensible



```
rg_nombre = "rg-terraform"
```

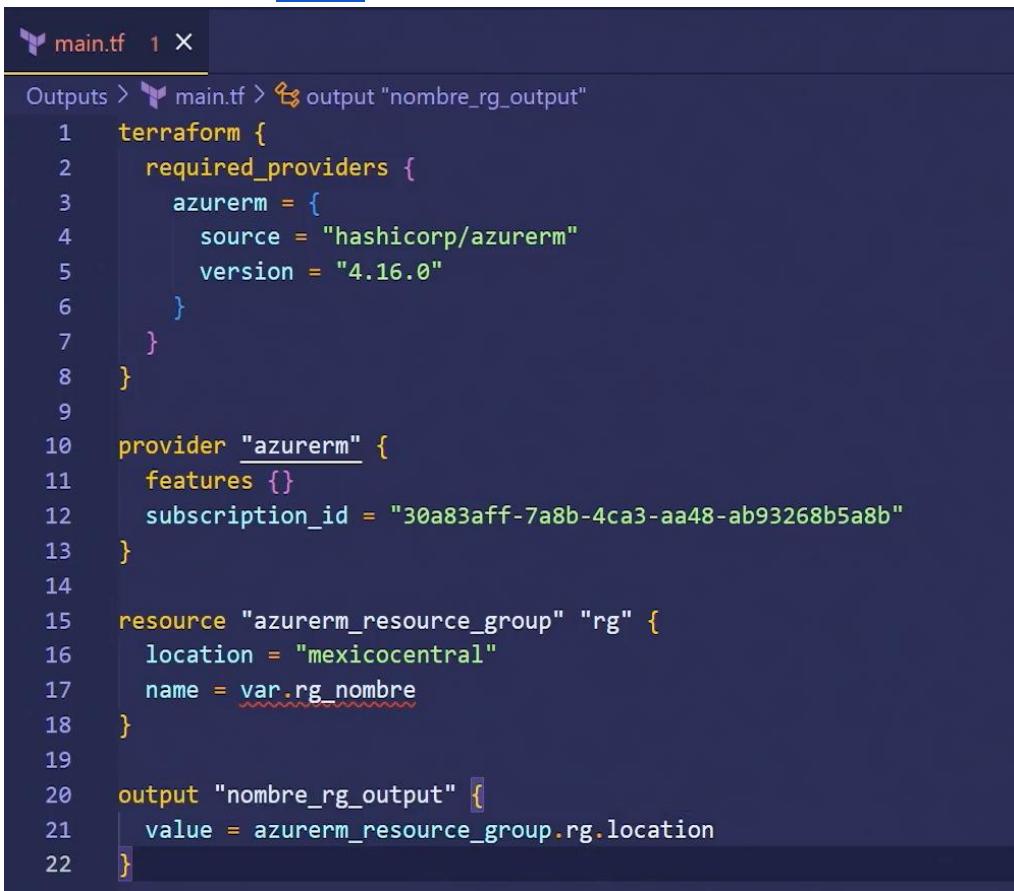
En cambio debe de crearse un archivo terraform.tfvars.example que si se puede subir a git



```
rg_nombre = "nombre del grupo de recursos"
```

## Outputs

En el mismo archivo [main.tf](#)

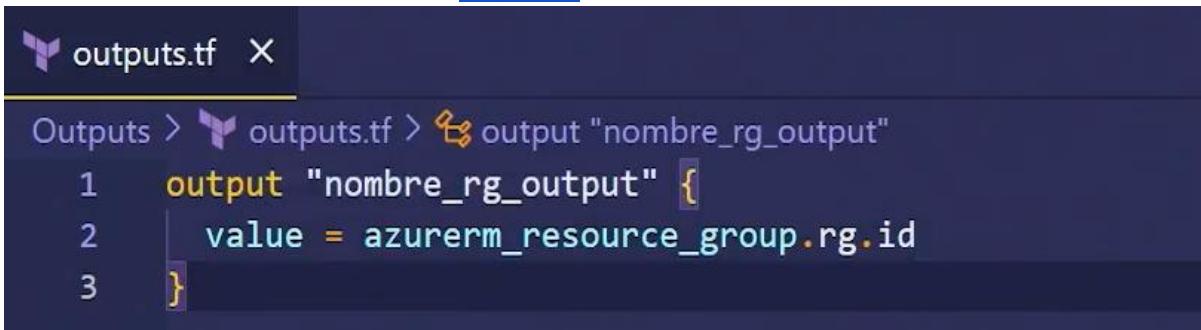


```
Outputs > main.tf > output "nombre_rg_output"
1  terraform {
2    required_providers {
3      azurerm = {
4        source  = "hashicorp/azurerm"
5        version = "4.16.0"
6      }
7    }
8  }
9
10 provider "azurerm" {
11   features {}
12   subscription_id = "30a83aff-7a8b-4ca3-aa48-ab93268b5a8b"
13 }
14
15 resource "azurerm_resource_group" "rg" {
16   location = "mexicocentral"
17   name     = var.rg_nombre
18 }
19
20 output "nombre_rg_output" {
21   value = azurerm_resource_group.rg.location
22 }
```

Este comando sirve para saber el valor de alguna variable del recurso

## Configurar output en otro archivo

Este archivo debe llamarse si o si ["outputs.tf"](#)



```
Outputs > outputs.tf > output "nombre_rg_output"
1  output "nombre_rg_output" {
2    value = azurerm_resource_group.rg.id
3  }
```

## Storage account (cuentas de almacenamiento)

```
provider "azurerm" {
  features {}
  subscription_id = "30a83aff-7a8b-4ca3-aa48-ab93268b5a8b"
}

resource "azurerm_resource_group" "rg" {
  location = "brazilsouth"
  name    = "grupoAlmacenamiento"
}

resource "azurerm_storage_account" "storage_account" {
  name          = "aminidevplatzi"
  resource_group_name = azurerm_resource_group.rg.name
  location      = azurerm_resource_group.rg.location
  account_tier   = "Standard"
  account_replication_type = "GRS"

  tags = {
    environment = "staging"
  }
}
```

Se pueden agregar nuevas propiedades que no aparecen por defecto

```
location = "brazilsouth"
account_tier = "Standard"
account_replication_type = "GRS"
public_network_access_enabled = false
```

## Obtener valores del Storage Account

```
output "cadenaConexion" {
  value = azurerm_storage_account.storage_account.primary_connection_string
}
```

Tener cuidado con valores sensibles como este que son de un access token

```
Planning failed. Terraform encountered an error while generating this plan.

Error: Output refers to sensitive values

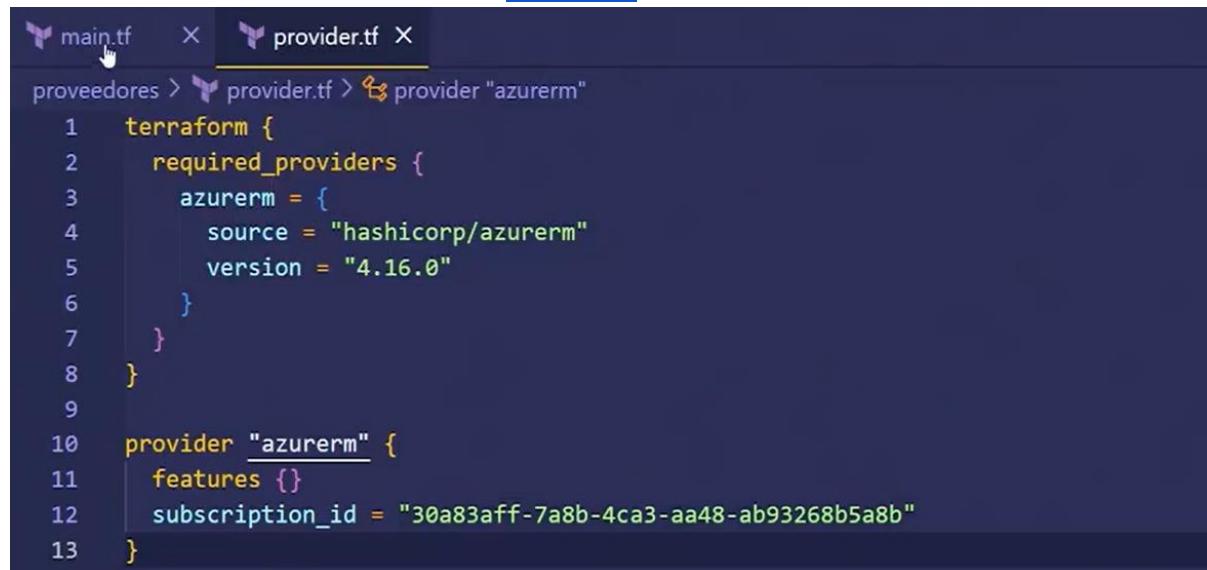
on main.tf line 33:
  33: output "cadenaConexion" {
```

Marcarlo como sensible

```
output "cadenaConexion" {
  value = azurerm_storage_account.storage_account.primary_connection_string
  sensitive = true
}
```

## Crear proveedor

Este archivo debe de llamarse si o si "[provider.tf](#)"



```
main.tf      provider.tf

proveedores > provider.tf > provider "azurerm"
1  terraform {
2    required_providers {
3      azurerm = {
4        source = "hashicorp/azurerm"
5        version = "4.16.0"
6      }
7    }
8  }
9
10 provider "azurerm" {
11   features {}
12   subscription_id = "30a83aff-7a8b-4ca3-aa48-ab93268b5a8b"
13 }
```

## Agregar proveedor Azure CAF

Sirve para colocar nombres a los recursos automáticamente y evitar que sea manual

```
provider "azurerm" {
  source  = "hashicorp/azurerm"
  version = "4.16.0"
}

provider "azurecaf" {
  source  = "aztfmod/azurecaf"
  version = "1.2.10"
}
```

Se debe de hacer un nuevo terraform init, al agregar un nuevo proveedor

Para cada recurso se tiene que crear un azurecaf

```
resource "azurerm_resource_group" "rg" {
  name     = "amines"
  location = "brazilsouth"
  tags = {
    environment = "staging"
  }
}

resource "azurerm_storage_account" "storage_account" {
  name                = "amines"
  resource_type       = "azurerm_storage_account"
  location            = "brazilsouth"
  account_tier        = "Standard"
  account_replication_type = "GRS"
  public_network_access_enabled = false
  tags = {
    environment = "staging"
  }
}
```

## Agregar alternativa de Azure CAF para otros proveedores

Kewin Daniel Guzman Diaz  
student • hace 4 meses

Segun tengo entendido en AWS no hay este proveedor pero si hay alternativas, como lo puede ser random\_string que también es muy bueno, yo lo uso en mi proyecto.

Su forma de implementar es sencilla

Aqui te dejo la doc: [random\\_string | Resources | hashicorp/random | Terraform | Terraform Registry](#)

Y mira un ejemplo:

```
resource "random_string" "suffix" {  
    length = 4  
    special = false  
    upper   = false  
}  
  
variable "env" {  
    default = "dev"  
}  
  
variable "project" {  
    default = "myapp"  
}  
  
resource "aws_s3_bucket" "example" {  
    bucket =  
        "rg-${var.env}-${var.project}-${random_string.suffix.result}"  
}
```

# Estado remoto en Azure

Para hacer esto es necesario crear un storage account manualmente

The screenshot shows the 'Create a storage account' wizard. It starts with a navigation bar: Home > Resource groups > EstadosTerraform > Marketplace > Storage account >. The main section is titled 'Create a storage account'. It has two tabs: 'Object details' (selected) and 'Advanced options'. A note says: 'Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.' The configuration fields are as follows:

- Subscription:** Azure Corp Account
- Resource group:** EstadosTerraform (with a 'Create new' link)
- Instance details:**
  - Storage account name:** estadoterraformamin
  - Region:** (South America) Brazil South (with a 'Deploy to an Azure Extended Zone' link)
  - Primary service:** Select a primary service (dropdown menu)
  - Performance:** Standard (radio button selected) - Recommended for most scenarios (general-purpose v2 account). Premium (radio button) - Recommended for scenarios that require low latency.
  - Redundancy:** Locally-redundant storage (LRS) (dropdown menu)

At the bottom are 'Previous', 'Next', and 'Review + create' buttons.

Primero hay que crear un contenedor

The screenshot shows the 'estadoterraformamin' storage account overview page. On the left, there's a sidebar with links like Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, Storage Mover, Partner solutions, and Data storage (Containers). The 'Containers' link is highlighted. The main area shows a list of containers with one entry: '\$logs'. On the right, a 'New container' panel is open, showing a form to create a new container. The 'Name' field contains 'states' with a red '2' highlight. The 'Anonymous access level' dropdown is set to 'Private (no anonymous access)'. A note at the bottom says: 'The access level is set to private because anonymous access is disabled on this storage account.' There's also an 'Advanced' section with a collapsed icon.

Ahora hay que crear el archivo que debe de llamarse si o si "[backend.tf](#)"

The screenshot shows a code editor with two tabs: 'main.tf' and 'backend.tf'. The 'backend.tf' tab is active and displays the following Terraform configuration:

```
1  terraform {  
2      backend "azurerm" {  
3          storage_account_name = "estadoterraformamin"  
4          container_name     = "states"  
5          key                = "estados.tfstate" ↗  
6      }  
7  }
```

A red arrow points to the 'key' parameter in the 'azurerm' block.

# Crear el SAS para el Storage Account

The screenshot shows the 'Shared access signature' configuration page in the Azure portal. It includes sections for Allowed services (Blob checked), Allowed resource types (Container checked, Object checked), Allowed permissions (Read, Write, Delete, List, Add, Create, Update, Process, Immutable storage, Permanent delete all checked), Blob versioning permissions (Enables deletion of versions checked), Allowed blob index permissions (Read/Write checked, Filter checked), and Start and expiry date/time fields (Start: 01/21/2025, End: 01/21/2025, Expiry time: 8:54:51 PM). A red box highlights the 'Shared access signature' link in the sidebar.

Copiar el valor

The screenshot shows the generated SAS token and URLs. It includes a 'Generate SAS and connection string' button, a 'Connection string' field with the URL `BlobEndpoint=https://estadoterraformamin.blob.core.windows.net/;QueueEndpoint=https://estadoterraformamin.queue.core.windows.net/;FileEndpo`, a 'SAS token' field with the value `sv=2022-11-02&ss=b&srt=co&sp=rwdlaciytfx&se=2025-01-29T01:54:51Z&st=2025-01-21T17:54:51Z&spr=https&sig=6Hm3RULMntxJyJD8JDRVO60ex`, and a 'Blob service SAS URL' field with the URL `https://estadoterraformamin.blob.core.windows.net/?sv=2022-11-02&ss=b&srt=co&sp=rwdlaciytfx&se=2025-01-29T01:54:51Z&st=2025-01-21T17:54:51Z&spr=https&sig=6Hm3RULMntxJyJD8JDRVO60ex`.

Ejecutar el terraform init de la siguiente manera incluyendo el backend

```
amineospinoza@CompuAmin:/mnt/c/Users/amine/Documents/Github/curso-terraform/estadoRemoto$ terraform init -backend-config="sas_token=sv=2022-11-02&ss=b&srt=co&sp=rwdlaciytfx&se=2025-01-29T01:54:51Z&st=2025-01-21T17:54:51Z&spr=https&sig=6Hm3RULMntxJyJD8JDRVO60exOR78EUACVdp6KimWB8%3D"
Initializing the backend...
```

Despues debe aplicar los siguientes comandos

```
terraform plan -out plan.out
```

```
terraform apply "plan.out"
```

A partir de crear este estado en Azure, ya no existirá el `terraform.tfstate` en local sino estará en la nube, el cual esta dentro del contenedor antes creado.

The screenshot shows the 'Containers' section of the Azure Storage account. It lists a single blob named 'estados.tfstate'. The blob details show it was modified on 1/21/2025 at 12:59:39, has an 'Hot (Inferred)' access tier, is a 'Block blob', and has a size of 12.49 KB. The 'Lease state' is 'Available'.

Importante: algunos usuarios reportan que no es necesario el SAS

## Módulo Data (no terraform)

Usar recursos que ya han sido creados anteriormente en Azure sin terraform

```
main.tf
data > main.tf > resource "azurerm_storage_account" "storage_account"
1   data "azurerm_resource_group" "imported_rg" {
2     name = "GrupoPrevio"
3   }
4
5   resource "azurerm_storage_account" "storage_account" {
6     name = "almacenamientoprevio"
7     resource_group_name = data.azurerm_resource_group.imported_rg.name
8     location = data.azurerm_resource_group.imported_rg.location
9     account_tier      = "Standard"
10    account_replication_type = "GRS"
11  }
```

A red arrow points to the opening brace of the `imported\_rg` data block at line 1.

Para proteger un recurso existente en producción y evitar que sea eliminado y recreado al ejecutar terraform apply

```
resource "aws_instance" "example" {
  ami           = "ami-12345678"
  instance_type = "t2.micro"

  lifecycle {
    prevent_destroy = true
  }
}
```

## Comandos de terraform

terraform fmt –recursive (formatear el texto, para ordenarlo)

terraform validate (validaciones básicas antes de ejecutar el plan)

# Máquinas virtuales (VM)

## Elementos a crear

- Grupo de recursos
- Red virtual
- Subred
- IP Pública
- Grupo de seguridad
- Interfaz de red
- Máquina virtual

```
resource "azurerm_resource_group" "rg" {
  location = var.location
  name     = var.resource_group_name
}

resource "azurerm_virtual_network" "my_terraform_network" {
  name          = "amines-vnet"
  address_space = ["10.0.0.0/16"]
  location      = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_subnet" "my_terraform_subnet" {
  name          = "amines-subnet"
  resource_group_name = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.my_terraform_network.name
  address_prefixes = ["10.0.1.0/24"]
}

resource "azurerm_public_ip" "my_terraform_public_ip" {
  name          = "amines-public-ip"
  location      = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  allocation_method = "Static"
}

resource "azurerm_network_security_group" "my_terraform_nsg" {
  name          = "amines-nsg"
  location      = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  security_rule {
    name          = "RDP"
    priority      = 1000
    direction     = "Inbound"
    access        = "Allow"
    protocol      = "*"
    source_port_range = "*"
    destination_port_range = "3389"
    source_address_prefix = "*"
    destination_address_prefix = "*"
  }
}
```

```

resource "azurerm_network_interface" "my_terraform_nic" {
  name          = "amines-nic"
  location      = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  ip_configuration {
    name          = "my_nic_configuration"
    subnet_id     = azurerm_subnet.my_terraform_subnet.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id       = azurerm_public_ip.my_terraform_public_ip.id
  }
}

resource "azurerm_network_interface_security_group_association" "nic_association" {
  network_interface_id      = azurerm_network_interface.my_terraform_nic.id
  network_security_group_id = azurerm_network_security_group.my_terraform_nsg.id
}

resource "azurerm_windows_virtual_machine" "main" {
  name          = "amines-vm"
  admin_username = var.vm_username
  admin_password = var.vm_password
  location      = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  network_interface_ids = [azurerm_network_interface.my_terraform_nic.id]
  size          = "Standard_DS1_v2"

  os_disk {
    name          = "myOsDisk"
    caching       = "ReadWrite"
    storage_account_type = "Premium_LRS"
  }

  source_image_reference {
    publisher = "MicrosoftWindowsServer"
    offer     = "WindowsServer"
    sku       = "2022-datacenter-azure-edition"
    version   = "latest"
  }
}

```

## Variables agregadas

```
maquinasVirtuales > variables.tf > variable "vm_password"
1   variable "location" {
2     type = string
3   }
4
5   variable "resource_group_name" {
6     type = string
7   }
8
9   variable "vm_username" {
10    type = string
11  }
12
13  variable "vm_password" {
14    type = string
15 }
```

## Valores para las variables

```
maquinasVirtuales > terraform.tfvars > vm_password
1   location = "eastus2"
2   resource_group_name = "maquinasVirtualesGroup"
3   vm_username = "aminespinoza"
4   vm_password = "Am0_Apr3nd3r$_"
```

Ver el código completo aquí

<https://github.com/platzi/curso-terraform/blob/main/maquinasVirtuales/main.tf>

## Instalar IIS en la VM

El código completo esta en la ruta anterior de git

```
resource "azurerm_virtual_machine_extension" "web_server_install" {
  name                  = "amines-wsi"
  virtual_machine_id    = azurerm_windows_virtual_machine.main.id
  publisher             = "Microsoft.Compute"
  type                 = "CustomScriptExtension"
  type_handler_version = "1.8"
  auto_upgrade_minor_version = true
  settings = <<SETTINGS
  {
    "commandToExecute": "powershell -ExecutionPolicy Unrestricted Install-WindowsFeature -Name IIS-WebServerRole -IncludeManagementTools"
  }
  SETTINGS
}
```

## Comunicación entre módulos en Terraform

La idea de esto es compatir variables o información entre dos módulos (carpeta de vscode), ya una podría estar creando un recurso que necesita la otra carpeta, para este caso se está creando una VM y en otra carpeta la SubNet, para esto es necesario obtener a través de un output los valores resultantes de la SubNet para pasarlo a la VM.

**IMPORTANTE:** como se ve en la imagen todos estas carpetas pertenecen a una general llamada “módulos” en la cual también tiene sus archivos de terraform.

The screenshot shows the VS Code interface with the following details:

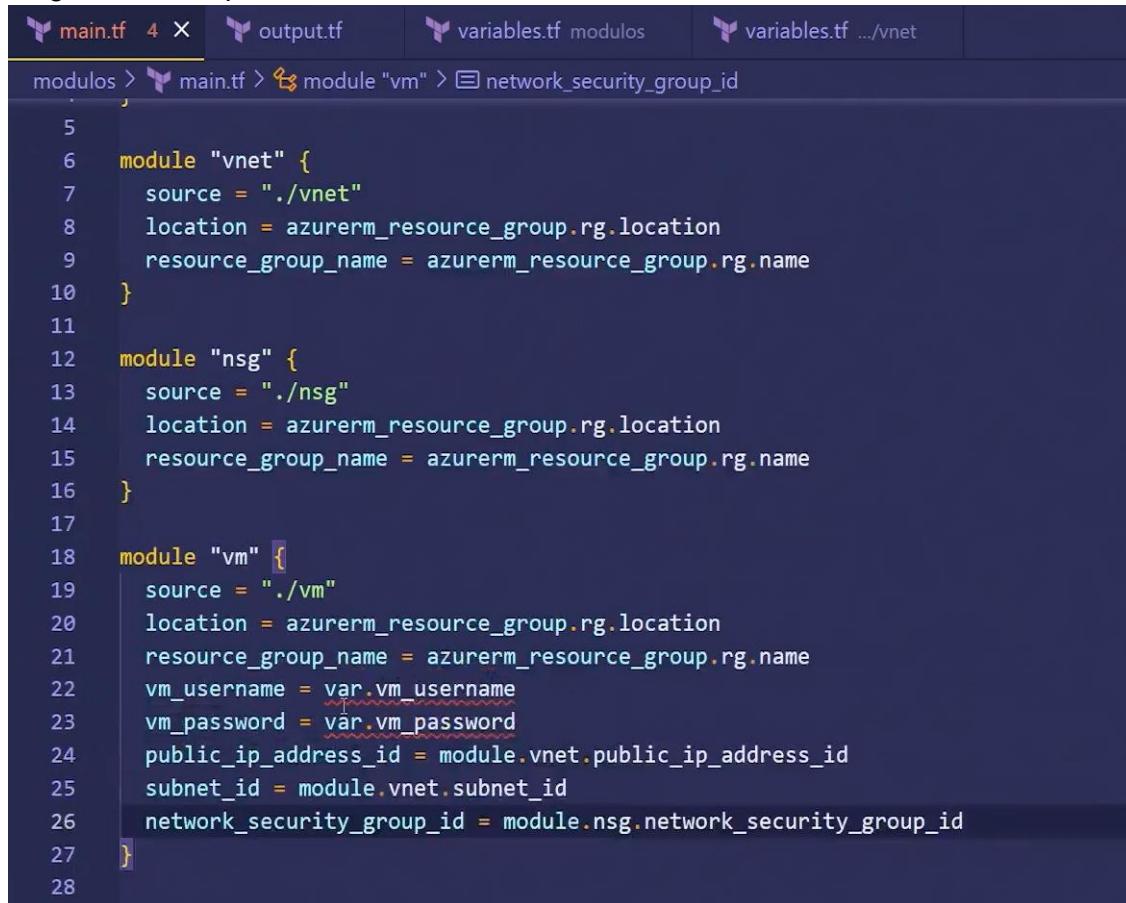
- EXPLORER:** Shows the project structure:
  - CURSO-TERRAFORM [WSL: UBU...]**:
    - almacenamiento
    - comandosBasicos
    - data
    - estadoRemoto
    - maquinasVirtuales
    - modulos**:
      - nsg
      - vm**:
        - main.tf
        - output.tf
        - variables.tf
    - vnet- OUTPUT:** Shows the code for the "vnet" module's "output.tf" file:

```
1  output "subnet_id" {
2    value = azurerm_subnet.my_terraform_subnet.id
3  }
4
5  output "public_ip_address_id" {
6    value = azurerm_public_ip.my_terraform_public_ip.id
7  }
```

Red annotations are present: a red arrow points to the "vm" folder in the Explorer; a red number "2" is next to the "vnet" folder in the Explorer; and a red number "3" is next to the "public\_ip\_address\_id" output line in the code.
- FILES:** Shows other files in the "vnet" folder: main.tf, provider.tf, terraform.tfvars, and variables.tf.

**IMPORTANTE:** Recordar que el nombre de los outputs, debe llamarse de igual manera que las variables que se van usar, es decir en el módulo de **vnet** el output `subnet_id` se llama igual que la variable del módulo de **vm** (`subnet_id`).

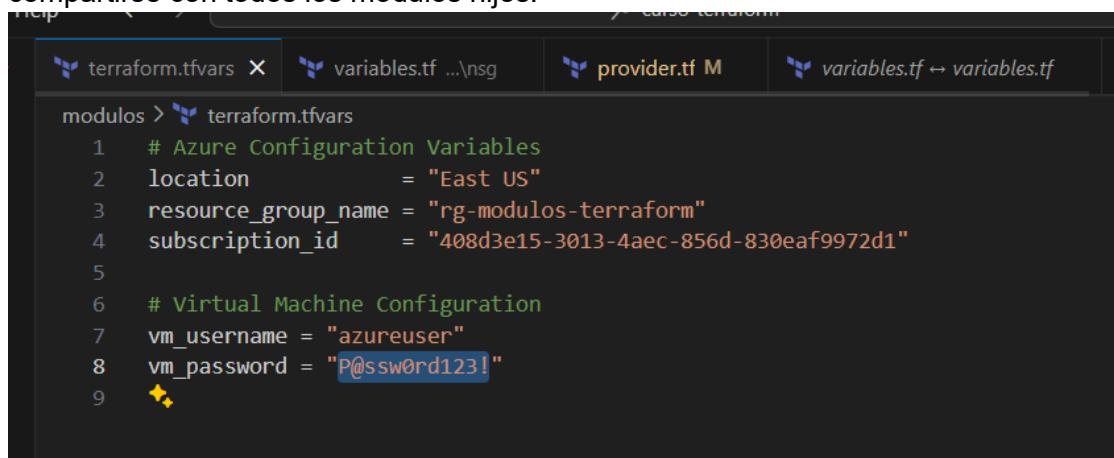
En la carpeta [main.tf](#) de la carpeta “módulos” vamos a crear la sección de módulos cargando las carpetas contenidas



```
modulos > main.tf > module "vnet" > module "nsg" > module "vm"
.
.
.
5
6 module "vnet" {
7   source = "./vnet"
8   location = azurerm_resource_group.rg.location
9   resource_group_name = azurerm_resource_group.rg.name
10 }
11
12 module "nsg" {
13   source = "./nsg"
14   location = azurerm_resource_group.rg.location
15   resource_group_name = azurerm_resource_group.rg.name
16 }
17
18 module "vm" {
19   source = "./vm"
20   location = azurerm_resource_group.rg.location
21   resource_group_name = azurerm_resource_group.rg.name
22   vm_username = var.vm_username
23   vm_password = var.vm_password
24   public_ip_address_id = module.vnet.public_ip_address_id
25   subnet_id = module.vnet.subnet_id
26   network_security_group_id = module.nsg.network_security_group_id
27 }
28 }
```

Luego ejecutar los comandos de terraform, si se agregan nuevos módulos se tiene que ejecutar nuevamente terraform init

El archivo con los valores de las variables puede estar en el módulo general, y este compartirse con todos los módulos hijos.



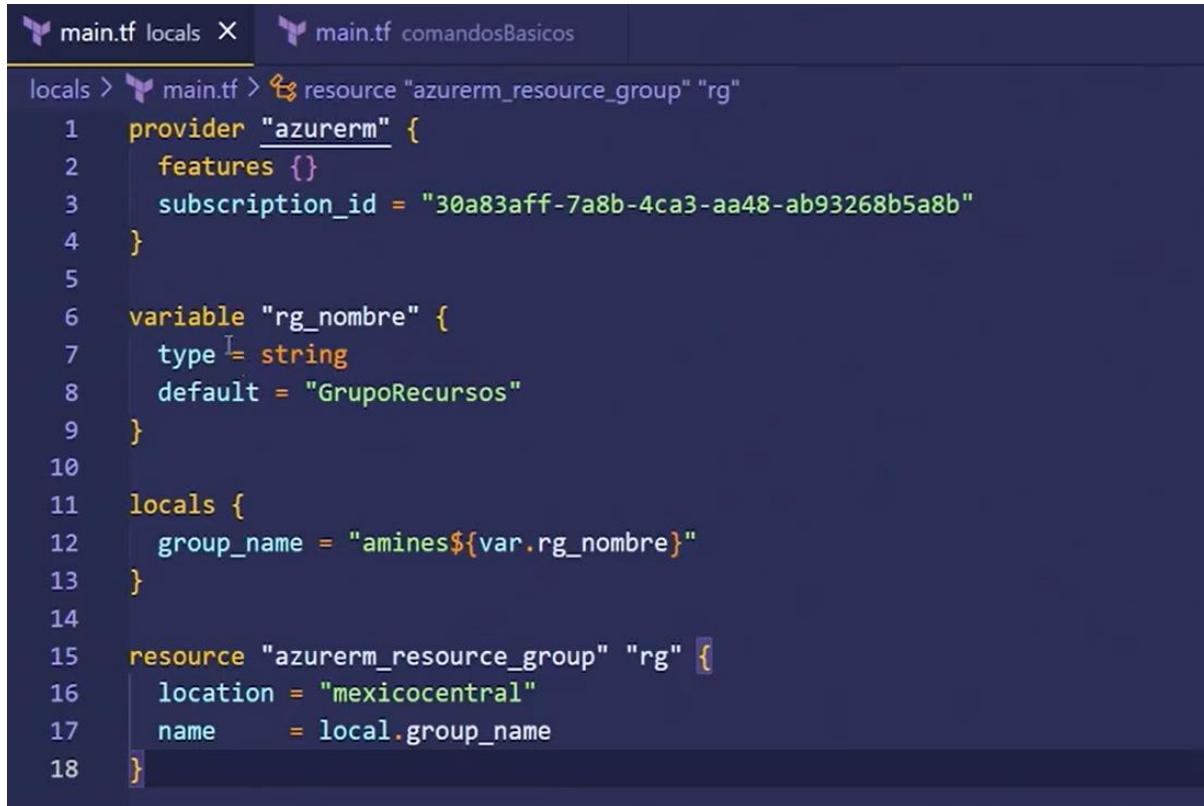
```
modulos > terraform.tfvars
.
.
.
1 # Azure Configuration Variables
2 location          = "East US"
3 resource_group_name = "rg-modulos-terraform"
4 subscription_id    = "408d3e15-3013-4aec-856d-830eaf9972d1"
5
6 # Virtual Machine Configuration
7 vm_username = "azureuser"
8 vm_password = "P@ssw0rd123"
9
```

## Locals

Le agrega un prefijo a nuestras variables es decir si:

- Local: amines
- Resource Group: GrupoPlatzi
- Resultante: aminesGrupoPlatzi

Se debe de definir un local por cada recurso.

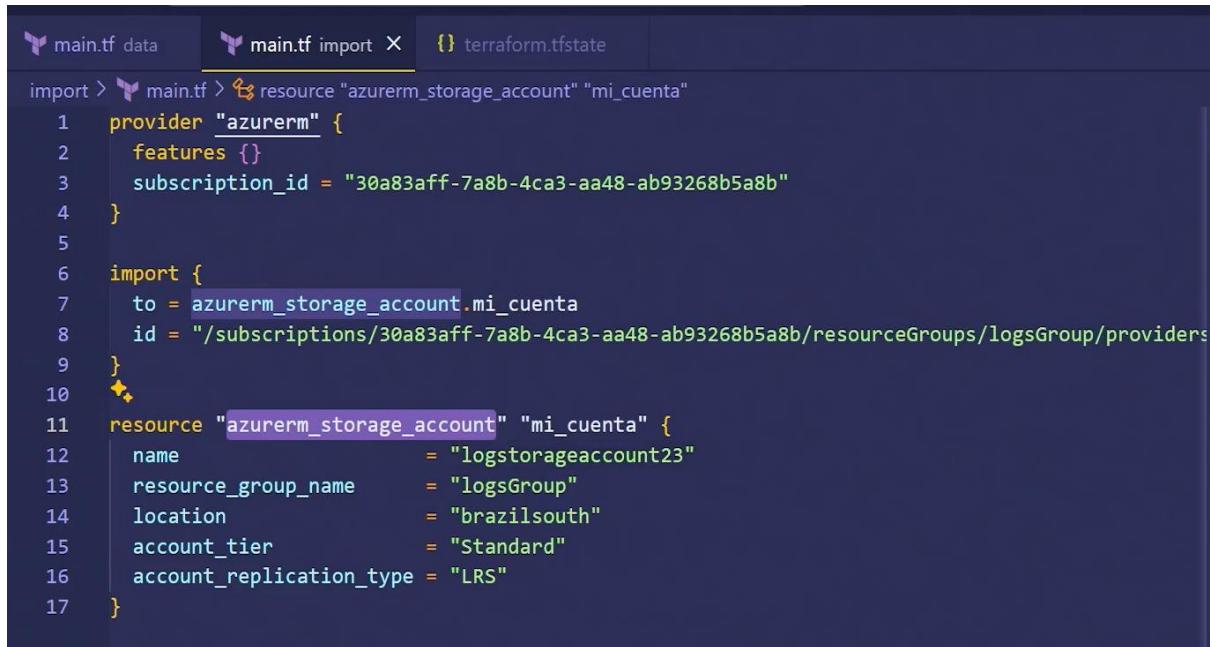


```
locals > main.tf > resource "azurerm_resource_group" "rg"
1 provider "azurerm" {
2   features {}
3   subscription_id = "30a83aff-7a8b-4ca3-aa48-ab93268b5a8b"
4 }
5
6 variable "rg_nombre" {
7   type = string
8   default = "GrupoRecursos"
9 }
10
11 locals {
12   group_name = "amines${var.rg_nombre}"
13 }
14
15 resource "azurerm_resource_group" "rg" {
16   location = "mexicocentral"
17   name     = local.group_name
18 }
```

## Importar recursos en terraform con Import

Para importar un recurso de la nube (Azure) es necesario obtener su id, como se muestra en la imagen, una forma fácil de obtener este id es ejecutar terraform apply y en el mensaje de error te dirá cual es este id.

Esta acción descarga todas las propiedades del recurso en local y además lo agrega en el state para poder modificarlo a voluntad usando terraform



```
import > main.tf > azurerm_storage_account "mi_cuenta"
1 provider "azurerm" {
2   features {}
3   subscription_id = "30a83aff-7a8b-4ca3-aa48-ab93268b5a8b"
4 }
5
6 import {
7   to = azurerm_storage_account.mi_cuenta
8   id = "/subscriptions/30a83aff-7a8b-4ca3-aa48-ab93268b5a8b/resourceGroups/logsGroup/providers"
9 }
10
11 resource "azurerm_storage_account" "mi_cuenta" {
12   name          = "logstorageaccount23"
13   resource_group_name = "logsGroup"
14   location      = "brazilsouth"
15   account_tier    = "Standard"
16   account_replication_type = "LRS"
17 }
```

## Diferencias entre Import y Data en terraform

Data: solo permite la lectura del recurso y obtener variables de ella, pero lo agrega al state de forma limitada y no permite modificarlo.

Import: Permite la modificación del recurso y lo agrega al state para ser alterado.

Clase 28: Diferencias entre import y data		
Monday, January 20, 2025	8:42 PM	
Diferencias Clave	import	data
Característica	import	data
Propósito	Asociar recursos existentes con Terraform	Ler información de recursos sin gestionarlos
Crea archivos de configuración	No	No
Afecta el estado	Sí (agrega el recurso al estado)	No (solo consulta datos)
Requiere definición previa en código	Sí	Sí
Control del recurso	Sí (después de importarlo)	No (solo lectura)

## Publicar módulos en Terraform

Para esto es necesario crear un repositorio en github de manera pública con la siguiente nomenclatura: **terraform-azurerm-(nombre del modulo)**

En este archivo se debe incluir los archivos de terraform como: [main.tf](#), [output.tf](#), [variables.tf](#), también considerar el Readme y la License .

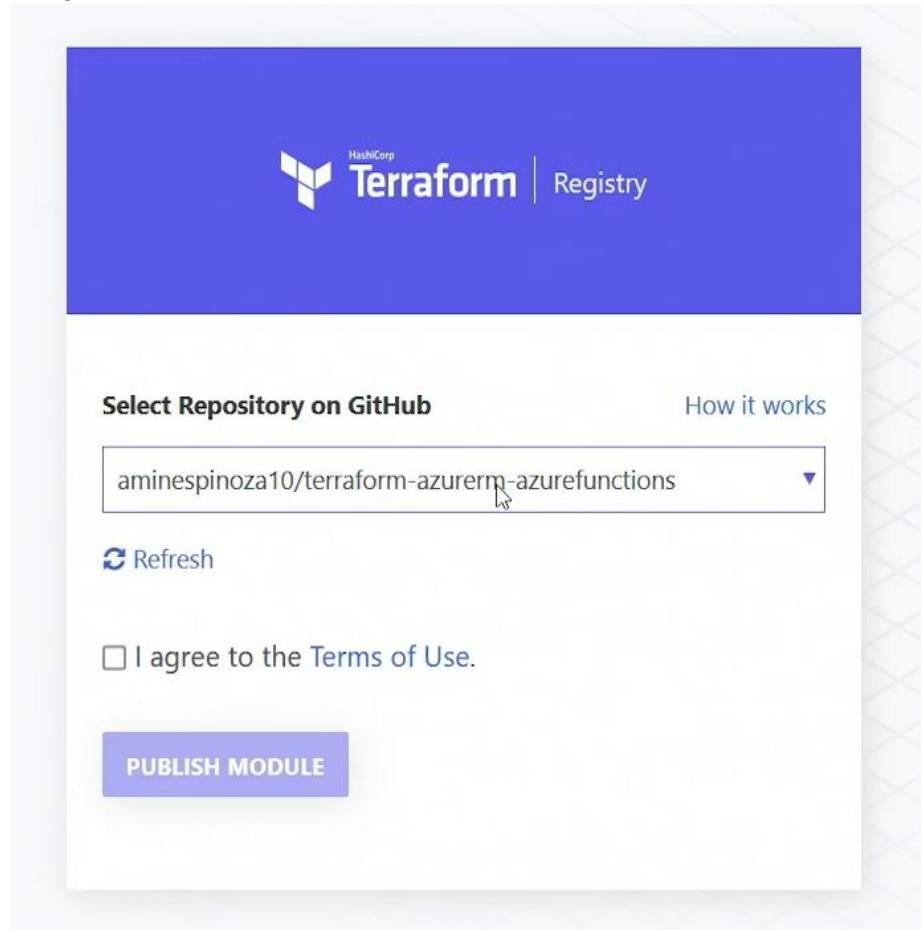
Al subirlo a github se debe crear una release

The screenshot shows the GitHub 'New Release' interface for a repository named 'terraform-azurerm-azurefunctions'. The title of the release is 'First version of my module'. The 'Write' tab is selected, showing a rich text editor with a placeholder: 'This is the first version of my Terraform module to deploy an Azure Function!'. Below the editor, there's a note about Markdown support and a file upload area. At the bottom, there's a checkbox for 'Set as a pre-release' and two buttons: 'Publish release' (highlighted in green) and 'Save draft'.

Luego dirigirse a terraform registry para subir el modulo

The screenshot shows the Terraform Registry interface. A search bar at the top finds 'azurerm'. On the left, there's a provider card for 'azurerm' with a provider icon, a provider name, and a provider URL. To the right, a dropdown menu is open under 'Publish' with 'Module' selected. A sidebar on the right displays download statistics: 'All versions' (21), 'Downloads this month' (63), 'Downloads this year' (63), and 'Downloads over all time' (4,212).

Luego deberá seleccionar el nombre del repositorio a subir



## CI/CD con Terraform y Github Actions

Es necesario crear primero un permiso para github action pueda manipular Azure

```
az ad sp create-for-rbac --name "terraformActions" --role contributor --scopes  
"/subscriptions/408d3e15-3013-4aec-856d-830eaf9972d1"
```

Se obtiene lo siguiente:

```
{  
  "appId": "ea7d02b9-c02f-40ec-9b18-29dd4ec35462",  
  "displayName": "terraformActions",  
  "password": "0p_8Q~TR3TaeYnOPoSFU542D7t1g2_ghx3xRgccP",  
  "tenant": "36df6968-506c-497e-b068-4b0c71cdada9"  
}
```

A partir de estos datos hay que dirigirse a Github

Settings -> Security -> Secrets and variables -> Actions -> Repository secrets -> New repository secret

CLIENT\_ID -> appId

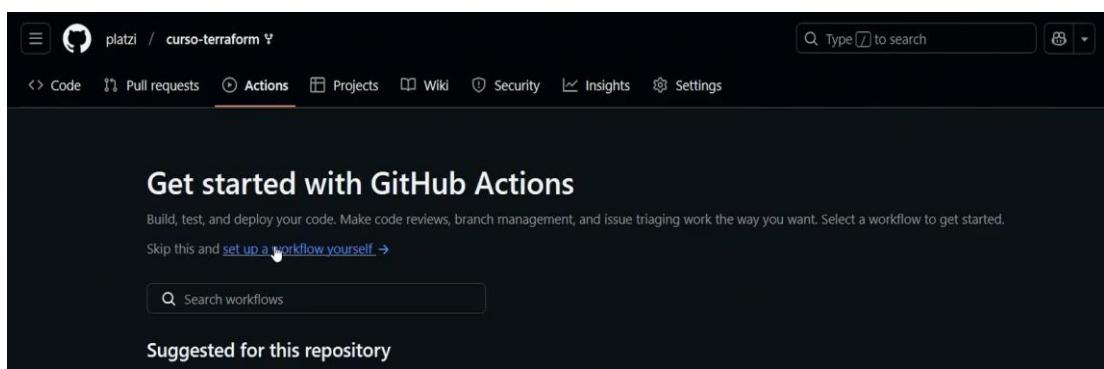
CLIENT\_SECRET -> password

SUBSCRIPTION\_ID -> subscription\_id

TENANT\_ID -> tenant

## Dirigirse a Actions

### Set up a workflow yourself



The screenshot shows the GitHub Actions setup interface. At the top, there's a navigation bar with links for Code, Pull requests, Actions (which is underlined), Projects, Wiki, Security, Insights, and Settings. Below the navigation is a search bar with placeholder text 'Type / to search'. The main content area has a dark background with white text. It features a heading 'Get started with GitHub Actions' followed by the subtext 'Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.' There's a link 'Skip this and set up a workflow yourself →'. Below this is a search bar labeled 'Search workflows'. At the bottom of the main content area, it says 'Suggested for this repository'.

### Crear el archivo de despliegue (terraformDeploy.yml)

```
name: "Terraform deploy"

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  terraform:
    env:
      ARM_CLIENT_ID: ${{ secrets.CLIENT_ID }}
      ARM_CLIENT_SECRET: ${{ secrets.CLIENT_SECRET }}
      ARM_SUBSCRIPTION_ID: ${{ secrets.SUBSCRIPTION_ID }}
      ARM_TENANT_ID: ${{ secrets.TENANT_ID }}
    name: 'Terraform'
    runs-on: ubuntu-latest
    defaults:
      run:
        working-directory: ${{ github.workspace }}/cicd
    steps:
      - uses: actions/checkout@v2
      - uses: hashicorp/setup-terraform@v1
```

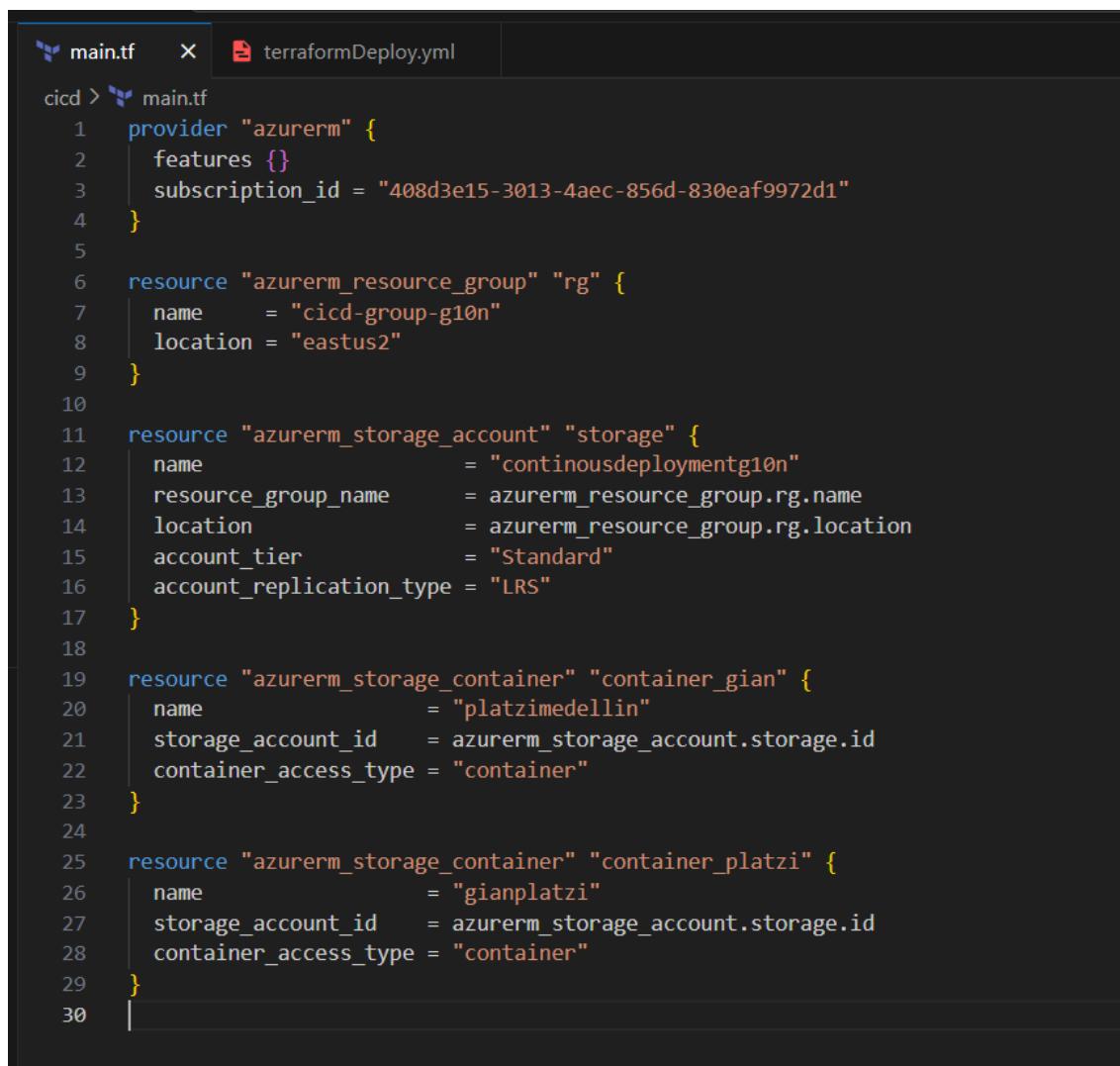
```
- name: Terraform init
  id: init
  run: terraform init

- name: Terraform validate
  id: validate
  run: terraform validate -no-color

- name: Terraform plan
  id: plan
  run: terraform plan -out plan.out

- name: Terraform apply
  id: apply
  run: terraform apply plan.out
```

## Archivo terraform para desplegar

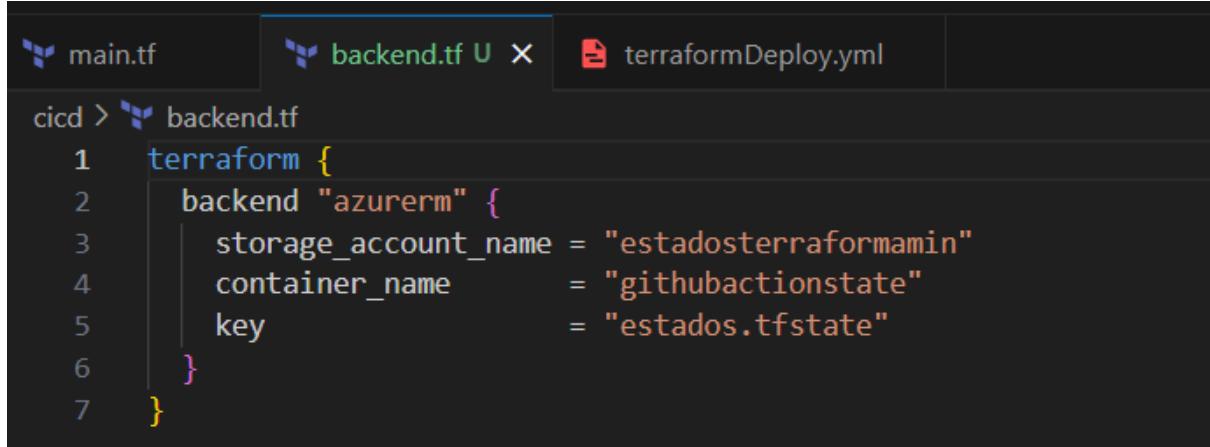


The screenshot shows a code editor with two tabs: 'main.tf' and 'terraformDeploy.yml'. The 'main.tf' tab is active, displaying Terraform configuration code. The 'terraformDeploy.yml' tab is visible in the background.

```
cicd > main.tf
1 provider "azurerm" {
2   features {}
3   subscription_id = "408d3e15-3013-4aec-856d-830eaf9972d1"
4 }
5
6 resource "azurerm_resource_group" "rg" {
7   name      = "cicd-group-g10n"
8   location  = "eastus2"
9 }
10
11 resource "azurerm_storage_account" "storage" {
12   name          = "continousdeploymentg10n"
13   resource_group_name = azurerm_resource_group.rg.name
14   location      = azurerm_resource_group.rg.location
15   account_tier    = "Standard"
16   account_replication_type = "LRS"
17 }
18
19 resource "azurerm_storage_container" "container_gian" {
20   name        = "platzimedellin"
21   storage_account_id = azurerm_storage_account.storage.id
22   container_access_type = "container"
23 }
24
25 resource "azurerm_storage_container" "container_platzi" {
26   name        = "gianplatzi"
27   storage_account_id = azurerm_storage_account.storage.id
28   container_access_type = "container"
29 }
30
```

# Estado Terraform con Github Actions

Crear archivo [backend.tf](#)



```
main.tf          backend.tf U X  terraformDeploy.yml
cicd > cicd > backend.tf
1  terraform {
2    backend "azurerm" {
3      storage_account_name = "estadoterraformamin"
4      container_name       = "githubactionstate"
5      key                  = "estados.tfstate"
6    }
7 }
```

Crear storage account

[Home](#) > [estado\\_terraform](#) > [Marketplace](#) > [Storage account](#) >

## Create a storage account

### Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription \*

Resource group \*  [Create new](#)

### Instance details

Storage account name \*

Region \*  [Deploy to an Azure Extended Zone](#)

Primary service

Performance \*  **Standard:** Recommended for most scenarios (general-purpose v2 account)

**Premium:** Recommended for scenarios that require low latency.

Redundancy \*

## Crear contenedor

The screenshot shows the 'Containers' section of the Azure Storage account 'estadoterraformgian'. The left sidebar includes links for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, Storage Mover, Partner solutions, Resource visualizer, Data storage (Containers, File shares, Queues, Tables), and Shared access signature. The main area displays a table with two items:

Name	Last modified
\$logs	18/08/2025, 7:16:26
githubactionstate	18/08/2025, 7:17:05

## Crear un SAS token (cambiar la fecha End)

The screenshot shows the 'Shared access signature' section of the Azure Storage account 'estadoterraformgian'. The left sidebar includes links for Partner solutions, Resource visualizer, Data storage (Containers, File shares, Queues, Tables), Security + networking (Networking, Front Door and CDN, Access keys), Shared access signature (Encryption, Microsoft Defender for Cloud), Data management, Settings, and Monitoring. The main area allows configuration of a SAS token with the following settings:

- Allowed services: Blob (checked), File, Queue, Table.
- Allowed resource types: Service, Container (checked), Object (checked).
- Allowed permissions: Read (checked), Write (checked), Delete (checked), List (checked), Add (checked), Create (checked), Update (unchecked), Process (checked), Immutable storage (checked), Permanent delete (checked).
- Blob versioning permissions: Enables deletion of versions (checked).
- Allowed blob index permissions: Read/Write (checked), Filter (checked).
- Start and expiry date/time:
  - Start: 08/18/2025, 7:03:01 AM
  - End: 08/25/2025, 3:18:01 PM

## Guardar y copiar el SAS token

The screenshot shows the 'Generate SAS and connection string' section of the Azure Storage account 'estadoterraformgian'. The left sidebar includes links for Security + networking (Networking, Front Door and CDN, Access keys), Shared access signature (Encryption, Microsoft Defender for Cloud), Data management, Settings, and Monitoring. The main area displays the generated SAS token and connection string:

Some routing options are disabled because the endpoints are not provisioned.

**Signing key:** key1

**Generate SAS and connection string**

**Connection string:** BlobEndpoint=https://estadoterraformgian.blob.core.windows.net/;QueueEndpoint=https://estadoterraformgian.queue.core.windows.net/;FileEndpoint=https://estadoterraformgian.file.core.windows.net...  
SAS token: sv=2024-11-04&ss=b&srt=co&sp=rwldlaciyfxf&se=2025-08-25T13:18:01Z&st=2025-08-18T05:03:01Z&spr=https&sig=OW2gpTUO19U8IDweKewMFz5fzoZgxNj6se%2FRRjsvZQ%3D

**Blob service SAS URL:** https://estadoterraformgian.blob.core.windows.net/?sv=2024-11-04&ss=b&srt=co&sp=rwldlaciyfxf&se=2025-08-25T13:18:01Z&st=2025-08-18T05:03:01Z&spr=https&sig=OW2gpTUO19U8IDweKewMFz...

## Agregar SAS token en los secretos de Github como se hizo anteriormente

The screenshot shows the 'Repository secrets' page in GitHub. On the left, there's a sidebar with 'Secrets and variables' selected. The main area lists several secrets: 'CLIENT\_ID', 'CLIENT\_SECRET', 'SAS\_TOKEN' (which has a red arrow pointing to it), 'SUBSCRIPTION\_ID', and 'TENANT\_ID'. Each secret has columns for 'Name', 'Last updated', and edit/delete icons.

Agregar la siguiente línea en el archivo yml

```
20 |   working-directory: ${{ github.workspace }}/cicd
21 |   steps:
22 |     - uses: actions/checkout@v2
23 |     - uses: hashicorp/setup-terraform@v1
24 |
25 |     - name: Terraform init
26 |       id: init
27+ |       run: terraform init -backend-config="sas_token=${{ secrets.SAS_TOKEN }}"
28+
29 |     - name: Terraform validate
30 |       id: validate
```

Hacer commit de los cambios y verificar que el estado se guarde en Azure

The screenshot shows the Microsoft Azure Storage Explorer interface. It's connected to a container named 'githubactionstate'. Inside, there's a single blob named 'estados.tfstate'. The interface includes standard file operations like Add Directory, Upload, Change access level, Refresh, Delete, Copy, Paste, Rename, Acquire lease, Break lease, and Edit columns.

Si necesitas autenticarte en Azure desde la terminal y el navegador no se abre automáticamente, ¿qué comando alternativo podrías usar?

Eige la respuesta correcta

**az login --use-device-code**