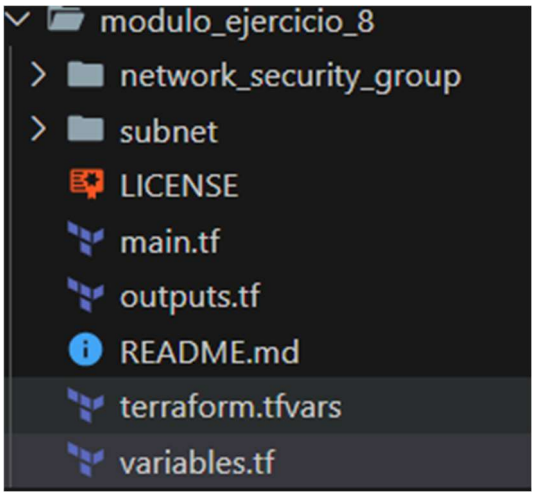
ENUNCIADO 8

```
 Configurar almacenamiento de estado remoto en Azure Blob Storage.

## Pre-requisitos

- Tener una cuenta de almacenamiento en Azure con un contenedor creado.

## Enunciado

Toma como base uno de los módulos desarrollados en ejercicios anteriores
y configura el almacenamiento de estado remoto para Terraform en Azure
Blob Storage.

Documenta los pasos necesarios para llevar a cabo el proceso con una
breve explicación (en una línea) de cada uno; desde los cambios
necesarios en azure, si los hubiera, hasta el resultado final tras la
destrucción de la infraestructura.
```

Voy a tomar como base el modulo del ejercicio 7, lo voy a renombrar y lo llamaré ejercicio 8:



En las variables de la raíz del ejercicio 8 añadimos una :

```hcl
#le añadimos un prefijo para añadirle el nombre de la empresa
variable "naming_prefix" {

  type = string
  default = "stemdo"
}
```

```
#numero aleatorio para el nombre del recurso
resource "random_integer" "rn"{
  max = 99999
  min = 10000
}
#creo al cuenta de almacenamiento
resource "azurerm_storage_account" "sa" {
  account_replication_type = "LRS"
  account_tier = "Standard"
  location = var.location
  name= "${lower(var.naming_prefix)}${random_integer.rn.result}"
  resource_group_name = var.resource_group_name

}
```

Creo la cuenta de almacenamiento

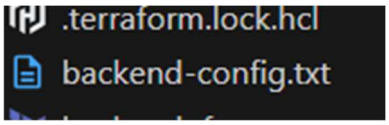
Después creo un contenedor que hará uso de esta cuenta de almacenamiento y añado un token sas para proteger y limitar los accesos a los recursos:

```
creo el contenedor
resource "azurerm_storage_container" "container" {
  name = "terraform-state"
  storage_account_name = azurerm_storage_account.sa.name

}
 #creo un token sas para el acceso al contenedor

 data "azurerm_storage_account_sas" "sas"{
  connection_string = azurerm_storage_account.sa.name
  https_only = true
  expiry = timeadd(timestamp(),"100d")
  start = timestamp()
  permissions {
    read = true
    write = true
    delete =true
    list = true
    add = true
    create = true
    update = false
    process = false
    filter = true
    tag = true
  }
  resource_types {
  service=true
```

```
  container=true
  object=true
  }
services {
  blob = true
  queue = false
  table = false
  file =false
}

 }
```

```
 #creamos el archivo de config para el backend que utilizaremos para
subir el terraform.tfstate
resource "local_file" "post-config" {
  depends_on = [azurerm_storage_container.container]
  filename              = "${path.module}/backend-config.txt"
  content     = <<EOF

storage_account_name  = "${azurerm_storage_account.sa.name}"
container_name        = "terraform.tfstate"
key                   = "terraform.tfstate"
sas_token             = "${data.azurerm_storage_account_sas.sas.sas}"
EOF

}
```

Hago un terraform init y un terrafirm apply y crea el fichero backend:



Ahora hay que enlazar este archivo de configuración de backend-config

Creamos un archivo llamado backen.tf para configurar el backend donde indicaremos el proveedor.

Para aplicar esta configuración lo que haremos será usar la inform de bcken-config y en este backend:

Terraform init backend_config.txt

PS C:\Users\palonso\Desktop\Workspace_VSC\TERRAFORM\ejercicio_terraform> terraform init -backend-config.txt

Initializing the backend...

Initializing modules...

Initializing provider plugins...

- Reusing previous version of hashicorp/azurerm from the dependency lock file

- Reusing previous version of hashicorp/random from the dependency lock file

- Reusing previous version of hashicorp/local from the dependency lock file

- Using previously-installed hashicorp/azurerm v3.0.2

- Using previously-installed hashicorp/random v3.6.1

- Using previously-installed hashicorp/local v2.5.1

╷

│ Warning: Missing backend configuration

│

│ -backend-config was used without a "backend" block in the configuration.

│

│ If you intended to override the default local backend configuration,

│ no action is required, but you may add an explicit backend block to your

│ configuration to clear this warning:

│

│ terraform {

│   backend "local" {}

│ }

│

│ However, if you intended to override a defined backend, please verify that

│ the backend configuration is present and valid.

│

╵

╷

│ Warning: Backend configuration ignored

```
╷
│   on modules\modulo_ejercicio_8\backend.tf line 2, in terraform:
│    2:   backend "azurerm" {
╵
│ Any selected backend applies to the entire configuration, so Terraform expects provider configurations only in the root module.
╷
│ This is a warning rather than an error because it's sometimes convenient to temporarily call a root module as a child module for testing
│ purposes, but this backend configuration block will have no effect.
╵
│ (and one more similar warning elsewhere)
I
```

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.