



Despliegue de Recursos de Datos de Azure con Terraform

Wagner Crivelini



Presentación:

Wagner Crivelini

- Consultor Senior de Microsoft Brasil
- Ingeniero de Datos
- Columnista de múltiples portales con +250 publicaciones



Agenda

[Introducción a Terraform](#)

[Ejemplo 1](#)

[Control de Estado](#)

[Scripts - Un Mejor Enfoque](#)

[Ejemplo 2](#)

[Múltiples Recursos](#)

[Ejemplo 3](#)

[Resumen](#)





Introducción Terraform

- Adopción de la Infraestructura como Código (IaC)
- Terraform (HashiCorp) es ampliamente utilizado:
 1. Declarativo (código limpio)
 2. Idempotente (consistencia)
 3. Agnóstico (múltiples proveedores de nube)
 4. Open source (contribuciones de la comunidad)



Terraform – Comandos Básicos

- Puedes instalarlo localmente y ejecutar vía VS CODE / CMD

<https://www.terraform.io/downloads>

- Comandos básicos:

C:\TEMP\terraform init

C:\TEMP\terraform validate #valida scripts

C:\TEMP\terraform plan #planea la ejecución de los scripts

C:\TEMP\terraform apply #ejecuta scripts

C:\TEMP\terraform destroy #borra los recursos

C:\TEMP\terraform show #presenta resultados del despliegue



Terraform – Scripts

- Creados en lenguaje HCL o JSON
- Eliges tu proveedor de nube (para Azure, usar **AzureRM**)

<https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>

- Informar valores de los parámetros del recurso que va a ser creado
- Utilizar **variables** en lugar de poner valores en el código
- Valorar la legibilidad del código (script único vs **múltiples scripts**)
- **IMPORTANTE:** despliegue de múltiples recursos es hecho en paralelo



Terraform – Lenguaje HCL

- Scripts empiezan con declaración de proveedores y sus versiones.

```
terraform {  
  required_providers {  
    azurerm = {  
      source  = "hashicorp/azurerm"  
      version = "~>3.0"  
      ##any azurerm version, from 3.0.0 or above  
    }  
  }  
}
```

- Despliegue de recursos debe seguir la sintaxis usada por AzureRM

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/resource_group)

```
resource "azurerm_resource_group" "rg" {  
  name      = "wagnerresourcegroup2024"  
  location = "East Us"  
}
```



Terraform – Ejecutar Scripts

- Como ejecutar los scripts

```
cd <path_to_scripts>
az login
rem to deploy to specific subscription run next command
az account set --subscription <subscriptionID>
terraform init
terraform validate
terraform plan [-out] <repositorio/arquivoPlan.tfplan>
terraform apply [-auto-approve] [<repositorio/arquivoPlan.tfplan>]
```



Ejemplo 1 : Despliegue Grupo de Recursos

DEMO



Control de Estado

- Después del primer despliegue, nuevos archivos serán creados en la carpeta del proyecto
- Archivo TFSTATE controla el estado de los recursos
- Siempre que se inicia un despliegue, Terraform busca por el TFSTATE.
- **JAMÁS PERCA/CAMBIE TFSTATE**



Para Cambiar o Borrar Recursos

- TFSTATE es la llave para despliegues exitosos
- Resultados obtenidos con nuevos despliegues:
 1. Si los scripts no han cambiado – nada ocurre
 2. Si han cambiado los valores de variables – recurso es cambiado
 3. Si el recurso es borrado del script – recurso va a ser borrado
 4. Si pierdes TFSTATE – se supone que el recurso no existe (puede ocurrir error)



Scripts - Un Mejor Enfoque

- Dividir tu script en varios es una manera mejor:
 1. Mejora gestión (ex: manejo de actualizaciones del proveedor)
 2. Mejora escalabilidad (ex: despliegue simultaneo de recursos)
 3. Mejora reusabilidad (ex: múltiples ambientes – DEV, TXT, PRD)

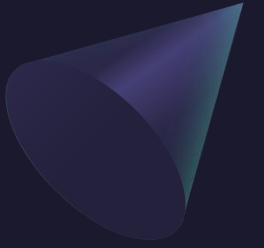


Terraform – Tipos de Scripts

- Es buena práctica dividir código en scripts distintos
- Proyecto puede utilizar:
 - **[main].tf** : archivo principal; puede contener proveedores and recursos
 - **variables.tf** : donde se define tipos de datos, descripciones y valores default
 - **locals.tf** : permite crear expresiones basadas en variables (Ejemplo: creación tags)
 - **output.tf** : para presentar valores generados durante el despliegue (Ejemplo: Ids).
 - **Archivos custom**: para mejorar organización del proyecto (Ejemplo:TFVARS)
 - **MODULES** : biblioteca de código reutilizable; encapsula la sintaxis de recursos



Porqué Utilizar Módulos

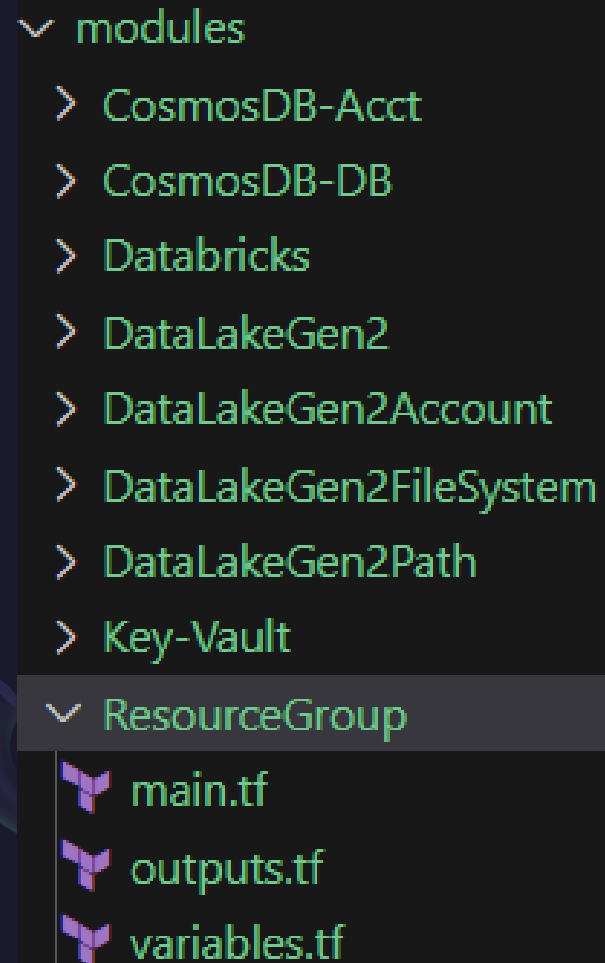


- Para aislar la sintaxis de recursos
- Biblioteca compartida por todo equipo de desarrollo
- Para crear un estándar
- Para manejar cambios de sintaxis entre versiones del proveedor



Organización de Scripts con Módulos

Crear una carpeta compartida



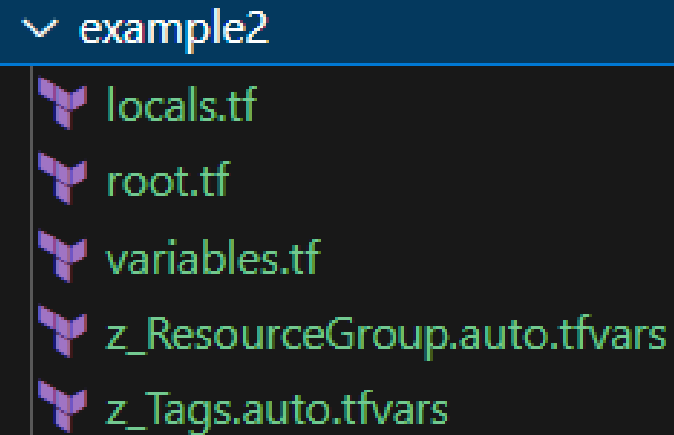
```

└─ modules
  ├── CosmosDB-Acct
  ├── CosmosDB-DB
  ├── Databricks
  ├── DataLakeGen2
  ├── DataLakeGen2Account
  ├── DataLakeGen2FileSystem
  ├── DataLakeGen2Path
  ├── Key-Vault
  └─ ResourceGroup
     ├── main.tf
     ├── outputs.tf
     └── variables.tf

```

A screenshot of a file explorer showing a shared folder structure. The 'modules' folder is expanded, showing a list of subfolders: CosmosDB-Acct, CosmosDB-DB, Databricks, DataLakeGen2, DataLakeGen2Account, DataLakeGen2FileSystem, DataLakeGen2Path, Key-Vault, and ResourceGroup. The 'ResourceGroup' folder is selected and expanded, showing three files: main.tf, outputs.tf, and variables.tf.

Crear carpetas para cada proyecto



```

└─ example2
  ├── locals.tf
  ├── root.tf
  ├── variables.tf
  ├── z_ResourceGroup.auto.tfvars
  └── z_Tags.auto.tfvars

```

A screenshot of a file explorer showing a project-specific folder structure. The 'example2' folder is expanded, showing a list of files: locals.tf, root.tf, variables.tf, z_ResourceGroup.auto.tfvars, and z_Tags.auto.tfvars.



Contenido de Scripts: Módulos

```
modules > ResourceGroup > main.tf
1  #reference https://registry.terraform.io/providers/h
2  resource "azurerm_resource_group" "ResourceGroup" {
3      name
4      location
5      tags
6  }
```

```
modules > ResourceGroup > outputs.tf
1  output "newResourceGroupName" {
2      value = azurerm_resource_group.ResourceGroup.name
3  }
4
5  output "id" {
6      value = azurerm_resource_group.ResourceGroup.id
7  }
8
9  #output "tenant_id" {
10     # value = azurerm_resource_group.ResourceGroup.tenant_id
11     #}
12
```

```
modules > ResourceGroup > variables.tf
1  variable "rg_name" {
2      type = string
3      description = "resource group name"
4  }
5
6  variable "location" {
7      type = string
8      description = "code related to Azure Region"
9  }
10
11
12  variable "tags" {
13      type = map(string)
14      description = "tags related to project"
15  }
```



Contenido de los Scripts Main & Variables

example2 > root.tf


```
1  # Terraform and Azure Provider configuration
2  terraform {
3      required_providers {
4          azurerm = {
5              source = "hashicorp/azurerm"
6              version = "~>3.0"
7              ##any azurerm version, from 3.0.0 or above
8          }
9      }
10 }
11 provider "azurerm" {
12     features {}
13 }
14
15 module "ResourceGroup1" {
16     source = "../modules/ResourceGroup"
17     rg_name = local.rgname1
18     location = var.location1
19     tags = local.tag
```

example2 > variables.tf


```
33 # #####
34 # variables related to RESOURCE GROUP
35 # #####
36 variable "resourceIdentifier" {
37     type = string
38     description = "prefix to resource group name"
39 }
40
41 variable "location1" {
42     type = string
43     description = "code related to Azure Region"
44 }
45
46 variable "shortlocation1" {
47     type = string
48     description = "short code related to Azure Region"
49 }
```




Contenido de los Scripts TFVARS & Locals

example2 >  z_Tags.auto.tfvars

```
1  company           = "WWI"
2  project            = "platdt"
3  costcenter         = "xyz"
4  environment        = "dev"
5  shortprovider      = "az"
6  resource_id        = "10"
7
```

example2 >  z_ResourceGroup.auto.tfvars

```
1
2  resourcegroup_idenfifier= "rg"
3  location              = "southcentralus"
4  shortlocation         = "ussc"
```

example2 >  locals.tf

```
1
2  locals {
3
4      rgname1 = format("%s-%s-%s-%s-01"
5          , var.shortprovider, var.shortlocation1
6          , var.environment, var.resourceIdentifier)
7
8      tag = {
9          company      = var.company
10         project       = "${var.company}-${var.project}"
11         costcenter    = var.costcenter
12         environment   = var.environment
13     }
14 }
```



Ejemplo 2 : Como Utilizar Módulos

DEMO



Múltiples Recursos

- Atención a convenciones de nomenclatura
<https://bit.ly/azurenames>
- Cuidado con los nombres únicos
- Se puede utilizar proveedores diferentes en el mismo proyecto



Nombres Únicos

- Nombres de recursos necesitan ser distintos dentro de su alcance
- Hay recursos con alcance global: “prueba y error”
- Alternativa: adicionar un valor aleatorio al nombre (**locals.tf**)

```
locals.tf
1  #Random ID for unique naming
2  resource "random_integer" "rand" {
3      min = 000001
4      max = 999999
5  }
6
7  locals {
8      randomSuffix = "${format("%06s", random_integer.rand.result)}"
9      rgname = "${var.rgPrefix}${var.rgName}-${local.randomSuffix}"
10 }
```

Dependencias entre Recursos

- Muchos recursos son dependientes de otro recurso “padre”
- Atención a 2 puntos principales:
 - Recurso “hijo” hay que se desplegar DESPUÉS que el padre sea creado
 - Colectar output del recurso “padre” (requerimientos para creación del hijo)

```
modules > SynapseWorkspace > outputs.tf
1  output "name" {
2    description = "The name of the resource created."
3    value       = azurerm_synapse_workspace.synapseworkspace.name
4  }
5
6  output "id" {
7    description = "The id of the resource created."
8    value       = azurerm_synapse_workspace.synapseworkspace.id
9  }
```

```
example3 > root.tf
230 module "synapsesqlpool" {
231   source              = "../modules/SynapseSQLPool"
232   synsqlpool_name     = var.synsqlpool_name
233   synsqlpool_wrkspcid = module.synapseworkspace.id
234   synsqlpool_sku      = var.synsqlpool_sku
235   synsqlpool_mode     = var.synsqlpool_mode
236
237   depends_on = [module.synapseworkspace]
238 }
239
```



Mensajes del Despliegue

- Terraform presenta información al final del despliegue



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

module.synapseworkspace.azure_rm_synapse_workspace.synapseworkspace: Still creating... [3m50s elapsed]
module.synapseworkspace.azure_rm_synapse_workspace.synapseworkspace: Still creating... [4m0s elapsed]
module.synapseworkspace.azure_rm_synapse_workspace.synapseworkspace: Still creating... [4m10s elapsed]
module.synapseworkspace.azure_rm_synapse_workspace.synapseworkspace: Still creating... [4m20s elapsed]
module.synapseworkspace.azure_rm_synapse_workspace.synapseworkspace: Still creating... [4m30s elapsed]
module.synapseworkspace.azure_rm_synapse_workspace.synapseworkspace: Still creating... [4m40s elapsed]
module.synapseworkspace.azure_rm_synapse_workspace.synapseworkspace: Creation complete after 4m49s [id=/subscriptions/56508de2-630e-4d64-a40f-45bfa674520b/resourceGroups/az-ussc-dev-rg-563645/providers/Microsoft.Synapse/workspaces/az-ussc-dev-syn-563645]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Creating...
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [10s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [20s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [30s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [40s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [50s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [1m0s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [1m10s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [1m20s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [1m30s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [1m40s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [1m50s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [2m0s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [2m10s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [2m20s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [2m30s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [2m40s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Still creating... [2m50s elapsed]
module.synapsesqlpool.azure_rm_synapse_sql_pool.synapsesqlpool: Creation complete after 2m54s [id=/subscriptions/56508de2-630e-4d64-a40f-45bfa674520b/resourceGroups/az-ussc-dev-rg-563645/providers/Microsoft.Synapse/workspaces/az-ussc-dev-syn-563645/sqlPools/whiterabbit]

Apply complete! Resources: 17 added, 0 changed, 0 destroyed.
PS C:\temp\Terraform101\example3>
```



Ejemplo 3 : Despliegue de Múltiplos Recursos

DEMO





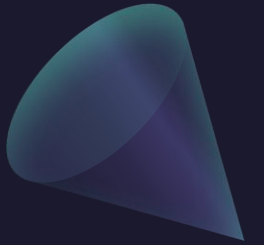
Resumen

- Terraform permite despliegue de docenas de recursos de Azure
- El HCL de Terraform es muy poderoso
- Los principios presentados se aplican a otros proveedores
- Varios proyectos open source adicionan funcionalidades
- Ejemplo: TFSEC & TRIVY
(<https://github.com/aquasecurity/trivy#how-to-pronounce-the-name-trivy>)



Donde Aprender Más

- [GitHub – todos archivos presentados en este taller:](https://github.com/wcrivelini/articles/tree/main/Azure_Terraform)
https://github.com/wcrivelini/articles/tree/main/Azure_Terraform
- [Database Deployment with Terraform - The Basics:](https://www.sqlservercentral.com/articles/database-deployment-with-terraform-the-basics)
<https://www.sqlservercentral.com/articles/database-deployment-with-terraform-the-basics>
- [Database Deployment with Terraform – Modules:](https://www.sqlservercentral.com/articles/database-deployment-with-terraform-modules)
<https://www.sqlservercentral.com/articles/database-deployment-with-terraform-modules>



Gracias

Wagner Crivelini

email

wagner.crivelini@microsoft.com

Linkedin

<https://www.linkedin.com/in/wagner-crivelini/>

