



Laboratorio
PRACTICA
TERRAFORM



CONTROL DE VERSIONES

Elaborado por: Jonatan Stiven Gutierrez	No. de Versión: 1.0.0
Revisado por:	Fecha de revisión:
Aprobado por:	Fecha de Aprobación:

Historia de Modificaciones

No. de Versión	Fecha de Versión	Autor	Revisado por	Aprobado por	Descripción
1.0.0	21/02/2024	Jonatan Stiven Gutierrez			Documento Original

Lista de distribución

Para	Acción*	Empresa	Firma/Medio de Entrega

* Tipos de acción: Aprobar, Revisar, Informar, Archivar, Complementar, Asistir a junta, Otras (por favor especificar)



Contenido

INTRODUCCION	4
PRERREQUISITOS	4
EJERCICIO 6:.....	5



INTRODUCCION

El siguiente documento proporciona una introducción detallada a los ejercicios realizados en el laboratorio.

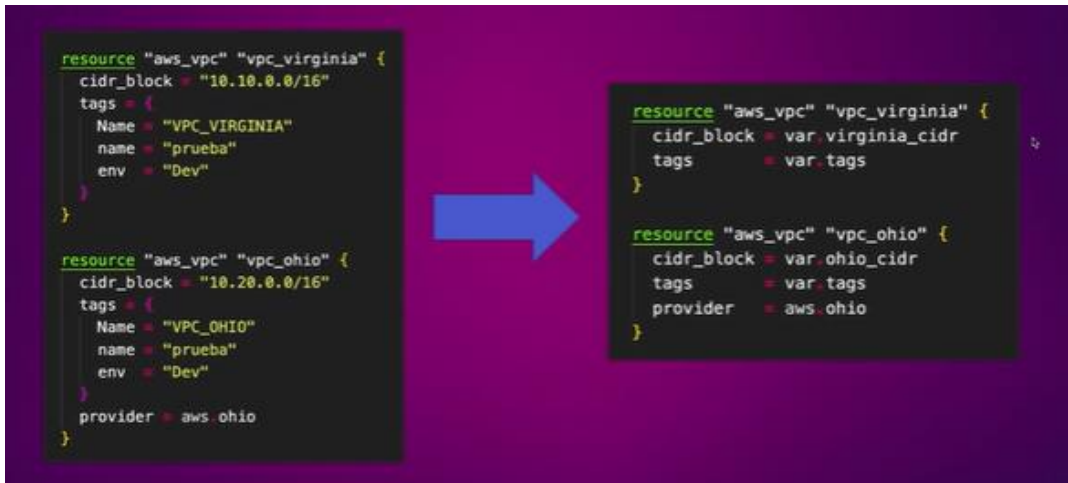
PRERREQUISITOS

- Tener entendimiento de los temas vistos anteriormente.



EJERCICIO 6:

En este ejercicio vamos a trabajar las variables en terraform.



1. Nos dirigimos a la consola y hacemos una copia de la carpeta practica_05, la cual llamaremos practica_06.
 - `cp -a practica_05 practica_06`
 - `cd practica_06`
 - `code .`

```
jonatan@ubuntu:~$ cd practica_terraform/
jonatan@ubuntu:~/practica_terraform$ cd practica_05
jonatan@ubuntu:~/practica_terraform/practica_05$ code .
jonatan@ubuntu:~/practica_terraform/practica_05$ cd ..
jonatan@ubuntu:~/practica_terraform$ cp -a practica_05 practica_06
jonatan@ubuntu:~/practica_terraform$ cd practica_06
jonatan@ubuntu:~/practica_terraform/practica_06$ ls -la
. .. providers.tf .terraform .terraform.lock.hcl terraform.tfstate terraform.tfstate.backup vpc.tf
jonatan@ubuntu:~/practica_terraform/practica_06$
```

2. Nos dirigimos al archivo "vpc.tf".
 - Agregamos lo siguiente al final del archivo.

```
17
18   variable "virginia_cidr" {
19     | default = "10.10.0.0/16"
20   }
21   variable "ohio_cidr" {
22     | default = "10.20.0.0/16"
23   }
```



3. Ahora editamos lo siguiente en el mismo archivo.
 - Editamos la línea 2 y la 10, debe quedar de la siguiente manera.

```
1 resource "aws_vpc" "vpc_virginia" {
2   cidr_block = var.virginia_cidr
3   tags = {
4     Name = "VPC_VIRGINIA"
5     env = "Dev"
6   }
7 }
8
9 resource "aws_vpc" "vpc_ohio" {
10  cidr_block = var.ohio_cidr
11  tags = {
12    Name = "VPC_OHIO"
13    env = "Dev"
14  }
15  provider = aws.Ohio
16 }
17
```

4. Ahora vamos a la consola y ejecutamos "terraform plan".
 - Como vemos nos toma el valor que pusimos en variables.

```
+ main_route_table_id      = (known after apply)
+ owner_id                 = (known after apply)
+ tags                    = {
+   Name = "VPC_OHIO"
+   env  = "Dev"
+ }
+ tags_all                 = {
+   Name = "VPC_OHIO"
+   env  = "Dev"
+ }
+ }

# aws_vpc.vpc_virginia will be created
+ resource "aws_vpc" "vpc_virginia" {
+   arn                    = (known after apply)
+   cidr_block             = "10.10.0.0/16"
+   default_network_acl_id = (known after apply)
+   default_route_table_id = (known after apply)
+   default_security_group_id = (known after apply)
+   dhcp_options_id        = (known after apply)
+   enable_dns_hostnames    = (known after apply)
+   enable_dns_support      = true
+   enable_network_address_usage_metrics = (known after apply)
+   id                     = (known after apply)
+   instance_tenancy        = "default"
+   ipv6_association_id     = (known after apply)
+   ipv6_cidr_block         = (known after apply)
+   ipv6_cidr_block_network_border_group = (known after apply)
+   main_route_table_id     = (known after apply)
+   owner_id                = (known after apply)
+   tags                    = {
+     Name = "VPC_VIRGINIA"
+     env  = "Dev"
+   }
+   tags_all                 = {
+     Name = "VPC_VIRGINIA"
+     env  = "Dev"
+   }
+ }

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

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
jonatan@ubuntu:~/practica_terraform/practica_06$
```



5. Ahora que pasaría si dejamos el valor de las variables vacías.

```
18 variable "virginia_cidr" {}
19
20 variable "ohio_cidr" {
21 }
```

- Ejecutamos terraform plan
- Como veremos nos pide colocar manualmente el valor de las variables

```
jonatan@ubuntu:~/practica_terraform/practica_06$ terraform plan
var.ohio_cidr
Enter a value: 
```

- Escribimos los valores
- Como veremos nos pide colocar manualmente el valor de las variables

```
var.ohio_cidr
Enter a value: 10.20.0.0/16

var.virginia_cidr
Enter a value: 10.10.0.0/16
```

- Como veremos el valor agregado lo muestra sin error alguno.

```
jonatan@ubuntu:~/practica_terraform/practica_06$ terraform plan
var.ohio_cidr
Enter a value: 10.20.0.0/16

var.virginia_cidr
Enter a value: 10.10.0.0/16

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_vpc.vpc_ohio will be created
+ resource "aws_vpc" "vpc_ohio" {
+   arn                               = (known after apply)
+   cidr_block                        = "10.20.0.0/16"
+   default_network_acl_id           = (known after apply)
+   default_route_table_id           = (known after apply)
+   default_security_group_id        = (known after apply)
+   dhcp_options_id                  = (known after apply)
+   enable_dns_hostnames              = (known after apply)
+   enable_dns_support                = true
+   enable_network_address_usage_metrics = (known after apply)
+   id                               = (known after apply)
+   instance_tenancy                 = "default"
+   ipv6_association_id              = (known after apply)
+   ipv6_cidr_block                   = (known after apply)
+   ipv6_cidr_block_network_border_group = (known after apply)
+   main_route_table_id              = (known after apply)
+   owner_id                         = (known after apply)
+   tags                             = {
+     "Name" = "VPC_OHIO"
+     "env"  = "Dev"
+   }
+   tags_all                         = {
+     "Name" = "VPC_OHIO"
+     "env"  = "Dev"
+   }
}
```



6. Usaremos variables de entorno.

- Teniendo en cuenta el id de las variables

```
variable "virginia_cidr" {  
  type = string  
}  
variable "ohio_cidr" {  
  type = string  
}
```

- Vamos a la consola y escribimos lo siguiente
- `export TF_VAR_virginia_cidr="10.10.0.0/16"`
- `export TF_VAR_ohio_cidr="10.20.0.0/16"`

```
jonatan@ubuntu:~/practica_terraform/practica_06$ export TF_VAR_virginia_cidr="10.10.0.0/16"  
jonatan@ubuntu:~/practica_terraform/practica_06$ export TF_VAR_ohio_cidr="10.20.0.0/16"  
jonatan@ubuntu:~/practica_terraform/practica_06$ terraform plan
```

- Ejecutamos "terraform plan", y como veremos ahora no nos pide ingresar datos, y toma las variables globales.

```
jonatan@ubuntu:~/practica_terraform/practica_06$ terraform plan  
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:  
+ create  
  
Terraform will perform the following actions:  
  
# aws_vpc.vpc_ohio will be created  
+ resource "aws_vpc" "vpc_ohio" {  
  + arn = (known after apply)  
  + cidr_block = "10.20.0.0/16"  
  + default_network_acl_id = (known after apply)  
  + default_route_table_id = (known after apply)  
  + default_security_group_id = (known after apply)  
  + dhcp_options_id = (known after apply)  
  + enable_dns_hostnames = (known after apply)  
  + enable_dns_support = true  
  + enable_network_address_usage_metrics = (known after apply)  
  + id = (known after apply)  
  + instance_tenancy = "default"  
  + ipv6_association_id = (known after apply)  
  + ipv6_cidr_block = (known after apply)  
  + ipv6_cidr_block_network_border_group = (known after apply)  
  + main_route_table_id = (known after apply)  
  + owner_id = (known after apply)  
  + tags = {  
    + "Name" = "VPC_OHIO"  
    + "env" = "Dev"  
  }  
  + tags_all = {  
    + "Name" = "VPC_OHIO"  
    + "env" = "Dev"  
  }  
}  
  
# aws_vpc.vpc_virginia will be created  
+ resource "aws_vpc" "vpc_virginia" {  
  + arn = (known after apply)  
  + cidr_block = "10.10.0.0/16"  
  + default_network_acl_id = (known after apply)  
  + default_route_table_id = (known after apply)  
  + default_security_group_id = (known after apply)
```




7. Para observar las variables de entorno se ejecuta el siguiente código.
- “env”, este comando es para la búsqueda general de variables de entorno.

```
jonatan@ubuntu:~/practica_terraform/practica_06$ env
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1926,unix/ubuntu:/tmp/.ICE-unix/1926
WINDOWID=10485767
QT_ACCESSIBILITY=1
EGL_VENDOR_LIBRARY_DIRS=/snap/konsole/28/etc/glvnd/egl_vendor.d:/snap/konsole/28/usr/share/glvnd/egl_vendor.d
SNAP_REVISION=28
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg:/snap/konsole/28/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
SNAP_REAL_HOME=/home/jonatan
SNAP_USER_COMMON=/home/jonatan/snap/konsole/common
LANGUAGE=es:en
LC_ADDRESS=es_CO.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=es_CO.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XDG_DATA_HOME=/home/jonatan/snap/konsole/common/.local/share
SHELL_SESSION_ID=54c66b125ee148d488cc5bb7e678a920
XDG_CONFIG_HOME=/home/jonatan/snap/konsole/common/.config
SNAP_INSTANCE_KEY=
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=es_CO.UTF-8
BAWF_DESKTOP_FILE_HINT=/var/lib/snapd/desktop/applications/konsole_konsole.desktop
GTK_MODULES=gail:atk-bridge
SNAP_EUID=1000
PWD=/home/jonatan/practica_terraform/practica_06
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=jonatan
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1954
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.L6QAJ2
SNAP_CONTEXT=7qyuX1Ub0Yr0_5L93EieuLY18285Ky0K7o2FJA-3328k48SoJt4Q
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
HOME=/home/jonatan
USERNAME=jonatan
IM_CONFIG_PHASE=1
LC_PAPER=es_CO.UTF-8
LANG=es_ES.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:cd=40;33:or=40;31:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:tar=01;31:tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:
```

- “env | grep TF_VAR”, este es para la búsqueda de variables de entorno que contengan “TF_VAR”.

```
jonatan@ubuntu:~/practica_terraform/practica_06$ env | grep TF_VAR
TF_VAR_ohio_cidr=10.20.0.0/16
TF_VAR_virginia_cidr=10.10.0.0/16
jonatan@ubuntu:~/practica_terraform/practica_06$
```

8. Otra forma de ejecutar una variable sin estar en el código es:
- terraform plan -var ohio_cidr="10.20.0.0/16"
9. Ahora vamos a borrar las variables de entorno creadas:
- unset TF_VAR_ohio_cidr
 - unset TF_VAR_virginia_cidr

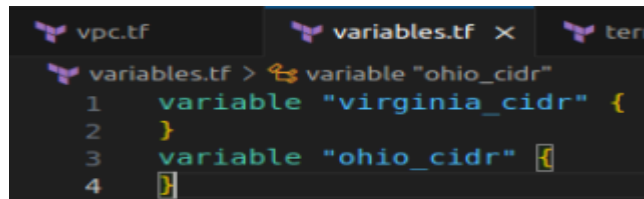


10. Ahora haremos el manejo de variables de la forma más recomendada:

- Creamos un archivo "variables.tf".
- Del archivo "vpc.tf", cortamos lo siguiente.

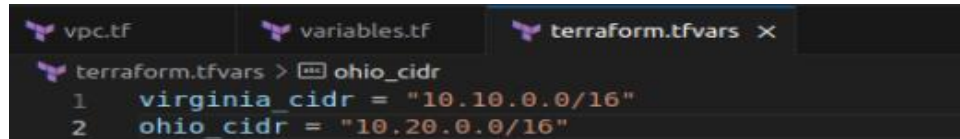
```
18 variable "virginia_cidr" {
19 }
20 variable "ohio_cidr" {}
21 }
```

- Pegamos en el archivo "variables.tf".

A screenshot of a code editor showing the file 'variables.tf'. The code contains two variable definitions: 'virginia_cidr' and 'ohio_cidr'.

```
vpc.tf  variables.tf x  terraform.tfvars
variables.tf > variable "ohio_cidr"
1 variable "virginia_cidr" {
2 }
3 variable "ohio_cidr" {}
4 }
```

- Creamos un archivo "terraform.tfvars", agregamos el valor de las variables.

A screenshot of a code editor showing the file 'terraform.tfvars'. The file contains two lines of variable assignments: 'virginia_cidr' and 'ohio_cidr'.

```
vpc.tf  variables.tf  terraform.tfvars x
terraform.tfvars > ohio_cidr
1 virginia_cidr = "10.10.0.0/16"
2 ohio_cidr = "10.20.0.0/16"
```



- Aplicamos el comando "terraform plan", y el valor de las variables las toma bien.

```
jonatan@ubuntu:~/practica_terraform/practica_06$ terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbol:
+ create

Terraform will perform the following actions:

# aws_vpc.vpc_ohio will be created
+ resource "aws_vpc" "vpc_ohio" {
  + arn                               = (known after apply)
  + cidr_block                        = "10.20.0.0/16"
  + default_network_acl_id           = (known after apply)
  + default_route_table_id           = (known after apply)
  + default_security_group_id        = (known after apply)
  + dhcp_options_id                  = (known after apply)
  + enable_dns_hostnames              = (known after apply)
  + enable_dns_support                = true
  + enable_network_address_usage_metrics = (known after apply)
  + id                               = (known after apply)
  + instance_tenancy                  = "default"
  + ipv6_association_id               = (known after apply)
  + ipv6_cidr_block                   = (known after apply)
  + ipv6_cidr_block_network_border_group = (known after apply)
  + main_route_table_id               = (known after apply)
  + owner_id                         = (known after apply)
  + tags                             = {
    + "Name" = "VPC_OHIO"
    + "env"  = "Dev"
  }
  + tags_all                         = {
    + "Name" = "VPC_OHIO"
    + "env"  = "Dev"
  }
}

# aws_vpc.vpc_virginia will be created
+ resource "aws_vpc" "vpc_virginia" {
  + arn                               = (known after apply)
  + cidr_block                        = "10.10.0.0/16"
  + default_network_acl_id           = (known after apply)
  + default_route_table_id           = (known after apply)
  + default_security_group_id        = (known after apply)
  + dhcp_options_id                  = (known after apply)
  + enable_dns_hostnames              = (known after apply)
  + enable_dns_support                = true
}
```

11. Terraform solo obtiene el valor de las variables por su nombre, por consecuencia debe respetar ciertas nomenclaturas, para que el sistema cargue de manera automática:
- El nombre con el .auto son para archivos siders, networkin, backends





12. Si queremos nombrar un archivo de manera diferente el sistema NO LO cargue de manera automática:
- Por ejemplo el archivo "terraform.tfvars", pero le cambiamos el nombre a "seti.tfvars"
 - Vamos a la consola
 - Terraform plan -var-file seti.tfvars
13. En caso de que tengamos una variable definida en dos lugares (variables de entorno y el archivo variables.tf), hay una prioridad en la elección de su definición.
- 4 mayor prioridad
 - 1 menor prioridad

Orden	Definición
1	Variable de entorno (export TF_VAR_ejemplo="valor")
2	terraform.tfvars
3	*.auto.tfvars (por orden alfabetico)
4	-var, -var-file (por linea de comandos)

Ejemplo 4 opción: terraform plan -var virginia_cidr="10.100.0.0/16"