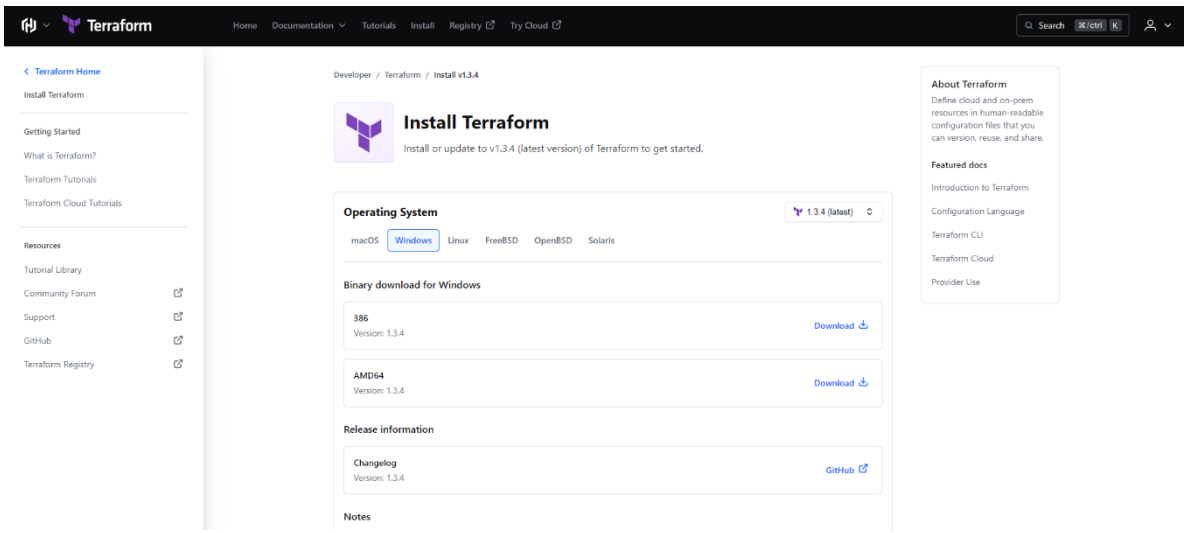
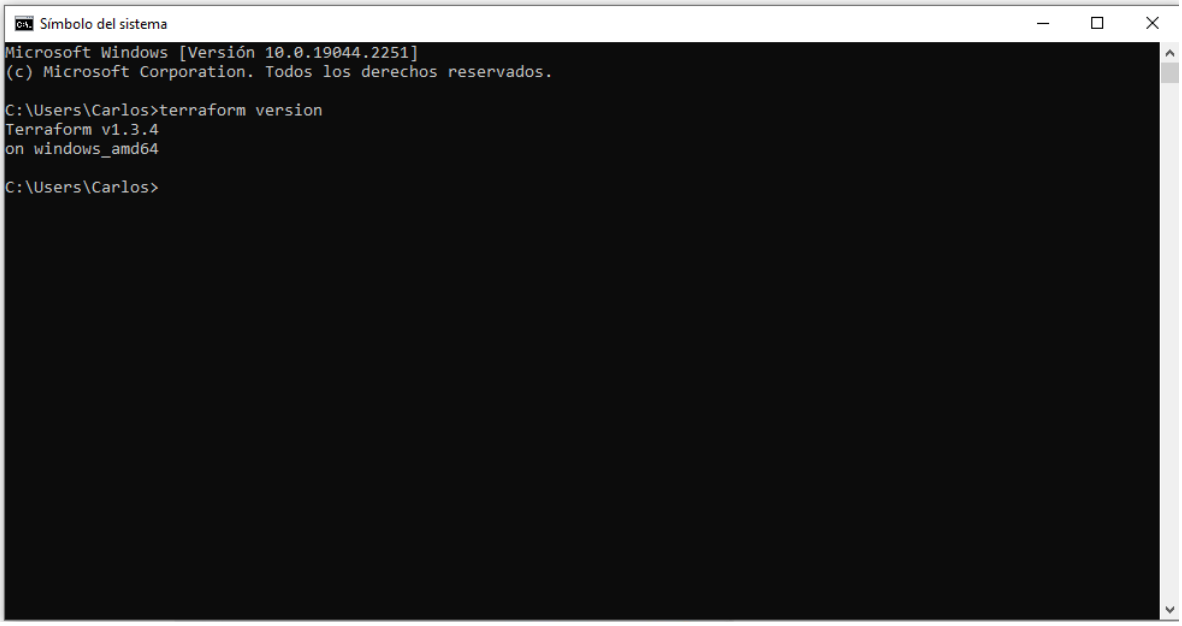
Carlos Alberto Arzuza Quiroz

Electiva de profundización II – Cloud Computer

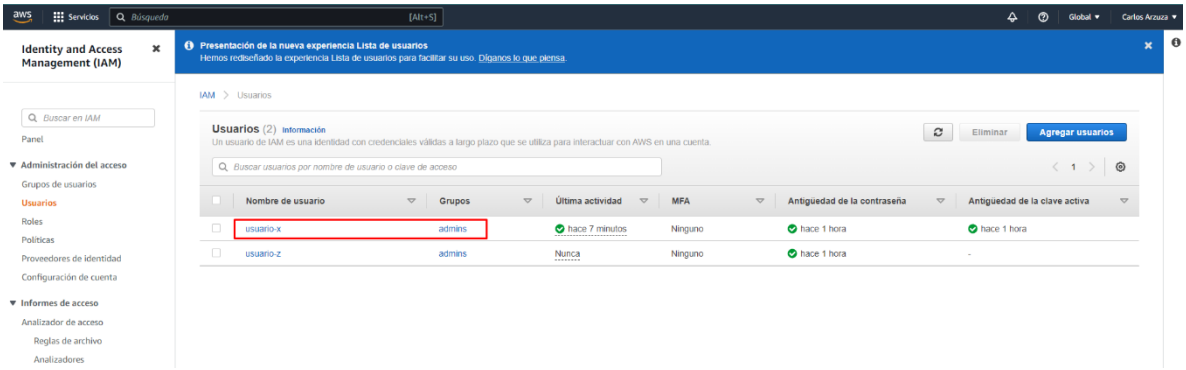
Laboratorio: Como crear una instancia EC2 en AWS con Terraform

1. Instalación de Terraform

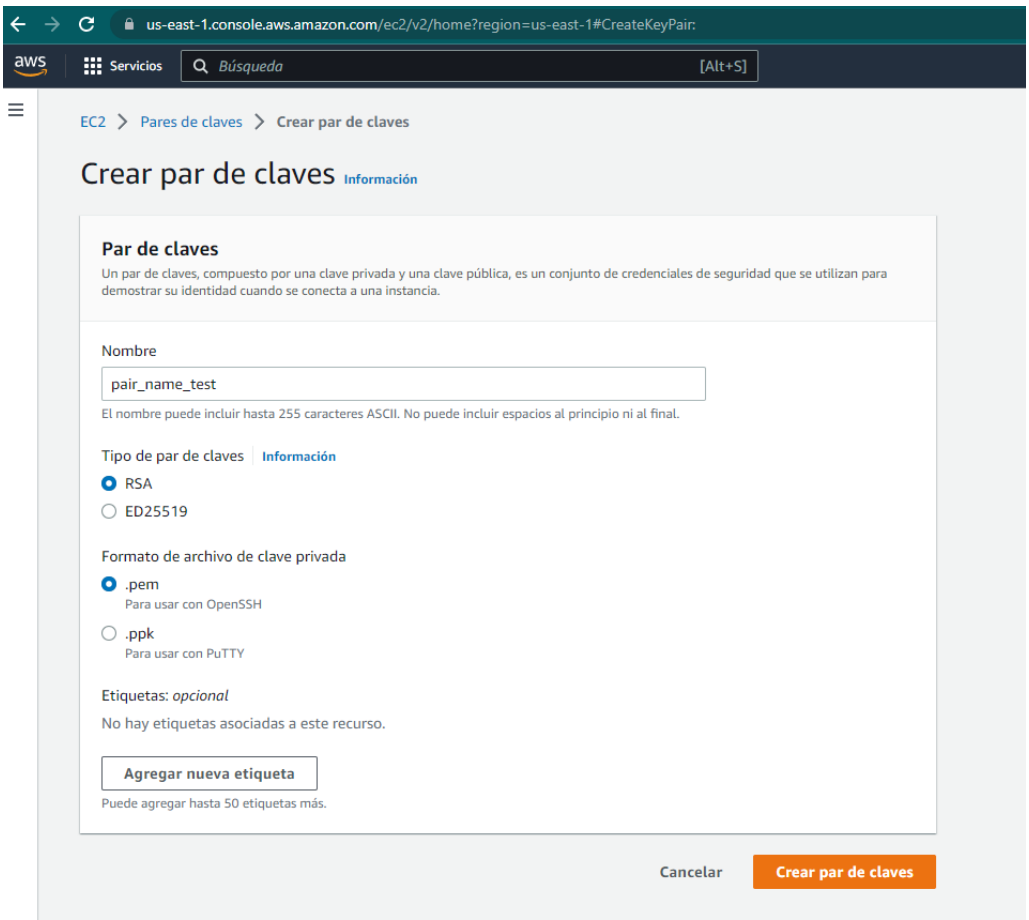


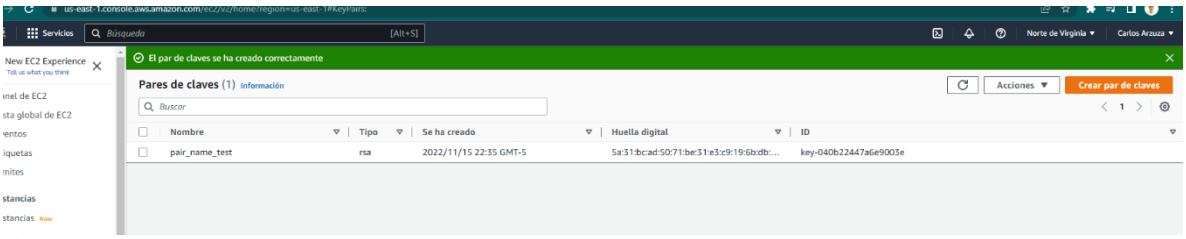


2. Usuario IAM creado y habilitado para acceder con acces key ID y secret Access key desde nuestro proyecto en Terraform



3. Creación de un par de claves con Amazon EC2





4. Ejecutamos “terraform init”. Este comando descarga e instala los plugins de los proveedores utilizados en la configuración. En nuestro caso es AWS.

```
Símbolo del sistema

C:\Users\Carlos\Documents\terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.39.0...
- Installed hashicorp/aws v4.39.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.

C:\Users\Carlos\Documents\terraform>
```

5. Ejecutamos “terraform plan”. Este comando se utiliza para ver los cambios que se producirán en la infraestructura.

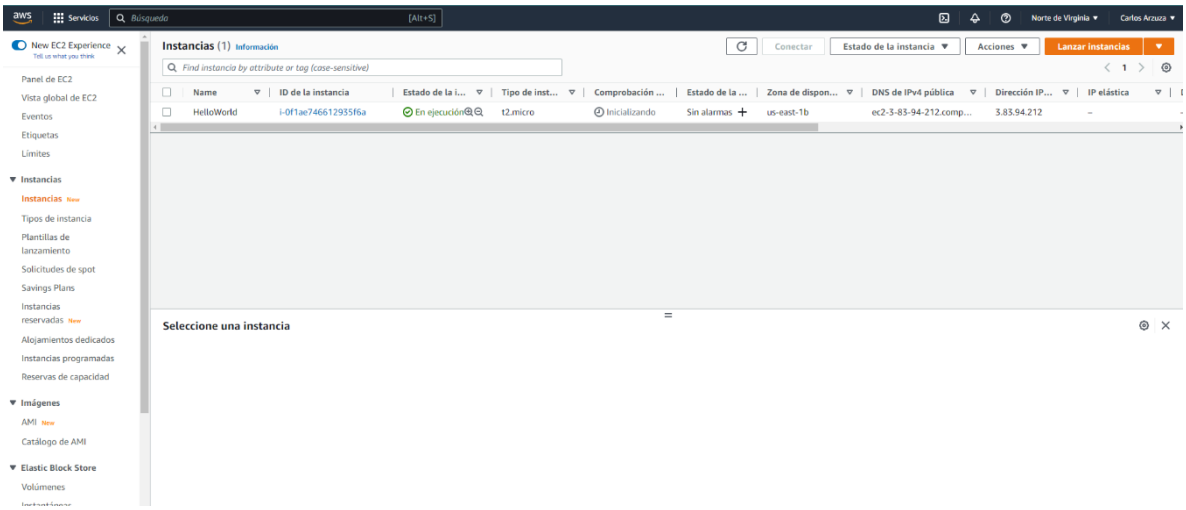
```
Símbolo del sistema

C:\Users\Carlos\Documents\terraform>terraform plan
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 1s [id=ami-072d6c9fae3253f26]

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_instance.my_instance will be created
  + resource "aws_instance" "my_instance" {
      + ami                                  = "ami-072d6c9fae3253f26"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
```

6. Ejecutamos “terraform apply”. Este comando creará los recursos en la AWS mencionados en el archivo main.tf.

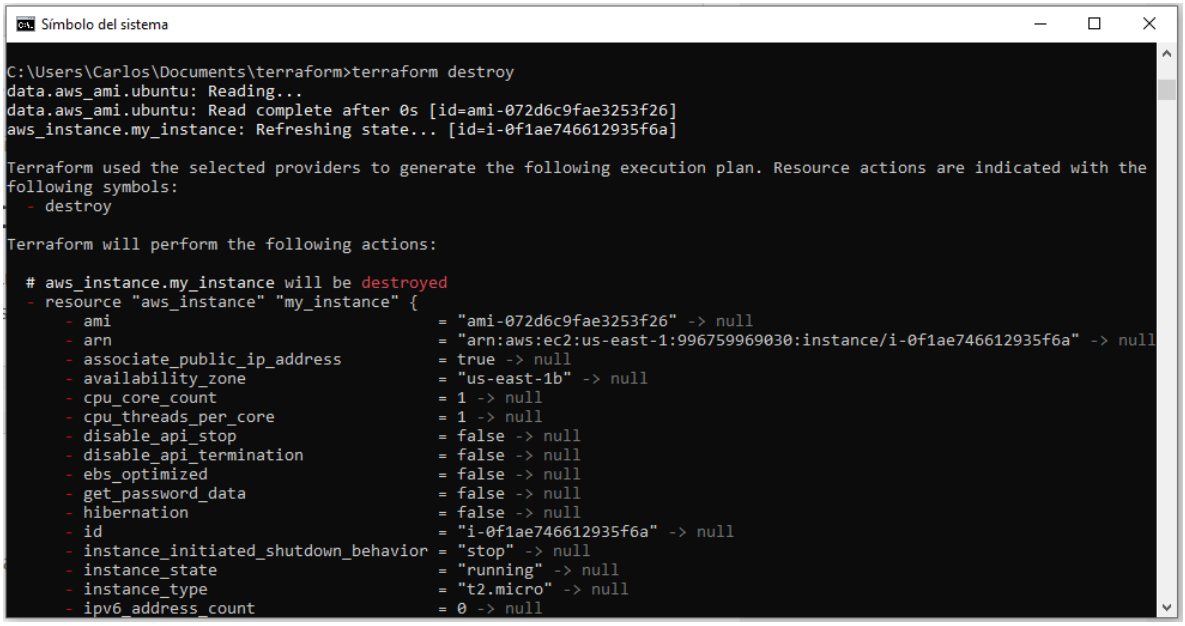
```
Símbolo del sistema
you run "terraform apply" now.

C:\Users\Carlos\Documents\terraform>terraform apply
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 0s [id=ami-072d6c9fae3253f26]

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_instance.my_instance will be created
  + resource "aws_instance" "my_instance" {
      + ami                                  = "ami-072d6c9fae3253f26"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```

```
Símbolo del sistema
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

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_instance.my_instance: Creating...
aws_instance.my_instance: Still creating... [10s elapsed]
aws_instance.my_instance: Still creating... [20s elapsed]
aws_instance.my_instance: Still creating... [30s elapsed]
aws_instance.my_instance: Creation complete after 33s [id=i-0f1ae746612935f6a]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

C:\Users\Carlos\Documents\terraform>
```

Verificamos en la consola de AWS EC2 si la instancia EC2 está Creada.



7. Por último, ejecutamos “terraform destroy” y eliminamos la Instancia EC2 creada mediante Terraform.



```
Símbolo del sistema
          - tags                  = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-0be15ad44fc2bdc51" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
        }
    }

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.my_instance: Destroying... [id=i-0f1ae746612935f6a]
aws_instance.my_instance: Still destroying... [id=i-0f1ae746612935f6a, 10s elapsed]
aws_instance.my_instance: Still destroying... [id=i-0f1ae746612935f6a, 20s elapsed]
aws_instance.my_instance: Still destroying... [id=i-0f1ae746612935f6a, 30s elapsed]
aws_instance.my_instance: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.

C:\Users\Carlos\Documents\terraform>
```

Verificamos en la consola de AWS EC2 si la instancia EC2 está Terminada.