


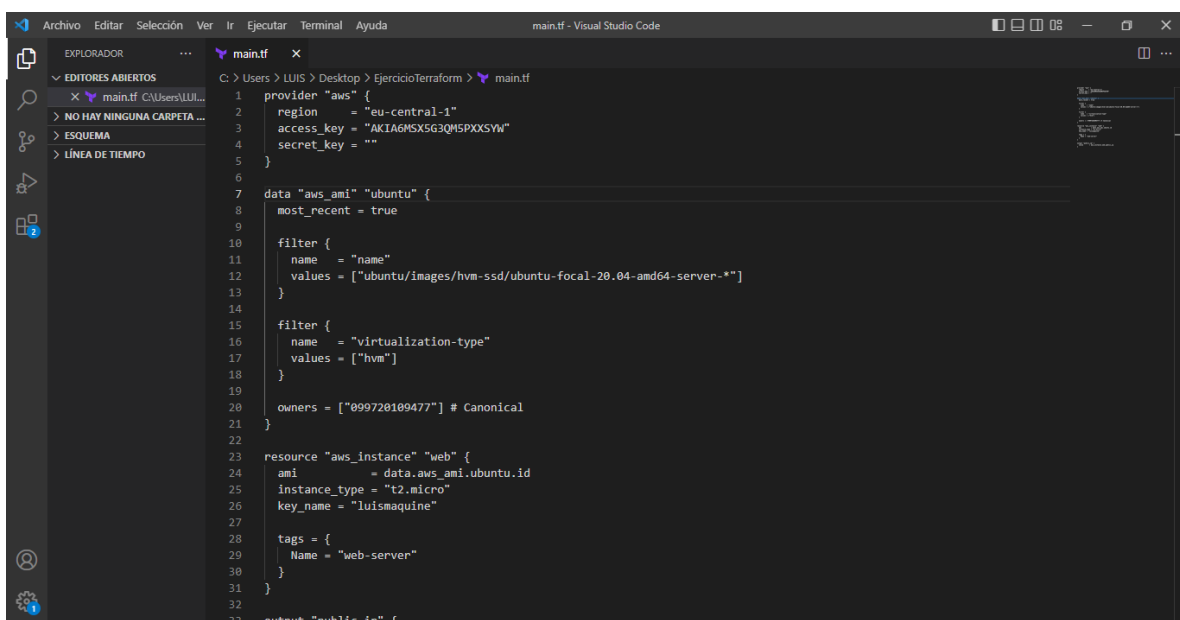
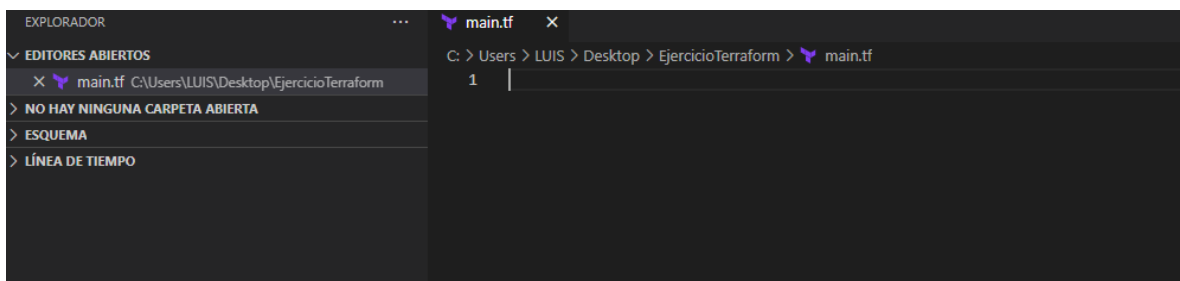
# LUIS EDUARDO GOMEZ SANTIAGO

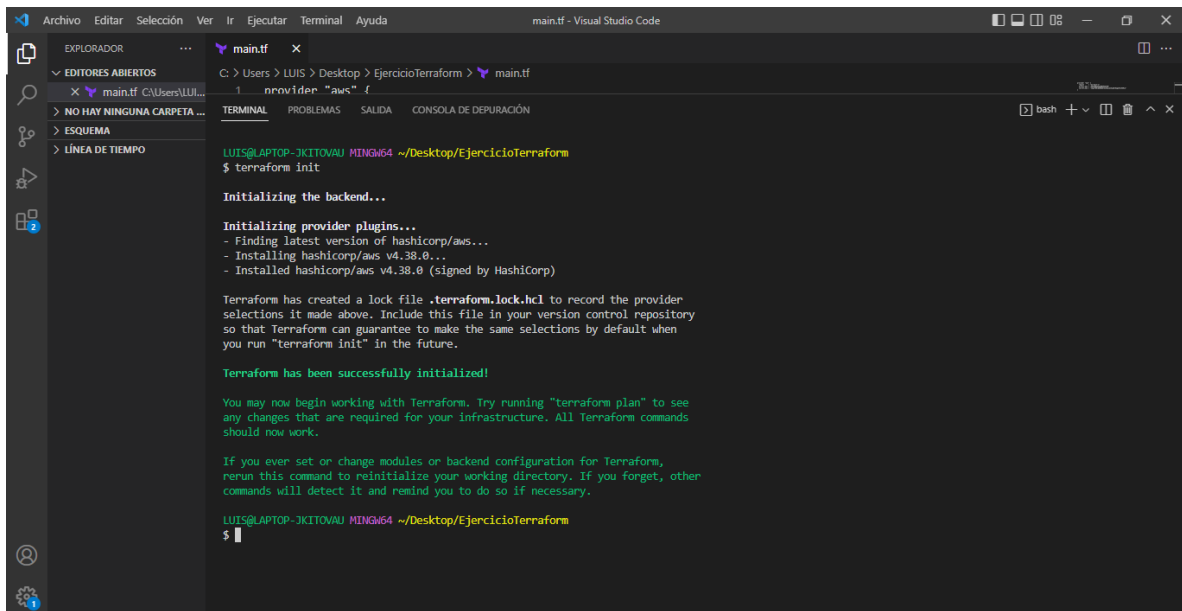
 Símbolo del sistema

```
Microsoft Windows [Versión 10.0.19043.2251]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\LUIS>terraform --version
Terraform v1.3.4
on windows_amd64

C:\Users\LUIS>
```





Visual Studio Code interface with the terminal open. The terminal shows the execution of the 'terraform init' command. The output indicates that Terraform is initializing the backend and provider plugins. It lists the steps: finding the latest version of hashicorp/aws, installing it, and confirming the installation. It also mentions the creation of a lock file and provides instructions on how to use Terraform.

```
C:\Users\LUIS\Desktop\EjercicioTerraform> main.tf
1  ...provider "aws" {

LUIS@LAPTOP-JKITOVAVU MINGW64 ~/Desktop/EjercicioTerraform
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.38.0...
- Installed hashicorp/aws v4.38.0 (signed by HashiCorp)

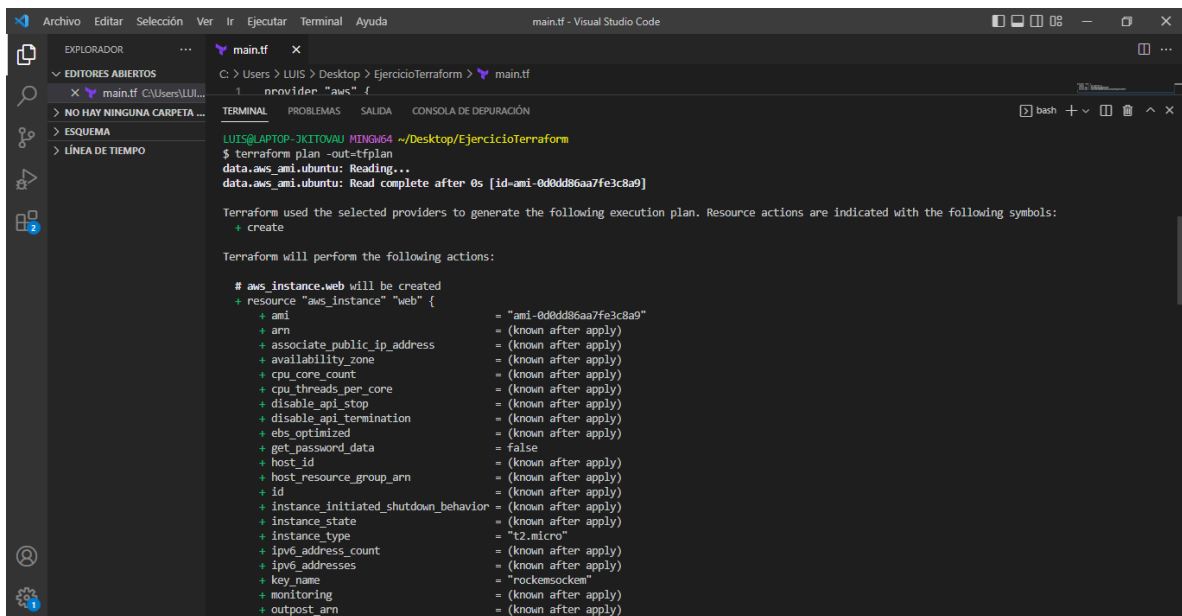
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.

LUIS@LAPTOP-JKITOVAVU MINGW64 ~/Desktop/EjercicioTerraform
$
```



Visual Studio Code interface with the terminal open. The terminal shows the execution of the 'terraform plan' command. The output displays the execution plan for the 'aws\_instance' resource, listing various attributes and their values. It indicates that the resource will be created and lists the actions Terraform will perform.

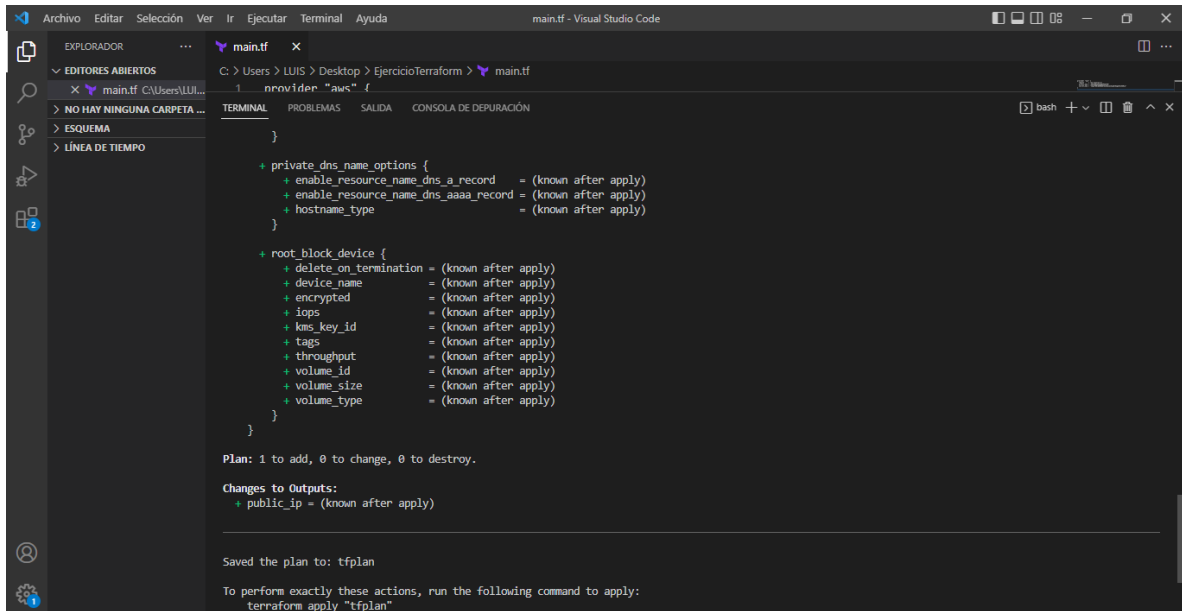
```
C:\Users\LUIS\Desktop\EjercicioTerraform> main.tf
1  ...provider "aws" {

LUIS@LAPTOP-JKITOVAVU MINGW64 ~/Desktop/EjercicioTerraform
$ terraform plan -out=tfplan
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 0s [id=ami-0d0dd86aa7fe3c8a9]

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.web will be created
+ resource "aws_instance" "web" {
  + ami                        = "ami-0d0dd86aa7fe3c8a9"
  + 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)
  + ipv6_addresses             = (known after apply)
  + key_name                   = "rocketsocketm"
  + monitoring                  = (known after apply)
  + outpost_arn                = (known after apply)
}
```



Visual Studio Code interface showing a Terraform configuration file named `main.tf`. The file defines an AWS instance with private DNS options and a root block device. The terminal shows the plan output, indicating that resources will be added.

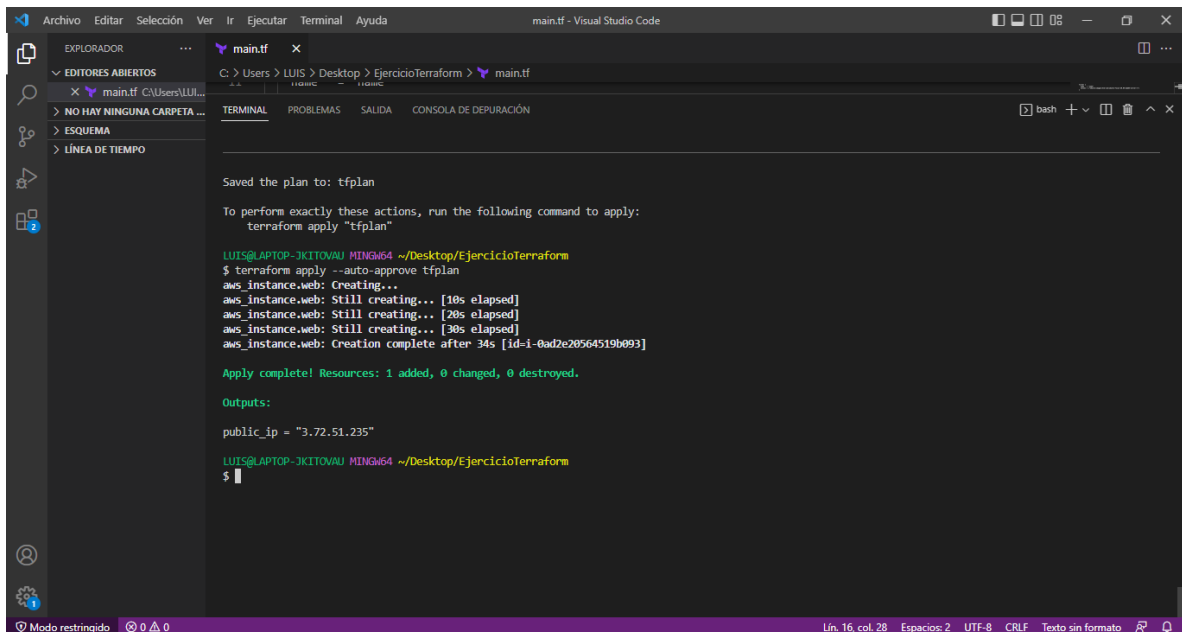
```
1. provider "aws" {  
  }  
  
+ private_dns_name_options {  
+   enable_resource_name_dns_a_record = (known after apply)  
+   enable_resource_name_dns_aaaa_record = (known after apply)  
+   hostname_type = (known after apply)  
+ }  
  
+ root_block_device {  
+   delete_on_termination = (known after apply)  
+   device_name = (known after apply)  
+   encrypted = (known after apply)  
+   iops = (known after apply)  
+   kms_key_id = (known after apply)  
+   tags = (known after apply)  
+   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.

Changes to Outputs:  
+ public\_ip = (known after apply)

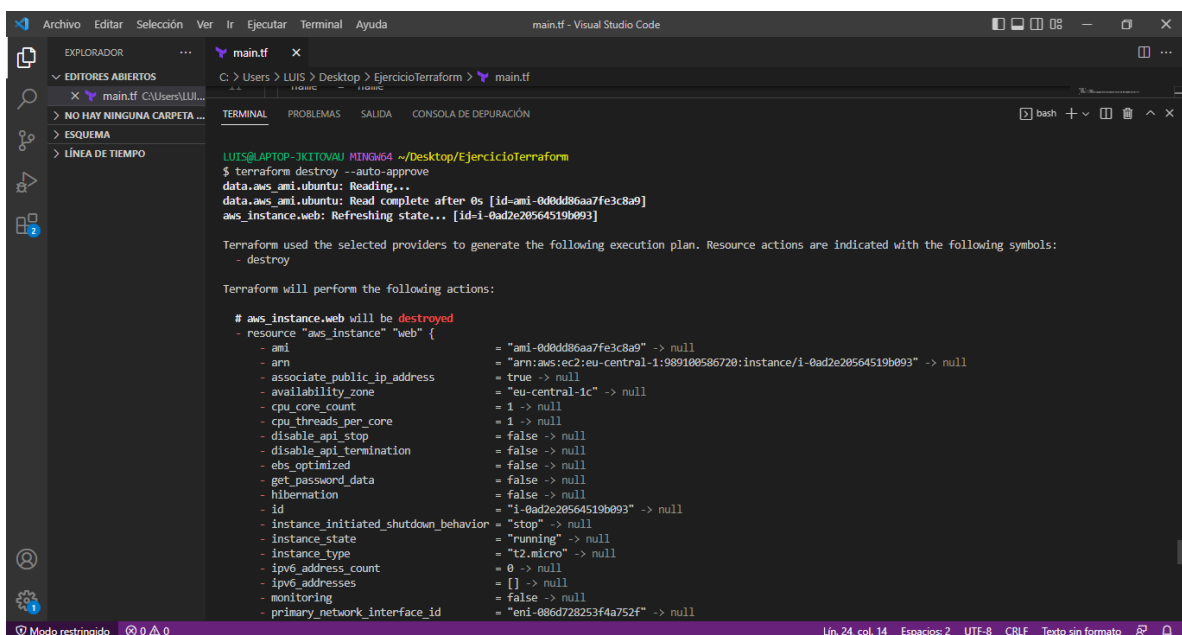
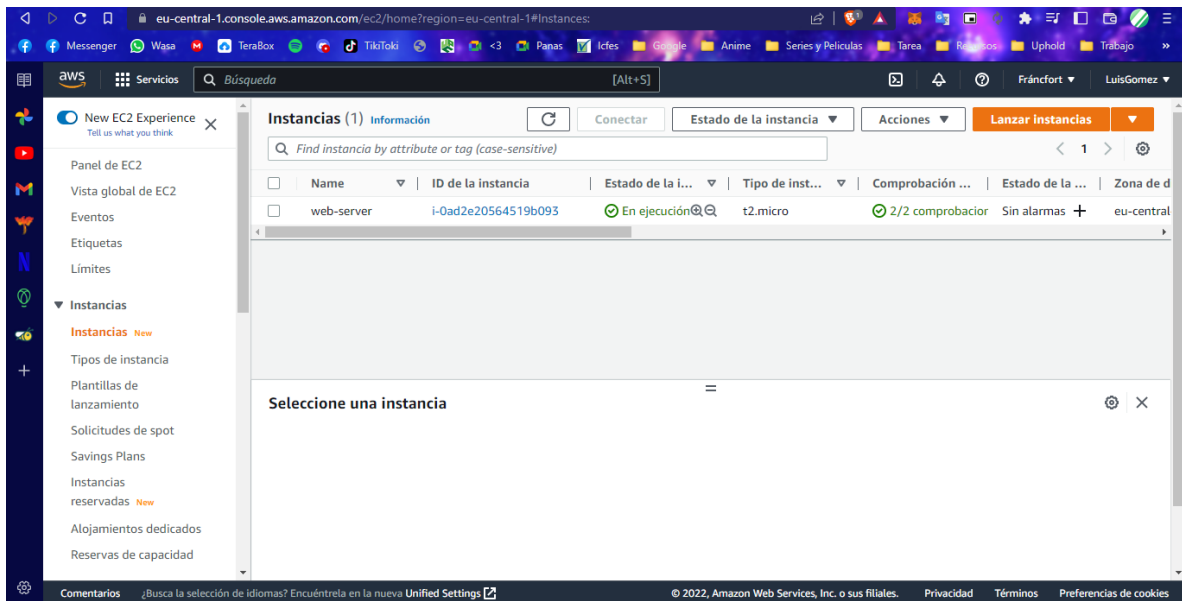
Saved the plan to: tfplan

To perform exactly these actions, run the following command to apply:  
terraform apply "tfplan"



Visual Studio Code interface showing the execution of the Terraform plan. The terminal displays the command `terraform apply --auto-approve tfplan` and the output showing the successful creation of the AWS instance.

```
LUIS@LAPTOP-JKITOVAV MINGW64 ~/Desktop/EjercicioTerraform  
$ terraform apply --auto-approve tfplan  
aws_instance.web: Creating...  
aws_instance.web: Still creating... [10s elapsed]  
aws_instance.web: Still creating... [20s elapsed]  
aws_instance.web: Still creating... [30s elapsed]  
aws_instance.web: Creation complete after 34s [id=i-0ad2e20564519b093]  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
  
Outputs:  
  
public_ip = "3.72.51.235"  
  
LUIS@LAPTOP-JKITOVAV MINGW64 ~/Desktop/EjercicioTerraform  
$
```



Visual Studio Code interface showing a terminal window with Terraform output. The terminal displays the configuration for a private DNS name options and a root block device, followed by the plan and changes to outputs. The output indicates that the instance is being destroyed and the resources are being destroyed.

```
main.tf - Visual Studio Code
C:\Users\LUIS\Desktop\EjercicioTerraform> terraform apply
- private_dns_name_options {
  - enable_resource_name_dns_a_record = false -> null
  - enable_resource_name_dns_aaaa_record = false -> null
  - hostname_type = "ip-name" -> null
}
- root_block_device {
  - delete_on_termination = true -> null
  - device_name = "/dev/sda1" -> null
  - encrypted = false -> null
  - iops = 100 -> null
  - tags = {} -> null
  - throughput = 0 -> null
  - volume_id = "vol-0955962032568cc76" -> null
  - volume_size = 8 -> null
  - volume_type = "gp2" -> null
}

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

Changes to Outputs:
  public_ip = "3.72.51.235" -> null
aws_instance.web: Still destroying... [id=i-0ad2e20564519b093, 10s elapsed]
aws_instance.web: Still destroying... [id=i-0ad2e20564519b093, 20s elapsed]
aws_instance.web: Still destroying... [id=i-0ad2e20564519b093, 30s elapsed]
aws_instance.web: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.

LUIS@LAPTOP-DKITOVWJ MINGW64 ~/Desktop/EjercicioTerraform
$
```

