**Modules in Terraform**

We know about work space in the terraform, it provides capability to maintain multiple state file in single configuration file at which same infrastructure is required to different environment like production, development, and testing so on.

But coming to Modules in the terraform, they are built in or readily available code they can be “**call”** while creating the infrastructure in any cloud provider.

That means In Terraform, a module is a fundamental building block for organizing and reusing infrastructure code.

**What is a Terraform Module?**

* A Terraform module is a collection of Terraform configuration files within a directory.
* It encapsulates a set of resources and their configurations, allowing you to create reusable components of your infrastructure.
* Essentially, it's a way to package and share Terraform code.

A module typically includes:

* **main.tf:** Contains the primary resource definitions.
* **variables.tf:** Defines input variables for the module.
* **outputs.tf:** Specifies output values that the module exposes.

Modules can be sourced from:

* Local paths (or) directory.
* GitHub repositories.
* Terraform Registry.

**Advantages of Using Terraform Modules**

✅ **Reusability**

* Modules allow you to reuse infrastructure code across multiple projects.

✅ **Maintainability**

* By breaking large configurations into smaller modules, it’s easier to manage and update them.

✅ **Consistency**

* Helps enforce best practices and uniformity across different environments (e.g., dev, staging, prod).

✅ **Simplified Management**

* Easier to troubleshoot and make changes as components are logically separated.

✅ **Encapsulation**

* Modules provide an abstraction layer, hiding the complexity of underlying infrastructure.

**Disadvantages of Using Terraform Modules**

❌ **Complexity Overhead**

* Managing and structuring multiple modules can become complex in large-scale projects.

❌ **Debugging Issues**

* When using remote or third-party modules, debugging can be harder compared to writing resources directly.

❌ **Versioning Conflicts**

* If a module is updated, it might break dependencies if not properly versioned.

❌ **Learning Curve**

* New users may find it challenging to understand module dependencies and usage.

Terraform automatically uses a **root module** (the main directory where Terraform is executed). You can also call **child modules** inside the root module to organize code better.

**Terraform Module Example:**

module "resource-group" {

source = "./modules/rg"

name = “RG”

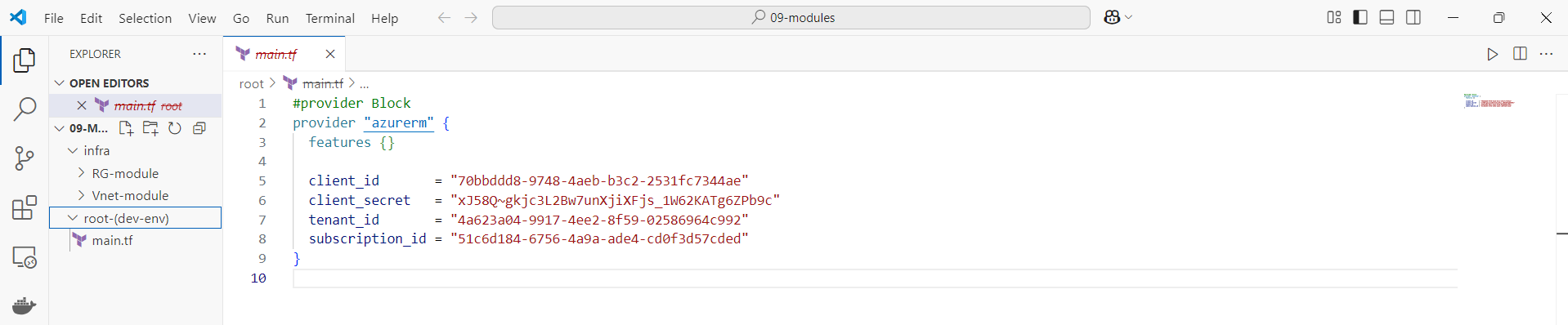
loacation = "eastus”

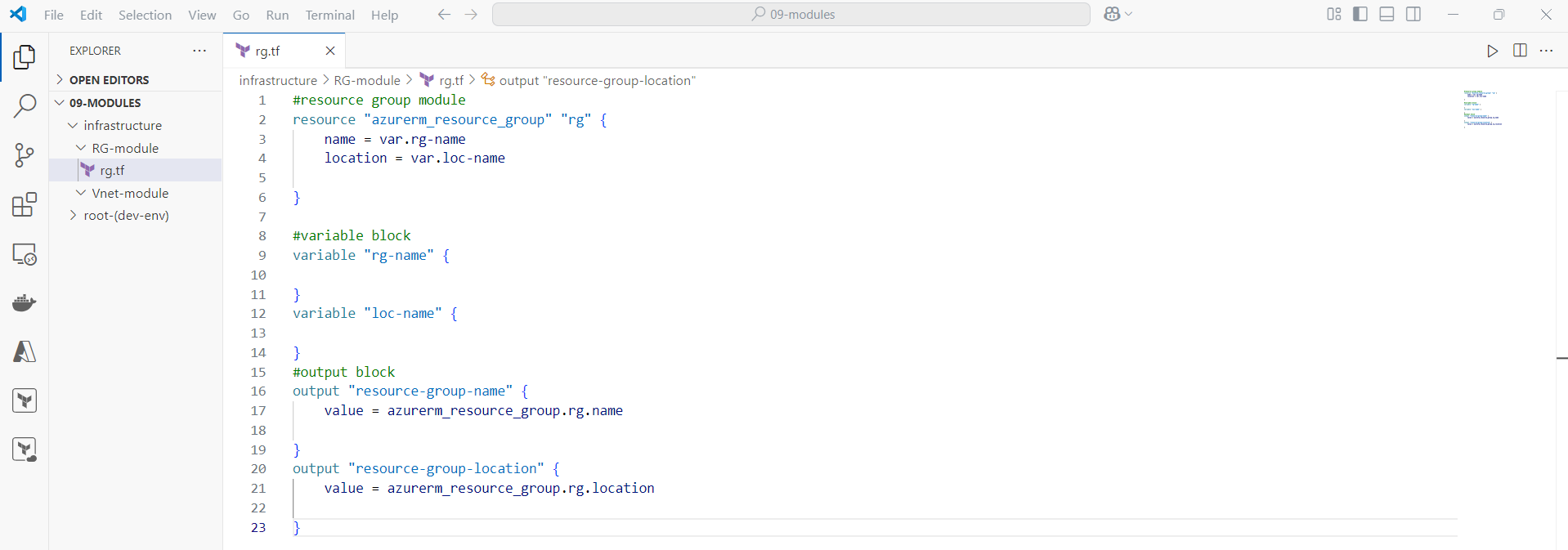
}

This example calls a **child module** named resource-group from ./modules/rg and passes an input variable name and location.

Now let’s create a child Module and call it from local directory to the root module of terraform configuration file.

Step1: Create two folders (root folder & infrastructure folder). And inside the infra folder create separate folders for each resource like RG, Vnet, Subnet, PIP and so on.

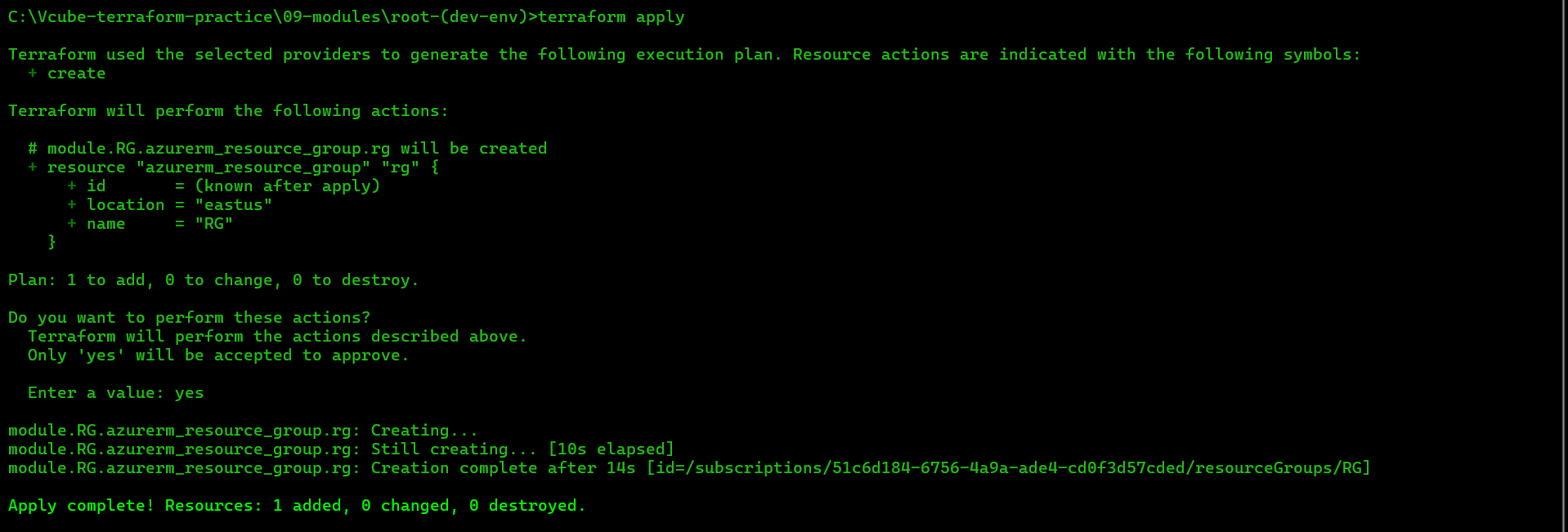


Step2: Create the Resource group module in RG-module directory.

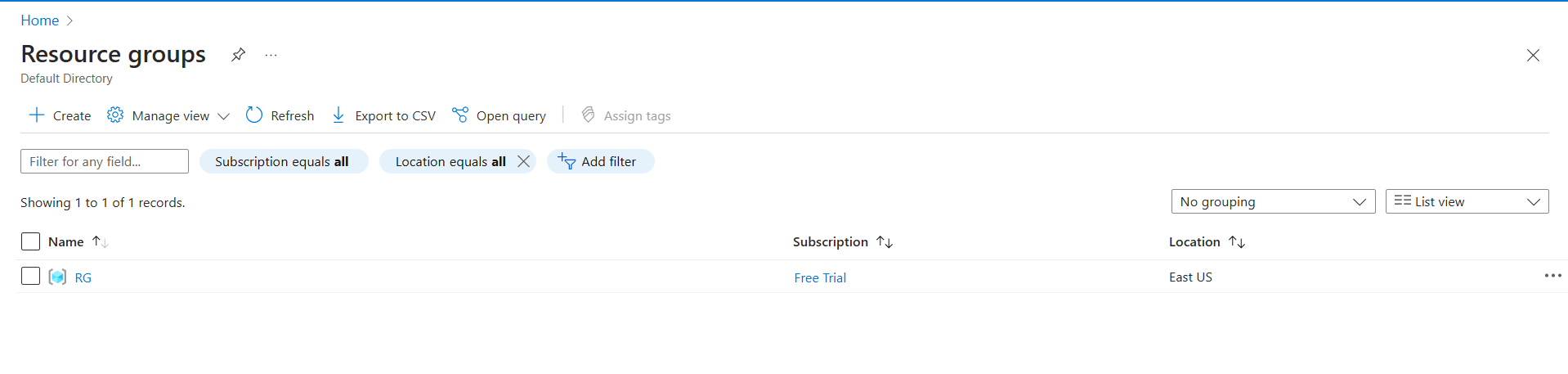
Step3: Now call this RG-module from root directory (dev-env) of main.tf file.



Step3: Then do the terraform apply form the root (dev-env) folder path.



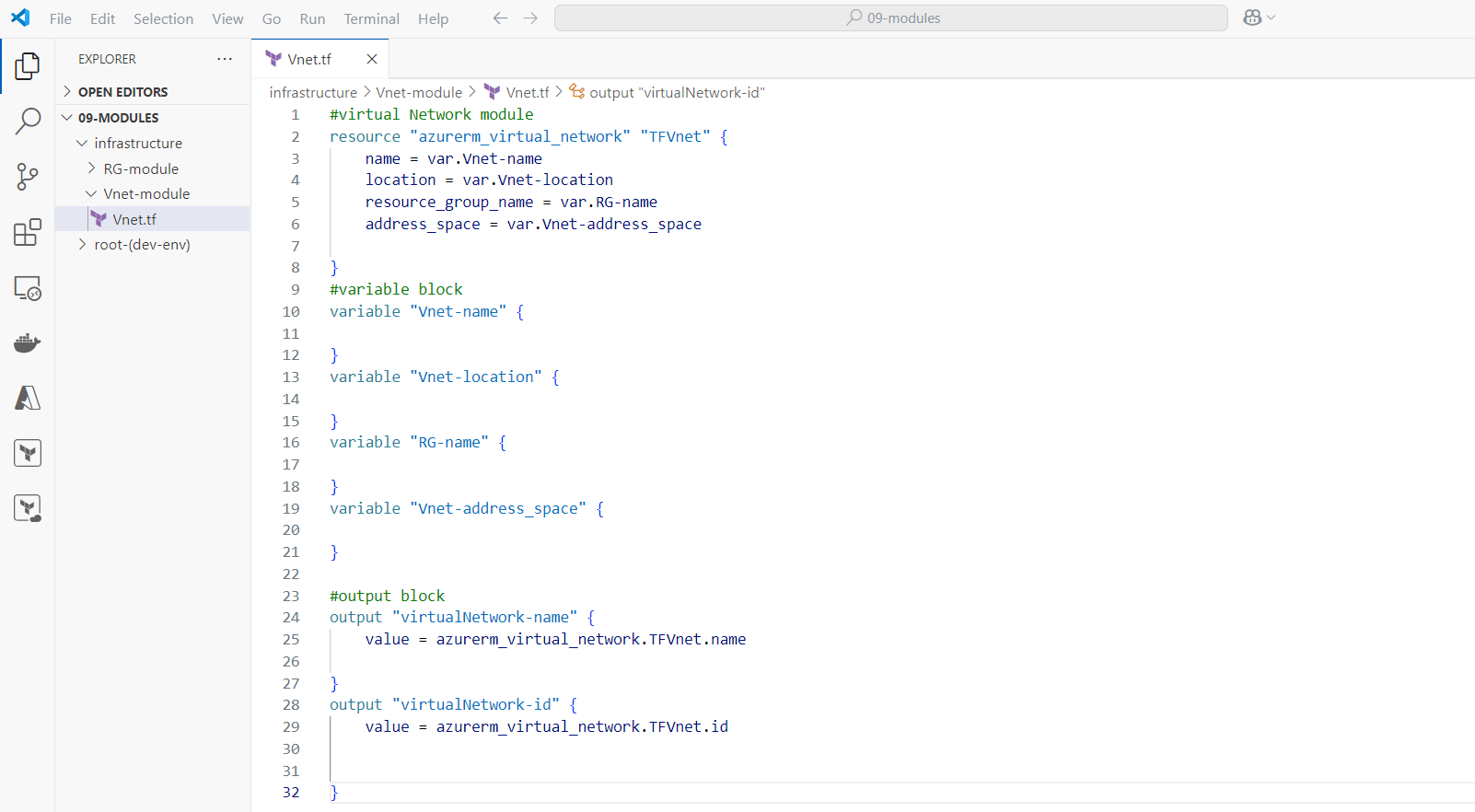
**Fig:** Terraform apply, to create resource group.



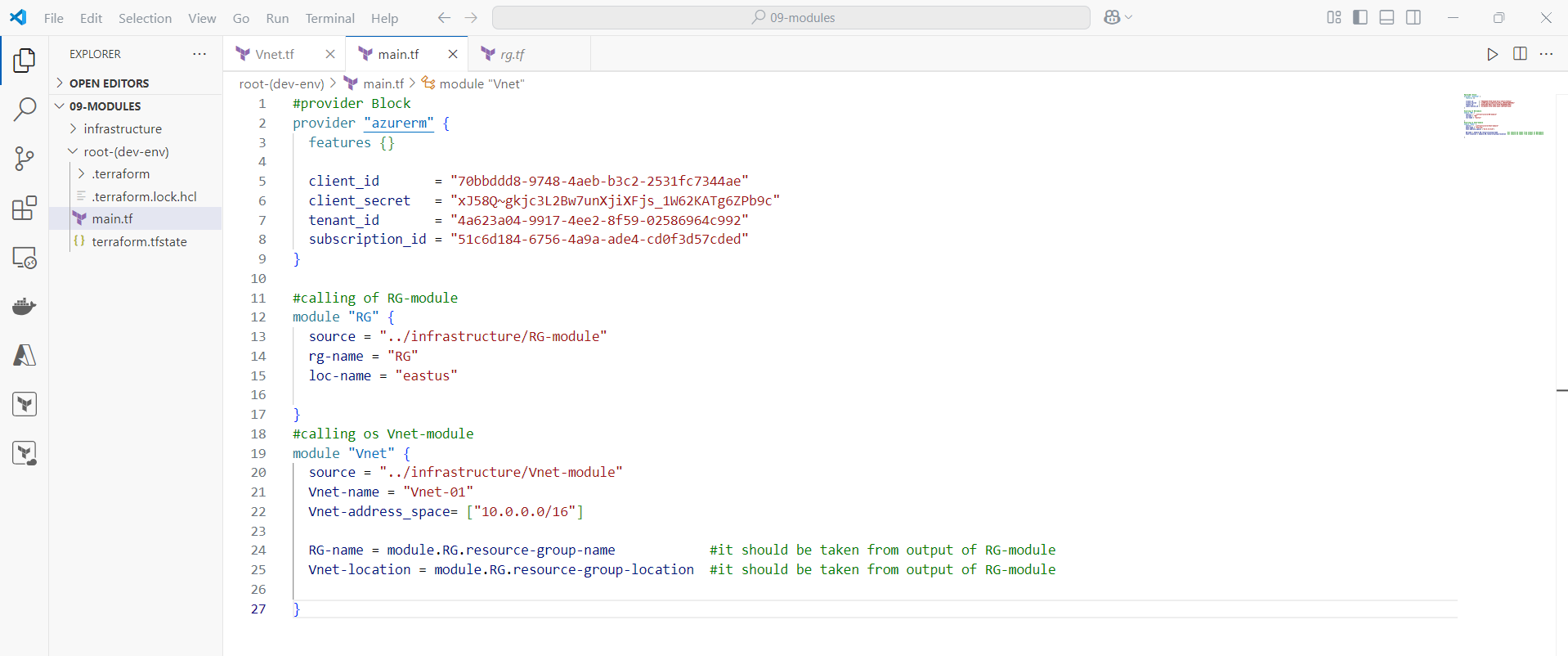
**Fig:** resource group (RG).

**Similarly create the Virtual network using module concept.**

**Step1:** Create a separate directory (Vnet-module) for virtual network and write its terraform code.

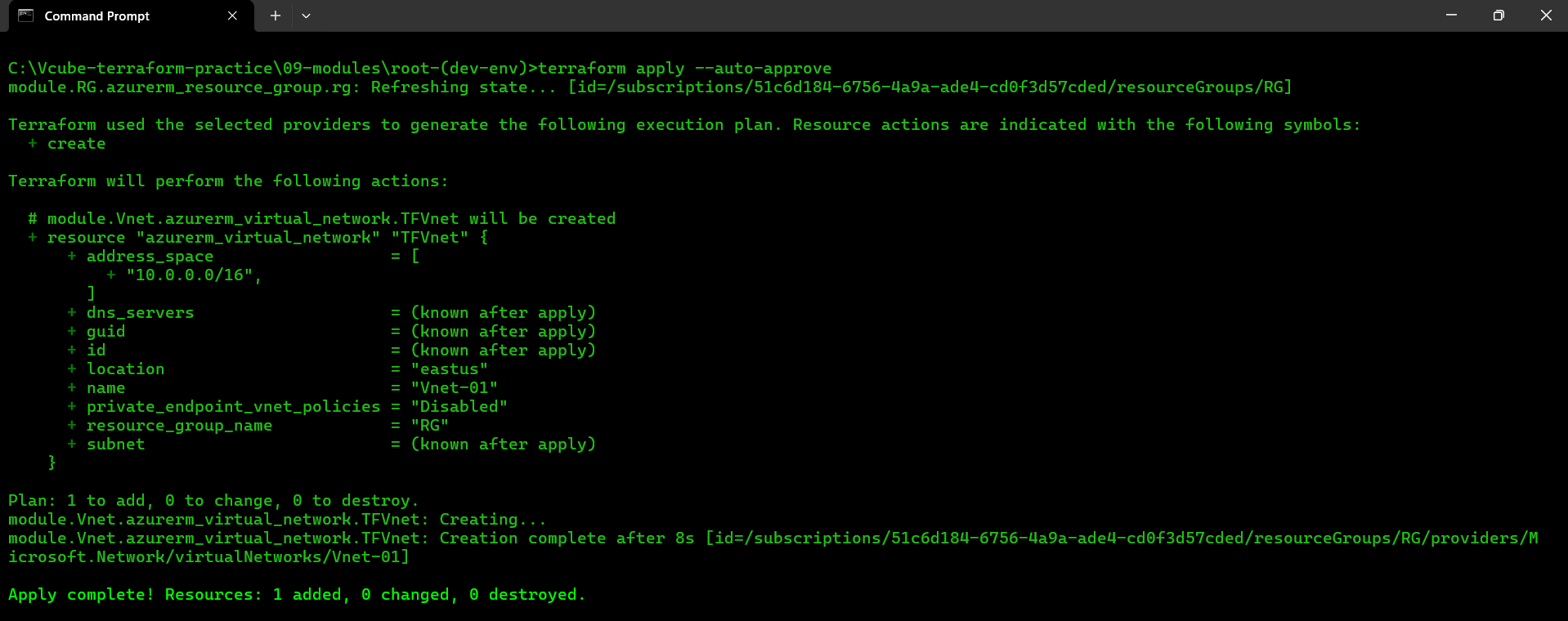


Step2: Now call this child module (Vnet-module) from root directory of main.tf file.

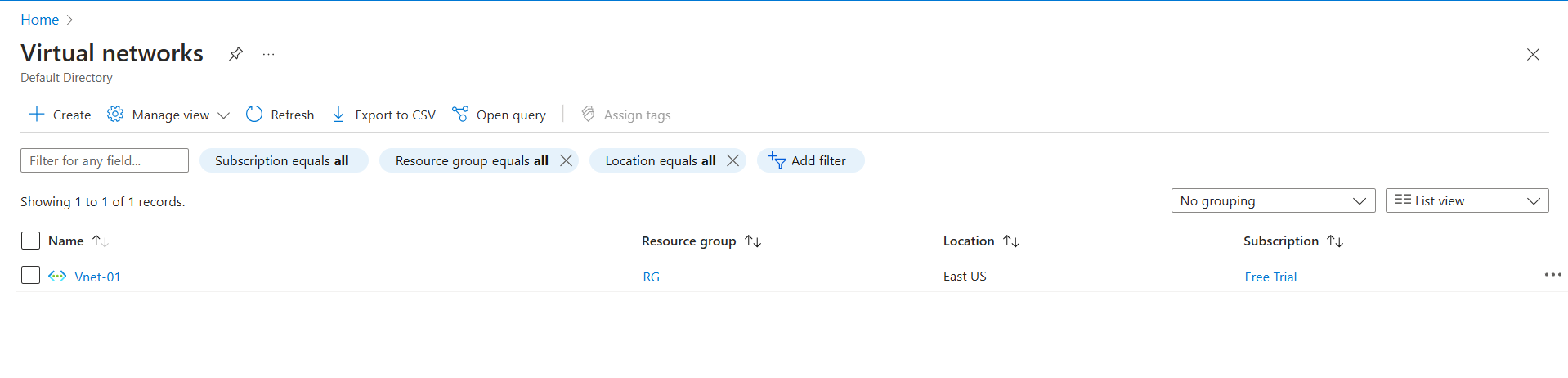


**Fig:** Calling of Vnet-module.

Step3: Then do the terraform apply form the root directory (dev-env) path.



**Fig:** Terraform apply to create Virtual network (Vnet-01).



**Fig:** Virtual Network (Vnet-01).

**Creation of Subnet using the module concept**

Step1: Create the separate directory (Subnet-module) for subnet and write its row code on it.

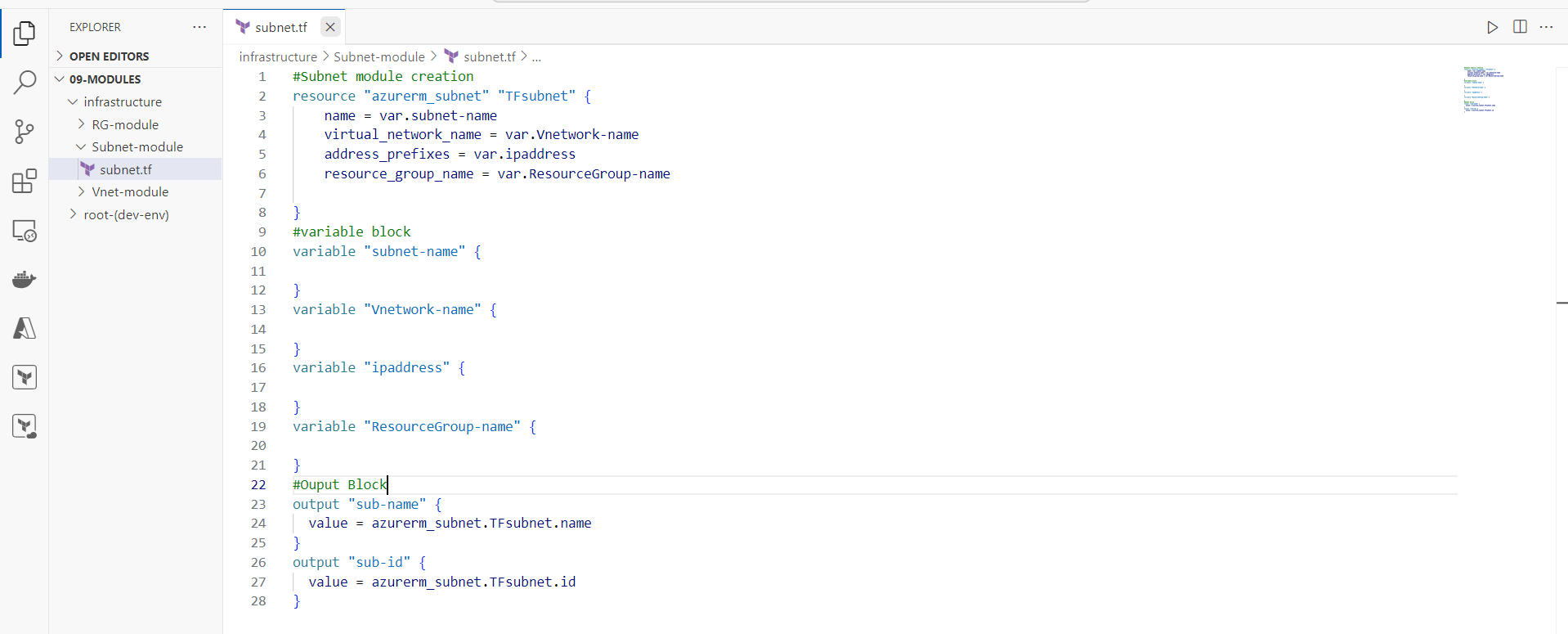
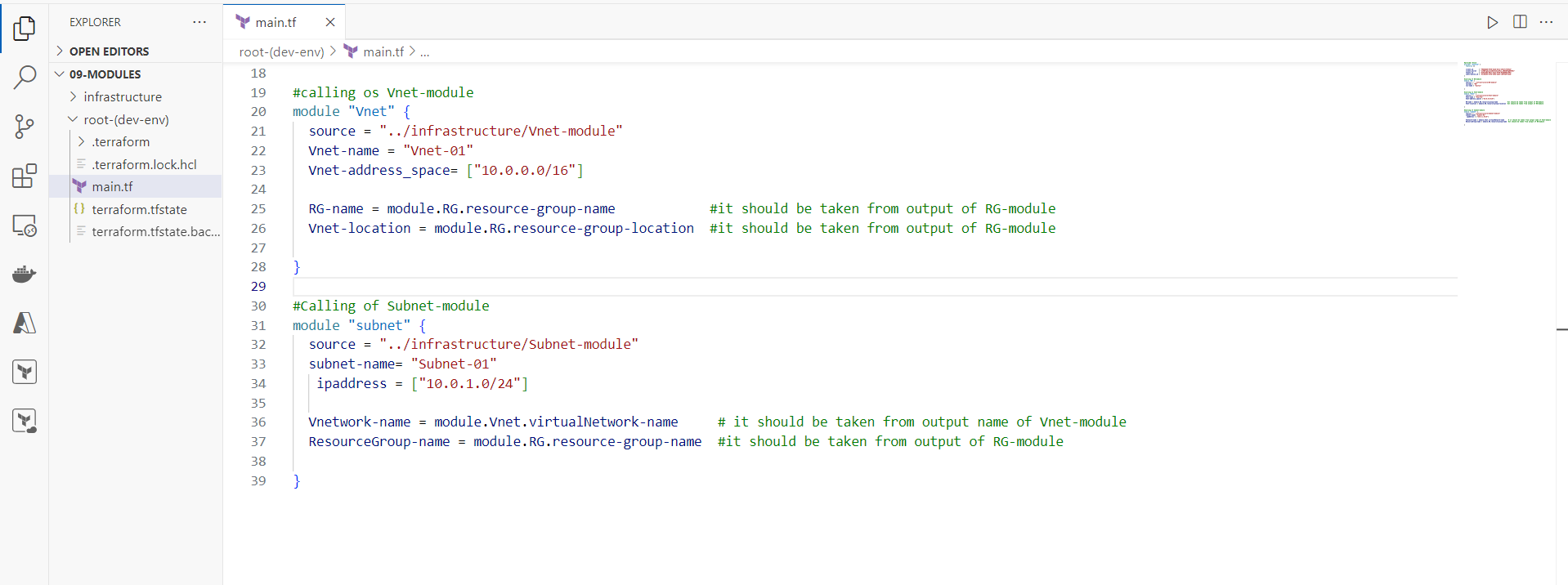


Fig: Subnet Module code in separate directory (Subnet-module).

Step2: Now call the Subnet-module from main.tf file (root module main.tf).

Step3: Do the terraform apply to create the subnet.

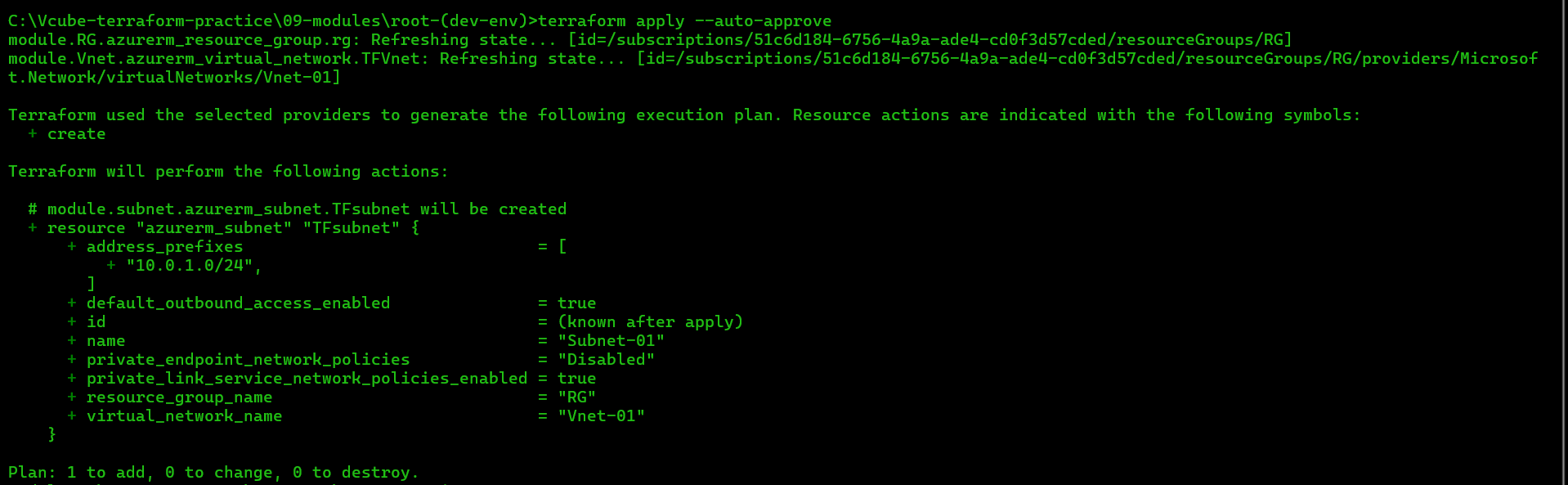


Fig: Terraform apply to create the subnet.

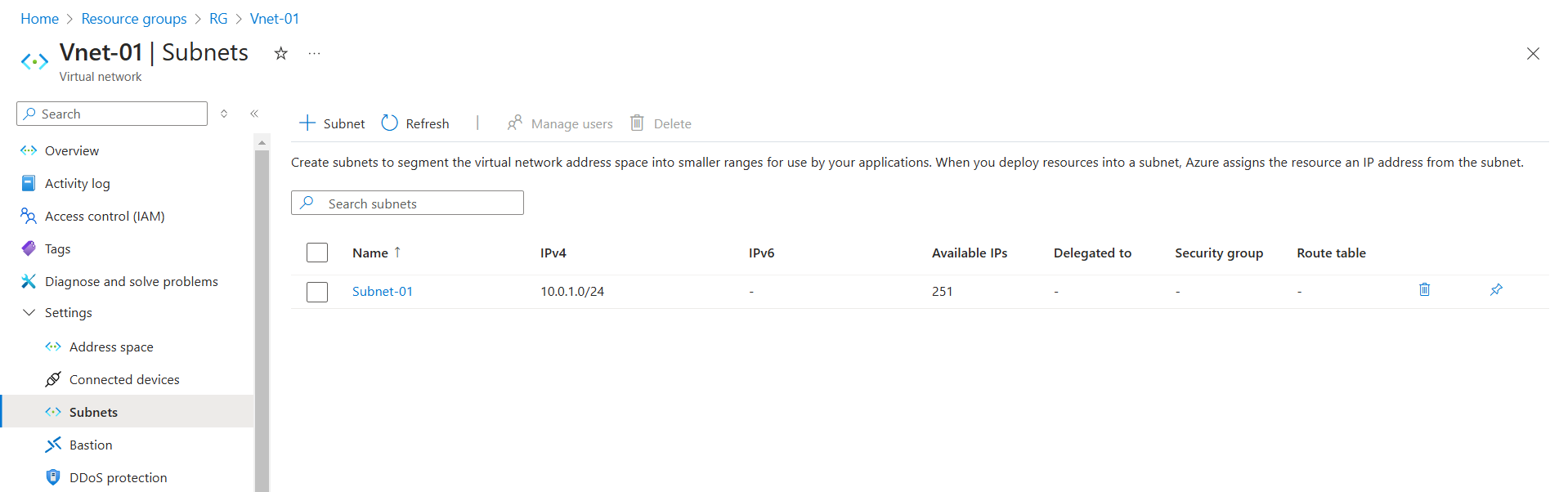


Fig: Subnet (subnet-01) is created.

**Create the Public IP using Terraform module concept**

Step1: Create a separate directory (PIP-module) for public IP and write its terraform code.



Step2: Now call the Public IP module (pip-module) from root directory (main.tf).

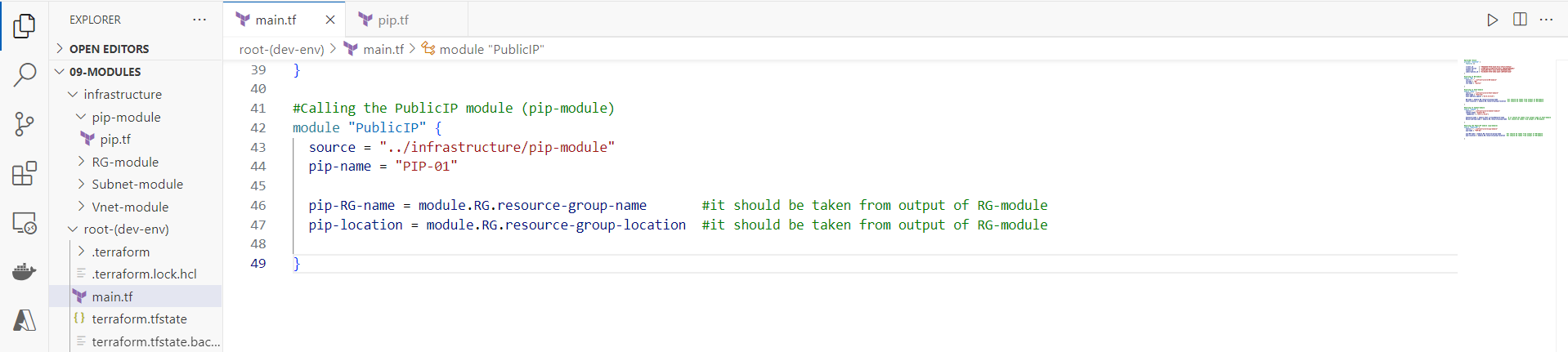


Fig: Calling of pip-module.

Step3: Do the terraform apply to create the public IP.

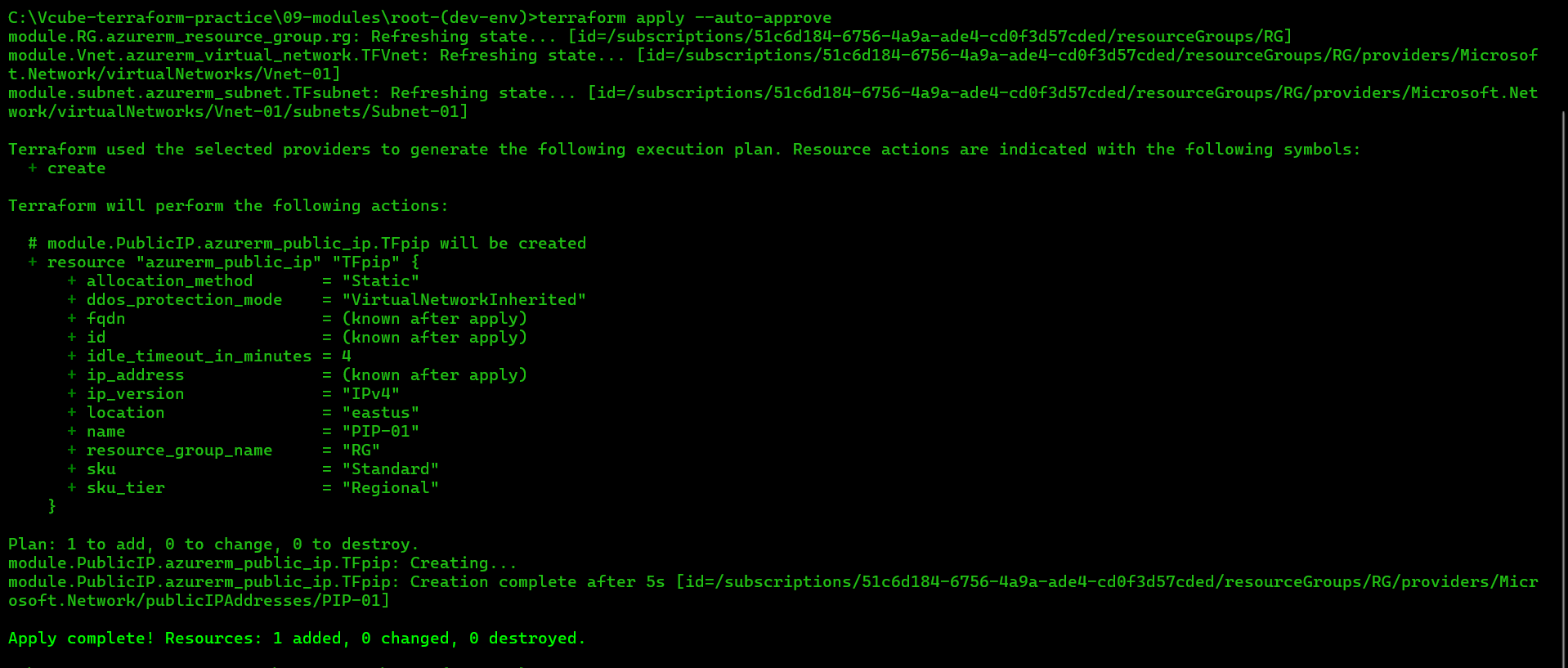


Fig: Terraform apply to create the Public IP.

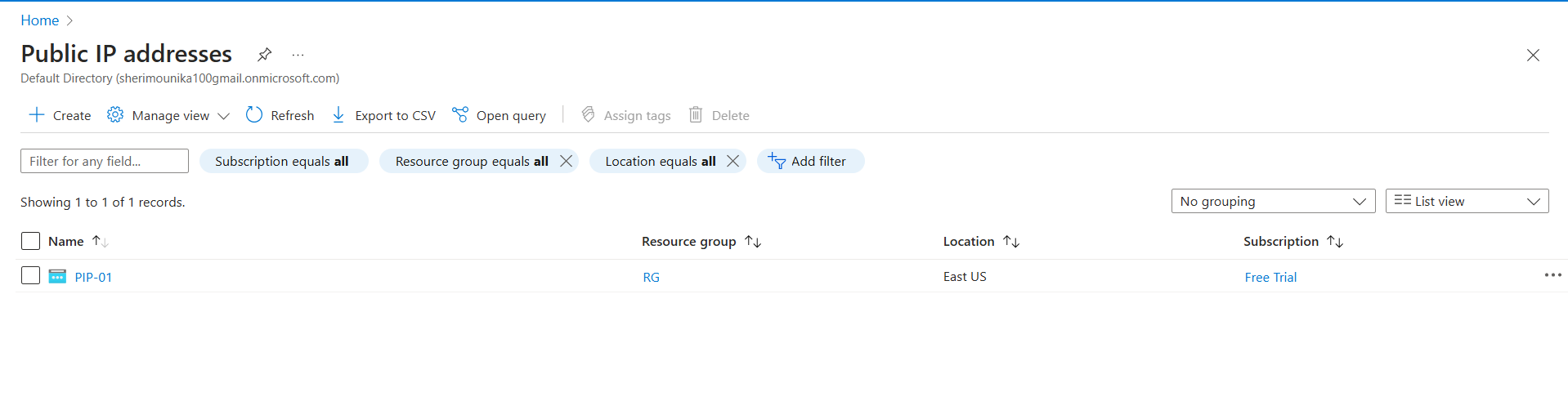
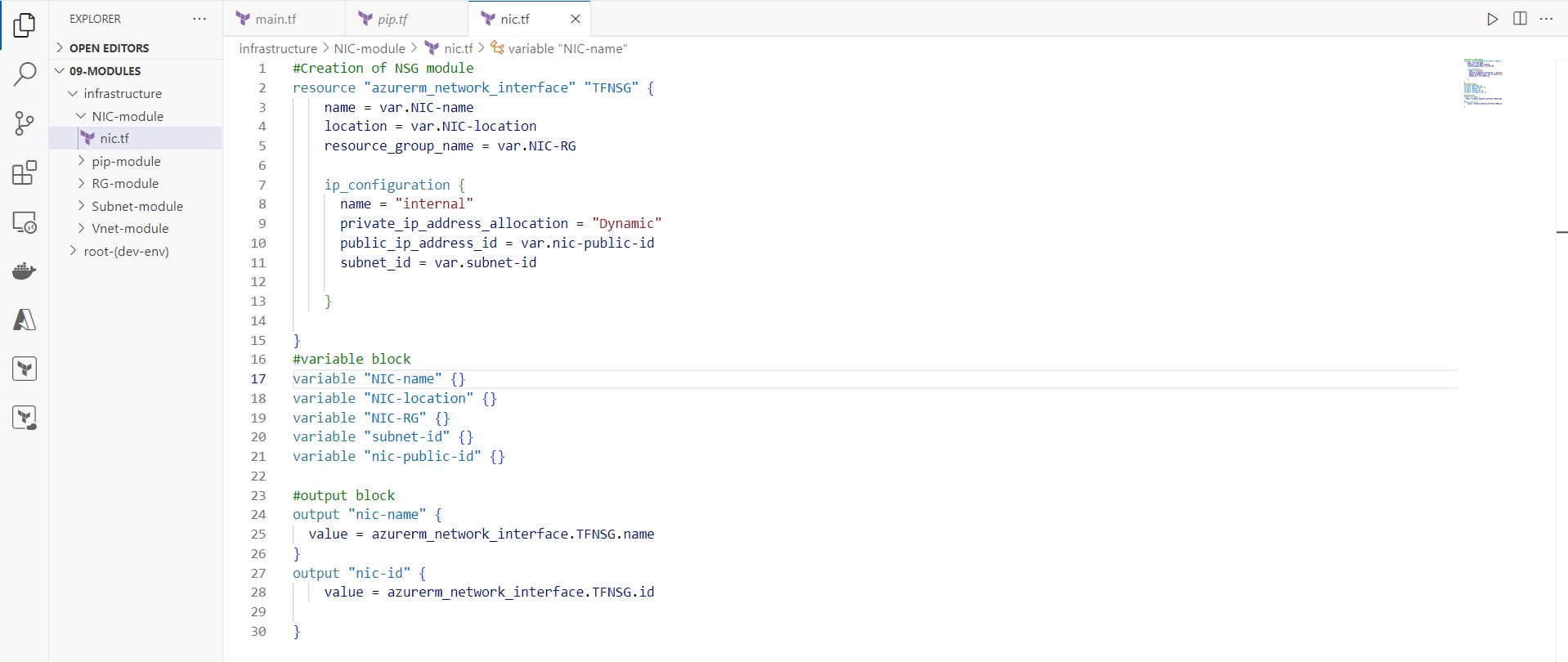


Fig: Public IP (PIP-01).

**Create the Network interface card (NIC) using module concept**

Step1: Create the separate directory (NIC-module) for NIC and write its terraform raw code.



Step2: Now call the NIC-module from root directory (root-dev-env) of main.tf.

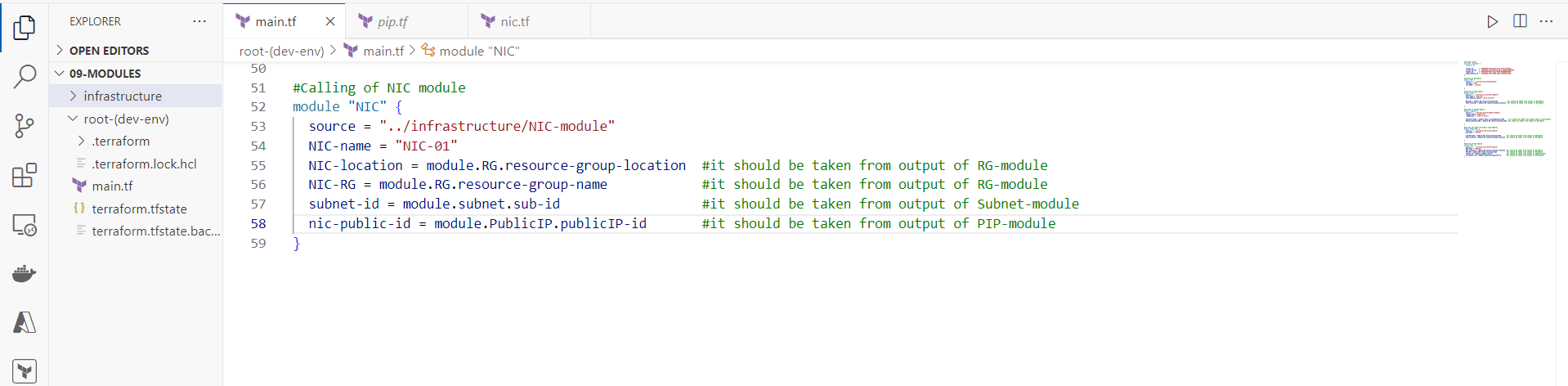


Fig: Calling of NIC-module.

Step3: Now do the terraform apply to create the NIC card (NIC-01).

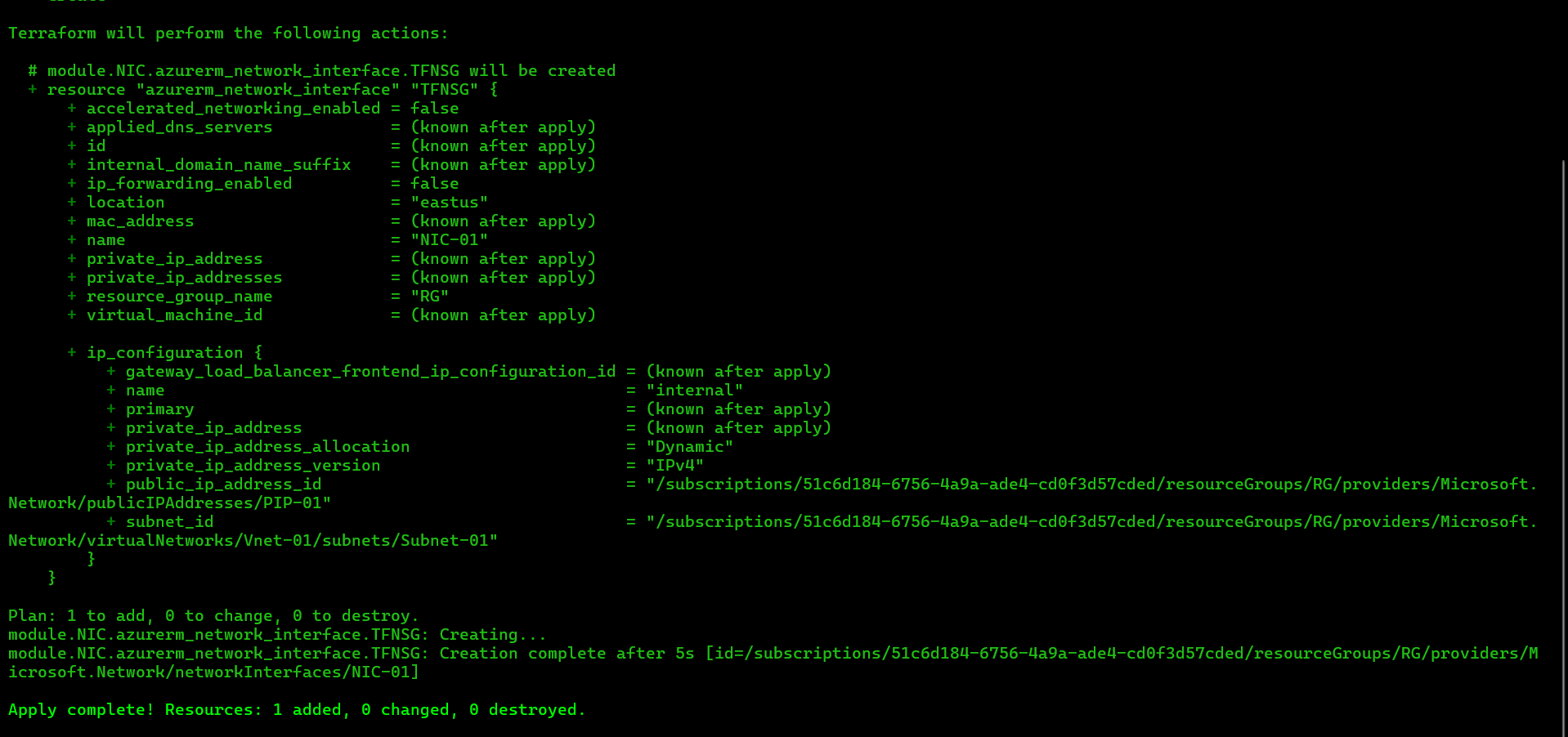


Fig: Terraform apply to create the NIC.

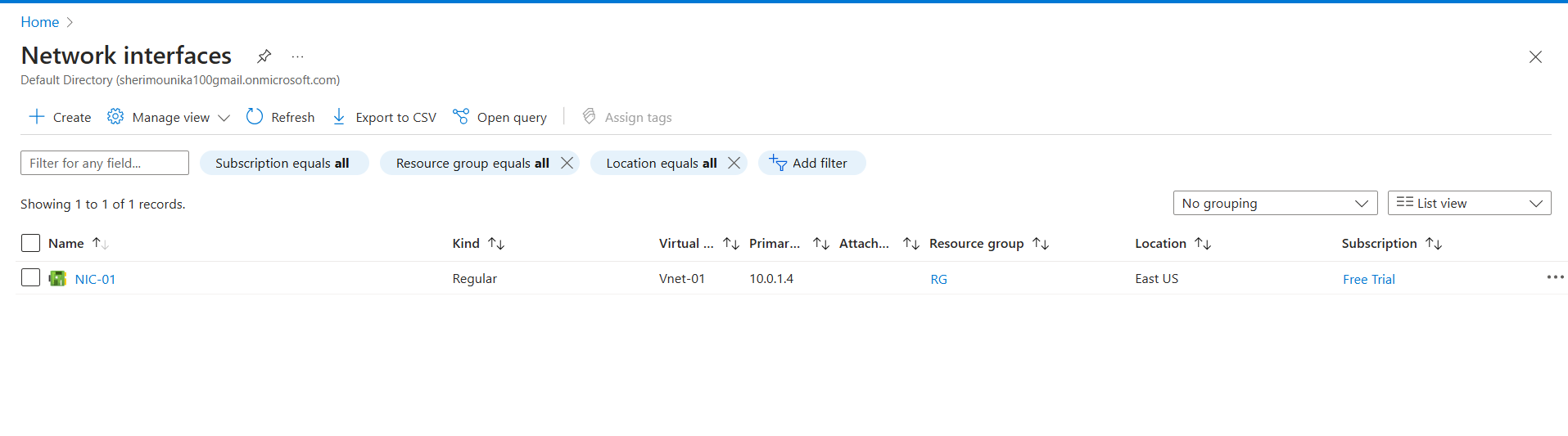
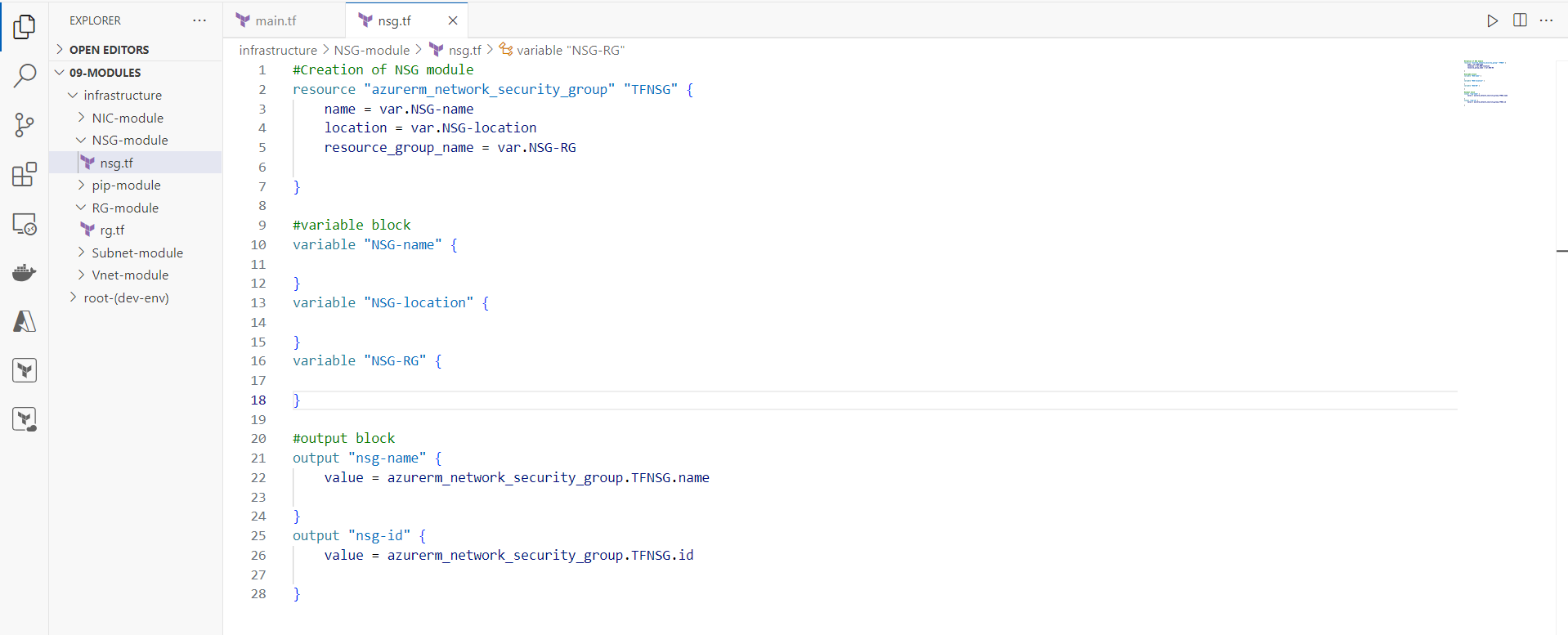


Fig: Network interface card (NIC-01).

**Create the Network security group using module concept**

Step1: Create the separate directory (NSG-module) for NSG and write its raw code.

Step2: Now call the NSG-module from root directory (dev-env) main.tf file.

Fig: calling of NSG module.

Step3: Do the terraform apply to create the Network security group (NSG-01).

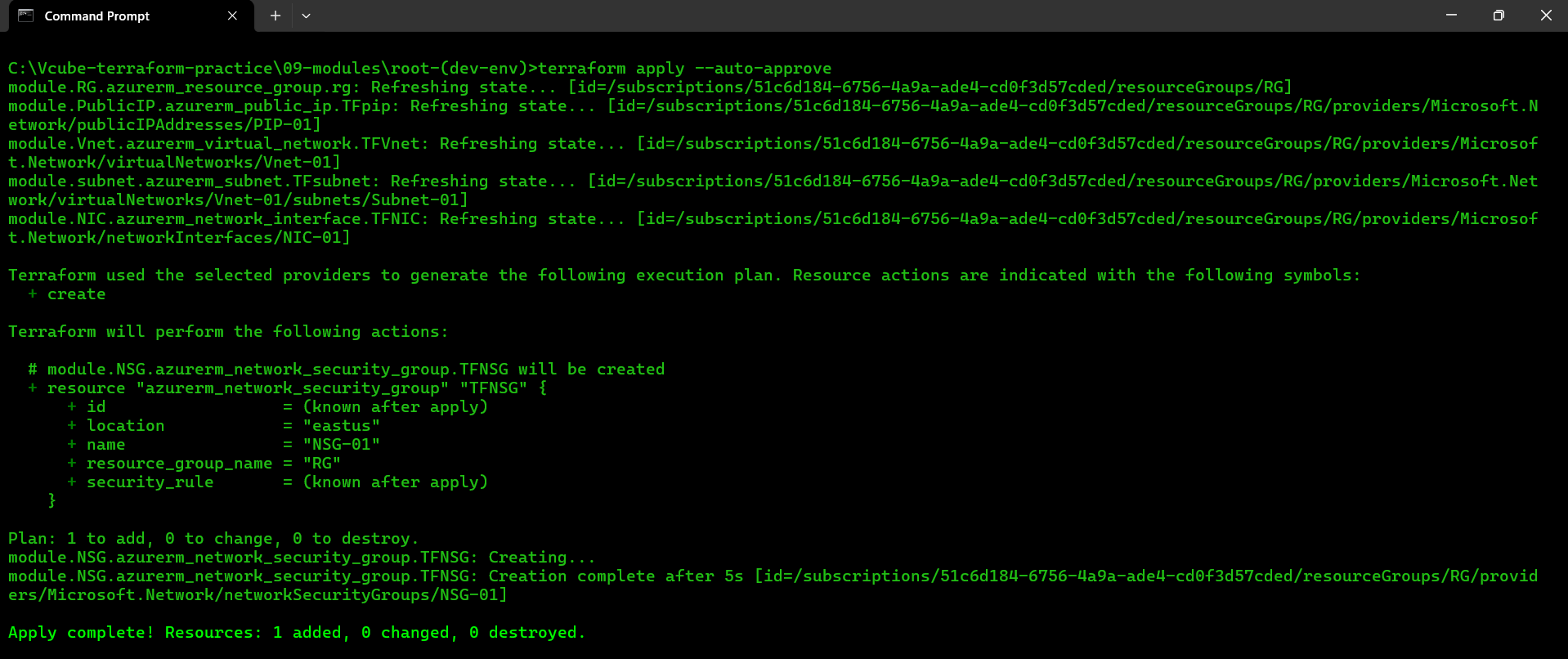


Fig: Terraform apply to create NSG-01.

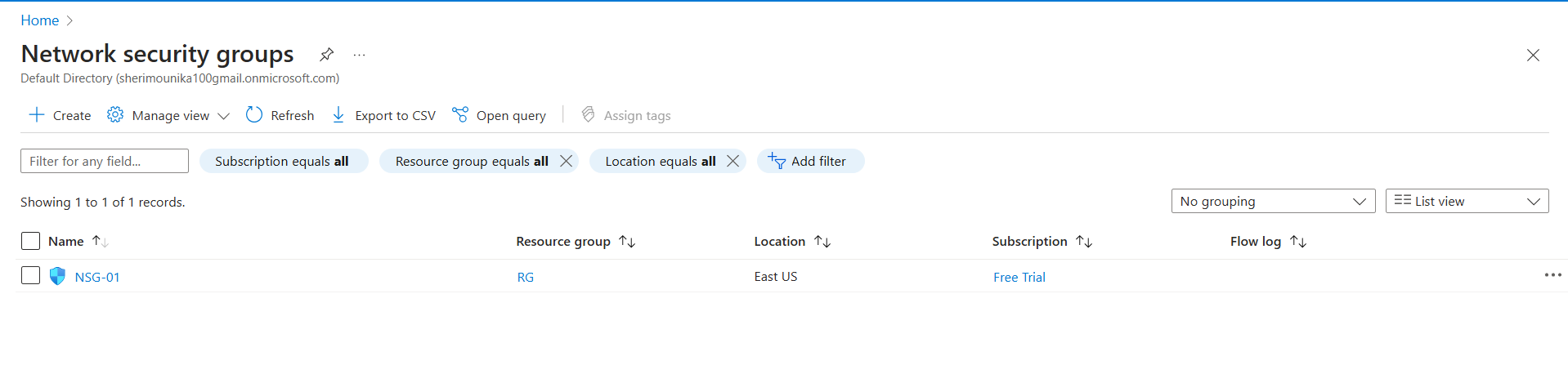


Fig: NSG-01.

**Create the NSG rules and assign it to the subnet using module concept**

Step1: Create a separate directory (rules-module) for NSG rule and write its code.



Fig: rule.tf configuration file



Fig: variable & output block of rule.tf file (Rules-module).

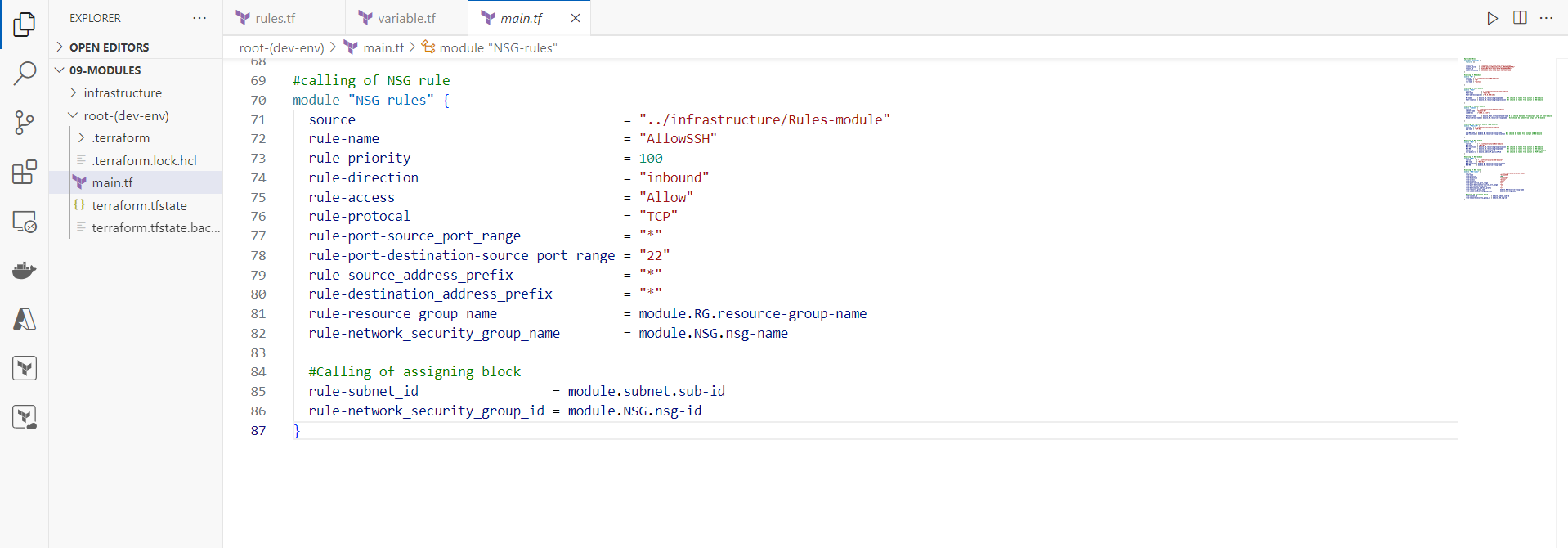
Step2: Now call the Rule module from root directory (dev-env) of main.tf file.

Fig: Calling of Rule-module.

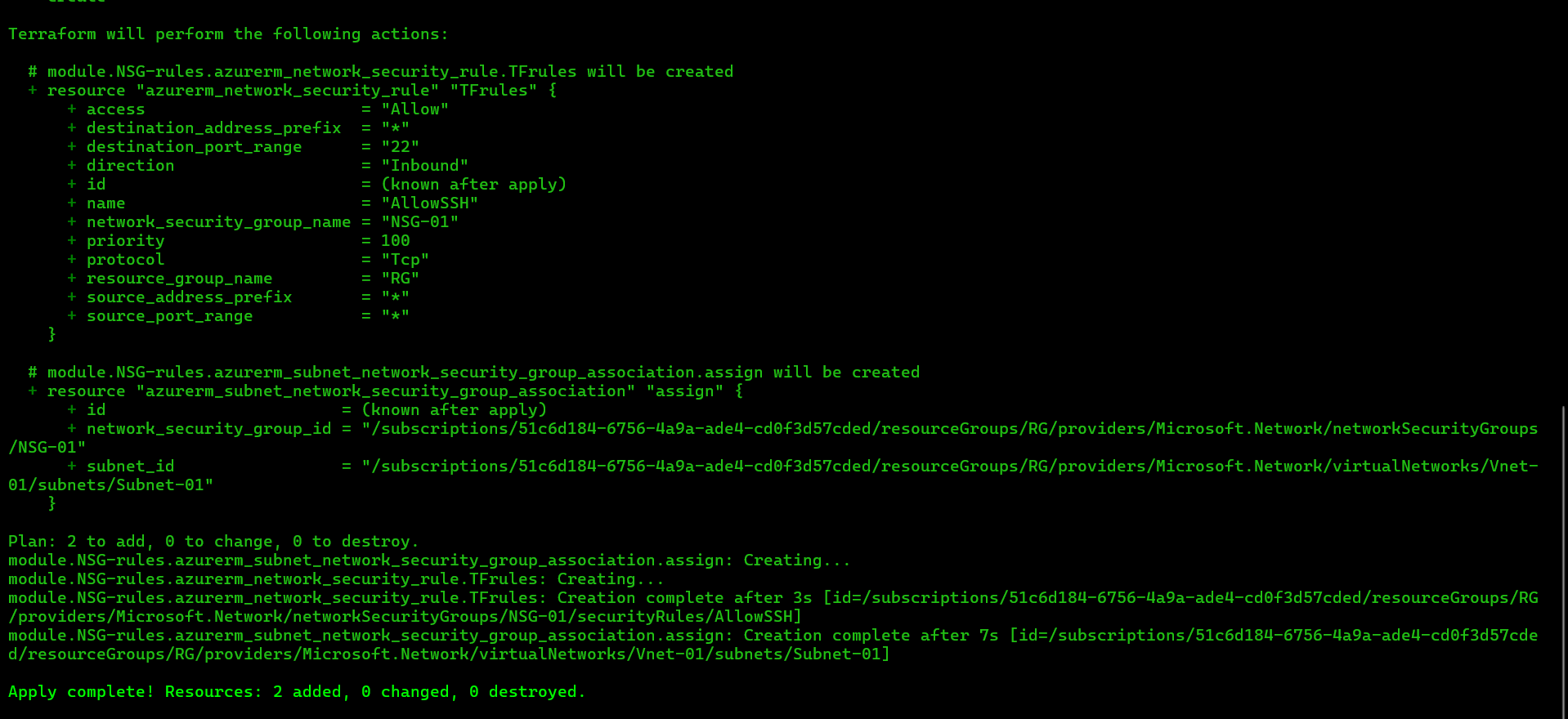
Step3: Do the terraform apply to create the NSG rules and assigning it to the Subnet-01

Fig: Terraform apply to create NSG rules.

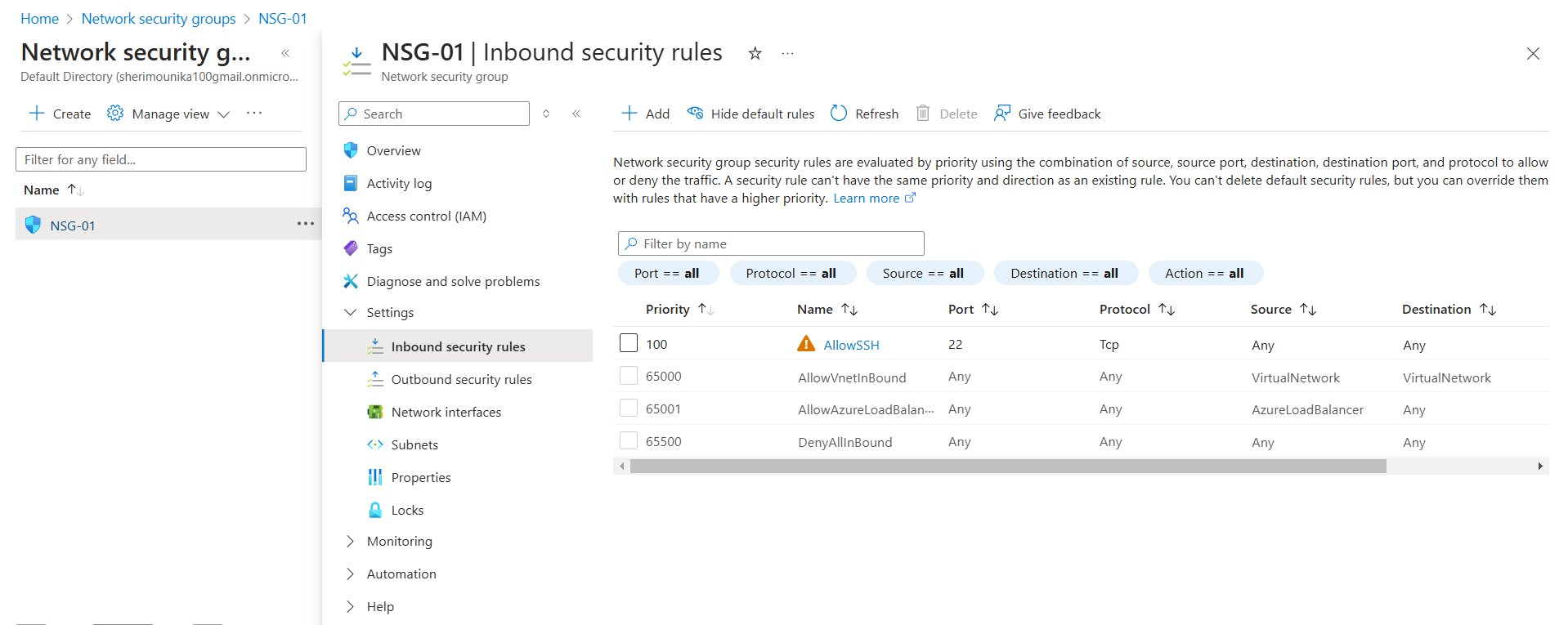


Fig: NSG rules allow SSH port.

**Create the Virtual Machine using module concept**

Step1: Create the separate directory (VM-module) for VM and write its terraform code.

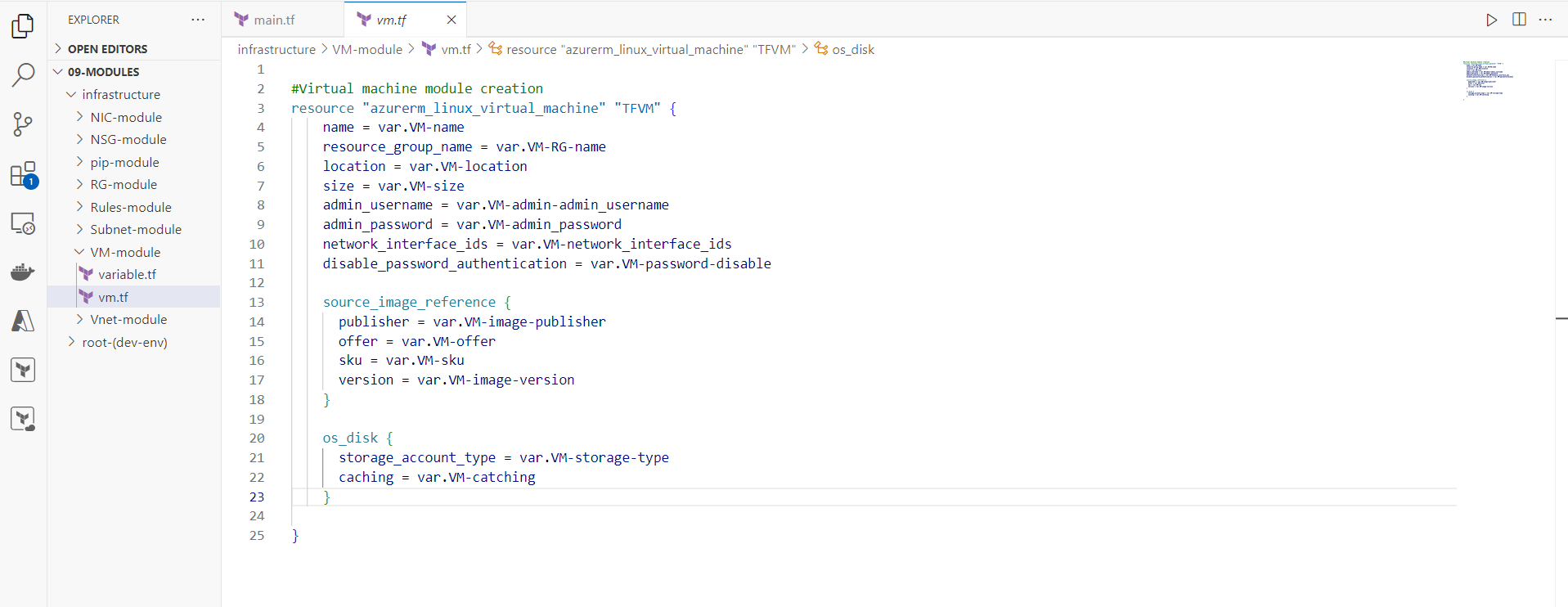


Fig: VM creation Module.

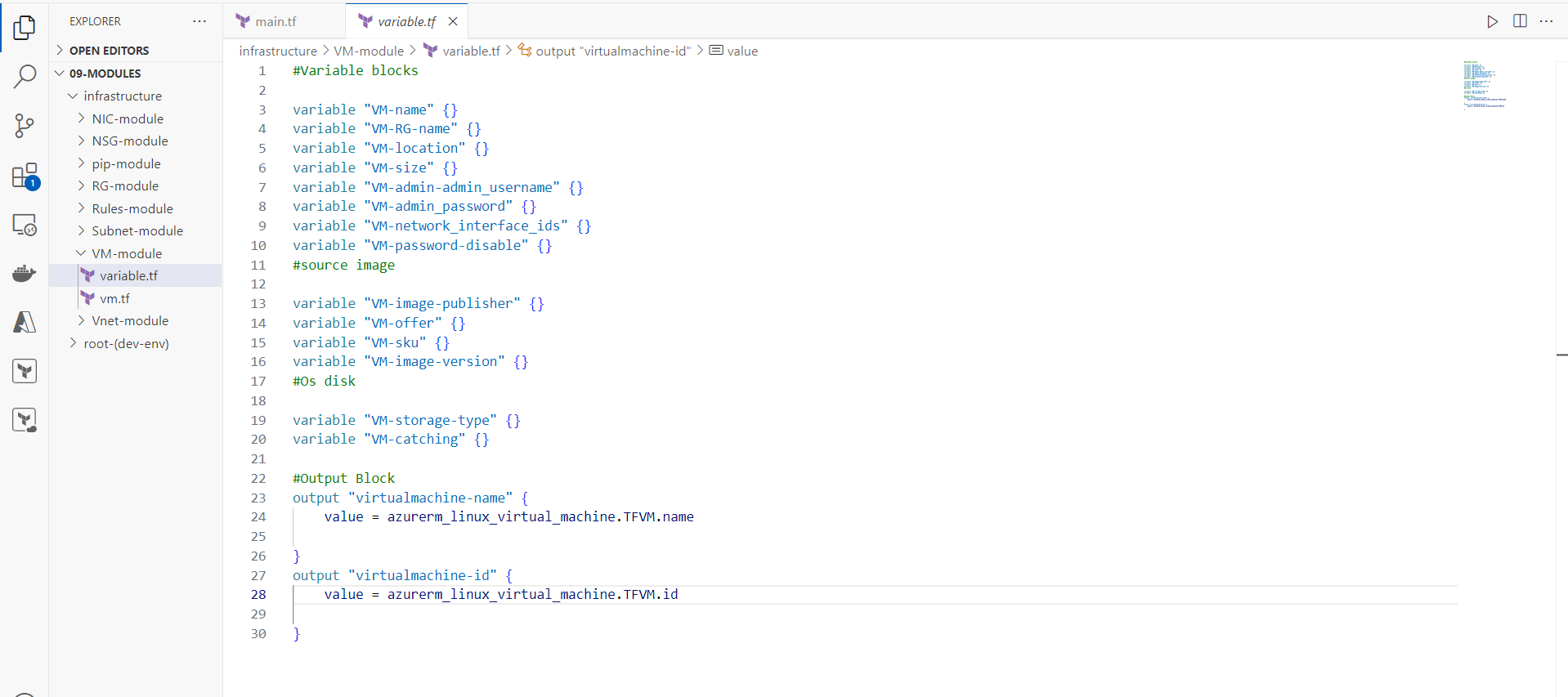


Fig: Variable & output blocks of VM-module.

Step2: Now call the VM-module from the root directory of main.tf file.

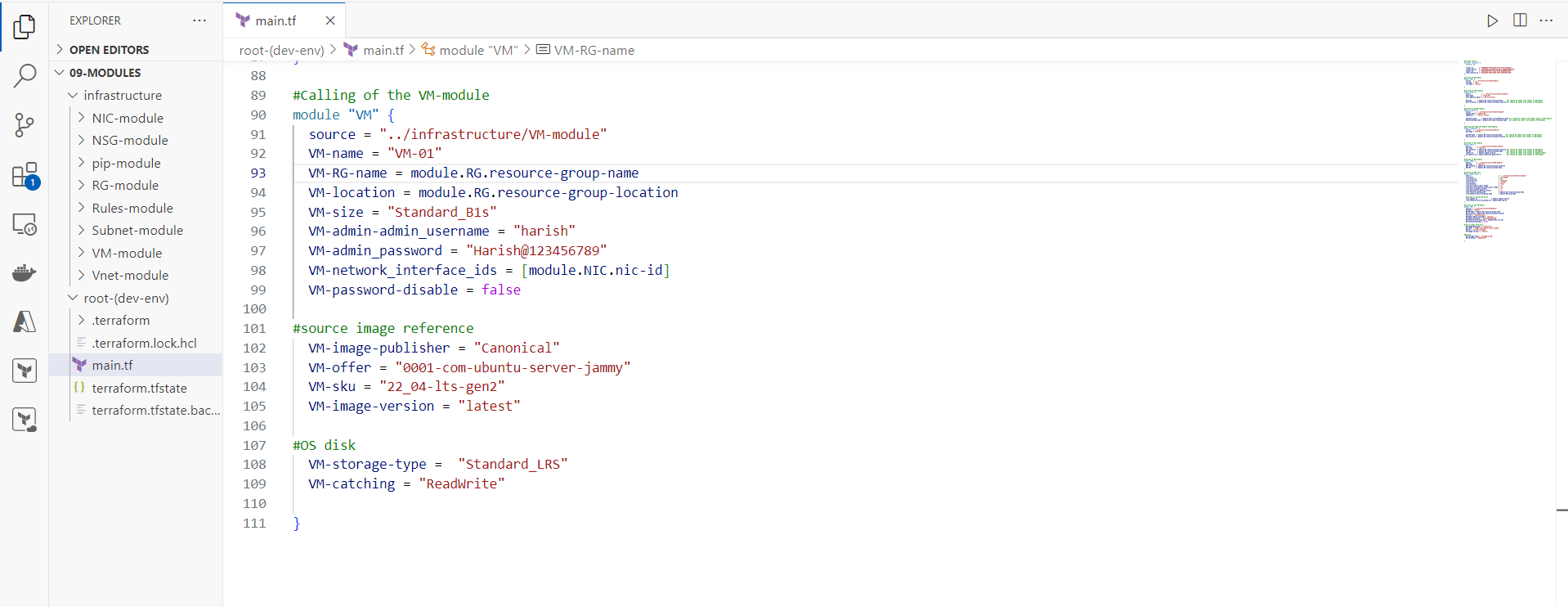


Fig: Calling of VM-module from main.tf file.

Step3: Do the terraform apply in order to create the VM from main.tf file path.

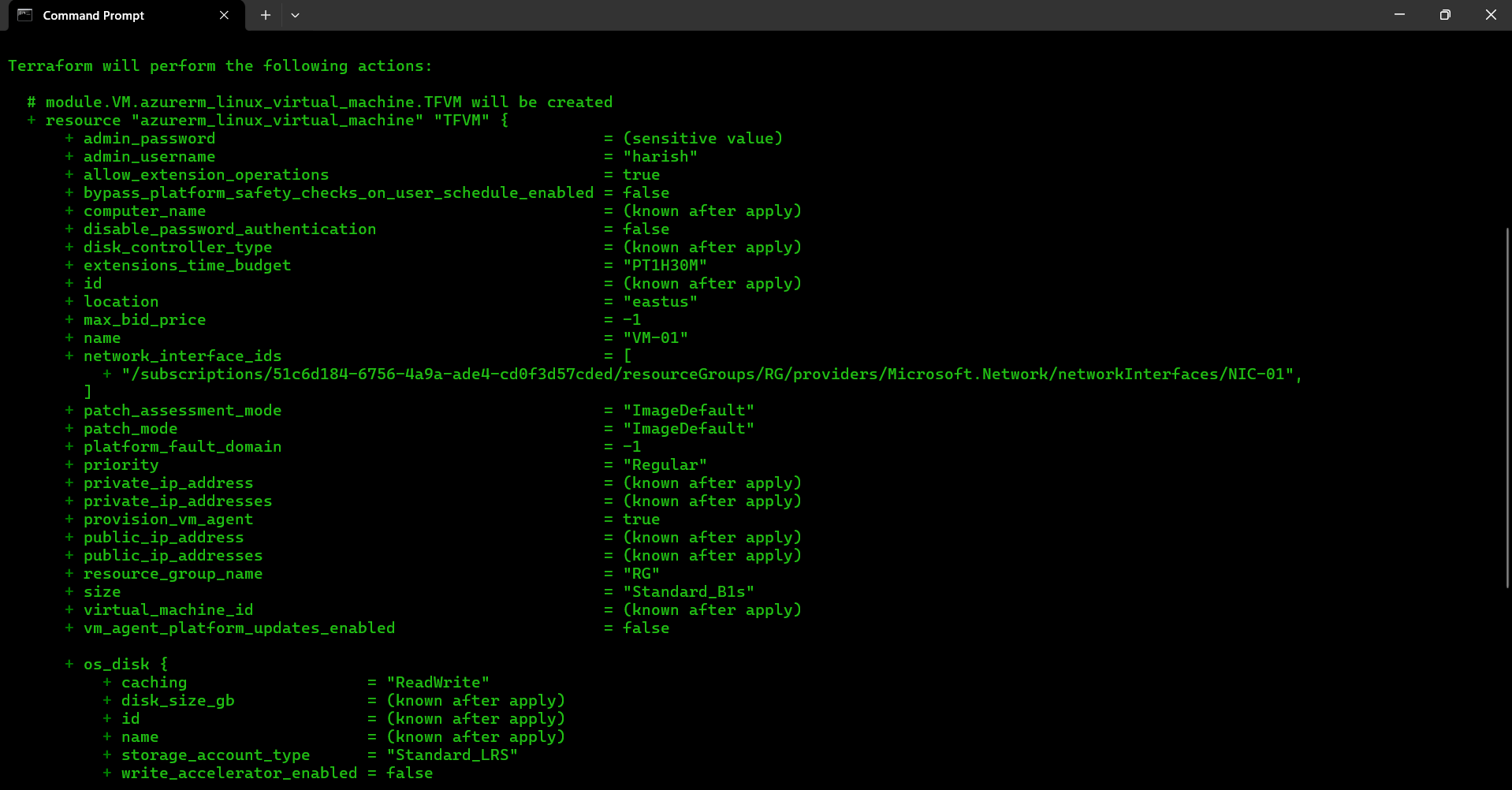


Fig: Terraform apply to create VM.

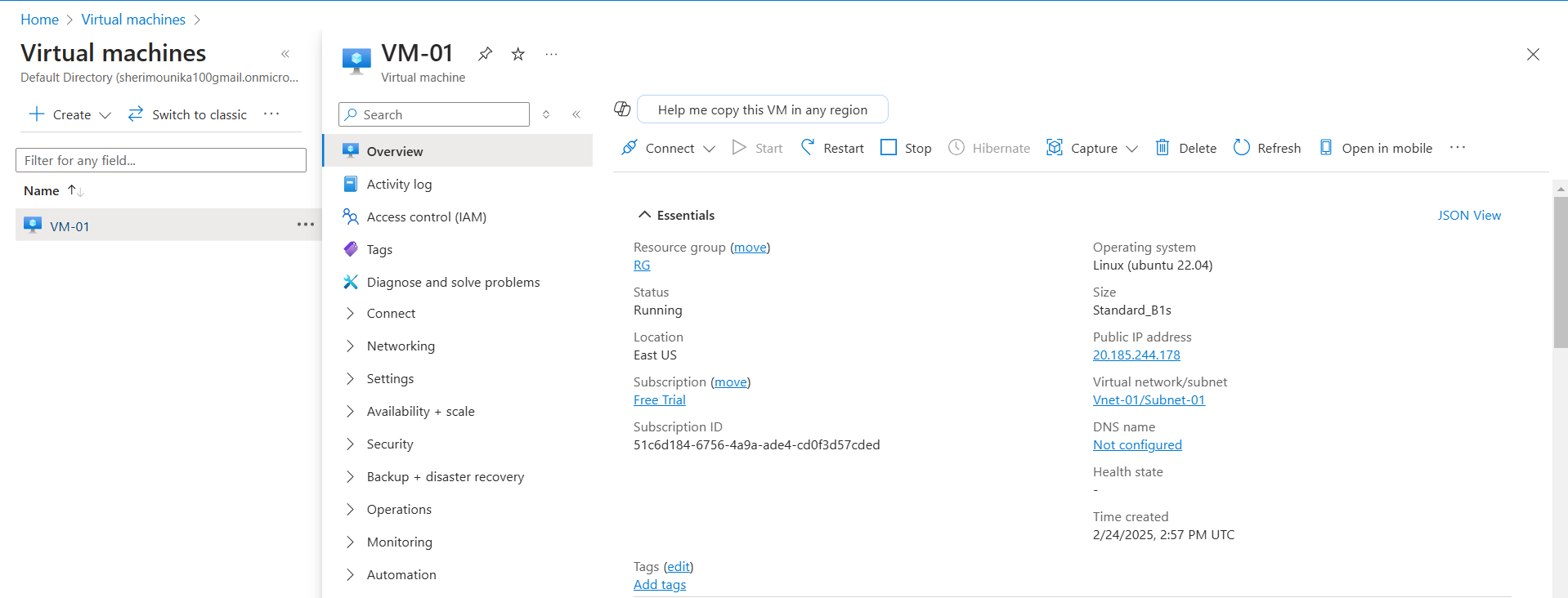


Fig: VM-01