**Basics of variables in terraform**

In Terraform, **variables** are used to make configurations more flexible, reusable, and maintainable. They allow you to parameterize values **instead of hardcoding** them into the configuration files.

**Key Uses of Variables in Terraform:**

1. **Parameterization** – Allows dynamic values to be passed instead of hardcoding them.
2. **Reusability** – The same Terraform configuration can be used with different values.
3. **Consistency** – Ensures that values remain the same across different modules and environments.
4. **Security** – Sensitive values like passwords and API keys can be stored securely.
5. **Scalability** – Makes it easy to deploy infrastructure with different configurations.

The main use of Terraform variables is to **make your infrastructure code reusable and adaptable**.

Generally Variables can be declared in separate terraform configuration file called **“variable.tf”** file.

In terraform a variable block mainly consist of three parameters

1. default
2. type (optional)
3. description (optional)
4. **Default:** It used to store the values of the variable.
5. **Type:** It is a data type. It may be string, number, Boolean, list, map, tuple and count.
6. **Description:** It is used to describe what the variable is used for.

Variable “example” {

default = “RG1”

type = string

description = “resource-Group-name”

Ex:

**Note:** If we not defined the “**type**” in the variable block it automatically set’s **“any”** bydefault.

Let us work with data types in the variables:

1. **string (“ ”) :**

#provider Block

provider "azurerm" {

  features {}

  client\_id       = "1f79e427-2ac4-4eb6-9ca0-f4dd4b3f31ee"

  client\_secret   = "KDQ8Q~H-\_St9118keMeU-ADFzsiY.3y.GMYEnbeS"

  tenant\_id       = "4a623a04-9917-4ee2-8f59-02586964c992"

  subscription\_id = "51c6d184-6756-4a9a-ade4-cd0f3d57cded"

}

#resource block

resource "resource\_group" "TFRG" {

    name = var.rg-name

    location = var.loc-name

}

**Fig:** main.tf file.

variable "rg-name"{

    type = string

    default = "RG-01"

    description = "resource group name"

}

variable "loc-name" {

    type = string

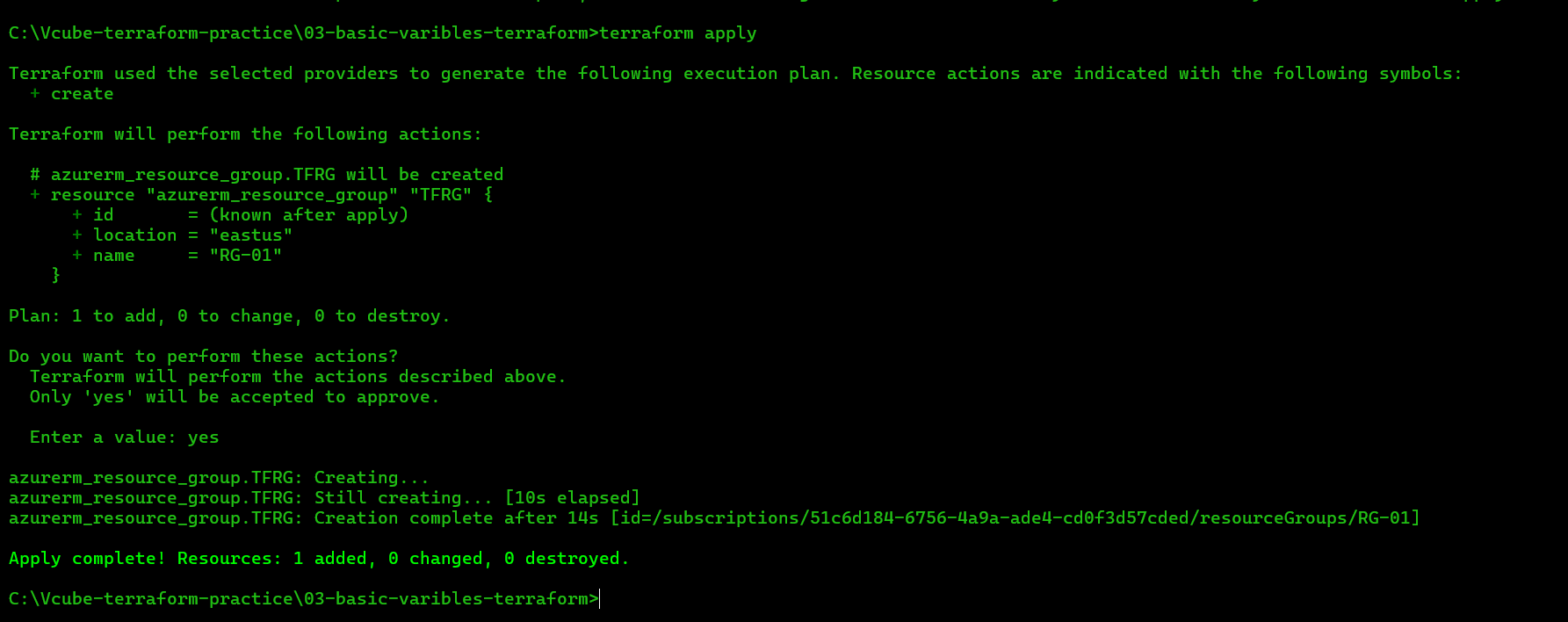
    default = "eastus"

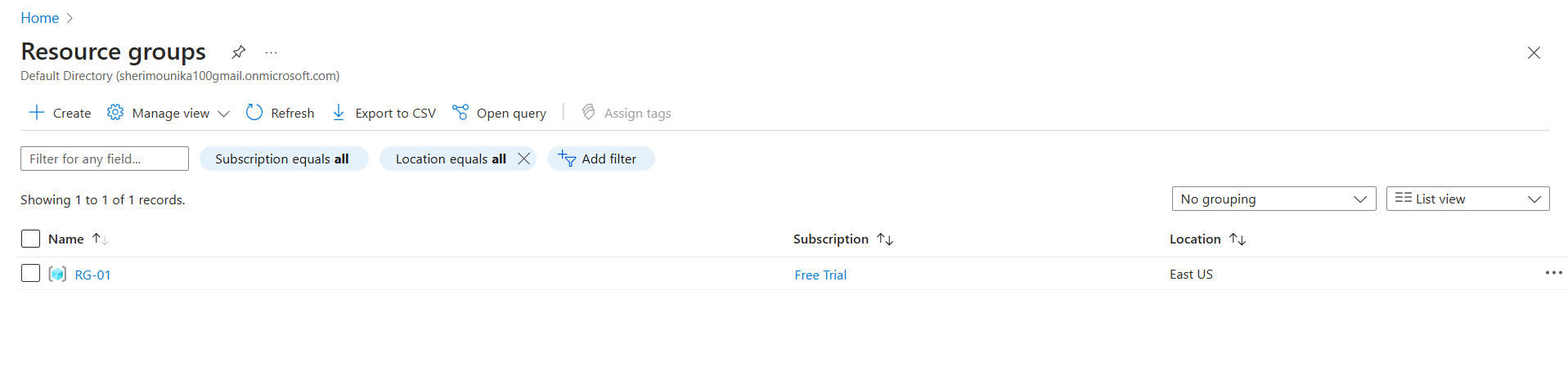
    description = "location name"

}

**Fig:** variable.tf file

Now execute the terraform configuration file using command **“terraform apply”.**





**Fig:** resource group is created using string data type in the variable block.

1. **List []** (index values start from “0”):

#provider Block

provider "azurerm" {

  features {}

  client\_id       = "1f79e427-2ac4-4eb6-9ca0-f4dd4b3f31ee"

  client\_secret   = "WJd8Q~OsrZms4FE1ZdY5QUTP1Or3LzIhrwlyOc3S"

  tenant\_id       = "4a623a04-9917-4ee2-8f59-02586964c992"

  subscription\_id = "51c6d184-6756-4a9a-ade4-cd0f3d57cded"

}

#resource block

resource "azurerm\_resource\_group" "TFRG" {

  name     = var.rg-name[0]

  location = var.loc-name[1]

}

**Fig:** main.tf file

variable "rg-name" {

  type        = list

  default     = ["RG-01", "RG1", "rg"]

  description = "list of resource group names"

}

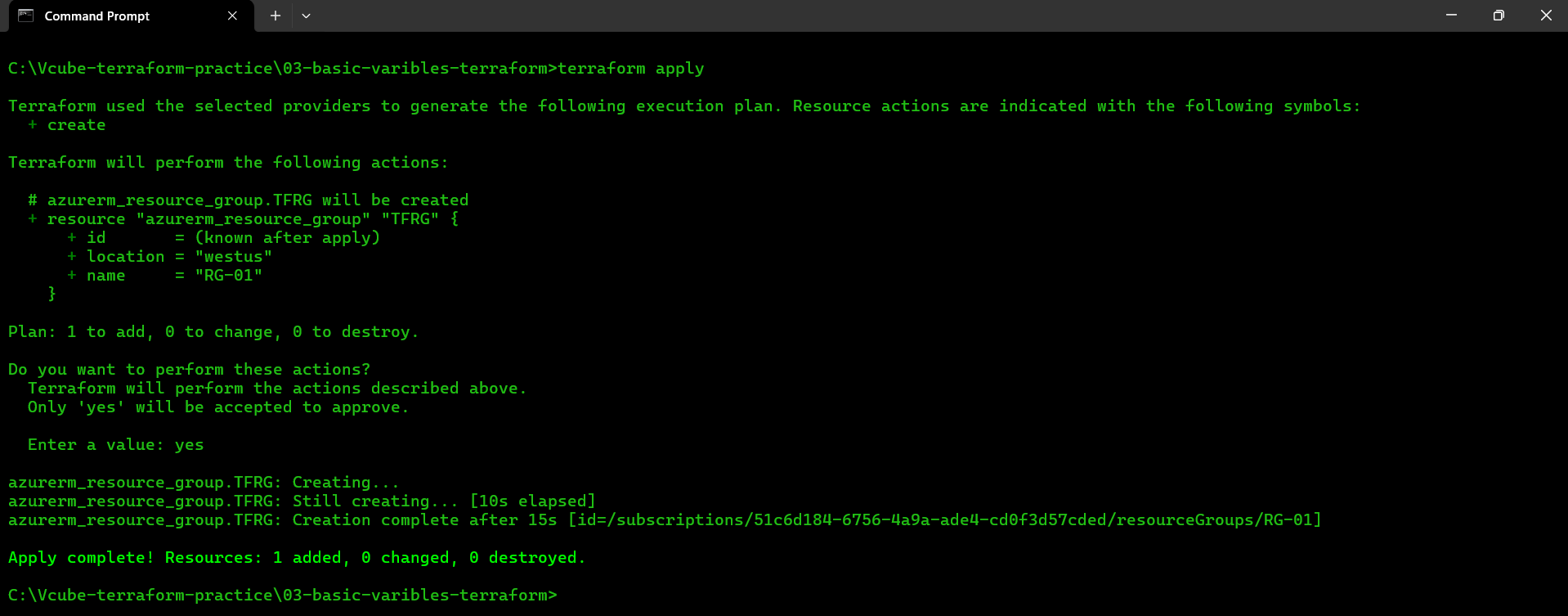
variable "loc-name" {

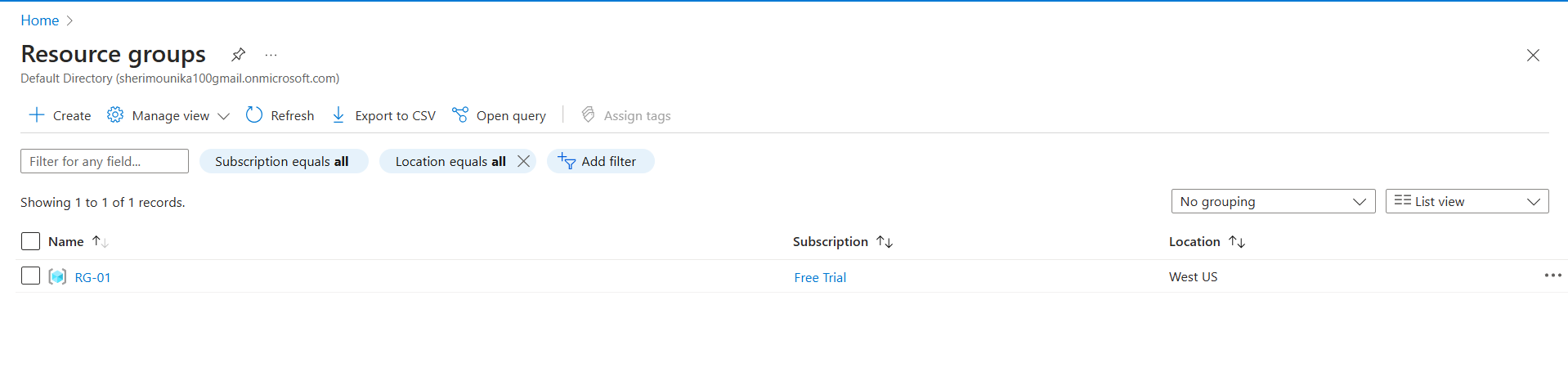
  type        = list

  default     = ["eastus", "westus", "centralindia"]

  description = "list of location name"

}

**Fig:** variable.tf file.



**Fig:** Resource group is created using **“list”** data type in variable block of terraform.

1. **Map {key-value pairs}:**

* In Terraform, a map is a data type that represents a collection of key-value pairs.
* **Unique keys:** Every key in a map must be unique. You can't have duplicate keys. And keys are always strings.
* **Values can be any data type:** While the values in a map can be any data type that Terraform supports.

**Ex:** {“marks”: 90, “name”: “Harish”}

**Note:** A Map can be consist duplicate elements.

#provider Block

provider "azurerm" {

  features {}

  client\_id       = "1f79e427-2ac4-4eb6-9ca0-f4dd4b3f31ee"

  client\_secret   = "WJd8Q~OsrZms4FE1ZdY5QUTP1Or3LzIhrwlyOc3S"

  tenant\_id       = "4a623a04-9917-4ee2-8f59-02586964c992"

  subscription\_id = "51c6d184-6756-4a9a-ade4-cd0f3d57cded"

}

#resource block

resource "azurerm\_resource\_group" "TFRG" {

  name     = var.rg-name["second"]

  location = var.loc-name["primary"]

}

**Fig:** main.tf file

variable "rg-name" {

  type        = map

  default     = {

    "first":"rg"

    "second":"RG1"

    "third":"Rg-01"

    }

  description = "list of resource group names"

}

variable "loc-name" {

  type        = map

  default     = {

    "primary":"eastus"

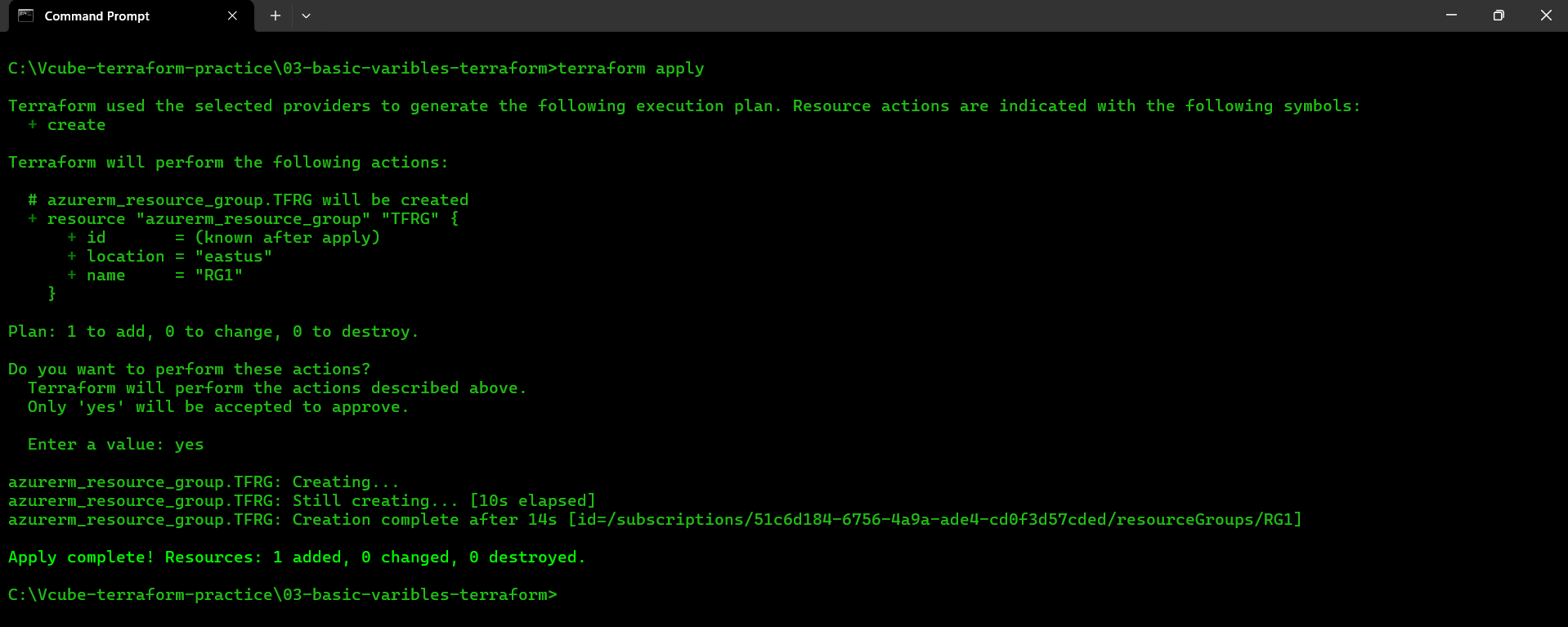
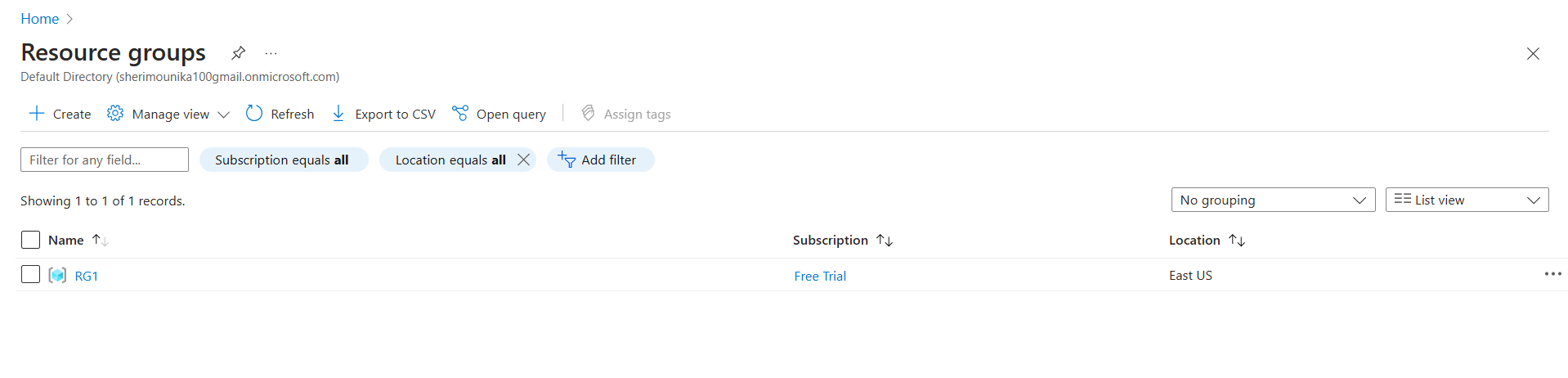
    "secondary":"westus"

  }

  description = "list of location name"

}

**Fig:** variable.tf file



**Fig:** Resource group is created using **“Map”** in variable block.

1. **Set []:**

A set is just similar as **“list”**, but Set cannot consist duplicate elements.

Ex: [“rage.”, “Rug”, “RG”]

1. **count:**

Count normally used with **“list”** data type in the variable.

By using “count” we can **create the multiple resource** with **single block of code**.

The primary use of count is to dynamically create multiple instances of a resource or module based on a number or an expression that evaluates to a number. This is incredibly useful when you need to provision a variable number of similar resources, such as:

* A set of virtual machines
* Multiple subnets
* Several load balancer listeners
* Multiple resource groups etc.

**Example:** Creating of two resource groups with single block of code.

#resource block

resource "azurerm\_resource\_group" "TFRG" {

  count    = 2

  name     = var.rg-name[count.index]

  location = var.loc-name[count.index]

}

**Fig:** main.tf file.

variable "rg-name" {

  type        = list(any)

  default     = ["RG01", "RG02", "RG03"]

  description = "list of resource group names"

}

variable "loc-name" {

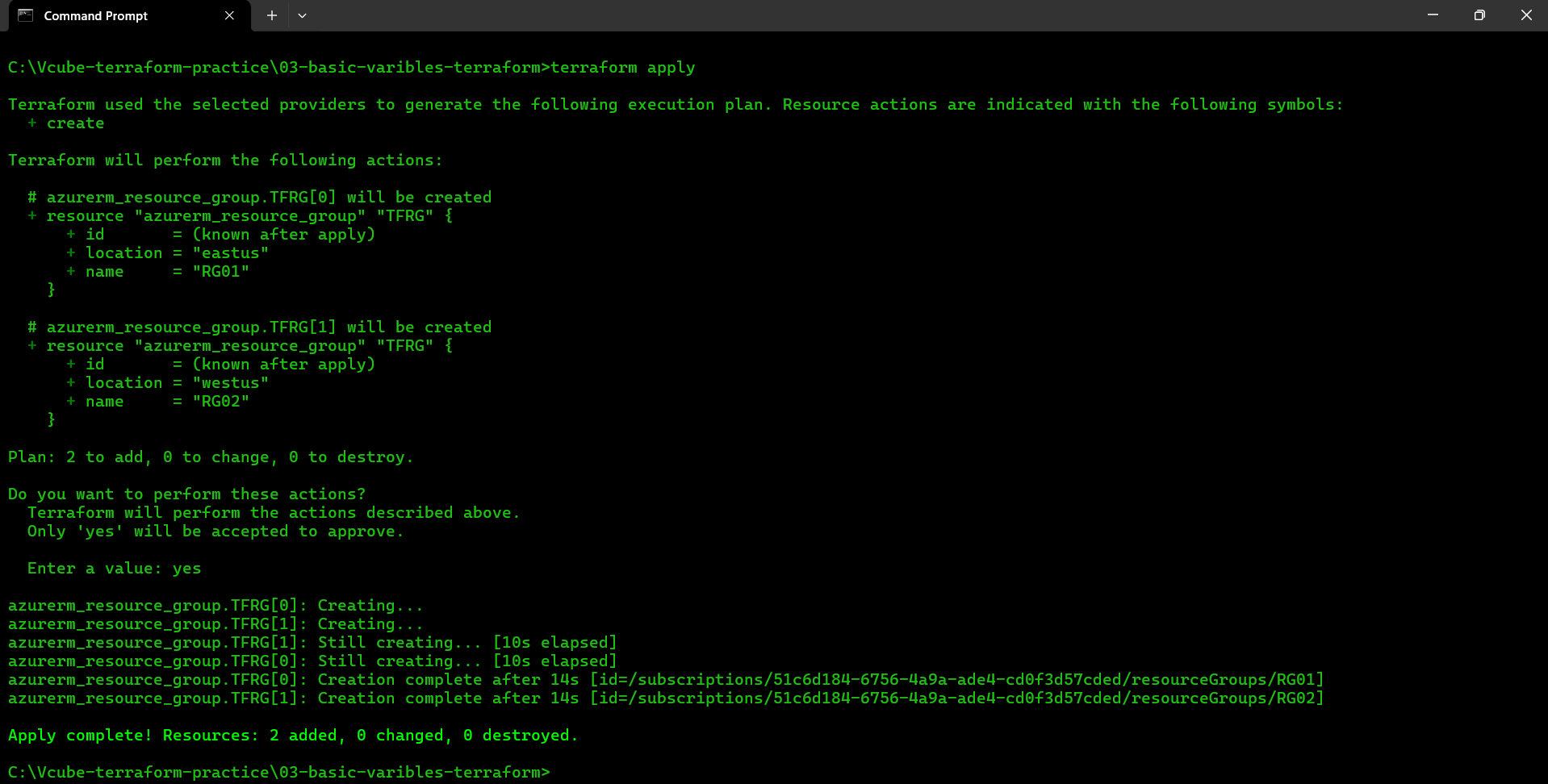
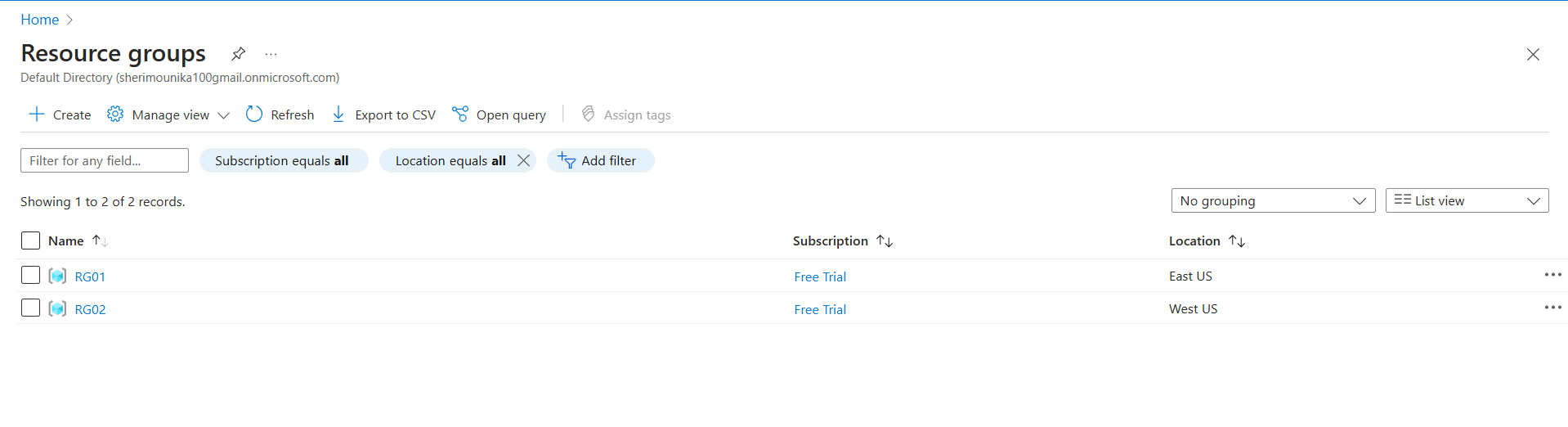
  type        = list(any)

  default     = ["eastus", "westus", "centralindia"]

  description = "list of location name"

}

**Fig:** variable.tf file



**Fig:** Two resources groups (RG01 & RG02) are created with single resource block using count

We can create the total number of resources group which are present in the list by using **"length”** as shown below command.

#provider Block

provider "azurerm" {

  features {}

  client\_id       = "1f79e427-2ac4-4eb6-9ca0-f4dd4b3f31ee"

  client\_secret   = "WJd8Q~OsrZms4FE1ZdY5QUTP1Or3LzIhrwlyOc3S"

  tenant\_id       = "4a623a04-9917-4ee2-8f59-02586964c992"

  subscription\_id = "51c6d184-6756-4a9a-ade4-cd0f3d57cded"

}

#resource block

resource "azurerm\_resource\_group" "TFRG" {

  count    = length(var.rg-name)

  name     = var.rg-name[count.index]

  location = var.loc-name[count.index]

}

**Fig:** main.tf file

variable "rg-name" {

  type        = list

  default     = ["RG01", "RG02", "RG03"]

  description = "list of resource group names"

}

variable "loc-name" {

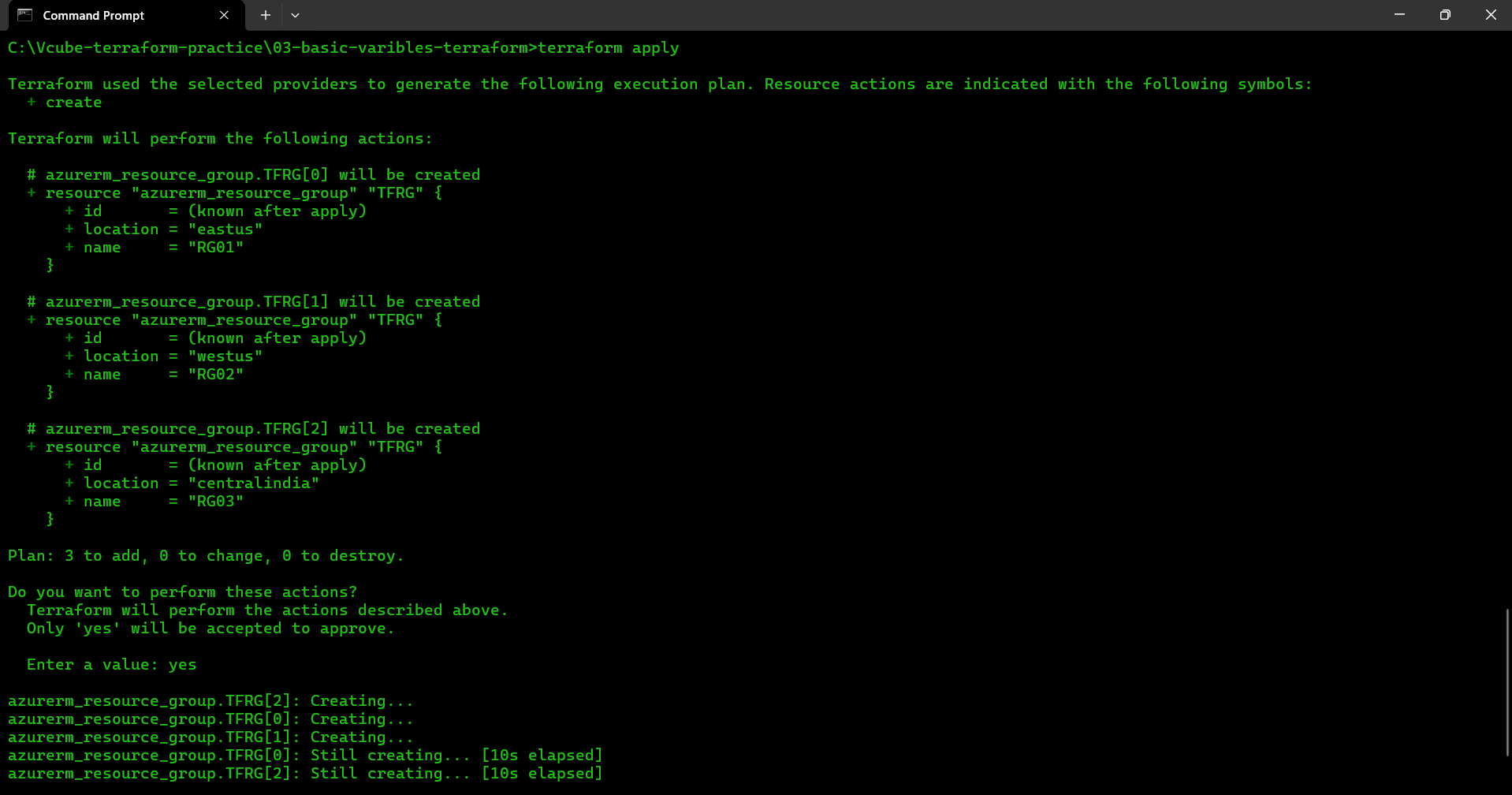
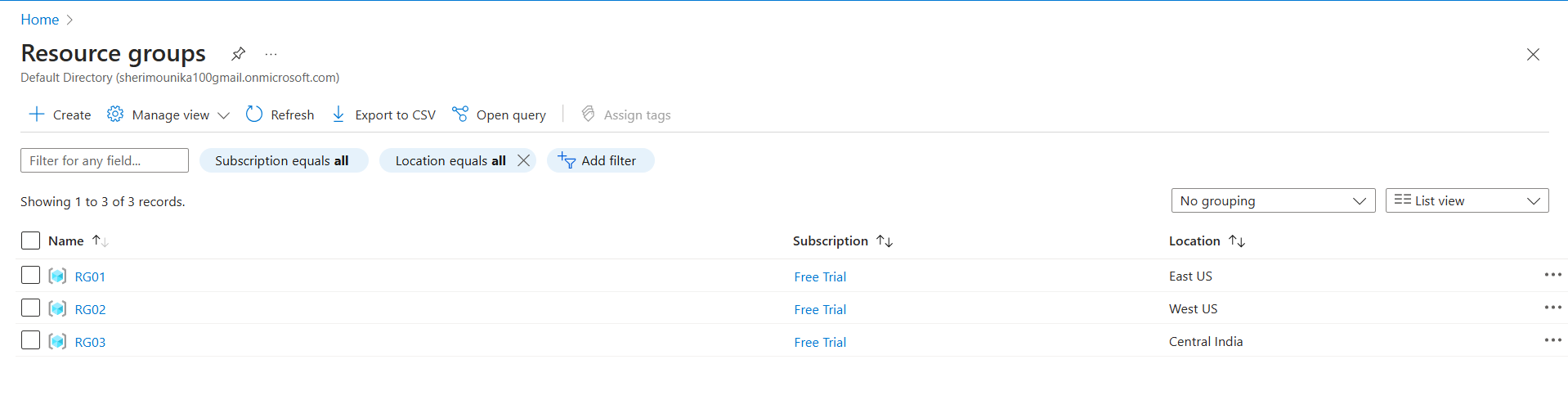
  type        = list

  default     = ["eastus", "westus", "centralindia"]

  description = "list of location name"

}

**Fig:** variable.tf file.



**Fig:** Resource groups are created with all index values of list (0, 1, 2) [RG01, RG02, RG03]