**Terraform Backend Block**

* We know that while running the **terraform apply** command the total information about the infrastructure (resources, their properties, and their relationships) all are stored in the **“terraform.tfstate”** file and which present in local machine.
* The terraform.tfstate file is a crucial component of Terraform, an Infrastructure as Code (IaC) tool.
* It serves as a snapshot of your managed infrastructure, storing the current state of your resources and their configurations.
* In real world this terraform.tfstate file cannot be stored locally, it can be stored remotely.
* That mean it can be stored in the **Backend Block.**
* Storing the terraform.tfstate file in a **backend block** (such as AWS S3, Azure Blob Storage, or Terraform Cloud) is important for **collaboration, security, and consistency**.
* **State Storage:** Terraform needs to keep track of the infrastructure it manages. This information, including the resources, their properties, and their relationships, is stored in a file called the "state file." The backend block tells Terraform where to store this state file.
* **Configuration:** The backend block is defined within your Terraform configuration files (usually in a terraform block) and specifies the type of backend you want to use along with any necessary configuration parameters.

**Why Backend Block is Important?**

**Collaboration:** In team environments, multiple people need to work with the same infrastructure. Storing the state file (terraform.tfstate) in a central location (using a remote backend cloud storage) ensures everyone is working with the same, up-to-date information.

**State Locking:** Some Backend provide a mechanism called "state locking." This prevents multiple Terraform operations from running simultaneously and potentially corrupting the state file.

**Security:** Remote Backend often provide better security for your state file, with features like encryption and access control.

**Versioning:** Some Backend offer versioning of the state file, allowing you to track changes and revert to previous states if needed.

In order to define the Backend Block in the terraform configuration file, first we have to create the storage account with container.

**Step1:** creation of resource group, storage account with container.

#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 group block

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

  name     = RG-01

  location = "eastus"

}

#storage account creation

resource "azurerm\_storage\_account" "TFSTAC" {

  name                     = "tfstate123"

  resource\_group\_name      = azurerm\_resource\_group.TFRG.name

  location                 = azurerm\_resource\_group.TFRG.location

  account\_tier             = "Standard"

  account\_replication\_type = "LRS"

}

#container creation

resource "azurerm\_storage\_container" "example" {

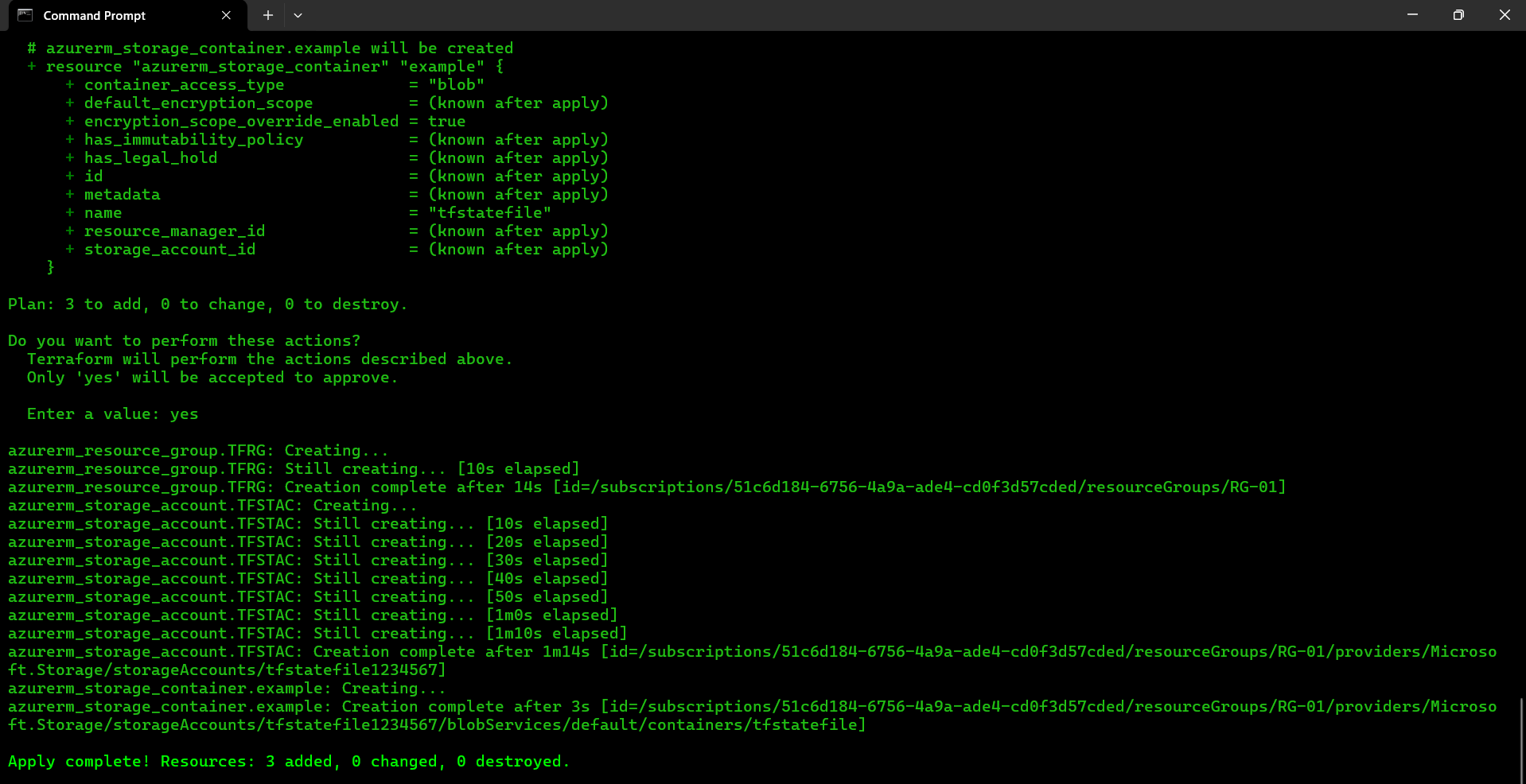
  name                  = "tfstatefile"

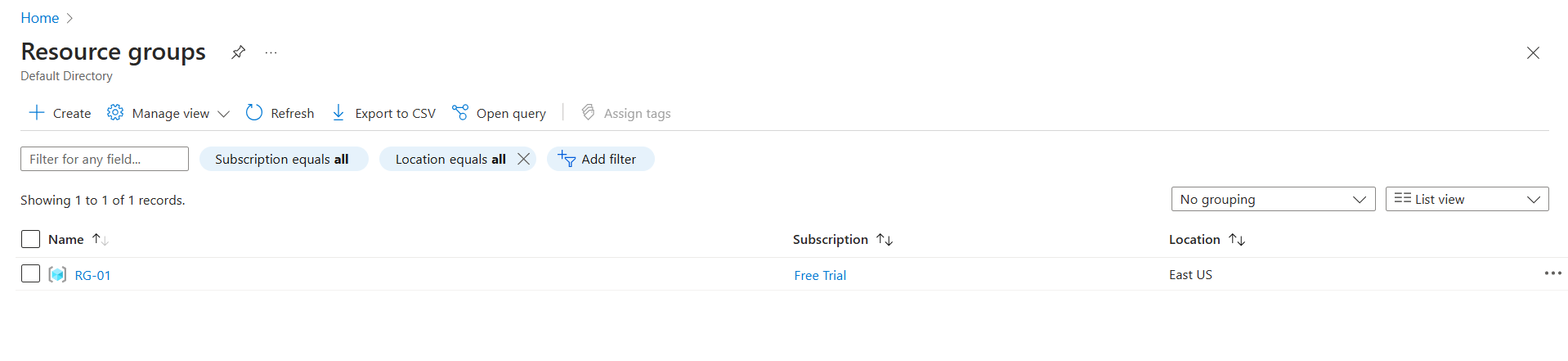
  storage\_account\_id    = azurerm\_storage\_account.TFSTAC.id

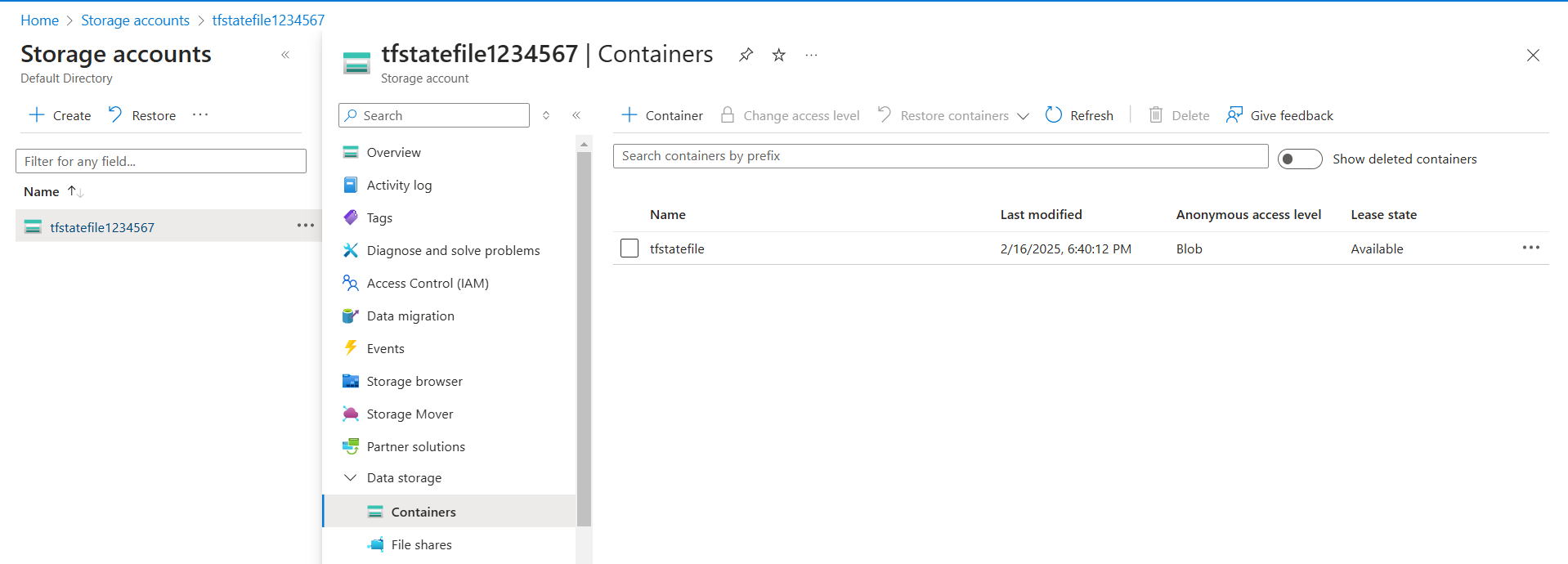
  container\_access\_type = "blob"

}

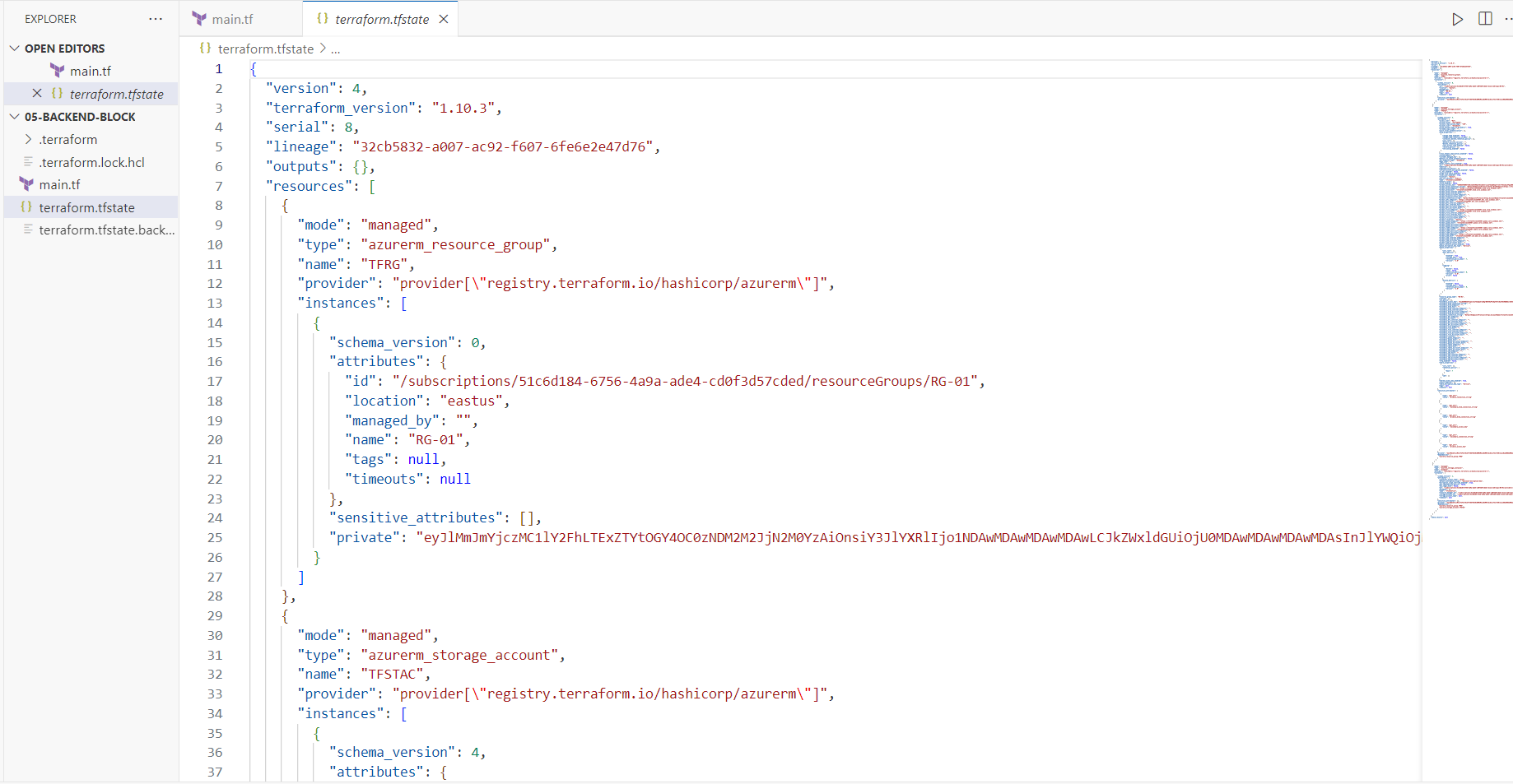
**Fig:** Main.tf file



**Fig:** Resource group is created



**Fig:** Storage account with container is created.

****

**Fig:** terraform.tfstate file before creation of Backend Block.

**Step2:** Creation of Backend Block to store the **terraform.tfstste** file in the storage account of container.

terraform {

  backend "azurerm" {

    resource\_group\_name  = "RG-01"

    storage\_account\_name = "tfstatefile1234567"

    container\_name       = "tfstatefile"

    key                  = "prod.terraform.tfstate"

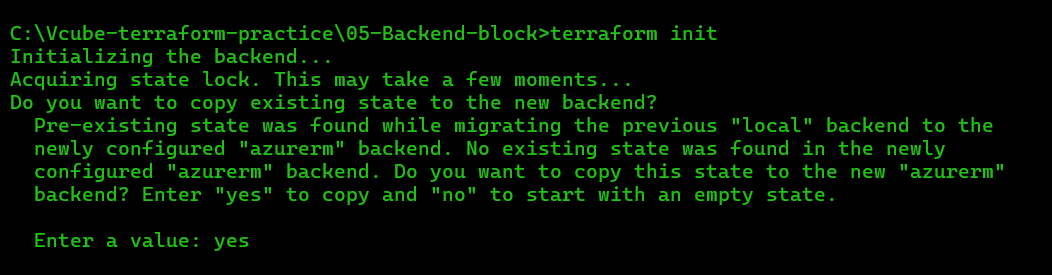
    access\_key           ="X2RUA5VVQR5fxNbcCm2ECH6q72Enow01CriaLm7ZCq86epnfln3v+VSCuAai4FpwwNO5IcDIQC9Z+AStvx7SBg=="

  }

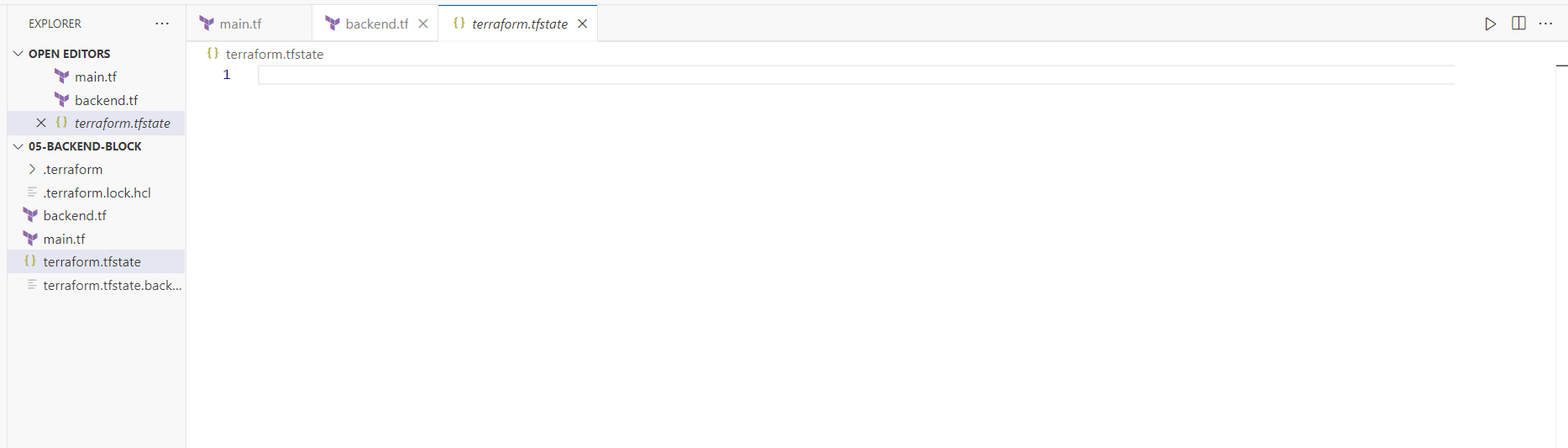
}

**Fig:** Backend Block.

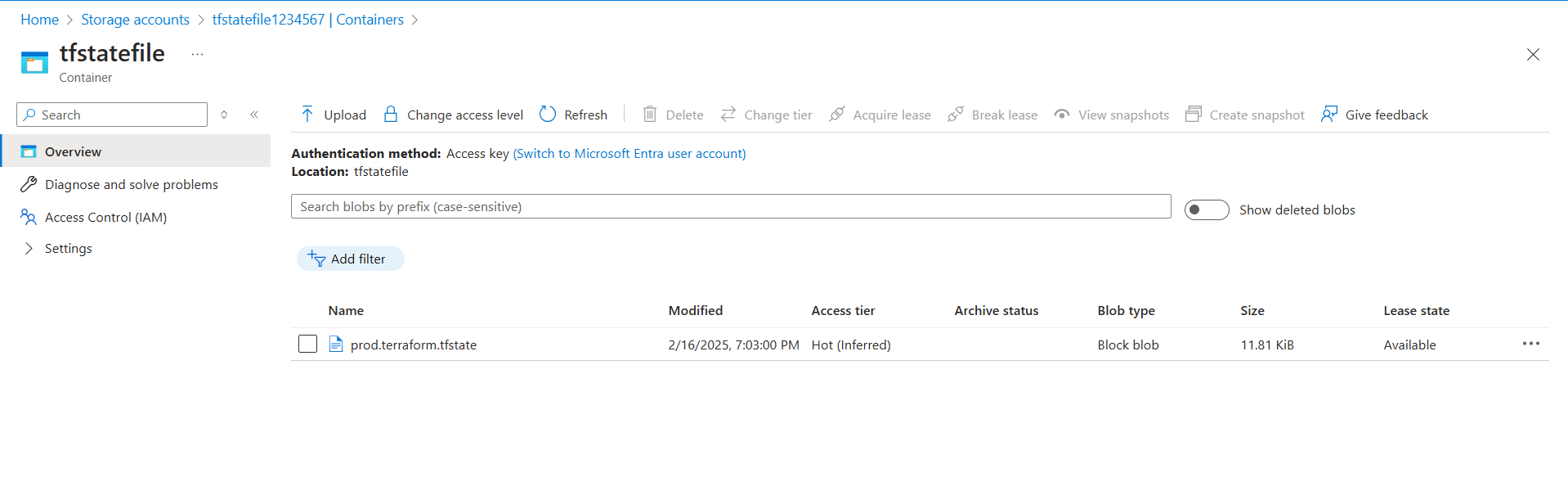
When we do **terraform init** command then it will ask whether to copy the state file into the storage account (remote backend) or not as show below figure.



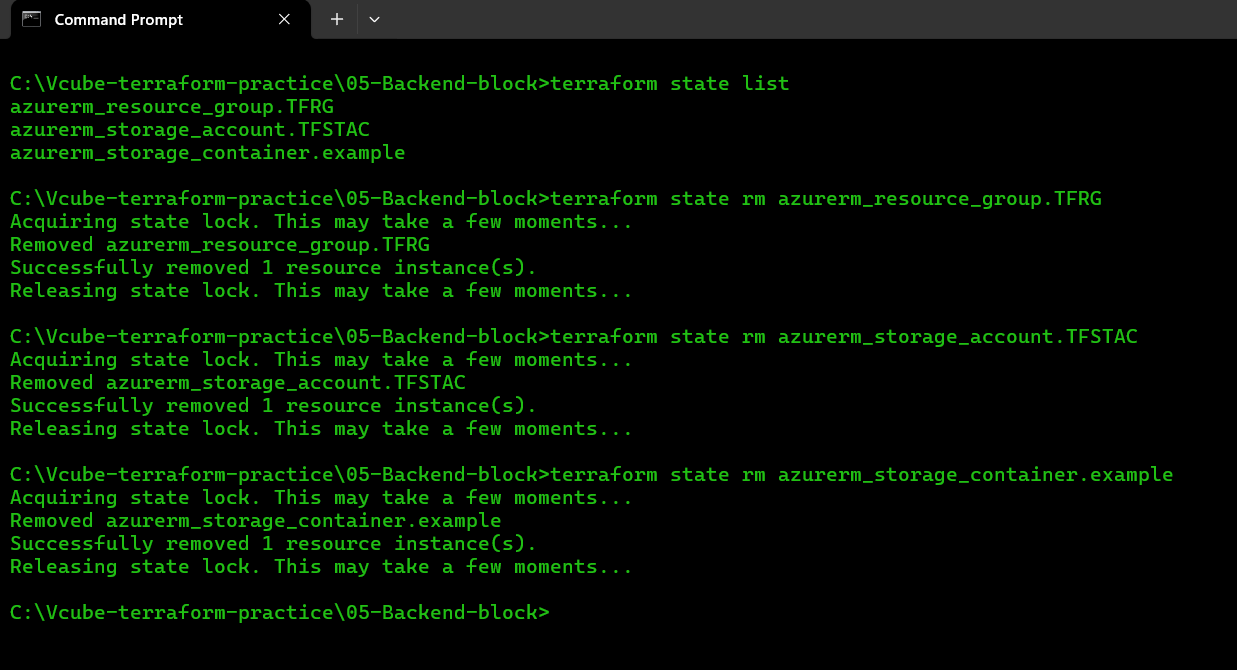
If we enter **“yes”** then it will migrate the state file into the azurerm backend (azure blob) by empting the local state file (local .tfstate file become empty) as show below figure.



Then the terraform.tfstate file migrates to the remote location (azure blob). Let’s check whether it is migrated or not.



**Note1:** After uploading the terraform.tfstate file in to the storage account, if we do **terraform destroy** command then it will delete all the resource including backend block, as a result we will **lost the terraform.tfstste file**. In order to prevent loosing of terraform.tfstate file we have to remove the backend block related information (resource group, storage account, container in which backed block is deployed) from terraform.tfstate file by using the command “**terraform state rm <resource>’**



**Note2:** “**terraform state rm <backend block related resource**>” it can only delete the metadata of resource (backend block) from terraform.tfstate file. So that while performing the terraform destroy command it won’t delete the backend block related resource from cloud, because the backend block related resource information not found in the terraform.tfstate file.

**Note2:** In general **terraform destroy** command it only delete resource whose information is present in the terraform.tfstate file.

The above performed task is just uploading or moving the local terraform.tfstate file into the remote location (AWS S3, azure Blob). So that in teem or collaborate environment (project group people) multiple people can read the file remotely, and based on the terraform.tfstate file they can update the changes and work on it easily.