Aniket Kumar Sinha for This is Learning
Posted on Jun 11 • Updated on Jun 13

Deploy Azure Infrastructure using Terraform Cloud

#azure #terraform #devops #github

Terraform Cloud (2 Part Series)

1 **Deploy Azure Infrastructure using Terraform Cloud**

2 Implementing Policy-as-Code to Terraform workflow usin...

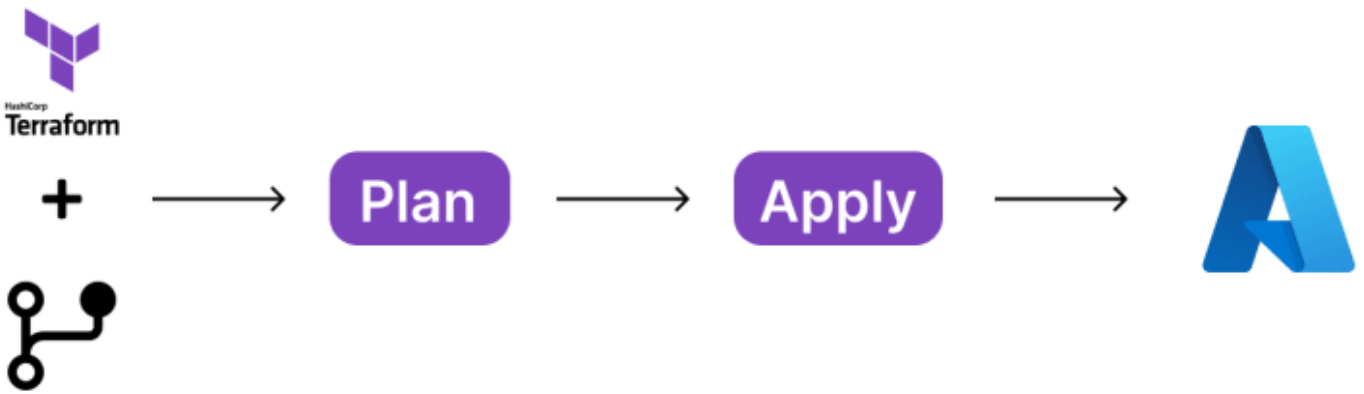
Note: You can get the whole code from this repository: [aniketkumarsinha/azure-terraform-infrastructure]

What is Terraform?

Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.

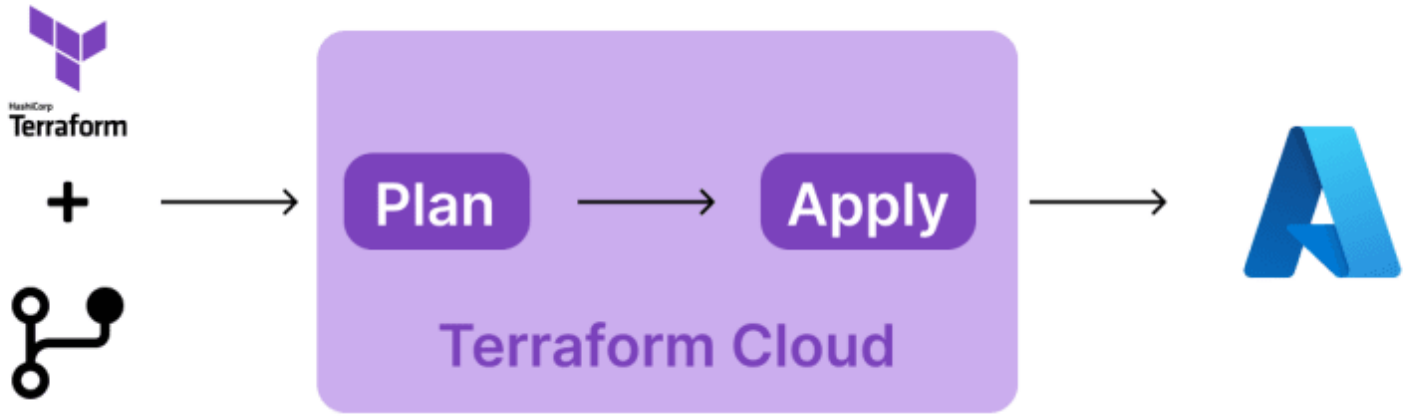
Terraform Flow

- First you have the terraform code.
- Then we have Terraform Plan phase. The terraform plan command creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure.
- Terraform Apply phase executes the actions proposed in a Terraform plan.
- And everything gets deployed over the CSP, here Azure.



What is Terraform Cloud?

Terraform Cloud is a managed service offering by HashiCorp that eliminates the need for unnecessary tooling and documentation for practitioners, teams, and organizations to use Terraform in production. It allows you to provision infrastructure in a remote environment that is optimized for the Terraform workflow.



In this blog, we will be creating Azure Infrastructure using Terraform and will be deploying it over to Azure using Terraform Cloud.

Infrastructure Code

We are creating a VM and resources related to it.

```
# Configure Azure Provider
terraform {
  required_providers {
     azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 3.59.0"
    }
  }
  required_version = ">= 0.14.9"
}

provider "azurerm" {
  features {}

  skip_provider_registration = "true"
}

variable "prefix" {
  default = "terraform"
}

resource "azurerm_resource_group" "rg" {
  name     = "${var.prefix}-ResourceGroup"
  location = "Central India"
}

resource "azurerm_virtual_network" "vnet" {
  name                = "${var.prefix}-VNet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.rg.location
```

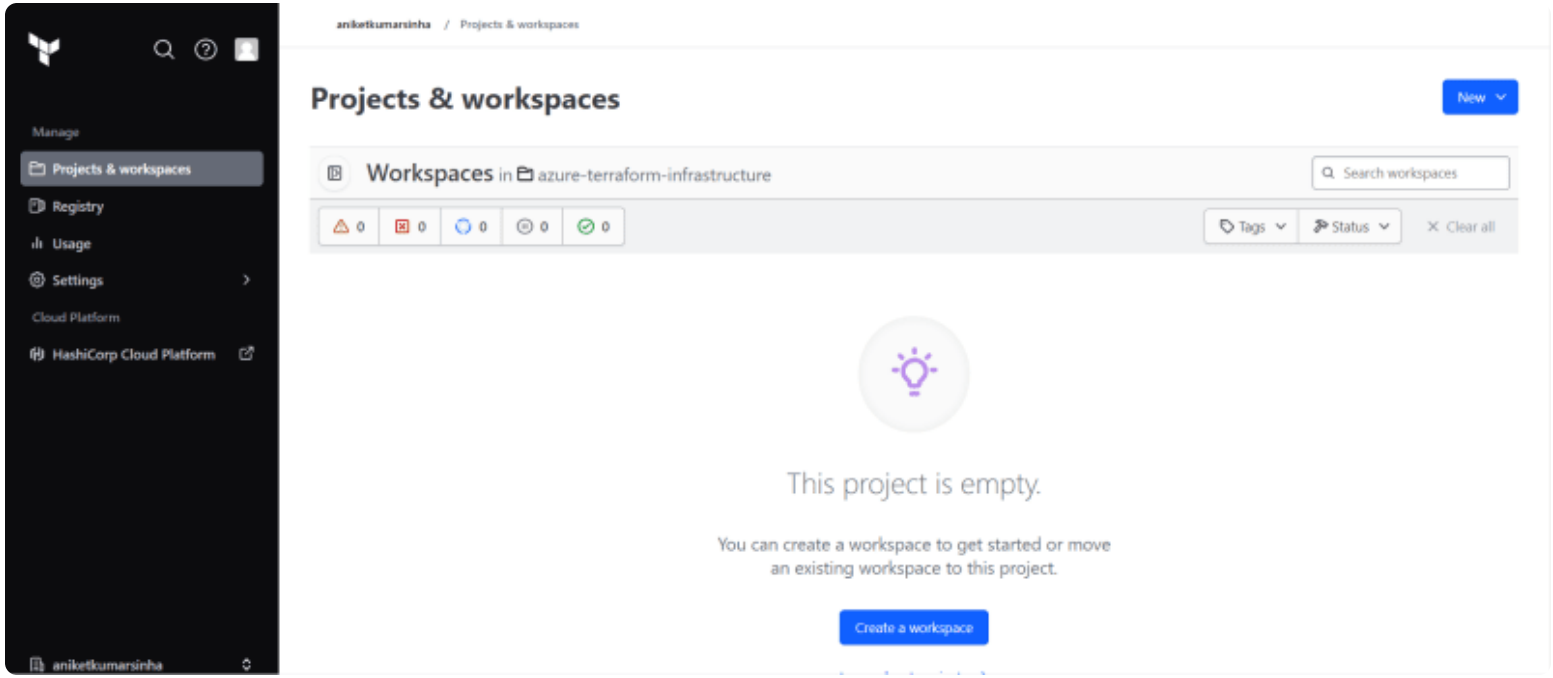
```
    resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_subnet" "internal" {
    name                 = "${var.prefix}-internal"
    resource_group_name  = azurerm_resource_group.rg.name
    virtual_network_name = azurerm_virtual_network.vnet.name
    address_prefixes     = ["10.0.2.0/24"]
}

resource "azurerm_network_interface" "nic" {
    name                = "${var.prefix}-NIC"
    location            = azurerm_resource_group.rg.location
    resource_group_name = azurerm_resource_group.rg.name

    ip_configuration {
      name                          = "tfconfiguration1"
      subnet_id                     = azurerm_subnet.internal.id
      private_ip_address_allocation = "Dynamic"
    }
}

resource "azurerm_virtual_machine" "vm" {
    name                  = "${var.prefix}-vm"
    location              = azurerm_resource_group.rg.location
    resource_group_name   = azurerm_resource_group.rg.name
    network_interface_ids = [azurerm_network_interface.nic.id]
    vm_size               = "Standard_DS1_v2"

    storage_image_reference {
      publisher = "Canonical"
      offer     = "UbuntuServer"
      sku       = "16.04-LTS"
```

```
64        version   = "latest"
65      }
66      storage_os_disk {
67        name              = "myosdisk1"
68        caching           = "ReadWrite"
69        create_option     = "FromImage"
70        managed_disk_type = "Standard_LRS"
71      }
72      os_profile {
73        computer_name  = "hostname"
74        admin_username = "tfadmin"
75        admin_password = "Password1234!"
76      }
77      os_profile_linux_config {
78        disable_password_authentication = false
79      }
80      tags = {
81        environment = "staging"
82      }
83    }
```

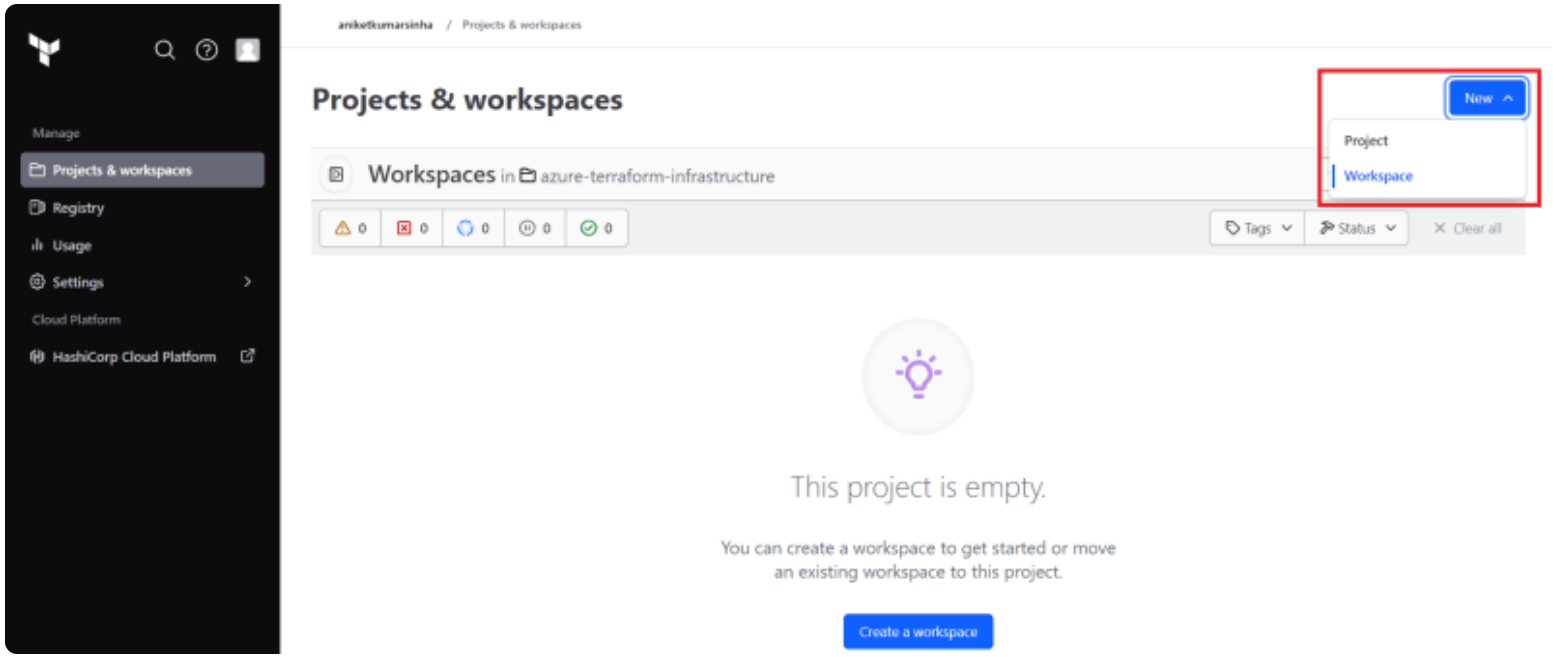
main.tf hosted with ❤ by GitHub view raw

Setting up Terraform Cloud

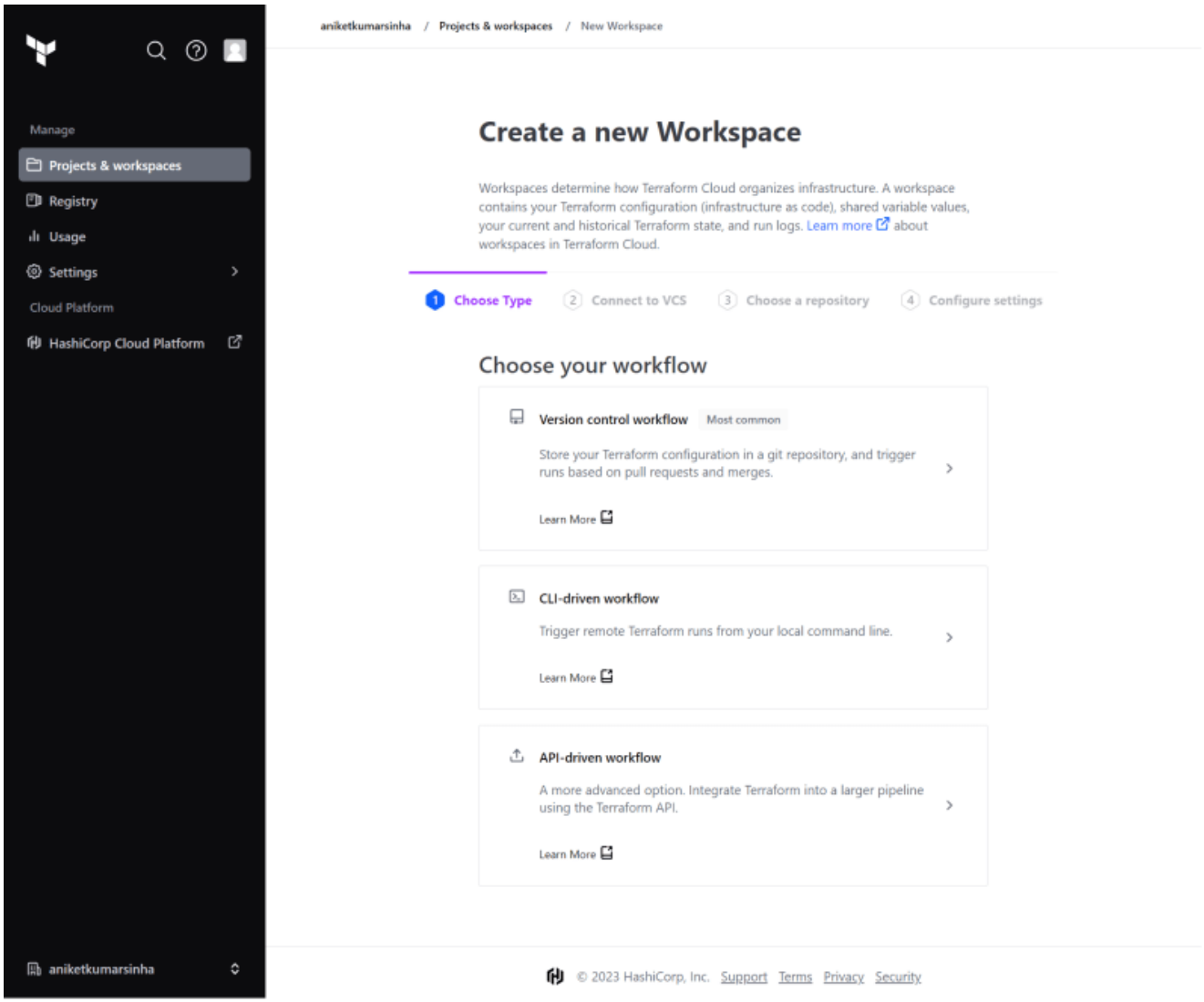
1. Create Terraform Cloud Account - Terraform Cloud
2. Create a Project in Terraform Cloud:



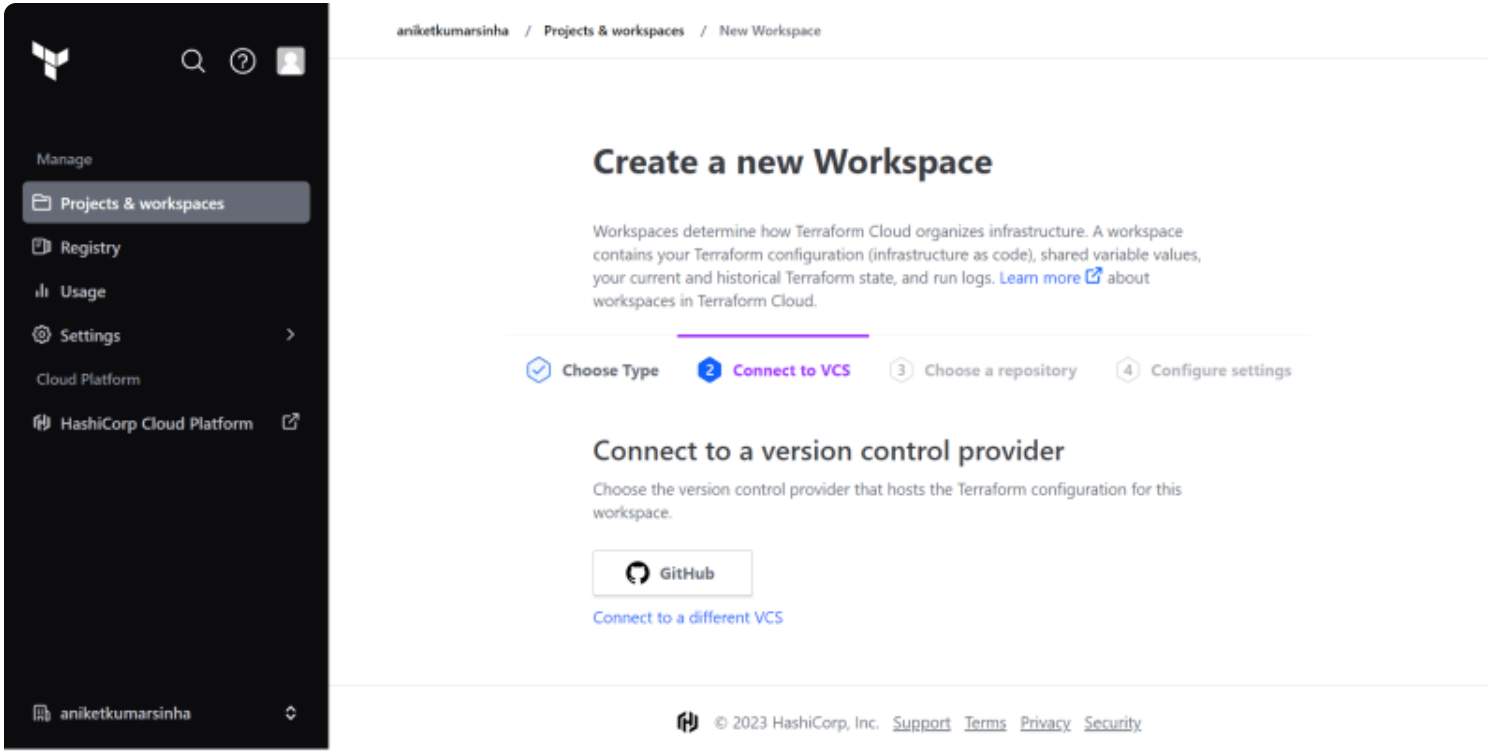
3. Create a Workspace.



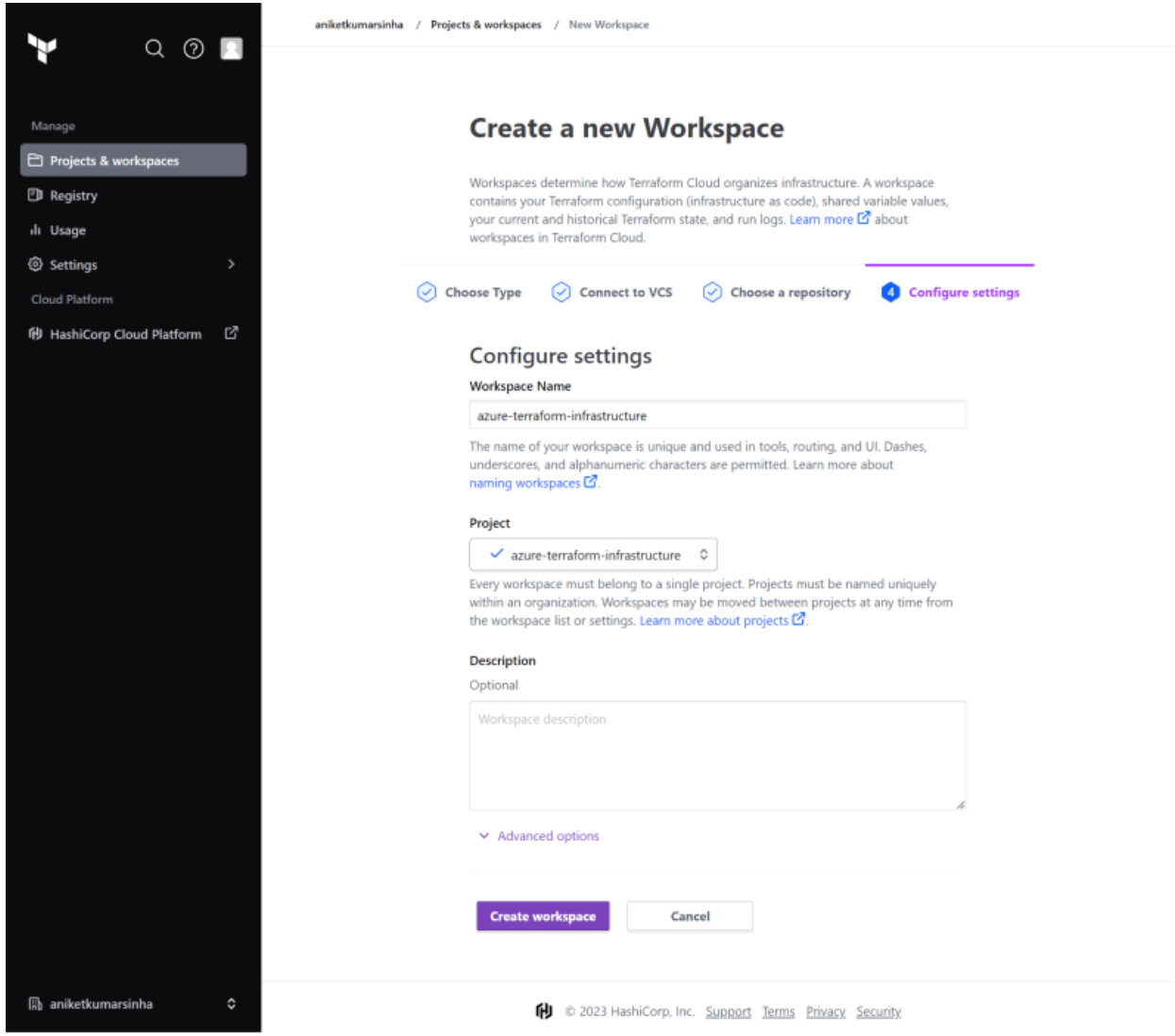
- Choose Version control workflow



- Connect to a version control provider

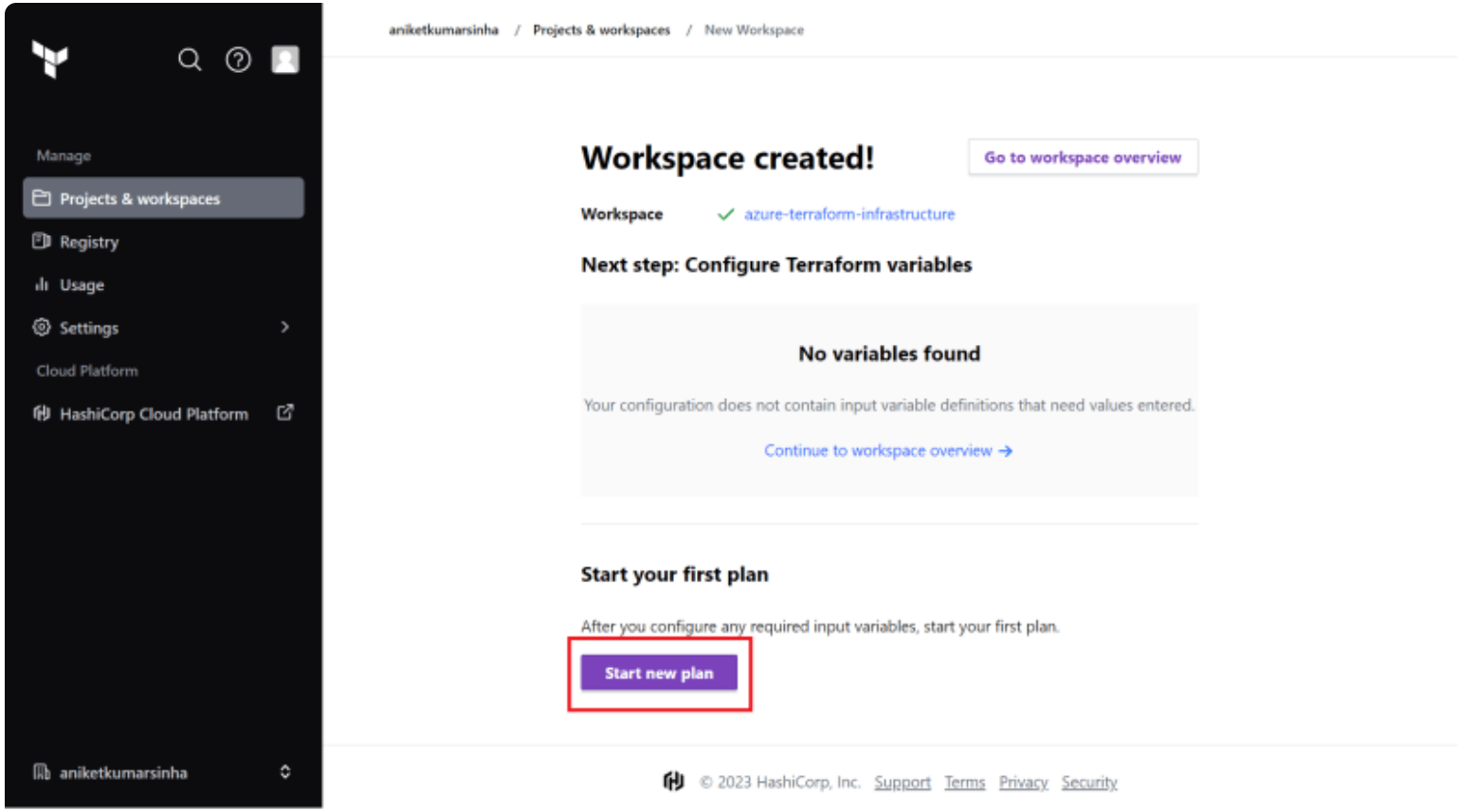


- Choose your Azure Infrastructure repository from your repository list. You can check the advance settings if you want to configure the workflow.

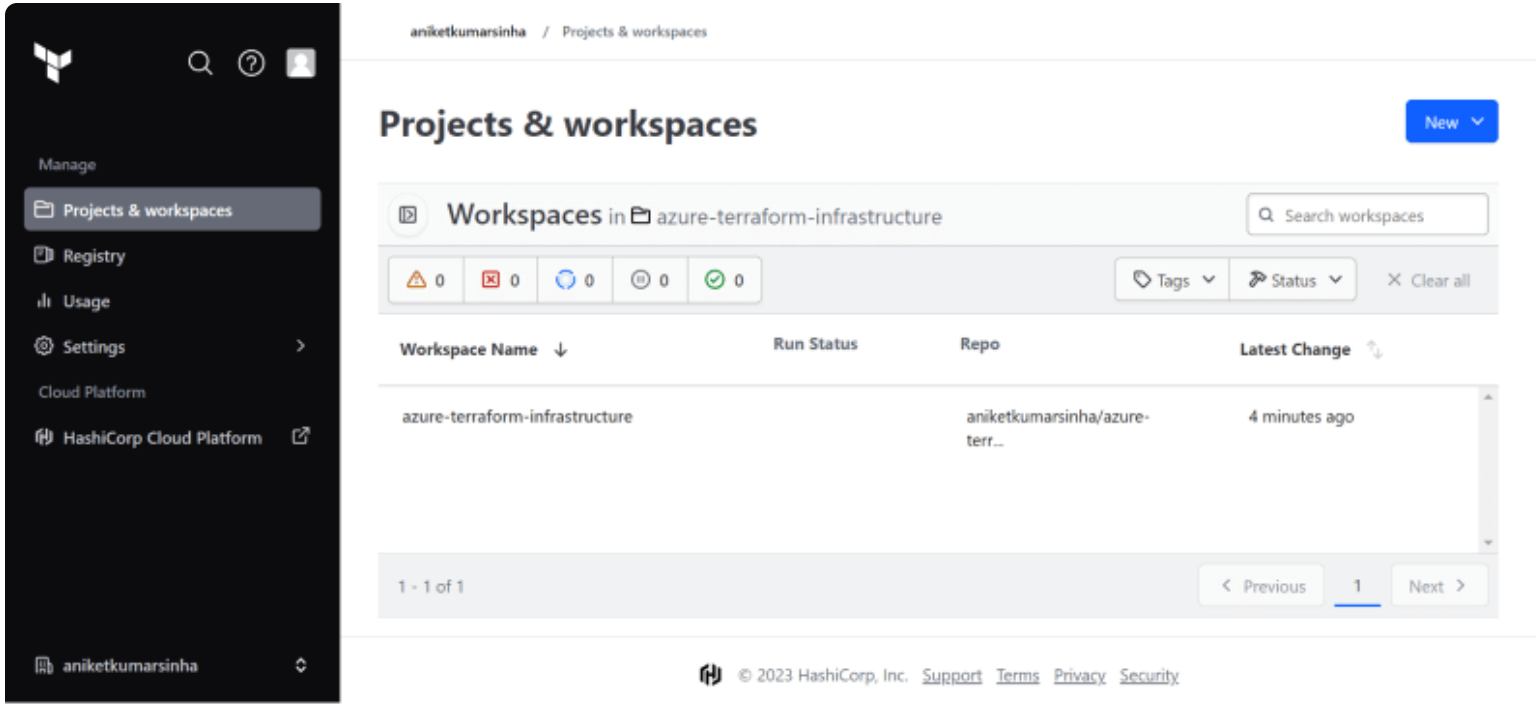


- Tap on **Create Workspace** button to create your workspace in terraform cloud Or Start new run from workspace overview page.

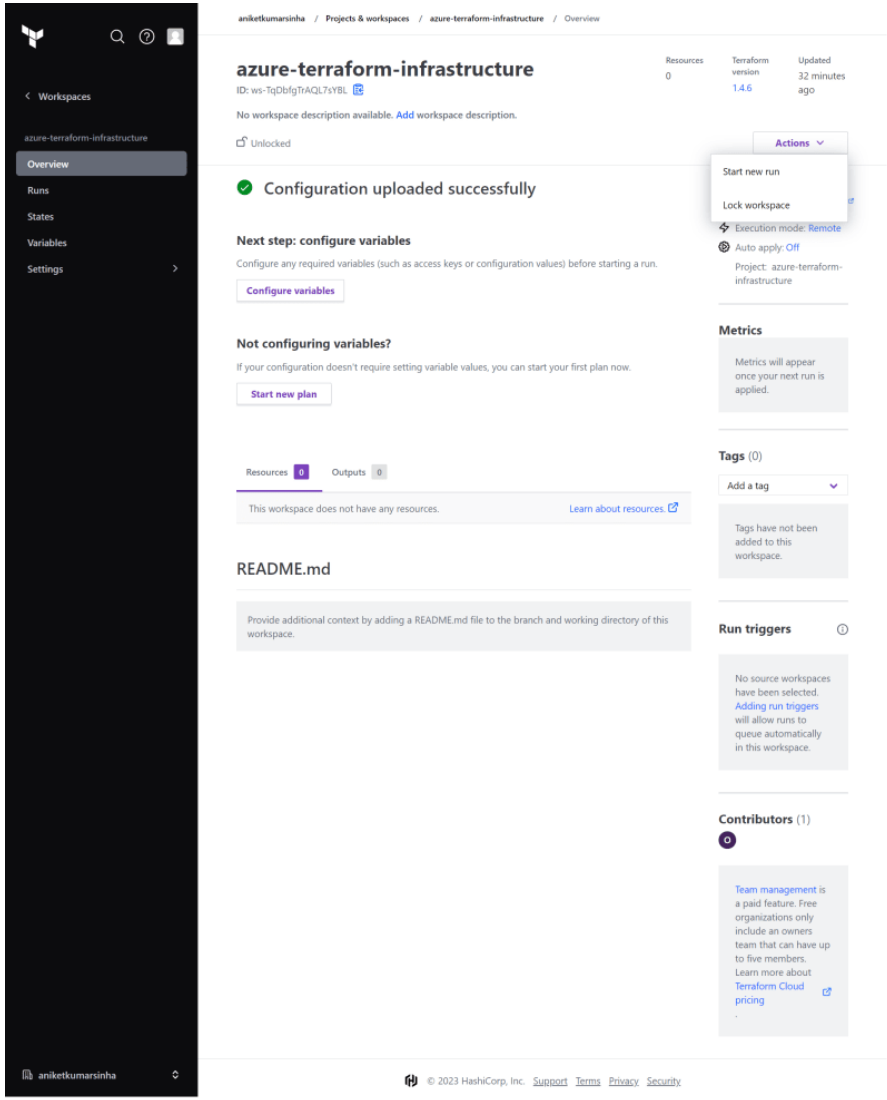
4. You can directly start your plan phase.



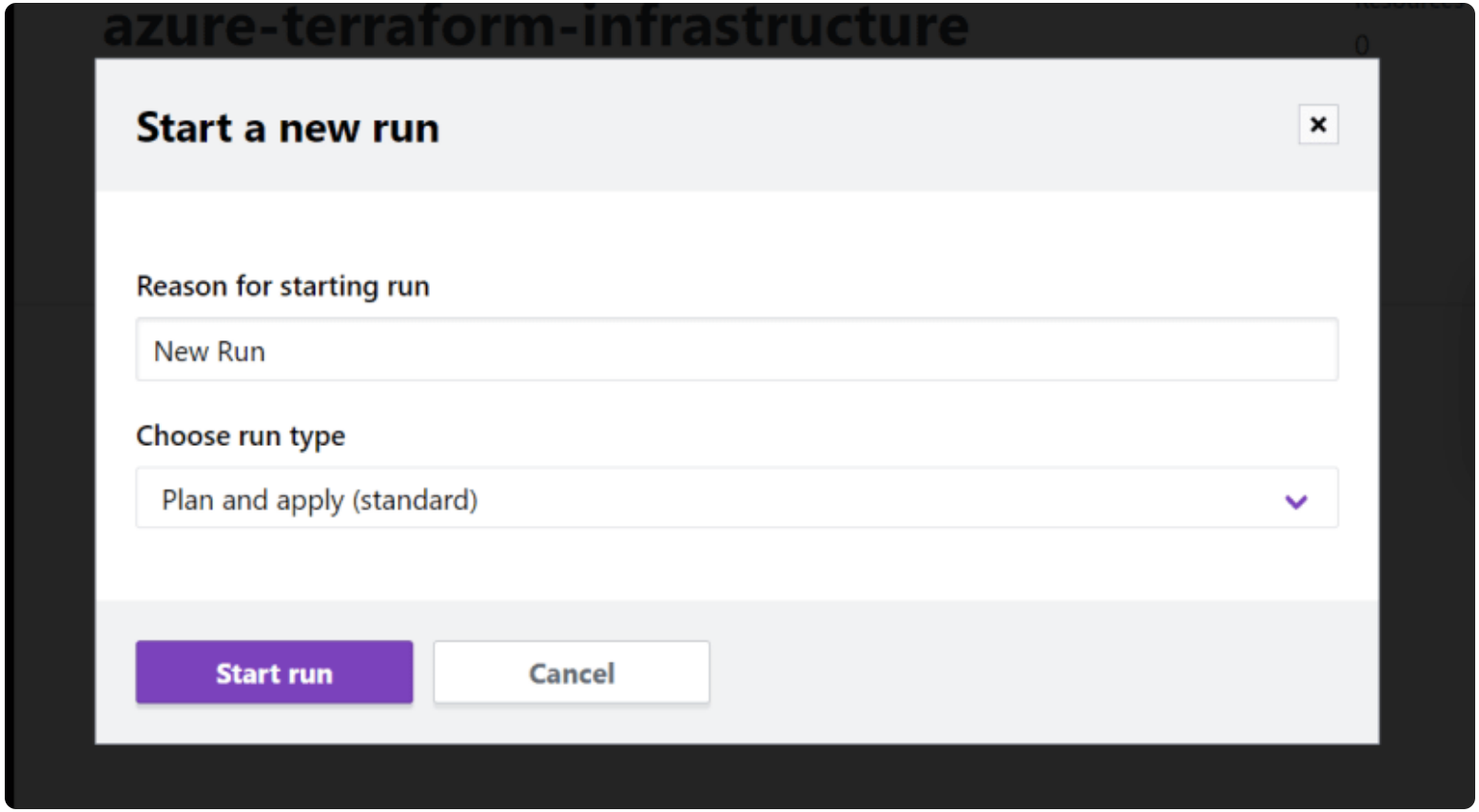
5. You can verify in your projects page that your workspace has been created.



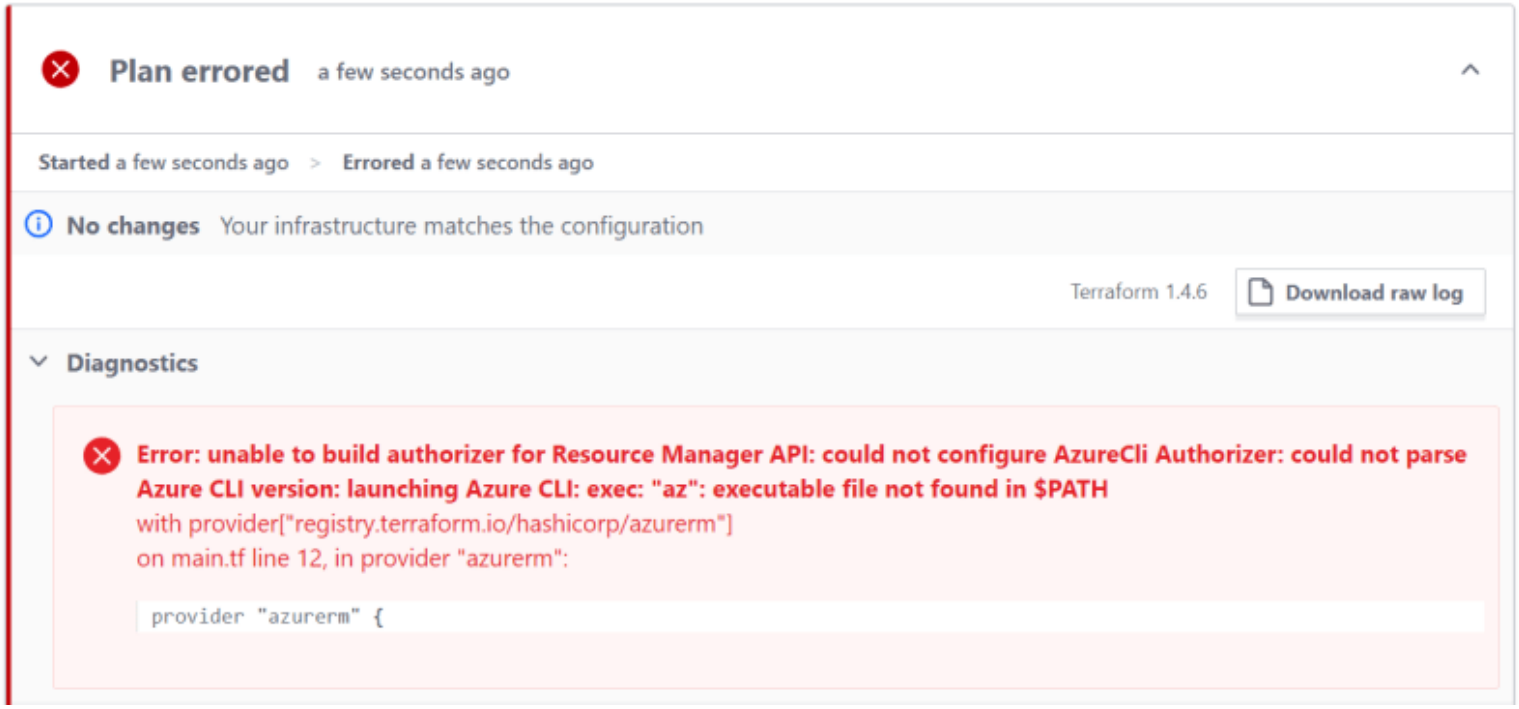
6. Open your workspace to start new run.



7. Choose your run type and start run.

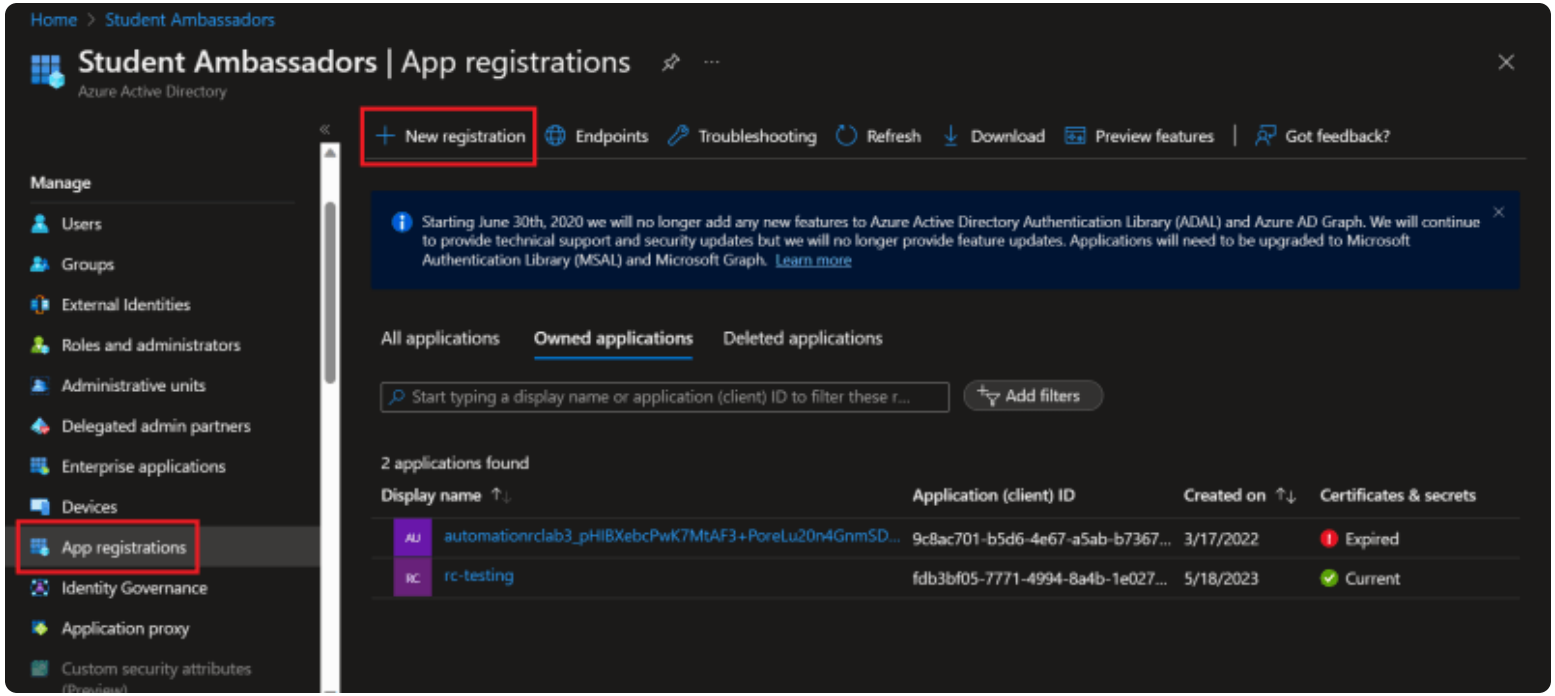


8. Ouch!! Errors!! We are getting error in provider phase. The reason behind this is that we have authorized our infrastructure to connect and write over our Azure. We will be creating an App provide all the necessary details to our infrastructure.

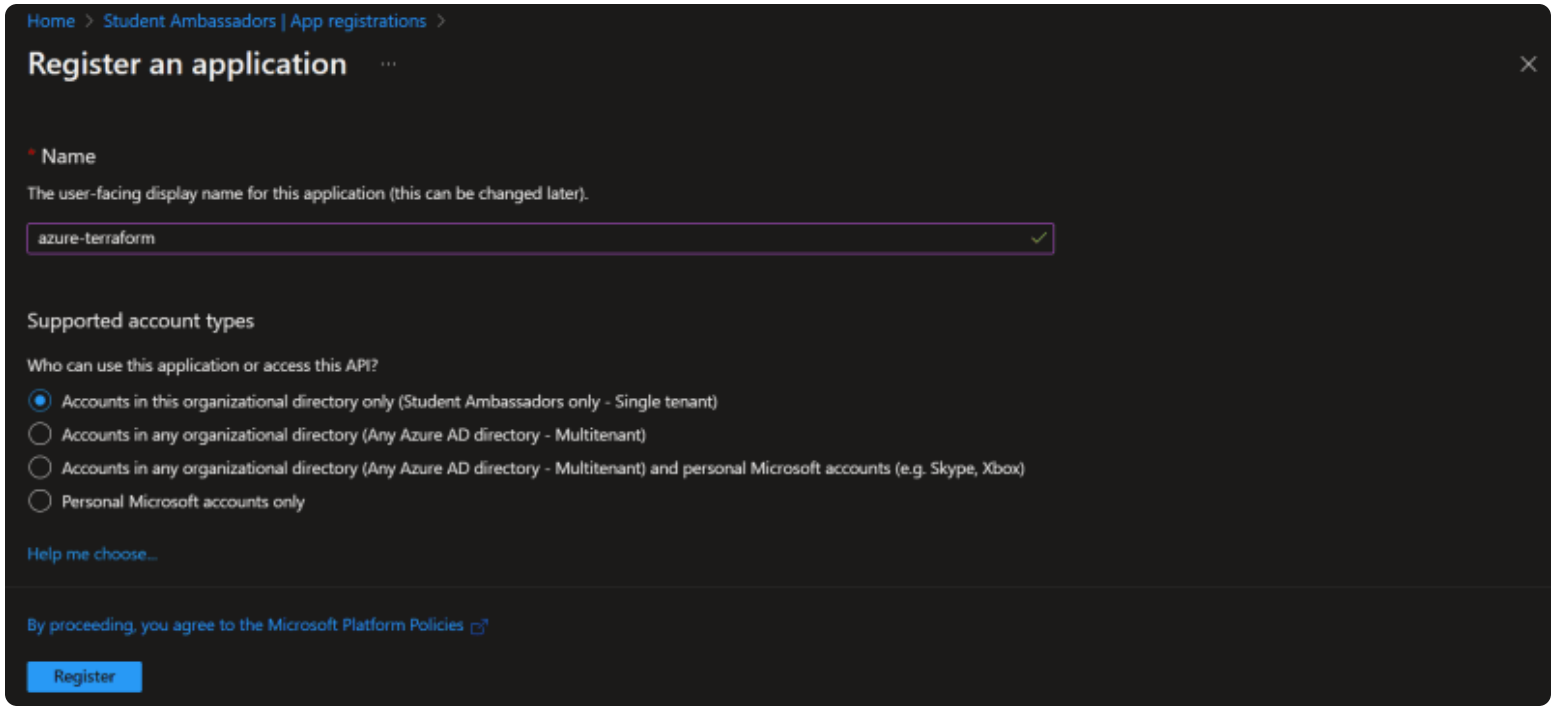


Authorizing Terraform Infrastructure to write over Azure

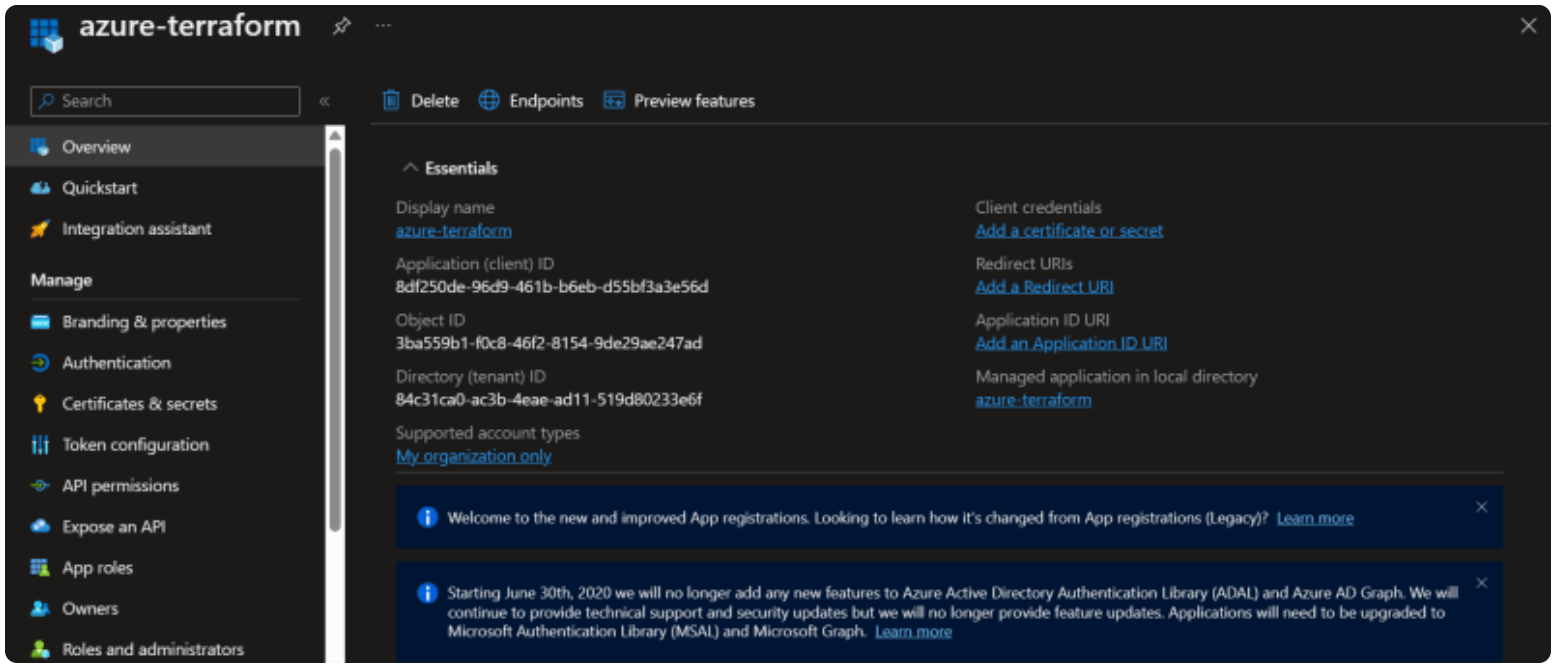
- We are creating an App over Azure to authorize using Client Id, Client Secret and Tenant Id. So in your Azure Portal move into Azure Active Directory, and open `App registrations` from left pane, And add a new registration.



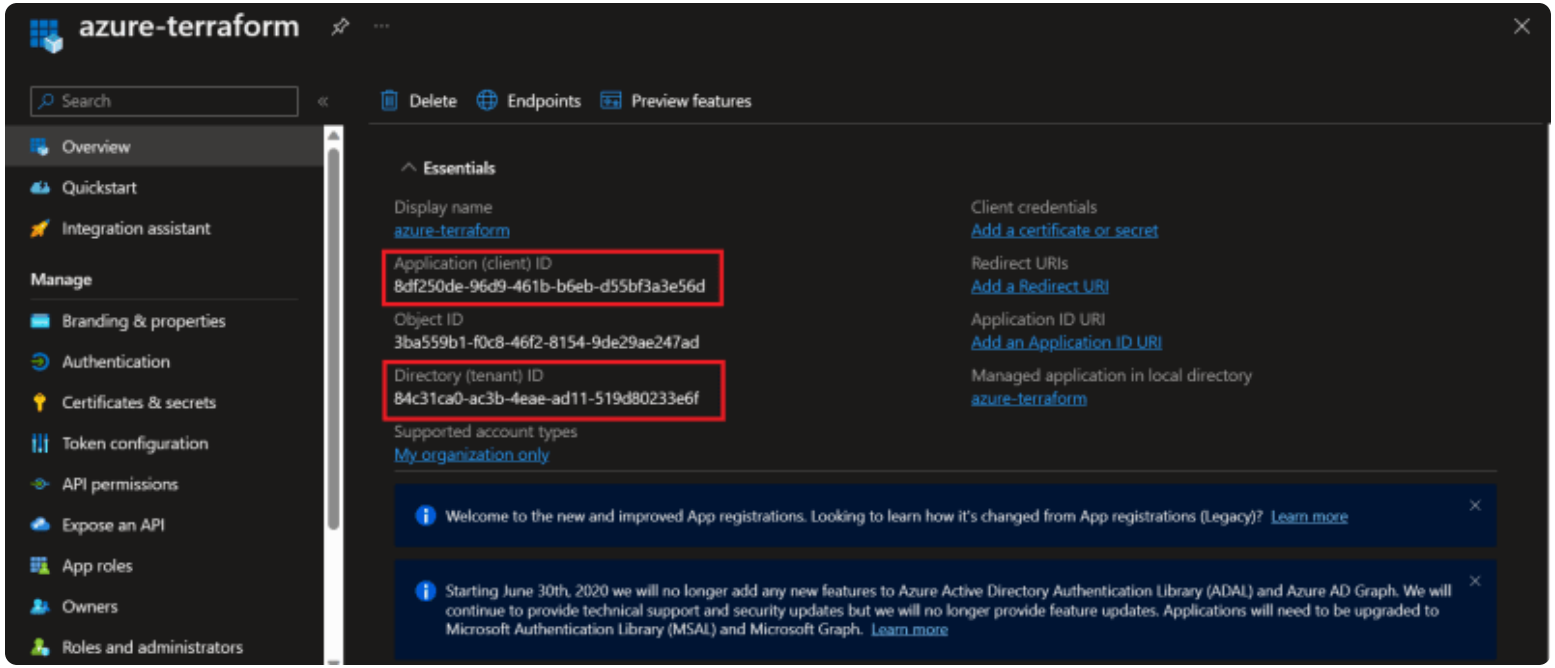
- Add your App details and register.



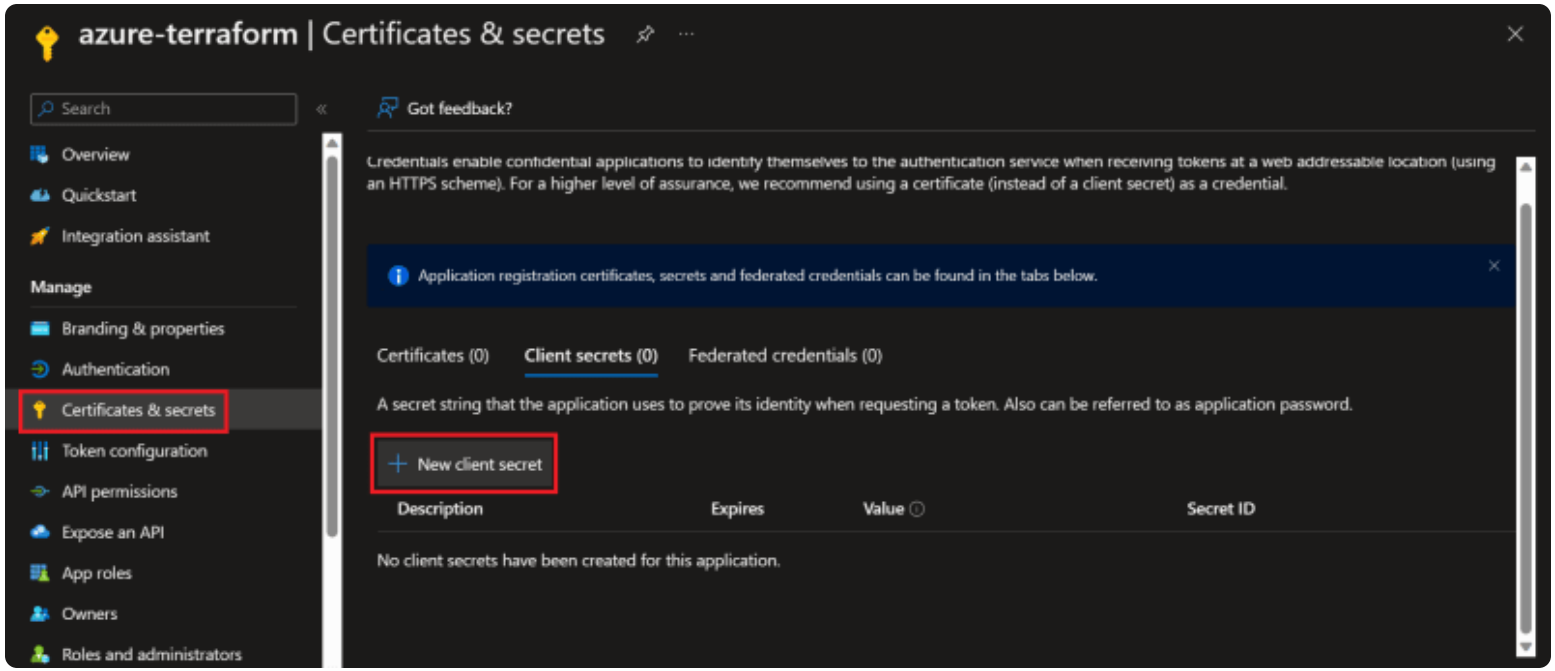
- So we have finally created our app.



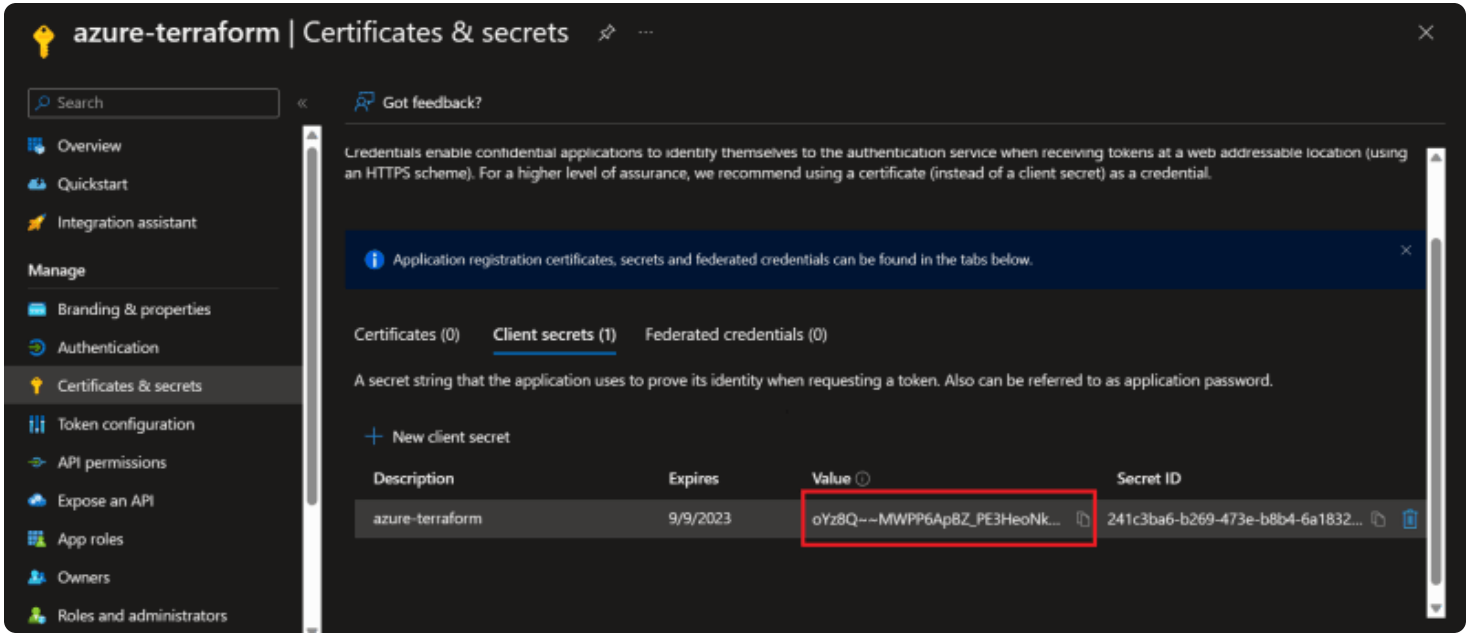
- To connect our Infrastructure we need 4 details
  - Client Id
  - Client Secret
  - Tenant Id
  - Subscription Id We would be getting this one by one.
- `Application (client) ID` is Client Id. `Directory (tenant) ID` is Tenant Id.



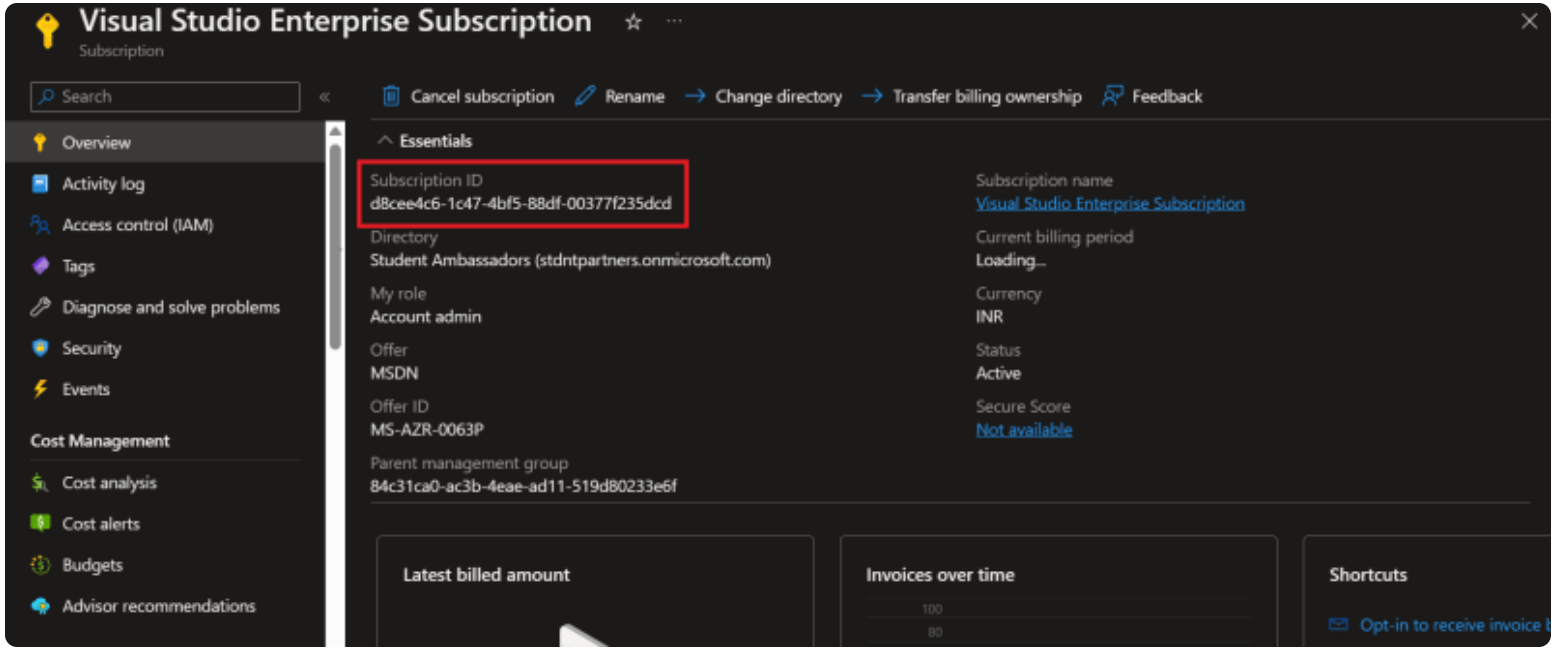
- To get the Client Secret, we first need to create the secret.



- Add description and expiry of this secret which you're creating.
- Copy this value under Value column, and save it somewhere as we wouldn't be able to access this later. This is our Client secret.



- Search Subscription from search box and open your subscription. Copy your subscription id.



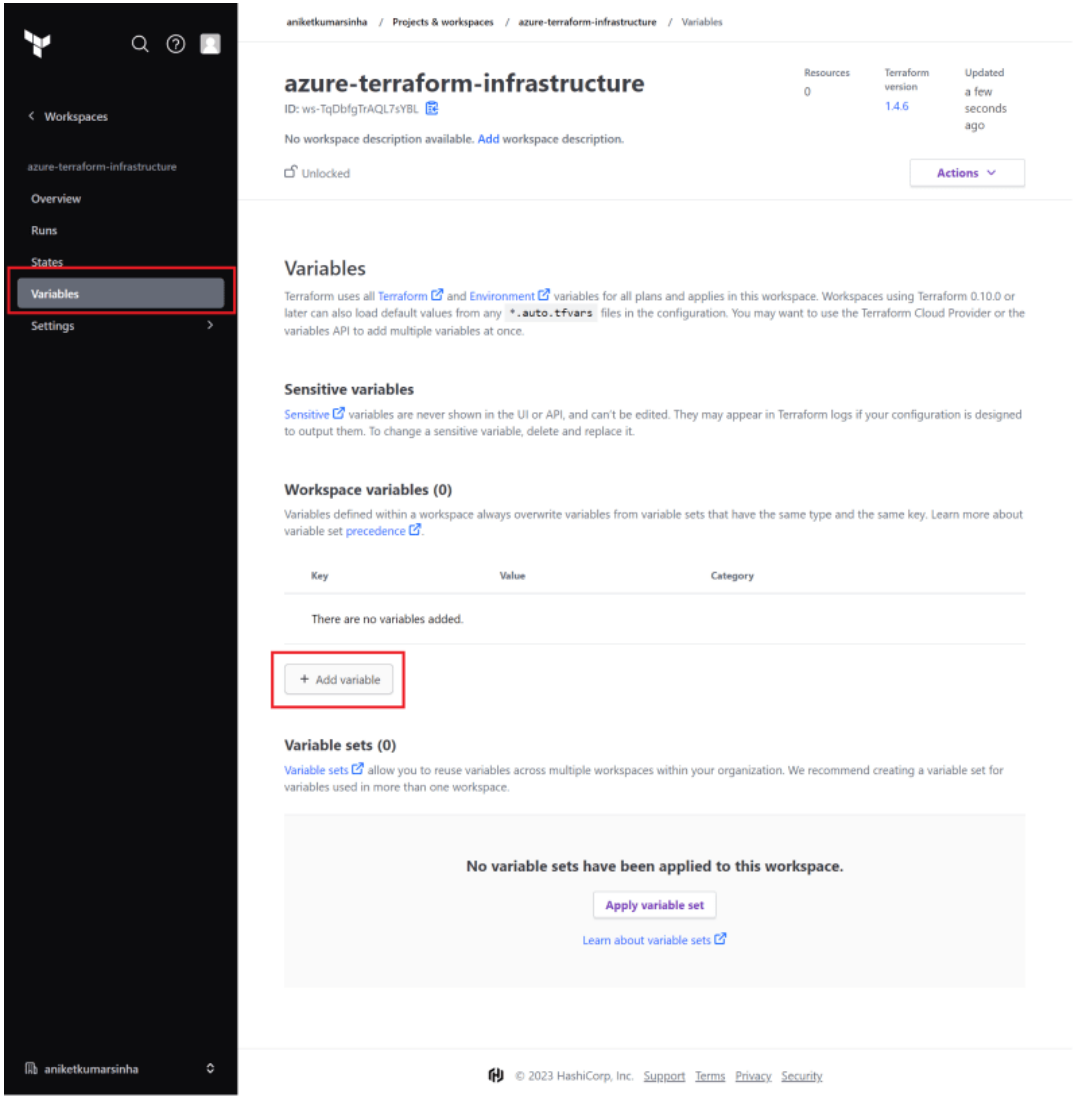
So now we have all the required values.

- Let's add these values in our Infrastructure.
  - In the provider block add all four details. We would be saving the values in Terraform Cloud variables for security purpose.

```
provider "azurerm" {
  features {}

  skip_provider_registration = "true"

  # Connection to Azure
  subscription_id = var.subscription_id
  client_id = var.client_id
  client_secret = var.client_secret
```

```
  tenant_id = var.tenant_id
}
```

- Add variables in Variables page.

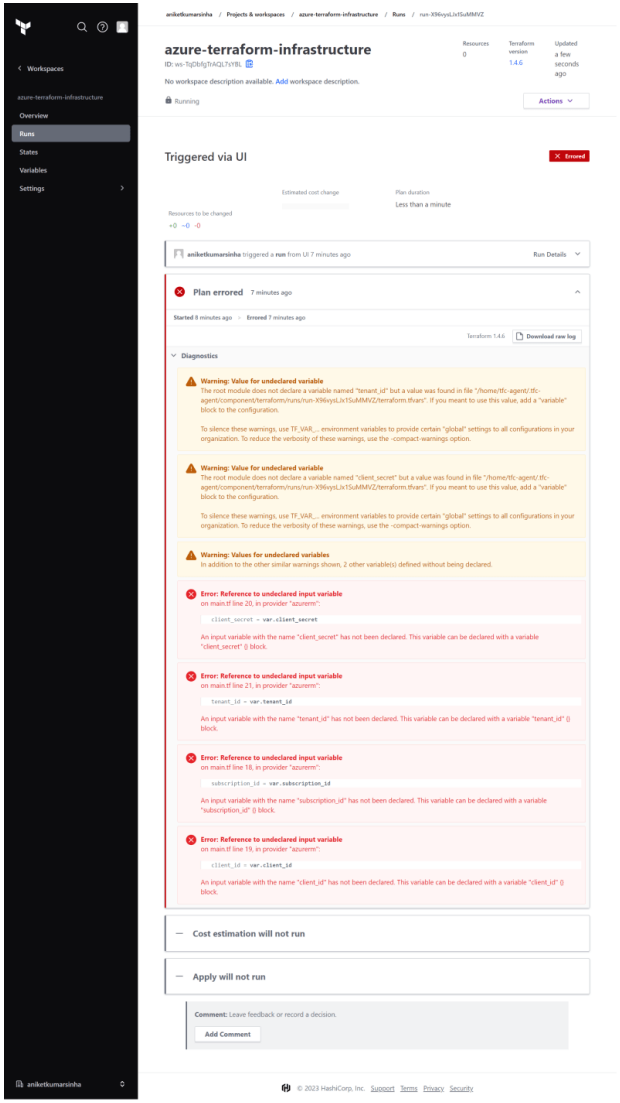


  - In `Key` add the words with var, i.e., var.`key`, and in `Value` add required Ids' and Secret.

Key	Value	Category	
client_id	8df250de-96d9-461b-b6eb-d55bf3a3e56d	terraform	•••
client_secret SENSITIVE	*Sensitive - write only*	terraform	•••
subscription_id	d8cee4c6-1c47-4bf5-88df-00377f235dcd	terraform	•••
tenant_id	84c31ca0-ac3b-4eae-ad11-519d80233e6f	terraform	•••

Let's rerun the workflow!

Ohhhhh!!!! ERROR AGAIN!!!!



But we are still left with some more code :P
We need to add variables.tf file mentioning about these variables.

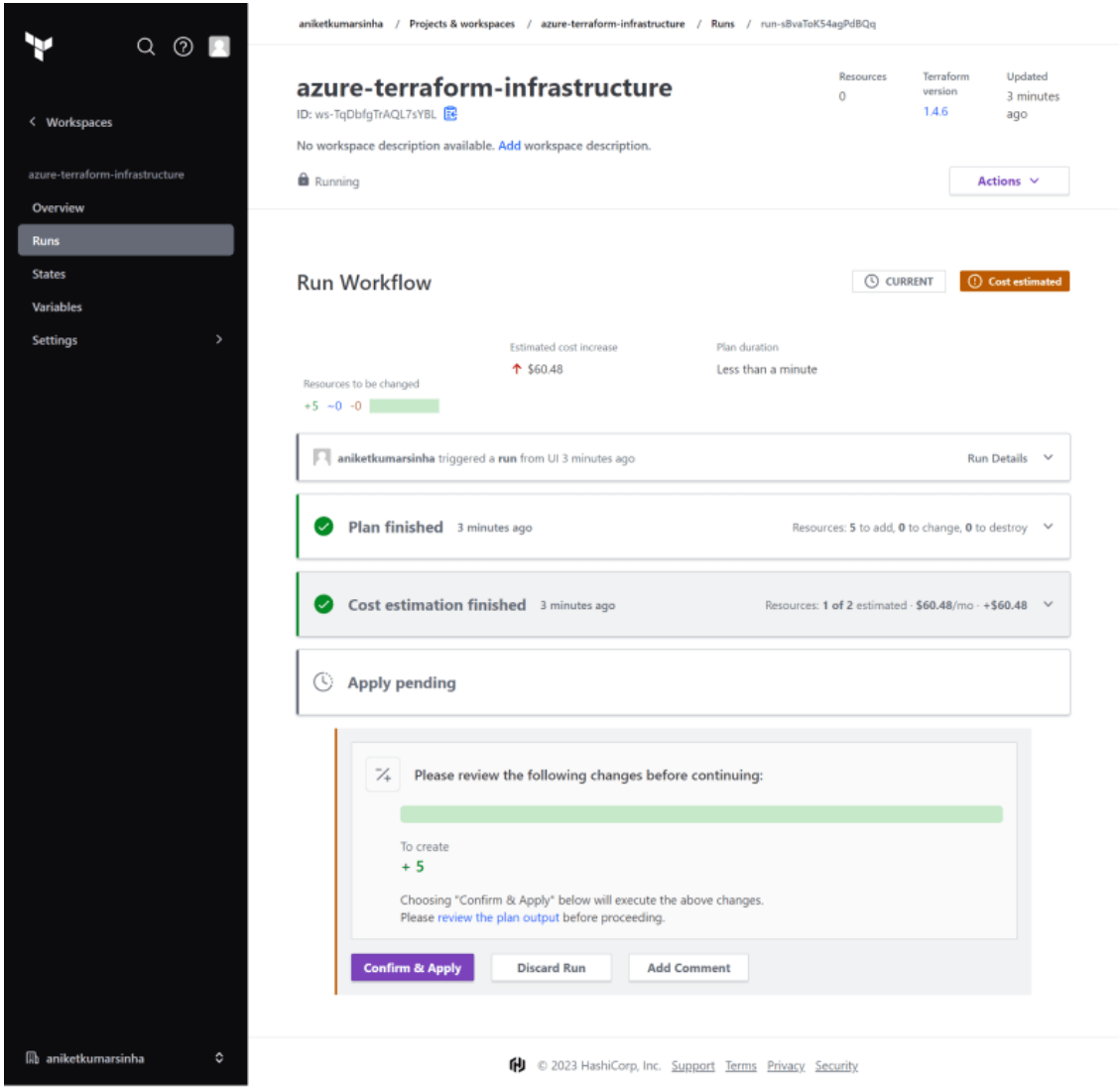
```
variable "client_id" {
  type = string
}

```

```
variable "client_secret" {
  type = string
}

variable "subscription_id" {
  type = string
}

variable "tenant_id" {
  type = string
}
```

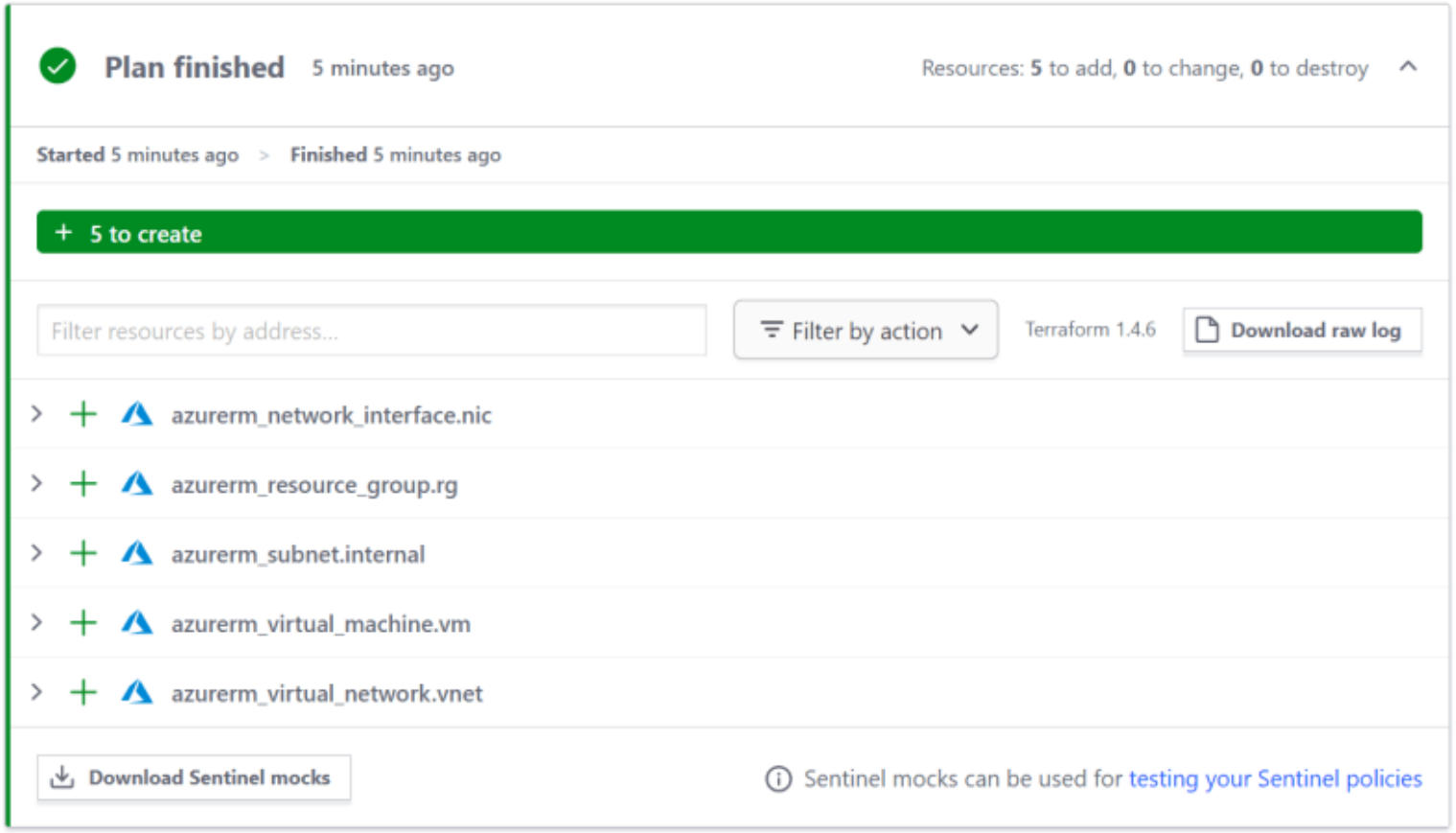
variables.tf hosted with ❤ by GitHub view raw

And now finally we can get the results from Plan and Apply phase. Let's rerun the pipeline.

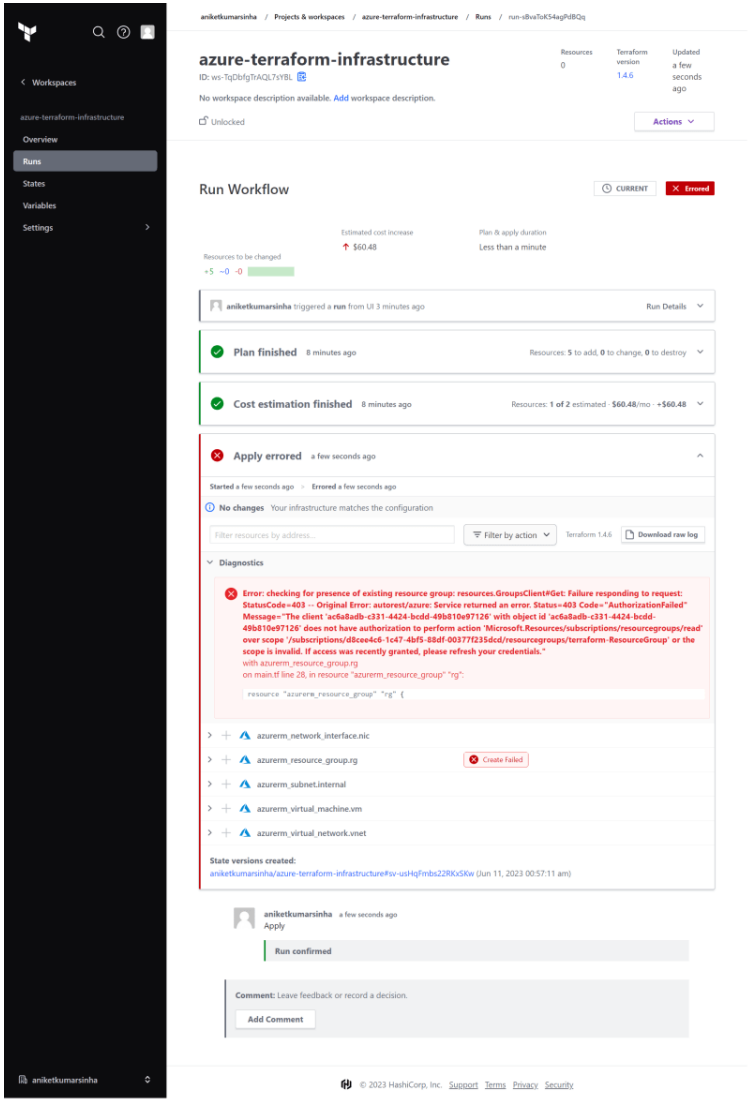
- Wohooo!! Our Plan phase ran successfully:



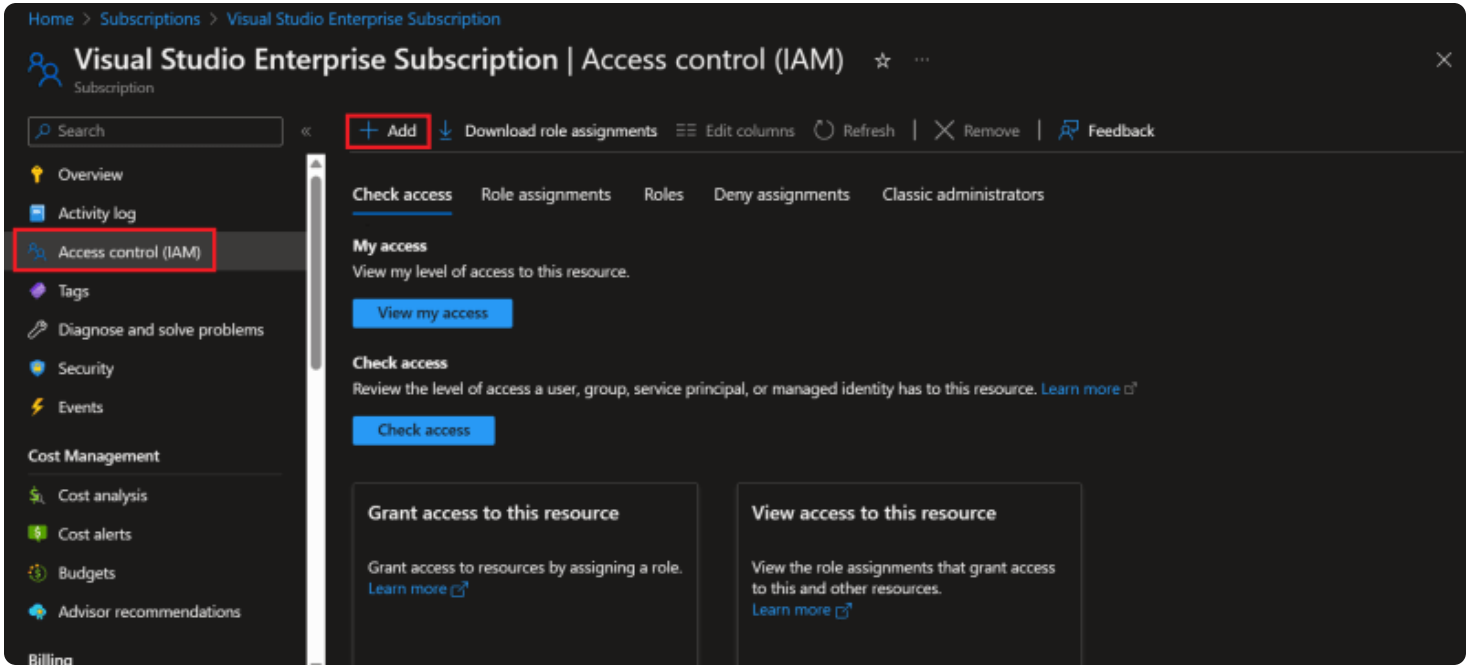
- Expand Plan phase to check what resources are getting created. And if every configuration is fine then tap on `Confirm & Apply` button at the end of phase. So by default the Apply phase does not run automatically, we need manual approval, this is to make sure that someone checks the Plan output and verifies everything and then accordingly approve for Apply or reject.


Plan finished
5 minutes ago
Resources: 5 to add, 0 to change, 0 to destroy
Started 5 minutes ago
Finished 5 minutes ago
5 to create
Filter resources by address...
Filter by action
Terraform 1.4.6
Download raw log
azurerm_network_interface.nic
azurerm_resource_group.rg
azurerm_subnet.internal
azurerm_virtual_machine.vm
azurerm_virtual_network.vnet
Download Sentinel mocks
Sentinel mocks can be used for testing your Sentinel policies

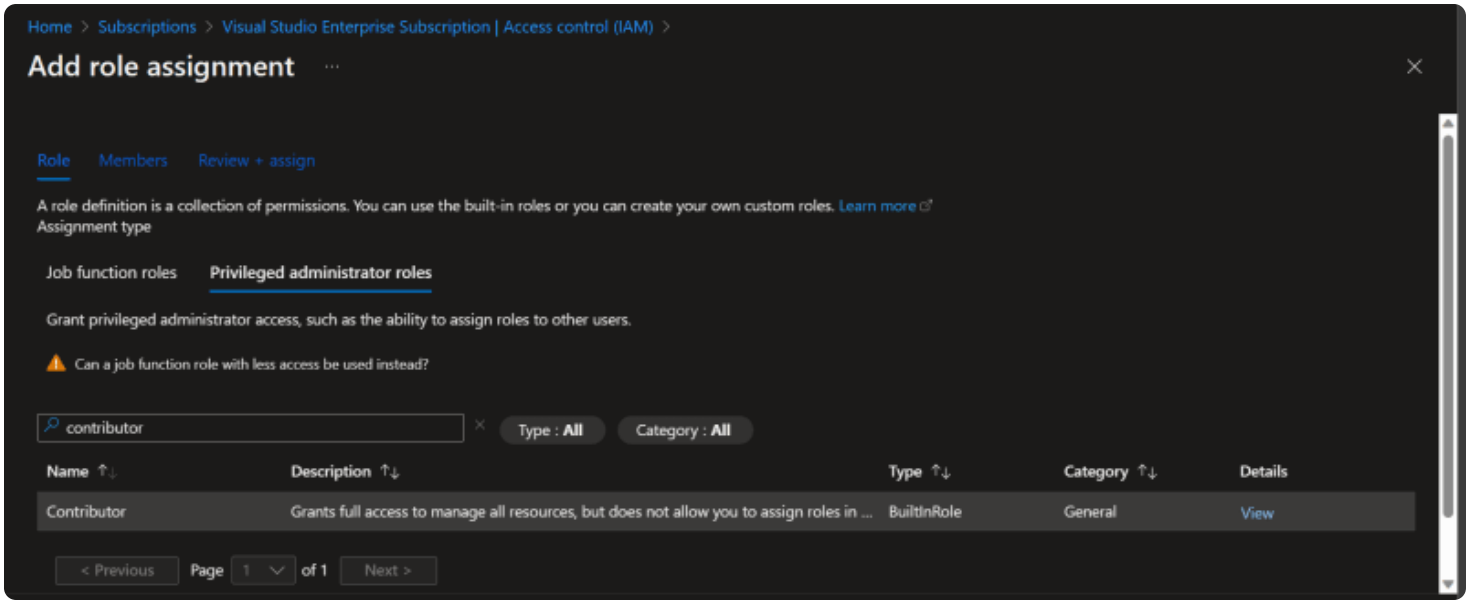
- Error again



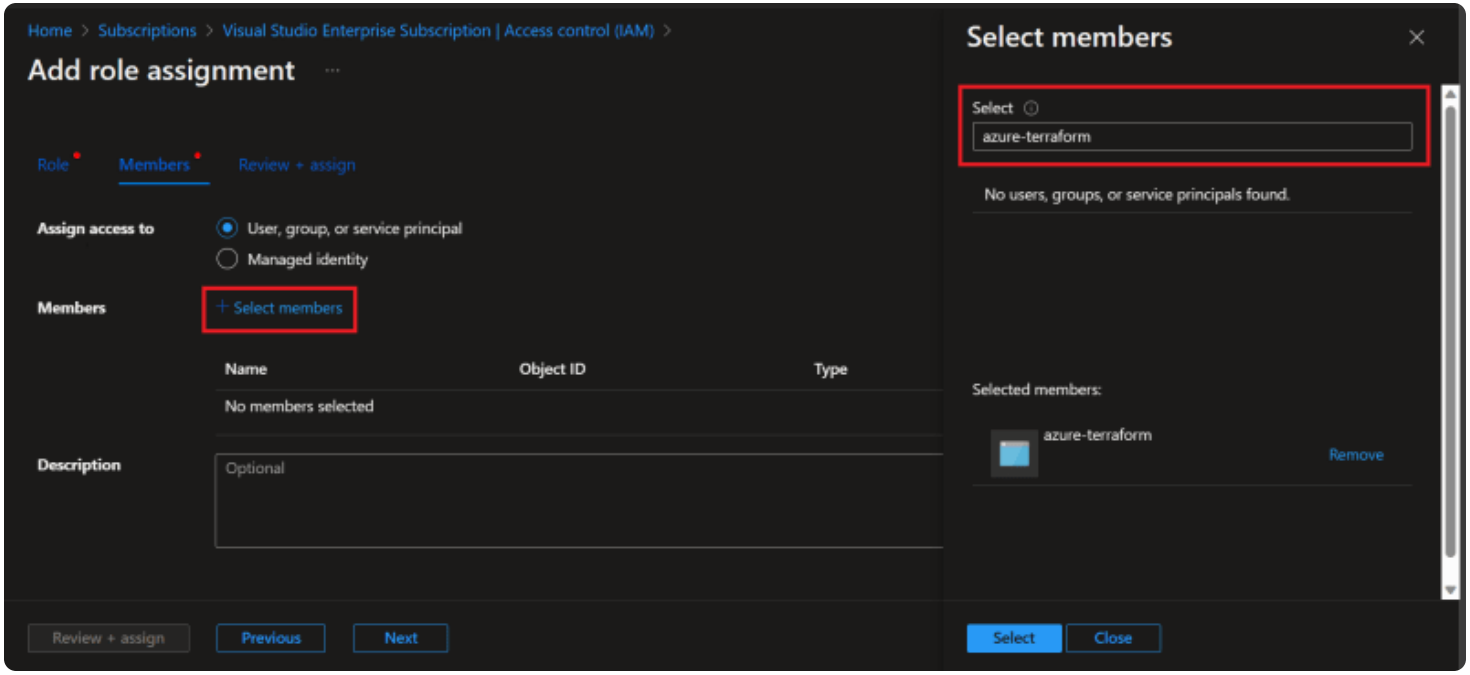
- It seems our Azure app don't have permission to add anything. Let's provide the contributor role to our app.
    - Under Subscription, got to Access Control (IAM) and Add a role there.



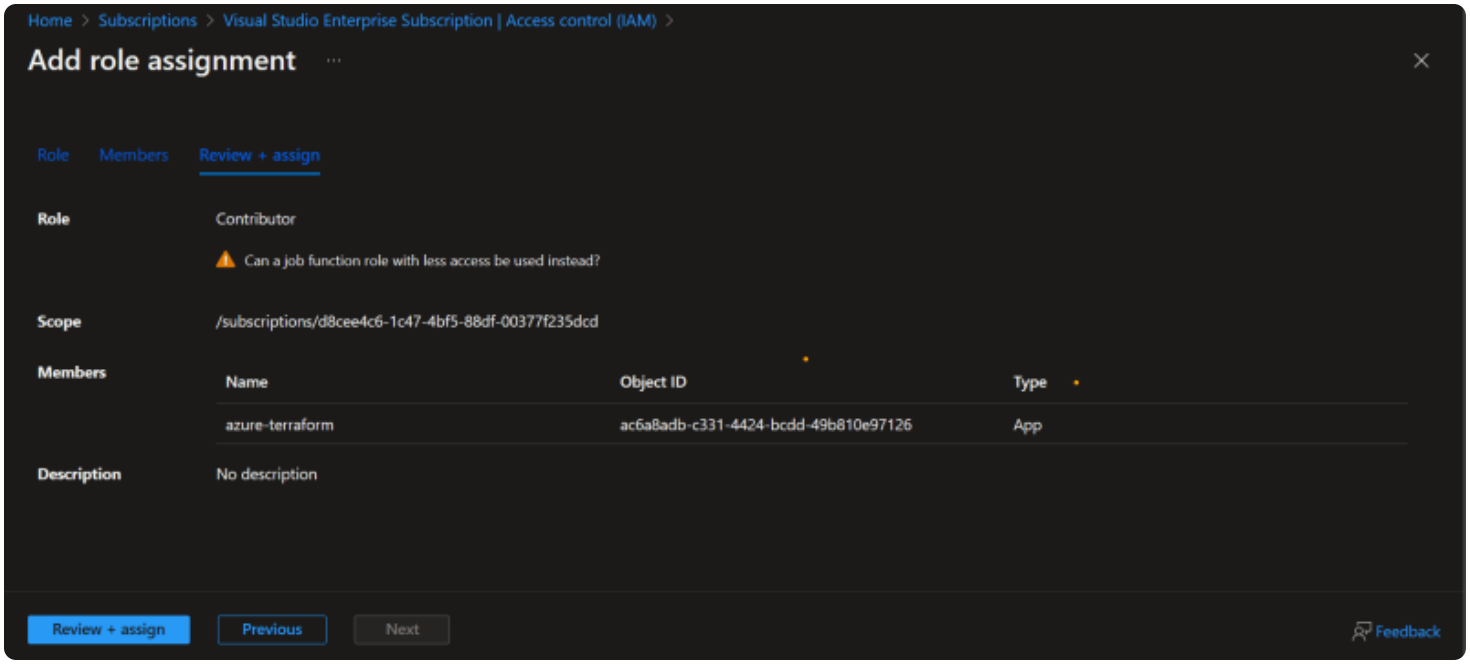
- Add a `Contributor` role under Privileged administrator roles.



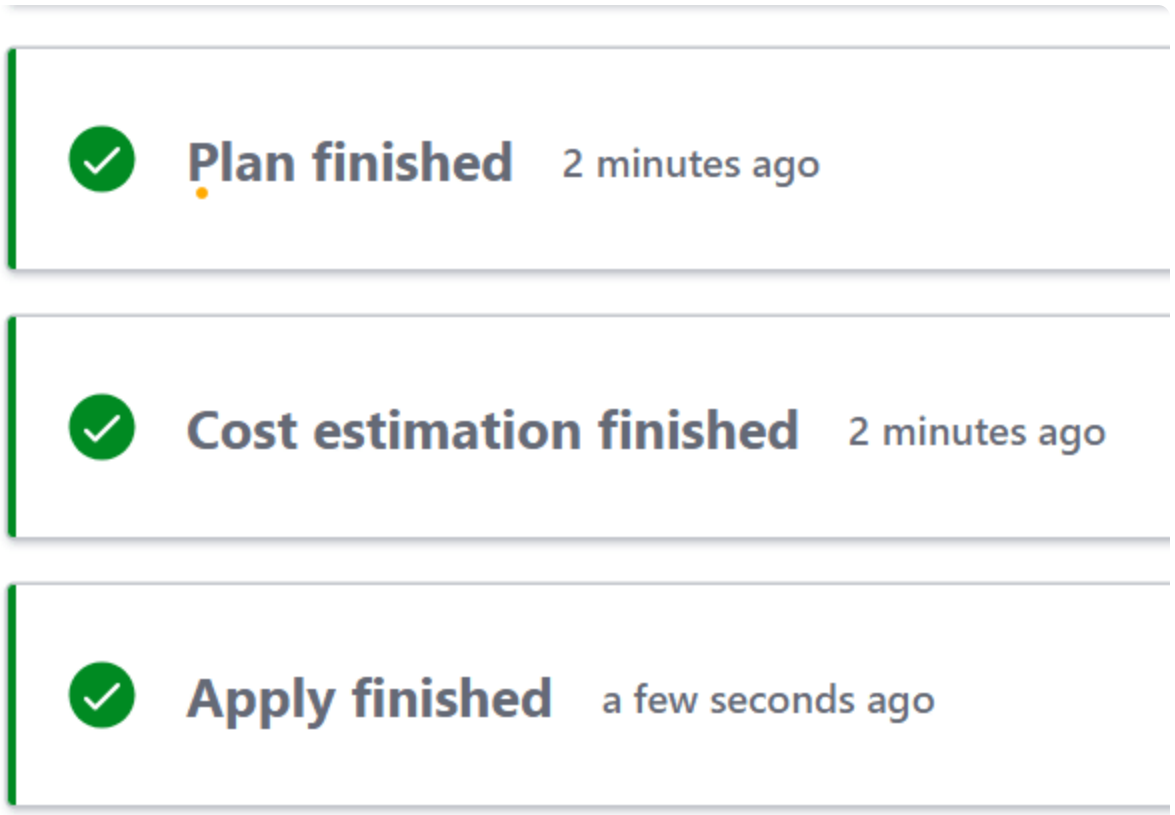
- Under Members tab, select our app as member.



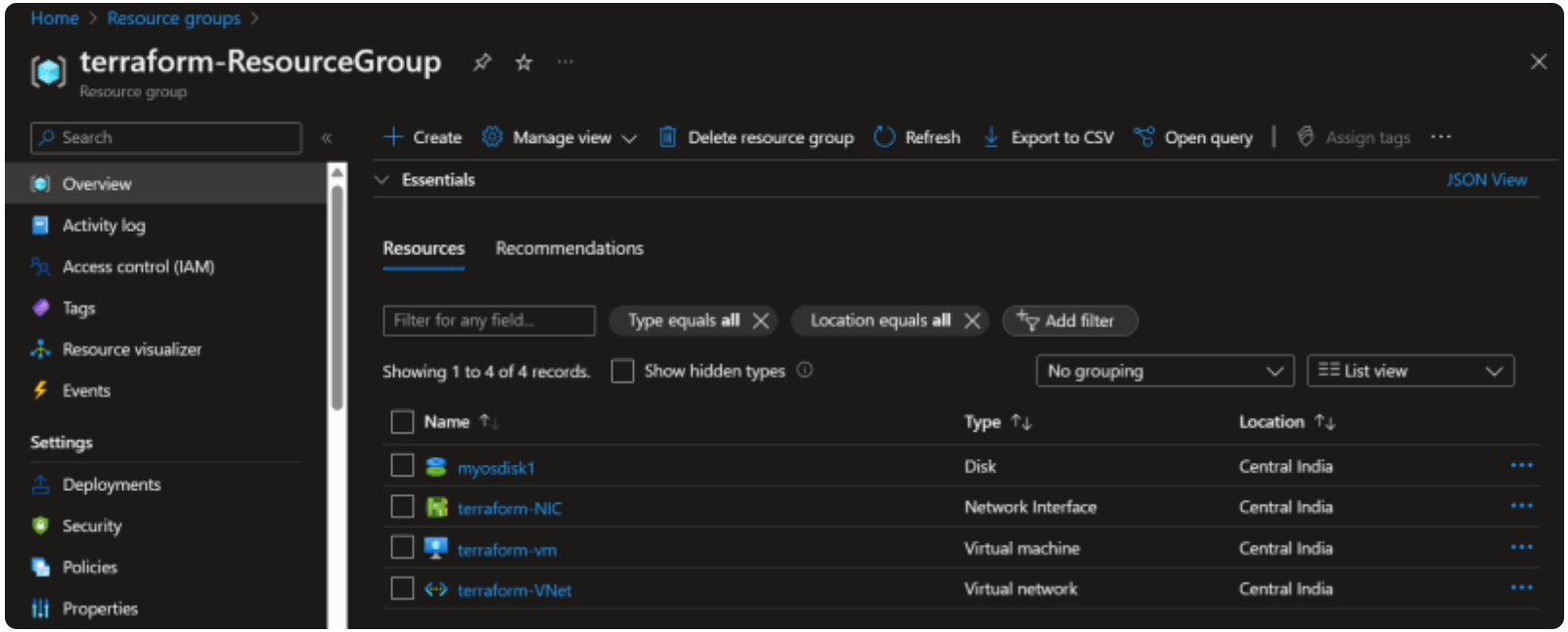
- And then tap on Review+Assign. So now our App has contributor role and can make changes over Azure subscription.



- Rerun the pipeline. And wollaahh!! Everything ran successfully!



Let's confirm over our Azure Portal too.
Yes we can see all our resources present under our subscription.



😍😍😍😍😍😍😍

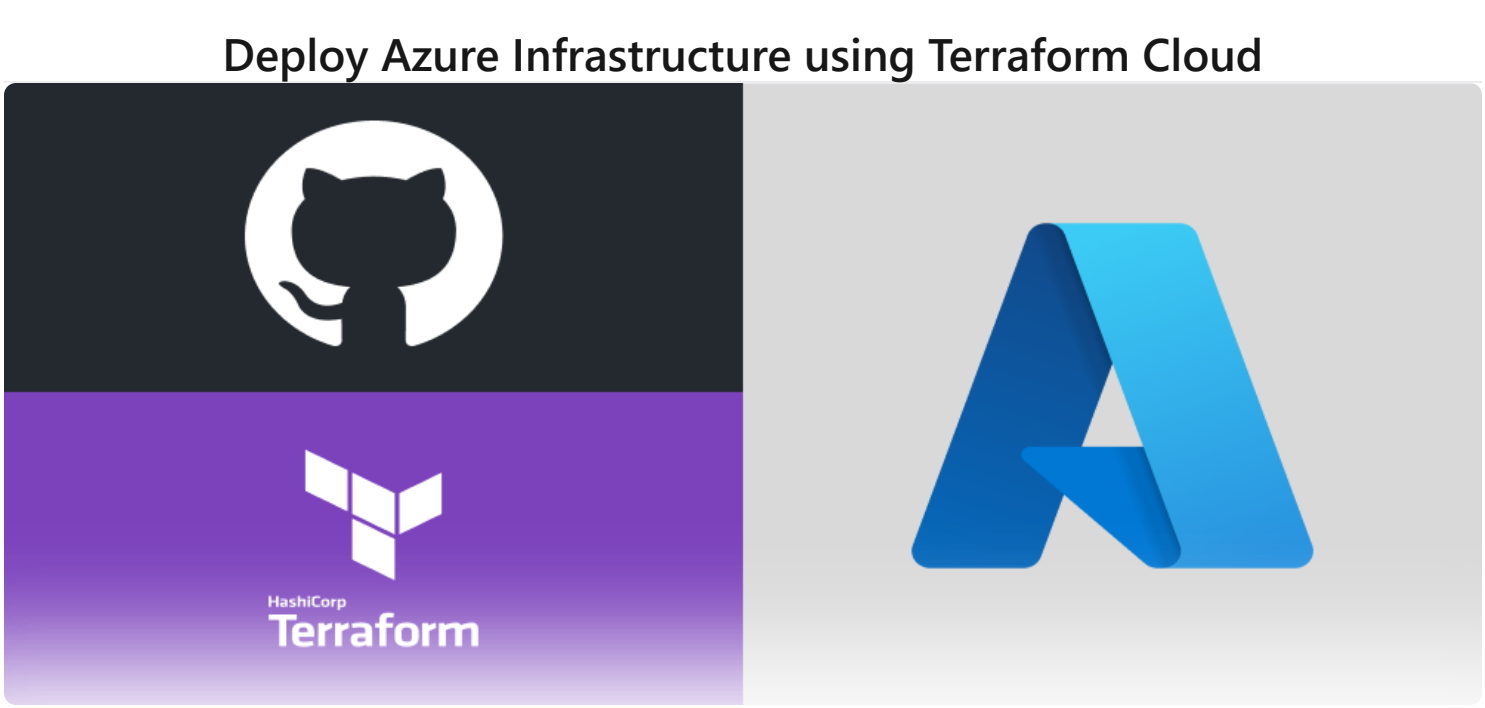
Damn!!!! We learnt so many things today. We first created the Terraform Infrastructure, setup our Terraform Cloud profile and then used it to deploy resources over Azure too. Now you can say you're a pro in Terraform and Terraform Cloud.

Let me provide you a bonus happiness. So the project which we configured over Terraform Cloud, automatically runs the workflow any branch is merged to the main branch or any new pushes are done to main branch.

You can get the whole Infrastructure as Code from this repository:

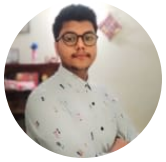
aniketkumarsinha / **azure-terraform-infrastructure**

Deploy Azure Infrastructure using Terraform Cloud



View on GitHub

Share if your like the blog and follow for more!!



Aniket Kumar Sinha Follow

Cloud Consultant at Rapid Circle | Gold Microsoft Learn Student Ambassador | GDSC Lead'21

Terraform Cloud (2 Part Series)

Top comments (0)

5. Conversations in DevOps are delving into the ease of **testing in production** with canary deployments and the efficiency gains from new **Docker Extensions**.

Say Goodbye to Docker Volumes 👋

Jonas Scholz · Nov 27

#docker #webdev #tutorial #devops

Explaining SSH to my Uber Driver

Jessica Wang · Nov 27

#ssh #softwareengineering #webdev #devops

🔥5 Must have tools to install on your Kubernetes cluster ✨🚀

Eden Federman for Odigos · Nov 27

#webdev #programming #devops #opensource

Testing in Production with Canary Deployments: A How-To Guide

Pradumna Saraf · Nov 29

#opensource #devops #development #programming

5 Docker Extensions to make your development life easier

Pradumna Saraf · Nov 27

#docker #development #programming #devops

TiL

This is Learning

Free, open and honest software education.

Read our welcome letter which is an open invitation for you to join.

Join us as a writer

More from This is Learning

The Most Upvoted Visual Studio Code Feature

#vscode #microsoft #github #productivity

Set up GitHub Codespaces for a .NET 8 application

#dotnet #github #productivity #docker

GitHub Codespaces: A Faster Way to Develop in the Cloud

#github #docker #productivity #webdev