***Terraform***

What is Terraform?

Terraform is an open-source IaC (Infrastructure-as-Code) tool for provisioning and managing infrastructure. It codifies infrastructure in configuration files that describe the desired state for your topology. Terraform enables the management of any infrastructure - such as public clouds, private clouds, and SaaS services - by using [Terraform providers](https://www.terraform.io/language/providers).

Benefits of Terraform with Azure?

**Automate infrastructure management**

The Terraform template-based configuration file syntax enables you to configure Azure resources in a repeatable and predictable manner. Automating infrastructure includes the following benefits:

* Lowers the potential for human errors while deploying and managing infrastructure.
* Deploys the same template multiple times to create identical development, test, and production environments.
* Reduces the cost of development and test environments by creating them on-demand.

**Understand infrastructure changes before being applied**

As a resource topology becomes complex, understanding the meaning and impact of infrastructure changes can be difficult.

The Terraform CLI enables users to validate and preview infrastructure changes before application of the plan. Previewing infrastructure changes in a safe manner has several benefits:

* Team members can collaborate more effectively by understanding proposed changes and their impact.
* Unintended changes can be caught early in the development process.

Terraform Installation?

Goto 🡪 <https://www.terraform.io/downloads>

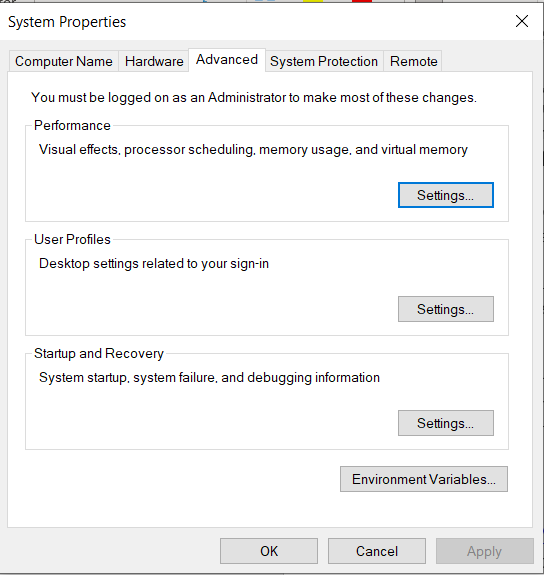
And download the binaries.

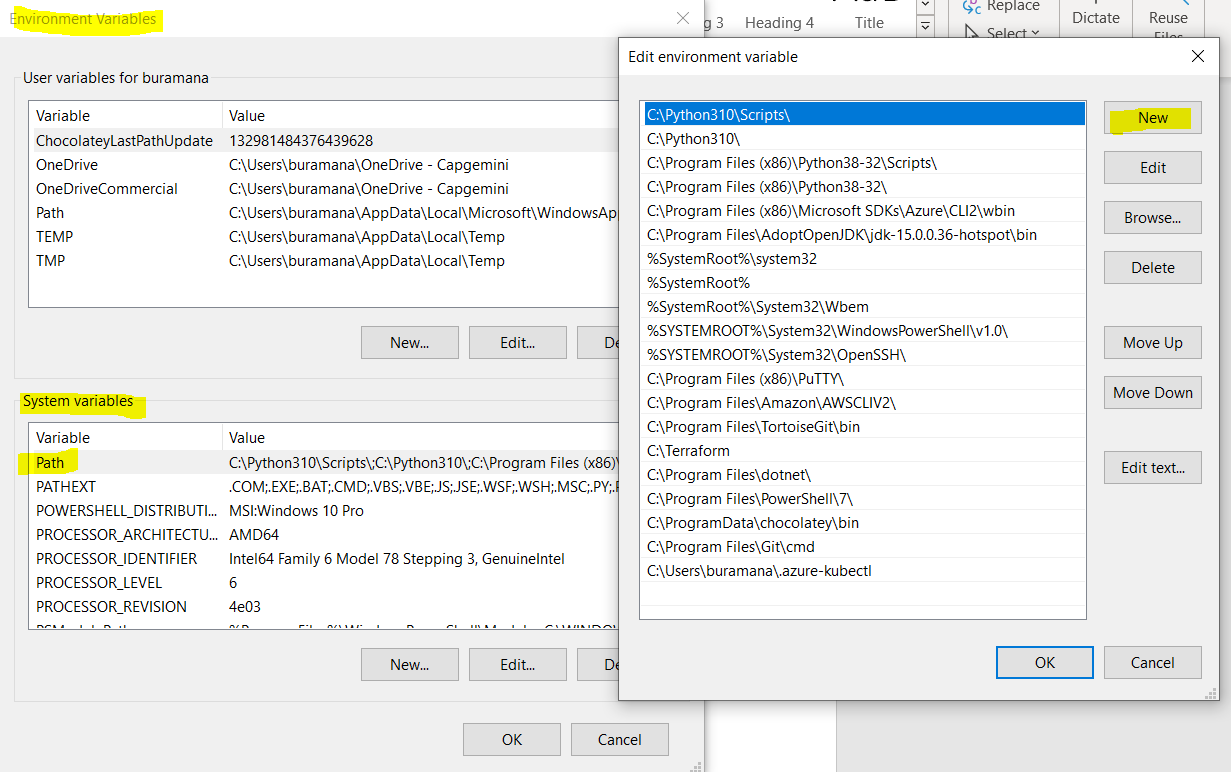
Extract the zip and copy the terraform.exe to desired folder.

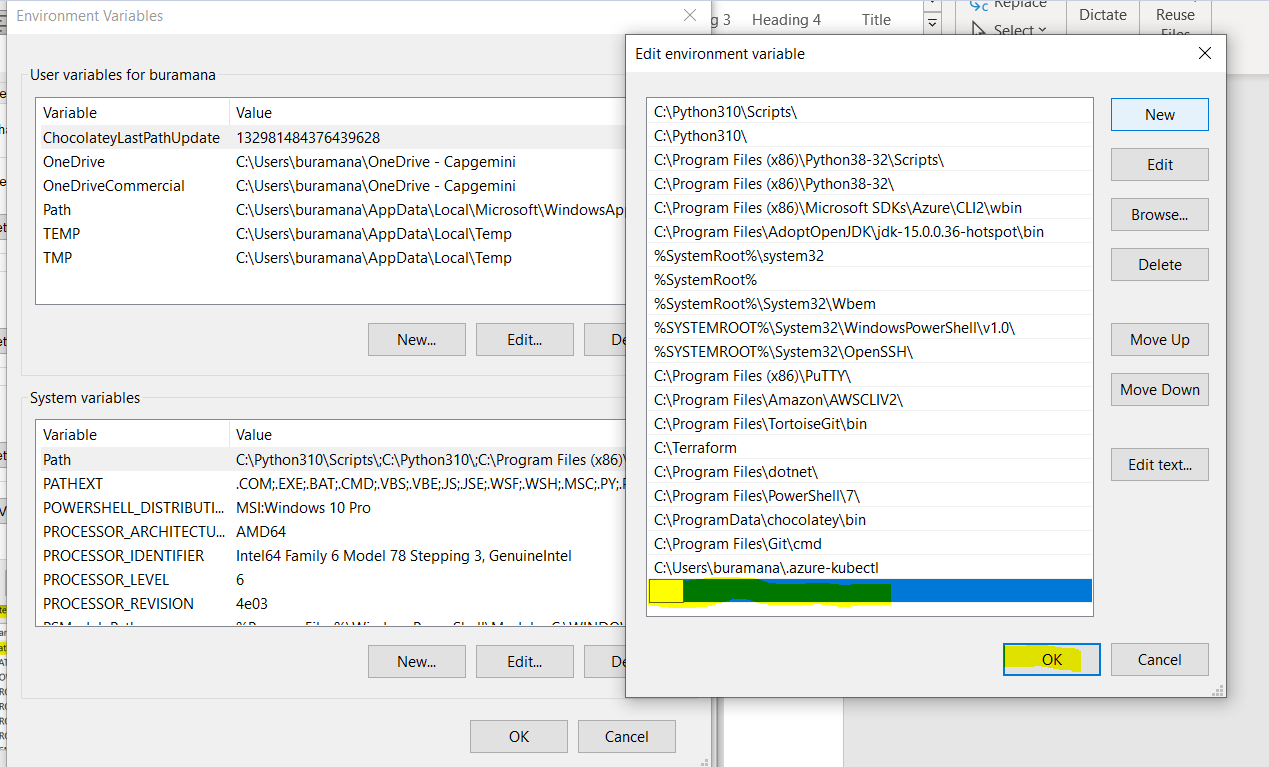
Set up the environment variables in windows to run terraform from anywhere in system:

Press Windows key and search for the environment variable and click the edit system environment variables.

Click on environment variable and search for ‘path’ in System variables. Click new and give the terraform.exe path, then click Ok, Ok and Apply.







Install VS code and then go to extensions, search for Terraform extension and install it to get the code suggestions and snippets.

Important Files in Terraform?

**main.tf** 🡪 which is called as root module and has the infrastructure configuration.

**variables.tf** 🡪 variables definition to use in configuration files

<**file-name>.tfvars** 🡪 Values for the variable which are defined in variables.tf to pass at runtime to main.tf

**terraform.tfstate** 🡪 this file will be created after terraform apply command and it has the infrastructure information in JSON format.

Important Terraform Commands?

**terraform init** 🡪 to initialise the working directory as terraform workspace, it will download the required plugins and modules for provider

**terraform validate** 🡪 to validate the configuration files.

**terraform plan 🡪**creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure.

**terraform apply 🡪** performs a plan just like terraform plan does, but then actually carries out the planned changes to each resource using the relevant infrastructure provider's API. It will create/deploy resources in target environment.

**terraform destroy 🡪** performs to delete the resources in target environment which are managed or created by terraform

Authenticating Azure using Azure Service Principal and Client Secret:

1. Login to Azure using Az cli

az login

It will be open the browser for password.

1. To list the subscriptions

az account list

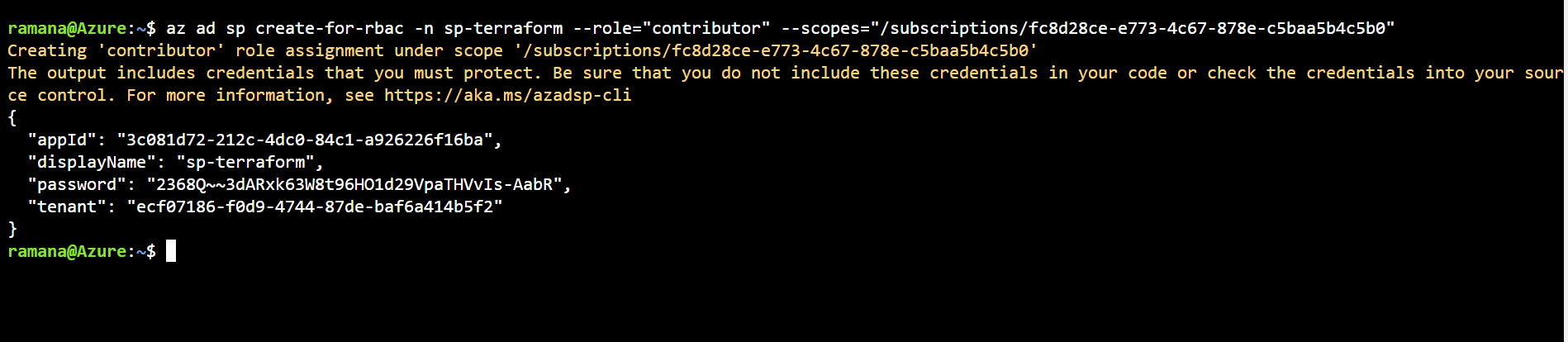
1. To set the subscription in which we need to create resources

az account set –subscription <subscriptionID/name>



1. To create service principal using azure cli

az ad sp create-for-rbac -n <name of the serviceprincipal> –role=”contributor” scopes="/subscriptions/SUBSCRIPTION\_ID"



Copy the result values to use for further in terraform configuration files.

In terraform provider block we can use service principal values as below.

provider "azurerm" {

  features {}

  subscription\_id = "fc8d28ce-e773-4c67-878e-c5baa5b4c5b0"

  client\_id       = "3c081d72-212c-4dc0-84c1-a926226f16ba"

  client\_secret   = "2368Q~~3dARxk63W8t96HO1d29VpaTHVvIs-AabR"

  tenant\_id       = "ecf07186-f0d9-4744-87de-baf6a414b5f2"

}

We can configure the service principal details as environment to login to azure instead giving values in terraform configuration file

Creating environment variables from command prompt

***In linux:***

***$ export ARM\_CLIENT\_ID="3c081d72-212c-4dc0-84c1-a926226f16ba"***

***$ export ARM\_CLIENT\_SECRET="2368Q~~3dARxk63W8t96HO1d29VpaTHVvIs-AabR"***

***$ export ARM\_SUBSCRIPTION\_ID="fc8d28ce-e773-4c67-878e-c5baa5b4c5b0"***

***$ export ARM\_TENANT\_ID="ecf07186-f0d9-4744-87de-baf6a414b5f2"***

***In Windows:***

***$env:ARM\_CLIENT\_ID="3c081d72-212c-4dc0-84c1-a926226f16ba"***

***$env:ARM\_CLIENT\_SECRET="2368Q~~3dARxk63W8t96HO1d29VpaTHVvIs-AabR"***

***$env:ARM\_SUBSCRIPTION\_ID="fc8d28ce-e773-4c67-878e-c5baa5b4c5b0"***

***$env:ARM\_TENANT\_ID="ecf07186-f0d9-4744-87de-baf6a414b5f2"***

==================================================================================

Terraform configuration files will be written in Harshi Corp Language(HCL).

Here is the high level syntax of terraform file

<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {

  # Block body

  <IDENTIFIER> = <EXPRESSION> # Argument

}

Example:

It is resource block

resource "azurerm\_resource\_group" "rg" {

  name     = "rg-terraform"

  location = "eastus"

}

Terraform configuration file to create resource group and virtual network.

terraform {

  required\_providers {

    azurerm = {

      source  = "hashicorp/azurerm"

      version = "=3.0.0"

    }

  }

}

# Configure the Microsoft Azure Provider

provider "azurerm" {

  features {}

  subscription\_id = "fc8d28ce-e773-4c67-878e-c5baa5b4c5b0"

  client\_id       = "3c081d72-212c-4dc0-84c1-a926226f16ba"

  client\_secret   = "2368Q~~3dARxk63W8t96HO1d29VpaTHVvIs-AabR"

  tenant\_id       = "ecf07186-f0d9-4744-87de-baf6a414b5f2"

}

resource "azurerm\_resource\_group" "rg" {

  name     = "rg-terraform"

  location = "eastus"

}

resource "azurerm\_virtual\_network" "vnet" {

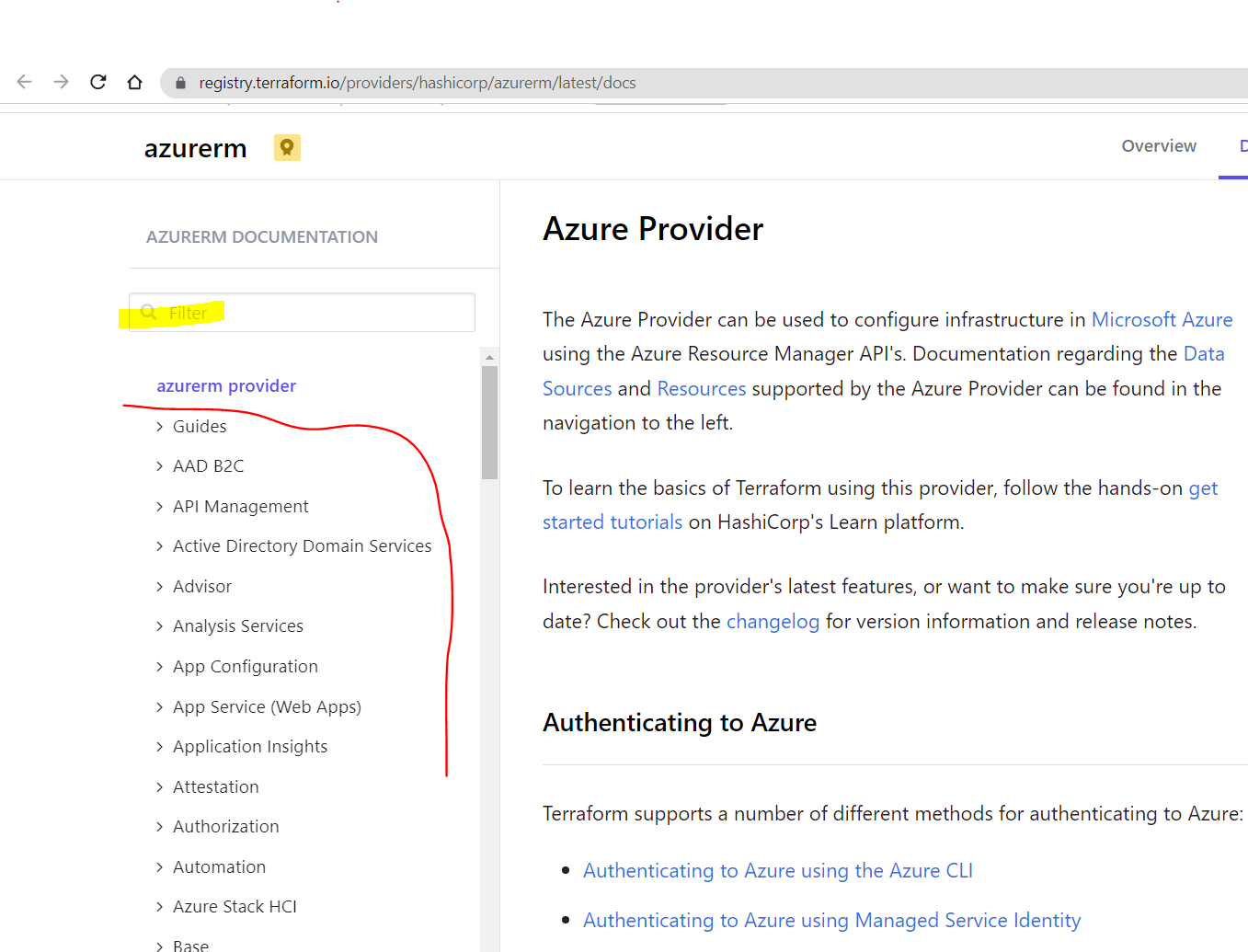
  name = "vnet-terraform"

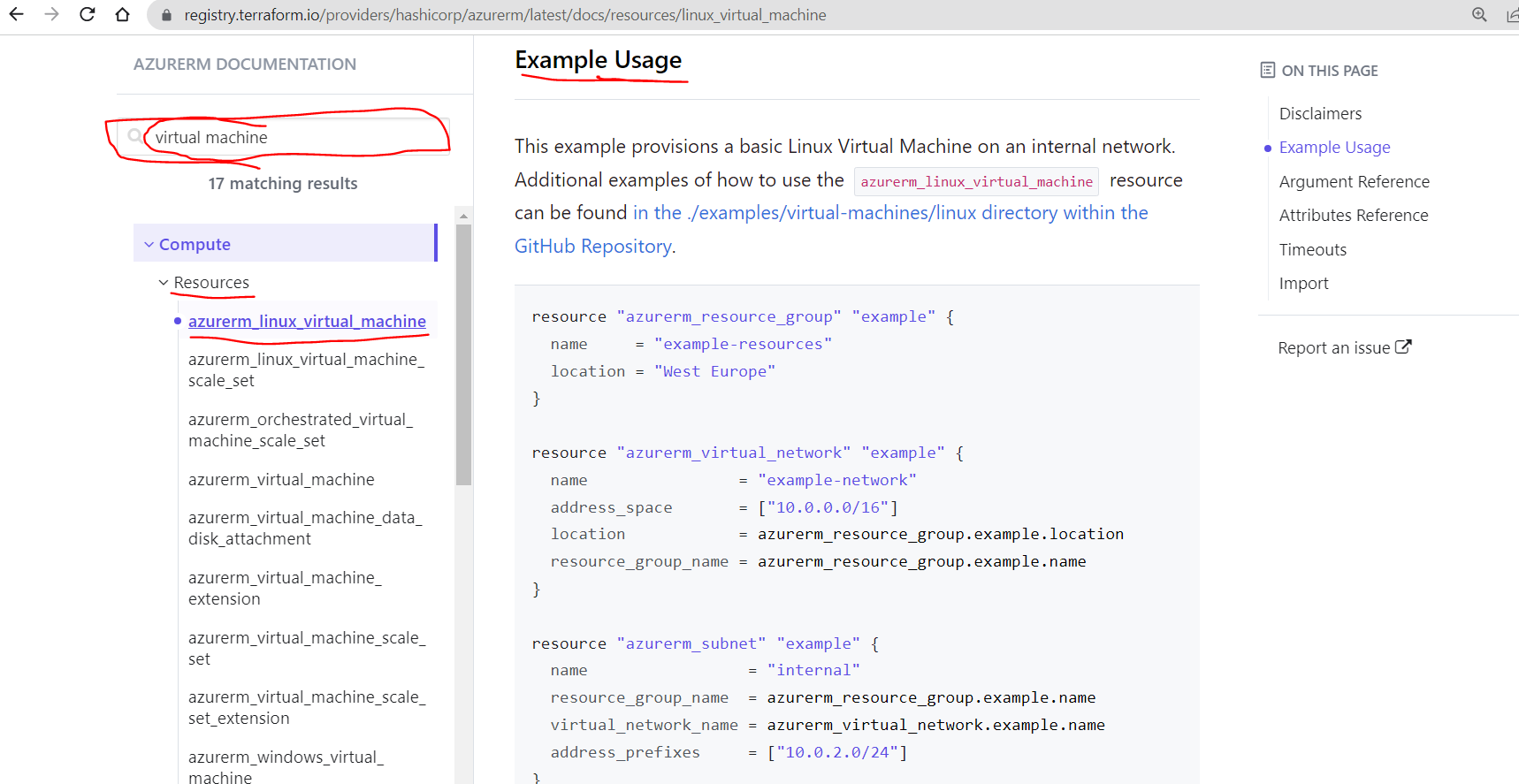
  resource\_group\_name = azurerm\_resource\_group.rg.name

  location = azurerm\_resource\_group.rg.location

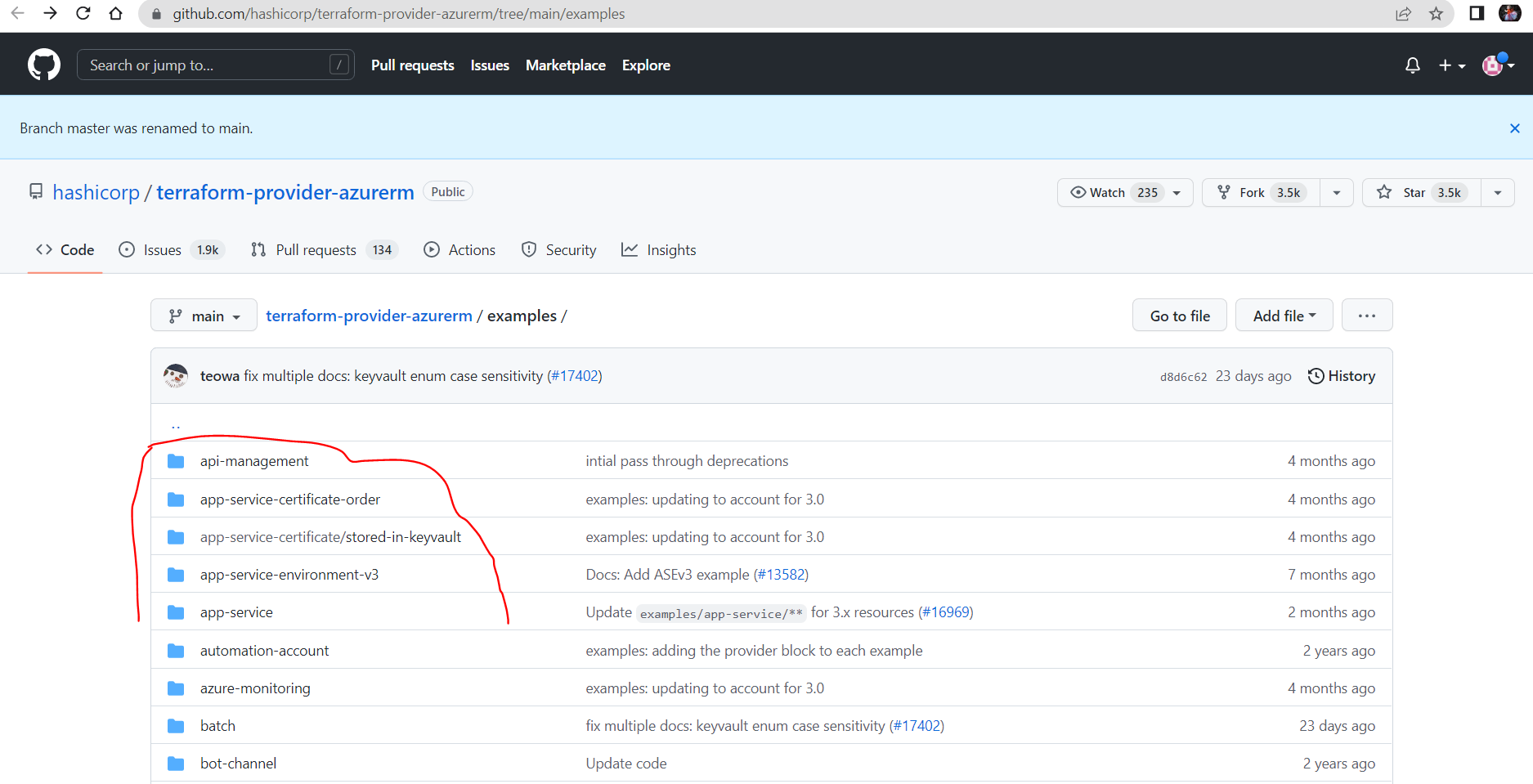
  address\_space = [ "10.1.0.0/16"]

}

For more details on terraform azure provider resource [Click Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs)



For sample terraform configuration files [Click here](https://github.com/hashicorp/terraform-provider-azurerm/tree/main/examples) to create azure resources.



For complete documentation on Terraform [Click Here](https://www.terraform.io/language)

To pass the variables in command line

*terraform plan -var "rg=rg-terraform" -var "location=westus" -var "vnet-name=vnet-terraform" -var 'ipAddress=[\"10.2.0.0/16\",\"172.16.1.0/24\"]'*

Backend for Statefile:

