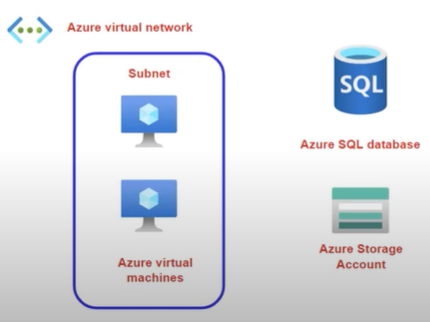
**Terraform:** it is an open-source tool and it is mainly used for provisioning and managing the cloud infrastructure, lets say for an example if we want to create a set of resources in Azure or AWS (as shown below) like Virtual network, Virtual Machine, Subnets, Storage accounts…etc,



With Terraform we can define a configuration file then we can submit the configuration file on to the terraform executable and then terraform will automatically deploy those resources based on our terraform configuration file, it basically a quick way to go ahead and automate the deployment of our resources, we can use terraform to build our environment in a matter of minutes using Infrastructure as code(IaC).

Terraform works with a variety of platforms, like Azure, AWS, GCP…etc, we can use Terraform to deploy the infrastructure in any of the cloud platforms as per our project requirements.

**Terraform configuration file::** This tells terraform how to manage the infrastructure, how to deploy/provision the infrastructure in cloud computing.

**Blocks/Modules-Terraform:** These are used to represent the configuration of an object, the resource block is used to represent the infrastructure that we want to deploy, the resource block will contain the resource type and the name as shown below

Text

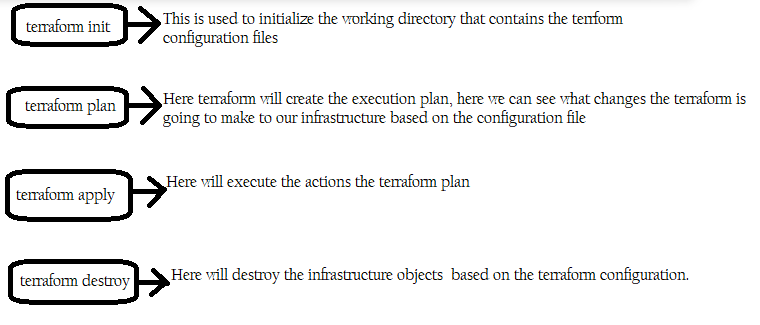
Description automatically generated

The resource type and the name will then become the resource identifier in the format of resource\_type.resource\_name

Here the name and location are the arguments to ensure the right properties values are in place when defining the resource and the arguments will differ depending on the type of resource that needs to be created.

If we want to deploy Azure Resources, then we use Azure provider and if we want to deploy AWS resource then we use AWS provider.

**Workflow/Steps for Terraform scripts executions:**

****

**Download and run the terraform:**

To download and run terraform in our laptop go to the link: <https://developer.hashicorp.com/terraform/downloads>

* After going on above link>>windows(if we working on windows laptop, here we have options for other different O/S also, choose accordingly)>>Download AMD64(here the zip folder will get downloaded in our laptop in downloads)>>create an **apps**(folder) in C drive>>paste the terraform zip folder in this **apps** folder>>extract(right click on the zip folder)>>Win RAR>>Extract Here) the zip file inside the apps folder>>and now delete the zip folder from our apps folder as we have extracted the terraform exe from zip
* Now come inside the control panel>>View by: Category>>System and Security>>System>>Advance system settings>> a new window will get open and in that go to Environment variables>>double click on Path(in below box)>>New>>C:\apps\>>
* Now open and search for cmd prompt in our laptop>>right click and run as administrator>>and pass the command to check the terraform version

**terraform version**>>command to check which version of terraform is installed.

* Now we have to install the visual studio code in our laptop and for this go to the below link to install visual studio code(VSC)

<https://code.visualstudio.com/>

* Now click on Download for windows>>wait for some **10-15** mins till the visual studio code(VSC) exe gets downloaded in our laptop>>double click on the visual studio code(VSC) and run it in our laptop and launch the Visual Studio Code(VSC)
* Create a tmp folder in C drive in our local laptop
* Click on file(on top) in VSC>>Open folder>>browse to tmp(this folder we can create manually in C drive to store our terraform config files)>>Select Folder.
* Now click on extensions (left hand side, 2nd last ikon) under VSC(here we can install various extensions that helps to extend the functionality of visual studio code and when it comes to terraform)>>on top search type **terraform>>Click on install for Hashicorp Terraform carefully(this is generic extension for terraform itself)** and wait for some **5-10 mins** till it’s get installed.
* Now in VSC search for terraform>>and **click on** **Install for Azure terraform carefully (this extension which also helps us to work with Azure terraform)** and wait for some **5-10 mins** till it’s get installed.
* Now all the configurations and installation is completed in our laptop and now click on File (at the very top)>>New file>>Text File Built-In>>click on Select a language>>in search type terraform and click with a mouse on it
* Click on File (at the very top)>>Save as>> and give the name as **main.tf** (and this is going to be our terraform configuration file) saving this file in C:\tmp folder (this we have created @ the top)
* We are not going to create a user in Azure cloud platform instead will create an application object which is similar to creating a user which is basically an another form of an identity in Azure AD, and the details of this identity can then be embedded in our terraform configuration files and this terraform configuration file is going to run on our local laptop and it is going to submit that file in terraform executable which we have downloaded and ran in our laptop above and then terraform will use the application object credentials to authenticate on to Azure Active Directory and once this authentication is completed it will then take the commands in the terraform configuration file and it will then issue API calls so that application programming interface calls to create the resources at Azure cloud computing platform finally..

**Creating an application object in Azure cloud portal:**

* Goto Microsoft Entra ID(in azure portal)>>App registrations(left side)>>New registration>>give Name as **terraform** and leave rest of the values to default and finally click on Register(@ the bottom).
* So here we have defined an identity that we can use now by our terraform configuration file

Now we have to ensure that we give permissions onto the object on our subscription so that it can create resources as part of our subscription, and we can do this via RBAC and for this we can give a contributor role to our subscription.

Now go to subscription>>Access control (IAM)>>+Add>>Add role assignment>>Contributor>>Next>>+Select members>>search: terraform>>click on it>>Select>>Next>>Review + assign.

To get the details for below code like Subscription\_id, client\_id…etc go to App registrations inside the Azure Active Directory/Microsoft Entra ID>>click on terraform>>copy client ID and paste all the details in below script in VSC one by one…for client secret value click on Add a certificate or secret>>+New client secret>>Description: secret>>Add>>copy the value key(in the middle) and paste it in VSC for client secret.

Copy the below code & paste it in VSC carefully.

//\*\*\*Script for terraform execution versioning\*\*\*\*//

terraform {

  required\_providers {

    azurerm = {

      source = "hashicorp/azurerm"

      version = "2.93.0"

    }

  }

}

//\*\*\*\*\*Script for Authentication & Authorization details\*\*\*\*\*\*\*\*\*//

provider "azurerm" {

  subscription\_id = ""

  client\_id = ""

  client\_secret = ""

  tenant\_id = ""

  features {

  }

}

Now will create the resource block for creating a resource group in VSC as shown in below code

//\*\*\*\*script to create a Resource Group\*\*//

resource "azurerm\_resource\_group" "RGcreations" {

  name = "NareshRG"

  location = "North Europe"

  }

**After passing all the above code click on File(on top)>>Save as and pass**

**File name: main & save as type: Terraform(\*.tf)(for sure) in tmp folder in c drive and ensure this tmp folder should not be having any files saved previously.**

Now click on Terminal(on top)>>New terminal (to start executing the commands one by one and these steps we called as workflow steps)

**dir>>**this command shows that our main.tf file is saved in c:\temp folder.

**clear>>** this command is to clear the screen

**Note:** Before executing the command **terraform init** in VSC terminal ensure you should be or your cursor should be at this path **C:\tmp** in your Visual Studio Code Terminal

**terraform init>>**this command is used to initialize our working directory; it downloads the terraform provider for Azurerm module.

**dir**>>with this command we are ensuring that main.tf file and terraform lock file is saved.

**clear>>** this command is to clear the screen.

**terraform plan -out main.tfplan>>**this command creates a plan based on what code we have written in main.tf file, with this command terraform will check whether a resource with this name is already exist in our subscription or not

**clear>>**this command is to clear the screen.

**terraform apply main.tfplan>>** Now this command is used to create the resources in Azure Cloud platform.

**Note:** Before executing the scripts always make sure the same name RG is not present in cloud portal, terraform object is created in our App registration.

**Multiple resource provisioning in Azure Cloud Computing using terraform scripting:**

**Note:** Whatever the changes we make in the main.tf file…like code addition, code modification always saves the file first and execute the terraform init command every time to deploy the changes followed with terraform plan & terraform apply commands.

Like wise we can deploy multiple resources using terraform scripting by going to the official link/documentation as shown below

<https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>

Terraform is an open-source, infrastructure as code software (IaC) tool, created by Hashicorp and written in the Go programming language.

Infrastructure as code is the process of managing infrastructure in a file or files, rather than manually configuring resources in a user interface (UI)

Here resources are nothing but virtual machines, Elastic IP, Security Groups, Network Interfaces...Terraform code is written in the Hashicorp Configuration Language (HCL) in files with the extension .tf

Terraform allows users to use Hashicorp Configuration Language (HCL) to create the files containing definitions of their desired resources on almost on any provider (AWS, GCP, Azure, Digital Ocean, OpenStack, etc) and automates the creation of those resources at the time of apply.

Table

Description automatically generated

**Terraform Scripts execution directly from Azure Portal:**

**** ****

**Below are some basic commands of Terraform:**

**terraform init:**

The terraform init command is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration

**terraform fmt:**

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style.

**terraform validate:**

The terraform validate command validates whether a configuration is syntactically valid or not.

**terraform plan:**

The terraform plan command is used to create an execution plan.

This command is a convenient way to check whether the execution plan for a set of changes matches your expectations without making any changes to real resources or to the state.

**terraform apply:**

terraform apply to create the infrastructure on Azure or AWS .

**terraform destroy:**

The terraform destroy command is used to destroy the Terraform-managed infrastructure.