**difference between arm and terraform**

1. **arm is developed by Microsoft and used only for azure**

| arm | Terraform |
| --- | --- |
| developed by microsoft | hashi corp |
| support only azure | any cloud platforms |
| u can’t able delete resource accounts | can delete |
| we have two types of deployments  1)incremental  2)decremental | we have state file — re usable  it check state of infrastructure |
|  |  |
|  |  |

used for infrastructure as code

installation

1)download terraform and copy to c drive by creating new folder

2)add the folder path in environmental variables (cmd will open)

pre - requisite

1)commandsƒ

init - its going to add the latest commands — it will provide required things to ur directory

validate - validate configuration and syntax

plan - shows changes required for the current configuration

apply - create the infrastructure in azure/aws environment

(terraform apply —-auto-approve — it will directly creates the resource with asking for permission)

destroy - to delete previously created infrastructure

2)open visual studi code

3)in the terminal , path should be terraform folder in c drive

eg— cd… — to get in to back folder

cd c:\terraformfolder — to change directory to terraform folder

4)download azure cli —- to connect visual studio to azure we need azure cli

5)az login in terminal — it will open the azure portal

6)down load terraform related plugins in visual studio

terraform should be saved with .tf extension

.tfstate (json file)—- it will have current state of infrastructure. this is mandatory to check the infrastructure for terraform.

this tfstate file will be there.if in case it is missing then excuse the command **terrafome refresh**.

It will create the tfstate file.

**Working**

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

1. terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "=2.46.0"

}

}

}

# Configure the Microsoft Azure Provider — this is to inform terraform about working infrastructure provider

provider "azurerm" {

features {}

}

# Create a resource group

resource "azurerm\_resource\_group" "example" {

name = "example-resources"

location = "West Europe"

}

**Variables**

2 types of variables

1. Variable.tf — in order to change parameter name we need to declare in variable.tf

variable storage name{

default= “storage1”

}

2) Just.tfvar—- in order to change values in variable.tf file

storagename = “storage2”

**i)input variables**

variable”name”{

type= string

default=[“south india”]

}

data types are

string

number

bool

list

map

**how to assign value to variable**

**output variables**

**local values**

**usage of variables**

* currently we are creating terraforms inn local and pushing to repos. by gitlab/azure repos/git bucket. will use this terraforms to create infrastructure or to build CI CD pipelines.

**Terraform Cloud**

**1)terraform pre requisites**

**2)provider - AWS/Azure**

**3)how to create resources**

**4)how to push changes on repos**

**5)CI/CD pipelines**

**6)terraform cloud**

**7)terrform backend - state file will be there**

**8)work space - where you can manage enivorment**

**9)how to integrate with azure devops/jenkins**

1. **Flexi basket**
2. **laptop raise**
3. **i success**
4. **induction document**
5. **internal chat box**