|  |  |
| --- | --- |
| Overview | 1 |
| Pre-Requisites | 2 |
| Terraform Cloud Nomenclature | 3 |
| Azure Repositories | 4 |
| Configuration Settings | 5 |
| Publish Modules | 6 |
| Workspace | 8 |
| Declare Variables | 9 |
| Workspace settings | 10 |
| Policy Sets | 11 |
| References | 12 |

CONTENT

**OVERVIEW**

This document describes Infrastructure as a Code (IaaC) to build resources in Azure using Terraform. To maintain consistency with terraform state among the team, Advantasure is using Terraform CLI to publishing run and deploy resources.

From TFC, we establish a connection with Version Control System (VCS) provider to pull the code using a webhooks upon authorization. In our case, an Azure DevOps service is our VCS where we create repository as per TFC nomenclature to publish modules or policies with Tags. We can create any number of VCS connections with the different/same provider.

This POC is designed for sample architecture to created module and re using modules and deploying to Azure cloud.

PRE-REQUISITES

1. Ensure you have Credentials to be used in Terraform modules Azure Devops services
2. Ensure sufficient access on Organization/ Project to manage repositories in Azure DevOps services.
   1. This to minimum files you needed to deploy VMS <https://dev.azure.com/AdvantasureExecution/Azure-Terraform-Build>
   2. If you wanted to download modules to your local system <https://dev.azure.com/AdvantasureExecution/Azure-Terraform-Module>
3. Install [Terraform CLI] (https://www.terraform.io/downloads.html)
4. Install [Azure CLI] (<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>)

Note :-

- Terraform - Authenticating using the Azure CLI` is going to be same for WindowsOS too.

# Azure CLI Login

az login

# List Subscriptions

az account list

# Set Specific Subscription (if we have multiple subscriptions)

az account set --subscription="SUBSCRIPTION\_ID"

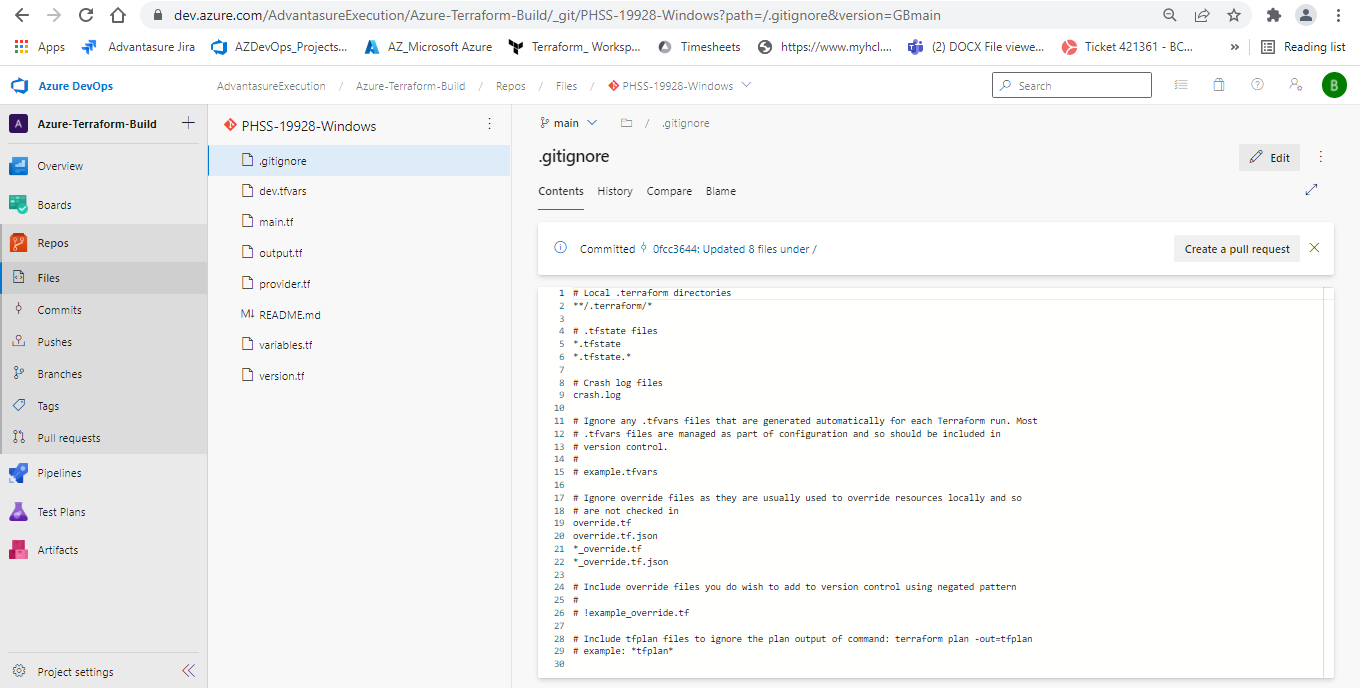
1. Install [VS Code Editor] (https://code.visualstudio.com/download)
2. Install [HashiCorp Terraform plugin for VS Code] (https://marketplace.visualstudio.com/items?itemName=HashiCorp.terraform)
3. Install [AZ Git Client]( https://dev.azure.com/AdvantasureExecution/Azure-Terraform-Module)

### Add VCS Provider:

How to Use Overview

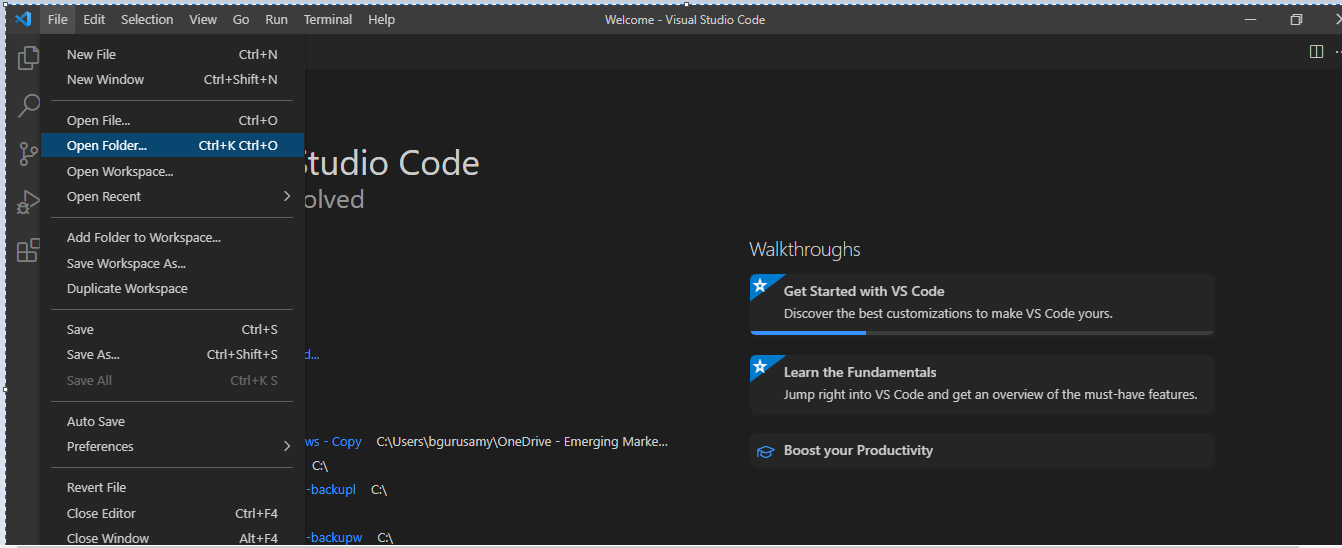
Select the appropriate BUILD FILE required for your infrastructure deployment and download it to your local desktop from Azure Repositories Artifact.

<https://dev.azure.com/AdvantasureExecution/Azure-Terraform-Build>



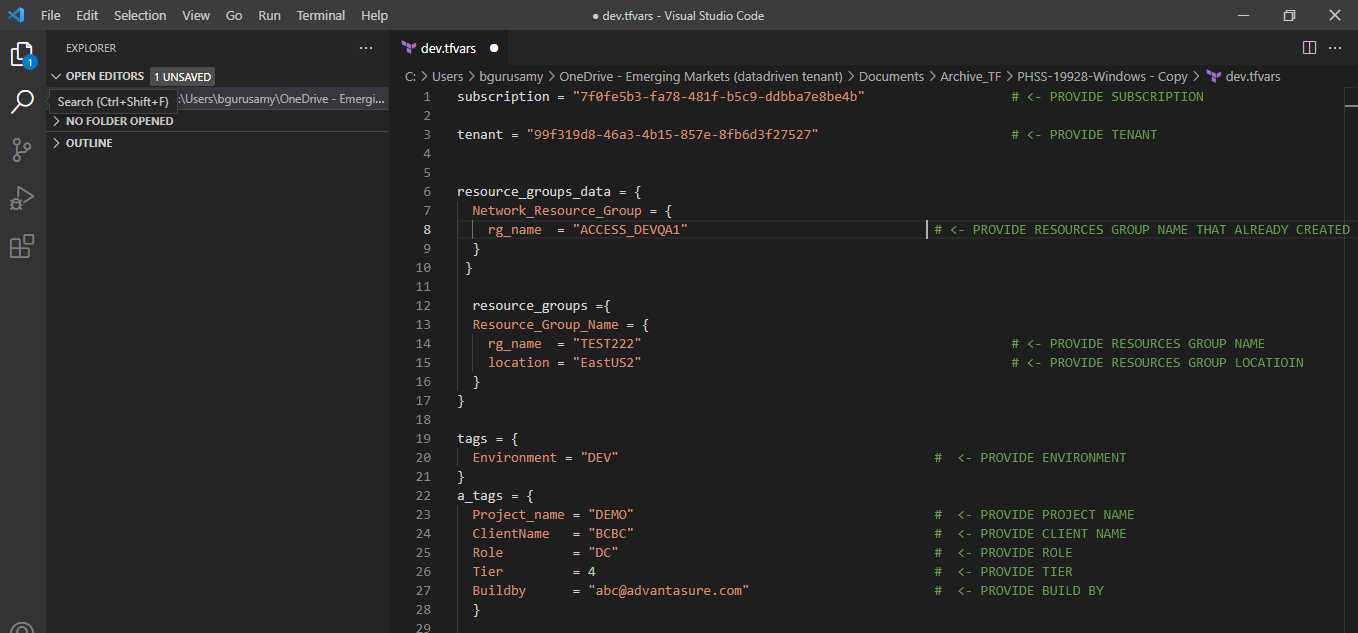
Customize the repo files

Open Visual Studio Code, and select file,and Open the Folder Option as shown below. The first step to go over the .TFVAR file.



Provide information mention in GREEN the resources that you wanted to build in Azure portal with naming convention.

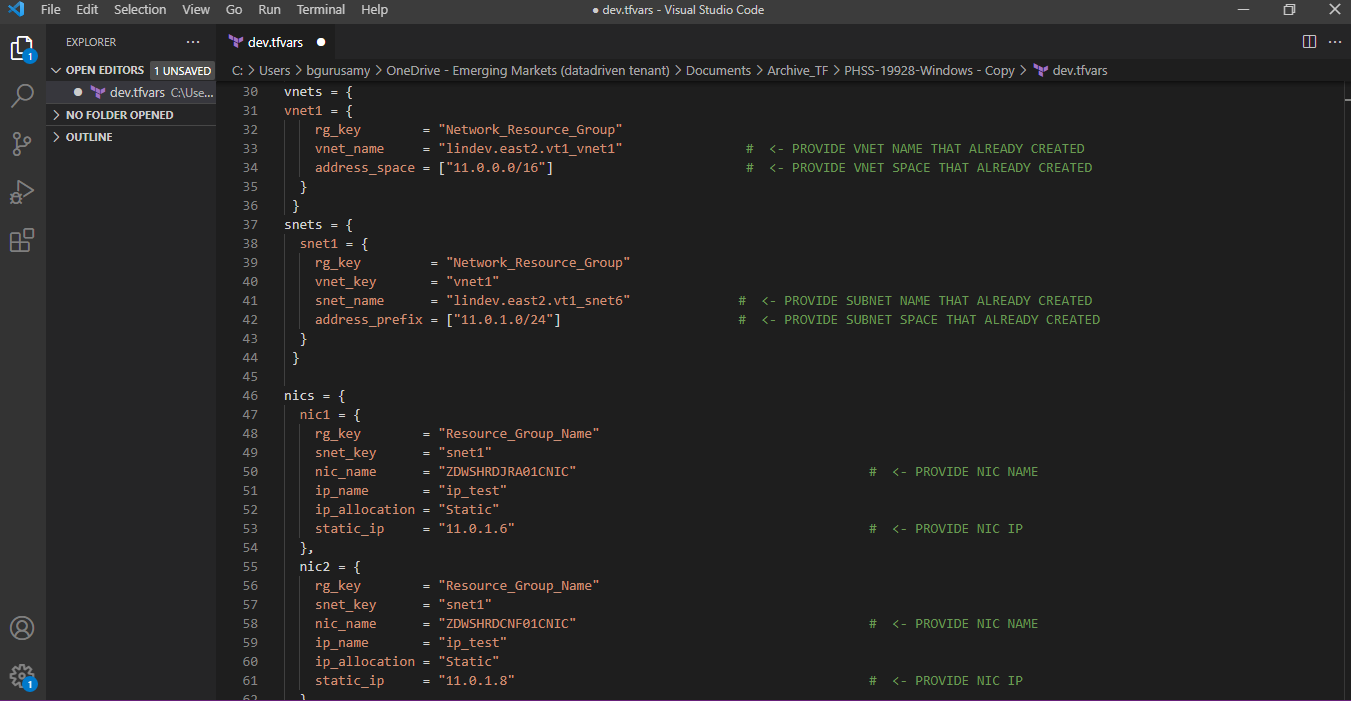
1. Please provide following details.
2. Subscription ID
3. Tenant ID
4. Resource Groups
5. rg\_name = "\*\*\*\*\*\*\*\*\*\*" # <- PROVIDE RESOURCES GROUP NAME WHICH IS ALREADY CREATED
6. rg\_name = “\*\*\*\*\*\*\*\*\*” # <- PROVIDE RESOURCES GROUP NAME -NEW RESOURCES TO BE CREATE.
7. Location = “\*\*\*\*\*\*\*\*” # <- PROVIDE RESOURCES GROUP LOCATIOIN
8. Tags
9. Environment =”\*\*\*” # <- PROVIDE ENVIRONMENT -NEW RESOURCES TO BE CREATE.
10. a\_tags = {
11. Project\_name = "\*\*\*\*" # <- PROVIDE PROJECT NAME
12. ClientName = "\*\*\*\*" # <- PROVIDE CLIENT NAME
13. Role = "\*\*" # <- PROVIDE ROLE
14. Tier = \* # <- PROVIDE TIER
15. Buildby = "\*\*\*\*\*\*\*\*\*" # <- PROVIDE BUILD BY



1. Vnets
2. Vnet name = “\*\*\*\*\*” # <- PROVIDE VNET NAME WHICH IS ALREADY CREATED
3. Address space = “\*\*\*\*\*” # <- PROVIDE VNET SPACE WHICH IS ALREADY CREATED
4. Snets
5. snet\_name = "\*\*\*\*\*\*\*\*\*\*\*" # <- PROVIDE SUBNET NAME WHICH IS ALREADY CREATED
6. address\_prefix = ["\*\*\*\*\*\*"] # <- PROVIDE SUBNET SPACE WHICH IS ALREADY CREATED
7. nics
8. nic name = "\*\*\*\*\*\*\*\*\*\*\*" # <- PROVIDE NIC NAME TO BE 77CREATED
9. static ip = "\*\*\*\*\*\*\*" # <- PROVIDE NIC IP TO VM TO BE CREATED.

IF YOU NEED TO CREATE MULTIPLE VMS YOU HAVE TO PROVIDE AS SHOWN BELOW

1. nic2
2. nic name = "\*\*\*\*\*\*\*\*\*\*\*" # <- PROVIDE NIC NAME TO BE 77CREATED
3. static ip = "\*\*\*\*\*\*\*” # <- PROVIDE NIC IP TO VM TO BE CREATED



1. windows vms
2. win1

rg\_key = "Resource\_Group\_Name"

vm\_name = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM\_NAME

vmSize = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM SIZE

vmosname = "\*\*\*\*\*\*\*\*” # <- PROVIDE VM OS NAME

vmCache = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM CACHE

vmDisk = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM DISK

osPublisher = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM OS PUBLISHER

osOffer = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM OS OFFER

osSKU = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM OS SKU

osVersion = "\*\*\*\*\*\*\*\*" # <- PROVIDE VM OS VERSION

os\_disk\_gb = \*\*\* # <- PROVIDE VM OS DISK GB

nic\_key = "nic1"

aset\_key = "aset1"

sg\_key = "sg1"

rsv\_key = "rsv1"

bkp\_key = "bkp1"

},

IF YOU NEED TO CREATE MULTIPLE VMS YOU HAVE TO PROVIDE AS SHOWN BELOW

1. win2

rg\_key = "Resource\_Group\_Name"

vm\_name = "ZDWSHRDJRA01C1" # <- PROVIDE VM\_NAME

vmSize = "Standard\_D16ds\_v4" # <- PROVIDE VM SIZE

vmosname = "ZDWSHRDJRA01C-O # <- PROVIDE VM OS NAME

vmCache = "ReadWrite" # <- PROVIDE VM CACHE

vmDisk = "Standard\_LRS" # <- PROVIDE VM DISK

osPublisher = "MicrosoftWindowsServer" # <- PROVIDE VM OS PUBLISHER

osOffer = "WindowsServer" # <- PROVIDE VM OS OFFER

osSKU = "2019-Datacenter" # <- PROVIDE VM OS SKU

osVersion = "latest" # <- PROVIDE VM OS VERSION

os\_disk\_gb = 128 # <- PROVIDE VM OS DISK GB

nic\_key = "nic1"

aset\_key = "aset1"

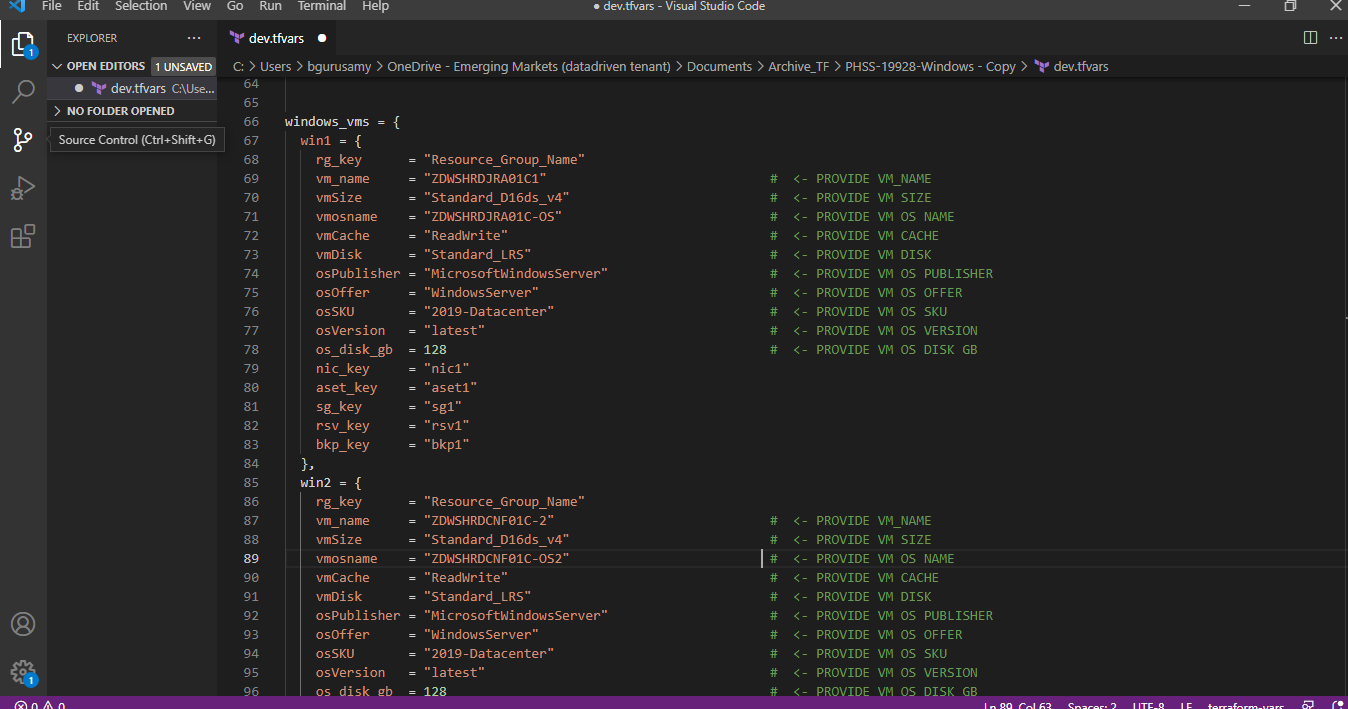
sg\_key = "sg1"

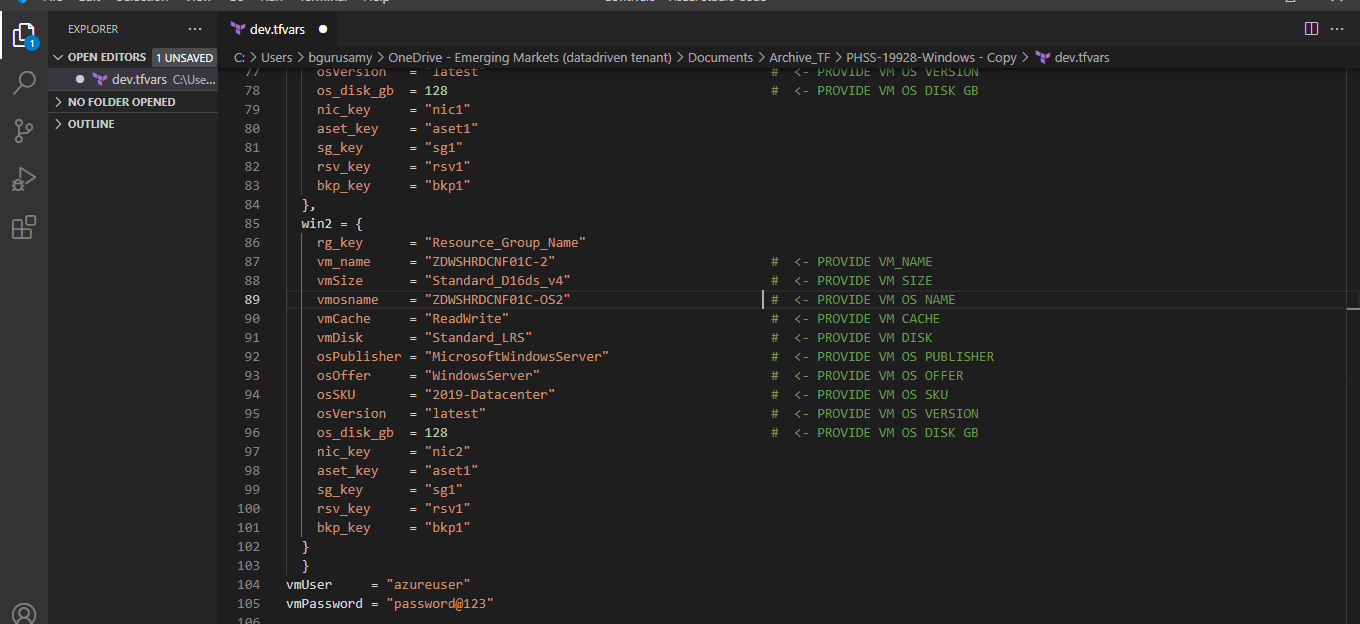
rsv\_key = "rsv1"

bkp\_key = "bkp1"

}

1. vmUser = "\*\*\*\*\*\*\*" # <- PROVIDE VM USERNAME
2. vmPassword = "\*\*\*\*\*\*\*" # <- PROVIDE VM PASSWORD





1. mdisks
2. mdisk1 = {

rg\_key = "Resource\_Group\_Name"

vm\_key = "win1"

name = "\*\*\*\*\*\*\*\*" # <- PROVIDE MANAGE DISK NAME

lun = 0 # <- PROVIDE MANAGE DISK LUN

caching = "\*\*\*\*\*\*\*\*" # <- PROVIDE MANAGE DISK CACHING

storage\_account\_type = "\*\*\*\*\*\*\*\*" # <- PROVIDE MANAGE DISK STORAGE ACCOUNT TYPE

create\_option = "Empty"

disk\_size\_gb = \*\*\* # <- PROVIDE MANAGE DISK SIZE

},

IF YOU NEED TO CREATE MULTIPLE MANAGE DISK YOU HAVE TO PROVIDE AS SHOWN BELOW

1. mdisk1 = {

rg\_key = "Resource\_Group\_Name"

vm\_key = "win1"

name = "ZDWSHRDJRA01C1-M" # <- PROVIDE MANAGE DISK NAME

lun = 0 # <- PROVIDE MANAGE DISK LUN

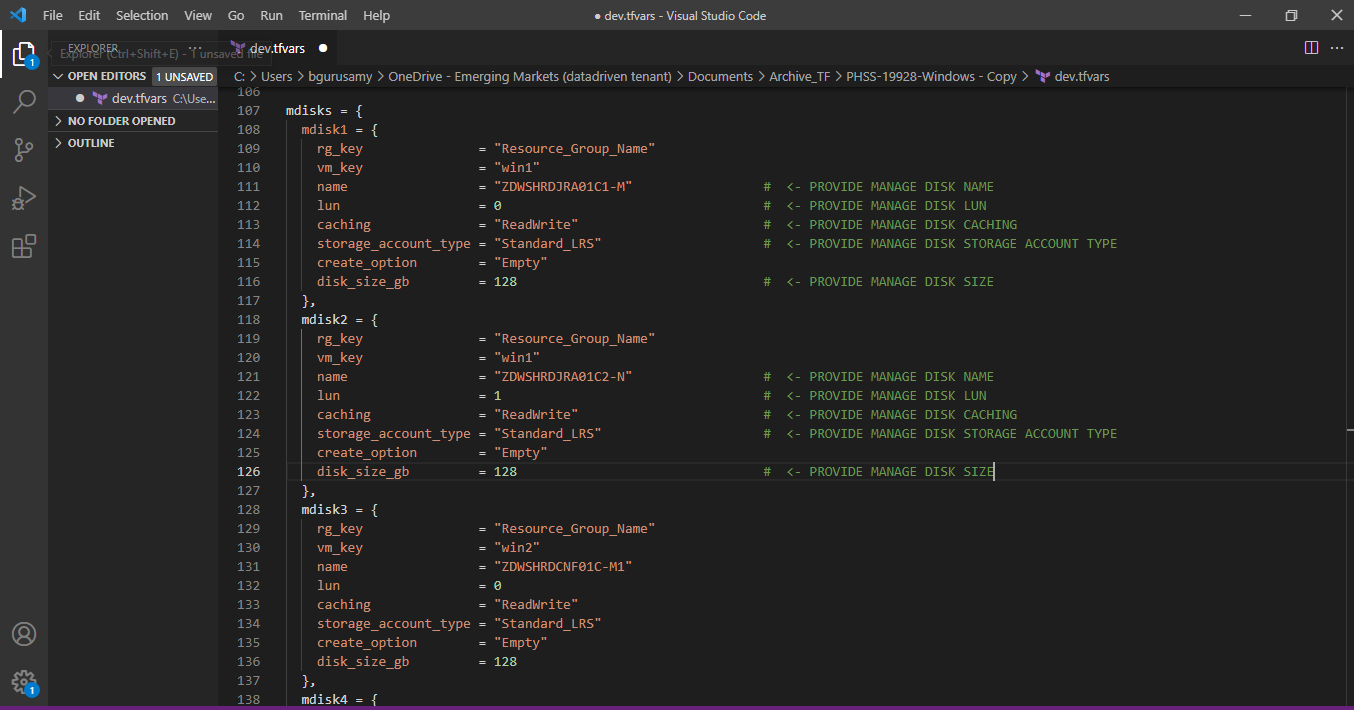
caching = "ReadWrite" # <- PROVIDE MANAGE DISK CACHING

storage\_account\_type = "Standard\_LRS" # <- PROVIDE MANAGE DISK STORAGE ACCOUNT TYPE

create\_option = "Empty"

disk\_size\_gb = 128 # <- PROVIDE MANAGE DISK SIZE

}



1. extensionsvm
2. ext1= {

vm\_key = "win1"

publisher = "Microsoft.Compute" # <- PROVIDE EXTENSION PUBLISHER

type = "BGInfo" # <- PROVIDE EXTENSION TYPE

version = "2.1" # <- PROVIDE EXTENSION VERSION

settings = null

},

1. ext2= {

vm\_key = "win1"

publisher = “ Microsoft.Azure.Monitoring.DependencyAgent" # <- PROVIDE EXTENSION PUBLISHER

type = " DependencyAgentWindows" # <- PROVIDE EXTENSION TYPE

version = " 9.8" # <- PROVIDE EXTENSION VERSION

settings = null

},

1. ext3= {

vm\_key = "win1"

publisher = "Microsoft.Compute" # <- PROVIDE EXTENSION PUBLISHER

type = "BGInfo" # <- PROVIDE EXTENSION TYPE

version = "2.1" # <- PROVIDE EXTENSION VERSION

settings = null

},

1. Ext4= {

vm\_key = "win2"

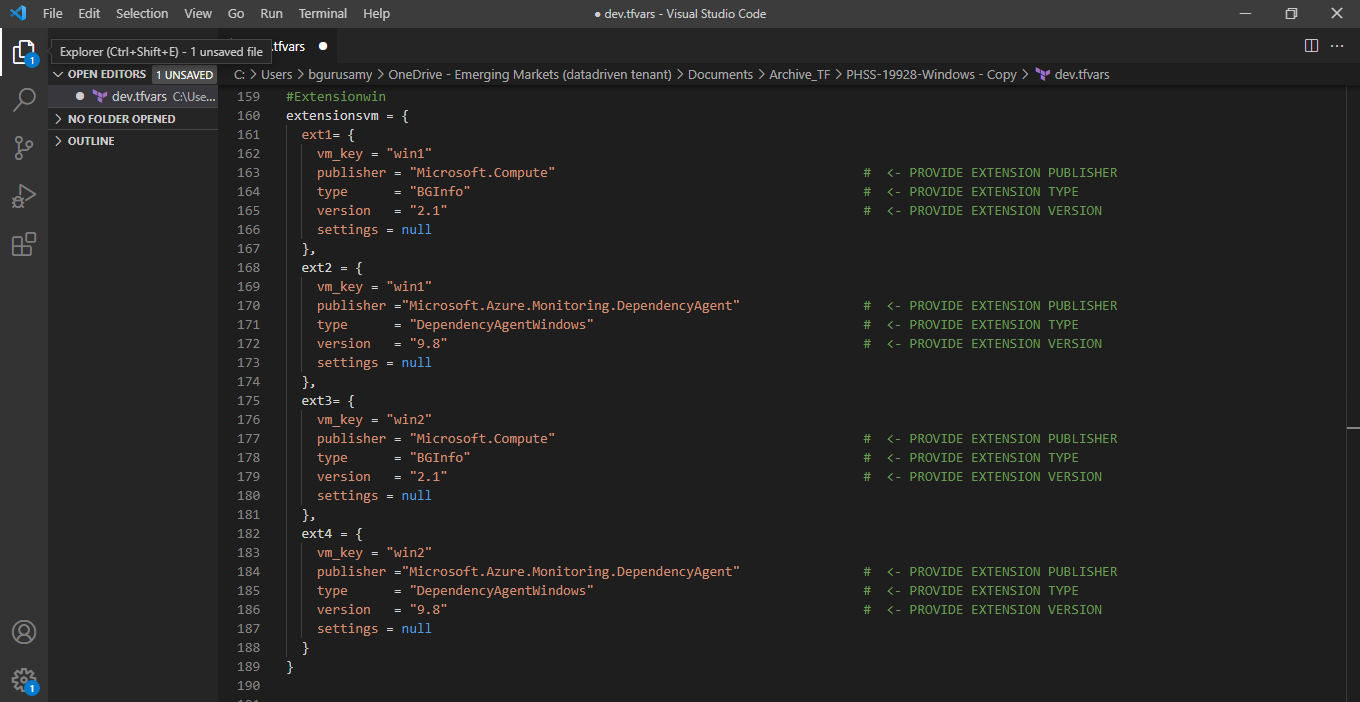
publisher = " Microsoft.Azure.Monitoring.DependencyAgent" # <- PROVIDE EXTENSION PUBLISHER

type = " DependencyAgentWindows " # <- PROVIDE EXTENSION TYPE

version = " 9.8" # <- PROVIDE EXTENSION VERSION

settings = null

}



1. storage\_account
2. sg1 = {

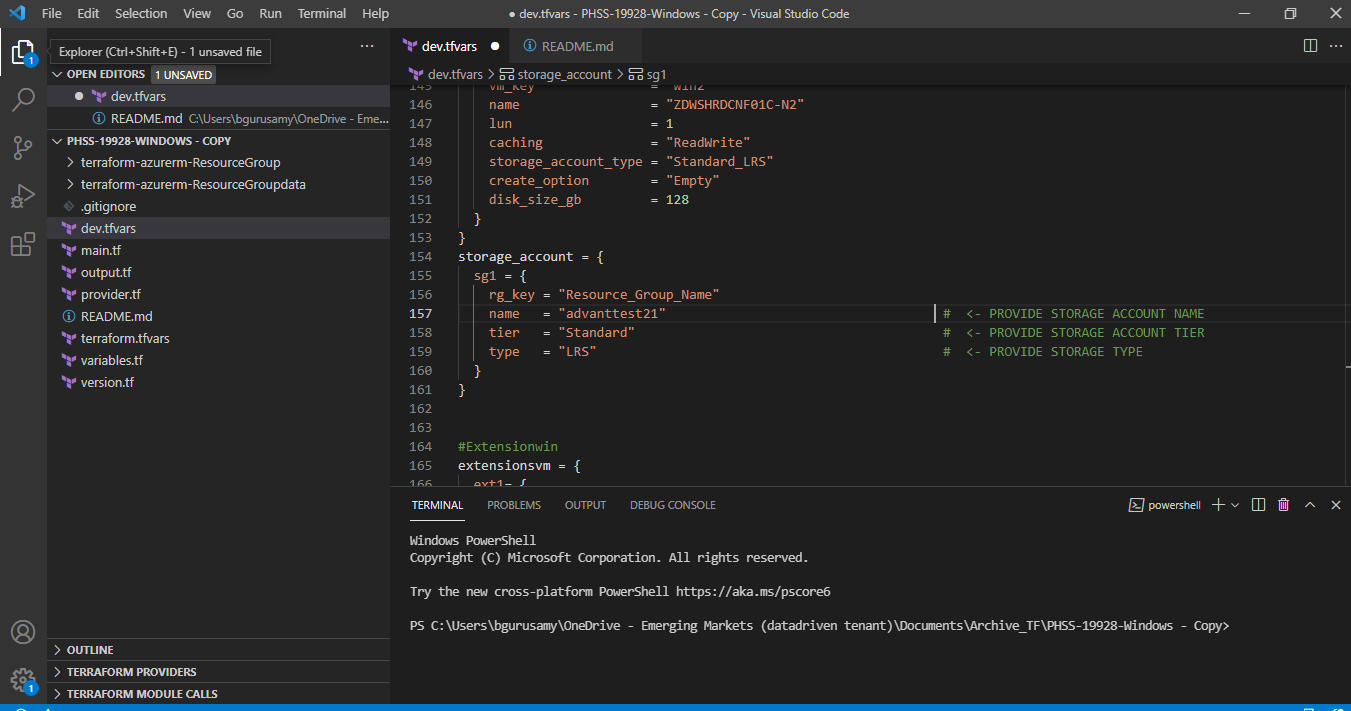
rg\_key = "Resource\_Group\_Name"

name = "advanttest21" # <- PROVIDE STORAGE ACCOUNT NAME

tier = "Standard" # <- PROVIDE STORAGE ACCOUNT TIER

type = "LRS" # <- PROVIDE STORAGE TYPE

}



Create and Deploy Resources.

Once all the changes are correctly done your local TFVAR FILE, the next step is to deploy the infrastructure resources to the Azure portal described command as follow.

title: Terraform Command Basics

description: Learn Terraform Commands like init, validate, plan, apply and destroy

## Step-01: Introduction

- Understand basic Terraform Commands

1. terraform init

Used to Initialize a working directory containing terraform config files. This is the first command that should be run after writing a new Terraform configuration Downloads Providers.

2. terraform validate

Validates the terraform configurations files in that respective directory to ensure they are syntactically valid and internally consistent.

3. terraform plan - terraform plan -var-file="dev.tfvars" -state="dev.tfstate"

Creates an execution plan - Terraform performs a refresh and determines what actions are necessary to achieve the desired state specified in configuration files and provide details resources to be created.

Used to apply the changes required to reach the desired state of the configuration.

4. terraform apply - terraform apply -var-file="dev.tfvars" -state="dev.tfstate" -auto-approve

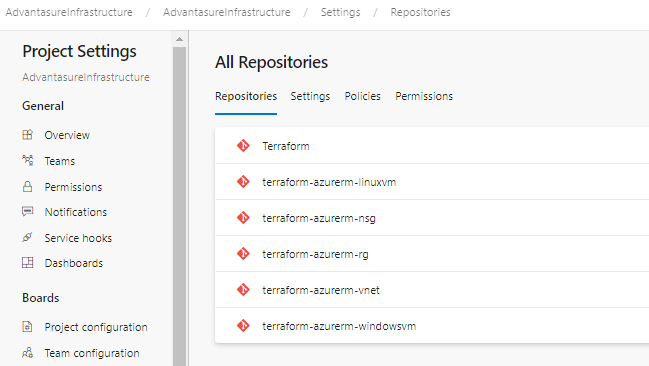
By default, apply scans the current directory for the configuration and applies the changes appropriately.

5. terraform destroy - terraform destroy -var-file="dev.tfvars" -state="dev.tfstate" -auto-approve

Used to destroy the Terraform-managed infrastructure.

AZURE REPOSITORIES

To Publish Modules In TFC, Repository name should be "terraform-<provider>-<name>". A new repository is required for each module.



Also, each repository must have tags to publish as module and should be like below.

