[Software & Platform Required: 1](#_Toc1433155968)

[Installation of Terraform 1](#_Toc305705214)

[Installation of V S Code 2](#_Toc839876579)

[Installation of Git 2](#_Toc191194980)

[Connect V S Code with Azure DevOps 3](#_Toc952095713)

[Install Terraform Extension on Azure DevOps 7](#_Toc994136097)

[If your Parallelism job is not working, then follow this setup. 8](#_Toc227515155)

[Steps to Run CI Pipeline. 9](#_Toc1323512756)

[1. Setup your Repos for Single VM creation. 9](#_Toc42664483)

[Code for main.tf (Single VM creation without variables.tf file). 10](#_Toc1992368746)

[Code for main.tf (Single VM creation with variables.tf file). 12](#_Toc757962102)

[Code for provider.tf 14](#_Toc2147074109)

[Code for variables.tf (Single VM file). 15](#_Toc1801790106)

[2. Setup your Repos for Multi VM creation (1st approach). 19](#_Toc887901094)

[Code for variables.tf (Multi VM 1st approach file). 19](#_Toc851566197)

[Code for main.tf (Multi VM 1st approach file). 20](#_Toc991372690)

[3. Setup your Repos for Multi VM creation (2nd approach). 22](#_Toc2136214410)

[Code for variables.tf (Multi VM 2nd approach file). 22](#_Toc581231617)

[Code for main.tf (Multi VM 2nd approach file). 25](#_Toc641532987)

[4. Steps to Setup your Pipelines to run CI. 28](#_Toc1549710430)

[NOTE: Change environmentServiceNameAzureRM with your subscription 31](#_Toc542962897)

[Steps to run CD pipeline. 35](#_Toc280556704)

Steps to initialize Azure DevOps CI/CD Pipeline using Terraform.

# Software & Platform Required:

1. Terraform
2. Visual Studio Code
3. Git bash

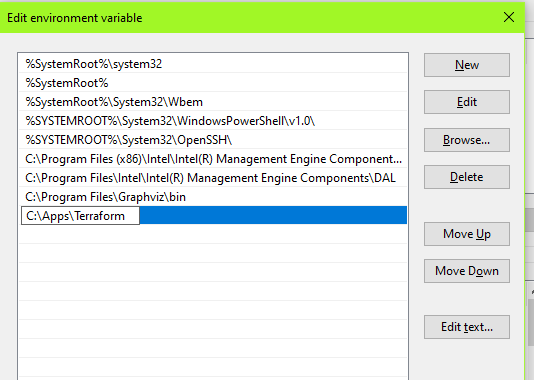
## Installation of Terraform

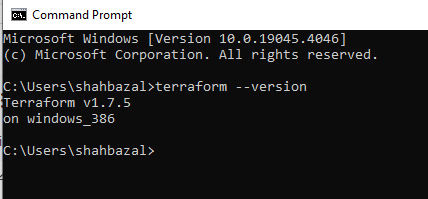
1. **Download the terraform from below link**

[Install | Terraform | HashiCorp Developer](https://developer.hashicorp.com/terraform/install)

1. **After downloading the terraform follow the below steps:**

* Download the terraform ZIP file from Terraform site.
* Extract the .exe from the ZIP file to a folder eg C:\Apps\Terraform copy this path location like C:\Apps\terraform\
* Add the folder location to your PATH variable, eg: Control Panel -> System -> System settings -> Environment Variables
* In System Variables, select Path > edit > new > Enter the location of the Terraform .exe, eg C:\Apps\Terraform then click OK

  
After completion of all the above steps, we can verify the installation.



## Installation of V S Code

1. **Download the V S Code from below link**

[Download Visual Studio Code - Mac, Linux, Windows](https://code.visualstudio.com/download)

## Installation of Git

1. **Download the Git from below link**

[Git - Downloads (git-scm.com)](https://git-scm.com/downloads)

After that Run the below command to set your Email and User Name:

1. $ git config --global user.name "John Doe"
2. $ git config --global user.email johndoe@example.com

For additional info you can check the below link:

[Git - First-Time Git Setup (git-scm.com)](https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup)

Now check your account with Visual Studio Code, ensure that you login the same account which you will use on your Azure DevOps portal.

# Connect V S Code with Azure DevOps

After completion of Setup follow the below steps:

1. Sign in Azure DevOps.
   * Use the below link for login/signup in Azure DevOps Portal

[Azure DevOps Services | Microsoft Azure](https://azure.microsoft.com/en-in/products/devops/)

1. Create Organization.
2. Create Project.
3. Click on Repos > files

5. Click on Initialize.

6. Click on Clone.

7. Click Clone in VS Code.

8. Select D drive (or any) as cloned repository.

9. Add all your required files for pipeline.

10. Go to Source Control in Visual Studio Code and give Some meaning full commit message.

11. Commit all changes every time after any edit or add and click on commit.

12. After commit always remember to do Sync.

13. On clicking on main as show in picture you can sync your Repos and local changes.

## Install Terraform Extension on Azure DevOps

**Steps:**

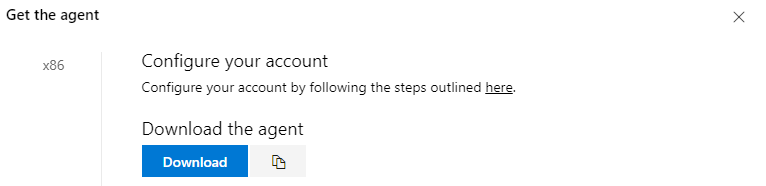
1. Go to Organization Settings.
2. Under the General section click Extensions.
3. Click Browse marketplace.
4. Search Terraform and install it.
5. Verify that in the installed extension section.

# If your Parallelism job is not working, then follow this setup.

**Additional Tool Setup (Azure Local Agent) to use your machine as server.**

Organization > Project > Settings > Agent pools > Agents > New Agent >

Download the Agent



After downloading the agent follow the below steps:

1. Create a Folder on C drive by Agent name.
2. Unzip the downloaded agent here.
3. Open the unzip directory on Command Prompt / PowerShell (Run as Administration).
4. Now Configure the agent by:
   1. .\config.cmd (run this command and allow the required permissions)
   2. .\run.cmd (to active the agent).
5. You can check your Agent under Agents section.

# Steps to Run CI Pipeline.

When you Completed the All steps / setup now it's time to run CI and CD pipeline.

## 1. Setup your Repos for Single VM creation.

1. After cloning with VS Code sync your all-required files.
2. Create a Resource Group, Storage Account and Container through azure portal to use as a backend\_bucket.

### Code for main.tf (Single VM creation without variables.tf file).

resource "azurerm\_resource\_group" "rg1" {

name = "TestRg1"

location = "East Us"

}

resource "azurerm\_virtual\_network" "vnet1" {

name = "testvnet1"

address\_space = ["10.0.0.0/16"]

location = azurerm\_resource\_group.rg1.location

resource\_group\_name = azurerm\_resource\_group.rg1.name

}

resource "azurerm\_subnet" "subnet1" {

name = "TestSubNet1"

resource\_group\_name = azurerm\_resource\_group.rg1.name

virtual\_network\_name = azurerm\_virtual\_network.vnet1.name

address\_prefixes = ["10.0.2.0/24"]

}

resource "azurerm\_network\_interface" "nic1" {

name = "TestNic1"

location = azurerm\_resource\_group.rg1.location

resource\_group\_name = azurerm\_resource\_group.rg1.name

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet1.id

private\_ip\_address\_allocation = "Dynamic"

}

}

resource "azurerm\_windows\_virtual\_machine" "vm1" {

name = "TestVm1"

resource\_group\_name = azurerm\_resource\_group.rg1.name

location = azurerm\_resource\_group.rg1.location

size = "Standard\_F2"

admin\_username = "adminuser"

admin\_password = "P@$$w0rd1234!"

network\_interface\_ids = [

azurerm\_network\_interface.nic1.id,

]

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = "MicrosoftWindowsServer"

offer = "WindowsServer"

sku = "2016-Datacenter"

version = "latest"

}

}

### Code for main.tf (Single VM creation with variables.tf file).

# Declaring resources using variables

resource "azurerm\_resource\_group" "rg1" {

name = var.resource\_group\_name

location = var.resource\_group\_location

tags = var.tags

}

resource "azurerm\_virtual\_network" "vnet1" {

name = var.vnet\_name

address\_space = var.vnet\_address\_space

location = azurerm\_resource\_group.rg1.location

resource\_group\_name = azurerm\_resource\_group.rg1.name

tags = var.tags

}

resource "azurerm\_subnet" "subnet1" {

name = var.subnet\_name

resource\_group\_name = azurerm\_resource\_group.rg1.name

virtual\_network\_name = azurerm\_virtual\_network.vnet1.name

address\_prefixes = var.subnet\_address\_prefixes

}

resource "azurerm\_network\_interface" "nic1" {

name = var.nic\_name

location = azurerm\_resource\_group.rg1.location

resource\_group\_name = azurerm\_resource\_group.rg1.name

tags = var.tags

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet1.id

private\_ip\_address\_allocation = "Dynamic"

}

}

resource "azurerm\_windows\_virtual\_machine" "vm1" {

name = var.vm\_name

resource\_group\_name = azurerm\_resource\_group.rg1.name

location = azurerm\_resource\_group.rg1.location

size = var.vm\_size

admin\_username = var.admin\_username

admin\_password = var.admin\_password

network\_interface\_ids = [

azurerm\_network\_interface.nic1.id,

]

tags = var.tags

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = var.os\_publisher

offer = var.os\_offer

sku = var.os\_sku

version = var.os\_version

}

}

### Code for provider.tf

terraform {

required\_version = ">=1.0"

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~>3.0"

}

random = {

source = "hashicorp/random"

version = "~>3.0"

}

}

}

provider "azurerm" {

features {}

}

### Code for variables.tf (Single VM file).

variable "resource\_group\_name" {

description = "The name of the resource group in which the resources will be created"

default = "terraform-compute"

}

variable "resource\_group\_location" {

type = string

default = "East US"

description = "Location for all resources."

}

variable "vnet\_name" {

description = "The name of the virtual network"

default = "testvnet1"

}

variable "vnet\_address\_space" {

description = "The address space for the virtual network"

type = list(string)

default = ["10.0.0.0/16"]

}

variable "subnet\_name" {

description = "The name of the subnet"

default = "TestSubNet1"

}

variable "subnet\_address\_prefixes" {

description = "The address prefixes for the subnet"

type = list(string)

default = ["10.0.2.0/24"]

}

variable "nic\_name" {

description = "The name of the network interface"

default = "TestNic1"

}

variable "vm\_name" {

description = "The name of the virtual machine"

default = "TestVm1"

}

variable "vm\_size" {

description = "The size of the virtual machine"

default = "Standard\_F2"

}

variable "admin\_username" {

description = "The admin username for the virtual machine"

default = "adminuser"

}

variable "admin\_password" {

description = "The admin password for the virtual machine"

default = "P@$$w0rd1234!"

}

variable "os\_publisher" {

description = "The publisher for the OS image"

default = "MicrosoftWindowsServer"

}

variable "os\_offer" {

description = "The offer for the OS image"

default = "WindowsServer"

}

variable "os\_sku" {

description = "The SKU for the OS image"

default = "2016-Datacenter"

}

variable "os\_version" {

description = "The version of the OS image"

default = "latest"

}

variable "tags" {

description = "A map of the tags to use on the resources."

type = map(string)

default = {

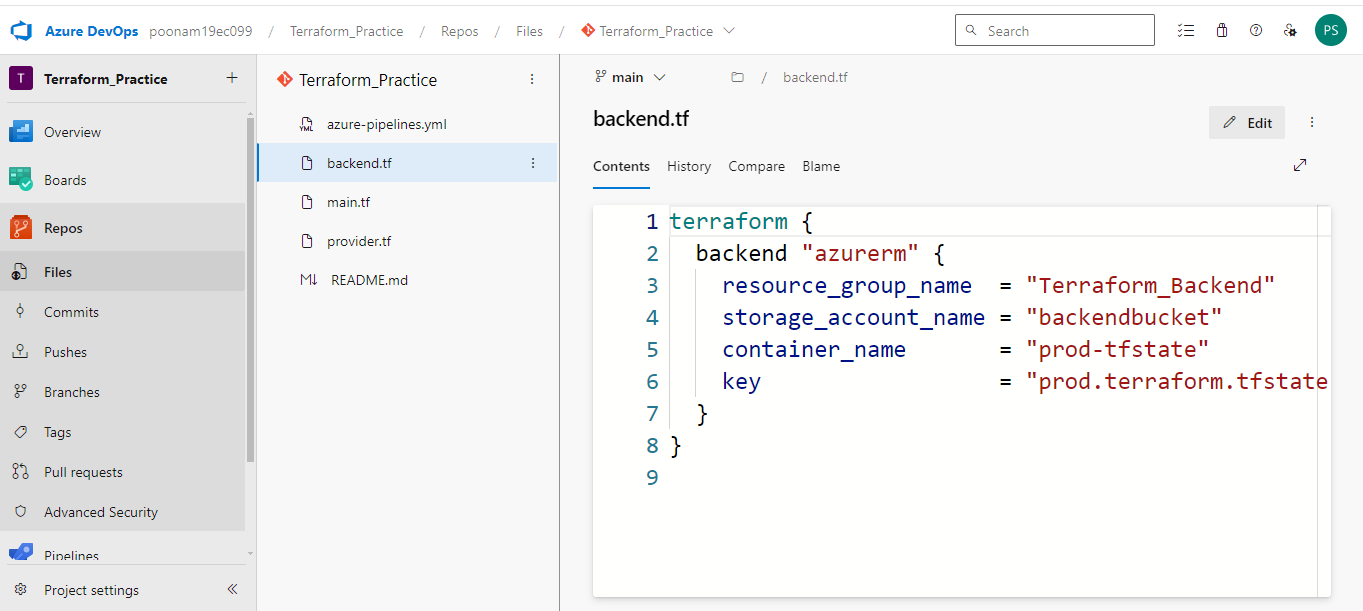
Environment = "Developer"

Department = "Terraform"

}

}

1. Add one more file and named as backend.tf, and use below code as a reference.



terraform {

backend "azurerm" {

resource\_group\_name = "Terraform\_Backend"

storage\_account\_name = "backendbucket"

container\_name = "prod-tfstate"

key = "prod.terraform.tfstate"

}

}

Follow the below link to know about azurerm backend

<https://developer.hashicorp.com/terraform/language/settings/backends/azurerm>

## 2. Setup your Repos for Multi VM creation (1st approach).

### Code for variables.tf (Multi VM 1st approach file).

variable "resource\_group\_name" {

description = "The name of the resource group in which the resources will be created"

default = "terraform-compute"

}

variable "resource\_group\_location" {

type = string

default = "East US"

description = "Location for all resources."

}

variable "vnet\_name" {

description = "The name of the virtual network"

default = "testvnet1"

}

variable "vnet\_address\_space" {

description = "The address space for the virtual network"

type = list(string)

default = ["10.0.0.0/16"]

}

variable "subnet\_name" {

description = "The name of the subnet"

default = "TestSubNet1"

}

variable "subnet\_address\_prefixes" {

description = "The address prefixes for the subnet"

type = list(string)

default = ["10.0.2.0/24"]

}

variable "vm\_size" {

description = "The size of the virtual machine"

default = "Standard\_F2"

}

variable "admin\_username" {

description = "The admin username for the virtual machine"

default = "adminuser"

}

variable "admin\_password" {

description = "The admin password for the virtual machine"

default = "P@$$w0rd1234!"

}

variable "os\_publisher" {

description = "The publisher for the OS image"

default = "MicrosoftWindowsServer"

}

variable "os\_offer" {

description = "The offer for the OS image"

default = "WindowsServer"

}

variable "os\_sku" {

description = "The SKU for the OS image"

default = "2016-Datacenter"

}

variable "os\_version" {

description = "The version of the OS image"

default = "latest"

}

variable "tags" {

description = "A map of the tags to use on the resources."

type = map(string)

default = {

Environment = "Developer"

Department = "Terraform"

}

}

### Code for main.tf (Multi VM 1st approach file).

# Define the Azure resource group

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

name = var.resource\_group\_name

location = var.resource\_group\_location

}

# Define the Azure virtual network

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

name = var.vnet\_name

address\_space = var.vnet\_address\_space

location = azurerm\_resource\_group.rg.location

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

}

# Define the subnet within the virtual network

resource "azurerm\_subnet" "subnet" {

name = var.subnet\_name

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

virtual\_network\_name = azurerm\_virtual\_network.vnet.name

address\_prefixes = var.subnet\_address\_prefixes

}

# Define the first Azure network interface

resource "azurerm\_network\_interface" "ni1" {

name = "Test-Nic-1"

location = azurerm\_resource\_group.rg.location

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

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet.id

private\_ip\_address\_allocation = "Dynamic"

}

}

# Define the second Azure network interface

resource "azurerm\_network\_interface" "ni2" {

name = "Test-Nic-2"

location = azurerm\_resource\_group.rg.location

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

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet.id

private\_ip\_address\_allocation = "Dynamic"

}

}

# Define the first Azure virtual machine

resource "azurerm\_windows\_virtual\_machine" "vm1" {

name = "TestVm-1"

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

location = azurerm\_resource\_group.rg.location

size = var.vm\_size

admin\_username = var.admin\_username

admin\_password = var.admin\_password

network\_interface\_ids = [azurerm\_network\_interface.ni1.id]

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = var.os\_publisher

offer = var.os\_offer

sku = var.os\_sku

version = var.os\_version

}

}

# Define the second Azure virtual machine

resource "azurerm\_windows\_virtual\_machine" "vm2" {

name = "TestVm-2"

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

location = azurerm\_resource\_group.rg.location

size = var.vm\_size

admin\_username = var.admin\_username

admin\_password = var.admin\_password

network\_interface\_ids = [azurerm\_network\_interface.ni2.id]

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = var.os\_publisher

offer = var.os\_offer

sku = var.os\_sku

version = var.os\_version

}

}

Provider.tf and backend.tf file will remain same as Single VM.

## 3. Setup your Repos for Multi VM creation (2nd approach).

### Code for variables.tf (Multi VM 2nd approach file).

# Variables  
 variable "resource\_group\_name" {  
 description = "The name of the resource group in which the resources will be created"  
 default = "terraform-Test1"  
 }

variable "resource\_group\_location" {  
 type = string  
 default = "East US"  
 description = "Location for all resources."  
 }

variable "vnet\_name" {  
 description = "The name of the virtual network"  
 default = "testvnet01"  
 }

variable "vnet\_address\_space" {  
 description = "The address space for the virtual network"  
 type = list(string)  
 default = ["10.0.0.0/16"]  
 }

variable "subnet\_name" {  
 description = "The name of the subnet"  
 default = "TestSubNet01"  
 }

variable "subnet\_address\_prefixes" {  
 description = "The address prefixes for the subnet"  
 type = list(string)  
 default = ["10.0.2.0/24"]  
 }

variable "nic\_name" {  
 description = "The name of the network interface"  
 type = list(string)  
 default = ["TestNic01", "TestNic02"]  
 }

variable "vm\_names" {  
 description = "The names of the virtual machines"  
 type = list(string)  
 default = ["TestVm01", "TestVm02"]  
 }

variable "vm\_size" {  
 description = "The size of the virtual machine"  
 default = "Standard\_F2"  
 }

variable "admin\_username" {  
 description = "The admin username for the virtual machine"  
 default = "adminuser"  
 }

variable "admin\_password" {  
 description = "The admin password for the virtual machine"  
 default = "P@$$w0rd1234!"  
 }

variable "os\_publisher" {  
 description = "The publisher for the OS image"  
 default = "MicrosoftWindowsServer"  
 }

variable "os\_offer" {  
 description = "The offer for the OS image"  
 default = "WindowsServer"  
 }

variable "os\_sku" {  
 description = "The SKU for the OS image"  
 default = "2016-Datacenter"  
 }

variable "os\_version" {  
 description = "The version of the OS image"  
 default = "latest"  
 }

variable "tags" {  
 description = "A map of the tags to use on the resources."  
 type = map(string)  
 default = {  
 Environment = "Developer"  
 Department = "Terraform"  
 }  
 }

### Code for main.tf (Multi VM 2nd approach file).

# Create resource group  
 resource "azurerm\_resource\_group" "rg" {  
 name = "terraform-rg"  
 location = var.resource\_group\_location  
 }

# Create virtual network  
 resource "azurerm\_virtual\_network" "vnet" {  
 name = var.vnet\_name  
 address\_space = var.vnet\_address\_space  
 location = azurerm\_resource\_group.rg.location  
 resource\_group\_name = azurerm\_resource\_group.rg.name  
 }

# Create subnet  
 resource "azurerm\_subnet" "subnet" {  
 name = var.subnet\_name  
 resource\_group\_name = azurerm\_resource\_group.rg.name  
 virtual\_network\_name = azurerm\_virtual\_network.vnet.name  
 address\_prefixes = var.subnet\_address\_prefixes  
 }

# Create network interfaces  
 resource "azurerm\_network\_interface" "nic" {  
 count = length(var.vm\_names)  
 name = var.nic\_names[count.index]  
 location = azurerm\_resource\_group.rg.location  
 resource\_group\_name = azurerm\_resource\_group.rg.name

ip\_configuration {  
 name = "ipconfig-${count.index}"  
 subnet\_id = azurerm\_subnet.subnet.id  
 private\_ip\_address\_allocation = "Dynamic"  
 }  
 }

# Create virtual machines  
 resource "azurerm\_virtual\_machine" "vm" {  
 count = length(var.vm\_names)  
 name = var.vm\_names[count.index]  
 location = azurerm\_resource\_group.rg.location  
 resource\_group\_name = azurerm\_resource\_group.rg.name  
 network\_interface\_ids = [azurerm\_network\_interface.nic[count.index].id]  
 vm\_size = var.vm\_size

storage\_os\_disk {  
 name = "${var.vm\_names[count.index]}-osdisk"  
 caching = "ReadWrite"  
 create\_option = "FromImage"  
 managed\_disk\_type = "Standard\_LRS"  
 }

os\_profile {  
 computer\_name = var.vm\_names[count.index]  
 admin\_username = var.admin\_username  
 admin\_password = var.admin\_password  
 }

os\_profile\_windows\_config {  
 enable\_automatic\_upgrades = true  
 provision\_vm\_agent = true  
 }

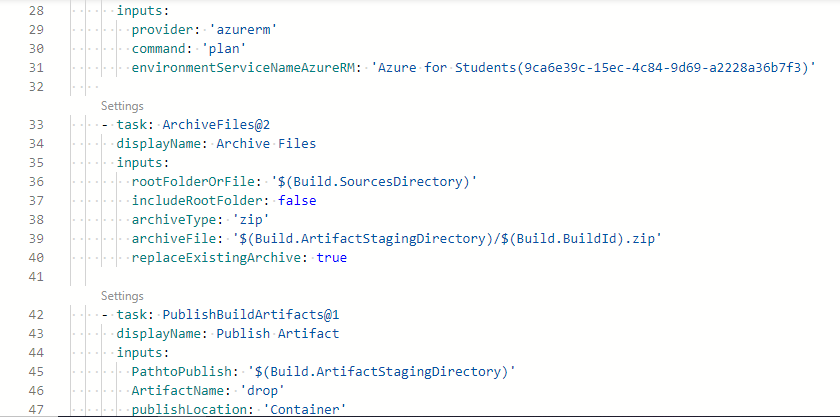
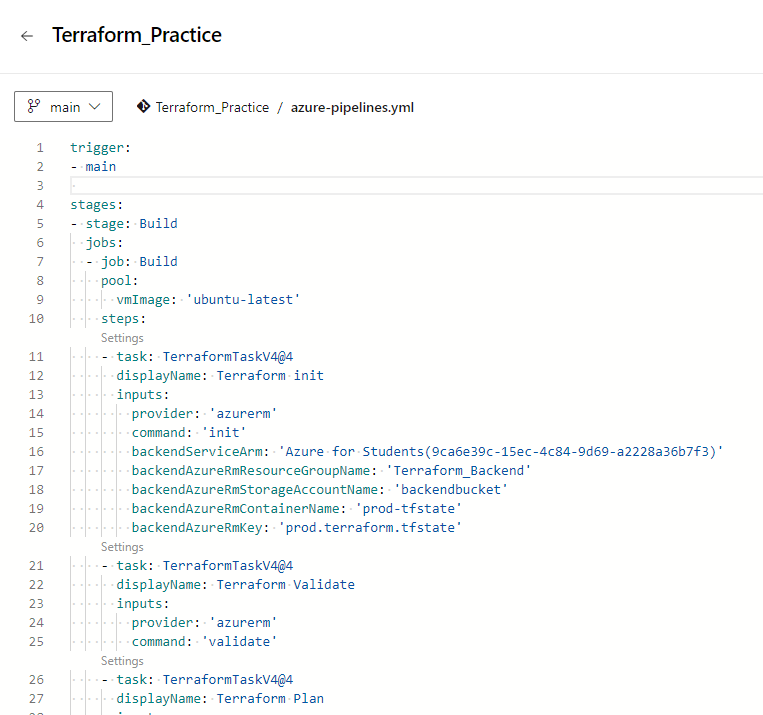
# Specify the source image reference  
 storage\_image\_reference {  
 publisher = var.os\_publisher  
 offer = var.os\_offer  
 sku = var.os\_sku  
 version = var.os\_version  
 }

tags = var.tags  
 }

## 4. Steps to Setup your Pipelines to run CI.

1. After the whole setup of files in Repos>files go to Pipelines>pipelines Section.
2. Click on New Pipeline.
3. At **Connect** section choose **Azure Repos Git**.
4. At **Select** section choose Y**our Project**.
5. At **Configure** section choose **Starter Pipeline**.
6. Now on Review Section you can write below .yml code.

Follow the same indentation as given in below image to ignore error.



trigger:

- main

stages:

- stage: Build

jobs:

- job: Build

pool:

vmImage: 'ubuntu-latest'

steps:

- task: TerraformTaskV4@4

displayName: Terraform init

inputs:

provider: 'azurerm'

command: 'init'

backendServiceArm: 'Azure for Students(9ca6e39c-15ec-4c84-9d69-a2228a36b7f3)'

backendAzureRmResourceGroupName: 'Terraform\_Backend'

backendAzureRmStorageAccountName: 'backendbucket'

backendAzureRmContainerName: 'prod-tfstate'

backendAzureRmKey: 'prod.terraform.tfstate'

- task: TerraformTaskV4@4

displayName: Terraform Validate

inputs:

provider: 'azurerm'

command: 'validate'

- task: TerraformTaskV4@4

displayName: Terraform Plan

inputs:

provider: 'azurerm'

command: 'plan'

environmentServiceNameAzureRM: 'Azure for Students(9ca6e39c-15ec-4c84-9d69-a2228a36b7f3)'

- task: ArchiveFiles@2

displayName: Archive Files

inputs:

rootFolderOrFile: '$(Build.SourcesDirectory)'

includeRootFolder: false

archiveType: 'zip'

archiveFile: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'

replaceExistingArchive: true

- task: PublishBuildArtifacts@1

displayName: Publish Artifact

inputs:

PathtoPublish: '$(Build.ArtifactStagingDirectory)'

ArtifactName: 'drop'

publishLocation: 'Container'

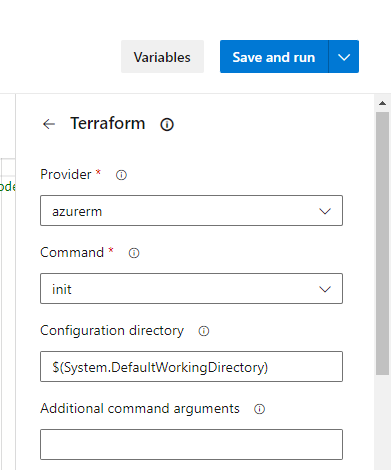
### NOTE: Change environmentServiceNameAzureRM with your subscription

Follow the below steps to change that:

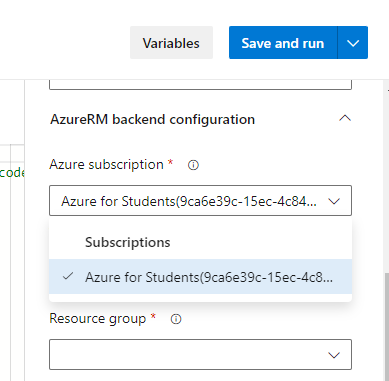
1.Click on Show Assistant.

2. On search box, search Terraform and choose terraform.

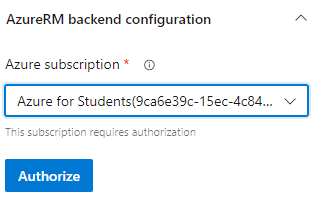
3. choose the operation/command you want to perform.



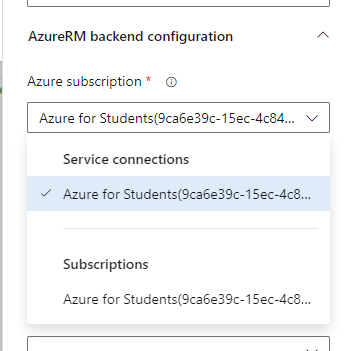
4. Configure azure subscription.



5. After selecting Subscription click on Authorize.



6. After authorizing select service connection for entire CI/CD pipeline



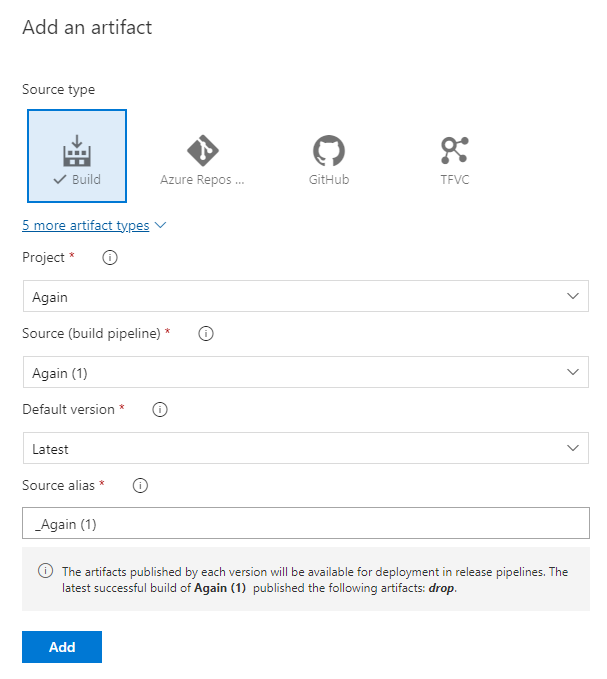
After completing all operations (init, validate, plan, etc.).

**Save and run pipeline.**

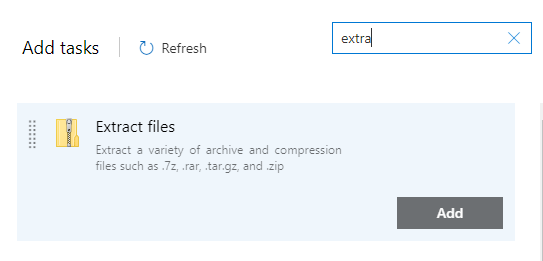
**CI Pipeline is Done.**

# Steps to run CD pipeline.

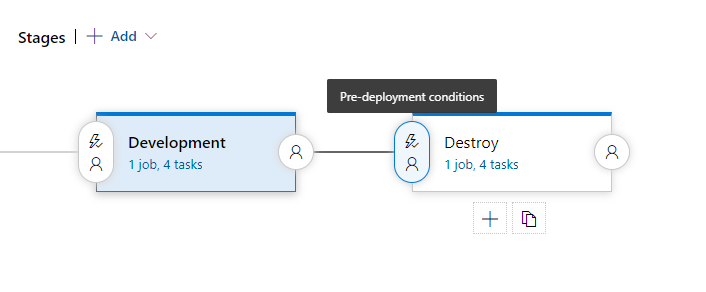
1. Go to Pipelines > Releases > New pipeline.
2. On Select a template click Empty job.
3. Give a meaning full name to your Stage and close that.
4. Click on Add an artifact.
5. From Source type choose **Build,**
   1. In project choose your **Project,**
   2. In Source (build pipeline) choose the **CI pipeline** for which you want to do CD operation,
   3. Leave reset field as default.
   4. Click on **add**.



1. Click on **job, task** section.
2. Click on **Agent job,**
   1. Change agent pool to **Azure Pipelines**,
   2. Agent specification to **ubuntu latest** (as show in below image).
   3. Leave reset field as default.
   4. Click on **+** symbol to add task on Agent job.
3. On search bar and search **Extract files** after that click on add.



1. Change **Destination folder** under **Extract files** and leave reset field as default.
2. Again, click on **+** symbol to add task on Agent job.
3. Now search Terraform and add **Terraform tool installer**.
4. Again, click on **+** symbol to add task on Agent job.
5. Again, search terraform and **Terraform.**
6. Under **Agent job** click on **Terraform** and edit the required fields as shown in below image.
7. For **Azure subscription, Resource group, Storage account,** **Container** and **Key,** do the same as you done on CI pipelines.
8. Again, Repeat the above step one more time and edit required fields as show in below image.
9. Now click on **Save,** after that **Ok** forpath.
10. Click on Pipeline and **Clone** your current job.
11. Click on view stage tasks.
12. Change stage name Destroy (or any).
13. Click on **Terraform apply** and change to **destroy**.
14. Click on **Pre-deployment conditions.**



1. Under Pre-deployment conditions:
   1. For select trigger choose **After Stage**
   2. Scroll down and enable Pre-deployment approvals and pass the user-name as approvers (admin name).
   3. Click on **Save.**
2. Now click on **Create release > Create**.

1. After clicking on Create your CD pipelines will trigger.
2. After getting Succeeded message of Devlopment Stage CD (for Creation of VM), you may Approve the Destroy Stage if your work is done.

Now your CI/CD Pipelines Creation and Deletion is done.