**Step-00: Introduction**

**Build Pipeline - CI**

* Implement Build Pipeline (Continuous Integration Pipeline)
* Use CopyFiles and PublishArtifacts Tasks in Build Pipeline

**Release Pipelines - CD**

* Implement Deployment stages Dev, QA, Stage and Prod
* In each stage implement below listed Tasks for a Ubuntu Agent
  + terraform install
  + terraform init
  + terraform validate
  + terraform plan
  + terraform apply -auto-approve
* Test both CI CD Pipelines
* [Azure DevOps Parallelism Free Tier Request Form](https://forms.office.com/pages/responsepage.aspx?id=v4j5cvGGr0GRqy180BHbR63mUWPlq7NEsFZhkyH8jChUMlM3QzdDMFZOMkVBWU5BWFM3SDI2QlRBSC4u)

**Step-01: Review Terraform Configs**

* **Folder:** Git-Repo-Files/terraform-manifests

**Step-01-01: c1-versions.tf**

# Terraform State Storage to Azure Storage Container (Values will be taken from Azure DevOps)

backend "azurerm" {

}

**Step-01-02: c7-01-web-linuxvm-input-variables.tf**

* Define Input Variables for VM Size and VM admin user name.
* If we required we can parameterize more arguments in azurerm\_linux\_virtual\_machine resource.

# Linux VM Input Variables Placeholder file.

variable "web\_linuxvm\_size" {

description = "Web Linux VM Size"

type = string

default = "Standard\_DS1\_v2"

}

variable "web\_linuxvm\_admin\_user" {

description = "Web Linux VM Admin Username"

type = string

default = "azureuser"

}

**Step-01-03: c7-05-web-linuxvm-resource.tf**

* Update arguments size, admin\_username and admin\_ssh\_key.username in Linux VM Resource

# Resource: Azure Linux Virtual Machine

resource "azurerm\_linux\_virtual\_machine" "web\_linuxvm" {

name = "${local.resource\_name\_prefix}-web-linuxvm"

#computer\_name = "web-linux-vm" # Hostname of the VM (Optional)

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

location = azurerm\_resource\_group.rg.location

size = var.web\_linuxvm\_size

admin\_username = var.web\_linuxvm\_admin\_user

network\_interface\_ids = [ azurerm\_network\_interface.web\_linuxvm\_nic.id ]

admin\_ssh\_key {

username = var.web\_linuxvm\_admin\_user

public\_key = file("${path.module}/ssh-keys/terraform-azure.pub")

}

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = "RedHat"

offer = "RHEL"

sku = "83-gen2"

version = "latest"

}

#custom\_data = filebase64("${path.module}/app-scripts/redhat-webvm-script.sh")

custom\_data = base64encode(local.webvm\_custom\_data)

}

**Step-01-04: terraform.tfvars**

# Generic Variables

business\_divsion = "hr"

resource\_group\_location = "eastus"

resource\_group\_name = "rg"

**Step-01-05: dev.tfvars**

# Environment Name

environment = "dev"

# Virtual Network Variables

vnet\_name = "vnet"

vnet\_address\_space = ["10.1.0.0/16"]

web\_subnet\_name = "websubnet"

web\_subnet\_address = ["10.1.1.0/24"]

app\_subnet\_name = "appsubnet"

app\_subnet\_address = ["10.1.11.0/24"]

db\_subnet\_name = "dbsubnet"

db\_subnet\_address = ["10.1.21.0/24"]

bastion\_subnet\_name = "bastionsubnet"

bastion\_subnet\_address = ["10.1.100.0/24"]

# Web Linux VM Variables

web\_linuxvm\_size = "Standard\_DS1\_v2"

web\_linuxvm\_admin\_user = "azureuser"

**Step-01-06: qa.tfvars**

# Environment Name

environment = "qa"

# Virtual Network Variables

vnet\_name = "vnet"

vnet\_address\_space = ["10.2.0.0/16"]

web\_subnet\_name = "websubnet"

web\_subnet\_address = ["10.2.1.0/24"]

app\_subnet\_name = "appsubnet"

app\_subnet\_address = ["10.2.11.0/24"]

db\_subnet\_name = "dbsubnet"

db\_subnet\_address = ["10.2.21.0/24"]

bastion\_subnet\_name = "bastionsubnet"

bastion\_subnet\_address = ["10.2.100.0/24"]

# Web Linux VM Variables

web\_linuxvm\_size = "Standard\_DS1\_v2"

web\_linuxvm\_admin\_user = "azureuser"

**Step-01-07: stage.tfvars**

# Environment Name

environment = "stage"

# Virtual Network Variables

vnet\_name = "vnet"

vnet\_address\_space = ["10.3.0.0/16"]

web\_subnet\_name = "websubnet"

web\_subnet\_address = ["10.3.1.0/24"]

app\_subnet\_name = "appsubnet"

app\_subnet\_address = ["10.3.11.0/24"]

db\_subnet\_name = "dbsubnet"

db\_subnet\_address = ["10.3.21.0/24"]

bastion\_subnet\_name = "bastionsubnet"

bastion\_subnet\_address = ["10.3.100.0/24"]

# Web Linux VM Variables

web\_linuxvm\_size = "Standard\_DS1\_v2"

web\_linuxvm\_admin\_user = "azureuser"

**Step-01-08: prod.tfvars**

# Environment Name

environment = "prod"

# Virtual Network Variables

vnet\_name = "vnet"

vnet\_address\_space = ["10.4.0.0/16"]

web\_subnet\_name = "websubnet"

web\_subnet\_address = ["10.4.1.0/24"]

app\_subnet\_name = "appsubnet"

app\_subnet\_address = ["10.4.11.0/24"]

db\_subnet\_name = "dbsubnet"

db\_subnet\_address = ["10.4.21.0/24"]

bastion\_subnet\_name = "bastionsubnet"

bastion\_subnet\_address = ["10.4.100.0/24"]

# Web Linux VM Variables

web\_linuxvm\_size = "Standard\_DS1\_v2"

web\_linuxvm\_admin\_user = "azureuser"

**Step-01-09: No Changes to files**

* c2-generic-input-variables.tf
* c3-locals.tf
* c4-random-resources.tf
* c5-resource-group.tf
* c6-01 to c6-07 Virtual Network Files
* c7-02-web-linuxvm-publicip.tf
* c7-03-web-linuxvm-network-interface.tf
* c7-04-web-linuxvm-network-security-group.tf
* c7-06-web-linuxvm-outputs.tf

**Step-02: Create Github Repository and Check-In Files**

**Step-02-01: Create new github Repository**

* **URL:** github.com
* Click on **Create a new repository**
* **Repository Name:** terraform-on-azure-with-azure-devops
* **Description:** Terraform on Azure with Azure IaC DevOps
* **Repo Type:** Public / Private
* **Initialize this repository with:**
* **CHECK** - Add a README file
* **CHECK** - Add .gitignore
* **Select .gitignore Template:** Terraform
* **CHECK** - Choose a license (Optional)
* **Select License:** Apache 2.0 License
* Click on **Create repository**

**Step-02-02: Clone Github Repository to Local Desktop**

# Clone Github Repo

git clone https://github.com/<YOUR\_GITHUB\_ID>/<YOUR\_REPO>.git

git clone https://github.com//terraform-on-azure-with-azure-devops.git

**Step-02-03: Copy files from Git-Repo-Files folder to local repo & Check-In Code**

* **Source Location:** Git-Repo-Files
* **Destination Location:** Copy all folders and files from Git-Repo-Files newly cloned github repository folder in your local desktop terraform-on-azure-with-azure-devops
* **Check-In code to Remote Repository**

# GIT Status

git status

# Git Local Commit

git add .

git commit -am "First Commit"

# Push to Remote Repository

git push

# Verify the same on Remote Repository

https://github.com//terraform-on-azure-with-azure-devops.git

**Step-03: Terraform Dependency Lock File Concept**

* Understand [Terraform Dependency Lock File Concept](https://www.terraform.io/docs/language/dependency-lock.html)

# Go to Local Git Repo

cd demo-repos

cd terraform-on-azure-with-azure-devops/terraform-manifests

# Delete `.terraform.lock.hcl`

Delete file if exists "`.terraform.lock.hcl`"

rm -rf .terraform.lock.hcl # Explicitly for students to get latest version of Providers on that respective day when you are learning this module

# Terraform Providers lock for multiple platforms

terraform providers lock -platform=windows\_amd64 -platform=darwin\_amd64 -platform=linux\_amd64

# Terraform Initialize

terraform init

Observation:

1. Provider plugins downloaded to ".terraform folder"

2. `.terraform.lock.hcl` created with command `terraform providers lock` used by `terraform init` to download those respective providers

3. We need to check-in this file `.terraform.lock.hcl` with our TF Configs to Git Repos for IaC DevOps Pipelines to ensure our provider versions doesnt get upgraded to latest versions and break our application.

# Delete ".terraform" folder

rm -rf .terraform

# We will ensure we are checking in `.terraform.lock.hcl`

`.terraform.lock.hcl`

# GIT Status

git status

# Git Local Commit

git add .

git commit -am "First Commit"

# Push to Remote Repository

git push

# Verify the same on Remote Repository

**Step-04: Create Azure DevOps Organization**

**Step-04-01: Create Azure DevOps Organization**

* Understand about [Azure DevOps Agents and Free-Tier Limits](https://docs.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops&tabs=ms-hosted)
* Navigate to https://dev.azure.com
* Click on Sign in to Azure DevOps
* Provide your Azure Cloud admin user
  + Username: XXXXXXXXXXXXXX
  + Password: XXXXXXXXXXXXXX
* Click on create **New Organization**
* **Name your Azure DevOps organization:** <any name>
* **We'll host your projects in:** Choose the location (Azure selects based on current location where you are accessing from)
* **Enter the characters you see:**
* Click on **Continue**

**Step-04-02: Request for Azure DevOps Parallelism**

* [Azure DevOps Parallelism Free Tier Request Form](https://forms.office.com/pages/responsepage.aspx?id=v4j5cvGGr0GRqy180BHbR63mUWPlq7NEsFZhkyH8jChUMlM3QzdDMFZOMkVBWU5BWFM3SDI2QlRBSC4u)

**Step-05: Install Terraform Extension for Azure DevOps**

* [Terraform Extension for Azure DevOps](https://marketplace.visualstudio.com/items?itemName=ms-devlabs.custom-terraform-tasks)

**Step-06: Create New Project in Azure DevOps Organization**

* Create a New Project in Azure DevOps Organization newly created
* Click on **New Project**
* **Project Name:** terraform-on-azure-with-azure-devops
* **Description:** terraform-on-azure-with-azure-devops
* **Visibility:** Private
* Click on **Create**

**Step-07: Understand Azure Pipelines**

* Understand about Azure Pipelines
* Pipeline Hierarchial Flow: Stages -> Stage -> Jobs -> Job -> Steps -> Task1, Task2

**Step-08: Create Azure CI (Continuous Integration) Pipeline (Build Pipeline)**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Pipelines
* Click on **New Pipeline**
* **Where is your code?:** GitHub
* Follow browser redirect steps to integrate with Github Account
* **Select a repository:** <any name>/terraform-on-azure-with-azure-devops
* **Configure your pipeline:** Starter Pipeline
* Rename the Pipeline file name to 01-terraform-azure-devops-ci-pipeline
* Build the below code using two tasks listed below
  + Copy Files
  + Publish Artifacts
* Click on **Save and Run** to Run the pipeline

trigger:

- main

# Stages

# Stage-1:

# Task-1: Copy terraform-manifests files to Build Artifact Directory

# Task-2: Publish build articats to Azure Pipelines

# Pipeline Hierarchial Flow: Stages -> Stage -> Jobs -> Job -> Steps -> Task1, Task2, Task3

stages:

# Build Stage

- stage: Build

displayName: Build Stage

jobs:

- job: Build

displayName: Build Job

pool:

vmImage: 'ubuntu-latest'

steps:

## Publish Artifacts pipeline code in addition to Build and Push

- bash: echo Contents in System Default Working Directory; ls -R $(System.DefaultWorkingDirectory)

- bash: echo Before copying Contents in Build Artifact Directory; ls -R $(Build.ArtifactStagingDirectory)

# Task-2: Copy files (Copy files from a source folder to target folder)

# Source Directory: $(System.DefaultWorkingDirectory)/terraform-manifests

# Target Directory: $(Build.ArtifactStagingDirectory)

- task: CopyFiles@2

inputs:

SourceFolder: '$(System.DefaultWorkingDirectory)/terraform-manifests'

Contents: '\*\*'

TargetFolder: '$(Build.ArtifactStagingDirectory)'

OverWrite: true

# List files from Build Artifact Staging Directory - After Copy

- bash: echo After copying to Build Artifact Directory; ls -R $(Build.ArtifactStagingDirectory)

# Task-3: Publish build artifacts (Publish build to Azure Pipelines)

- task: PublishBuildArtifacts@1

inputs:

PathtoPublish: '$(Build.ArtifactStagingDirectory)'

ArtifactName: 'terraform-manifests'

publishLocation: 'Container'

* Verify First Run logs
* Rename the Pipeline name to Terraform Continuous Integration CI Pipeline

**Step-09: Sync Local Git Repo**

* A new file named 01-terraform-azure-devops-ci-pipeline.yaml will be added git Remote Repo
* Sync the same thing to your local Git Repo

# Local Git Repo

git pull

**Step-10: Azure Release Pipelines Introduction**

1. Understand Azure Release Pipelines
2. What are we going to implement as part of Release Pipelines ?
3. Review the Infra we are going to provision.
4. Understand where Terraform State files will be stored for 4 environments.
5. Demonstrate Continuous Delivery by making a change to our TF Configs atleast for one environment (Prod)

**Step-11: Create Azure Resource Manager Service Connection for Azure DevOps**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) > Project Settings -> Pipelines -> Service Connections
* Click on **New Service Connection**
* **Choose a service or connection type:** Azure Resource Manager
* **Authentication Method:** Service principal (automatic)
* **Scope level:** Subscription
* **Subscription:** Select Subsciption if we have many
* **Username:** Azure Cloud Admin User
* **Password:** XXXXXXXXX
* **Resource Group:** leave empty
* **Service connection name:** terraformiacdevops1
* **Description (optional):** terraformiacdevops1 Service Connection used for CICD Pipelines
* **Security:** CHECK Grant access permission to all pipelines (leave to default checked)
* Click on **Save**

**Step-12: Create Storage Account for storing Terraform State Files**

* Create Storage Account, Storage Container if not created.
* We have already created that as part of [Section-24-Step-02](https://github.com/stacksimplify/terraform-on-azure-cloud/tree/main/24-Terraform-Remote-State-Storage#step-02-create-azure-storage-account) which will re-use here.

# Terraform State Storage to Azure Storage Container

resource\_group\_name = "terraform-storage-rg"

storage\_account\_name = "terraformstate201"

container\_name = "tfstatefiles"

key = "dev-terraform.tfstate"

**Step-13: Release Pipelines - Create Dev Stage**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Releases
* Click on **New Release Pipeline**
* **Pipeline Name:** Terraform-CD

**Dev Stage**

* **Stage Name:** Dev Stage
* **Stage Owner:** [@gmail.com](mailto:stacksimplify@gmail.com) (your-azure-admin-id)
* Click on **1 Job, 0 Task**

**Agent Job**

* **Display Name:** Terraform Ubuntu Agent
* **Agent Pool:** Azure Pipelines
* **Agent Specification:** Ubuntu latest image
* Rest all leave to defaults

**Task-1: Terraform Tool Installer**

* **Display Name:** Install Terraform latest version
* **Version:** 1.0.5 (as on today)
* **Important Note:** Get latest terraform version number from [Terraform Downloads page](https://www.terraform.io/downloads.html)

**Task-2: Terraform: init**

* **Display Name:** Terraform: init
* **Provider:** azurerm
* **Command:** init
* **Configuration directory:** Select by browsing it (Example: $(System.DefaultWorkingDirectory)/\_Terraform Continuous Integration CI Pipeline/terraform-manifests)
* **Additional command arguments:** Nothing leave empty
* **AzureRM backend configuration**
* **Azure subscription:** terraformiacdevops1 (Select the service connection created in step-10)
* **Resource group:** terraform-storage-rg
* **Storage account:** terraformstate201
* **Container:** tfstatefiles
* **Key:** dev-terraform.tfstate
* Rest all leave to defaults

**Task-3: Terraform: validate**

* **Display Name:** Terraform: validate
* **Provider:** azurerm
* **Command:** validate
* **Configuration directory:** Select by browsing it (Example: $(System.DefaultWorkingDirectory)/\_Terraform Continuous Integration CI Pipeline/terraform-manifests)
* **Additional command arguments:** Nothing leave empty
* Rest all leave to defaults

**Task-4: Terraform: plan**

* **Display Name:** Terraform: plan
* **Provider:** azurerm
* **Command:** plan
* **Configuration directory:** Select by browsing it (Example: $(System.DefaultWorkingDirectory)/\_Terraform Continuous Integration CI Pipeline/terraform-manifests)
* **Additional command arguments:** -var-file=dev.tfvars
* **Azure subscription:** terraformiacdevops1 (Select the service connection created in step-10)
* Rest all leave to defaults

**Task-5: Terraform: apply -auto-approve**

* **Display Name:** Terraform: apply -auto-approve
* **Provider:** azurerm
* **Command:** validate and apply
* **Configuration directory:** Select by browsing it (Example: $(System.DefaultWorkingDirectory)/\_Terraform Continuous Integration CI Pipeline/terraform-manifests)
* **Additional command arguments:** -var-file=dev.tfvars -auto-approve
* **Azure subscription:** terraformiacdevops1 (Select the service connection created in step-10)
* Rest all leave to defaults
* Click on \**Save* to save the release-pipeline.

**Step-14: Release Pipeline - Artifacts Settings**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Releases -> Terraform-CD

**Step-14-01: Add Artifacts**

* Click on **Add Artifacts**
* **Source Type:** Build
* **Project:** terraform-on-azure-with-azure-devops
* **Source (build pipeline):** Terraform Continuous Integration CI Pipeline
* **Default version:** Latest (leave to default)
* **Source alias:** \_Terraform Continuous Integration CI Pipeline (leave to default)
* Click on **Add**

**Step-14-02: Enable Continuous deployment trigger**

* **Continuous deployment trigger:** Enabled
* Rest all leave to defaults

**Step-15: Trigger Build (CI) and Release (CD) Pipelines**

* Make a minor change in git repo and push the changes from local git repo

## In any file add some changes

Example: Add some comment in any of the \*.tf files (Just for testing)

# Git Status

git status

# Git Commit

git commit -am "CICD-Test-1"

# Git Push

git push

**Step-16: Review Build (CI) Pipeline and Release Pipeline(CD) Logs**

**Verify Build Pipeline Logs**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Pipelines -> Terraform Continuous Integration CI Pipeline

**Verify Release Pipeline Logs**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Releases -> Terraform CD

**Step-17: Verify Dev Resources created in Azure Portal**

**Verify dev-terraform.tfsate file**

* Go to Storaage Accounts -> terraform-rg-storage -> terraformstate201 -> tfstatefiles
* Verify the file dev-terraform.tfstate

**Verify Dev Resources in Azure Portal**

1. Azure Virtual Network
2. Azure Subnets
3. Azure Public IP
4. Azure Linux Virtual Machine

**Step-18: Create Stages listed below by cloning Dev Stage in Releases**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Releases -> Terraform CD -> Edit
* Updates include the following for QA, Stage and Prod

**Task-1: Terraform: init**

* Update Key to respective environment
* **QA Key:** qa-terraform.tfsate
* **Stage Key:** stage-terraform.tfstate
* **Prod Key:** prod-terraform.tfstate

**Task-2: Terraform: plan**

* Update Additional command arguments to respective environment
* **QA Additional command arguments:** -var-file=qa.tfvars
* **Stage Additional command arguments:** -var-file=stage.tfvars
* **Prod Additional command arguments:** -var-file=prod.tfvars

**Task-3: Terraform: apply -auto-approve**

* Update Additional command arguments to respective environment
* **QA Additional command arguments:** -var-file=qa.tfvars -auto-approve
* **Stage Additional command arguments:** -var-file=stage.tfvars -auto-approve
* **Prod Additional command arguments:** -var-file=prod.tfvars -auto-approve

**Step-19: Add Pre-Deployment Approval and Post Deployment Approvals**

* **Pre-Deployment Approvals:** QA, Stage and Prod
* **Post-Deployment Approvals:** Stage

**Step-20: Trigger Build (CI) and Release (CD) Pipelines**

* Make a minor change in git repo and push the changes from local git repo

## In any file add some changes

Example: Add some comment in any of the \*.tf files (Just for testing)

# Git Status

git status

# Git Commit

git commit -am "CICD-Test-2"

# Git Push

git push

**Step-21: Verify Resources created in Azure Portal for QA, Stage and Prod Environments**

**Verify TFState File for Dev, QA and Prod**

* Go to Storaage Accounts -> terraform-rg-storage -> terraformstate201 -> tfstatefiles
* Verify the files listed below
* qa-terraform.tfstate
* stage-terraform.tfstate
* prod-terraform.tfstate

**Verify Resources in Azure Portal for QA, Stage and Prod Environments**

1. Azure Virtual Network
2. Azure Subnets
3. Azure Public IP
4. Azure Linux Virtual Machine

**Step-22: Change web\_linuxvm\_admin\_user to Prod Environment**

# File: prod.tfvars

#web\_linuxvm\_admin\_user = "azureuser"

web\_linuxvm\_admin\_user = "produser" # Enable during step-21

# Git Status

git status

# Git Commit

git commit -am "Changed Prod VM adminuser name to produser"

# Git Push

git push

**Step-23: Review Build (CI) Pipeline and Release Pipeline(CD) Logs**

**Verify Build Pipeline Logs**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Pipelines -> Terraform Continuous Integration CI Pipeline

**Verify Release Pipeline Logs**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Releases -> Terraform CD
* **Dev Stage:** Review Logs
* **QA Stage:** Approve (Pre-Deployment Approval) and Review Logs
* **Staging Stage:** Approve (Pre-Deployment Approval) and Review logs and also do Post-Deployment Approval
* **Prod Stage:** Approve (Pre-Deployment Approval)

**Verify Virtual Machines in Azure Portal**

* Go to -> Virtual Machines
* Verify VM hr-prod-web-linuxvm and get the Public IP

# Connect to prod VM using SSH

ssh -i ssh-keys/terraform-azure.pem produser@<Prod-VM-Public-IP>

**Step-24: Disable Build (CI) Pipeline**

* Go to Azure DevOps -> Organization -> Project (terraform-on-azure-with-azure-devops) -> Pipelines -> Pipelines -> Terraform Continuous Integration CI Pipeline
* Settings -> Disabled -> Click on **Save**
* This will help us if by any chance you made any accidental commits to your git repo we don't get any unexpected surprise azure bills.

**Step-25: Delete Resources or Clean-Up**

**Delete Resources**

* Go to Azure Portal -> Resource Groups -> Delete Resource Groups for All Environments
* Dev
* QA
* Staging
* Prod

**Delete Terraform State Files**

* Go to Azure Portal -> Storage Containers -> terraformstate201 -> Containers -> tfstatefiles -> Delete all files
* dev-terraform.tfstate
* qa-terraform.tfstate
* stage-terraform.tfstate
* prod-terraform.tfstate