**Scenario 1.**

1. If we are using the classic ci/cd process, then we need to check the option for continuous integration so that when ever the developer commits the code in to the repository, immediately the built process will start.

Else if we are using the pipeline.yml then , we can mention bellow things.

**Trigger:**

**-Main/master**

1. We can create a multi stage pipeline in order to achieve the requirement. Below is a sample multistage pipeline.yml file

stages:

- stage: Build

jobs:

- job: BuildJob

steps:

- script: echo Building!

- stage: Web

jobs:

- job: Deploy

steps:

- script: echo Deploying the code!

- stage: Test

jobs:

- job: TestOne

steps:

- script: echo Test One

- job: TestTwo

steps:

- script: echo Test Two

- stage: API

jobs:

- job: API

steps:

- script:

5. we can add the approvers in the cicd pipeline process before a specific task so that it the task will not be performed unless the approver approves it .

**Scenario 2:**

**2.**

List the tools you will create and store the Terraform templates.

Answer:

You need to create different terraform templates like

Terraform modules which is used for the reusability of the code.

Terraform Variable file which will contain the variables.

Terraform main.tf file which will contain the main resource block that we need to create in the infrastructure.

Terraform state file that will contain the state and the config details of the infrastructure we are creating using main.tf.

**3.**

Explain the process and steps to create automated deployment pipeline.

Answer:

A deployment pipeline typically follows 3 main steps (though you may also have more): **build, test, deploy**. This is the pipeline that supports your ability to automate the deployment process and ensures that code moves from being committed to deployment quickly.

Below are the steps need to be followed to create a automated deployment process.

If we are using the classic ci/cd process, then we need to check the option for continuous integration so that when ever the developer commits the code in to the repository, immediately the built process will start.

Else if we are using the pipeline.yml then , we can mention bellow things.

**Trigger:**

**-Main/master**

Inside the pipeline.yml file, we can mention different tasks that need to be performed as per the requirement so that it will process automatically.

4. I have added the main.tf file in the repo for the same.

5. Explain how will you access the password stored in Key Vault and use it as Admin Password in the VM Terraform template.

Answer:

You first need to create a data resource to the azure key vault to get the key vault resource ID:

data "azurerm\_key\_vault" "keyvault" {

name = "${var.keyvault\_name}"

resource\_group\_name = "${var.resourcegroup\_name}"

}

And then use **azurerm\_key\_vault\_secret** to get the secret with the key vault resource Id:

data "azurerm\_key\_vault\_secret" "win\_admin\_pass" {

name = "${var.secret\_name}"

key\_vault\_id = "${data.azurerm\_key\_vault.keyvault.id}"

}