Answer Sheet

Scenario1:

**1.The build should trigger as soon as anyone in the dev team checks in code to master branch.**

Ans: We need to define trigger in build definition that means point at which we want build to get trigger, in above case, we want as soon as master branch is updated so,

Trigger = branch (master)

Definition:

YAML

trigger:

branches:

include:

- refs/heads/master

**2. There will be test projects which will create and maintained in the solution along the Web and API. The trigger should build all the 3 projects - Web, API and test.**

Ans: Taking three different solution files for 3 projects under same repository,

Step1: NuGet restore \*\*\\*.sln

Step2: build solution file Web.sln

**Command**: msbuild.exe web.sln ( we can define additional argument mention below in YAML)

**YAML**

- task: MSBuild@1

name: MSBuild\_3

displayName: Build solution web.sln

inputs:

**solution: web.sln**

**msbuildArchitecture**: x64

**platform**: $(BuildPlatform)

**configuration**: $(BuildConfiguration)

**msbuildArguments**: /p:DeployOnBuild=true /p:BuildingProject=true /p:CreatePackageOnPublish=false /p:NugetRepository=$(NugetRepository)

**Step3**: Building API Solution API.sln

Command: msbuild.exe API.sln

**Step4**: Build solution file Test.sln

Command: msbuild.exe Test.sln

**Step5: Testing assemblies** generated in above steps using VSTest task

**YAML**

- task: VSTest@2

name: VSTest\_4

displayName: Test Assemblies \*\*\\*test\*.dll;-:\*\*\obj\\*\*

inputs:

searchFolder: $(build.artifactstagingdirectory)

**The build should not be successful if any test fails.**

Answer: We need to add false in option **continueOnError**

Ex: **continueOnError: False**

Q3. The deployment of code and artifacts should be automated to Dev environment.

Ans: **Enable continuous integration** option in build definition and **Continuous deployment trigger in release definition, also we need to define trigger as “After release” for Development environment** so that as soon as build is succeeded deployment will get trigger for the same otherwise even if continuous deployment trigger is enabled and Development stage is not set as “After release” deployment will not happen.

Q4. Upon successful deployment to the Dev environment, deployment should be easily promoted to QA and Prod through automated process.

Q5 The deployments to QA and Prod should be enabled with Approvals from approvers only.

Ans: To meet above condition we need to apply trigger on both QA and PROD stages in same pipeline, below is the setting:

a) Trigger (**After release)**

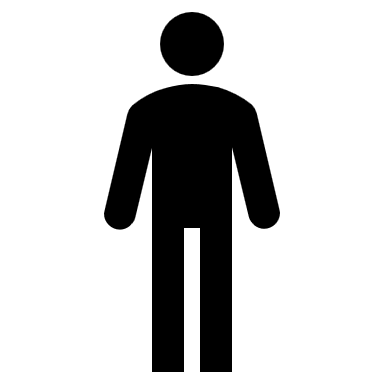
Development (successful)

b) Trigger (**After stage = Development**)

QA (deployment successful)

c) Trigger (**After stage = QA**) and pre-deployment approvals and add Approvers

PROD (deployment)



PROD

Approver

Development

QA

Scenario2:

Q1: What are different artifacts you need to create - name of the artifacts and its purpose.

Build artifact: It will contain all the terraform files and templates form source code management (either git,github,tfvc etc) and will be used during deployment.

Q2. List the tools you will to create and store the Terraform templates.

Ans: SCM tools such as Azure Devops git , Github, tfvc etc

Q3: Explain the process and steps to create automated deployment pipeline

Step1: Build: To consolidate source code create a drop (as in this case only network and infrastructure included)

Step2: Deployment: (All the below task requires Azure service connection with all valid permissions)

1. In release pipeline add task terraform installer
2. Add task in pipeline to initialize terraform

**command**: terraform init (argument “resource group” “Storage account” “Container” “Key” need to be define

1. Once terraform initialize and terraform state is set, add task terraform plan
2. Finally, add terraform validate and apply command -auto-approve
3. If we have any App to be deployed we can deploy using Azure App service task

Q4. Create a sample Terraform template you will use to deploy Below services:

Vnet

2 Subnet

NSG to open port 80 and 443

1 Window VM in each subnet

1 Storage account



Q5. Explain how will you access the password stored in Key Vault and use it as Admin Password in the VM Terraform template.

Ans. To access key vault password in terraform template as VM admin password, below are the two steps.

Step1 **Add data source as key Vault**

**data** "azurerm\_key\_vault\_secret" "mySecret" {

name = "Admin"

vault\_uri = "[https://yourKeyVault.vault.azure.net/](https://yourkeyvault.vault.azure.net/)"

}

Step2: **Reference in the operating system profile section of the deployment config. As given in attached sample.tf**

os\_profile {

computer\_name = "tfvm"

admin\_username = "Admin"

admin\_password = "${data.azurerm\_key\_vault\_secret.mySecret.value}"

}