Q2 – SCENARIO

Macro Life, a healthcare company has recently setup the entire Network and Infrastructure on Azure.
The infrastructure has different components such as Virtual N/W, Subnets, NIC, IPs, NSG etc.
The IT team currently has developed PowerShell scripts to deploy each component where all the properties of each resource is set using PowerShell commands.
The business has realized that the PowerShell scripts are growing over period of time and difficult to handover when new admin onboards in the IT.
The IT team has now decided to move to Terraform based deployment of all resources to Azure.
All the passwords are stored in a Azure Service known as key Vault. The deployments needs to be automated using Azure DevOps using IaC(Infrastructure as Code).
1) What are different artifacts you need to create - name of the artifacts and its purpose
2) List the tools you will to create and store the Terraform templates.
3) Explain the process and steps to create automated deployment pipeline.
4) 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
5) Explain how will you access the password stored in Key Vault and use it as Admin Password in the VM
Terraform template.

Please find a solution to this scenario below:

1) By Artifacts, if you mean resources in Azure then I'll be creating a **storage account** in order to store **tfstatefiles** in a **container.** This **storage account** will be part of a separate **resource group** will also be created that will contain anything related to **Terraform state files.**

   Also, a **key vault** will have to be created that **contains secrets/passwords** that will be consumed by Terraform when provisioning **Network and Infrastructure resources**. The reason for manual creation of this key vault is to avoid storing passwords in the code.

2) I will use **VisualStudio Code** in order to create Terraform templates and store them in **Github/Azure DevOps GIT**. Any changes done to Terraform templates will have to be done via a **PR (PullRequest)** with enforced branch policies that encourage peer reviews and as a result, knowledge transfer.

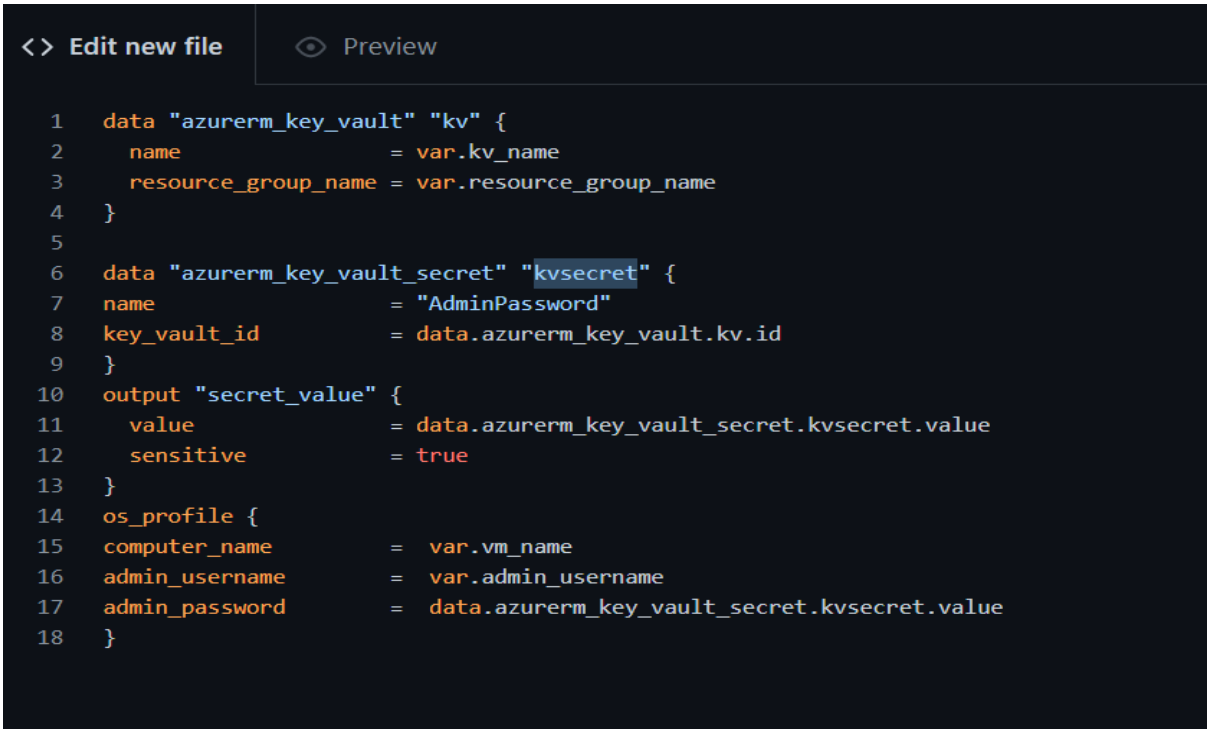
3) I've created a **sample-terraform-deploy.yml** file that can be found here: https://github.com/bolvinfernandes/MaerskDevopsTest/blob/main/MacroLife/sample-terraform-deploy.yml
   This file does a terraform module install followed by a plan and apply.

   For the pipeline referring to this yml file I will use **AzureDevops** and pass the `{Environment}` variable at build runtime. Value of this variable can be **dev/test/QA/PROD**. The idea is to use the **same pipeline to provision multiple environments**.

4) Rather than creating a single template I've created multiple modules which is more granular and aids debugging when things don't go well. Please find them here: https://github.com/bolvinfernandes/MaerskDevopsTest/tree/main/MacroLife

5) Using **Data Sources** it's possible to **retrieve Azure Key Vault secrets** (along with their values) and then **store the (data) password in a variable** which can then be used to set the **AdminPassword for the VM** in question.

- You can find the below snippet here: https://github.com/bolvinfernandes/MaerskDevopsTest/blob/main/MacroLife/kv-retrieve-secret.tf



- We could use powershell

get-azurekeyvaultsecret -vaultName "<vault-name>" -name "<secret-name>"

- Also, we could use the following in **AzureDevops to link keyvault secrets to a variable group**