Challenge #1

A 3-tier environment is a common setup. Use a tool of your choosing/familiarity create these resources on a cloud environment (Azure/AWS/GCP). Please remember we will not be judged on the outcome but more focusing on the approach, style and reproducibility.

Solutions 1:

To create a 3-tier environment on Azure, we are using Azure CLI, which is a command-line tool that we can use to manage Azure resources. The Azure CLI is available on Windows, macOS, and Linux, and it can be installed using the following command:

 # Install the Azure CLI on Windows
Invoke-WebRequest -Uri https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi; Start-Process msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'; rm .\AzureCLI.msi

To create a 3-tier environment on Azure, we will need to create the following resources:

A resource group: *A resource group is a logical container for Azure resources. We can create a resource group using the az group create command.*

A virtual network: *A virtual network is a logical network in Azure that we can use to connect your resources. We can create a virtual network using the az network vnet create command.*

Subnets*: Subnets are segments of a virtual network that you can use to isolate resources. We can create subnets using the az network vnet subnet create command.*

Network security groups: *Network security groups are used to control inbound and outbound traffic to resources in a virtual network. We can create network security groups using the az network nsg create command.*

Virtual machines: *Virtual machines are the compute resources in a 3-tier environment. We can create virtual machines using the az vm create command.*

 # Create a resource group
az group create --name my-resource-group --location eastus

 # Create a virtual network
az network vnet create --name my-vnet --resource-group my-resource-group --address-prefixes 10.0.0.0/16

 # Create subnets
az network vnet subnet create --name my-subnet-1 --resource-group my-resource-group --vnet-name my-vnet --address-prefix 10.0.0.0/24
az network vnet subnet create --name my-subnet-2 --resource-group my-resource-group --vnet-name my-vnet --address-prefix 10.0.1.0/24

 # Create network security groups
az network nsg create --name my-nsg-1 --resource-group my-resource-group
az network nsg create --name my-nsg-2 --resource-group my-resource-group

 # Create virtual machines

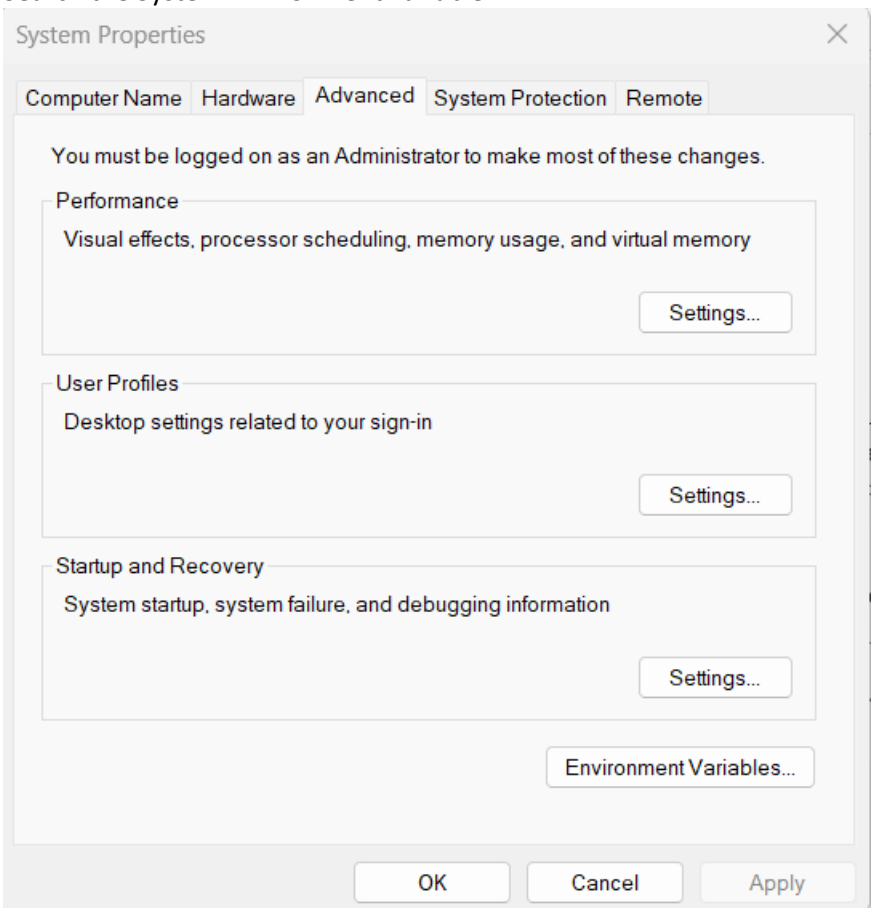
az vm create --name my-vm-1 --resource-group my-resource-group --vnet-name my-vnet --subnet my-subnet-1 --nsg my-nsg-1 --image UbuntuLTS --admin-username azureuser --generate-ssh-keys
az vm create --name my-vm-2 --resource-group my-resource-group --vnet-name my-vnet --subnet my-subnet-2 --nsg my-nsg-2 --image UbuntuLTS --admin-username azureuser –

Solution 2:

Terraform Installation:

1. First, I downloaded terraform for my local system from the website (https://developer.hashicorp.com/terraform/downloads?product_intent=terraform)
2. Select windows - Version: 1.3.6
3. Extract the file and create a folder on C drive with the name Terraform, pasted the terraform.exe into C:\Terraform
4. Configure the system environment variable

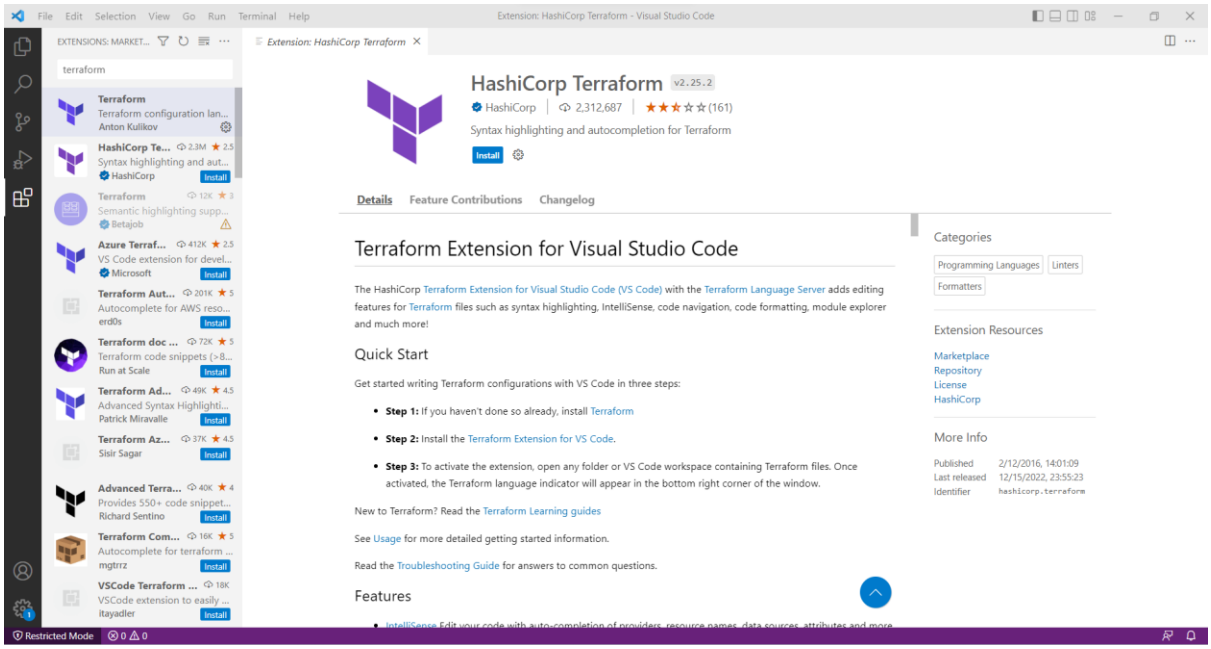
- Search the System Environment variable



-
- Click on the Environment variable
- Edit Path for User Variable and Add the System variable

Code Development in the Visual Studio Code:

1. Downloaded the Extensions – Terraform, Hashicorp Terraform, Azure Account, Azure Resource Manager, Azure Resources



2. ***Downloaded the GIT (***Git - Downloading Package (git-scm.com)***)***
3. ***Setup up Connectivity to Azure (Services Principal Needs to be created)***

- *Az Login*
- *Az Account Set –Subscription (Subscription Detail)*
- *Az login services principal (Username, Password & Tenant ID)*
- *CD Terraform*

To create a 3-tier environment on Azure using Terraform, we will need to define the following resources:

A resource group: We can define a resource group using the azurerm_resource_group resource.

A virtual network: We can define a virtual network using the azurerm_virtual_network resource.

Subnets: We can define subnets using the azurerm_subnet resource.

Network security groups: We can define network security groups using the azurerm_network_security_group resource.

Virtual machines: We can define virtual machines using the azurerm_virtual_machine resource.

3-tier environment on Azure using Terraform:

```
# Configure the Azure provider
provider "azurerm" {
  version = "2.36.0"
}

# Create a resource group
resource "azurerm_resource_group" "my-resource-group" {
  name     = "my-resource-group"
  location = "eastus"
}
```

```
# Create a virtual network
resource "azurerm_virtual_network" "my-vnet" {
  name                = "my-vnet"
  resource_group_name = azurerm_resource_group.my-resource-group.name
  address_space       = ["10.0.0.0/16"]
}

# Create subnets
resource "azurerm_subnet" "my-subnet-1" {
  name                 = "my-subnet-1"
  resource_group_name  = azurerm_resource_group.my-resource-group.name
  virtual_network_name = azurerm_virtual_network.my-vnet.name
  address_prefix       = "10.0.0.0/24"
}

resource "azurerm_subnet" "my-subnet-2" {
  name                 = "my-subnet-2"
  resource_group_name  = azurerm_resource_group.my-resource-group.name
  virtual_network_name = azurerm_virtual_network.my-vnet.name
  address_prefix       = "10.0.1.0/24"
}

# Create network security groups
resource "azurerm_network_security_group" "my-nsg-1" {
  name                = "my-nsg-1"
  resource_group_name = azurerm_resource_group.my-resource-group.name
}

resource "azurerm_network_security_group" "my-nsg-2" {
  name                = "my-nsg-2"
  resource_group_name = azurerm_resource_group.my-resource-group.name
}

# Create virtual machines
resource "azurerm_virtual_machine" "my-vm-1" {
  name                  = "my-vm-1"
  resource_group_name   = azurerm_resource_group.my-resource-group.name
  location              = azurerm_resource_group.my-resource-group.location
  network_interface_ids = [azurerm_network_interface.
```

Challenge #2
We need to write code that will query the meta data of an instance within AWS or Azure or GCP and provide a json formatted output.
The choice of language and implementation is up to you.

1. *Library will be azure-sdk-for-python*
2. *pip install azure-sdk-for-python*
3.

```
import json
```

from azure.mgmt.compute import ComputeManagementClient
from azure.common.credentials import ServicePrincipalCredentials

 # Create a compute management client
credentials = ServicePrincipalCredentials(
client_id='your-client-id',
secret='your-client-secret',
tenant='your-tenant-id'
)
client = ComputeManagementClient(credentials, 'your-subscription-id')

 # Call the virtual_machines method to get the metadata of the instance
vm = client.virtual_machines.get('your-resource-group', 'your-vm-name')

 # Print the metadata in JSON format
print(json.dumps(vm.as_dict(), indent=2))

3.We have a nested object. We would like a function where you pass in the object and a key and *get back the value.*

Example Inputs
object = {"a":{"b":{"c":"d"}}}
key = a/b/c
object = {"x":{"y":{"z":"a"}}}
key = x/y/z
value = a

Answer:

```
def exam_get_value(obj, key):

    keys = key.split('/')
    for k in keys:
        if k in obj:
            obj = obj[k]
        else:
            return None
    return obj

obj1 = {'a': {'b': {'c': 'd'}}}
key1 = 'a/b/c'
print(exam_get_value(obj1, key1))

obj2 = {'x': {'y': {'z': 'a'}}}
key2 = 'x/y/z'
print(exam_get_value(obj2, key2))
```