

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

With an established proficiency in Azure Cloud technologies, I have chosen Azure as the preferred platform for the implementation of the 3-tier architecture. Leveraging Azure's Cloud Shell affords me the advantage of rapid and streamlined resource creation.

Within my organizational context, Ansible serves as the orchestration tool for the seamless generation and management of Virtual Machines, aligning with our operational practices.

The envisioned 3-tier architecture comprises distinct layers. The Presentation Tier predominantly caters to web browsers on client machines, facilitating access to the targeted website. Subsequent to presentation, the Application Tier becomes pivotal, housing the core logic. This tier processes client inputs, executing intricate operations and orchestrating interactions with the Database Tier. The Database Tier serves as the repository for the application's data.

An integral initial step involves the establishment of a virtual network encompassing subnets for both the Application and Database Tiers. In addition, a dedicated client subnet has been configured to visually emphasize the presentation layer.

PS /home/shrikant> az group create --name kpmg-tst-rg --location eastus

```
PS /home/shrikant> az group create --name kpmg-tst-rg --location eastus
{
  "id": "/subscriptions/9c5017e9-6f33-4699-81c4-be51c4d8aecb/resourceGroups/kpmg-tst-rg",
  "location": "eastus",
  "managedBy": null,
  "name": "kpmg-tst-rg",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

```
PS /home/shrikant> az network vnet create --resource-group kpmg-tst-rg --name kpmg-tst-vnet --address-prefixes 10.0.0.0/22
```

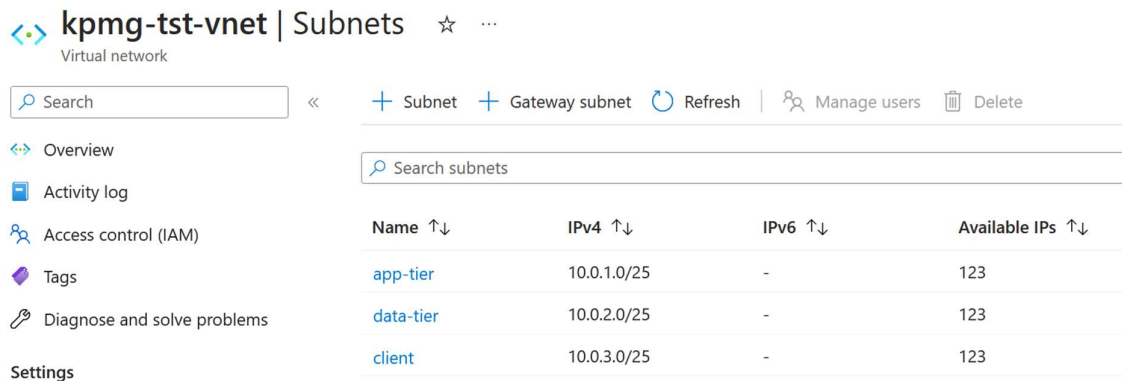
```
PS /home/shrikant> az network vnet create --resource-group kpmg-tst-rg --name kpmg-tst-vnet --address-prefixes 10.0.0.0/22
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
        "10.0.0.0/22"
      ]
    },
    "enableDdosProtection": false,
    "etag": "W/\\4762ef40-c9e1-41bf-99b2-1593bf00afee\\\"",
    "id": "/subscriptions/9c5017e9-6f33-4699-81c4-be51c4d8aeb/resourceGroups/kpmg-tst-rg/providers/Microsoft.Network/virtualNetworks/kpmg-tst-vnet",
    "location": "eastus",
    "name": "kpmg-tst-vnet",
    "provisioningState": "Succeeded",
    "resourceGroup": "kpmg-tst-rg",
    "resourceGuid": "7b4c4eaa-557c-4cb0-99d0-af6f8b32d936",
    "subnets": [],
    "type": "Microsoft.Network/virtualNetworks",
    "virtualNetworkPeerings": []
  }
}
```

Creating Subnets Under the VNET

```
az network vnet subnet create --resource-group kpmg-tst-rg --vnet-name kpmg-tst-vnet --name app-tier --address-prefixes 10.0.1.0/25
```




```
az network vnet subnet create --resource-group kpmg-tst-rg --vnet-name kpmg-tst-vnet --name data-tier --address-prefixes 10.0.2.0/25
```

```
az network vnet subnet create --resource-group kpmg-tst-rg --vnet-name kpmg-tst-vnet --name client --address-prefixes 10.0.3.0/25
```



Name	IPv4	IPv6	Available IPs
app-tier	10.0.1.0/25	-	123
data-tier	10.0.2.0/25	-	123
client	10.0.3.0/25	-	123

I've established Network Security Groups (NSGs) for each subnet, featuring a standardized set of rules. Specifically, a rule for Remote Desktop Protocol (RDP) on port 3389, a rule for Hypertext Transfer Protocol (HTTP) on port 80 for the app-tier NSG, and a rule for SQL port 1433 on the data-tier subnet have been implemented. These rules are designed for ingress traffic.

<input type="checkbox"/> Name ↑↓	Type ↑↓	Location ↑↓
<input type="checkbox"/>  app-tst-nsg	Network security group	East US
<input type="checkbox"/>  client-tst-nsg	Network security group	East US
<input type="checkbox"/>  db-tst-nsg	Network security group	East US


Associate the NSGs to the respective subnets

[+ Subnet](#)
[+ Gateway subnet](#)
[Refresh](#)
[Manage users](#)
[Delete](#)

<input type="text" value="Search subnets"/>					
Name ↑↓	IPv4 ↑↓	IPv6 ↑↓	Available IPs ↑↓	Delegated... ↑↓	Security group ↑↓
app-tier	10.0.1.0/25	-	123	-	app-tst-nsg
client	10.0.3.0/25	-	123	-	client-tst-nsg
data-tier	10.0.2.0/25	-	123	-	db-tst-nsg

Now for the application tier I will use Virtual Machine Scale Sets (VMSS) as its scalable and relevant solution to host application.

```
az vmss create --resource-group kpmg-tst-rg --name myvmss --image MicrosoftWindowsServer:WindowsServer:2022-datacenter-smalldisk:latest --instance-count 2 --admin-username syerge --admin-password <mypassword> --authentication-type password --vm-sku Standard_B2ms
```


myvmss | Instances
 ☆ ...

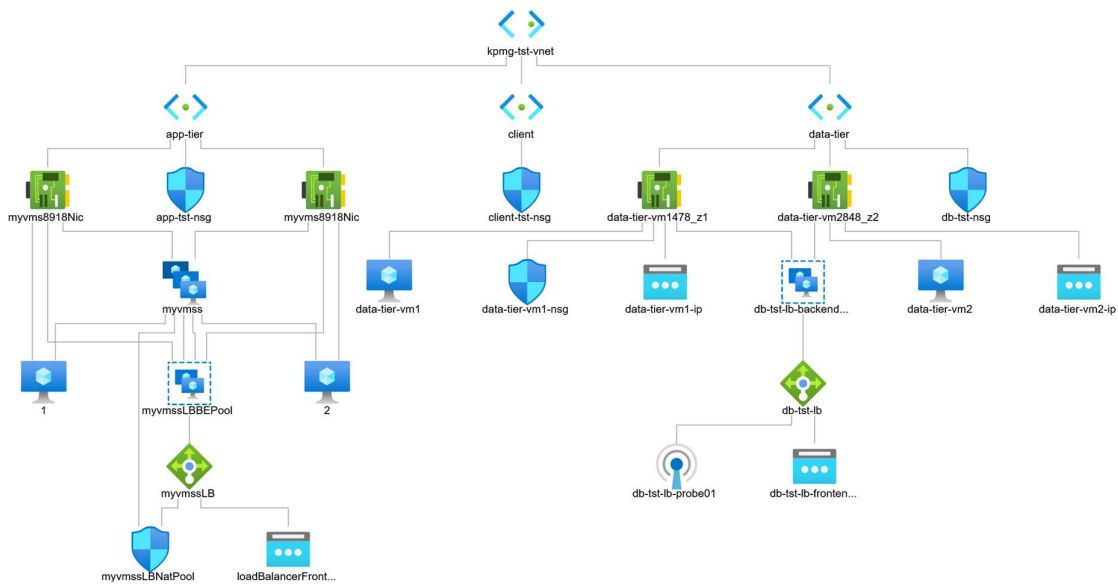
Virtual machine scale set

«
▶ Start
↺ Restart
□ Stop
↺ Reimage
🗑 Delete
↑ Upgrade
🔄 Refresh
🔒 Protection Policy

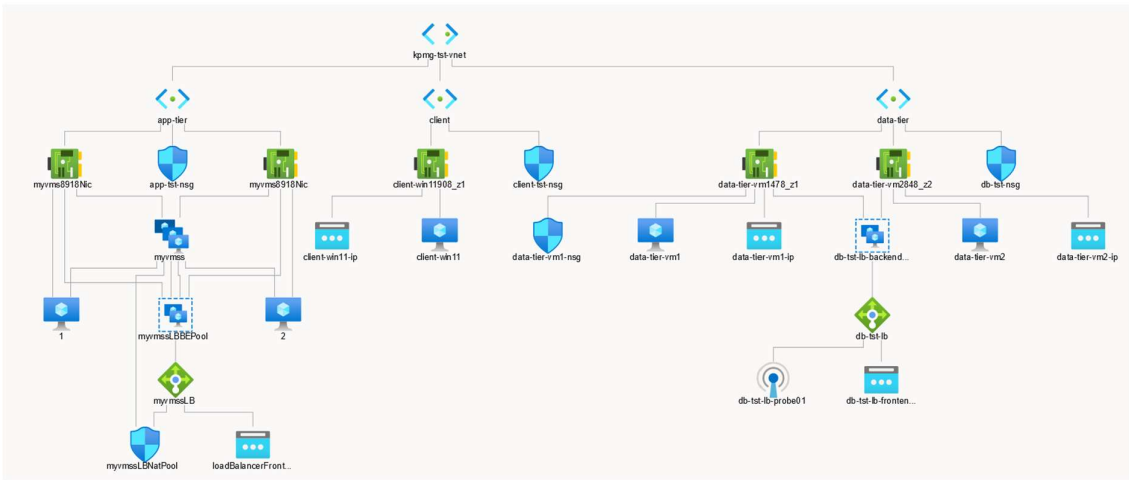
[Overview](#)
[Activity log](#)
[Access control \(IAM\)](#)
[Tags](#)
[Diagnose and solve problems](#)

Instance ↑↓	Computer name ↑↓	Status ↑↓	Protection policy ↑↓	Provisioning sta... ↑↓
<input type="checkbox"/> myvmss_1	myvms8918000001	🟢 Running		Succeeded
<input type="checkbox"/> myvmss_2	myvms8918000002	🟢 Running		Succeeded

With the application side now prepared, we can move forward to configure the Database tier. In this phase, I'm opting to create the Database tier VMs via the portal, as this allows me to explore any new options that have been incorporated into the portal form. It's a chance to discover and appreciate the enhancements firsthand! Additionally, I'm integrating a standard load balancer to effectively distribute traffic across the two database nodes. Consequently, the entire Database tier setup is now in place, culminating in the presentation of the comprehensive network topology.



I have created a client machine just to show the client side and here is the final figure.



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.

I intend to employ an Ansible playbook to retrieve the details of the Azure Virtual Machines. The relevant files have been uploaded under the 'Challenge2' directory. For security considerations, I have redacted links and IDs from the output.

Code: https://raw.githubusercontent.com/syerge/kpmg/main/Challenge2/vm_info.yaml

Output:

https://raw.githubusercontent.com/syerge/kpmg/main/Challenge2/vm_info_output.json
<https://github.com/syerge/kpmg/blob/main/Challenge2/Ansible%20output.png>

Challenge #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. The choice of language and implementation is up to you.

Code: <https://raw.githubusercontent.com/syerge/kpmg/main/Challenge3/NestedValue.ps1>

Output: <https://github.com/syerge/kpmg/blob/main/Challenge3/NestedValueOutput.png>