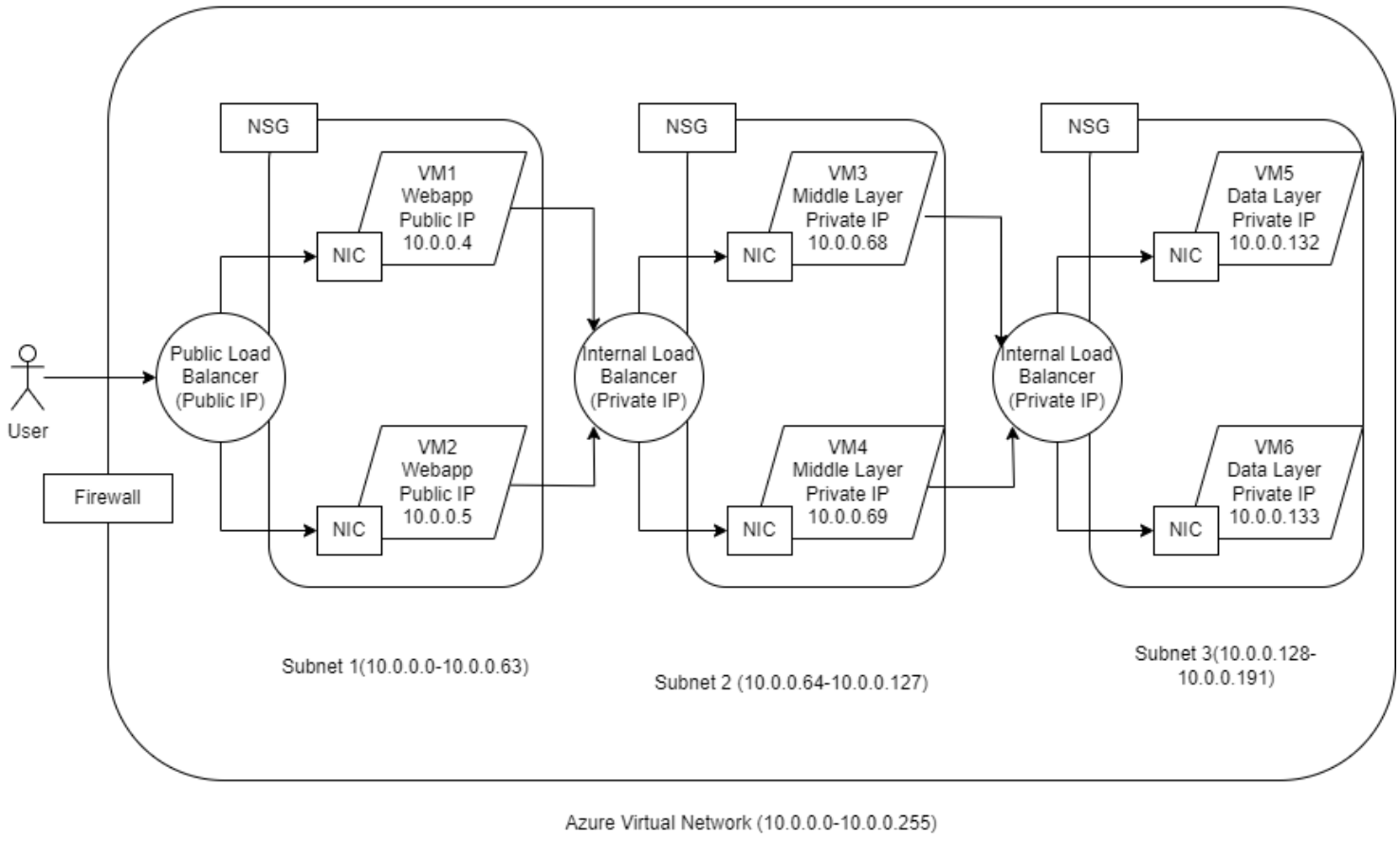
KPMG Challenges

Challenge 1 :

3 Tier Architecture in Azure :



The above diagram represents 3 Tier Architecture.

Steps :

1. Create Vnet named Challenge1Vnet; IPv4 CIDR Block : 10.0.0.0/24

2. Create Public Subnet named Subnet1;

3. Create 2 VMs VM1 and VM2 and attach these VMs with Subnet1 while creation.

4. Create Public Subnet named Subnet2;

5. Create 2 VMs VM3 and VM4 and attach these VMs with Subnet2 while creation.

6. Create Public Subnet named Subnet3;

7. Create 2 VMs VM5 and VM6 and attach these VMs with Subnet3 while creation.

8. Create a Public Load Balancer and add VM1 and VM2 in the Target group/ Backend pool so that it can distribute the traffic between these 2 VMs.

9. Create an internal Load Balancer and add VM3 and VM4 in the Target group/ Backend pool which will distribute the traffic within VM3 and VM4.

10. Create an internal Load Balancer and do the same for VM5 and VM6.

11. Create Firewall which will give security at the VNet level.

12. Create the NSG (Network Security Group) and attach it with all subnets. Open the necessary ports.(e.g. if there is any SQL server open the port 3306 and enable the port 80 or 8080 for any web server)

13. Create Internet Gateway and attach it with the Vnet.

14. Create Public Route Table and integrate it with Internet Gateway and Subnet.

15. Create private Route Table and attach it with NAT Gateway and attach it with Subnet 3.

16. You can Install SQL server in VM 5 and VM 6 according to requirement and build an application in the middle layer.

Challenge 2 :

In AWS, Metadata is the data about your instance that you can use to configure or manage the running instance.

Every instance has access to its own metadata using any HTTP client request, such as,
curl http://169.254.169.254/latest/meta-data

get_key.py :

```
from get_metadata import get_metadata

def gen_dict_extract(key, var):
    if hasattr(var, 'items'):
        for k, v in var.items():
            if k == key:
                yield v
            if isinstance(v, dict):
                for result in gen_dict_extract(key, v):
                    yield result
            elif isinstance(v, list):
                for d in v:
                    for result in gen_dict_extract(key, d):
                        yield result

def find_key(key):
    metadata = get_metadata()
    return list(gen_dict_extract(key, metadata))

if __name__ == '__main__':
    key = input("What key would you like to find?\n")
    print(find_key(key))
```

get.metadata.py :

```
import requests
import json
def expand_tree(url, arr):
    output = {}
    for item in arr:
        new_url = url + item
        r = requests.get(new_url)
        text = r.text
        if item[-1] == "/":
            list_of_values = r.text.splitlines()
            output[item[:-1]] = expand_tree(new_url, list_of_values)
        elif is_json(text):
            output[item] = json.loads(text)
        else:
            output[item] = text
    return output

def get_metadata():
    initial = ["meta-data/"]
    result = expand_tree(metadata_url, initial)
    return result

def get_metadata_json():
    metadata = get_metadata()
    metadata_json = json.dumps(metadata, indent=4, sort_keys=True)
    return metadata_json

def is_json(myjson):
    try:
        json.loads(myjson)
    except ValueError:
        return False
    return True

if __name__ == '__main__':
    print(get_metadata_json())
```

Challenge 3 :

//use all the packages according to the requirement.

```
namespace GetNestedValueByKey {
    class Program {
        static void Main(string[] args) {
            static object GetNestedValueByKey(JObject obj, string key) {
                var keys = key.Split('/');
                JToken temp = obj;
                foreach(var k in keys) {
                        temp = temp[k];
                        if (temp == null) return $ "Key {key} not exists";
                }
                return temp;
            }

            JObject obj = JObject.Parse("{\"a\":{\"b\":{\"c\":\"d\"}}}");
            var value = GetNestedValueByKey(obj, "a/b/c");
            Console.WriteLine(value);
            obj = JObject.Parse("{\"x\":{\"y\":{\"z\":\"a\"}}}");
            value = GetNestedValueByKey(obj, "x/y/z");
            Console.WriteLine(value);
        }
    }
}
```