Week 3 Assignment

To achieve the task of creating VNETs, subnets, and VMs in Microsoft Azure, you can follow these steps. The process involves using the Azure Portal, Azure CLI, or Azure PowerShell. I'll outline the steps using Azure CLI, as it can be easily executed in a script or a command line interface. This script will:

1. Create two Virtual Networks (VNETs) each with its own subnet.
2. Deploy a Windows VM in one subnet and a Linux VM in another subnet within each VNET.
3. Set up VNET peering to allow the VMs in different VNETs to communicate.

Here is a step-by-step Azure CLI script to accomplish the task:

Step 1: Set Variables

Set variables for resource names, locations, and other configurations.

```
resourceGroup="MyResourceGroup"
location="eastus"
vnet1="MyVnet1"
vnet2="MyVnet2"
subnet1="MySubnet1"
subnet2="MySubnet2"
subnet3="MySubnet3"
subnet4="MySubnet4"
nsg="MyNSG"
windowsVM="MyWindowsVM"
linuxVM="MyLinuxVM"
windowsVM2="MyWindowsVM2"
linuxVM2="MyLinuxVM2"
adminUsername="azureuser"
adminPassword="YourPassword123!"
```

Step 2: Create Resource Group

```
az group create --name $resourceGroup --location $location
```

Step 3: Create VNETs and Subnets

```
# Create VNET1 with two subnets
```

```
az network vnet create --resource-group $resourceGroup --name $vnet1 --address-prefix 10.0.0.0/16 \ --subnet-name $subnet1 --subnet-prefix 10.0.1.0/24
az network vnet subnet create --resource-group $resourceGroup --vnet-name $vnet1 \ --name $subnet2 --address-prefix 10.0.2.0/24
```

```
# Create VNET2 with two subnets
```

```
az network vnet create --resource-group $resourceGroup --name $vnet2 --address-prefix 10.1.0.0/16 \ --subnet-name $subnet3 --subnet-prefix 10.1.1.0/24
```

```
az network vnet subnet create --resource-group $resourceGroup --vnet-name $vnet2 \ --name $subnet4 --address-prefix 10.1.2.0/24
```

Step 4: Create Network Security Group (NSG)

```
# Create a network security group
```

```
az network nsg create --resource-group $resourceGroup --name $nsg
```

```
# Create NSG rules to allow RDP, SSH, and ICMP (ping)
```

```
az network nsg rule create --resource-group $resourceGroup --nsg-name $nsg --name AllowRDP --protocol Tcp --direction Inbound --priority 1000 --source-address-prefixes '*' --source-port-ranges '*' --destination-address-prefixes '*' --destination-port-ranges 3389 --access Allow
```

```
az network nsg rule create --resource-group $resourceGroup --nsg-name $nsg --name AllowSSH --protocol Tcp --direction Inbound --priority 1001 --source-address-prefixes '*' --source-port-ranges '*' --destination-address-prefixes '*' --destination-port-ranges 22 --access Allow
```

```
az network nsg rule create --resource-group $resourceGroup --nsg-name $nsg --name AllowICMP --protocol Icmp --direction Inbound --priority 1002 --source-address-prefixes '*' --source-port-ranges '*' --destination-address-prefixes '*' --destination-port-ranges '*' --access Allow
```

Step 5: Associate NSG with Subnets

```
# Associate NSG with subnets
```

```
az network vnet subnet update --resource-group $resourceGroup --vnet-name $vnet1 --name $subnet1 --network-security-group $nsg
```

```
az network vnet subnet update --resource-group $resourceGroup --vnet-name $vnet1 --name $subnet2 --network-security-group $nsg
```

```
az network vnet subnet update --resource-group $resourceGroup --vnet-name $vnet2 --name $subnet3 --network-security-group $nsg
```

```
az network vnet subnet update --resource-group $resourceGroup --vnet-name $vnet2 --name $subnet4 --network-security-group $nsg
```

Step 6: Create VMs

```
# Create a Windows VM in VNET1/Subnet1

az vm create --resource-group $resourceGroup --name $windowsVM --image Win2019Datacenter --admin-username $adminUsername --admin-password $adminPassword --vnet-name $vnet1 --subnet $subnet1 --nsg $nsg --public-ip-address ""

# Create a Linux VM in VNET1/Subnet2

az vm create --resource-group $resourceGroup --name $linuxVM --image UbuntuLTS --admin-username $adminUsername --admin-password $adminPassword --vnet-name $vnet1 --subnet $subnet2 --nsg $nsg --public-ip-address ""

# Create a Windows VM in VNET2/Subnet3

az vm create --resource-group $resourceGroup --name $windowsVM2 --image Win2019Datacenter --admin-username $adminUsername --admin-password $adminPassword --vnet-name $vnet2 --subnet $subnet3 --nsg $nsg --public-ip-address ""

# Create a Linux VM in VNET2/Subnet4

az vm create --resource-group $resourceGroup --name $linuxVM2 --image UbuntuLTS --admin-username $adminUsername --admin-password $adminPassword --vnet-name $vnet2 --subnet $subnet4 --nsg $nsg --public-ip-address ""
```

Step 7: Set Up VNET Peering

```
# Create peering from VNET1 to VNET2

az network vnet peering create --resource-group $resourceGroup --name Vnet1ToVnet2 --vnet-name $vnet1 --remote-vnet $vnet2 --allow-vnet-access

# Create peering from VNET2 to VNET1

az network vnet peering create --resource-group $resourceGroup --name Vnet2ToVnet1 --vnet-name $vnet2 --remote-vnet $vnet1 --allow-vnet-access
```

Step 8: Test Connectivity

After the VMs are created and the VNET peering is set up, you can test connectivity by pinging between the VMs. Use the Azure portal or connect via RDP/SSH to each VM and ping the private IP addresses of the other VMs.

Summary

This script creates two VNETs, each with two subnets, deploys a Windows VM and a Linux VM in each VNET, and sets up VNET peering between the two VNETs. This allows VMs in different VNETs to communicate with each other. Make sure to replace placeholder values with your specific information and ensure passwords meet Azure's requirements.