-Om Wabale

Azure Virtual Network (VNet), Subnets and Peering – Deep Dive

Azure Virtual Network (VNet) is a **software-defined private network** in the cloud. Microsoft describes a VNet as "the fundamental building block for your private network in Azure". In effect, a VNet is analogous to an on-premises network: you define a private IP address space (via CIDR notation), segment it into subnets, and attach Azure resources (like VMs) to those subnets. Resources in the same VNet (or peered VNets) can communicate over their private IPs. Azure routes traffic between subnets and VNets by default, and provides built-in isolation and routing. You can connect VNets in the same region or across regions; this is called **VNet peering** (more below).

- **Address space & CIDR**: Each VNet must be created with an IPv4 CIDR block (e.g. 10.0.0.0/16) that defines its address range. The Azure portal even shows you how many addresses a prefix yields. For example, a /16 block provides 65,536 addresses. The smallest allowed subnet is /29 (8 IPs). Remember that Azure *reserves* 5 IPs in every subnet (network ID, broadcast, and three internal addresses), so a /29 yields only 3 usable addresses. Azure VNets support sizes from /16 down to /29, and **subnets cannot overlap** other VNets' address ranges.
- **Subnets**: A VNet is subdivided into one or more subnets. Each subnet is a contiguous range within the VNet's address space. When you create a VNet in the portal, a default subnet (often named default) is automatically created. You can add or adjust subnets as needed. Subnets are where you place resources: for example, you might have one subnet for Windows servers and another for Linux servers, or public and private subnets in a hub/spoke architecture. Subnet sizes (CIDR prefixes) should be chosen based on expected number of hosts; common practice is to use a /24 (256 addresses) by default, which gives 251 usable IPs (256 minus 5 reserved). Subnets can be public (with internet connectivity) or private; you can also designate a special "GatewaySubnet" when using VPN/ExpressRoute. Network Security Groups (NSGs) are typically associated at the subnet or NIC level to filter traffic.
- **VNet Peering**: Peering enables direct, private IP connectivity between two VNets. Azure supports peering **within a region** (same-region peering) or **across regions** (Global VNet peering). In either case the VMs in peered VNets can communicate as if on the same network. A key requirement is that the two VNets' address ranges must *not overlap*. Peering is non-transitive: traffic flows directly between the two peer VNets. Portal or CLI creation of a peering link is straightforward (see steps below). By default, Azure enables bidirectional communication when you allow "VNet access" in the peering settings. Advanced options (gateway transit, forwarded traffic) are available but optional. In summary, use peering to make two VNets act like one network for VM-to-VM traffic.

Prerequisites

- **Azure Subscription & Resources**: You need an active Azure subscription. You will create resources in a Resource Group (RG). If you don't have one, create an RG (via portal, CLI `az group create`, or PowerShell `New-AzResourceGroup`) in a chosen region.
- **Region**: We'll use a region "close to Mumbai" as requested. Azure's **Indian regions** include *Central India* (Pune), *South India* (Chennai), and *West India* (Mumbai). For example, selecting **Central India** or **South India** as the region when creating VNets/VMs is appropriate for geographic proximity. (These region names appear in Azure CLI/portal as `centralindia`, `southindia`, etc.)
- **Tools**: You can use the Azure Portal GUI *and* Azure CLI (or PowerShell) interchangeably. Azure Cloud Shell (browser-based CLI) is also convenient. Ensure Azure CLI version ≥2.0.28 or Azure PowerShell module ≥5.4.1.

Creating VNets and Subnets (Portal and CLI/PowerShell)

Below is a walkthrough of creating a VNet with two subnets, then deploying VMs into them. We cover both Azure Portal steps (with illustrative screenshots) and equivalent CLI/PowerShell commands. All examples assume the RG "test-rg" (or another of your choice) and location **centralindia** (Central India).

1. Create a Virtual Network

- **Portal** – In the Azure portal, search for **Virtual networks** and click **+ Create**. In the **Basics** tab, fill in project details (select your subscription and either an existing RG or "Create new" – e.g. `test-rg`). Enter a **Name** for the VNet (e.g. `vnet-1`). Choose **Region** = Central India (or South India).

  Under **IP Addresses**, accept the default address space of `10.0.0.0/16` (or specify your chosen CIDR). This gives ~65,536 addresses. Click **Review + create** and then **Create** to provision the VNet. By default Azure adds a subnet named `default` covering `10.0.0.0/24`.

- **CLI** – Use `az network vnet create` with the address prefix and an initial subnet.

  For example:

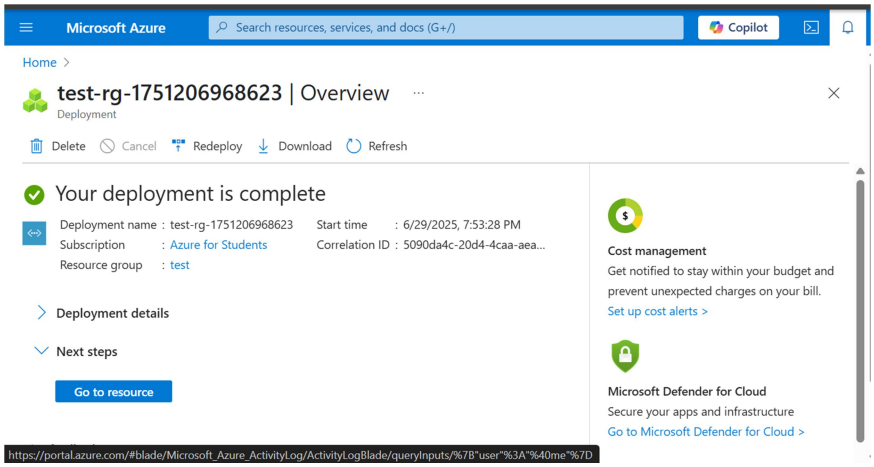
```
az network vnet create -g test-rg -n vnet-1 --address-prefixes 10.0.0.0/16 \
   --subnet-name subnet1 --subnet-prefixes 10.0.0.0/24 --location centralindia
```

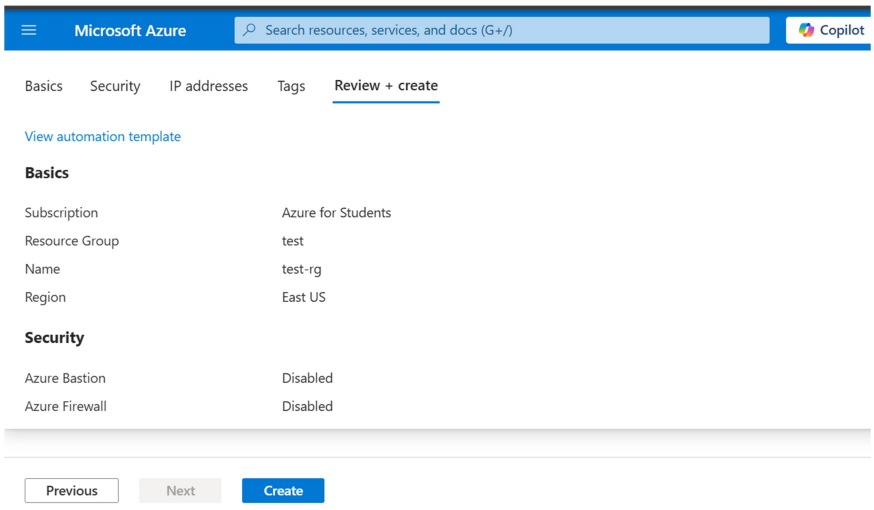
  This creates VNet `vnet-1` in resource group `test-rg` with address space `10.0.0.0/16` and a subnet `subnet1` = `10.0.0.0/24`. (Azure CLI docs show a similar example of creating a VNet with a subnet in one command

- **PowerShell** – Alternatively, use Azure PowerShell:

```
$vnet = @{
    Name              = 'vnet-1'
    ResourceGroupName = 'test-rg'
    Location          = 'centralindia'
    AddressPrefix     = '10.0.0.0/16'
}
$virtualNetwork = New-AzVirtualNetwork @vnet
$subnetConfig = Add-AzVirtualNetworkSubnetConfig -Name 'subnet1' `
    -VirtualNetwork $virtualNetwork -AddressPrefix '10.0.0.0/24'
$virtualNetwork | Set-AzVirtualNetwork
```

This creates the VNet and adds a subnet subnet1 of size /24.





2. Create Additional Subnets

You may want two subnets (e.g. one for Windows VMs, one for Linux VMs). If you only used the portal's default subnet during VNet creation, add a second one now.

- **Portal** – In the VNet's **Subnets** blade, click **+ Subnet**. Give it a name (e.g. subnet2), and set the **Subnet address range** (e.g. 10.0.1.0/24). This must be inside the VNet's 10.0.0.0/16 range and not overlap the other subnet. Click **Add** to save.

- **CLI** – You can add subnets with:

```
az network vnet subnet create -g test-rg --vnet-name vnet-1 \
  --name subnet2 --address-prefix 10.0.1.0/24
```

- **PowerShell** – Similarly:

```
Add-AzVirtualNetworkSubnetConfig -Name 'subnet2' `
  -VirtualNetwork $virtualNetwork -AddressPrefix '10.0.1.0/24' |
  Set-AzVirtualNetwork
```

Now the VNet vnet-1 has two subnets (10.0.0.0/24 and 10.0.1.0/24). By default, NSGs (network security groups) allow all intra-VNet traffic, so VMs in these subnets can talk to each other unless you explicitly add rules.

Launching Virtual Machines

We will create **one Windows VM** in subnet1 and **one Linux VM** in subnet2. This tests cross-subnet connectivity.

3. Create a Windows VM in Subnet1

- **Portal** – Search for **Virtual machines** and select **+ Create > Azure virtual machine**. In the **Basics** tab, choose the same **Resource Group** (test-rg). Enter a VM name (e.g. win-vm). Choose **Region = Central India** to match the VNet. Under **Image**, pick a Windows Server image (e.g. *Windows Server 2022 Datacenter*). Choose a VM size (for example, **Standard_B1ms** or **D2s_v3**) – a small size is sufficient for testing.

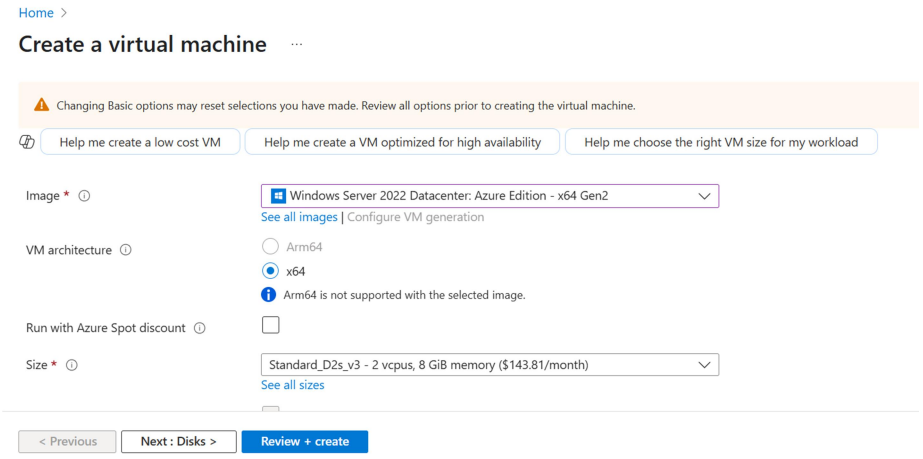
  Set up **Administrator account** (e.g. username azureuser and a strong password). Under **Inbound port rules**, you can choose **None** or just allow **RDP (3389)** so you can remote into the Windows VM if desired. Then go to the **Networking** tab: select **Virtual network = vnet-1** and **Subnet = subnet1**. For this exercise you do not need a Public IP (the VMs will communicate internally), so you may set "Public IP = None" if you want. Finally click **Review + create** and then **Create**. This deploys win-vm into subnet1 of vnet-1.

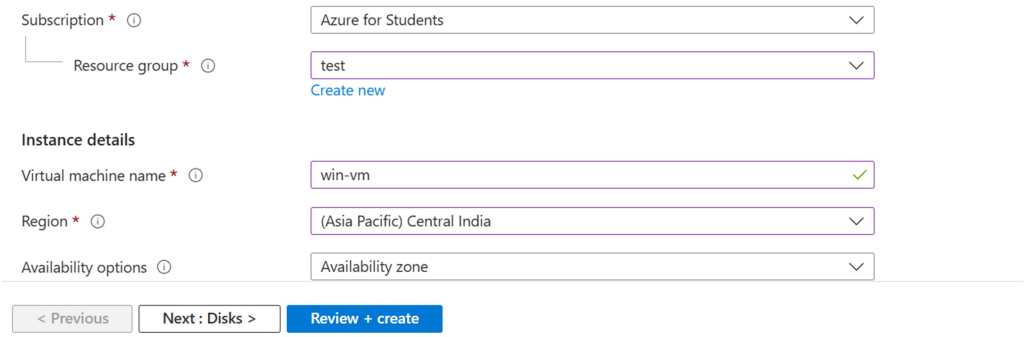
- **CLI** – Using Azure CLI, you can run (example for Windows):

```
az vm create -g test-rg -n win-vm \
  --image Win2022Datacenter \
  --admin-username azureuser --admin-password 'MyP@ssw0rd123!' \
  --vnet-name vnet-1 --subnet subnet1 \
  --no-wait
```

  This creates a Windows VM named win-vm attached to subnet1. (Use az vm open-port if you need to open RDP/HTTPS etc.) For a Linux VM, you would use --image Ubuntu2204 and --authentication-type ssh or --generate-ssh-keys. The key point is specifying --vnet-name and --subnet as above.

- **PowerShell** – With Az module, you could similarly use New-AzVM with -VirtualNetwork and -SubnetName parameters. The Azure docs provide an example for PowerShell VM creation that deploys into an existing subnet (using the same pattern as CLI above).





Home >

Create a virtual machine ...

Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.

Help me create a low cost VM | Help me create a VM optimized for high availability | Help me choose the right VM size for my workload

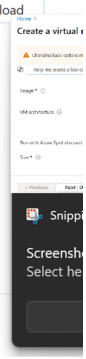
Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ⓘ
- ○ None
- ◉ Allow selected ports

Select inbound ports * RDP (3389)

This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

< Previous | Next : Disks > | Review + create



Home >

Create a virtual machine ...

Help me create a low cost VM | Help me create a VM optimized for high availability | Help me choose the right VM size for my workload

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * ⓘ (new) win-vm-vnet
Create new

Subnet * ⓘ (new) default (10.0.0.0/24)

Public IP ⓘ (new) win-vm-ip
Create new

NIC network security group ⓘ
- ○ None
- ◉ Basic
- ○ Advanced

Home >

Create a virtual machine ...

Validation passed

Help me create a low cost VM | Help me create a VM optimized for high availability | Help me choose the right VM size for my workload

Basics | Disks | Networking | Management | Monitoring | Advanced | Tags | **Review + create**

Price

1 X Standard D2s v3
by Microsoft
Terms of use | Privacy policy

Subscription credits apply ⓘ
0.1970 USD/hr
Pricing for other VM sizes

TERMS

4. Create a Linux VM in Subnet2

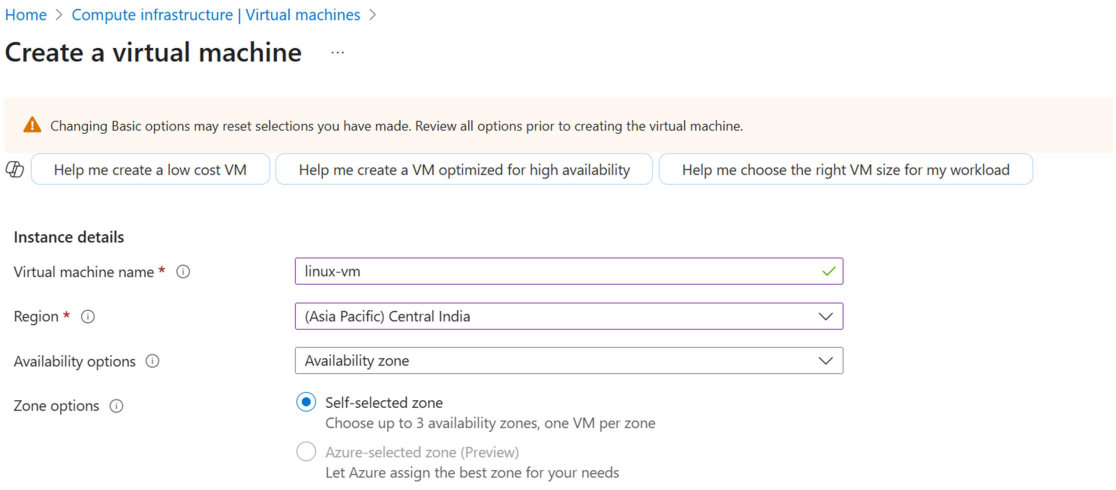
Repeat similar steps for a Linux VM:

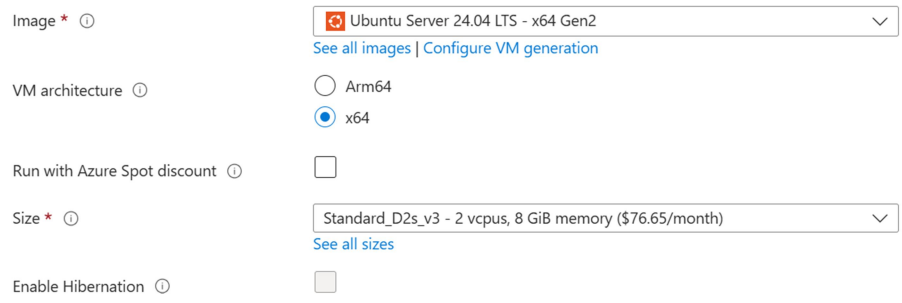
- **Portal** – **+ Create > Virtual machine**. Fill **Instance details**: name linux-vm, Resource Group test-rg, Region = Central India, **Image** = *Ubuntu Server 22.04 LTS*. Size = e.g. **Standard_B1ms**. Under **Authentication type**, choose SSH public key (or password). Then **Networking**: Virtual network = vnet-1, Subnet = subnet2. Inbound ports: allow SSH (22) if you want to SSH in. Click **Review + create** then **Create**.

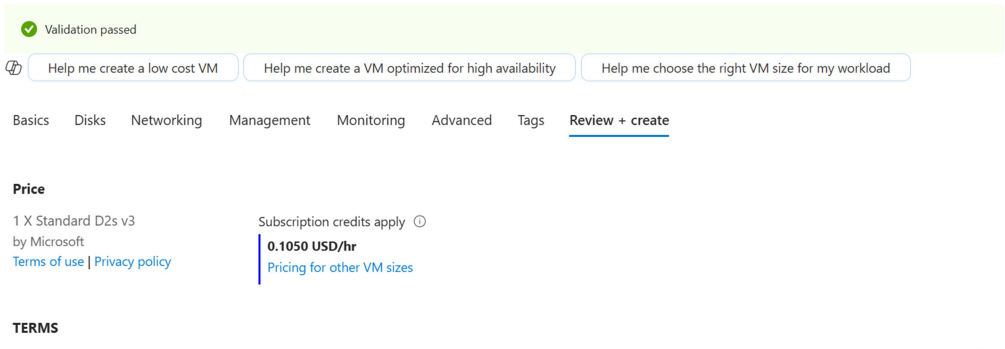
- **CLI** – For example:

```
az vm create -g test-rg -n linux-vm \
  --image Ubuntu2204 \
  --admin-username azureuser \
  --generate-ssh-keys \
  --vnet-name vnet-1 --subnet subnet2
```

This sets up a Linux VM linux-vm in subnet2







Once both VMs are running, each will have a private IP (e.g. 10.0.0.4 and 10.0.1.4). By default, intra-VNet traffic is allowed. You can log in (SSH or RDP) to each VM and test connectivity:

- **Ping test (intra-VNet)**: From **win-vm**, open Command Prompt/PowerShell and ping 10.0.1.x (the Linux VM's IP). From **linux-vm**, run ping 10.0.0.x. Azure network-level defaults allow this if no NSG or firewall blocks it. (On Windows, the OS firewall may block ICMP by default; in that case run netsh advfirewall firewall add rule name="Allow ICMP" protocol=icmpv4:8,any dir=in action=allow to permit ping.) As Microsoft's documentation notes, "you should be able to ping a private address range as long as there is a path available and there are no NSGs or OS-level firewalls blocking".

If pings do not work, verify that: a) the VMs indeed have IPs in the correct subnets; b) no Network Security Group rule is blocking ICMP/SSH/RDP. You can attach an NSG to a subnet or NIC via Azure Portal or CLI (az network nsg rule etc.) if fine-grained control is needed.

VNet Peering (Connect Two VNets)

Now create a second VNet (vnet-2) and peer it with vnet-1 so that their VMs can also communicate across VNets.

5. Create a Second VNet and Subnet

Just as before, make another VNet in the same region:

- **Portal** – **Virtual networks > + Create**. Name it vnet-2, Resource Group test-rg, Region = Central India, Address space (e.g. 10.1.0.0/16). In IP Addresses, use default subnet subnet1 = 10.1.0.0/24. Create it.

- **CLI** –

```
az network vnet create -g test-rg -n vnet-2 \
  --address-prefixes 10.1.0.0/16 \
  --subnet-name subnet1 --subnet-prefixes 10.1.0.0/24 \
  --location centralindia
```

6. Configure VNet Peering

Portal – In the Azure portal, go to **Virtual networks**, select **vnet-1**, then **Peerings** (under Settings) and click **+ Add**. Fill in:

- **Name**: e.g. vnet1-to-vnet2.
- **Virtual network deployment model**: (leave default Resource Manager).
- **Remote virtual network**: Select Subscription and then vnet-2.
- **Peering settings**: Ensure **Allow virtual network access** (the default checkbox) is checked for both directions. This enables VMs in vnet-2 to talk to vnet-1 and vice versa. Then click **Add**.

Repeat in **vnet-2**: Peerings > +Add, Name vnet2-to-vnet1, Remote VNet = vnet-1, allow access. The portal peering configuration page looks like this:

CLI – Use `az network vnet peering create` twice (once for each direction):

```
# Peer vnet-1 to vnet-2
az network vnet peering create -g test-rg \
   -n vnet1-to-vnet2 \
   --vnet-name vnet-1 \
   --remote-vnet /subscriptions/<subId>/resourceGroups/test-rg/providers/Microsoft.Network/virtualNetworks/vnet-
2 \
   --allow-vnet-access

# Peer vnet-2 to vnet-1
az network vnet peering create -g test-rg \
   -n vnet2-to-vnet1 \
   --vnet-name vnet-2 \
   --remote-vnet /subscriptions/<subId>/resourceGroups/test-rg/providers/Microsoft.Network/virtualNetworks/vnet-
1 \
   --allow-vnet-access
```

(If both VNets are in the same subscription, you can often just use `--remote-vnet vnet-2` by name; otherwise use the full resource ID as above.) The `--allow-vnet-access` flag permits VM-to-VM traffic through the peering.

PowerShell – Similarly, one would run:

```
$peer1 = @{
   Name                = 'vnet1-to-vnet2'
   VirtualNetwork      = (Get-AzVirtualNetwork -Name 'vnet-1' -ResourceGroupName 'test-rg')
   RemoteVirtualNetworkId = (Get-AzVirtualNetwork -Name 'vnet-2' -ResourceGroupName 'test-rg').Id
}
Add-AzVirtualNetworkPeering @peer1

$peer2 = @{
   Name                = 'vnet2-to-vnet1'
   VirtualNetwork      = (Get-AzVirtualNetwork -Name 'vnet-2' -ResourceGroupName 'test-rg')
   RemoteVirtualNetworkId = (Get-AzVirtualNetwork -Name 'vnet-1' -ResourceGroupName 'test-rg').Id
}
Add-AzVirtualNetworkPeering @peer2
```

This creates reciprocal peerings. After creation, the status should show as **Connected** (you can click Refresh after a few seconds as per docs).

7. Verify Peering Connectivity

With peering in place, try pinging the VMs across VNets. From `win-vm` (in vnet-1), ping the Linux VM's IP (in vnet-2). From `linux-vm`, ping `win-vm`'s IP. Because "Allow VNet access" was enabled on both sides, the traffic is allowed. As before, ensure OS firewalls permit ICMP. If needed, use `Test-NetConnection -ComputerName <IP>` on Windows or `ping` on Linux.

Note: No extra routing or gateways are needed; Azure handles the peering route automatically. The private IPs of the VMs should now be reachable just like same-VNet communication. (If

you had created a Network Virtual Appliance (NVA) or VPN gateway, you could configure gateway transit or use remote gateways, but that is beyond this basic scenario.)

Summary

In summary, we defined two Azure VNets with non-overlapping CIDR blocks, created subnets within them, and deployed one Windows and one Linux VM into separate subnets. By default, VMs in the same VNet could ping each other using their private IPs. We then peered the two VNets (in the same Azure region), enabling cross-VNet VM connectivity. All steps were shown using both the Azure Portal (with screenshots) and equivalent Azure CLI/PowerShell commands. Key points:

- **CIDR/Address Space**: Define VNet ranges in CIDR; each subnet is a subset of that range. Azure reserves 5 IPs per subnet.
- **Subnets**: Segment the VNet; assign VMs to subnets. Configure NSGs as needed.
- **Peering**: Connect VNets (same-region or global) so resources can talk privately. The peering requires no overlapping IP ranges and is created twice (bidirectional) if using CLI/PS.
- **Testing**: VMs use their private IPs. Ping should succeed if firewalls permit.

By following these steps, one can build a multi-VNet Azure network topology and deploy VMs with full connectivity across subnets and VNets.

Sources: Microsoft Azure documentation (portal walkthroughs and CLI/PowerShell references) and Azure networking blogs were used to confirm CIDR, subnet, and peering details. The screenshots are from Azure docs showing the portal UI.