A **Virtual Network (VNet)** in Azure is a **logical isolation of the Azure cloud** dedicated to your subscription.
Think of it as a virtual version of your on-premises network but hosted in Microsoft's global cloud infrastructure.

Key points about VNets:

- They enable **communication** between Azure resources using **private IPs**.
- They can be segmented into multiple **subnets** for better isolation and control.
- You can connect VNets to each other (via **peering**) or to on-premises networks (via VPNs).
- You can configure **security rules** to control traffic flow.

Example real-world use case:
A company hosts its web servers and database servers in different subnets within the same VNet. This helps apply fine-grained access policies between them.

Understanding CIDR Ranges

CIDR (Classless Inter-Domain Routing) is a method for allocating IP address blocks more efficiently than the old class-based system.

When creating a VNet, you define an **address space** using CIDR notation, such as `10.0.0.0/16`.
This means:

- The network starts at `10.0.0.0`.
- `/16` indicates the subnet mask (255.255.0.0), allowing about **65,536 addresses**.

Subnets are smaller address blocks carved from the VNet's space.
For example:

- `Subnet-1`: `10.0.1.0/24` → 256 addresses.
- `Subnet-2`: `10.0.2.0/24` → 256 addresses.

Important:

- Subnet CIDRs must fit within the parent VNet's CIDR range.

- Subnets must not overlap.
- Plan subnets to reserve space for future growth.

Planning VNet & Subnet Address Spaces

Before creating a VNet, always plan:

1. **Total IP address space needed:**
   Estimate the number of VMs, services, and future growth.
2. **Subnets:**
   Divide the VNet logically — for example:
   - **Frontend subnet** for web servers.
   - **Backend subnet** for databases.
   - **Management subnet** for admin tasks.
3. **Avoid Overlaps:**
   When using multiple VNets with peering, each must have unique CIDR ranges to prevent conflicts.
4. **Public vs Private IPs:**
   VMs typically get a private IP inside the VNet.

For admin access, you may assign public IPs temporarily.

Example Plan:

- **VNet-1:** `10.0.0.0/16`
  - Subnet-A: `10.0.1.0/24`
  - Subnet-B: `10.0.2.0/24`
- **VNet-2:** `10.1.0.0/16`
  - Subnet-A: `10.1.1.0/24`

What is VNet Peering?

VNet Peering allows two VNets to connect and communicate as if they are one.
It is Microsoft Azure's way of creating a private link between separate networks.

Features:

- Secure — uses Azure's backbone network.
- Low latency — no VPN or extra hardware needed.
- Resources in either VNet communicate using **private IPs**, not public Internet.
- You can peer VNets **within the same region** or **across regions**.

Why Peering is used:

- For multi-tier applications split across VNets.
- For large organizations with multiple departments using separate VNets.
- For secure data replication between regions.

Types of VNet Peering

Azure supports **two main types** of VNet Peering:

Intra-region Peering

- Connects VNets within the same Azure region.
- Example: Two VNets in East US region.

Global VNet Peering

- Connects VNets in different Azure regions.
- Example: A VNet in East US peered with one in West Europe.
- Useful for geo-redundancy and cross-region failover.

Key Points:

- Both VNets must have non-overlapping IP ranges.
- You can allow or disallow forwarded traffic.
- Peering is **non-transitive** by default (VMs in VNet-A peered with VNet-B cannot reach VNet-C automatically).

Prerequisites & Setup

Before implementing the use case, ensure:

1. **Azure Account** — Active subscription with enough free credits.
2. **Azure Portal Access** — via https://portal.azure.com
3. **Basic Tools** — Remote Desktop (for Windows VM) & SSH client like PuTTY or Windows Terminal (for Linux VM).
4. **Resource Group** — Logical container to hold related Azure resources.
5. **Proper permissions** — Account must have contributor or owner rights to create VNets, VMs, and NSGs.
6. **Screenshots Plan** — Keep a folder ready to store images at each step for your final report.

Use Case — Goals & Architecture

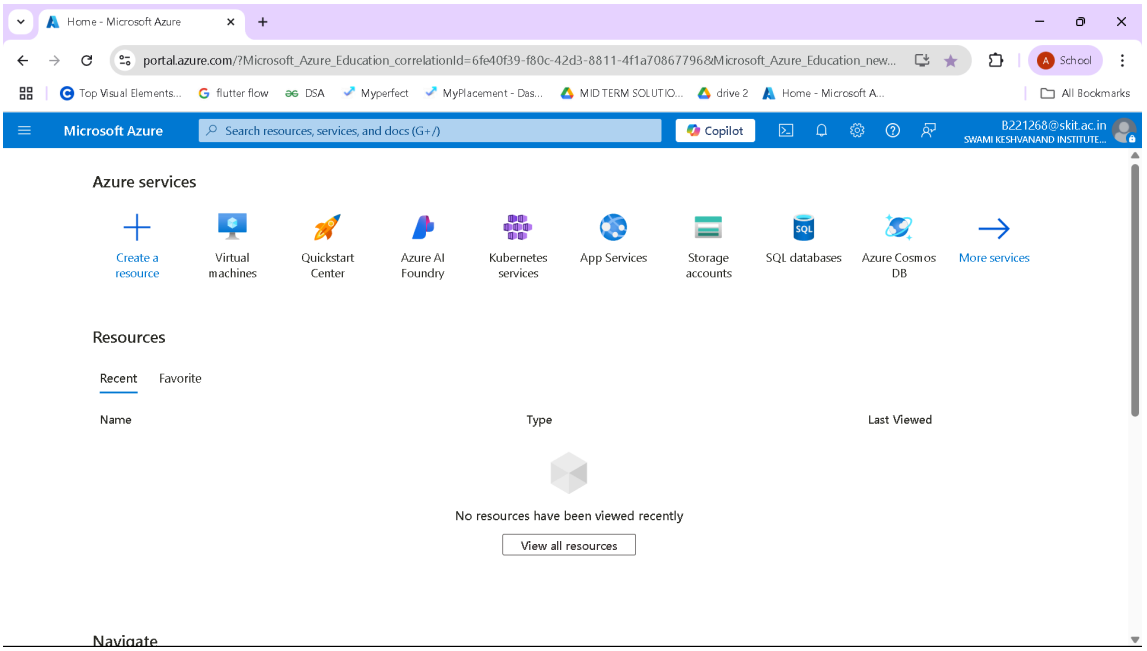
Objective:

1. Deploy `VNet-1` with **two subnets** — one for Windows VM and one for Linux VM.
2. Launch Windows VM in Subnet-1 and Linux VM in Subnet-2.

3. Ensure VMs can **ping** each other via private IPs.
4. Deploy `VNet-2` with its own subnet.
5. Peer `VNet-1` and `VNet-2` to enable communication between their resources.

Architecture Diagram

- `VNet-1` → Subnet-1 → Windows VM
- `VNet-1` → Subnet-2 → Linux VM
- `VNet-2` → Subnet-1 → Test VM (optional)
- Peering arrows between `VNet-1` & `VNet-2`

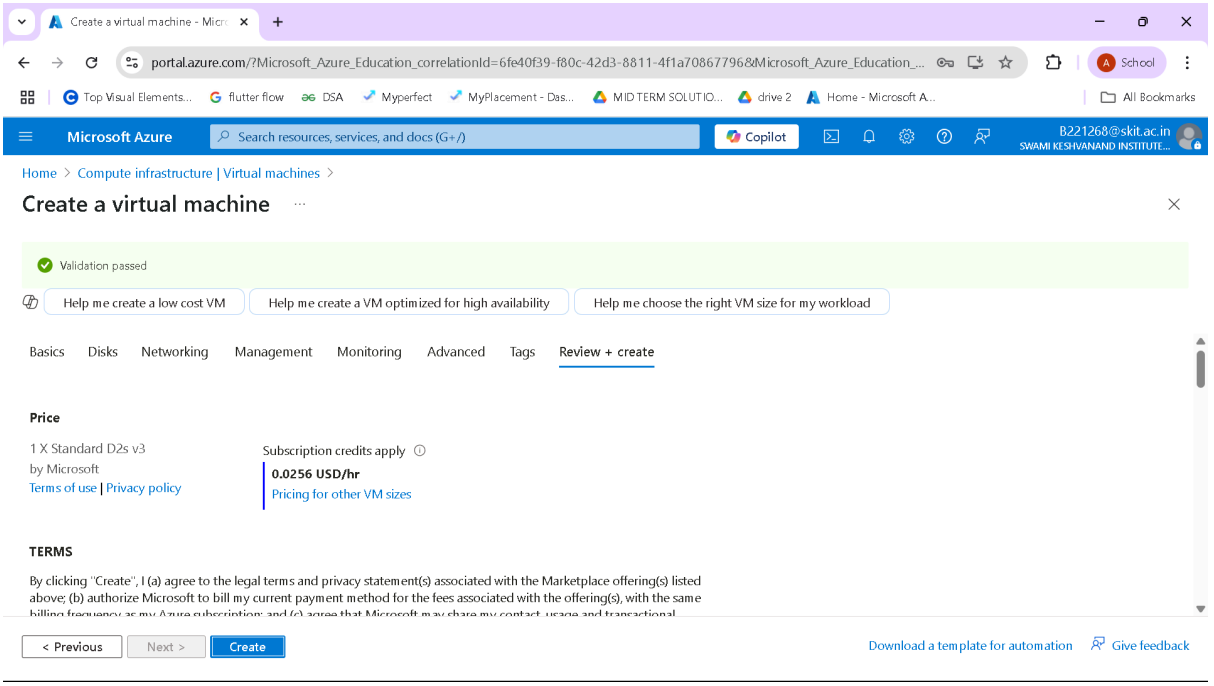


Implementation — Create First VNet & Subnets

Steps:

1. In **Azure Portal**, search **Virtual Networks** → Click **Create**.
2. Select or create a **Resource Group**, e.g., `MyResourceGroup`.
3. Give your VNet a clear name: `VNet-1`.
4. Choose **Region**: e.g., East US.
5. Set **Address Space** to `10.0.0.0/16`.
6. Under **Subnets**, add:
    - `Subnet-1`: `10.0.1.0/24` (for Windows VM)
    - `Subnet-2`: `10.0.2.0/24` (for Linux VM)

7. Click **Review + Create**, then **Create**.



Implementation — Deploy Windows VM

Steps:

1. Go to **Virtual Machines → Create**.
2. Select the **same Resource Group**.
3. VM Name: `Windows-VM`.
4. Region: Same as VNet-1.
5. Image: **Windows Server 2022 Datacenter**.

6. Size: Small size (`Standard_B1s` for demo).

7. Set admin **Username & Password** — note them securely.

8. Under **Networking**:

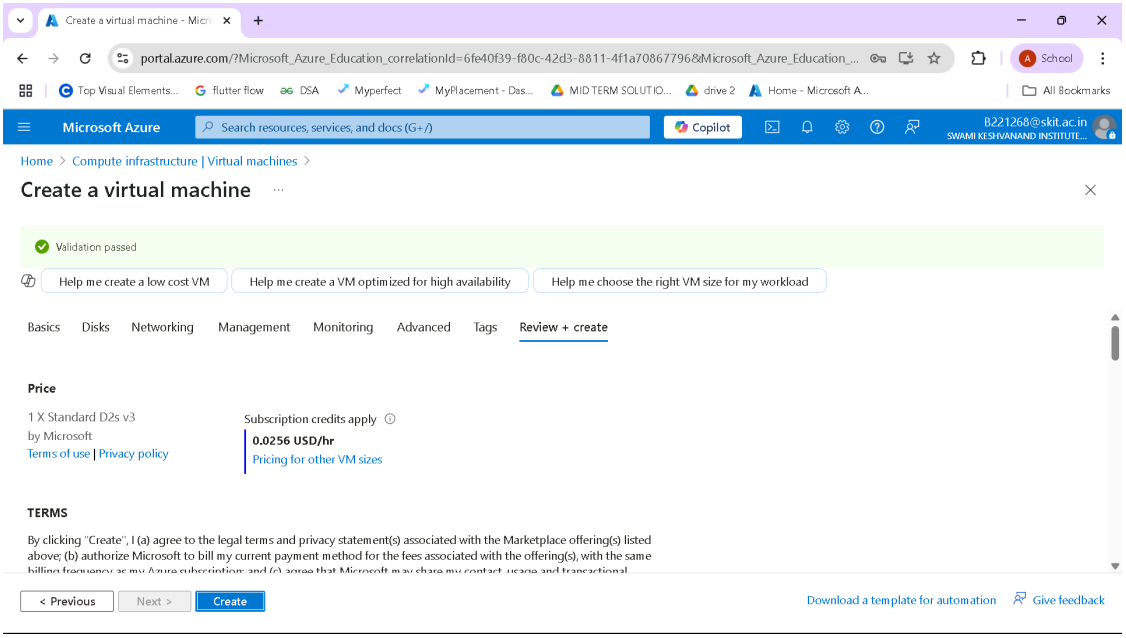
    - Virtual Network: `VNet-1`

    - Subnet: `Subnet-1`

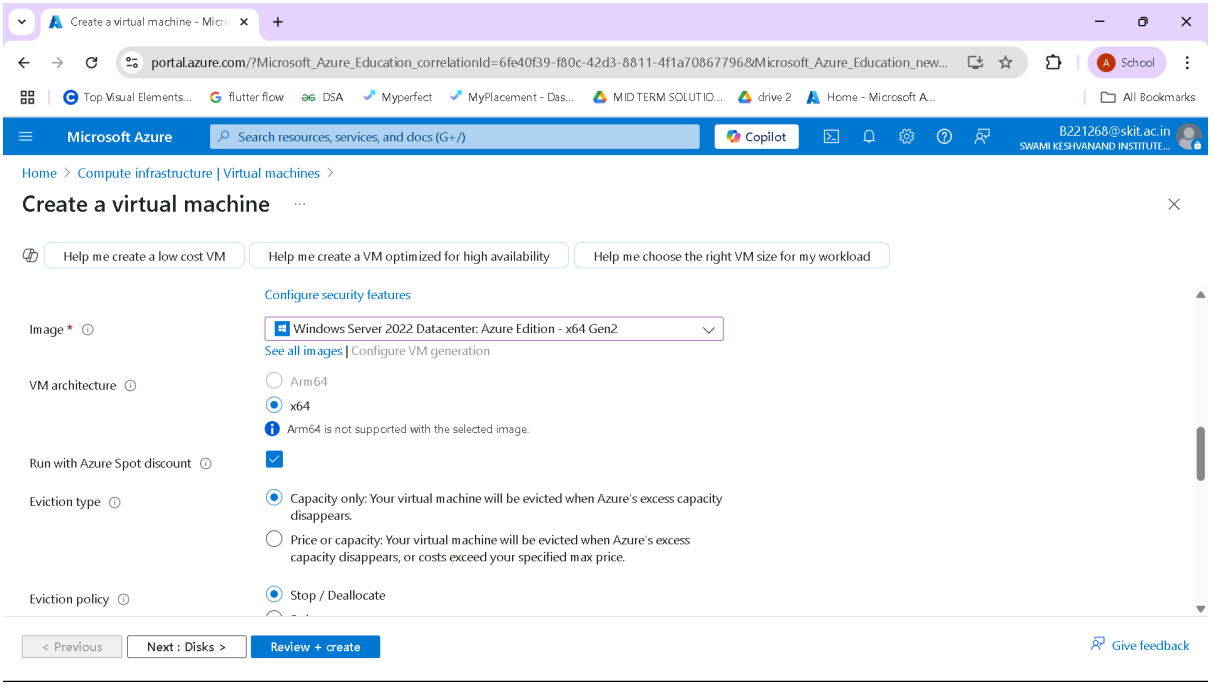
    - Public IP: Enabled (for RDP)

    - NSG: Basic, ensure Port 3389 (RDP) is allowed.

9. Click **Review + Create → Create**.



CSI ASSIGNMENT 4



10. Implementation — Deploy Linux VM

Steps:

1. Go to **Virtual Machines → Create**.
2. Use same Resource Group.
3. VM Name: `Linux-VM`.
4. Image: Ubuntu 22.04 LTS.
5. Region: Same as VNet-1.
6. Set Username & SSH key/password.

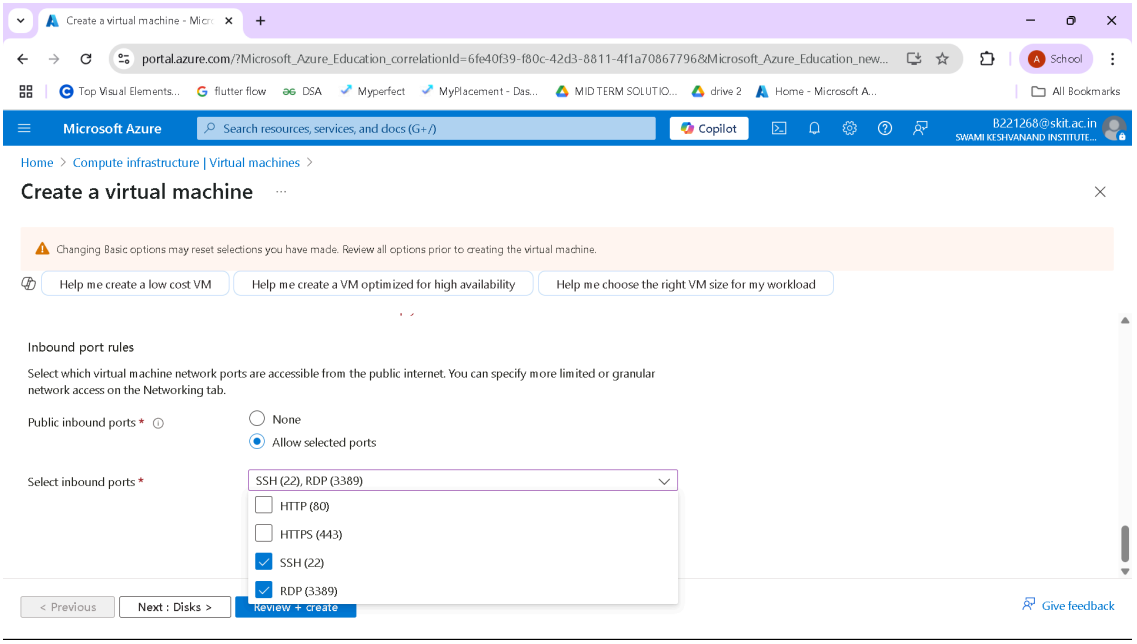
7. **Networking**:
    - Virtual Network: `VNet-1`
    - Subnet: `Subnet-2`
    - Public IP: Enabled (for SSH)
    - NSG: Basic, ensure Port 22 (SSH) is allowed.
    - Click **Review + Create** → **Create**.

Implementation — Configure ICMP & Test

Allow Ping (ICMP):

1. For both VMs, go to **Networking** → **Network Security Group (NSG)** linked to the NIC.
2. Add **Inbound Security Rule**:
    - Source: Any
    - Protocol: ICMP
    - Action: Allow
    - Priority: 300

CSI ASSIGNMENT 4



Test Communication:

- Find **Private IPs** of both VMs.
- RDP into Windows VM → Open Command Prompt → `ping <Linux VM Private IP>`.
- SSH into Linux VM → `ping <Windows VM Private IP>`.

Implementation — Create Second VNet & Peering

Create VNet-2:

1. Create **Virtual Network** → Name: `VNet-2`.

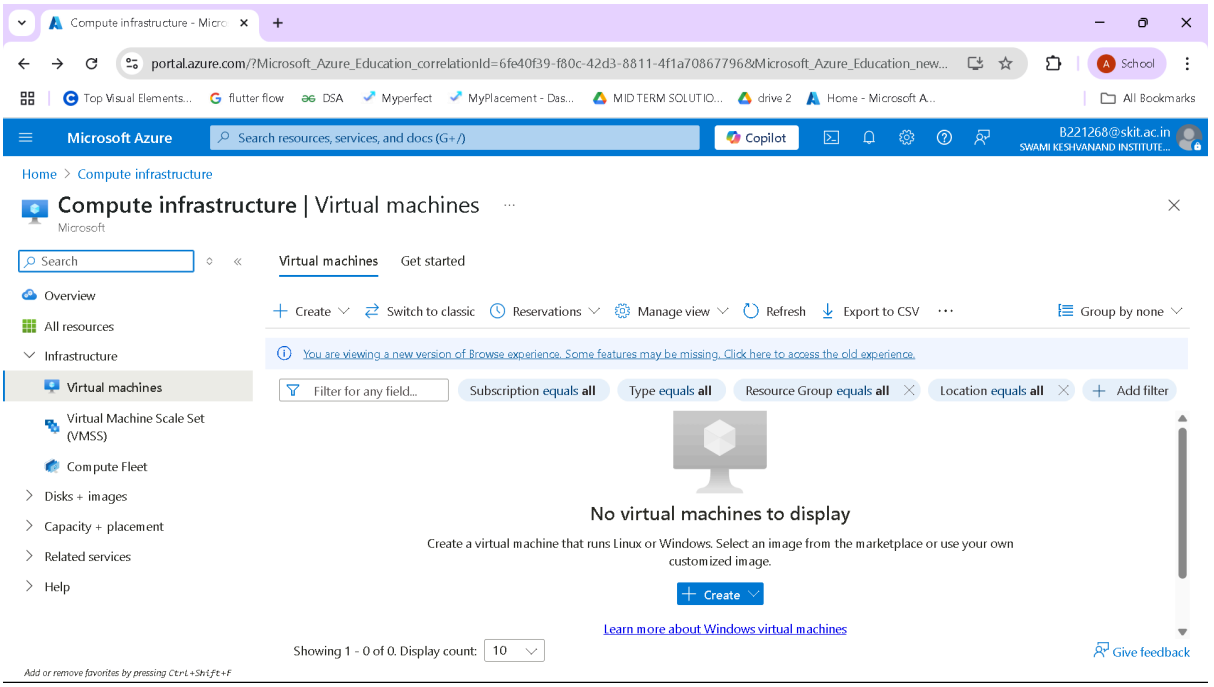
2. Address Space: `10.1.0.0/16`.
3. Add Subnet-1: `10.1.1.0/24`.
4. Deploy test VM if needed (optional).

VNet Peering:

1. Go to `VNet-1` → Peerings → **Add**.
    - Name: `VNet1-to-VNet2`.
    - Select remote VNet: `VNet-2`.
    - Enable traffic in both directions.

CSI ASSIGNMENT 4

2. Repeat for VNet-2 → Peerings → Add back to VNet-1.



3. Next, click on create button.