R&D Report
CIDR Ranges, Subnet Configuration, and VNet Peering

Prepared by: Virat Pandey

Internship: Cloud Infrastructure & Security – Celebal Technology

Date: June 2025

Table of Contents

1. Introduction

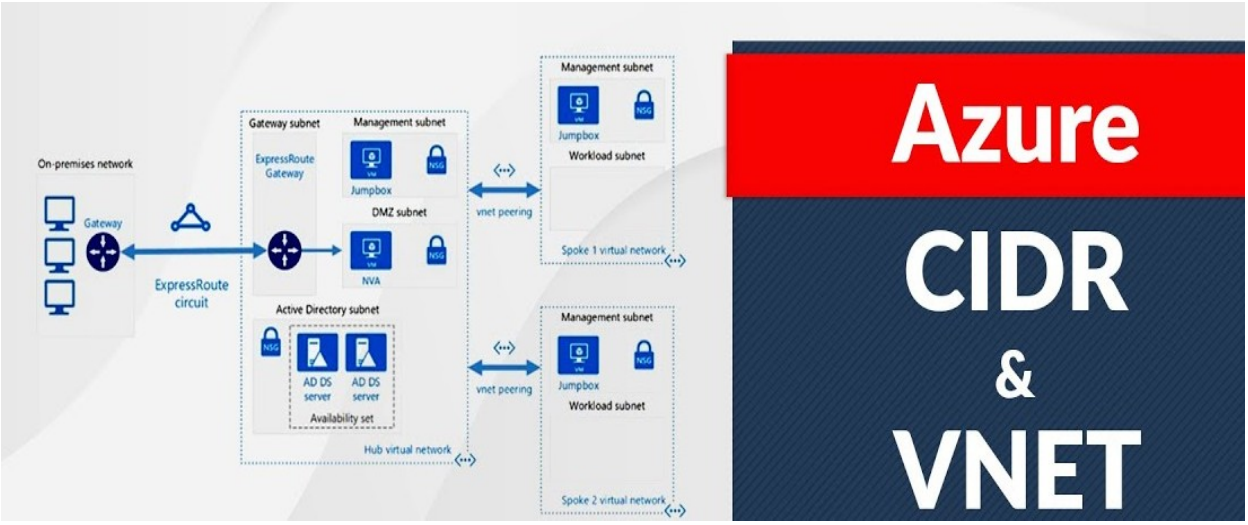
This report documents the practical implementation of Azure Virtual Network (VNet) configuration, CIDR-based subnetting, and inter-VNet peering. The goal was to build a simulated cloud network environment where multiple virtual machines (VMs) are deployed in isolated subnets across different VNets and can communicate securely and efficiently through private IP ranges.

The implementation showcases real-world usage of CIDR blocks, subnet design, and Azure VNet peering for scalable and secure network architectures.

2. Overview of CIDR and VNet Concepts

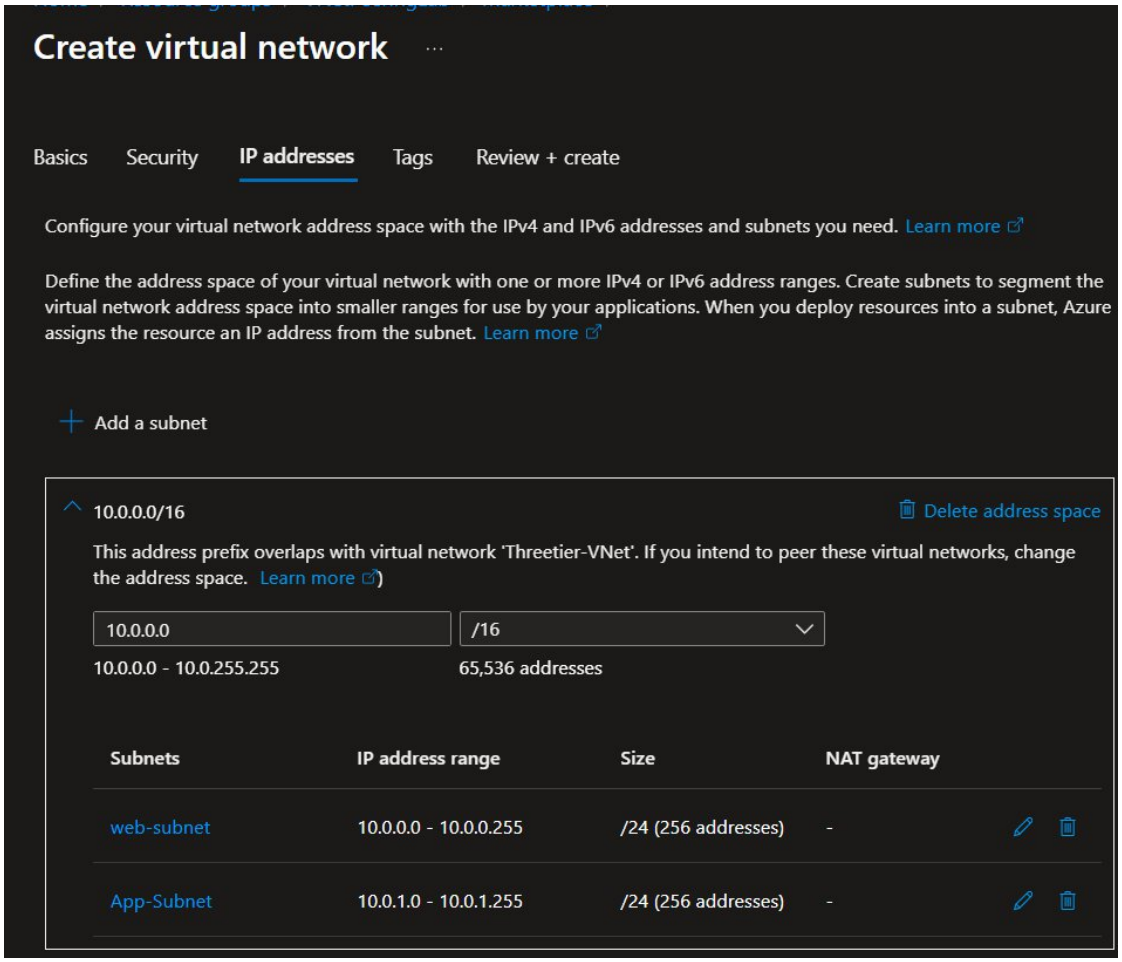
CIDR (Classless Inter-Domain Routing) allows flexible allocation of IP addresses and efficient subnetting. A VNet in Azure uses CIDR ranges to define its IP space, and each subnet is a subdivision of that space.

In this assignment, multiple subnets were created within a VNet using different CIDR blocks. Network Security Groups (NSGs) were configured to allow ping (ICMP), and a second VNet was created and peered with the first to allow inter-VNet communication.



3.1 Creating Virtual Network and Subnets

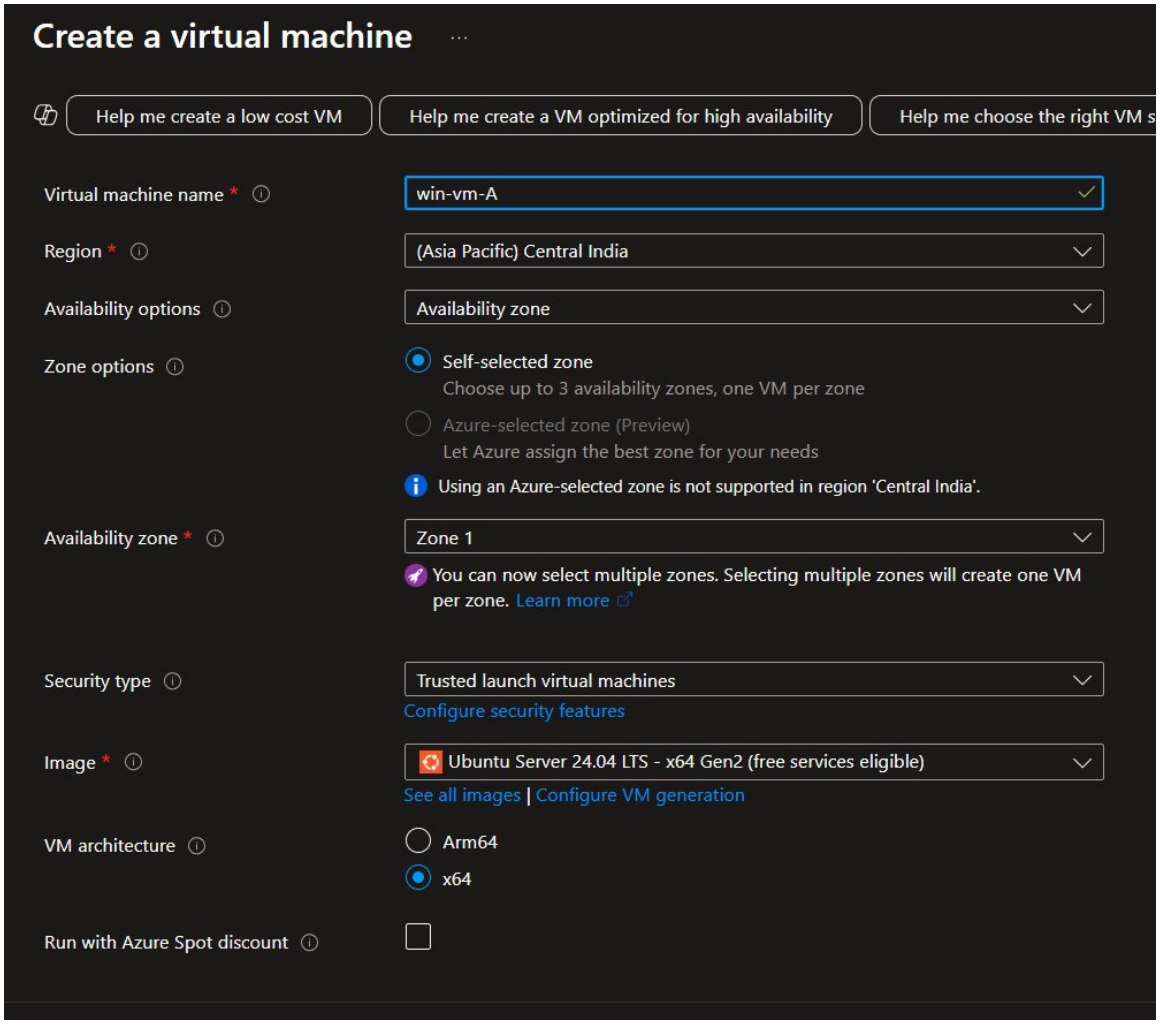
- Resource Group: VNetPeeringLab
- Virtual Network Name: VNet-Project-A
- Address Space: 10.0.0.0/16
- Subnets Created:
  - Web-Subnet: 10.0.0.0/24
  - App-Subnet: 10.0.1.0/24



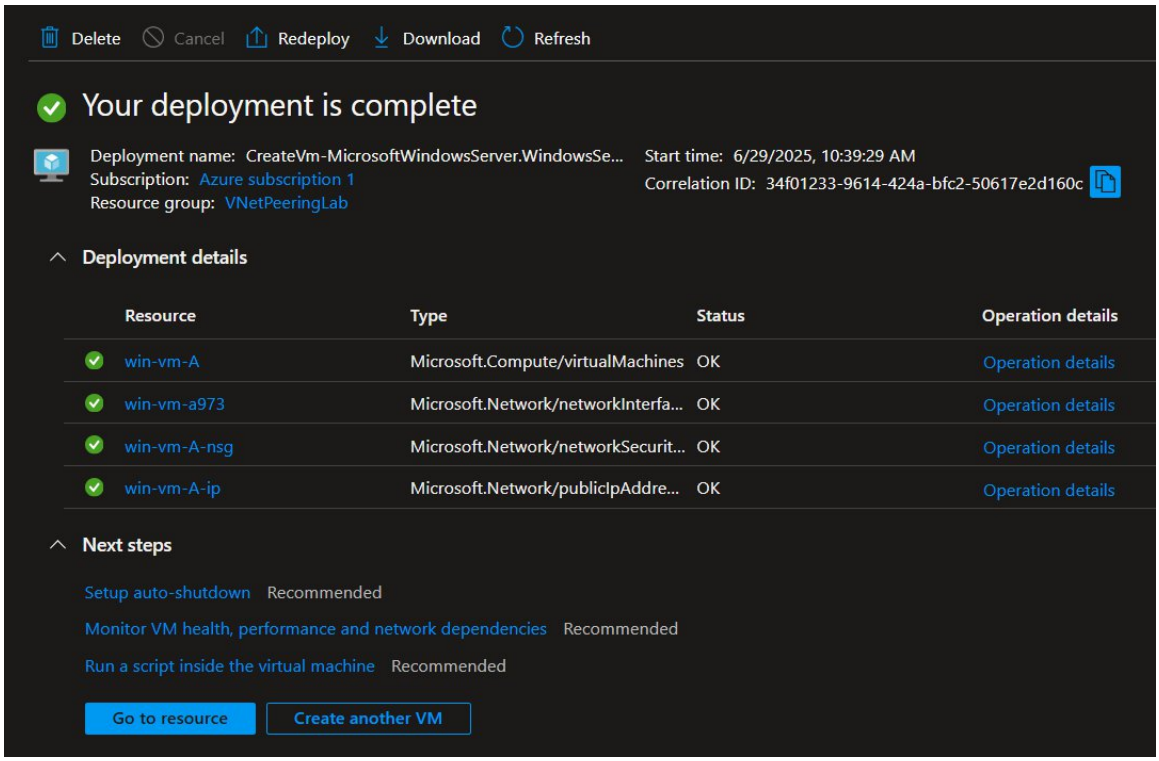
As part of the implementation, a virtual network named **VNet-Project-A** was created to define a custom IP address space using the CIDR block 10.0.0.0/16. This VNet acts as the foundational layer for deploying subnets and virtual machines. It enables isolated and secure communication between resources, and serves as the core network for configuring address ranges, routing, and connectivity in the simulated cloud environment.

3.2 Deploying VMs in Separate Subnets

- win-vm-A: Deployed in Web-Subnet
- linux-vm-A: Deployed in App-Subnet
- Size: Standard B1s
- Public IP: Assigned
- Inbound Port Rule: RDP (Windows), SSH (Linux)

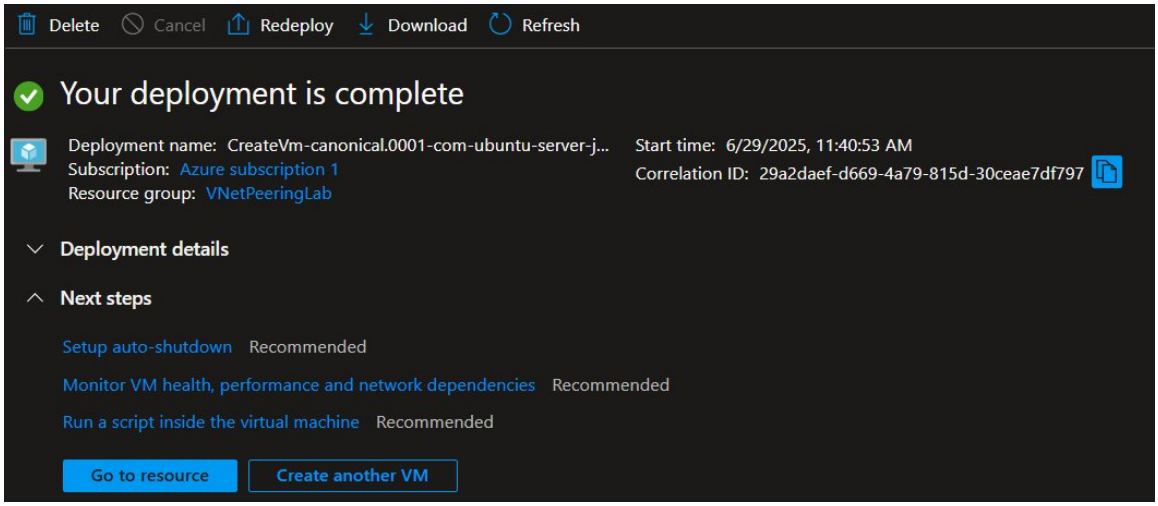


Here we have created our first virtual machine that is windows virtual machine in the web subnet now we will do the same for linux based virtual machine now we will create another virtual machine and it will be placed in the app subnet. Remember both the vm are inside different subnets but are within the same virtual network so they can ping each other which we will see in the upcoming screenshots that how we peered between to virtual machines in different subnets inside a same network.



Here this is the windows vm in the web subnet we have deployed this

Now the linux vm inside the app subnet-

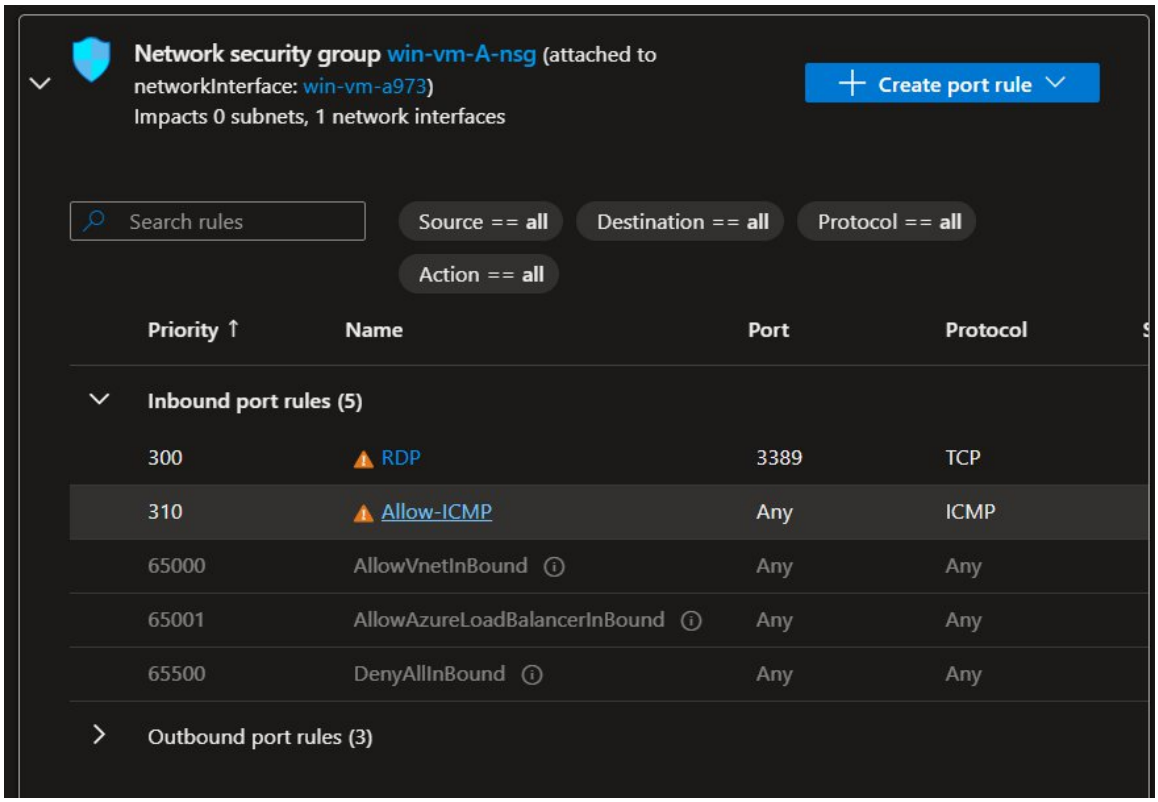


This concludes the creation of virtual machines inside different subnets but within the same

Virtual network. Now we will enable the internal communication between these two virtual machines we will b sending a ping request form the windows vm to the linux based vm

3.3 Enabling Internal Communication Between VMs

- ICMP (ping) was not enabled by default.
- Added inbound NSG rule:
  - Protocol: ICMP
  - Source: Any
  - Destination: Any
  - Port: *
  - Action: Allow



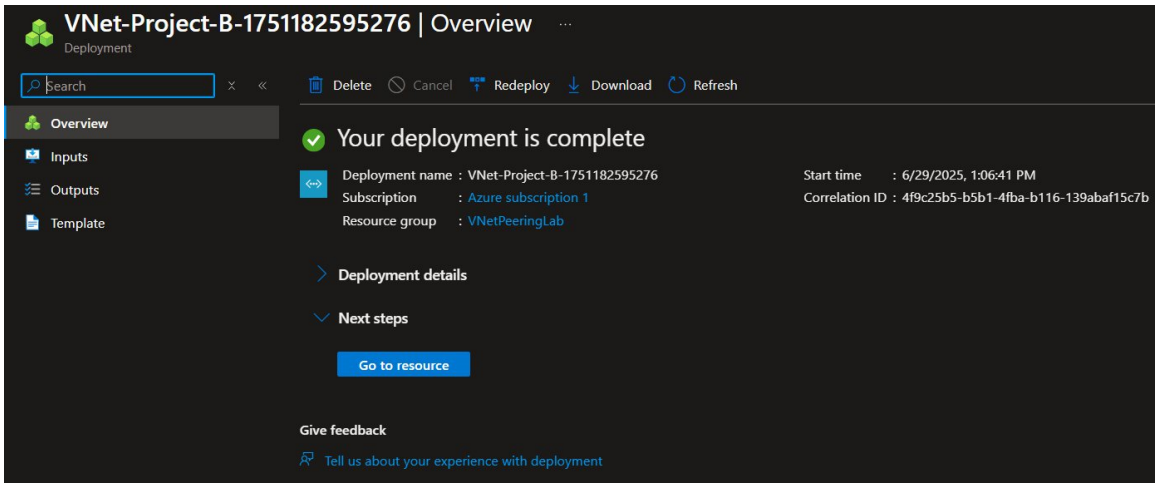
Here we created a new port rule inside the network security group so that the verifying can be carried out as the ICMP (ping) was not enabled by default. So first we enabled that and then we proceeded with creating a second virtual network for peering.

We already have a virtual network-A so for peering and connection testing we have to create one more virtual network

We will name this as virtual network-B and deploy a virtual machine in this vnet also via adding a subnet first.

3.4 Creating Second VNet for Peering

- Virtual Network Name: VNet-Project-B
- Address Space: 10.1.0.0/16
- Subnet: b-subnet (10.1.0.0/24)



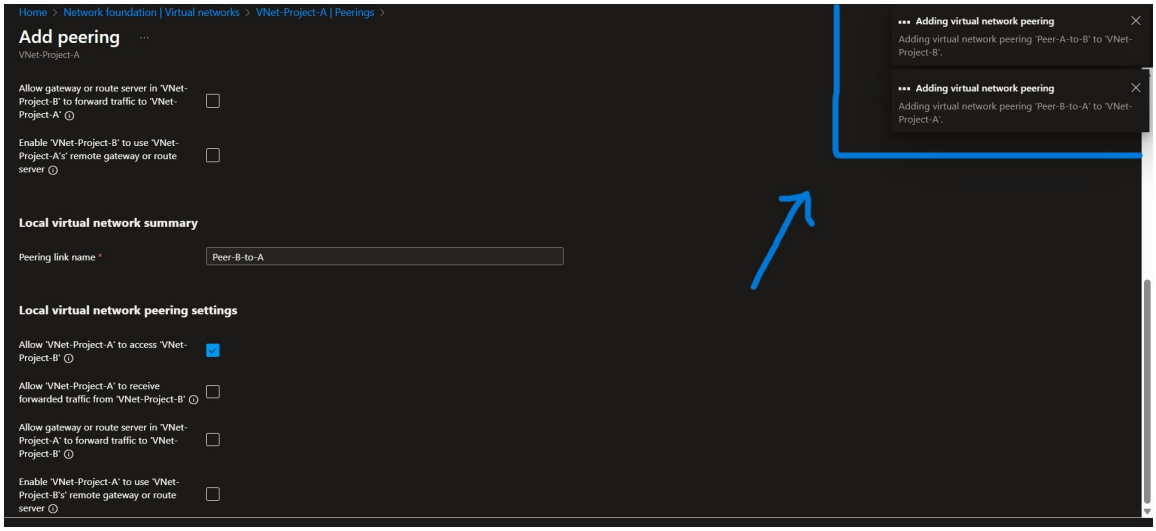
3.5 Peering VNet-Project-A and VNet-Project-B

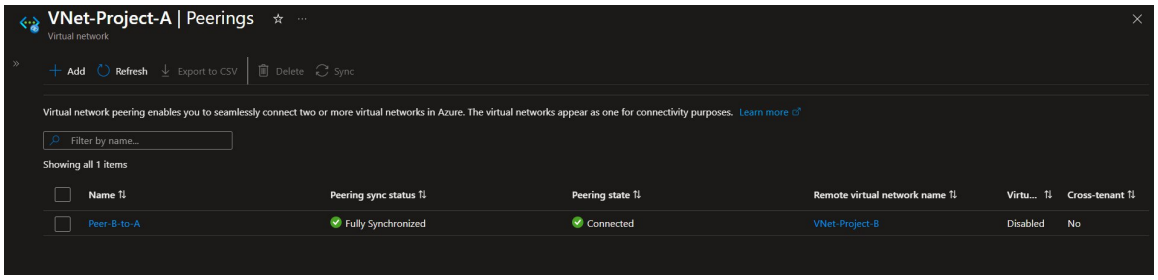
Peering A → B:

- Name: Peer-A-to-B
- Allow traffic: ✔
- Forwarded traffic, gateway: ✖

Peering B → A:

- Name: Peer-B-to-A
- Allow traffic: ✔
- Forwarded traffic, gateway: ✖*

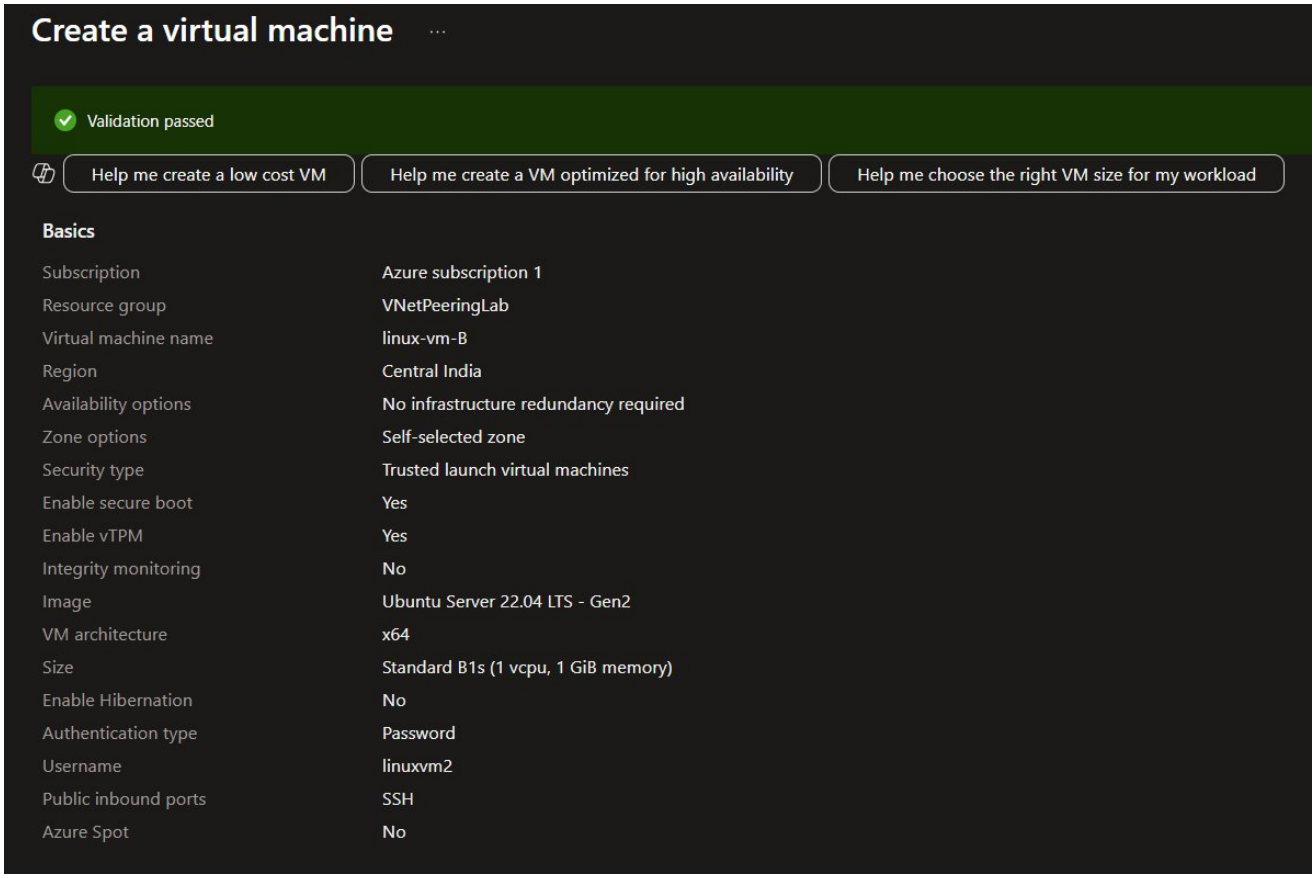


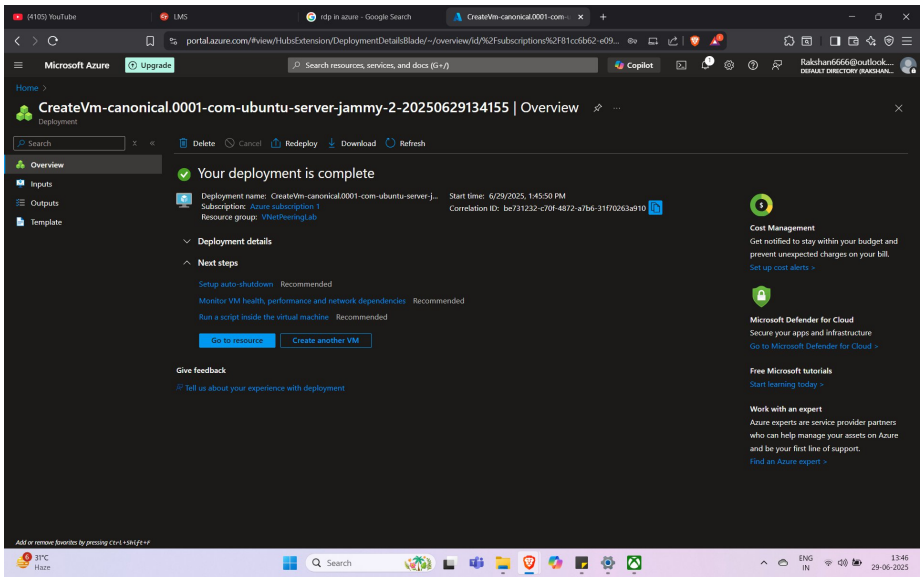


Here we have configured the peering setting in the above picture.

3.6 Deploying VM in VNet-B

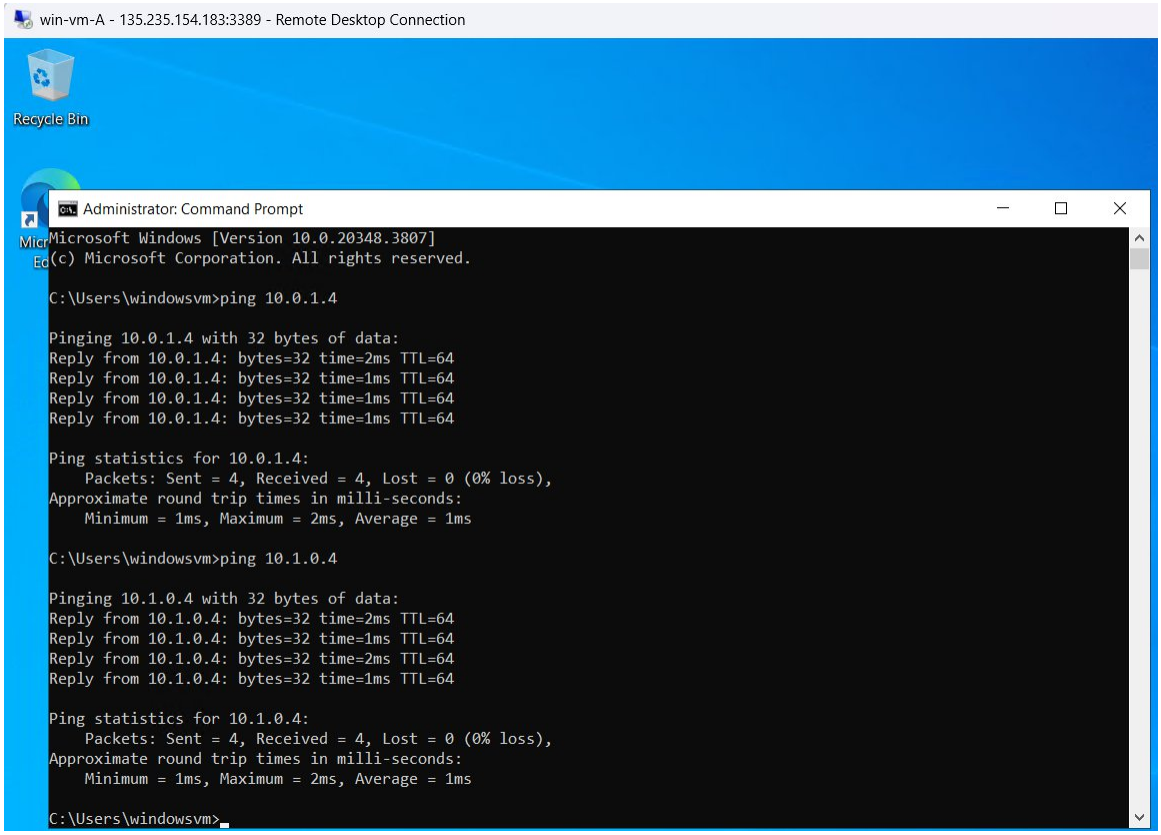
- VM Name: linux-vm-B
- Subnet: b-subnet
- Size: Standard B1s (or alternative due to quota)
- Public IP: Assigned
- Inbound Rule: SSH (Port 22)





3.7 Testing Ping Across Peered VNets

- From win-vm-A (in VNet-A) → pinged linux-vm-B (in VNet-B) using private IP
- Result: Successful reply



The above screenshot includes the terminal responses of two successful ping operations: first, between the Windows and Linux VMs within VNet-Project-A (intra-VNet communication), and second, between the Windows VM in VNet-Project-A and the Linux VM in VNet-Project-B (inter-VNet communication using VNet Peering). This confirms that both subnet-level and VNet-level connectivity have been successfully implemented.4. Result Verification

All steps were executed successfully. VMs in separate subnets were able to communicate through private IPs after NSG modification. Peering between VNet-A and VNet-B enabled cross-VNet communication, verified via ping.

4. Conclusion

This implementation demonstrated a clear understanding of CIDR-based subnetting, NSG configurations, and VNet peering within Microsoft Azure. These skills are essential for designing scalable, secure, and modular cloud infrastructures that mimic real-world enterprise setups.