Assignment- 4

Prepared by: Sanchit Mathur
Summer Intern
CELEBAL TECHNOLOGY

Table of Contents

Executive Summary

This document shares what I learned while working on Azure Virtual Network as part of my internship of Azure Virtual Networks (VNets), focusing on CIDR ranges, subnet configuration, and VNet peering mechanisms. The document includes a practical use case demonstrating the creation of multiple VNets with cross-subnet VM communication and VNet peering implementation.

Key Objectives:

- Understand Azure VNet architecture and CIDR implementation
- Configure subnets with appropriate IP address ranges
- Implement VNet peering for cross-network communication
- Deploy Windows and Linux VMs across different subnets
- Establish and test inter-subnet and inter-VNet connectivity

---

Azure Virtual Network Overview

What is Azure Virtual Network?

Azure Virtual Network (VNet) is a fundamental building block for private networks in Azure. It enables Azure resources to securely communicate with each other, the internet, and on-premises networks. VNets are similar to traditional networks in on-premises data centers but provide additional benefits of Azure's infrastructure.

Key Features:

- **Isolation and Segmentation**: Provides network isolation using private IP address spaces
- **Internet Communication**: Enables resources to communicate with the internet

- **Inter-Resource Communication**: Allows secure communication between Azure resources
- **On-Premises Connectivity**: Connects to on-premises networks via VPN or ExpressRoute
- **Traffic Filtering**: Implements network security groups and firewalls
- **Traffic Routing**: Controls traffic routing between subnets and networks

VNet Components:

1. **Address Space**: The private IP address range for the VNet
2. **Subnets**: Logical divisions within the VNet
3. **Network Security Groups (NSGs)**: Firewall rules for traffic filtering
4. **Route Tables**: Custom routing configurations
5. **VNet Peering**: Connections between VNets
6. **Gateways**: VPN and ExpressRoute connections

CIDR Ranges and IP Addressing

Understanding CIDR (Classless Inter-Domain Routing)

CIDR notation is a method for describing IP address ranges and subnet masks. It uses the format IP_ADDRESS/PREFIX_LENGTH, where the prefix length indicates how many bits are used for the network portion.

Common CIDR Ranges:

CIDR Notation	Subnet Mask	Number of IPs	Usable IPs	Use Case
/16	255.255.0.0	65,536	65,531	Large enterprise networks
/20	255.255.240.0	4,096	4,091	Medium enterprise networks
/24	255.255.255.0	256	251	Small networks, single subnets
/28	255.255.255.240	16	11	Very small subnets

Azure Reserved IP Addresses

Azure reserves 5 IP addresses in each subnet:

- **x.x.x.0**: Network address
- **x.x.x.1**: Reserved for Azure gateway
- **x.x.x.2**: Reserved for Azure DNS
- **x.x.x.3**: Reserved for Azure DNS
- **x.x.x.255**: Network broadcast address

Private IP Address Ranges (RFC 1918):

- **10.0.0.0/8**: 10.0.0.0 - 10.255.255.255
- **172.16.0.0/12**: 172.16.0.0 - 172.31.255.255
- **192.168.0.0/16**: 192.168.0.0 - 192.168.255.255

Subnets in Azure VNets

Subnet Design Principles

Subnets are logical divisions within a VNet that help organize and secure resources. Proper subnet design is crucial for network performance, security, and management.

Subnet Planning Considerations:

1. **Size Requirements**: Plan for current and future resource needs
2. **Security Boundaries**: Group resources with similar security requirements
3. **Service Requirements**: Some Azure services require dedicated subnets
4. **Growth Planning**: Allow room for expansion
5. **IP Address Efficiency**: Avoid waste while maintaining flexibility

Subnet Types:

1. **Default Subnet**: Standard subnet for general resources
2. **Dedicated Subnets**: Required for specific services (App Service, SQL MI)
3. **Gateway Subnet**: Required for VPN and ExpressRoute gateways
4. **Azure Firewall Subnet**: Dedicated subnet for Azure Firewall
5. **Azure Bastion Subnet**: Dedicated subnet for Azure Bastion

Best Practices for Subnet Design:

- Use hierarchical addressing schemes
- Implement network security groups per subnet
- Consider traffic flow patterns
- Plan for service-specific subnet requirements
- Document subnet purposes and ownership

VNet Peering and Types

What is VNet Peering?

VNet peering connects two Azure virtual networks, enabling resources in different VNets to communicate as if they were in the same network. Traffic between peered VNets is routed through Microsoft's private backbone network.

Types of VNet Peering:

1. Regional VNet Peering

- Connects VNets in the same Azure region
- Low latency, high bandwidth
- No additional costs for data transfer
- Supports all Azure services

2. Global VNet Peering

- Connects VNets across different Azure regions
- Uses Microsoft's global backbone network
- Additional charges for cross-region data transfer
- Supports most Azure services with some limitations

VNet Peering Characteristics:

- **Non-transitive**: If VNet A peers with VNet B, and VNet B peers with VNet C, VNet A cannot communicate with VNet C
- **Bidirectional**: Requires configuration on both VNets
- **No gateway required**: Direct connection between VNets
- **Cross-subscription support**: Can peer VNets in different subscriptions

- **Address space overlap**: Cannot peer VNets with overlapping address spaces

Prerequisites

Azure Account Requirements:

1. **Azure Subscription**: Active Azure subscription with appropriate permissions
2. **Resource Group**: Container for organizing resources
3. **Permissions**: Contributor or Owner role on the subscription
4. **Quotas**: Sufficient VM core quotas for deployment

Network Planning:

1. **IP Address Planning**: Non-overlapping CIDR ranges
2. **DNS Configuration**: Custom DNS servers if required
3. **Security Planning**: Network security group rules
4. **Routing Requirements**: Custom routing if needed

VM Requirements:

1. **VM Sizes**: Appropriate VM sizes for Windows and Linux
2. **Operating Systems**: Windows Server 2019/2022, Ubuntu 20.04 LTS
3. **Storage**: Managed disks for VMs
4. **Authentication**: SSH keys for Linux, RDP for Windows

Tools and Access:

1. **Azure Portal**: Web-based management interface
2. **Azure CLI**: Command-line interface (optional)
3. **PowerShell**: Azure PowerShell module (optional)

4. **SSH Client**: For Linux VM access
5. **RDP Client**: For Windows VM access

Use Case Implementation

Scenario Overview

Objective: Create a comprehensive Azure networking solution that demonstrates:

1. VNet creation with multiple subnets
2. VM deployment across different subnets
3. Inter-subnet communication
4. VNet peering implementation
5. Cross-VNet communication

Architecture Design:

VNet 1: Production Network

- **VNet Name**: VNet-Production
- **Address Space**: 10.1.0.0/16
- **Subnets**:
    - Web-Subnet: 10.1.1.0/24 (Windows VM)
    - App-Subnet: 10.1.2.0/24 (Linux VM)

VNet 2: Development Network

- **VNet Name**: VNet-Development
- **Address Space**: 10.2.0.0/16
- **Subnets**:
    - Dev-Subnet: 10.2.1.0/24 (Test VMs)

VM Deployment Plan:

1. **Windows VM**: Deploy in Web-Subnet (10.1.1.0/24)
2. **Linux VM**: Deploy in App-Subnet (10.1.2.0/24)
3. **Test VMs**: Deploy in Dev-Subnet (10.2.1.0/24)

Connectivity Requirements:

- Windows VM ↔ Linux VM (same VNet)
- Production VMs ↔ Development VMs (via VNet peering)
- Internet access for all VMs (outbound)

Step-by-Step Configuration Guide

Phase 1: Create Resource Group

1. **Navigate to Azure Portal**
   - Go to https://portal.azure.com
   - Sign in with your Azure credentials
2. **Create Resource Group**
   - Search for "Resource Groups" in the top search bar
   - Click "Create"
   - Fill in details:
     - **Subscription**: Select your subscription
     - **Resource Group**: RG-VNet-Demo
     - **Region**: East US 2
   - Click "Review + Create" → "Create"

Phase 2: Create VNet-Production

1. **Navigate to Virtual Networks**
   - Search for "Virtual Networks" in Azure Portal

   - Click "Create virtual network"

2. **Basics Tab Configuration**
   - **Subscription**: Select your subscription
   - **Resource Group**: RG-VNet-Demo
   - **Name**: VNet-Production
   - **Region**: East US 2
3. **IP Addresses Tab Configuration**
   - **IPv4 address space**: 10.1.0.0/16
   - **Subnet Configuration**:
     - **Subnet 1**:
       - Name: Web-Subnet
       - Address range: 10.1.1.0/24
     - **Subnet 2**:
       - Name: App-Subnet
       - Address range: 10.1.2.0/24

4. **Security Tab**
   - **BastionHost**: Disable
   - **DDoS Protection**: Basic
   - **Firewall**: Disable
5. **Review and Create**
   - Review all configurations
   - Click "Create"

Phase 3: Create VNet-Development

1. **Create Second VNet**
   - **Name**: VNet-Development
   - **Address Space**: 10.2.0.0/16
   - **Subnet**:
     - Name: Dev-Subnet
     - Address range: 10.2.1.0/24

Phase 4: Create Network Security Groups

1. **Create NSG for Web-Subnet**
   - Search for "Network Security Groups"
   - Click "Create"
   - **Name**: NSG-Web-Subnet
   - **Resource Group**: RG-VNet-Demo
   - **Location**: East US 2
2. **Configure Inbound Rules for Web NSG**
   - **Rule 1**: Allow RDP
     - Priority: 100
     - Source: Any
     - Destination: Any
     - Service: RDP
     - Action: Allow
   - **Rule 2**: Allow HTTP
     - Priority: 110

        - Source: Any
        - Destination: Any
        - Service: HTTP
        - Action: Allow
    - **Rule 3**: Allow ICMP
        - Priority: 120
        - Source: Any
        - Destination: Any
        - Protocol: ICMP
        - Action: Allow

3. **Create NSG for App-Subnet**
    - **Name**: NSG-App-Subnet
    - **Inbound Rules**:
        - **Rule 1**: Allow SSH
            - Priority: 100
            - Service: SSH
            - Action: Allow
        - **Rule 2**: Allow ICMP
            - Priority: 110
            - Protocol: ICMP
            - Action: Allow
4. **Associate NSGs with Subnets**
    - Go to VNet-Production → Subnets
    - Click Web-Subnet → Associate NSG-Web-Subnet

- Click App-Subnet → Associate NSG-App-Subnet

Phase 5: Deploy Windows VM

1. **Create Windows VM**
    - Search for "Virtual Machines"
    - Click "Create" → "Azure virtual machine"
2. **Basics Tab**
    - **Subscription**: try azure free.
    - **Resource Group**: RG-VNet-Demo
    - **VM Name**: VM-Windows-Web
    - **Region**: East US 2
    - **Image**: Windows Server 2019 Datacenter
    - **Size**: Standard_B2s
    - **Username**: azureuser(set as defult)
    - **Password**: ComplexPassword123!
3. **Networking Tab**
    - **Virtual Network**: VNet-Production
    - **Subnet**: Web-Subnet
    - **Public IP**: Create new (VM-Windows-Web-ip)
    - **NIC NSG**: None (using subnet NSG)
4. **Create VM**
    - Review and create the VM

Phase 6: Deploy Linux VM

1. **Create Linux VM**

  - **VM Name**: VM-Linux-App
  - **Image**: Ubuntu Server 20.04 LTS
  - **Size**: Standard_B2s
  - **Authentication**: SSH public key
  - **Username**: azureuser(by default)
  - **SSH Key**: Generate new key pair
2. **Networking Configuration**
  - **Virtual Network**: VNet-Production
  - **Subnet**: App-Subnet
  - **Public IP**: Create new (VM-Linux-App-ip)

Phase 7: Configure VNet Peering

1. **Create Peering from VNet-Production to VNet-Development**
  - Go to VNet-Production
  - Click "Peerings" in the left menu
  - Click "Add"
  - **Peering Configuration**:
    - **Name of peering**: Prod-to-Dev
    - **Remote virtual network**: VNet-Development
    - **Allow forwarded traffic**: Yes
    - **Allow gateway transit**: No
    - **Use remote gateway**: No
2. **Create Reverse Peering**
  - Go to VNet-Development
  - Click "Peerings"

   - Click "Add"
   - **Peering Configuration**:
     - **Name of peering**: Dev-to-Prod
     - **Remote virtual network**: VNet-Production
     - **Allow forwarded traffic**: Yes
3. **Verify Peering Status**
   - Both peerings should show "Connected" status

Phase 8: Create Test VM in Development VNet

1. **Create Test VM**
   - **VM Name**: VM-Test-Dev
   - **Image**: Ubuntu Server 20.04 LTS
   - **Virtual Network**: VNet-Development
   - **Subnet**: Dev-Subnet

Testing and Validation

Phase 9: Test Inter-Subnet Communication

1. **Connect to Windows VM**
   - Use RDP to connect to VM-Windows-Web
   - Open Command Prompt as Administrator
2. **Test Ping to Linux VM**
3. ping 10.1.2.4

(Replace with actual IP of Linux VM)

4. **Test from Linux VM**
   - SSH to VM-Linux-App

   - Test ping to Windows VM:

5. ping 10.1.1.4

Phase 10: Test VNet Peering Communication

1. **Test from Production to Development**
   - From Windows VM, ping the Test VM:
2. ping 10.2.1.4
3. **Test from Development to Production**
   - From Test VM, ping Windows VM:
4. ping 10.1.1.4

Phase 11: Network Troubleshooting Commands

1. **Windows VM Diagnostics**
2. ipconfig /all
3. route print
4. nslookup google.com
5. telnet 10.1.2.4 22
6. **Linux VM Diagnostics**
7. ip addr show
8. ip route
9. nslookup google.com
10. netstat -rn
11. nc -zv 10.1.1.4 3389

Monitoring and Management

1. **Azure Monitor**

- Set up basic monitoring to check VM and network health.

2. **Cost Optimization**
   - Monitor data transfer costs
   - Optimize VM sizes
   - Use Azure Reserved Instances
   - Regular cost reviews

Troubleshooting

Common Issues and Solutions

1. **VM Cannot Ping Each Other**
   - Check NSG rules
   - Verify subnet routes
   - Check Windows Firewall
   - Verify IP addresses
2. **VNet Peering Not Working**
   - Verify peering status
   - Check address space overlap
   - Verify NSG rules
   - Check route tables
3. **Cannot Connect to VMs**
   - Check public IP configuration
   - Verify NSG rules
   - Check VM firewall
   - Verify credentials

Diagnostic Tools

1. **Azure Network Watcher**
   - IP flow verify
   - Next hop analysis
   - Security group view
   - VPN diagnostics
2. **Azure Portal Diagnostics**
   - Resource health
   - Activity logs
   - Metrics and alerts
   - Connection troubleshoot

Conclusion

This comprehensive R&D document has covered the essential aspects of Azure Virtual Networks, including CIDR ranges, subnet configuration, and VNet peering implementation. The practical use case demonstrated the creation of a multi-VNet environment with cross-subnet and cross-VNet communication.

Key Achievements:

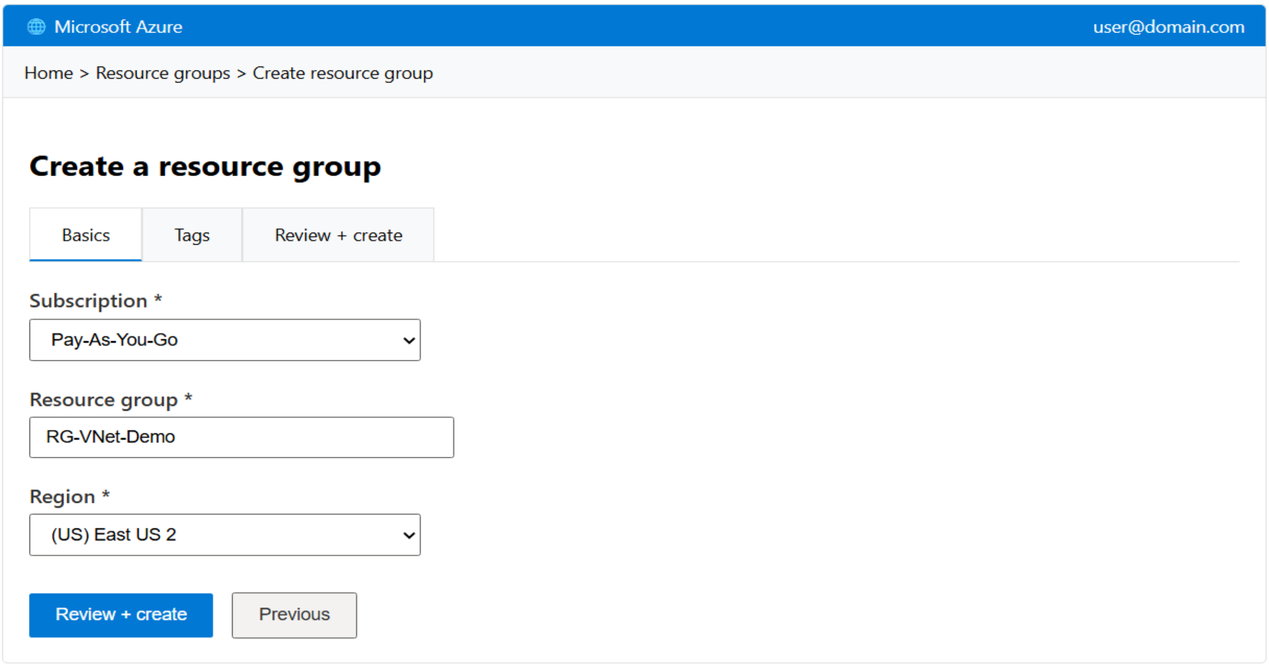
1. **Successful VNet Creation**: Implemented two VNets with proper CIDR planning
2. **Subnet Configuration**: Created multiple subnets with appropriate security controls
3. **VM Deployment**: Successfully deployed Windows and Linux VMs

4. **Inter-Subnet Communication**: Established communication between VMs in different subnets
5. **VNet Peering**: Implemented and tested VNet peering connectivity
6. **Security Implementation**: Applied NSGs and proper access controls

Learning Outcomes:

- Understanding of Azure networking fundamentals
- Practical experience with CIDR planning
- VNet peering configuration and troubleshooting
- Security best practices implementation
- Network monitoring and diagnostics

Screenshots

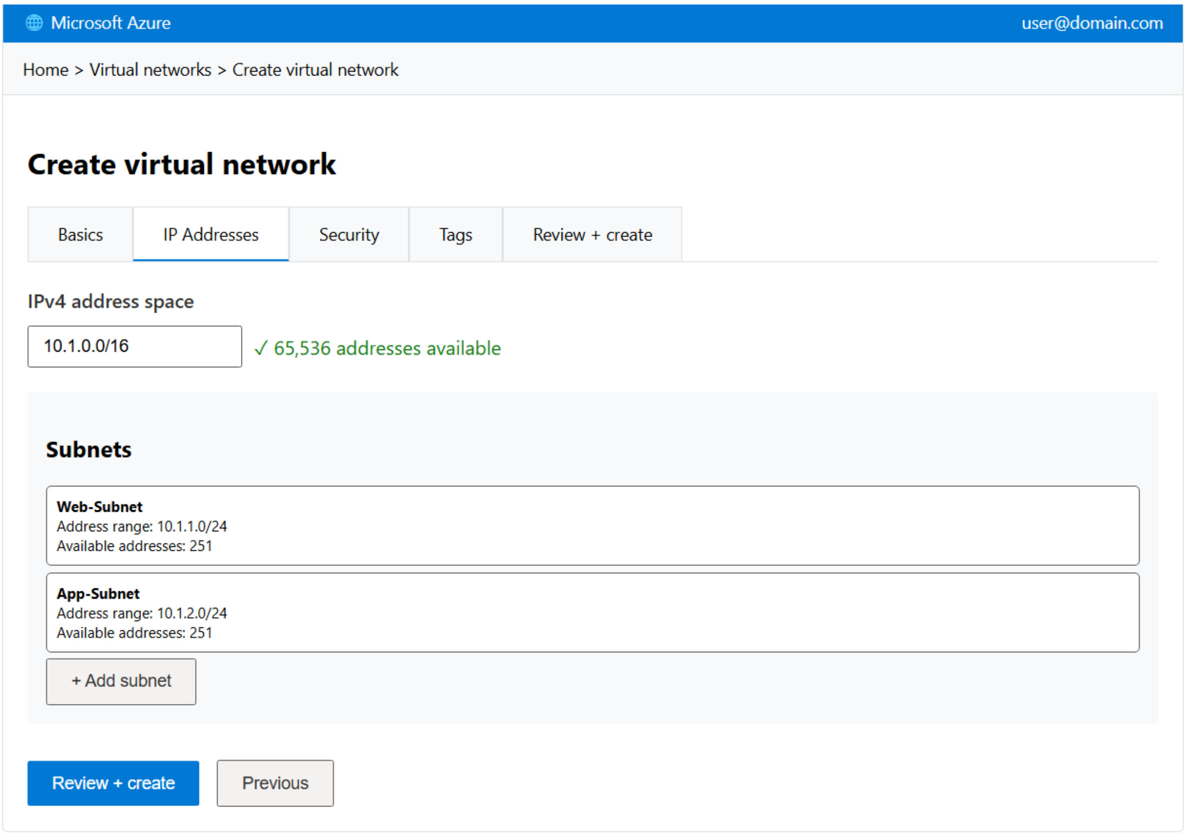


Create a resource group

Subscription *: Pay-As-You-Go

Resource group *: RG-VNet-Demo

Region *: (US) East US 2



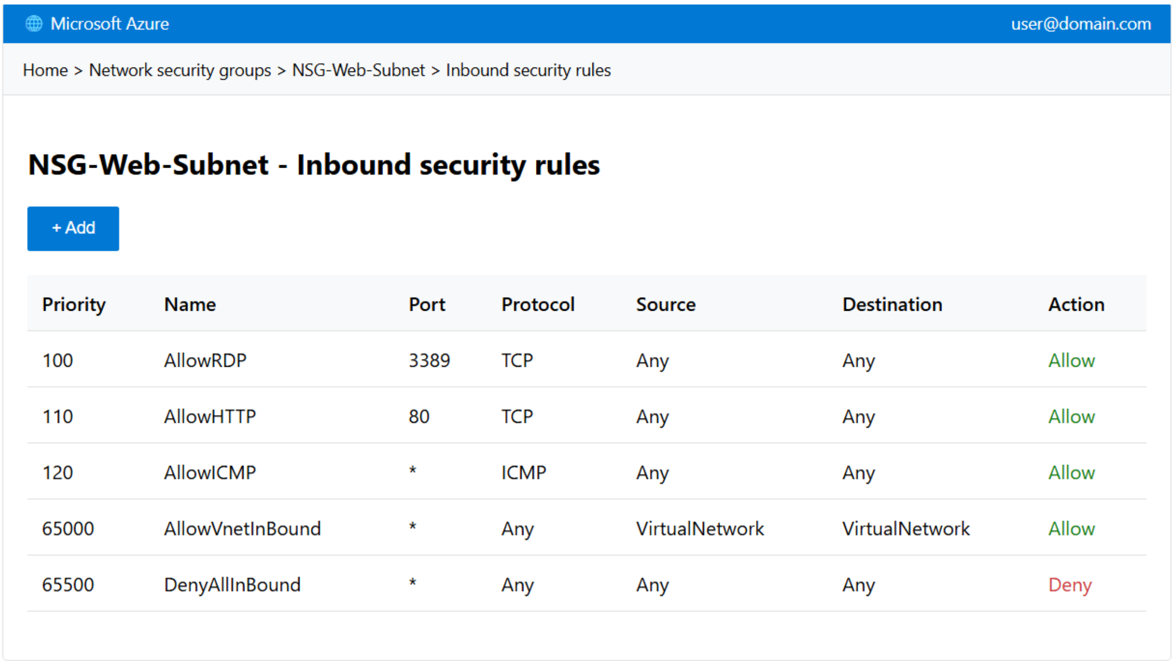
Create virtual network

IPv4 address space: 10.1.0.0/16 ✓ 65,536 addresses available

Subnets

Web-Subnet
Address range: 10.1.1.0/24
Available addresses: 251

App-Subnet
Address range: 10.1.2.0/24
Available addresses: 251



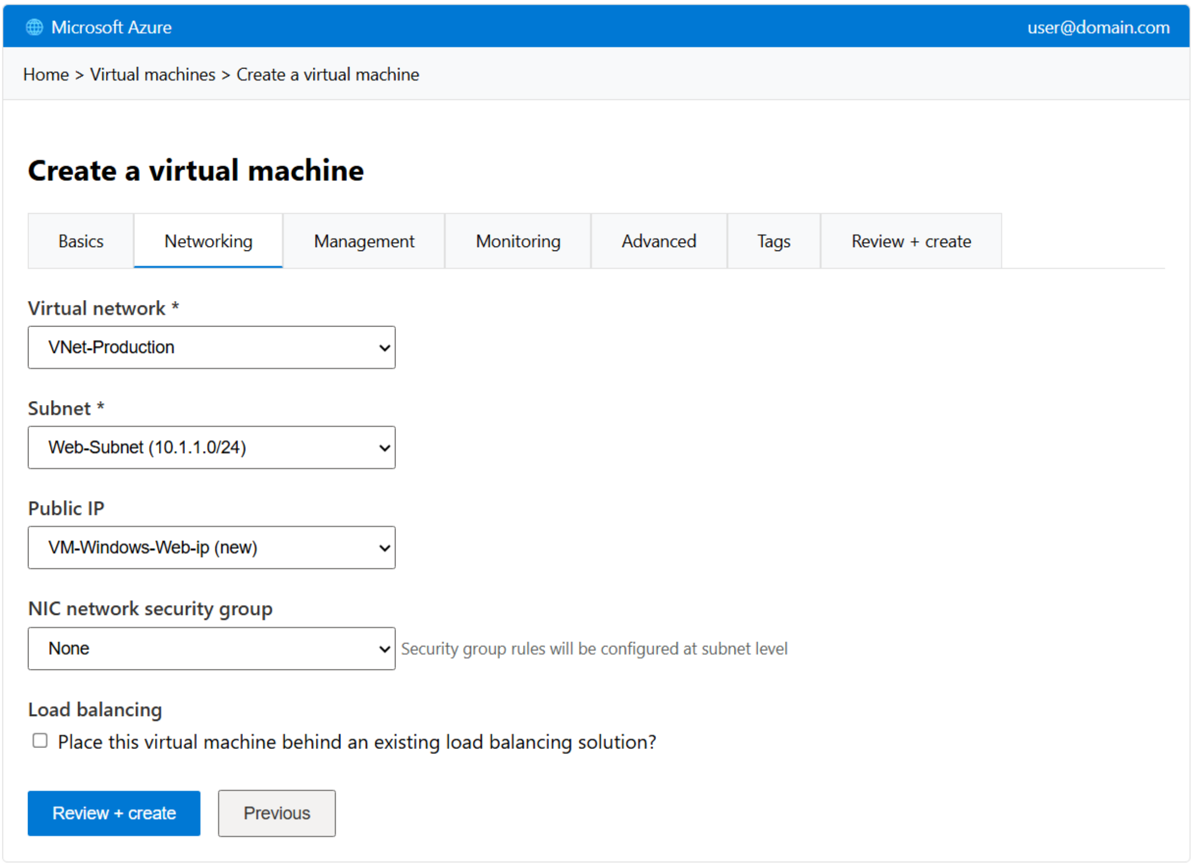
Microsoft Azure user@domain.com

Home > Network security groups > NSG-Web-Subnet > Inbound security rules

NSG-Web-Subnet - Inbound security rules

+ Add

Priority	Name	Port	Protocol	Source	Destination	Action
100	AllowRDP	3389	TCP	Any	Any	Allow
110	AllowHTTP	80	TCP	Any	Any	Allow
120	AllowICMP	*	ICMP	Any	Any	Allow
65000	AllowVnetInBound	*	Any	VirtualNetwork	VirtualNetwork	Allow
65500	DenyAllInBound	*	Any	Any	Any	Deny



Microsoft Azure user@domain.com

Home > Virtual machines > Create a virtual machine

Create a virtual machine

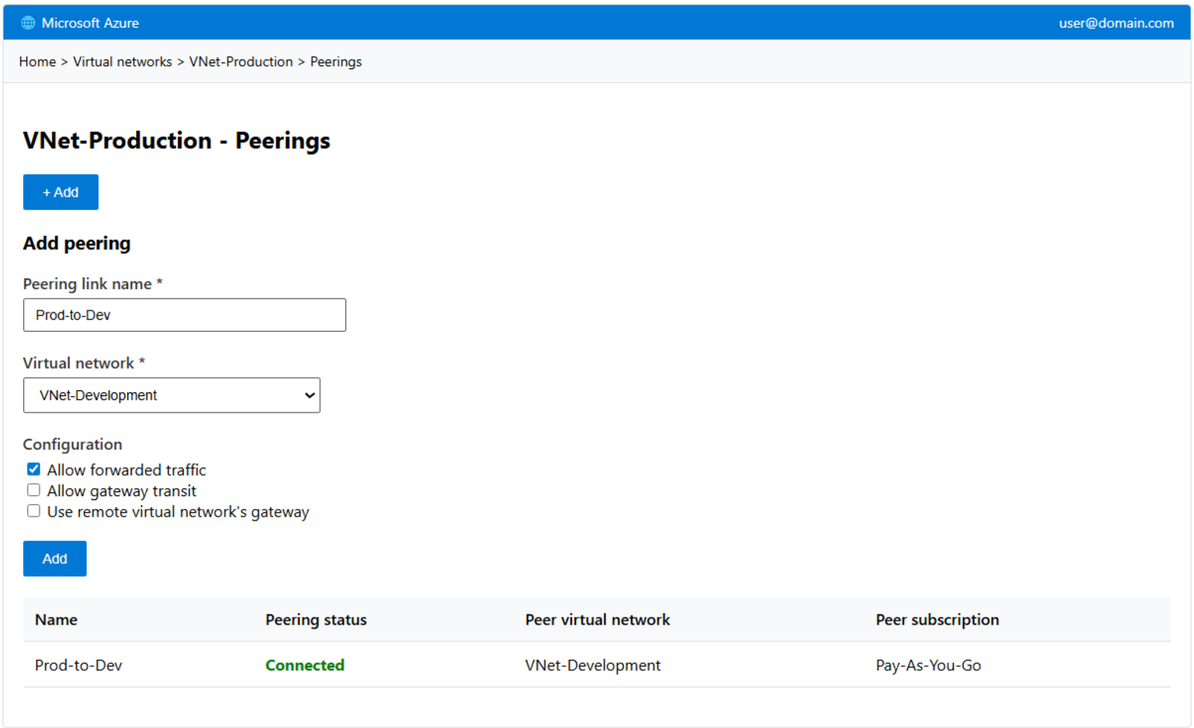
Virtual network *
VNet-Production

Subnet *
Web-Subnet (10.1.1.0/24)

Public IP
VM-Windows-Web-ip (new)

NIC network security group
None
Security group rules will be configured at subnet level

Load balancing
☐ Place this virtual machine behind an existing load balancing solution?



Microsoft Azure user@domain.com

Home > Virtual networks > VNet-Production > Peerings

VNet-Production - Peerings

+ Add

Add peering

Peering link name *

Prod-to-Dev

Virtual network *

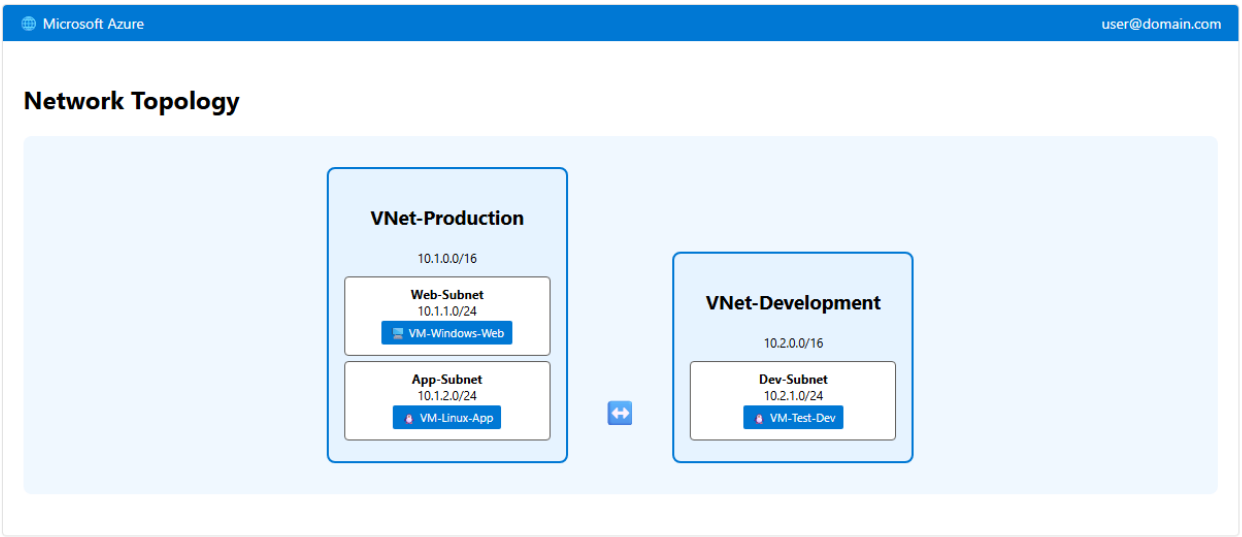
VNet-Development

Configuration

☑ Allow forwarded traffic
□ Allow gateway transit
□ Use remote virtual network's gateway

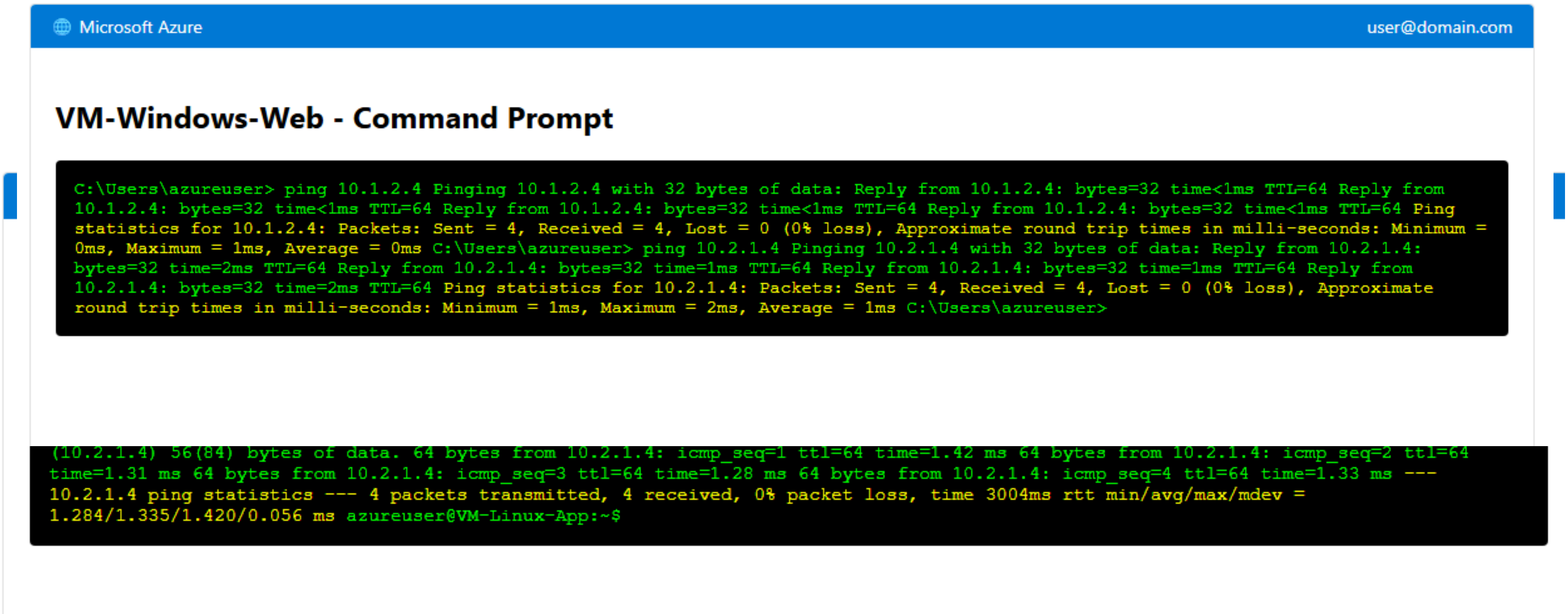
Add

Name	Peering status	Peer virtual network	Peer subscription
Prod-to-Dev	Connected	VNet-Development	Pay-As-You-Go



Microsoft Azure user@domain.com

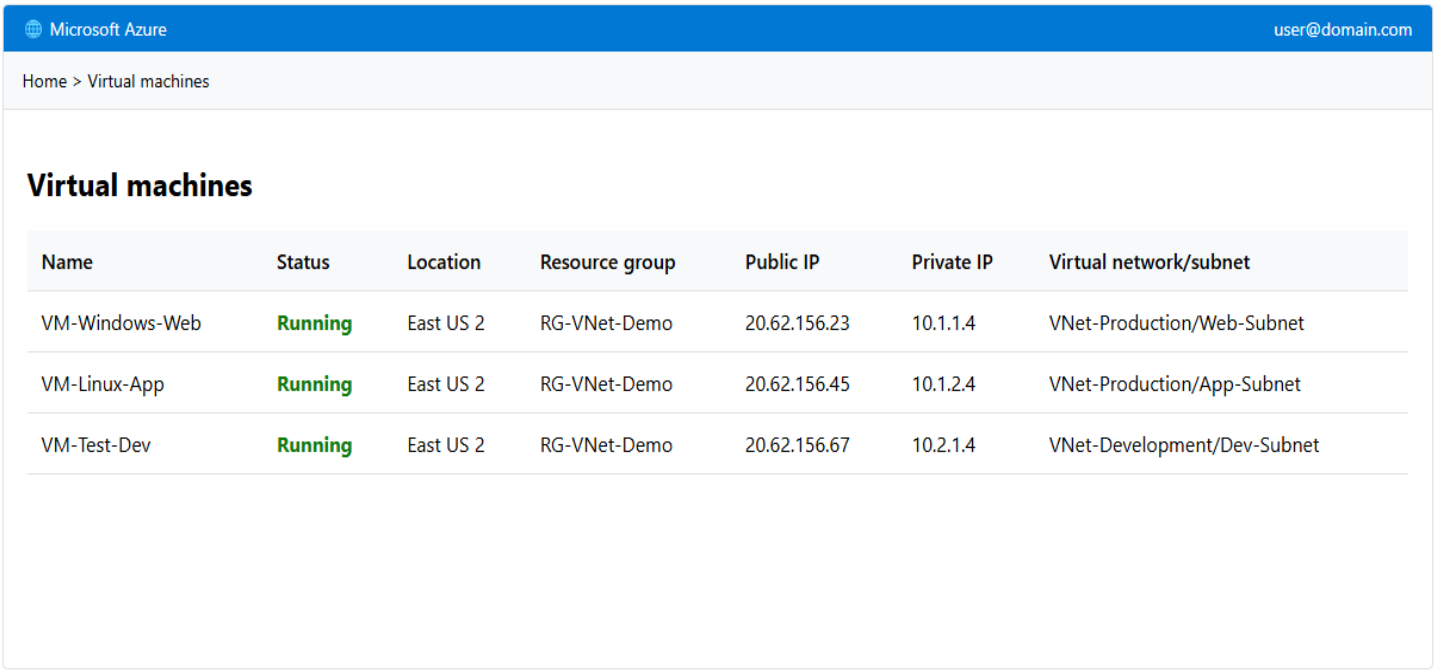
Network Topology



VM-Windows-Web - Command Prompt

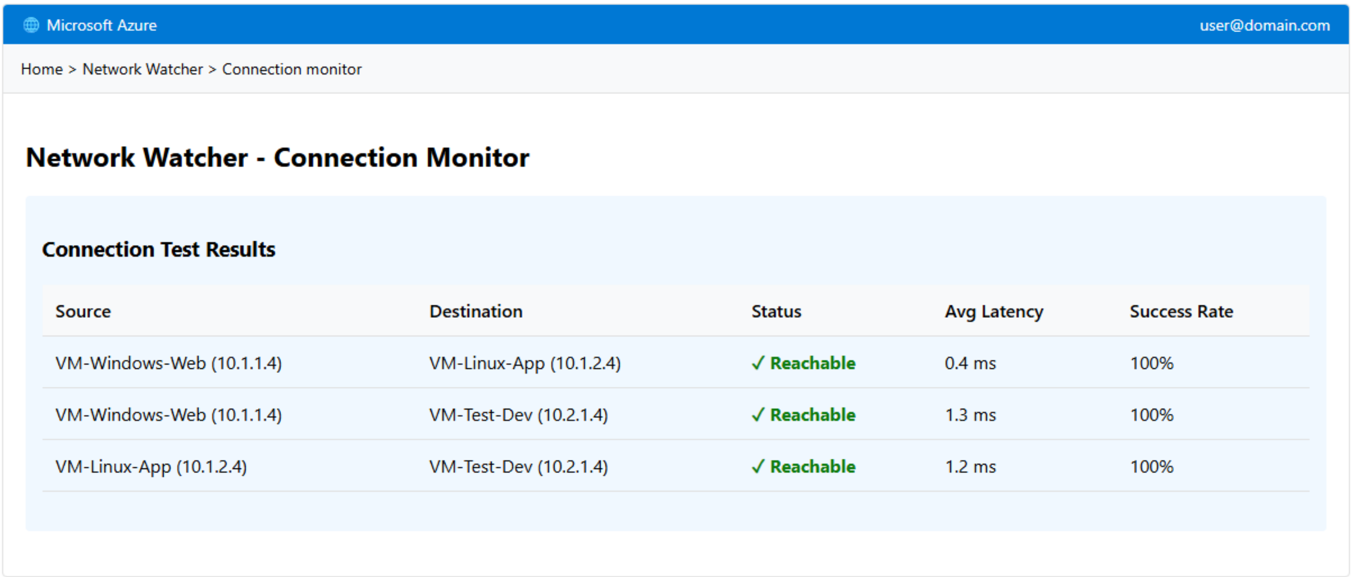
```
C:\Users\azureuser> ping 10.1.2.4 Pinging 10.1.2.4 with 32 bytes of data: Reply from 10.1.2.4: bytes=32 time<1ms TTL=64 Reply from
10.1.2.4: bytes=32 time<1ms TTL=64 Reply from 10.1.2.4: bytes=32 time<1ms TTL=64 Reply from 10.1.2.4: bytes=32 time<1ms TTL=64 Ping
statistics for 10.1.2.4: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds: Minimum =
0ms, Maximum = 1ms, Average = 0ms C:\Users\azureuser> ping 10.2.1.4 Pinging 10.2.1.4 with 32 bytes of data: Reply from 10.2.1.4:
bytes=32 time=2ms TTL=64 Reply from 10.2.1.4: bytes=32 time=1ms TTL=64 Reply from 10.2.1.4: bytes=32 time=1ms TTL=64 Reply from
10.2.1.4: bytes=32 time=2ms TTL=64 Ping statistics for 10.2.1.4: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate
round trip times in milli-seconds: Minimum = 1ms, Maximum = 2ms, Average = 1ms C:\Users\azureuser>
```

```
(10.2.1.4) 56(84) bytes of data. 64 bytes from 10.2.1.4: icmp_seq=1 ttl=64 time=1.42 ms 64 bytes from 10.2.1.4: icmp_seq=2 ttl=64
time=1.31 ms 64 bytes from 10.2.1.4: icmp_seq=3 ttl=64 time=1.28 ms 64 bytes from 10.2.1.4: icmp_seq=4 ttl=64 time=1.33 ms ---
10.2.1.4 ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3004ms rtt min/avg/max/mdev =
1.284/1.335/1.420/0.056 ms azureuser@VM-Linux-App:~$
```



Virtual machines

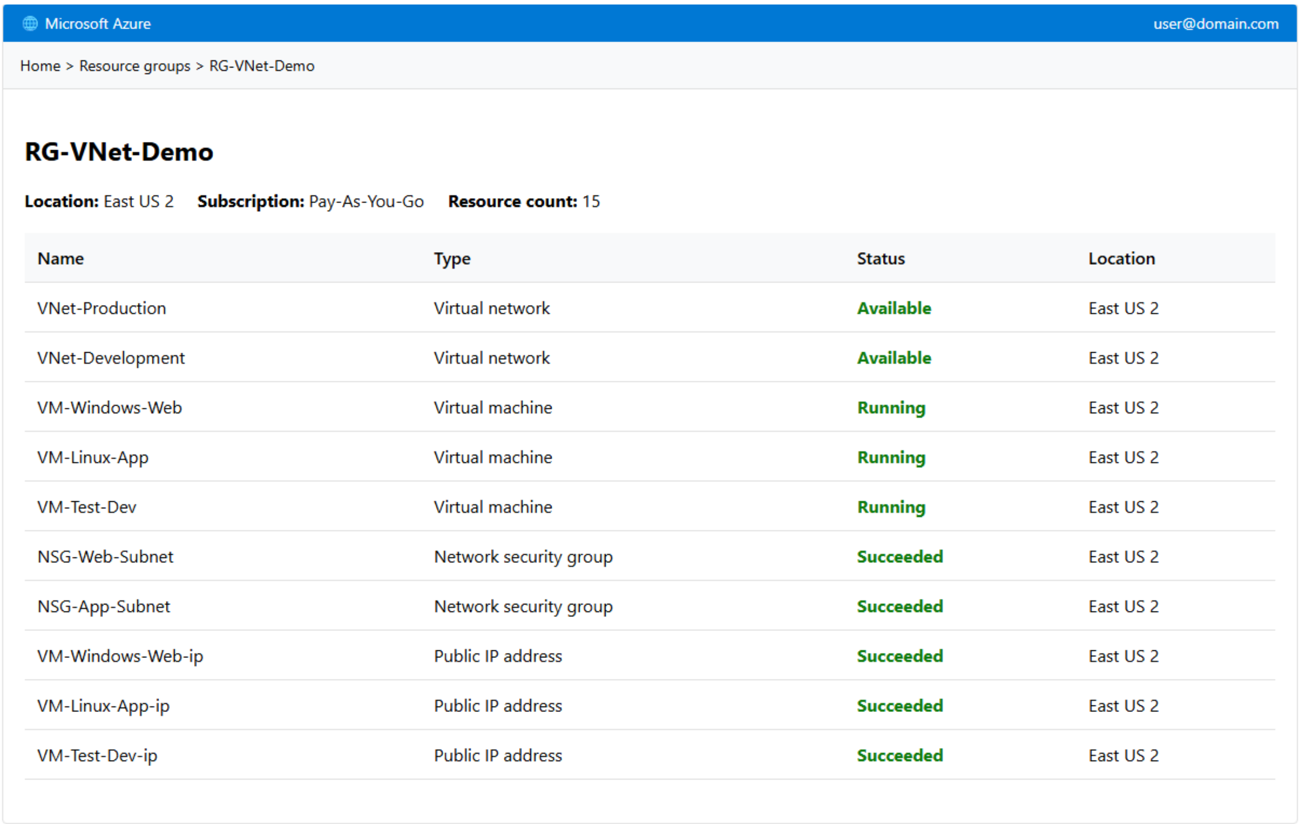
Name	Status	Location	Resource group	Public IP	Private IP	Virtual network/subnet
VM-Windows-Web	**Running**	East US 2	RG-VNet-Demo	20.62.156.23	10.1.1.4	VNet-Production/Web-Subnet
VM-Linux-App	**Running**	East US 2	RG-VNet-Demo	20.62.156.45	10.1.2.4	VNet-Production/App-Subnet
VM-Test-Dev	**Running**	East US 2	RG-VNet-Demo	20.62.156.67	10.2.1.4	VNet-Development/Dev-Subnet



Network Watcher - Connection Monitor

Connection Test Results

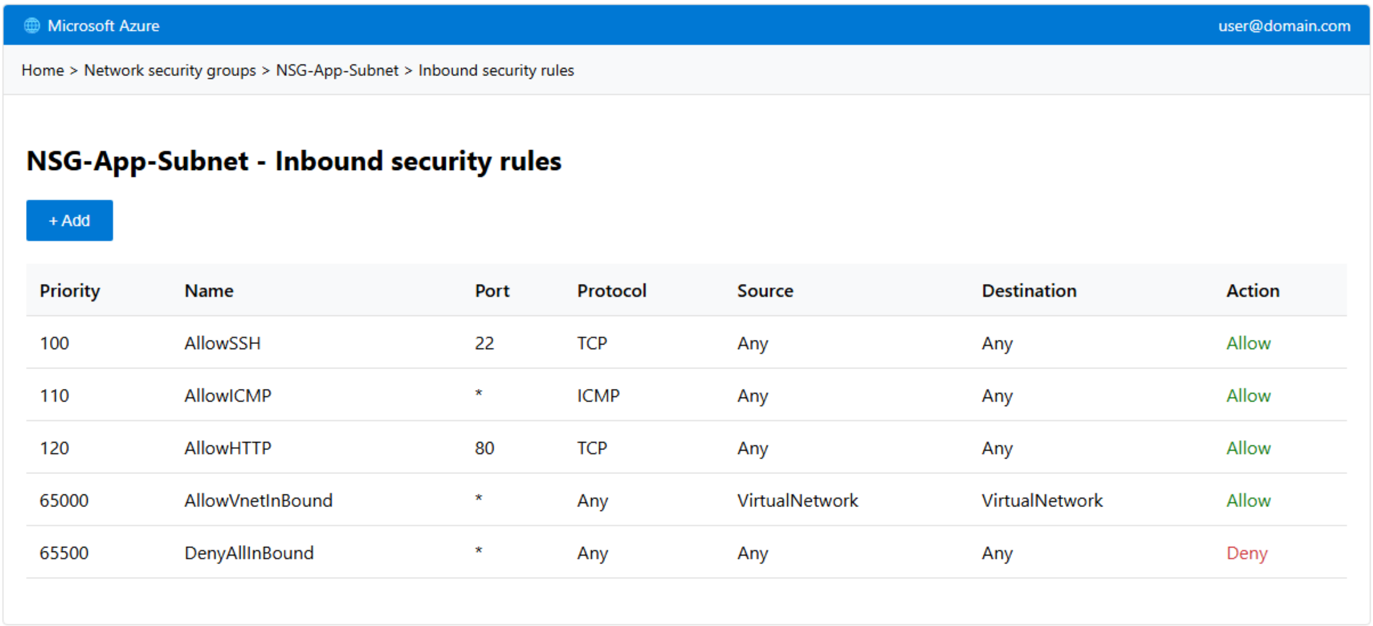
Source	Destination	Status	Avg Latency	Success Rate
VM-Windows-Web (10.1.1.4)	VM-Linux-App (10.1.2.4)	✓ Reachable	0.4 ms	100%
VM-Windows-Web (10.1.1.4)	VM-Test-Dev (10.2.1.4)	✓ Reachable	1.3 ms	100%
VM-Linux-App (10.1.2.4)	VM-Test-Dev (10.2.1.4)	✓ Reachable	1.2 ms	100%



RG-VNet-Demo

Location: East US 2 **Subscription:** Pay-As-You-Go **Resource count:** 15

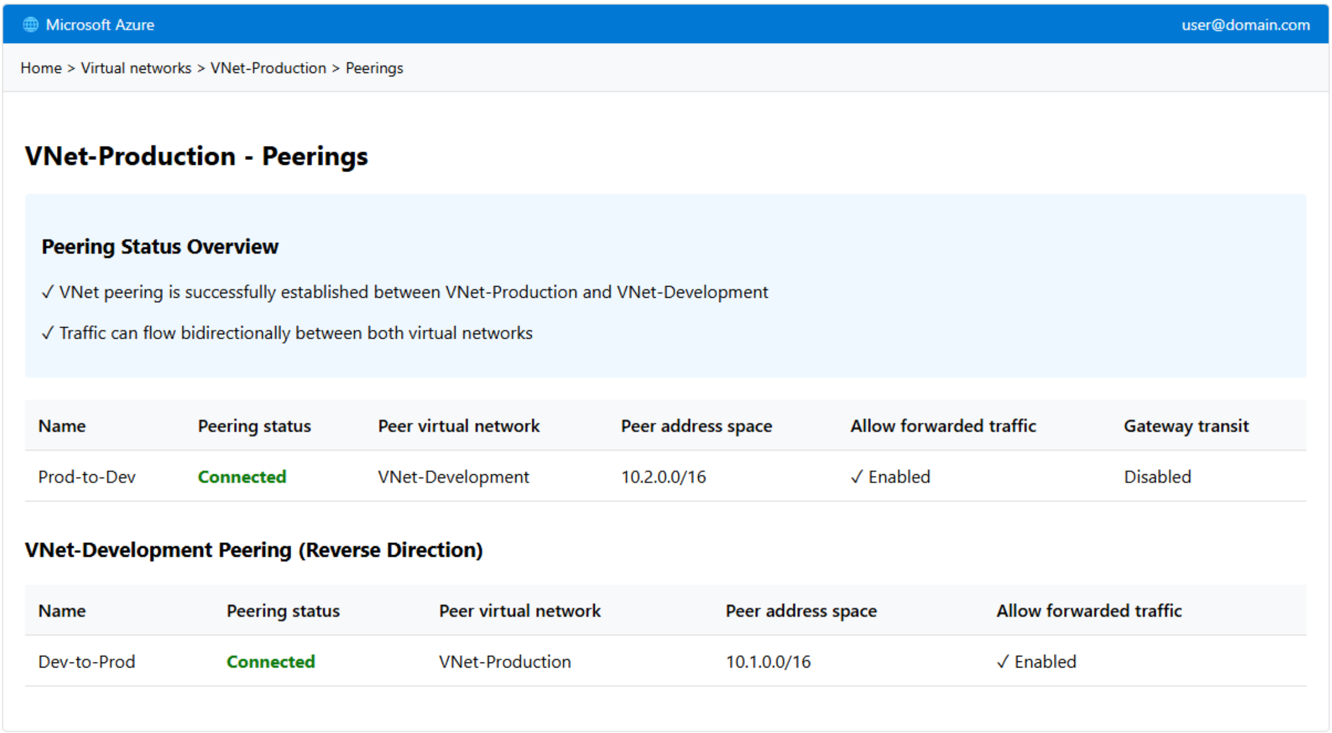
Name	Type	Status	Location
VNet-Production	Virtual network	Available	East US 2
VNet-Development	Virtual network	Available	East US 2
VM-Windows-Web	Virtual machine	Running	East US 2
VM-Linux-App	Virtual machine	Running	East US 2
VM-Test-Dev	Virtual machine	Running	East US 2
NSG-Web-Subnet	Network security group	Succeeded	East US 2
NSG-App-Subnet	Network security group	Succeeded	East US 2
VM-Windows-Web-ip	Public IP address	Succeeded	East US 2
VM-Linux-App-ip	Public IP address	Succeeded	East US 2
VM-Test-Dev-ip	Public IP address	Succeeded	East US 2



Microsoft Azure
user@domain.com
Home > Network security groups > NSG-App-Subnet > Inbound security rules

NSG-App-Subnet - Inbound security rules

Priority	Name	Port	Protocol	Source	Destination	Action
100	AllowSSH	22	TCP	Any	Any	Allow
110	AllowICMP	*	ICMP	Any	Any	Allow
120	AllowHTTP	80	TCP	Any	Any	Allow
65000	AllowVnetInBound	*	Any	VirtualNetwork	VirtualNetwork	Allow
65500	DenyAllInBound	*	Any	Any	Any	Deny



VNet-Production - Peerings

Peering Status Overview

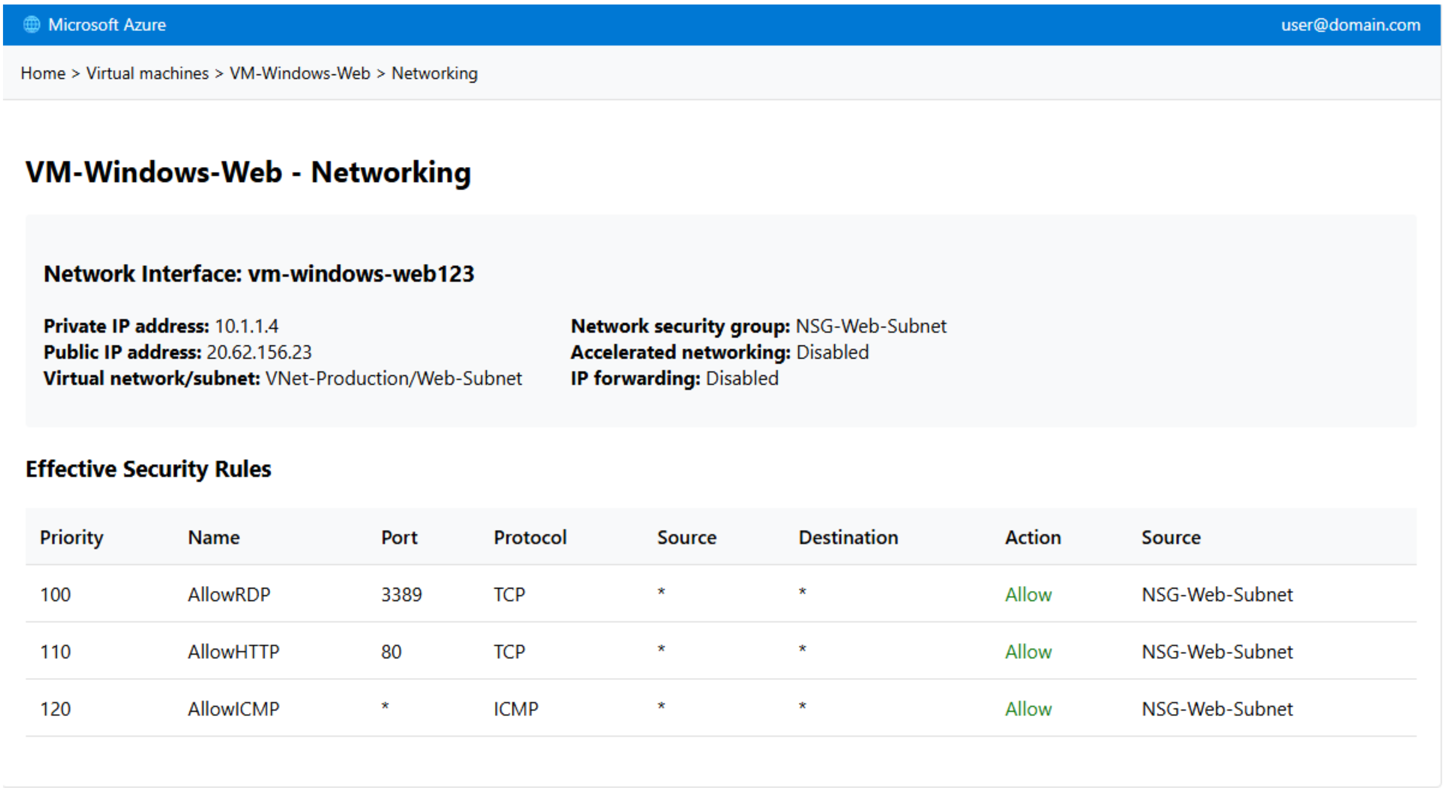
✓ VNet peering is successfully established between VNet-Production and VNet-Development

✓ Traffic can flow bidirectionally between both virtual networks

Name	Peering status	Peer virtual network	Peer address space	Allow forwarded traffic	Gateway transit
Prod-to-Dev	**Connected**	VNet-Development	10.2.0.0/16	✓ Enabled	Disabled

VNet-Development Peering (Reverse Direction)

Name	Peering status	Peer virtual network	Peer address space	Allow forwarded traffic
Dev-to-Prod	**Connected**	VNet-Production	10.1.0.0/16	✓ Enabled



VM-Windows-Web - Networking

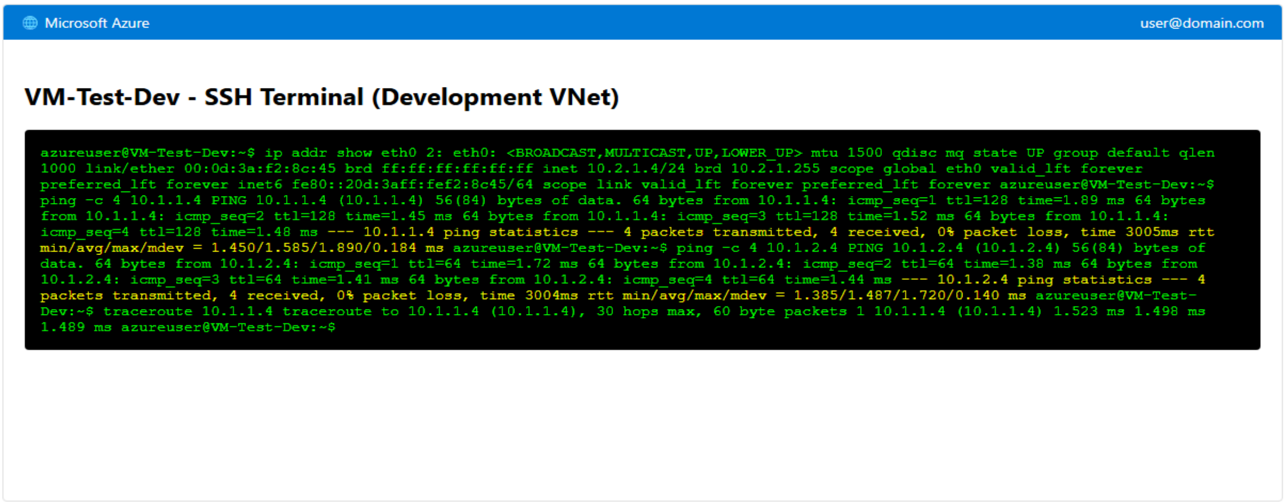
Network Interface: vm-windows-web123

Private IP address: 10.1.1.4
Public IP address: 20.62.156.23
Virtual network/subnet: VNet-Production/Web-Subnet

Network security group: NSG-Web-Subnet
Accelerated networking: Disabled
IP forwarding: Disabled

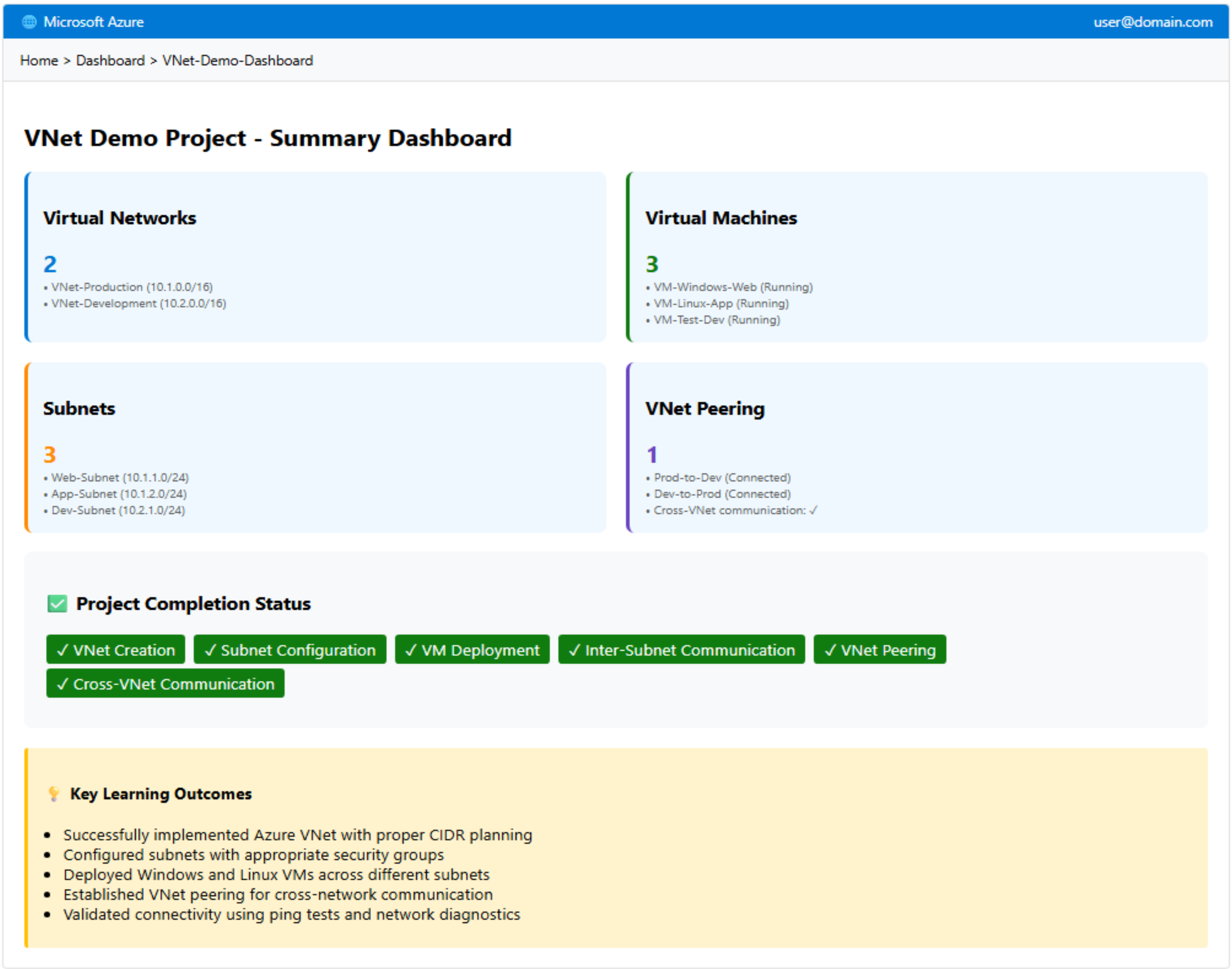
Effective Security Rules

Priority	Name	Port	Protocol	Source	Destination	Action	Source
100	AllowRDP	3389	TCP	*	*	Allow	NSG-Web-Subnet
110	AllowHTTP	80	TCP	*	*	Allow	NSG-Web-Subnet
120	AllowICMP	*	ICMP	*	*	Allow	NSG-Web-Subnet



VM-Test-Dev - SSH Terminal (Development VNet)

```
azureuser@VM-Test-Dev:~$ ip addr show eth0 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
1000 link/ether 00:0d:3a:f2:8c:45 brd ff:ff:ff:ff:ff:ff inet 10.2.1.4/24 brd 10.2.1.255 scope global eth0 valid_lft forever
preferred_lft forever inet6 fe80::20d:3aff:fef2:8c45/64 scope link valid_lft forever preferred_lft forever azureuser@VM-Test-Dev:~$
ping -c 4 10.1.1.4 PING 10.1.1.4 (10.1.1.4) 56(84) bytes of data. 64 bytes from 10.1.1.4: icmp_seq=1 ttl=128 time=1.89 ms 64 bytes
from 10.1.1.4: icmp_seq=2 ttl=128 time=1.45 ms 64 bytes from 10.1.1.4: icmp_seq=3 ttl=128 time=1.52 ms 64 bytes from 10.1.1.4:
icmp_seq=4 ttl=128 time=1.48 ms --- 10.1.1.4 ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3005ms rtt
min/avg/max/mdev = 1.450/1.585/1.890/0.184 ms azureuser@VM-Test-Dev:~$ ping -c 4 10.1.2.4 PING 10.1.2.4 (10.1.2.4) 56(84) bytes of
data. 64 bytes from 10.1.2.4: icmp_seq=1 ttl=64 time=1.72 ms 64 bytes from 10.1.2.4: icmp_seq=2 ttl=64 time=1.38 ms 64 bytes from
10.1.2.4: icmp_seq=3 ttl=64 time=1.41 ms 64 bytes from 10.1.2.4: icmp_seq=4 ttl=64 time=1.44 ms --- 10.1.2.4 ping statistics --- 4
packets transmitted, 4 received, 0% packet loss, time 3004ms rtt min/avg/max/mdev = 1.385/1.487/1.720/0.140 ms azureuser@VM-Test-
Dev:~$ traceroute 10.1.1.4 traceroute to 10.1.1.4 (10.1.1.4), 30 hops max, 60 byte packets 1 10.1.1.4 (10.1.1.4) 1.523 ms 1.498 ms
1.489 ms azureuser@VM-Test-Dev:~$
```



VNet Demo Project - Summary Dashboard

Virtual Networks

2

- VNet-Production (10.1.0.0/16)
- VNet-Development (10.2.0.0/16)

Virtual Machines

3

- VM-Windows-Web (Running)
- VM-Linux-App (Running)
- VM-Test-Dev (Running)

Subnets

3

- Web-Subnet (10.1.1.0/24)
- App-Subnet (10.1.2.0/24)
- Dev-Subnet (10.2.1.0/24)

VNet Peering

1

- Prod-to-Dev (Connected)
- Dev-to-Prod (Connected)
- Cross-VNet communication: ✓

✅ Project Completion Status

✓ VNet Creation ✓ Subnet Configuration ✓ VM Deployment ✓ Inter-Subnet Communication ✓ VNet Peering ✓ Cross-VNet Communication

💡 Key Learning Outcomes

- Successfully implemented Azure VNet with proper CIDR planning
- Configured subnets with appropriate security groups
- Deployed Windows and Linux VMs across different subnets
- Established VNet peering for cross-network communication
- Validated connectivity using ping tests and network diagnostics