Azure Hub-Spoke Network Topology

Complete Technical Guide

Ahmed Abdelwahed
ahmed@abdelwahed.me
www.abdelwahed.me
LinkedIn

LEVEL 1 — INTERMEDIATE: Foundation & Build

1. What Is the Hub-Spoke Topology?

The Hub-Spoke network topology is the recommended enterprise reference architecture in Azure. It separates shared network services (Hub) from individual workload environments (Spokes), connected via VNet Peering. This design mirrors how traditional campus networks function — but delivered as software-defined infrastructure in the cloud.

Hub VNet — Shared Services

- Azure Firewall or NVA (traffic inspection)
- VPN Gateway (site-to-site / P2S)
- ExpressRoute Gateway (private WAN)
- Azure Bastion (secure VM access)
- Private DNS Resolver
- Network Watcher
- DDoS Protection Plan

Spoke VNets — Workload Environments

- spoke-prod 10.1.0.0/16
- spoke-dev 10.2.0.0/16
- spoke-dmz 10.3.0.0/16
- spoke-identity 10.4.0.0/16
- spoke-data 10.5.0.0/16
- spoke-mgmt 10.6.0.0/16

1.1 IP Addressing Plan

Before creating any VNet, design your IP address plan carefully. Overlapping address spaces cannot be peered. Use RFC 1918 private ranges and allocate generously — VNet address spaces cannot be re-sized after resources are deployed in them.

VNet Name	Address Space	Purpose	Reserved Subnets
vnet-hub-eastus	10.0.0.0/16	Shared Services Hub	GatewaySubnet, AzureFirewallSubnet, AzureBastionSubnet
vnet-spoke-prod	10.1.0.0/16	Production workloads	snet-web, snet-app, snet-db, snet-mgmt
vnet-spoke-dev	10.2.0.0/16	Development workloads	snet-web, snet-app, snet-db
vnet-spoke-dmz	10.3.0.0/16	Internet-facing services	snet-frontend, snet-waf, snet-backend
vnet-spoke-data	10.5.0.0/16	Data & Analytics	snet-sql, snet-synapse, snet-storage

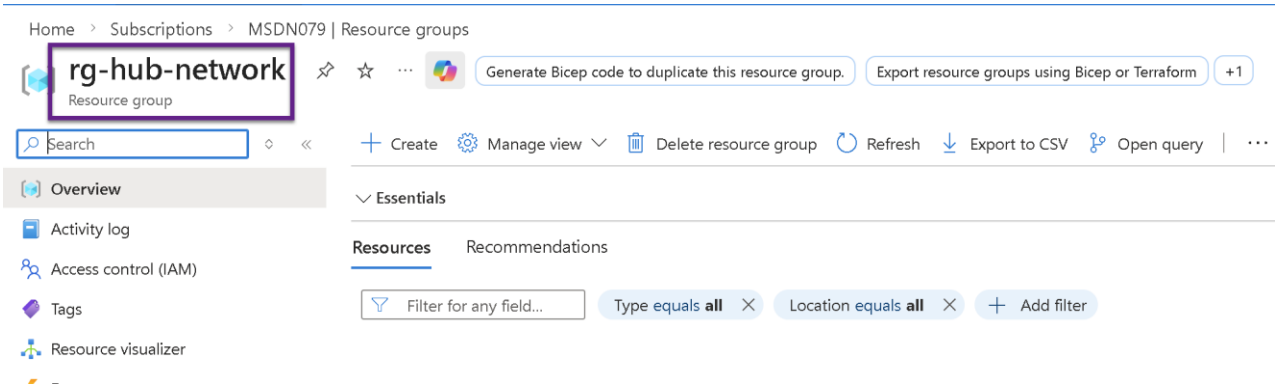
List and select your subscriptions

```
az account list --output table
az account set --subscription "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
```

1.2 Step-by-Step: Create the Hub VNet

Step 1 — Create the Hub Resource Group

```
# Azure CLI
az group create \
  --name rg-hub-network \
  --location eastus \
  --tags Environment=Hub  CostCenter=Network  Owner=Ahmed
```



Step 2 — Deploy the Hub VNet with Required Subnets

The Hub VNet requires three Azure-reserved subnet names: GatewaySubnet (VPN/ER Gateway), AzureFirewallSubnet (Azure Firewall), and AzureBastionSubnet. These names are mandatory and cannot be changed.

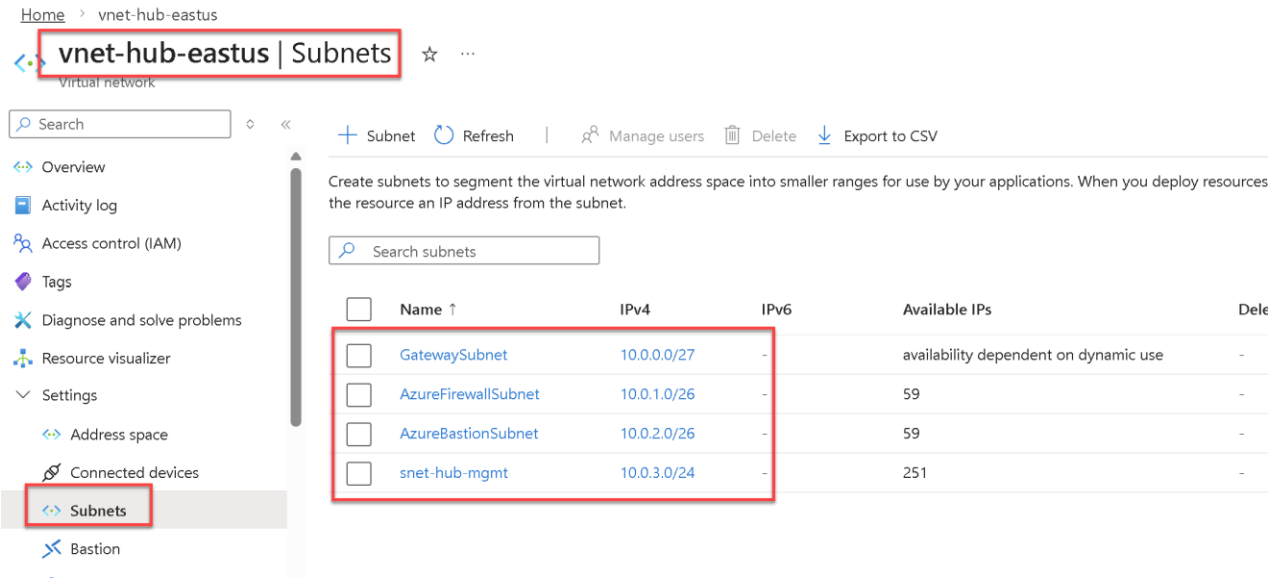
```
# Create Hub VNet
az network vnet create \
  --resource-group rg-hub-network \
  --name vnet-hub-eastus \
  --address-prefix 10.0.0.0/16 \
  --location eastus

# GatewaySubnet — required for VPN/ExpressRoute Gateway (/27 minimum)
az network vnet subnet create \
  --resource-group rg-hub-network \
  --vnet-name vnet-hub-eastus \
  --name GatewaySubnet \
  --address-prefix 10.0.0.0/27

# AzureFirewallSubnet — required for Azure Firewall (/26 minimum)
az network vnet subnet create \
  --resource-group rg-hub-network \
  --vnet-name vnet-hub-eastus \
  --name AzureFirewallSubnet \
  --address-prefix 10.0.1.0/26
```

```
# AzureBastionSubnet — required for Bastion (/26 minimum)
az network vnet subnet create \
  --resource-group rg-hub-network \
  --vnet-name vnet-hub-eastus \
  --name AzureBastionSubnet \
  --address-prefix 10.0.2.0/26

# Management subnet
az network vnet subnet create \
  --resource-group rg-hub-network \
  --vnet-name vnet-hub-eastus \
  --name snet-hub-mgmt \
  --address-prefix 10.0.3.0/24
```



> ⚠ **Subnet Size Requirements**
> GatewaySubnet minimum is /29 but Microsoft recommends /27 or larger to support future gateway SKU upgrades. AzureFirewallSubnet and AzureBastionSubnet both require a minimum of /26.

1.3 Step-by-Step: Create a Spoke VNet

```
# Create Spoke resource group
az group create \
  --name rg-spoke-prod \
  --location eastus \
  --tags Environment=Production  CostCenter=App  Owner=Ahmed

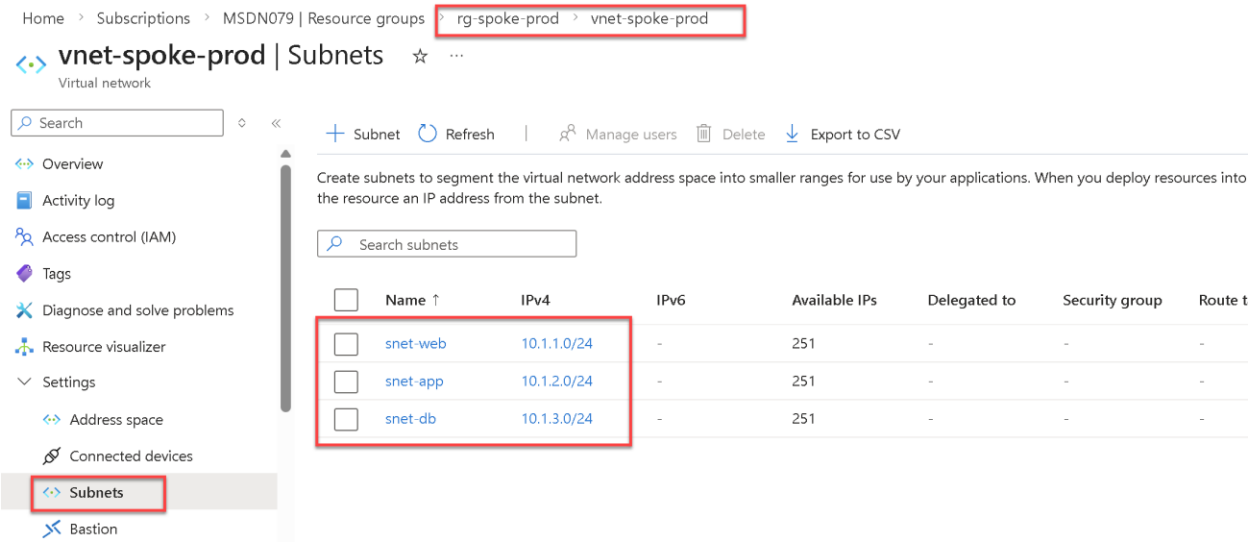
# Create Production Spoke VNet
az network vnet create \
  --resource-group rg-spoke-prod \
  --name vnet-spoke-prod \
  --address-prefix 10.1.0.0/16 \
  --location eastus

# Web tier subnet
az network vnet subnet create \
```

```
  --resource-group rg-spoke-prod \
  --vnet-name vnet-spoke-prod \
  --name snet-web \
  --address-prefix 10.1.1.0/24


# Application tier subnet
az network vnet subnet create \
  --resource-group rg-spoke-prod \
  --vnet-name vnet-spoke-prod \
  --name snet-app \
  --address-prefix 10.1.2.0/24

# Database tier subnet
az network vnet subnet create \
  --resource-group rg-spoke-prod \
  --vnet-name vnet-spoke-prod \
  --name snet-db \
  --address-prefix 10.1.3.0/24
```



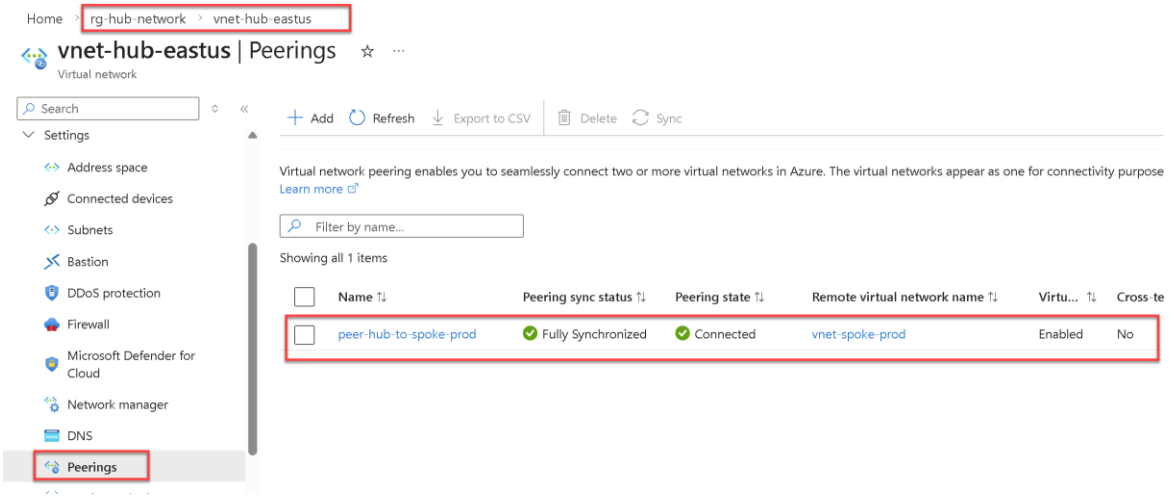
1.4 Establishing VNet Peering

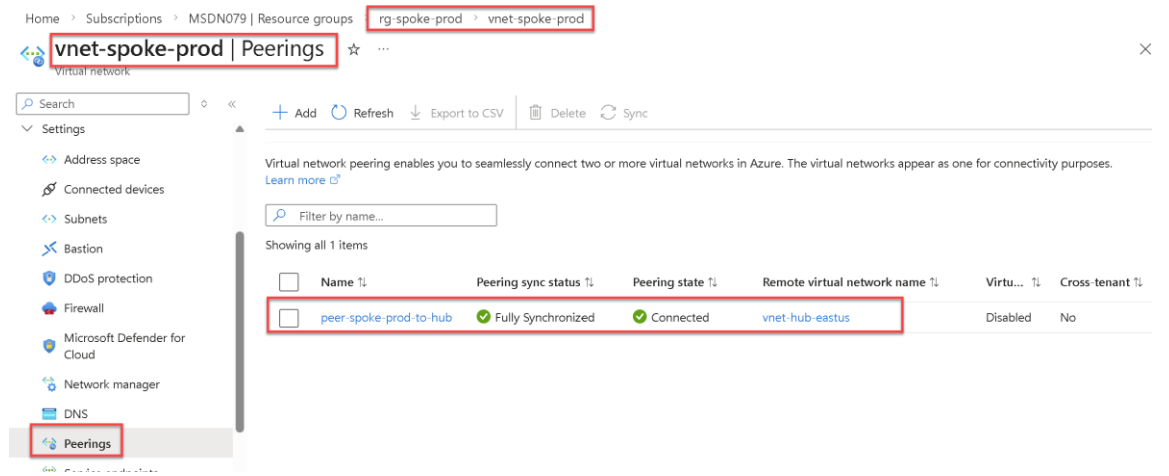
VNet Peering must be created in BOTH directions — from Hub to Spoke AND from Spoke to Hub. A one-sided peering shows as 'Initiated' and will not pass traffic until the other side is also configured.

```
# --- Peering: Hub → Spoke-Prod ---
az network vnet peering create \
  --resource-group rg-hub-network \
  --vnet-name vnet-hub-eastus \
  --name peer-hub-to-spoke-prod \
  --remote-vnet /subscriptions/b15d7-xxxxx-xxxxxx-xxxxxx-xxxxxxx/resourceGroups/rg-spoke-
prod/providers/Microsoft.Network/virtualNetworks/vnet-spoke-prod \
  --allow-vnet-access \
  --allow-forwarded-traffic \
  --allow-gateway-transit

# --- Peering: Spoke-Prod → Hub ---
az network vnet peering create \
  --resource-group rg-spoke-prod \
```

```
  --vnet-name vnet-spoke-prod \
  --name peer-spoke-prod-to-hub \
  --remote-vnet /subscriptions/b15d7-xxxxx-xxxxxx-xxxxxx-xxxxxxx/resourceGroups/rg-hub-network/providers/Microsoft.Network/virtualNetworks/vnet-hub-eastus \
  --allow-vnet-access \
  --allow-forwarded-traffic
  --use-remote-gateways #you can disable this option and add it later
```





Peering States Explained

'Initiated' = only one side created. 'Connected' = both sides created and healthy. Traffic ONLY flows when BOTH peers show 'Connected'. Always verify both sides after creating peering.

LEVEL 2 — ADVANCED: Security, Routing & Automation

2. Azure Firewall in the Hub

Azure Firewall Premium is the recommended centralized traffic inspection point in a Hub-Spoke topology. All traffic from spokes is forced through the firewall using User-Defined Routes (UDRs) — this ensures every east-west and north-south flow is inspected, logged, and controlled by policy.

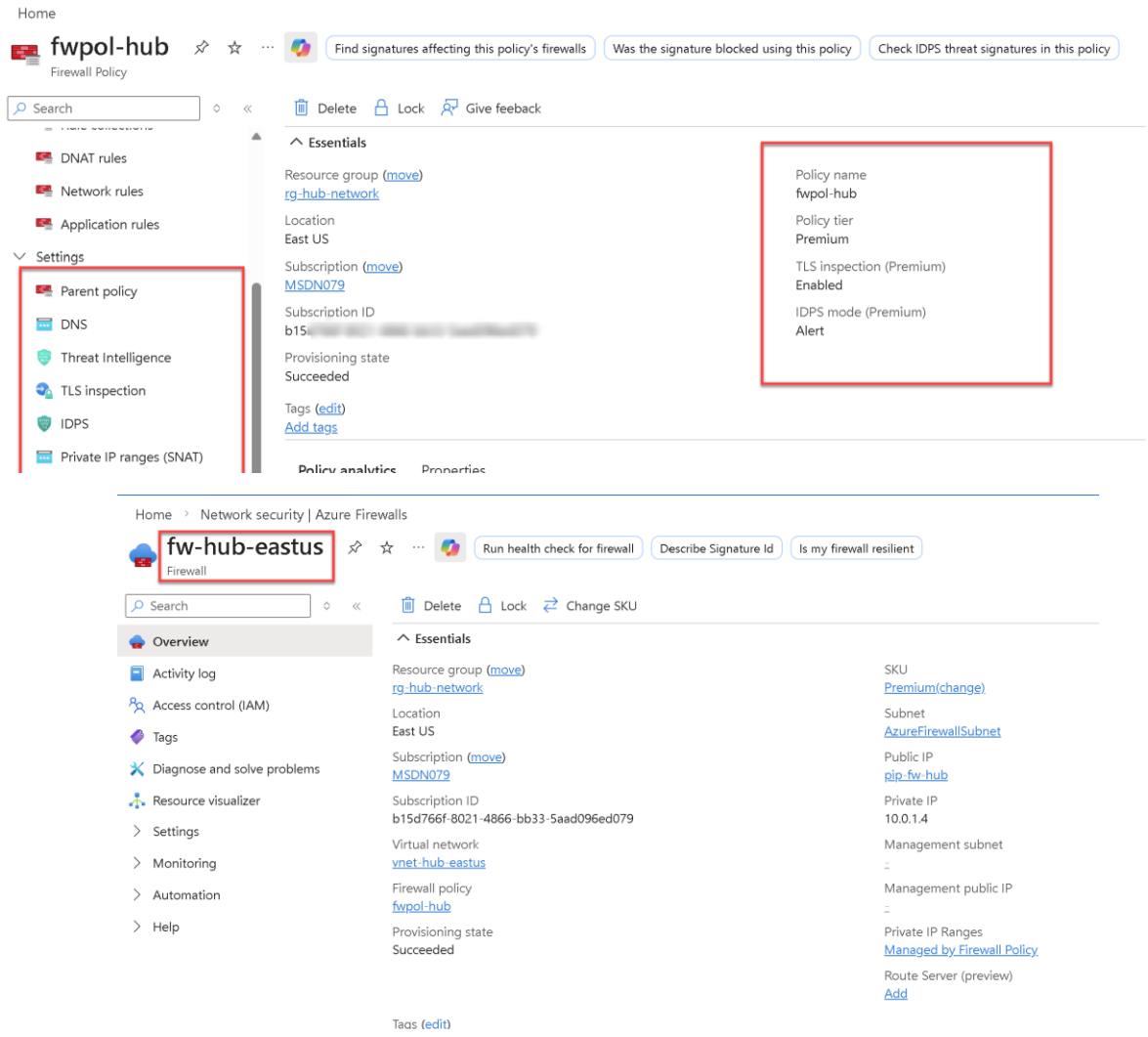
2.1 Deploy Azure Firewall Premium

```
# Create Public IP for Firewall
az network public-ip create \
  --resource-group rg-hub-network \
  --name pip-fw-hub \
  --sku Standard \
  --zone 1 2 3

# Create Firewall Policy (Premium tier enables IDPS & TLS inspection)
az network firewall policy create \
  --resource-group rg-hub-network \
  --name fwpol-hub \
  --sku Premium \
  --threat-intel-mode Alert

# Deploy Azure Firewall
az network firewall create \
  --resource-group rg-hub-network \
  --name fw-hub-eastus \
  --location eastus \
  --sku AZFW_VNet \
  --tier Premium \
  --firewall-policy fwpol-hub \
  --zones 1 2 3

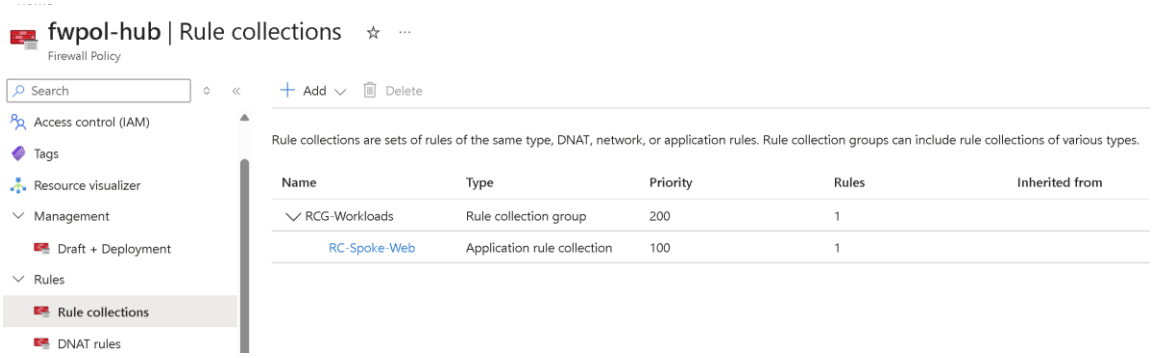
# Assign IP configuration
az network firewall ip-config create \
  --resource-group rg-hub-network \
  --firewall-name fw-hub-eastus \
  --name ipconfig-fw-hub \
  --vnet-name vnet-hub-eastus \
  --public-ip-address pip-fw-hub
```



2.2 Firewall Policy Rule Collections

Firewall policies are organized in Rule Collection Groups → Rule Collections → Individual Rules. Traffic is evaluated top-down with explicit Deny taking priority. Always add a final Deny-All rule to ensure no traffic passes uninspected.

Rule Collection Group	Priority	Rule Collection	Rule Type	Action
RCG-Infrastructure	100	RC-DNS	Network	Allow
RCG-Infrastructure	100	RC-NTP	Network	Allow
RCG-Workloads	200	RC-Spoke-Web	Application	Allow
RCG-Workloads	200	RC-Spoke-DB	Network	Allow
RCG-EastWest	300	RC-Spoke-to-Spoke	Network	Deny
RCG-Internet	400	RC-WebCategories	Application	Allow
RCG-Default	65000	RC-Deny-All	Network	Deny



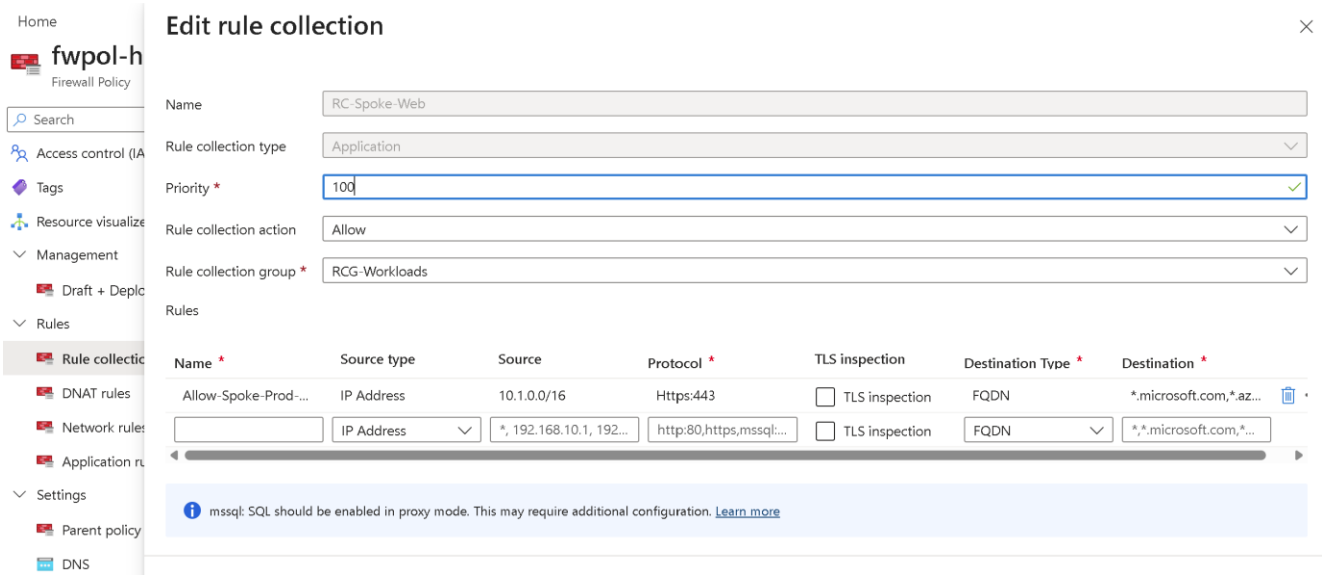
Sample Application Rule — Allow HTTPS for Spoke-Prod

```
# PowerShell — Add Application Rule to Firewall Policy
$fwPolicy = Get-AzFirewallPolicy -Name 'fwpol-hub' -ResourceGroupName 'rg-hub-network'

$appRule = New-AzFirewallPolicyApplicationRule `
  -Name 'Allow-Spoke-Prod-HTTPS' `
  -SourceAddress '10.1.0.0/16' `
  -TargetFqdn '*.microsoft.com','*.azure.com','*.windows.net' `
  -Protocol 'https:443'

$ruleCollection = New-AzFirewallPolicyFilterRuleCollection `
  -Name 'RC-Spoke-Web' `
  -Priority 100 `
  -Rule $appRule `
  -ActionType 'Allow'

$rcGroup = New-AzFirewallPolicyRuleCollectionGroup `
  -Name 'RCG-Workloads' `
  -Priority 200 `
  -RuleCollection $ruleCollection `
  -FirewallPolicyObject $fwPolicy
```



3. User-Defined Routes (UDR) — Force Tunneling Through Firewall

By default, Azure routes traffic between subnets using system routes. UDRs override this behavior, forcing all spoke traffic to the Azure Firewall's private IP (10.0.1.4) as the next hop. This is the key mechanism that makes centralized inspection work.

3.1 Understanding Route Priority

Route Type	Source	Priority	Can Override?
System routes	Azure (automatic)	Lowest	Yes, by UDR
User-Defined Routes	Administrator	High	Yes, by BGP (if allowed)
BGP routes	VPN/ExpressRoute	Medium	Yes, by UDR
Service Tag routes	Azure services	Varies	Yes, by UDR

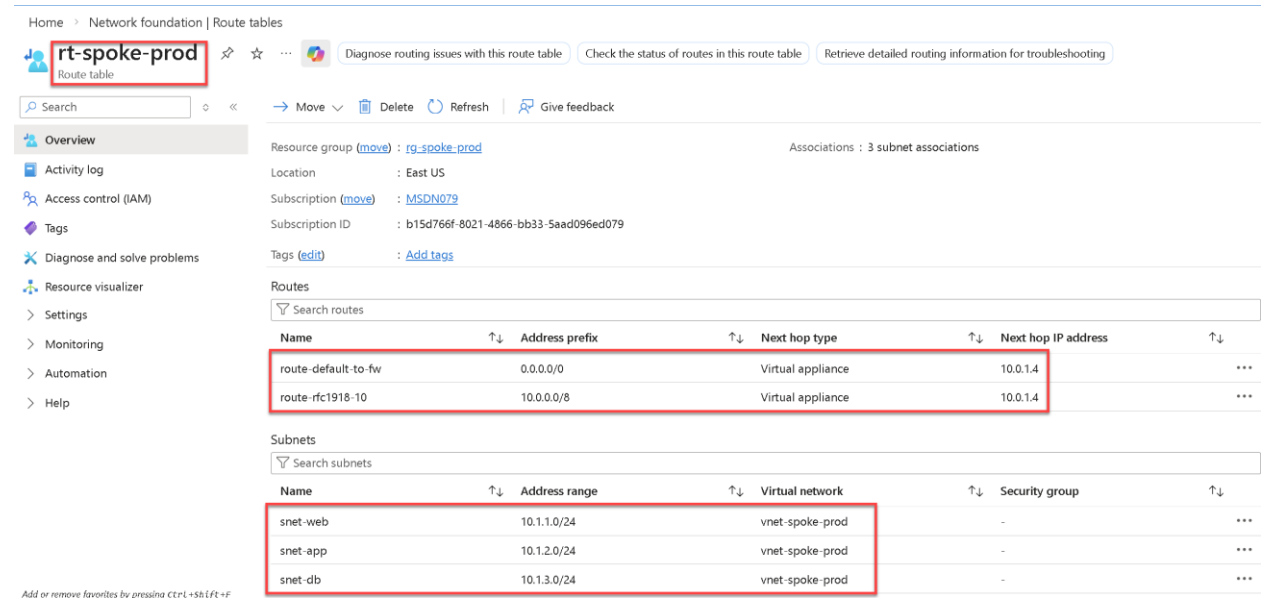
3.2 Create UDR for Spoke-Prod

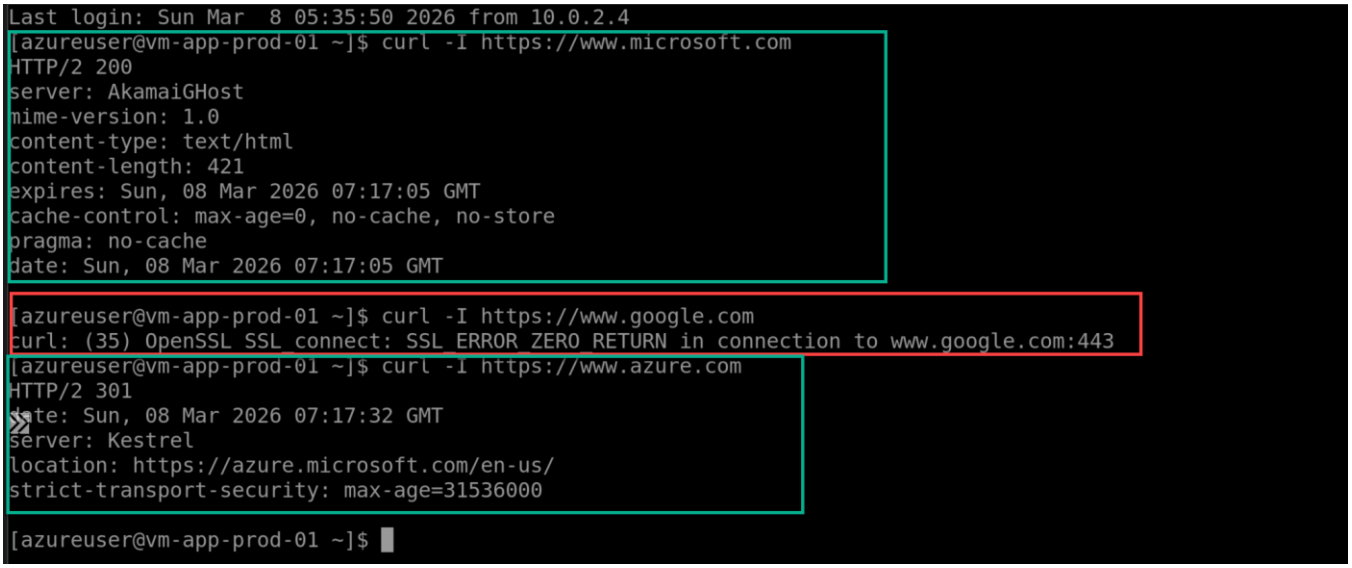
```
# Create Route Table for Spoke-Prod
az network route-table create \
  --resource-group rg-spoke-prod \
  --name rt-spoke-prod \
  --location eastus \
  --disable-bgp-route-propagation true

# Default route → Azure Firewall (0.0.0.0/0 forces ALL traffic through firewall)
az network route-table route create \
  --resource-group rg-spoke-prod \
  --route-table-name rt-spoke-prod \
  --name route-default-to-fw \
  --address-prefix 0.0.0.0/0 \
  --next-hop-type VirtualAppliance \
  --next-hop-ip-address 10.0.1.4

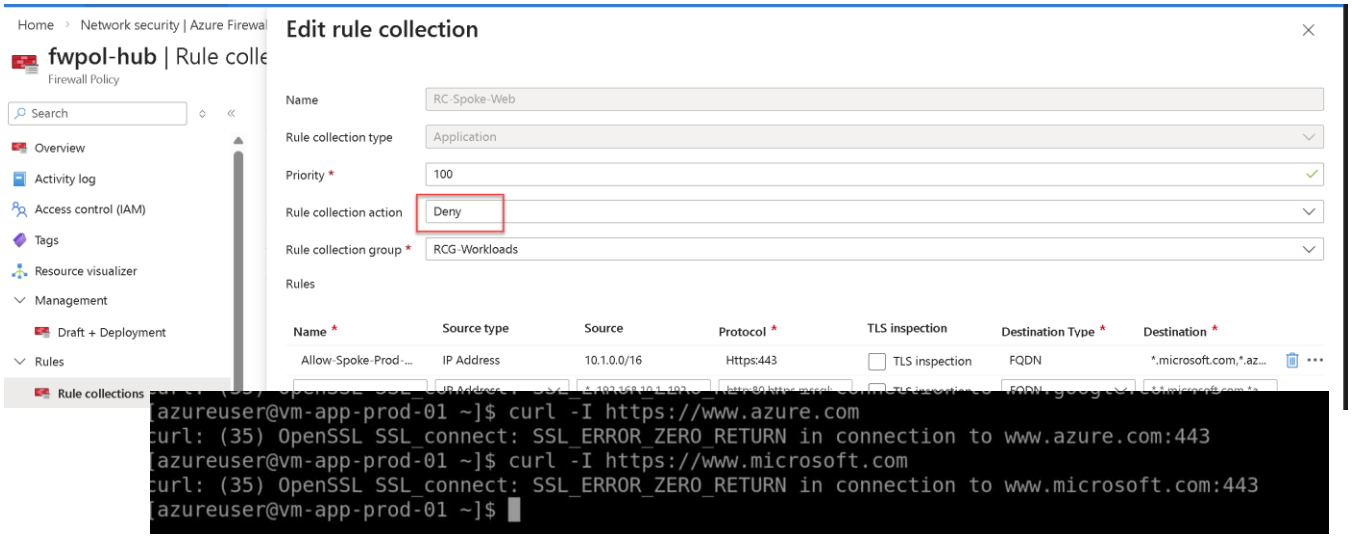
# RFC 1918 routes (ensure all private traffic goes through firewall)
az network route-table route create \
  --resource-group rg-spoke-prod \
  --route-table-name rt-spoke-prod \
  --name route-rfc1918-10 \
  --address-prefix 10.0.0.0/8 \
  --next-hop-type VirtualAppliance \
  --next-hop-ip-address 10.0.1.4

# Associate route table with spoke subnets
for subnet in snet-web snet-app snet-db; do
  az network vnet subnet update \
    --resource-group rg-spoke-prod \
    --vnet-name vnet-spoke-prod \
    --name $subnet \
    --route-table rt-spoke-prod
done
```





If you set deny action, all traffic will blocked



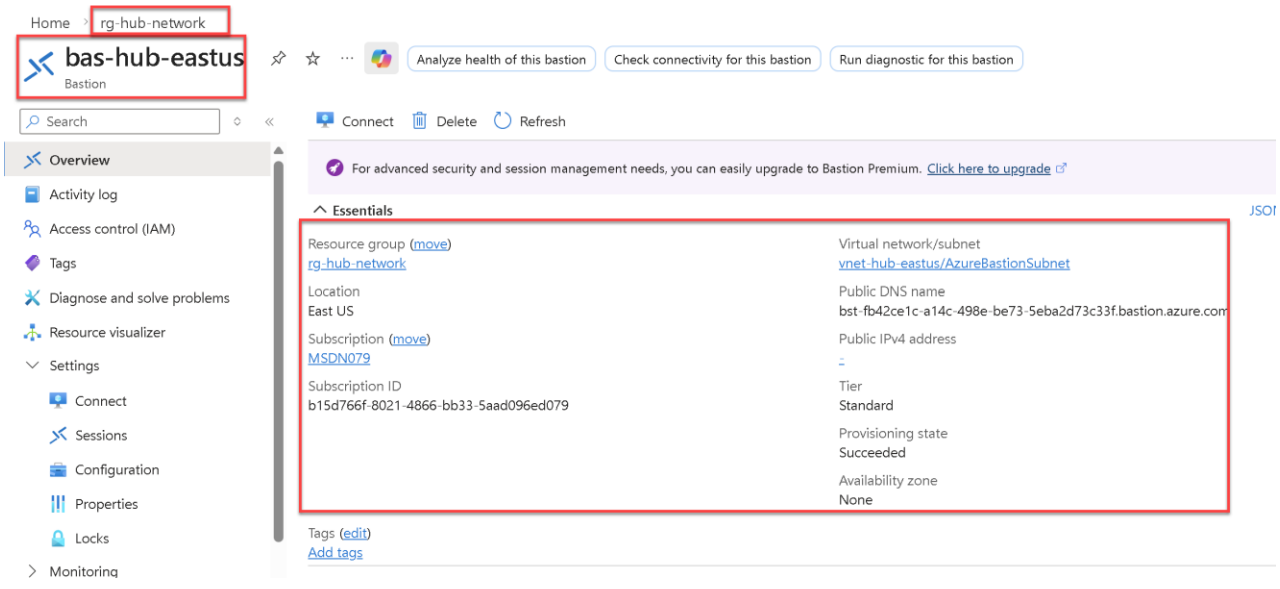
4. Azure Bastion — Secure VM Access

Azure Bastion provides browser-based RDP/SSH access to VMs without exposing them to the internet. Deployed in the Hub's AzureBastionSubnet, it provides connectivity to VMs across ALL peered spokes — no need to deploy Bastion per spoke.

4.1 Deploy Azure Bastion (Standard SKU)

```
# Create Bastion Public IP
az network public-ip create \
  --resource-group rg-hub-network \
  --name pip-bastion-hub \
  --sku Standard \
  --zone 1 2 3

# Deploy Bastion — Standard SKU enables native client, tunneling, and shareable links
az network bastion create \
  --resource-group rg-hub-network \
  --name bas-hub-eastus \
  --vnet-name vnet-hub-eastus \
  --public-ip-address pip-bastion-hub \
  --sku Standard \
  --enable-tunneling true \
  --enable-ip-connect true
```



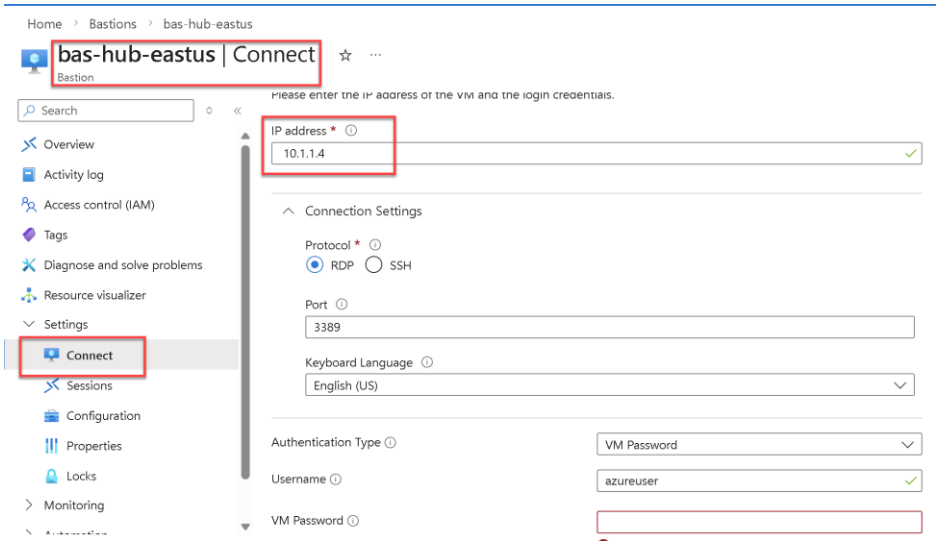
4.2 Connectivity using Bastion: Provision Two VMs One for Windows Server and Another for Rocky Linux

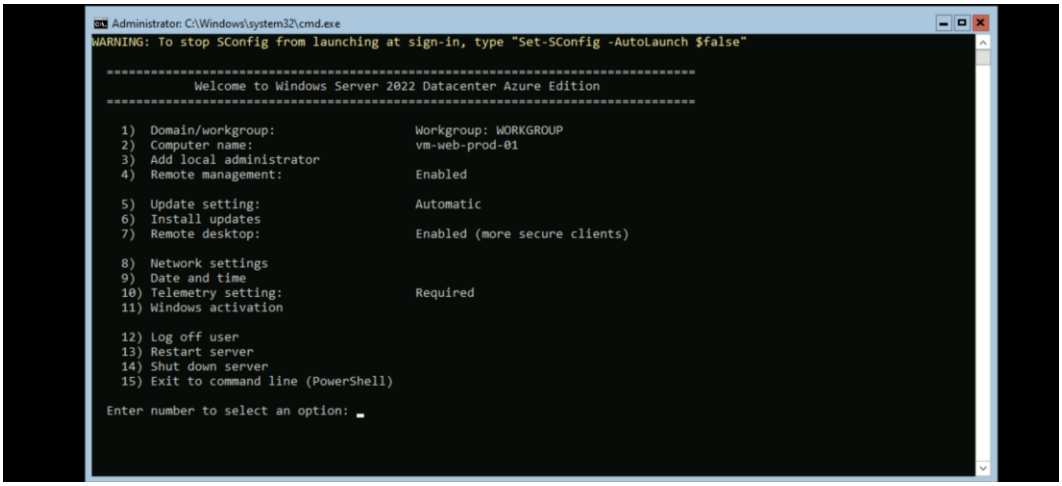
```
# Create Windows server VM in spoke
az vm create \
  --resource-group rg-spoke-prod \
  --name vm-web-prod-01 \
  --image Win2022AzureEditionCore \
  --size Standard_B2ms \
  --admin-username azureuser \
  --subnet snet-web \
  --vnet-name vnet-spoke-prod \
  --public-ip-sku Standard

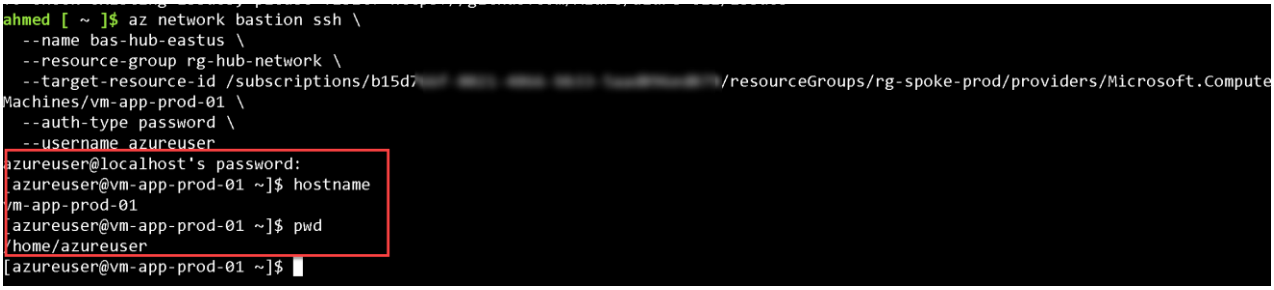
# Create Rocky Linux VM in spoke
az vm create \
  --resource-group rg-spoke-prod \
  --name vm-app-prod-01 \
  --image resf:rockylinux-x86_64:9-base:9.6.20250531 \
  --size Standard_B2ms \
  --admin-username azureuser \
  --authentication-type password \
  --subnet snet-app \
  --vnet-name vnet-spoke-prod \
  --public-ip-sku Standard
```

4.3 Connect to Spoke VM via Bastion

Because Bastion is in the Hub and the Hub is peered to all spokes, you can connect to any spoke VM from the single Bastion deployment. No public IP on the VM is required.





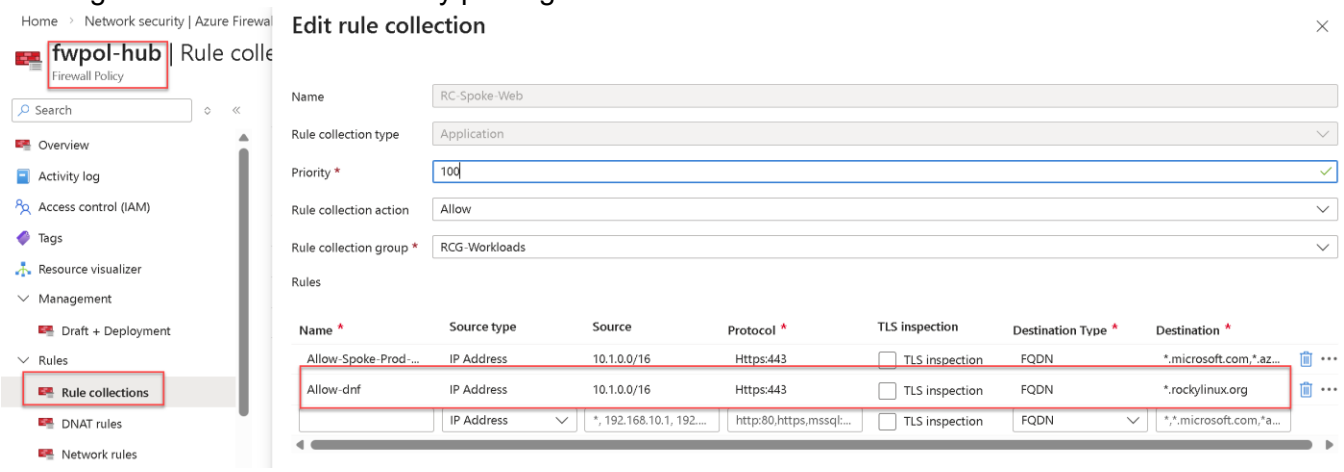


If you delete the peering, you won't be able to reconnect.

> **Cost Optimisation**
> One Bastion Standard SKU in the Hub covers all spoke VMs via peering. Deploying individual Bastion instances per spoke is unnecessary and significantly more expensive. The Hub-Spoke model makes this a natural cost saving.

4.4 Set up an Nginx website and permit only inbound traffic via the firewall

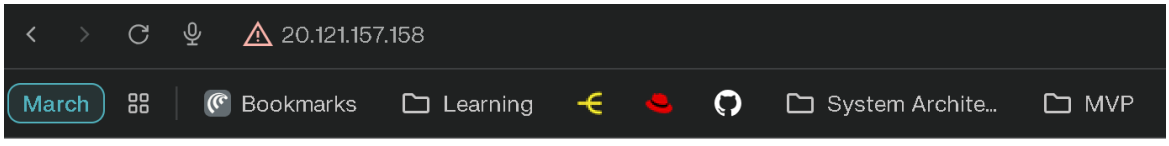
Initially, I include an additional role that enables outbound connections to the Rocky Linux repositories, allowing us to install the necessary packages.




```
# Install Nginx and simple site
sudo dnf install -y nginx
sudo systemctl enable --now nginx
curl -I http://127.0.0.1

# Create DNAT rule collection group (once)
az network firewall policy rule-collection-group create \
  --resource-group rg-hub-network \
  --policy-name fwpol-hub \
  --name RCG-DNAT \
  --priority 100

# Add DNAT rule to publish Nginx (NSG not required)
az network firewall policy rule-collection-group collection add-nat-collection \
  --resource-group rg-hub-network \
  --policy-name fwpol-hub \
  --rule-collection-group-name RCG-DNAT \
  --name RC-Nginx-DNAT \
  --collection-priority 100 \
  --action DNAT \
  --rule-name Allow-Nginx-HTTP \
  --source-addresses '*' \
  --destination-addresses 20.121.157.158 \
  --destination-ports 80 \
  --ip-protocols TCP \
  --translated-address 10.1.2.4 \
  --translated-port 80
```



Hello from Nginx via Azure Firewall

Also, if you remove the peering, the connection will go down.