# Group1 Project Report – Gloria Ayoola

# Deployment of a VNet-Based Multi-Tier Architecture

## Objectives

This project aimed to design and deploy a Virtual Network (VNet)-based multi-tier architecture on Microsoft Azure to support secure, scalable, and high-performance cloud application hosting. The deployment aimed to demonstrate segmentation of workloads into web, application, and database tiers while implementing best practices in networking, security, and resource management.

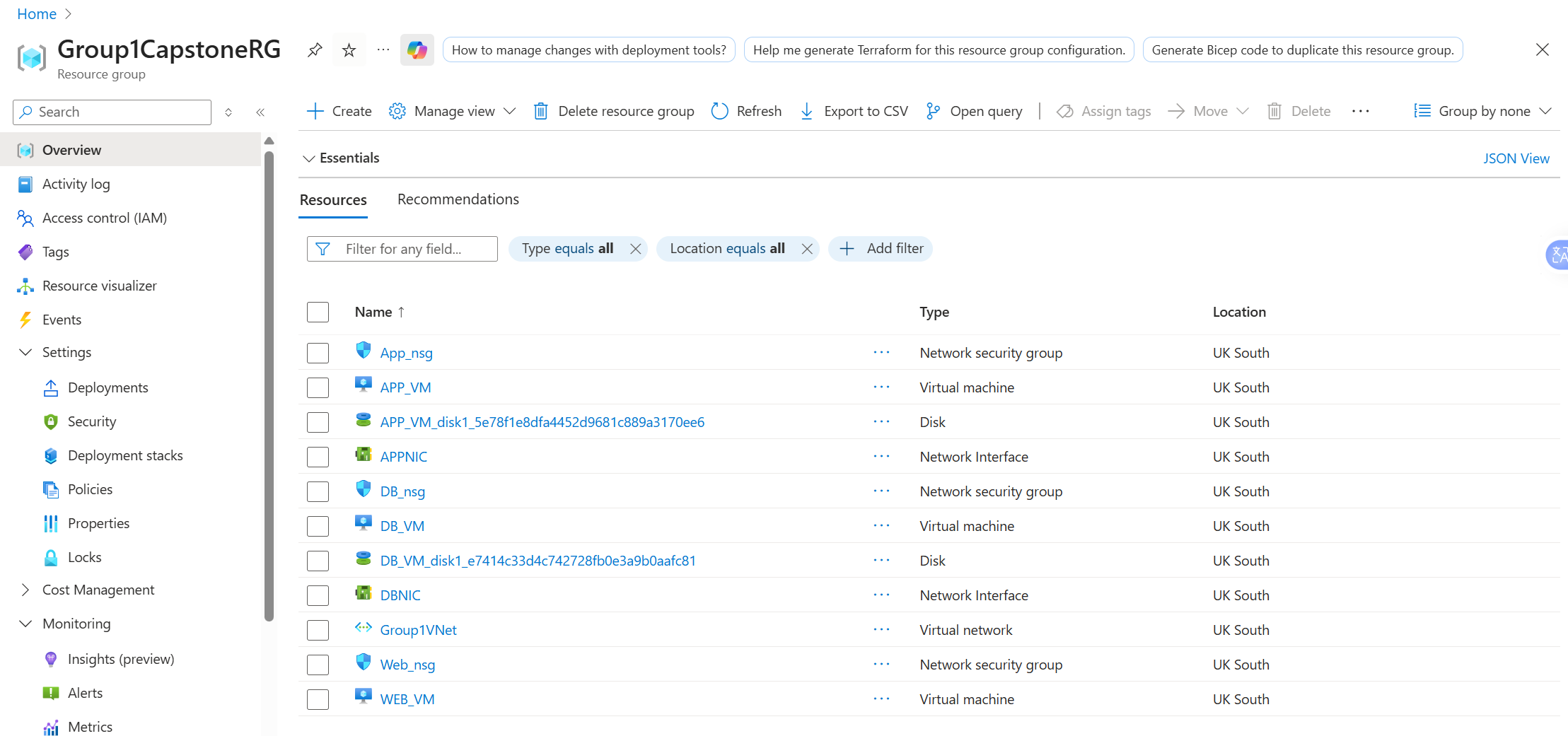
## Steps Taken

1. Created a resource group to host all project resources within a defined Azure region.
2. Configured a Virtual Network (VNet) and divided it into three subnets representing the web, application, and database tiers.
3. Deployed Network Security Groups (NSGs) for each subnet to control inbound and outbound traffic using security rules allowing necessary communication (Web → App → DB).
4. Provision virtual machines in each subnet representing the tiers: a web server (frontend), an application server (middleware), and a database server (backend).
5. Deploy NIC for each virtual machine
6. SSH into each VM and verify connectivity rules (e.g., ping tests)
7. Write a Bash script to automate the full deployment.

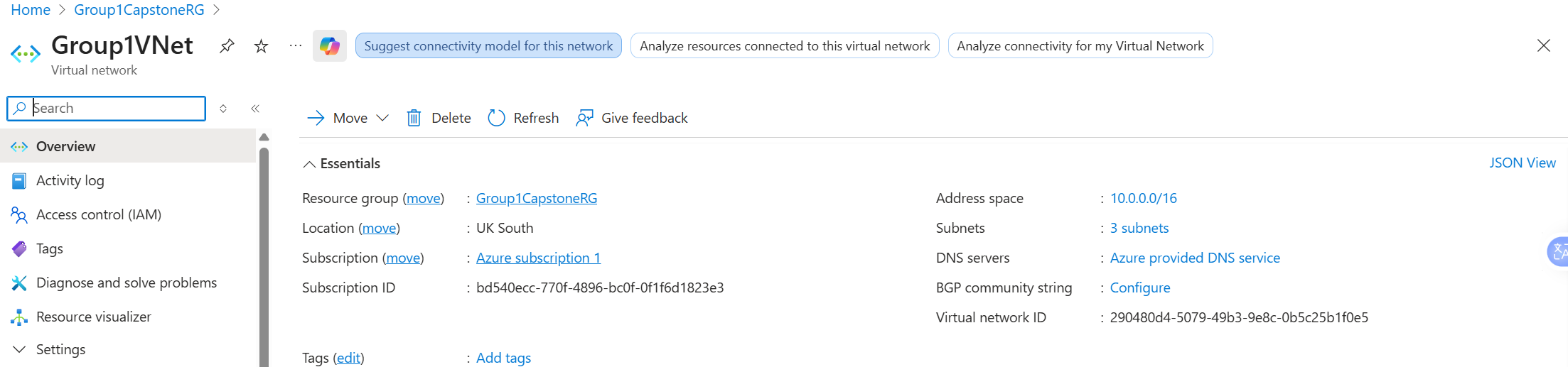
## Results

The deployment successfully created a functional multi-tier architecture within Azure as shown in the screenshots below. Each tier was properly isolated using subnetting and NSGs, and secure communication was achieved between tiers. The web tier was publicly accessible, while the application and database tiers were kept private. Testing confirmed successful communication between the multi-tier architecture.

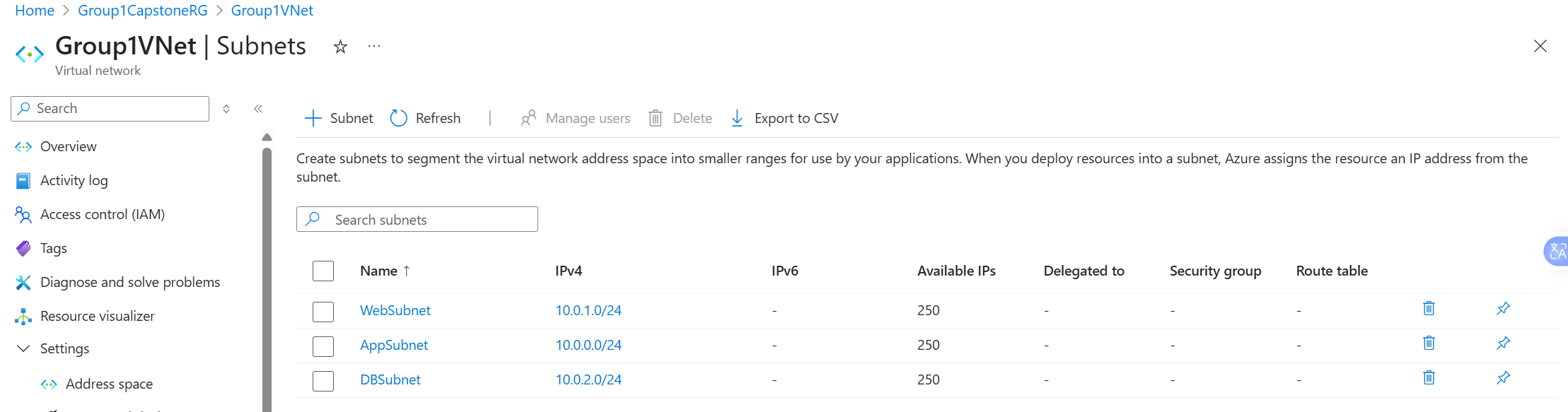
**RESOURCE GROUP – Group1CapstoneRG**



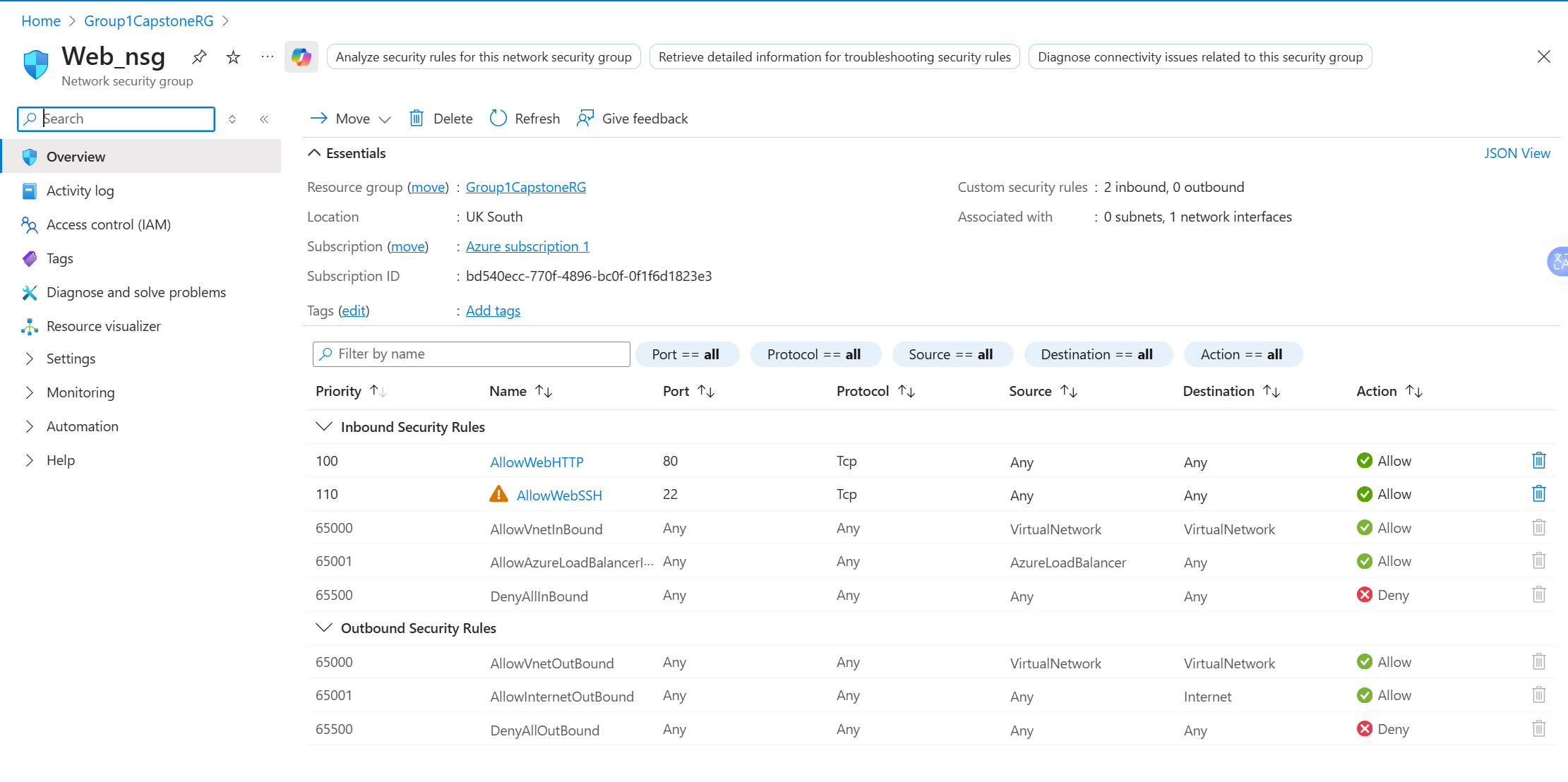
**CREATION OF VRITUAL NETWORK (Group1VNet) WITH THREE SUBNETS**



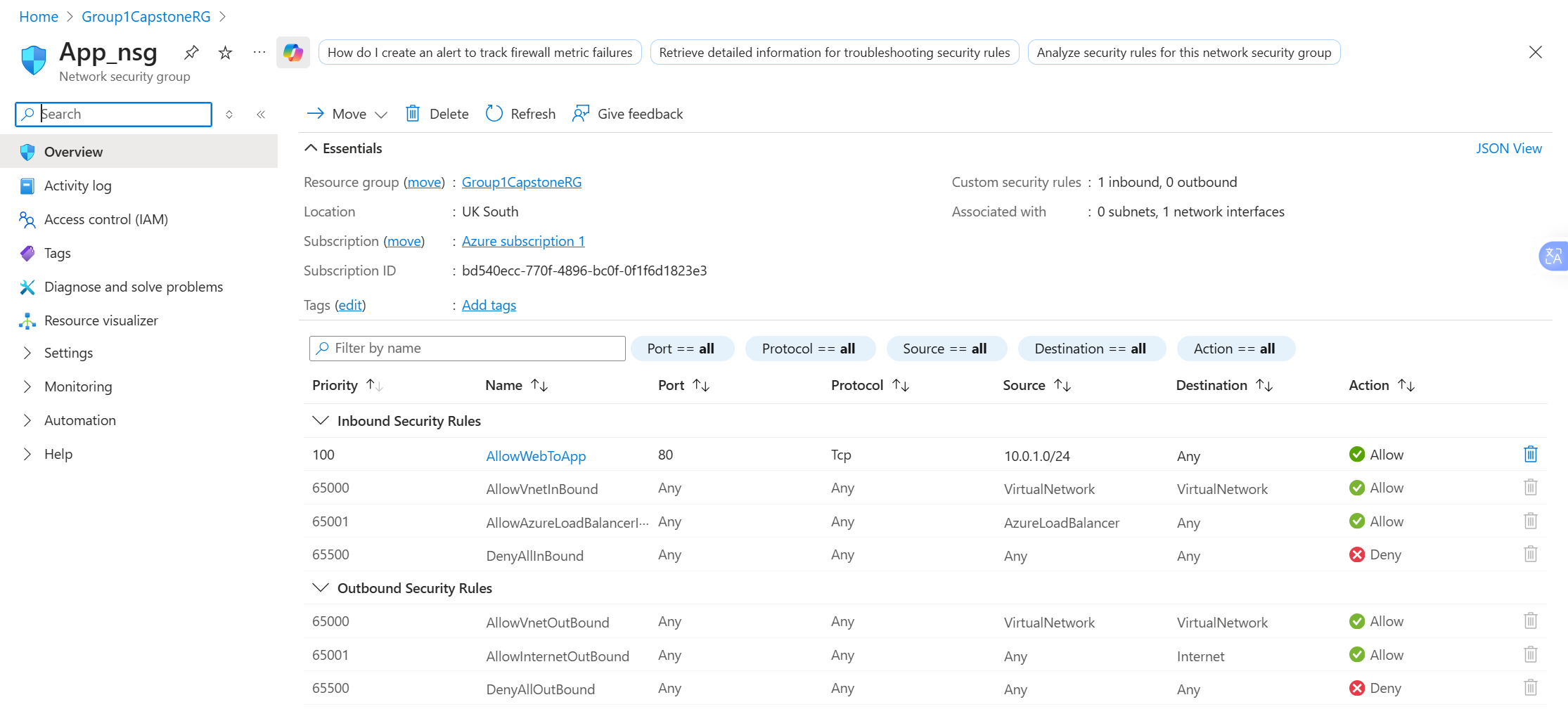
**THREE SUBNETS – WebSubnet, Appsubnet AND DBSubnet CREATED**



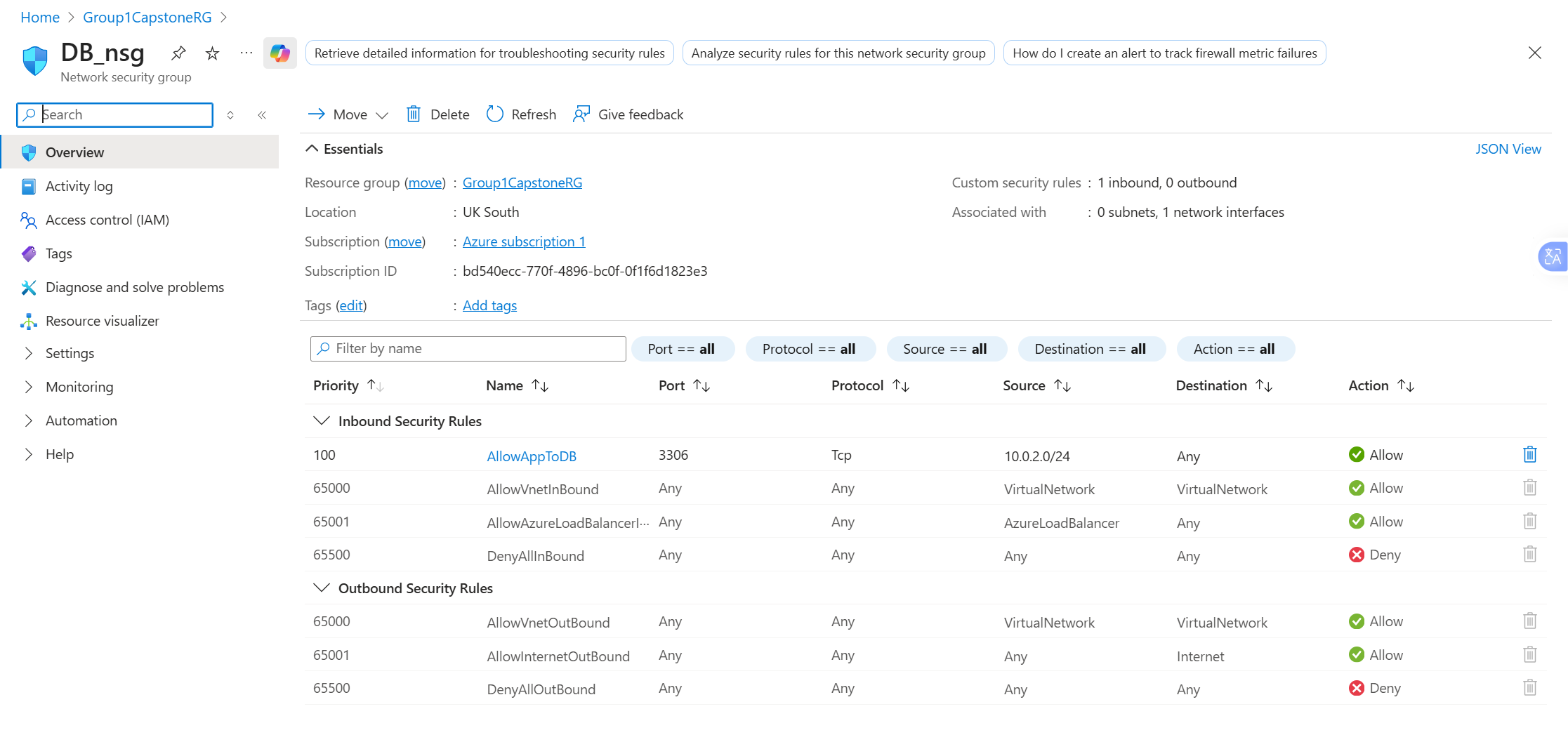
WEB NETWORK SECURITY GROUP (Web\_nsg) created with the tier rules



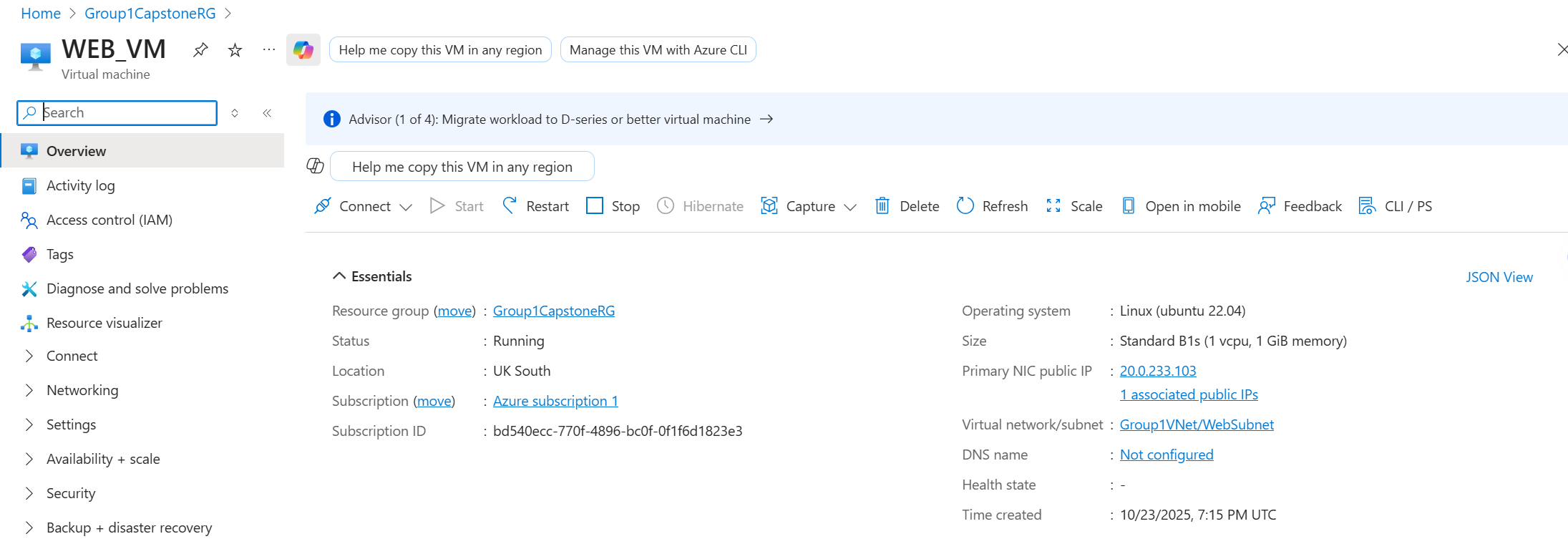
APP NETWORK SECURITY GROUP CREATED (App\_Nsg) WITH THE TIER RULE.



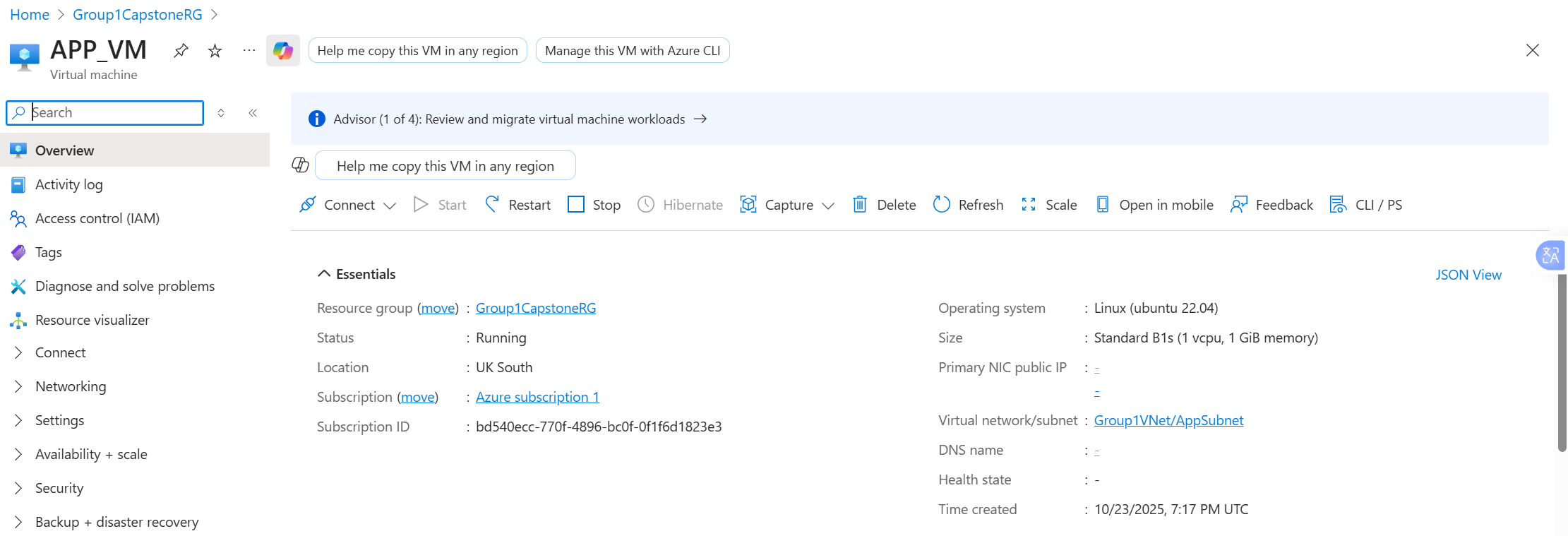
DB NETWORK SECURITY GROUP CREATED (DB\_nsg) WITH THE TIER RULE



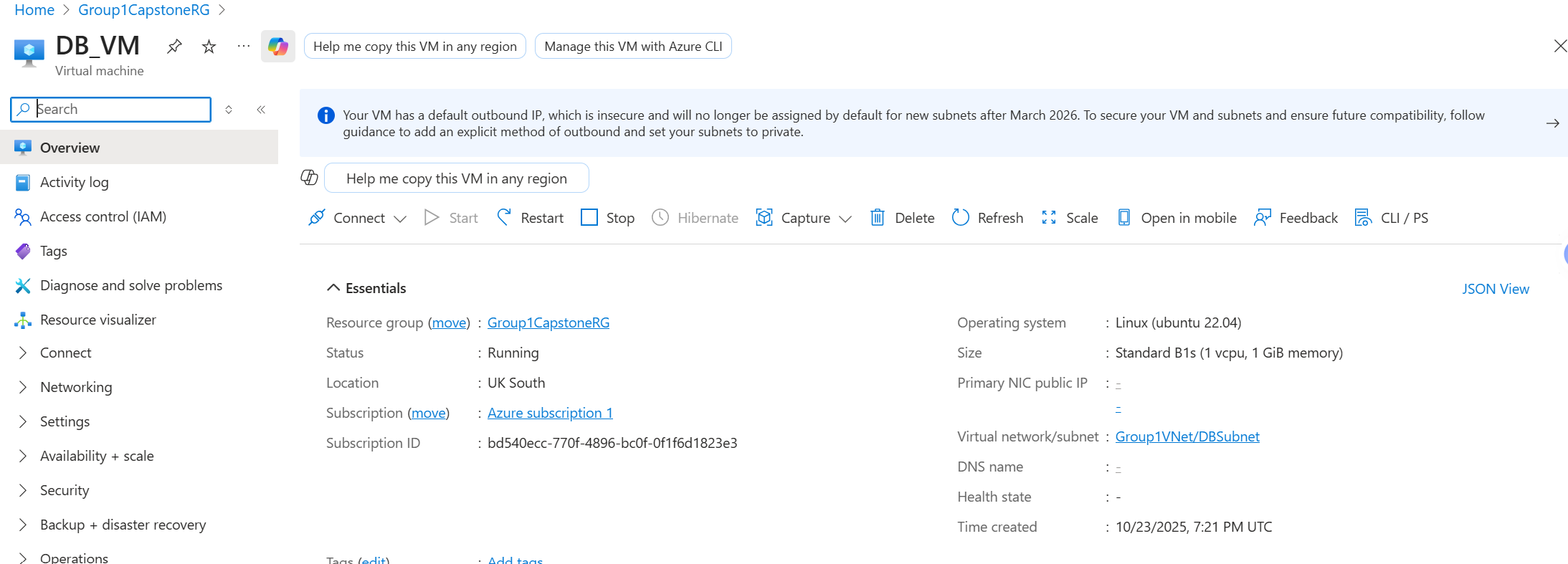
DEPLOYMENT OF WEB VIRTUAL MACHINE (WEB\_VM)



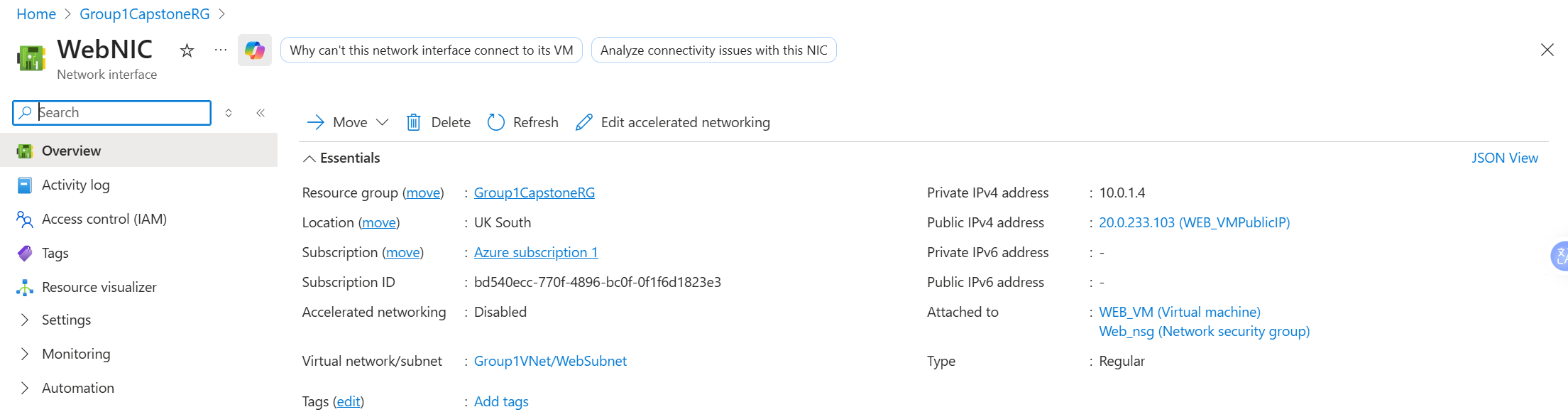
DEPLOYMENT OF APP VIRTUAL MACHINE (APP\_VM)



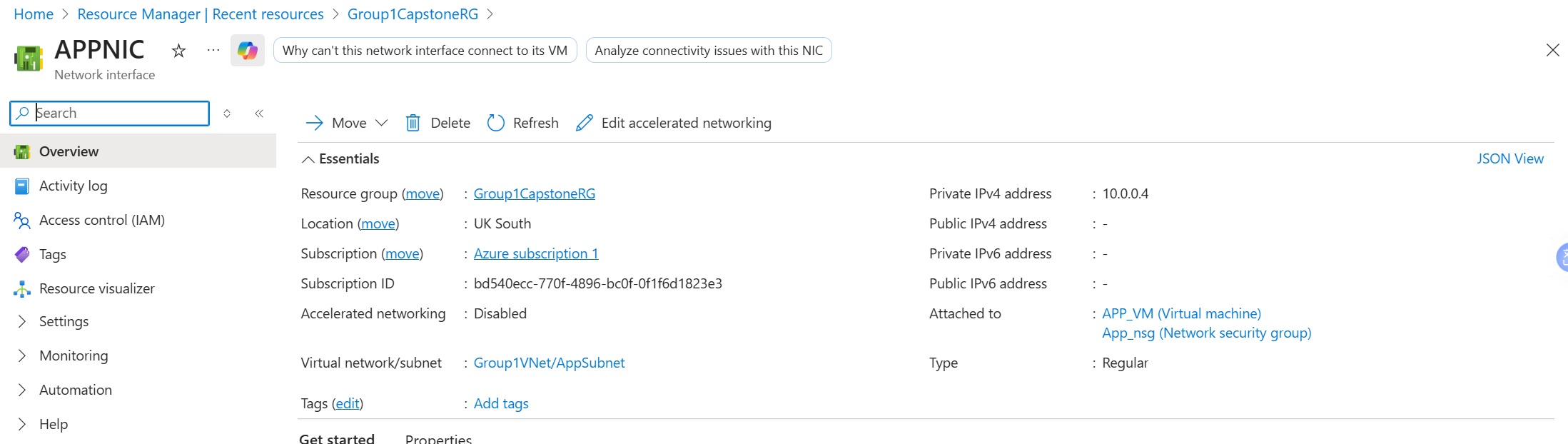
DEPLOYMENT OF DB VIRTUAL MACHINE (DB\_VM)



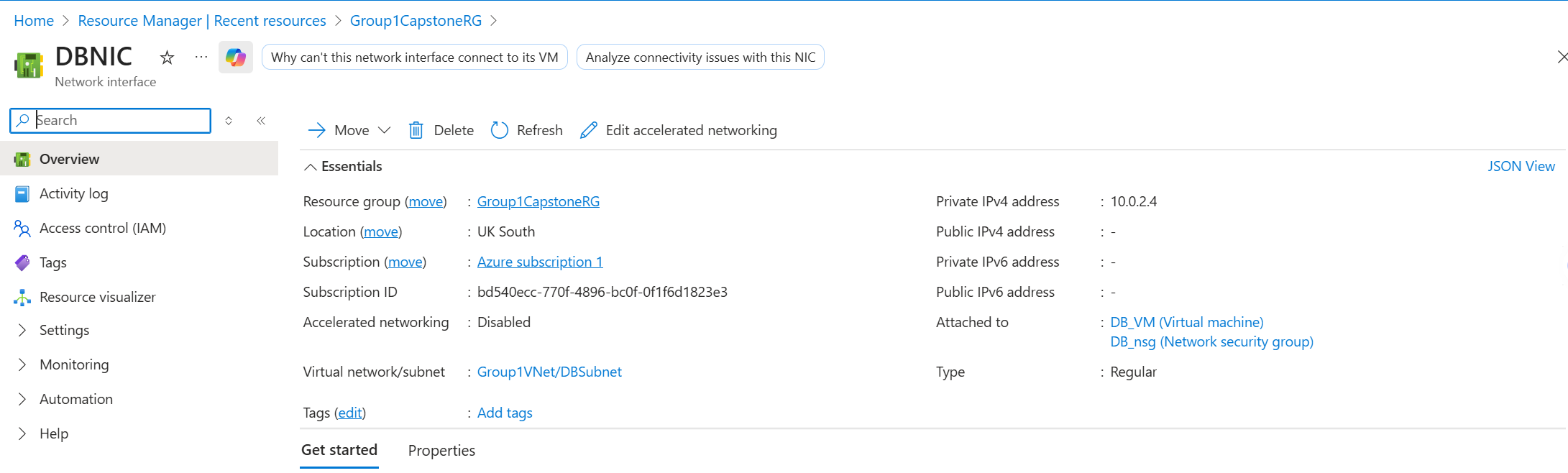
PROVISIONING OF WEB NETWORK INTERFACE CONNECTION (WEBNIC)



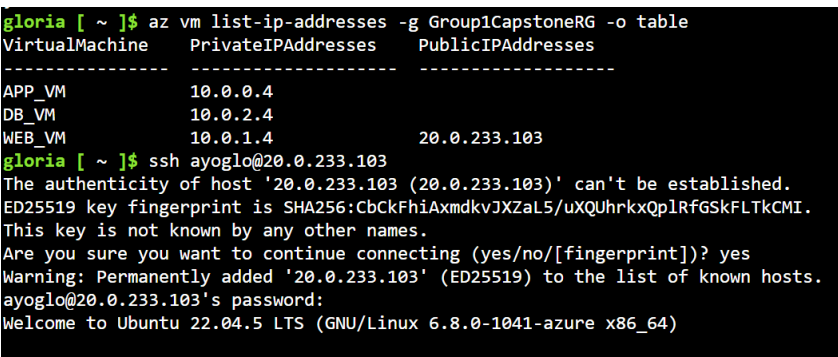
PROVISIONING OF APP NETWORK INTERFACE CONNECTION (APPNIC)



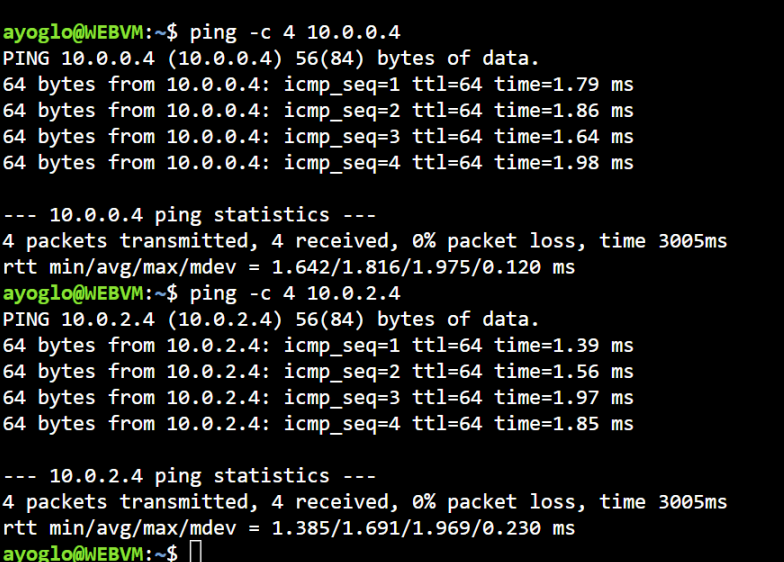
PROVISIONING OF DB NETWORK INTERFACE CONNECTION (DBNIC)



**IP ADDRESSES FOR THE 3 VIRTUAL MACHINES AND SSH INTO WEB\_VM**



**PING TEST TO SHOW CONNECTION OF WEB TO APP TO DB**



## Discussion

## The project demonstrated network segmentation and secure architecture design principles using Azure resources. The tiered structure allowed clear separation of concerns:

## Web tier: handled user-facing requests.

## App tier: processed business logic.

## DB tier: managed data securely, accessible only by the application tier.

## Using Network Security Groups, ensured traffic was restricted to only required communication paths. SSH testing validated that communication policies were properly enforced. Automating the setup with a Bash script reinforced Infrastructure as Code (IaC) principles, ensuring consistent deployment in multiple environments. Integration with GitHub supported version control, making collaboration and rollback easier.

## Conclusion

## The project achieved all stated objectives. A secure, scalable, and well-structured multi-tier virtual network was successfully deployed and validated using Azure CLI. This approach mirrors modern enterprise cloud designs where security and network isolation are key components of infrastructure deployment. Automation through Bash scripting and GitHub integration provided a reliable and efficient deployment workflow

## Recommendations

* Use Azure Bastion for secure SSH access without exposing Public IPs.
* 2.Deploy load balancers for Web and App tiers to improve fault tolerance.
* Implement Azure Monitor and Log Analytics for continuous performance monitoring and alerting.
* Use Azure Policy and Role-based access (RBAC) to enforce compliance and governance across all deployed resources.
* Consider integrating Application Gateway with Web Application Firewall (WAF) for advanced web traffic security.
* 7. Extend the architecture to include disaster recovery (DR) and backup solutions for business continuity.

**BASH SCRIPT FOR AUTOMATION OF FULL DEPLOYMENT**

#!/bin/bash

# ==========================================

# GROUP 1 CAPSTONE PROJECT

# ==========================================

# Variables

RG="Group1CapstoneRG"

LOC="uksouth"

VNET="Group1VNet"

WEB\_SUBNET="WebSubnet"

APP\_SUBNET="AppSubnet"

DB\_SUBNET="DBSubnet"

WEB\_NSG="Web\_nsg"

APP\_NSG="App\_nsg"

DB\_NSG="DB\_nsg"

WEB\_VM="WEB\_VM"

APP\_VM="APP\_VM"

DB\_VM="DB\_VM"

ADMIN\_USER="your\_username"

ADMIN\_PASS="Gloriaayool@69" #password should contain at least a symbol, a capital letter and not less than 12 characters

VM\_SIZE="Standard\_B1s"

# Create Resource Group

echo "Creating Resource Group in $LOC..."

az group create --name $RG --location $LOC

# Create Virtual Network and Subnets

echo "Creating VNet and Subnets..."

az network vnet create -g $RG -n $VNET --address-prefix 10.0.0.0/16 --subnet-name $WEB\_SUBNET --subnet-prefix 10.0.1.0/24

az network vnet subnet create -g $RG --vnet-name $VNET -n $APP\_SUBNET --address-prefix 10.0.2.0/24

az network vnet subnet create -g $RG --vnet-name $VNET -n $DB\_SUBNET --address-prefix 10.0.3.0/24

# Create Network Security Groups

echo "Creating NSGs..."

az network nsg create -g $RG -n $WEB\_NSG --location $LOC

az network nsg create -g $RG -n $APP\_NSG --location $LOC

az network nsg create -g $RG -n $DB\_NSG --location $LOC

# Configure NSG Rules

echo "Configuring NSG Rules..."

# --- Web Tier Rules ---

az network nsg rule create -g $RG --nsg-name $WEB\_NSG -n AllowWebHTTP --priority 100 --protocol Tcp --destination-port-ranges 80 --access Allow --direction Inbound

az network nsg rule create -g $RG --nsg-name $WEB\_NSG -n AllowWebSSH --priority 110 --protocol Tcp --destination-port-ranges 22 --access Allow --direction Inbound

# --- App Tier Rules ---

az network nsg rule create -g $RG --nsg-name $APP\_NSG -n AllowWebToApp --priority 100 --protocol Tcp --source-address-prefixes 10.0.1.0/24 --destination-port-ranges 80 --access Allow --direction Inbound

# Allow SSH from Web subnet (optional)

az network nsg rule create -g $RG --nsg-name $APP\_NSG -n AllowSSHFromWeb --priority 120 --protocol Tcp --source-address-prefixes 10.0.1.0/24 --destination-port-ranges 22 --access Allow --direction Inbound

# --- DB Tier Rules ---

az network nsg rule create -g $RG --nsg-name $DB\_NSG -n AllowAppToDB --priority 100 --protocol Tcp --source-address-prefixes 10.0.2.0/24 --destination-port-ranges 3306 --access Allow --direction Inbound

# Allow SSH from App subnet (optional)

az network nsg rule create -g $RG --nsg-name $DB\_NSG -n AllowSSHFromApp --priority 120 --protocol Tcp --source-address-prefixes 10.0.2.0/24 --destination-port-ranges 22 --access Allow --direction Inbound

# Create NICs and Attach NSGs

echo "Creating NICs..."

az network nic create -g $RG -n WebNIC --vnet-name $VNET --subnet $WEB\_SUBNET --network-security-group $WEB\_NSG

az network nic create -g $RG -n AppNIC --vnet-name $VNET --subnet $APP\_SUBNET --network-security-group $APP\_NSG

az network nic create -g $RG -n DbNIC --vnet-name $VNET --subnet $DB\_SUBNET --network-security-group $DB\_NSG

# Create Public IP for Web VM

echo "Creating Public IP for Web\_VM..."

az network public-ip create -g $RG -n Web\_VMPublicIP --sku Standard --allocation-method Static

# Update Web NIC with Public IP

echo "Attaching Public IP to WebNIC..."

az network nic ip-config update -g $RG --nic-name WebNIC --name ipconfig1 --public-ip-address WebVMPublicIP

# Deploy Linux VMs

echo "Deploying Linux VMs in $LOC..."

az vm create -g $RG -n $WEB\_VM --nics WebNIC --image Ubuntu2204 --size $VM\_SIZE --admin-username $ADMIN\_USER --admin-password $ADMIN\_PASS --public-ip-sku Standard --location $LOC

az vm create -g $RG -n $APP\_VM --nics AppNIC --image Ubuntu2204 --size $VM\_SIZE --admin-username $ADMIN\_USER --admin-password $ADMIN\_PASS --location $LOC

az vm create -g $RG -n $DB\_VM --nics DbNIC --image Ubuntu2204 --size $VM\_SIZE --admin-username $ADMIN\_USER --admin-password $ADMIN\_PASS --location $LOC

# Output Public and Private IPs

echo "Deployment Complete! Fetching IP Addresses..."

az vm list-ip-addresses -g $RG -o table

# Final Instructions

echo ""

echo "Deployment successful!"

echo "Use SSH to connect and verify connectivity:"

echo " ssh $ADMIN\_USER@<Web\_VM\_Public\_IP>"

echo "Then test private connections from inside WebVM:"

echo " ping 10.0.2.4 # App\_VM"

echo " ping 10.0.3.4 # DB\_VM"