Azure Project: WireGuard VPN & Private Web Server Lab (Cloud Shell)

Objective

The objective of this lab is to deploy a secure WireGuard VPN server on an Ubuntu VM in Microsoft Azure using Cloud Shell (CLI), and to configure a private web server accessible only through the VPN tunnel. This project demonstrates VPN setup, NSG configuration, private networking, and cloud-init automation.

Tools & Services Used

- - Azure Cloud Shell (Bash)
- - Azure Resource Group
- - Ubuntu Linux Virtual Machines
- - WireGuard VPN
- - Nginx Web Server
- - Network Security Groups (NSGs)
- - cloud-init Automation

Step-by-Step Implementation

Step 1: Create Resource Group

Command:
RG=rg-wg-demo
LOC=eastus
az group create -n $RG -l $LOC

Step 2: Create Virtual Network and Subnets

Command:
az network vnet create -g $RG -n vnet-wg --address-prefixes 10.10.0.0/16 --subnet-name snet-vpn --subnet-prefix 10.10.0.0/24
az network vnet subnet create -g $RG --vnet-name vnet-wg -n snet-app --address-prefixes 10.10.1.0/24

```
mose [ ~ ]$ az network vnet subnet create -g $RG --vnet-name $VNET -n $SNET_APP \
  --address-prefixes 10.10.1.0/24
{
  "addressPrefix": "10.10.1.0/24",
  "delegations": [],
  "etag": "W/\"d2301dca-518d-47d2-857d-32a2bcf54ae8\"",
  "id": "/subscriptions/a63c3193-6450-4099-95e2-8c41ba85ae57/resourceGroups/rg-vpn-lab/providers/Microsoft.Network/virtualNetworks/vnet-vpn/subnets/s
net-app",
  "name": "snet-app",
  "privateEndpointNetworkPolicies": "Disabled",
  "privateLinkServiceNetworkPolicies": "Enabled",
  "provisioningState": "Succeeded",
  "resourceGroup": "rg-vpn-lab",
  "type": "Microsoft.Network/virtualNetworks/subnets"
}
```

Step 3: Configure Network Security Groups (NSGs)

Command:

az network nsg create -g $RG -n nsg-wg

az network nsg rule create -g $RG --nsg-name nsg-wg -n allow-ssh --priority 1000 --access Allow --protocol Tcp --direction Inbound --destination-port-ranges 22

az network nsg rule create -g $RG --nsg-name nsg-wg -n allow-wireguard --priority 1001 --access Allow --protocol Udp --direction Inbound --destination-port-ranges 51820

az network nsg create -g $RG -n nsg-app

az network nsg rule create -g $RG --nsg-name nsg-app -n allow-http-from-vpn --priority 1000 --access Allow --protocol Tcp --direction Inbound --source-address-prefixes 10.10.0.0/24 --destination-port-ranges 80

az network nsg rule create -g $RG --nsg-name nsg-app -n allow-ssh-from-vpn --priority 1001 --access Allow --protocol Tcp --direction Inbound --source-address-prefixes 10.10.0.0/24 --destination-port-ranges 22

```
mose [ ~ ]$ az network nsg create -g $RG -n nsg-wg
az network nsg rule create -g $RG --nsg-name nsg-wg -n allow-ssh \
  --priority 1000 --access Allow --protocol Tcp --direction Inbound --destination-port-ranges 22
az network nsg rule create -g $RG --nsg-name nsg-wg -n allow-wireguard \
  --priority 1001 --access Allow --protocol Udp --direction Inbound --destination-port-ranges $WG_PORT

az network nsg create -g $RG -n nsg-app
```

```
mose [ ~ ]$ az network nsg rule create -g $RG --nsg-name nsg-app -n allow-http-from-vpn \
  --priority 1000 --access Allow --protocol Tcp --direction Inbound \
  --source-address-prefixes 10.10.0.0/24 --destination-port-ranges 80
{
  "access": "Allow",
  "destinationAddressPrefix": "*",
  "destinationAddressPrefixes": [],
  "destinationPortRange": "80",
  "destinationPortRanges": [],
  "direction": "Inbound",
  "etag": "W/\"c60b3182-3dc5-40fe-bdab-84ac5e50da20\"",
  "id": "/subscriptions/a63c3193-6450-4099-95e2-8c41ba85ae57/resourceGroups/rg-vpn-lab/providers/Microsoft.Network/networkSecurityGroups/nsg-app/secu
rityRules/allow-http-from-vpn",
  "name": "allow-http-from-vpn",
  "priority": 1000,
  "protocol": "Tcp",
  "provisioningState": "Succeeded",
  "resourceGroup": "rg-vpn-lab",
  "sourceAddressPrefix": "10.10.0.0/24",
  "sourceAddressPrefixes": [],
  "sourcePortRange": "*",
  "sourcePortRanges": [],
  "type": "Microsoft.Network/networkSecurityGroups/securityRules"
}
```

```
mose [ ~ ]$ az network nsg rule create -g $RG --nsg-name nsg-app -n allow-ssh-from-vpn \
  --priority 1001 --access Allow --protocol Tcp --direction Inbound \
  --source-address-prefixes 10.10.0.0/24 --destination-port-ranges 22
{
  "access": "Allow",
  "destinationAddressPrefix": "*",
  "destinationAddressPrefixes": [],
  "destinationPortRange": "22",
  "destinationPortRanges": [],
  "direction": "Inbound",
  "etag": "W/\"2965ef08-c5a9-45e7-a3ed-5d9f4bb1e785\"",
  "id": "/subscriptions/a63c3193-6450-4099-95e2-8c41ba85ae57/resourceGroups/rg-vpn-lab/providers/Microsoft.Network/networkSecurityGroups/nsg-app/secu
rityRules/allow-ssh-from-vpn",
  "name": "allow-ssh-from-vpn",
  "priority": 1001,
  "protocol": "Tcp",
  "provisioningState": "Succeeded",
  "resourceGroup": "rg-vpn-lab",
  "sourceAddressPrefix": "10.10.0.0/24",
  "sourceAddressPrefixes": [],
  "sourcePortRange": "*",
  "sourcePortRanges": [],
  "type": "Microsoft.Network/networkSecurityGroups/securityRules"
}
```

Step 4: Associate NSGs with Subnets

Command:

az network vnet subnet update -g $RG --vnet-name vnet-wg -n snet-vpn
--network-security-group nsg-wg

az network vnet subnet update -g $RG --vnet-name vnet-wg -n snet-app
--network-security-group nsg-app

```
mose [ ~ ]$ az network vnet subnet update -g $RG --vnet-name $VNET -n $SNET_VPN --network-security-group nsg-wg
az network vnet subnet update -g $RG --vnet-name $VNET -n $SNET_APP --network-security-group nsg-app
```

Step 5: Create Public IP and NIC

Command:

az network public-ip create -g $RG -n pip-wg --sku Standard --allocation-method Static

az network nic create -g $RG -n nic-wg --vnet-name vnet-wg --subnet snet-vpn
--network-security-group nsg-wg --public-ip-address pip-wg

```
mose [ ~ ]$ az network public-ip create -g $RG -n pip-wg --sku Standard --allocation-method Static
[Coming breaking change] In the coming release, the default behavior will be changed as follows when sku is Standard and zone is not provided: For zo
nal regions, you will get a zone-redundant IP indicated by zones:["1","2","3"]; For non-zonal regions, you will get a non zone-redundant IP indicated
 by zones:null.
{
  "publicIp": {
    "ddosSettings": {
      "protectionMode": "VirtualNetworkInherited"
    },
    "etag": "W/\"60aef697-1b09-4ec8-8fc0-c7a2b545939c\"",
    "id": "/subscriptions/a63c3193-6450-4099-95e2-8c41ba85ae57/resourceGroups/rg-vpn-lab/providers/Microsoft.Network/publicIPAddresses/pip-wg",
    "idleTimeoutInMinutes": 4,
    "ipAddress": "74.235.225.65",
    "ipTags": [],
    "location": "eastus",
    "name": "pip-wg",
    "provisioningState": "Succeeded",
    "publicIPAddressVersion": "IPv4",
    "publicIPAllocationMethod": "Static",
    "resourceGroup": "rg-vpn-lab",
    "resourceGuid": "062ea444-308d-4522-bf51-9b80fbf22764",
    "sku": {
      "name": "Standard",
      "tier": "Regional"
    },
    "type": "Microsoft.Network/publicIPAddresses"
  }
}
mose [ ~ ]$ az network nic create -g $RG -n nic-wg --vnet-name $VNET --subnet $SNET_VPN \
  --network-security-group nsg-wg --public-ip-address pip-wg
```

Step 6: Deploy WireGuard VM (VPN Server)

Command:

az vm create -g $RG -n wg-vm --image Ubuntu2404 --size Standard_B1s --admin-username

azureuser --generate-ssh-keys --nics nic-wg --custom-data wg-cloudinit.yaml --public-ip-sku Standard

```
mose [ ~ ]$ RG=rg-wg-demo
WGVM=wg-vm
WEBVM=web-vm2  # <-- new name to avoid the customData error

# Reuse wg-vm's subnet
NICID=$(az vm show -g $RG -n $WGVM --query "networkProfile.networkInterfaces[0].id" -o tsv)
SUBNETID=$(az network nic show --ids "$NICID" --query "ipConfigurations[0].subnet.id" -o tsv)

# Create web-vm2 private-only with your cloud-init
az vm create \
  -g $RG -n $WEBVM \
  --image Ubuntu2404 --size Standard_B1s \
  --admin-username azureuser --generate-ssh-keys \
  --subnet "$SUBNETID" \
  --public-ip-address "" \
  --custom-data ~/web-cloudinit.yaml

# Private IP to test
WEBPRIVIP=$(az vm list-ip-addresses -g "$RG" -n "$WEBVM" -o tsv \
  --query "[0].virtualMachine.network.privateIpAddresses[0]")
echo "web-vm2 private IP: $WEBPRIVIP"
The default value of '--size' will be changed to 'Standard_D2s_v5' from 'Standard_DS1_v2' in a future release.
{
  "fqdns": "",
  "id": "/subscriptions/a63c3193-6450-4099-95e2-8c41ba85ae57/resourceGroups/rg-wg-demo/providers/Microsoft.Compute/virtualMachines/web-vm2",
  "location": "eastus",
  "macAddress": "00-22-48-1D-3E-D9",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.6",
  "publicIpAddress": "",
  "resourceGroup": "rg-wg-demo"
}
web-vm2 private IP: 10.0.0.6
```

Step 7: Verify WireGuard Service

Command:
sudo systemctl status wg-quick@wg0 --no-pager
sudo wg show
sudo journalctl -u wg-quick@wg0 -n 200 --no-pager

```
azureuser@wg-vm:~$ sudo systemctl status wg-quick@wg0 --no-pager
sudo wg show
sudo journalctl -u wg-quick@wg0 -n 200 --no-pager
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
     Loaded: loaded (/usr/lib/systemd/system/wg-quick@.service; enabled; preset: enabled)
     Active: active (exited) since Sat 2025-10-04 08:28:01 UTC; 8min ago
       Docs: man:wg-quick(8)
             man:wg(8)
             https://www.wireguard.com/
             https://www.wireguard.com/quickstart/
             https://git.zx2c4.com/wireguard-tools/about/src/man/wg-quick.8
             https://git.zx2c4.com/wireguard-tools/about/src/man/wg.8
    Process: 2506 ExecStart=/usr/bin/wg-quick up wg0 (code=exited, status=0/SUCCESS)
   Main PID: 2506 (code=exited, status=0/SUCCESS)
        CPU: 28ms
```

Step 8: Create App NIC and Web Server Cloud-init

Command:
cat > ~/web-cloudinit.yaml <<'YAML'
#cloud-config
package_update: true
packages: [nginx]
write_files:
 - path: /var/www/html/index.html
   permissions: "0644"
   content: |
     <h1>Private Web (VPN-only)</h1>
     If you can see this page, your WireGuard tunnel to Azure is working.

YAML
az network nic create -g $RG --network-security-group nsg-app -n nic-web --vnet-name vnet-wg --subnet snet-app

```
azureuser@wg-vm:~$ cat > ~/web-cloudinit.yaml <<'YAML'
#cloud-config
package_update: true
packages: [nginx]
write_files:
  - path: /var/www/html/index.html
    permissions: "0644"
    content: |
      <h1>Private Web (VPN-only)</h1>
      <p>If you can see this page, your WireGuard tunnel to Azure is working.</p>
runcmd:
  - [ systemctl, enable, --now, nginx ]
YAML
```

Step 9: Deploy Private Web VM

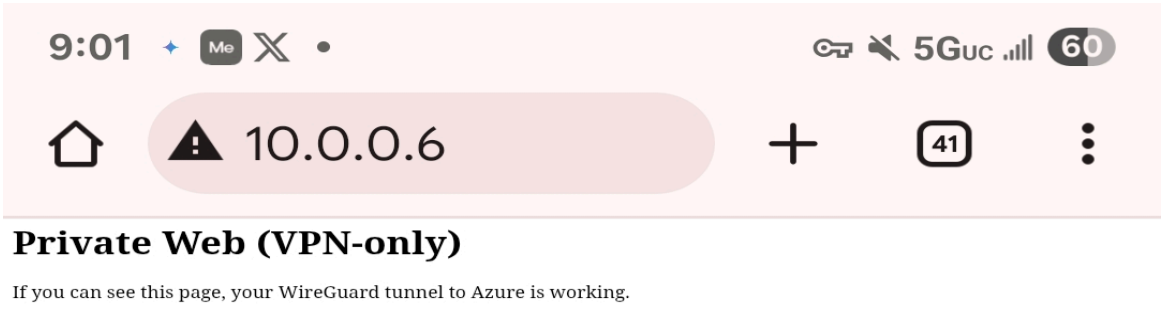
Command:
az vm create -g $RG -n web-vm2 --image Ubuntu2404 --admin-username azureuser --generate-ssh-keys --subnet snet-app --public-ip-address '' --custom-data ~/web-cloudinit.yaml

```
mose [ ~ ]$ RG=rg-wg-demo
LOC=eastus
VM=web-vm
IMG=Ubuntu2404
SIZE=Standard_B1s
ADMIN=azureuser

az vm create \
  -g $RG -n $VM \
  --image $IMG --size $SIZE \
  --admin-username $ADMIN --generate-ssh-keys \
  --custom-data ~/web-cloudinit.yaml \
  --public-ip-sku Standard
The default value of '--size' will be changed to 'Standard_D2s_v5' from 'Standard_DS1_v2' in a future release.
{
  "fqdns": "",
  "id": "/subscriptions/a63c3193-6450-4099-95e2-8c41ba85ae57/resourceGroups/rg-wg-demo/providers/Microsoft.Compute/virtualMachines/web-vm",
  "location": "eastus",
  "macAddress": "7C-1E-52-67-F4-E2",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.5",
  "publicIpAddress": "74.235.227.233",
  "resourceGroup": "rg-wg-demo"
}
```

Step 10: Test Connectivity via VPN

Command:
curl http://10.0.0.6
Expected Result:
Private Web (VPN-only)
If you can see this page, your WireGuard tunnel to Azure is working.



Results

- - Deployed a functional WireGuard VPN server on Azure using Cloud Shell (CLI).
- - Configured NSG rules and subnet associations for VPN and application layers.
- - Created a private web server accessible only through the VPN tunnel.
- - Verified end-to-end secure communication using curl and browser testing.