mushroom NETWORKS

Networks on autopilot®

TruffleV HA Configuration Template for Azure

Executive Summary

Brief overview of the solution:

This design document presents a template design for VPN over VLL tunnel implementation for connecting branch offices to Azure Cloud infrastructure. VPN over VLL presents a more robust and agile HA (high availability) compared to traditional designs such as BGP over VPN.

Target audience:

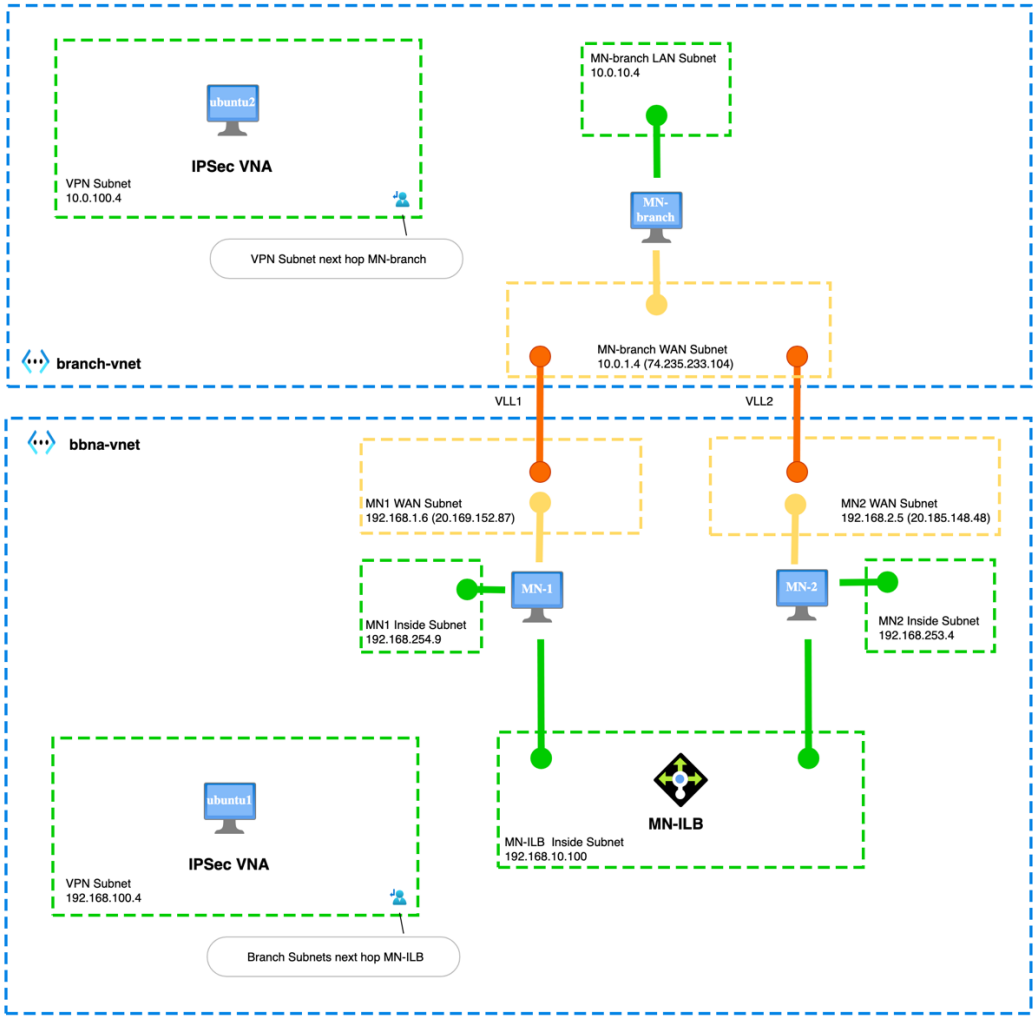
Network administrators implementing secure branch-to-Azure connectivity.

Architecture Overview

Network Topology

The architecture diagram below represents a sample design and could be used as a template for designing a network for specific project requirements. In our design, we will use an IPSec VPN between a HA (High Availability) setup of Truffle virtual machines (VM) in the azure cloud (bbna-vnet) and in a separate Azure vnet to simulate a single branch office network (branch-vnet).

Network Diagram after MN-ILB installation:



Components Descriptions

Virtual Networks:

- branch-vnet: Represents branch office network (simulated in Azure for demonstration purposes).
- bbna-vnet: Represents your main Azure virtual network environment.

Mushroom Networks Appliances:

- MN-VM1: Mushroom Networks Truffle VNA (Virtual Network Appliance) as the primary VLL tunnel endpoint in Azure (bbna-vnet).
- MN-VM2: Secondary VLL tunnel endpoint in Azure for HA.
- MN-VM-branch: VLL tunnel endpoint at the branch office (branch-vnet).

IPsec Endpoints

- ubuntu1: Ubuntu VM running StrongSwan in Azure.
- ubuntu2: Ubuntu VM running StrongSwan in the branch network.

Connection Elements

- VLL Tunnels: Secure tunnel connections between Azure and branch Mushroom Networks VNAs.
- IPsec Tunnel: Encrypted VPN connection running over the VLL tunnels between StrongSwan endpoints.
- Internal Load Balancer: Distributes traffic between MN-VM1 and MN-VM2 for high availability.
- Health Probe: Monitors health condition of a VNA.

Network Addressing Scheme

What follows is a sample subnet addressing design for demonstration purposes and the specifics should be modified to support the project requirements.

Branch Environment: branch-vnet: 10.0.0.0/16

- MN-VM-branch WAN subnet: 10.0.1.0/24
  - MN-VM-branch WAN1 IP: private:10.0.1.4, public:74.235.233.104
- MN-VM-branch LAN subnet: 10.0.10.0/24
  - MN-VM-branch LAN IP: private: 10.0.10.4
- IPSec VPN endpoint (ubuntu2) subnet: 10.0.100.0/24
  - IPSec VPN endpoint (ubuntu2) LAN IP: 10.0.100.4

Azure Environment: bbna-vnet: 192.168.0.0/16

- MN-VM1 WAN Subnet: 192.168.1.0/24
  - MN-VM1 WAN interface: 192.168.1.6
- MN-VM1 LAN subnet: 192.168.254.0/24
  - MN-VM1 LAN interface 192.168.254.9
- MN-VM2 WAN Subnet : 192.168.2.0/24
  - MN-VM2 WAN interface: 192.168.2.5
- MN-VM2 LAN subnet: 192.168.253.0/24
  - MN-VM2 LAN interface 192.168.253.4
- IPSec VPN endpoint (ubuntu1) Subnet: 192.168.100.0/24
  - IPSec VPN endpoint (ubuntu1): 192.168.100.4
- ILB Subnet: 192.168.10.0/24
  - Load Balancer frontend: 192.168.10.100

Routing

- Traffic from ubuntu1 to ubuntu2 routes through the ILB (192.168.10.100)
- ILB directs traffic to either MN-VM1 or MN-VM2 based on LB rule and on health
- Each MN VM is configured to forward traffic through its VLL tunnel to/from MN-VM-branch
- Traffic from ubuntu2 to ubuntu1 routes through the MN-branch

Networks on autopilot®

Configuration Template – Phase I

Deploy MN-VM-branch, MN-VM1 and MN-VM2 virtual machines in Azure

1. Download the vhd file from the link provided by Mushroom Networks.

2. Create a storage blob in your Azure account and upload the vhd file.

3. Create an "image" in Azure within your "Resource group" that has the vhd blob and your region. Choose Linux as "OS type" and Gen 1 as "VM generation". Choose the storage blob that has your vhd file, standard HDD, and read/write "host caching" options.

4. Create a VM using the newly created Azure image. Standard_D2s_v4 or better is recommended as VM size. Choose "SSH public key" for authentication type, enter a "Key pair name", choose None for "Public inbound ports" and choose Other for "License type".

5. Note that prior to being activated by Mushroom Networks, the VM start may give a time out message in Azure which is expected behavior. The VM image will have single eth0 interface for the LAN port. Create and attach a new interface to the new VM with a public IP which will be used for WAN1. Note down both your public and private IP address for this interface.

6. Within the security group of your newly created WAN interface, add a port rule to allow all inbound traffic for all ports and protocols.

7. Using the "serial console" connect option, connect to your VM.

8. Once Mush CLI is available, you can configure your WAN1 port with the correct private IP address that your WAN1 interface has. To accomplish this, run the following command in the Mush CLI: "change wan/1 StaticInterface: ip "192.168.2.5" gateway "192.168.2.1" netmask "255.255.255.0" dns1 "8.8.8.8" failover "false" alias "false" mtu "1500". Remember to replace the 192.168.2.5 and 192.168.2.1 with your correct private IP addresses. You can check your changes via "list wan" command.

9. Once the public IP is assigned to your WAN1, you will be able to remotely access your unit from https://<your-public-ip-address>:8081. Once you confirmed remote access, contact Mushroom Networks support for enabling your Truffle V license.

Install StrongSwan in bbna-vnet

1. Create the subnet for StrongSwan in bbna-vnet:

   ```
   az network vnet subnet create –resource-group bbna –vnet-name bbna-vnet –name bbna-strongswan-subnet –address-prefix 192.168.100.0/24
   ```

2. Create the network security group:

   ```
   az network nsg create --resource-group bbna --name bbna-strongswan-nsg
   ```

3. Modify NSG to allow IPSec traffic:

   ```
   az network nsg rule create --resource-group bbna --nsg-name bbna-strongswan-nsg --name AllowESP --priority 1030 --protocol Esp --destination-port-range "*" --access Allow --source-address-prefix 192.168.1.0/24 --destination-address-prefix 192.168.100.0/24
   ```

4. Create the VM that will host StrongSwan:

   ```
   az vm create --resource-group bbna --name ubuntu1-strongswan --image Ubuntu2204 --admin-username azureuser --generate-ssh-keys --vnet-name bbna-vnet --subnet bbna-strongswan-subnet --nsg bbna-strongswan-nsg --private-ip-address 192.168.100.4 --size Standard_B2s
   ```

5. Now that we have the VM ready, let's SSH into it to install StrongSwan:
   Ssh azureuser@172.173.224.39

6. Once you're connected to the VM, run these commands to install StrongSwan:
   sudo apt update && sudo apt upgrade -y
   sudo apt install -y strongswan strongswan-pki libcharon-extra-plugins

We will create the branch-vnet components next.

Create and configure the branch-vnet and required subnets

1. First, let's create a new virtual network for the branch office:
   az network vnet create --resource-group bbna --name branch-vnet --address-prefix 10.0.0.0/16 --location eastus

2. Create subnets within branch-vnet .We'll need two subnets - one for the Mushroom Networks VM (WAN subnet) and one for the StrongSwan VM:
   az network vnet subnet create --resource-group bbna --vnet-name branch-vnet --name branch-wan-subnet --address-prefix 10.0.1.0/24
   az network vnet subnet create --resource-group bbna --vnet-name branch-vnet --name branch-strongswan-subnet --address-prefix 10.0.100.0/24

3. Create NSG for the StrongSwan VM in branch-vnet
   az network nsg create --resource-group bbna --name branch-strongswan-nsg

4. Add security rules for IPSec to the NSG
   az network nsg rule create --resource-group bbna --nsg-name branch-strongswan-nsg --name AllowSSH --priority 1000 --protocol Tcp --destination-port-range 22 --access Allow

   az network nsg rule create --resource-group bbna --nsg-name branch-strongswan-nsg --name AllowIKE --priority 1010 --protocol Udp --destination-port-range 500 --access Allow

   az network nsg rule create --resource-group bbna --nsg-name branch-strongswan-nsg --name AllowNAT-T --priority 1020 --protocol Udp --destination-port-range 4500 --access Allow

   az network nsg rule create --resource-group bbna --nsg-name branch-strongswan-nsg --name AllowESP --priority 1030 --protocol Esp --destination-port-range "*" --access Allow --source-address-prefix 192.168.100.0/24 --destination-address-prefix 10.0.100.0/24

Create the Ubuntu VM with StrongSwan for the branch-vnet

1. Create Ubuntu VM:
   az vm create --resource-group bbna --name ubuntu2-strongswan --image Ubuntu2204 --admin-username azureuser --generate-ssh-keys --vnet-name branch-vnet --subnet branch-strongswan-subnet --nsg branch-strongswan-nsg --private-ip-address 10.0.100.4 --size Standard_B2s

2. Now let's SSH into your ubuntu2-strongswan VM and install StrongSwan. Use the public IP address from your output, e.g.:
   ssh azureuser@172.203.145.139

3. Once you're connected to the VM, install StrongSwan with these commands:
   ```
   sudo apt update && sudo apt upgrade -y
   sudo apt install -y strongswan strongswan-pki libcharon-extra-plugins
   ```

Configure StrongSwan on ubuntu1 (bbna-vnet side)

1. SSH into ubuntu1-strongswan and edit the IPSec configuration:
   ```
   ssh azureuser@[ubuntu1-public-ip]
   ```

2. Edit ipsec.conf file.
   ```
   sudo nano /etc/ipsec.conf
   ```

   Replace the content with:
   ```
   config setup
      charondebug="all"
      uniqueids=yes

   conn bbna-to-branch
      type=tunnel
      auto=start
      keyexchange=ikev2
      ike=aes256-sha256-modp2048!
      esp=aes256-sha256!
      left=192.168.100.4
      leftsubnet=192.168.100.0/24
      right=10.0.100.4
      rightsubnet=10.0.100.0/24
      authby=secret
      ikelifetime=1h
      lifetime=8h
      dpddelay=30
      dpdtimeout=120
      dpdaction=restart
   ```

3. Edit the secrets file:
   ```
   sudo nano /etc/ipsec.secrets
   ```

   Add this line:
   ```
   192.168.100.4 10.0.100.4 : PSK "StrongSecretPassphrase123!"
   ```

Configure StrongSwan on ubuntu2 (branch-vnet side)

1. SSH into ubuntu2-strongswan and edit the IPSec configuration:
   ```
   ssh azureuser@172.203.145.139
   ```

2. Edit ipsec.conf file:
   ```
   sudo nano /etc/ipsec.conf
   ```

   Replace the content with:
   ```
   config setup
      charondebug="all"
      uniqueids=yes

   conn branch-to-bbna
      type=tunnel
      auto=start
      keyexchange=ikev2
      ike=aes256-sha256-modp2048!
   ```

```
esp=aes256-sha256!
left=10.0.100.4
leftsubnet=10.0.100.0/24
right=192.168.100.4
rightsubnet=192.168.100.0/24
authby=secret
ikelifetime=1h
lifetime=8h
dpddelay=30
dpdtimeout=120
dpdaction=restart
```

3. Edit the secrets file:

   ```
   sudo nano /etc/ipsec.secrets
   ```

   Add this line:

   ```
   10.0.100.4 192.168.100.4 : PSK "StrongSecretPassphrase123!"
   ```

Configure UDR (User Defined Routes) for StrongSwan in bbna-vnet

Please note that when we later introduce the ILB, we will update this UDR.

1. Create route table for bbna StrongSwan subnet

   ```
   az network route-table create --resource-group bbna --name bbna-strongswan-routes
   ```

2. Add route to direct traffic to branch subnet via MN-VM1

   ```
   az network route-table route create --resource-group bbna --route-table-name bbna-strongswan-routes --name to-branch-via-mn --address-prefix 10.0.100.0/24 --next-hop-type VirtualAppliance --next-hop-ip-address 192.168.1.6
   ```

3. Associate route table with bbna StrongSwan subnet

   ```
   az network vnet subnet update --resource-group bbna --vnet-name bbna-vnet --name bbna-strongswan-subnet --route-table bbna-strongswan-routes
   ```

Configure UDR for StrongSwan in branch-vnet

1. Create route table for branch StrongSwan subnet

   ```
   az network route-table create --resource-group bbna --name branch-strongswan-routes
   ```

2. Add route to direct traffic to bbna subnet via MN-VM-branch

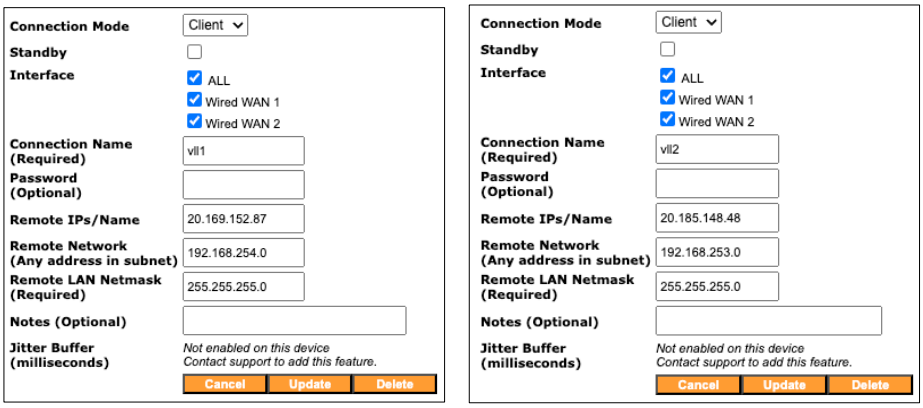
   ```
   az network route-table route create --resource-group bbna --route-table-name branch-strongswan-routes --name to-bbna-via-mn --address-prefix 192.168.100.0/24 --next-hop-type VirtualAppliance --next-hop-ip-address 10.0.10.4
   ```

3. Associate route table with branch StrongSwan subnet

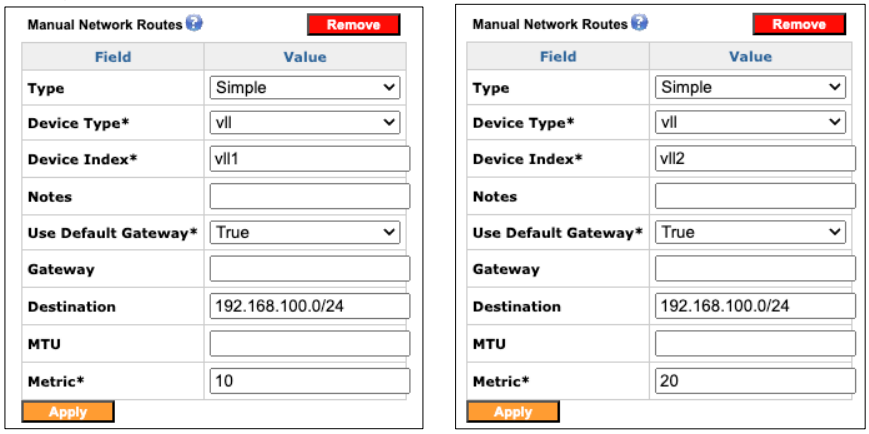
   ```
   az network vnet subnet update --resource-group bbna --vnet-name branch-vnet --name branch-strongswan-subnet --route-table branch-strongswan-routes
   ```

Configure MN-branch

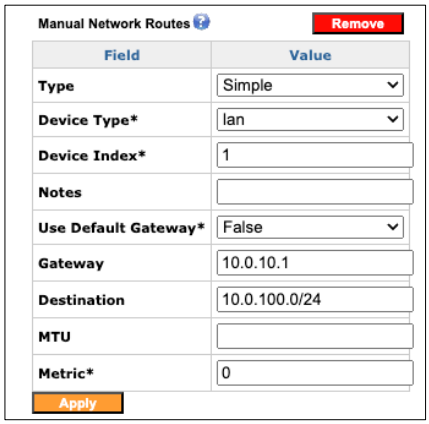
1. Configure VLL tunnel clients for VLL1 (and VLL2 for later) from the “Remote LAN(VLL)” section of the “Home” tab



2. Configure VLL1 and VLL2 routes under the “Manual Network Routes“ section of the “ADVANCED” tab

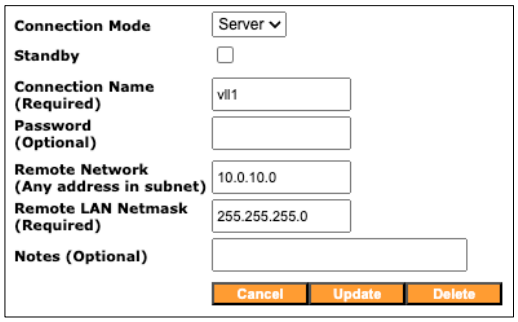


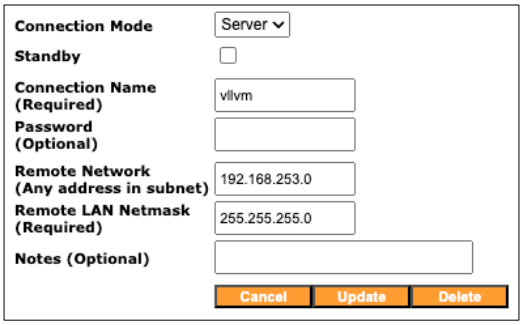
3. Configure route for IPSec destination traffic



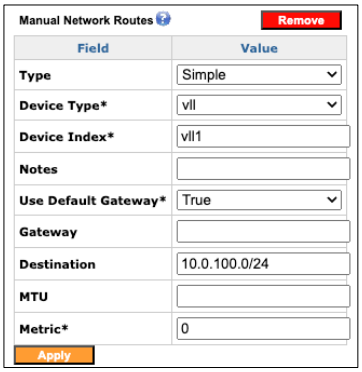
Configure MN-VM1

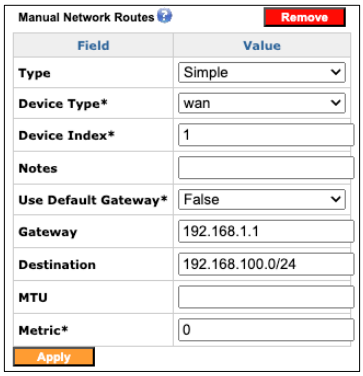
1. Add “VLL1 server” and “VLLVM server” configurations under the “Remote LAN” section of the “HOME” tab

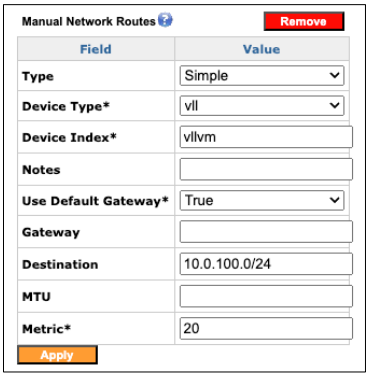




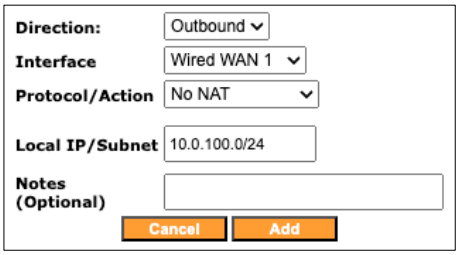
2. Add routes for destinations 10.0.100.0/24 (ubuntu2) and 192.168.100.0/24 (ubuntu1) under the "Manual Network Routes" section of the "ADVANCED" tab







3. Added a no-nat rule on MN-VM1 for traffic with source IP 10.0.100.0/24 (ubuntu2) under the "Outbound" section of the "FIREWALL" tab



Test IPSec prior to adding MN-VM2 and ILB

1. Restart IPSec tunnels on both ubuntu1 and ubuntu2 after ssh'ing into the units

   sudo ipsec restart

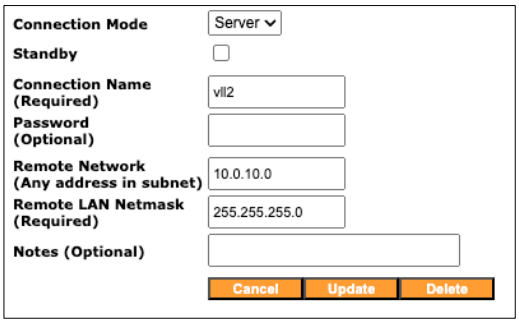
2. Test connectivity via ping test between ubuntu1 and ubuntu2. As an additional step, you can use "sudo tcpdump -n esp" on the receive side to make sure ping packets are going via the IPSec tunnel.

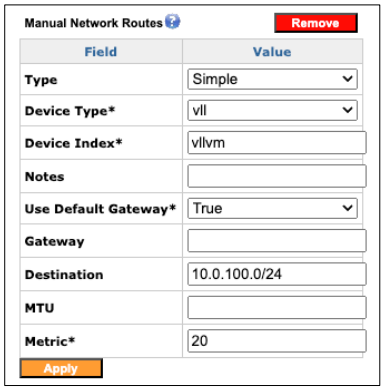
This completes the phase1 of the configuration. In phase2 below, we will configure MN-VM2 and the ILB.

Configuration Template – Phase II

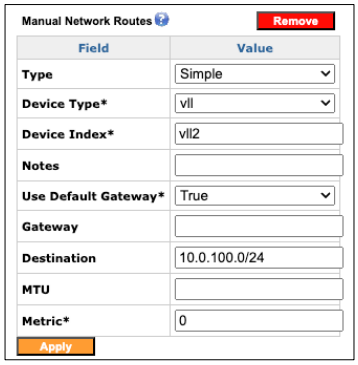
Configure MN-VM2 to be used for High Availability (HA)

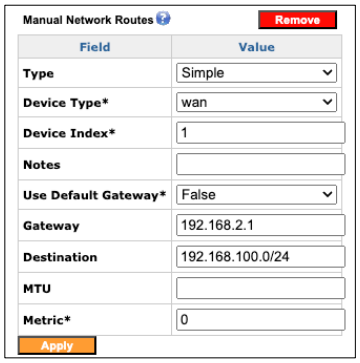
1. Add “VLL2 server” and “VLLVM client” configurations under the “Remote LAN” section of the “HOME” tab

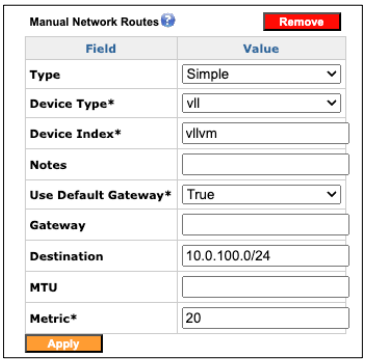




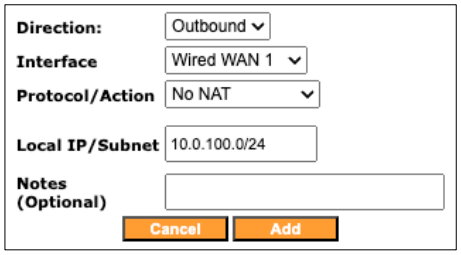
2. Add routes for destinations 10.0.100.0/24 (ubuntu2) and 192.168.100.0/24 (ubuntu1) under the “Manual Network Routes” section of the “ADVANCED” tab







3. Added a no-nat rule on MN-VM1 for traffic with source IP 10.0.100.0/24 (ubuntu2) under the “Outbound” section of the “FIREWALL” tab



Configure the ILB with “standard” SKU

1. Create dedicated subnet (example: 192.168.10.0/24)

```
az network vnet subnet create \
    --resource-group bbna \
    --vnet-name bbna-vnet \
    --name bbna-ilb-subnet \
    --address-prefix 192.168.10.0/24
```

2. Create the ILB in this subnet with static IP address:

```
az network lb create \
```

```
    --resource-group bbna \
    --name bbna-mn-ilb \
    --sku Standard \
    --vnet-name bbna-vnet \
    --subnet bbna-ilb-subnet \
    --frontend-ip-name bbna-mn-ilb-frontend \
    --backend-pool-name bbna-mn-vm-pool \
    --private-ip-address 192.168.10.100
```

3. Configure Health Probe for TCP protocol on port 8081 with 5 second intervals

```
az network lb probe create \
        --resource-group bbna \
        --lb-name bbna-mn-ilb \
        --name mn-health-probe \
        --protocol Tcp \
        --port 8081 \
        --interval 5
```

4. Configure the ILB with source IP affinity with HA ports

```
az network lb rule create \
    --resource-group bbna \
    --lb-name bbna-mn-ilb \
    --name IPsecHAPortRule \
    --protocol All \
    --frontend-port 0 \
    --backend-port 0 \
    --frontend-ip-name bbna-mn-ilb-frontend \
    --backend-pool-name bbna-mn-vm-pool \
    --probe-name mn-health-probe \
    --load-distribution SourceIP
```

5. Add MN-VM1 to the backend pool

```
az network nic ip-config address-pool add \
    --resource-group bbna \
    --nic-name <your_WAN_nic_name_of_MN_VM1> \
    --ip-config-name ipconfig1 \
    --lb-name bbna-mn-ilb \
    --address-pool bbna-mn-vm-pool
```

6. Update the UDR for ubuntu1 route table to use the ILB

```
az network route-table route update \
    --resource-group bbna \
    --route-table-name bbna-strongswan-routes \
    --name to-branch-via-mn \
    --address-prefix 10.0.100.0/24 \
    --next-hop-type VirtualAppliance \
    --next-hop-ip-address 192.168.10.100
```

Configure UDR for ubuntu1 and verify IPSec connectivity

1. Update the route table to use the ILB's IP address. We need to modify the User-Defined Route (UDR) for ubuntu1's subnet to point to the Internal Load Balancer instead of directly to MN-VM1:

```
az network route-table route update \
  --resource-group bbna \
  --route-table-name bbna-strongswan-routes \
  --name to-branch-via-mn \
```

```
--address-prefix 10.0.100.0/24 \
--next-hop-type VirtualAppliance \
--next-hop-ip-address 192.168.10.100
```

2. Restart IPSec tunnels on both ubuntu1 and ubuntu2 after ssh'ing into the units
   ```
   sudo ipsec restart
   ```

3. Test connectivity via ping test between ubuntu1 and ubuntu2. As an additional step, you can use "sudo tcpdump -n esp" on the receive side to make sure ping packets are going via the IPSec tunnel.

Add MN-VM2 to the network

1. Add MN-VM2 to the ILB backbend address pool
   ```
   az network nic ip-config address-pool add \
    --resource-group bbna \
    --nic-name <your_WAN_nic_name_of_MN_VM2> \
    --ip-config-name ipconfig1 \
    --lb-name bbna-mn-ilb \
    --address-pool bbna-mn-vm-pool
   ```

Verify IPSec connectivity and MN-VM1 (and MN-VM2) failures

1. Restart IPSec tunnels on both ubuntu1 and ubuntu2 after ssh'ing into the units
   ```
   sudo ipsec restart
   ```

2. Test connectivity via ping test between ubuntu1 and ubuntu2. As an additional step, you can use "sudo tcpdump -n esp" on the receive side to make sure ping packets are going via the IPSec tunnel.

3. During on-going ping tests, bring down MN-VM1 and observe IPSec traffic failing over to the MN-VM2.

4. Bring back MN-VM1 and repeat the test with bringing down MN-VM2.

Notes

1. Adjust Azure inbound/outbound port rules based on your corporate security requirements.
2. The following ports should be open: udp/123 (ntp), udp/9899 (vll), udp and tcp/18927 (vll), tcp/18926(vll), tcp/7022(remote maintenance), tcp/8081 (remote GUI access and health monitor)
3. Make sure to enable "packet forwarding" for VNA interfaces
4. If IPSec endpoint is designed to have different inbound and outbound IP addresses, adjust the corresponding UDR routes accordingly.
5. If preferred, MN-VM1 WAN interface, MN-VM2 WAN interface and the ILB frontend IP addresses can all be in the same subnet.
6. If there is no option for having unique IP addresses for the branch office IPSec endpoints, you will need to use a "NAT" firewall rule instead of the "no NAT" firewall rule, whereby each NAT rule will require its own Azure IP address to identify each branch.