CIAB Mesh VPN Internetwork Overlay Architecture

Using BGP, VxLAN, BGP VRF and LXD

to implement LXD VM & Container

Multi-Node, Multi-Cloud/Hybrid Systems

Single Tenant Configuration

2023 by Brian Mullan (bmullan.mail@gmail.com)

The Cloud in a Box (CIAB) Mesh VPN is a software-defined wide-area network that is abstracted from hardware, creating a virtualized network overlay.

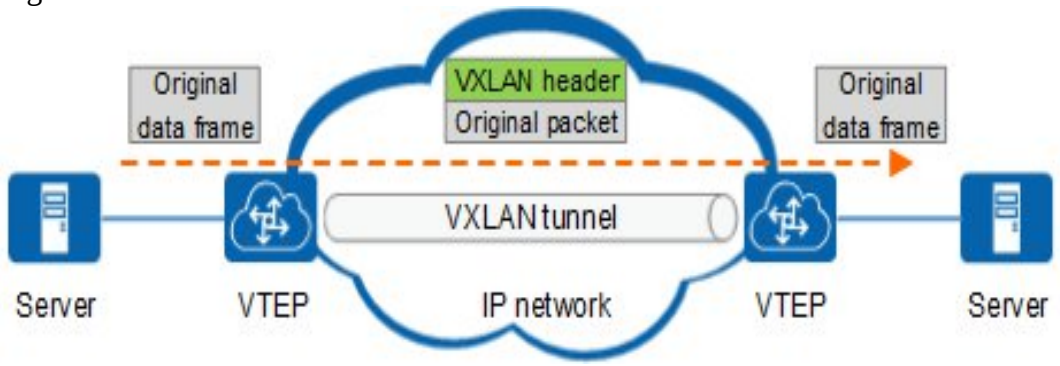
CIAB Mesh VPN Internet Overlay Architecture

The CIAB Mesh VPN Internet Overlay Architecture is implemented using several open source linux networking tools/apps, Routing and protocol capabilities including:

Free Range Routing (FRR) - FRR is an IP routing protocol suite for Linux and Unix platforms which includes protocol daemons for BGP, OSPF, RIP etc.

Virtual eXtensible LAN (**VxLAN**) – **an overview of VxLAN and Linux by Vincent Bernat**

Figure 1-1 VXLAN network model



A VxLAN network is a virtual Layer 2 network constructed on a Layer 3 network to enable communication of hosts at Layer 3

BGP w/Route Reflectors – **a general overview**

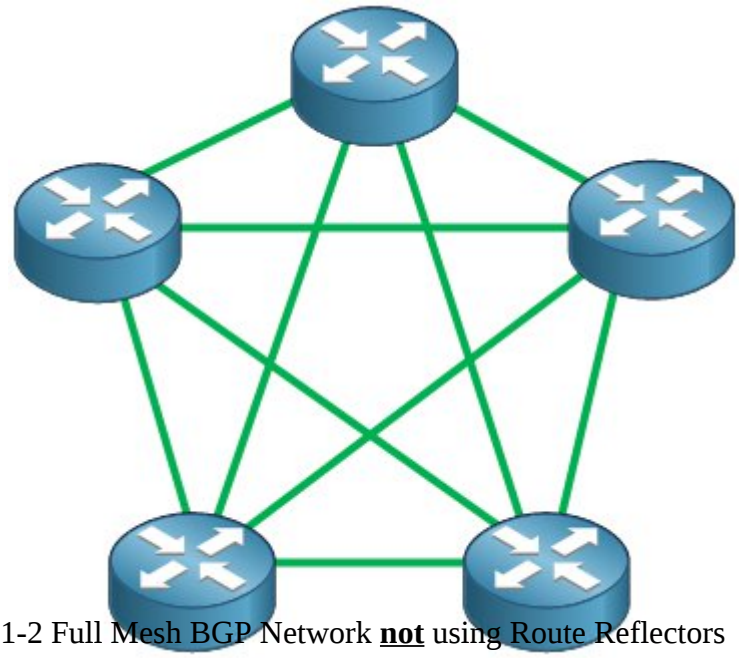


Figure 1-2 Full Mesh BGP Network **not** using Route Reflectors

BGP Route reflectors (RR) are one method to get rid of the full mesh of iBGP Peers in your network. While still providing *Routing information to all nodes about all nodes.*

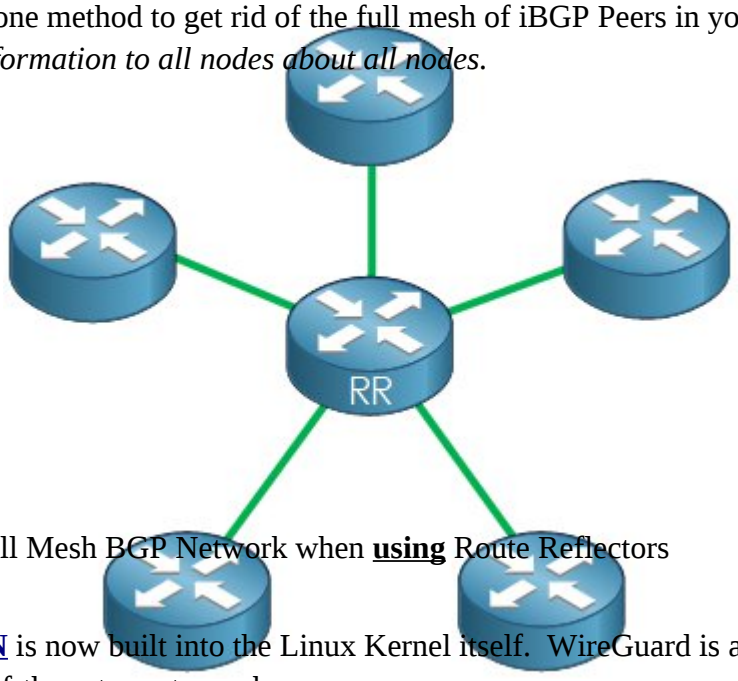


Figure 1-3 Full Mesh BGP Network when **using** Route Reflectors

WireGuard – **WireGuard VPN** is now built into the Linux Kernel itself. WireGuard is a fast and modern VPN that utilizes state-of-the-art cryptography.

LXD – **LXD is a System VM and Container Hypervisor**

VxWireGuard-Generator - **Utility to generate VxLAN over Wireguard mesh SD-WAN configuration**

VxWireGuard Background

VxWireguard-Generator (vwgen) has three layers of IP addresses that it adds to every CIAB Network Node's Wireguard config file (re //etc/wireguard/ciabmesh.net):

The "**public ip**" address -
They are the your Node/Server/Host's real Public IP Addresses.

Wireguard uses these addresses to establish VPN Tunnels between Nodes.

The VxLAN Virtual Tunnel End Point (VTEP) address -
They are allocated randomly from 169.254.0.0/16 and IPv6 link-local address space, used to build the VxLAN network overlay.

The IP addresses used by Wireguard: -
They are allocated from pool-ipv4 and pool-ipv6 you specify during configuration.

Since all nodes are then connected to one *"virtual" LAN*, preferably you may want these addresses to reside in one single subnet..

If you are not satisfied with the automatic allocation, for IPv4, you can edit the address manually; for IPv6, you can add more than one address and set one of them as Primary.

Goals of the CIAB Mesh VPN Internetworking Overlay Project

The ideal solution to satisfy the goals of this project will include my identified Key Performance Indicator (**KPI**) success factors:

1. **Security and Secure communications**
2. **Open Source**
3. **Supports use of LXD Containers & VMs and LXD related technologies**
4. **Easy/simple installation, configuration and expansion**
5. **Multi-node (re Multi LXD Host/Server) capable, intranet or Internet**
6. **Multi-Cloud and Hybrid Cloud capable**

Installation Guide
CIAB Mesh VPN Internetworking Overlay

2023 Brian Mullan (bmullan.mail@gmail.com)

Assumptions

A) Installation is being done on either Ubuntu 22.04 Desktop or Server systems.

- By "system" we mean a some PC/Laptop/Server, Virtual Machines or a Cloud Instances. Any or all of which can be part of your CIAB Mesh VPN Overlay Network.

B) Make sure LXD is installed & “init’d” on all Nodes.

C) **Steps # 5, 6, 7, 8, 9, 10, 11 – (***do these first on 1 Node of your choice***)**
These steps **need only be performed on only one node** because they will perform configurations that will be used later to configure ***all*** nodes.

D) **Steps # 1, 2, 3, 4, 12, 13, 14, 15**
These steps **must be** performed on **ALL** *Nodes/Hosts/Servers/VMs intended to utilize the CIAB Mesh VPN.*

The steps create configuration files for FRR’s BGP (/etc/frr/bgpd.conf) edited for appropriate IP addresses in the *template supplied in Appendix #1*.

E) This Guide Installation does **not** cover multi-tenant configuration although I have done this using additional configuration steps contained in a separate document.

Step 1

Create an initial Ubuntu LTS Node using:

- a Server or
- a VM or
- a Cloud-Instance

Whichever you intend to become the Nodes in your Mesh VPN Overlay.

Step 2

Install Free Range Routing (FRR) **on each** of your Nodes:

$ sudo apt install frr -y

Install Wireguard and Wireguard-tools **on each** of your Nodes:

$ sudo apt install wireguard wireguard-tools -y

Step 3

Edit the FRR daemons Config File on each of your Nodes (use Nano, Vi or whatever text editor):

$ sudo nano /etc/frr/daemons

Change :

bgpd=no
to: **bgpd=yes**

Then save the file w/changes made in Step 3.

Step 4

Use wget to retrieve the ciabvpn-utility-scripts.tar.gz from the Github repo:

$ cd

$ mkdir ciabscripts

$ cd ./ciabscripts

NOTE: ***the following is 1 single line*** *to be executed:*

> **$ wget https://github.com/bmullan/CIAB.Full-Mesh.VPN.Wireguard.FRR.BGP.VXLAN.Internet.Overlay.Architecture/raw/master/ciabvpn-scripts.tar.xz**

Copy (scp) the ***ciabvpn-scripts.tar.gz*** file to each Node that will be configured to join the CIAB Mesh VPN. (maybe your Home directory on each

> **example:** scp ./ciabvpn*.gz yourID@Node2_IP:/home/yourID/

Then ssh to each Node as yourID and do the following to uncompress the CIAB scripts onto that NODE and install them into /*usr/*bin

> **$ cd**
>
> **$ mkdir ciabscripts**
>
> **$ mv ./ciabvpn*.gz ./ciabscripts/**
>
> **$ cd ./ciabscripts**
>
> **$ tar -xvf *.gz**
>
> **$ chmod +x ./*.sh**
>
> **$ chmod +x ./generate***
>
> **$ sudo mv *.sh /*usr/*bin/**
>
> **$ sudo mv generate* /*usr*/bin/**

The files that you just put into /user/bin were:

Script Name	Purpose
wg-meshup.sh	Start Wireguard on a Node
wg-meshdown.sh	Shutdown Wireguard on a Node
NOTE: Wireguard must be UP on a Node before Starting FRR	
frr-start.sh	Start FRR on a Node

frr-restart.sh	Stop then Start FRR on a Node
frr-status.sh	Show FRR Status on a Node
frr-stop.sh	Stop FRR on a Node

NOTE: *You must "start" FRR before executing any of the following...*

bgp-start.sh	Start FRR's BGP on a Node
bgp-restart.sh	Stop/Start FRR's BGP on a Node
bgp-stop.sh	Stop FRR's BGP on a Node

NOTE: the following is a compiled binary program whose Source Code can be found on Github at: **https://github.com/althea-net/generate-ipv6-address**

generate-ipv6-address	**Used by wg-meshup.sh** to assign a unique IPv6 address to that node's Wireguard Interface Generate-ipv6-address generates IPv6 addresses from a given prefix and either a given MAC-48 address (an Ethernet hardware address) or a randomly drawn host number.

Step 5

Install/configure **VxWireguard-Generator**.

VxWireguard-Generator is a tool which will be used to first generate a Master WireGuard Mesh Config file.

After the Master Config file has been built we will use the VxWireguard-Generator again to extract each individual Node's Wireguard/VxLAN **ciabmesh.conf** config file.

Log into the Node using ssh or however you do it.

Make sure UNZIP is installed:

$ sudo apt install unzip -y

Use "wget" to retrieve the VxWireguard-Generator .ZIP file from github (install wget if its not already).

$ wget https://github.com/m13253/VxWireguard-Generator/archive/master.zip

Unzip the VxWireguard-Generator

$ unzip ./master.zip

Change into the unzipped Directory that was created:

$ cd ./VxWireguard-Generator-master

Install pre-requisites for VxWireguard-Generator.

Install python3-pip:

$ sudo apt install python3-pip -y

Install VxWirguard-Generator misc files

$ sudo pip3 install -r requirements.txt

$ python3 setup.py build

$ sudo python3 setup.py install --force

Step 6

Create a Master config file for our CIAB Mesh VPN using ***VxWireguard-Generator***

First, create a work directory (let's call it "/opt/ciabvpn"):

$ sudo mkdir /opt/ciabvpn
$ sudo chown yourID:yourID /*opt*/ciabvpn

change into that work directory

$ cd /opt/ciabvpn

NOTE: *The **VxWireguard-Generator's main cli tool is "vwgen"**.*

To view a vwgen's help page:

$ vwgen --help

Usage vwgen <sub-commands> [<args>]

Available sub-commands

***show**: Shows the current configuration of the mesh network*

***showconf:** Generate a configuration file for a given node*

***add**: Add new nodes to the mesh network*

***set**: Change the configuration of nodes*

***del:** Delete nodes from the mesh network*

***blacklist**: Manage peering blacklist between specified nodes*

***zone**: Generate BIND-style DNS zone records*

***genkey**: Generates a new private key and writes it to stdout*

***genpsk**: Generates a new preshared key and writes it to stdout*

***pubkey**: Reads a private key from stdin and writes a public key to stdout*

You may pass '--help' to any of these sub-commands to view usage.

Start to build the Master VxWireguard-Configuration file (we'll call ours ciabmesh):

$ vwgen add ciabmesh

This will create a file a Master config file called "**ciabmesh.conf**"

Set the ***IPv4 and IPv6 Address Pool*** for our CIAB Mesh Internet Overlay.

The following will cause *VxWireguard-Generator* to utilize IP addresses from these IPv4 and IPv6 address Pools for the configuration of your CIAB Mesh Internetwork.

$ vwgen set ciabmesh pool-ipv4 172.16.10.0/12 pool-ipv6 2001:db8:42::/64

Step 7

Decide now if you want to create an exact number of Node Config files or if you want to generate more (many) than you need today.

Then execute the next step.

Example: say you only want to create configs for our demo's 3 nodes

$ vwgen add ciabmesh node1 node2 node3

The above command will ***inject*** modifications into our Master "ciabmesh.conf" file with separate Sections for each Node (ie "node1", "node2" etc).

Config Hint: if you want to create *many* Node configs using a "root" name part and a sequential Node number here is an example of how to do that:

Say you want to create a Master config file that has 50 Node configs. The name "root" is to be "mycn" and we want 50 of them:

```
$ vwgen add ciabmesh $(seq -f 'node%.f' 50)
```

This would create a Master config file with 50 Node Configs each named: ***node01, node02 … node49, node50****, Each with its VxLAN TEP IP, the Node's public ip and its encryption key to become part of the CIAB Mesh network.*

Background Note

*The **Internet Assigned Numbers Authority (IANA)** regulates what Port Numbers may be used for what purposes on the Internet.*

Port Number Ranges and Well Known Ports

A port number uses 16 bits and so can therefore have a value from 0 to 65535 decimal

*Port numbers are divided into **ranges** as follows:*

***Port numbers 0-1023 – Well known port range**. These are allocated to server services by the Internet Assigned Numbers Authority (IANA). e.g Web servers normally use port 80 and SMTP servers use port 25 (see diagram above).*

***Ports 1024-49151- Registered Port range** -These can be registered for services with the IANA and should be treated as semi-reserved. User written programs should not use these ports.*

***Ports 49152-65535** – These are used by client programs and y**ou are free to use these** in client programs. When a Web browser connects to a web server the browser will allocate itself a port in this range. Also known as ephemeral ports.*

Given the previous Note, you should pick and use an Ephemeral Port Number between 49152-65535 for your CIAB Mesh Internetwork configuration use.

For this document and our examples, *I'll use Port 50000.*

Step 8

Set **public ip** of node1, node2 and node3, t***o each Node's Public Interface IP address*** (note: either IPv4 or IPv6 will work).

Example: suppose Node1 is a Cloud Instance and it's Internet facing <***public ip***> address is 132.65.71.21 then the command for Node1 would be:

$ vwgen set ciabmesh node node1 endpoint '132.65.71.21:50000' listen-port 50000

Now, execute the following for each of your Nodes substituting each Node's Public Interface IP address:

$ vwgen set ciabmesh node node1 endpoint '<public ip node1>:50000' listen-port 50000

$ vwgen set ciabmesh node node2 endpoint '<public ip node2>:50000' listen-port 50000

$ vwgen set ciabmesh node node3 endpoint '<public ip node3>:50000' listen-port 50000

Step 9

Show all information we have so far for cursory verification.

IMPORTANT NOTE: The following *will also show you **each Node's VTEP IP*** !

$ vwgen show ciabmesh | more

Note:

*You may want to execute the above command now and record/copy the **VTEP IP** for each Node.*

Step 10

vwgen, the VxWireguard-Generator tool, can be used to parse and extract from the Master Config file (**ciabvpn.conf**), each Node's individual Wireguard/VxLAN Config file.

Note: Repeat this for each Node or create a script to loop and do it for you

Example: Use **vwgen** to extract each Node's FRR config file from the Master config file.

$ vwgen showconf ciabmesh node1 > node1.conf

$ vwgen showconf ciabmesh node2 > node2.conf

$ vwgen showconf ciabmesh node3 > node3.conf

Step 11

Copy (scp or whatever) each Node's "*nodeN.conf*" file to create **that** actual Node's Wireguard config file located in that Node's own:

/etc/wireguard/ciabmesh.conf

So for Node1, you will be copying node1.conf and renaming it to ciabmesh.conf In the ***/etc/wireguard*** directory of the Node1 Server (or VM orcloud-instance).

Note: When you installed FRR in Step 2 it would have created /etc/frr/

Step 12

Copy the Free Range Routing (FRR) CIAB BGP and VRF Configuration Template from this document's ***Appendix #1*** to each node's **/etc/frr/** directory and renaming it "**frr.conf**" (re **/etc/frr/frr.conf**)

Step 13

Edit **/etc/frr/frr.conf** on **each** Node and where indicated in the template add the requested IP address information.

*Remember, you can **get each Node's VTEP (Virtual Tunnel End Point) IP address** by executing in the directory where you created the original "master" config file:*

$ vwgen show ciabmesh | more

Before Proceeding Please Reboot each of your Nodes Now !

Step 14

Start up the CIAB Mesh VPN Internetworking Overlay on each/all Nodes

SSH into each of your Nodes and start both Wireguard and FRR on all Nodes using the appropriate CIAB MESH VPN utility bash script from those you copied to each node previously in Step #4

In each Node execute:

$ sudo wg-meshup.sh

$ sudo frr-start.sh

$ sudo bgp-restart.sh

Step 15

On each Node execute to show that Node's interfaces information:

$ ip addr

*You should see **several additional interfaces** on each **Node** now, including:*

vciabmesh	***The Node's VxLAN VTEP Interface and IP***
ciabmesh	***The Node's Wireguard VPN Interface and IP***

Validate that from any Host/Node you can now successfully Ping the IP address of any other Host/Node's LXD Container(s).

***If there is a problem, often is with a mistake made entering IP addresses in one or more Node's /etc/frr/frr.conf* file**

Congratulations

You now have a fully encrypted, Full-Mesh VPN Overlay Network that should work with your Servers and VM's anywhere they are located whether in-house, inter-divisions, or inter-Cloud to interconnect LXD Virtual Machines & LXD Containers.

Now your Applications and Users in any location can easily access Distributed Applications and Databases etc anywhere in the Internet with CIAB's MESH VPN.

Appendix #1

Raw (ie uncommented) /etc/frr/frr.conf script follows

Note: copy this to create each node's initial */etc/frr/frr.conf then edit that file*

log syslog informational

hostname node1

password <enter some password>

enable password <enter some "enable" password if you use FRR's VTYSH to do configs>

router bgp 64512

bgp router-id <insert "***this***" NODEs VTEP IP address here>

bgp neighbors

neighbor <insert Node1's VTEP IP address here> remote-as 64512

neighbor <insert Node2's VTEP IP address here> remote-as 64512

neighbor <insert Node3's VTEP IP address here> remote-as 64512

route reflectors

neighbor <insert Node1's VTEP IP address here> route-reflector-client

neighbor <insert Node2's VTEP IP address here> route-reflector-client

neighbor <insert Node3's VTEP IP address here> route-reflector-client

Identify this Node's LXD network (on Host execute $ lxc list command).

This will advertise each Node's LXD network through the

BGP Route Reflector's to the other Nodes in the CIAB Mesh VPN.

network 10.0.1.0/24

Appendix #2

Fully commented /etc/frr/frr.conf script

CIAB BGP and VRF Configuration Template for Nodes in a CIAB Secure Mesh Internet Overlay.

#===

Name: ***CIAB BGP and VRF Configuration Template***

CIAB Mesh VPN using BGP, WireGuard and VxLAN

```
#                       2023 Brian Mullan (bmullan.mail@gmail.com)
#==================================================================
# The following is the BGP and BGP Route Reflector Config file for
# each NODE in the CIAB MeshVPN.
#
# There are comments explaining where and what to change for each
# individual NODE.
#
# Copy this Template and edit it to insert appropriate IP addresses etc.
#
# Then SCP the template file to its corresponding Server/Host/Node
# and rename it on each node  to "/etc/frr/frr.conf" as it will become the Free Range Routing's
# (FRR) config file when FRR is run.
#
#===={Begin FRR BGP Configuration - Copy everything below this line}==
#  NOTE:
#     If you use Quagga comments begin with an exclamation "!"
#     If you use FRR you can use either a Hash or exclamation mark.
#     I recommend using FRR as it seems to be moving faster that quaga todaay.
#
#==================================================================
# Default to using syslog. /etc/rsyslog.d/45-frr.conf places the log
# in /var/log/frr/frr.log
log syslog informational


#==================================================================
# CIAB Ubuntu BGP and BGP  Route Reflectors Configuration
# Each Node will require is FRR config file located in - /etc/frr/frr.conf
#
# 2023 brian mullan (ciab) for Architectural Design combining
```

```
# LXD, BGP, BGP Route Reflectors, VxLAN and WireGuard to create
# an Internet Overlay Network VPN.
#=================================================================
# FRR BGPd configuration file for CIAB network node.
#
# Node's FRR config includes the Virtual Routing and Forwarding (re VRFs)
# defined for each of the other LXD Host/Server nodes in the rest of the CIAB Mesh Network.
#=================================================================
hostname node1
password <enter some password>
enable password <enter some "enable" password if you use FRR's VTYSH to do configs>

#=================================================================
# bgp multiple-instances...
#=================================================================
# Define Autonomous System (AS) number to use with BGP  (re 64512) and Tenant Network.
#
# IMPORTANT NOTES:
#
# 1) You can ONLY use IANA approved "Private" AS Numbers.
#    REFER to Section 5 - https://tools.ietf.org/html/rfc6996
#
# 2) Both Quagga & FRR BGPd support "multiple VRF instance"
#    configurations where in the Config you can designate
#    Multiple AS Numbers.  To utilize this in multi-tenant use-cases
#    please research this topic if your intent is to implement one unique AS number
#    for each Multi-Tenant.
#
#    The FRR Documentation for this topic can be found here:
#    http://docs.frrouting.org/en/latest/bgp.html#multiple-autonomous-systems
```

#

Although this Guide/Template doesn’t cover Multi-Instance BGP AS configuration

If more than one AS is defined then a Node’s BGP would know to participate in more than

node AS Network.

#===

#===={Define Tenant BGP AS numbers}===

BGP AS for tenant1

router bgp 64512

#===

Node0 is a Node using its VRFs to provide

Routing information to each of its Peers.

#

Add BGP Peer nodes. One entry for each WireGuardud LXD Host/Server

node using that Host/Server's VxLAN Tunnel End Point (TEP)

interface IP address and the same Autonomous System (AS) number as

defined above.

#

#===

ID “this” BGP Node by its own VxLAN Tunnel End Point (TEP) IP

address. These will the 172.16.10.x addresses from our IPv4 “Pool” we specified

when we used vx-wireguard-generator to create the Master config file and later

extracted each Node’s individual WireGuard config.

#===

bgp router-id <insert "***this***" Node’s VTEP IP address here>

#===

As Node0 is a Node using its VRFs to provide

Routing information to each of its sub-ordinate Peers.

#

Add BGP Peer nodes. One entry for each WireGuardud LXD Host/Server

Node using that Host/Server's WireGuard VxLAN interface IP address

and the same Autonomous System (AS) number as defined above.

#

NOTE:

Again, if this is a Multi-Tenant or a Multi-Network configuration

EACH of these "could" be in a different AS Number which would

effectively separate Traffic like a VPN would do except this would

also Route via BGP.

#

This Node0 would have to have more than 1 "router bgp <AS number>"

statement though so it’d know to participate in more than one BGP

AS network routing operation.

#

#===

Format:

neighbor <WireGuard Node1’s VTEP IP> <AS number>

#

#

Hint: If you were configuring with FRR BGP Multi-AS support you could “conceptually”

think of each BGP AS number as a Tenant ID. Again, we are not doing that here.

Node0 -

neighbor <insert VTEP IP address here> remote-as 64512

Node1 -

neighbor <insert VTEP IP address here> remote-as 64512

Node2 -

neighbor <insert VTEP IP address here> remote-as 64512

#===={example Route Reflector config statements}=====================

we ID the Route Reflector Clients nodes

Node0 -

neighbor <insert VTEP IP address here> route-reflector-client

Node1 -

neighbor <insert VTEP IP address here> route-reflector-client

Node2 -

neighbor <insert VTEP IP address here> route-reflector-client

#===

Tell BGP to advertise/share "this" NODE's "local" LXD IP Subnet (/24 subnet of LXDBR0)

route info to its BGP AS Peers

NOTE 1: *my example is using 10.0.1.x, yours is likely different*

NOTE 2: you can list multiple LXD subnets here if you have more than one LXD bridge

#===

network 10.0.1.0/24