

Contents

- [1 Objectives](#)
- [2 Prerequisites](#)
- [3 Topology](#)
- [4 Install Process](#)
 - ◆ [4.1 Step 1: Installing DevStack](#)
 - ◆ [4.2 Step 2: CSR Preparation](#)
 - ◆ [4.3 Step 3: Adapting OpenStack To Work With The CSR](#)
 - ◆ [4.4 Step 4: OpenStack Startup](#)
 - ◆ [4.5 Step 5: CSR Configuration](#)
 - ◆ [4.6 Step 6: Setting up the other host](#)
- [5 Usage](#)
- [6 CSR Restart](#)
- [7 Reference Information](#)

Objectives

Note: This guide is for users of OpenStack Icehouse or Juno releases. If you are using the Kilo or Cisco Juno+ release, check out [these notes](#).

This guide provides instructions on how to set up a CSR to provide site-to-site IPsec VPNaaS for an OpenStack cloud computing environment. The CSR will operate as an out-of-band virtual router connected in parallel with a companion Neutron router to provide the ability to establish VPN connections for tenants of OpenStack.

For simplicity, the guide will use DevStack as a way to start up OpenStack. However, the same procedures can be applied to a traditional OpenStack deployment (albeit with some manual steps at this time).

The OpenStack VPNaaS feature is an experimental release, and APIs may change in the future. These procedures describe a manual process to set up VPN using the CSR. The goal in the future is to integrate the CSR into OpenStack, so that use with VPNaaS (and other services) will be seamless.

Note: This software is provided "as is," and in no event does Cisco warrant that the software is error free or that customer will be able to operate the software without problems or interruptions.

Prerequisites

There are several things needed to setup an OpenStack cloud for VPN operation with CSR:

- A CSR 3.13 .iso image (EFT image should be available in May 2014)
- CSR license (contact Cisco Sales)
- Use OpenStack Icehouse RC1 or newer (keep in mind that changes in OpenStack and CSR images need to be in sync)
- Helper scripts for using up the CSR under OpenStack - <https://github.com/CiscoSystems/openstack-csr>
- Adequate hardware (CPU/memory) for running OpenStack with CSR images (TODO link to UCS...)
- Connectivity to another OpenStack (e.g. DevStack) cloud or compatible IPsec VPN site-to-site device (virtual or physical).

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

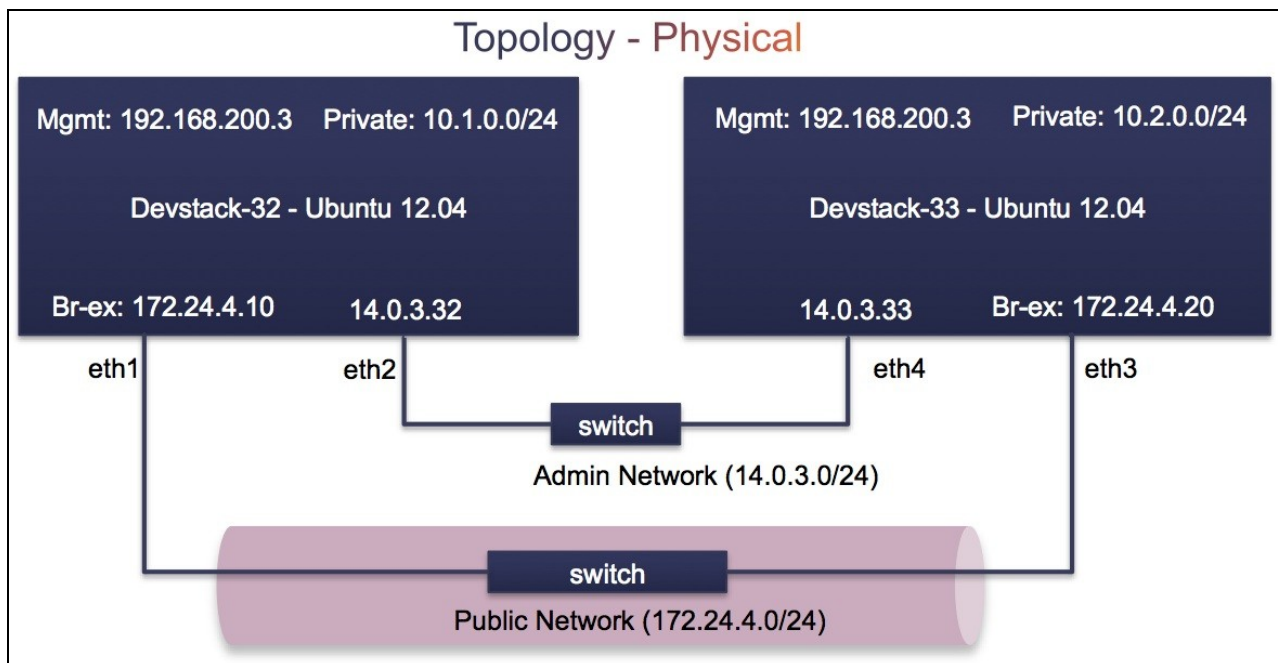
Assumptions:

- You have a basic understanding of how to start up DevStack (<http://devstack.org/>).
- You have some familiarity with how to configure (from a user's perspective) OpenStack's VPN with either Horizon or the Neutron client.
- You're using Ubuntu Server 14.04 LTS (64 bit, obviously) on your host will all current updates. Using a different operating system is left as an exercise for the reader.

In this guide, we'll focus on the detailed steps needed to set up one of the two sites for the IPSec site-to-site VPN connectivity. The example uses the same method for the other site (with just brief info on the different IPs used), and you can follow the same instructions to create a complete site-to-site connection with two clouds using a CSR. Alternately, you can employ the reference OpenStack VPN implementation or a compatible virtual or physical VPN device for the other end of the site-to-site VPN connection.

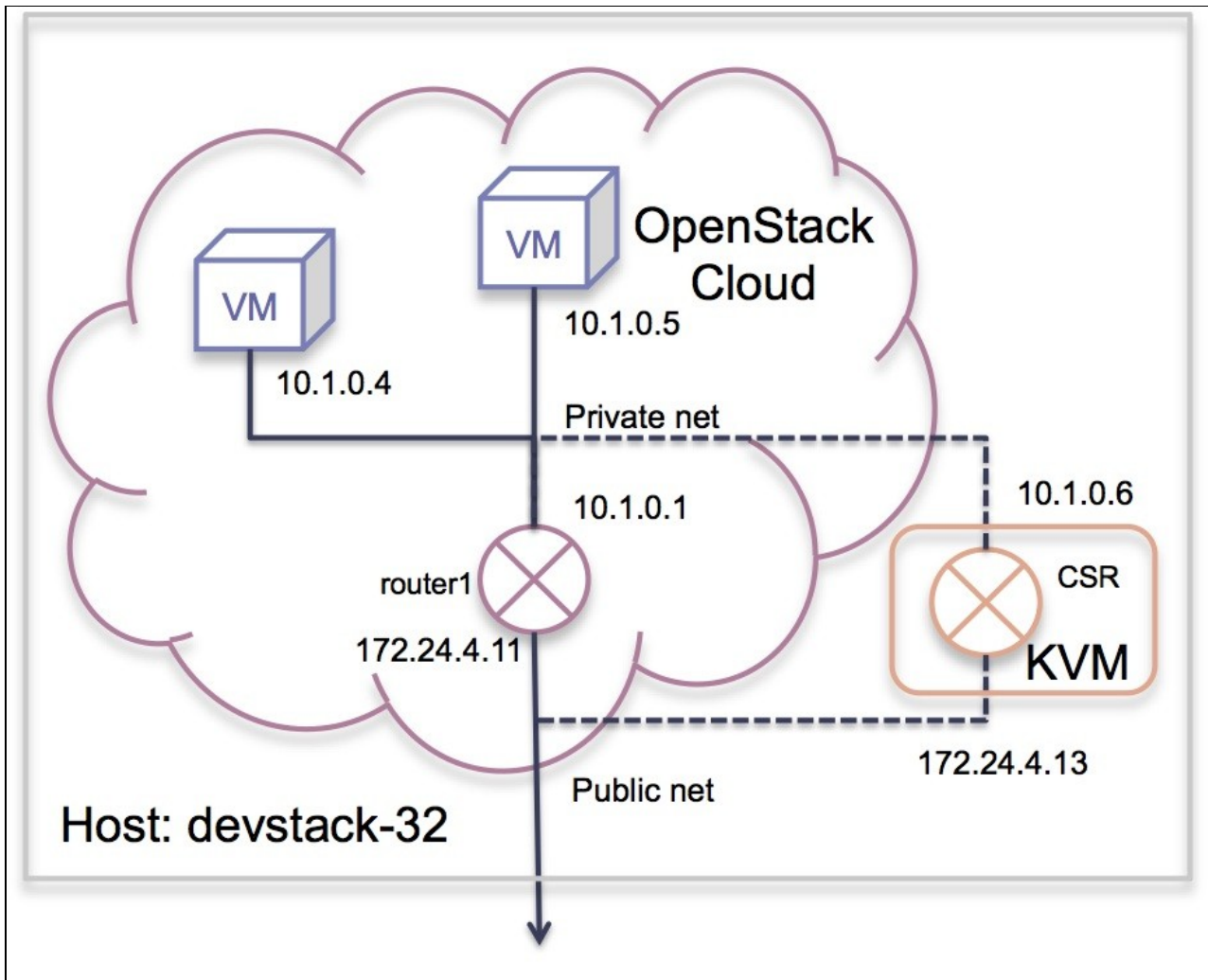
Topology

In the example configuration, we have this physical topology:

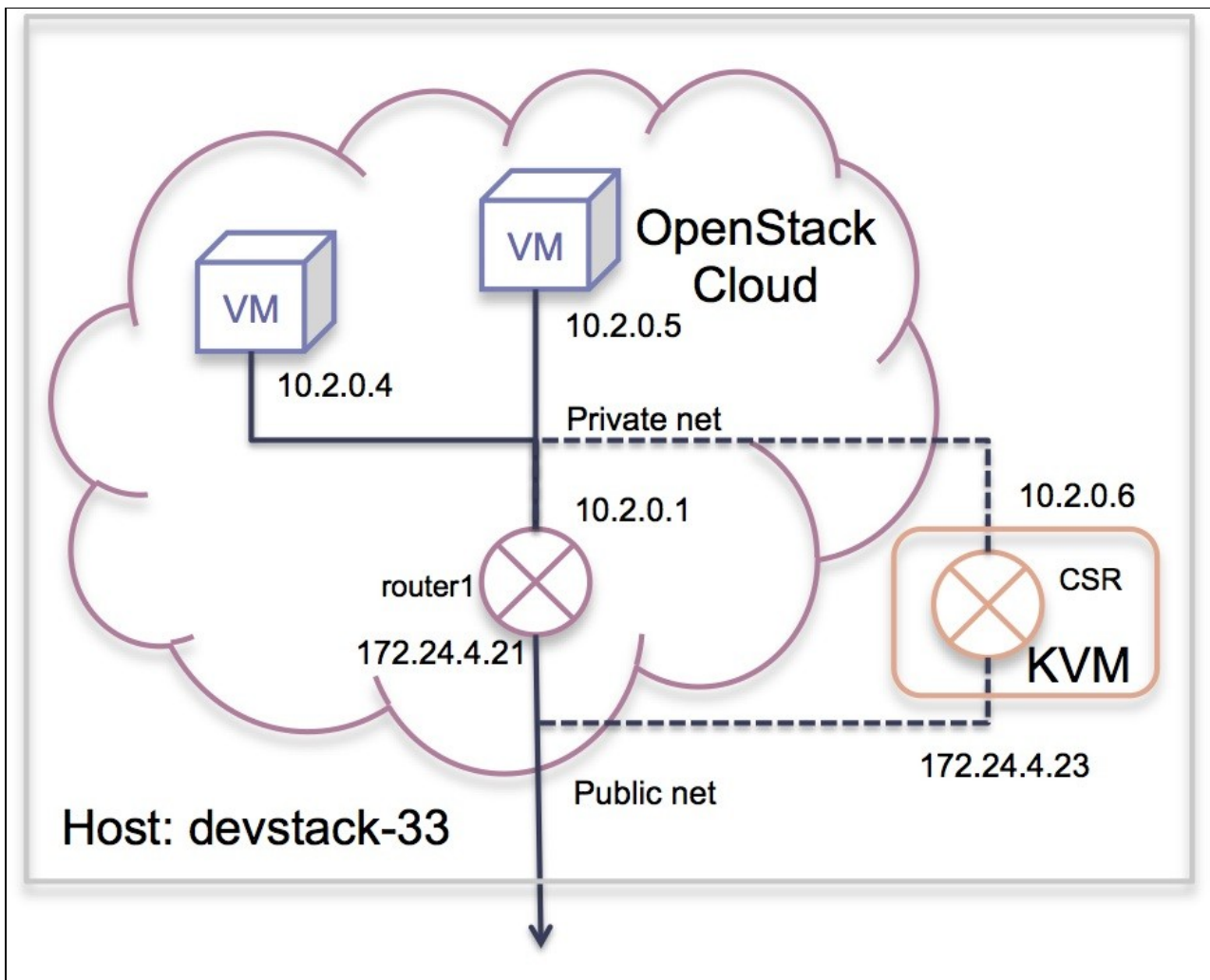


There are two UCS systems, which will each host an OpenStack (using DevStack) cloud for the site-to-site VPN end-points. A physical connection between the two nodes will be used for the "public" network (172.24.4.0/24), where traffic between the private networks (10.1.0.0/24 and 10.2.0.0/24) will be encrypted. There is an internal network on the node (192.168.200.0/24) for management of the CSR during operation, and another network (14.0.3.0/24) for setup of the clouds.

Logically, we have the following topology for the first cloud, running on the devstack-32 host:



You can see the IPs that will be used for public and private networks. For the other host, devstack-33, we have:



The key information in these drawings are the IP addresses used for the CSR (private and public), the public IP of the peer CSR, and the physical ethernet interface being used.

Install Process

Step 1: Installing DevStack

Recording of steps 1 & 2:

<https://cisco.webex.com/ciscosales/lsr.php?RCID=177e1f5a051546189673ec5fe2dddc2f>

The first thing we need to do, is obtain all of the OpenStack code and install it on the system. We'll use DevStack to do that, but at the same time, we'll start to prepare DevStack for use with the CSR. Obtain the latest DevStack repo with:

```
cd
git clone git://github.com/openstack-dev/devstack.git
cd devstack
```

For the devstack-32 host, we have these attributes to consider:

- Local network (FIXED_RANGE) will be 10.1.0.0/24 with router (GW) using 10.1.0.1
- Public GW at 172.24.4.10 on the 172.24.4.0/24 public network (will be the br-ex interface).
- Set of floating IPs reserved on the public network in the range 172.24.4.11 - 172.24.4.19

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

- Using eth1 for the public network interface.

Here is the **localrc** file for devstack-32, using the above information:

```
GIT_BASE=http://git.openstack.org
DEST=/opt/stack
disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
enable_service tempest
enable_service q-vpn

API_RATE_LIMIT=False
VOLUME_BACKING_FILE_SIZE=4G

FIXED_RANGE=10.1.0.0/24
FIXED_NETWORK_SIZE=256
PRIVATE_SUBNET_NAME="mysubnet"
NETWORK_GATEWAY=10.1.0.1

FLOATING_RANGE=172.24.4.0/24
PUBLIC_SUBNET_NAME="yoursubnet"
PUBLIC_INTERFACE=eth1
OVS_PHYSICAL_BRIDGE=br-ex
PUBLIC_NETWORK_GATEWAY=172.24.4.10
Q_FLOATING_ALLOCATION_POOL=start=172.24.4.11,end=172.24.4.19
Q_USE_SECGROUP=False

VIRT_DRIVER=libvirt
SWIFT_REPLICAS=1
export OS_NO_CACHE=True
SCREEN_LOGDIR=/opt/stack/screen-logs
SYSLOG=True
SKIP_EXERCISES=boot_from_volume,client-env
ROOTSLEEP=0
ACTIVE_TIMEOUT=60
BOOT_TIMEOUT=90
ASSOCIATE_TIMEOUT=60
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=token
# Temp try ML2
# Q_PLUGIN=openvswitch
Q_USE_DEBUG_COMMAND=True

Q_VPN_EXTRA_CONF_FILES=(/opt/stack/neutron/etc/neutron/plugins/cisco/cisco_vpn_agent.ini)

# RECLONE=No
# OFFLINE=True
RECLONE=yes
OFFLINE=false
```

All of the OpenStack software will be installed, by doing **./stack.sh**. Once completed, you can run **./unstack.sh** and comment out the RECLONE line in localrc: Now that Neutron code has been installed, you can modify localrc to turn off recloning and set offline operation:

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

```
RECLONE=No
OFFLINE=True
# RECLONE=yes
# OFFLINE=false
```

Step 2: CSR Preparation

For this step, we have these attributes for the devstack-32 host:

- Host IP is 14.0.3.32
- Local (private) IP for CSR will be 10.1.0.6
- Public IP for CSR will be 172.24.4.13

NOTE: We've setup an account called **openstack** on our system, with the usual sudo permissions. This wiki assumes the files are under the account "openstack"; e.g. /home/openstack. If you have a different account, edit the commands accordingly.

Obtain the helper scripts that will be used to assist in installing, starting, and hooking the CSR to Neutron:

```
cd
git clone https://github.com/CiscoSystems/openstack-csr csr
cd csr
mkdir 3.13
```

Download your ISO CSR image into the 3.13 directory (if you happen to have a OVA image, it can be converted to an ISO - procedures TODO).

Copy the config.ini.sample to config.ini file and set the version to 3.13, the filename in the ISO environment variable to match the image placed in the 3.13 directory, and use the date of the image in the IMAGE environment variable. Also, set the HOST environment variable to the IP of the host the CSR will run on.

```
ISO="/home/openstack/csr/$VERS/csr1000v-universalk9.BLD_MCP_DEV_LATEST_20140325_064532.iso"
...
HOST="14.0.3.32"
```

NOTE: If you are using this same method on the far end's setup, be sure to change the UUID and MAC addresses, so that there are no conflicts on the public interfaces (do this on all I/Fs).

To do the install, run **sudo ./install-csr** as root. From another system (e.g. a Ubuntu desktop jump box) with access to this host, VNC into the host, on port zero (e.g. 14.0.3.32:0) to access the console of the CSR and watch the boot process. It takes 15-20 minutes, so be patient. When asked "Would you like to enter the initial configuration dialog", select yes and the following selections can be made:

- Basic management => yes
- Defaults for settings for host
- enable secret **lab**, enable password **lab** (ignore the warning about them being the same, and re-enter again)
- virtual term password **cisco**, which will be used when accessing via REST
- SNMP management "yes" and choose all defaults by pressing enter
- Enter **GigabitEthernet1** for management interface.
- Enter "yes" to configure IP and set an IP address and subnet mask for management access (e.g. 192.168.200.2/255.255.255.0)
- Select to save the configuration (option 2).

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

On the host, you can enter these to commands, to temporarily setup an interface to talk to the CSR:

```
sudo ovs-vsctl add-port br-csr to_csr -- set interface to_csr type=internal
sudo ifconfig to_csr 192.168.200.3/24 up
```

You can then Telnet to the CSR using the management IP set up (e.g. 192.168.200.2) with the password "cisco", enter enable mode "en" with password "lab", and provide the basic configuration:

```
config t
interface GigabitEthernet2
 ip address 172.24.4.13 255.255.255.0
 negotiation auto
 no shut
!
interface GigabitEthernet3
 ip address 10.1.0.6 255.255.255.0
 negotiation auto
 no shut
!
!
interface VirtualPortGroup0
 ip unnumbered GigabitEthernet1
!
transport-map type persistent webui http-restapi
 secure-server
!
virtual-service csr_mgmt
 vnic gateway VirtualPortGroup0
 guest ip address 192.168.200.20
 activate
!
transport type persistent webui input http-restapi
!
ip http server
ip http authentication local
ip http secure-server
ip route 0.0.0.0 0.0.0.0 GigabitEthernet1 192.168.200.1
ip route 192.168.200.20 255.255.255.255 VirtualPortGroup0
!
ip ssh version 2
username stack priv 15 secret cisco

remote-management
 restful-api local-port 443

license boot level premium
```

You must enter "yes" for the licensing, and when it is completed, "end" configuration and enter "write" to save the configuration. At this point, the basic CSR installation is completed, although the system must be rebooted to complete the licensing process. This will be done later, but for now, on the host, press control-C to shutdown the CSR VM. Use the following command to remove the temporary bridge that was created to install the CSR:

```
sudo ovs-vsctl del-br br-csr
```

Just for reference, on the other node, we would set up the configuration on the CSR with:

```
config t
interface GigabitEthernet2
 ip address 172.24.4.23 255.255.255.0
 negotiation auto
```

Step 2: CSR Preparation

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

```
no shut
!
interface GigabitEthernet3
 ip address 10.2.0.6 255.255.255.0
 negotiation auto
 no shut
!
!
interface VirtualPortGroup0
 ip unnumbered GigabitEthernet1
!
transport-map type persistent webui http-restapi
 secure-server
!
virtual-service csr_mgmt
 vnic gateway VirtualPortGroup0
 guest ip address 192.168.200.20
 activate
!
transport type persistent webui input http-restapi
!
ip http server
ip http authentication local
ip http secure-server
ip route 0.0.0.0 0.0.0.0 GigabitEthernet1 192.168.200.1
ip route 192.168.200.20 255.255.255.255 VirtualPortGroup0
!
ip ssh version 2
username stack priv 15 secret cisco

remote-management
 restful-api local-port 443

license boot level premium
```

The next step is to configure the CSR for REST access, but first OpenStack (DevStack) must be installed and started up...

Step 3: Adapting OpenStack To Work With The CSR

Recording of steps 3 and 4 (note, a file edit mistake was made at timestamp 02:15, but was corrected at timestamp 21:00 and there is useful info inbetween)

<https://cisco.webex.com/ciscosales/lsr.php?RCID=57b455ec881d4b82ad312c34da278b2a>

To prepare DevStack so that it starts up with the Cisco device driver and service driver, we can modify DevStack's library files and change the init files used. For the selection of the Cisco .ini file, the changes are already merged into DevStack and you only need to set the Q_VPN_EXTRA_CONF_FILES, as indicated later on, in the loclarc file.

To cause the Cisco service driver to be enabled as the default, apply this patch (or manually make these changes):

```
patch -p 1 << EOT
diff --git a/lib/neutron b/lib/neutron
index 02dcaf6..452281b 100644
--- a/lib/neutron
+++ b/lib/neutron
@@ -728,6 +728,7 @@ function _configure_neutron_fwaas {
     function _configure_neutron_vpn {
```


Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

```
neutron_vpn_install_agent_packages
neutron_vpn_configure_common
+ neutron_vpnaas_configure_driver
}

# _configure_neutron_plugin_agent() - Set config files for neutron plugin agent
diff --git a/lib/neutron_plugins/services/vpn b/lib/neutron_plugins/services/vpn
index d920ba6..a676fdc 100644
--- a/lib/neutron_plugins/services/vpn
+++ b/lib/neutron_plugins/services/vpn
@@ -18,6 +18,10 @@ function neutron_vpn_configure_common {
    _neutron_service_plugin_class_add $VPN_PLUGIN
}

+function neutron_vpnaas_configure_driver() {
+  iniset_multiline $NEUTRON_CONF service_providers service_provider "VPN:cisco:neutron.services.vpn"
+}
+
function neutron_vpn_stop {
  local ipsec_data_dir=$DATA_DIR/neutron/ipsec
  local pids
EOT
```

The alternative to using the patch is to modify `/etc/neutron/neutron.conf`, each time, after stacking, to have this line uncommented (instead of the OpenSwan VPN line), and then restart the process:

```
service_provider=VPN:cisco:neutron.services.vpn.service_drivers.cisco_ipsec.CiscoCsrIPsecVPNDriver
```

It's more work, so the above patch is the preferred method.

After the service driver is setup, we want to enable the Cisco device driver by uncommenting this line in `/opt/stack/neutron/etc/vpn_agent.ini`:

```
vpn_device_driver=neutron.services.vpn.device_drivers.cisco_ipsec.CiscoCsrIPsecDriver
```

Before a VPN connection can be established, the CSR information must be placed in `/opt/stack/neutron/etc/neutron/plugins/cisco/cisco_vpn_agent.ini` (the name and location are specified in the DevStack localrc file using the `Q_VPN_EXTRA_CONF_FILES` environment variable). This can be specified in advance, as we have done here, by predicting what IP addresses will be assigned for the CSR. We know that the Neutron router created when DevStack starts, will get the first floating IP address available. The CSR, when started, will get the next available IP (which will be the third address, as a compute probe IP will be created). The REST management IP and other parameters are static. See the section below on CSR Restart for info on how to set up this information just prior to making the VPN connection.

For devstack-32, we have these attributes:

- Neutron router IP 172.24.4.11
- CSR IP 172.24.4.13

These are used in the file as follows:

```
[cisco_csr_rest:172.24.4.11]
rest_mgmt = 192.168.200.20
tunnel_ip = 172.24.4.13
username = stack
password = cisco
timeout = 60
```

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

The IP on the **[cisco_csr_rest:...]** line is the public IP for the Neutron router that is associated with the CSR (172.24.4.11 for devstack-32). Private network packets sent to the Neutron router, will be forwarded to the CSR, which will encrypt and send them over the tunnel established on the public network defined. The **rest_mgmt** IP is the predefined CSR IP address on the management network that will be used for REST messages. The **tunnel_ip** will be the public interface IP for the CSR (172.24.4.13 for devstack-32), and will be on the same subnet as the router's public interface (typically, Neutron will assign the first floating IP to the router, and the third to the CSR). This must match the IP that is assigned by Neutron, when the CSR is later started (see below).

Step 4: OpenStack Startup

Prior to starting DevStack for use with the CSR, the external bridge should be created and the ethernet interface added, and brought up. We'll create a script (prep.32) to run, as it will be used later.

```
cd /home/openstack/devstack
mkdir scripts
cat << EOF | tee scripts/prep.32
sudo ovs-vsctl del-br br-ex
sudo ifconfig eth1 down
sudo ifconfig eth1 up
sudo ovs-vsctl add-br br-ex
sudo ifconfig br-ex up
sudo ovs-vsctl add-port br-ex eth1
EOF

chmod 755 scripts/prep.32

scripts/prep.32
```

Note that we are specifying the physical interface, eth1, for this host so that Neutron can use this public network from br-ex.

If you are using a firewall, you may have to disable proxy for the CSR IPs with the following `no_proxy` command. Make sure the command is entered in the same terminal window where DevStack will be run.

```
export no_proxy=$no_proxy,192.168.200.20,192.168.200.2
```

At this point, you can start DevStack, by using **./stack.sh**. Once this startup has completed and everything is running, SSH and ping can be enabled on the private network, and two VMs created:

```
cat << EOT | tee scripts/build-vms.32
source openrc admin admin

# Allow admin tenant access to private net (needed?)
neutron net-update --shared=True private

source openrc admin demo

nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0

PRIVATE_NET=$(neutron net-list | grep private | cut -f 2 -d'|' | cut -f 2 -d' ')

nova boot --flavor 1 --image cirros-0.3.1-x86_64-uec --nic net-id=$PRIVATE_NET peter
nova boot --flavor 1 --image cirros-0.3.1-x86_64-uec --nic net-id=$PRIVATE_NET paul
EOT

chmod 755 scripts/build-vms.32
```

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

scripts/build-vms.32

Note, for the other host, you can use different image names (e.g. mary, thomas), and name the scripts appropriately (e.g. build-vms.33). You don't have to create VMs, but by not doing so, it changes the IP address assignment on the CSR. See below, for information on how to deal with that. For this example, we'll make sure we create two, so that everything lines up correctly.

To setup a management network for the CSR and port in OVS to access the CSR, do the following:

```
cat << EOF | tee scripts/csr
source openrc admin admin

# Create a management network for CSR
neutron net-create mgmt
neutron subnet-create --disable-dhcp --name=mgmt-subnet mgmt 192.168.200.0/24

sudo ovs-vsctl add-port br-int to_csr -- set interface to_csr type=internal
sudo ifconfig to_csr 192.168.200.3/24 up
EOF

chmod 755 scripts/csr

scripts/csr
```

Step 5: CSR Configuration

Recording of step 5 at

<https://cisco.webex.com/ciscosales/lsr.php?RCID=5b6a7040b8ea40ef8e36af3d39fd93b6>

Now that the Neutron network is all set up (and the two VMs created), the CSR VM can be booted up. In a separate terminal (because this will need to remain running as long as the CSR is up), use the following commands:

```
cd /home/openstack/csr
sudo ./startvm.sh
```

The CSR will boot up (this takes 5-7 mins), after which you can then Telnet to the CSR using the management IP set up (e.g. 192.168.200.2). The boot process can be monitored by VNC into the host at port 50 (e.g. 13.0.3.32:50, for the devstack-32 host). The configuration done previously, should be there, and interfaces should all be up/up. For GigabitEthernet2 (on public net) and GigabitEthernet3 (on private net), you should verify the IP addresses and make sure they match what was created by the Neutron binding (and make sure the public IP matches what you placed in the cisco_vpn_agent.ini file). Use:

```
cd /home/openstack/devstack
source openrc admin demo
neutron port-list
```

and check **public_p** and **private_p** IPs for GigabitEthernet2 and GigabitEthernet3, respectively. As mentioned before, the IPs should match, as we created two VMs, so the CSR should be at 10.x.0.6. If either of the addresses are wrong, then either they were entered incorrectly on the CSR or scripts, a different number of VMs were created, or the CSR was stopped and restarted (as Neutron will assign the next available IP each time the CSR is started). If things don't match, alter the CSR configuration as mentioned in the section CSR Restart, below.

You can verify that /etc/neutron/neutron.conf has the Cisco service driver specified as the default, /etc/neutron/vpnagent.ini has the Cisco device driver uncommented, and you can check the

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

/opt/stack/screen-logs/ log files to ensure that the Cisco drivers are loaded and used, and that the CSR settings from cisco_vpn_agent.ini are loaded (screen-q-svc.log and screen-q-vpn.log).

Note: In the future, each time you start DevStack, create VMs, and then start the CSR, as long as you use the same sequence, the private IP address will be the same.

Step 6: Setting up the other host

Recording of step 6 and usage at

<https://cisco.webex.com/ciscosales/lsr.php?RCID=01197bff7bfd4ad0866662fdf1c5b690>

As mentioned, the example configuration has a second host, devstack-33, which we set up in a similar manner. With this host we have different attributes. For the CSR preparation, we have:

- Host IP is 14.0.3.33
- Local (private) IP for CSR will be 10.2.0.6
- Public IP for CSR will be 172.24.4.23

For OpenStack setup, and the localrc file, we use:

- Local network (FIXED_RANGE) will be 10.2.0.0/24 with router (GW) using 10.2.0.1
- Public GW at 172.24.4.20 on the 172.24.4.0/24 public network.
- Set of floating IPs reserved on the public network in the range 172.24.4.21 - 172.24.4.29
- Using eth3 for the public network interface.
- We specify to use the cisco_vpn_agent.ini file via the Q_VPN_EXTRA_CONF_FILES

For reference, here is the file used:

```
GIT_BASE=http://git.openstack.org
DEST=/opt/stack
disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
enable_service tempest
enable_service q-vpn

API_RATE_LIMIT=False
VOLUME_BACKING_FILE_SIZE=4G

FIXED_RANGE=10.2.0.0/24
FIXED_NETWORK_SIZE=256
NETWORK_GATEWAY=10.2.0.1
PRIVATE_SUBNET_NAME=mysubnet

PUBLIC_SUBNET_NAME=yoursubnet
PUBLIC_INTERFACE=eth3
OVS_PHYSICAL_BRIDGE=br-ex
FLOATING_RANGE=172.24.4.0/24
PUBLIC_NETWORK_GATEWAY=172.24.4.20
Q_FLOATING_ALLOCATION_POOL="start=172.24.4.21,end=172.24.4.29"
Q_USE_SECGROUP=False

VIRT_DRIVER=libvirt
```

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

```
SWIFT_REPLICAS=1
export OS_NO_CACHE=True
SCREEN_LOGDIR=/opt/stack/screen-logs
SYSLOG=True
SKIP_EXERCISES=boot_from_volume,client-env
ROOTSLEEP=0
ACTIVE_TIMEOUT=60
BOOT_TIMEOUT=90
ASSOCIATE_TIMEOUT=60
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=token
# Temp try ML2
# Q_PLUGIN=openvswitch
Q_USE_DEBUG_COMMAND=True

Q_VPN_EXTRA_CONF_FILES=(/opt/stack/neutron/etc/neutron/plugins/cisco/cisco_vpn_agent.ini)

# RECLONE=No
# OFFLINE=True
RECLONE=yes
OFFLINE=False
```

For adapting the OpenStack setup for the CSR, the `cisco_vpn_agent.ini` will use these attributes:

- Neutron router IP 172.24.4.21
- CSR IP 172.24.4.23

Do all of the same steps, and be sure to verify pings and IPs on the CSR.

Usage

At this point, DevStack should be able to communicate with the CSR via the REST API. To check that, we'll use some basic commands.

Note: If you are behind a firewall, check that proxy is disabled for the CSR IPs with "echo \$no_proxy". If it is not disabled, disable it using this command:

```
export no_proxy=$no_proxy,192.168.200.20,192.168.200.2
```

To verify access to the CSR's REST APIs use:

```
curl -X POST https://192.168.200.20/api/v1/auth/token-services -H "Accept:application/json" -H "C
```

This should succeed and return an authorization token for the CSR. In addition, you should verify that pings work on the CSR. It should be able to ping the local VMs and the far-end CSR's public interface.

Now that we know there is REST management access to the CSR from the host, and the CSR has connectivity on the public and private interfaces, we can use the Neutron client CLI commands or Horizon to configure a site-to-site IPsec connection between the two clouds.

Note: Because the CSR is running out-of-band, we need to tell the Neutron router to redirect packets destined to the far end, to the CSR.

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

Using the command line and the Neutron client, here are the steps on devstack-32. CSR_PORT is set to the port of the CSR at 10.1.0.6.

```
source openrc admin demo

# Setup static route to CSR for VPN link
CSR_PORT=`neutron port-list | grep private_p | cut -d' ' -f 11 | cut -f 2 -d'"'"`
neutron router-update router1 --routes type=dict list=true destination=10.2.0.0/24,nextthop=$CSR_PORT

neutron vpn-ikepolicy-create ikepolicy1
neutron vpn-ipsecpolicy-create ipsecpolicy1
neutron vpn-service-create --name myvpn --description "My vpn service" router1 mysubnet

neutron ipsec-site-connection-create --name vpnconnection1 --vpnservice-id myvpn --ikepolicy-id ikepolicy1
--ipsecpolicy-id ipsecpolicy1 --peer-address 172.24.4.23 --peer-id 172.24.4.23 \
--peer-cidr 10.2.0.0/24 --psk secret
```

Here are the steps to setup the connection on devstack-33. CSR_PORT is set to 10.2.0.6:

```
source openrc admin demo

# Setup static route to CSR for VPN link
CSR_PORT=`neutron port-list | grep private_p | cut -d' ' -f 11 | cut -f 2 -d'"'"`
neutron router-update router1 --routes type=dict list=true destination=10.1.0.0/24,nextthop=$CSR_PORT

neutron vpn-ikepolicy-create ikepolicy1
neutron vpn-ipsecpolicy-create ipsecpolicy1
neutron vpn-service-create --name myvpn --description "My vpn service" router1 mysubnet

neutron ipsec-site-connection-create --name vpnconnection1 --vpnservice-id myvpn --ikepolicy-id ikepolicy1
--ipsecpolicy-id ipsecpolicy1 --peer-address 172.24.4.13 --peer-id 172.24.4.13 \
--peer-cidr 10.1.0.0/24 --psk secret
```

You should be able to ping from a VM in the devstack-32 cloud to a VM in the devstack-33 cloud. The `neutron vpn-service-status` and `ipsec-site-connection-list` commands can be used to see the status of the IPsec connection. Note: It takes time for the VPN connection to be negotiated.

CSR Restart

As mentioned earlier, Devstack builds up a router, using the private network in the localrc. Neutron will assign an IP on the private net for the router, DHCP, and some compute probe. Then when you create VMs, that allocates the next IPs. Given that we create two VMs, the next IP is 10.0.1.6. When we create the CSR, it will get that IP. It is by design, then, that we picked an IP for the CSR of 10.0.1.6, in expectation that it would be created **after** two VMs are created. Everything works out as planned.

If, however, you then stop and then restart the CSR, it will get the NEXT IP, 10.0.1.7 for the private interface and 172.24.4.14 for the public interface. Do it again, and you get 10.0.1.8/172.24.4.15, etc. In other words, if you try to restart the CSR, it will start and attach to Neutron, but Neutron will assign different IPs to the ports than what is configured in the CSR and setup in the `cisco_vpn_agent.ini` file.

This can be rectified by logging into the CSR (use the management interface at 192.168.200.2) and set the GigabitEthernet2 interface IP to the new public IP assigned by neutron, and set the GigabitEthernet3 interface IP to the new private IP address assigned by Neutron. Use "neutron port-list" to see the settings. For public IP address changes, you'll also want to change the tunnel_up address in the `cisco_vpn_agent_ini` file in `/opt/stack/neutron/etc/neutron/plugins/cisco/` area (as called out by the `Q_VPN_EXTRA_CONF_FILES` in

Install_and_Setup_of_Cisco_Cloud_Services_Router_(CSR)_for_OpenStack_VPN

the DevStack localrc file.

When the next attempt is made to create a VPN connection, the INI file will be used for the updated router information. Remember, you'll need to do these two things:

- Update the static route that redirects private net traffic to the CSR, so that it uses the new private IP.
- Change the peer_address and peer_id fields used on the far-end node, to refer to the new public IP for the CSR.

Similar to the restart case, if you didn't create two VMs before starting the CSR, the private IP created for the CSR in Neutron will not match what was configured for the interface. For example, if you did not create any VMs, then the CSR would have been assigned 10.1.0.4 as a private IP. You can use the management port and change the configuration on GigabitEthernet3 to use that IP. The static route will need to be updated as well, so that packets are redirected to the CSR.

Just remember that you can always use the "neutron port-list" command to see what IPs have been assigned to the CSR, and then make sure that the interface configurations match (and possibly the public IP in the .INI file).

Reference Information

TODO: OS link, VPN APIs, CSR pages,...