Introduction into VTI ROUTER APPLIANCE

Table of Content

Disclaimer

VTI appliance created in EIS Group and published into public domain under GPL licence.

```
#########################################################
# (c) EIS Group LTD, 2016 (scripts and setup)           #
# (License) GPL                                         #
# Donated to OpenSource by EIS Group, 2016.             #
# Authors: Alexei Roudnev , Andrey Saveliev             #
#                                                       #
#       open-source@eisgroup.com                        #
#  or   aprudnev@gmail.com                              #
#  URL; http://eisgroup.com                             #
#########################################################
```

Software is based on OpenSource Linux software and is provided ‘As Is’, no guarantees.
(This document original is stored / can be commented / here: https://docs.google.com/document/d/1E55SmAjCtrJE7UWIcjIlGdVPF3O_mBhbfDZz1y7jmJE/edit?usp=sharing)
https://github.com/eis-group-opensource/vti-router

Introduction

This appliance was created to provide IPSEC VPN services, compatible with public cloud providers (such as AwS or Azure - AWS is for Amazon Web Services cloud, and Azure is Microsoft's cloud - see Azure Specifics section in ‘Implementation’ document) and at the same time avoid high costs of dedicated hardware (the only compatible with these clouds and their VPN-s is relatively new hardware) and have enough flexibility.

Problem is common:

- Your company decided to utilize AWS for some project, and it requires site-to-site VPN with your network, and possible with your client’s network as well.
- AWS supports very flexible, new IPSEC model which is called VTI - Virtual Tunnel Interface - when it is IPSEC in network, but looks as normal interface inside the router or firewall; it allows normal routing (via BGP or static or other protocols), simple access lists and so on;
- Most companies got 2 problems with this:
    - Their firewalls can be incompatible, as most firewalls do not support VTI at all.
    - Even if they have compatible hardware, it usually is designed to provide INTRANET access or to provide policy based IPSEC, and you do not want to bring AWS network (AWS call it VPC - Virtual Private Cloud) into your INTRANET.
- Company need a method to integrate some of DMZ zones, dedicated for the project, with AWS VPC, created in Amazon.
- In some cases, we want to interconnect some other networks, and it can not be done on existing hardware.

Our solution is simple:

- We created Virtual Appliance - virtual system, based on CentOS7 Linux, with updated kernel, specially installed and well tested software, and a lot of management scripts.
- This appliance can be installed, as a Virtual Server on Vmware (or onto any centos linux), into any sandbox DMZ, integrated with internal routing, and establish VPN connections with Amazon or other such appliance;
- It is free, so two appliances can be installed if required for redundancy;
- Appliance has tools to generate VPN for AWS automatically from generic configuration file, provided by Amazon; or manually, by manually entering VPN data.

Appliance supports:

- IPSEC in VTI mode, compatible with AWS (and possible, with Azure);
- Interruption-less adding/deleting VTI tunnels;
- BGP routing for internal and external connections (like AWS); OSPF routing; static routing;

- SNMP monitoring; Zabbix and other monitors can be added as required.
- Optional support (not included) for VRRP and Zabbix agent.

Appliance parts are:

- Standard CentOS7.2 (or higher)
- Kernel 4 (as default kernel has a serious bugs)
- Strongswan for IPSEC and VTI implementation
- Quagga for the routing implementation
- IPTABLES for firewall protection
- SNMPD for snmp monitoring
- Scripts, which can be split into 3 groups”
    - SETUP scripts to setup or reset system (they allow to set up VTI router on plain CentOS7 as well, not just use this appliance)
    - Parsing and tunnel creation scripts to create tunnels out fo AWS data or manually
    - Management scripts to enable,disable, add, delete, check status, and make healthcheck, on the tunnels.
- This can be used as pre-configured appliance, OR can be used to add all these features to existing CentOS7 Linux (not tested).

Appliance requirements are:

- Direct connection to OUTSIDE network (can be replaced by properly configured NAT translation, but it was never tested),
- Connection to DMZ or other network you want to integrate with AWS networks;
- VMware ESXi 5 or higher (if you import appliance) or compatible; it can work on other VM systems but ovf file _may_ need to be adjusted; OR Linux CentOS 7 with packets installed as mentioned in scripts, access to DMZ and to OUTSIDE network, 1 -2 GB of RAM, 2 or more CPU (HW solution was not tested, but we do not see any reason, why it should not work).

Appliance performance - it provides about 400Mbit IPSEC bandwidth when running as a Virtual Server on average speed VMWare ESXi host; performance can be much higher on faster hosts.

Appliance Use Case

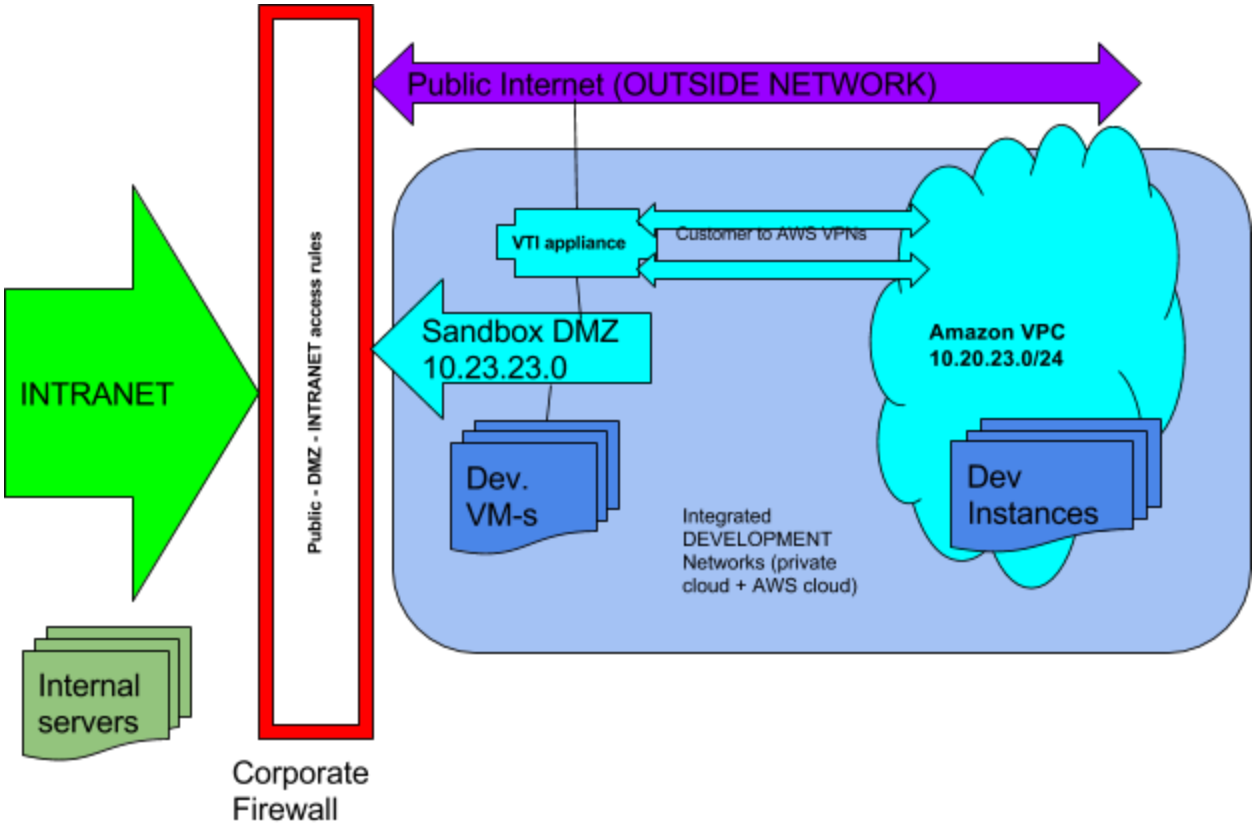
This is common use case (but it is not the only one). We have AWS project, and must integrate AWS VPC (Virtual Private Network) into our network and provide a flexible way to make development in our own network and on AWS.

1) Create SandBOX DMZ for the project; set up firewall rules. For example, 10.23.23.0/24
2) Create AWS VPC and network; for example, 10.20.23.0/24
3) Install 1 or 2 VTI appliances into our SandBOX DMZ, with 1 interface in this DMZ and 1 interface in OUTSIDE network (maybe it can be done by reusing interface and

configuring pass thru for IPSEC on firewall, but it decrease performance and was not tested). Integrate them with internal routing, if required. Assign INSIDE and OUTSIDE IP and BGP AS number (can be private). Default routing is via INSIDE (which in turn connected to SandBOX DMZ) so systems can not be scanned or seen from OUTSIDE.

4) Add these VTI appliances into AWS as ‘Customer Gateways’ and create VPN-s with them (each VPN has 1 customer gateway and 2 IP for AWS gateways);
5) Download VPN descriptions from AWS, copy them into appliances, and parse using scripts; parsing creates VTI interfaces descriptions (2 for each AWS VPN);
6) Add these VTI interfaces using scripts; enable them using scripts; check status using scripts;
7) Set up health check script to run every 10 minutes (it helps to re-establish if something went wrong);
8) Check routing, adjust quagga routing as required;
9) Enjoy.
10) If necessary disable VTI tunnel(s) or remove it.
11) It can be repeated many times, as number of VTI interfaces is not limited by the scripts.

Here is an architecture of integrated (Private + Cloud) network
Picture 1.



Appliance usage in example.

Let's show, how to setup VPN using our VTI appliance, for such environment.
(Notice: test was done with real IP and VPN, which was, then, disabled for security purposes).

I set up, OUTSIDE of this document:

- Extra DMZ23 on our firewalls, which is 10.23.23.0/24, with access 'Intranet -> DMZ23 OK, DMZ23 -> OUTSIDE OK on http and https, DNS and NTP allowed. Few test VM's exists in this network. Access from this network to INTRANET is not allowed, so I can allow 3d party users to access it, if it became essential (or we can set up rules, proxies and so on).
- We made OUTSIDE network available inside our cloud (but with strict access rules, so only network admin can access it. In some cases it may require extra interface on hosts, in others it is just dedicated VLAN shared on the same L2 switches (but without any L3 interfaces).
- AWS VPC, with network 10.20.23.0/24, and one test instance running in it. Propagation of routes is allowed.

- Now I prepare our test, to do it:

Router initial setup.

- I clone router (I can import it from downloads repository, using ‘Vmware import’ feature) and name it dev2-lab23-gw02 .
- I assign 2 static IP to this router, one in DMZ23 and one in OUTSIDE network. IT is important that virtual appliance can reach outside network. One option is to allow OUTSIDE network inside the cloud. It is safe, in my case, as no any IP exists on this network, except IP of our router, and router has devault inside DMZ23 network and firewall providing extra protection of outside network. DMZ23 network is connected to eth0 and OUTSIDE network is connected to eth1. IP / names assigned are: (it is just an example)
  dev2-lab23-gw02 - 10.23.23.5
  dev2-lab23-gw02ext - 63.2xx.2xx.38
  AS number - we assign AS 65003, which is private AS (so can be used in private connections only).
- When done, we start router, login as a root, (change root password as appliance comes with known root), and then run initial configuration script:
  - Enter host name. If you repeat script, it will remember settings you enter last time for the same host, and propose to reuse them. It allows to repeat few initializations, when you debug initialization.
  - Enter domain name.
  - Enter IP address for internal network (eth0), netmask and gateway. (Notice - THIS gateway will be used for default traffic, such as update requests from router itself);
  - Enter list of dns servers
  - Enter external IP, external netmask, external gateway (Notice: this gateway will be used ONLY for the traffic to remote VPN access points);
  - REINSTALL - if yes, packets (such as strongswan, quagga and so on) will be reinstalled. Do not select it until you surem that router has access to CentOS and epel repositories. This feature should be used, when you setup router on your own CentOS7 , not in downloaded appliance.
  - RECONFIGURE - if selected, all IPSEC and routing configurations will be reset to initial (almost empty) values.

  See Screen1 and Screen2 screenshots in attachment.
- Now, restart router, and check all access (ping both default gateways, DNS setting, access to CentOS7 yum repositories, system time and time zone, /etc/ntp.conf). Now, once static IP is assigned, I can login via SSH, configure public key, disable password login on sshd, and do other enhansements, which are outside of our scope. What is important, they should be done in THIS time.

- If necessary, configure OSPF and BGP routing between your router and the rest of your network. We configured it so that we
  - restrict announces from AWS by route-map and access-list
  - Redistribute them to OSPF which in turn peered with firewalls
  - I used IBGP to redistribute routes to internal network, but it is not common case; more commonly you will use ospf on firewalls, or static routes.
  - Remember, quagga is not cisco, even if it looks as a cisco. You can manage it by
    vtysh
    sh run
    conf t
    <config update>
    ^D
    wr mem
    ^D
    But be careful, as it is not cisco and is not so well debugged and bug-free. For example, when you add route-map and want to use it in bgp, make sure that bgp is started and running, before adding, else it _may_ be added to other routers only. So always double check the results.
  - I will skip this step and show only interaction with AWS. There is example of configuration in /etc/quagga/quagga-config.SAMPLE directory. Here is my test configuration:

```
dev2-lab23-gw02.sjclab.exigengroup.com# sh run
Building configuration...

Current configuration:
!
!
interface eth0
 ipv6 nd suppress-ra
!
interface eth1
 ipv6 nd suppress-ra
!
interface lo
!
router bgp 65003
  network 10.23.23.0/24
!
router ospf
 redistribute bgp route-map from_aws
 network 10.23.23.0/24 area 0.0.0.0
!
access-list from_aws remark Announces we can accept from AWS and redistribute
access-list from_aws permit 10.23.24.0/24
!
route-map from_aws permit 10
 match ip address from_aws
```

```
!
ip forwarding
!
line vty
!
end
wr mem
^D
<ping internal gateway here>
<ping external gateway here>
<check time and time zone>
<check syslog settings>
```

- Now we are ready for the next step - describe our gateway on AWS as ‘Customer gateway’, create VPN and download VPN description file for the parsing.

VPN Configuration.

Create VPN on AWS and Download.

First, you must describe your VTI router as ‘Customer gateway’ in AWS. It is very simple, as not too much information required - you enter Type, external IP address, BGP ASN (if using BGP).

ID:	cgw-54c8114a (63.2xx.2xx.xx) | dev2-lab23-gw02
State:	available
Type:	ipsec.1
IP address:	63.2xx.2xx.xx
BGP ASN:	65003
VPC:	vpc-2fdeae4b (10.20.23.0/24) | AWS0-LAB23-VPC

Second, you create VPN between VPC and thus ‘Customer Gateway’. It looks like this:

VPN ID:	vpn-05e0fe17 | AWS23_dev2lab23gw02
State:	available
Virtual Private Gateway:	vgw-4de33a53 | AWS0-LAB23-GW
Customer Gateway:	cgw-54c8114a (63.2xx.2xx.xx) | dev2-lab23-gw02
Type:	ipsec.1
VPC:	vpc-2fdeae4b (10.20.23.0/24) | AWS0-LAB23-VPC
Routing:	Dynamic

And now, you go and download VPN description in ‘Generic’ format. Copy this file into the router into ~/scripts/WORK/ and name it reasonably. It looks like this:
(Notice: ~/scripts is symlink to /usr/local/scripts):

```
cd ~/scripts/WORK
cat aws23_dev2lab23gw02.txt
(See Screen-3 in attachment for an example, with obfuscated PSK keys).
```

Parse VPN description file and apply (add and enable)

Now, we are ready for the final stages. You must create vti interfaces and enable them.
Interfaces have numbers, 1 - 99, and can be in one of 3 states, from VTI system point of view:

- Available - property file created (see files layout)
- Disabled (vti interface created in strongswan but disabled by moving files into .DISABLED subdirectory, so inactive);
- Enabled (vti interface is active and try to establish packets).

First, verify that you do not have any vti created, as they all must be unique. Cd into ~/scripts and run

```
[root@dev2-lab23-gw02 scripts]# VTI -list
Active: *
Disabled: *
Available: *
```

So, we do not have any interfaces available yet. Available means that property file exists in **`/usr/local/scripts/WORK/<router-name>.d/vtiNN.properties`** file.

There are 3 ways to create such file.

- Manually, by test editor.
- Manually, by VTI_ADD.sh script (and answering the questions). The benefit is that it can create files for 2 routers on both sides, if you configure VTI-APPLIANCE to VTI-APPLIANCE tunnel.
- By automatic parsing of AWS description file. To do it, select vti numbers (in our case, vti1 and vti2; do not use names with zeroes like vti01, they cause system confusion) and run VTI_ADDAWS.sh script:

```
./VTI_ADDAWS.sh WORK/aws23_dev2lab23gw02.txt vti1 vti2
Using directory WORK/dev2-lab23-gw02.d - you can change it by -d <dir> option
grep: WORK/dev2-lab23-gw02.d/*.properties: No such file or directory
grep: WORK/dev2-lab23-gw02.d/*.properties: No such file or directory
Created WORK/dev2-lab23-gw02.d/vti1.properties
Created WORK/dev2-lab23-gw02.d/vti2.properties
[root@dev2-lab23-gw02 scripts]# VTI -list
Active: *
Disabled: *
Available: vti1 vti2
```

- As you can see, this command created property files for vti1 and vti2 virtual interfaces, so they became ‘Available’ for the system. Files are in WORK/<hostname>.d directory, we can inspect them and/or edit if necessary:

```
cat WORK/`hostname -s`.d/vti2.properties
### Generated Thu Oct 27 20:29:20 PDT 2016
VTI="vti2"
VTI_MTU="1420"
VTI_OIP_LOCAL="63.2xx.2xx.38"
VTI_OIP_REMOTE="52.39.14.106"
LOCAL_GW="63.2xx.2xx.1"
VTI_IIP_LOCAL="169.254.13.110/30"
VTI_IIP_REMOTE="169.254.13.109/30"
VTI_PSK="xxxx0kFNtuW0QdRD_ksToB1byxxxxxo."
VTI_MARK="24"
#### OPTIONAL
ID="aws23_dev2lab23gw02_GW2"
AWS_ID="vpn-05e0fe17"
VTI_BGP_LOCAL_AS="65003"
VTI_BGP_REMOTE_AS="7224"
VTI_BGP_LOCAL_IP="169.254.13.110/30"
VTI_BGP_REMOTE_IP="169.254.13.109"
```

- Next step is to add vti into ipsec system by running VTI -add:
  `VTI -add vti1 vti2`
  (See an example of output in attachment in Screen 4).
- It created vti1 and vti2 as disabled and created all required routing and bgp. You may wish to adjust bgp and routing, for example by adding inbound filter:

```
[root@dev2-lab23-gw02 scripts]# VTI -list
Active: *
Disabled: vti1 vti2
Available: vti1 vti2
[root@dev2-lab23-gw02 scripts]# vtysh

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

dev2-lab23-gw02.sjclab.exigengroup.com# sh run
Building configuration...

Current configuration:
!
!
interface eth0
 ipv6 nd suppress-ra
!
interface eth1
 ipv6 nd suppress-ra
!
interface lo
!
router bgp 65003
 bgp router-id 63.2xx.2xx.38
 network 10.23.23.0/24
 neighbor 169.254.13.21 remote-as 7224
 neighbor 169.254.13.109 remote-as 7224
!
router ospf
```

```
 redistribute bgp route-map from_aws
 network 10.23.23.0/24 area 0.0.0.0
!
ip route 52.25.31.90/32 63.2xx.2xx.1
ip route 52.39.14.106/32 63.2xx.2xx.1
!
access-list from_aws remark Announces we can accept from AWS and redistribute
access-list from_aws permit 10.20.23.0/24
!
route-map from_aws permit 10
 match ip address from_aws
...
dev2-lab23-gw02.sjclab.exigengroup.com# conf t
dev2-lab23-gw02.sjclab.exigengroup.com(config)# router bgp 65003
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# neighbor 169.254.13.21
route-map from_aws in
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# neighbor 169.254.13.109
route-map from_aws in
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# ^Z
dev2-lab23-gw02.sjclab.exigengroup.com# wr mem
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
Configuration saved to /etc/quagga/bgpd.conf
[OK]
dev2-lab23-gw02.sjclab.exigengroup.com# exit
```

- As you can see, it created static routing for remote (AWS) gateways, added BGP records for the peers (we added filter manually) and actually created vti description files and scripts, but did not add VTI themselves into strongswan configuration, so tunnels are still inactive. You can see it by running VTI -status.
- Next, activate VTI tunnels - run VTI -enable vti1 vti2

```
[root@dev2-lab23-gw02 scripts]# VTI -enable vti1 vti2
vti1 enabled
Reloading strongSwan IPsec configuration...
generating QUICK_MODE request 981536986 [ HASH SA No KE ID ID ]
sending packet: from 63.2xx.2xx.38[4500] to 52.25.31.90[4500] (460 bytes)
received packet: from 52.25.31.90[4500] to 63.2xx.2xx.38[4500] (444 bytes)
parsed QUICK_MODE response 981536986 [ HASH SA No KE ID ID ]
CHILD_SA vti1.aws23_dev2lab23gw02_GW1{2} established with SPIs cbf7ef1e_i 356d83c6_o
and TS 0.0.0.0/0 === 0.0.0.0/0
connection 'vti1.aws23_dev2lab23gw02_GW1' established successfully
vti2 enabled
Reloading strongSwan IPsec configuration...
generating QUICK_MODE request 1476949665 [ HASH SA No KE ID ID ]
sending packet: from 63.2xx.2xx.38[4500] to 52.39.14.106[4500] (460 bytes)
received packet: from 52.39.14.106[4500] to 63.2xx.2xx.38[4500] (444 bytes)
parsed QUICK_MODE response 1476949665 [ HASH SA No KE ID ID ]
CHILD_SA vti2.aws23_dev2lab23gw02_GW2{5} established with SPIs c9ab231b_i a63b9ad8_o
and TS 0.0.0.0/0 === 0.0.0.0/0
connection 'vti2.aws23_dev2lab23gw02_GW2' established successfully
Reloading strongSwan IPsec configuration...
```

- Now, check the status. VTI -status wil show long output; you can run

```
VTI -status | grep vti
```

```
vti2.aws23_dev2lab23gw02_GW2: #2, ESTABLISHED, IKEv1,
e57a6f663d51aa9c:bf8e9bbd4c228ebb
  vti2.aws23_dev2lab23gw02_GW2: #4, reqid 2, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
  vti2.aws23_dev2lab23gw02_GW2: #5, reqid 2, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
  vti2.aws23_dev2lab23gw02_GW2: #7, reqid 2, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
vti1.aws23_dev2lab23gw02_GW1: #1, ESTABLISHED, IKEv1,
b69a7eadbe4a6639:dc27788ab4f37f72
  vti1.aws23_dev2lab23gw02_GW1: #1, reqid 1, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
  vti1.aws23_dev2lab23gw02_GW1: #2, reqid 1, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
  vti1.aws23_dev2lab23gw02_GW1: #3, reqid 1, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
  vti1.aws23_dev2lab23gw02_GW1: #6, reqid 1, INSTALLED, TUNNEL-in-UDP,
ESP:AES_CBC-128/HMAC_SHA2_256_128/MODP_2048_256
vti1: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
vti2: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
169.254.13.20   0.0.0.0       255.255.255.252 U             0 0            0 vti1
169.254.13.108  0.0.0.0       255.255.255.252 U             0 0            0 vti2
```

- Notice that both tunnels are established, tunnel type is TUNNEL-in-UDP (as AWS uses IPSEC in NAT mode which is UDP tunnel), and both VTI interfaces are up and got IP addresses. Wait about 5 minutes, and you should see BGP connections established and routing coming from AWS:

```
vti1: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
vti2: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
10.20.23.0     169.254.13.21   255.255.255.0   UG            0 0            0 vti1
169.254.13.20   0.0.0.0       255.255.255.252 U             0 0            0 vti1
169.254.13.108  0.0.0.0       255.255.255.252 U             0 0            0 vti2

tcp     0      0 0.0.0.0:179                 0.0.0.0:*               LISTEN
tcp     0      0 169.254.13.22:48432         169.254.13.21:179       ESTABLISHED
tcp     0      0 169.254.13.110:42948        169.254.13.109:179      ESTABLISHED
tcp6    0      0 :::179                      :::*                    LISTEN
```

- You can add other vti interfaces, disable / enable / remove existing ones (see VTI -help).
- Now, check that health-monitor works properly. It is running every 10 minutes and helps to re-establish IPSEC connections if they fail somehow:

```
cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
```

```
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
*/10 * * * * root /etc/strongswan/restart_vti.sh > /tmp/restart_vti.log 2>& 1
[root@dev2-lab23-gw02 scripts]# /etc/strongswan/restart_vti.sh
vti1.aws23_dev2lab23gw02_GW1 OK
vti2.aws23_dev2lab23gw02_GW2 OK
```

- Restart router; make sure that everything comes up after the restart (BGP re-connection takes up to a few minutes, so be patient). Add snmp and zabbix monitoring, as necessary. Test performance. Enjoy.

Notice about your firewall and traffic between this DMZ and AWS.

If you have servers in the same DMZ, where VTI router(s) are configured, you _MAY_ experience unsymmetrical firewall routing, when traffic is between the server in this DMZ and AWS. The reason is that, traffic from the server TO AWS go usually to the firewall first, which in turn must forward it to VTI router, Traffic from AWS go directly. It may cause a few issues. Solutions are

- Separate servers and VTI into 2 different DMZ (recommended!)
- Set up default gateway on VTI routers and use OSPF to route traffic between them and INTRANET (you may want to use VRRP for redundancy)
- Allow firewall to pass non-symmetrical traffic, see this article: http://packetflow.io/2014/03/asa-hairpinning-and-tcp-state-bypass.html

Here is configuration example (for ASA 8.2.5), it is not full as few other places are changed as well. Other firewalls may need different setting.

```
class-map dmz23_bypass
 match access-list dmz23_bypass
policy-map dmz23_bypass_policy
 class dmz23_bypass
  set connection advanced-options tcp-state-bypass
!
service-policy dmz23_bypass_policy interface dmz23
```

ATTACHMENT. Screenshots.

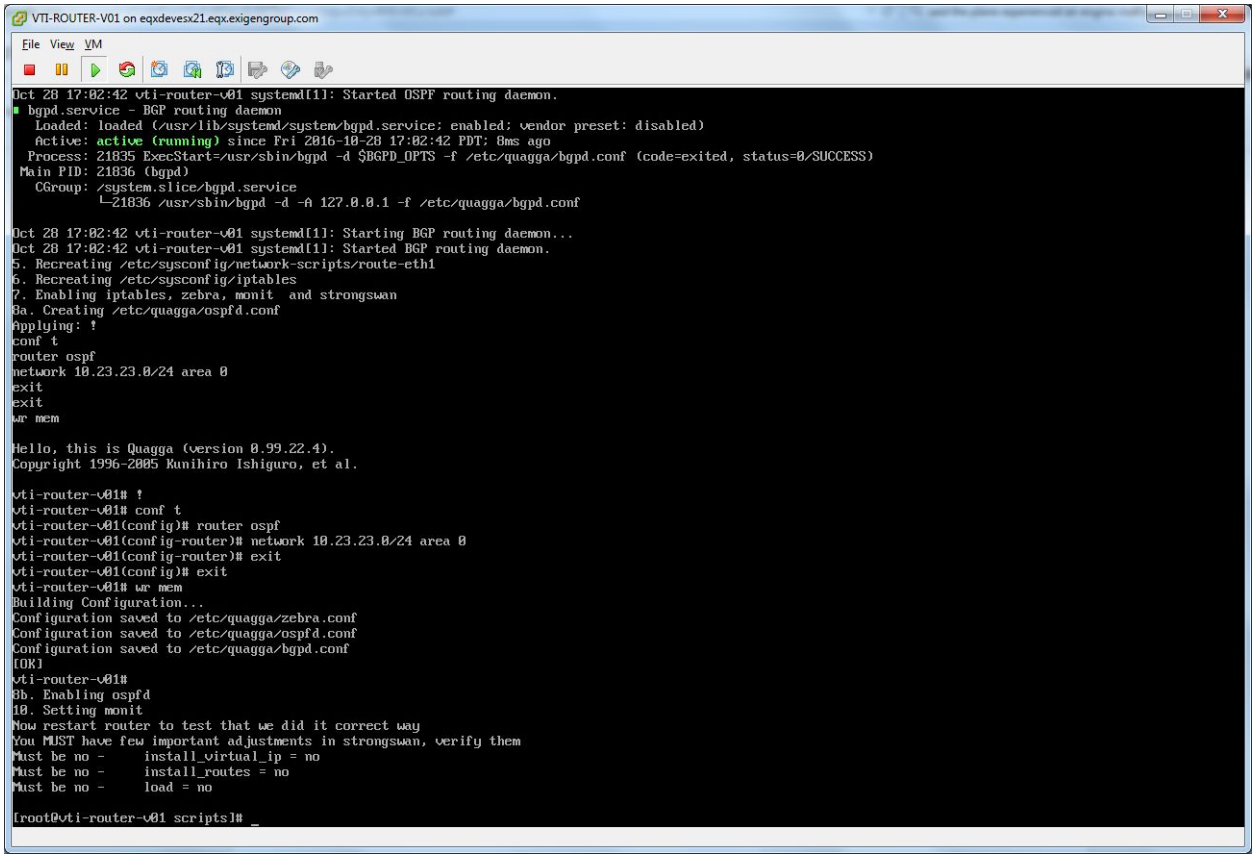
Screen 1. Initial setup.

```
[root@vti-router-v01 scripts]# ./INITIAL_SETUP.sh
Enter short hostname
host= [vti-router-v01]_dev2-lab23-gw02
Enter short hostname
host= [dev2-lab23-gw02]_
Enter domain name
domain= [net.exigengroup.com]_sjclab.exigengroup.com
Enter my INSIDE IP address
IP= []_10.23.23.5
Enter netmask for 10.23.23.5
NETMASK= [255.255.255.0]_255.255.255.0
Enter gateway for 10.23.23.5
GATEWAY= [10.23.23.1]_
Enter list of DNS servers (IP, space delimited)
dnsservers= [10.25.168.31 10.23.5.31 192.168.8.24 ]_10.21.5.31 10.23.5.31
Enter OUTSIDE IP address
EXT_IP= []_63.2NN.2NN.38
Enter OUTSIDE IP netmask
EXT_NETMASK= [255.255.255.0]_
Enter OUTSIDE IP Gateway
EXT_GW= [63.2NN.2NN.1]_
Enter LOCAL AS number, no BGP if empty
LOCAL_AS= []_65003
Enter list of OSPF networks in N.N.N.N/prefix format delimited by spaces
OSPF_NETWORKS= []_10.23.23.0/24
SNMP v1/v2 communities (no SNMP if empty)
SNMP_COMM= []_TestSnmp
Reinstall or install software and system variables?
(require INTERNET connection)
REINSTALL= [n]_
Reset IPSEC and quagga configuration files?
RESET= [y]_

*** VERIFY:
HOST dev2-lab23-gw02.sjclab.exigengroup.com
IP: 10.23.23.5
NETMASK: 255.255.255.0
GATEWAY: 10.23.23.1
EXT_IP:  63.2NN.2NN.38
EXT_NETMASK: 255.255.255.0
EXT_GW:  63.2NN.2NN.1
LOCAL_AS: 65003
OSPF_NETWORKS: 10.23.23.0/24
RESET: y
REINSTALL: n
SNMP_COMM: TestSnmp

^C to abort, ENTER to proceed_
```

Screen 2. Initial setup execution log (obfuscated):



Screen 3. Example of AWS VPN configuration.

#VPN description file example:

```
 cd ~/scripts/WORK ; cat aws23_dev2lab23gw02.txt
Amazon Web Services
Virtual Private Cloud

VPN Connection Configuration
================================================================================
AWS utilizes unique identifiers to manipulate the configuration of
a VPN Connection. Each VPN Connection is assigned a VPN Connection Identifier
and is associated with two other identifiers, namely the
Customer Gateway Identifier and the Virtual Private Gateway Identifier.

Your VPN Connection ID                : vpn-05e0fe17
Your Virtual Private Gateway ID       : vgw-4de33a53
Your Customer Gateway ID              : cgw-54c8114a

A VPN Connection consists of a pair of IPSec tunnel security associations (SAs).
It is important that both tunnel security associations be configured.


IPSec Tunnel #1
================================================================================
#1: Internet Key Exchange Configuration

Configure the IKE SA as follows:
Please note, these sample configurations are for the minimum requirement of AES128, SHA1, and
DH Group 2.
You will need to modify these sample configuration files to take advantage of AES256, SHA256,
or other DH groups like 2, 14-18, 22, 23, and 24.
The address of the external interface for your customer gateway must be a static address.
Your customer gateway may reside behind a device performing network address translation (NAT).
To ensure that NAT traversal (NAT-T) can function, you must adjust your firewall !rules to
unblock UDP port 4500. If not behind NAT, we recommend disabling NAT-T.
  - Authentication Method     : Pre-Shared Key
  - Pre-Shared Key            : xxxx1DEJB2wtXzc4jYRGvADIhVLxxxxx
  - Authentication Algorithm : sha1
  - Encryption Algorithm      : aes-128-cbc
  - Lifetime                  : 28800 seconds
  - Phase 1 Negotiation Mode : main
  - Perfect Forward Secrecy  : Diffie-Hellman Group 2

#2: IPSec Configuration

Configure the IPSec SA as follows:
Please note, you may use these additionally supported IPSec parameters for encryption like
AES256 and other DH groups like 1,2, 5, 14-18, 22, 23, and 24.
  - Protocol                  : esp
  - Authentication Algorithm : hmac-sha1-96
  - Encryption Algorithm      : aes-128-cbc
  - Lifetime                  : 3600 seconds
  - Mode                      : tunnel
  - Perfect Forward Secrecy  : Diffie-Hellman Group 2
```

```
IPSec Dead Peer Detection (DPD) will be enabled on the AWS Endpoint. We
recommend configuring DPD on your endpoint as follows:
  - DPD Interval              : 10
  - DPD Retries               : 3

IPSec ESP (Encapsulating Security Payload) inserts additional
headers to transmit packets. These headers require additional space,
which reduces the amount of space available to transmit application data.
To limit the impact of this behavior, we recommend the following
configuration on your Customer Gateway:
  - TCP MSS Adjustment        : 1387 bytes
  - Clear Don't Fragment Bit : enabled
  - Fragmentation             : Before encryption

#3: Tunnel Interface Configuration

Your Customer Gateway must be configured with a tunnel interface that is
associated with the IPSec tunnel. All traffic transmitted to the tunnel
interface is encrypted and transmitted to the Virtual Private Gateway.



The Customer Gateway and Virtual Private Gateway each have two addresses that relate
to this IPSec tunnel. Each contains an outside address, upon which encrypted
traffic is exchanged. Each also contain an inside address associated with
the tunnel interface.

The Customer Gateway outside IP address was provided when the Customer Gateway
was created. Changing the IP address requires the creation of a new
Customer Gateway.

The Customer Gateway inside IP address should be configured on your tunnel
interface.

Outside IP Addresses:
  - Customer Gateway                : 63.2xx.2xx.38
  - Virtual Private Gateway          : 52.25.31.90

Inside IP Addresses
  - Customer Gateway                : 169.254.13.22/30
  - Virtual Private Gateway          : 169.254.13.21/30

Configure your tunnel to fragment at the optimal size:
  - Tunnel interface MTU      : 1436 bytes

#4: Border Gateway Protocol (BGP) Configuration:

The Border Gateway Protocol (BGPv4) is used within the tunnel, between the inside
IP addresses, to exchange routes from the VPC to your home network. Each
BGP router has an Autonomous System Number (ASN). Your ASN was provided
to AWS when the Customer Gateway was created.

BGP Configuration Options:
  - Customer Gateway ASN                    : 65003
  - Virtual Private  Gateway ASN            : 7224
  - Neighbor IP Address                     : 169.254.13.21
```

```
  - Neighbor Hold Time        : 30

Configure BGP to announce routes to the Virtual Private Gateway. The gateway
will announce prefixes to your customer gateway based upon the prefix you
assigned to the VPC at creation time.


IPSec Tunnel #2
================================================================================
#1: Internet Key Exchange Configuration

Configure the IKE SA as follows:
Please note, these sample configurations are for the minimum requirement of AES128, SHA1, and
DH Group 2.
You will need to modify these sample configuration files to take advantage of AES256, SHA256,
or other DH groups like 2, 14-18, 22, 23, and 24.
The address of the external interface for your customer gateway must be a static address.
Your customer gateway may reside behind a device performing network address translation (NAT).
To ensure that NAT traversal (NAT-T) can function, you must adjust your firewall !rules to
unblock UDP port 4500. If not behind NAT, we recommend disabling NAT-T.
  - Authentication Method     : Pre-Shared Key
  - Pre-Shared Key            : xxxxNG0kFNtuW0QdRD_ksToB1byJxxxx.
  - Authentication Algorithm : sha1
  - Encryption Algorithm      : aes-128-cbc
  - Lifetime                  : 28800 seconds
  - Phase 1 Negotiation Mode : main
  - Perfect Forward Secrecy  : Diffie-Hellman Group 2

#2: IPSec Configuration

Configure the IPSec SA as follows:
Please note, you may use these additionally supported IPSec parameters for encryption like
AES256 and other DH groups like 1,2, 5, 14-18, 22, 23, and 24.
  - Protocol                  : esp
  - Authentication Algorithm : hmac-sha1-96
  - Encryption Algorithm      : aes-128-cbc
  - Lifetime                  : 3600 seconds
  - Mode                      : tunnel
  - Perfect Forward Secrecy  : Diffie-Hellman Group 2

IPSec Dead Peer Detection (DPD) will be enabled on the AWS Endpoint. We
recommend configuring DPD on your endpoint as follows:
  - DPD Interval              : 10
  - DPD Retries               : 3

IPSec ESP (Encapsulating Security Payload) inserts additional
headers to transmit packets. These headers require additional space,
which reduces the amount of space available to transmit application data.
To limit the impact of this behavior, we recommend the following
configuration on your Customer Gateway:
  - TCP MSS Adjustment        : 1387 bytes
  - Clear Don't Fragment Bit : enabled
  - Fragmentation             : Before encryption

#3: Tunnel Interface Configuration
```

```
Your Customer Gateway must be configured with a tunnel interface that is
associated with the IPSec tunnel. All traffic transmitted to the tunnel
interface is encrypted and transmitted to the Virtual Private Gateway.



The Customer Gateway and Virtual Private Gateway each have two addresses that relate
to this IPSec tunnel. Each contains an outside address, upon which encrypted
traffic is exchanged. Each also contain an inside address associated with
the tunnel interface.

The Customer Gateway outside IP address was provided when the Customer Gateway
was created. Changing the IP address requires the creation of a new
Customer Gateway.

The Customer Gateway inside IP address should be configured on your tunnel
interface.

Outside IP Addresses:
  - Customer Gateway                : 63.2xx.2xx.38
  - Virtual Private Gateway          : 52.39.14.106

Inside IP Addresses
  - Customer Gateway                : 169.254.13.110/30
  - Virtual Private Gateway          : 169.254.13.109/30

Configure your tunnel to fragment at the optimal size:
  - Tunnel interface MTU      : 1436 bytes

#4: Border Gateway Protocol (BGP) Configuration:

The Border Gateway Protocol (BGPv4) is used within the tunnel, between the inside
IP addresses, to exchange routes from the VPC to your home network. Each
BGP router has an Autonomous System Number (ASN). Your ASN was provided
to AWS when the Customer Gateway was created.

BGP Configuration Options:
  - Customer Gateway ASN                    : 65003
  - Virtual Private  Gateway ASN             : 7224
  - Neighbor IP Address                     : 169.254.13.109
  - Neighbor Hold Time        : 30

Configure BGP to announce routes to the Virtual Private Gateway. The gateway
will announce prefixes to your customer gateway based upon the prefix you
assigned to the VPC at creation time.



Additional Notes and Questions
================================================================================

  - Amazon Virtual Private Cloud Getting Started Guide:
        http://docs.amazonwebservices.com/AmazonVPC/latest/GettingStartedGuide
  - Amazon Virtual Private Cloud Network Administrator Guide:
        http://docs.amazonwebservices.com/AmazonVPC/latest/NetworkAdminGuide
  - XSL Version: 2009-07-15-1119716
```

Screen 4 - execution if **VTI -add vti1 vti2** command.

```
1. Creating /etc/strongswan/vti.d/vti1.init, vti1.secret, vti1.conf
quagga: Adding ip route 52.25.31.90/32 63.2xx.2xx.1

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

dev2-lab23-gw02.sjclab.exigengroup.com# conf t
dev2-lab23-gw02.sjclab.exigengroup.com(config)# ip route 52.25.31.90/32 63.2xx.2xx.1
dev2-lab23-gw02.sjclab.exigengroup.com(config)# exit
dev2-lab23-gw02.sjclab.exigengroup.com# write mem
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
Configuration saved to /etc/quagga/bgpd.conf
[OK]
dev2-lab23-gw02.sjclab.exigengroup.com#
quagga: Adding neighbor 169.254.13.21 remote-as 7224

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

dev2-lab23-gw02.sjclab.exigengroup.com# conf t
dev2-lab23-gw02.sjclab.exigengroup.com(config)# router bgp 65003
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# neighbor 169.254.13.21
remote-as 7224
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# neighbor 169.254.13.21
local-as 65003
% Cannot have local-as same as BGP AS number
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# exit
dev2-lab23-gw02.sjclab.exigengroup.com(config)# exit
dev2-lab23-gw02.sjclab.exigengroup.com# write mem
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
Configuration saved to /etc/quagga/bgpd.conf
[OK]
dev2-lab23-gw02.sjclab.exigengroup.com#
***INFO: vti1 created as DISABLED, run /usr/local/scripts/VTI_OPS.sh -enable vti1 to
activate
***INFO: You may wish to edit /etc/strongswan/vti.d/vti1.init and
/etc/strongswan/vti.d/.DISABLED/vti1.conf first
1. Creating /etc/strongswan/vti.d/vti2.init, vti2.secret, vti2.conf
quagga: Adding ip route 52.39.14.106/32 63.2xx.2xx.1

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

dev2-lab23-gw02.sjclab.exigengroup.com# conf t
dev2-lab23-gw02.sjclab.exigengroup.com(config)# ip route 52.39.14.106/32
63.2xx.2xx.1
dev2-lab23-gw02.sjclab.exigengroup.com(config)# exit
dev2-lab23-gw02.sjclab.exigengroup.com# write mem
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
Configuration saved to /etc/quagga/bgpd.conf
[OK]
dev2-lab23-gw02.sjclab.exigengroup.com#
quagga: Adding neighbor 169.254.13.109 remote-as 7224
```

```
Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

dev2-lab23-gw02.sjclab.exigengroup.com# conf t
dev2-lab23-gw02.sjclab.exigengroup.com(config)# router bgp 65003
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# neighbor 169.254.13.109
remote-as 7224
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# neighbor 169.254.13.109
local-as 65003
% Cannot have local-as same as BGP AS number
dev2-lab23-gw02.sjclab.exigengroup.com(config-router)# exit
dev2-lab23-gw02.sjclab.exigengroup.com(config)# exit
dev2-lab23-gw02.sjclab.exigengroup.com# write mem
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
Configuration saved to /etc/quagga/bgpd.conf
[OK]
dev2-lab23-gw02.sjclab.exigengroup.com#
***INFO: vti2 created as DISABLED, run /usr/local/scripts/VTI_OPS.sh -enable vti2 to
activate
***INFO: You may wish to edit /etc/strongswan/vti.d/vti2.init and
/etc/strongswan/vti.d/.DISABLED/vti2.conf first
No changes
[root@dev2-lab23-gw02 scripts]#
```