

# GÈANT Global Testbed Service Engineering, Installation, and Configuration Guide

---

## Outline

|   |    |
|---|----|
| Abstract.....   | 4  |
| 1 Architecture overview.....                                  | 4  |
| 1.1 CSF and POD internal devices .....                        | 5  |
| 1.2 GÉANT GTS WAN connectivity .....                          | 7  |
| 2 Function parts and their configuration .....                | 8  |
| 2.1 Virtual machines provisioning.....                        | 8  |
| 2.2 Networking.....   | 8  |
| 2.2.1 Control plane network .....                             | 9  |
| 2.2.2 Data plane network.....                                 | 11 |
| 2.2.3 Public network .....                                    | 11 |
| 3 GTS components installation and configuration .....         | 12 |
| 3.1 Hardware and software prerequisites .....                 | 12 |
| 3.1.1 Minimal Hardware Requirements .....                     | 12 |
| 3.1.2 Minimal Software Requirements .....                     | 14 |
| 3.2 Central Server Facility .....                             | 15 |
| 3.2.1 Orchestration server CSF0.....                          | 15 |
| 3.2.2 Gateway server CSF1.....                                | 43 |
| 3.2.3 Storage server CSF2.....                                | 44 |
| 3.3 Common node.....  | 48 |
| 3.3.1 Compute node.....                                       | 48 |
| 3.3.2 Data plane router .....                                 | 53 |
| 3.3.3 OpenFlow fabric .....                                   | 55 |
| 3.3.4 Rack switch.....  | 55 |
| Appendix I: Service management quick reference .....          | 56 |
| A - Start and stop GTS core services on CSF0.1 .....          | 56 |
| B - Restart OpenStack services on CSF0.2 .....                | 57 |
| C - Restart services on CSF0.3 .....                          | 57 |
| Appendix II – OpenFlow switch configuration at GTS pods ..... | 58 |
| A - Installation of the Device .....                          | 58 |
| A.1 Mount the switch in the 19-inch rack .....                | 58 |
| A.2 Ground the device .....                                   | 58 |
| A.3 Connect the power supply and cords .....                  | 58 |
| A.4 Install the fan trays .....                               | 58 |

|   |    |
|---|----|
| A.5 Connect the console cable .....                   | 59 |
| A.6 Verify the installation and power on switch ..... | 60 |
| A.7 Software upgrades via USB .....                   | 60 |
| A.8 Troubleshoot the device .....                     | 60 |
| B - Basic Configuration .....                         | 61 |
| B.1 Configure multiple SSH login on the device .....  | 61 |
| B.2 Configure IP address and subnet information ..... | 61 |
| B.3 Configure manager user "admin" und password ..... | 61 |
| B.4 Cable Map .....                                   | 62 |
| References .....                                      | 62 |

|   |                                       |
|---|---------------------------------------|
| Document Version: 1.1   | Date: September, 1 <sup>st</sup> 2015 |
| <b>Authors</b><br>Michal Hazlinsky, CESNET<br>Milos Zdravkovic, AMRES/RCUB<br>Aleksander Weinert, PSNC<br>Buelent Arslan, DFN<br>Susanne Naegele-Jackson, DFN<br>Fabio Farina, GARR<br>Paolo Velati, GARR |                                       |

## Abstract

This document summarizes briefly the architecture of the GÉANT Global Testbed Service (GTS) facility and discusses in detail the steps needed to install and configure all the components needed to setup a complete deployment of the service.

The objective of the document is twofold. First of all, we aim at providing an unexperienced administrator a step-by-step guide to configure a GTS domain starting from scratch. Secondly, the document is a living logbook of the experiences that the GTS staff has collected facing the deployment of such a rich and complex ecosystem of devices and software platforms.

## 1 Architecture overview

The GÉANT Testbed Service (GTS) allows the set-up of customized packet-based testbed environments. The advantage of such testbed environments in GTS is that they are completely isolated from other testbed domains, allowing researchers and experimenters to simulate their experiments over resources in a real network without having to worry about affecting other testbeds or production services. To build such a testbed domain, the user selects testbed components from a pool of virtualized resources via a graphical user interface (GUI) or defines the environment using a Domain Specific Language (DSL) description.

To offer such a customized experimental network facility it is necessary to put the following components in place:

- Physical resources that are to be made available to experimenters at a variety of geographical locations or PODS (Points of Distribution); these locations then serve as common nodes in the testbed service. The elementary atomic elements that the end users compose to build their experiments are taken from the pool of resources provided by the PODS.
- A Central Server Facility (CSF) where components and servers are located. These are needed for enabling the GTS instance and are not required at every location. The resources for CSF are reserved exclusively for overall service control. No user experiments run on CSF components and servers. It is good practice to replicate a CSF deployment as backup, in order to ensure availability and resiliency of the testbed service core components.

CSF is a separate entity from the Common node. However, it is recommended to collocate CSF with one common node to simplify the connectivity between the data plane router and control plane VRFs.

In GTS, resources that are offered to users comprise mainly virtual machines (VMs), virtual circuits (VCs) and instances of OpenFlow switches (OFXs), but other types of resources could be integrated into the facility as well as long as certain control primitives for GTS processing are made available for these resources. Typically, for GTS to setup a testbed, the following steps have to be processed: a user defines a new project via a graphical user interface (GUI) and builds a testbed description that represents the testbed environment required for the experiment. For each project, a testbed control agent (TCA) is created that manages all project related functions. Once the testbed description is complete, it is analysed by the GTS service engine for any obvious errors or omissions. If there is nothing critical, the Service proceeds to locate and reserve the indicated testbed resources. After the reservation of all resources, the experimenter requests the GTS Service Engine to instantiate the testbed. As resources are initialized and brought into service, the control of those resources is passed to the researcher's own testbed control agent. Instantiating resources are available to the users when the selected resources have been activated. In other words, for every resource that is to be available in the testbed service, it is necessary to develop control modules that are capable of reserving, activating, deactivating, releasing and querying that resource. As these control primitives can be written in a way that they specifically map to a resource, any type of resource could be made available in the service.

### 1.1 CSF and POD internal devices

The architecture of the GÉANT Testbed Service distinguishes between the central server facilities (CSF) and common nodes or Points of Distribution (PODS). The CSF holds all servers that are required for the overall management and operation of the whole GTS infrastructure and that do not stand for any specific point of distribution or common node.

Common nodes or general PODs typically comprise:

- One or more compute nodes for delivering virtual machine resources,
- a data plane router for delivering virtual circuits and virtual routers,
- OpenFlow fabric for delivering OpenFlow resources,
- and a rack switch.

Components of a CSF are:

- an orchestration physical server CSF0 hosting
  - a virtual server for GTS core services CSF0.0,
  - a virtual server for the OpenStack controller CSF0.1,
  - a virtual server for OpenNSA CSF0.2,
  - a virtual server running OpenStack Network (Neutron) CSF0.3;
- a gateway server CSF1 for letting users access GTS
  - a GTS service GW CSF1.0;
- a storage server CSF2 for hosting users' experiments configurations, input and output data.

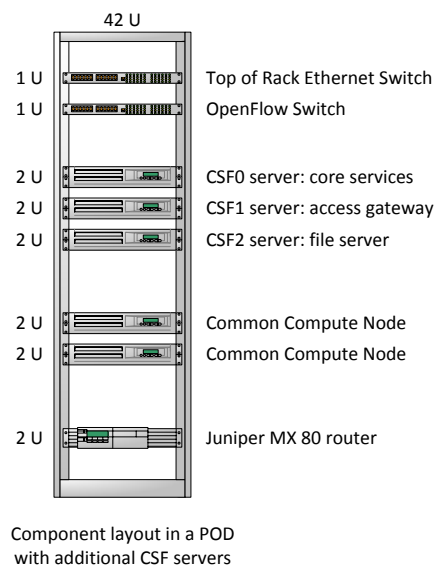
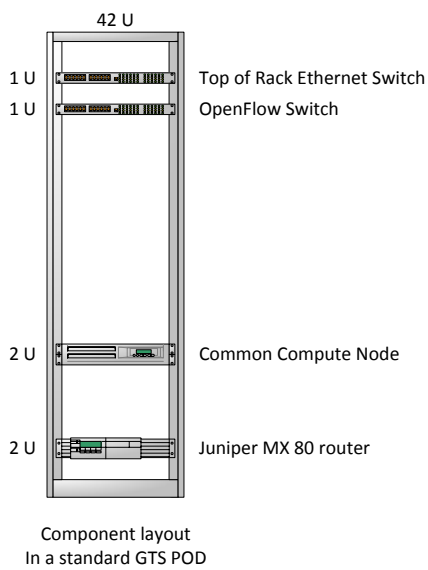
These physical components are required for this framework to be able to create VMs, to have a separate router to manage the data plane and another switch responsible for the control plane. A switch with OpenFlow capability is required if OpenFlow resources are to be included in the testbed service.

GÉANT GTS implementation supports officially the following equipment:

- Dell 520 servers are used to support virtual machine provisioning,
- Juniper MX-80 routers are in place for data plane connections.
- Juniper EX-4200 switches are responsible for the management of the control plane.
- HP 5900 OpenFlow switches where users' flowspace in the HP Openflow switches are separated based on VLAN isolation.

GTS ensures distinct customized work areas based on a control plane that allows virtual routing and forwarding with multiple parallel instances of routing tables.

The following figure shows the rack units layout for common nodes at the PODS and for the Central Service Facility.



## 1.2 GÈANT GTS WAN connectivity

The GÈANT Global Testbed Service has its own layer 2 network infrastructure provisioned over GEANT lambdas. GTS network consist by GTS nodes currently interconnected as full mesh with 10GE links provided by GEANT. For better scalability in the future, the topology will have to be changed. Currently planned choice is hypercube. Figure 1 shows the four nodes deployed in version 1.1.

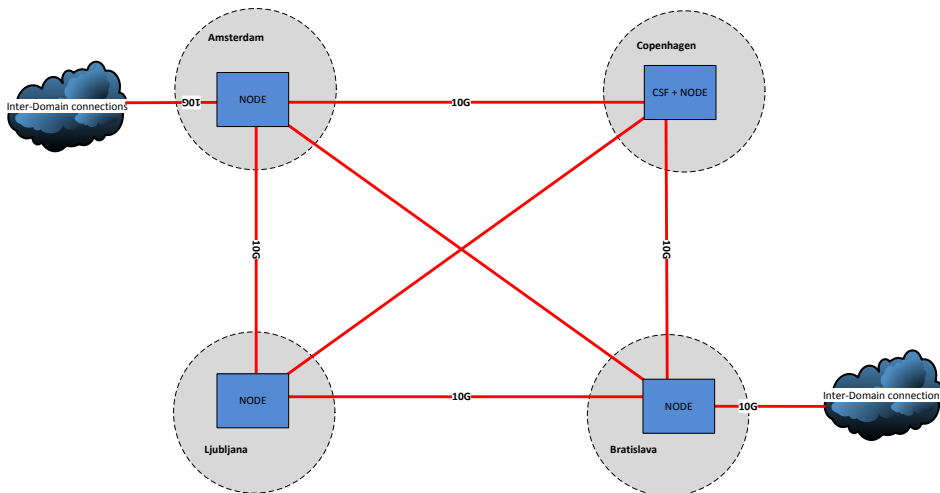


Figure 1: Overall topology

All four nodes are equally equipped and have equal capabilities. In addition there is the Central Server Facility collocated in the Copenhagen node. There are three servers in the CSF and all GTS orchestration software runs there. It is good practise to duplicate CSF functions at least in one other node to improve overall GTS resilience.

## 2 Function parts and their configuration

This chapter will describe the design of GTS infrastructure and main functional parts. An important part of GTS are computing resources: in current version of the service, computing resources means virtual machines, but providing of bare metal servers is also on the roadmap.

Networking is the second function part of GTS. There are several important aspects to solve like separation and isolation of data flows produced by individual projects, providing external access into isolated testbed environment, provide access to internet for all active resources and finally to secure independent management access to all equipment to secure ability to force shutdown of any resource in case of unexpected issues.

Next sections will describe this to pars in detail.

### 2.1 Virtual machines provisioning

The creation and the life-cycle management of virtual machines in GTS is through the OpenStack cloud stack. Essential OpenStack services for virtual machines provisioning must be installed and configured to permit GTS drive the creation of compute resources for the users.

The OpenStack components that are needed by GTS core services are the following:

- Keystone: the OpenStack authentication and authorization service is wrapped inside GTS and keeps track of the account linking between the GTS and OpenStack internal credentials.
- Nova: manages the lifecycle of the VMs on the PODs.
- Glance and Cinder: manage the boot images with the OS for the VMs and enables the snapshotting of running instances
- Neutron: used in GTS to provide basic connectivity among the VMs.

### 2.2 Networking

Networking in GTS is divided into two independent networks based on their primary purposes. As shown on figure 2 both networks are interconnecting all GTS pods (racks), but each in a different way. Both networks are described below in separate subsections.

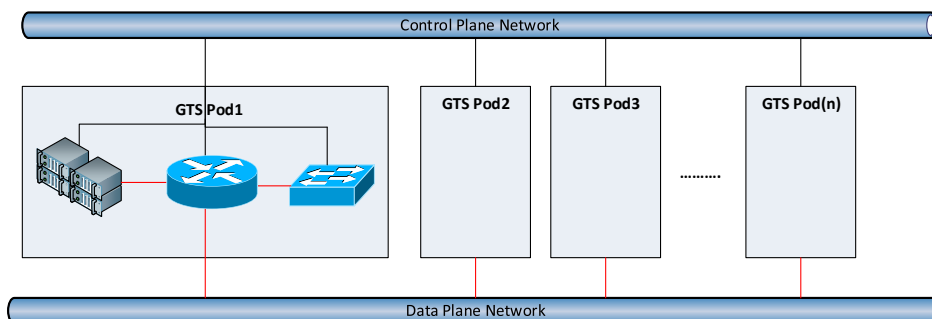


Figure 2: GTS networking overview



The next picture bellow shows the complete scheme of rack patch connections. There you can see networks' realization and physically connection to every device in the current rack design. The node with Central Server facility (CSF, see section 3.2) is shown on this picture, when no CSF is in the node, just skip that three CSF servers with all their connections and no public VLAN is configured in rack switch, because pod without CSF has no direct connection to public Internet.

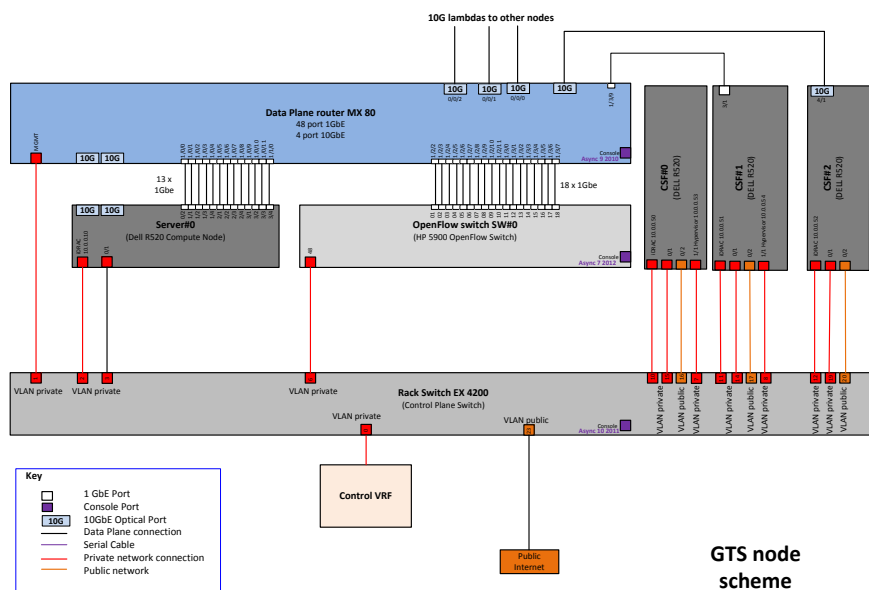


Figure 3: Rack connections scheme

### 2.2.1 Control plane network

The control network serves for administration and orchestration purposes and interconnects all pods and their IP subnets. In the current version of GTS, every pod has its own private CIDR block and the range of each block has been set to /26 (64 addresses including network and broadcast IP address). Schematic view on the control network is shown on figure 4.

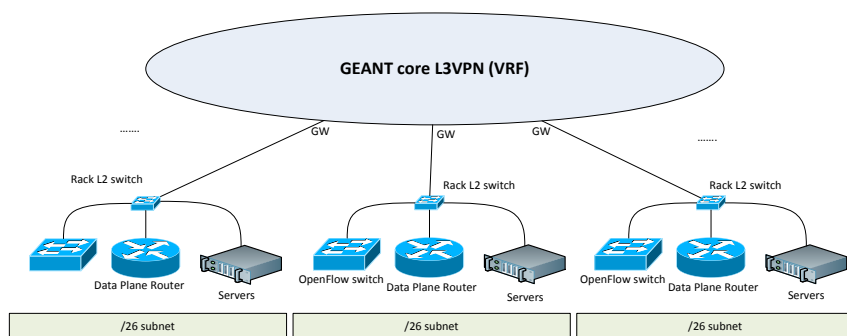


Figure 4: Control network logical scheme

Following table will list all devices in the GTS pod with all interfaces connected to private control network along with their purpose. Rack switch is the device that represents that private broadcast domain so it will not be in this list, since all private network links from any other device lead into this switch.

| Device              | Interfaces                | Purpose  |
|---------------------|---------------------------|--|
| Compute node server | eth0                      | Management and service access into the server OS and to hypervisor   |
| Data Plane router   | Mngt                      | Out of band management Ethernet interface – management and service purpose   |
| OpenFlow switch     | Port Gi1/48               | Management and service purpose   |
| Geant core router   | GTS VRF interface         | GTS private control network  |
| CSF0 server         | eth0; eth12; iDrac (IPMI) | Private network OVS bridge; Management and service access into the server OS and to hypervisor; Server remote management |
| CSF1 server         | eth0; eth12, iDrac (IPMI) | Private network OVS bridge; Management and service access into the server OS and to hypervisor; Server remote management |
| CSF2 server         | eth0, iDrac (IPMI)        | Management and service access into the server OS and to hypervisor; Server remote management                             |

Table 1: List of devices and their interfaces in private network

The private control network has several main purposes.

- To provide administrator access to all devices
- Serve as internal network for communication between GTS orchestration components
- Provide internet access to all devices through service access gateway (VM on CSF1)

#### Routing in control network

The access gateway (VM running on CSF1, see 3.2.2.1) is the main Internet gateway for entire control network, it is running NAT service and also the OpenVPN service to provide external access for SA2 personal into control network. It is good practise to give its own CIDR block of IP addresses, when collocated with common node it can share the L2 infrastructure, but having its own IP subnet is providing more flexibility and independence for CSF devices (for example when migration is needed). It is also good to create point to point IP link between nearest router (this will be VRF instance in GTS implementation) and OpenVPN server. In this way all devices in control network will have access to OpenVPN server and also internet GW hosted on the same VM through their subnet's default GW.

This will make IP addressing of PODs (and also of CSF devices) unified regardless where is the OpenVPN server hosted.

2.2.2 Data plane network

The Core of the Data plane network is made by data plane routers (Juniper MX-80, one in each pod) directly interconnected by 10GE lambdas. IP/MPLS domain is provisioned over this topology and all data plane routers represent PE routers here. Any device in any pod that hosts resources has several links to data plane router as shown on the figure below.

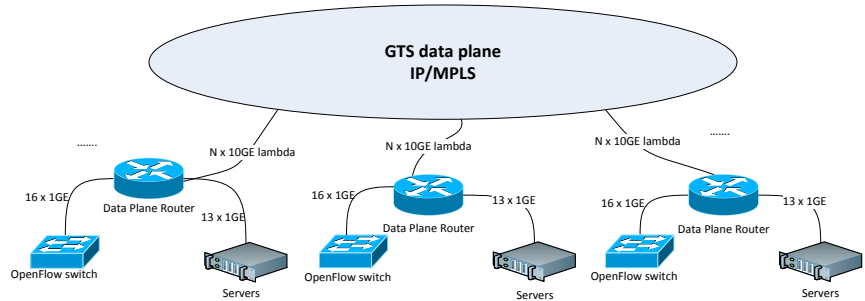


Figure 5: Data plane network logical scheme

General Purpose VPLS

There is VPLS service configured between all PEs in the data plane. It serves for provision project VLANs for GTS user’s projects. These VLANs are used for interconnect all active resources belonging to the project to provide management access to users and Internet connectivity.

2.2.3 Public network

Public network interconnects all public interfaces of CSF servers.

| Device            | Interfaces          | Purpose                    |
|-------------------|---------------------|----------------------------|
| CSF0 server       | eth1                | Public network OVS bridge; |
| CSF1 server       | eth1                | Public network OVS bridge; |
| Geant core router | Public IP interface | Internet connectivity      |

### 3 GTS components installation and configuration

This chapter describes the elements of the GTS architecture and provides installation and configuration instructions for every role.

#### 3.1 Hardware and software prerequisites

Some common preparatory steps need to be satisfied before deploying services over CSF and Common nodes. As CSF exploits virtualization for creating its internal component, the resources provided by the physical servers hosting it, and the size of the virtual machines running the components need to be planned. Operating system deployment on the nodes is also a common, standard task that need to be covered for every element at the CSF and at the Common nodes. For simplicity we assume that CSF physical servers, CSF virtual machines and Common nodes physical servers run the same version of Ubuntu Linux OS.

##### 3.1.1 Minimal Hardware Requirements

This section describes the hardware configuration for every physical and virtual node used by the GÉANT GTS deployment. These are the minimal recommended setup for running GTS core services.

CSF0 Orchestration physical server:

- CPUs with virtualisation extensions (VT-x/AMD-v)
- 32 GB of RAM
- Fault tolerant disk configuration such as RAID 1 (mirroring) with 50 GB of space
- 4 Ethernet ports (two in control plane, one in data plane and one publicly visible)

CSF0.0 GTS core services virtual machine:

- X vCPUs
- X GB of RAM
- 80 GB of HDD
- 2 Ethernet port

CSF0.1 OpenStack Controller virtual machine:

- 4 vCPUs
- 4 GB of RAM
- 80 GB of HDD
- one Ethernet port attached to control plane through *br-ctl* virtual switch on CSF0

CSF0.2 OpenNSA virtual machine:

- 1 vCPUs
- 2 GB of RAM
- 10 GB of HDD
- 1 Ethernet port

CSF0.3 OpenStack Network (Neutron) virtual machine:

- 1 vCPUs
- 2 GB of RAM
- 10 GB of HDD
- one Ethernet port attached to control plane through *br-ctl* virtual switch on CSF0
- one Ethernet port attached to data plane through *br-data* virtual switch on CSF0

Keeping Neutron in a separate virtual machine is mainly a recommendation for keeping the configuration isolated and simple. For users with experience in OpenStack Administration, Neutron could be deployed on the CSF0.1 VM together with the OpenStack controller.

CSF1 Gateway physical server:

- CPUs with virtualisation extensions (VT-x/AMD-V)
- 32 GB of RAM
- Fault tolerant disk configuration such as RAID 1 (mirroring) with 500 GB of space
- Total of 6 Ethernet ports as following:
  - one BCM5720 embedded NIC (2 x 1G port)
  - one 2 x 10G port NIC of any type (currently not in use)
  - three BCM5719 Adapters (1 x 1G port)

CSF1.0 GTS service GW virtual machine:

- 1 vCPUs
- 1 GB of RAM
- 10 GB of HDD
- 2 Ethernet port

CSF2 Storage physical server:

- relatively large storage capacity (multiple disks in some reliable RAID configuration)
- high network throughput (channel bonding of two or more 1 Gbps Ethernet ports)

Common nodes are the physical servers at the PODs that deliver compute resources to the experimentations as virtual machines.

Common node (Compute)

- CPUs with virtualisation extensions (VT-x/AMD-V)
- 32 GB of RAM
- Fault tolerant disk configuration such as RAID 1 (mirroring) with 500 GB of space
- Total of 12 Ethernet ports as following:
  - one BCM5720 embedded NIC (2 x 1G port)
  - one 2 x 10G port NIC of any type (currently not in use)
  - three BCM5719 Adapters (4 x 1G port)

### 3.1.2 Minimal Software Requirements

The operating system used on every node is the Ubuntu 12.04 LTS server edition 64-bit (amd64 architecture) with the addition of OpenStack Havana repositories.

There are two ways to install and configure CSF0. It is possible to download the experimental automated ISO image installer from our FTP server or follow the manual installation and configuration process described in the rest of this section.

Standard Ubuntu installation is the suggested prerequisite for kick-starting the physical and virtual nodes both on CSF and on Common nodes. The default user on the system is expected to be named "taas". Once the installation is complete, the user should login and gain root privileges using **sudo su** command.

The following commands will add OpenStack Havana repositories, upgrade the system and install common software:

```
apt-get update
apt-get install software-properties-common
apt-get install python-software-properties
add-apt-repository cloud-archive:havana
apt-get update
apt-get -y upgrade
apt-get -y dist-upgrade
apt-get -y install ntp openssh-server
```

Add **iburst** option at the end of each **server** line in `/etc/ntp.conf` and restart NTP daemon using **service ntp restart** command:

```
server 0.ubuntu.pool.ntp.org iburst
server 1.ubuntu.pool.ntp.org iburst
server 2.ubuntu.pool.ntp.org iburst
server 3.ubuntu.pool.ntp.org iburst
```

You can check the status of NTP synchronisation using **ntpq -p** command (there should be exactly one server marked with \*).

## 3.2 Central Server Facility

CSF is the name for the group of servers that manage the entire GTS infrastructure. In general, it is functionally completely independent on any common node, but it makes sense to collocate these servers along with different nodes. It simplifies connecting of CSF into GTS internal networks. CSF needs to have public IP connectivity in place that is not necessary in common node.

### 3.2.1 Orchestration server CSF0

CSF0 is mostly used as a Libvirt host for management VMs. This section describes how to configure virtualization on CSF0 physical server using KVM.

#### **STEP 0: Install minimal software requirements**

See section 3.1.2 for installing the common packages

#### **STEP 1: Configure networking**

The following is an example of how `/etc/network/interfaces` should look like:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

# control plane interface
auto eth0
iface eth0 inet static
    address CSF0_CTLnet_IP
    netmask 255.255.255.192
    gateway TaaSAG_IP #Taas access gateway IP addr
    dns-nameservers YOUR_DNSsrv_IP
    up ip route add CTL_NET via VRF_gw # static route for ctl network since def. route leads via
#taas GW to internet

# control plane interface for br-ctl
auto eth1
iface eth1 inet manual
    up ip link set eth1 promisc on

# data plane interface for br-data
auto eth2
iface eth2 inet manual
    up ip link set eth2 promisc on

# public plane interface for br-pub
auto eth3
iface eth3 inet manual
    up ip link set eth3 promisc on
```

Interfaces can be identified using `lshw -class network | less` command and renamed in `/etc/udev/rules.d/70-persistent-net.rules`. To apply these changes **reboot** the system.

### STEP 2: Install packages.

The following commands will install the necessary packages:

```
apt-get install --no-install-recommends ubuntu-desktop  
apt-get install qemu-kvm libvirt-bin openvswitch-switch virt-manager
```

Installing of desktop environment is optional. It can be useful for managing VMs using graphical apps like virt-manager.

### STEP 3: Configure OpenVSwitch.

Create the following OVS instances (one for each network -plane):

```
ovs-vsctl add br-ctl  
ovs-vsctl add-port br-ctl eth1  
ovs-vsctl add br-data  
ovs-vsctl add-port br-data eth2  
ovs-vsctl add br-pub  
ovs-vsctl add-port br-pub eth3
```

Create the symbolic link to gnome terminal and **reboot** the system:

```
ln -s /usr/bin/gnome-terminal /home/taas/Desktop
```

### STEP 4 – Create the CSF0.x Virtual Machines.

Login to the desktop environment.

Double click on gnome-terminal icon then type **sudo virt-manager**.

From there you can create OpenStack controller and other VMs described in the sections 3.1.1.

Once done you have to connect the VMs to the OpenVSwitch for enabling virtual networking.

To attach the VM virtual NICs to appropriate networking planes use the **virsh edit <vm-name>** command to change the networking configuration as demonstrated in the following example:

```
<interface type='bridge'  
  <mac address='52:54:00:71:b1:b6'/>  
  <source bridge='br-ctl'/>  
  <virtualport type='openvswitch'/>  
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>  
</interface>
```

You can permanently disable graphical environment in order to save some hardware resources:

```
echo "manual" >> /etc/init/lightdm.override  
reboot
```

You can turn it on again using **start lightdm** command or if you want to enable it permanently, just remove the /etc/init/lightdm.override file.



### 3.2.1.1 GTS core services server CSF0.0

All user interactions with GTS infrastructure go through GTS core server so there is no need to expose OpenStack Controller or OpenNSA to the outside world. In particular, as we want to let user access the VMs through VNC we need that the Nova VNC proxy service, usually a part of OpenStack controller, is installed alongside with the GTS core application.

#### STEP 0: Install minimal software requirements and VNC Proxy

See section 3.1.2 for installing the common packages

The following commands will add OpenStack Havana repositories, upgrade the system and install all the necessary packages:

```
apt-get --no-install-recommends install nova-novncproxy
```

Make sure that the “controller” can be resolved in /etc/hosts. For example:

```
controller 10.0.0.62
```

The content of /etc/nova/nova.conf should be:

```
[DEFAULT]

my_ip=PRIVATE_IP_ADDRES_OF_THIS_NODE
rabbit_host = controller
rabbit_password =ADMIN_PASS
rabbit_userid = guest
rabbit_port = 5672
auth_strategy=keystone
rpc_backend = nova.rpc.impl_kombu
```

Invoke **service nova-novncproxy restart** command.

#### STEP 1: CREATING USER taas

- Checking if the user taas is already present

To check if the taas user is already present please run this command:

```
userabc@TaasCore:~$ getent passwd | grep taas
```

if the user is present the output should be similar to this:

```
taas:x:1000:1000:taas,,,:/home/taas:/bin/bash
```

If there is no output present that means that the user taas has to be created.

- Creating user taas

To create user taas please run those commands:

```
userabc@TaasCore:~$ sudo adduser taas
```

(please give the password when asked and additional information if you want.)

```
userabc@TaasCore:~$ sudo usermod -aG sudo taas
```

## STEP 2: JAVA JDK INSTALLATION

- Download Java JDK from ftp server

To install Java JDK simply download the packet jdk-6u45-linux-x64.bin from the Oracle servers

<http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html#jdk-6u45-oth-JPR>

It will download file jdk-6u45-linux-x64.bin. Copy this file to /usr/lib/jvm/ directory on TaaS Core VM. Change the attributes to make this file executable.

```
taas@TaasCore:/usr/lib/jvm$ chmod +x jdk-6u45-linux-x64.bin
```

Run this file:

```
taas@TaasCore:/usr/lib/jvm$ ./jdk-6u45-linux-x64.bin
```

It will extract the Java files in jdk1.6.0\_45 folder.

- Set installed Java as default version.

Run those commands (first three commands are split across two lines per command due to their length)

```
taas@TaasCore:~$ sudo update-alternatives --install "/usr/bin/java"
"java" "/usr/lib/jvm/jdk1.6.0_45/bin/java" 1

taas@TaasCore:~$ sudo update-alternatives --install "/usr/bin/javac"
"javac" "/usr/lib/jvm/jdk1.6.0_45/bin/javac" 1

taas@TaasCore:~$ sudo update-alternatives --install "/usr/bin/javaws"
"javaws" "/usr/lib/jvm/jdk1.6.0_45/bin/javaws" 1

taas@TaasCore:~$ sudo update-alternatives --config java
```

After this command, output will be similar to the following (although it could be different on each system). Read through the list and find the number for the Oracle JDK installation (/usr/lib/jvm/jdk1.6.0\_45/bin/java)

There are 2 choices for the alternative java (providing /usr/bin/java).

| Selection | Path                                 | Priority | Status      |
|-----------|--------------------------------------|----------|-------------|
| * 0       | /usr/lib/jvm/java-6-sun/jre/bin/java | 63       | auto mode   |
| 1         | /usr/lib/jvm/java-6-sun/jre/bin/java | 63       | manual mode |
| 2         | /usr/lib/jvm/jdk1.6.0_45/bin/java    | 1        | manual mode |

```
Press enter to keep the current choice[*], or type selection number:
```

In this case, number 2 should be chosen.

```
taas@TaasCore:~$ sudo update-alternatives --config javac
```

Follow steps similar to those listed above if you see a list of options (choose `/usr/lib/jvm/jdk1.6.0_45/bin/javac`). In case where there was no the OpenJDK javac binary installed before, the output will look like the following:

```
There is only one alternative in link group javac:
/usr/lib/jvm/jdk1.6.0_45/bin/javac
Nothing to configure.
```

```
taas@TaasCore:~$ sudo update-alternatives --config javaws
```

If you get a list of options, just type in the number of the path to the Oracle javaws command (`/usr/lib/jvm/jdk1.6.0_45/bin/javaws`), and press Enter. If there was no previous OpenJDK version of javaws the output will be simple again.

Please test to ensure everything is setup correctly. Proceed with those commands:

```
taas@TaasCore:~$ java -version
```

The output should be:

```
java version "1.6.0_45"
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.45-b01, mixed mode)
```

```
taas@TaasCore:~$ javac -version
```

The output should be:

```
javac 1.6.0_45
```

```
taas@TaasCore:~$ javaws -version
```

The output should be:

```
Java(TM) Web Start 1.6.0_45
[which is followed by a long usage message.]
```

- `JAVA_HOME` and `PATH` environment variables.

Edit the following file with your favourite editor (for instance):

```
taas@TaasCore:~$ sudo vi /etc/environment
```

Enter the following at the bottom of the file:

```
JAVA_HOME="/usr/lib/jvm/jdk1.6.0_45"
```

Find the line where PATH environment variable is and add

```
:/usr/lib/jvm/jdk1.6.0_45/bin
```

Do not forget to separate previous part with colon and finish the line with quotation marks. The example of the PATH environment variable could look like this:

```
PATH="/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/jvm/jdk1.6.0_45/bin"
```

Save the file and update its content by the command:

```
taas@TaasCore:~$ source /etc/environment
```

To check if variables are working properly type the following command:

```
taas@TaasCore:~$ echo $JAVA_HOME
```

You should see the following output:

```
/usr/lib/jvm/jdk1.6.0_45
```

Lastly, verify that JAVA\_HOME is correctly set for the sudo user:

```
taas@TaasCore:~$ sudo env | grep JAVA_HOME
```

You should see the result like this:

```
JAVA_HOME=/usr/lib/jvm/jdk1.6.0_45
```

The JDK 6.0 update 45 is installed.

### STEP 3: KARAF INSTALLATION

- Download Karaf packet from ftp server.

To install Karaf simply download the packet apache-karaf-2.3.3.deb from ftp server

```
wget http://archive.apache.org/dist/karaf/2.3.3/apache-karaf-2.3.3.tar.gz
```

Unpack it with

```
tar xvzf apache-karaf-2.3.3.tar.gz
```

that will install Karaf in /home/taas/ directory. Additional configuration is needed depending on installation environment.

- **Karaf configuration files.**


It is important to configure couple of Karaf files. Edit them with your favourite editor.

```
/home/taas/apache-karaf-2.3.3/etc taas.rm.cfg
```

| Parameter                      | Value                 |
|--------------------------------|-----------------------|
| persistenceUnit                | = rmProduction        |
| scheduleForQueryResourcesDelay | = 10000               |
| stpsConfigFile                 | = /home/taas/stps.cfg |

```
/home/taas/apache-karaf-2.3.3/etc/taas.rca.os.cfg
```

| Parameter   | Value   |
|---|---|
| providerId  | OpenStack   |
| admin.user  | admin   |
| admin.pass  |   |
| admin.role  | admin   |
| admin.tenant  | admin   |
| service.identity  | http://OPENSTACK_IP:5000/v2.0   |
| service.keystone  | http://OPENSTACK_IP:35357/v2.0  |
| service.nova  | http://OPENSTACK_IP:8774/v2   |
| service.neutron   | http://OPENSTACK_IP:9696/v2.0   |
| (Where OPENSTACK_IP is the private IP address dedicated for VM with OpenStack service.) |   |
| vm.flavor   | 1   |
| vm.waitTimeout  | 300   |
| allowedComputeNodes   | cph-compute0,bra-compute0,lju-compute0  |
| hosts.reuse   | false   |
| persistenceUnit   | rcaOsProductionUnit   |
| vm.image  | The value for this parameter can be found by opening browser and load the page of OpenStack Horizon: <a href="http://OPENSTACK_IP/horizon">http://OPENSTACK_IP/horizon</a> . Login as admin (credentials specified above). On the left column (below openstack sign) choose Admin bookmark, then in System Panel choose Images, mark the default VM image and the Overview will be displayed. The parameter is the ID value. On the example from the picture below it is 892e6d63-73b6-4091-8a5f-53ea3b648e54 |



**openstack**

DASHBOARD

ProjectAdmin

System Panel

Overview

Hypervisors

Instances

Volumes

Flavors

Images

Networks

Routers

Defaults

System Info

Identity Panel

Projects

Users

Image Detail

Logged in as: admin

Settings

Help

Sign Out

Overview

Image Overview

Info

Name

sa2host-v2

ID

892e6d63-73b6-4091-8a5f-53ea3b648e54

Status

Active

Public

True

Protected

False

Checksum

0e514c95ec4720306d7fd1379d80bda4

Created

2014-10-09T09:17:25

Updated

2014-10-09T09:17:39

Specs

Size

1.9 GB

Container Format

BARE

Disk Format

QCOW2

Min Disk


10.0GB

Min RAM

1GB

Custom Properties

|              |   |
|--------------|---|
| vm.network   | <p>To find this parameter, staying on the web page of OpenStack Horizon, on the left column, please mark one position lower, i.e. Networks. Select the appropriate network from presented. An Overview of the network will be shown. The requested parameter is the ID value. On the example from the picture below it is</p> <p>b3af869b-dd51-4cf3-b82f-12045574a268</p> |
| hosts.tenant | <p>The value for this parameter can be taken from the same web page, one line lower: it is Project ID value. On the example from the picture below it is</p> <p>c173862b09904203b4941695e857a313</p>  |



**openstack**

DASHBOARD

Project Admin

System Panel

- Overview
- Hypervisors
- Instances
- Volumes
- Flavors
- Images
- Networks
- Routers
- Defaults
- System Info

Identity Panel

- Projects
- Users

Network Detail: taas

Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

### Network Overview

**Name**  
taas

**ID**  
b3af669b-dd51-4c63-b82f-12045574a268

**Project ID**  
c173862b09904203b4941695e857a313

**Status**  
ACTIVE

**Admin State**  
UP

**Shared**  
Yes

**External Network**  
No

**Provider Network**  
Network Type: vlan  
Physical Network: physnet1  
Segmentation ID: 1000

### Subnets

[+ Create Subnet](#) [Delete Subnets](#)

| <input type="checkbox"/> | Name | CIDR             | IP Version | Gateway IP    | Actions  |
|--------------------------|------|------------------|------------|---------------|--|
| <input type="checkbox"/> | taas | 192.168.100.0/24 | IPv4       | 192.168.100.1 | <a href="#">Edit Subnet</a> <a href="#">More</a> |

Displaying 1 item

### Ports

[+ Create Port](#)

| Name                 | Fixed IPs | Device Attached | Status | Admin State | Actions |
|----------------------|-----------|-----------------|--------|-------------|---------|
| No items to display. |           |                 |        |             |         |

Displaying 0 items

```
/home/taas/apache-karaf-2.3.3/etc/taas.rca.vc.cfg
```

| Parameter             | Value  |
|-----------------------|--|
| ProviderNsa           | urn:ogf:network:taas-cph:nsa   |
| RequesterNsa          | urn:ogf:network:taascore-client-cph:nsa  |
| OpennsaUrl            | http://OPENNSA_IP:9443/NSI/services/CS2<br>(Where OPENNSA_IP is the private IP address of VM dedicated for OpenNSA service.) |
| ClientEndpointAddress | TAASCORE_IP:9000<br>(Where TAASCORE_IP is the private IP address dedicated for TaaS Core VM.)                                |
| HttpServiceContext    | /nsi   |
| persistenceUnit       | rca-vc-production  |

```
/home/taas/apache-karaf-2.3.3/etc/taas.rca.openflow.cfg
```

| Parameter   | Value          |
|---|----------------|
| providerId  | OpenFlow       |
| persistenceUnit   | productionUnit |
| ofX.user<br>(Where X is successive number for<br>OpenFlow Switches)     | Admin          |
| ofX.password<br>(Where X is successive number for<br>OpenFlow Switches) | sa2.2014       |

```
/home/taas/apache-karaf-2.3.3/bin/setenv
```

In this file the size of the heap must be set for Karaf process. Add those lines at the end of the file:

```
export JAVA_PERM_MEM=512M
export JAVA_MAX_PERM_MEM=512M
```

File setenv must have executable right, so make sure that the command:

```
Home/taas/apache-karaf-2.3.3/bin$ ls -al setenv
```

will give similar result:

```
-rwxr-xr-x 1 taas taas 1877 Oct 21 11:16 setenv
```

Otherwise please use command:

```
Home/taas/apache-karaf-2.3.3/bin$ chmod 755 setenv
```

#### STEP 4: DERBY INSTALLATION

- Download Derby packet from ftp server.

To install Derby simply download the packet db-derby-10.10.1.1-bin.tar.gz from ftp server

```
taas@TaasCore:~$ wget https://archive.apache.org/dist/db/derby/db-derby-10.10.1.1/db-derby-10.10.1.1-bin.tar.gz
```

Unpack the packet with the command:

```
taas@TaasCore:~$ tar -xvzf db-derby-10.10.1.1-bin.tar.gz
```

that will unpack Derby in /home/taas/db-derby-10.10.1.1-bin directory.



- Environment variables.

Edit the following file with your favorite editor (for instance):

```
taas@TaasCore:~$ sudo vi /etc/environment
```

Enter the following at the bottom of the file:

```
DERBY_INSTALL="/home/taas/db-derby-10.10.1.1-bin"  
CLASSPATH="/home/taas/db-derby-10.10.1.1-bin/lib/derbyclient.jar:/home/taas/db-  
derby-10.10.1.1-bin/lib/derbytools.jar:/home/taas/db-derby-10.10.1.1-  
bin/lib/derbynet.jar:."
```

Save the file and update its content by the command:

```
taas@TaasCore:~$ source /etc/environment
```

To check if variables are working properly type the following command:

```
taas@TaasCore:~$ echo $DERBY_INSTALL
```

You should see the following output:

```
/home/taas/db-derby-10.10.1.1-bin
```

and this:

```
taas@TaasCore:~$ echo $CLASSPATH
```

You should see the following output:

```
/home/taas/db-derby-10.10.1.1-bin/lib/derbyclient.jar:/home/taas/db-derby-  
10.10.1.1-bin/lib/derbytools.jar:/home/taas/db-derby-10.10.1.1-  
bin/lib/derbynet.jar:..
```

#### **STEP 5: TOMCAT INSTALLATION**

- Download Tomcat packet from ftp server.

To install Tomcat simply download the packet `apache-tomcat-7.0.54.tar.gz` from ftp server

```
taas@TaasCore:~$ wget https://archive.apache.org/dist/tomcat/tomcat-  
7/v7.0.54/bin/apache-tomcat-7.0.54.tar.gz
```

Unpack the packet with the command:

```
taas@TaasCore:~$ tar -xvzf apache-tomcat-7.0.54.tar.gz
```

that will unpack Tomcat in /home/taas/apache-tomcat-7.0.54 directory.

- Environment variable and configuration file.

Environment variable must be added. Edit the following file with your favourite editor, (for instance):

```
taas@TaasCore:~$ sudo vi /etc/environment
```

Place the following line at the bottom of this file:

```
CATALINA_HOME="/home/taas/apache-tomcat-7.0.54"
```

Save the file and update its content by the command:

```
taas@TaasCore:~$ source /etc/environment
```

To check if variables are working properly type the following command:

```
taas@TaasCore:~$ echo $CATALINA_HOME
```

You should see the following output:

```
/home/taas/apache-tomcat-7.0.54
```

Couple of parameters should be set in a following file:

```
/home/taas/apache-tomcat-7.0.54/bin/setenv.sh
```

The content of this file should look like this (it's one single line):

```
export JAVA_OPTS="-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=5000  
-Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false  
-Djava.rmi.server.hostname=TAASCORE_IP -Xms256m -Xmx1024m -XX:PermSize=512m  
-XX:MaxPermSize=512m"
```

TAASCORE\_IP is the public IP address dedicated for TaaS Core VM.

File setenv.sh must have executable right, so make sure that the command:

```
taas@TaasCore:~/apache-tomcat-7.0.54/bin$ ls -al setenv.sh
```

will give similar result:

```
-rwxr-xr-x 1 taas taas 283 Oct 21 11:13 setenv.sh
```

Otherwise please use command:

```
taas@TaasCore:~/apache-tomcat-7.0.54/bin$ chmod 755 setenv.sh
```

- Update the GUI Version.

Now the GUI component is installed as web application over Tomcat. The following instructions apply both for installing from scratch and updating the GUI version.

Make sure that the following folder is empty:

```
/home/taas/apache-tomcat-7.0.54/webapps
```

If there is something in this folder, delete it by this command:

```
taas@TaasCore:~/apache-tomcat-7.0.54/webapps$ rm -rf *
```

To copy the latest version of the GUI, login on the page:

<https://artifactory.geant.net/artifactory/taas-snapshot-local/net/geant/taas/admin-gui/>

and find the folder with the latest version: 0.1.XXXX-SNAPSHOT (where XXXX is the highest number). Inside of the folder there will be file with the .war extension, with the name similar to: admin-gui-0.1.XXXX-SNAPSHOT.war

This file must be copied to the home folder on TaaS Core VM. In order to do so please type this command:

```
taas@TaasCore:~$ wget https://artifactory.geant.net/artifactory/taas-snapshot-local/net/geant/taas/admin-gui/0.1.XXXX-SNAPSHOT/admin-gui-0.1.XXXX-SNAPSHOT.war
```

(where XXXX is the highest version number)

Having this file in home folder move it to the folder apache-tomcat-7.0.54/webapps with name changed to ROOT.war, using this command:

```
taas@TaasCore:~$ mv admin-gui-0.1.XXXX-SNAPSHOT.war apache-tomcat-7.0.54/webapps/ROOT.war
```

XXXX is the version number.

NOTE: In case of GUI version update, the following steps are not needed. Skip directly to STEP 7 to stop and restart the services.

#### **STEP 6: CREATING USER task3 FOR MONITORING**

Users from SA2 Task3 must have easy access to log files for monitoring purposes. To create user for them please run this command:

Fabio Farina 28/4/2015 13:50

**Comment [1]:** Here is the admin-gui deployment instructions

```
taas@TaasCore:~$ sudo adduser task3
```

Set the password settled with those users.

In order to SA2T3 Users have easy access to Karaf logs from home folder please run this command from newly created task3 home folder:

```
taas@TaasCore:/home/taas3$ sudo ln -s /home/taas/apache-karaf-2.2.3/data/log log
```

This command will create symbolic link named log in their home folder.

### STEP 7: Starting the GTS core services

- **GTS core Karaf first run**

In order to run the software for the first time, you should login as TaaS user and perform presented commands from the folder /home/taas/apache-karaf-2.3.3/

Start derby:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ nohup ../db-derby-10.10.1.1-bin/bin/startNetworkServer &
```

Start karaf:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ./bin/start
```

Start Tomcat:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ../apache-tomcat-7.0.54/bin/catalina.sh jpda start
```

- **Karaf restart**

If there is a need for restart, the infrastructure user should perform the following steps in order. User should be logged in as taas and perform presented commands from the folder /home/taas/apache-karaf-2.3.3/

- **Stopping GTS core**

Stop Karaf:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ./bin/stop
```

It could happen that karaf will not stop so easily. It's a good habit to check if its process is really ended. If not, user is forced to kill that process. To do so, first find the process number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ps ax | grep karaf.jar | grep -v grep | awk '{print $1}'
```

Fabio Farina 28/4/2015 13:53

**Comment [2]:** I think install command for GTS Core should go here.

If no number is displayed it means that Karaf has ended correctly. If a number is displayed, that means that user has to kill the process with that number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ sudo kill -9 NUMBER
```

where *NUMBER* is the number given as a result of the previous command.

#### Stop Derby:

To stop derby user has to proceed with similar steps as with killing karaf process. First, user has to specify the derby process number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ps ax | grep  
NetworkServerControl | grep -v grep | awk '{print $1}'
```

then kill process with a given number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ sudo kill -9 NUMBER
```

where *NUMBER* is the number given as a result of the previous command.

#### Stop Tomcat:

In order to stop tomcat user has to proceed with similar steps as with killing derby process. First, user has to specify the tomcat process number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ps ax | grep tomcat | grep -v  
grep | awk '{print $1}'
```

then kill process with the given number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ sudo kill -9 NUMBER
```

where *NUMBER* is the number given as a result of the previous command.

#### Delete three karaf folders:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ rm -rf taas/  
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ rm -rf taasDB/  
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ rm -rf data/
```

#### NOTE to update GUI version:

Administrator should keep GUI updated to the new versions. To do so, please repeat procedures described STEP 5 at the bullet point "Update GUI Version." Then restart software as described in this section.

### 3.2.1.2 OpenStack controller server CSF0.1

GTS uses OpenStack as a backend for VM provisioning. The controller is a collection of OpenStack API daemons that act as mediators between GTS core and compute nodes. The controller also contains CLI client tools and web interface (Horizon) which are used internally by infrastructure staff for the purpose of initial setup and debugging.

The root logical volume can be extended for additional Glance images or as an alternative, Glance images can be held on a CSF2 and mounted on /var/lib/glance as a CIFS share)Ubuntu installation for the controller is straightforward (manual IP address configuration, use the entire disk to set up LVM).

#### STEP 0: Install minimal software requirements and OpenStack controller

See section 3.1.2 for installing the common packages

Modify the hosts file on the controller. The /etc/hosts file should contain an entry that looks something like the following example:

```
10.0.0.62 pra-controller controller
```

where 10.0.0.62 and pra-controller are the IP address and the hostname entered during the Ubuntu installation.

The following command adds OpenStack Havana repositories, upgrade the system and install all the necessary packages:

```
apt-get -y install python-mysqldb mysql-server rabbitmq-server keystone glance novnc nova-api  
nova-cert nova-conductor nova-consoleauth nova-doc nova-ajax-console-proxy nova-scheduler  
python-novaclient memcached libapache2-mod-wsgi openstack-dashboard cinder-api cinder-  
scheduler python-cinderclient linux-headers-`uname -r` nmap neutron-server  
  
apt-get -y remove --purge openstack-dashboard-ubuntu-theme
```

#### STEP 1: configure the OpenStack database

Configure MySQL and change the default password of RabbitMQ user:

```
mysql_secure_installation (answer yes to all the question, no on password change)  
rabbitmqctl change_password guest ADMIN_PASS
```

Enable MySQL to accept remote connections and restart the DBMS:

```
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf  
service mysql restart
```

Database creation:

```
mysql -u root -p <<EOF  
DROP DATABASE nova;  
DROP DATABASE cinder;  
DROP DATABASE glance;  
DROP DATABASE keystone;  
DROP DATABASE neutron;  
  
CREATE DATABASE nova;
```

```

CREATE DATABASE cinder;
CREATE DATABASE glance;
CREATE DATABASE keystone;
CREATE DATABASE neutron;

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'DB_PASSWORD';

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'DB_PASSWORD';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'DB_PASSWORD';

FLUSH PRIVILEGES;
EOF

```

**NOTE:** In case of mistakes in setting keystone db during the installation, the simplest option is to drop the databases and create them from scratch.

#### **STEP 1: configure the Openstack controller services**

Edit /etc/keystone/keystone.conf and add the following keys in the related sections.

```

[DEFAULT]
admin_token = YOUR_ADMIN_TOKEN
...
[sql]
connection = mysql://keystone:DB_PASSWORD@localhost/keystone

```

Synchronize Keystone database and restart its service:

```

keystone-manage db_sync
service keystone restart

```

Initial environment needed for keystone popluation:

```

export OS_SERVICE_ENDPOINT="http://controller:35357/v2.0"
export OS_SERVICE_TOKEN= YOUR_ADMIN_TOKEN

```

The following bash script will populate Keystone (remember to edit before invoking):

```

#!/bin/bash

## START - EDIT HERE
ADMIN_PASSWORD="ADMIN_PASS"

```

```

MYSQL_USER=root
MYSQL_DATABASE=keystone
MYSQL_HOST=localhost
MYSQL_PASSWORD=ROOT_DB_PASS;

KEYSTONE_HOST=controller
NEUTRON_HOST=controller
export OS_SERVICE_TOKEN=" YOUR_ADMIN_TOKEN "
## END - EDIT HERE

# Modify these variables as needed
ADMIN_PASSWORD=${ADMIN_PASSWORD:-password}
SERVICE_PASSWORD=${SERVICE_PASSWORD:-$ADMIN_PASSWORD}
export OS_SERVICE_ENDPOINT="http://controller:35357/v2.0"
SERVICE_TENANT_NAME=${SERVICE_TENANT_NAME:-service}
echo "debug $SERVICE_TENANT_NAME"

KEYSTONE_REGION=RegionOne

# Shortcut function to get a newly generated ID
function get_field() {
    while read data; do
        if [ "$1" -lt 0 ]; then
            field="$(\$(NF$1))"
        else
            field="\${$1 + 1}"
        fi
        echo "$data" | awk -F'[ \t]*\\|\\[ \t]*' '{print $field}'
    done
}

# Tenants
ADMIN_TENANT=$(keystone tenant-create --name=admin | grep " id " | get_field 2)
SERVICE_TENANT=$(keystone tenant-create --name=$SERVICE_TENANT_NAME | grep " id " | get_field 2)

# Users
ADMIN_USER=$(keystone user-create --name=admin --pass="$ADMIN_PASSWORD" --
email=admin@domain.com | grep " id " | get_field 2)
NOVA_USER=$(keystone user-create --name=nova --pass="$SERVICE_PASSWORD" --tenant-id
$SERVICE_TENANT --email=nova@domain.com | grep " id " | get_field 2)
GLANCE_USER=$(keystone user-create --name=glance --pass="$SERVICE_PASSWORD" --tenant-id
$SERVICE_TENANT --email=glance@domain.com | grep " id " | get_field 2)
NEUTRON_USER=$(keystone user-create --name=neutron --pass="$SERVICE_PASSWORD" --tenant-id
$SERVICE_TENANT --email=neutron@domain.com | grep " id " | get_field 2)
CINDER_USER=$(keystone user-create --name=cinder --pass="$SERVICE_PASSWORD" --tenant-id
$SERVICE_TENANT --email=cinder@domain.com | grep " id " | get_field 2)

# Roles
ADMIN_ROLE=$(keystone role-create --name=admin | grep " id " | get_field 2)
MEMBER_ROLE=$(keystone role-create --name=Member | grep " id " | get_field 2)

# Add Roles to Users in Tenants
keystone user-role-add --user-id $ADMIN_USER --role-id $ADMIN_ROLE --tenant-id $ADMIN_TENANT
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $NOVA_USER --role-id $ADMIN_ROLE
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $GLANCE_USER --role-id $ADMIN_ROLE
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $NEUTRON_USER --role-id $ADMIN_ROLE
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $CINDER_USER --role-id $ADMIN_ROLE

```



```
# Create services
COMPUTE_SERVICE=$(keystone service-create --name nova --type compute --description 'OpenStack
Compute Service' | grep " id " | get_field 2)
VOLUME_SERVICE=$(keystone service-create --name cinder --type volume --description 'OpenStack Volume
Service' | grep " id " | get_field 2)
IMAGE_SERVICE=$(keystone service-create --name glance --type image --description 'OpenStack Image
Service' | grep " id " | get_field 2)
IDENTITY_SERVICE=$(keystone service-create --name keystone --type identity --description 'OpenStack
Identity' | grep " id " | get_field 2)
EC2_SERVICE=$(keystone service-create --name ec2 --type ec2 --description 'OpenStack EC2 service' | grep "
id " | get_field 2)
NETWORK_SERVICE=$(keystone service-create --name neutron --type network --description 'OpenStack
Networking service' | grep " id " | get_field 2)

# Create endpoints
keystone endpoint-create --region $KEYSTONE_REGION --service-id $COMPUTE_SERVICE --publicurl
'http://"$KEYSTONE_HOST":8774/v2/$(tenant_id)s' --adminurl
'http://"$KEYSTONE_HOST":8774/v2/$(tenant_id)s' --internalurl
'http://"$KEYSTONE_HOST":8774/v2/$(tenant_id)s'
keystone endpoint-create --region $KEYSTONE_REGION --service-id $VOLUME_SERVICE --publicurl
'http://"$KEYSTONE_HOST":8776/v1/$(tenant_id)s' --adminurl
'http://"$KEYSTONE_HOST":8776/v1/$(tenant_id)s' --internalurl
'http://"$KEYSTONE_HOST":8776/v1/$(tenant_id)s'
keystone endpoint-create --region $KEYSTONE_REGION --service-id $IMAGE_SERVICE --publicurl
'http://"$KEYSTONE_HOST":9292' --adminurl 'http://"$KEYSTONE_HOST":9292' --internalurl
'http://"$KEYSTONE_HOST":9292'
keystone endpoint-create --region $KEYSTONE_REGION --service-id $IDENTITY_SERVICE --publicurl
'http://"$KEYSTONE_HOST":5000/v2.0' --adminurl 'http://"$KEYSTONE_HOST":35357/v2.0' --internalurl
'http://"$KEYSTONE_HOST":5000/v2.0'
keystone endpoint-create --region $KEYSTONE_REGION --service-id $EC2_SERVICE --publicurl
'http://"$KEYSTONE_HOST":8773/services/Cloud' --adminurl
'http://"$KEYSTONE_HOST":8773/services/Admin' --internalurl
'http://"$KEYSTONE_HOST":8773/services/Cloud'
keystone endpoint-create --region $KEYSTONE_REGION --service-id $NETWORK_SERVICE --publicurl
'http://"$NEUTRON_HOST":9696/' --adminurl 'http://"$NEUTRON_HOST":9696/' --internalurl
'http://"$NEUTRON_HOST":9696/'
```

Unset initial environment:

```
unset OS_SERVICE_ENDPOINT
unset OS_SERVICE_TOKEN
```

Create the file /root/admin\_env and **source** its content:

```
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL="http://controller:35357/v2.0"
```

This gives you administrative privileges for OpenStack CLI.

Modify the following entries in /etc/glance/glance-api.conf and /etc/glance/glance-registry.conf:

```
[DEFAULT]
...
admin_password = ADMIN_PASS
...
```

```
[keystone_authtoken]
...
sql_connection = mysql://glance:DB_PASSWORD@localhost/glance
...
```

The [filter:authtoken] section in /etc/glance/glance-api-paste.ini and /etc/glance/glance-registry-paste.ini should contain:

```
auth_host=controller
admin_user=glance
admin_tenant_name=service
admin_password=ADMIN_PASS
```

Sync Glance database and restart services using the commands

```
glance-manage db_sync
service glance-registry restart
service glance-api restart
```

The [filter:authtoken] section in /etc/nova/api-paste should contain:

```
auth_host = controller
auth_port = 5000
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = ADMIN_PASS
```

Replace the content of /etc/nova/nova.conf with the following:

```
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:DB_PASSWORD@controller/nova

[DEFAULT]
# General
verbose=True
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova

my_ip=IP_ADDRESS_OF_THIS_HOST
rabbit_host = controller
rabbit_password = ADMIN_PASS
rabbit_userid = guest
rabbit_port = 5672
auth_strategy=keystone
rpc_backend = nova.rpc.impl_kombu

# Networking
network_api_class=nova.network.neutronv2.api.API
neutron_admin_username=neutron
```

```

neutron_admin_password= ADMIN_PASS
neutron_admin_auth_url=http://controller:35357/v2.0/
neutron_auth_strategy=keystone
neutron_admin_tenant_name=service
neutron_url=http://controller:9696/
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver

### NOTE: these two parameters control metadata-agent on Neutron
neutron_metadata_proxy_shared_secret = ADMIN_PASS
service_neutron_metadata_proxy = true

# Security Groups
firewall_driver=nova.virt.firewall.NoopFirewallDriver
security_group_api=neutron

# Compute
compute_driver=libvirt.LibvirtDriver
connection_type=libvirt

# Cinder
volume_api_class=nova.volume.cinder.API

# Glance
glance_host=controller

```

Sync Nova database using **nova-manage db sync** command. Restart Nova services:

```

service nova-api restart
service nova-cert restart
service nova-consoleauth restart
service nova-scheduler restart
service nova-conductor restart

```

Note: nova-novncproxy service should be installed on GTS core server.

The [filter:authtoken] section in /etc/cinder/api-paste should contain:

```

auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000
admin_tenant_name = service
admin_user = cinder
admin_password = ADMIN_PASS

```

The content of /etc/cinder/cinder.conf should look like this:

```

[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_config = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder

```

```
volumes_dir = /var/lib/cinder/volumes

rpc_backend = cinder.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = ADMIN_PASS
```

```
[database]
connection = mysql://cinder:DB_PASSWORD@controller/cinde
```

Synchronize Cinder database using **cinder-manage db sync** command. Restart Cinder services:

```
service cinder-scheduler restart
service cinder-api restart
```

The [filter:authtoken] section in /etc/neutron/api-paste.ini should contain:

```
auth_host=controller
auth_uri=http://controller:5000
admin_user=neutron
admin_tenant_name=service
admin_password=ADMIN_PASS
```

Modify /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini (physnet2 maps to public network and physnet1 corresponds to data plane):

```
[ovs]
tenant_network_type = vlan
network_vlan_ranges = physnet2,physnet1:1000:1100
[agent]
[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

VLAN range 1000:1100 should be allowed on the trunk between access switch and compute nodes.

Modify the content of /etc/neutron/neutron.conf:

```
[DEFAULT]
state_path = /var/lib/neutron
lock_path = $state_path/lock
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
allow_overlapping_ips = True
control_exchange = neutron
rabbit_host = controller
rabbit_password = ADMIN_PASS
rabbit_port = 5672
rabbit_userid = guest
notification_driver = neutron.openstack.common.notifier.rabbit_notifier
[quotas]
[agent]
root_helper = sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = neutron
```

```
admin_password = ADMIN_PASS
signing_dir = $state_path/keystone-signing
[database]
connection = mysql://neutron:DB_PASSWORD@controller/neutron
[service_providers]
service_provider=LOADBALANCER:Haproxy:neutron.services.loadbalancer.drivers.haproxy.plugin_driver.HaproxyOnHostPluginDriver:default
```

Restart Neutron API service:

```
service neutron-server restart
```

### **STEP 3 (optional): reboot OpenStack controller**

Although not necessary, it is highly recommended that you reboot the controller VM at this point.

#### **3.2.1.3 OpenNSA server CSF0.2**

OpenNSA is the component taking care of creating point-to-point virtual circuits according to the topology descriptions provided by the users.

### **STEP 0: Install minimal software requirements and OpenNSA packages**

See section 3.1.2 for installing the common packages

Update packages and install the required software:

```
sudo apt-get update && apt-get -y upgrade
sudo apt-get install -y git postgresql postgresql-plpython-9.1 postgresql-server-dev-9.1 python
python-minimal python-pip python-dev python-openssl python-setuptools python-dateutil python-
twisted python-pip
sudo pip install --upgrade twister pycopg2 pycrypto pyasn pyasn1
```

Initialize and Start the PostgreSQL:

```
service postgresql initdb
service postgresql start
ln -s /etc/init.d/postgresql /etc/rc3.d/S99postgresql
ln -s /etc/init.d/postgresql /etc/rc3.d/K99postgresql
```

### **STEP 2: Prepare the environment for OpenNSA**

Crear user opennsa

```
sudo adduser opennsa
```

(please give the password when asked and additional information if you want.)

Create required folder opennsa:

```
sudo mkdir /usr/src/opennsa
sudo chown opennsa:opennsa /usr/src/opennsa
```

## **STEP 2: Install OpenNSA**

Change the user to opennsa and install OpenNSA:

```
sudo su - opennsa
git clone -b nsi2-r99 https://dev.niif.hu/vargat/opennsa.git
cd /usr/src/opennsa
git checkout nsi2-r99
python2.7 setup.py build
exit
sudo python2.7 setup.py install
```

## **STEP 3: configure the OpenNSA database**

Create the database:

```
cp datafiles/schema.sql /tmp/
sudo su - postgres
createdb opennsa
createuser -RSD opennsa
exit
sudo su - opennsa
psql opennsa
opennsa=# \i /tmp/schema.sql
exit
<CTRL+D>
exit
```

## **STEP 4: configure OpenNSA services**

It's important to keep the server time accurate (NTP):

```
ln -s /etc/init.d/ntpd /etc/rc3.d/S99ntpd
ln -s /etc/init.d/ntpd /etc/rc3.d/K99ntpd
/etc/init.d/ntpd start
```

Generate SSH keys:

```
sudo su - opennsa
ssh-keygen -t rsa -N '' -f ~/.ssh/opennsa_rsa.key
```

Press <ENTER> 3 times. The keys opennsa\_rsa.key and opennsa\_rsa.key.pub will be created under ~opennsa/.ssh/

The key needs to have this format for Twisted recognize it:

```
ssh-rsa (3 or more lines) opennsa@OpenNSA
```

Edit these two configuration files accordingly to your environment:

```
sudo vi /usr/src/opennsa/opennsa.conf

[service]
network=<YOUR_NETWORK_NAME>
```

```
logfile=
nrmmmap=opennsa.nrm
host=<SERVER_FQDN>
database=opennsa
dbuser=opennsa
dbpassword=<YOUR_PASS>
tls=false
```

```
sudo vi /usr/src/opennsa/opennsa.nrm
```

```
# type      name      remote      labels      bandwidth  interface
#
# Assuming that my IXP is IXP_A, follow an example:
#
# AMPATH <-- 2/2 IXP_A 2/1 --> NORDUNET
#
# bi-ethernet nordunet  nordunet#IXP_A-(in|out)  vlan:1779-1787  1000  2/1
# bi-ethernet ampath    ampath#IXP_A-(in|out)    vlan:1779-1787  1000  2/2
#
# type      name      remote      labels      bandwidth  interface
bi-ethernet ps        -            vlan:1780-1787  1000  em0
bi-ethernet port1    NetworkA#portX-(in|out)  vlan:1781-1787  1000  em1
bi-ethernet port2    NetworkB#portY-(in|out)  vlan:1782-1787  1000  em2
```

Create a ".opennsa-cli" file under ~opennsa/:

```
echo -e "bandwidth=200\nhost=localhost\nport=7080\nstarttime=+1\nendtime=+20" >
~opennsa/.opennsa-cli
```

The starttime and the endtime represent when the circuit will start and end in seconds

The file /home/opennsa/.opennsa-cli should look like this:

```
bandwidth=200
host=localhost
port=7080
starttime=+1
endtime=+20
```

Start the OpenNSA:

```
su - opennsa
cd /usr/src/opennsa
twistd -ny opennsa.tac
(-n to not create a daemon. There is also an init.d script)
```

You should see:

```
2013-07-02 14:17:08-0400 [-] Log opened.
2013-07-02 14:17:08-0400 [-] twistd 13.1.0 (/usr/local/bin/python2.7 2.7.3) starting up.
2013-07-02 14:17:08-0400 [-] reactor class: twisted.internet.epollreactor.EPollReactor.
2013-07-02 14:17:08-0400 [-] OpenNSA service initializing
2013-07-02 14:17:08-0400 [-] Provider URL: http://<SERVER_FQDN>:9080/NSI/services/CS2
2013-07-02 14:17:08-0400 [-] Topology URL: http://<SERVER_FQDN>:9080/NSI/topology/<YOUR_NETWORK>.xml
2013-07-02 14:17:08-0400 [-] Site starting on 9080
2013-07-02 14:17:08-0400 [-] Starting factory <twisted.web.server.Siteinstance at 0x1df39e0>
2013-07-02 14:17:08-0400 [-] OpenNSA service started
```

#### **STEP 5: configure Juniper MX router for OpenNSA**

Configure your MX router to support authentication using SSH keys:

In general, you have to copy your public key (opennsa\_rsa.key.pub) to the network device and set the configuration to use it. You have to copy previously created public key opennsa\_rsa.key.pub to Juniper MX router:

```
taas@lab1-opennsa:/home/opennsa/.ssh$ scp opennsa_rsa.key.pub  
taas@IP_ADDRESS_OF_A_ROUTER:.
```

Log into your Juniper MX80 router and:

```
taas@lab1-mx80> configure  
taas@lab1-mx80# set system login user opennsa authentication load-key-file opennsa_rsa.key.pub
```

Now you are ready to run OpenNSA.

#### **3.2.1.4 Neutron server CSF0.3**

This virtual machine is used for experimental purposes only and must be powered off in production environment. It contains Neutron L2, L3 and DHCP agents which must not be used at the same time as gateway instances on CSF1.

Note: Installation and configuration of neutron-metadata-agent are not described in this section. The OpenStack version currently used by GTS let the controller manage the behaviour of the metadata agent. Therefore the parameters that enable the communication between the controller and Neutron are on the controller side, see section 3.2.1.2 related to /etc/nova/nova.conf.

#### **STEP 0: Install minimal software requirements**

See section 3.1.2 for installing the common packages

#### **STEP 1: Configure networking**

The following is an example of how /etc/network/interfaces should look like:

```
# The loopback network interface  
auto lo  
iface lo inet loopback  
  
# The primary network interface  
  
# attached to br-ctl on csf0  
auto eth0  
iface eth0 inet static  
    address 192.168.0.12  
    netmask 255.255.255.0  
    gateway 192.168.0.1
```



```
dns-nameservers 8.8.8.8
```

```
# attached to br-data on csf0
# used for br-eth1 data plane (trunk to VM instances)
auto eth1
iface eth1 inet manual
    up ip address add 0/0 dev $IFACE
    up ip link set $IFACE up
    down ip link set $IFACE down

# attached to br-pub on bgd-csf0
# used for br-ex public net (external/public network)
auto eth2
iface eth2 inet manual
    up ip link set $IFACE promisc on
    up ip address add 0/0 dev $IFACE
    up ip link set $IFACE up
    down ip link set $IFACE down
```

Change the names of the interfaces and their address ranges and invoke the **service networking restart** command.

Make sure that the “controller” can be resolved in /etc/hosts. For example:

```
controller 10.0.0.62
```

Edit and add to /etc/sysctl.conf:

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

Invoke **sysctl -p** to apply these changes.

### **STEP 2: Install and configure Neutron packages**

The following command installs all the necessary packages:

```
apt-get -y install python-mysqldb neutron-dhcp-agent neutron-l3-agent neutron-plugin-  
openvswitch-agent
```

Modify the content of /etc/neutron/neutron.conf:

```
[DEFAULT]
state_path = /var/lib/neutron
lock_path = $state_path/lock
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
allow_overlapping_ips = True
control_exchange = neutron
rabbit_host = controller
rabbit_password = ADMIN_PASS
rabbit_port = 5672
rabbit_userid = guest
notification_driver = neutron.openstack.common.notifier.rabbit_notifier
[quotas]
[agent]
```

```

root_helper = sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = neutron
admin_password = ADMIN_PASS
signing_dir = $state_path/keystone-signing
[database]
connection = mysql://neutron:DB_PASSWORD@controller/neutron
[service_providers]
service_provider=LOADBALANCER:Haproxy:neutron.services.loadbalancer.drivers.haproxy.plugin_driver.HaproxyOnHostPluginDriver:default

```

The following lines are needed in /etc/neutron/dhcp\_agent.ini:

```

[DEFAULT]
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
use_namespaces = True

```

Similar configuration is used for L3 agent in /etc/neutron/l3\_agent.ini:

```

[DEFAULT]
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
external_network_bridge = br-ex

```

Modify /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini

Note that physnet2 maps to public network and physnet1 corresponds to data plane

```

[ovs]
tenant_network_type = vlan
network_vlan_ranges = physnet2,physnet1:1000:1100
integration_bridge = br-int
bridge_mappings = physnet2:br-ex,physnet1:br-eth1
[agent]
[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
[database]
connection = mysql://neutron:ADMIN_PASS@controller/neutron

```

VLAN range 1000:1100 should be allowed on the trunk between access switch and physical port on CSF0 (br-data).

### **STEP 2: Configure OpenVSwitch for Neutron**

The following commands will create OVS switches used by Neutron:

```

ovs-vsctl add-br br-int
ovs-vsctl add-br br-ex
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-ex eth2
ovs-vsctl add-port br-eth1 eth1

```

Restart all three agents:

```
service neutron-plugin-openvswitch-agent restart
service neutron-dhcp-agent restart
service neutron-l3-agent restart
```

Although not necessary, it is highly recommended that you reboot the server manually at this point.

### 3.2.2 Gateway server CSF1

This server is dedicated to host gateway VMs which provide access into testbeds default management network [see 3.2.1.1]. It also hosts the TaaS gateway VM which provides access into infrastructure private control network used to manage all equipment and for service orchestration purposes [see 3.2.2.1]).

#### **STEP 0: Install minimal software requirements**

See section 3.1.2 for installing the common packages

#### **STEP 1: Install Common node**

CSF1 is actually a default Common compute node from the OpenStack point of view, but it is dedicated by TaaS service to host GW VMs and orchestration software is configured to not to lunch user's host VMs here. New GW VM is launched by TaaS orchestration software for each project when new one is created.

The TaaS GW is hosted on this server and is lunched directly by local libvirt (not by OpenStack). This VM will be described in separate chapter [3.2.2.1].

Requirements and installation process are the same as normal compute node with some minor changes in configuration of networking. To install the CSF1 just follow the process of installing the Common node [3.3.1] and **skip** the VC-support part.

#### **STEP 2: Configure OpenVSWitch**

That means not to create and run VC-support script. Instead, you need to create ovs bridges br-ctl and br-pub and map one free physical interface to each.

```
ovs-vsctl add-br br-pub
ovs-vsctl add-br br-ctl
ovs-vsctl add-port br-pub eth4
ovs-vsctl add-port br-pub eth5
```

Bridge br-ctl will be used by TaaS GW to access the private control network, br-pub will be used by all GW VMs to reach public internet. There will be also br-int (described in compute node installation procedure) which will serve to project GW VMs to reach testbed's inner management network.

To enable automated provisioning of user's GW VMs and its proper connecting to public network, following xml file named "testpub.xml" is needed to be created and stored in "/home/taas/".

```
cd /home/taas
vim testpub.xml
```

This is the content of the file:

```
<interface type='bridge'>
<source bridge='br-pub'/>
<virtualport type='openvswitch'/>
<model type='virtio'/>
</interface>
```

#### 3.2.2.1 *GTS service GW CSF1.0*

Description of function, network connectivity demands, minimal performance ...., software installed, configurations ... --- ME

#### 3.2.2.2 *GTS users GW CSF1.x*

$x \in \{1...n\}$

Description of function, network connectivity demands, minimal performance ...., software installed, configurations ... --- ME

### 3.2.3 Storage server CSF2

Beside gateway and VPN server roles, UAG instance provides a CIFS network file system which is automatically mounted on each VM inside the project. The actual disk space for these shares comes from a Cinder agent installed on CSF2. The shared file system is not exposed to the Internet for security reasons and the external access is only possible through VPN.

The automated ISO installer for CSF2 will also be available in the near future.

#### **STEP 0: Install minimal software requirements**

See section 3.1.2 for installing the common packages

#### **STEP 1: Install and configure Storage server**

The Storage server CSF2 is based on the OpenStack Cinder service. One of the main advantages of using Cinder as backend for network file system feature is horizontal scalability; the capacity extension is just a matter of adding more CSF2 nodes. The only special requirement during the installation is to allocate the "cinder-volumes" logical volume group and leave about 10GB for root and swap partitions.

The following command installs all the necessary packages:

```
apt-get -y install ifenslave cinder-volume python-mysqldb ethtool
```

The first step when creating 802.3ad LACP interface is to load bonding module:

```
stop networking
echo bonding >> /etc/modules
```

Fabio Farina 31/3/2015 17:33

**Comment [3]:** Missing sections. How can we get contribution here?

```
modprobe bonding
```

Change the network configuration as following (this example assumes only two Ethernet ports):

```
#eth0 is manually configured, and slave to the "bond0" bonded NIC
auto eth0
iface eth0 inet manual
bond-master bond0

#eth1 ditto, thus creating a 2-link bond.
auto eth1
iface eth1 inet manual
bond-master bond0

# bond0 is the bonded NIC and can be used like any other normal NIC.
# bond0 is configured using static network information.
auto bond0
iface bond0 inet static
address 192.168.1.10
gateway 192.168.1.1
netmask 255.255.255.0
# bond0 uses standard IEEE 802.3ad LACP bonding protocol
bond-mode 4
bond-miimon 100
bond-lacp-rate 0
bond-slaves eth0 eth1
```

Apply these changes by starting the networking service again:

```
start networking
```

To check the status of bond interface use **ifconfig bond0** and **more /proc/net/bonding/bond0** commands.

Make sure that the "controller" can be resolved in /etc/hosts. For example:

```
controller 10.0.0.62
```

The [filter:authtoken] section in /etc/cinder/api-paste should contain:

```
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000
admin_tenant_name = service
admin_user = cinder
admin_password = ADMIN_PASS
```

The content of /etc/cinder/cinder.conf should be:

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_config = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
```

```
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes

rpc_backend = cinder.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = ADMIN_PASS

[database]
connection = mysql://cinder:DB_PASSWORD@controller/cinde
```

Restart cinder-volume and iSCSI target service:

```
service cinder-volume restart
service tgt restart
```

Although not necessary, it is highly recommended that you reboot the server manually at this point.

BREAK: Inside the UAG VM

As a part of UAG instance creation, GTS core will create and attach the Cinder with the following commands or equivalent API calls:

```
cinder create --display_name <VOLUME_NAME> 500
nova volume-attach <UAG_ID> <VOLUME_ID> auto
```

The volume will stay attached to the VM independently of reboots and power offs.

Before destroying UAG instance, GTS core will detach the Cinder volume and delete it using following two commands or equivalent API calls:

```
nova volume-detach <UAG_ID> <VOLUME_ID>
cinder delete <VOLUME_NAME>
```

The first time the UAG VM is accessed by the GTS application through SSH on the public interface, it gains root privileges and invoke the /root/create-share.sh script:

```
/root/create-share.sh
```

The script, shown in a following box, take care of connecting the Cinder volume to the VM file system, exposing the storage as a network CIFS endpoint to the testbed resources.

In particular, to support the shared file system the UAG template has been modified as follows:

- Installation of **samba** and **lvm2** packages
- Modifying SAMBA configuration in /etc/samba/smb.conf
- Adding /root/create-share.sh script

Packages can be installed manually with the following two commands:

```
apt-get update
apt-get install lvm2 samba
```

Fabio Farina 31/3/2015 16:17

**Comment [4]:** Where this script comes from?  
Missing URL

The following lines in /etc/smb.conf should be modified:

```
...
interfaces = eth0
...
bind interfaces only = yes
...
[share]
comment = Ubuntu File Server Share
path = /srv/samba/share
browsable = yes
guest ok = yes
read only = no
create mask = 0755
```

The create-share.sh script should be placed under /root directory:

```
#!/bin/bash

DEVICE="/dev/vdc"
MP="/srv/samba/share"

/sbin/pvcreate $DEVICE
/sbin/vgcreate samba-vg $DEVICE
/sbin/lvcreate --name samba-lv --extents 100%FREE samba-vg
/sbin/mkfs.ext4 /dev/samba-vg/samba-lv

/bin/mkdir -p $MP
/bin/chown nobody.nogroup /srv/samba/share/

/bin/mount $DEVICE $MP

UUID=`blkid -o export /dev/samba-vg/samba-lv | /usr/bin/head -n 1`

echo "$UUID $MP ext4 defaults 0 0" >> /etc/fstab

/usr/sbin/service smbd restart
/usr/sbin/service nmbd restart
```

The script also needs root owner and execute permissions:

```
chown root:root /root/create-share.sh
chmod u+x /root/create-share.sh
```

To boot the shared file system automatically, the end-user VM templates need additional entry in /etc/fstab:

```
//172.16.0.254/share /mnt/share cifs defaults,_netdev,uid=root,gid=root,forceuid,forcegid 0 0
```

Mount point can be created with:

```
mkdir -p /mnt/share
```

## 3.3 Common node

### 3.3.1 Compute node

The purpose of compute node is to act as a hypervisor for end-user VMs.

#### **STEP 0: LVM layout during installation and minimal software requirements**

GTS core does not support persistent block level storage but this feature might be added in the future. For this reason, 10% of the effective disk space must be allocated for the “cinder-volumes” volume group. This group needs to be created during the Ubuntu installation and should not host any manually created logical volumes.

As usual, see section 3.1.2 for installing the common packages

#### **STEP 1: Networking configuration**

Interfaces should be identified using **lshw -class network | less** command and renamed in `/etc/udev/rules.d/70-persistent-net.rules` as following:

- eth0 is the first port on embedded NIC (control plane)
- eth1 is the second port on embedded NIC (data plane)
- eth2 and eth3 correspond to ports on 10G NIC (reserved for the future use)
- eth4 – eth15 are assigned to 4 NICs (used for virtual circuit provisioning).

After saving changes in `70-persistent-net.rules` **reboot** the system. The following `udev.awk` script can be used to automate renaming procedure:

```
#!/usr/bin/awk -f

BEGIN{
    RS="*";
    FS="\n";
    bcm5720_counter=0;
    tenG_counter=2;
    bcm5719_counter=4;
}

# Explicit conditions avoid non-standard records (e.g. blanks before first *)
# Only eth0 and eth1 are sorted
{

    if (index($0,"BCM5720"))
    {
        eth[bcm5720_counter++]=substr($9,16);
        if (bcm5720_counter == 2 && eth[1] < eth[0])
        {
            temp=eth[0];
            eth[0]=eth[1];
            eth[1]=temp;
        }
    }

    if (index($0,"BCM5719"))
```



```

{
    eth[bcm5719_counter++]= substr($9,16);
}

if (index($0,"10 Gigabit"))
{
    eth[tenG_counter++]= substr($9,16);
}

}

# awk -F " " 'BEGIN{OFS=" "} { if (index($0,"00:1a:a0:cc:5a:26")) {$8=" NAME=\"eth33\""} {print} }'
70-persistent-net.rules
END{
    rules_path="/etc/udev/rules.d/70-persistent-net.rules";
    # The loop is an overkill but it does the job...
    for (i= 0; i <= 15; i++)
    {
        ethX="\ " NAME=\\\"eth\"i\"\\\"\\\"";
        system("cp " rules_path " tmp.rules");
        system("awk -F \" \" 'BEGIN{OFS=\\\" \" } { if (index($0,\\\"eth\"i\\\"\\\")) {$8=\\\"ethX\\\"} {print} }' tmp.rules >
" rules_path);
        system("rm tmp.rules");
    }
}

```

Add execute permissions and invoke the script in the following manner:

```

chmod u+x udev.awk
lshw -class network > lshw.txt
./udev.awk lshw.txt

```

Do not try to pipe the output of **lshw** command directly to the script.

Edit /etc/network/interfaces:

```

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 10.0.0.18
    netmask 255.255.255.192
    gateway 10.0.0.56
    # dns-* options are implemented by the resolvconf package, if installed
    dns-nameservers 8.8.8.8

auto eth1
iface eth1 inet manual
up ip link set dev eth1 up
up ip link set dev eth1 promisc on

auto eth4

```

```

iface eth4 inet manual
up ip link set dev eth4 up
up ip link set dev eth4 promisc on

auto eth5
iface eth5 inet manual
up ip link set dev eth5 up
up ip link set dev eth5 promisc on

# Same for eth6 - eth15
....

```

After editing invoke the **service networking restart** command.

### STEP 2: Install and configure Compute

The following command installs the needed packages:

```

apt-get -y install qemu-kvm libvirt-bin openvswitch-switch python-mysqldb nova-compute-kvm
python-guestfs neutron-plugin-openvswitch-agent cinder-volume ethtool

```

The following command solves some bug (please see OpenStack documentation for details):

```

chmod 0644 /boot/vmlinuz*

```

Make sure that the “controller” can be resolved in /etc/hosts. For example:

```

controller 10.0.0.62

```

### STEP 3: Configure OpenVSwitch

Create the OVS instances required by Neutron L2 agent:

```

ovs-vsctl add-br br-int
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1

```

The purpose of these software switches is to provide data plane connectivity to end-user VMs.

Create the script /root/VC-support.sh with the following content:

```

#!/bin/bash

for i in `seq 4 15`;
do
    ovs-vsctl add-br br$i
    ovs-vsctl add-port br$i eth$i
    cd /home/taas
    echo "<interface type='bridge'>" > testeth$i.xml
    MAC=`ifconfig eth$i | grep -o -E '([[:xdigit:]]{1,2}:){5}([[:xdigit:]]{1,2})'`
    LOW=`echo $MAC | awk -F ":" '{print $4 ":" $5 ":" $6}'`
    echo " <mac address='52:54:00:$LOW'/>" >> testeth$i.xml
    echo " <source bridge='br$i'/>" >> testeth$i.xml
    echo " <virtualport type='openvswitch'/>" >> testeth$i.xml
    echo " <target dev='testnet$i'/>" >> testeth$i.xml

```

```
echo "</interface>" >> testeth$i.xml
done
```

and invoke it using **bash /root/VC-support.sh** command. This will create libvirt xml files that are needed for virtual circuit provisioning.

#### **STEP 4: Configure OpenStack plugins**

Modify /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini (physnet1 corresponds to the data plane):

```
[ovs]
tenant_network_type = vlan
network_vlan_ranges = physnet1:1000:1100
[agent]
[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

VLAN range 1000:1100 should be on trunk between the switch and eth1 on compute nodes.

Modify the content of /etc/neutron/neutron.conf:

```
[DEFAULT]
state_path = /var/lib/neutron
lock_path = $state_path/lock
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
control_exchange = neutron
rabbit_host = controller
rabbit_password = ADMIN_PASS
rabbit_port = 5672
rabbit_userid = guest
notification_driver = neutron.openstack.common.notifier.rabbit_notifier
[quotas]
[agent]
root_helper = sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = neutron
admin_password = ADMIN_PASS
signing_dir = $state_path/keystone-signing
[database]
connection = mysql://neutron:DB_PASSWORD@controller/neutron
[service_providers]
service_provider=LOADBALANCER:Haproxy:neutron.services.loadbalancer.drivers.haproxy.plugin_driver.HaproxyOnHostPluginDriver:default
```

Restart L2 agent:

```
service neutron-plugin-openvswitch-agent restart
```

Replace the content of /etc/nova/nova.conf with the following:

```
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:DB_PASSWORD@controller/nova

[DEFAULT]

# General
verbose=True
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova

my_ip=IP_ADDRESS_OF_THIS_HOST
rabbit_host = controller
rabbit_password = ADMIN_PASS
rabbit_userid = guest
rabbit_port = 5672
auth_strategy=keystone
rpc_backend = nova.rpc.impl_kombu

# Networking
network_api_class=nova.network.neutronv2.api.API
neutron_admin_username=neutron
neutron_admin_password= ADMIN_PASS
neutron_admin_auth_url=http://controller:35357/v2.0/
neutron_auth_strategy=keystone
neutron_admin_tenant_name=service
# change to network depending on where neutron-api package is installed:
neutron_url=http://controller:9696/
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver

# Security Groups
firewall_driver=nova.virt.firewall.NoopFirewallDriver
security_group_api=neutron

# Compute
compute_driver=libvirt.LibvirtDriver
connection_type=libvirt

# Cinder
volume_api_class=nova.volume.cinder.API

# Glance
glance_host=controller

# novnc
vnc_enabled=True
novncproxy_base_url=http://PUBLIC-IP-ADDRES-OF-GTS-CORE:6080/vnc_auto.html
vncserver_proxycient_address=IP_ADDRESS_OF_THIS_HOST
vncserver_listen=0.0.0.0
```

Restart nova-compute service:

```
service nova-compute restart
```

The [filter:authtoken] section in /etc/cinder/api-paste should contain:

```
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000
admin_tenant_name = service
admin_user = cinder
admin_password = ADMIN_PASS
```

The content of /etc/cinder/cinder.conf should be:

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_config = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes

rpc_backend = cinder.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = ADMIN_PASS

[database]
connection = mysql://cinder:DB_PASSWORD@controller/cinde
```

Restart cinder-volume and iSCSI target service:

```
service cinder-volume restart
service tgt restart
```

Although not necessary, it is highly recommended that you reboot the server manually at this point.

### 3.3.2 Data plane router

GTS data plane router is the key part of testbed service. It is used for realizing every single connection between resources within the service. Basic requirements are:

- Sufficient number interfaces to make as many connections as needed between devices in the pod
- 10GE interfaces for core inter-pod connections

- As wide as possible support of flexible Ethernet services. Following features are mandatory:
  - Flexible vlan tagging (Q-in-Q, vlan mapping, Ethernet circuit cross connection)
  - MPLS-TE, RSVP and BGP signalled L2 services, VPLS
- Capability to manage configuration by third party software

In case of GTS Juniper MX-80-48T has been chosen. Currently provisioned operation system version is JUNOS 12.3R6.6 built 2014-03-13.

The description of default configuration of GTS data plane router will follow. First important part is the management access via control network. This configuration has to be done before deploying the router on site.

- Users
  - admin; class super-user; password
  - opennsa; class super-user; auth ssh-rsa (This user will be used by OpenNSA software to provision virtual circuits.)
  - taas; class super-user; auth ssh-rsa (Optional, but recommended. This is common user configured on all deployed devices.)
- services: ssh; telnet; ntp
- MNG ip interface fxp0 to proper subnet and IP
- Static route for def gw for proper subnet

Following configuration is needed to have GTS router fully deployed in default configuration and ready to operate.

- Configure IP and MPLS on 10GE interfaces (data plane core links)
- Configure VPLS tagged interface ge-1/0/0; vlans 1000-1100 (For neutron networks, trunk to the Compute0. In pods, where CSF servers are placed, interfaces which leads to this servers have to be configured in this VPLS as well.)
- Init interfaces ge-1/0/1 – ge-1/1/0 for links to the compute0 and check cabling between compute node
- Optional: repeat previous for next compute nodes if any, or for HP switch links (no common trunk to the HP in current version of TaaS arch.)
- Configure Lo0 interface by IP address
- Configure AS number 65250
- Configure RSVP interface all (exclude ifaces Lo0.0 and fxp0.0)
- Configure MPLS for 10GE ifaces
- Configure mpls LSPs to the rest of MXes for NEUTRON VPLS
- Configure internal BGP
  - Local address (loopback)
  - Advertise-peer-as
  - Family I2VPN auto discovery; signalling
  - Local-as 65250
  - And RR neighbour which is CPH MX80 10.0.16.3 (keep in mind to configure all neighbours on RR)

- Configure OSPF for 10GE links – area 0.0.0.0
- Configure VPLS NEUTRON
  - Vlan id all
  - Interface ge-1/0/0.0 [ge-1/1/1.0 ...]
  - For CSF node ifaces to CSF servers
  - Route distinguisher 65250:1; vrf-target target:65250:1
  - No-tunnel-services
  - Site identifier
- SAVE clean config to file
  - Name of the def. file is “sa2\_init\_clean0.cfg”. You can name the file as you wish or even create more versions. File will be saved into the users home dir on internal memory.
  - Copy actual file to the /var/tmp/configs/ and name it by name “sa2\_init\_clean0.cfg”. (During maintenance reconfiguration will be loaded file /var/tmp/configs/ sa2\_init\_clean0.cfg and this is hardcoded in the redeployment script in current version)

### 3.3.3 OpenFlow fabric

Description of function, network connectivity demands, software installed, configurations ... --- ME

### 3.3.4 Rack switch

Description of function, network connectivity demands, software installed, configurations ... - ME

Fabio Farina 31/3/2015 17:36

Comment [5]: Missing paragraphs

## Appendix I: Service management quick reference

### A - Start and stop GTS core services on CSF0.1

- **GTS core Karaf first run**

In order to run the software for the first time, you should login as taas user and perform presented commands from the folder /home/taas/apache-karaf-2.3.3/

Start derby:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ nohup ../db-derby-10.10.1.1-  
bin/bin/startNetworkServer &
```

Start karaf:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ./bin/start
```

Start tomcat:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ../apache-tomcat-  
7.0.54/bin/catalina.sh jpda start
```

- **Karaf restart**

If there is a need for restart, the infrastructure user should perform the following steps in order. User should be logged in as taas and perform presented commands from the folder /home/taas/apache-karaf-2.3.3/

- **Stopping GTS core**

Stop Karaf:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ./bin/stop
```

It could happen that karaf will not stop so easily. It's a good habit to check if it's process is really ended. If not, user is forced to kill that process. To do so, first find the process number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ps ax | grep karaf.jar | grep  
-v grep | awk '{print $1}'
```

If no number is displayed it means that karaf has ended correctly. If a number is displayed, that means that user has to kill the process with that number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ sudo kill -9 NUMBER
```

where *NUMBER* is the number given as a result of the previous command.

Stop derby:



To stop derby user has to proceed with similar steps as with killing karaf process. First, user has to specify the derby process number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ps ax | grep  
NetworkServerControl | grep -v grep | awk '{print $1}'
```

then kill process with a given number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ sudo kill -9 NUMBER
```

where *NUMBER* is the number given as a result of the previous command.

Stop tomcat:

In order to stop tomcat user has to proceed with similar steps as with killing derby process. First, user has to specify the tomcat process number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ ps ax | grep tomcat | grep -v  
grep | awk '{print $1}'
```

then kill process with the given number:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ sudo kill -9 NUMBER
```

where *NUMBER* is the number given as a result of the previous command.

Delete three Karaf folders:

```
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ rm -rf taas/  
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ rm -rf taasDB/  
taas@TaasCore:/home/taas/apache-karaf-2.3.3$ rm -rf data/
```

## B - Restart OpenStack services on CSF0.2

```
service keystone restart  
service glance-registry restart  
service glance-api restart  
service nova-api restart  
service nova-cert restart  
service nova-consoleauth restart  
service nova-scheduler restart  
service nova-conductor restart  
service cinder-scheduler restart  
service cinder-api restart  
service neutron-server restart
```

## C - Restart services on CSF0.3

```
service neutron-plugin-openvswitch-agent restart  
service neutron-dhcp-agent restart  
service neutron-l3-agent restart
```

## Appendix II – OpenFlow switch configuration at GTS pods

This appendix discusses how OpenFlow fabrics have been deployed and configured on the GÉANT pods across the Europe.

### A - Installation of the Device

Please perform the following steps:

- Mount the switch in the 19-inch rack
- Ground the device
- Install the fan trays
- Connect the power supply and cords
- Connect the console cable
- Verify the installation and power on switch
- Software upgrades and access to Boot ROM menus
- Troubleshoot the device

#### A.1 Mount the switch in the 19-inch rack

Slide the switch to its location within the rack and attach it to the rack with the mounting brackets (for more information see Ref. **Error! Reference source not found.**, pp. 11-18).

#### A.2 Ground the device

After connecting the grounding cable to the switch, connect the end of the grounding cable to the grounding strip (for more information see Ref. **Error! Reference source not found.**, p. 17).

#### A.3 Connect the power supply and cords

Unpack the power supply units (it should be two units) and check that the power supply model is correct. Power Supply: JC680A HP 650W AC Power Supply for HP 58X0AF.

Slide power supplies slowly into the slots and check that both power adapters are plugged in correctly.

Connect the female connector of the AC power cord supplied with the power supply and the other end of the cord to an AC power outlet. There is a cable tie you can use to secure the power cord to the handle of the power supply (for more information see Ref. **Error! Reference source not found.**, p. 21).

#### A.4 Install the fan trays

**CAUTION:** The following details must be observed at all times in order not to void the warranty of the switch:

- The HP 5900 requires two same direction air flow fan trays to function properly.
- Do not operate the switch with only one working fan tray for more than 24 hours.
- Remove a failed fan tray only when you are ready to replace it.
- You must not operate the switch without any fan tray for more than 2 minutes.
- The switch must be operated inside of the temperature range 0°C to 45°C (32°F to 113°F) degrees.

To install the fan trays, slide the trays slowly into the slots and check that they are plugged in correctly. There is a screw on each tray that must be fastened to secure the tray to the chassis (for more information see Ref. **Error! Reference source not found.**, p. 20).

#### A.5 Connect the console cable

The first time the switch is accessed, log in must be to the CLI via the console port (6) of the switch (rear panel). See Figure1.

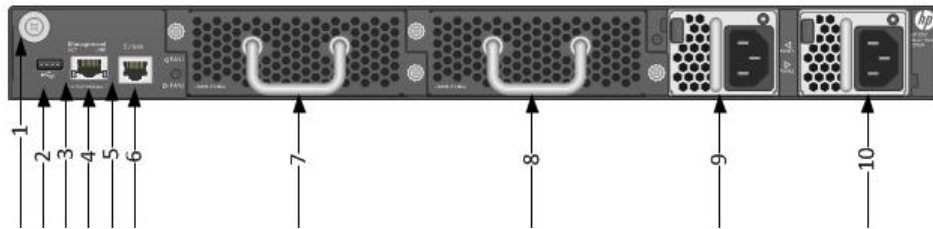


Figure 6 HP 5900 rear panel description

- (1) Ground point
- (2) USB port
- (3) ACT LED
- (4) Management Ethernet port
- (5) LINK LED
- (6) Console port
- (7) Fan tray 1
- (8) Fan tray 2
- (9) Power supply 1
- (10) Power supply 2

- Connect the DB-9 female connector of the console cable to the serial port of the PC.
- Connect the RJ-45 connector of the console cable to the console port of the switch, as point (6) in Figure 1.
- Turn on the PC
- On the PC, launch the terminal emulation program, create a connection that uses the serial port connected to the device, and set the port properties so the port properties match the following console port default settings (for more information see Ref. **Error! Reference source not found.**, p. 26):
  - Bits per second – 9600 bps
  - Flow Control - None
  - Parity – None
  - Stop bits – 1
  - Data bits – 8
  - Emulation – VT100

## A.6 Verify the installation and power on switch

Verify that everything has been setup properly before you power on the switch. Make sure that

- The power cord is connected to the switch
- The power voltage is correct
- The console cable is connected
- The PC and configuration terminal has started and has been set to the default settings as described in [Error! Reference source not found.]
- Then power on the switch.

## A.7 Software upgrades via USB

The HP5900 series have a USB port on them that you can use to transfer files.

- Download the upgrade software image to the USB flash drive and plug the flash drive into the switch
- Logging in through the console port for the first device access
- At the default user view prompts <HP>, enter commands to configure the device or view the running status of the device. To get help, enter ?
- Copy the file “5900\_5920-CMW710-R2307.ipe” to the flash memory of the switch
  - <HP>copy usb0:/5900\_5920-CMW710-R2307.ipe flash:/  
Copy usb0:/5900\_5920-CMW710-R2307.ipe to flash:/5900\_5920-CMW710-R2307.ipe?[Y/N]:Y  
Copying file usb0:/5900\_5920-CMW710-R2307.ipe to flash:/5900\_5920-CMW710-R2307.ipe.....
- When the copy has completed you will need to reload the switch.
  - <HP>boot-loader file flash:/5900\_5920-CMW710-R2307.ipe slot 1 main  
Images in IPE:  
5900\_5920-cmw710-boot-r2307.bin  
5900\_5920-cmw710-system-r2307.bin  
This command will set the main startup software images. Continue? [Y/N]:Y  
Add images to target slot.  
The specified file list will be used as the main startup software images at the next reboot on slot 1.  
<HP>reboot  
  
Start to check configuration with next startup configuration file, please wait.....DONE!  
This command will reboot the device. Continue? [Y/N]:Y

Wait until you see the message “System is starting...”

## A.8 Troubleshoot the device

Please check Ref. Error! Reference source not found., pp. 38-39 in case you experience any of the following problems:

- Power supply failure
- Fan failure: HP 5900 Switch Series –Error “FAN\_DIRECTION\_NOT\_PREFERRED”
  - [HP] fan prefer-direction slot 1 port-to-power  
[HP]%Jan 12 00:22:59:911 2011 HP DEV/5/FAN\_RECOVERED: Fan 1 recovered.  
%Jan 12 00:22:59:912 2011 HP DEV/5/FAN\_RECOVERED: Fan 2 recovered.

```
[HP]display fan
Slot 1
  FAN 1
  State : Normal
  Wind Direction :Port-to-Power
  Prefer Wind Direction :Port-to-Power
  FAN 2
  State : Normal
  Wind Direction :Port-to-Power
  Prefer Wind Direction :Port-to-Power
[HP]
```

- Problems with the configuration terminal/environment

## B - Basic Configuration

The following steps have to be completed in order to baseline the switch for operation within TaaS.

- Configure SSH login on the device
- Configure IP address and subnet information
- Configure default user (with restricted access rights)
- Configure passwords

### B.1 Configure multiple SSH login on the device

```
<HP>system-view
[HP]ssh server enable
[HP]line vty 0 40
[HP-line-vty0-40]authentication-mode password
[HP-line-vty0-40]authentication-mode scheme
[HP-line-vty0-40]protocol inbound ssh
[HP-line-vty0-40]quit
[HP]
```

Michal Hažlinský 15/4/2015 15:16

**Comment [6]:** It is needed to generate new pair of keys when enabling SSH on HP switches ... Steps for that are missing here .. I and I think .. It is also needed to enable ssh for each user .. (not sure if necessary or optional)

### B.2 Configure IP address and subnet information

```
<HP>system-view
[HP]sysname name [LON,CPH,AMS,CM ...]

<HP> system-view
[HP] ip vpn-instance MNG
[HP-vpn-instance-MNG]description CTL NET
[HP-vpn-instance-MNG]address-family ipv4
[HP-vpn-instance-MNG]quit

[HP] interface vlan-interface 1
[HP-Vlan-interface1]ip binding vpn-instance MNG
[HP-Vlan-interface1] ip address A.B.C.0 255.255.255.192
[HP-Vlan-interface1] quit

[HP]ip route-static vpn-instance MNG 0.0.0.0 0 10.0.0.X permanent description CTL
ROUTE
```

### B.3 Configure manager user “admin” und password

```
<HP>system-view
[HP] local user admin class manage
New local user added.
[HP-luser-manage-admin]password simple [you cause]
[HP-luser-manage-admin]service-type ssh
[HP-luser-manage-admin]service-type telnet
[HP-luser-manage-admin]service-type terminal
[HP-luser-manage-admin]service-type ftp
```

```
[HP-luser-manage-admin]authorization-attribute user-role network-admin
[HP-luser-manage-admin]authorization-attribute user-role network-operator
[HP-luser-manage-admin]quit
```

## B.4 Cable Map

| HP5900 cabling map      |                    |           |                           |
|-------------------------|--------------------|-----------|---------------------------|
| Source Port             | Destination Device | Port      | Purpose                   |
| 1                       | MX80               | ge-1/2/2  | VX0                       |
| 2                       | MX80               | ge-1/2/3  | VX0                       |
| 3                       | MX80               | ge-1/2/4  | VX0                       |
| 4                       | MX80               | ge-1/2/5  | VX0                       |
| 5                       | MX80               | ge-1/2/6  | VX0                       |
| 6                       | MX80               | ge-1/2/7  | VX0                       |
| 7                       | MX80               | ge-1/2/8  | VX1                       |
| 8                       | MX80               | ge-1/2/9  | VX1                       |
| 9                       | MX80               | ge-1/2/10 | VX1                       |
| 10                      | MX80               | ge-1/2/11 | VX1                       |
| 11                      | MX80               | ge-1/3/0  | VX1                       |
| 12                      | MX80               | ge-1/3/1  | VX1                       |
| 13                      | MX80               | ge-1/3/2  | VX2                       |
| 14                      | MX80               | ge-1/3/3  | VX2                       |
| 15                      | MX80               | ge-1/3/4  | VX2                       |
| 16                      | MX80               | ge-1/3/5  | VX2                       |
| 17                      | MX80               | ge-1/3/6  | VX2                       |
| 18                      | MX80               | ge-1/3/7  | VX2                       |
| Reserved for future use | .                  | .         | .                         |
| Reserved for future use | .                  | .         | .                         |
| Reserved for future use | .                  | .         | .                         |
| Reserved for future use | .                  | .         | .                         |
| 48                      | EX4200             | port 20   | Control plane mgmt access |

## References

- [1] Hewlett Packard Development Company, HP 5920&5900 Switch Series Installation Guide, Part number: 5998- 2852, Document version: 6W101-20130123, available from [http://h20628.www2.hp.com/km-ext/kmcsdirect/emr\\_na-c03189333-4.pdf](http://h20628.www2.hp.com/km-ext/kmcsdirect/emr_na-c03189333-4.pdf)
- [2] Manuals for HP Switch 5900 can be downloaded from [http://h20565.www2.hp.com/portal/site/hpsc/template.PAGE/public/psi/manualsResults/?javax.portlet.begCacheTok=com.vignette.cachetoken&javax.portlet.endCacheTok=com.vignette.cachetoken&javax.portlet.prp\\_e97ce00a6f76436cc859bfdeb053ce01=wsrpnavigationalState%3Daction%253Dmanualslist%257Cviewall%253Dtrue%257Clang%253Den&javax.portlet.tpst=e97ce00a6f76436cc859bfdeb053ce01&sp4ts.oid=5221896&ac.admitted=1384269988038.876444892.492883150](http://h20565.www2.hp.com/portal/site/hpsc/template.PAGE/public/psi/manualsResults/?javax.portlet.begCacheTok=com.vignette.cachetoken&javax.portlet.endCacheTok=com.vignette.cachetoken&javax.portlet.prp_e97ce00a6f76436cc859bfdeb053ce01=wsrpnavigationalState%3Daction%253Dmanualslist%257Cviewall%253Dtrue%257Clang%253Den&javax.portlet.tpst=e97ce00a6f76436cc859bfdeb053ce01&sp4ts.oid=5221896&ac.admitted=1384269988038.876444892.492883150)

