



GÉANT Testbed Service (GTS) User and Resource Guide

Version 4.1

June 2017

Document Code: <GN-16-001>

Authors:

© GEANT Limited on behalf of the GN4-2 project.

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

Contents

1	Introduction	4
2	Testbed topology and resources	5
3	How to get started	6
3.1	Access to GTS	6
3.2	Login to the GUI	6
4	Working in the testbed environment	9
4.1	Defining a testbed with DSL	10
4.2	Reserving testbed resources	13
4.3	Activating testbed resources	15
4.4	Setting reservation windows	17
4.5	Deactivating testbed resources	18
4.6	Releasing testbed resources	19
4.7	Querying details	20
4.8	Internet Access Gateway (IAGW)	23
4.8.1	Internet connectivity	23
4.8.2	VPN access	23
4.8.3	Persistent Shared Project Folder	31
4.9	Example: Working with GTS OpenFlow switches (VSI)	32
5	Help and Support	39
6	Introduction to Domain Specific Language	40
6.1	DSL in GTS	40
6.2	Quick Start	40
	Appendix I: Resource Guide	44
I.	Composite types	44
II.	Host	45
III.	Link	47
IV.	VSI	48
V.	External Domain	50
VI.	Bare Metal Server (BMS)	52
i.	<i>Introduction to Bare Metal Servers</i>	52
ii.	<i>PERC H710 Mini RAID configuration</i>	54

<i>iii.</i>	<i>PERC H730 Mini RAID configuration</i>	58
<i>iv.</i>	<i>Installation of OS on BMS</i>	59
VII.	BNF Grammar	61
Appendix II: Additional examples		62
I.	Examples: One host	62
II.	Example: Two hosts linked together	63
III.	Example: Triangle between three locations	65
IV.	Example: Three triangles built using DSL code iterations	67
V.	Example: Two OpenFlow switches (VSI), each with two ports, one host and one separate controller	68
VI.	Example: One host directly connected with a Bare Metal Server	70

1 Introduction

Welcome to the new GÉANT Testbeds Service (GTS)! This new production service gives you the opportunity to setup your own customized experimental platform in order to be able to test novel concepts in networking and telecommunications.

This new service is made available in several releases and will grow further over time. This user guide describes features available in GTS Version 4.1.

The service allows you to set up an isolated customized packet-based testbed environment and enables you to conduct your particular experiment over resources in a real network without having to worry about impacting other testbeds or production services. After a short registration process a first time user has access to a pool of virtualized resources that can be programmed and reserved using a Domain Specific Lanmole (DSL) description or selected via a graphical user interface. An Internet Access Gateway (IAGW) provides the possibility to load software into the testbed. The system also allows the selection of an external interface as a resource whenever a testbed needs to be linked to external nodes.

The GTS facility is intended for use by researchers who are part of the GÉANT Community and its associated OpenCall projects [FAR-2014, NAE-2016, SZE-2014]. Other researchers will be provided access on a space available basis.

2 Testbed topology and resources

Currently, GTS allows you to select resources from the following geographical locations: Amsterdam, Hamburg, London, Madrid, Milan, Paris and Prague (see

Figure 1). All these locations are sitting on the GÉANT backbone network point of presences. Later on, it is expected that new resource locations will be available outside of GÉANT, in the NRENs or other peer networks' domains.

Available resources in Version 4 of GTS include:

- Hosts (VMs with data ports and integrated monitoring in the GTS GUI; implemented using OpenStack)
- Virtual circuits (Ethernet pipe with data ports; implemented using Network Service Interface (NSI) with 10 GE connectivity)
- Virtual Switch Instances (VSIs) (fully virtualized OpenFlow switch instances (OpenFlow Specification 1.3) with data ports)
- External domain interfaces
- Bare Metal Servers (BMSs) (single-tenant physical server)

The detailed description of the resources, their parameters and default attributes can be found in the Appendix I.

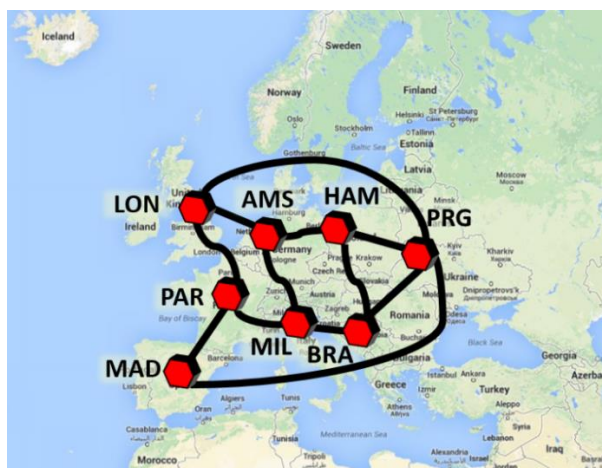


Figure 1: GTS physical infrastructure topology¹.

¹ Please note: As of GTS v4 Ljubljana is no longer available.

3 How to get started

This section describes the steps you need to complete until your testbed setup is ready for use.

3.1 Access to GTS

The GTS front-end web Graphical User Interface (GUI) is available at: <https://gts4.geant.net>. NOTE: Whenever you get the message “The service is currently undergoing maintenance, and may not be fully functional. Check back later!” it practically means that we are upgrading and verifying the software/hardware functions of the environment. The system may not be fully functional while the notice is displayed.

3.2 Login to the GUI

The GUI is the primary user interface of GTS but there are other means of accessing the testbed services. The GUI is built on a restful API in front of the GTS core software. In later versions of the software, the API will also be open for application designers and system integrators.

First time users have to register by clicking on ‘Register’ and then sending an email to gts-operations@lists.geant.org (Figure 2); users that are already registered can login with their user names and passwords as shown in Figure 3. When you first register, please describe briefly in the email which project name you would like to use, describe the project and list approximately what type of resources and what quantities you expect you will be using. Also please define your username and provide your real name, e-mail address. By creating or requesting an account you also agree to the Terms of Service of GTS.

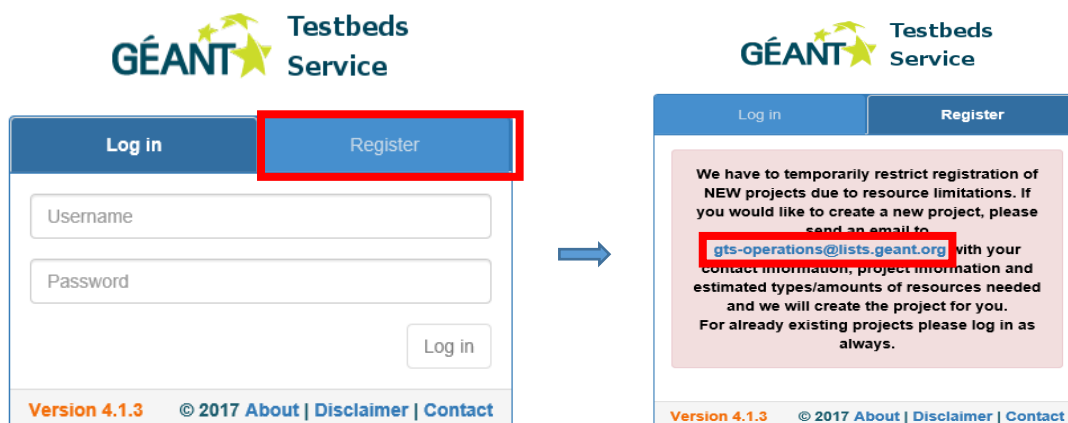


Figure 2: First-time registration

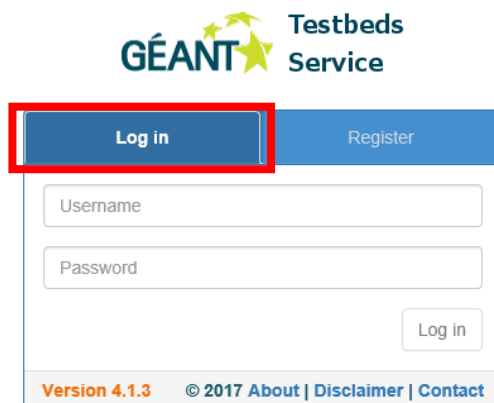


Figure 3: Login as registered user

Once you are logged in you have access to the VPN information by clicking on the blue information icon next to your project's name (Figure 4). Logout of your project using the dropdown menu below your project name (Figure 5). This dropdown menu also gives you access to your user profile and allows updates if needed (Figure 6).

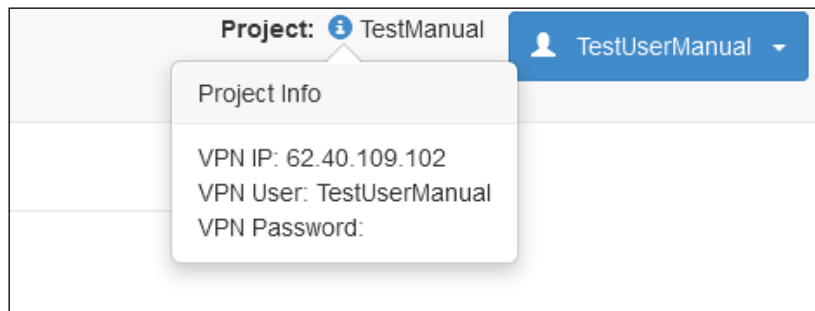


Figure 4: Project and VPN related information

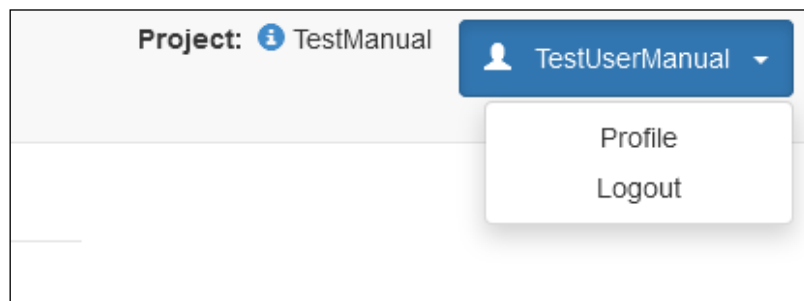


Figure 5: Log out of the project via the dropdown menu next to the project's name

In the current implementation of Version 4.1, new user registrations always end up in new projects. If you wish to add multiple users to your single (already existing) project, please contact the GTS service managers.

Edit User Details

Username

Name

e-mail

Password *(necessary only for password change)*

Confirm Password

Role

Project

Figure 6: Update or reset user profile

During a login process you may find a disclaimer that indicates that the testbed facility is currently under maintenance and its use may be limited (Figure 7).

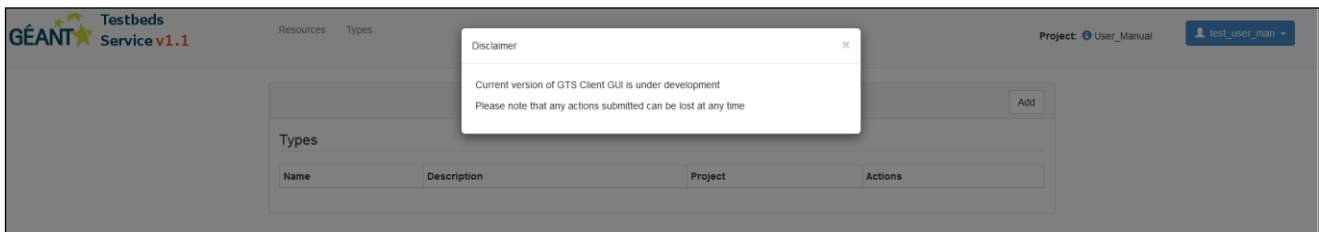


Figure 7: Disclaimer

4 Working in the testbed environment

The GTS GUI acts as an agent to the testbed resources and allows a user to build and monitor an isolated domain of interacting resources, i.e. a user's testbed is independent from other users' testbeds and can be operated without any impact on other testbed environments.

After logging in to GTS the user builds a testbed description for the Testbed Control Agent (TCA). In a next step, the testbed description document is fed to the Resource Manager (RM) which in turn parses the document, allocates resources, and sets up the testbed control plane. The testbed is then activated and the user controls it via the GUI.

For example, the user may decide to define router nodes at first and may then browse and select a set of PC resources to act as the end systems. Each of these resources has data plane ports defined as integral components of the resource. The user must then define the connectivity among these nodal resources by indicating how the various nodal resources are to be interconnected via these network links. At the end the user saves the testbed description to the project's repository and submits the testbed description to the Service Engine for processing. For some experiments it may be necessary to select virtual resources that are geographically apart or at certain locations. In such use cases users may select a resource's location from the list of available installations.

The GTS service engine analyzes the testbed description for any obvious errors or omissions, and finding nothing critical, the Service proceeds to locate and reserve the indicated resources. Note: this is an iterative process – the researcher may have only defined a partial set of resources and wishes to reserve these first, where upon the user will then define additional resources to be incorporated into the testbed. Upon successfully reserving all resources, the researcher requests the GTS Service Engine to instantiate the testbed. As resources are initialized and brought into service, control of those resources are passed to the researcher's own testbed control agent. Instantiating resources is complete, when the selected resources have been activated.

The user can program the desired testbed environments and reserve resources using a Domain Specific Language (DSL) description or select resources via the GUI (for more information on DSL please see Chapter 6). This reservation of resources also allows a user to specify start and end times during when these resources should be activated and made available for experimentation. Only when all requested resources are available the reservation process completes with success.

Reserved resources can then be **activated** by the user and at this time all resources will actually be allocated. At some time during an experiment a user may wish to **deactivate** the resources, but keep the reservation of resources valid for use at a later point in time. Once an experiment is finished and the project is no longer needed, the user should also **release** all resources so that resources can be made available to other projects. Resources can also be **queried** for status information.

4.1 Defining a testbed with DSL

After you have logged in as a registered user you are now able to upload Domain Specific Language (DSL) descriptor files that set up the type of experimental environment that you would like to have in your testbed (for more information on the DSL please see Chapter 6).

You can define your own resource types or classes with DSL code (for example: an OpenFlow switch connected to a controller). To add a DSL file, please click on “Types” (top navigation menu, Figure 8) and then select “Add” (Figure 9) which will open up a window (see Figure 10) where you can browse through your files to upload an already existing DSL file. Click on ‘Submit’ to finish your testbed description.

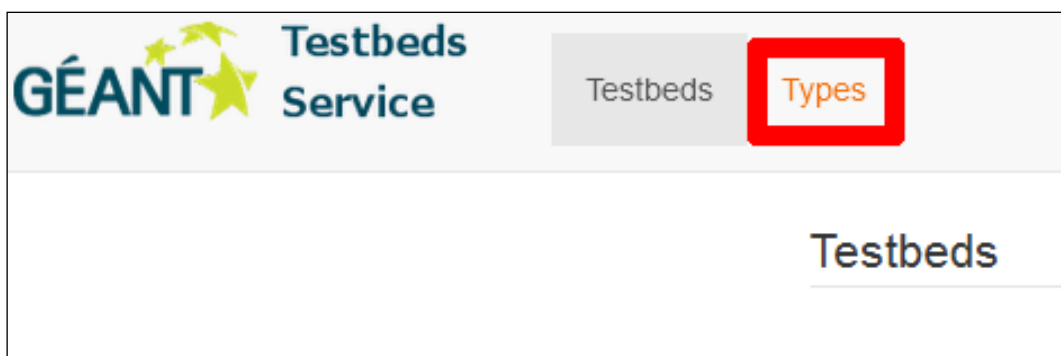


Figure 8: Select “Types”



Figure 9: Add “Type”

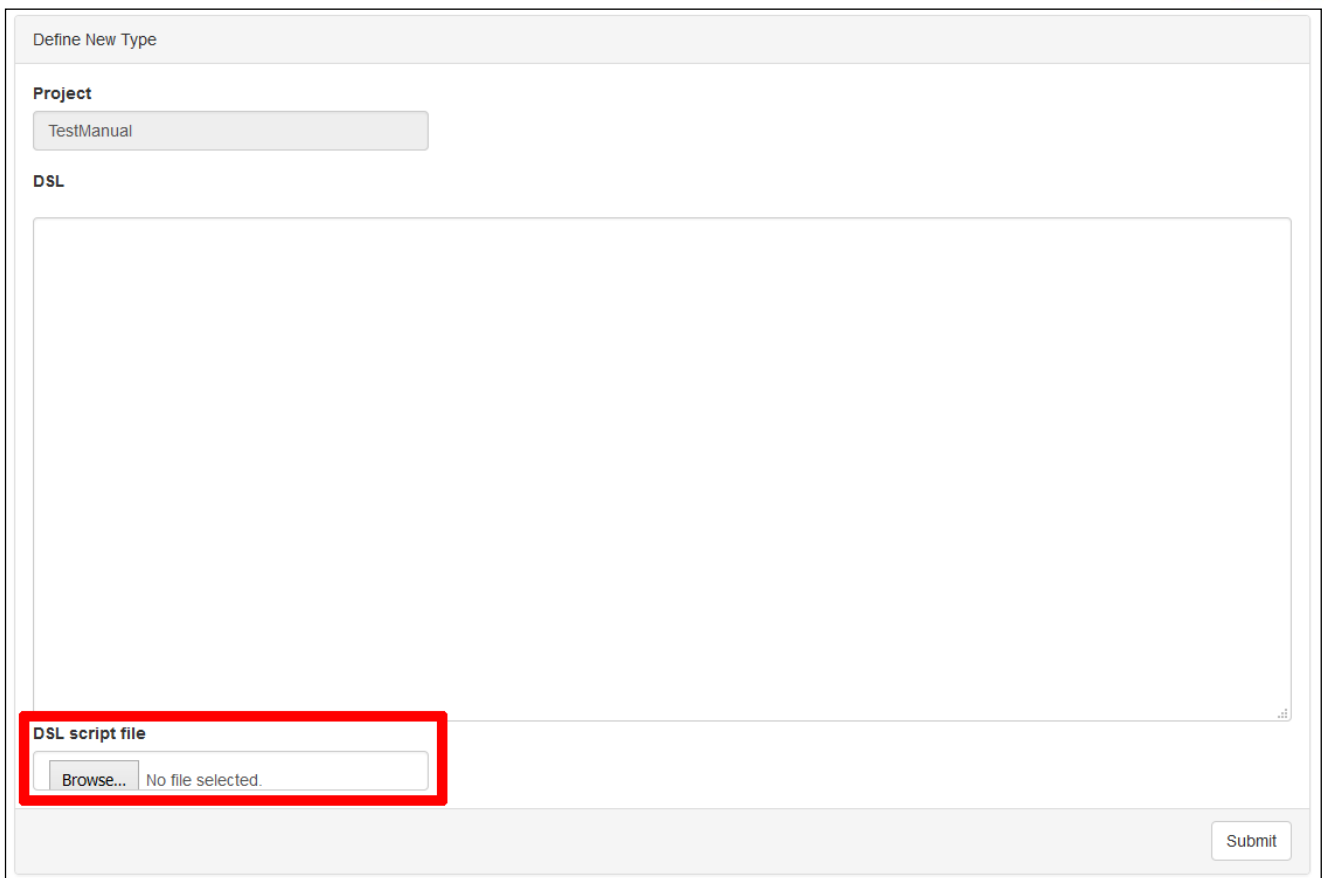


Figure 10: Submitting DSL code via an already existing file

Another option is to configure your testbed by filling in your DSL description using the editor window of the GUI (Figure 11): Just type in your code and click on 'Submit'.

Define New Type

Project
TestManual

DSL

```
Triangle {  
  description = "Triangle testbed with three hosts."  
  id="TriangleTestbed"  
  
  host {  
    id="h1"  
    port { id="eth1" }  
    port { id="eth2" }  
  }  
  
  host {  
    id="h2"  
    port { id="eth1" }  
    port { id="eth2" }  
  }  
  
  host {  
    id="h3"  
    port { id="eth1" }  
    port { id="eth2" }  
  }  
}
```

DSL script file
Browse... No file selected.

Submit

Figure 11: Submitting DSL code directly

4.2 Reserving testbed resources

After submitting your DSL code the system takes you back to the 'Types' screen and you will see the new testbed type you have added to your project (Figure 12)

Types		
Name	Description	Actions
Triangle	Triangle testbed with three hosts.	 

Figure 12: Added testbed type

When you click on the type's name you have read-only access to your DSL code. For a created testbed type you can choose the following actions:

- You can RESERVE the testbed by clicking on the green button.
- The red button allows you to 'Undefine' or delete the type and the DSL file is dropped.

After you selected the green button to reserve your testbed you will see the following information on your screen that lists your new resource type (Figure 13):



Testbeds					
Provider ID	ID	Status	Type	Actions	
+ 321	TriangleTestbed	RESERVED	Testbed (Triangle) 6	 	

Figure 13: Reserved new resource type

You can expand the listed information to show you more details on the individual resource components that were reserved as part of your new type by clicking on the '+' sign next to the Provider ID on the left (Figure 14):



Testbeds					
Provider ID	ID	Status	Type	Actions	
- 321	TriangleTestbed	RESERVED	Testbed (Triangle) 6	 	
	Host-1464089637720	RESERVED	Host		
	Host-1464089638483	RESERVED	Host		
	Host-1464089639090	RESERVED	Host		
	OpenNsaLink-GT-a18136acac	RESERVED	Link		
	OpenNsaLink-GT-162254d444	RESERVED	Link		
	OpenNsaLink-GT-67c6322cd4	RESERVED	Link		

Figure 14: Listing of reserved resources

The reservation process is complete when your resources show the status 'RESERVED'.

Note: In case you specify a certain location for a resource and your request cannot be granted because not enough resources are available at that time at that location then your request may go through if you resubmit your DSL without the specification of a location (if that is not a requirement for your experiment in some way).

4.3 Activating testbed resources

Before *reserved* resources can be used, they must be *activated*. Click on the green button on the right to start the activation process (Figure 15).



Testbeds					
Provider ID	ID	Status	Type	Actions	
+ 321	TriangleTestbed	RESERVED	Testbed (Triangle) 6	 	

Figure 15: Activate a resource

The activation of resources prompts the user to indicate a start time and end time. The activation can happen in two ways:

1. If you specified your reservation time window (start time and end time) in your DSL, the resources will automatically go from reserved state to active when the time window starts (Figure 16). During that defined time window (i.e. when your reservation is valid) you can actually deactivate and activate your resources as you wish.
2. If you did not specify a reservation window in your DSL, your reservation is valid starting from the current time until practically forever. This will likely change in later versions of GTS! In this case you can activate and deactivate your resources whenever you want.



Select Reservation to activate					
Provider ID	ID	Status	Type	Actions	
<p>▶ Reservation Id: 321 Start time: 24-May-2016 11:33:00 GMT End time: 31-Jan-2526 22:59:59 GMT</p>					
321	TriangleTestbed	RESERVED	Testbed (Triangle) 6	 	

Figure 16: Reservation start and end time

You can follow the status of your resources while it is changing from 'RESERVED' to 'ACTIVE' (Figure 17):


Testbeds						
Provider ID	ID	Status	Type	Actions		
321	TriangleTestbed	ACTIVATING	Testbed (Triangle) 6			
Host-1464089637720	h3	ACTIVE	Host			
Host-1464089638483	h2	ACTIVE	Host			
Host-1464089639090	h1	ACTIVATING	Host			
OpenNsaLink-GT-a18136acac	l3	ACTIVE	Link			
OpenNsaLink-GT-162254d444	l2	ACTIVE	Link			
OpenNsaLink-GT-67c6322cd4	l1	ACTIVE	Link			

Figure 17: Activating resources



Testbeds						
Provider ID	ID	Status	Type	Actions		
321	TriangleTestbed	ACTIVE	Testbed (Triangle) 6	 		
Host-1464089637720	h3	ACTIVE	Host			
Host-1464089638483	h2	ACTIVE	Host			
Host-1464089639090	h1	ACTIVE	Host			
OpenNsaLink-GT-a18136acac	l3	ACTIVE	Link			
OpenNsaLink-GT-162254d444	l2	ACTIVE	Link			
OpenNsaLink-GT-67c6322cd4	l1	ACTIVE	Link			

Figure 18: All resources are activated

For activated resources the status turns to 'ACTIVE' (Figure 18); please allow a few minutes for processing as all resources are created dynamically and not just picked from an already existing pool.

4.4 Setting reservation windows

This version does not allow the setting of specific start- and end times yet for activation and deactivation processes. Simply select the provided default settings by clicking on them (Figure 19) (***Please note***: Closing the window is not enough to choose the default values!):



Figure 19: Selecting default start- and end-times

4.5 Deactivating testbed resources

At some time during an experiment a user may wish to deactivate the resources, but keep the reservation of resources valid for use at a later point in time. In other words: Deactivated resources are not released and returned to other users until the experimenter actually *releases* the resources. Deactivated resources can be returned to an active state by simply activating them again at any time during the experiment.

To deactivate a resource, simply click on the yellow button on the right (Figure 20):


Provider ID	ID	Status	Type	Actions
321	TriangleTestbed	ACTIVE	Testbed (Triangle) 6	
Host-1464089637720	h3	ACTIVE	Host	
Host-1464089638483	h2	ACTIVE	Host	
Host-1464089639090	h1	ACTIVE	Host	
OpenNsaLink-GT-a18136acac	l3	ACTIVE	Link	
OpenNsaLink-GT-162254d444	l2	ACTIVE	Link	
OpenNsaLink-GT-67c6322cd4	l1	ACTIVE	Link	

Figure 20: Deactivation of a resource

The start- and end-time reservation window will pop up again where you can specify the time when your resources are to be deactivated (in this version only available with default setting); select the (default) setting by clicking on the line with the start and end times (closing the window is not enough!) and wait for the status of the resources to change to 'RESERVED' again (Figure 21). But that does not mean that they are released (please see section 4.6 for more details); instead they remain reserved for the user to be available for the next activation process.

Provider ID	ID	Status	Type	Actions
321	TriangleTestbed	ACTIVE	Testbed (Triangle) 6	

Select Reservation to deactivate

Reservation Id: 321 Start time: 24-May-2016 11:33:00 GMT End time: 31-Jan-2526 22:59:59 GMT

Figure 21: Selecting an end-time

4.6 Releasing testbed resources

Resources that are no longer of interest should be *released* so that these resources can be made available to other projects and users.



Testbeds					
Provider ID	↕ ID	↕ Status	↕ Type	↕ Actions	
+ 342	TriangleTestbed	RESERVED	Testbed (Triangle) 6		 

Figure 22: Releasing resources

To release the testbed resource click on the red button on the right as marked in Figure 22.

4.7 Querying details

The individual resources that were reserved for your testbed become visible when you click on the '+' sign to the left of the Provider ID of your resource (Figure 23); each resource can then also be accessed by clicking on its link (Figure 24):




Testbeds					
Provider ID	ID	Status	Type	Actions	
 342	TriangleTestbed	RESERVED	Testbed (Triangle) 6		

Figure 23: Obtaining details via Provider ID




Testbeds					
Provider ID	ID	Status	Type	Actions	
 342	TriangleTestbed	RESERVED	Testbed (Triangle) 6		
Host-1465200373058	h3	RESERVED	Host		
Host-1465200374206	h2	RESERVED	Host		
Host-1465200375170	h1	RESERVED	Host		
OpenNsaLink-GT-abdd0d2a18	l3	RESERVED	Link		
OpenNsaLink-GT-4ecec2ca07	l2	RESERVED	Link		
OpenNsaLink-GT-276c394f16	l1	RESERVED	Link		

Figure 24: List of individual testbed resources


By clicking on a resource you can also obtain more information on ports, locations as well as reservation start and end times (Figure 25):

Resource: Host

Provider id	Id	Status	Description	Ports	Location	Console
Host-1481289272973	host1	ACTIVE			lab1	http://195.113.161.164:8080/vnc_auto.html?token=c11c8d4b-78c6-4896-9ff9-2da490b5e75a

Reservations

Id	Start Time	End Time
100	Fri, 09 Dec 2016 13:14:00 GMT	Thu, 31 Jan 2528 22:59:59 GMT

Statistics 

Key	Value
vdb_write	0
vdb_read	757780
memory-rss	332356
vda_read	133879808
tan90R5590D-5n tv drom	0

Figure 25: Details of a testbed resource, in this case details of a Host resource including statistics

To obtain a tree structure of your testbed resource and to see port adjacencies or adjacencies for external ports, click on the Provider ID itself (Figure 26 and Figure 27):



Provider ID	ID	Status	Type	Actions
342	TriangleTestbed	RESERVED	Testbed (Triangle) 6	 

Figure 26: Access to tree structure

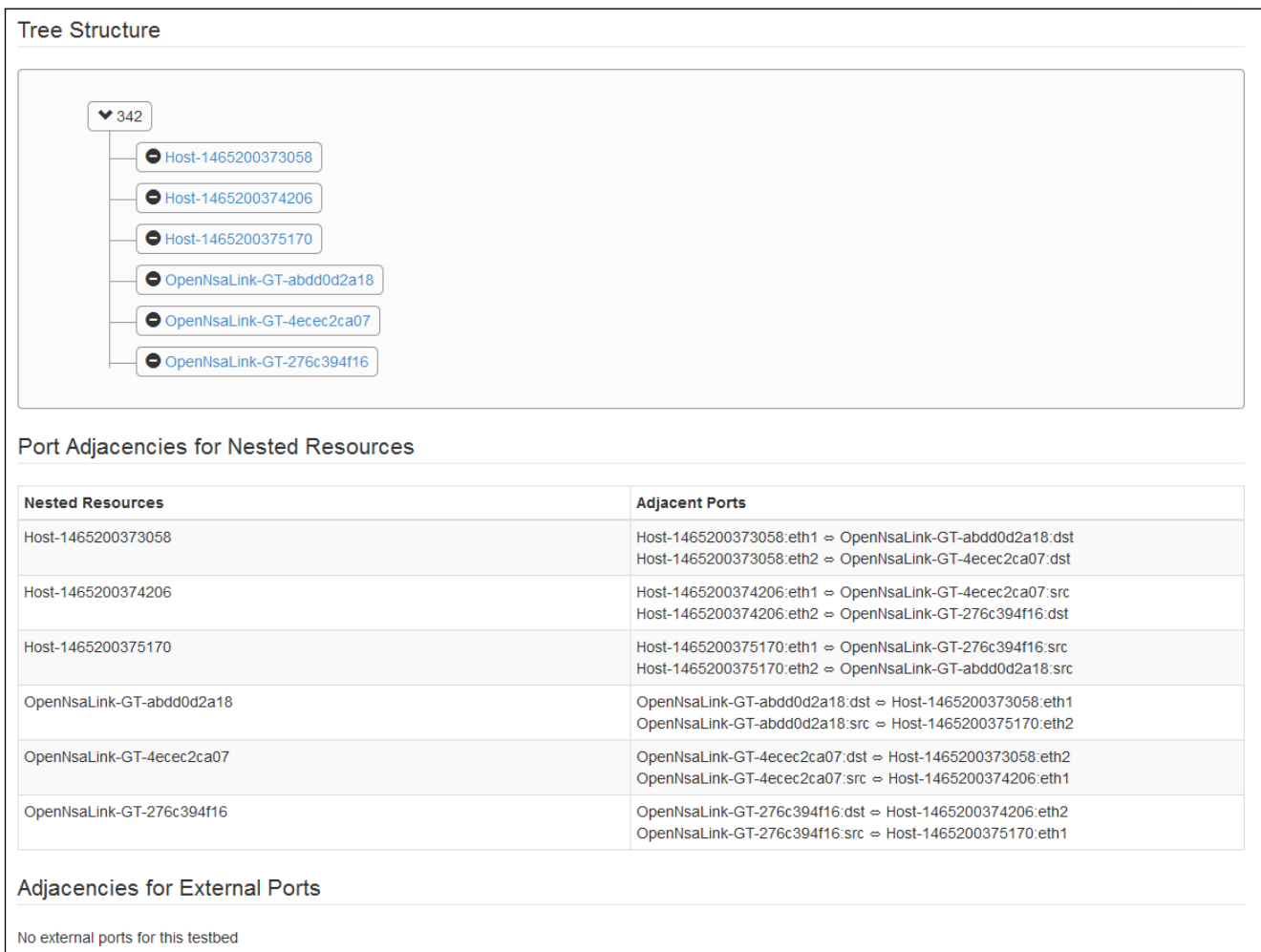


Figure 27: Tree structure and port adjacencies

4.8 Internet Access Gateway (IAGW)

As a part of its creation, each project in GTS gets a separate broadcast domain. This broadcast segment is then used to connect all VMs in your project regardless of their testbed and location. The segment will also host an IAGW instance which provides your VMs with the basic Internet connectivity (DHCP and SNAT), VPN access and a shared file system.

4.8.1 Internet connectivity

The Internet connectivity from your VMs is realized through preconfigured DHCP client on the first Ethernet port (i.e. `eth0`). The rest of the ports (`eth1`, `eth2`, and so on) are not preconfigured and are there for point-to-point links as specified in your DSL. So do not use `eth0` to run tests in your testbed, this might break it!

DHCP server on the IAGW announces a private subnet range and the source network address translation (SNAT) is used to enable VMs to reach the Internet.

4.8.2 VPN access

Because of the private subnet range and SNAT, VMs in your project are not exposed on the Internet. Port forwarding and the floating public IP addresses are not supported at the moment and are considered for some future versions of GTS. However, if you need a direct IP connectivity from your personal computer to any of the VMs in your project, you can use the VPN functionality described in the following sub-sections.

4.8.2.1 VPN connectivity from Windows

For systems using Windows, set up a new VPN network connection via the Control Panel and its Network and Internet options.

To set up the VPN use the IP address shown when you click on the blue 'i' icon next to your project name (Figure 28: VPN information and Figure 29: VPN IP addressFigure 28) and also the VPN user name and password you defined during the initial project registration process.

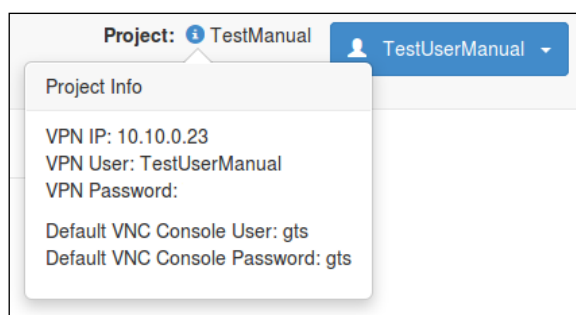


Figure 28: VPN information

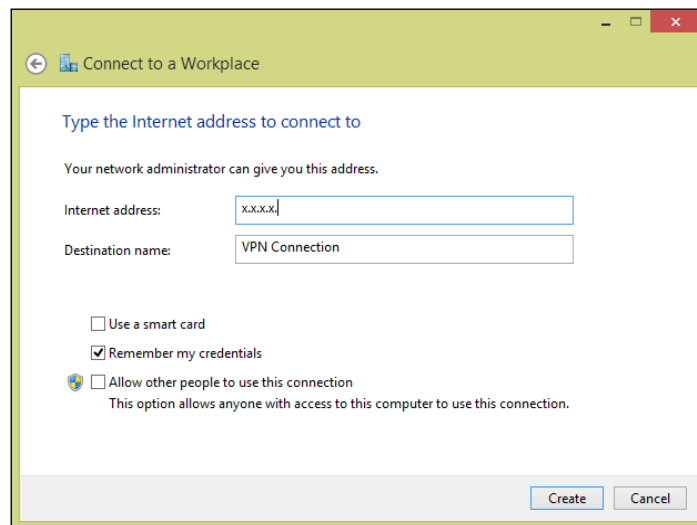


Figure 29: VPN IP address

To prevent Routing and/or Gateway problems on Windows you have to adjust some VPN network properties. Right click on the name of the VPN network in the Open Network and Sharing Center:

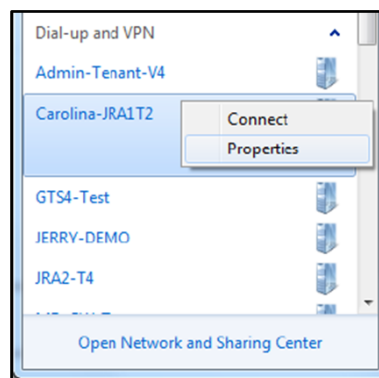


Figure 30: Open Network and Sharing Center

Then choose Internet Protocol Version 4 (TCP/IPv4) from Networking Tab and click the Properties Button. The TCP/IPv4 Properties will be displayed.

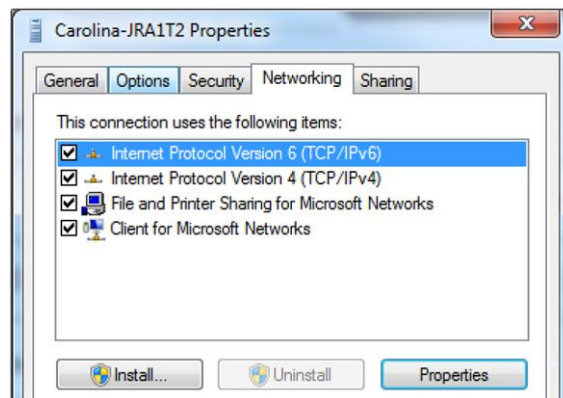


Figure 31: TCP/IPv4 Networking

There click the Advanced Button and make sure that the checkboxes in the IP Settings Tab have the same configuration like in Figure 31:

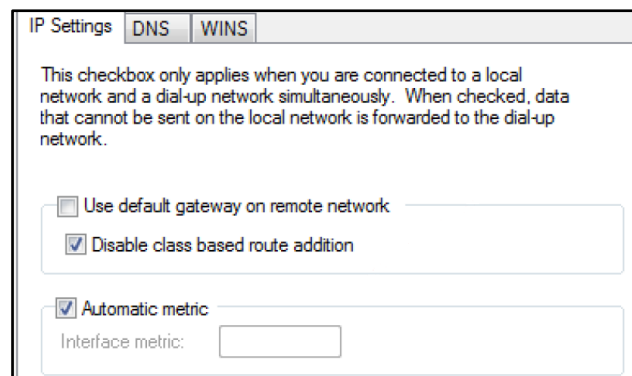


Figure 32: IP Settings for TCP/IPv4

Then repeat the same procedure for Internet Protocol Version 6 (TCP/IPv6) (see also Figure 31) until the IP Settings Tab for Ipv6 is displayed. There the configuration has to be set as follows:

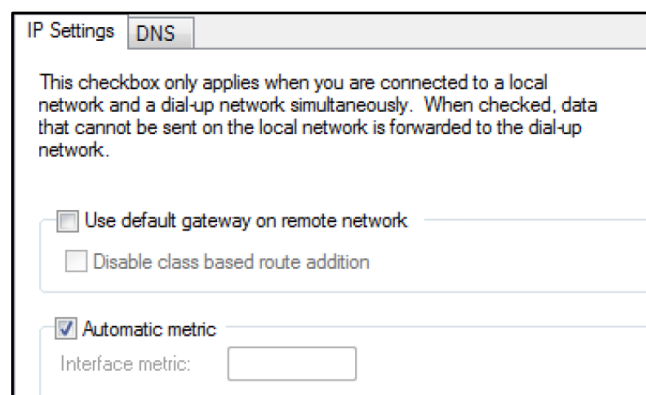


Figure 33: IP Settings for TCP/IPv6

For example, once you are connected via VPN, you can use programs like WinSCP or PuTTY to transfer files or SSH to a testbed host. In order to do so, you need to access the console of the host that you would like to connect to by clicking on the appropriate host (Figure 34) which will then provide a console link (Figure 35).










Resources						
Provider ID	ID	Status	Type	Actions		
2491	8963620	ACTIVE	Testbed			
Host-1425297436046	h3	ACTIVE	Host			
Host-1425297436919	h2	ACTIVE	Host			
Host-1425297437784	h1	ACTIVE	Host			
OpenNsaLink-TA-T863cc624f840	l3	ACTIVE	Link			
OpenNsaLink-TA-T6ee5ff17d8b1	l2	ACTIVE	Link			
OpenNsaLink-TA-T115cf06baffe	l1	ACTIVE	Link			

Figure 34: Accessing the console of a host

Resource: Host						
Provider id	Id	Status	Description	Ports	Location	Console
Host-1465200373058	h3	ACTIVE		eth1 eth2	ams	http://62.40.126.12:6080/vnc_auto.html?token=b71365e5-3c17-45a8-b148-a3076bd19c62

Reservations		
Id	Start Time	End Time
891	Mon, 06 Jun 2016 08:06:00 GMT	Thu, 31 Jan 2526 23:59:59 GMT

Figure 35: Console link

Click on the console link; you will then be prompted for user name and password. Look for the IP address provided on the screen for `eth0` (Figure 36); this is the IP address you should use for your file transfer program along with user name and password. If you don't see an IP address here, use the `ifconfig` command to find out the IP address for interface `eth0`.

```

Ubuntu 14.04.1 LTS sa2host tty1
sa2host login: gts
Password:
Last login: Fri Mar  6 10:11:08 CET 2015 on tty1
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Fri Mar  6 10:11:08 CET 2015

System load:  0.0          Processes:            68
Usage of /:   16.3% of 6.76GB    Users logged in:    0
Memory usage: 1%          IP address for eth0: 172.16.0.11
Swap usage:  0%

Graph this data and manage this system at:
https://landscape.canonical.com/

gts@sa2host:~$ _

```

Figure 36: IP address for eth0

4.8.2.2 VPN connectivity with Debian / Ubuntu

For a VPN connection to GTS using Debian / Ubuntu proceed with the following steps:

- Get into root environment
\$ sudo su
- Switch into directory 'peers'
cd /etc/ppp/peers
- Setup up a VPN configuration file 'projectname' and adjust the values marked in red (Figure 37):

```

### Build VPN connection to GTS VPN Server
pty "pptp <VPN-IP> --nolaunchpppd --nobuffer --timeout 10"

### GTS VPN - user name
name <VPN user name>

### This command reconnects to GTS VPN server after a disconnect
persist

### MTU value should be standard value
mtu 1400

### used for scripts in ip-up / ip-down
ipparam <YOURPROJECTNAME>

### Terminate after n consecutive failed connection attempts.
### A value of 0 means no limit. The default value is 10.
maxfail 0

### IMPORTANT: Do not use this option for the GTS VPN connection.
### Otherwise all packets are routed over this connection.
### Please comment it out:
#defaultroute

### Some useful settings
remotename <YOURPROJECTNAME>
lock
noauth
nobsdcomp
nodeflate

#Require encrypt
require-mppe-128

```

Figure 37: Adjust VPN configuration file

To find out the correct IP and user name that you should set in this VPN configuration file click on the blue 'i' icon next to your project name (Figure 38) and use the values you set during your initial project registration and then save the file.

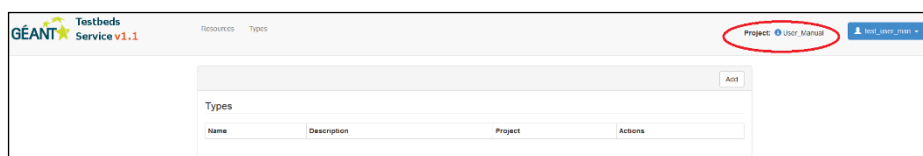


Figure 38: VPN information

- In the next step adjust the file 'chap-secrets' in directory /etc/ppp/ indicating your VPN user name, your project name and VPN password that you set during your initial project registration (Figure 43):

```
# Secrets for authentication using CHAP
# client server secret          IP addresses
<VPN USER> <YOURPROJECTNAME> "<VPN PASSWORD>" *
#end
```

Figure 39: Adjust file 'chap-secrets'

- Then switch to the directory ip-up.d
cd ip-up.d/
- and set the project route with an executable file 'yourprojectname.route' (Figure 44)

```
#!/bin/sh
#IMPORTANT: Use here the network IP of interface eth0 of your
# GTS VM (NOT the VPN IP!)
if [ "${PPP_IPPARAM}" = "<YOURPROJECTNAME>" ]; then
    /sbin/route add -net <YOUR NETWORK IP>/24 add dev <PPP
INTERFACE NAME>
fi
```

Figure 40: Set project route with executable file

and make sure you use the network IP address of eth0 of your host. To find this IP address click on the appropriate host (Figure 41) which will then provide a console link (Figure 42).










Provider ID	ID	Status	Type	Actions
2491	8963620	ACTIVE	Testbed	  
Host-1425297436046	h3	ACTIVE	Host	
Host-1425297436919	h2	ACTIVE	Host	
Host-1425297437784	h1	ACTIVE	Host	
OpenNsaLink-TA-T863cc624f840	i3	ACTIVE	Link	
OpenNsaLink-TA-T6ee5ff17d8b1	i2	ACTIVE	Link	
OpenNsaLink-TA-T115cf06baffe	i1	ACTIVE	Link	

Figure 41: Accessing the console of a host

Provider id	Id	Status	Description	Ports	Location	Console
Host-1425297436919	h2	ACTIVE		port22 port21	AMS	http://62.40.126.10:6080/vnc_auto.html?token=85b2f30a-3612-429a-8688-5bb5be0d83fb

Id	Start Time	End Time
2554	Mon, 02 Mar 2015 11:57:00 GMT	Thu, 31 Jan 2526 22:59:59 GMT

Figure 42: Console link

Click on the console link; you will then be prompted for user name and password. Look for the IP address provided on the screen for `eth0` (Figure 36); this is the IP address you should set as **<YOUR NETWORK IP>** in the file `'yourprojectname.route'` as shown in Figure 44.

In the file `'yourprojectname.route'` you also define the **<PPP INTERFACE NAME>** (see Figure 40) as `pppx` where 'x' would be '0' if you have no other VPNs defined (or else would be incremented to the appropriate number, but please **note**: only count VPNs set up with `ppp`, not VPNs set up with other clients such as OpenVPN).

- This file `'yourprojectname.route'` should then be made into an executable file

```
# chmod 700 yourprojectname.route
```

- To connect with the VPN use

```
# pon projectname
```

- To disconnect the VPN use

```
# poff projectname
```

- **To test:** Type

```
# ifconfig
```

and look for your `pppx` with address 172.16.0.1 (Figure 43):

```
ppp0      Link encap:Point-to-Point Protocol
          inet addr:172.16.0.1  P-t-P:172.16.0.253  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1396  Metric:1
          RX packets:14584 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13996 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:2568663 (2.4 MiB)  TX bytes:966234 (943.5 KiB)
```

Figure 43: check for `pppx`

- and then check if you can ping your host, for example

```
# ping -c 5 172.16.0.10
```

- if your ping does not work, check with

```
# ip route show
```

if you have the following entry:

```
172.16.0.0/24 dev ppp0 scope link
```

4.8.2.3 VPN connectivity with OS X/macOS

Starting with macOS 10.8 (Sierra) Apple has dropped support for the PPTP VPN that GTS is using. To continue using the VPN connection to your testbed with macOS 10.8, you need a 3rd party client. There are multiple clients that support PPTP VPN, the GTS operations team has successfully tested the Flow VPN client (<https://www.flowvpn.com/download-mac/>) on macOS 10.8.

4.8.3 Persistent Shared Project Folder

IAGW instance also acts as a read-write shared folder (SMB/CIFS share) for the project. This shared directory is by default mounted on `/home/gts/project-share` but this can be easily disabled or changed for each VM individually (just edit the appropriate line in `/etc/fstab` file). The folder is “project-persistent” in the sense that activating or releasing a testbed will not have any effect on its content. Currently, a user can store up to 3GB of data in this project-share.

To access this directory from your personal computer, you need to VPN to the project as described in one of the previous sub-sections. This, for example, gives you the ability to easily “drag-and-drop” the content which then becomes available to all hosts in your project. Figure 44 shows how this can be done from a Windows Explorer:

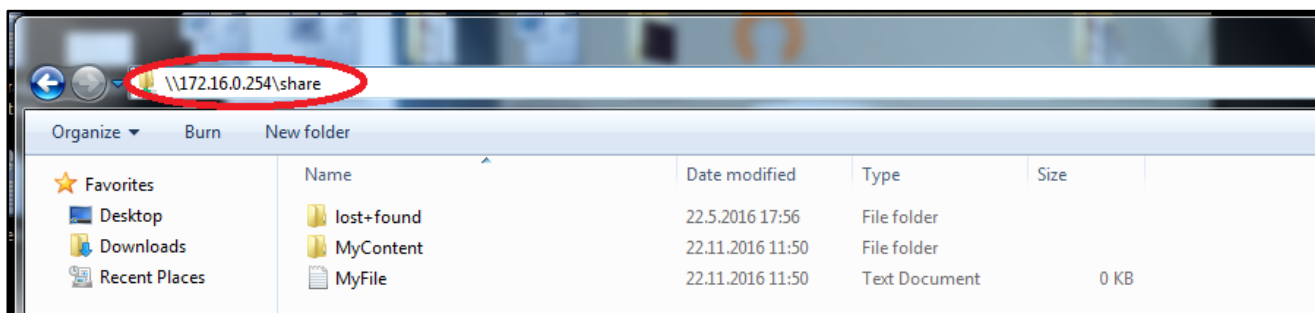


Figure 44: Accessing shared folder from Windows Explorer

The path used to access the shared directory from the Windows Explorer is `\\172.16.0.254\share` where `172.16.0.254` is the private IP address of the IAGW (the gateway address for the subnet announced via DHCP).

The content of the share is not accessible from the Internet.

4.9 Example: Working with GTS OpenFlow switches (VSI)

This chapter guides you through a DSL example that setups an OpenFlow environment with two VSIs² OpenFlow switches, two hosts and one controller. In the end we will have a working OpenFlow network environment. The complete DSL code is available on page 68 (“Example: Two OpenFlow switches (VSI), each with two ports, one host and one separate controller”). Below we will explain in small parts the DSL code.

Step 1: Define the three hosts, one will act as the controller host

Three hosts are defined, one host will act as the OpenFlow controller. The other two hosts will act as testing environment for our OpenFlow environment.

```
twoCorsaVSIController2Hosts {
    description = "Two corsa VSI each with 2 ports 1 host and a
controller"
    id = "twoVsiCorsaTst"

    host {
        id="host1"
        port { id="port1" }
    }

    host {
        id="host2"
        port { id="port1" }
    }

    host {
        id="controller"
        port { id="port1" }
        port { id="port2" }
    }
}
```

² Please note that in GTS v4 OFXs are no longer available and are replaced with fully virtualized Virtual Switch Instances (VSIs).

Step 2: Define the two VSI OpenFlow Switches

```

vsi {
  id="vsi1"
  location="LON"
  switchIPv4Addr="10.10.100.1"
  switchIPv4Mask="255.255.255.0"
  switchMode="hard"

  controller {
    ipv4="10.10.100.100"
    port="6653"}

  port {
    id="port1"
    logicalPort=1}
  port {
    id="port2"
    logicalPort=2}
  port {
    id="port9"
    mode="CONTROL"
  }
}

vsi {
  id="vsi2"
  location="LON"
  switchIPv4Addr="10.10.101.1"
  switchIPv4Mask="255.255.255.0"
  switchMode="hard"

  controller {
    ipv4="10.10.101.100"
    port="6653"}

  port {
    id="port1"
    logicalPort=1}
  port {
    id="port2"
    logicalPort=2}
  port {
    id="port9"
    mode="CONTROL"
  }
}

```

In this section the two OpenFlow switches are defined in the DSL. Each switch has a number of attributes, like the IP address of the controller port, the IP address of the controlling OpenFlow controller, the ports (including logical numbers), etc. Please see the **Error! Reference source not found.** section in Appendix I: Resource Guide for detailed information about the attributes.

Step 3: Connect the two hosts with the two switches

Now the two hosts need to be connected with their respective switches. This is done by adding a link for each and connecting the endpoints of the links with adjacencies to the ports of the switches/hosts.

```

link {
    id="h1vsi"
    port { id="src" }
    port { id="dst" }
}

link {
    id="h2vsi"
    port { id="src" }
    port { id="dst" }
}

adjacency host1.port1, h1vsi.src
adjacency vsi1.port1, h1vsi.dst

adjacency host2.port1, h2vsi.src
adjacency vsi2.port1, h2vsi.dst

```

Step 4: Connect the controller host with the two control ports of the VSIs

Additionally you need to connect the two ports of the controller host with the controller ports of the two switches. So you define two links and again use adjacencies to connect the endpoints of the links with the ports.

Step 5: Connect the two switches with each other

```

link {
    id="controllervsi1"
    port { id="src" }
    port { id="dst" }
}

link {
    id="controllervsi2"
    port { id="src" }
    port { id="dst" }
}

adjacency controller.port1, controllervsi1.src
adjacency vsi1.port9, controllervsi1.dst
adjacency controller.port2, controllervsi2.src
adjacency vsi2.port9, controllervsi2.dst

```

```

link {
    id="vsilvsi2"
    port { id="src" }
    port { id="dst" }
}

adjacency vsi1.port2, vsilvsi2.src
adjacency vsi2.port2, vsilvsi2.dst

```

Don't forget to close the surrounding "twoCorsaVSIController2Hosts {" block with a "}".

Step 6: Configuring the controller instance

Once you have reserved your testbed, activate the resources and make sure that all resources are marked as 'ACTIVE'.

- find your controller in the list of resources
- click on the controller in order to get a link to its console (as you would do to get access to the console of any VM; see Figure 45, Figure 46, compare also with Figure 42)



Testbeds							
Provider ID	ID	Status	Type	Actions			
- 103	twoVsiCorsaTst	ACTIVE	Testbed (twoCorsaVSIController2Hosts)	 			
Host-1487598228991	controller	ACTIVE	Host				
Host-1487598229360	host2	ACTIVE	Host				
Host-1487598229724	host1	ACTIVE	Host				
VSI-6ca4f25d-42e3-41b6-a137-d34dc2ba3197	vsi2	ACTIVE	VSI				
VSI-d90d5ba8-8ae8-4e13-ad58-4a7fd4fc5760	vsi1	ACTIVE	VSI				
OpenNsaLink-GT-fcc1394c46	controllervsi2	ACTIVE	Link				
OpenNsaLink-GT-8aff967f5	controllervsi1	ACTIVE	Link				
OpenNsaLink-GT-0924bc12c0	vsi1vsi2	ACTIVE	Link				
OpenNsaLink-GT-e5caf4e4e6	h2vsi	ACTIVE	Link				
OpenNsaLink-GT-5fc584dfd5	h1vsi	ACTIVE	Link				

Figure 45: Access to controller VM

Resource: Host						
Provider id	Id	Status	Description	Ports	Location	Console
Host-1487598228991	controller	ACTIVE		port2 port1	lab1	http://195.113.161.164:6080/vnc_auto.html?token=7f0a937d-07ab-4cfe-bcac-8c55f11c9d26 ⓘ Credentials Username: gts Password: gts

Reservations		
Id	Start Time	End Time
10	Mon, 20 Feb 2017 13:43:00 GMT	Thu, 31 Jan 2526 22:59:59 GMT

Figure 46: Link to controller console

- at the console you will be prompted for your user name and password

- at the moment it is still necessary to manually configure the interfaces of the controller (in later versions of GTS it will be possible to simply set this via the GUI); this is because GTS currently does not support port mapping.

- **please note**: eth0 should NOT be used, as it is currently reserved for internal processes!

- configure the eth1 and eth2 interfaces of controller (as already defined in the DSL example):

```
$ sudo su
# ifconfig eth1 up 10.10.100.100 netmask 255.255.255.0
# ifconfig eth2 up 10.10.101.100 netmask 255.255.255.0
and check for incoming ARP packets on both interfaces with
# tcpdump -i eth1
# tcpdump -i eth2
```

You will see the incoming echo request from the VSI like this:

```
#16:49:58.702223 IP 10.10.100.1.59938 > 10.10.100.100.6653: Flags [S], seq
688712374, win 26880, options [mss 8960,sackOK,TS val 1019200571 ecr 0,nop,wscale
7], length 0 i.e. a VSI with IP address 10.10.100.1 requests IP 10.10.100.100
```

- open the interfaces configuration file with your editor

```
# vim /etc/network/interfaces
```

- add the following entries to the file:

```
auto eth1
iface eth1 inet static
address 10.10.100.100
netmask 255.255.255.0
network 10.10.100.0
```

```
auto eth2
iface eth2 inet static
address 10.10.101.100
netmask 255.255.255.0
network 10.10.101.0
```

- start and stop both interfaces with

```
# ifdown eth1
# ifup eth1
# ifdown eth2
# ifup eth2
```

- check if you can ping both switches

```
# ping -c 5 10.10.100.1
# ping -c 5 10.10.101.1
```

Step 7: Install ONOS as OpenFlow controller

- add and install the dependencies required for ONOS

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update -y
$ sudo apt-get install git zip oracle-java8-installer oracle-java8-set-
default python2.7 python2.7-dev -y
```

- download and compile ONOS

```
$ git clone https://github.com/opennetworkinglab/onos.git ~/onos
```

```
$ cd ~/onos
$ touch -f ~/.bashrc
$ echo "export ONOS_ROOT=$PWD" >> ~/.bashrc
$ . ~/.bashrc
$ tools/build/onos-buck build onos --show-output
```

- **Run ONOS**

```
$ tools/build/onos-buck build onos --show-output
$ export ONOS_APPS=drivers,openflow,proxyarp,mobility
$ tools/build/onos-buck run onos-local -- clean debug
```

- **Press Ctrl+D+A to detach from the running screen session, now we connect ONOS to the running server**

```
$ tools/test/bin/onos localhost

# Change OpenFlow port (if required, to match that from the DSL)
onos> cfg set
org.onosproject.openflow.controller.impl.OpenFlowControllerImpl
openflowPorts 6653
```

- **Now that the basic setup of ONOS has been finished, let's configure ONOS so that the two hosts can connect to each other. We do this by using manual intents in ONOS**

- **First check that the two VSI switches are connected to ONOS**

```
onos> devices
id=of:000026fa26251242, available=true, local-status=connected 14s ago,
role=MASTER, type=SWITCH, mfr=Corsa, hw=CDP2100-A00, sw=corsa-ovs-datapath
1.5.71, serial=None, driver=default, channelId=10.10.100.1:52516,
managementAddress=10.10.100.1, protocol=OF_13
id=of:00005a5efef9e344, available=true, local-status=connected 13s ago,
role=MASTER, type=SWITCH, mfr=Corsa, hw=CDP2100-A00, sw=corsa-ovs-datapath
1.5.71, serial=None, driver=default, channelId=10.10.101.1:60386,
managementAddress=10.10.101.1, protocol=OF_13
```

- **Now setup the point-to-point intents, we need to cover all the connections between the switch ports. So you need to get the two OpenFlow ids from the devices command above and add intents between the two ports on each switch**

```
onos> add-point-intent of:000026fa26251242/1 of:000026fa26251242/2
onos> add-point-intent of:000026fa26251242/2 of:000026fa26251242/1
onos> add-point-intent of:00005a5efef9e344/1 of:00005a5efef9e344/2
onos> add-point-intent of:00005a5efef9e344/2 of:00005a5efef9e344/1
```

Step 8: Assign IP addresses to the two hosts and test the network connection

- **We're almost finished now, we just need to assign IP addresses to the two hosts and test the network connection by using the ping command.**

- **Login to host1 and execute the following commands**

```
$ sudo bash
# ifconfig eth1 up 192.168.0.1 netmask 255.255.255.0
```

- Login to host2 and execute the following commands

```
$ sudo bash
# ifconfig eth1 up 192.168.0.2 netmask 255.255.255.0
```
- Congratulations, you've now setup your own OpenFlow environment in GTS. You should be able to ping host2 from host1 and vice versa. So execute the following command on host1 to check your network connection that was established with the help of OpenFlow

```
$ ping -c 5 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=32.3 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.752 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.692 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=0.667 ms
64 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=0.696 ms
```

5 Help and Support

For help and support please contact

- GÉANT Operations Centre
mailto: gts-operations@lists.geant.org
- phone: +44 1223 866140

You also find this support information by clicking on 'About' at the bottom of the GTS home page (Figure 47).



Figure 47: Access to support

In case you forgot your password, you can use the password hash calculator at <https://www.dailycred.com/article/bcrypt-calculator> to generate a hash for your new password (use 10 rounds for the hash generation). Please then send us your generated hash and we will update your account.

Additional information and training videos are available at https://www.geant.org/Services/Connectivity_and_network/GTS/.

6 Introduction to Domain Specific Language

6.1 DSL in GTS

GTS uses its own Domain Specific Language (DSL) to describe topologies. It is used by users to specify their requests, but also by GTS service to represent the allocated topology that conforms to users' requirements. DSL not only allows defining requests, but also offers Groovy [GRO-2014] programming language syntax that eases up the process of specifying scalable, complex topologies. In this section only DSL syntax is described. Readers should get familiar with Groovy programming language grammar if they want to use programming language features. For basic request definition knowing only DSL syntax is enough. The section is divided into two parts: Quick start and DSL grammar definition. The first offers introduction with examples where after five minutes users will know all concepts and should be able to define basic requests. The latter provides information about all resource types, their parameters and BNF (Backus–Naur Form) grammar of the DSL.

6.2 Quick Start

For every virtual resource there is a resource class or type, called a 'template' that defines all possible attributes of that resource. The process of creating a resource instance from this class template is referred to as 'instantiation'. In a testbed, a researcher can instantiate several instances of a resource and distinguish them by assigning each resource instance a unique name [DEL-D61].

Each resource class has

- a class name / an identifier of that class (not instance!) (name cannot start with a digit)
- external ports (where each port has a port name and a set of attributes for those ports)
- class attributes
- internal resources
- adjacencies among internal resources
- and a set of resource control primitives defined for that resource class.

Resources can be either atomic or composite. Atomic resources denote virtual resources that will be allocated for the user. Those that are composite are logical containers of other resources. This feature allows to define scalable, complex topologies, because composite types form reusable building blocks that can be combined together.

Before going into formal BNF grammar definition let's start with a simple example to get familiar with the DSL syntax. The code snippet below is a definition of a type containing one host. The type called *HostWithTwoPorts* is a composite that contains one host. The host has two interfaces called ports *p1* and *p2*. Each resource instance

definition consists of a resource type name and a set of parameters and ports specified in curly braces. The parent-child relation is specified by defining the resource instance within the definition of the composite resource.

In this example the host has *HostWithTwoPorts* composite type as a parent.

```

type HostWithTwoPorts {
    description = "Host that has two ports."
    id = "Host that has two ports"
// This is a comment.
    host {
        id = "host"
        port { id = "p1"}
        port { id = "p2"}
    }
}

```

Let's imagine a more complex topology where we have three hosts connected to each other forming a triangle. We will reuse *HostWithTwoPorts* type to specify three hosts. Each resource can be connected with another

```

type Triangle {
    description = "Triangle using PRG, AMS and BRA hosts."
    id = "Triangle"

    host { id = "h1"
        description = "PRG HostWithTwoPorts" //
        location = "prg" // location for VM, optional
        port { id = "p1"}
        port { id = "p2"}
    }

    host { id = "h2"
        description = "AMS HostWithTwoPorts"
        location = "ams"
        port { id = "p1"}
        port { id = "p2"}
    }

    host { id = "h3"
        description = "BRA HostWithTwoPorts"
        location = "bra"
        port { id = "p1"}
        port { id = "p2"}
    }

//Note. Do not define Links that is not used. It will cause error.
    link { id = "l1"
        description = "Link between h1 and h2"
        port { id = "src"}
        port { id = "dst"}
    }
//Note. Ports for links have reserved names src and dst. Do not use for Links other names.
    link { id = "l2" ... }
    link { id = "l3" ... }
}

```

resource via a transport resource called *link*. Links have exactly two ports called *src* and *dst* denoting source and sink. In *links* other port names except “src” and “dst” are prohibited. We need three hosts and three links.

The id parameter in each resource is a user defined identifier that is unique only within the given DSL. It does not have to be a globally unique identifier. It helps users specifying properties and adjacencies. We used ids to define locations of internal nodes. The other benefit is when debugging what went wrong during instantiation of the request. Having hundreds of resources with meaningful identifiers greatly helps to find the cause of the problem in the request. The last step that we need in order to define the triangle are adjacencies between resources. We are using composites to connect to link resources.

In order to expose internal details, ie. ports of the underlying host, logical ports must be defined in the composite and connected to the host's ports. Then these logical ports will be used to connect to link ports. Now we can define triangle adjacencies.

```
type Triangle {
  ...
  adjacency h1.p1, l1.src //Link l1 connect port p1 at host h1 to
  adjacency h2.p1, l1.dst //port p1 at host h2

  adjacency h1.p2, l2.src //Link l2 connect port p2 at host h1 to
  adjacency h3.p2, l2.dst //port p2 at host h3

  adjacency h2.p2, l3.src //Link l3 connect port p2 at host h2 to
  adjacency h3.p1, l3.dst //port p1 at host h3
}
```

We specify that host *h1* port *p1* is connected to source of link *l1*. Link *l1* sink is connected to host *h2* port *p1* and so on. As you see we use user specified identifiers to help us denote specific resource instances.

Groovy programming language features come in handy when there is a need to specify complex topologies with possibly hundreds of connections. It can be done by defining procedures for doing so. Even the Triangle example can benefit from using programming. This way we do not have to define an additional type called `HostWithTwoPorts`. Take a look at the example below. We use a loop three times to define three hosts and three links. To ease up iterating over resources we defined two lists `hosts` and `links`. The second loop is used to connect all resources together. Please, check DSL below:

```
type Triangle {
    description = "Triangle using Groovy language iterators to define
adjacencies."
    id = "t1"

    def hosts = []
    def links = []

    3.times { idx ->
        def h1 = host {
            id = "host$idx"
            port { id = "p1" }
            port { id = "p2" }
        }
        hosts << h1

        def l1 = link {
            id = "link$idx"
            port { id = "src" }
            port { id = "dst" }
        }
        links << l1 //

        adjacency h1.p1, l1.src
    }

    3.times { idx -> adjacency hosts[(idx + 1) % 3].p2, links[idx].dst }
}
```

Appendix I: Resource Guide

Each resource type has a different set of properties that can be specified or read by users. This section provides a list of resource types and possible properties with description of their field types and possible ranges.

I. Composite types

Aggregate logical resources are specified by this type.

List of parameters:

Parameter	Type	Description	Example
description	String <read, write, optional>	Characterizes the type for the user.	Host that has two ports.
id	String <read, write, optional>	User defined identifier. If not specified it is autogenerated by the service.	h1
providerId	String <read-only>	Globally unique identifier of the resource instance. Resource instances have this value set by the service.	Host-1505196483255
status	RESERVED, ACTIVATING, ACTIVE, DEACTIVATING, RELEASING <read-only>	Current state of the resource.	ACTIVE
ports	Map<id, port> <read only>	Map of ports defined by the user.	port { id = "p1" } port { id = "p2" }

Example:

```

type HostWithTwoPorts {
  description = "Host that has two ports."
  id = "Type id"

  host {
    id = "h1"
    port { id = "p1" }
    port { id = "p2" }
  }
}

```

II. Host

A virtual machine is depicted by this type. Each default VM has the following components preinstalled:

- Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-37-generic x86_64)³

List of parameters:

Parameter	Type	Description	Example
location	PRG, BRA, AMS, MIL, HAM, PAR, MAD, LON <read, write, optional>	location of the resource	PRG
id	String <read, write, optional>	User defined identifier. If not specified it is autogenerated by the service.	h1
providerId	String <read-only>	Globally unique identifier of the resource instance. Resource instances have this value set by the service.	Host-1464179133289
status	RESERVED, ACTIVATING, ACTIVE, DEACTIVATING, RELEASING <read-only>	Current state of the resource.	ACTIVE
ports	Map<id, port> <read only>	Map of ports defined by the user.	port { id = "p1" } port { id = "p2" }

³ Other images are possible; please contact gts-operations@lists.geant.org.

Default configuration		Comments
Host { }	cpuSpeed == 2.3	The minimum clock speed of the CPU in GHz.
	image = "<string>"	The ISO image of the operating system installed on the host by default. The current operation system is "Ubuntu 14.04"
	disk == 20	Disk space coming with the host* defined in GB. <i>* persistent storage (NFS) of 3GB is also available in folder "project-share" of testbed.</i>
	consoleAccess == "<public URL>"	Public URL of the console access to the host.
port { }	Framing == "<Eth>"	Only Ethernet (802.1Q) framing is available.

The current default configuration is c2r2h20 (2 virtual cores = 1 physical core, 2 GB RAM and 20 GB hard disk space); but other flavors and images are possible. Please contact gts-operations@lists.geant.org.

Example:

```
host {
  id = "h1"
  location = "prg"
  port { id = "p1" }
}
```

III. Link

Represents a virtual circuit between resources. Always has exactly two ports for source and sink.

List of parameters:

Parameter	Type	Description	Example
id	String <read, write, optional>	User defined identifier. If not specified it is autogenerated by the service.	l1
providerId	String <read-only>	Globally unique identifier of the resource instance. Resource instances have this value set by the service.	OpenNsaLink-GT-228a44578b
status	RESERVED, ACTIVATING, ACTIVE, DEACTIVATING, RELEASING <read-only>	Current state of the resource.	ACTIVE
ports	Map<id, port> <read only>	Map of exactly two ports source and sink. Note that the two port ids always have to be "src" and "dst".	port { id = "src" } port { id = "dst" }

Default configuration		Comments
link { }	capacity = {1...10000}	Capacity of the link defined in Mbps. Default value is 10 Mbps
port { }	Framing = "<Eth>"	Only Ethernet (802.1Q) framing is available
	lineRate = {1, 10}	Nominal line rate of the port defined in Gbps. Default value is 1 Gbps.
	directionality = {uniDir, biDir}	Directionality of the port. Default value is bidirectional: "biDir"

Example:

```
link {
  id = "l1"
  port { id="src" }
  port { id="dst" }
}
```

IV. VSI

As of GTS 4.1 VSI (Virtual Switch Instance) is the new OpenFlow resource, which can be backed by either a OVS instance or by a hardware switch. Please note that one of the VSI ports must be denoted with mode = "CONTROL" to inform the service that this port is going to be a control port.

GTS currently uses Corsa DP 2100 Series switches which support OpenFlow specification 1.3 [CZI-2016].

List of parameters:

Parameter	Type	Description	Example
location	PRG, BRA, AMS, MIL, HAM, PAR, MAD, LON <read, write, optional>	Location of the resource	PRG
id	String <read, write, optional>	User defined identifier. If not specified it is autogenerated by the service.	vsi1
providerId	String <read-only>	Globally unique identifier of the resource instance. Resource instances have this value set by the service.	VSI-5af7dc69-228c-4a93-8ed1-e1f66aa1245
status	RESERVED, ACTIVATING, ACTIVE, DEACTIVATING, RELEASING <read-only>	Current state of the resource.	ACTIVE
ports	Map<id, port>	Map of ports defined by the user.	port { id = "p1" } port { id = "p2" }
controllers	List<controller>	Controller configuration	controller {ipv4="10.10.100.100" port="6653"}
switchIPv4Addr	IPv4 address	IPv4 address of the fabric switch	10.10.100.1
switchIPv4Mask	IP mask	IP mask of fabric subnet	255.255.255.0
switchMode	String	(currently unused parameter)	

Example:

```

type OneVSI {
    description = "Two hosts connected to OpenFlow switch, and a controller."

    host { id = "h1"
        location = "PRG"
        port { id = "port1" }
    }
    host { id = "h2"
        location = "PRG"
        port { id = "port1" }
    }
    host { id = "controller"
        location = "PRG"
        port { id = "port1" }
    }
    link { id = "l1"
        port { id="src" }
        port { id="dst" }
    }
    link { id = "l2"
        port { id="src" }
        port { id="dst" }
    }
    link { id = "lc"
        port { id="src" }
        port { id="dst" }
    }

    vsi { id = "vs1"
        location = "PRG"
        switchIPv4Addr="10.10.100.1"
        switchIPv4Mask="255.255.255.0"
        controller {
            ipv4="10.10.100.100"
            port="6653"}
        port { id = "port1"
            logicalPort=1 }
        port { id = "port2"
            logicalPort=2 }
        port {
            id="port3"
            mode="CONTROL"}
    }

    adjacency h1.port1, l1.src
    adjacency vs1.port1, l1.dst
    adjacency h2.port1, l2.src
    adjacency vs1.port2, l2.dst
    adjacency controller.port1, lc.src
    adjacency vs1.port3, lc.dst

```

V. External Domain

The External Domain resource [SOB-2015] represents an endpoint in some facility that is outside the GTS service area. This facility is reachable by a preprovisioned virtual circuit that has been established outside of the scope of a normal GTS testbed life cycle by the GTS service management team. To request such a preprovisioned virtual circuit, click on “Services” in the top navigation menu and then click on “Request External Domain”. In this form you can fill in the details for the preprovisioned virtual circuit and request it by sending this information to the GTS operations team.

When the virtual circuit has been provisioned, the external domain can be used with the following DSL code:

```
ExternalDomain {
    id = "ex1"
    location = "lab1"
    port { id = "ep1" }
}
```

Users may specify the following properties of the external domain:

- *id* – alphanumeric string, case sensitive; The ID is a user chosen name of the External Domain resource (unique within the scope of the Testbed DSL) that can be used within Adjacency specifications to reference a particular ED object/port pair, or for other debugging purposes during the life cycle of the testbed;
- *location* – choice of “AMS”, “BRA”, “PRG”, “LON”, “HAM”, “MIL”; case insensitive. The GTS pop location defines where the External Domain virtual circuit appears on the edge of the GTS domain.
- *Port id* - choice of “ep1”, or “ep2”. The port construct specifies a port ID that is preconfigured within GTS (by the service management team) to be the GTS endpoint for an External Domain virtual circuit. The port id must be either “ep1” or “ep2”. These port ids, at the associated location, are configured to represent different External Domains at different times. The user and the GTS service management team must coordinate to arrange for an external Domain facility to be attached to the GTS domain, and to identify the specific External Domain <location>/<portid> pair which will terminate the virtual circuit going to the particular External Domain. For convenience, the user is able to use the External Domain identifier and the port ID to reference the External domain in adjacency specifications.

To reiterate: It is the “location”/“port id” pair that maps to a particular External Domain at any particular time. And within the testbed DSL description, an External Domain resource instance can be specified and referenced much like a Host resource. Below you can find an example with a host “h1” which is defined to be in Prague that is connected to an external domain called “ProtoGENI-SL” which meets GTS in London and presents a port id “ep1” that other testbed resources can connect to. The host *h1* port *eth1* is connected via link *l1* to external domain *ProtoGENI-SL* port *ep1*.

Example:

```
ExternalExample {
  host {
    id = "h1"
    port { id = "eth1" }
    port { id = "eth2" }
  }

  link {
    id = "l1"
    port {id = "src" }
    port {id = "dst" }
  }

  ExternalDomain {
    id = "ProtoGENI-SL"
    location = "LON"
    port { id = "ep1" }
  }

  adjacency h1.eth1, l1.src
  adjacency ProtoGENI-SL.ep1, l1.dst
}
```

For more information please contact gts-operations@lists.geant.org.

VI. Bare Metal Server (BMS)

i. Introduction to Bare Metal Servers

Represents a physical server that is controlled by the testbed user. Currently the following flavors are available:

- Dell R520 (8C/16T Intel Xeon E5-2450 v2 @ 2.5 GHz, 32 Gb ECC DDR3 1600 MHz RAM, 2xSAS, 372 GB, 6.0 Gb/s HDD (must be configured as RAID0 or RAID1 in PERC H710 controller first). Servers are in AMS, LON, MIL
- Dell R520 (8C/16T Intel Xeon E5-2450 @ 2.1 GHz, 32 Gb ECC DDR3 1600 MHz RAM, 2xSAS, 372 GB, 6.0 Gb/s HDD (must be configured as RAID0 or RAID1 in PERC H710 controller first). Servers are in HAM
- Dell R530 (10C/20T Intel Xeon E5-2650 v3 @ 2.3 GHz, 32 Gb ECC DDR4 2133 MHz RAM, 2xSAS, 372 GB, 6.0 Gb/s HDD (must be configured as RAID0 or RAID1 in PERC H730 controller first). Servers are in MAD

Notes for BMS in GTS version 4.1.3:

When activated for the first time during the reservation, the BMS instance will be in its initial (“factory default”) state. This practically means that you will need to check the initial settings such as BIOS time and date, RAID configuration and other, and adjust them to your needs. See sections *Introduction to Bare Metal Servers*, *PERC H710 Mini RAID configuration* and *PERC H730 Mini RAID configuration* with examples of PERC 710/730RAID controllers configuration and screenshots of Ubuntu 16.04 server installations.

Default timezones on virtual machines are Central European Time/Central European Summer Time. Configure right time and timezone at BMS, or change timezone at VMs.

Restrictions for BMS in GTS version 4.1.3:

- In GTS version 4.1.3 only one port can be configured at each BMS for connection to other DSL resources (other BMS, VM or VSI)
- Unfortunately, the Internet connectivity for the BMS is not supported out of the box in this version of the GTS.

So in GTS version 4.1.3 the following testbed configuration is recommended for experiments with BMS: One BMS and one VM for VPN. With this configuration you can have full IP connectivity from your local machine to your bare metal server using VPN.

As a temporary solution GTS staff will also configure manually an additional management interface. In this case, please, do not deactivate/release the testbed yourself, but instead write to GTS support to do this. Before the testbed deactivation we need to manually delete any additionally configured interface on the BMS.

Parameter	Type	Description	Example
id	String <read, write, optional>	User defined identifier. If not specified it is autogenerated by the service.	s1
location	PRG, BRA, AMS, MIL,, PAR, MAD, LON <read, write, optional>	location of the bare-metal server	PRG
image	String <read, write, optional>	The Operating System image to pre-install on this bare-metal server	"ubuntu-14.04.3-server-amd64.iso"
port	Map<id, port>	Map of one port as defined by the user.	port { id = "port1" }

Example:

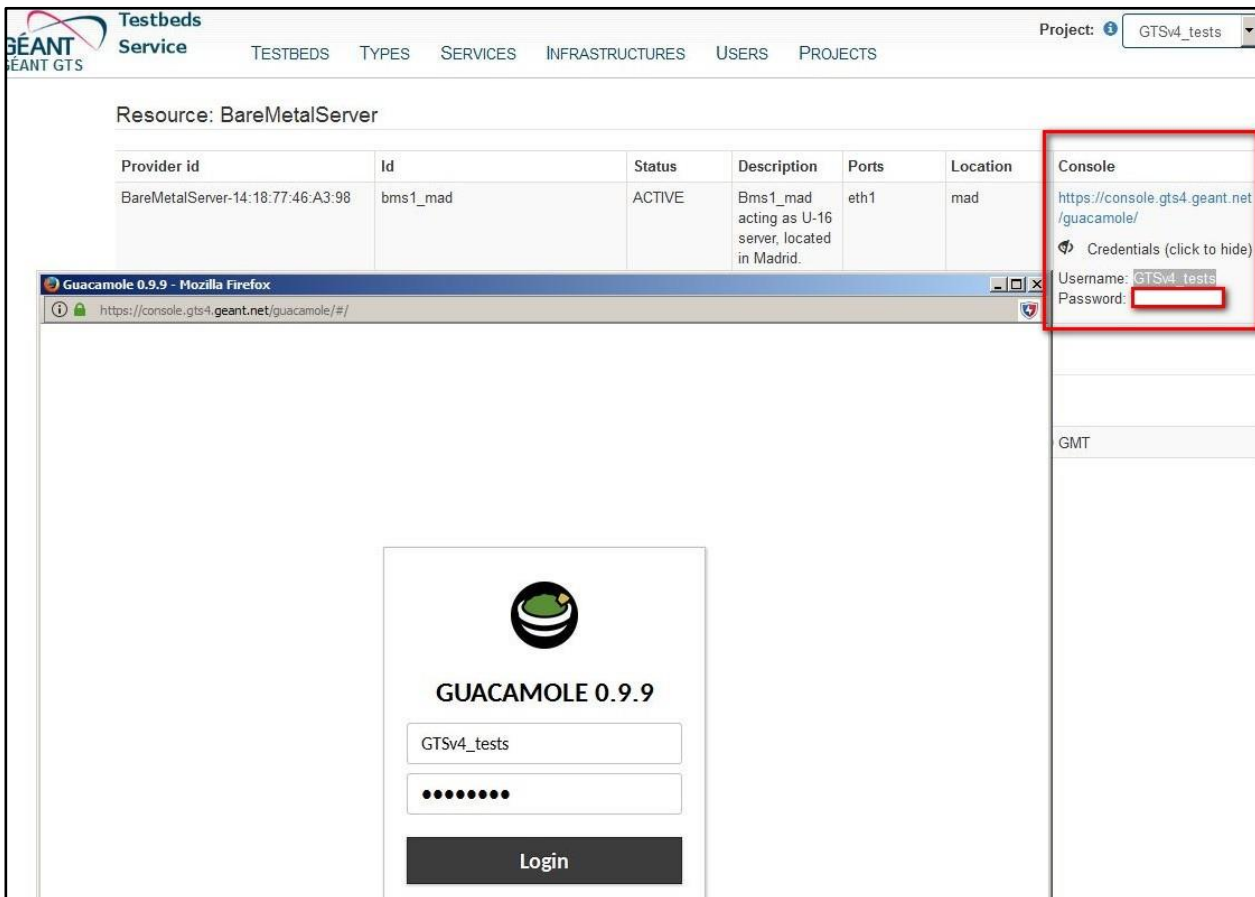
```

OneBMS_and_OneVM_in_LON {
  id = "OneBMS_and_OneVM_in_LON"
  description = "NI:BMS in LON connected to VM in LON"
  //   vm_in_lon      bms1_lon
  //   eth1 <----->eth1
  //
  baremetalserver { id = "bms1_lon"
    description = "BMS with ubuntu-16.04-server, located in London."
    location="lon"
    image = "ubuntu-16.04-server.iso"
    port { id = "eth1" }
  }
  host { id = "vm_lon"
    description = "VM located in London."
    location = "lon"
    port { id = "eth1" }
  }
  link { id = "eth1_link"
    port { id="src" }
    port { id="dst" }
  }
  adjacency eth1_link.src, bms1_lon.eth1
  adjacency eth1_link.dst, vm_lon.eth1
}

```

ii. PERC H710 Mini RAID configuration

After testbed activation open BMS console and connect to Guacamole [GUA-2017], using “Credentials”, that are below the link to Guacamole (see Figure 48):



The screenshot shows the GEANT Testbeds Service interface. At the top, there is a navigation menu with options: TESTBEDS, TYPES, SERVICES, INFRASTRUCTURES, USERS, and PROJECTS. The current project is 'GTSv4_tests'. Below the navigation, the resource 'BareMetalServer' is displayed. A table lists the resources:

Provider id	Id	Status	Description	Ports	Location
BareMetalServer-14:18:77:46:A3:98	bms1_mad	ACTIVE	Bms1_mad acting as U-16 server, located in Madrid.	eth1	mad

To the right of the table, there is a 'Console' section with the URL <https://console.gts4.geant.net/guacamole/>. Below the URL, there is a 'Credentials (click to hide)' section with the following information:

- Username: GTSv4_tests
- Password:

In the foreground, a 'Guacamole 0.9.9 - Mozilla Firefox' window is open, showing the Guacamole login page. The page has the Guacamole logo and the text 'GUACAMOLE 0.9.9'. Below this, there are two input fields: the first contains 'GTSv4_tests' and the second contains a masked password '.....'. A 'Login' button is located at the bottom of the login form.

Figure 48: Connecting to Guacamole

On BMS startup press F2 to enter DELL Server System Setup menu. When you see in console the OS installation window, select Language and then select “Boot from first hard disk.” Press any key. On boot press F2.

1. Select Device Settings (see Figure 49)
2. Select Integrated RAID Controller Dell PERC H710.
3. Select Virtual Disk Management
4. Select Create Virtual Disk

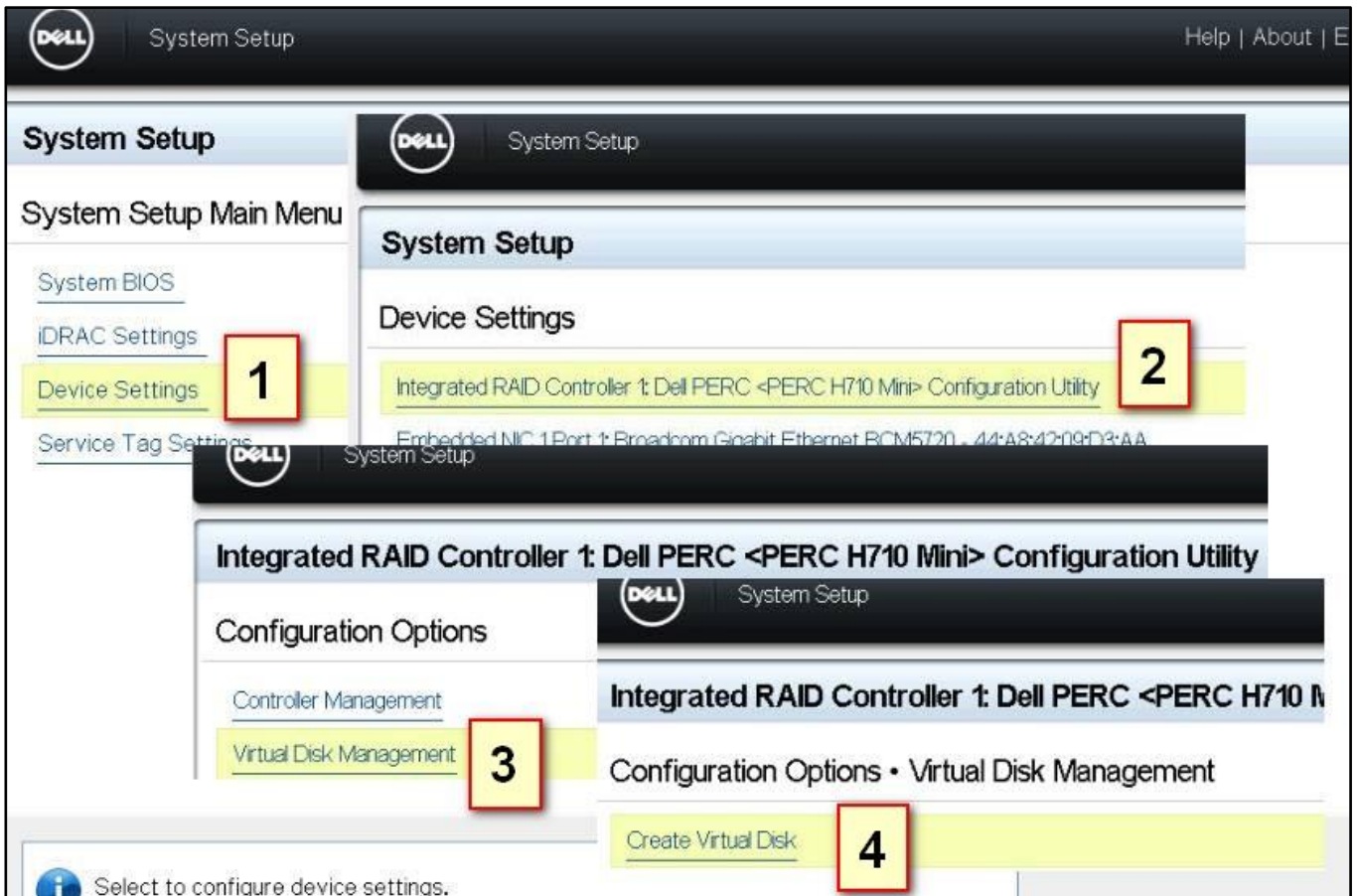


Figure 49: DELL server system setup

Then continue with the following steps as shown in Figure 50:

5. Select RAID 1
6. Select Select Physical Disks
7. Select "Both"
8. Check All, and
9. Apply Changes / Ok

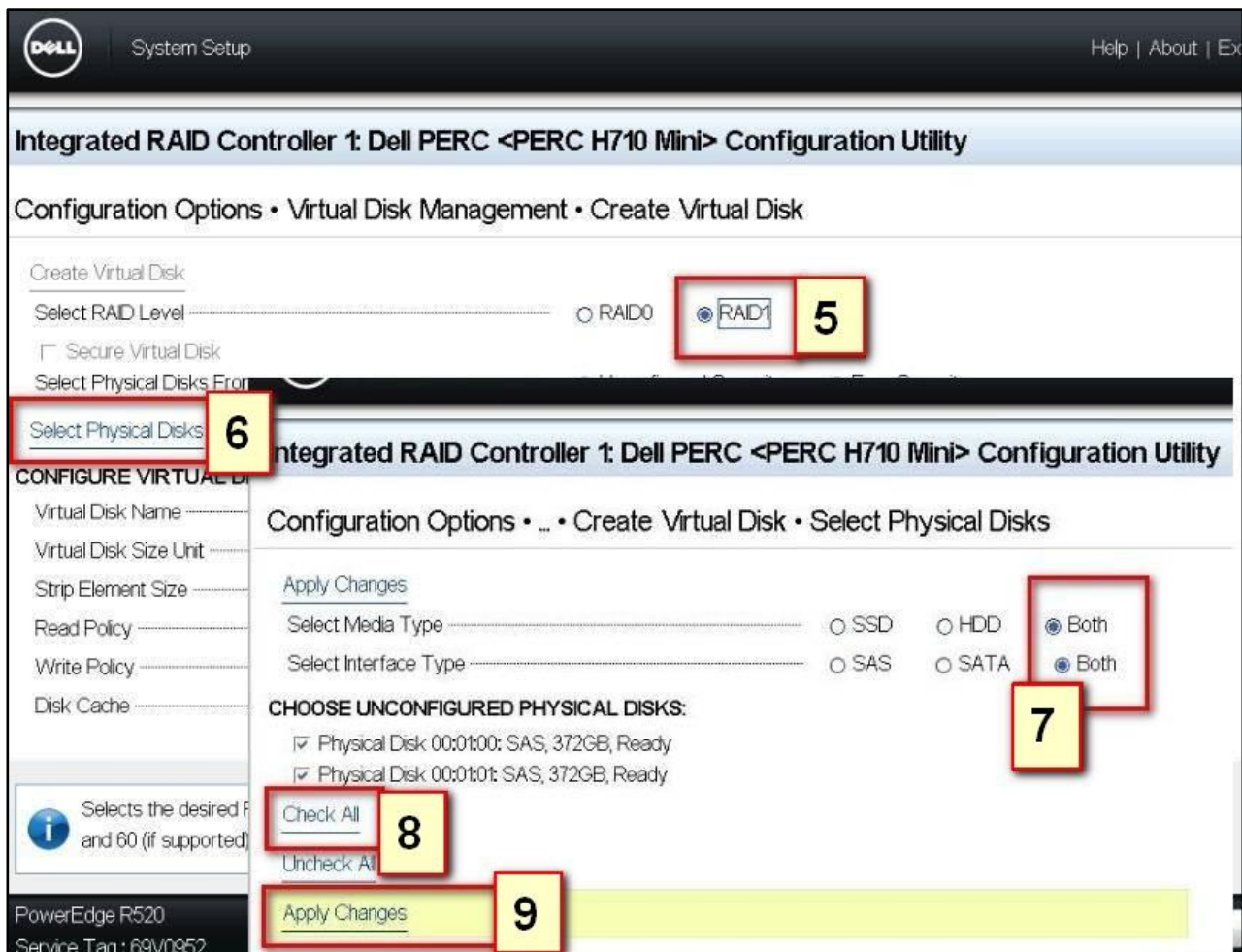


Figure 50: setup RAID PERC H710 Mini

10. Enter Virtual Disk Name
11. Click Create Virtual Disk and confirm operation (CheckBox, "Yes", "Ok")
12. Click "Back" and "Finish" several times to "Exit" from Setup.



Figure 51: set virtual disk

iii. PERC H730 Mini RAID configuration

After Testbed activation open BMS console and connect to Guacamole, using “Credentials”, that are below link to Guacamole (see Figure 48).

On BMS startup press F2 to enter DELL Server System Setup menu. When you see in console the OS installation window, select Language and then select “Boot from first hard disk.” Press any key. On boot press F2.

1. Select Device Settings.
2. Select Integrated RAID Controller Dell PERC H730.
3. Select Create Profile Based Virtual Disk
4. Select Generic RAID 1

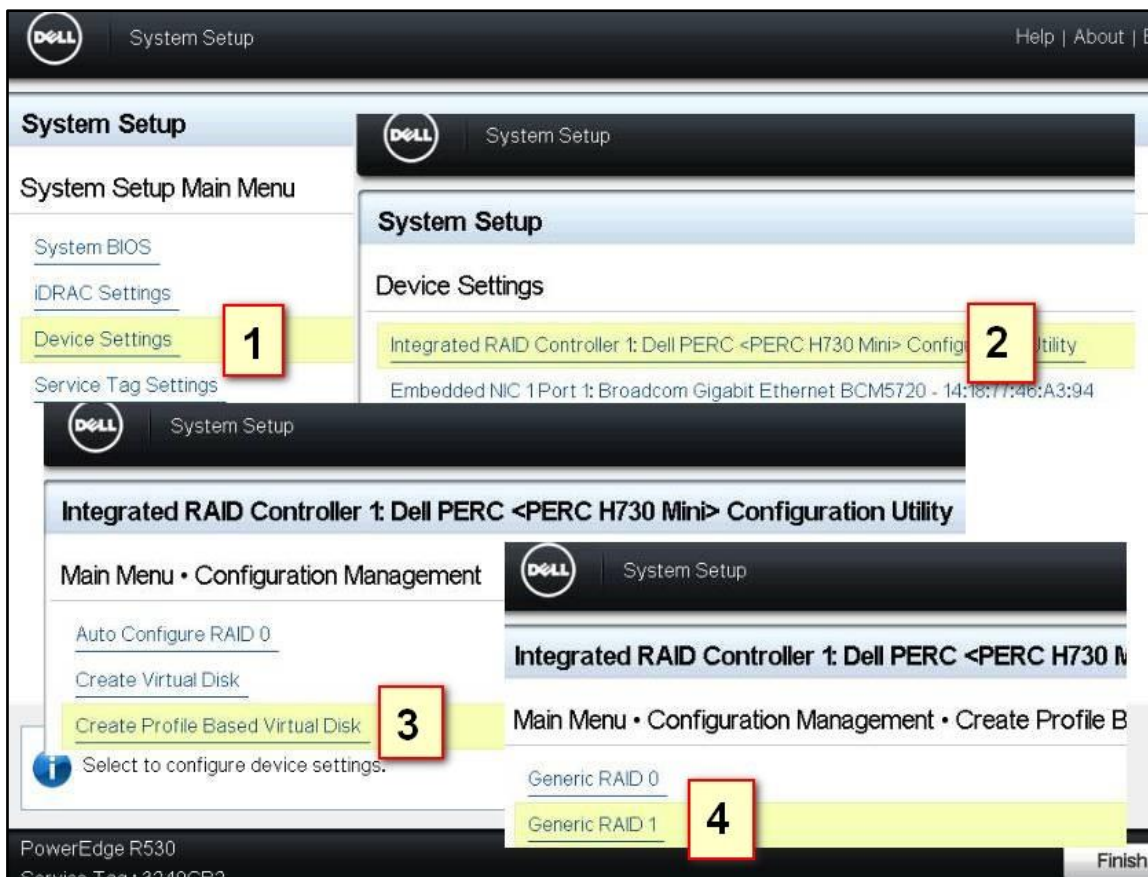


Figure 52: setup RAID PERC H370 Mini

iv. Installation of OS on BMS

After Testbed activation open BMS console and connect to Guacamole, using “Credentials”, that are below link to Guacamole (see Figure 48). Create RAID1 or RAID0 (see *PERC H710 Mini RAID configuration* and *PERC H730 Mini RAID configuration*). Without RAID configuration you can only use Live-CD image to test BMS.

OS Ubuntu 16.4 server installation example:

1. Select Language in next window click “Install Ubuntu Server”. If you do not configure RAID, select in menu “Boot from first hard disk” and press any key. Go to sections *PERC H710 Mini RAID configuration* or *PERC H730 Mini RAID configuration*
2. Select Language, Country (use other to select country in Europe). *Note: Default timezones at Virtual Machines are Central European Time/Central European Summer Time. Configure right time and timezone at BMS, or change timezone at VMs.*
3. Select Country to base default locale settings on.
4. “No” or detect keyboard layout. Use “Tab” or arrows to change selected items in menu.
5. Select “Country of origin for the keyboard” and keyboard layout.
6. Wait. Reconnect Guacamole if needed.
7. Configure the network. Select first 10 Gb Ethernet interface for configuration. Use settings, that you plan for BMS port in your network configuration. *Note: in GTS v 4.1.3 only one port at BMS can be configured.*

```

[!!!] Configure the network

Your system has multiple network interfaces. Choose the one to use as the primary network
interface during the installation. If possible, the first connected network interface
found has been selected.

Primary network interface:

eno1: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
eno2: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
enp10s0f0: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet
enp10s0f1: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet
enp12s0f0: Broadcom Corporation NetXtreme BCM5719 Gigabit Ethernet PCIe
enp12s0f1: Broadcom Corporation NetXtreme BCM5719 Gigabit Ethernet PCIe
enp12s0f2: Broadcom Corporation NetXtreme BCM5719 Gigabit Ethernet PCIe
enp12s0f3: Broadcom Corporation NetXtreme BCM5719 Gigabit Ethernet PCIe

```

8. At this point, you can choose to "Configure network manually" or leave this for later by selecting the "Do not configure network at this time."
9. Regardless of your choice in step 8, the gateway and the DNS server should remain unspecified as this is just a point-to-point link.
10. Enter host name, skip domain name, enter user name and password.
11. Advice: Do not “Encrypt your home directory”
12. Unmount mounted partitions if exist.
13. Select “Partitioning method” and select disk to partition.
14. Remove if exist existing logical volume data. Write changes to disk.
15. Select volume group size. Write changes to disk.
16. Blank for no proxy.
17. Select “No automatic updates”.
18. Choose software to install (press blank to select). Select OpenSSH server necessarily.
19. Install the GRUB boot loader to the master boot record.

20. Continue.
21. Installation complete. Wait for system to boot.
22. If OS does not boot and you see in console the OS installation window, select Language and then select "Boot from first hard disk." Press any key. On boot press F2 and change in BIOS boot order to "Hard drive C" first:
System BIOS/Boot Settings/BIOS Boot Settings/Boot Sequence/Change order/Hard drive C/+ to move to first position. Back/Back/Finish/Yes/Ok/Finish/Yes.

VII. BNF Grammar

<property> ::= <key>=<value>

<properties> ::= <property>(;|\n)<properties>

<port> ::= port { <properties> }

<atomic> ::= <atomic-type> { <properties> <ports> }

<atomic-type> ::= host | link | ofx | vsi | baremetalserver

<composite> ::= type <name> { <properties> <resources> <ports> <adjacencies> }

<resources> ::= <resource> <resources>

<resource> ::= <composite> | <atomic>

<adjacencies> ::= <adjacency><adjacencies>

<adjacency> ::= adjacency <port-id>, <port-id>

Appendix II: Additional examples

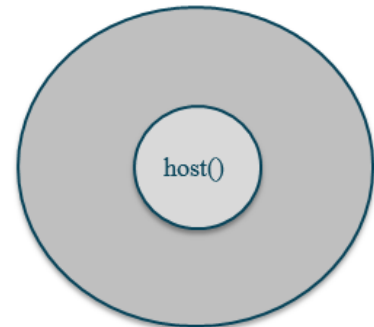
I. Examples: One host

The following example shows how one host can be described with DSL:

```
testbed {
  id = "OneHost"

  host {
  }
}
```

testbed ("OneHost")

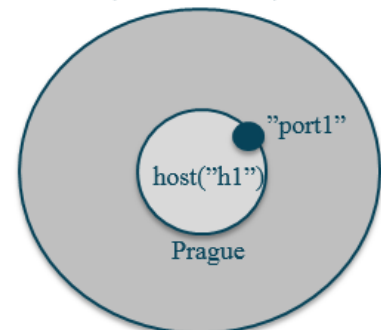


If you need to have one host at a specific location for your experiment then you must specify this in your DSL code:

```
testbed {
  id = "OneHost"
  description = "One host in Prague"

  host {
    id = "h1"
    location = "PRG"
    port { id = "port1" }
  }
}
```

testbed ("OneHost")



II. Example: Two hosts linked together

This example shows in two steps how you can define two hosts and a link and connect them to each other:

Step 1: Describe both hosts and the link:

```
HostsLine {
  description = "PRG host linked with
  BRA host"

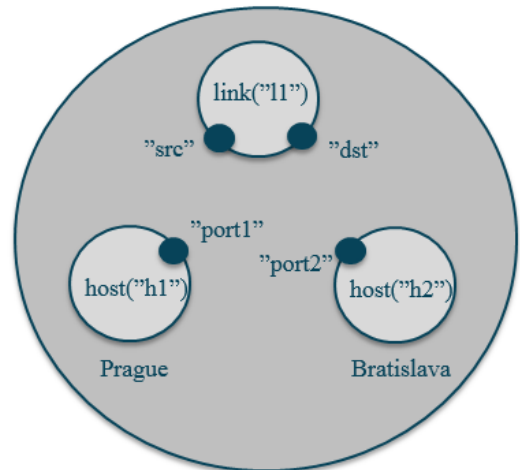
  host {
    id = "h1"
    location = "PRG"
    port { id = "port1" }
  }

  host {
    id = "h2"
    location = "BRA"
    port = { id = "port2" }
  }

  link {
    id = "l1"
    port { id = "src" }
    port { id = "dst" }
  }

  ...
}
```

HostsLine ()



Step 2: Describe the adjacencies to actually connect the entities:

```

HostsLine {
  description = "PRG host linked with BRA
host"

  host {
    id = "h1"
    location = "PRG"
    port { id = "port1" }
  }

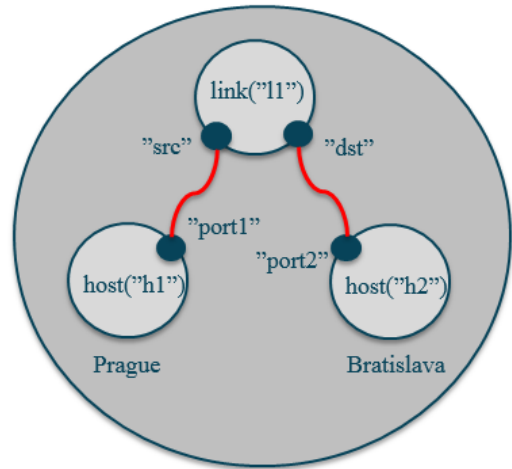
  host {
    id = "h2"
    location = "BRA"
    port { id = "port2" }
  }

  link {
    id = "l1"
    port { id = "src" }
    port { id = "dst" }
  }

  adjacency h1.port1, l1.src
  adjacency h2.port2, l1.dst
}

```

HostsLine ()



III. Example: Triangle between three locations

For a triangle to be set up between three different locations you need to first describe the three hosts and specify their locations in your DSL code:

Step 1: Describe hosts and locations:

```

type triangle {
  description = "Triangle between PRG,
  BRA, and MAD"

  host {
    id = "h1"
    location = "PRG"
    port { id = "port11" }
    port { id = "port12" }
  }

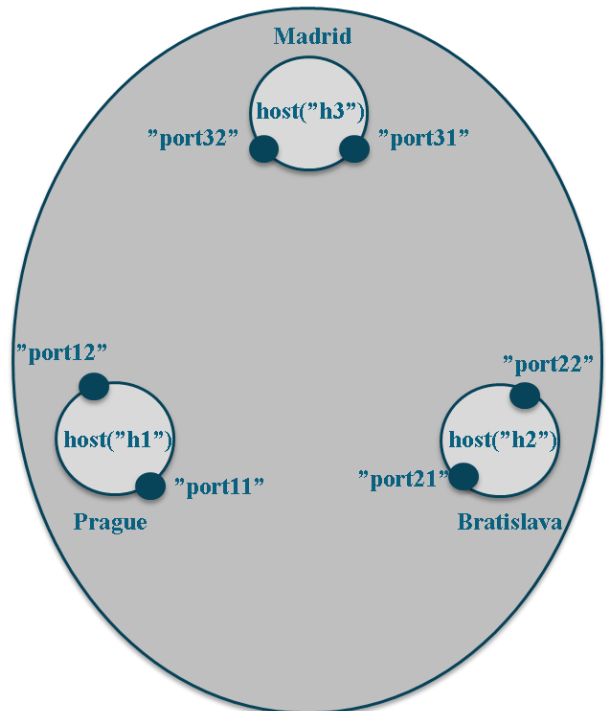
  host {
    id = "h2"
    location = "BRA"
    port { id = "port21" }
    port { id = "port22" }
  }

  host {
    id = "h3"
    location = "MAD"
    port { id = "port31" }
    port { id = "port32" }
  }

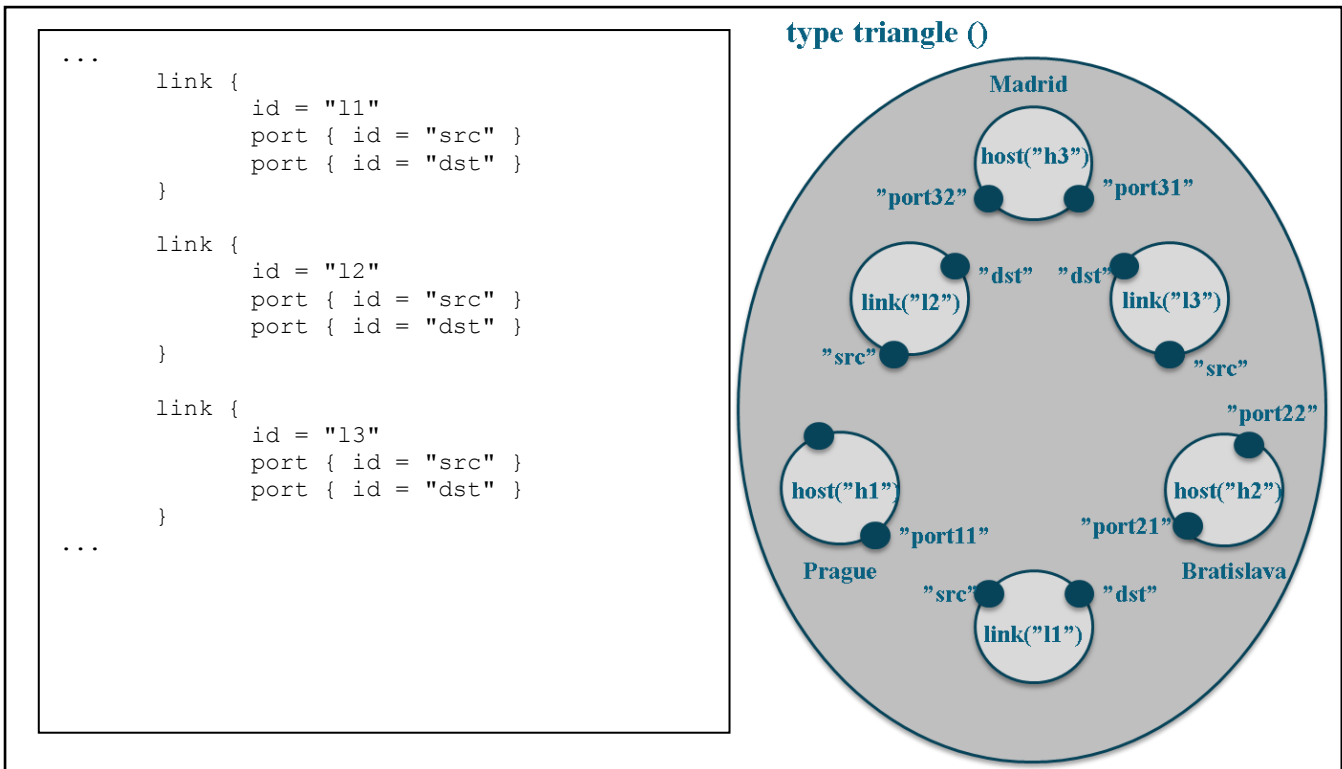
  ...
}

```

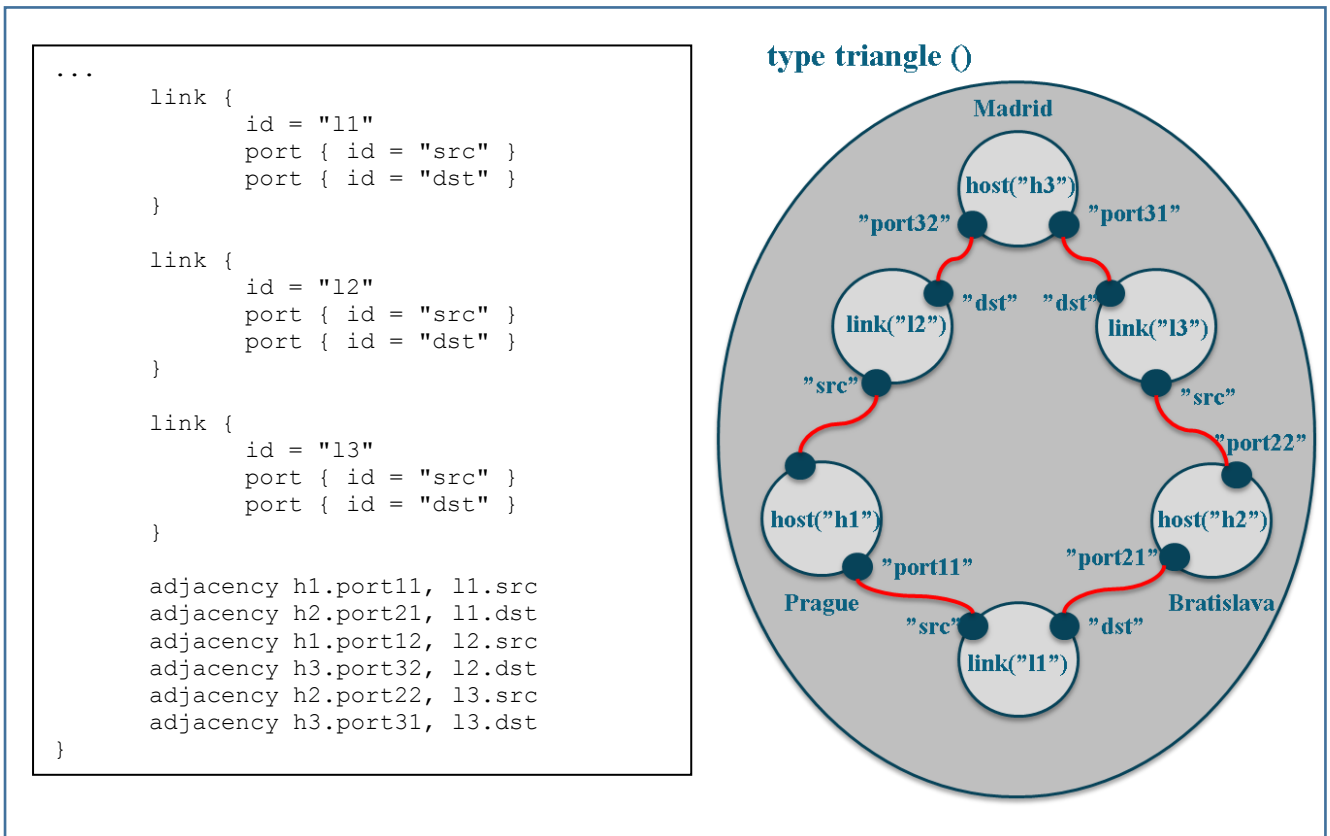
type triangle ()



Step 2: Describe the three links between them:

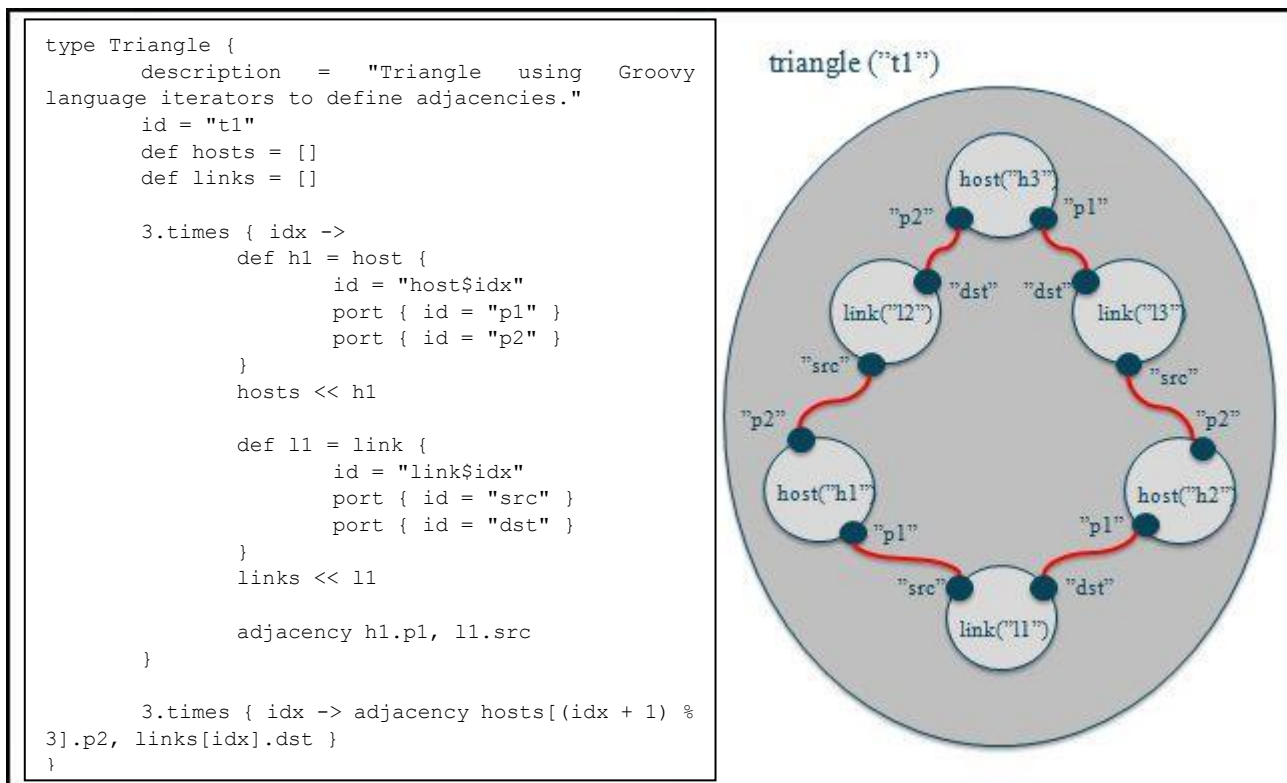
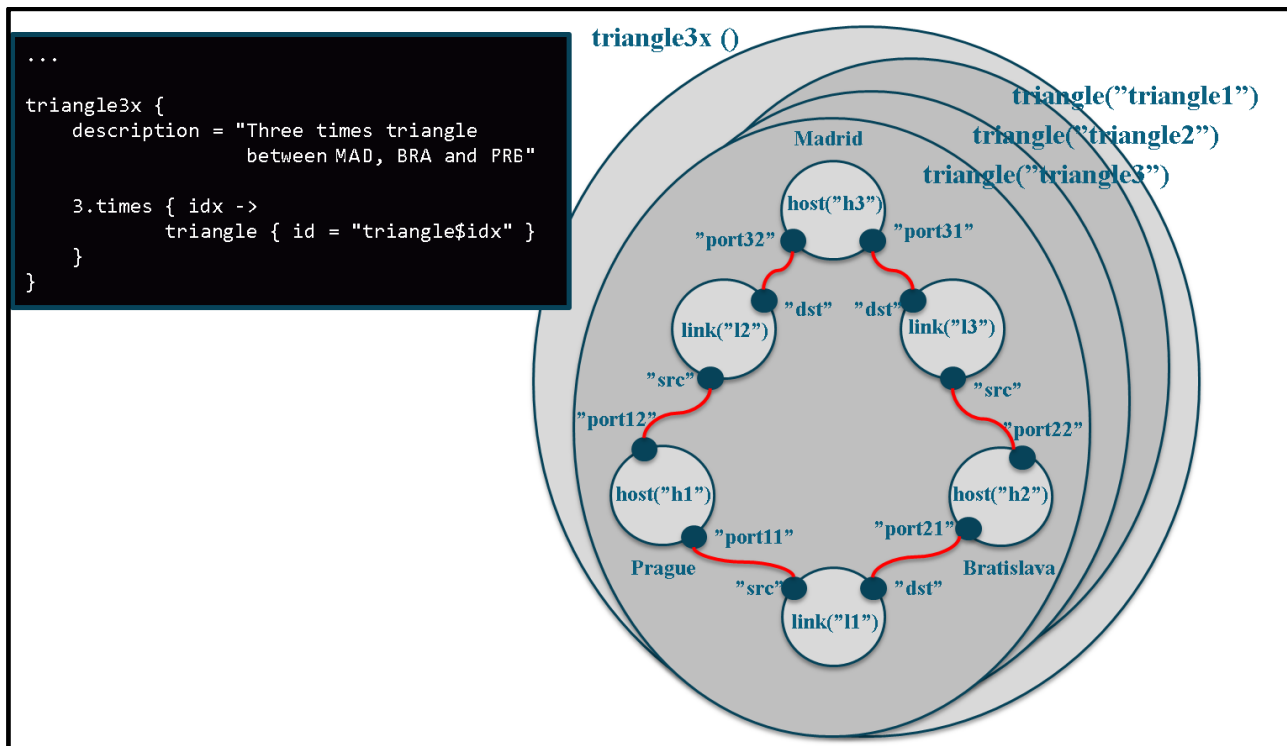


Step 3: Describe the adjacencies to connect them:



IV. Example: Three triangles built using DSL code iterations

The following describes three triangles being setup between Prague, Bratislava and Madrid:



V. Example: Two OpenFlow switches (VSI), each with two ports, one host and one separate controller

```

twoCorsaVSIController2Hosts {
  description = "Two Corsa VSI, each with two ports, one host and a controller"
  id = "twoVsiCorsaTst"
  host {
    id="host1"
    port { id="port1" }
  }
  host {
    id="host2"
    port { id="port1" }
  }
  host {
    id="controller"
    port { id="port1" }
    port { id="port2" }
  }
}

vsi {
  id="vsi1"
  location="HAM"
  switchIPv4Addr="10.10.100.1"
  switchIPv4Mask="255.255.255.0"
  switchMode="hard"

  controller {
    ipv4="10.10.100.100"
    port="6653"
  }

  port {
    id="port1"
    logicalPort=1}
  port {
    id="port2"
    logicalPort=2}

  port {
    id="port9"
    mode="CONTROL"
  }
}

vsi {
  id="vsi2"
  location="MAD"
  switchIPv4Addr="10.10.101.1"
  switchIPv4Mask="255.255.255.0"
  switchMode="hard"

  controller {
    ipv4="10.10.100.100"
    port="6653"
  }

  port {
    id="port1"
    logicalPort=1
  }
}

```

```
port {
  id="port2"
  logicalPort=2
}

port {
  id="port9"
  mode="CONTROL"
}

link {
  id="h1vsi"
  port { id="src" }
  port { id="dst" }
}

link {
  id="h2vsi"
  port { id="src" }
  port { id="dst" }
}

link {
  id="vsilvsi2"
  port { id="src" }
  port { id="dst" }
}

link {
  id="controllervsi1"
  port { id="src" }
  port { id="dst" }
}

link {
  id="controllervsi2"
  port { id="src" }
  port { id="dst" }
}

adjacency host1.port1, h1vsi.src
adjacency vsi1.port1, h1vsi.dst
adjacency host2.port1, h2vsi.src
adjacency vsi2.port1, h2vsi.dst
adjacency vsi1.port2, vsilvsi2.src
adjacency vsi2.port2, vsilvsi2.dst
adjacency controller.port1, controllervsi1.src
adjacency vsi1.port9, controllervsi1.dst
adjacency controller.port2, controllervsi2.src
adjacency vsi2.port9, controllervsi2.dst
}
```

VI. Example: One host directly connected with a Bare Metal Server

```
OneHostBms {
  id = "OneHostBMS"

  host {
    id="h1"
    port { id="port1" }
  }

  baremetalserver {
    id = "bms1"
    cpuCores = 2
    image = "imageID"
    port { id = "port2" }
  }

  link {
    id="l1"
    port { id="src" }
    port { id="dst" }
  }

  adjacency h1.port1, l1.src
  adjacency bms1.port2, l1.dst
}
```

References

- CZI-2016 Cziva R., Sobieski J., Kumar Y., High-Performance Virtualized SDN Switches for Experimental Network Testbeds, SC16 – INDIS, Austin, TX, USA, Nov. 8, 2016, https://scinet.supercomputing.org/workshop/sites/default/files/INDIS16_paper01.pdf
- DEL-D61 Deliverable D6.1 (DS2.3.1): Architecture Description: Dynamic Virtualised Packet Testbeds Service
- DLL-2017 <http://www.dell.com>.
- FAR-2014 F. Farina, P. Szegedi, J. Sobieski, GÉANT World Testbed Facility - Federated and distributed Testbeds as a Service facility of GÉANT, published at FIDC 2014 in Karlskrona, Sweden, 2014
- GUA-2017 Apache Guacamole Clientless Remote Desktop Gateway, <https://guacamole.incubator.apache.org/>.
- GRO-2014 GROOVY, <http://groovy.codehaus.org/>
- NAE-2016 Naegele-Jackson S., Sobieski J., Gutkowski J., Hažlinský M., Creating Automated Wide-Area Virtual Networks with GTS – Overview and Future Developments, 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) (Luxembourg City, 12/12/16 - 12/15/16)
- RYU-2014 RYU Project Team, RYU SDN Framework, <http://osrg.github.io/ryu-book/en/Ryubook.pdf>, available on September 29, 2014
- SOB-2015 J. Sobieski, S. Naegele-Jackson, B. Pietrzak, M. Hazlinsky, F. Farina and K. Kramaric, GÉANT Testbeds Service (GTS) - GÉANT Testbed Service - External Domain Ports: A demo on multiple domain connectivity, European Workshop on Software Defined Networks (EWSDN), Bilbao, Sept. 30 - Oct. 2, 2015
- SZE-2014 P. Szegedi, User's Training 1 – The Basics.