

LIGHTNING PROFESSOR MANUAL

Lightning 0.8.6

12/09/2018

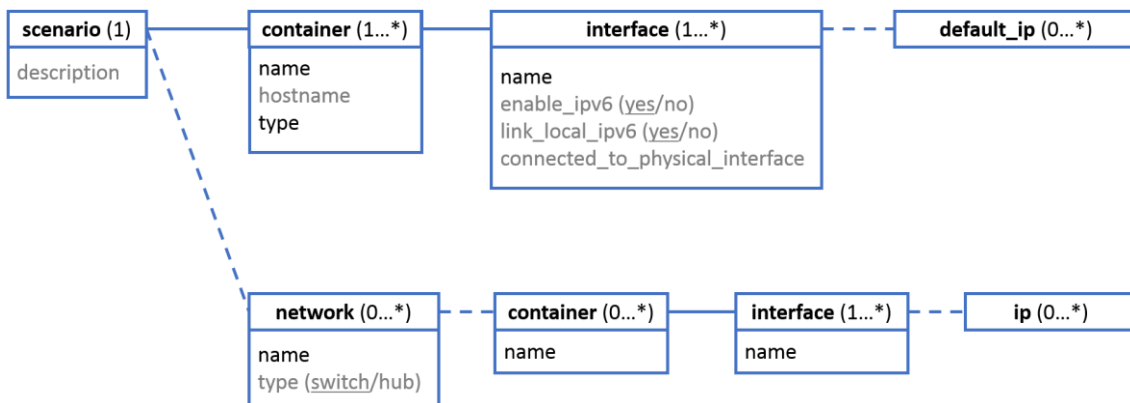
This professor manual will explain how to create repositories of personalized scenarios and Docker images to be used with **Lightning** (<https://github.com/ptoribi/lightning>), the network simulator based on Docker containers. It also explains how to create new scenario files that comply with the Lightning XML Schema Definition.

For installing the software please refer to the “*Installation Manual*”. For information about how to use this program, please take a look to the “*User Manual*”.

Write your own scenario files (XML / direct execution)

Every **XML scenario file** must have a certain syntax and comply with the Lightning XSD (it is defined in the file `lightning-xml-schema.xsd`, you can find it in the installation directory).

The elements and attributes that can populate your scenario file are graphically represented in the following diagram:



The upper part of each box shows the element name, and the rest of the box contains its attributes. The attributes in black are mandatory and the ones in grey, optional. When the name is followed by several options separated by slashes (option1/option2/optionN), the attribute must be constrained to one of those values, appearing underlined in the diagram the default one in case the attribute is not present in your scenario file.

The diagram has to be readen this way: each file must have only one scenario, with an optional description attribute, each scenario must have one or more containers inside with a mandatory name and type and an optional hostname attributes; each scenario can also have networks with a mandatory name and an optional type attributes, if the type attribute is not specified, the default value is set to “switch”... and so on.

Scenarios can also be defined in a **direct execution scenario file**, where instead of using a markup language, functions are used directly (native Lightning functions and personalized ones stored in the files `functions` and `personalized_functions` respectively).

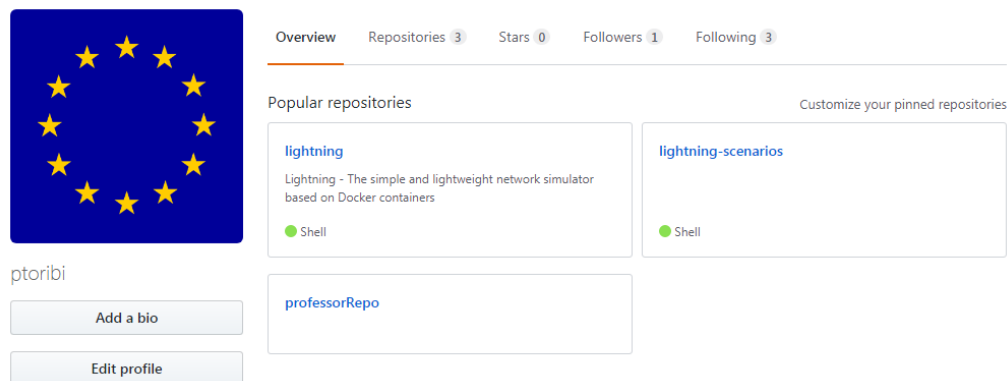
At the end of this manual you could find two examples of Lightning scenarios, each one defined in XML [I] [II] and direct execution files [I] [II].

Use your own scenarios (stored in the cloud)

- **Sign in and create a Git repository**

First, you will have to create an account in one of the several Git servers that offer this service, or to use a local Git server already deployed in your organization. Probably the most famous online Git repository is [GitHub](#).

It is up to you choose one Git server provider, create an account and start a new repository.



Online Git server account with some repositories already created

As an example, in the next steps of this guide we will assume that the repository used by the professor is:

<https://github.com/ptoribi/professorRepo.git>

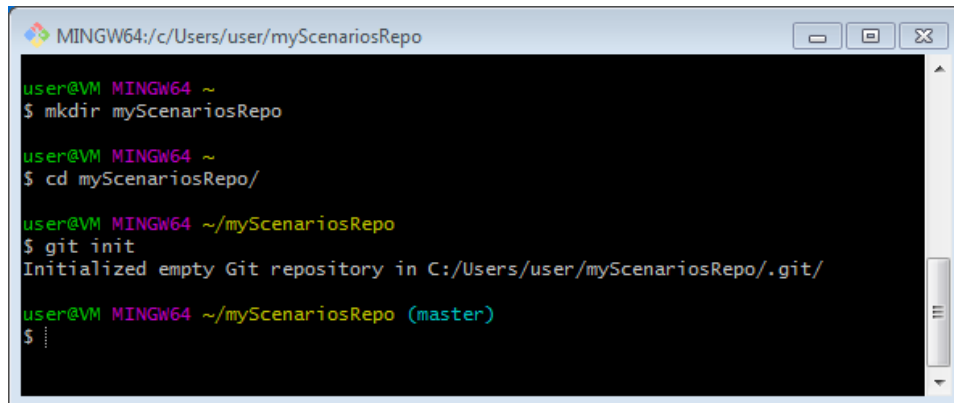
- **Install the Git version control software in your computer**

Please download the proper package for your operating system and follow the installation instructions that you will find in the Git Downloads page: <https://git-scm.com/downloads>

- **Create a git local repository in your machine**

Create a folder in your machine, go inside that folder and initialize a new Git repository:

```
$ mkdir FOLDER_NAME
$ cd FOLDER_NAME
$ git init
```

A screenshot of a terminal window titled 'MINGW64:/c/Users/user/myScenariosRepo'. The terminal shows the following commands and output: 'user@VM MINGW64 ~', '\$ mkdir myScenariosRepo', 'user@VM MINGW64 ~', '\$ cd myScenariosRepo/', 'user@VM MINGW64 ~/myScenariosRepo', '\$ git init', 'Initialized empty Git repository in C:/Users/user/myScenariosRepo/.git/', 'user@VM MINGW64 ~/myScenariosRepo (master)', '\$...'.

```
user@VM MINGW64 ~
$ mkdir myScenariosRepo

user@VM MINGW64 ~
$ cd myScenariosRepo/

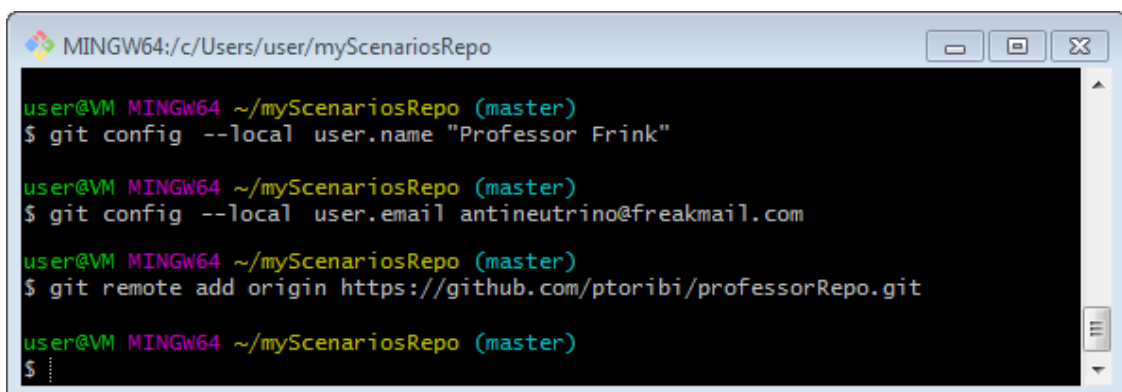
user@VM MINGW64 ~/myScenariosRepo
$ git init
Initialized empty Git repository in C:/Users/user/myScenariosRepo/.git/

user@VM MINGW64 ~/myScenariosRepo (master)
$ ...
```

Your local Git repository needs some basic configurations before working (this step only has to be done once):

```
$ git config --local user.name "Your Name"
$ git config --local user.email yourEmail@server.com
$ git remote add origin
https://github.com/ptoribi/professorRepo.git
```

The argument `--local` means that data will only be used for that repository.

A screenshot of a terminal window titled 'MINGW64:/c/Users/user/myScenariosRepo'. The terminal shows the following commands and output: 'user@VM MINGW64 ~/myScenariosRepo (master)', '\$ git config --local user.name "Professor Frink"', 'user@VM MINGW64 ~/myScenariosRepo (master)', '\$ git config --local user.email antineutrino@freakmail.com', 'user@VM MINGW64 ~/myScenariosRepo (master)', '\$ git remote add origin https://github.com/ptoribi/professorRepo.git', 'user@VM MINGW64 ~/myScenariosRepo (master)', '\$...'.

```
user@VM MINGW64 ~/myScenariosRepo (master)
$ git config --local user.name "Professor Frink"

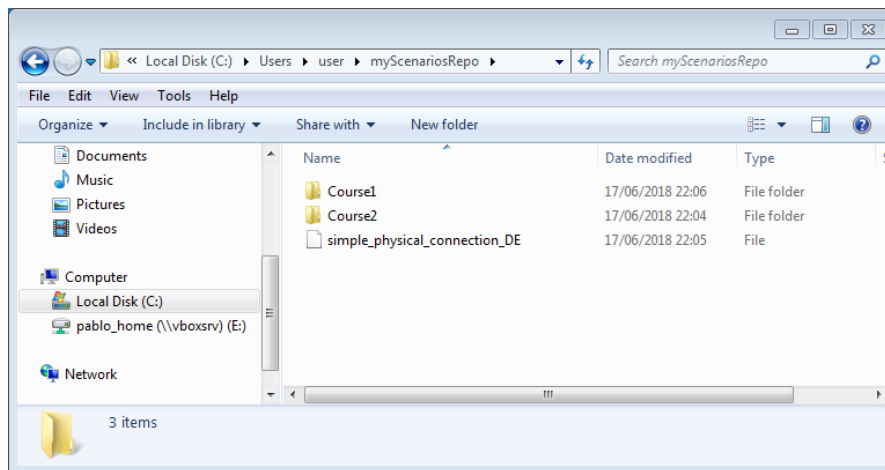
user@VM MINGW64 ~/myScenariosRepo (master)
$ git config --local user.email antineutrino@freakmail.com

user@VM MINGW64 ~/myScenariosRepo (master)
$ git remote add origin https://github.com/ptoribi/professorRepo.git

user@VM MINGW64 ~/myScenariosRepo (master)
$ ...
```

Now is time to populate your new repository with scenario files. It is possible to classify your network scenarios in folders inside your repository. You can find some sample scenarios in the Lightning example repository:

<https://github.com/ptoribi/lightning-scenarios>



- **Apply changes and upload to server**

Once you have finished, it is time to add the files to the repository and commit the changes:

```
$ git add .
$ git commit -m "Your comment describing the changes"
```

```

MINGW64:/c/Users/user/myScenariosRepo
user@VM MINGW64 ~/myScenariosRepo (master)
$ git add .
warning: LF will be replaced by CRLF in Course1/S1.xml.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in Course2/simple.xml.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in simple_physical_connection_DE.
The file will have its original line endings in your working directory.

user@VM MINGW64 ~/myScenariosRepo (master)
$ git commit -m "Scenarios for next networking lab session"
[master (root-commit) b09606c] Scenarios for next networking lab session
3 files changed, 149 insertions(+)
create mode 100644 Course1/S1.xml
create mode 100644 Course2/simple.xml
create mode 100644 simple_physical_connection_DE

```

Finally, the file changes in your repository must be uploaded:

```
$ git push origin master
```

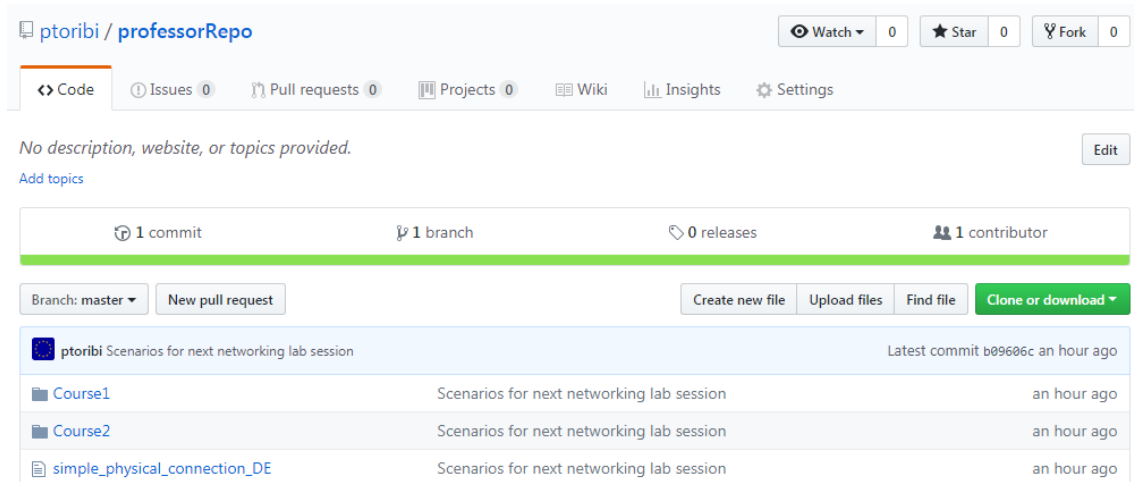
```

MINGW64:/c/Users/user/myScenariosRepo
user@VM MINGW64 ~/myScenariosRepo (master)
$ git push origin master
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 1.50 KiB | 765.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/ptoribi/professorRepo.git
 * [new branch]      master -> master

```

- **Use your new scenarios repository with Lightning**

Now you can browse your repository to check that everything worked properly:



Please ask the system administration of your organization to set the value of your personalized repository in the `variables.conf` file of the Lightning installations:

```
# Git repository for SCENARIOS
GIT_REPO_SCENARIOS="https://github.com/ptoribi/professorRepo.git"
```

After changing the value, a “lightning update” has to be performed on each machine, so the new scenarios are downloaded and installed into the system. The next time Lightning will be executed, the new scenarios will be available:

```
Terminal
File Edit View Search Terminal Help
student@uc3m:~$ lightning

*****
**  Universidad Carlos III de Madrid (UC3M)  **
**                Lightning 0.8.5             **
*****

[lightning] usage:
lightning          -> display usage message and list available scenarios
lightning stop     -> close running scenario
lightning purge    -> ONLY use this option in case "stop" doesn't work
lightning start SCENARIO -> start selected network scenario
lightning update   -> update Lightning to the latest version available

Available scenarios:

simple_physical_connection_DE - Example (DE): scenario with one PC connected to the physical host interface enp0s3
Course1/S1 - Scenario for Lab. Session N°1
Course2/simple - Simple scenario with 2 PCs

Your shared folder between host systems is located
at /home/student/lightning-shared-folder in your main system, and will
be mounted as /home/student inside the simulated host systems.

student@uc3m:~$
```

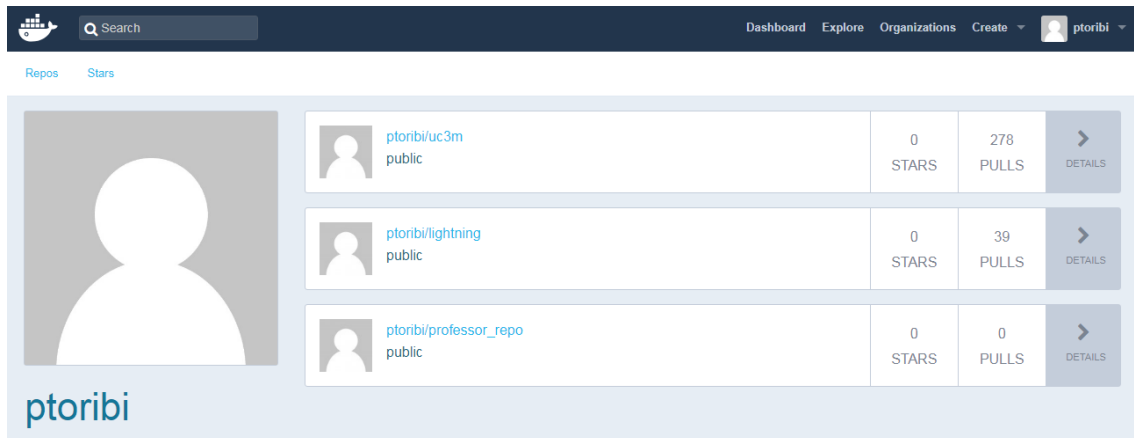
Use your own Docker images

- **Signing in and creating a Docker registry**

A place for hosting Docker images is called a *registry*. First, you have to create an account in one of the several Docker registries that offer this service. Probably the most famous online Docker registry is [Docker Hub](#).

It is up to you chose one Docker registry provider, create an account and start a new repository.

As an example, in the next steps of this guide we will assume that the repository used by the professor is: `ptoribi/professor_repo`



Online Docker registry account with some repositories already created

- **Install the Docker Community Edition software in your computer**

Please download the proper package for your operating system and follow the installation instructions that you will find in the Docker Documentation page: <https://docs.docker.com/>

In the left panel go to “*Get Docker*” → “*Docker CE*” → Select your operating system and follow the guide.

- **Get the Lightning Docker base image**

For the time being there are available two Docker images for Lightning, called “host” and “router”. For this example, it will only be used the first one, but the procedure is analogous for the other.

The first step will be to download and install the Docker image in the system:

```
# docker pull ptoribi/lightning:host
```

- **Create a container of that image**

Now that we have the Docker image, containers based on it can be created. We will create one called `miContainerTipoHost` for performing modifications:

```
# docker run -it --privileged -v /home/professor:/mnt --name  
miContainerTipoHost ptoribi/lightning:host
```

The `--privileged` argument will start the container in privileged mode for performing all type of changes without any restriction. The argument `-v` will mount the directory `/home/professor` of the main operating system into the `/mnt` directory of the container, this can be used for saving files into the container or installing software packages.

Once all the modifications needed are performed, we can simply exit the container by typing “`exit`”:

```
[root@container]# exit
```

Now the container has stopped but its file remains in the system. For making further changes in the future it can be run again by executing:

```
# docker start -i miContainerTipoHost
```

- **Create a new image from the modified container**

An image can be created by packing that modified container. The name of the Docker repository has to be specified followed by a tag that names the image, in that case, “`host`”:

```
# docker commit miContainerTipoHost  
ptoribi/professor_repo:host
```

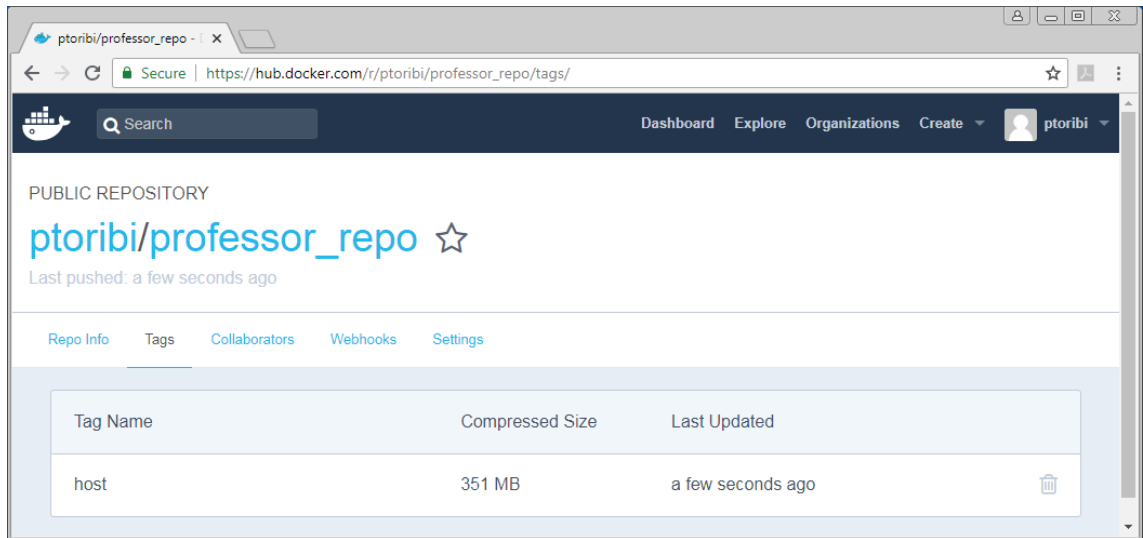
For uploading the changes to the registry, first we have to login into the service:

```
# docker login
```

And finally, uploading the modified image:

```
# docker push ptoribi/professor_repo:host
```

Now you can browse your repository to check that everything worked properly:



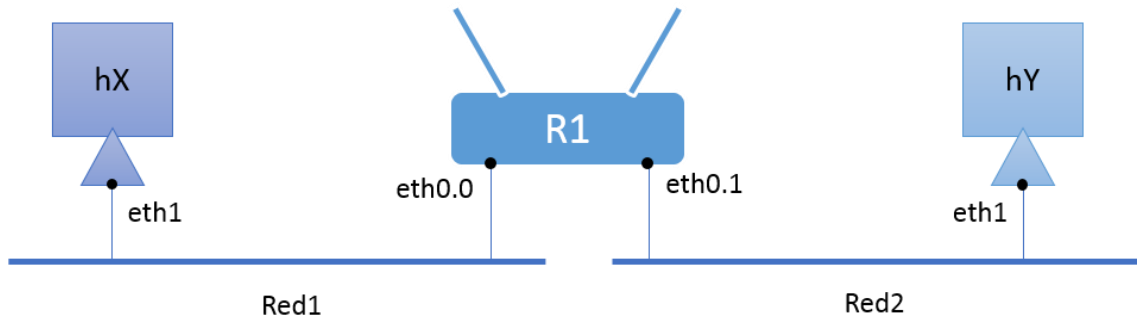
Please ask the system administration of your organization to set the value of your personalized repository in the `variables.conf` file of the Lightning installations:

```
# Docker images repository
DOCKER_IMAGE_host="ptoribi/professor_repo:host"
```

After changing the value, a “lightning update” has to be performed on each machine, so the new Docker images are downloaded and installed into the system. The next time Lightning will be executed, it will use the new filesystems.

EXAMPLE FILE I (XML)

Network scenario that defines two networks connected with a router. Each network is connected to a host network interface and a router's network interface:

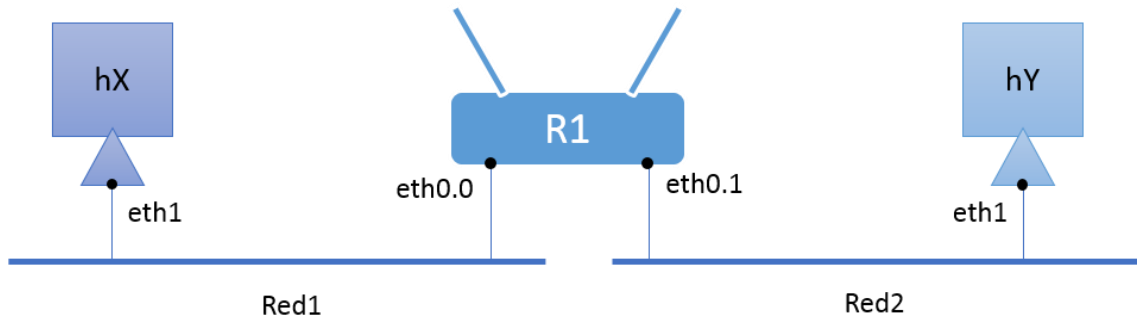


```
1. <?xml version="1.0" encoding="UTF-8" ?>
2.
3. <scenario description="Simple scenario with 2 PCs and one router"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:noNamespaceSchemaLocation="lightning-xml-schema.xsd">
6.
7.   <container name="hX" hostname="hostX" type="host">
8.     <interface name="eth1" enable_ipv6="yes" link_local_ipv6="no"/>
9.   </container>
10.
11.  <container name="hY" hostname="hostY" type="host">
12.    <interface name="eth1" enable_ipv6="yes" link_local_ipv6="no"/>
13.  </container>
14.
15.  <container name="R1" hostname="Router1" type="router">
16.    <interface name="eth0.0" enable_ipv6="yes" link_local_ipv6="no">
17.      <default_ip>192.168.0.1/24</default_ip>
18.    </interface>
19.    <interface name="eth0.1" enable_ipv6="yes" link_local_ipv6="no">
20.      <default_ip>192.168.1.1/24</default_ip>
21.    </interface>
22.    <interface name="eth0.2" enable_ipv6="yes" link_local_ipv6="no">
23.      <default_ip>192.168.2.1/24</default_ip>
24.    </interface>
25.    <interface name="eth0.3" enable_ipv6="yes" link_local_ipv6="no">
26.      <default_ip>192.168.3.1/24</default_ip>
27.    </interface>
28.    <interface name="eth0.4" enable_ipv6="yes" link_local_ipv6="no">
29.      <default_ip>192.168.4.1/24</default_ip>
30.    </interface>
31.    <interface name="wlan0" enable_ipv6="yes" link_local_ipv6="no">
32.      <default_ip>192.168.5.1/24</default_ip>
33.    </interface>
34.  </container>
35.
36.
37.  <network name="Red1" type="switch">
38.
39.    <container name="hX">
40.      <interface name="eth1">
```

```
41.     <ip>192.100.100.101/24</ip>
42.     </interface>
43. </container>
44.
45.     <container name="R1">
46.         <interface name="eth0.0"/>
47.     </container>
48.
49. </network>
50.
51.
52. <network name="Red2" type="switch">
53.
54.     <container name="hy">
55.         <interface name="eth1">
56.             <ip>192.100.100.102/24</ip>
57.         </interface>
58.     </container>
59.
60.     <container name="R1">
61.         <interface name="eth0.1"/>
62.     </container>
63.
64. </network>
65.
66.
67. </scenario>
```

EXAMPLE FILE I (DIRECT EXECUTION)

Network scenario that defines two networks connected with a router. Each network is connected to a host network interface and a router's network interface:

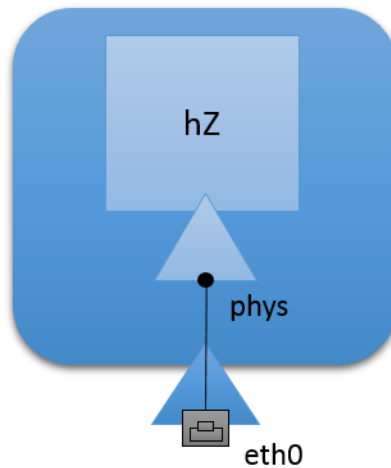


```
1. #!/bin/bash
2.
3. # DESCRIPTION: Simple scenario with 2 PCs and one router
4.
5. LIGHTNING_INSTALLATION_PATH=$(dirname $(readlink -f $(which lightning)))
6. source $LIGHTNING_INSTALLATION_PATH/functions
7. source $LIGHTNING_INSTALLATION_PATH/personalized_functions
8.
9. # CONTAINER CREATION & CONFIGURATION
10. create_container hX hostX host
11. create_interface hX eth1
12. clear_interface hX eth1
13.
14. create_container hY hostY host
15. create_interface hY eth1
16. clear_interface hY eth1
17.
18. create_container R1 Router1 router
19. create_interface R1 eth0.0
20. clear_interface R1 eth0.0
21. configure_ip R1 eth0.0 192.168.0.1/24
22. create_interface R1 eth0.1
23. clear_interface R1 eth0.1
24. configure_ip R1 eth0.1 192.168.1.1/24
25. create_interface R1 eth0.2
26. clear_interface R1 eth0.2
27. configure_ip R1 eth0.2 192.168.2.1/24
28. create_interface R1 eth0.3
29. clear_interface R1 eth0.3
30. configure_ip R1 eth0.3 192.168.3.1/24
31. create_interface R1 eth0.4
32. clear_interface R1 eth0.4
33. configure_ip R1 eth0.4 192.168.4.1/24
34. create_interface R1 wlan0
35. clear_interface R1 wlan0
36. configure_ip R1 wlan0 192.168.5.1/24
37.
38.
```

```
39. # NETWORK CREATION & CONNECTIONS
40. create_net Red1
41. connect_machine hX eth1 Red1
42. configure_ip hX eth1 192.100.100.101/24
43. connect_machine R1 eth0.0 Red1
44.
45. create_net Red2
46. connect_machine hY eth1 Red2
47. configure_ip hY eth1 192.100.100.102/24
48. connect_machine R1 eth0.1 Red2
49.
50.
51. # SHOW CONTAINERS CLI
52. display_host hX
53. display_host hY
54. display_router R1
```

EXAMPLE FILE II (XML)

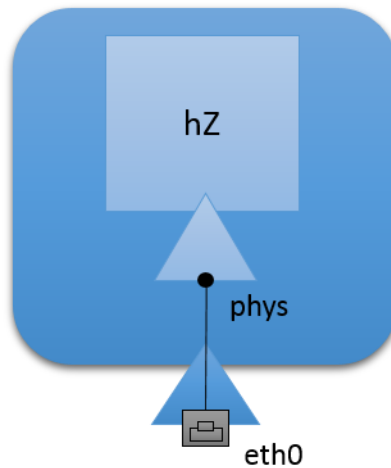
Network scenario with one host directly connected to the physical network segment of the PC running Lightning:



```
1. <?xml version="1.0" encoding="UTF-8" ?>
2.
3. <scenario description="Example: scenario with one PC connected to the physical
4.   host interface eth0"
5.   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6.   xsi:noNamespaceSchemaLocation="lightning-xml-schema.xsd">
7.   <container name="hZ" hostname="hostPhysConn" type="host">
8.     <interface name="phys" enable_ipv6="yes" link_local_ipv6="no" connected_to
9.       _physical_interface="eth0"/>
10.   </container>
11. </scenario>
```

EXAMPLE FILE II (DIRECT EXECUTION)

Network scenario with one host directly connected to the physical network segment of the PC running Lightning:



```
1. #!/bin/bash
2.
3. # DESCRIPTION: Example: scenario with one PC connected to the physical host in
   terface eth0
4.
5. LIGHTNING_INSTALLATION_PATH=$(dirname $(readlink -f $(which lightning)))
6. source $LIGHTNING_INSTALLATION_PATH/functions
7. source $LIGHTNING_INSTALLATION_PATH/personalized_functions
8.
9. # CONTAINER CREATION & CONFIGURATION
10. create_container hZ hostPhysConn host
11. connect_machine_physical hZ phys eth0
12. clear_interface hZ phys
13.
14.
15. # SHOW CONTAINERS CLI
16. display_host hZ
```