# Configuring ODL with Router emulator

There can be number of ways how this configuration can be achieved. The easiest is using a virtual image of linux machine which we found from an ODL tutorial. Other is to use a real Linux server and install the softwares.

## Linux Virtual Image

Download the virtual machine from the below link. The link will have the instructions on how to use the machine. This machine should have java, maven and opendaylight code on it. We just need to build opendaylight on this machine.

We would also need to install docker and the netopeer docker image. Netopeer is the netconf server which we will be using for emulating the router.

<http://sdnhub.org/tutorials/sdn-tutorial-vm/>

Once the virtual image is downloaded and booted using virtual box. We need to install ODL and netopeer.

### Installing Docker and netopeer docker image

$ wget -qO- https://get.docker.com/gpg | sudo apt-key add -

$ wget -qO- https://get.docker.com/ | sudo sh

$ sudo docker pull sdnhub/netopeer

### Installing ODL

$ cd ~/SDNHub\_Opendaylight\_Tutorial

$ mvn clean install -nsu

## Linux machine running Netopeer and ODL running on Linux/Windows

### Installing Docker and netopeer docker image

Installing netopeer docker image will be the same as above for a virtual linux image.

### Installing ODL

For installing the ODL you will need to first clone the ODL code from github. Also make sure java and maven are install on the machine. If not, please install java and maven on the machine.

https://git.opendaylight.org/gerrit/controller

Once the code is cloned it needs to be built using mven. Same as above.

mvn clean install –nsu.

## Steps for creating and registering the router.

### Virtual linux image

If using the virtual image the SDNHub\_Opendaylight\_Tutorial/netconf-exercise directory should have the scripts required for build the router.

#### Create router

Change the existing router.yang with the yang model below. This is a dummy yang model created for this POC.



Run $ ./spawn\_router.sh router1

#### Registering router with ODL controller

Start ODL and install the Netconf feature.

$ cd distribution/opendaylight-karaf/target/assembly

$ ./bin/karaf

opendaylight-user@root> feature:install odl-netconf-connector-all

Run the register script

./register\_netconf\_device.sh router1 <ip address of generated router>

### Linux Machine

Download the spawn\_router and register\_netconf\_device scripts from <https://github.com/sdnhub/SDNHub_Opendaylight_Tutorial/tree/master/netconf-exercise>

#### Creating router

Same as above.

#### Registering router with ODL

Wherever the code is checkout. Find the ./bin/karaf in the target and run it.

$ ./bin/karaf

opendaylight-user@root> feature:install odl-netconf-connector-all

## Invoking the REST URL for sending the configuration to the router.

Once the router is registered with the ODL controller. The YANG model will be exposed by the ODL controller via RESTCONF.

We can use the below REST API to send information to the router via ODL.

<http://localhost:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/router1/yang-ext:mount/router:router>

PUT json/application

Sample JSON

{

"router":{

"ipsla":[{

"cos-entry-number":"40102",

"destination-address":"12.12.12.12",

"source-address":"10.10.1.1",

"customer-name":"Hello Inc"

},

{

"cos-entry-number":"40103",

"destination-address":"10.11.12.12",

"source-address":"10.10.1.1",

"customer-name":"ABC Inc"

},

{

"cos-entry-number":"40105",

"destination-address":"12.5.12.12",

"source-address":"10.10.1.2",

"customer-name":"test Inc"

}]

}

}

GET can be invoked on the same URL to fetch the current configuration.