## Setting up Consul

After getting consul installed you need to create a directory to store Consul configuration and service definition files. A standard location to create the configuration directory is /etc/consul.d. Remember the location you use because you will need it when starting the Consul Agent.

The only configuration for Consul saved to a file is to enable Consul Connect on the node serving as the consul server. At the time this document was created the Consul server was Debian-10-Consul-LXC-4 on Server 3. The configuration file, basic\_config.json, to enable Connect looks like:

{

"connect": {

"enabled": **true**

}

}

The rest of the Consul configuration has been done though command line arguments. The rest of this document will show the command line arguments used to start Consul. Most of the command line arguments can be moved to a configuration file on each host. Information for creating Consul configuration files can be found here: <https://www.consul.io/docs/agent/options>.

## Setting up Consul Services

First, we will set up the service that we want to direct traffic towards. In this case we want to direct traffic from OVS (Mininet) to ONOS. When this document was written ONOS was in the container ONOS-copy on Server 1. Here is what the configuration, onos.json, for ONOS looks like:

{

"service": {

"name": "onos",

"port": 6653,

"connect": {

"sidecar\_service": {}

}

}

}

Notice we are directing traffic to the OVS port for ONOS at 6653. The sidecar\_service is telling Consul Connect that we will need a proxy setup for this service. Consul will then create the proxy service automatically and we will need to start the proxy manually as shown later.

Next, we need to set up a service definition for Mininet so that it can contact ONOS through the proxy. Mininet is in a separate container from ONOS. The Mininet container at the time this document was created was Debian-10-Consul-LXC-3 on Server 3. Here is the configuration, mininet.json, for Mininet:

{

"service": {

"name": "mininet",

"connect": {

"sidecar\_service": {

"proxy": {

"upstreams": [

{

"destination\_name": "onos",

"local\_bind\_port": 12000

}

]

}

}

}

}

}

In Mininet’s service definition we are defining a proxy listening on port 12000 that will direct traffic to the ONOS service. When we start Mininet we can set the controller IP to localhost and port to 12000 and its traffic will be directed to ONOS through the proxy.

## Starting Consul

Now that all of the Consul servers and clients are configured, we can start Consul. First, start the Consul Agent that will be the server. Currently the Consul Server is run on Server3 in the container Debian-10-Consul-LXC-3. The Consul Server Agent is started using the following command:

root@Debian-10-Consul-LXC-3:~# consul agent -server -node=agent-server -bind=192.168.16.239 -client=0.0.0.0 -dns-port=53 -bootstrap -data-dir=/tmp/consul -config-dir=/etc/consul.d &

Then, we can start any Consul client next. Lets start the ONOS Consul Agent. Currently ONOS is being run on Server 1 in container ONOS-copy. Before starting Consul ensure that ONOS has started and is running. We can check if ONOS is running by using the following command:

root@ONOS-copy:~# systemctl status onos

We should see ONOS is loaded and active as in Figure 1. We can then ssh into the ONOS CLI and run a few commands to ensure ONOS has fully loaded using the following command:

doughertypat@Debian-10-ONOS-VM-1:~$ ssh karaf@localhost -p 8101

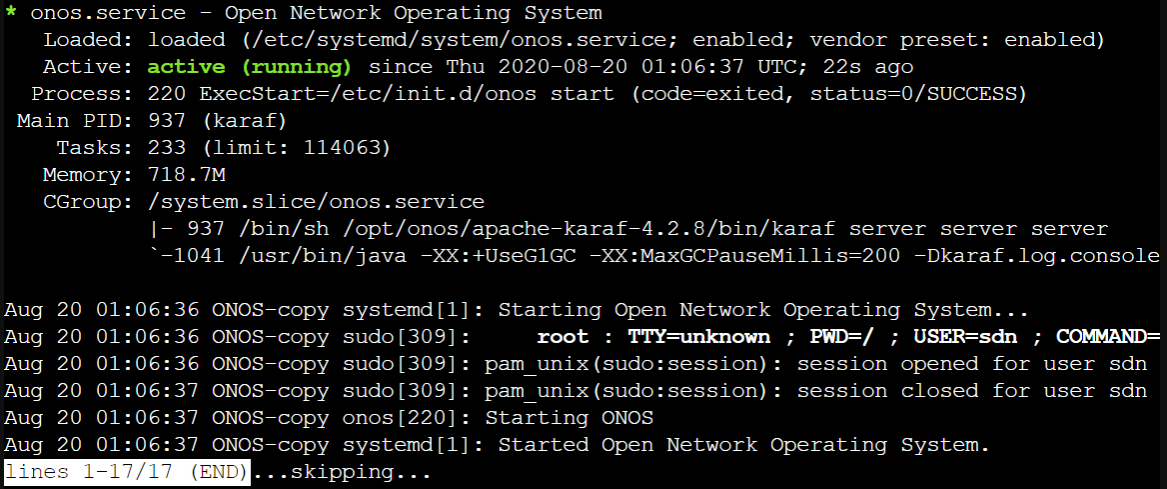


Figure . ONOS Service Status

Once an ssh connection is established login to ONOS using the default password, karaf, or your user defined password. Then a few ONOS commands can be run to ensure ONOS is running properly as shown in Figure 2.

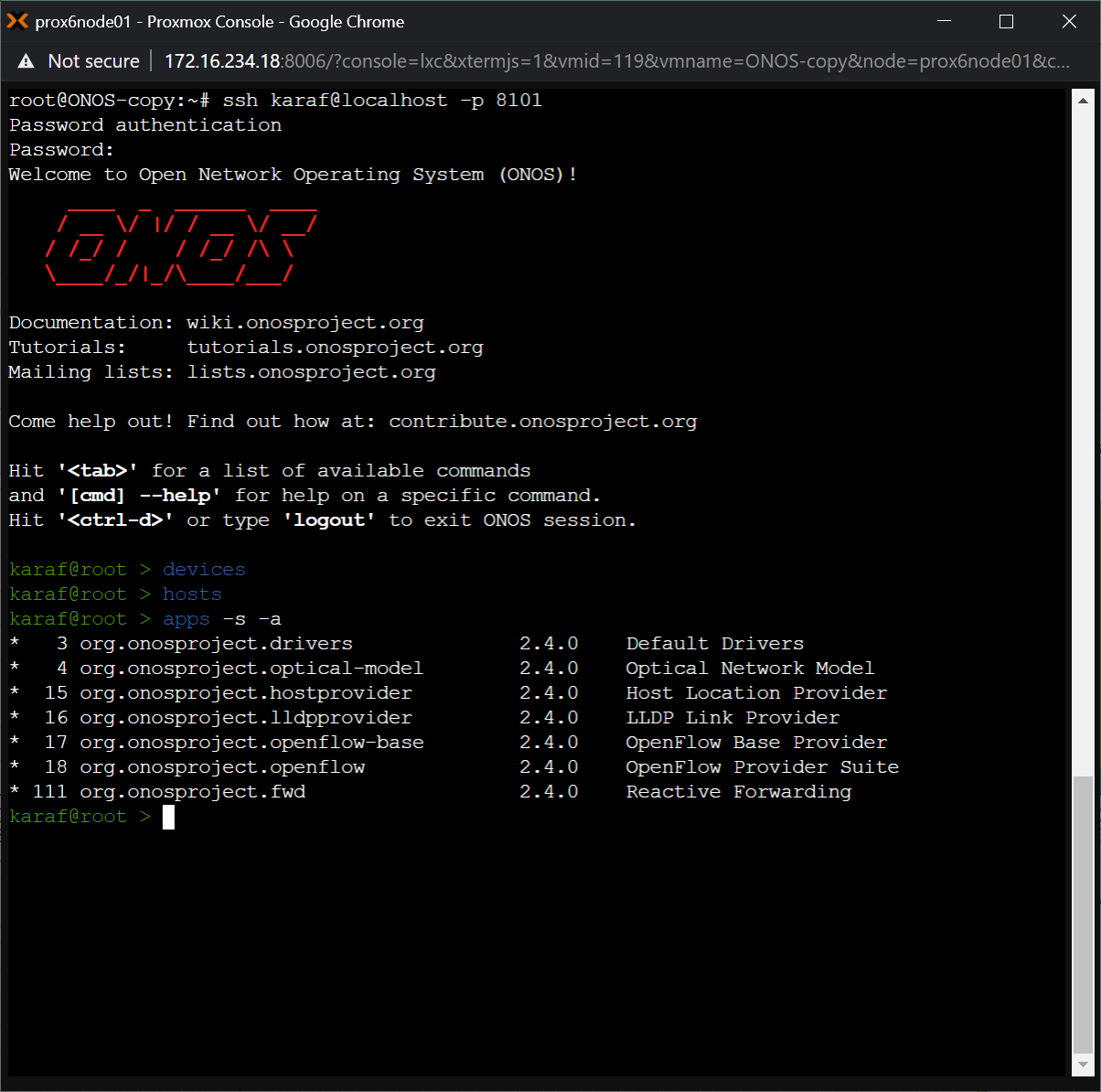


Figure . ONOS CLI

If ONOS has not started correctly you may see errors such as those shown in Figure 3. In this case you should look at the log file in your onos directory (ONOS\_HOME). It is typically: /opt/onos/log/karaf.log. Some troubleshooting steps are covered in a separate section at the end of this document.

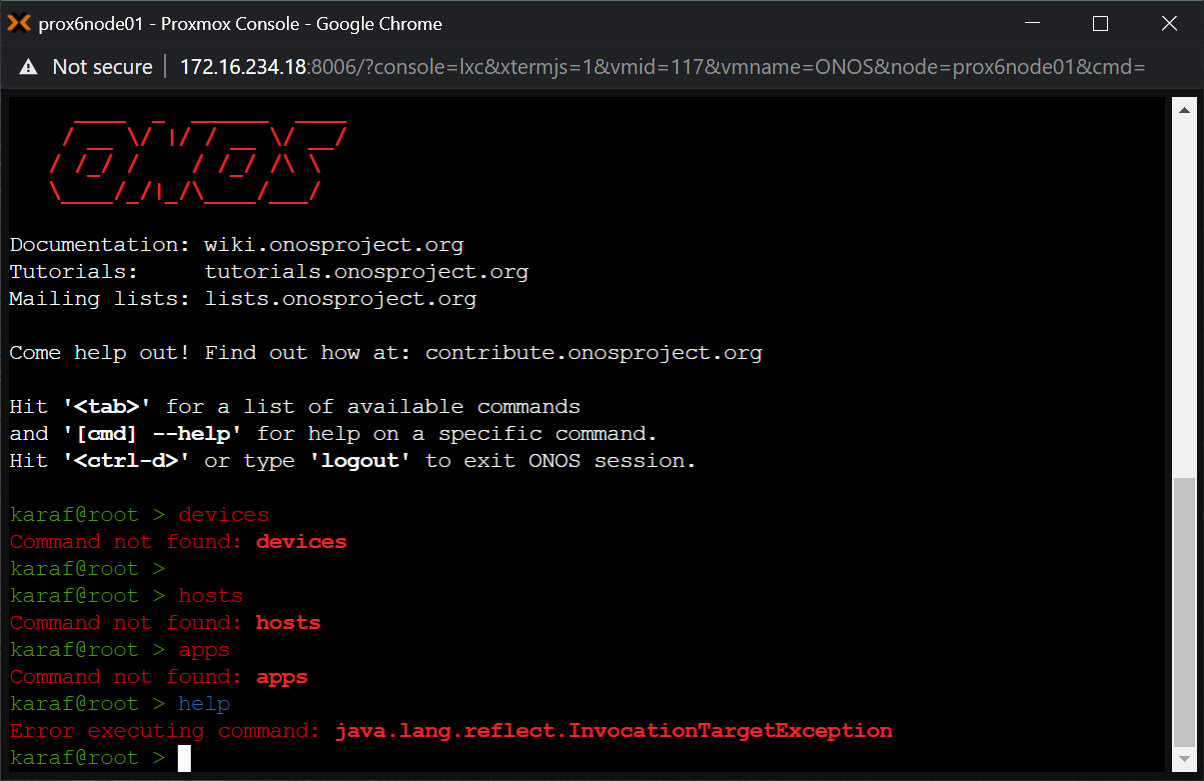


Figure . ONOS CLI Command Errors

Once it is confirmed that ONOS is started and running properly we can start Consul. The following commands will start Consul and the Connect Proxy:

root@ONOS-copy:/etc/consul.d$ sudo consul agent -bind=192.168.16.209 -advertise=192.168.16.209 -node=agent-onos -dns-port=53 -config-dir=/etc/consul.d -data-dir=/tmp/consul -enable-local-script-checks -join=192.168.16.239 &

root@ONOS-copy:/etc/consul.d$ sudo consul connect proxy --sidecar-for onos

As mentioned in the Setting up Consul Services section, we must manually start the Connect Proxy and the second command is doing just that.

Finally, we will start the Consul client on the Mininet LXC. The following commands start Consul and the Connect Proxy:

root@Debian-10-Consul-LXC-4:/etc/consul.d# consul agent -bind=192.168.16.238 -advertise=192.168.16.238 -node=agent-one -dns-port=53 -config-dir=/etc/consul.d -data-dir=/tmp/consul -enable-local-script-checks -join=192.168.16.239 &

root@Debian-10-Consul-LXC-4:/etc/consul.d# consul connect proxy --sidecar-for mininet

## Using the Services

Now that we have Consul started we can use the services we just defined. ONOS should already be running so we need to start Mininet with the controller directed to port 12000. We can do that using the following command:

root@Debian-10-Consul-LXC-4:/etc/consul.d# mn --controller=remote,ip=127.0.0.1,port=12000

Once Mininet has started open another terminal on the same container (or use screen) and verify that OVS is connected to ONOS. To do this use the following command:

root@Debian-10-Consul-LXC-4:/etc/consul.d# ovs-vsctl show

You should get a response that looks like this:

root@Debian—10—Consu1—LXC 4 : /etc/consul .d# ovs—vsctl 
1bdebfc6 0053-49b8 98eO 68c7491e2fbO 
Bridge "sl " 
Controller "tcp:127.0.0.1:12000" 
is connected: true 
Controller "ptcp: 6654 " 
fail mode: secure 
Port "sl " 
Interface "sl " 
type: internal 
Port "sl —eth2 " 
Interface "s1—eth2 " 
Port "sl—ethl " 
Interface "sl—ethl " 
ovs version: "2.10.1" 
root@Debian—10—Consu1—LXC 4 : /etc/consul . 
show 

Figure . OVS Switch Configuration

If “is\_connected: true” is not shown then there is a problem with the connection between OVS and ONOS. Assuming you have a successful connection, next you should verify that ONOS sees the OVS switch. To do this we will ssh into the ONOS CLI as we did earlier. Once logged into the CLI run the following command:

karaf@root > devices

You should see a response similar to this:

karaf@root > devices 
id=of: 0000000000000001, available=true, local—status=connected 9s ago, role=MASTER, type—SWITCH, mfr=Nicira, 
eria1=None, chassis=l, driver=ovs, channelId—127.0.0.1:34896, managementAddress=127.0.0.1, protocol=OF 14 
Inc., 
hw=Open vSwitch, 
sw 2.10.1, 
s 

Figure . ONOS Devices Output

Now that we have confirmed ONOS and OVS are connected and both recognize eachother, we can attempt to use ONOS to control OVS. In the Mininet terminal enter:

mininet > pingall

If you have reactive forwarding disabled, you should see the pings fail as shown in Figure 6.

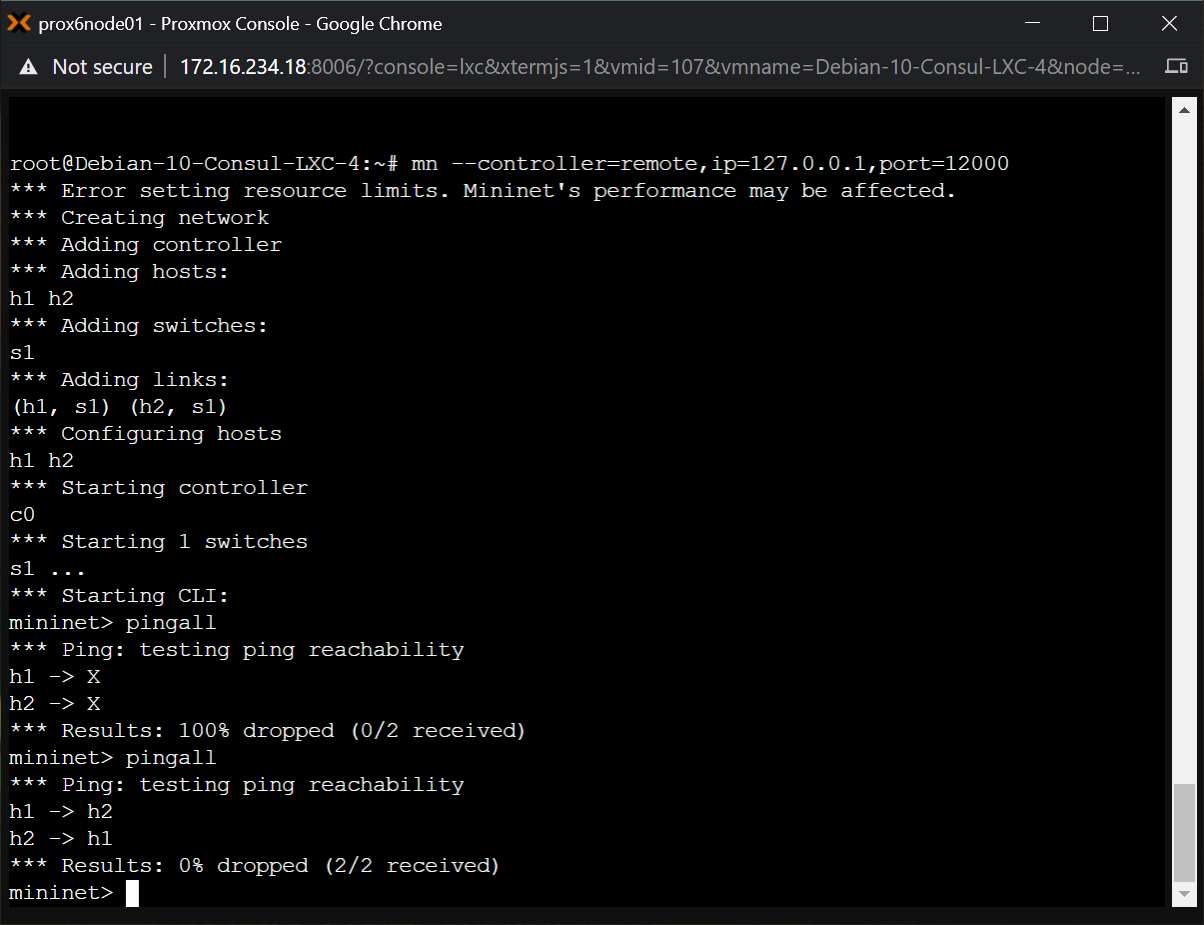


Figure . Mininet Ping Failure

In the ONOS CLI issue the following command to activate reactive forwarding:

karaf@root > app activate fwd

Now in the Mininet terminal ping all the hosts again and you should see a successful result as shown in Figure 7.

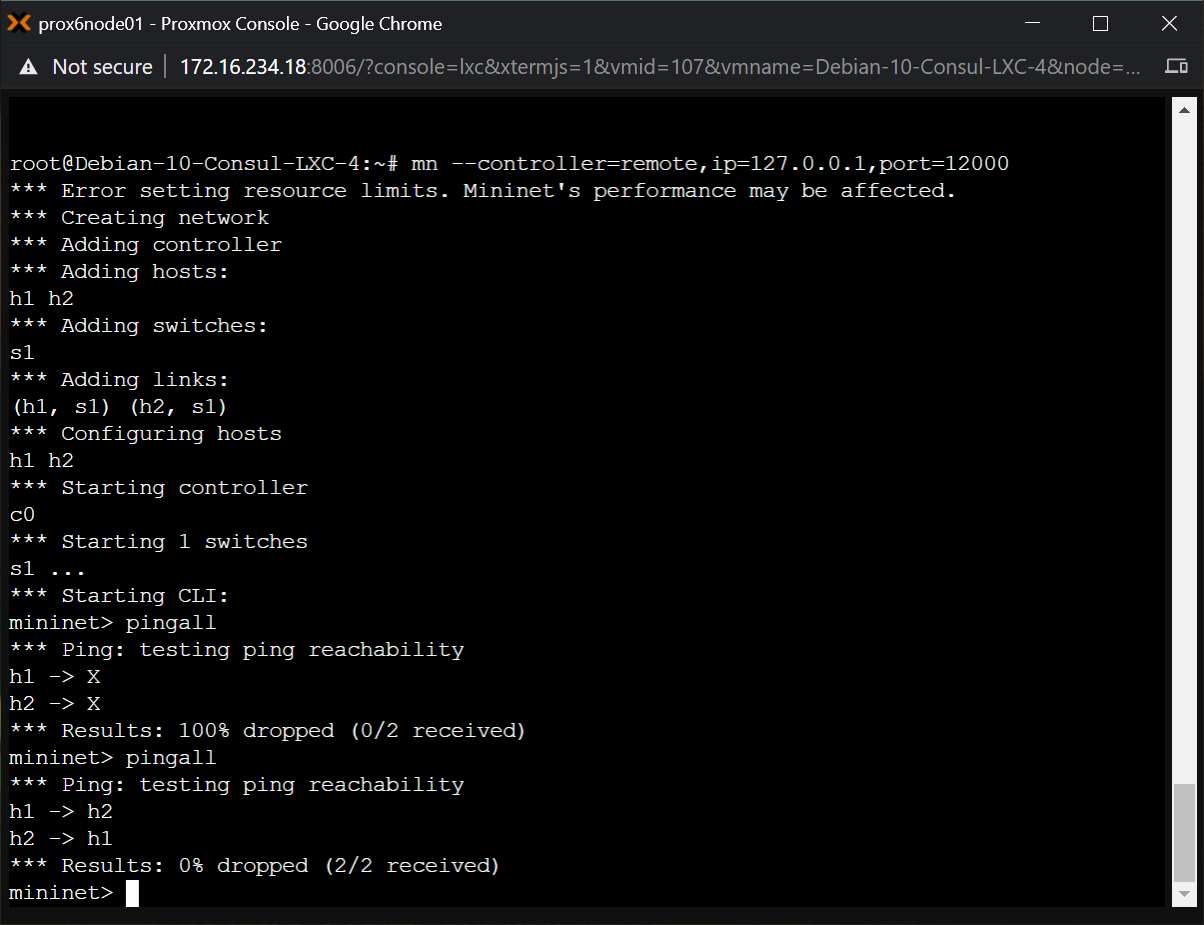


Figure 7. Mininet Ping Success

Congratulations you have successfully configured Consul to allow a connection between Mininet and ONOS!

## Troubleshooting

### Java Version

When starting ONOS as a service it seemed to load correctly. It showed loaded and active when systemctl status was run. But when commands were run in the CLI, error resulted as shown in Figure 3. In the logs there were ‘missing requirement’ errors as shown in Figure 8.

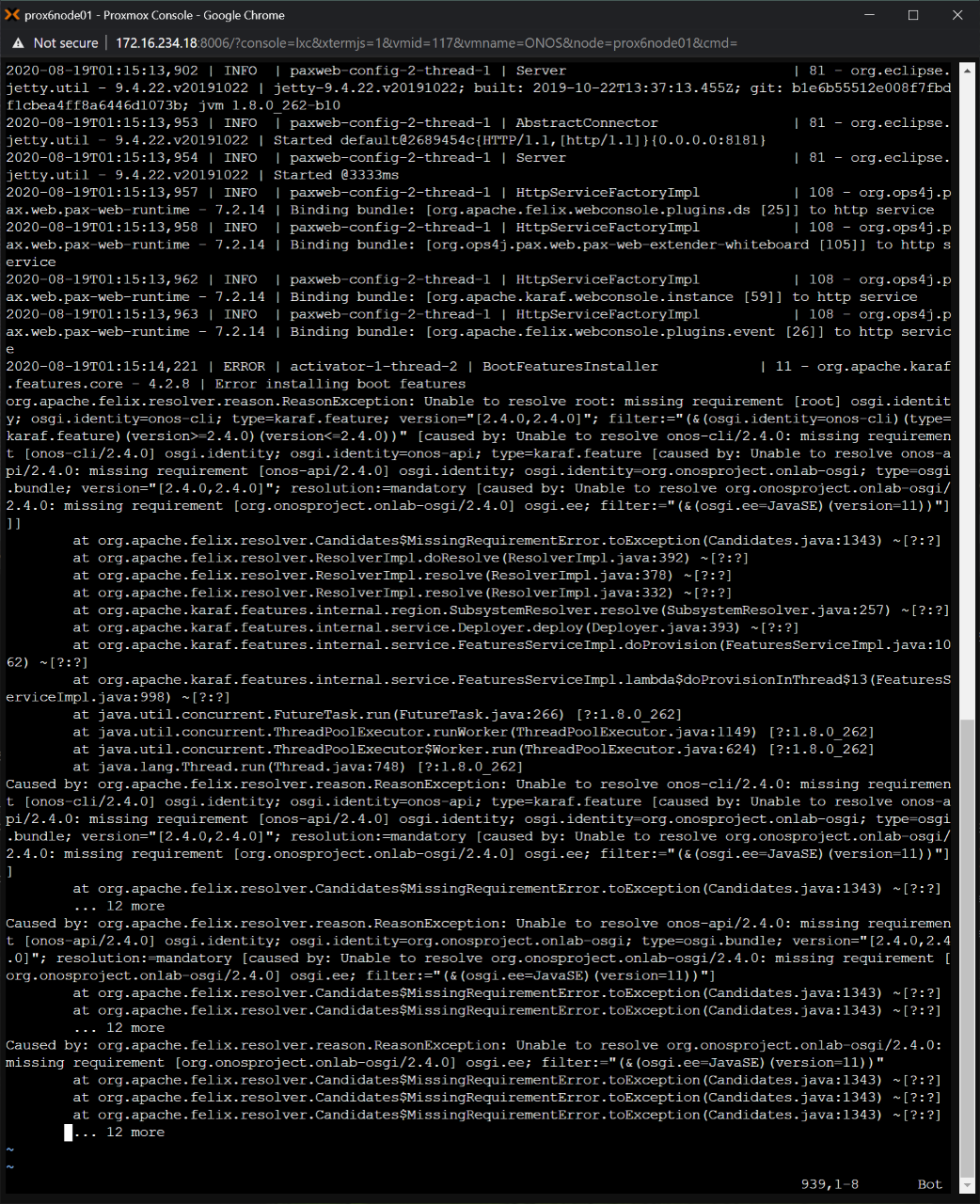


Figure . ONOS Missing Requirement Errors

It was noted that there were repeated references in the errors to osgi.ee=JavaSE version=11. The version currently installed was Java 1.8. ONOS claims that their product is compatible with Java 1.8 but it was decided to try installing Java 11 anyways. The package openjdk-11-jdk was installed. It was verified that update-alternatives for java and javac had the new version 11 selected. JAVA\_HOME was updated to the path shown in update-alternatives for Java 11’s home folder. The PATH was also updated to include JAVA\_HOME/bin. After all this ONOS was restarted using systemctl restart onos. ONOS started properly and the commands worked properly in the CLI. ONOS was also able to connect to Mininet successfully.