RAMNIRANJAN JHUNJHUNWALA COLLEGE
GHATKOPAR (W), MUMBAI - 400 086

DEPARTMENT OF INFORMATION TECHNOLOGY

2020 - 2021

M.Sc.(I.T.) SEM IV

Software Defined Networking

Name: Jitendra Sharma

Roll No.: 12

Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE

(AUTONOMOUS)

Opposite Ghatkopar Railway Station, Ghatkopar West, Mumbai-400086

CERTIFICATE

This is to certify that **Mr. Jitendra Sharma** with Seat No. **12** has successfully completed the necessary course of experiments in the subject of **Software Defined Networking** during the academic year **2020 – 2021** complying with the requirements of **RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE AND COMMERCE**, for the course of M.Sc**. (IT)** semester -IV.

Internal Examiner Date:

Head of Department College Seal External Examiner

Index

PRACTICAL NO: 01

Aim: Wireshark and bash Script

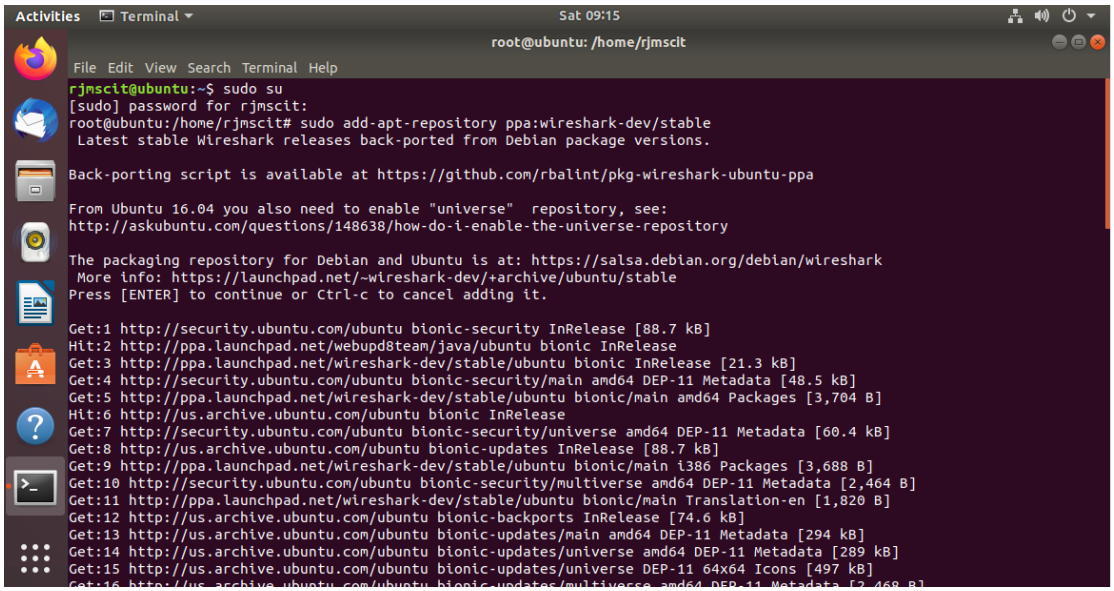
Wireshark is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.

Wireshark has a rich feature set which includes the following:

- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Standard three-pane packet browser
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- The most powerful display filters in the industry
- Rich VoIP analysis
- Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® (compressed and uncompressed), Sniffer® Pro, and NetXray®, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Coloring rules can be applied to the packet list for quick, intuitive analysis
- Output can be exported to XML, PostScript®, CSV, or plain text

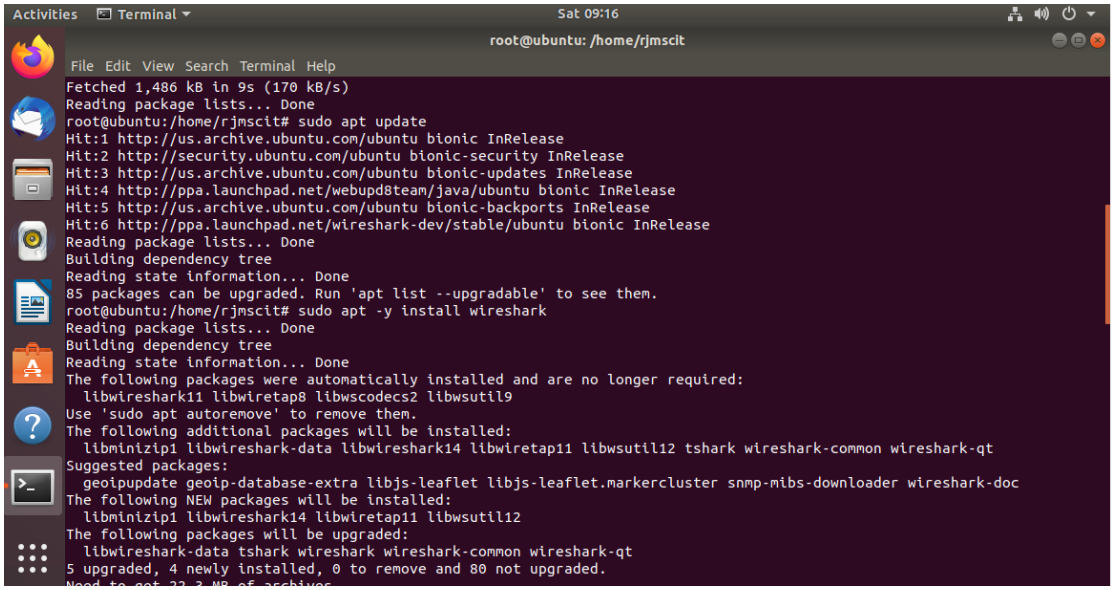
Step 1 -> Install Wireshark

>> sudo add-apt-repository ppa:wireshark-dev/stable



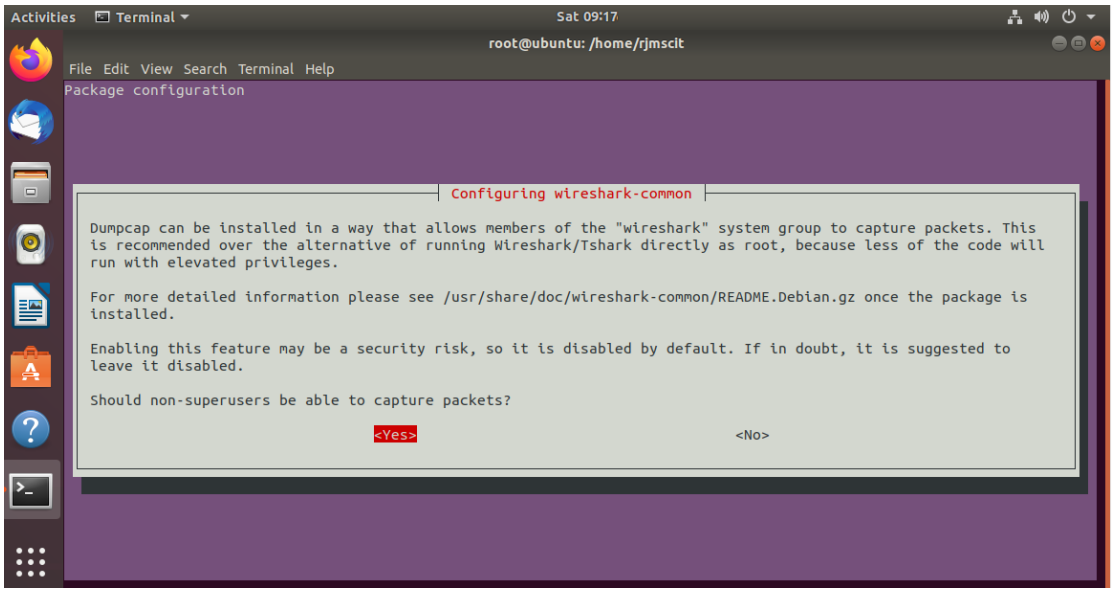
>> sudo apt update

>> sudo apt -y install wireshark



>> sudo dpkg-reconfigure wireshark-common

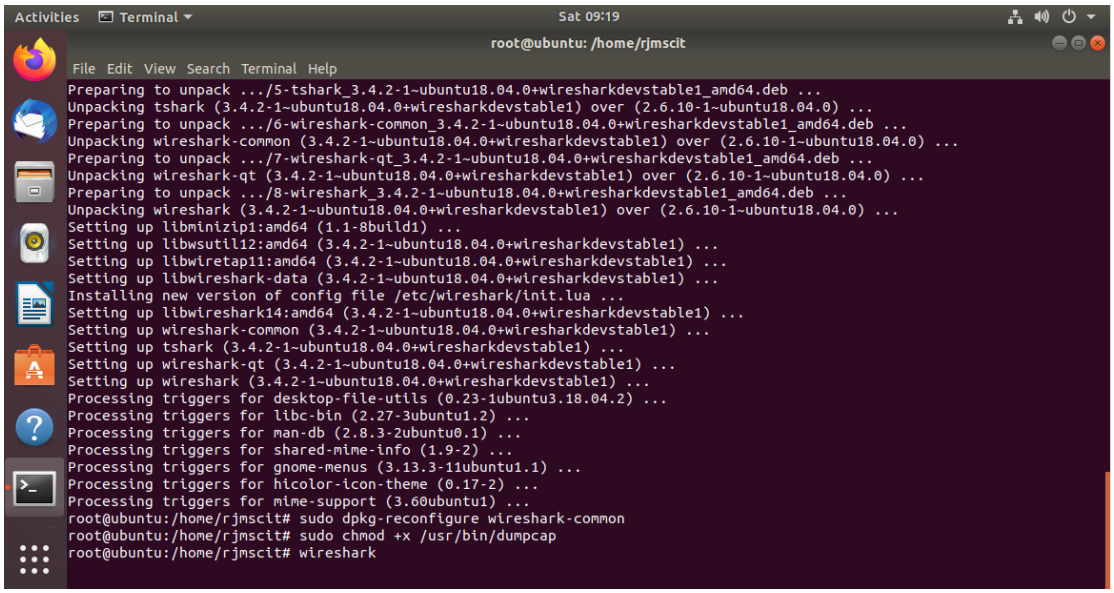
then press yes

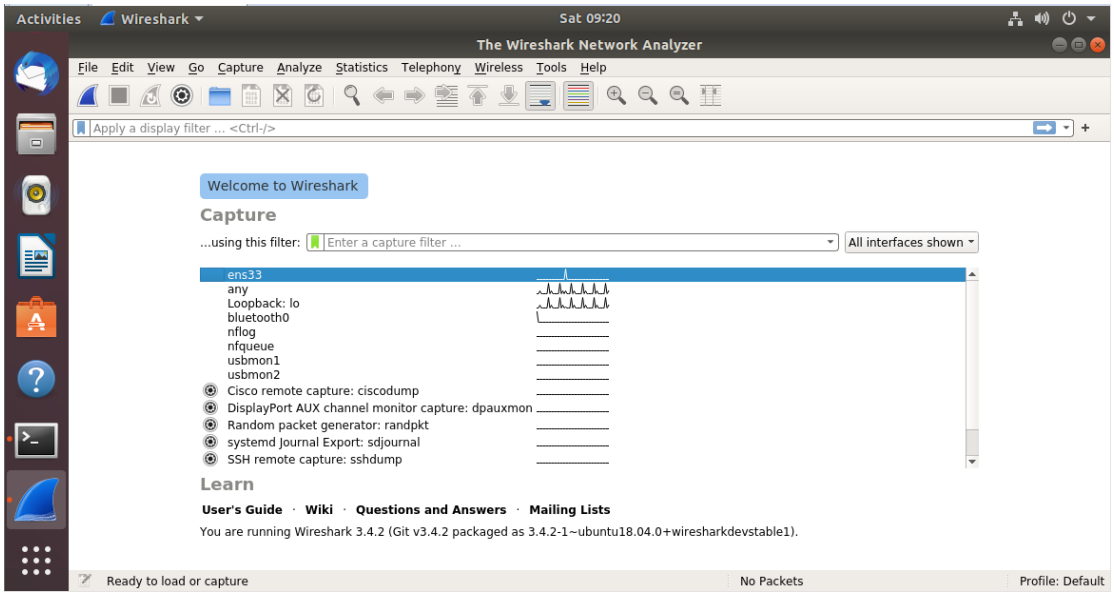


>> sudo chmod +x /usr/bin/dumpcap

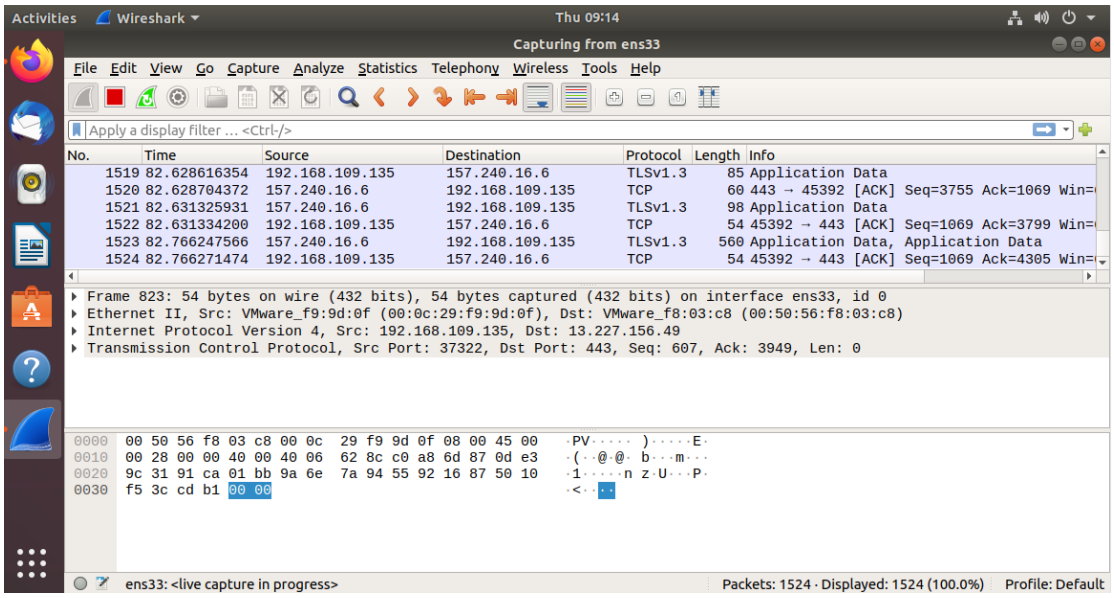
Step 2 -> Starting Wireshark

Launching a Wireshark application using CLI
Type wireshark in CLI and press enter

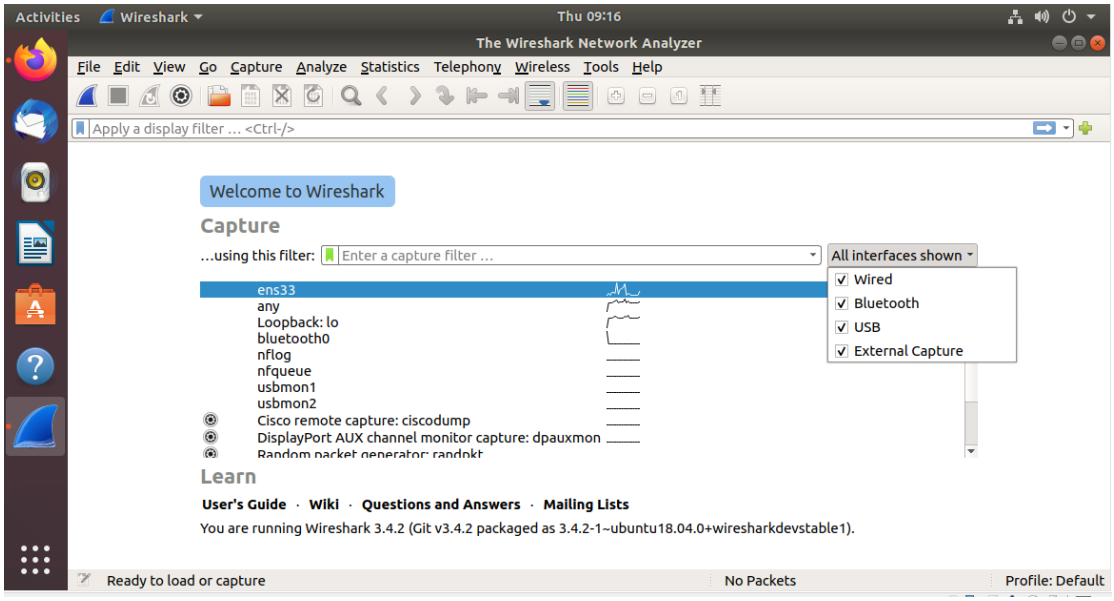




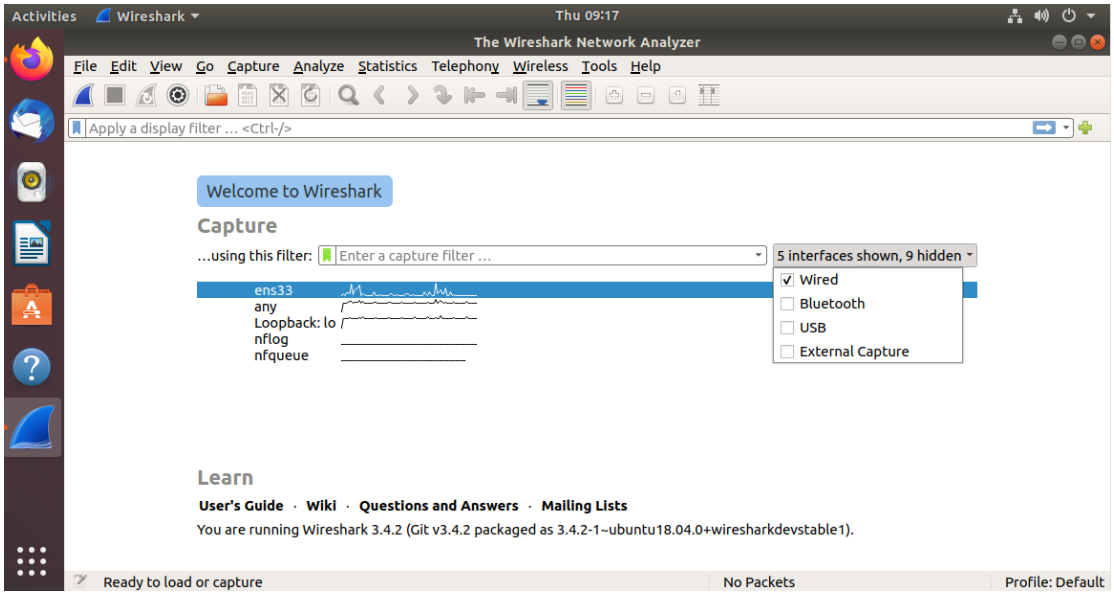
To test packets capturing, select interface to use and click "**Start capturing packets**" button.



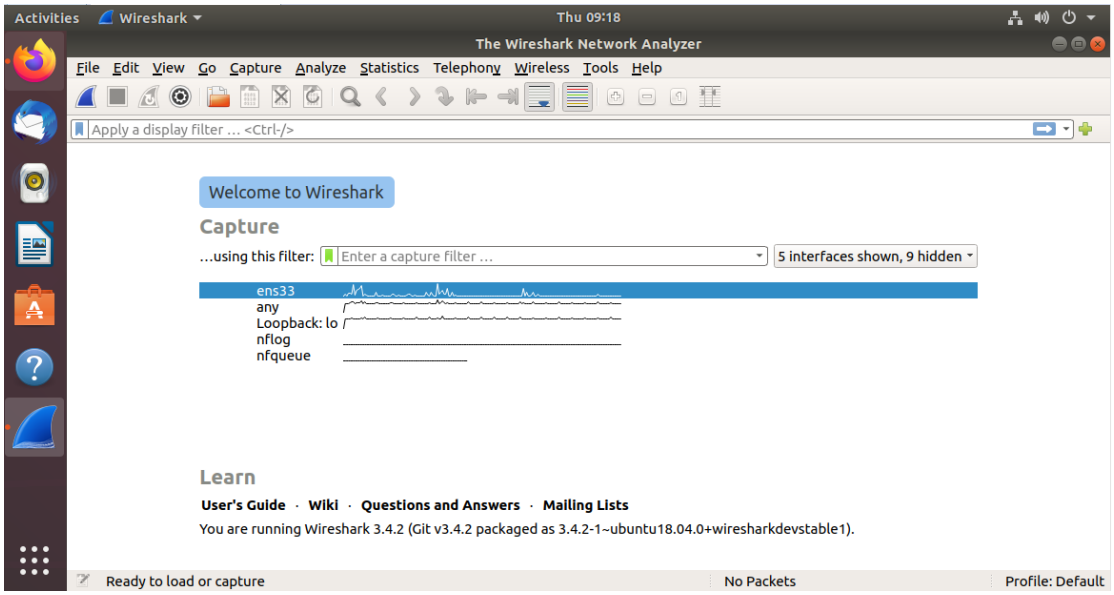
There are many types of interfaces you can monitor using Wireshark, for example**, Wired, Wireless, USB** and many external devices. You can choose to show specific types of interfaces in the welcome screen from the marked section of the screenshot below.



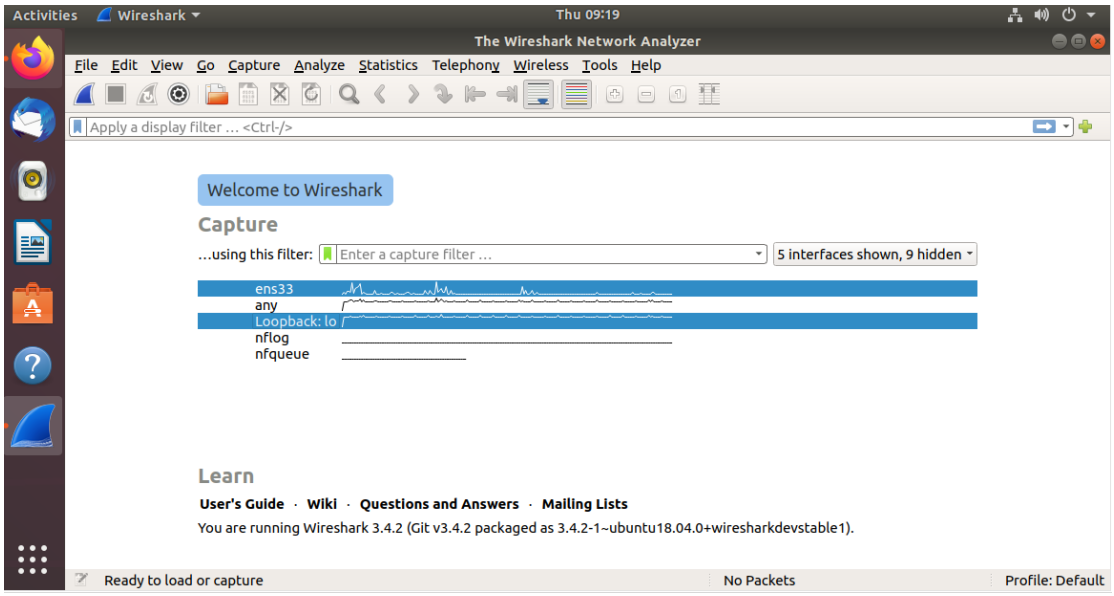
Here, I listed only the **Wired** network interfaces.



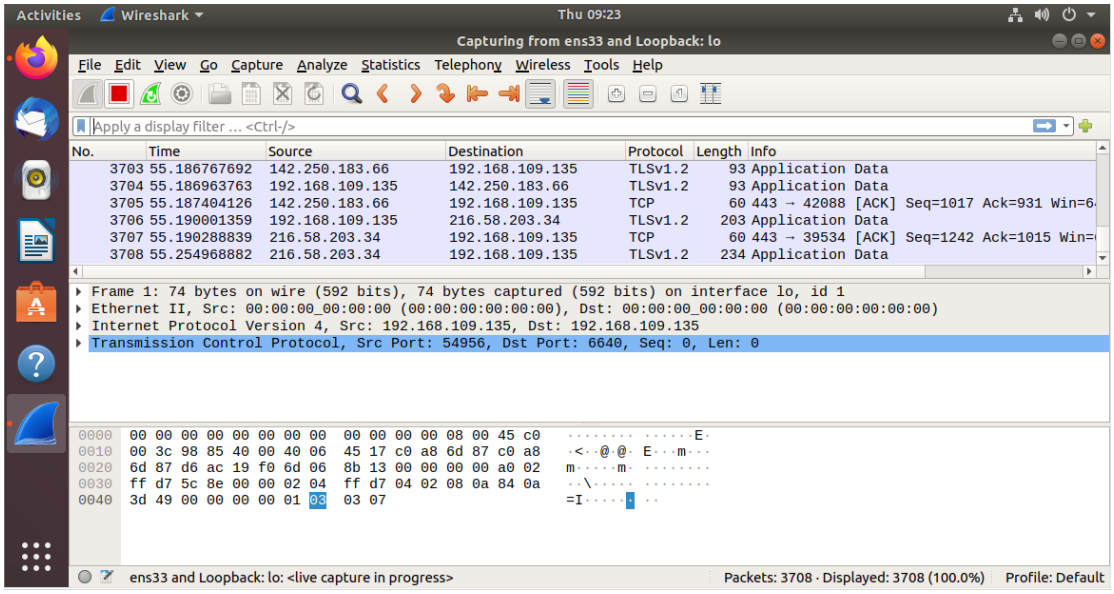
Now to start capturing packets, just select the interface (in my case interface **ens33**) and click on the **Start capturing packets icon** as marked in the screenshot below. You can also double click on the interface that you want to capture packets to and from to start capturing packets on that particular interface.



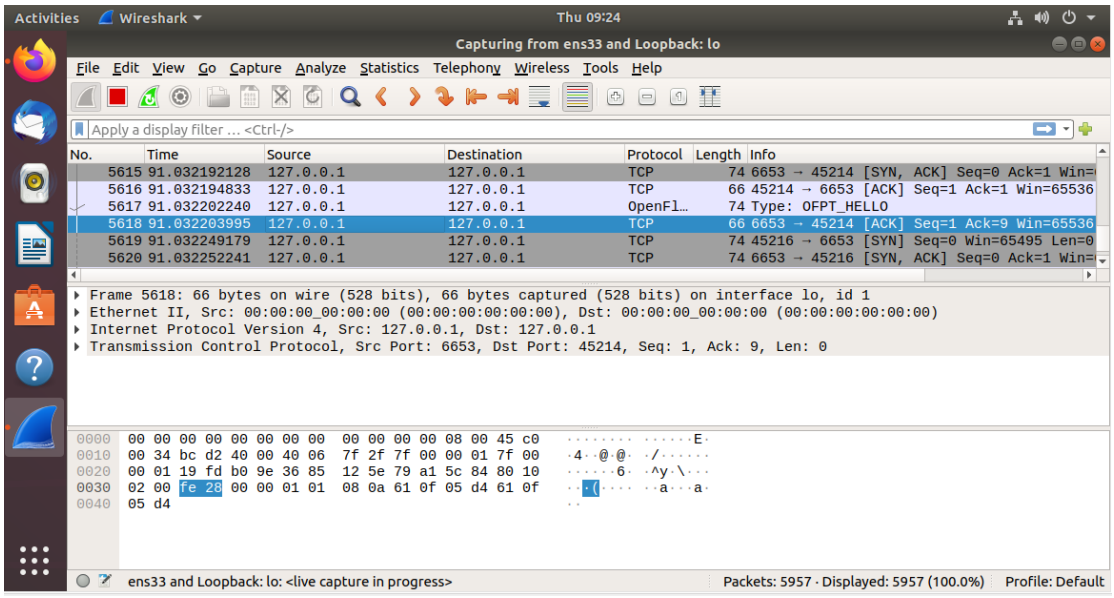
You can also capture packets to and from multiple interfaces at the same time. Just press and hold **<Ctrl>** and click on the interfaces that you want to capture packets to and from and then click on the **Start capturing packets** icon as marked in the screenshot below.



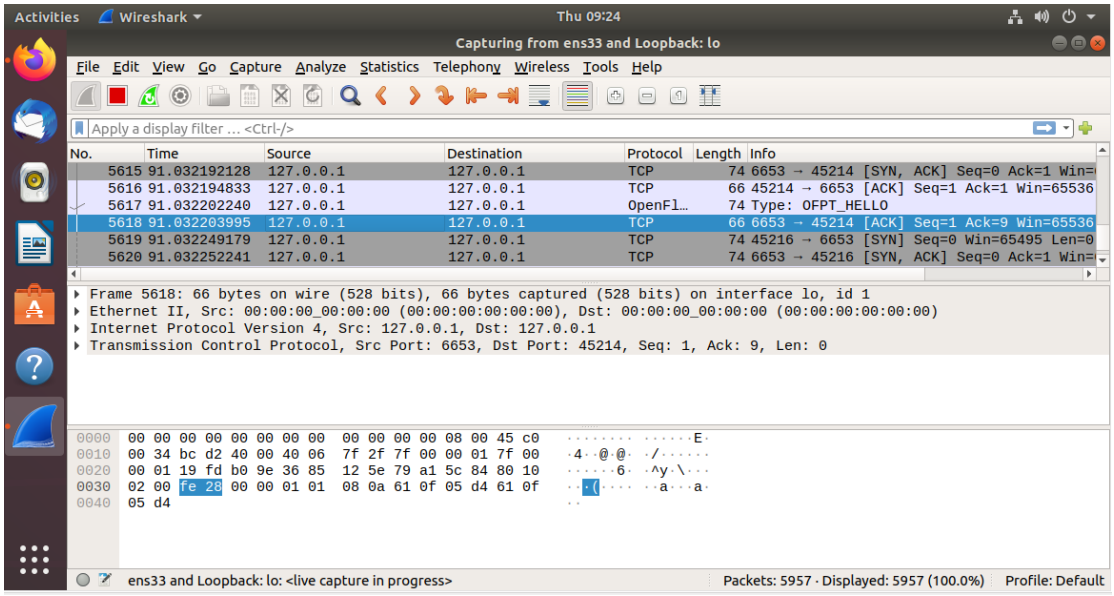
I pinged google.com from the terminal and as you can see, many packets were captured.



Now you can click on a packet to select it. Selecting a packet would show many information about that packet. As you can see, information about different layers of TCP/IP Protocol is listed.

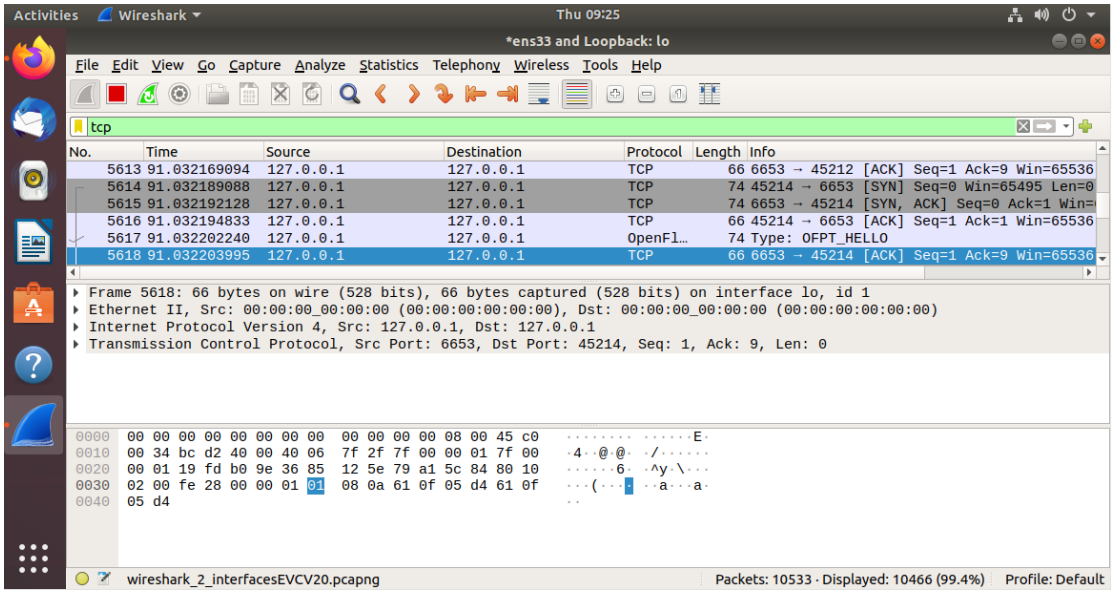


You can also see the RAW data of that particular packet.



Filtering Packets Using Wireshark:

On a busy network thousands or millions of packets will be captured each second. So the list will be so long that it will be nearly impossible to scroll through the list and search for certain type of packet. The good thing is, in Wireshark, you can filter the packets and see only the packets that you need. To filter packets, you can directly type in the filter expression in the textbox as marked in the screenshot below.

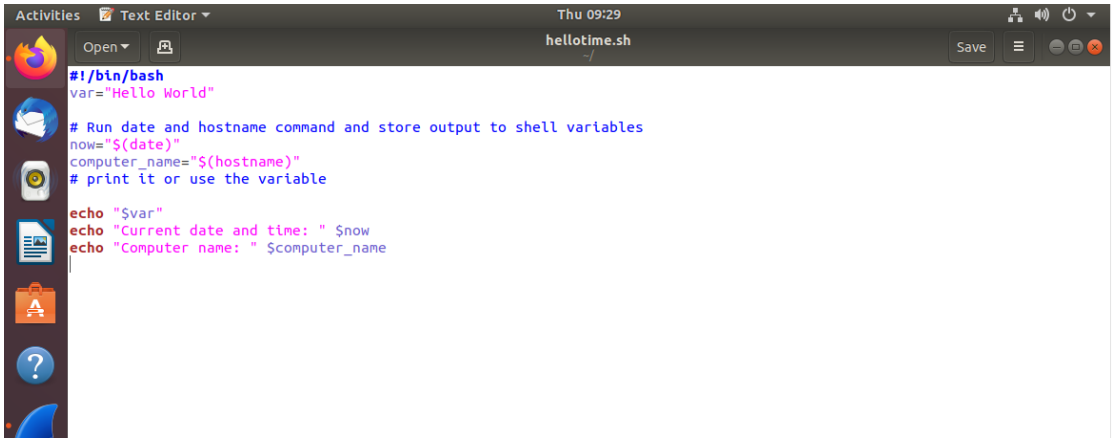


Bash Scripts

open text editor and save file with .sh; run in terminal

Bash script to print current date and time and Hostname.

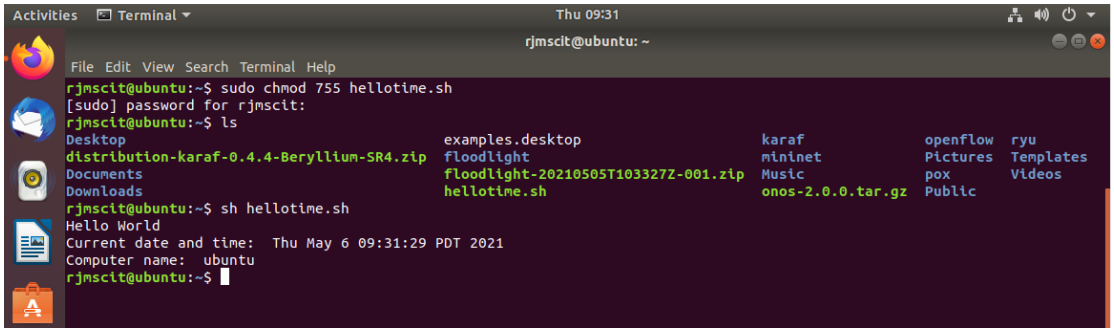
```
#!/bin/bash
var="Hello World"

# Run date and hostname command and store output to shell variables
now="$(date)"

computer_name="
$(hostname)"

# print it or use the variable

echo "$var"
echo "Current date and time: " $now
echo "Computer name: " $computer_name
```



After creating the script give execute permission to the file/script.

>> chmod 755 hellotime.sh

>> sh hellotime.sh



PRACTICAL NO: 02

Theory: HP controller, HP Network Protector, HP Network Visualizer, HP Network Optimizer, Cisco XNC Controller

Hewlett-Packard

HP is the first to offer SDN technologies for all three critical layers—infrastructure, control software and application—to simplify networks and improve agility across the enterprise. These layers create a complete, open SDN hardware and software solution that provides a single point of control for the entire network.

The **infrastructure layer** delivers open programmable access through OpenFlow, a networking protocol that automates hardware configurations.

- New SDN functionality in the infrastructure layer enables clients to simplify network configuration. HP today announced nine additional switch models providing OpenFlow-enabled support for HP Flex Network architecture, offering clients a flexible and programmable standards-based interface. The addition of the new HP 3800 switch series to the 16 existing models in the product portfolio reaffirms HP as the only major networking vendor with more than 15 million installed OpenFlow-enabled ports. (1)

  The control-software layer creates a centralized view of the network.

- Within the control layer, the new HP Virtual Application Networks SDN Controller abstracts the physical hardware from the logical deployment, providing a centralized view and automating network configuration of all devices in the infrastructure. By eliminating thousands of manual CLI entries, the controller enables network administrators to easily and flexibly program and scale their network environment for single-touch automated applications. It also provides application program interfaces (APIs) to third-party developers to integrate custom enterprise applications.

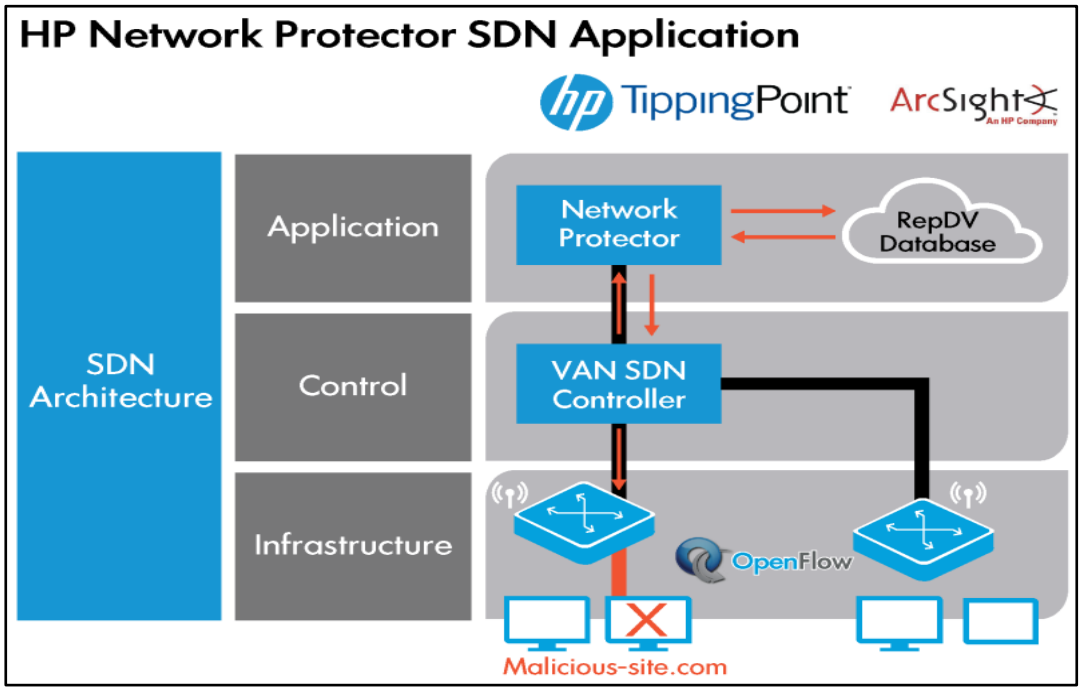
  “We recognized the need to scale our perimeter firewall capacity to cope with the increase of internet traffic,” said Jean-Michel Juanito, communication systems group leader, IT Department, CERN. “In the framework of the CERN open lab R&D collaboration, we are developing with HP’s software defined network technologies a load-balancing application for the Virtual Application Networks SDN controller to distribute network traffic across multiple devices including firewalls and servers, increasing simplicity while reducing cost and bandwidth bottlenecks.”

The **application layer** delivers open programmable interfaces to automate applications across the network.

- New HP Virtual Cloud Networks software enables cloud providers to deliver automated and scalable public cloud services to enterprises. Using this software, enterprises can create an isolated virtual cloud network environment through a self-

service public cloud infrastructure, providing them complete control for introducing new services and applications to their users.

Also at the application layer, the new HP Sentinel Security software application automates network access control and intrusion prevention security for enterprise campus networks with existing OpenFlow-enabled switch hardware through HP's controller. As a result, clients can eliminate the complexity and expense of dedicated networking hardware appliances, while achieving scalable security needed for bring-your-own-device (BYOD) initiatives.



HP Network Protector

The HP Network Protector SDN Application leverages HP Networking, Tipping Point, and ArcSight products to deliver a converged solution that addresses security threats in a completely new way by leveraging the network itself. The HP Network Protector enables network intelligence on network infrastructure devices. The application uses the HP VAN SDN (Virtual Application Network Software-Defined Networking) Controller and OpenFlow to program the network infrastructure with security intelligence from the HP Tipping Point Reputation Digital Vaccine (RepDV). This turns the entire network infrastructure into security-enforcement devices, providing visibility and threat protection against more than one million malicious botnets, malware, and spyware sites.

Some of the key features of the solution include:

- Runs on HP VAN SDN Controller
- Complimentary to Tipping Point IPS solutions
- Delivers real-time threat characterization with HP Tipping Point DVLabs Database

- Protects from over 1,000,000+ botnet, malware, spyware, and malicious sites
- OpenFlow enabled switches gain ability to detect malware, botnets, and other threats
- Ability to create custom whitelist and blacklist
- Improves visibility and accuracy with ArcSight Integration
- Dynamic switch learning with HPN OpenFlow enabled switches distributes detection into switch infrastructure

HP Network Visualizer

The HP Network Visualizer SDN Application by utilizing HP VAN SDN Controller provides dynamic traffic capture with real-time detailed network monitoring allowing for fast network diagnosis and verification, rapid transition from incident to fix.

Features

1. Real-time Visibility and Diagnosis

Network Visualizer provides dynamic traffic capture to diagnose the root cause of the network. It proactively monitors the network to reduce the number of help desk issues.

2. Low Cost, Simple and Automated Troubleshooting

Network Visualizer allows for simple troubleshooting that requires high level network detail. The application eliminates the need for any expensive manual network tapping tools for troubleshooting.

3. Fast Transition from Incident to Fix

Network Visualizer solves network issues in a matter of seconds versus minutes.

Network Optimizer

HP Network Optimizer SDN Application for Microsoft Lync enables automated provisioning of network policy and quality of service to provide an enhanced user experience. The Network Optimizer Application dynamically provisions the end-to-end network path and Quality of Service (QoS) policy via the HP Virtual Application Networks (VAN)SDN Controller, reducing the need for manual, device-by-device configuration via the CLI, which greatly simplifies policy deployment and reduces the likelihood of human errors.

Network Optimizer provides:

- Traffic classification – Identify application traffic requiring preferential treatment
- Policy enforcement – Prioritize time sensitive traffic based QoS profiles, supports up to 10K users within a single SDN domain
- Visibility into per call SDN operation in terms of DSCP marking applied
- Granular sorting and display filtering – by IP address, media type, call quality and jitter
- History records management
- Configurable global template and phone trust

Cisco XNC Controller

Cisco created the Cisco XNC Controller, in order to keep up with the changing software-defined networking (SDN) environments. Its support of OpenFlow, the most widely used SDN communications standard, helps it integrate into varied SDN deployments to enable organizations to better control and scale their networks. As of mid-2015, Cisco has retired the Cisco XNC Controller — see What the Cisco XNC Controller Tells Us About OpenDaylight to learn about XNC's demise.

As an SDN Controller, which is the "brains" of the network, Cisco XNC uses OpenFlow to communicate information "down" to the forwarding plane (switches and routers), with southbound APIs, and "up" to the applications and business logic, with northbound APIs. It enables organizations to deploy and even develop a variety of network services, using representational state transfer application program interfaces (REST APIs), as well as Java APIs.

The XNC is Cisco's implementation of the OpenDaylight stack. Cisco is a contributor to the OpenDaylight initiative, which is focused on developing open standards for SDN that promote innovation and interoperability. Cisco XNC is designed to deliver the cutting edge OpenDaylight technologies as commercial, enterprise-ready solutions.

Practical No: 03

Aim: Install ODL(Open Daylight) controller

ODL is one of the most versatile and largest open-source controllers. It is helpful for automating larger area networks and it is scalable. ODL is written in Java. Compared to all other SDN Controllers, this controller is the best out of all and it is well-known for its security.

The OpenFlow protocol, considered the first SDN standard, defines the open communications protocol that allows the SDN Controller to work with the forwarding plane and make changes to the network. This gives businesses the ability to better adapt to their changing needs, and have greater control over their networks.

The OpenDaylight Controller exposes open northbound APIs, which are used by applications. These applications use the controller to collect information about the network, run algorithms to conduct analytics, and then use the OpenDaylight Controller to create new rules throughout the network.

The OpenDaylight Controller is implemented solely in software and is kept within its own Java Virtual Machine (VM) (JVM). This means it can be deployed on hardware and any operating system platforms that support Java. The controller uses these tools:

- Maven for build automation
- OSGi for dynamically loading bundles and packaging JAR files
- JAVA interfaces for event listening, specifications, and forming patterns.
- REST APIs such as topology manager, host tracker, flow programmer, and static routing.

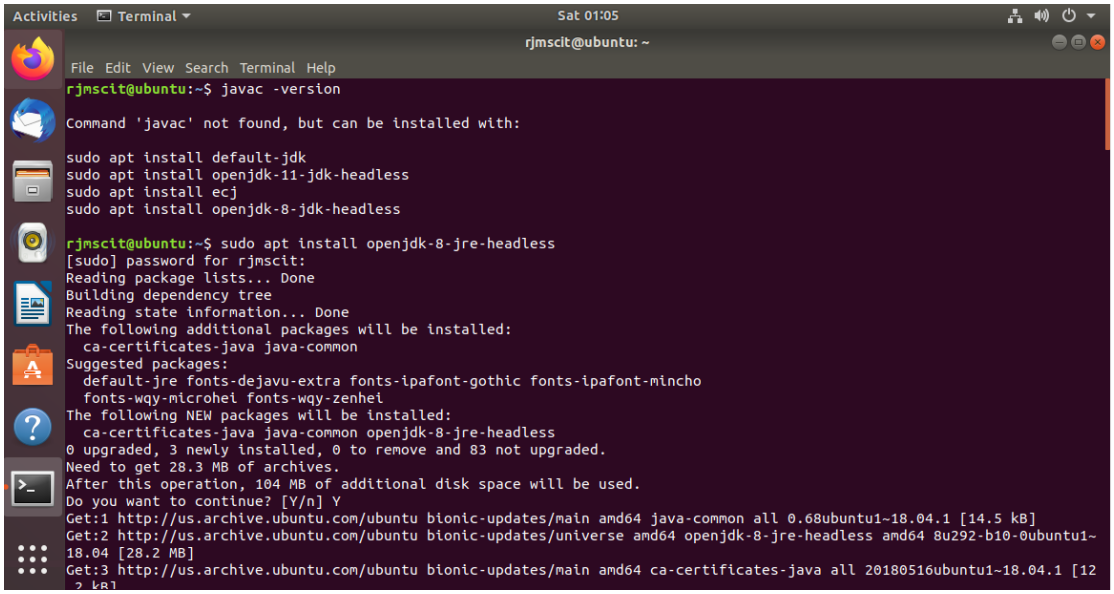
STEP 1: Install java and Set JAVA_HOME

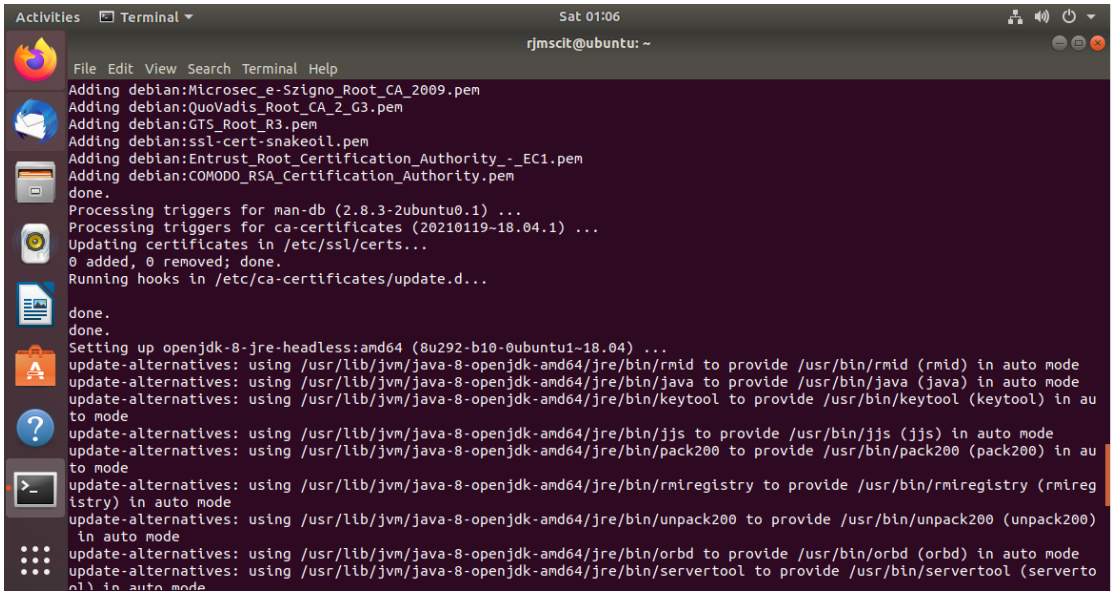
>>javac -version

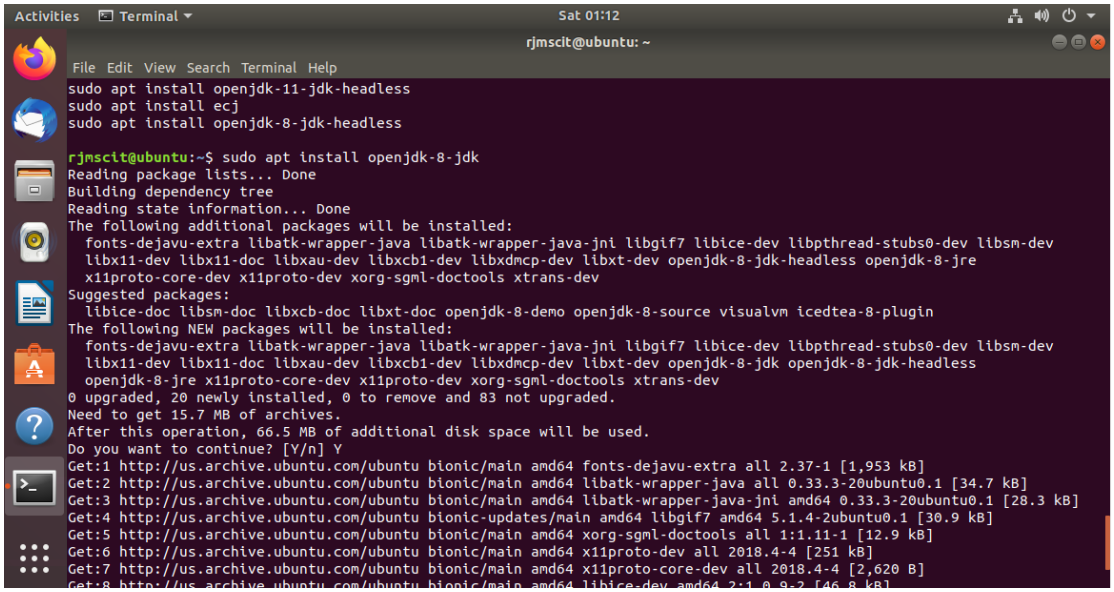
>>sudo apt install openjdk-8-jre-headless

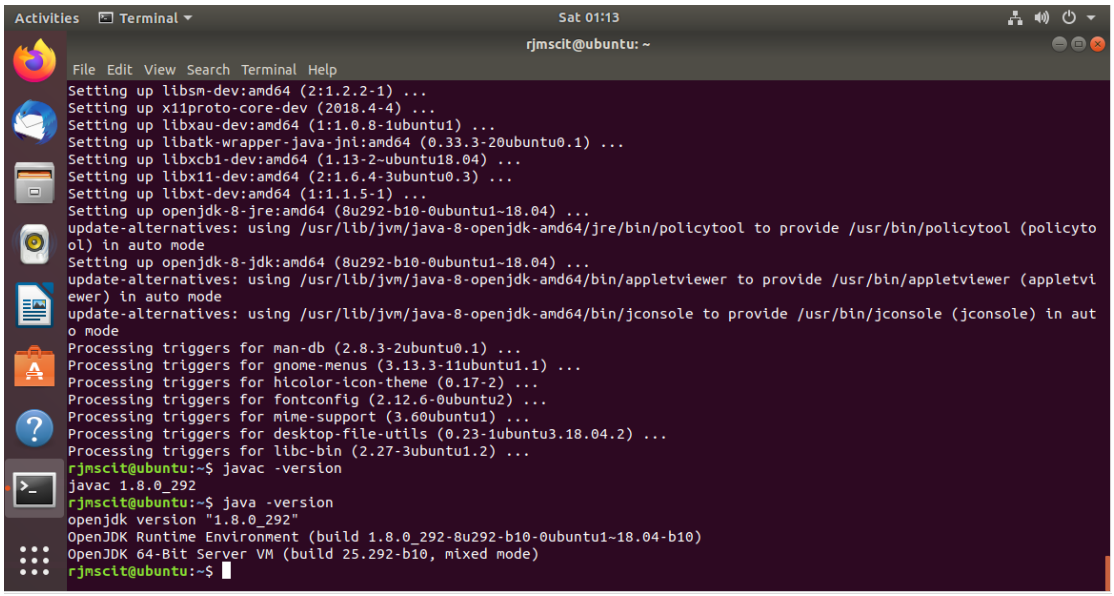
>>sudo apt install openjdk-8-jdk

>>java -version


Activities
Terminal
Sat 01:05
rjmscit@ubuntu: ~
File Edit View Search Terminal Help
rjmscit@ubuntu:~$ javac -version
Command 'javac' not found, but can be installed with:
sudo apt install default-jdk
sudo apt install openjdk-11-jdk-headless
sudo apt install ecj
sudo apt install openjdk-8-jdk-headless
rjmscit@ubuntu:~$ sudo apt install openjdk-8-jre-headless
[sudo] password for rjmscit:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
ca-certificates-java java-common
Suggested packages:
default-jre fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho
fonts-wqy-microhei fonts-wqy-zenhei
The following NEW packages will be installed:
ca-certificates-java java-common openjdk-8-jre-headless
0 upgraded, 3 newly installed, 0 to remove and 83 not upgraded.
Need to get 28.3 MB of archives.
After this operation, 104 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 java-common all 0.68ubuntu1~18.04.1 [14.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 openjdk-8-jre-headless amd64 8u292-b10-0ubuntu1~
18.04 [28.2 MB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ca-certificates-java all 20180516ubuntu1~18.04.1 [12


Activities
Terminal
Sat 01:06
rjmscit@ubuntu: ~
File Edit View Search Terminal Help
Adding debian:Microsec_e-Szigno_Root_CA_2009.pem
Adding debian:QuoVadis_Root_CA_2_G3.pem
Adding debian:GTS_Root_R3.pem
Adding debian:ssl-cert-snakeoil.pem
Adding debian:Entrust_Root_Certification_Authority_-_EC1.pem
Adding debian:COMODO_RSA_Certification_Authority.pem
done.
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ca-certificates (20210119~18.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
Setting up openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/rmid to provide /usr/bin/rmid (rmid) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java to provide /usr/bin/java (java) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/keytool to provide /usr/bin/keytool (keytool) in au
to mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/jjs to provide /usr/bin/jjs (jjs) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/pack200 to provide /usr/bin/pack200 (pack200) in au
to mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/rmiregistry to provide /usr/bin/rmiregistry (rmireg
istry) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/unpack200 to provide /usr/bin/unpack200 (unpack200)
in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (serverto
ol) in auto mode





>> sudo nano /etc/environment

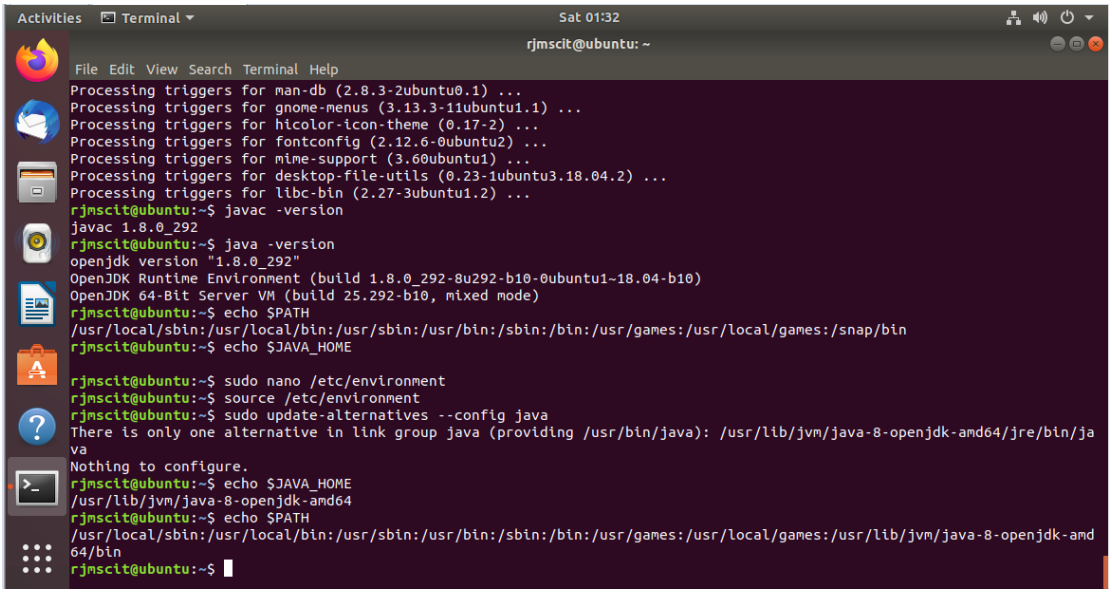
>>paste at first line - JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"

>>paste at end of Path - /usr/lib/jvm/java-8-openjdk-amd64/bin

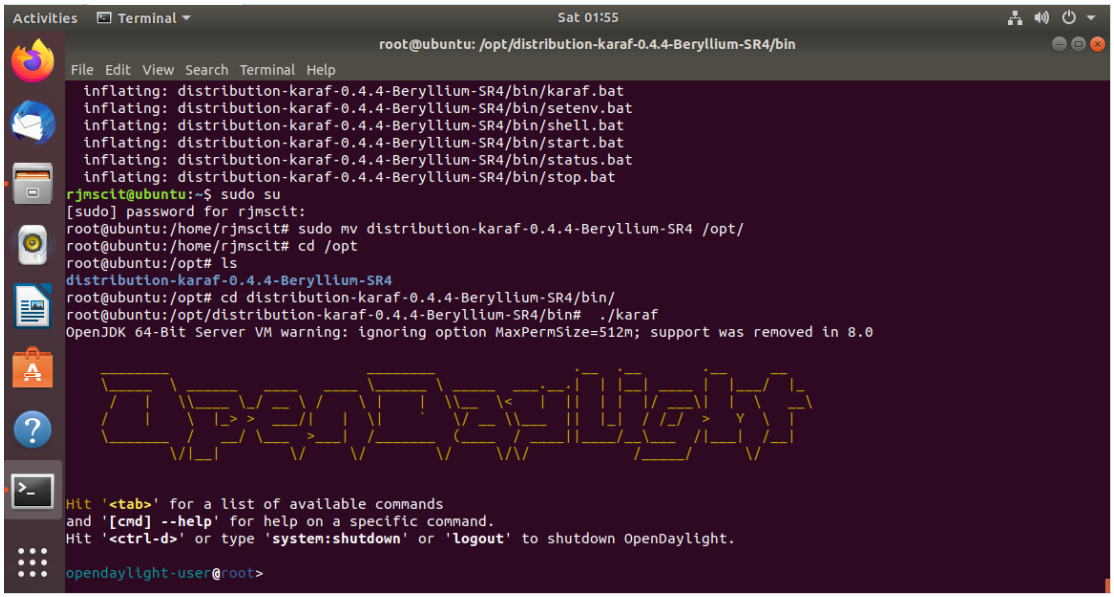
>>source /etc/environment

>>echo $JAVA_HOME

>>echo $PATH



STEP 2: Download **distribution-karaf-0.6.0-Carbon.zip** and unzip it Move to the directory distribution-karaf-0.6.0-Carbon and type: ./bin/karaf



STEP 3: Install features

following ARE ODL-DLUX-ALL FEATURES

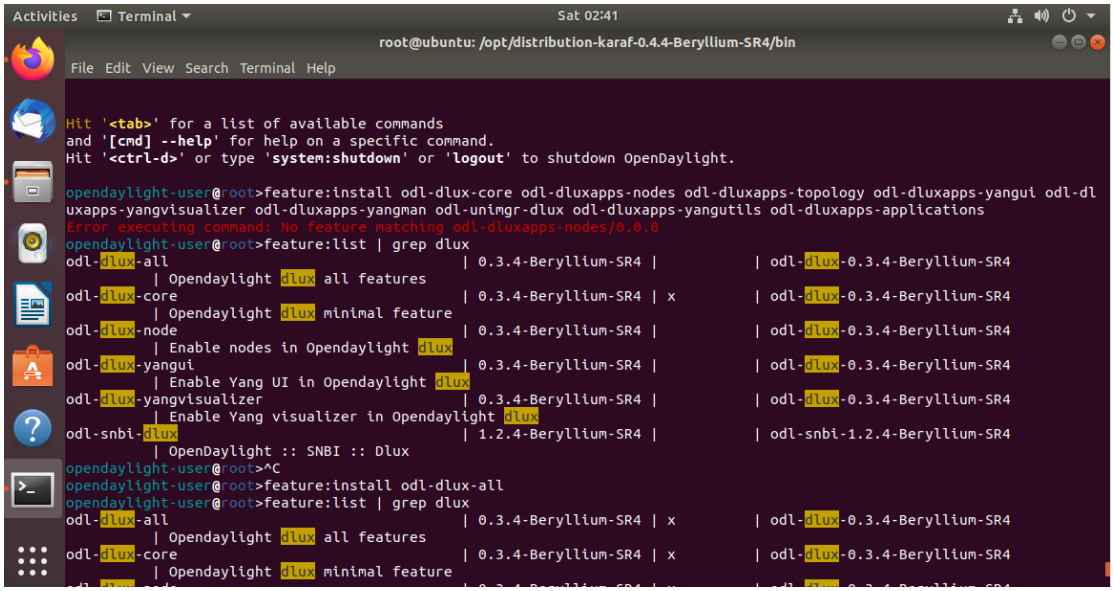
opendaylight-user@root> feature:install odl-dlux-core
opendaylight-user@root>feature:install odl-dluxapps-nodes
opendaylight-user@root>feature:install odl-dluxapps-topology
opendaylight-user@root>feature:install odl-dluxapps-yangui
opendaylight-user@root>feature:install odl-dluxapps-yangvisualizer
opendaylight-user@root>feature:install odl-dluxapps-yangman

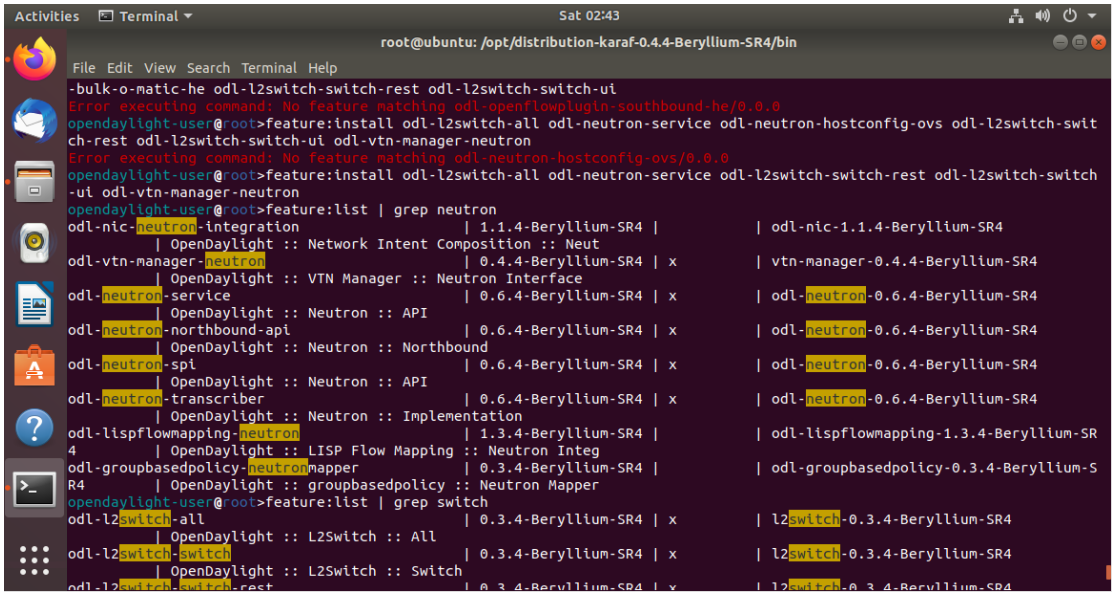
opendaylight-user@root>feature:install odl-dlux-all
opendaylight-user@root> feature:install odl-restconf odl-l2switch-switch odl- mdsal-apidocs
opendaylight-user@root> feature:install odl-l2switch-all

opendaylight-user@root> feature:install odl-vtn-manager-neutron odl-neutron- service odl-neutron-hostconfig-ovs

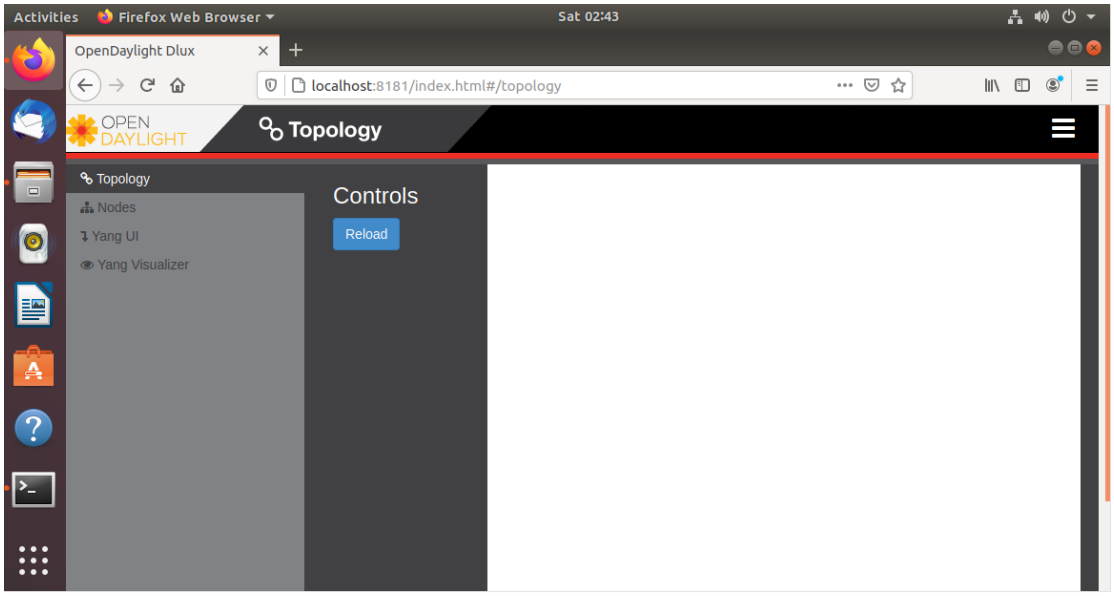
opendaylight-user@root>feature:install odl-ovsdb-library odl-restconf-all odl- ovsdb-southbound-api odl-ovsdb-southbound-impl odl-ovsdb-southbound-impl- rest

opendaylight-user@root>feature:install odl-netvirt-openstack





Even ODL has dashboard DLUX feature which gives GUI web interface to ODL controller at URL **http://localhost:8181/index.html OR http://ip:8181/index.html** on which ODL user can log in (by default username and password is 'admin') and check network topology on the browser of the system where ODL is running.



Practical No: 04

Aim: Implement OVS

Open vSwitch was created by the team at Nicira, that was later acquired by VMware. OVS was intended to meet the needs of the open source community, since there was no a feature-rich virtual switch offering designed for Linux-based hypervisors, such as KVM and XEN. OVS has quickly become the de facto virtual switch for XEN environments, and it is now playing a large part in other open source projects, like OpenStack.

OVS supports NetFlow, sFlow, port mirroring, VLANs, LACP, etc. From a control and management perspective, Open vSwitch leverages OpenFlow and the Open vSwitch Database (OVSDB) management protocol, which means it can operate both as a soft switch running within the hypervisor, and as the control stack for switching silicon. Other important ways OVS is incorporated in software-defined networking (SDN) include:

- OVS is critical to many SDN deployments in data centers because it ties together all the virtual machines (VMs) within a hypervisor instance on a server
- It is the first entry point for all the VMs sending traffic to the network and is the ingress point into overlay networks running on top of physical networks in the data center
- Using OVS for virtual networking is considered the core element of many datacenter SDN deployments and the main use case is multi-tenant network virtualization
- OVS can also be used to direct traffic between network functions in service chaining use cases

OVS differs from the commercial offerings from VMware and Cisco. One point worth noting about OVS is that there is not a native SDN Controller or manager, like the Virtual Supervisor Manager (VSM) in the Cisco 1000V or vCenter in the case of VMware's distributed switch. Open vSwitch is meant to be controlled and managed by third party controllers and managers.

Once OpenFlow and OVSDB plugins are installed, ODL will starts listening for OpenFlow and OVSDB connections at 6653 and 6640 port respectively. This can be checked at controller by running the following commands:

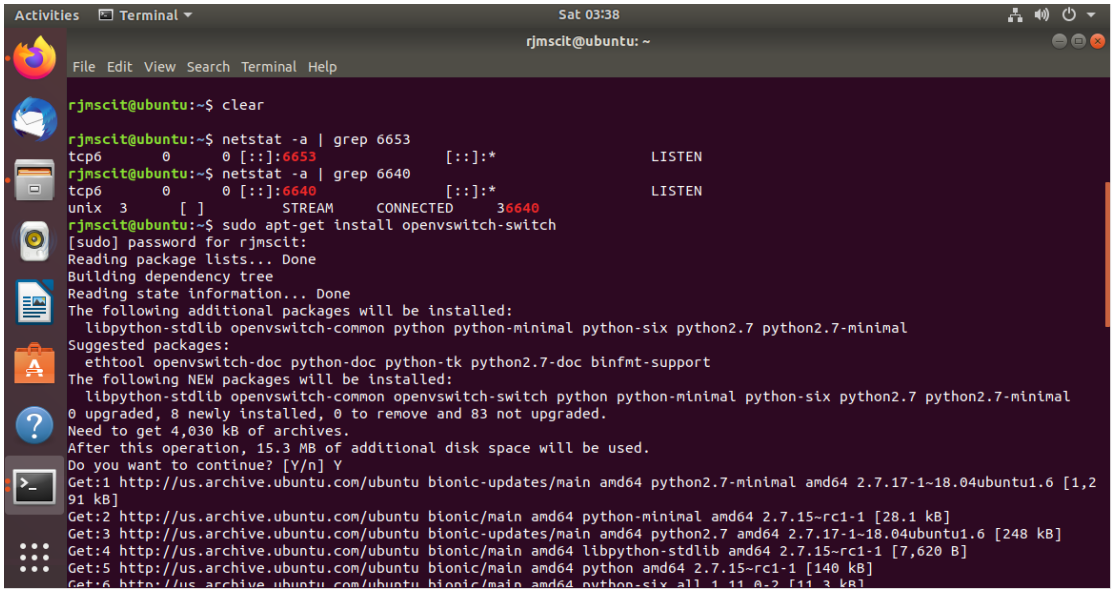
First run ODL then enter below command in terminal:

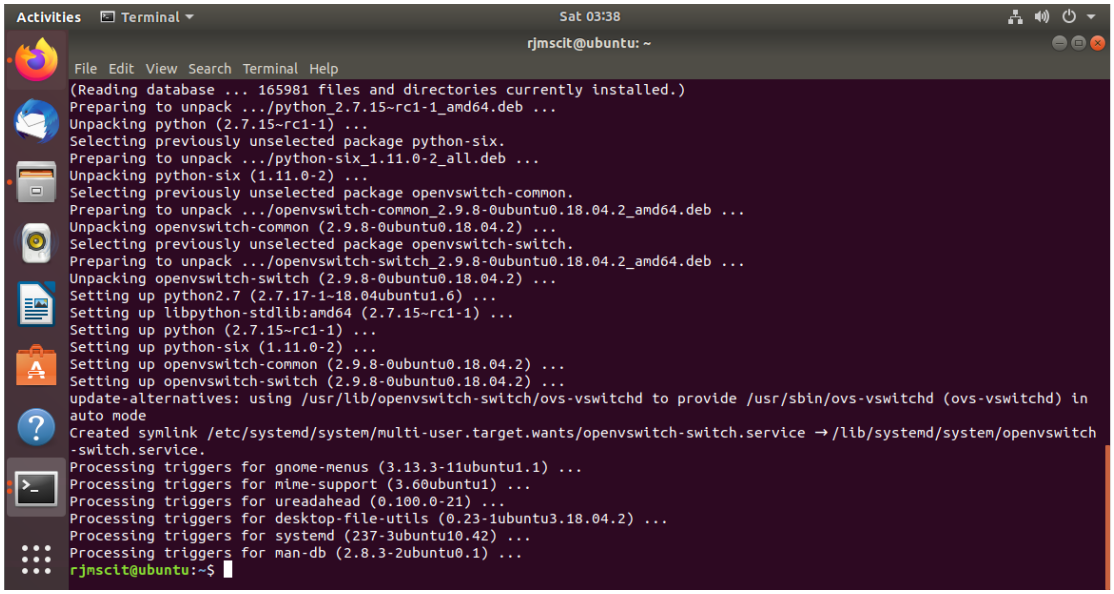
>>netstat -a | grep 6653

>>netstat -a | grep 6640

And install openvswitch

>>sudo apt-get install openvswitch-switch



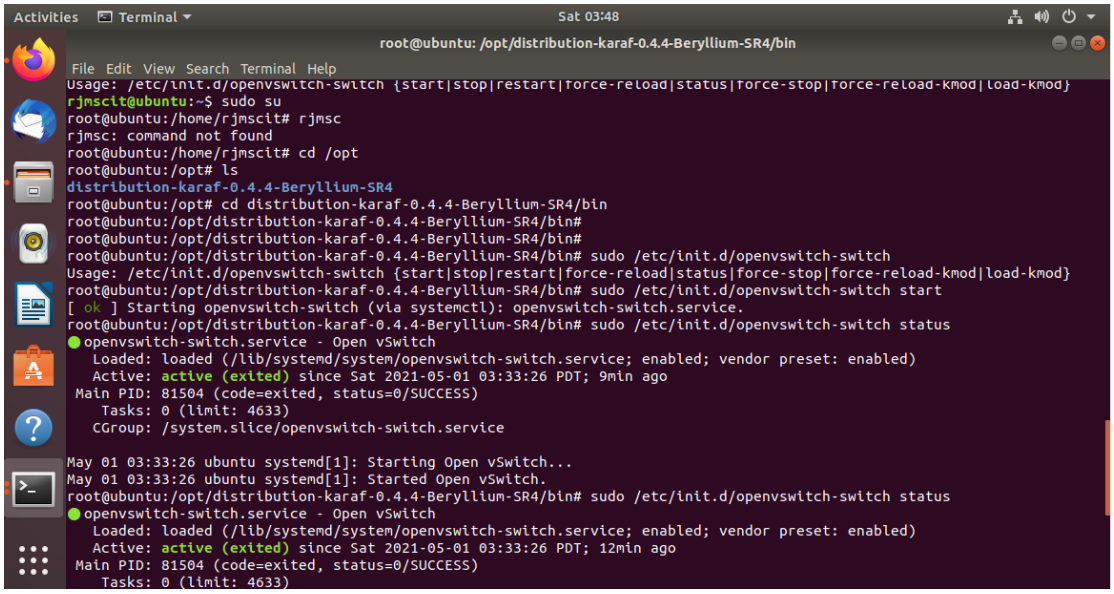


<u>Now go to bin repository of Karaf and run below command to start openvswitch:</u>

>>sudo /etc/init.d/openvswitch-switch start

>>sudo /etc/init.d/openvswitch-switch status

>>sudo ovs-vsctl show



At OVS side, below command is run to initiate OVSDB connection with controller:

ovs-vsctl set-manager tcp:<controller-IP>:6640

To confirm that the connection is established, below command can be run which shows 'is_connected' flag is set to true in the output.

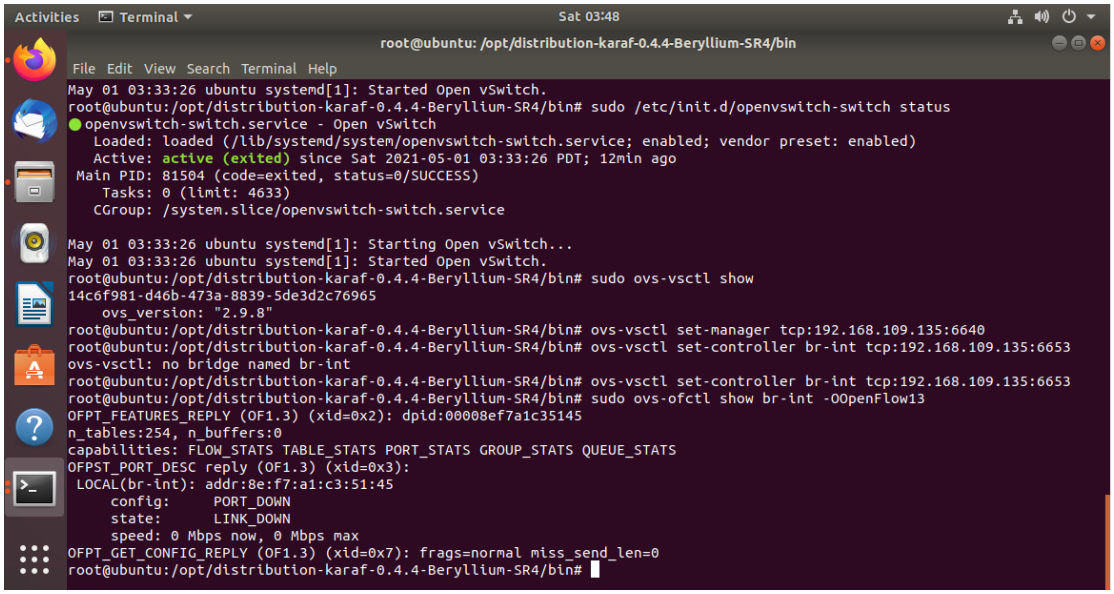
OpenFlow connection is made on bridge, so either you can create bridge on OVS or ODL can also create bridge on OVS by sending configuration to OVS via OVSDB connection. Over the bridge in OVS, below command can be run to connect bridge to ODL, which establishes establishing OpenFlow connection between ODL and OVS.

ovs-vsctl set-controller <bridge name> tcp:<controller-IP>:6653

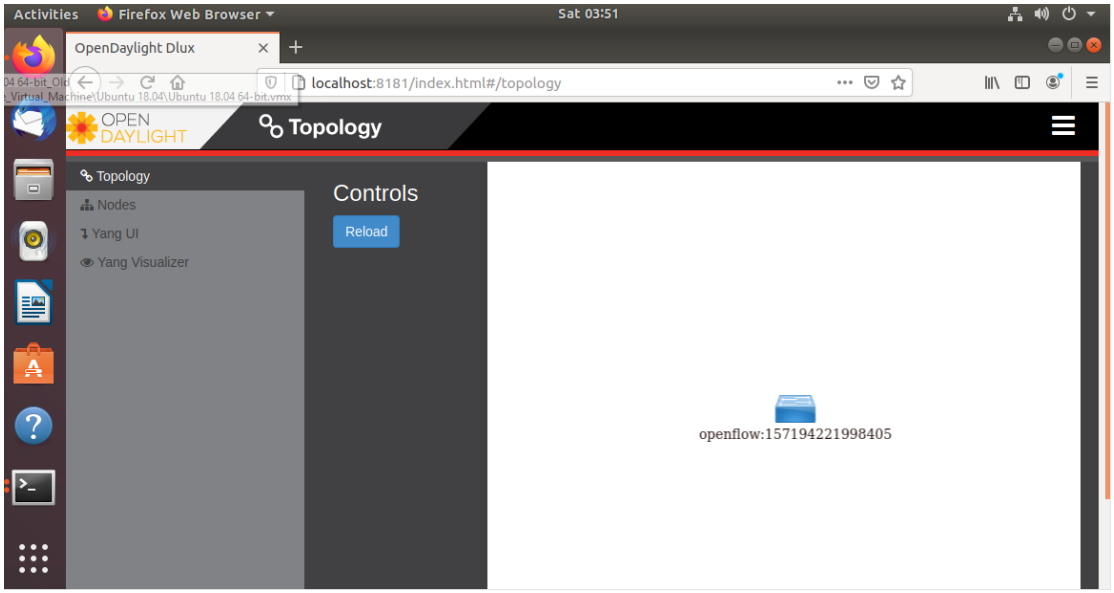
ovs-vsctl set-controller br-int tcp: 192.168.109.135:6653

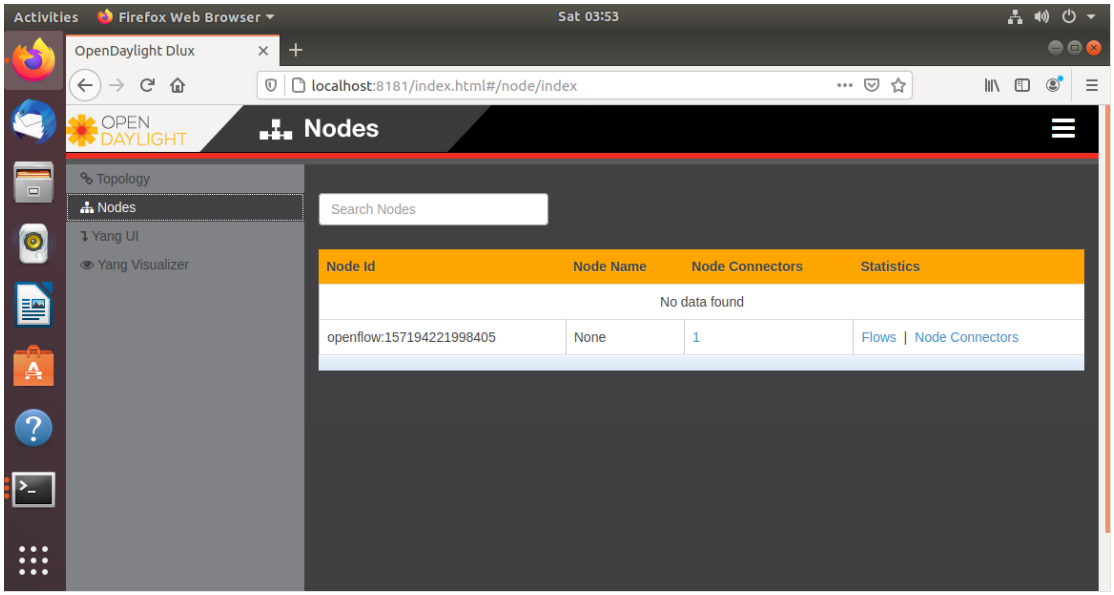
At the OVS side, the following command can be run to show details of bridge.

sudo ovs-ofctl show br-int -OOpenFlow13



Restart odl and login to view OpenFlow switch




Activities
Firefox Web Browser
Sat 03:53
OpenDaylight Dlux
localhost:8181/index.html#/node/index
OPEN DAYLIGHT
Nodes
Topology
Nodes
Yang UI
Yang Visualizer
Search Nodes
Node Id
Node Name
Node Connectors
Statistics
No data found
openflow:157194221998405
None
1
Flows | Node Connectors

PRACTICAL NO: 05

Aim: Implement Mininet on ODL

Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

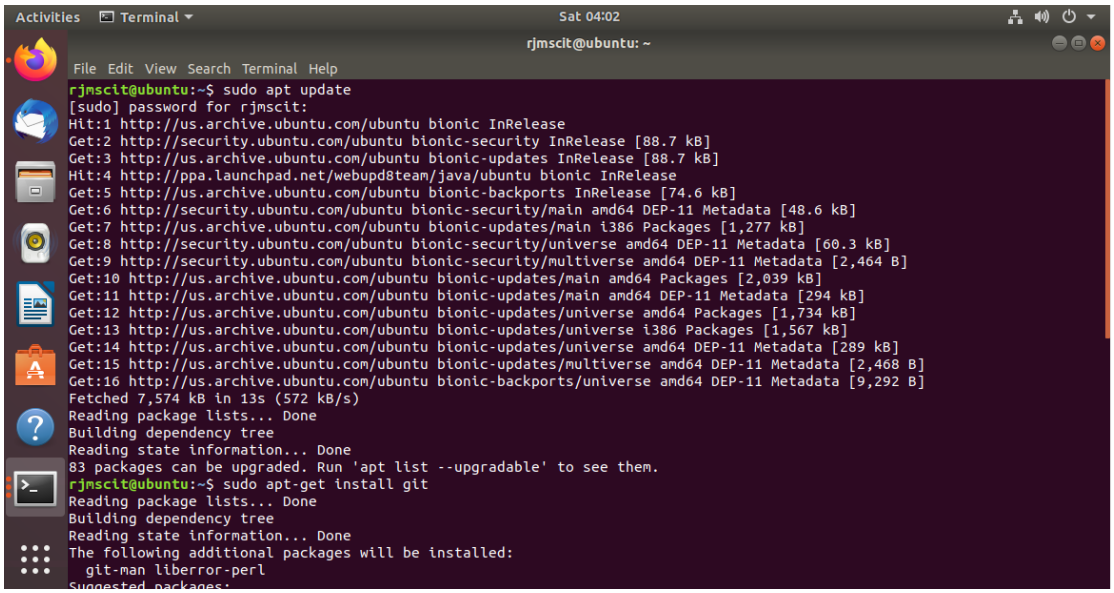
Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.

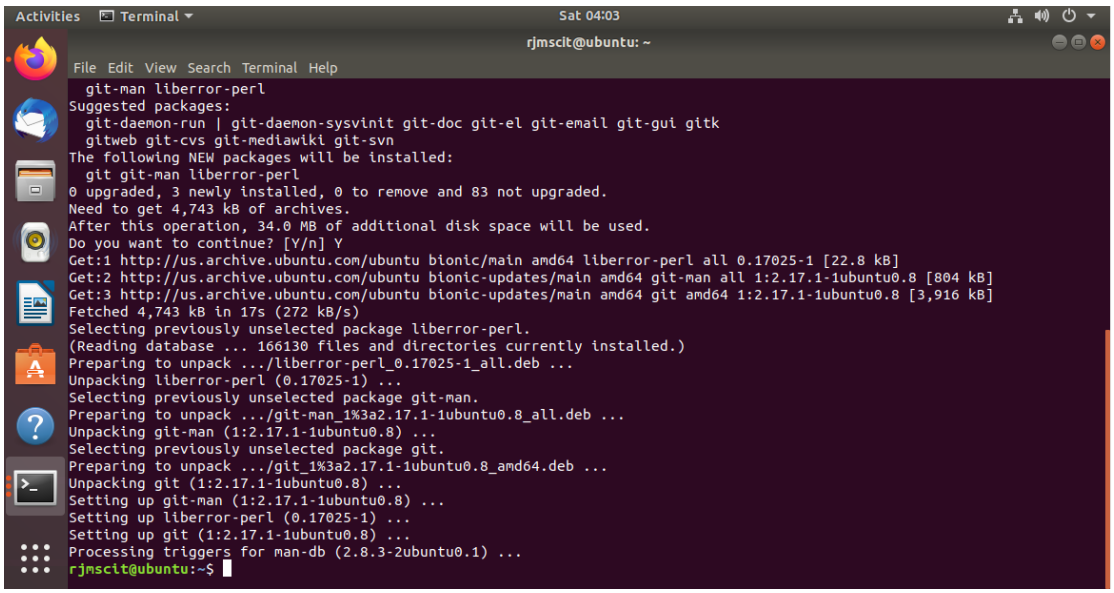
Mininet:

- Provides a simple and inexpensive network testbed for developing OpenFlow applications
- Enables multiple concurrent developers to work independently on the same topology
- Supports system-level regression tests, which are repeatable and easily packaged
- Enables complex topology testing, without the need to wire up a physical network
- Includes a CLI that is topology-aware and OpenFlow-aware, for debugging or running network-wide tests
- Supports arbitrary custom topologies, and includes a basic set of parametrized topologies
- is usable out of the box without programming, but
- also Provides a straightforward and extensible Python API for network creation and experimentation
- Mininet provides an easy way to get correct system behavior (and, to the extent supported by your hardware, performance) and to experiment with topologies.
- Mininet networks run real code including standard Unix/Linux network applications as well as the real Linux kernel and network stack (including any kernel extensions which you may have available, as long as they are compatible with network namespaces.)
- Because of this, the code you develop and test on Mininet, for an OpenFlow controller, modified switch, or host, can move to a real system with minimal changes, for real-world testing, performance evaluation, and deployment. Importantly this means that a design that works in Mininet can usually move directly to hardware switches for line-rate packet forwarding.

>> sudo apt update

>> sudo apt-get install git



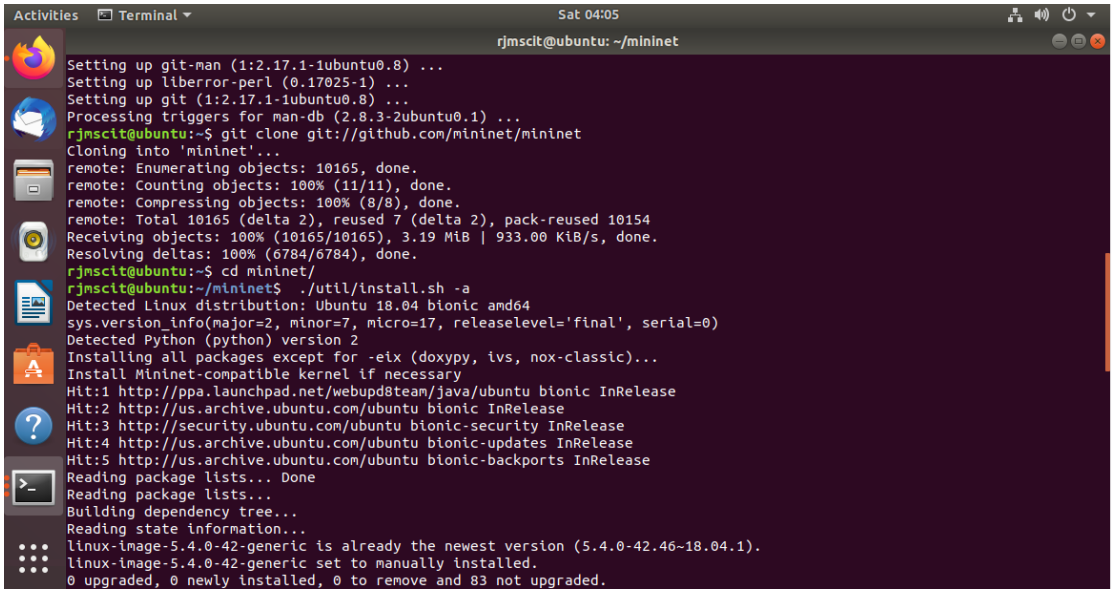


>> git clone git://github.com/mininet/mininet

>> cd mininet

The Mininet project provides an install script. Run the script.

>> ./util/install.sh -a (to install all feature of mininet)



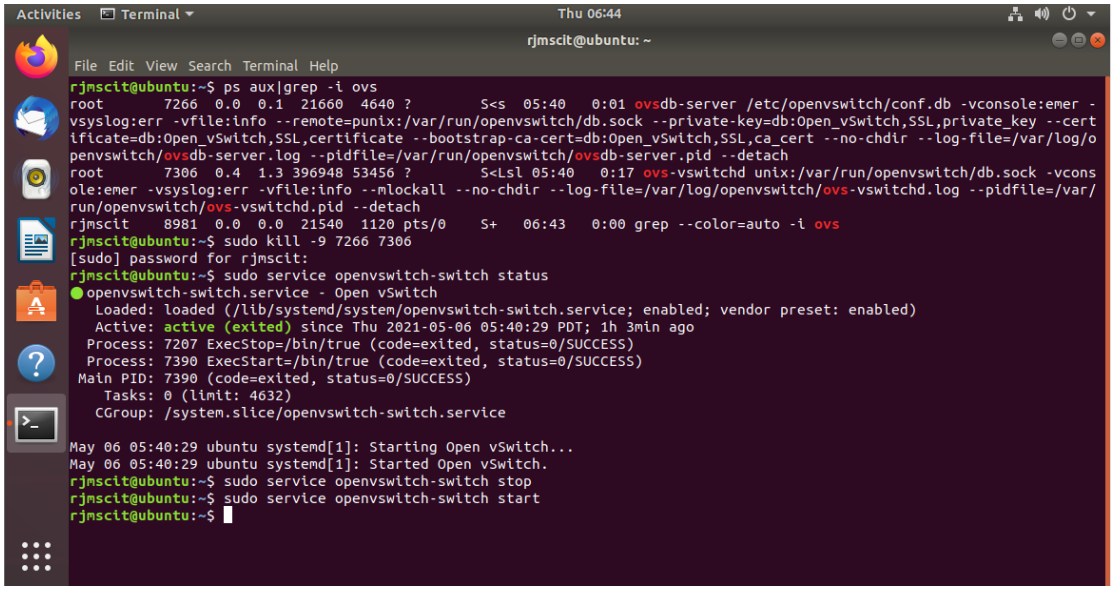
>> ps aux|grep -i ovs

>> sudo kill -9 7266 7306

>> sudo service openvswitch-switch status

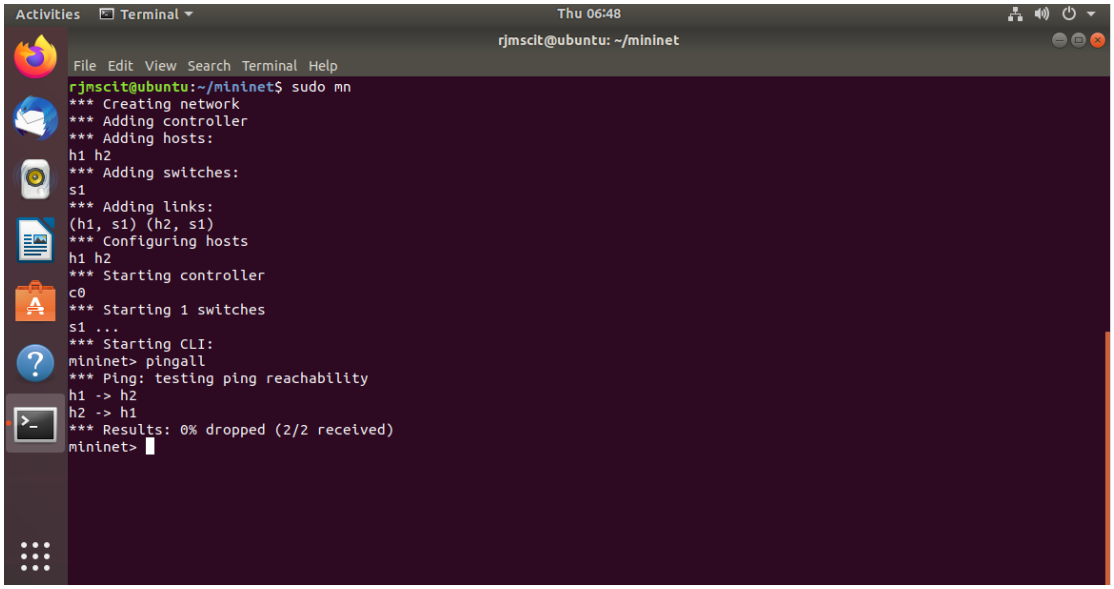
>> sudo service openswitch-switch stop

>> service openvswitch-switch start

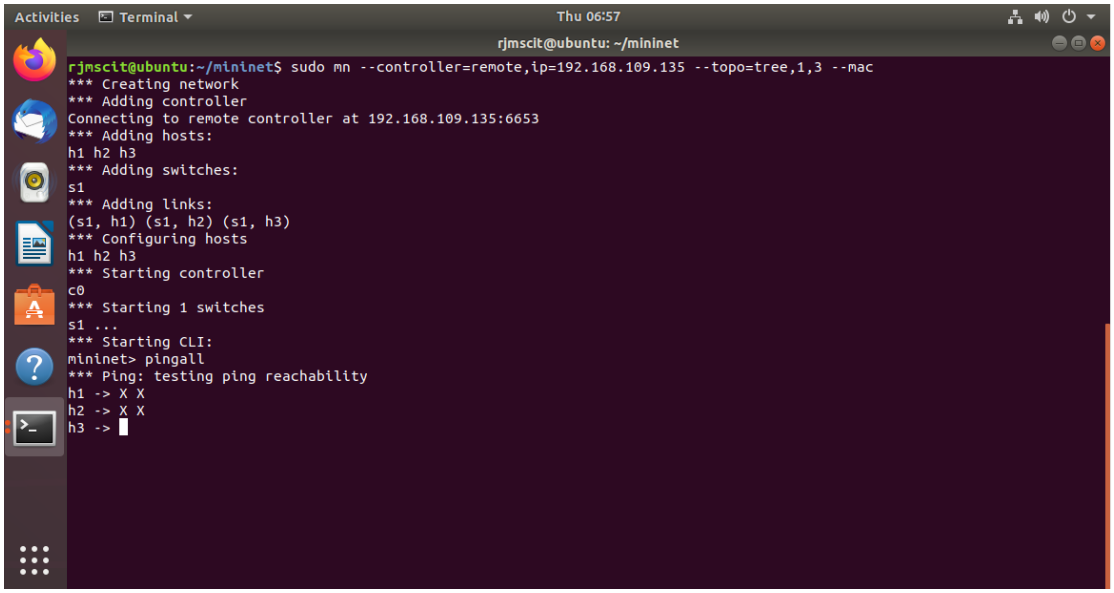


>> sudo mn

>> sudo mn --test pingall



>> sudo mn --controller=remote,ip=192.168.109.135 --topo=tree,1,3 --mac

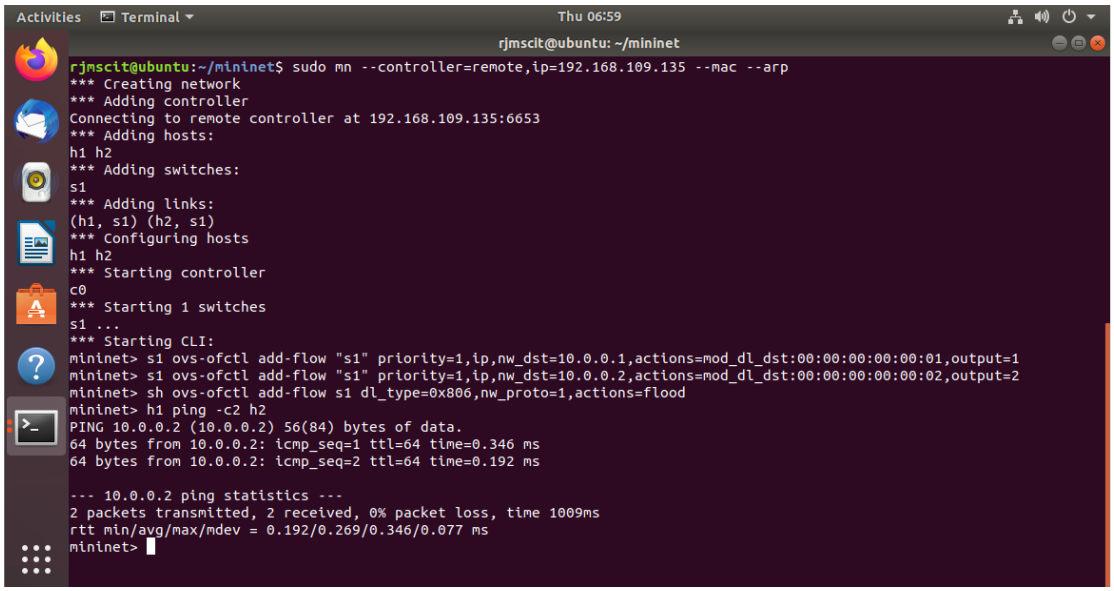


sudo mn --controller=remote,ip=192.168.109.135 --mac --arp

mininet> s1 ovs-ofctl add-flow "s1"
priority=1,ip,nw_dst=10.0.0.1,actions=mod_dl_dst:00:00:00:00:00:01,output=1

mininet> s1 ovs-ofctl add-flow "s1"
priority=1,ip,nw_dst=10.0.0.2,actions=mod_dl_dst:00:00:00:00:00:02,output=2

mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood
mininet> h1 ping -c2 h2



PRACTICAL NO: 06

Aim: Install RYU controller with mininet topology

Ryu Controller is an open, software-defined networking (SDN) Controller designed to increase the agility of the network by making it easy to manage and adapt how traffic is handled. In general, the SDN Controller is the brain of the SDN environment, communicating information down to the switches and routers with southbound APIs, and up to the applications and business logic with northbound APIs. The Ryu Controller is supported by NTT and is deployed in NTT cloud data centers as well.

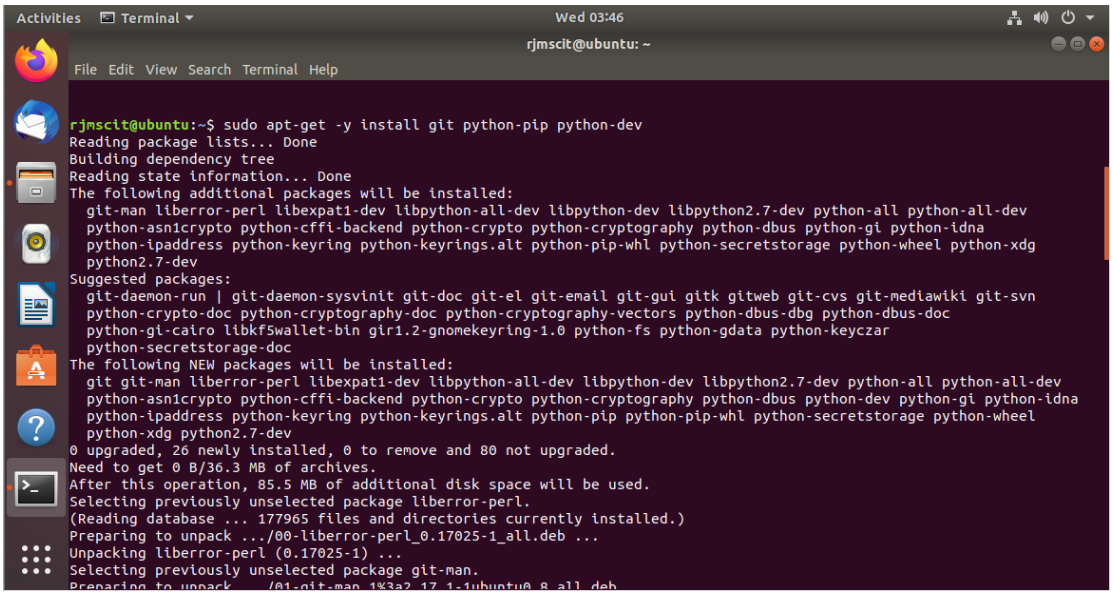
The Ryu Controller provides software components, with well-defined application program interfaces (APIs), that make it easy for developers to create new network management and control applications. This component approach helps organizations customize deployments to meet their specific needs; developers can quickly and easily modify existing components or implement their own to ensure the underlying network can meet the changing demands of their applications.

The Ryu Controller source code is hosted on GitHub and managed and maintained by the open Ryu community. OpenStack, which runs an open collaboration focused on developing a cloud operating system that can control the compute, storage, and networking resources of an organization, supports deployments of Ryu as the Network Controller.

Written entirely in Python, all of Ryu's code is available under the Apache 2.0 license and open for anyone to use. The Ryu Controller supports NETCONF and OF-config network management protocols, as well as OpenFlow, which is one of the first and most widely deployed SDN communications standards. It also supports Nicira extensions.

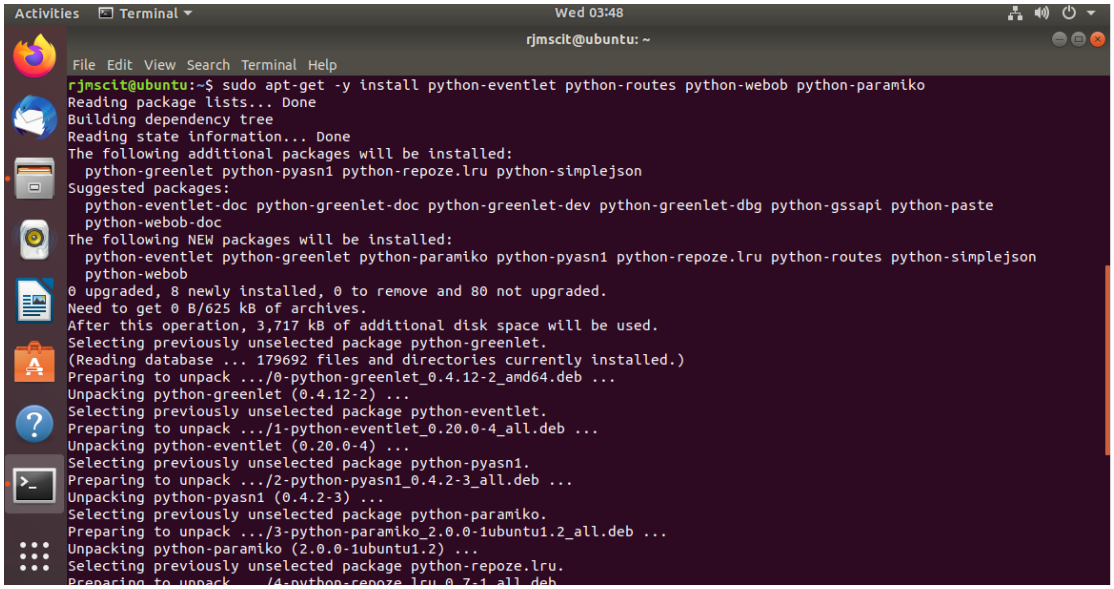
Step 1 -> Install tools

>> sudo apt-get -y install git python-pip python-dev



Step 2 -> Install python packages

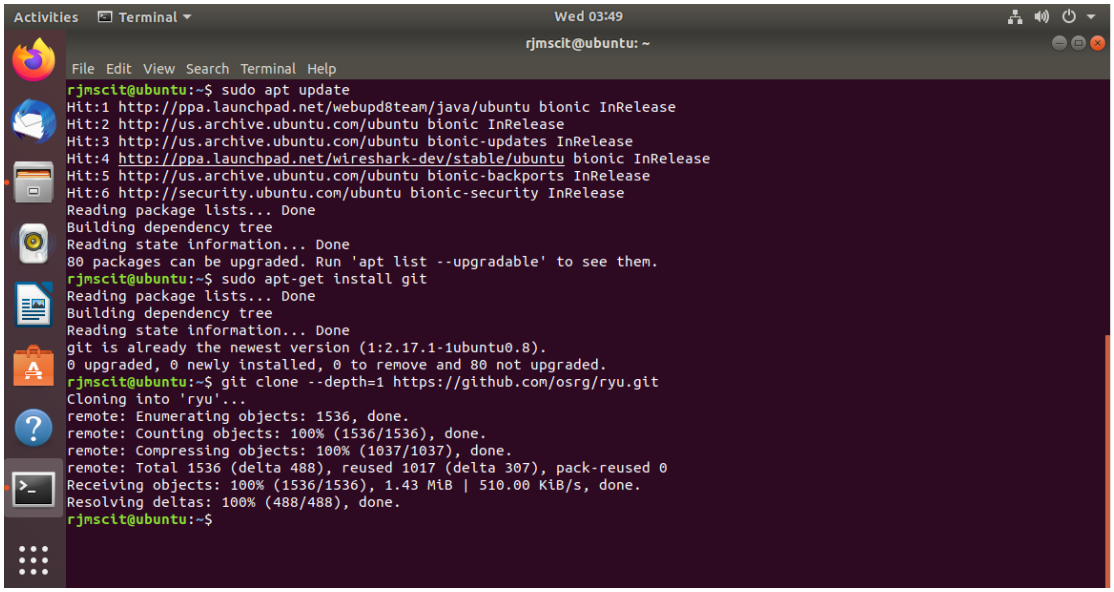
>> sudo apt-get -y install python-eventlet python-routes python-webob python-paramiko



Step 3 -> Clone RYU git Repo

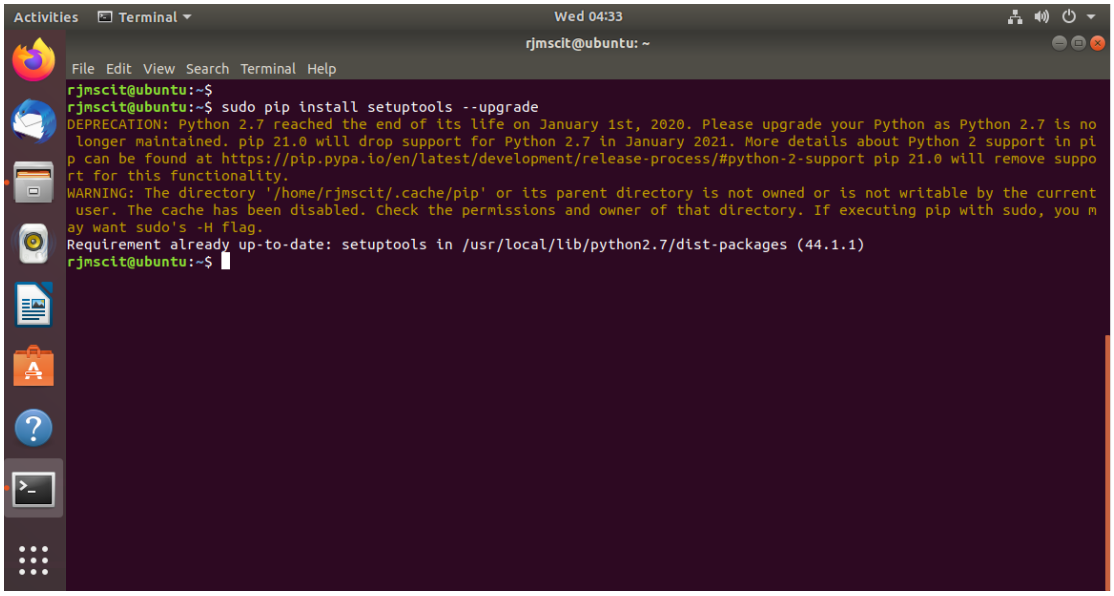
>> sudo apt-get install git

>> git clone --depth=1 https://github.com/osrg/ryu.git



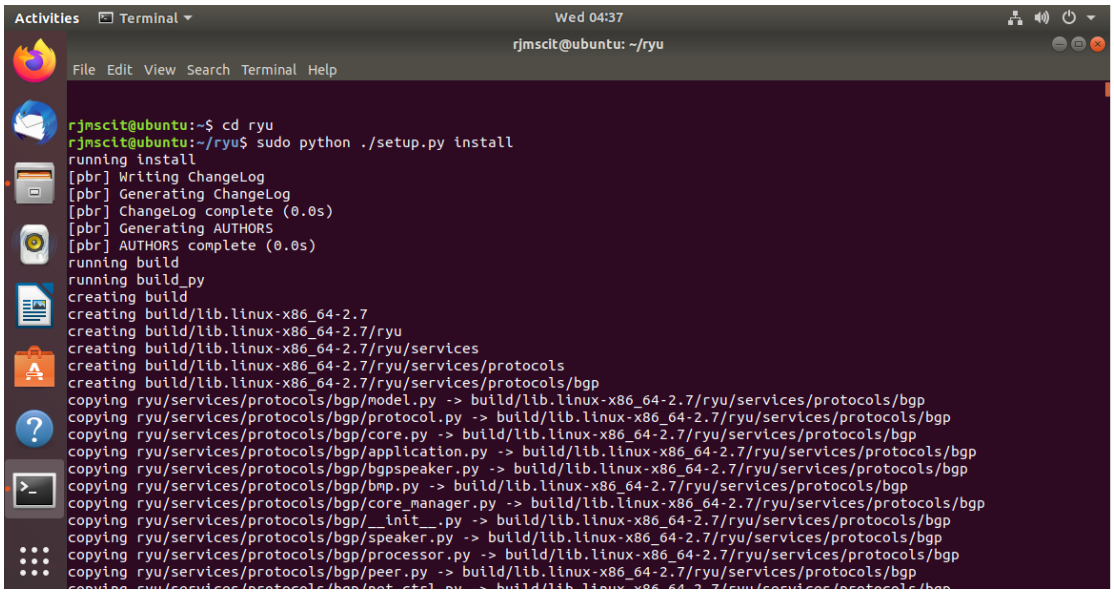
Step 4 -> Install RYU

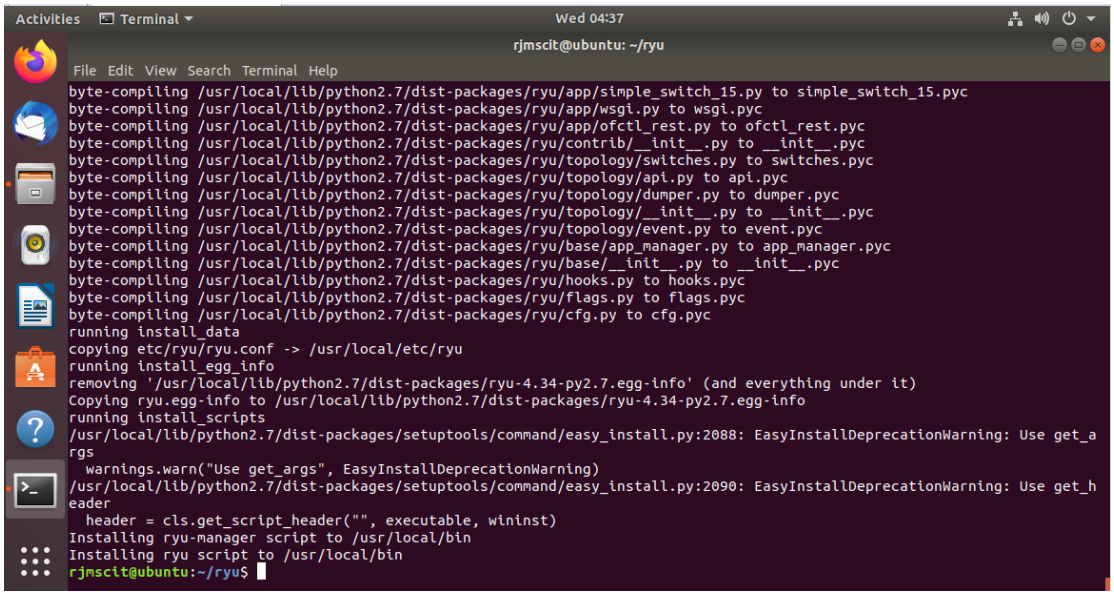
>> sudo pip install setuptools --upgrade



>> cd ryu

>> sudo python ./setup.py install



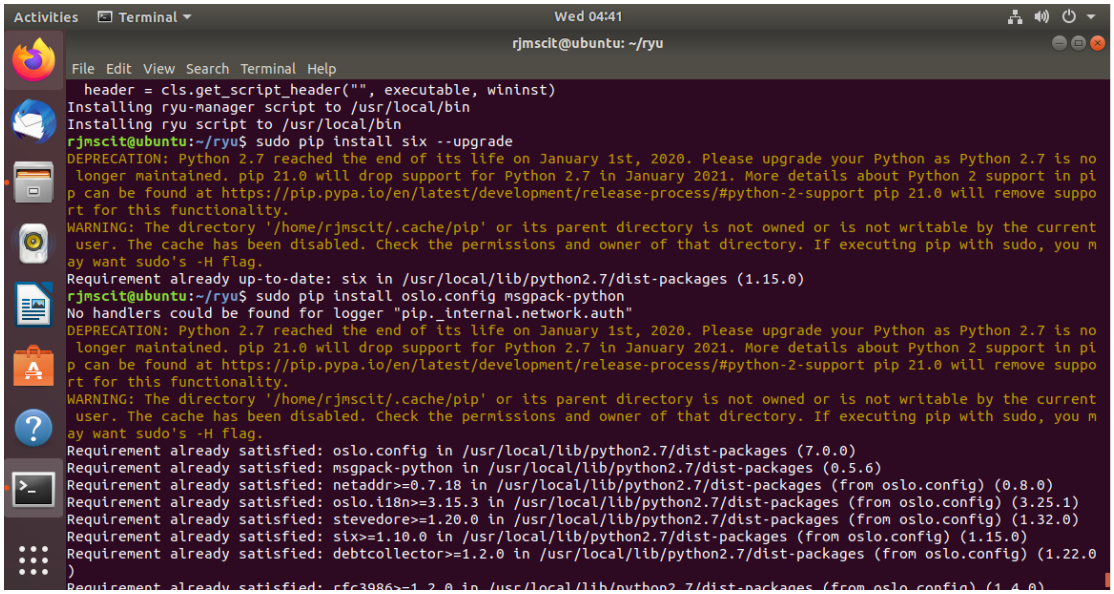


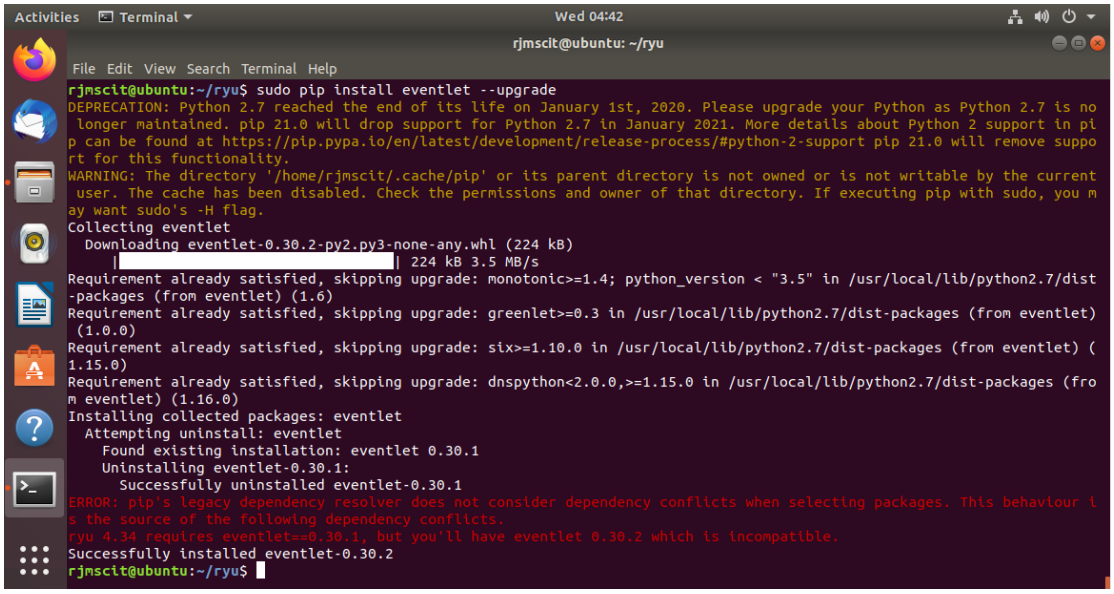
Step 5 -> Install and Update python packages

>> sudo pip install six –upgrade

>> sudo pip install oslo.config msgpack-python

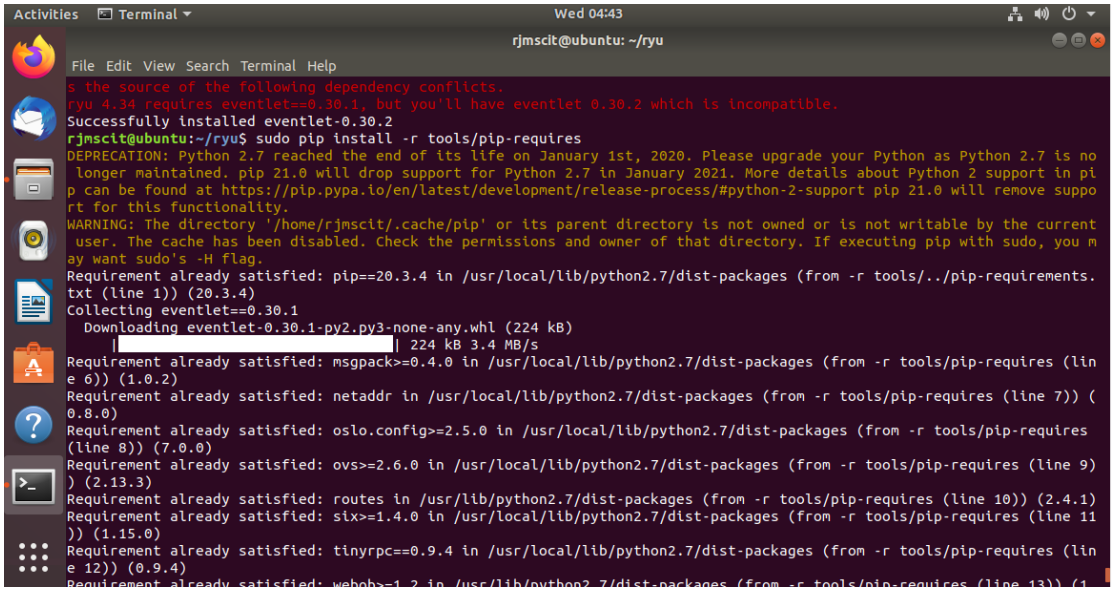
>> sudo pip install eventlet –upgrade



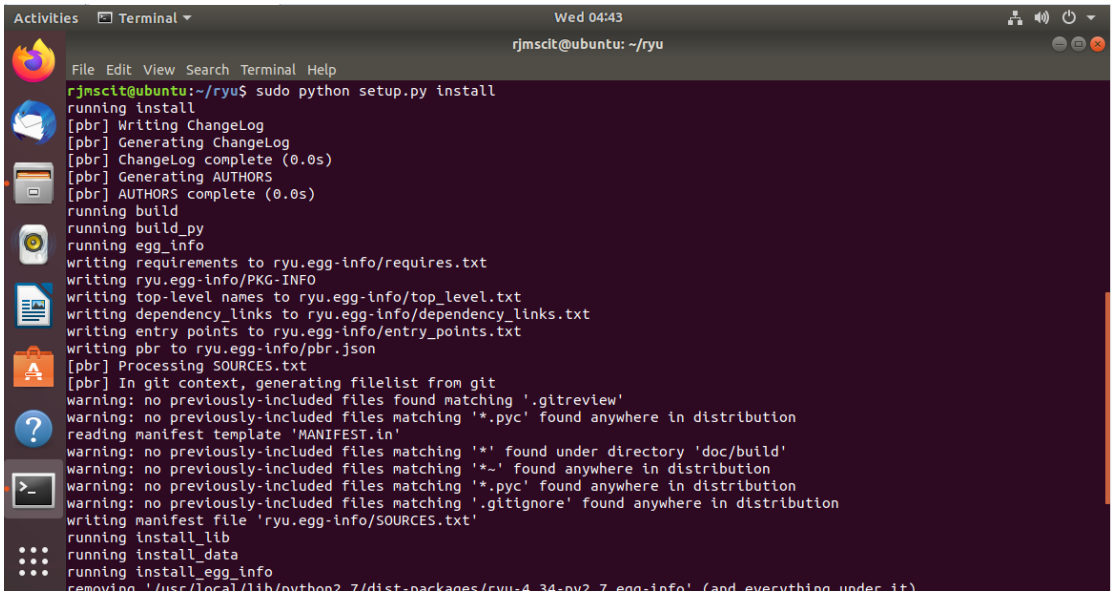


Step 6 -> Test ryu-manager

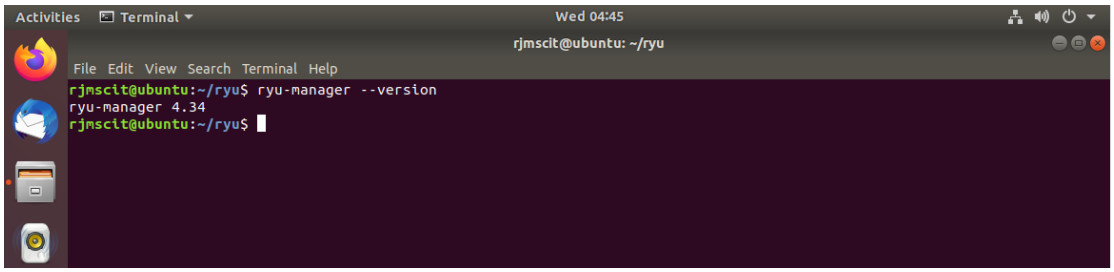
>>sudo pip install -r tools/pip-requires



>> sudo python setup.py install



>> ryu-manager --version



As we already perform OVS/ODL practical therefore ports are already in use, we need to kill them,

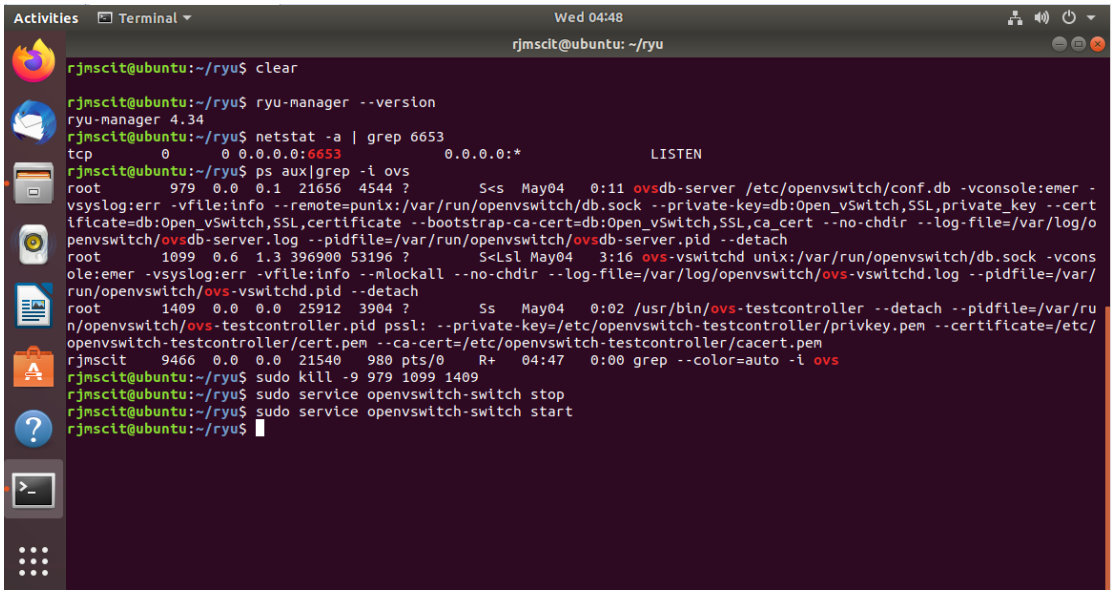
>> netstat -a | grep 6653

>> ps aux|grep -i ovs

>> sudo kill -9 970 1029 1223 11131

>> sudo service openvswitch-switch stop

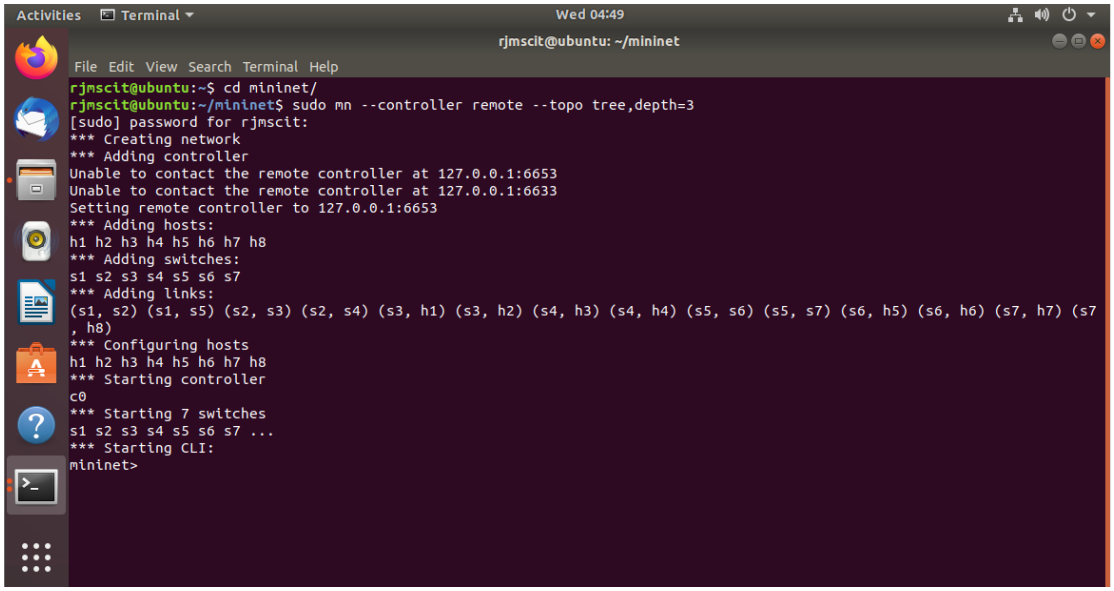
>> sudo service openvswitch-switch start



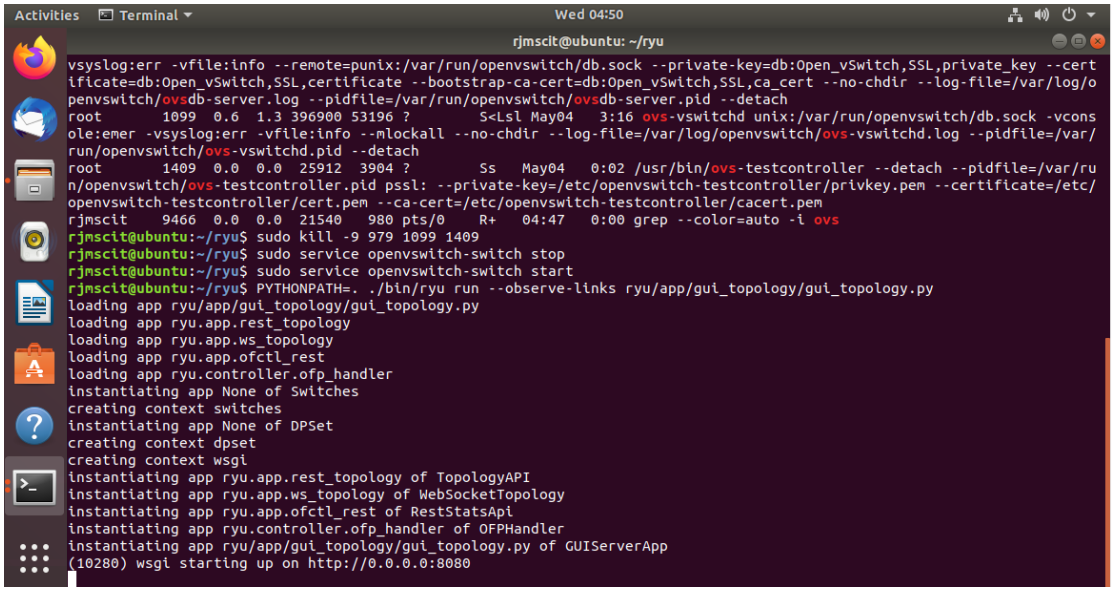
Topology Viewer

Run mininet in new terminal (or join your real environment):

>> sudo mn --controller remote --topo tree,depth=3



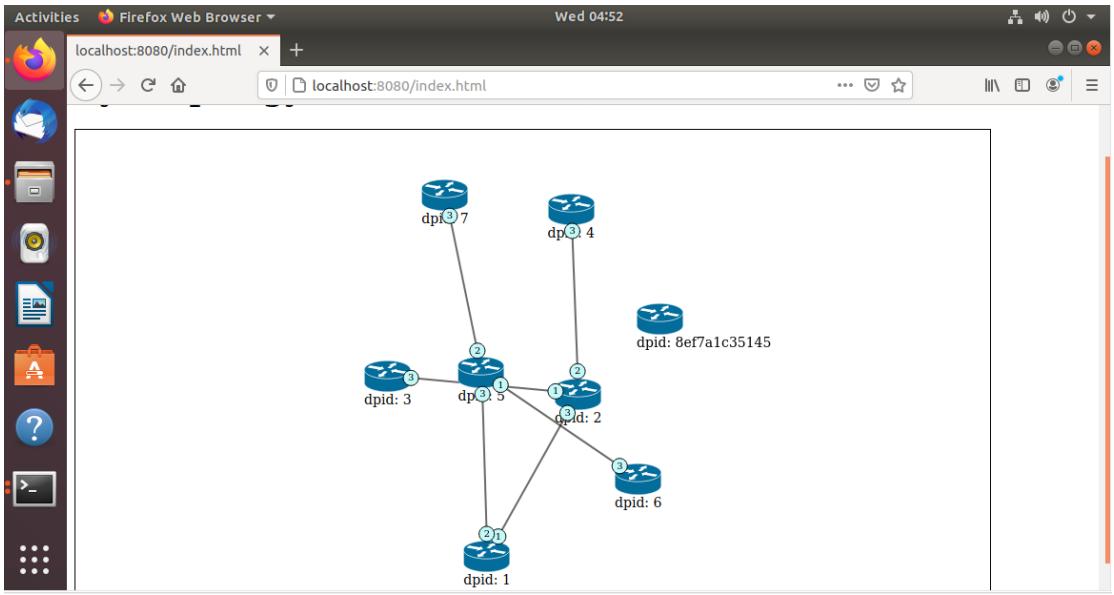
>> PYTHONPATH=. ./bin/ryu run --observe-links ryu/app/gui_topology/gui_topology.py



Now open Mozilla and run below URL:

>> localhost:8080/index.html

Output:



PRACTICAL NO: 07

Aim: Install floodlight on Ubuntu

Floodlight Controller is an SDN Controller developed by an open community of developers, many of which from Big Switch Networks, that uses with the OpenFlow protocol to orchestrate traffic flows in a software-defined networking (SDN) environment. OpenFlow is one of the first and most widely used SDN standards; it defines the open communications protocol in an SDN environment that allows the SDN Controller (brains of the network) to speak to the forwarding plane (switches, routers, etc.) to make changes to the network.

The SDN Controller is responsible for maintaining all of the network rules and providing the necessary instructions to the underlying infrastructure on how traffic should be handled. This enables businesses to better adapt to their changing needs and have better control over their networks.

The Floodlight Controller can be advantageous for developers because it offers them the ability to easily adapt software and develop applications and is written in Java. Included are representational state transfer application program interfaces (REST APIs) that make it easier to program interface with the product, and the Floodlight website offers coding examples that aid developers in building the product.

Tested with both physical and virtual OpenFlow-compatible switches, the Floodlight Controller can work in a variety of environments and can coincide with what businesses already have at their disposal. It can also support networks where groups of OpenFlow-compatible switches are connected through traditional, non-OpenFlow switches.

The Floodlight Controller is compatible with OpenStack, a set of software tools that help build and manage cloud computing platforms for both public and private clouds. Floodlight can be run as the network backend for OpenStack using a Neutron plugin that exposes a networking-as-a-service model with a REST API that Floodlight offers.

Floodlight Controller:

The Floodlight Open SDN Controller is an enterprise-class, Apache-licensed, Java- based OpenFlow Controller and intended to run with standard JDK tools and ant.

Highlights:

- Easy to set up with minimal dependencies
- Supports a broad range of virtual and physical OpenFlow switches
- Can handle mixed OpenFlow and non- OpenFlow networks.
- It can manage multiple “islands” of OpenFlow hardware switches
- Designed to be high-performance

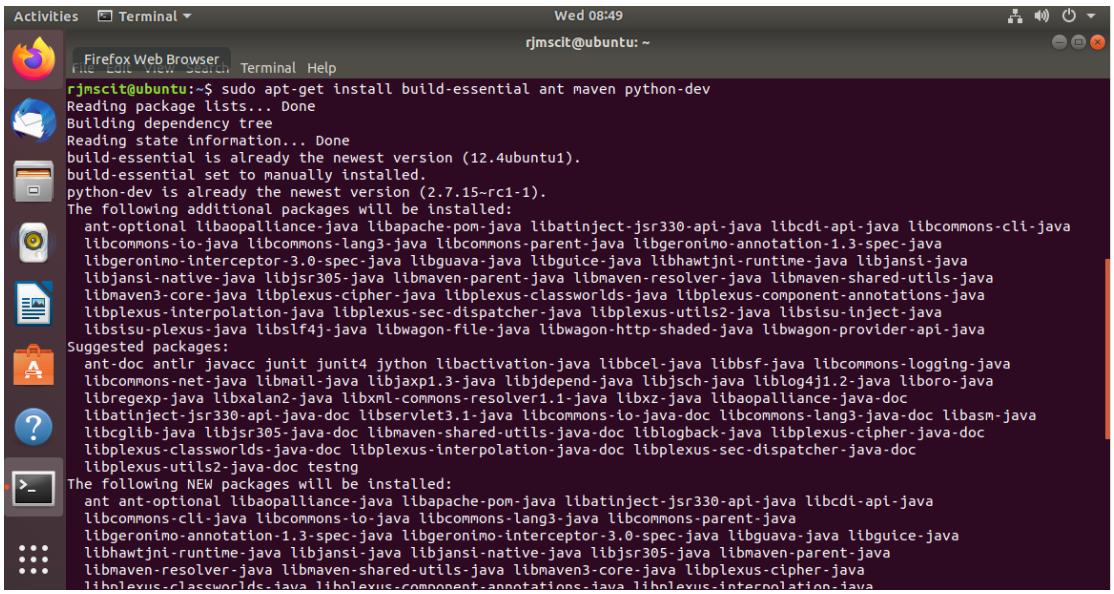
Installation:

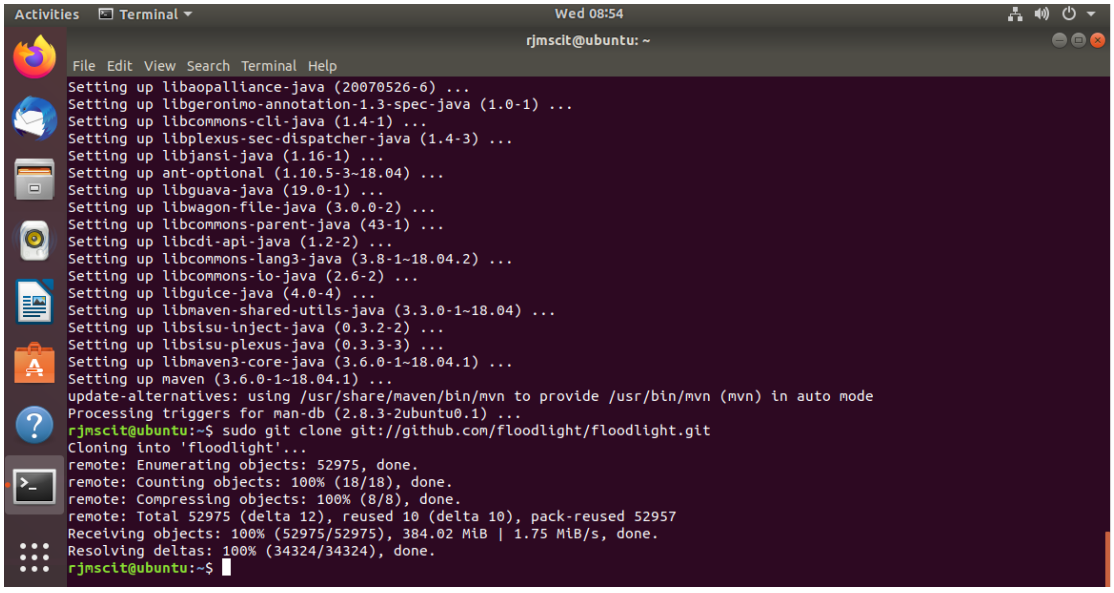
Prerequisites:

- Java development kit
  - JDK 8 for Floodlight master and above
  - JDK 7 for Floodlight v1.2 and below
- Ant to build
- Python development package

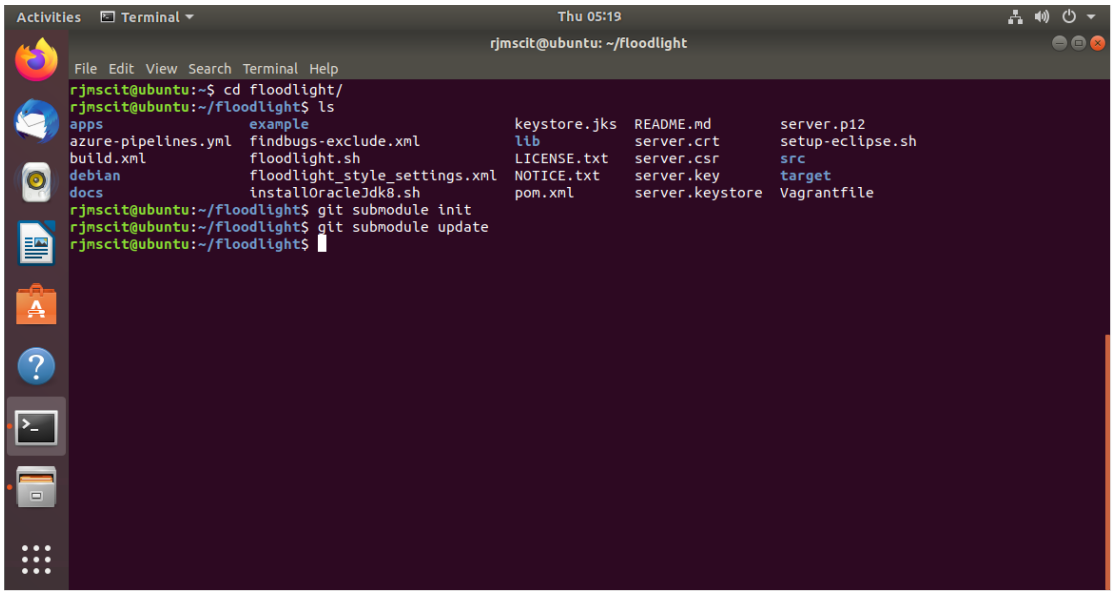
-To download dependencies for Floodlight master and above:

>> sudo apt-get install build-essential ant maven python-dev

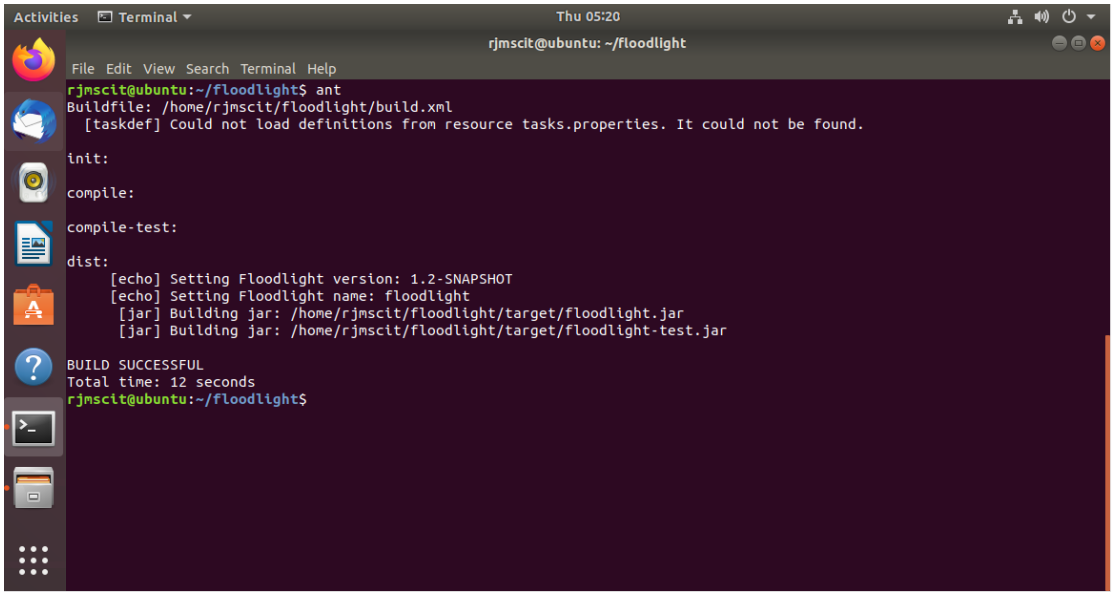




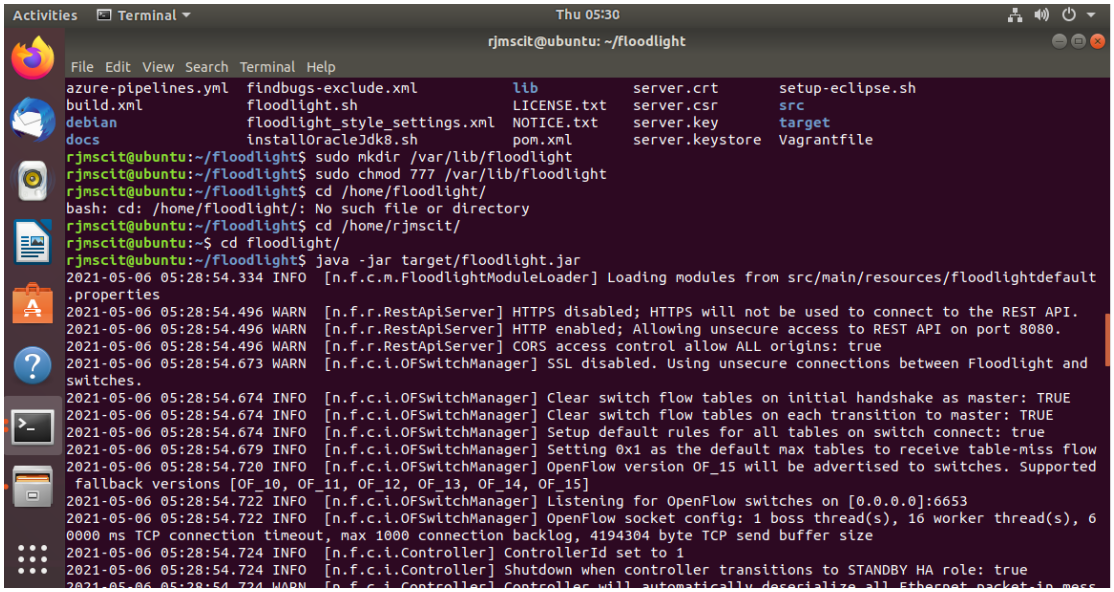
>> cd floodlight
>> git submodule init
>> git submodule update



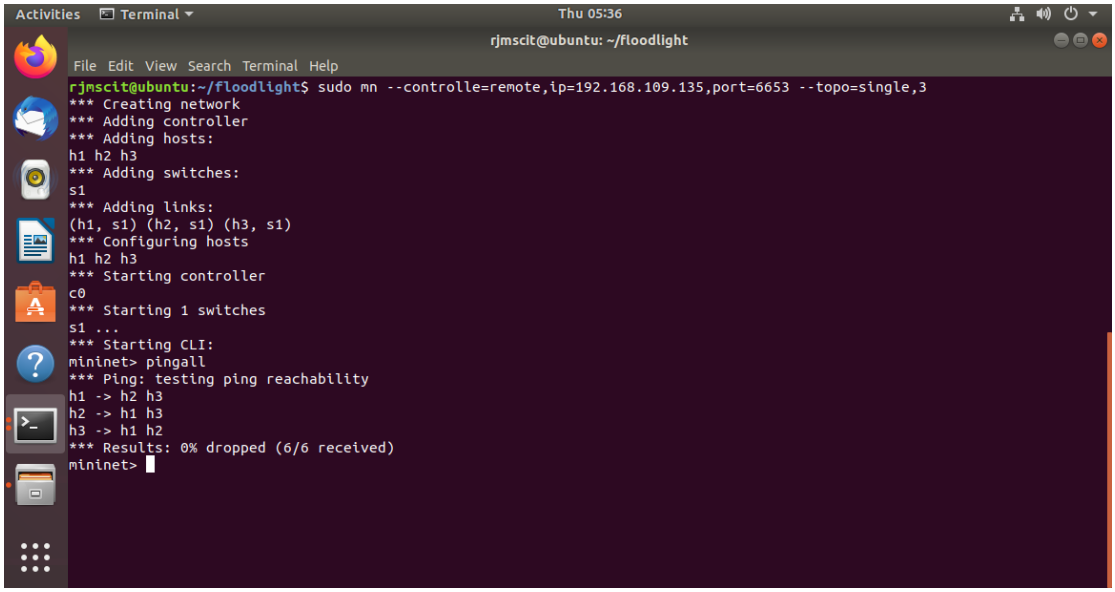
>> ant



>> sudo mkdir /var/lib/floodlight
>> sudo chmod 777 /var/lib/floodlight
>> cd /home/floodlight/
>> java -jar target/floodlight.jar

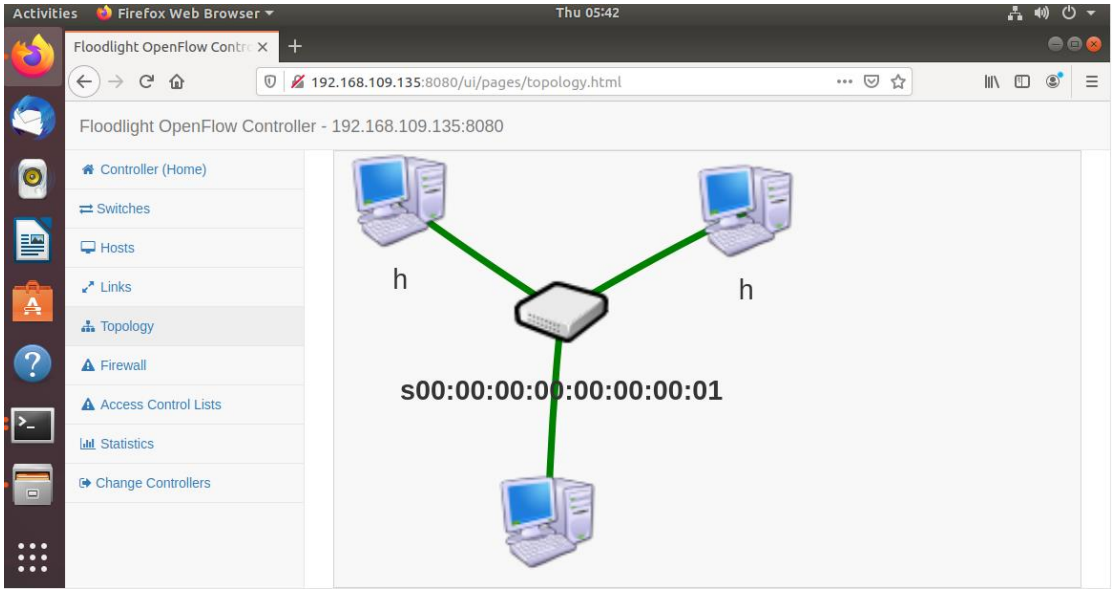


>> sudo mn --controlle=remote,ip=103.242.117.180,port=6653 --topo=single,3



We created a single topology with 3 host 1 switch and one controller. This can be view in the following URL. Go to web browser and type,

>>http:// 192.168.109.135:8080/ui/index.html


Activities
Firefox Web Browser
Thu 05:42
Floodlight OpenFlow Contr
192.168.109.135:8080/ui/pages/topology.html
Floodlight OpenFlow Controller - 192.168.109.135:8080
Controller (Home)
Switches
Hosts
Links
Topology
Firewall
Access Control Lists
Statistics
Change Controllers
h
h
s00:00:00:00:00:00:00:01

PRACTICAL NO: 08

Aim: Install ONOS controller on Ubuntu

Open Network Operating System (ONOS®) is the leading open source SDN controller for building next-generation SDN/NFV solutions.

ONOS was designed to meet the needs of operators wishing to build carrier-grade solutions that leverage the economics of white box merchant silicon hardware while offering the flexibility to create and deploy new dynamic network services with simplified programmatic interfaces. ONOS supports both configuration and real-time control of the network, eliminating the need to run routing and switching control protocols inside the network fabric. By moving intelligence into the ONOS cloud controller, innovation is enabled and end-users can easily create new network applications without the need to alter the dataplane systems.

The ONOS platform includes:

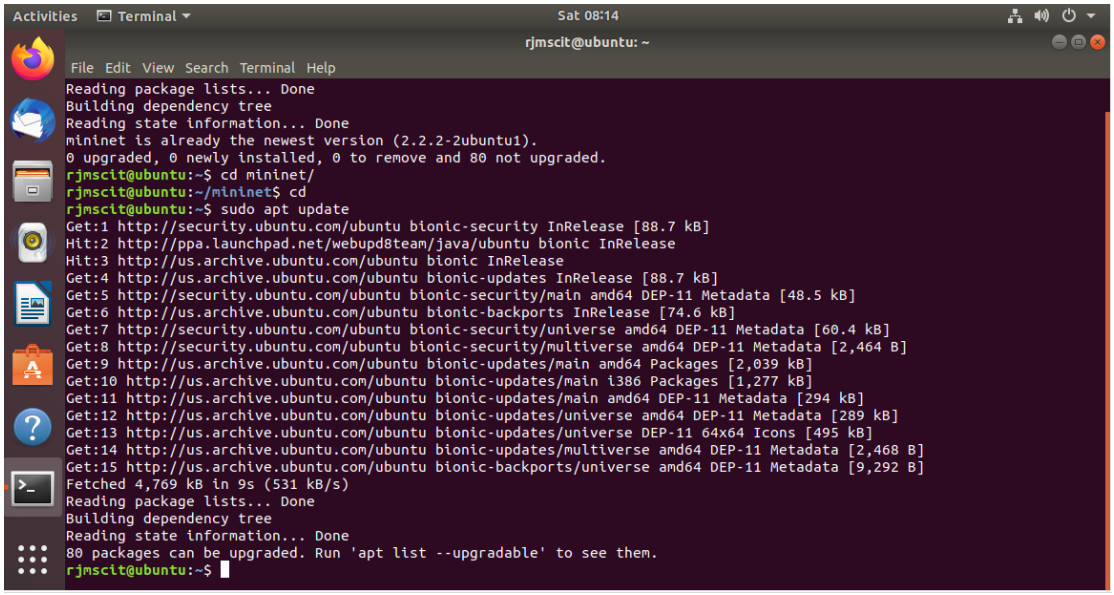
- A platform and a set of applications that act as an extensible, modular, distributed SDN controller.
- Simplified management, configuration and deployment of new software, hardware & services.
- A scale-out architecture to provide the resiliency and scalability required to meet the rigors of production carrier environments.

ONOS (Open Network Operating System) is an operating system (OS) designed to help network service providers build carrier-grade software-defined networks architected for high scalability, availability and performance. Although specifically designed to address the needs of service providers, ONOS can also act as a software-defined networking (SDN) control plane for enterprise campus local area networks (LANs) and data center networks.

Check java version and set it to java 8

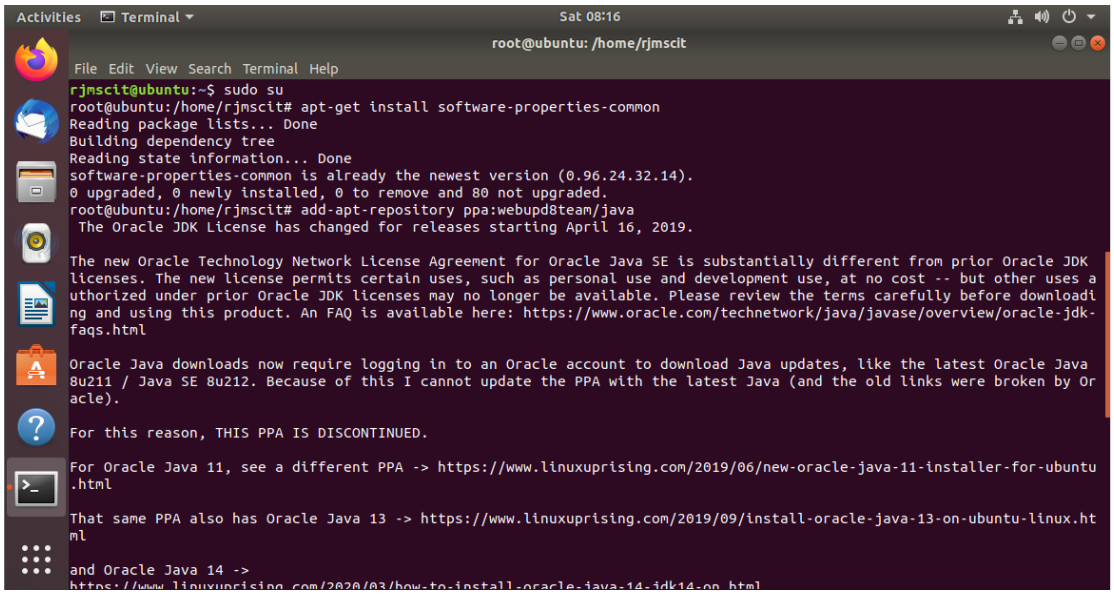
Java -version

>> sudo apt update

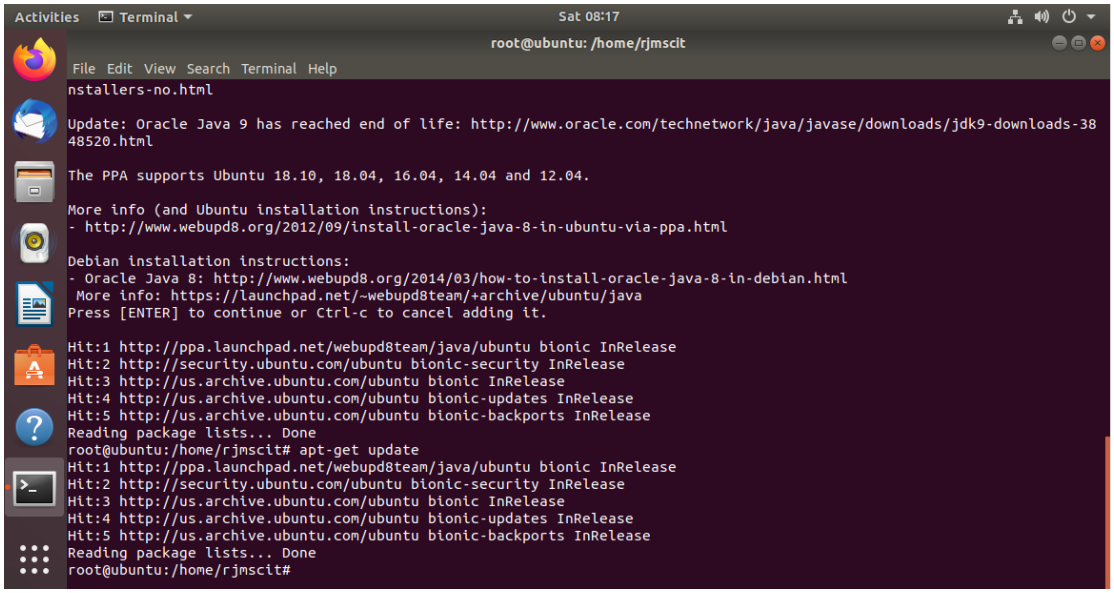


>> apt-get install software-properties-common

>> add-apt-repository ppa:webupd8team/java

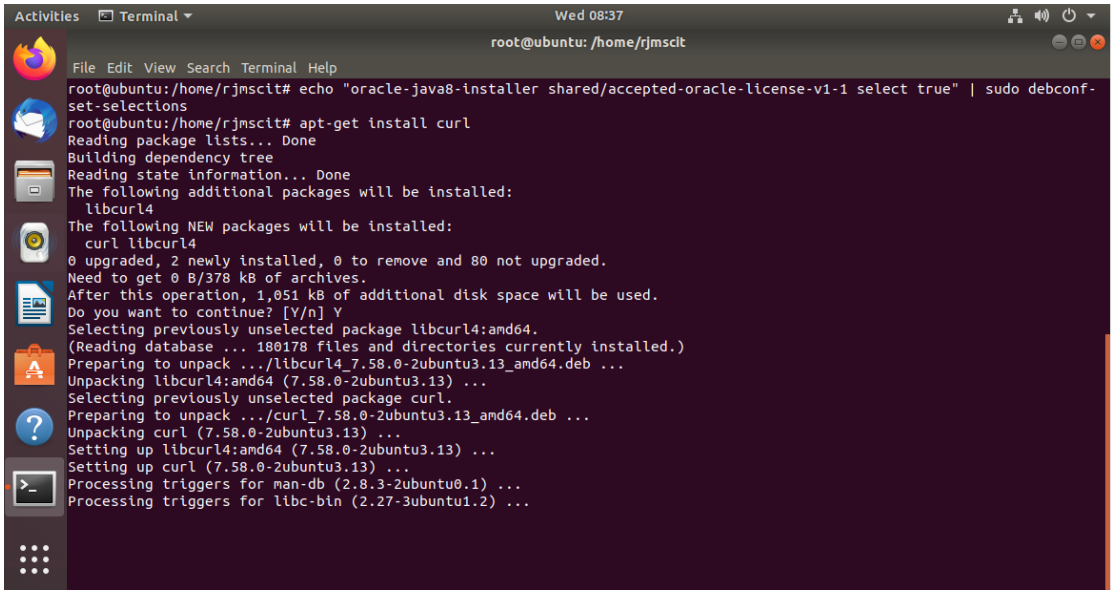


>> apt-get update



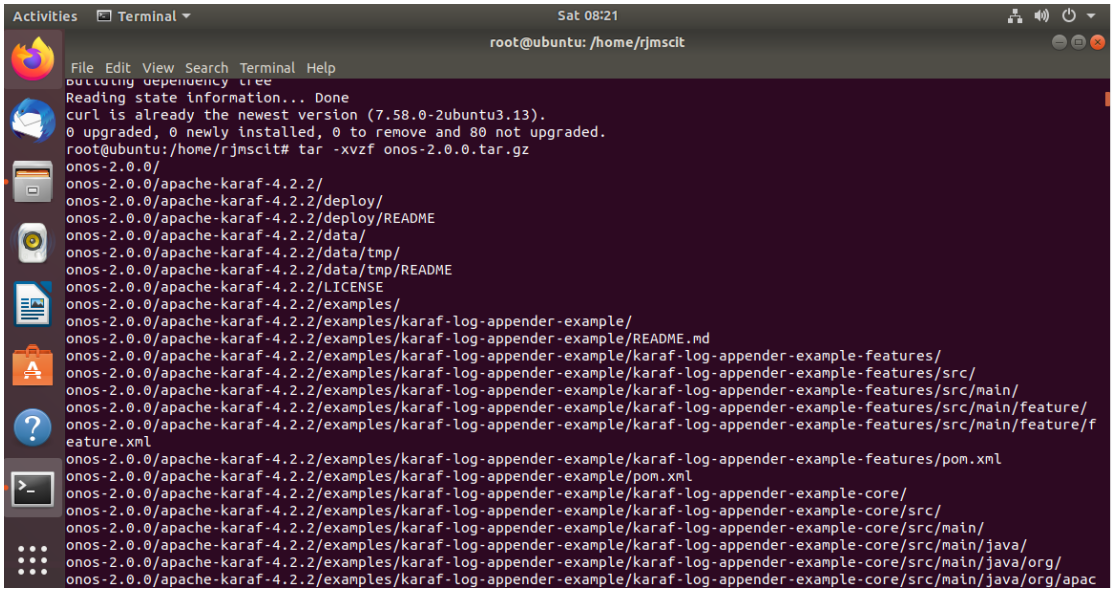
>> echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 select true" | sudo debconf-set-selections

>> apt-get install curl



Move ONOS software setup from Main system to Virtual system and extract from below command:

>> tar -xvzf onos-2.0.0.tar.gz

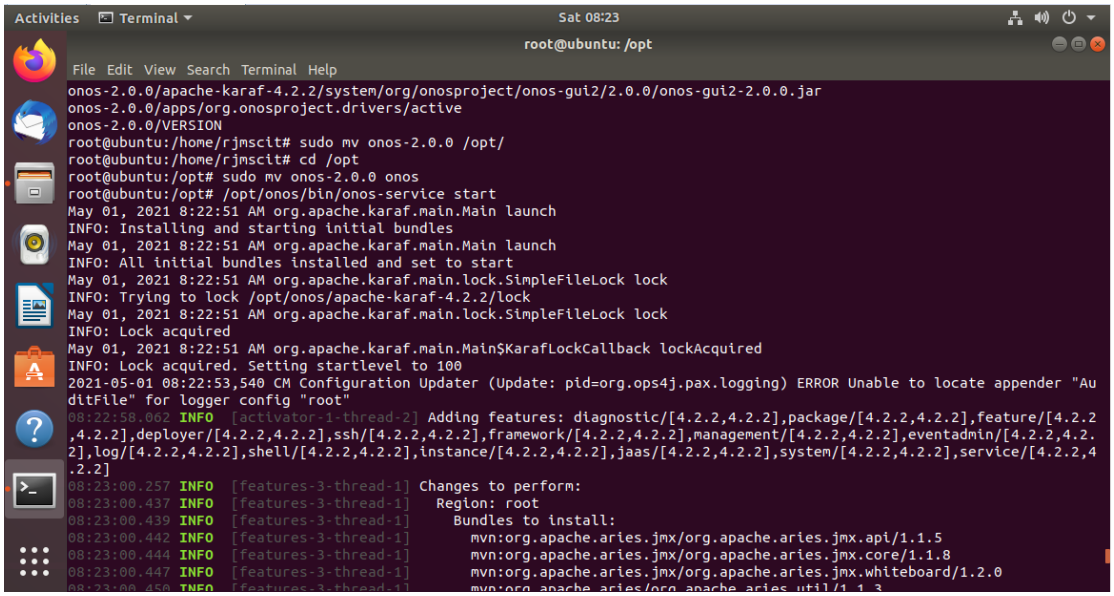


>> sudo mv onos-2.0.0 /opt/

>> cd /opt

>> sudo mv onos-2.0.0 onos

>> /opt/onos/bin/onos-service start



then open mozilla and run below URL to see ouput:

>> 127.0.0.1:8181/onos/ui/index.html

user:onos

pass:rocks