SOFTWARE DEFINED NETWORKING

MSc IT- Part 2 Semester 4

Roll no. 04

Omkar Gunjal

E-Journal

Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE

(AUTONOMOUS)

Opposite Ghatkopar Railway Station, Ghatkopar West, Mumbai-400086

CERTIFICATE

This is to certify that **Mr. Omkar Gunjal** with Seat No.**04** 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

_____ _____

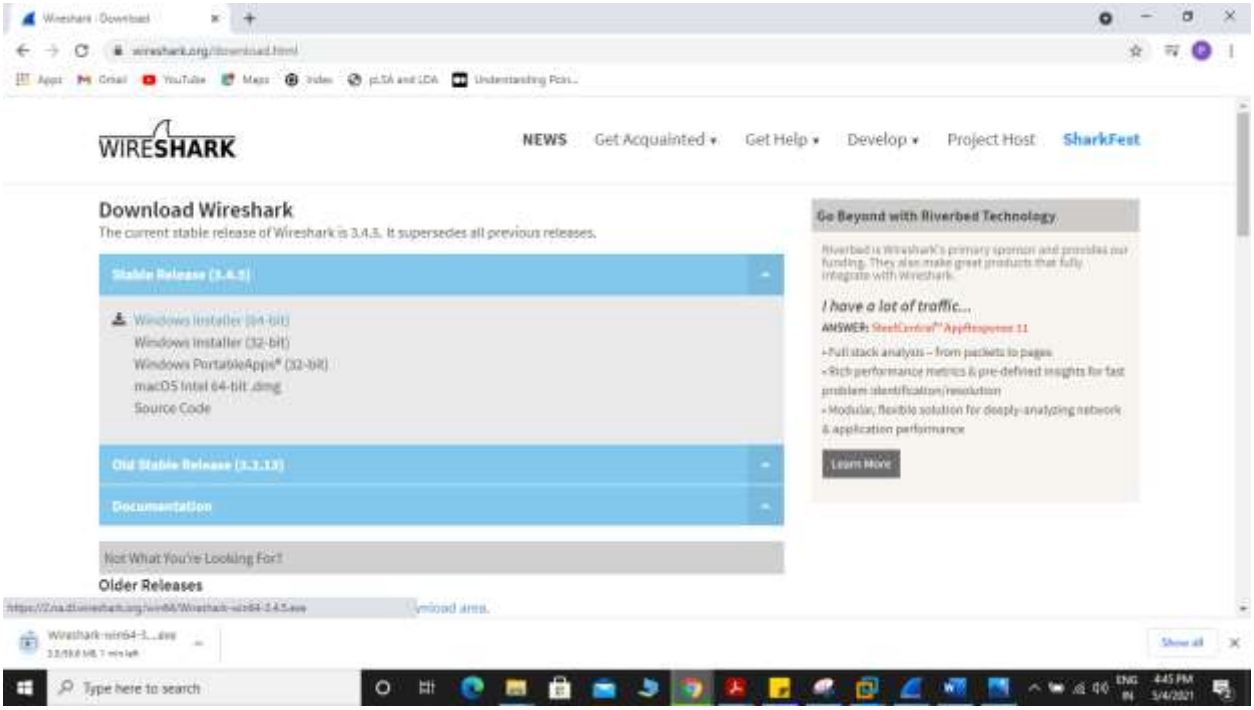
Head of Department College Seal External Examiner

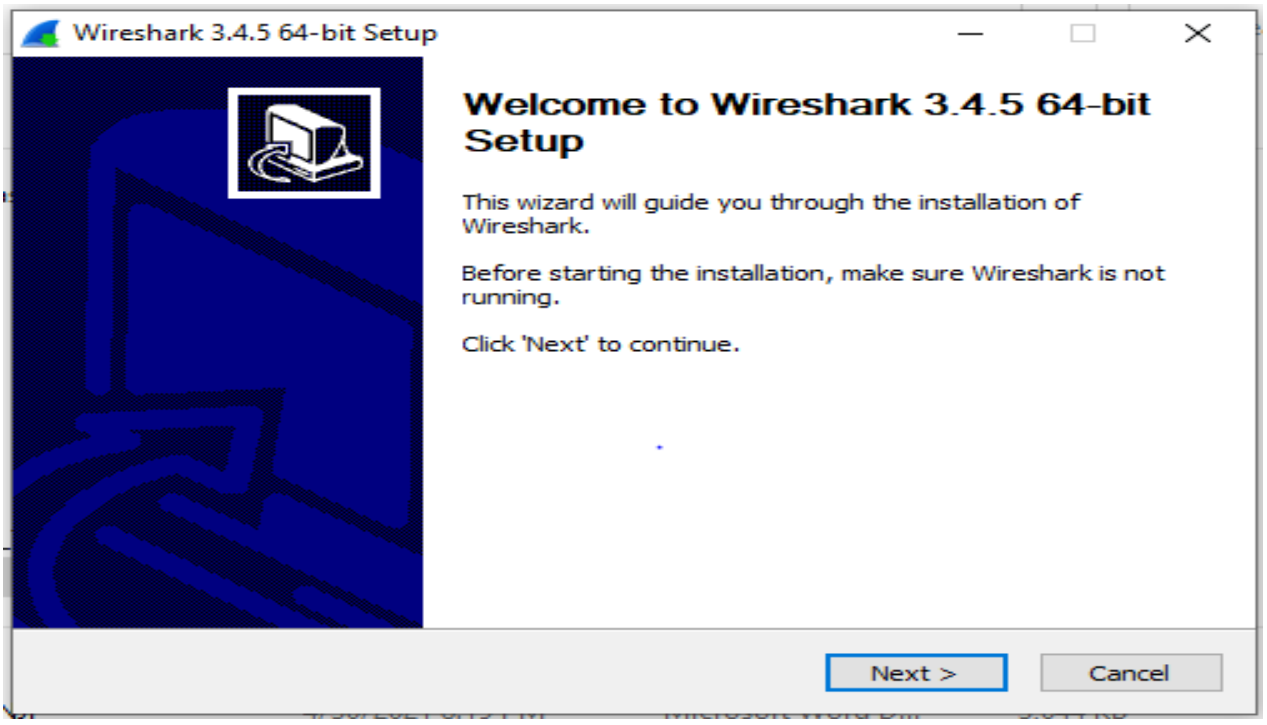
INDEX

PRACTICAL NO: 01

Wireshark and bash Script

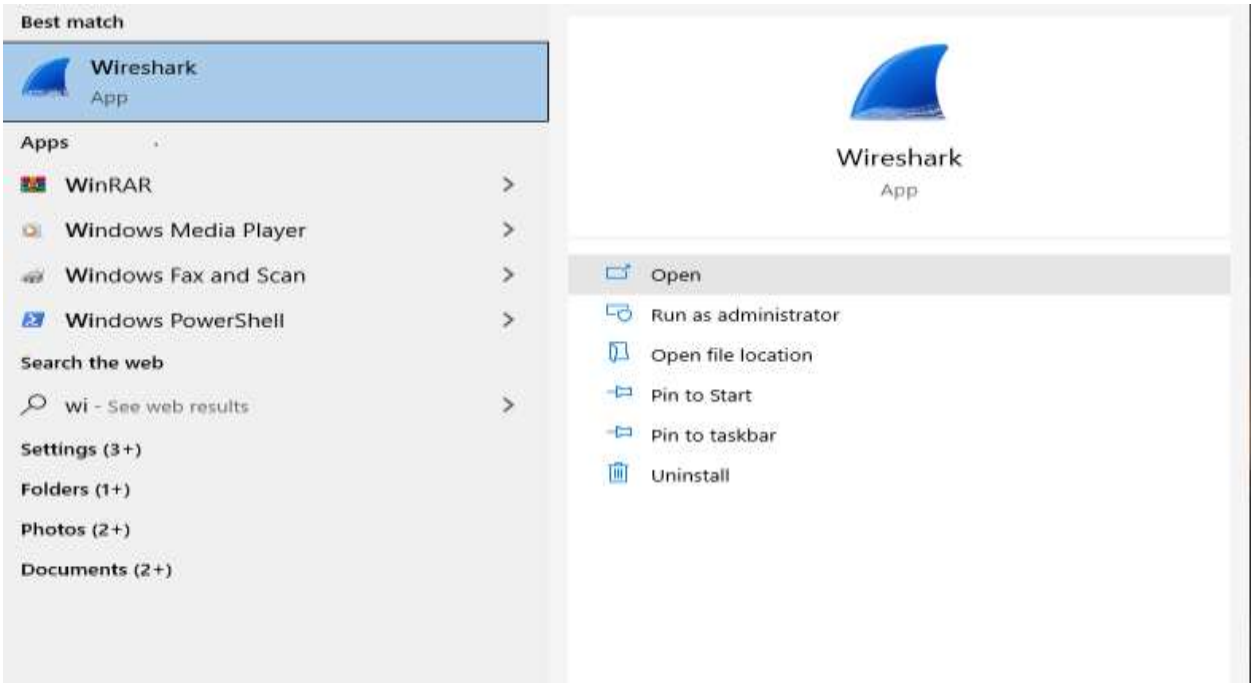
Install Wireshark on Windows 10



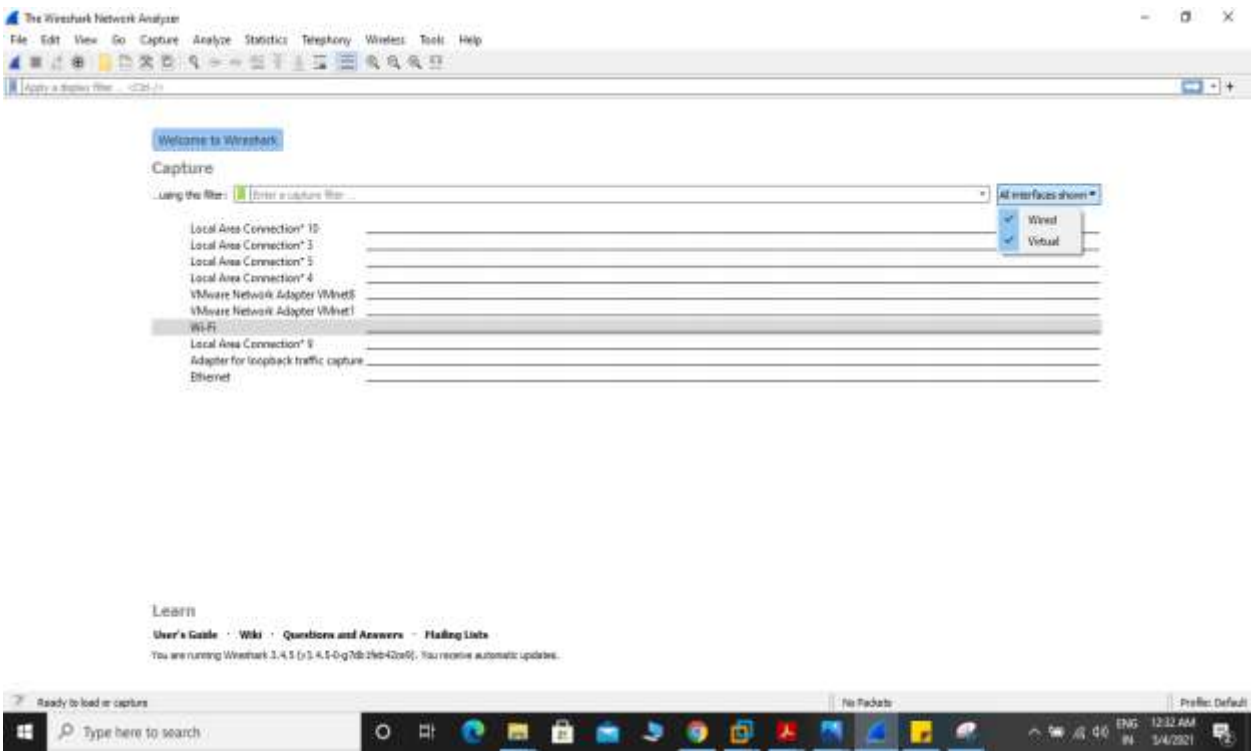


Starting Wireshark

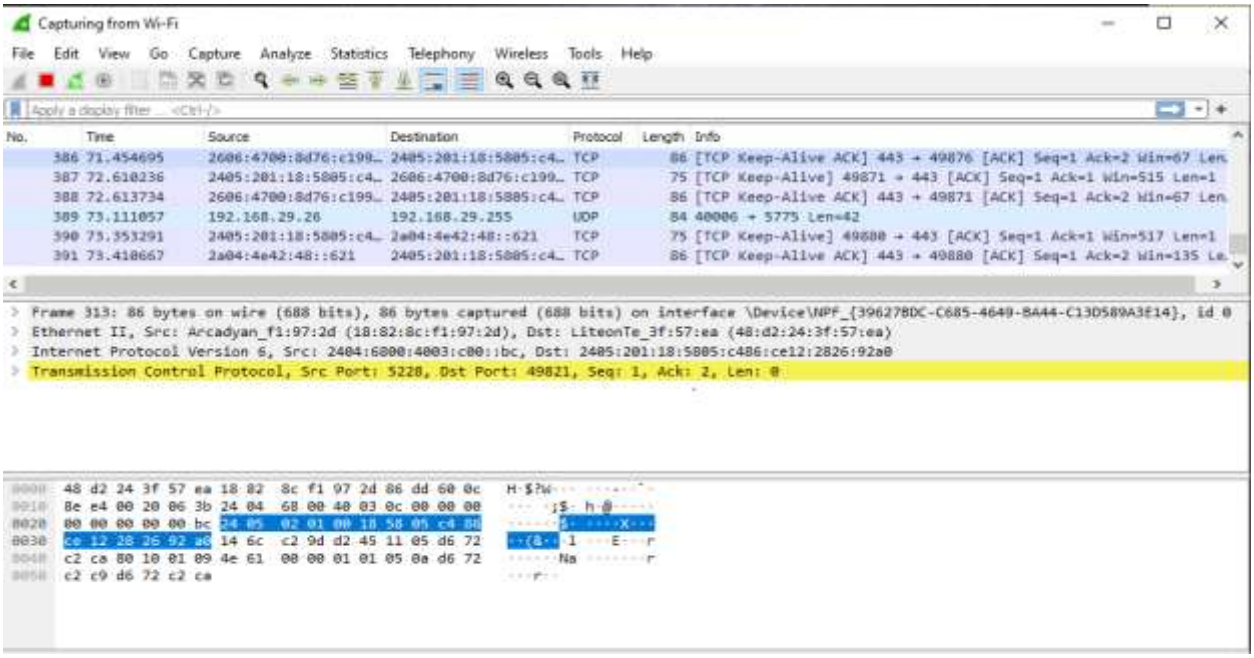
Launching a Wireshark application can be done from the start button.



Now, Open the Wireshark.

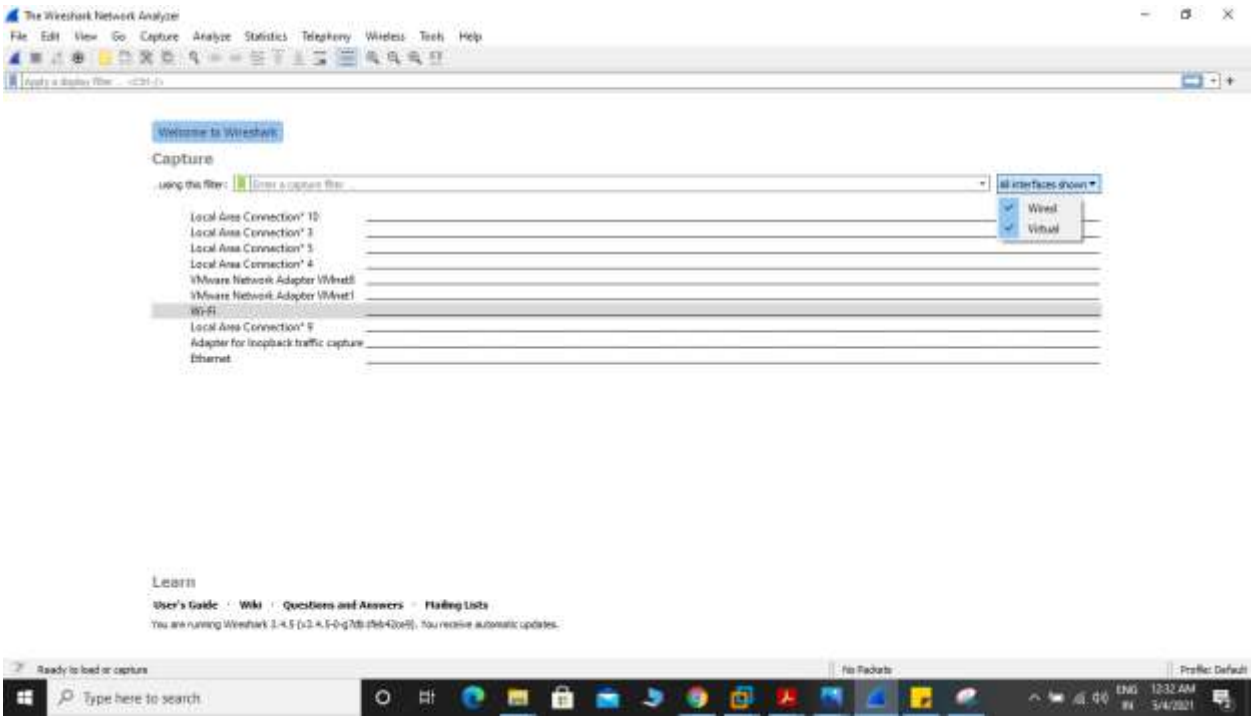


To test packets capturing, select interface to use and click "**Start capturing packets**" button.

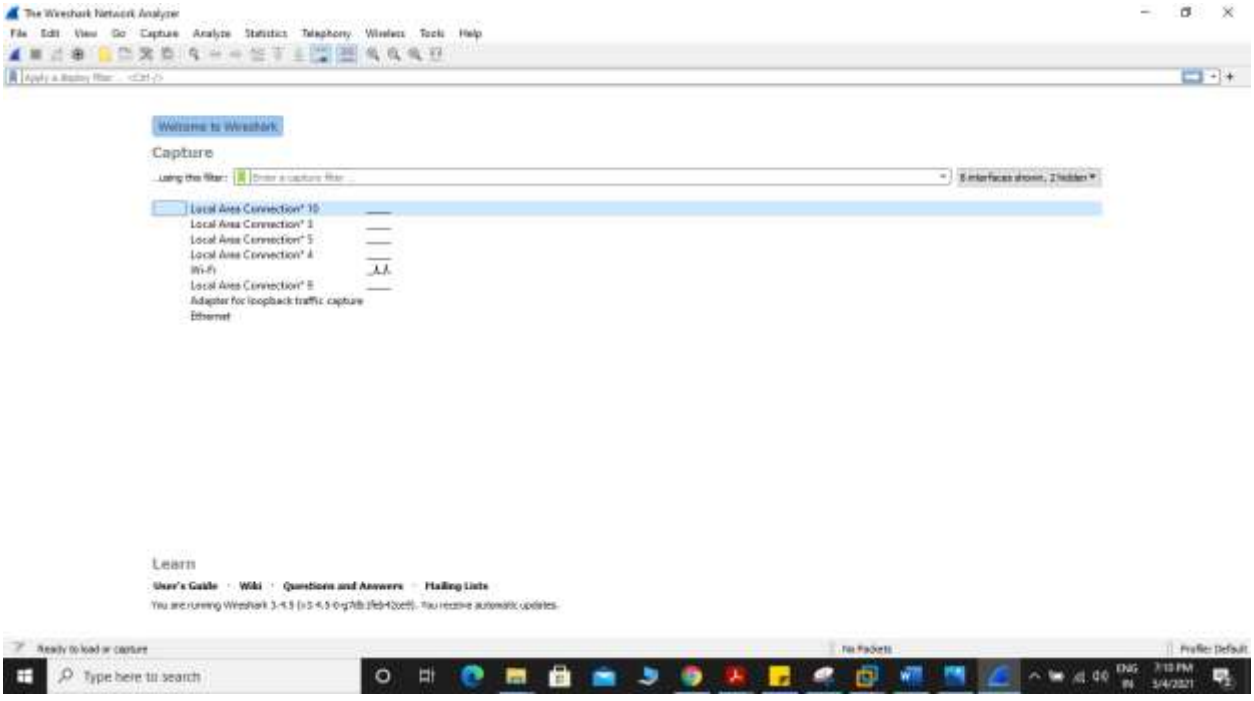


There are many types of interfaces you can monitor using Wireshark, for example, **Wired**, **Wireless**, USB and many external devices. You can choose to show specific

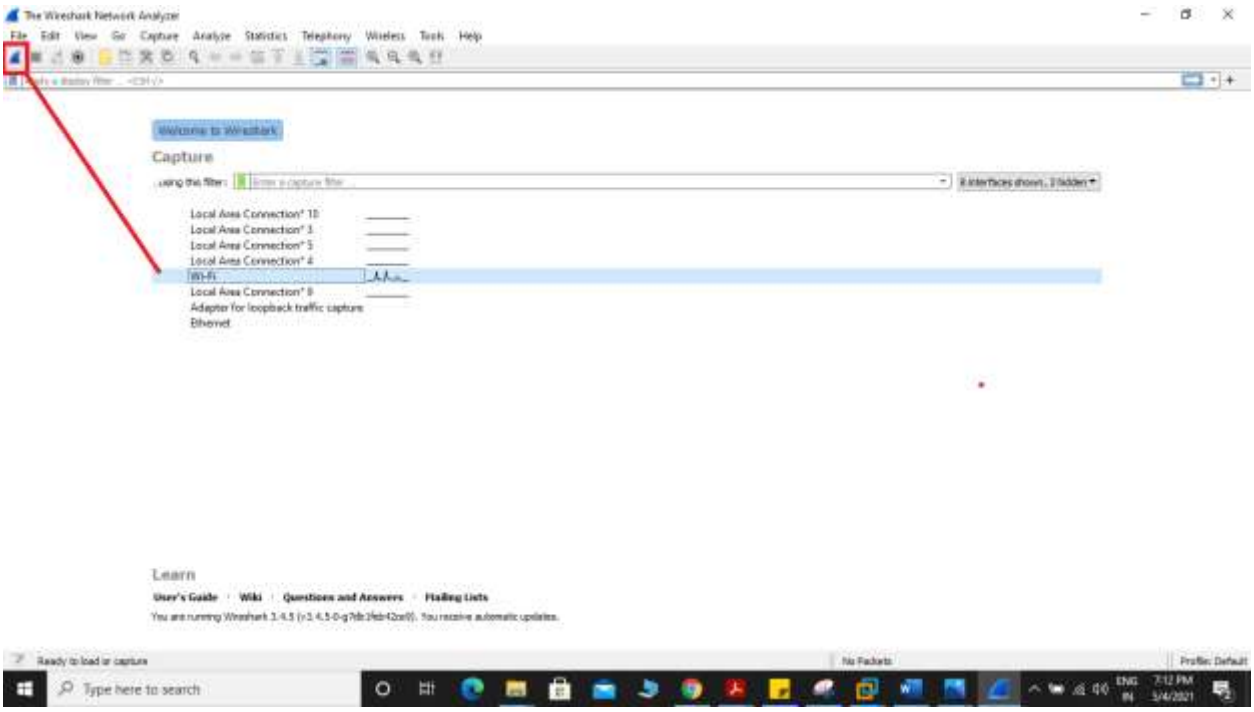
types of interfaces in the welcome screen from the marked section of the screenshot below.



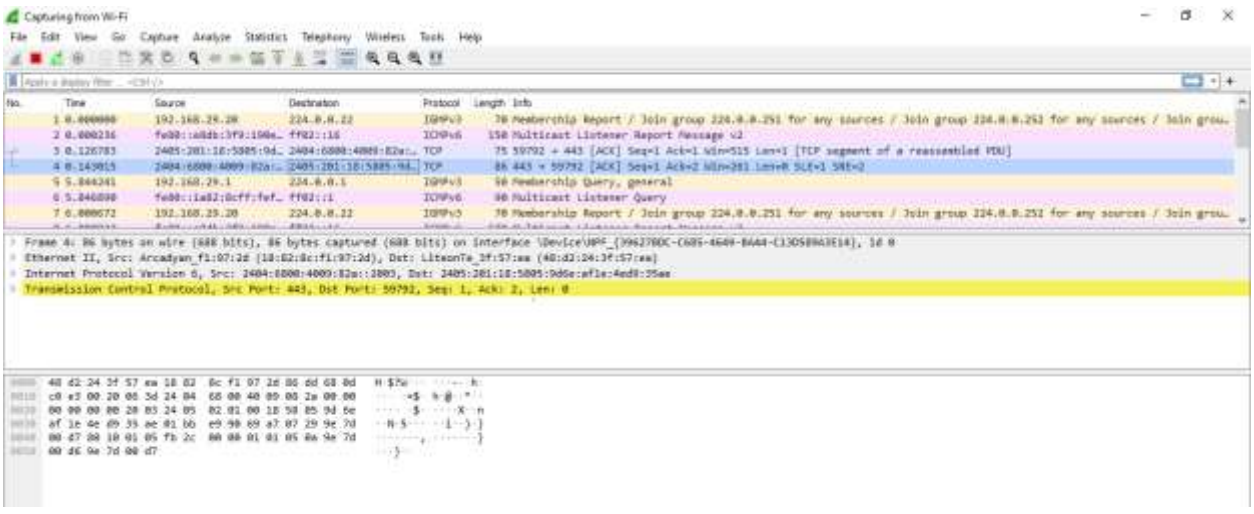
Here, I listed only the **Wired** network interfaces.



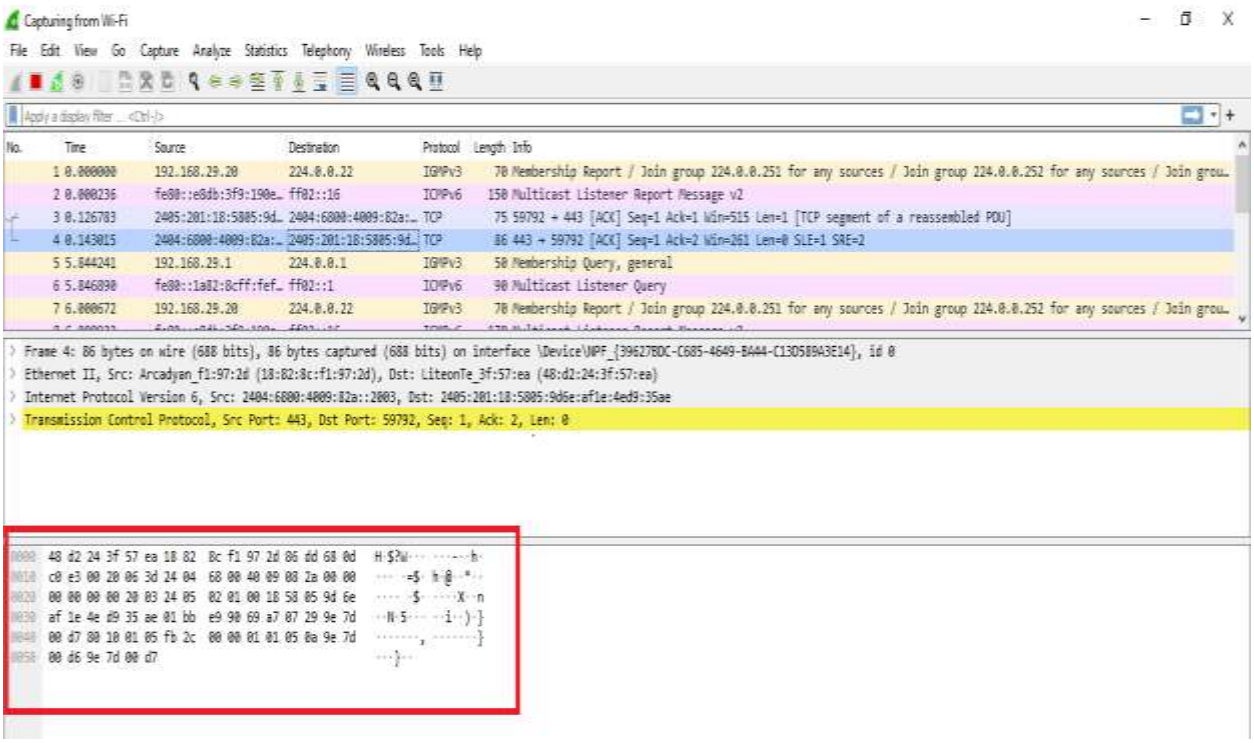
Now to start capturing packets, just select the interface (in my case interface **Wifi**) and click on the **Start capturing packets** icon as marked in the screenshot below. You can also double click on the interface that you want to capture packets to and from to start capturing packets on that particular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



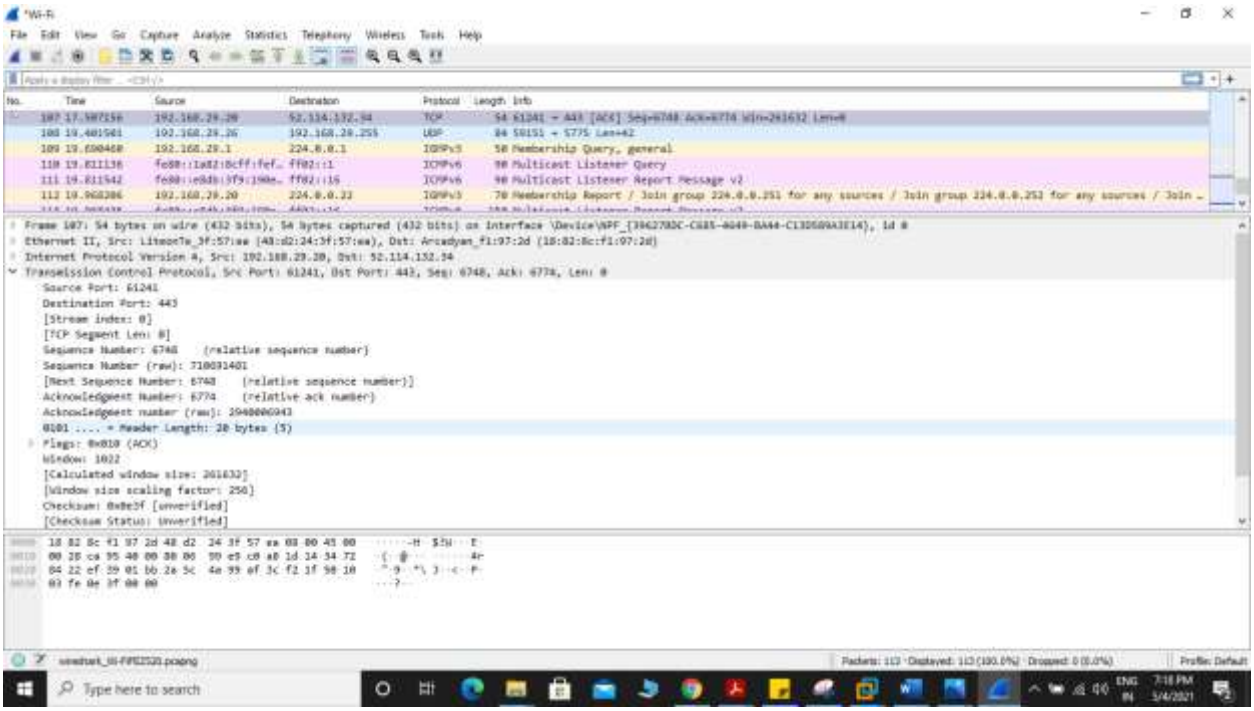
Now you can click on a packet to select it. Selecting a packet would show many information about that packet. As you can see, information about different layers of TCP/IP Protocol is listed.



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



You can also click on the arrows to expand packet data for a particular TCP/IP Protocol Layer.



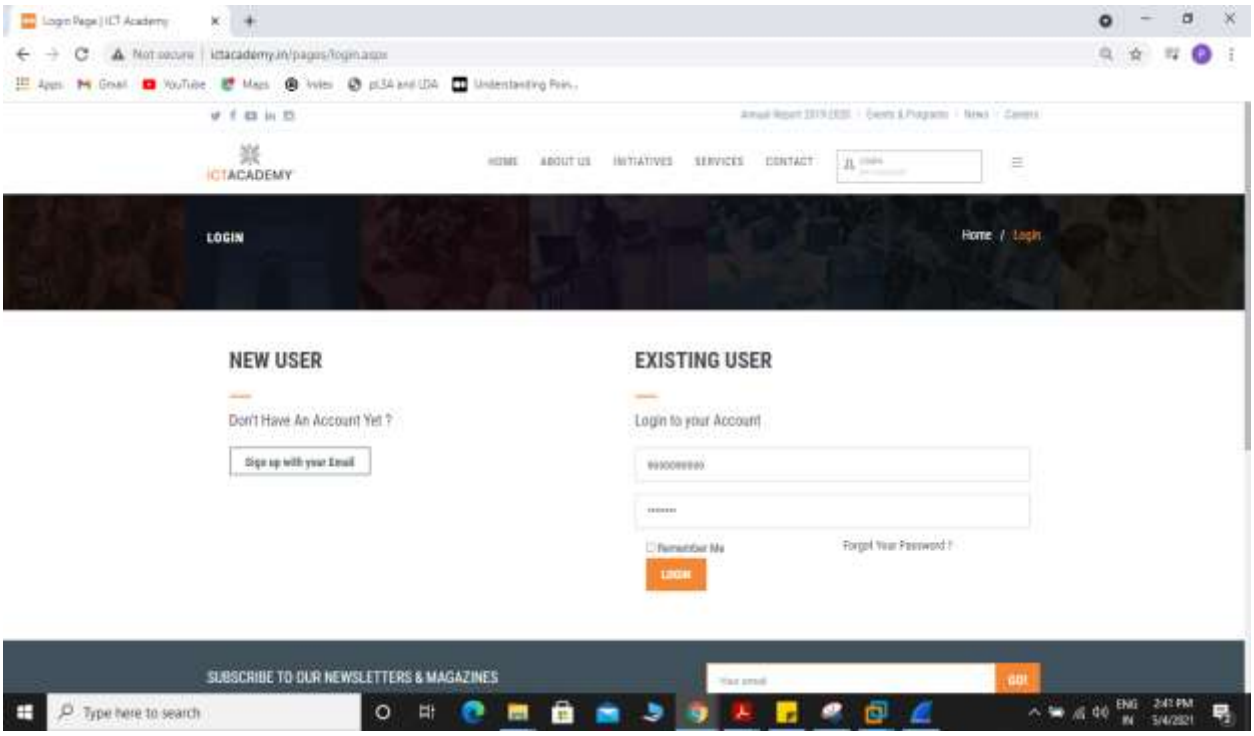
Filtering Packets Using Wireshark:

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.

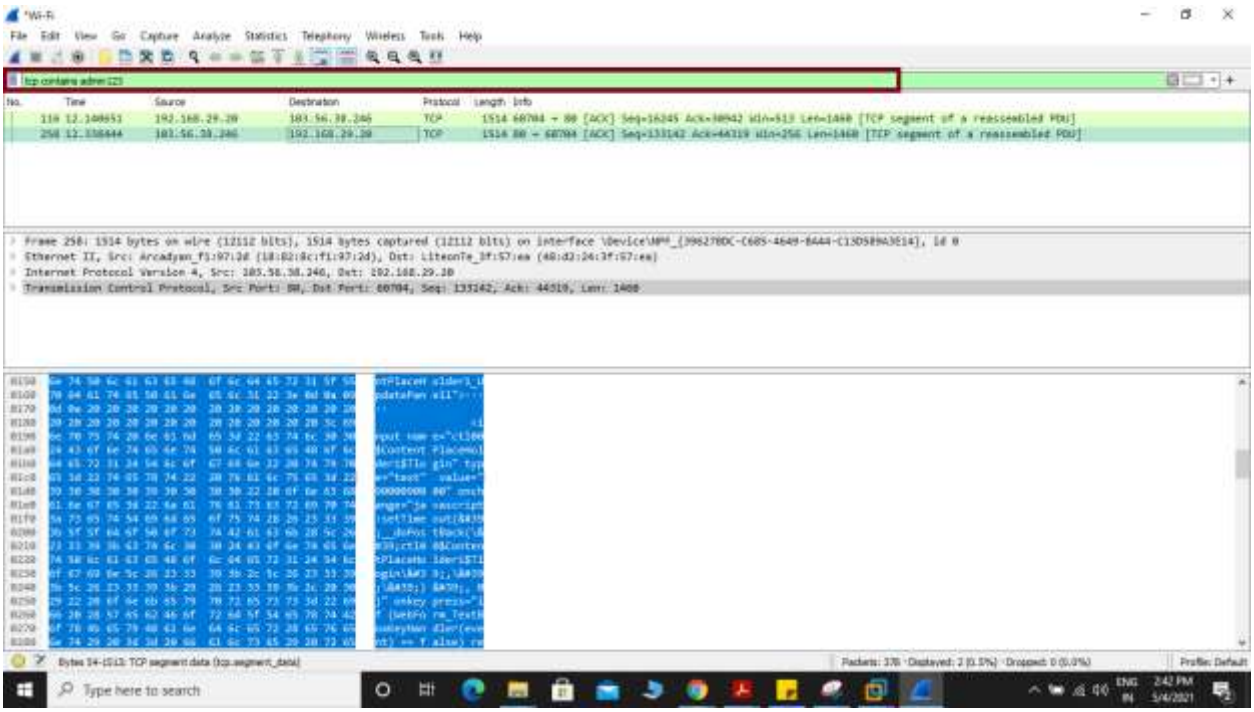
The good thing is, in Wireshark, you can filter the packets and see only the packets that you need.

To Demonstrate lets go on to an unsecured site from google by starting capturing new packets.

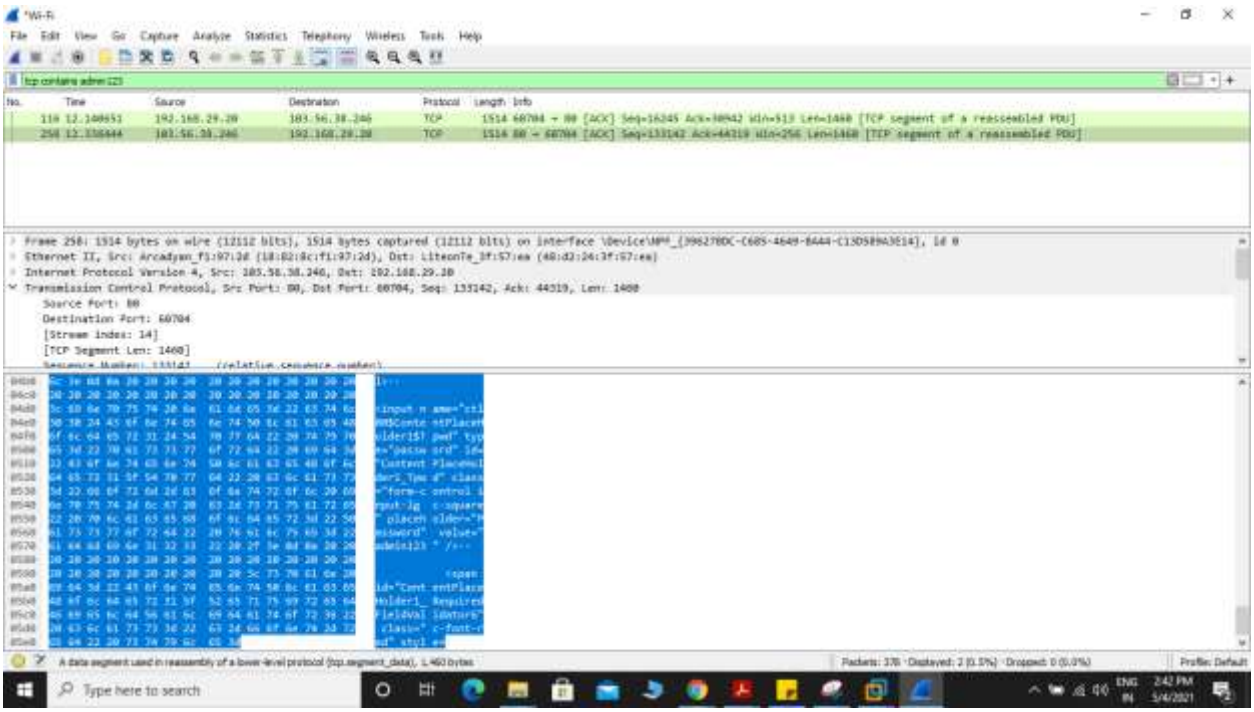
Let's go to http://www.ictacademy.in/pages/login.aspx

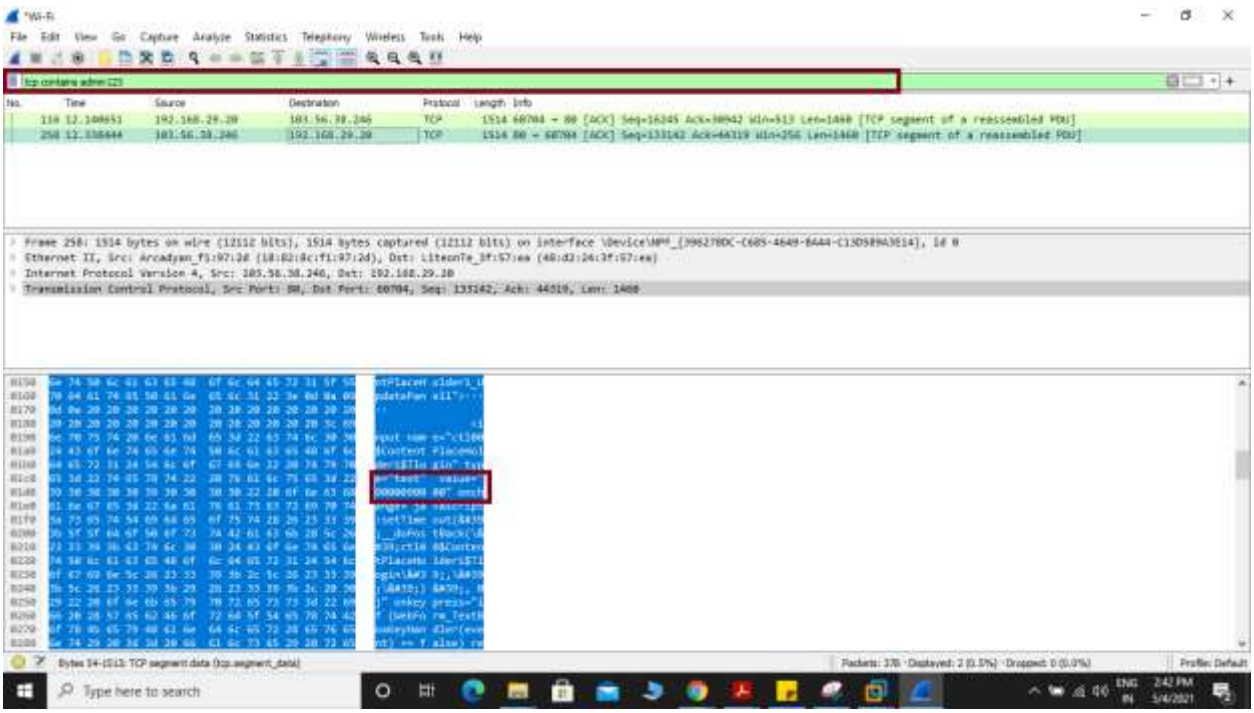


To filter packets, you can directly type in the filter expression in the textbox as marked in the screenshot below.



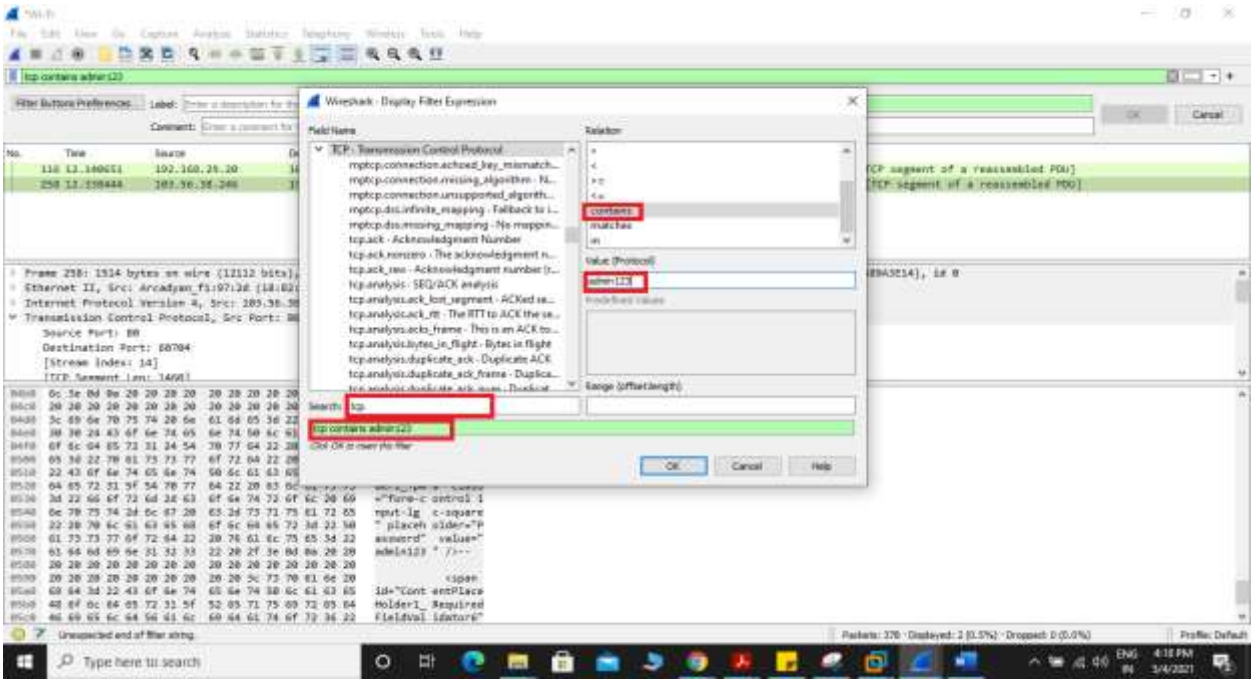
You can also filter packets captured by Wireshark graphically. To do that, click on the **Expression…** button as marked in the screenshot below.



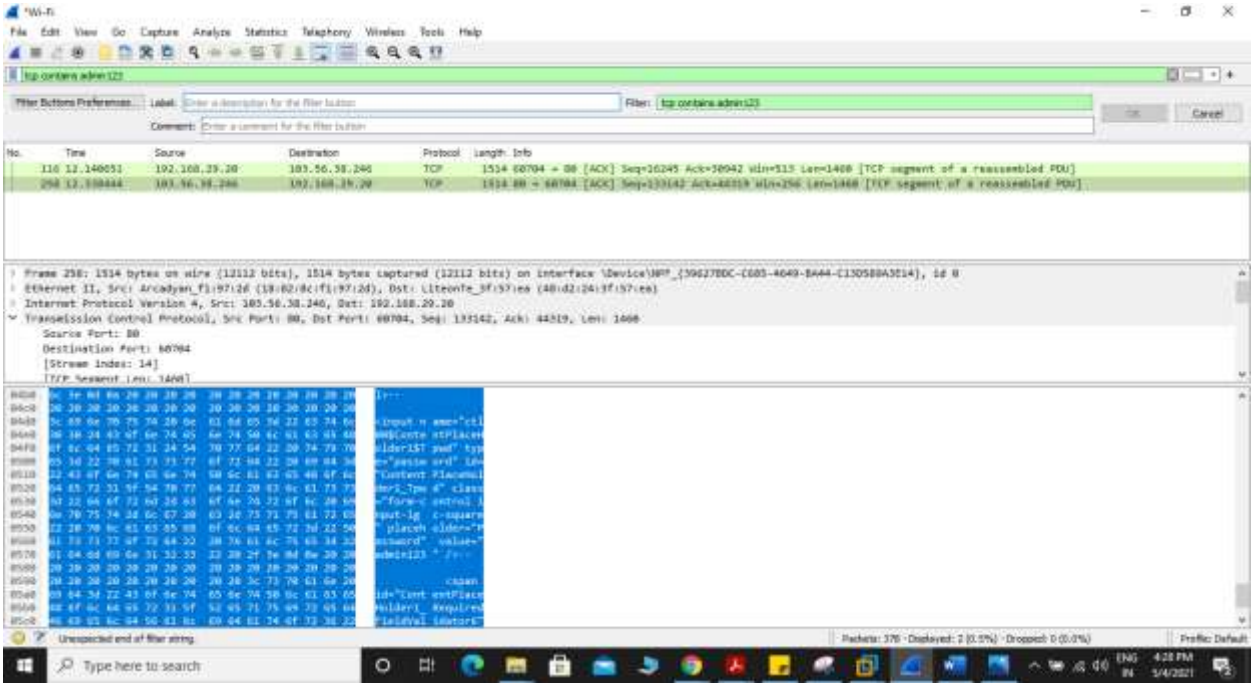


A new window should open as shown in the screenshot below. From here you can create filter expression to search packets very specifically.

In the **Field Name** section almost all the networking protocols are listed. The list is huge. You can type in what protocol you're looking for in the **Search** textbox and the **Field Name** section would show the ones that matched.

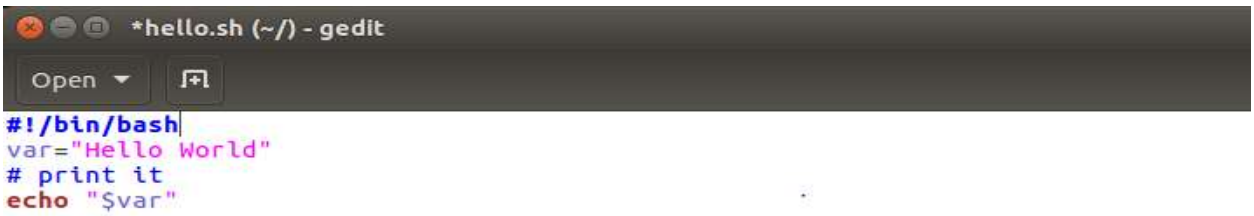


In this article, I am going to filter out all the DNS packets. So I selected **DNS Domain Name System** from the **Field Name** list. You can also click on the **arrow** on any protocol



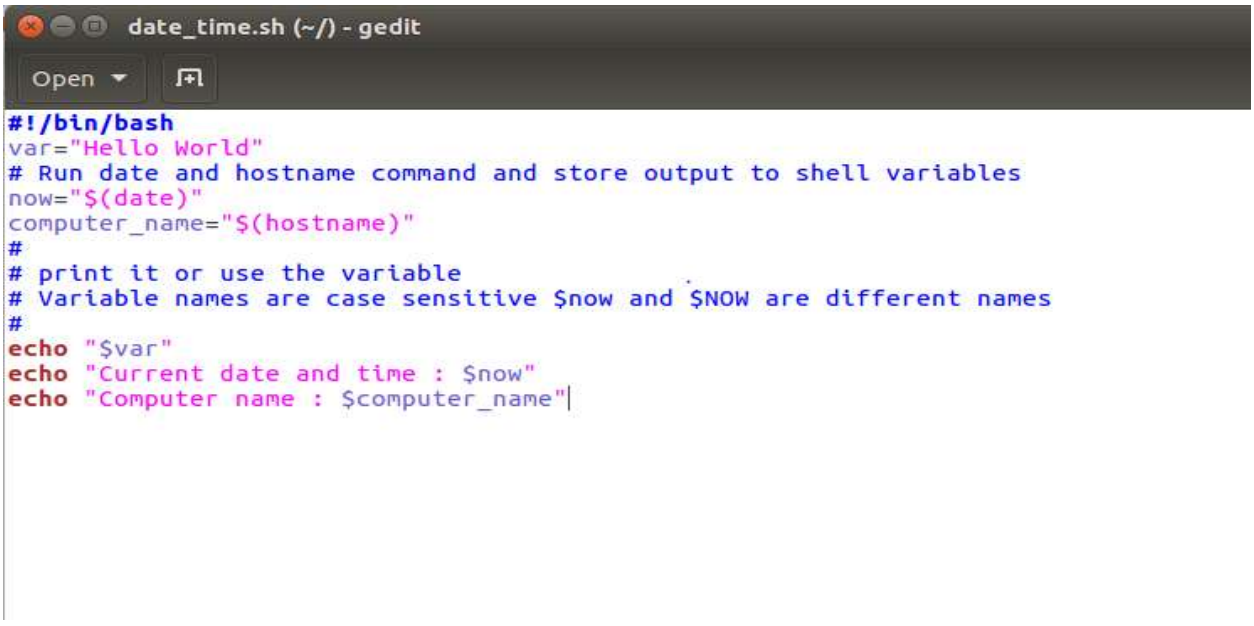
Bash Scripts

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

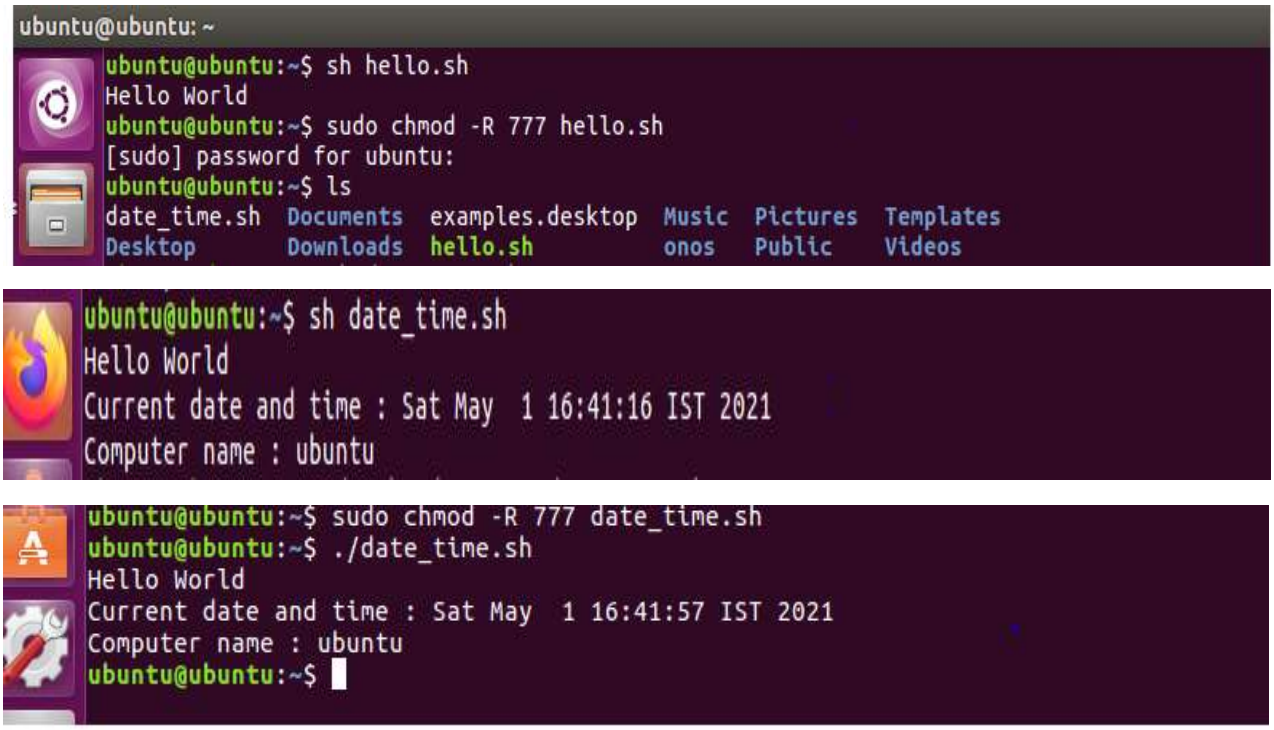


Bash script to print current date and time and Hostname .

```
#!/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
# Variable names are case sensitive $now and $NOW are different names
#
echo "$var"
echo "Current date and time : $now"
echo "Computer name : $computer_name"
```



output of above programs. Following snap shows two ways to run a shell script:



PRACTICAL NO : 02

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

HP Controller

- **The HPE Virtual Application Networks (VAN) SDN Controller provides a unified control point in an SDN-enabled network, simplifying management, provisioning, and orchestration.**
- **This enables delivery of a new generation of application-based network services. It also provides open application programming interfaces (APIs) to allow developers to create innovative solutions to dynamically link business requirements to network infrastructure via either custom Java programs or general-purpose RESTful control interfaces.**
- **The VAN SDN Controller is designed to operate in campus, data center, or service provider environments**
- The HPE VAN SDN Controller provides a unified control point in an OpenFlow-enabled network, simplifying management, provisioning, and orchestration and enabling delivery of a new generation of application-based network services.
- In the Hewlett Packard Enterprise Software Defined Networking (SDN) architecture, the control and data planes of the network are decoupled from each other, centralizing network intelligence and abstracting the underlying network infrastructure from applications.
- Controller software directly provisions physical and virtual switches under its control via the industry-standard OpenFlow protocol. Network ports, links, and topologies are all directly visible, enabling centralized policy administration and more effective path selection based on a dynamic, global view of the network.
- This dramatically simplifies the orchestration of multi-tenant environments and the enforcement of network policy for both mobile clients and servers. The HPE VAN SDN Controller is designed to operate in a variety of computing environments, including campus, data center, service provider, private cloud, and public cloud.

The HP SDN controller delivers:

- Open programmable interfaces. Our SDN controller delivers tight integration between the network and business system, with open,

- programmable interfaces that enable the orchestration of applications and automation of network functions.
- Your developers can use the language of their choice and leverage the extensible RESTful API for the creation of SDN applications that will unleash new levels of innovation.
- Centralized, resilient control. Our controller provides centralized, resilient control of the SDN network, including functions such as discovery of the network topology and shortest path forwarding through the network.
- Highly available and scalable. We designed our SDN controller to meet high availability and scaling requirements through a scale-out teaming model. Our SDN controller can be clustered, so that if any one controller in the network fails, another in the cluster will take over.
- Robust security. Security is an important factor of our controller. It uses robust authentication and authorization methods so that SDN applications can interact with the controller while preventing unauthorized applications from gaining network access. The southbound connections between the OpenFlow switches and the HP SDN controller are also secured and encrypted.
- Full Integration with HP Intelligent Management Center (IMC). Administrators can use the HP Intelligent Management Center (IMC) SDN Manager for full fault, configuration, accounting, performance, and security management for the HP SDN controller and OpenFlow infrastructure.
- HP SDN Manager leverages flow monitoring, topology mapping, and troubleshooting to provide full SDN management through the same interface as the wired, physical, and virtual network. IMC provides full controller application life cycle management and monitoring, enhanced reporting, and SDN network visualization.

Product features

Proactive flow processing :

Enables highly scalable, centrally orchestrated SDN networks

Reactive flow processing :

Enables dynamic monitoring of new flows or endpoints

Graphical user interface (GUI) :

Facilitates controller administration and API documentation

Northbound APIs :

Leverage the controller's extensible RESTful HTTPS interface; provide an abstract representation of the underlying OpenFlow network and allow external applications running above the controller to exert deliberative, business-level control over the network

Provide the services necessary to support a full management platform such as HPE Intelligent Management Center (IMC)

Native APIs :

Allow Java applications to run within the controller as a collection of OSGi bundles that enable high performance event and packet processing

Scale-out architecture :

Uses scalable, resilient database frameworks, allowing expansion beyond a single standalone controller to a high-availability cluster; based on open-source in-memory database systems, including Hazelcast (for strict consistency), Cassandra (for eventual consistency), and a PostgreSQL relational database, persistent data can be shared among multiple controllers to deliver a scale-out approach to the control of large or demanding networks.

High availability :

Provides a "2n+1" active consistency model, which allows three controllers to manage individual subsets of the network while sharing a common network view; the failure of one control component generates a rapid response by the cluster to provide continued network operations.

Controller security :

Delivers security at multiple levels; HTTPS is used for the REST API, and the authentication of users and applications is performed by way of the Keystone identity service; controller-to-switch communications are secured through the Transport Layer Security (TLS) encryption protocol, as specified in the OpenFlow standard.

Link service module :

Utilizes LLDP messages to discover physical links between switches in the

control domain and monitors port state changes and notifies applications of link event changes; it is also able to identify multi-hop links where non-OpenFlow devices separate controlled network segments.

Topology service module :

Creates a network graph based on information from the link service; identifies ports, computes the shortest path between nodes, and creates a broadcast tree, avoiding network loops; in future releases, this module will enable multi-path services.

Node manager service module :

Monitors ARP, DHCP, and IP packets from edge ports; allows the module to provide a cache of MAC and IP addresses for each end point, which provides identification of devices or users attached to the network.

Path service module :

Utilizes information from the node and topology services to program an end-to-end unidirectional L2 path through the control domain for new network flows; drops unknown source addresses and supports flooding for unknown destinations; can be disabled for normal packet processing, or can be replaced with a more sophisticated program as desired by the application programmer.

Path diagnostic service :

Validates network paths and generates protocol-specific test packets (ICMP, DHCP, UDP, TCP, etc.) that can be inserted into the network and observed at various switches along the path; this provides network administrators with trace-route functionality.

OpenFlow control interface :

Uses a generalized approach in the controller's southbound interface to processing OpenFlow 1.0 and 1.3 messages; this provides an efficient and intuitive mechanism for monitoring and programming various network components and for processing new flow messages; packets are translated into a set of rich data types by the controller, which allows Java applications to easily

consume or create messages or packets. Netconf and SNMP southbound drivers are also available.

Flexible packet processing :

Enables a hybrid of both OpenFlow and normal packet processing with the HP SDN architecture; access control lists can be provisioned centrally, for example, while L2 or L3 forwarding decisions can be made using standard network protocols; this allows SDN concepts to be applied incrementally to the network, starting with the application of network policy and extending to exception-based forwarding, adding value without replacing traditional switching or routing.

- An enterprise-class platform for the delivery of a broad range of network innovations
- An extensible, scalable, and resilient controller architecture
- Compliance with OpenFlow 1.0 and 1.3 protocols
- Support for Hewlett Packard Enterprise and H3C OpenFlow-enabled switches
- Secure authentication using a local or remote Keystone server
- Controller teaming for distributed platform High Availability (HA) and scalability
- Embedded applications that provide common network services
- Open APIs to enable third-party SDN application developers to deliver innovative solutions to dynamically link business requirements to network infrastructure using either custom Java programs or general-purpose RESTful control interfaces, including functions to extend the controller REST API and UI.
- Integration with HPE Intelligent Management Center (IMC). HPE IMC provides full controller application life cycle management and monitoring, enhanced reporting and SDN network visualization

About the controller embedded applications

List of controller embedded applications

The HPE VAN SDN Controller includes a default set of core network service applications that are installed as modules on the controller. The following applications are embedded in the controller and are installed when you install the controller:

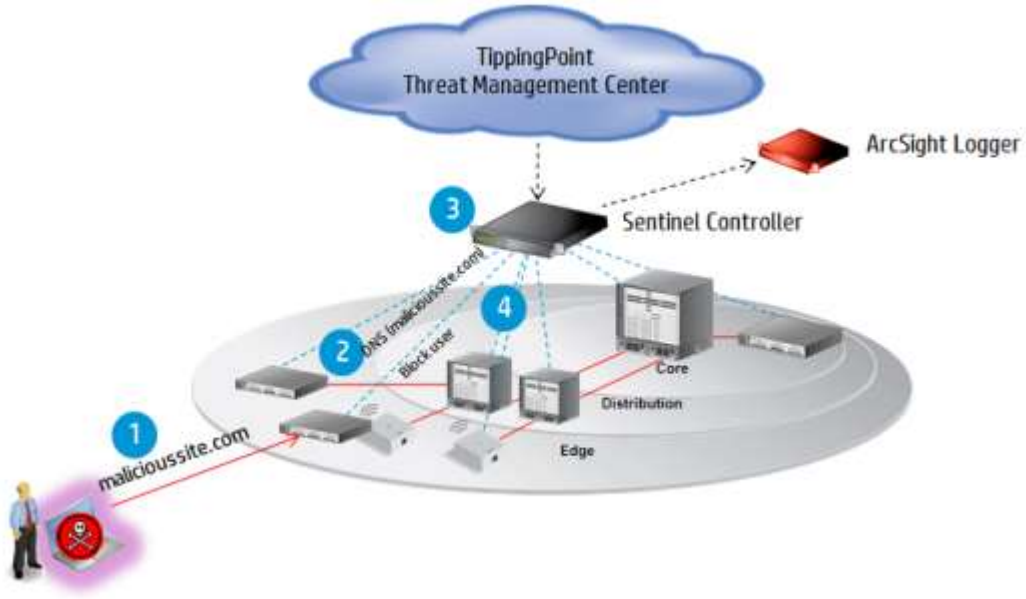
• OpenFlow Link Discovery

- OpenFlow Node Discovery
    - Path Daemon
- Path Diagnostics
- Topology Manager
- Topology Viewer

HP Network Protector

Network Protector – SDN App By HP

The **Network Protector SDN Application** enables automated network posture assessment and real-time security across SDN-enabled network, provides simple security for Bring Your Own Device (BYOD). This turns the entire network infrastructure into security-enforcement devices, providing visibility and threat protection against more than one million malicious bot-nets, malware and spyware sites.



Security for Bring Your Own Device (BYOD)

- Brings a new level of threat visibility automation and control to organizations that support bringing your own device (BYOD) for network connectivity.

- Scales up to thousands of endpoints supporting enterprise organizations.
- Decreases the time IT spends on security problems, from days or weeks to hours.

Enables Automated Network-posture Assessment

- Improves your network visibility and accuracy.
- Prioritizes specific DNS traffic (e.g. business critical) and restricts non-critical DNS traffic (e.g. social media).

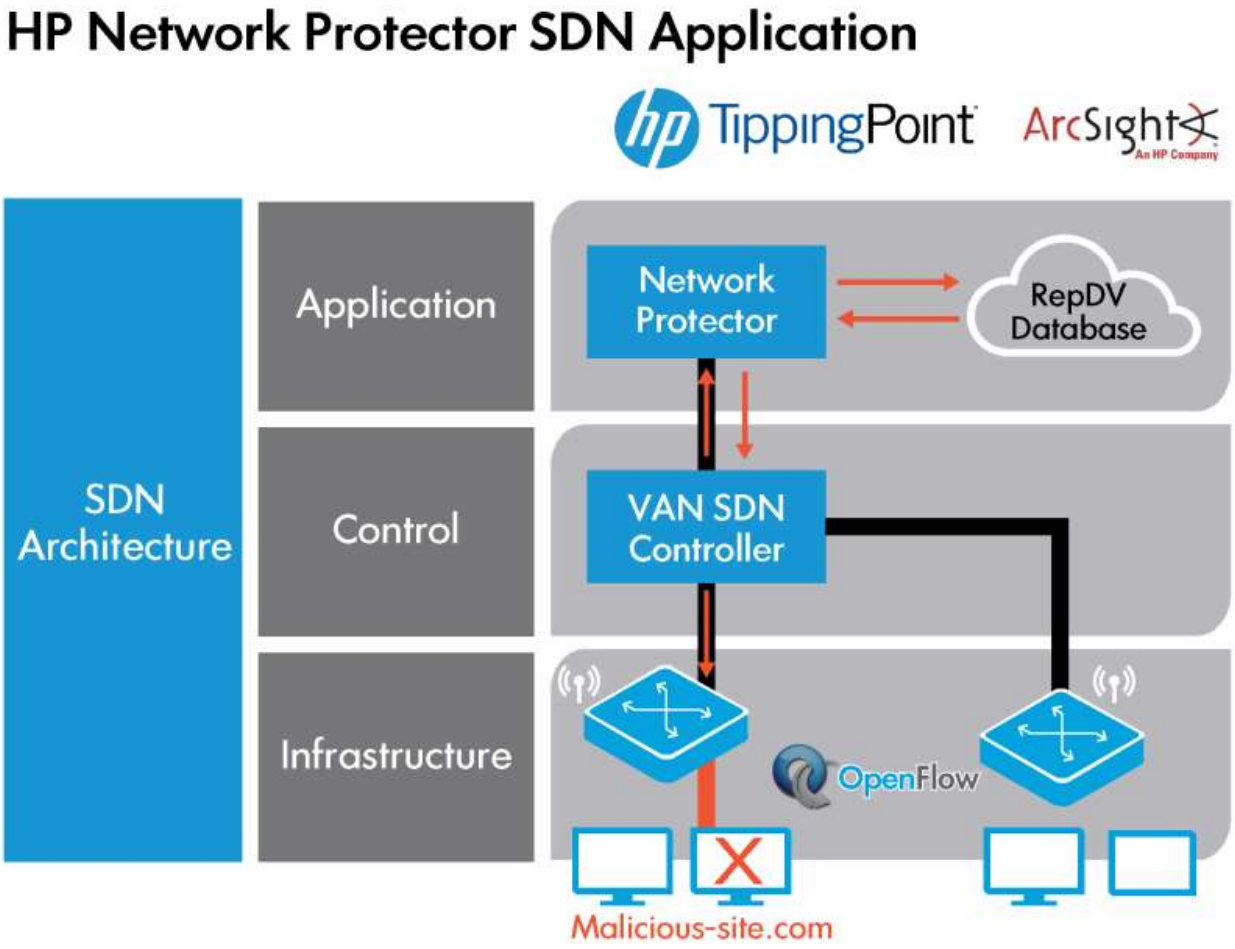
Provides Real-time Threat Detection Across Enterprise Campus Networks

- Protects from over one million malicious botnet, malware and spyware sites.
- Enables real-time threat characterization with the HP TippingPoint Reputation Digital Vaccine (RepDV) cloud service data base.
- Can address cloud-based threat intelligence.

Proactive IT Management of Threats

- Allows flow-based dynamic access control lists (ACL), bringing security to the next level.
- Allows for per switch and device inspection throttling.
- Provides enhanced white/black/grey list user policy routing.

HP Network Protector Solution Overview



The HP Network Protector SDN Application leverages HP Networking, TippingPoint, 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 TippingPoint Reputation Digital Vaccine (RepDV).

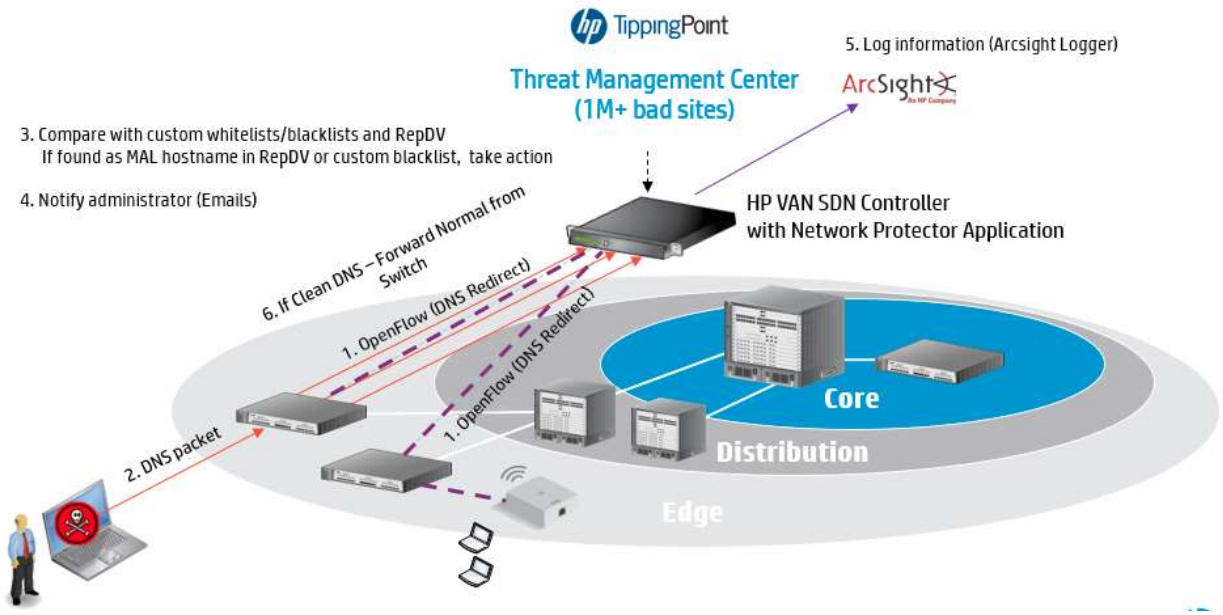
This turns the entire network infrastructure into security-enforcement devices, providing visibility and threat protection against more than one million malicious botnets, malware, and spyware sites.

Some of the key features of the solution include:

- Runs on HP VAN SDN Controller
- Complimentary to TippingPoint IPS solutions
- Delivers real-time threat characterization with HP TippingPoint 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

1: High Level HP Network Protector Architecture



HP VAN SDN Controller

- The HP Network Protector is deployed as an application that runs on top of the HP VAN SDN Controller.
- As a stand-alone application bundled with the controller, it leverages several controller features and subsystems like Application Manager, Pipeline Manager, Licensing infrastructure, Cassandra Database, SKI UI framework, REST API framework, Audit, Alert, Support logs, and others.

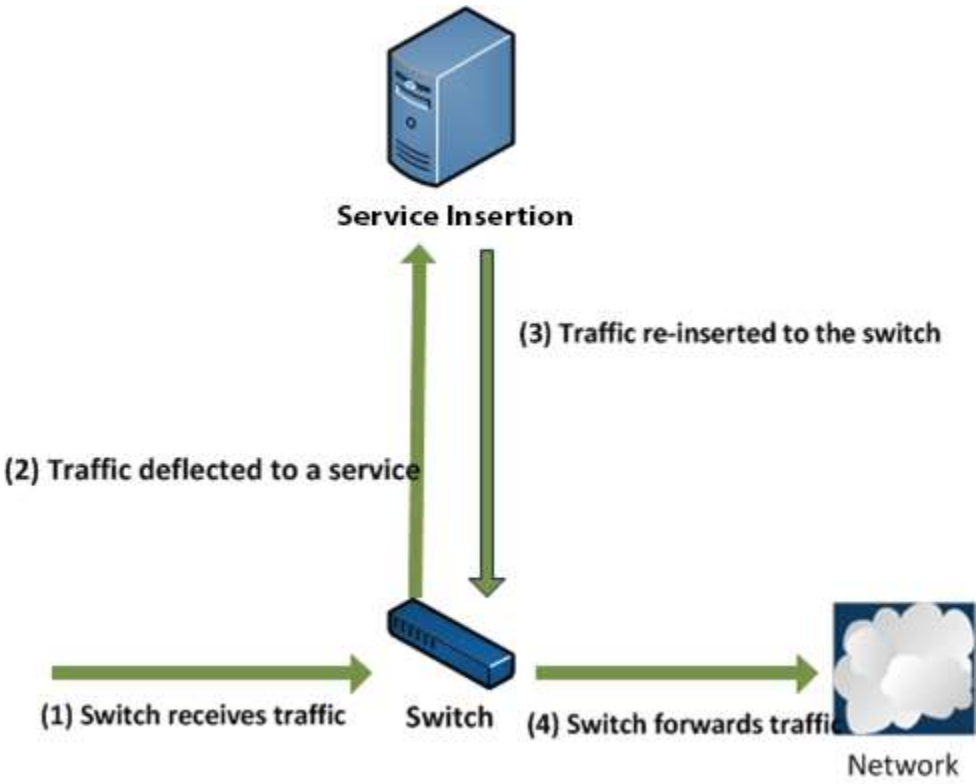
OpenFlow enabled switches

- One of the basic requirements for the application is the OpenFlow enabled switchs. OpenFlow is the mechanism by which the application instructs the discovered switches to redirect all DNS traffic towards itself.
- There are several security policies supported in the application, which are implemented by using the OpenFlow protocol to push desired flows on the switches. Currently, OpenFlow 1.0 and OpenFlow 1.3 versions are supported.
- The switch firmware plays an important role in the proper functioning of the application. Provision switch firmware version 15.15 and above support an additional switch feature called **Service Insertion**, which helps send DNS data traffic to the switch using switch hardware, bypassing the switch CPU, thereby enhancing performance.
- Packet processing using the switch CPU is slower than the packet processing using switch hardware.
- Switch capabilities and extensions or lack thereof have significant impact on how much actual packet processing needs to be handled by the application.
- The base level OpenFlow switch with no Service Insertion is the most rudimentary environment and all inspected traffic and control is shared on the OpenFlow interface port. Best performance is achieved with switches that support OpenFlow and Service Insertion.
- The application scalability and performance also depends on the hardware and software specifications on the server on which it is deployed.

Service Insertion

- The application uses OpenFlow to redirect DNS traffic from the switch to the application and compares it against RepDV to make forwarding or blocking decisions. However, packets of new flows need to be copied to switch CPU for processing and then redirected to the application, limiting performance in the range of tens of megabits per second.
- To maximize the performance and keep the switch to what it does best, packets switching, the switch hardware is used to pipe traffic directly to application, yielding potential performance in the gigabits per second range or line rate.
- In other words, packets are forwarded by switch hardware instead by switch CPU through a Service Insertion tunnel. The desired best performance for application is achieved with switches that support OpenFlow and tunnel technology for Service Insertion.

2: Service insertion mechanism



Link aggregation

This application supports link aggregation, in which multiple links between the switches and the controller are combined to:

- Increase throughput over that provided by a single link

- Provide redundant connections to the controller in case of a single link failure.

To implement link aggregation, all the links must have the same bandwidth.

HP TippingPoint RepDV Cloud Service

- TippingPoint Reputation Digital Vaccine (RepDV) is a subscription service that enables organizations to monitor and block inbound and outbound communications with known malicious and undesirable hosts. RepDV is a robust security intelligence feed powered by advanced analytics and a global reputation database of IPv4, IPv6, and DNS names.
- The RepDV database includes more than a million known malicious or undesirable hosts collected from HP TippingPoint ThreatLinQ global intelligence network, DVLabs malware repository and honeypot network, third-party commercial sources, and open source black lists.
- A threat score of 1 to 100 is assigned to each entry based on DVLabs analysis of the activity, source, category, and threat.
- Customers can tune RepDV policies based on reputation score, category, or geolocation to meet custom security requirements. RepDV is updated multiple times a day to stay ahead of emerging threats and reduce customers' security risk.
- The application interfaces with the RepDV Cloud service to download the RepDV database and update its local copy of the same. This database forms the basis of DNS hostname comparisons after being filtered based on policies defined within the application.
- The application polls for updates every two hours (adjustable from the GUI) from the service to keep itself updated of new threats.

ArcSight CEF Logger

- ArcSight is the universal log management solution to unify logs across the IT to collect, store, and search. HP ArcSight Logger can improve

compliance, risk management, security intelligence, IT operations, and efforts that prevent insider and advanced persistent threats.

- This universal log management solution collects machine data from any log-generating source and unifies the data for searching, indexing, reporting, analysis, and retention. And in the age of BYOD and mobility, it enables comprehensive management of increasing volume of log data from an increasing number of sources.
- The HP Network Protector supports ArcSight CEF syslog output so events can be sent directly to ArcSight Logger for enterprise visibility. ArcSight CEF is compatible with many generic syslog servers and supports all standard syslog servers.

HP Intelligent Management Center

- The HP Network Protector exposes REST interface which IMC can use to get information about the policies and statistics.
- The application can also use IMC UAM module's REST API for getting the end user correlation using the IP and MAC address of the end point as collected from the DNS packets received in the application.

HP Network Visualizer

Network Visualizer – SDN App By HP

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.

HP Network Optimizer

Network Optimizer – SDN App By HP

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

Optimize – Multilayer Network Optimizer

- Optimize is an application run periodically to defragment the network. When connections are initially provisioned, the Navigate multilayer path computation engine finds their optimal paths through the network; however, over time as connections churn and resources are freed up or new resources are added, the network may become sub-optimally configured and utilized. In the aggregate, connections may take more average hops, incur more average latency, require more resources, and cause more current or potential blocking than is necessary.
- Optimize interrogates the topology and connections database and the network policy database and determines how raw capacity could be re-allocated, and existing connections re-groomed, to better optimize the network.
- It then generates the sequence of steps to migrate the client services connections and the underlying transport connections to their new home paths with minimal disruption, making use of temporary bandwidth and make-before-break techniques as appropriate.
- Optimize defragments the network and minimizes stranded bandwidth—delaying as long as possible the need to add new physical resources to the network. This saves CAPEX.

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.
- The Cisco Extensible Network Controller (XNC) is the first commercial version of the OpenDaylight controller. Continuing Cisco's commitment to open networking, Cisco XNC 1.5 offers programmability support for new network devices.
- With Cisco XNC, you gain more control as you make the network application-aware. The programmatic access to the network helps to automate network behavior, so the network responds faster to changing business and application needs.
- Cisco XNC and Cisco XNC applications offer customers a low-risk, blueprint approach to introducing software-defined networking (SDN) into their environments. Using a centralized controller along with the local control plane provides a highly secure, scalable, and resilient deployment model.

As a result, Cisco XNC provides the functionality required for production environments, such as:

- Monitoring, topology-independent forwarding (TIF), high availability and network slicing applications

- Advanced troubleshooting and debugging capabilities
- Support for the Cisco Open Network Environment (ONE) Platform Kit (onePK), in addition to its OpenFlow support.

Cisco XNC can run on a virtual machine (VM) or on a bare-metal service and can be used to manage any third-party switches, as long as they support OpenFlow. It uses the Open Services Gateway Initiative (OSGi) framework, which offers the modular and extensibility needs that business-critical application requires.

Cisco XNC features include:

- An extensible, open, and modular architecture based on OpenDaylightMultiprotocol support using OpenFlow and Cisco's One Platform Kit (onePK)Highly secure communication using HTTPS and Transport Layer SecurityCluster-based active-active deployment support for high-availability and scalabilityThe benefits of Cisco XNC and Cisco XNC applications include:
- Extensibility of Cisco XNC functions using Java or Python and the ability to support multiple protocols for device communicationConsistent management access through the GUI or through Java OSGi or representational state transfer (REST) northbound APIsNetwork end-to-end forwarding rules programming, plus flow-level visibility for statistics and troubleshootingIn addition to providing all the advanced features that the market expects of an SDN controller, the Cisco XNC 1.5 software release provides many unique features that improve control of the network.
- Multiprotocol southbound support such as OpenFlow 1.0.
- The features that support network visibility and programmability, including network topology discovery, network device management, forwarding rule programming, and also access to detailed network statistics.
- A service abstraction layer that enables modular southbound interface support, such as OpenFlow 1.0.
- Consistent administrative access.
- The Security features like role-based access control and integration with external Active Directory or TACACS for authentication and authorization.
- Troubleshooting tools such as analysis collection and diagnostic packet injection.

Cisco XNC 1.5 features include:

GUI enhancements to better manage flow rules

- Cisco onePK plugin for programming device functions
- Automation of the link-layer packet flows across point-to-point ports on a switch
- Monitor Manager advancements that support QinQ

Monitor Manager Embedded – Enabling single-switch deployment for network traffic monitoring solution

Cisco XNC applications include:

Monitor Manager :

Monitor Manager provides traffic visibility for troubleshooting, analysis, reporting, and archiving using a centralized, policy-focused approach.

Topology-Independent Forwarding :

Explore the use of application-aware network forwarding functions with the Topology-Independent Forwarding (TIF) module. TIF extends beyond traditional routing concepts and can use metrics to create a forwarding path end to end. These metrics include nonconventional ones like bandwidth, currency (dollar cost), and other custom properties.

Network Slicing :

With Cisco XNC and network slicing, users can partition the network according to physical and logical criteria for multiple user communities. Slicing provides the logical separation required to manage network traffic domains extending beyond VLANs.

Virtual Patch Panel :

Automate port interconnections within a switch or across a network without have to make physical rewiring.

Controller and Application Product Packaging :

Cisco XNC and Cisco XNC applications are available as standalone products and in bundles. The Monitor Manager, TIF, network slicing, the virtual patch panel, and other individual applications require the Cisco XNC controller as a base module.

PRACTICAL NO : 03

Install ODL(OpenDayLight) controller

ODL - OpenDaylight is an open source SDN controller / framework, hosted by the Linux Foundation. It's one of the more popular (open source) SDN controllers at the moment.

One of the southbound interface protocols it supports is OpenFlow. To test OpenDaylig3ht, we'll need some switches that support OpenFlow.

The following outline records the steps necessary to install OpenDaylight on Ubuntu LTS 18.04

1. Prepare the operating system
2. Install the Java JRE
3. Download OpenDaylight
4. Install OpenDaylight
5. Create a *systemd* service configuration file
6. Install and enable the *systemd* OpenDaylight service

- **Prepare operating system**

Run an ***apt-get*** update to ensure that your server receives all of the most recent security and application packages.

Step1 : sudo apt-get update

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ sudo apt-get update
[sudo] password for ubuntu:
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [2,039 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [1,277 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [295 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1,734 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [1,566 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [289 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,468 B]
Get:12 http://us.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [9,288 B]
Fetched 7,464 kB in 38s (193 kB/s)

(appstreamcli:2619): GLib-CRITICAL **: g_strchug: assertion 'string != NULL' failed

(appstreamcli:2619): GLib-CRITICAL **: g_strchomp: assertion 'string != NULL' failed

(appstreamcli:2619): GLib-CRITICAL **: g_strchug: assertion 'string != NULL' failed

(appstreamcli:2619): GLib-CRITICAL **: g_strchomp: assertion 'string != NULL' failed

(appstreamcli:2619): GLib-CRITICAL **: g_strchug: assertion 'string != NULL' failed

(appstreamcli:2619): GLib-CRITICAL **: g_strchomp: assertion 'string != NULL' failed
AppStream cache update completed, but some metadata was ignored due to errors.
Reading package lists... Done
```

Now, install the following convenience packages

Step 2 : sudo apt-get -y install unzip vim wget

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ sudo apt-get -y install unzip vim wget
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libc-bin libc-dev-bin libc6 libc6-dbg libc6-dev libgpm2 libidn2-0 libpsl5 libpython3.6
  libpython3.6-minimal libpython3.6-stdlib libreadline7 libssl1.1 libunistring2 locales publicsuffix
  vim-common vim-runtime vim-tiny xxd
Suggested packages:
  glibc-doc gpm ctags vim-doc vim-scripts indent
The following NEW packages will be installed:
  libidn2-0 libpsl5 libpython3.6 libpython3.6-minimal libpython3.6-stdlib libreadline7 libssl1.1
  libunistring2 publicsuffix vim vim-runtime xxd
The following packages will be upgraded:
  libc-bin libc-dev-bin libc6 libc6-dbg libc6-dev libgpm2 locales unzip vim-common vim-tiny wget
11 upgraded, 12 newly installed, 0 to remove and 1439 not upgraded.
Need to get 28.2 MB of archives.
After this operation, 77.8 MB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc-dev-bin amd64 2.27-3ubuntu1.4 [7
1.8 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc6-dev amd64 2.27-3ubuntu1.4 [2,58
5 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc6-dbg amd64 2.27-3ubuntu1.4 [5,16
3 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 locales all 2.27-3ubuntu1.4 [3,611 kB
]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc6 amd64 2.27-3ubuntu1.4 [2,832 kB
]
```

- **Install the Java JRE**

Installation of OpenDaylight via the release **zip** archive requires the JAVA 8 runtime environment.

Run the following command to install the JRE

Step 3 : sudo apt-get -y install openjdk-8-jre

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ sudo apt-get -y install openjdk-8-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libatk-wrapper-java libatk-wrapper-java-jni libpng16-16 openjdk-8-jre-headless zlib1g
Suggested packages:
  icedtea-8-plugin fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei
  fonts-indic
The following NEW packages will be installed:
  libatk-wrapper-java libatk-wrapper-java-jni libpng16-16
The following packages will be upgraded:
  openjdk-8-jre openjdk-8-jre-headless zlib1g
3 upgraded, 3 newly installed, 0 to remove and 1436 not upgraded.
Need to get 28.6 MB of archives.
After this operation, 879 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 zlib1g amd64 1:1.2.11.dfsg-0ubuntu2 [56.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libpng16-16 amd64 1.6.34-1ubuntu0.18.
04.2 [176 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 libatk-wrapper-java all 0.33.3-20ubuntu0.1 [3
4.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 libatk-wrapper-java-jni amd64 0.33.3-20ubuntu
0.1 [28.3 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 openjdk-8-jre amd64 8u282-b08-0ub
untu1~18.04 [69.7 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 openjdk-8-jre-headless amd64 8u28
2-b08-0ubuntu1~18.04 [28.2 MB]
Fetched 28.6 MB in 1min 38s (290 kB/s)
(Reading database ... 202432 files and directories currently installed.)
```

Run the following command. If it does not point to *JAVA 8*, be sure to select version 8 from the list.

Step 4 : sudo update-alternatives --config java

```
ubuntu@ubuntu:~$ sudo update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-8-openjdk-a
md64/jre/bin/java
Nothing to configure.
ubuntu@ubuntu:~$ echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' >> ~/.bashrc
ubuntu@ubuntu:~$ source ~/.bashrc
ubuntu@ubuntu:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-openjdk-amd64/jre
ubuntu@ubuntu:~$
```

Copy the link to the binary above, as you will need this information in the next step.

With the path in hand, run the following command to update your **BASHRC** file.

Step 5 : echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' >> ~/.bashrc

Now ***source*** your ***BASHRC*** file and then check to ensure ***$JAVA_HOME*** lives in the environment.

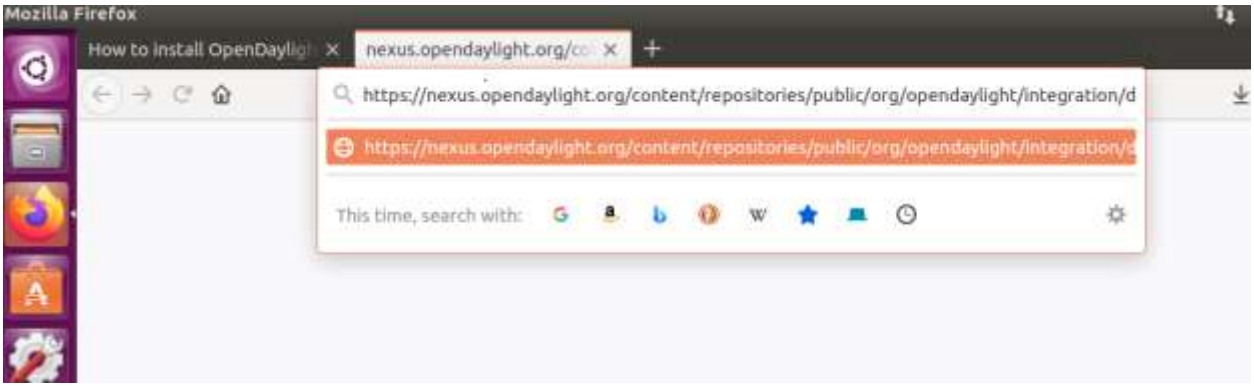
Step 6 : source ~/.bashrc

Double check that ***$JAVA_HOME*** ends with ***/jre***.

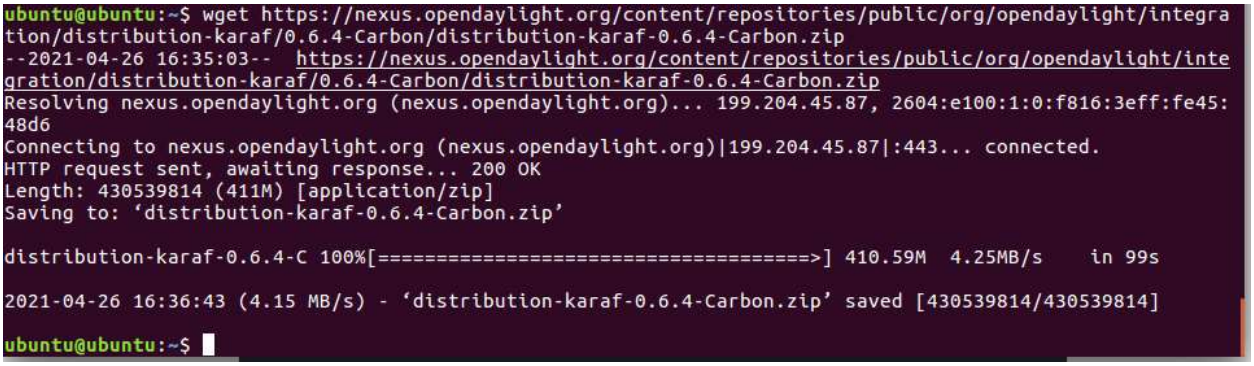
Step 7 : echo $JAVA_HOME

- **Download the OpenDaylight Zip Archive**

Simply right click the Version name, select 'Copy Link' and then run the following command. The following example depicts the command you need to execute to download *Carbon*.



Step 8 : wget https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.6.4-Carbon/distribution-karaf-0.6.4-Carbon.zip



- **Install OpenDaylight**

First, make a directory for the binary.

Step 9 : sudo mkdir /usr/local/karaf

Move the zip archive to the install workspace and deflate the archive. Be sure to use the correct version. I downloaded version ***0.8.4*** and yours may be different.

Step 10 : sudo mv distribution-karaf-0.6.4-Carbon.zip /usr/local/karaf

Step 11 : sudo unzip /usr/local/karaf/distribution-karaf-0.6.4-Carbon.zip -d /usr/local/karaf/

Install ***karaf*** into user space.

Step 12 : sudo update-alternatives --install /usr/bin/karaf karaf /usr/local/karaf/distribution-karaf-0.6.4-Carbon/bin/karaf 1

sudo update-alternatives --config karaf

which karaf

```
ubuntu@ubuntu:~$ sudo update-alternatives --install /usr/bin/karaf karaf /usr/local/karaf/distribution-ka
raf-0.6.4-Carbon/bin/karaf 1
update-alternatives: using /usr/local/karaf/distribution-karaf-0.6.4-Carbon/bin/karaf to provide /usr/bin
/karaf (karaf) in auto mode
ubuntu@ubuntu:~$ sudo update-alternatives --config karaf
There is only one alternative in link group karaf (providing /usr/bin/karaf): /usr/local/karaf/distributi
on-karaf-0.6.4-Carbon/bin/karaf
Nothing to configure.
ubuntu@ubuntu:~$ which karaf
/usr/bin/karaf
ubuntu@ubuntu:~$
```

Execute the ***karaf*** command via sudo and pass the ***-E*** flag to keep the ***$JAVA_HOME*** environment variable.

Step 13 : sudo -E karaf

```
ubuntu@ubuntu:~$ sudo -E karaf
Apache Karaf starting up. Press Enter to open the shell now...
100% [========================================================================]

Karaf started in 9s. Bundle stats: 64 active, 64 total

    ________                ________                .__  .__       .__    __
    \_____  \ ______ ____ ____\_____ \ ____ ___.__.|  | |__| ____ |  |__/  |_
     /   |   \\____ \_/ __ \/    \  _  \\__  <   |  ||  | |  |/ ___\|  |  \   __\
    /    |    \  |_> >  ___/   |   \  \/ / __ \\___  ||  |_|  / /_/  >   Y  \  |
    \_______  /   __/ \___  >___|  /_____/ (____  / ____||____/\___  /|___|  /__|
            \/|__|        \/     \/             \/\/          /_____/      \/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

You can verify that Karaf runs via a ***netstat***.

Step 14 : sudo netstat -an | grep 8181

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ sudo netstat -an | grep 8181
[sudo] password for ubuntu:
tcp6       0      0 :::8181                 :::*                    LISTEN
ubuntu@ubuntu:~$
```

STEP 15 : Install features following ARE ODL-DLUX-ALL FEATURES

opendaylight-user@root> feature:install odl-l2switch-switch-ui

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 //

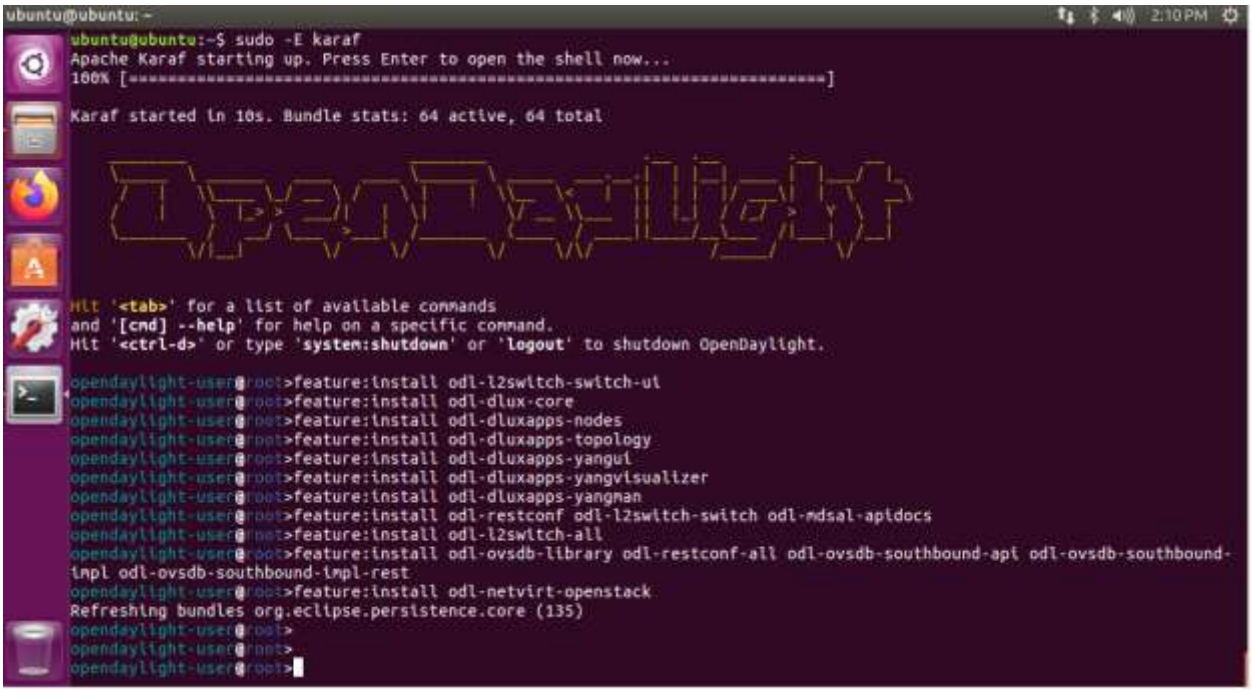
opendaylight-user@root> feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs

opendaylight-user@root> feature:install odl-l2switch-all

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

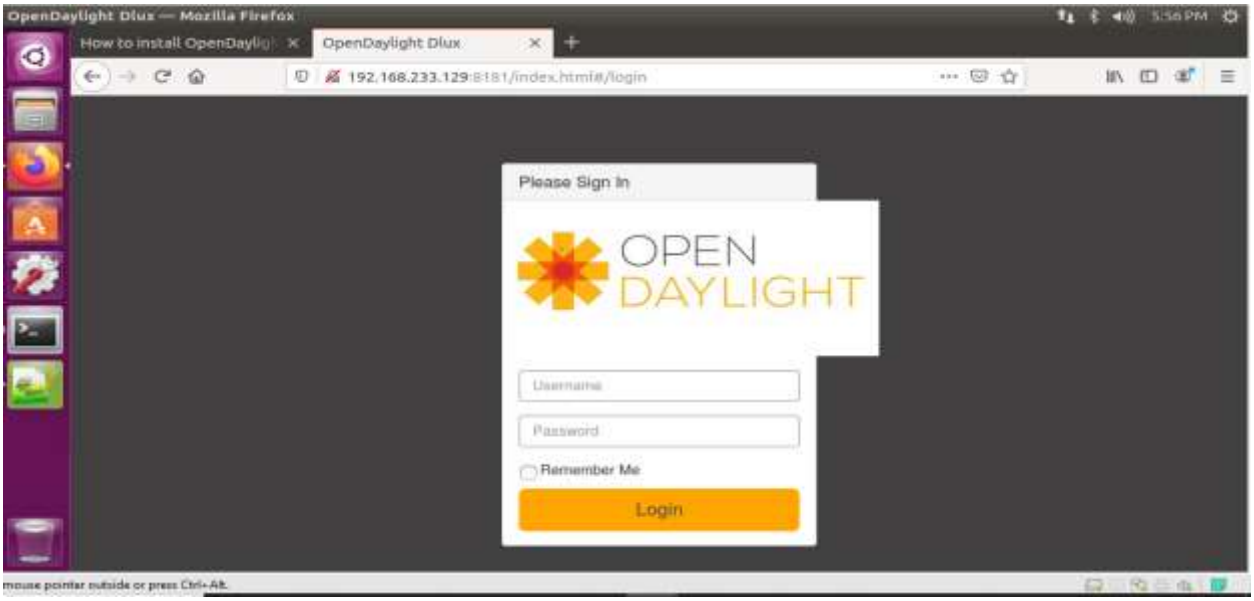
opendaylight-user@root>feature:install odl-ovsdb-library odl-restconf-all odl-ovsdb-southbound-api odl-ovsdb-southbound-impl odl-ovsdb-southbound-impl-rest

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

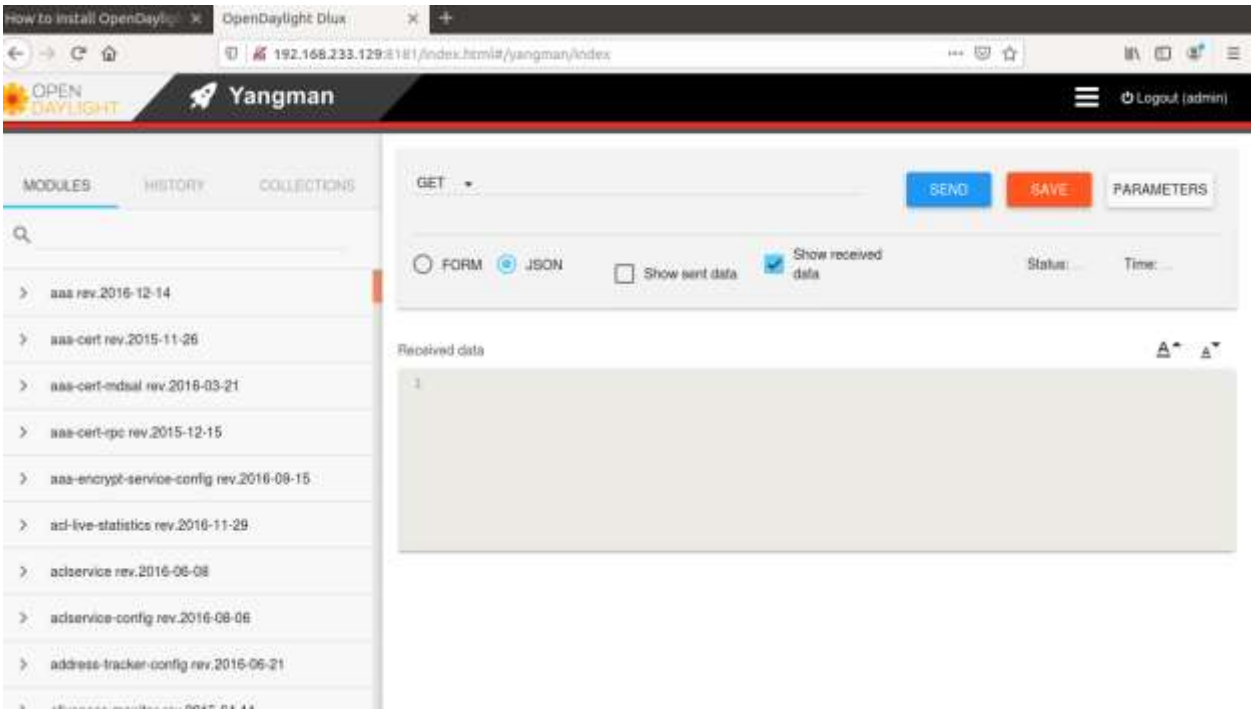


Alternatively (assuming your firewall/ security groups permit it), you can go to your URL and log into the DLUX console using credentials ***admin***/***admin***. Be sure to put your IP address in the following URL (Keep the port as ***8181***).

Step 16 : http://localhost:8181/index.html



If you log in with ***admin***/***admin***, you will see the ***DLUX*** console.



PRACTICAL NO : 04

Implement OVS

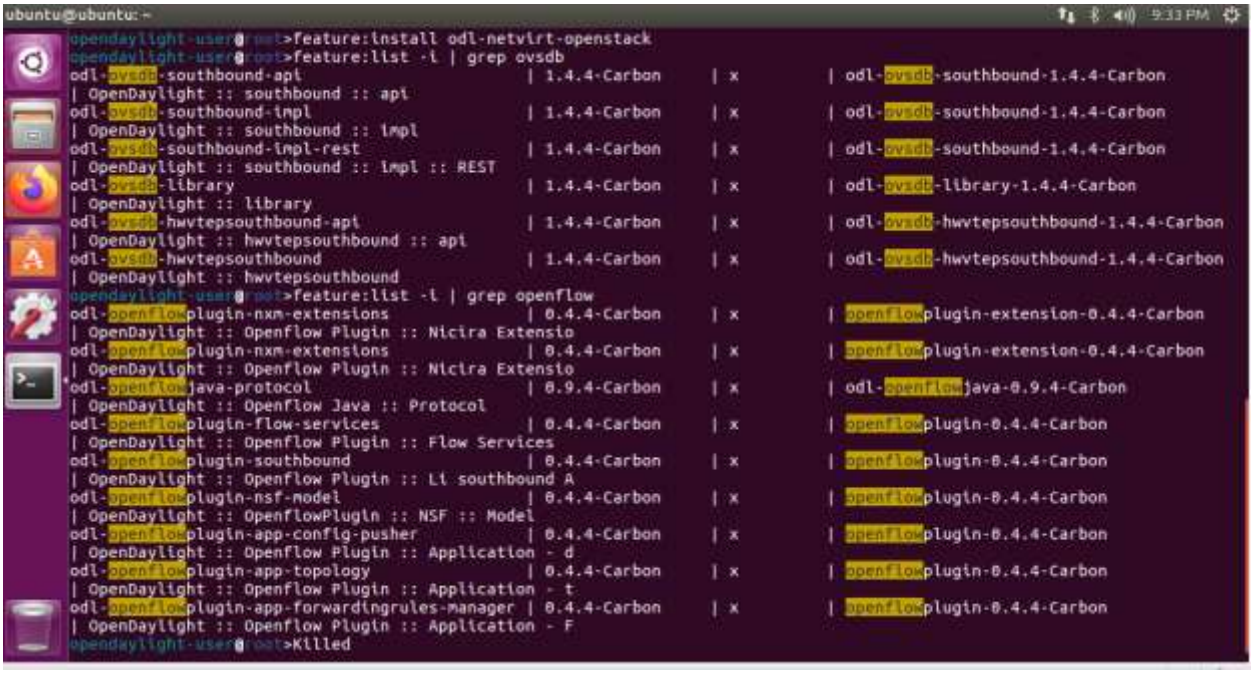
OpenShift Container Platform uses a software-defined networking (SDN) approach to provide a unified cluster network that enables communication between pods across the OpenShift Container Platform cluster. This pod network is established and maintained by the OpenShift SDN, which configures an overlay network using Open vSwitch (OVS).

OVSDB is OVS database management protocol which defines schema for OVS database and specification for communication between controller and switch. SDN controller uses this protocol to control OVS switch by populating its database with the intended configuration.

As we need to manage OVS switches to which OVSDB and Openflow connections will be made, so Openflow and OVSDB plugins need to be installed into ODL. which is to be done This can be done as shown below:

Step 1 : feature:list -i | grep openflow

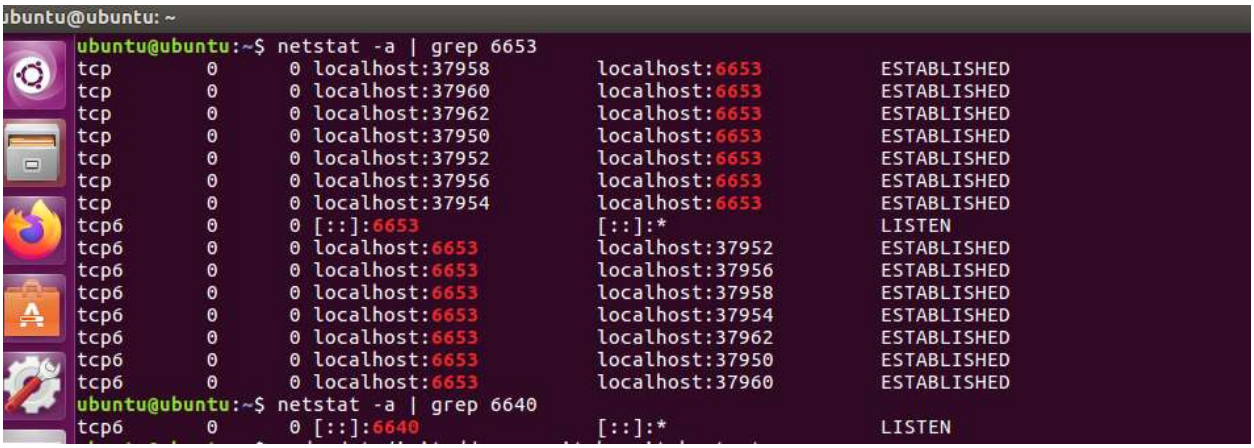
feature:list -i | grep ovsdb



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

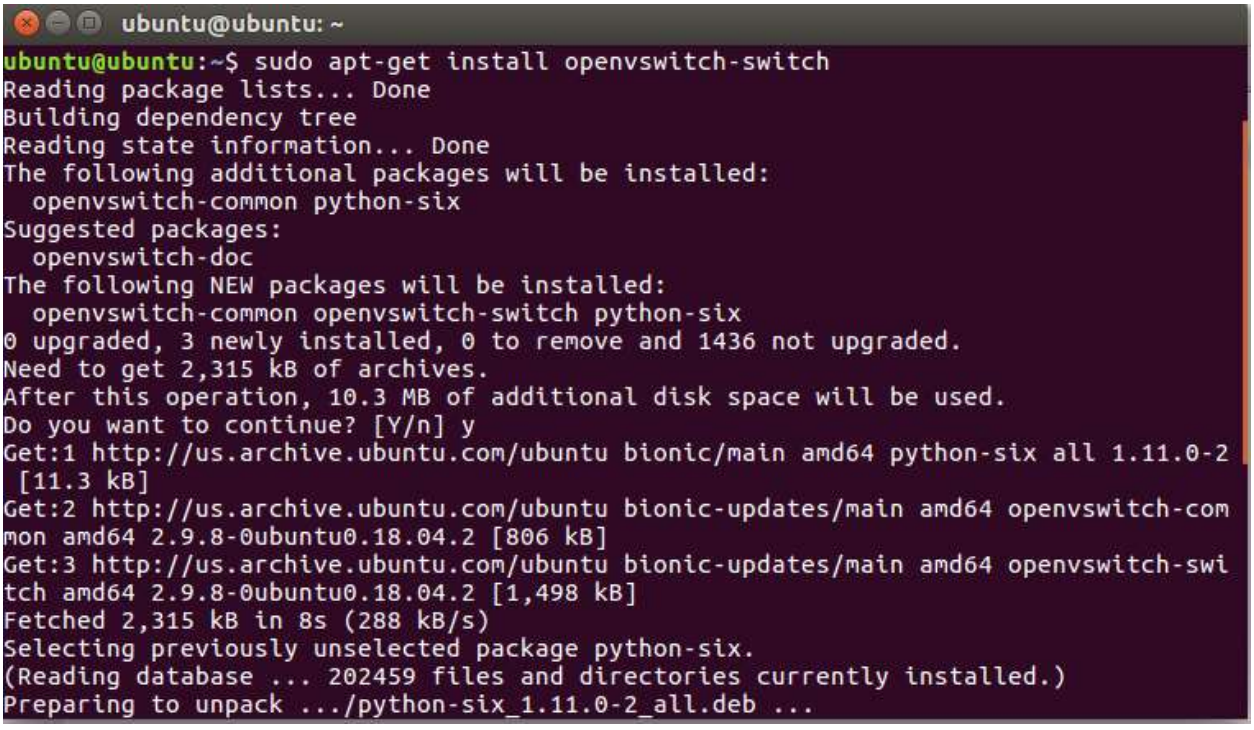
Step 2 : netstat -a | grep 6653

netstat -a | grep 6640



Run this command to install openvswitch

Step 3 : sudo apt-get install openvswitch-switch



It is noted that OVS switch should already be running before executing any OVS command. To run OVS on Ubuntu, run the following command with root permissions:

Step 4 : sudo /etc/init.d/openvswitch-switch start

It is available as service, so you can check its status as well.

Step 5 : sudo /etc/init.d/openvswitch-switch status

```
ubuntu@ubuntu:~$ sudo /etc/init.d/openvswitch-switch start
[sudo] password for ubuntu:
 * ovsdb-server is already running
 * ovs-vswitchd is already running
 * Enabling remote OVSDB managers
ubuntu@ubuntu:~$ sudo /etc/init.d/openvswitch-switch status
ovsdb-server is running with pid 1044
ovs-vswitchd is running with pid 1151
```

Once, OVS is running, you can execute its CLI commands also. For example,

Step 6 : sudo ovs-vsctl show

```
ubuntu@ubuntu:~$ sudo ovs-vsctl show
e7846c50-b49e-400c-b382-af8997bcb747
    Bridge "s5"
        Controller "ptcp:6658"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port "s5-eth3"
            Interface "s5-eth3"
                error: "could not open network device s5-eth3 (No such device)"
        Port "s5-eth1"
            Interface "s5-eth1"
                error: "could not open network device s5-eth1 (No such device)"
        Port "s5"
            Interface "s5"
                type: internal
        Port "s5-eth2"
            Interface "s5-eth2"
                error: "could not open network device s5-eth2 (No such device)"
    Bridge "s6"
        Controller "ptcp:6659"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port "s6"
            Interface "s6"
                type: internal
        Port "s6-eth2"
            Interface "s6-eth2"
                error: "could not open network device s6-eth2 (No such device)"
        Port "s6-eth3"
            Interface "s6-eth3"
```

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

Step 7 : ovs-vsctl set-manager tcp:127.0.0.1:6640

sudo ovs-vsctl show

```
ubuntu@ubuntu:~$ sudo ovs-vsctl set-manager tcp:127.0.0.1:6640
ubuntu@ubuntu:~$ sudo ovs-vsctl show
e7846c50-b49e-400c-b382-af8997bcb747
    Manager "tcp:127.0.0.1:6640"
        is_connected: true
    Bridge "s5"
        Controller "ptcp:6658"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port "s5-eth3"
            Interface "s5-eth3"
                error: "could not open network device s5-eth3 (No such device)"
        Port "s5-eth1"
            Interface "s5-eth1"
                error: "could not open network device s5-eth1 (No such device)"
        Port "s5"
            Interface "s5"
                type: internal
        Port "s5-eth2"
            Interface "s5-eth2"
                error: "could not open network device s5-eth2 (No such device)"
    Bridge br-int
        Controller "tcp:127.0.0.1:6653"
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
    Bridge "s6"
        Controller "ptcp:6659"
        Controller "tcp:127.0.0.1:6653"
```

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.

Step 8 : ovs-vsctl set-controller br-int tcp:127.0.0.1:6653

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

```
ubuntu@ubuntu:~$ sudo ovs-vsctl set-controller br-int tcp:127.0.0.1:6653
ubuntu@ubuntu:~$ sudo ovs-vsctl show
e7846c50-b49e-400c-b382-af8997bcb747
    Manager "tcp:127.0.0.1:6640"
        is_connected: true
    Bridge "s5"
        Controller "ptcp:6658"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port "s5-eth3"
            Interface "s5-eth3"
                error: "could not open network device s5-eth3 (No such device)"
        Port "s5-eth1"
            Interface "s5-eth1"
                error: "could not open network device s5-eth1 (No such device)"
        Port "s5"
            Interface "s5"
                type: internal
        Port "s5-eth2"
            Interface "s5-eth2"
                error: "could not open network device s5-eth2 (No such device)"
    Bridge br-int
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
    Bridge "s6"
        Controller "ptcp:6659"
```

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

Step 9 : sudo ovs-ofctl show br-int -OOpenFlow13

```
ubuntu@ubuntu:~$ sudo ovs-ofctl show br-int -OOpenFlow13
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:00003000d3164e6b
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 LOCAL(br-int): addr:30:00:d3:16:4e:6b
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x7): frags=normal miss_send_len=0
ubuntu@ubuntu:~$
```

Even ODL has dashboard DLUX feature which gives GUI web interface to ODL controller at url http://localhost: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.


localhost:8181/index.html#/topology
OPEN DAYLIGHT
Topology
Logout (admin)
Controls
Reload
openflow:81885942504520

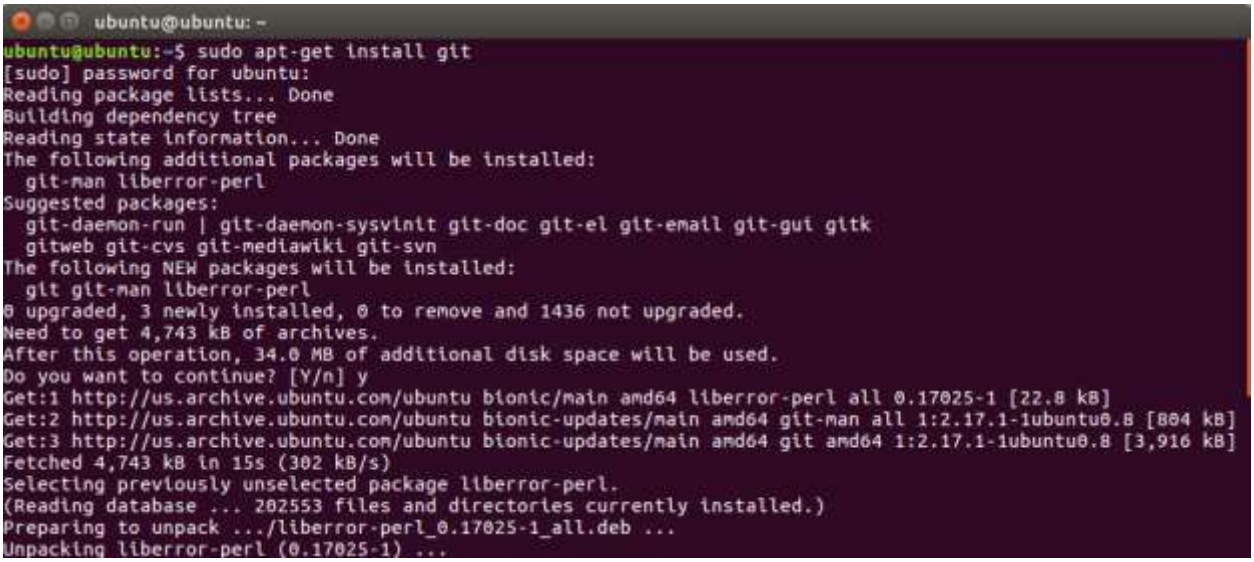
PRACTICAL NO : 05

Implement mininet on ODL

Mininet - Mininet provides a virtual test bed and development environment for software-defined networks (SDN).

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

Step 1 : sudo apt-get install git



Step 2 : git clone git://github.com/mininet/mininet

After installation creates a folder in the home directory named mininet that contains the project file structure.

Step 3 : cd mininet/

```
ubuntu@ubuntu:~$ cd mininet/
ubuntu@ubuntu:~/mininet$ ls
bin           custom  doc       INSTALL  Makefile  mnexec.c   setup.py
CONTRIBUTORS  debian  examples  LICENSE  mininet   README.md  util
ubuntu@ubuntu:~/mininet$
```

To find the latest beta version of Mininet, list all tagged releases in the Mininet project.

Step 4 : git tag

```
ubuntu@ubuntu: ~/mininet
bin           custom  doc       INSTALL  Makefile  mnexec.c   setup.py
CONTRIBUTORS  debian  examples  LICENSE  mininet   README.md  util
ubuntu@ubuntu:~/mininet$ git tag
1.0.0
2.0.0
2.1.0
2.1.0p1
2.1.0p2
2.2.0
2.2.1
2.2.2
2.3.0
2.3.0b1
2.3.0b2
2.3.0d3
2.3.0d4
2.3.0d5
2.3.0d6
2.3.0rc1
2.3.0rc2
cs244-spring-2012-final
ubuntu@ubuntu:~/mininet$ git checkout -b cs244-spring-2012-final
Switched to a new branch 'cs244-spring-2012-final'
ubuntu@ubuntu:~/mininet$
```

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

Step 5 :util/install.sh -a

```
ubuntu@ubuntu: ~/mininet
ubuntu@ubuntu:~/mininet$ util/install.sh -a
Detected Linux distribution: Ubuntu 16.04 xenial amd64
sys.version_info(major=2, minor=7, micro=12, releaselevel='final', serial=0)
Detected Python (python) version 2
Installing all packages except for -eix (doxypy, ivs, nox-classic)...
Install Mininet-compatible kernel if necessary
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [1,697 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [294 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [324 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu bionic-updates/restricted i386 Packages [24.5 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [290 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,468 B]
Get:11 http://us.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [9,288 B]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/main i386 Packages [973 kB]
Get:13 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [48.9 kB]
Get:14 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [60.4 kB]
Get:15 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Fetched 3,977 kB in 8s (491 kB/s)

(appstreamcli:11494): GLib-CRITICAL **: g_strchug: assertion 'string != NULL' failed
```

Step 6 : ps aux|grep -i ovs

sudo kill -9 18342 18386 31373

```
ubuntu@ubuntu:~/mininet$ ps aux|grep -i ovs
root       6687  0.0  0.1  21404  3248 ?        S<s  18:29   0:00 ovsdb-server /etc/openvswitch/conf.db -vconsol
e:emer -vsyslog:err -vfile:info --remote=punix:/var/run/openvswitch/db.sock --private-key=db:Open_vSwitch,SSL,pr
ivate_key --certificate=db:Open_vSwitch,SSL,certificate --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert --no-chd
ir --log-file=/var/log/openvswitch/ovsdb-server.log --pidfile=/var/run/openvswitch/ovsdb-server.pid --detach
root       6742  0.0  0.5  26940 10340 ?        S<Ls 18:29   0:00 ovs-vswitchd unix:/var/run/openvswitch/db.sock
 -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-file=/var/log/openvswitch/ovs-vswitchd.log
--pidfile=/var/run/openvswitch/ovs-vswitchd.pid --detach
ubuntu    19190  0.0  0.0  14416   968 pts/17   S+   19:42   0:00 grep --color=auto -i ovs
ubuntu@ubuntu:~/mininet$ sudo kill -9 6687 6742 19190
ubuntu@ubuntu:~/mininet$
```

Installing mininet

Step 7 : sudo apt-get install mininet

```
ubuntu@ubuntu: ~/mininet
ubuntu@ubuntu:~/mininet$ sudo apt-get install mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cgroup-bin cgroup-tools iperf libcgroup1 python-pkg-resources socat
Suggested packages:
  python-setuptools
The following NEW packages will be installed:
  cgroup-bin cgroup-tools iperf libcgroup1 mininet python-pkg-resources socat
0 upgraded, 7 newly installed, 0 to remove and 1435 not upgraded.
Need to get 819 kB of archives.
After this operation, 2,896 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 libcgroup1 amd64 0.41-8ubuntu2 [42.0 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 cgroup-tools amd64 0.41-8ubuntu2 [66.2 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 iperf amd64 2.0.10+dfsg1-1ubuntu0.18.04.
2 [60.5 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 python-pkg-resources all 39.0.1-2 [128 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 cgroup-bin all 0.41-8ubuntu2 [2,576 B]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 socat amd64 1.7.3.2-2ubuntu2 [342 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 mininet amd64 2.2.2-2ubuntu1 [178 kB]
Fetched 819 kB in 3s (229 kB/s)
Selecting previously unselected package libcgroup1:amd64.
```

Step 8 : sudo service openvswitch-switch status/stop

ps aux|grep -i mn

```
ubuntu@ubuntu:~/mininet$ sudo service openvswitch-switch status/stop
Usage: /etc/init.d/openvswitch-switch {start|stop|restart|force-reload|status|force-stop|force-reload-kmod|load-
kmod}
ubuntu@ubuntu:~/mininet$ ps aux|grep -i mn
ubuntu    20268  0.0  0.0  14416  1004 pts/17   S+   19:49   0:00 grep --color=auto -i mn
```

Step 9 : sudo mn

```
ubuntu@ubuntu:~/mininet$ sudo mn
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
ovs-vsctl: unix:/var/run/openvswitch/db.sock: database connection failed (No such file or directory)
ovs-vsctl exited with code 1
*** Error connecting to ovs-db with ovs-vsctl
Make sure that Open vSwitch is installed, that ovsdb-server is running, and that
"ovs-vsctl show" works correctly.
You may wish to try "service openvswitch-switch start".
ubuntu@ubuntu:~/mininet$
```

Services is used to start and see status of openvswitch

Step 10 : service openvswitch-switch start

service openvswitch-switch status

```
ubuntu@ubuntu:~/mininet$ service openvswitch-switch start
ubuntu@ubuntu:~/mininet$ service openvswitch-switch status
● openvswitch-switch.service - Open vSwitch
   Loaded: loaded (/lib/systemd/system/openvswitch-switch.service; enabled; vendor preset: enabled)
   Active: active (exited) since Mon 2021-04-26 18:29:28 IST; 1h 25min ago
 Main PID: 6750 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/openvswitch-switch.service

Apr 26 18:29:28 ubuntu systemd[1]: Starting Open vSwitch...
Apr 26 18:29:28 ubuntu systemd[1]: Started Open vSwitch.
Apr 26 19:53:16 ubuntu systemd[1]: Started Open vSwitch.
ubuntu@ubuntu:~/mininet$
```

Ping is used to assess the total time it takes to packet send and receives an acknowledgment from the connected network.

Step 11 : sudo mn –test pingall

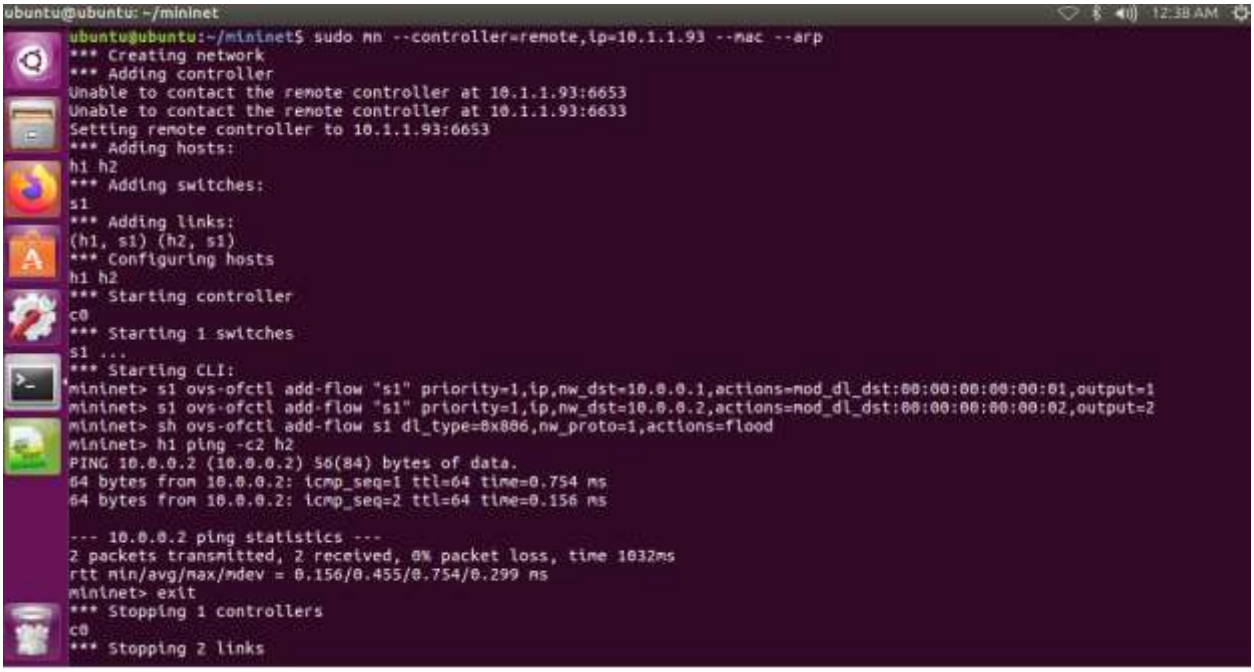
```
ubuntu@ubuntu: ~/mininet
ubuntu@ubuntu:~/mininet$ sudo mn --test pingall
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 0.782 seconds
ubuntu@ubuntu:~/mininet$
```

Step 12 : sudo mn --controller=remote,ip=10.1.1.93 --topo tree,3 –mac

```
ubuntu@ubuntu:~/mininet$ sudo mn --controller=remote,ip=10.1.1.93 --topo tree,3 --mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 10.1.1.93:6653
Unable to contact the remote controller at 10.1.1.93:6633
Setting remote controller to 10.1.1.93:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7
, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 14 links
..............
*** Stopping 7 switches
s1 s2 s3 s4 s5 s6 s7
*** Stopping 8 hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Done
completed in 52.500 seconds
ubuntu@ubuntu:~/mininet$
```

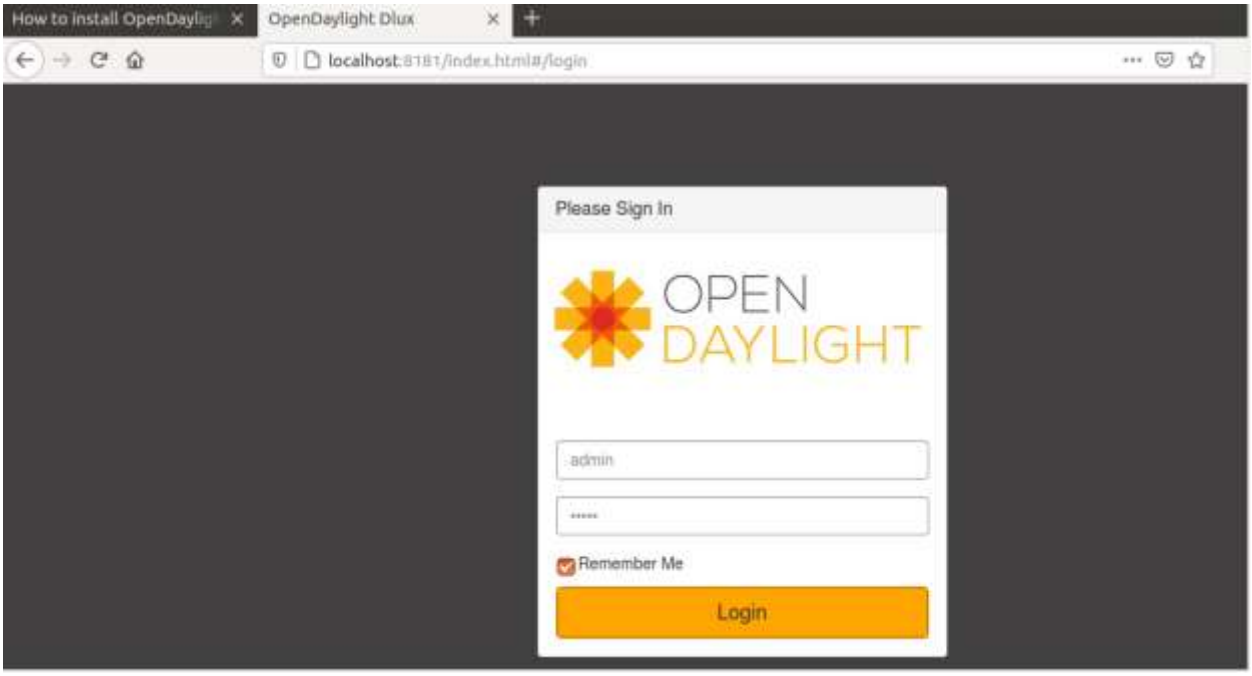
With different topology:

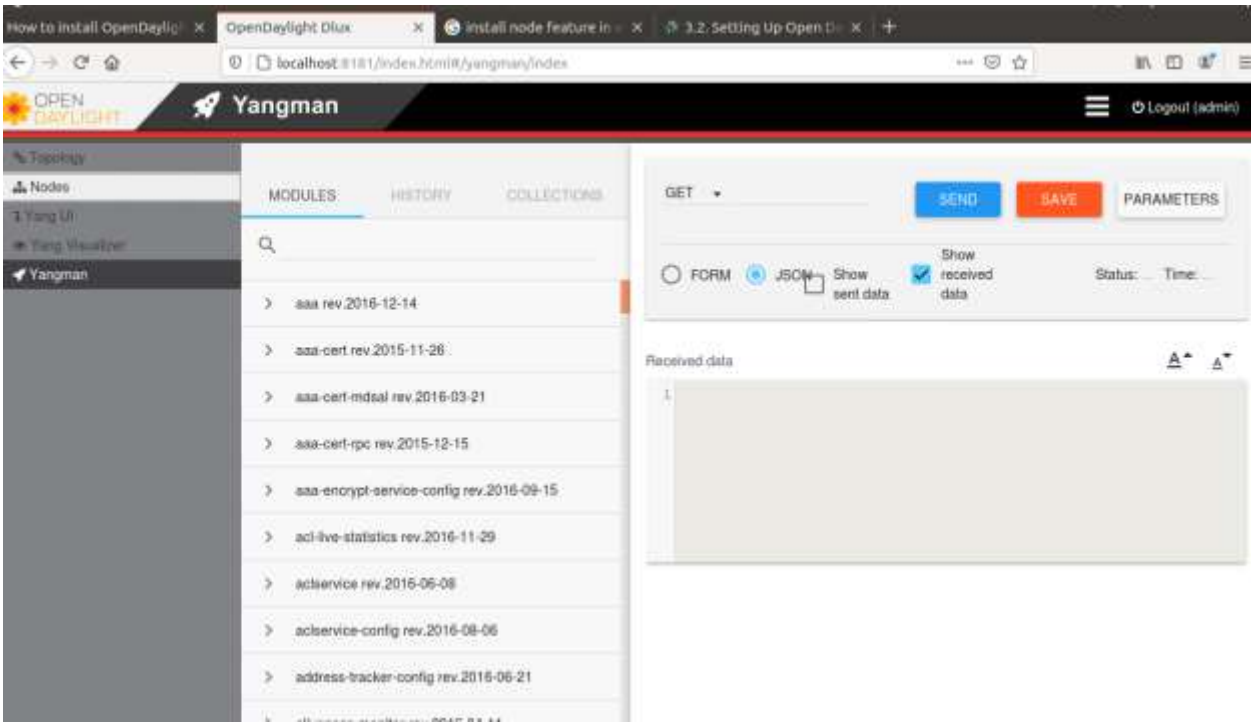
Step 13 : sudo mn --controller=remote,ip=10.1.1.93 --mac --arp



Start the opendaylight(odl)

Step 14 : http://localhost:8181/index.html#/node/index




OpenDaylight Dlux
localhost:8181/index.html#/yangman/index
Yangman
Logout (admin)
Topology
Nodes
Yang UI
Yang Visualizer
Yangman
MODULES
HISTORY
COLLECTIONS
aaa rev.2016-12-14
aaa-cert rev.2015-11-26
aaa-cert-mdsal rev.2016-03-21
aaa-cert-rpc rev.2015-12-15
aaa-encrypt-service-config rev.2016-09-15
acl-live-statistics rev.2016-11-29
aclservice rev.2016-06-08
aclservice-config rev.2016-08-06
address-tracker-config rev.2016-06-21
GET
SEND
SAVE
PARAMETERS
FORM
JSON
Show sent data
Show received data
Status:
Time:
Received data

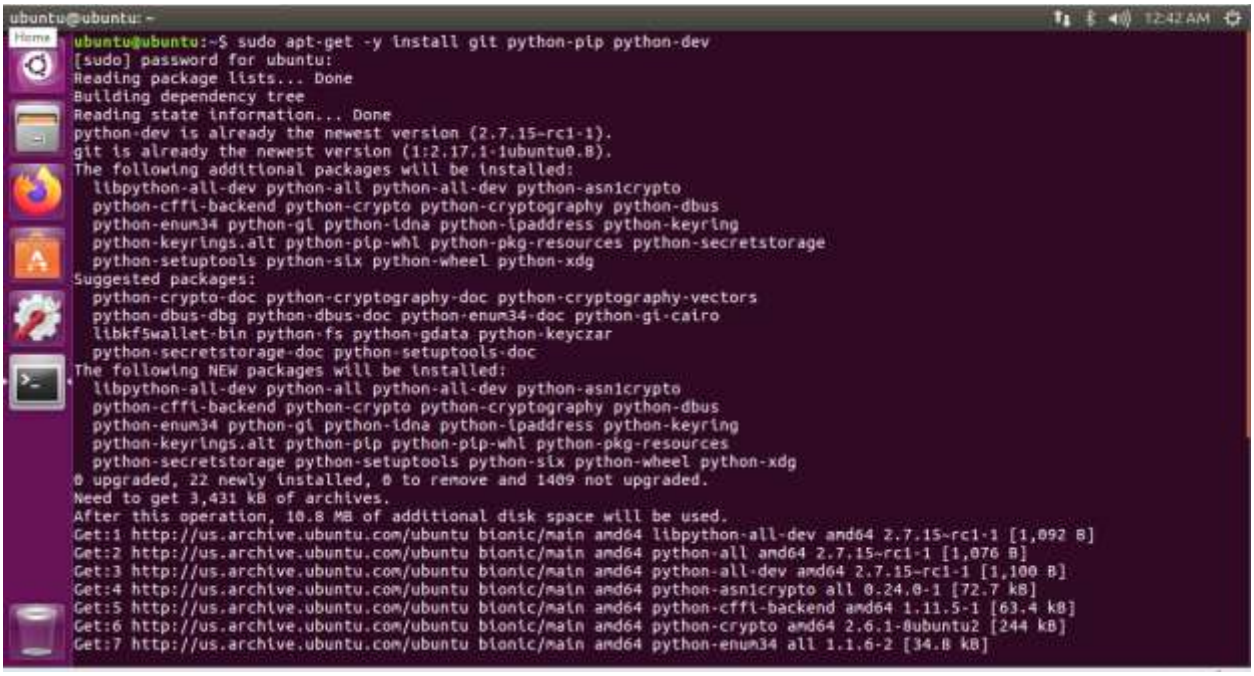
PRACTICAL NO: 06

Install RYU controller with mininet topology

It is one of the SDN controller specially designed for the agility of the network and for managing the higher traffic rate. Ryu includes well-defined software components along with API. Ryu makes the developers develop a new application and manage various other networking devices.

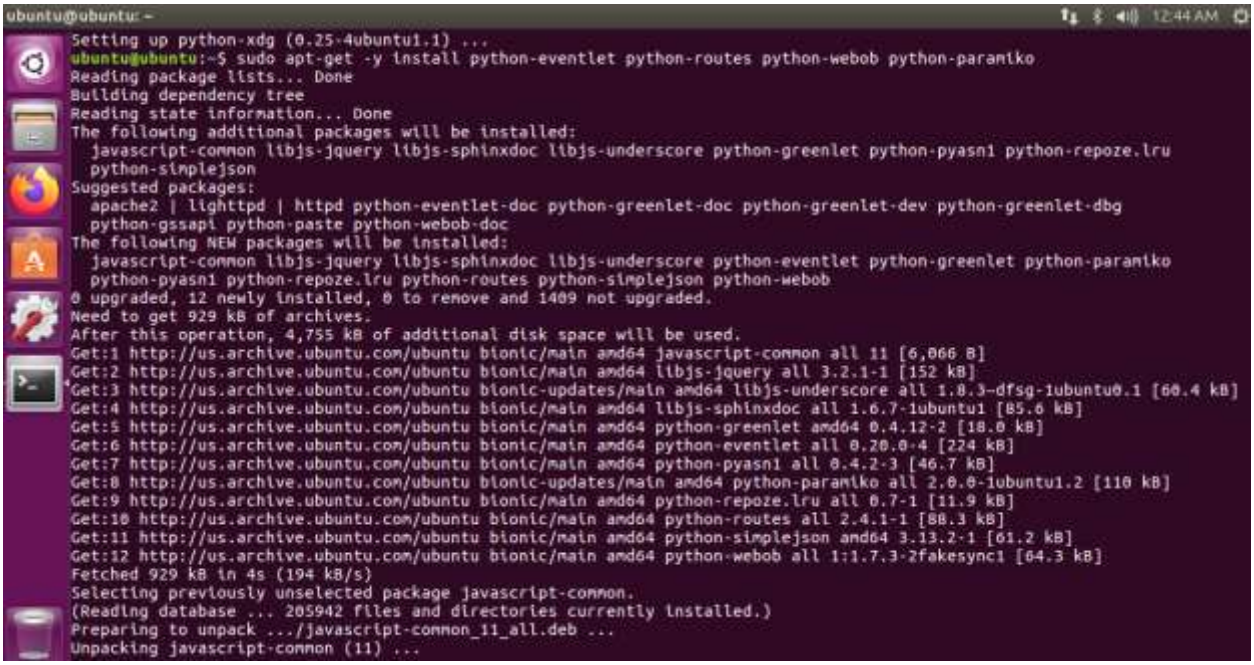
Step 1 : sudo apt-get -y install git python-pip python-dev

Install tools



Step 2 : sudo apt-get -y install python-eventlet python-routes python-webob python-paramiko

Install python packages

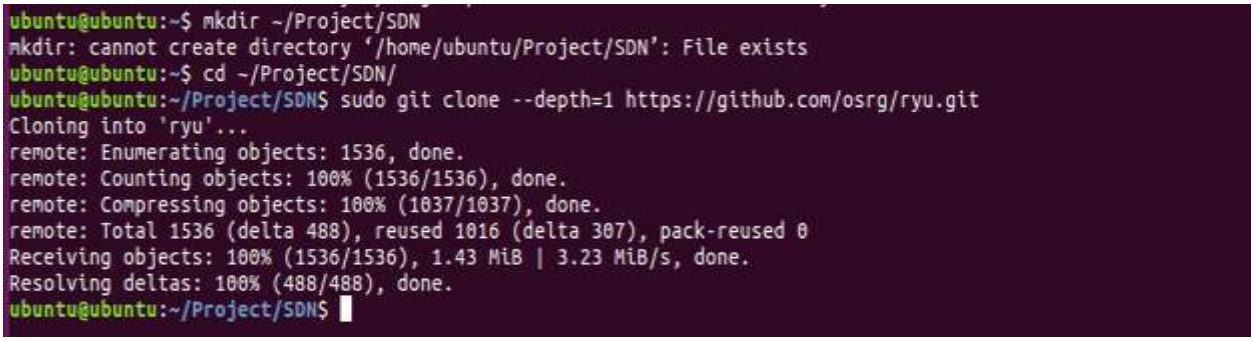


Step 3 : mkdir ~/Project/SDN

cd ~/Project/SDN/

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

Create a directory and Clone RYU git Repository



Step 4 : sudo pip install setuptools –upgrade

cd ryu;

sudo python ./setup.py install

Install RYU

```
ubuntu@ubuntu:~/Project/SDN$ sudo pip install setuptools --upgrade
The directory '/home/ubuntu/.cache/pip/http' or its parent directory is not owned by the current user and the cache has
been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's
 -H flag.
The directory '/home/ubuntu/.cache/pip' or its parent directory is not owned by the current user and caching wheels has
been disabled. check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H fla
g.
Collecting setuptools
  Downloading https://files.pythonhosted.org/packages/e1/b7/182161210a13158cd3ccc41ee19aadef54496b74f2817cc147006ec932b4
/setuptools-44.1.1-py2.py3-none-any.whl (583kB)
    100% |████████████████████████████████| 583kB 405kB/s
Installing collected packages: setuptools
  Found existing installation: setuptools 39.0.1
    Not uninstalling setuptools at /usr/lib/python2.7/dist-packages, outside environment /usr
Successfully installed setuptools-44.1.1
ubuntu@ubuntu:~/Project/SDN$
```

```
ubuntu@ubuntu: ~/Project/SDN/ryu                                                    12:52 AM
Successfully installed setuptools-44.1.1
ubuntu@ubuntu:~/Project/SDN$ cd ryu;
ubuntu@ubuntu:~/Project/SDN/ryu$ sudo python ./setup.py install
running install
[pbr] Writing ChangeLog
[pbr] Generating ChangeLog
[pbr] ChangeLog complete (0.0s)
[pbr] Generating AUTHORS
[pbr] AUTHORS complete (0.0s)
running build
running build_py
creating build
creating build/lib.linux-x86_64-2.7
creating build/lib.linux-x86_64-2.7/ryu
creating build/lib.linux-x86_64-2.7/ryu/services
creating build/lib.linux-x86_64-2.7/ryu/services/protocols
creating build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/protocol.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/net_ctrl.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/speaker.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/base.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/model.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/bmp.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/__init__.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/constants.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/peer.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/application.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/bgpspeaker.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/core.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/core_manager.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/bgp_sample_conf.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
copying ryu/services/protocols/bgp/processor.py -> build/lib.linux-x86_64-2.7/ryu/services/protocols/bgp
creating build/lib.linux-x86_64-2.7/ryu/topology
copying ryu/topology/api.py -> build/lib.linux-x86_64-2.7/ryu/topology
```

```
ubuntu@ubuntu: ~/Project/SDN/ryu                                                    12:53 AM
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/xflow/netflow.py to netflow.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/xflow/__init__.py to __init__.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/xflow/sflow.py to sflow.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/snortlib.py to snortlib.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/type_desc.py to type_desc.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/stringify.py to stringify.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/ovs/vsctl.py to vsctl.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/ovs/bridge.py to bridge.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/ovs/db_client.py to db_client.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/ovs/__init__.py to __init__.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/ovs/vswitch_idl.py to vswitch_idl.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/lp.py to lp.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/netconf/__init__.py to __init__.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/netconf/constants.py to constants.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/netdevice.py to netdevice.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/ofctl_v1_2.py to ofctl_v1_2.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/lib/stplib.py to stplib.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/base/__init__.py to __init__.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/ryu/base/app_manager.py to app_manager.pyc
running install_data
creating /usr/local/etc/ryu
copying etc/ryu/ryu.conf -> /usr/local/etc/ryu
running install_egg_info
Copying ryu.egg-info to /usr/local/lib/python2.7/dist-packages/ryu-4.34-py2.7.egg-info
running install_scripts
/usr/local/lib/python2.7/dist-packages/setuptools/command/easy_install.py:2088: EasyInstallDeprecationWarning: Use get_a
rgs
  warnings.warn("Use get_args", EasyInstallDeprecationWarning)
/usr/local/lib/python2.7/dist-packages/setuptools/command/easy_install.py:2090: EasyInstallDeprecationWarning: Use get_h
eader
  header = cls.get_script_header("", executable, wininst)
Installing ryu-manager script to /usr/local/bin
Installing ryu script to /usr/local/bin
ubuntu@ubuntu:~/Project/SDN/ryu$
```

Step 5. sudo pip install six –upgrade

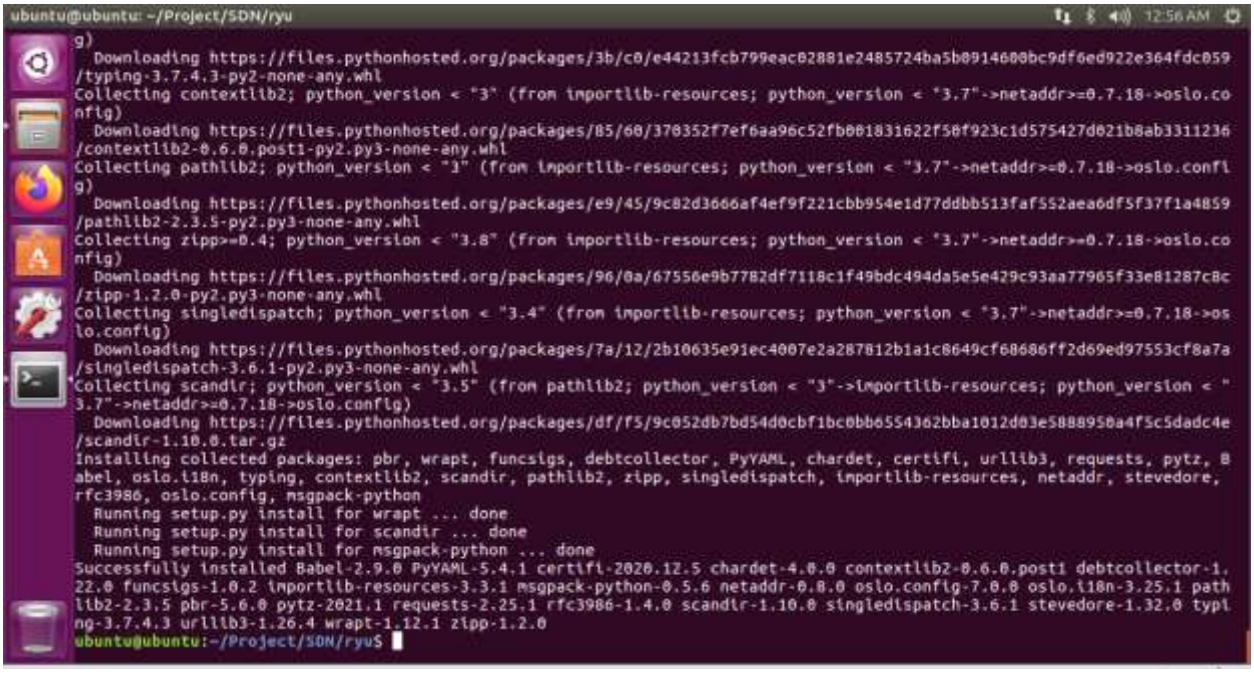
sudo pip install oslo.config msgpack-python

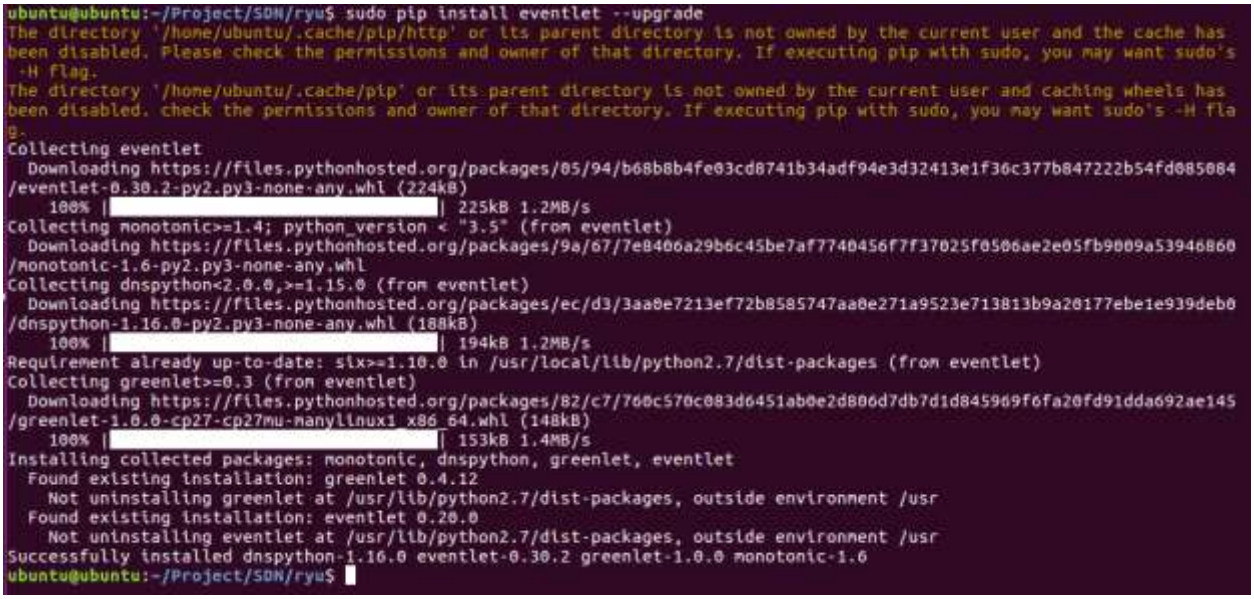
sudo pip install eventlet –upgrade

Install and Update python packages

```
ubuntu@ubuntu:~/Project/SDN/ryu$ sudo pip install six --upgrade
The directory '/home/ubuntu/.cache/pip/http' or its parent directory is not owned by the current user and the cache has
been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's
 -H flag.
The directory '/home/ubuntu/.cache/pip' or its parent directory is not owned by the current user and caching wheels has
been disabled. check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H fla
g.
Collecting six
  Downloading https://files.pythonhosted.org/packages/ee/ff/48bde5c0f013094d729fe4b0316ba2a24774b3ff1c52d924a8a4cb04078a
/six-1.15.0-py2.py3-none-any.whl
Installing collected packages: six
  Found existing installation: six 1.11.0
    Not uninstalling six at /usr/lib/python2.7/dist-packages, outside environment /usr
Successfully installed six-1.15.0
ubuntu@ubuntu:~/Project/SDN/ryu$
```

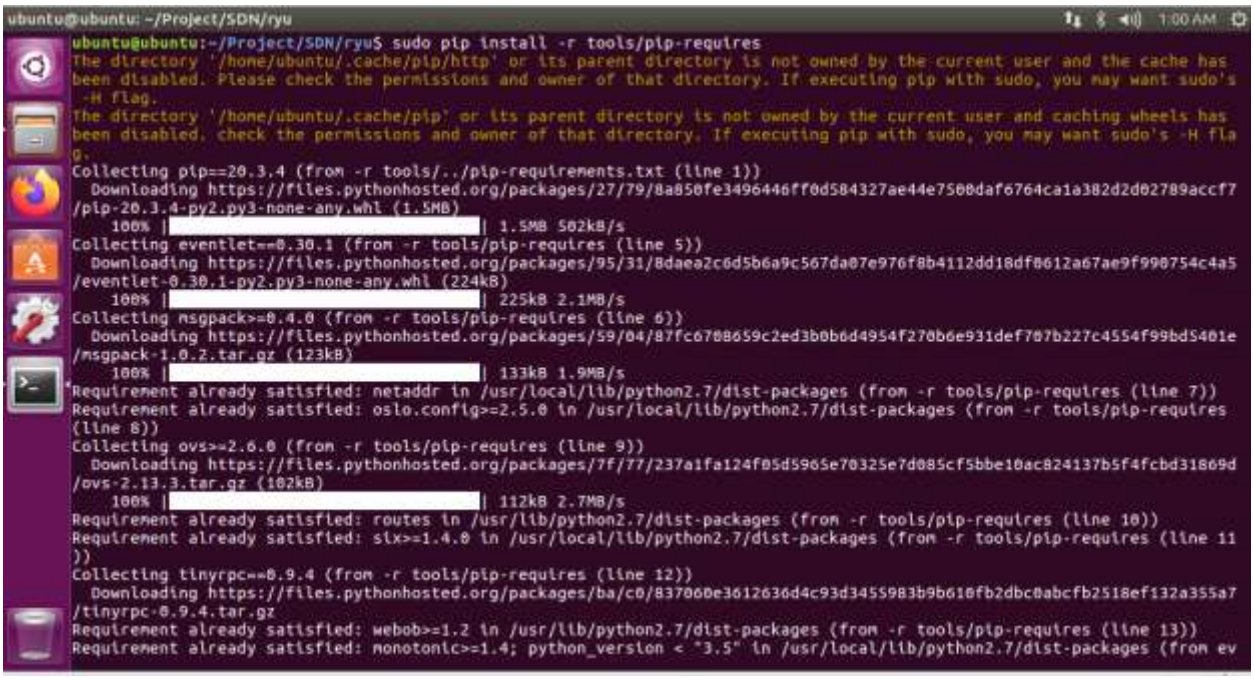
```
ubuntu@ubuntu: ~/Project/SDN/ryu                                                    12:55 AM
ubuntu@ubuntu:~/Project/SDN/ryu$ sudo pip install oslo.config msgpack-python
The directory '/home/ubuntu/.cache/pip/http' or its parent directory is not owned by the current user and the cache has
been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's
 -H flag.
The directory '/home/ubuntu/.cache/pip' or its parent directory is not owned by the current user and caching wheels has
been disabled. check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H fla
g.
Collecting oslo.config
  Downloading https://files.pythonhosted.org/packages/5c/45/53255cdbabefabc4ac8ae36671ecbd8a92a6d029363fcae9126a27fb81d7
/oslo.config-7.0.0-py2.py3-none-any.whl (125kB)
    100% |████████████████████████████████| 133kB 603kB/s
Collecting msgpack-python
  Downloading https://files.pythonhosted.org/packages/8a/20/6eca772d1a5830336f84aca1d8198e5a3f4715cd1c7fc36d3cc7f7185091
/msgpack-python-0.5.6.tar.gz (138kB)
    100% |████████████████████████████████| 143kB 1.2MB/s
Requirement already satisfied: enum34>=1.0.4; python_version == "2.7" or python_version == "2.6" or python_version == "3
.3" in /usr/lib/python2.7/dist-packages (from oslo.config)
Collecting debtcollector>=1.2.0 (from oslo.config)
  Downloading https://files.pythonhosted.org/packages/ca/ab/e34b13877f84a198b043166a82baf0ae8b9ed1daa83b6ebde776e8628b0a
/debtcollector-1.22.0-py2.py3-none-any.whl
Collecting PyYAML>=3.12 (from oslo.config)
  Downloading https://files.pythonhosted.org/packages/ba/d4/3cf562876e0cda0405e65d351b835077ab13990e5b92912ef2bf1a2280e0
/PyYAML-5.4.1-cp27-cp27mu-manylinux1_x86_64.whl (574kB)
    100% |████████████████████████████████| 583kB 624kB/s
Collecting requests>=2.18.0 (from oslo.config)
  Downloading https://files.pythonhosted.org/packages/29/c1/24814557f1d22c56d50280771a17307e6bf87b70727d975fd6b2ce6b014a
/requests-2.25.1-py2.py3-none-any.whl (61kB)
    100% |████████████████████████████████| 61kB 2.8MB/s
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python2.7/dist-packages (from oslo.config)
Collecting oslo.i18n>=3.15.3 (from oslo.config)
  Downloading https://files.pythonhosted.org/packages/4e/a4/e2c71ea714266097b2edc5188462cfc8f0fffd277ef148a33430199b4a0a
/oslo.i18n-3.25.1-py2.py3-none-any.whl (47kB)
    100% |████████████████████████████████| 51kB 3.7MB/s
Collecting netaddr>=0.7.18 (from oslo.config)
```



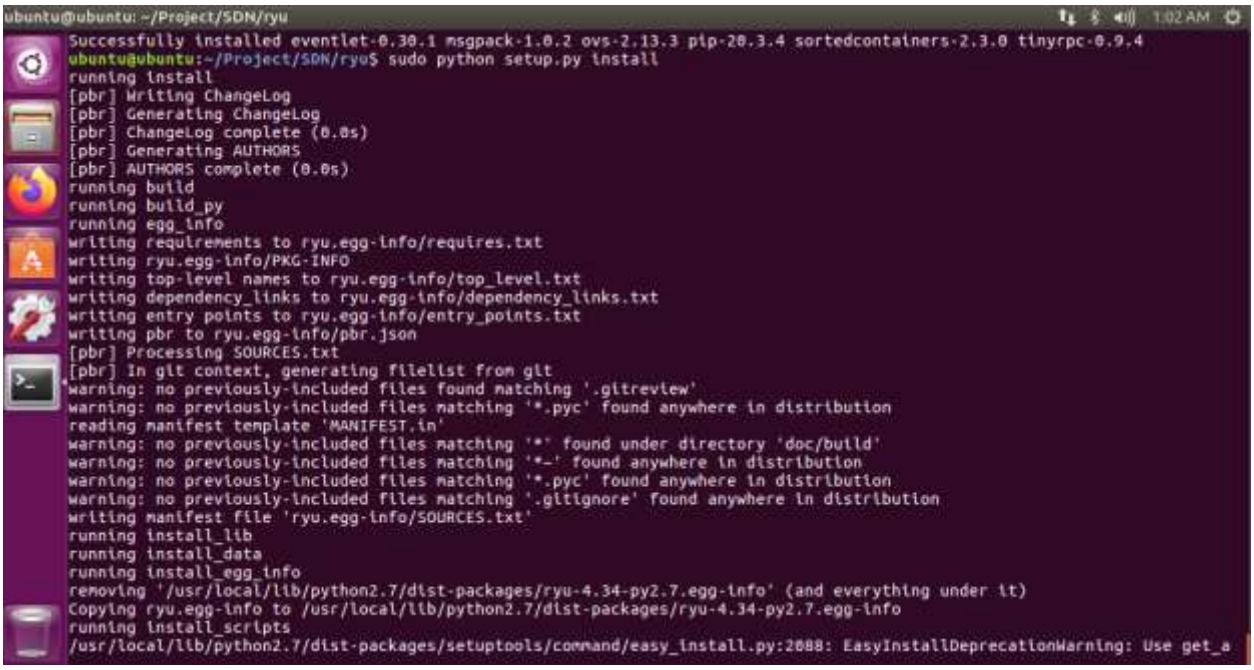


Step 6.: sudo pip install -r tools/pip-requires

Test ryu-manager

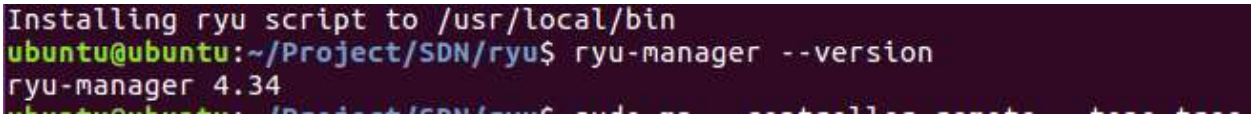


sudo python setup.py install



Step 7 : ryu-manager --version

ryu-manager –version



Topology Viewer

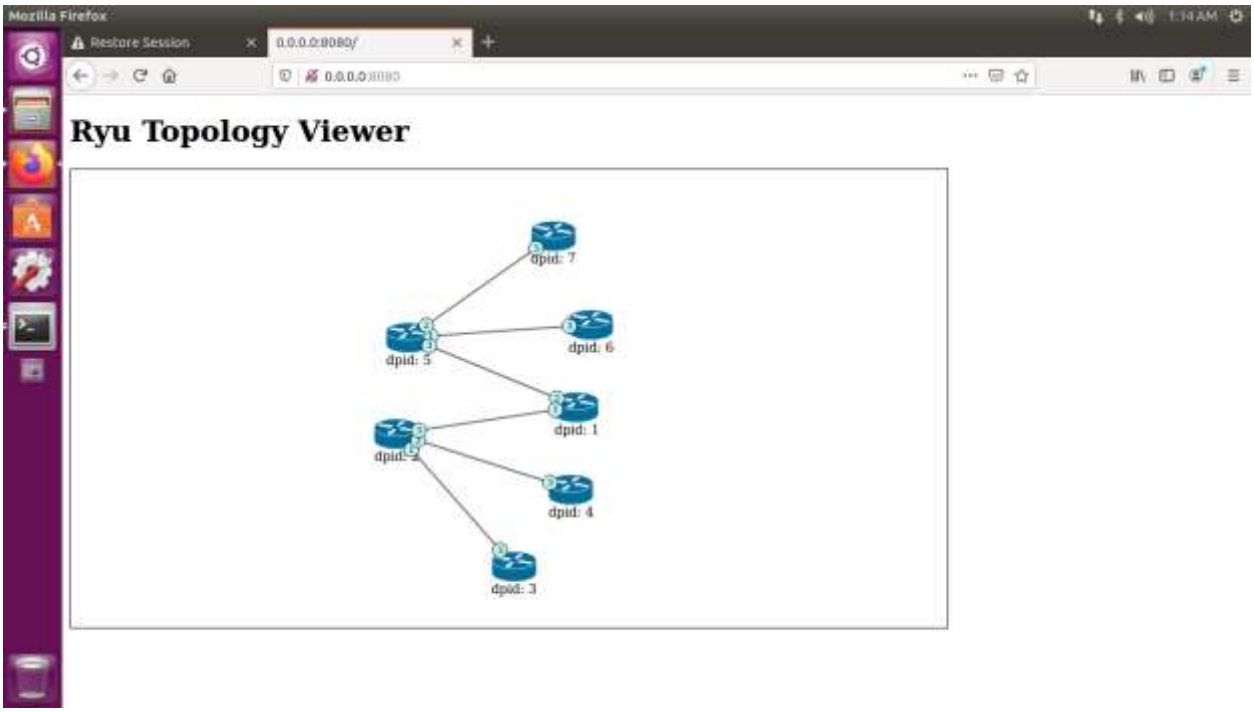
Run mininet (or join your real environment):

Step 8 : sudo mn --controller remote --topo tree,depth=3

```
ubuntu@ubuntu:~/Project/SDN/ryu$ sudo mn --controller remote --topo tree,depth=3
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7
, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 14 links
..............
*** Stopping 7 switches
s1 s2 s3 s4 s5 s6 s7
*** Stopping 8 hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Done
completed in 12.625 seconds
ubuntu@ubuntu:~/Project/SDN/ryu$
```

Step 9 : PYTHONPATH=. ./bin/ryu run --observe-links ryu/app/gui_topology/gui_topology.py

```
ubuntu@ubuntu:~/Project/SDN/ryu$ PYTHONPATH=. ./bin/ryu run --observe-links ryu/app/gui_topology/gui_topology.py
loading app ryu/app/gui_topology/gui_topology.py
loading app ryu.app.rest_topology
loading app ryu.app.ws_topology
loading app ryu.app.ofctl_rest
loading app ryu.controller.ofp_handler
instantiating app None of Switches
creating context switches
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.rest_topology of TopologyAPI
instantiating app ryu.app.ws_topology of WebSocketTopology
instantiating app ryu.app.ofctl_rest of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu/app/gui_topology/gui_topology.py of GUIServerApp
(6003) wsgi starting up on http://0.0.0.0:8080
(6003) accepted ('127.0.0.1', 59292)
127.0.0.1 - - [28/Apr/2021 01:12:29] "GET / HTTP/1.1" 304 177 0.032901
127.0.0.1 - - [28/Apr/2021 01:12:32] "GET /ryu.topology.css HTTP/1.1" 304 177 0.001521
(6003) accepted ('127.0.0.1', 59300)
127.0.0.1 - - [28/Apr/2021 01:12:32] "GET /ryu.topology.js HTTP/1.1" 304 177 0.001120
(6003) accepted ('127.0.0.1', 59302)
127.0.0.1 - - [28/Apr/2021 01:12:32] "GET /v1.0/topology/switches HTTP/1.1" 200 2501 0.002261
127.0.0.1 - - [28/Apr/2021 01:12:32] "GET /v1.0/topology/links HTTP/1.1" 200 2822 0.003103
127.0.0.1 - - [28/Apr/2021 01:12:32] "GET /router.svg HTTP/1.1" 200 3647 0.029091
```


Mozilla Firefox
Restore Session
0.0.0.0:8080/
Ryu Topology Viewer
dpid: 7
dpid: 6
dpid: 5
dpid: 1
dpid: 2
dpid: 4
dpid: 3

PRACTICAL NO: 07

Install floodlight on Ubuntu

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

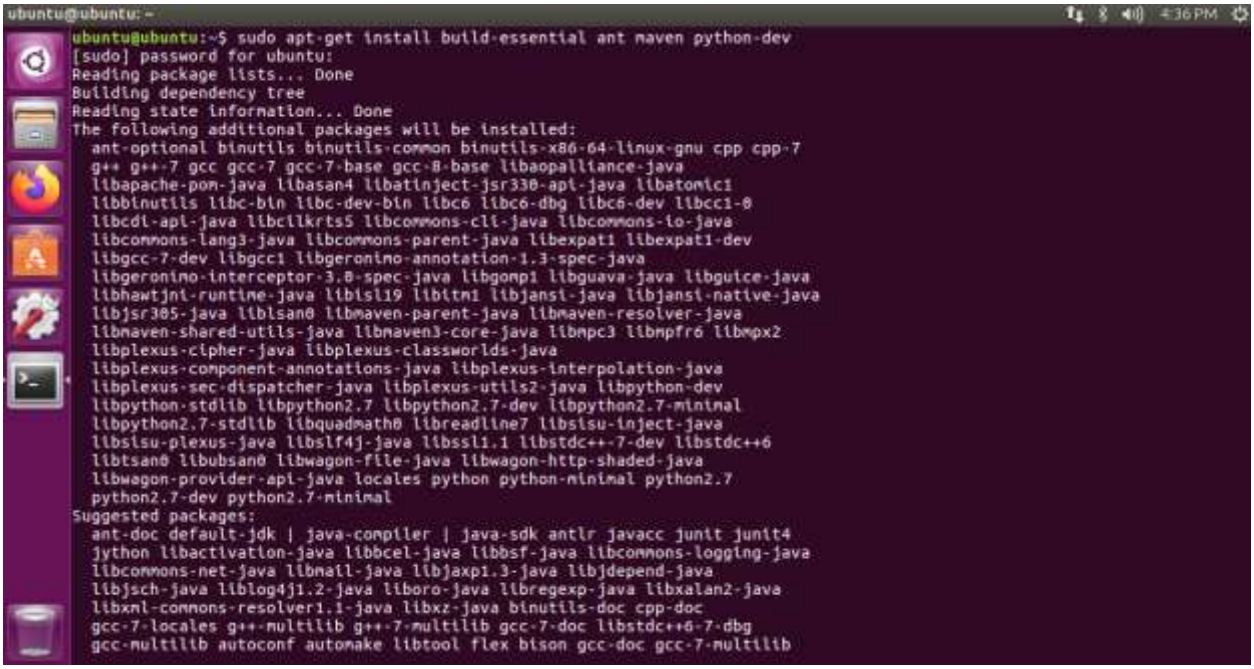
Installation:

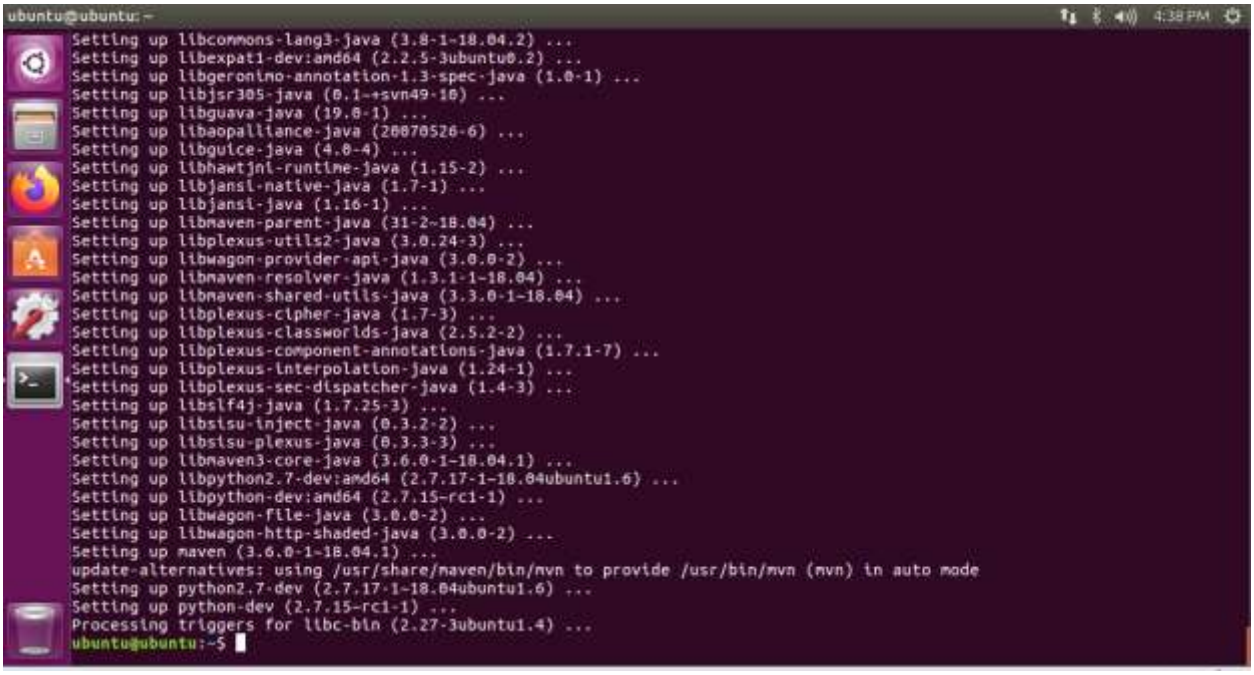
Prerequisites:

- 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:

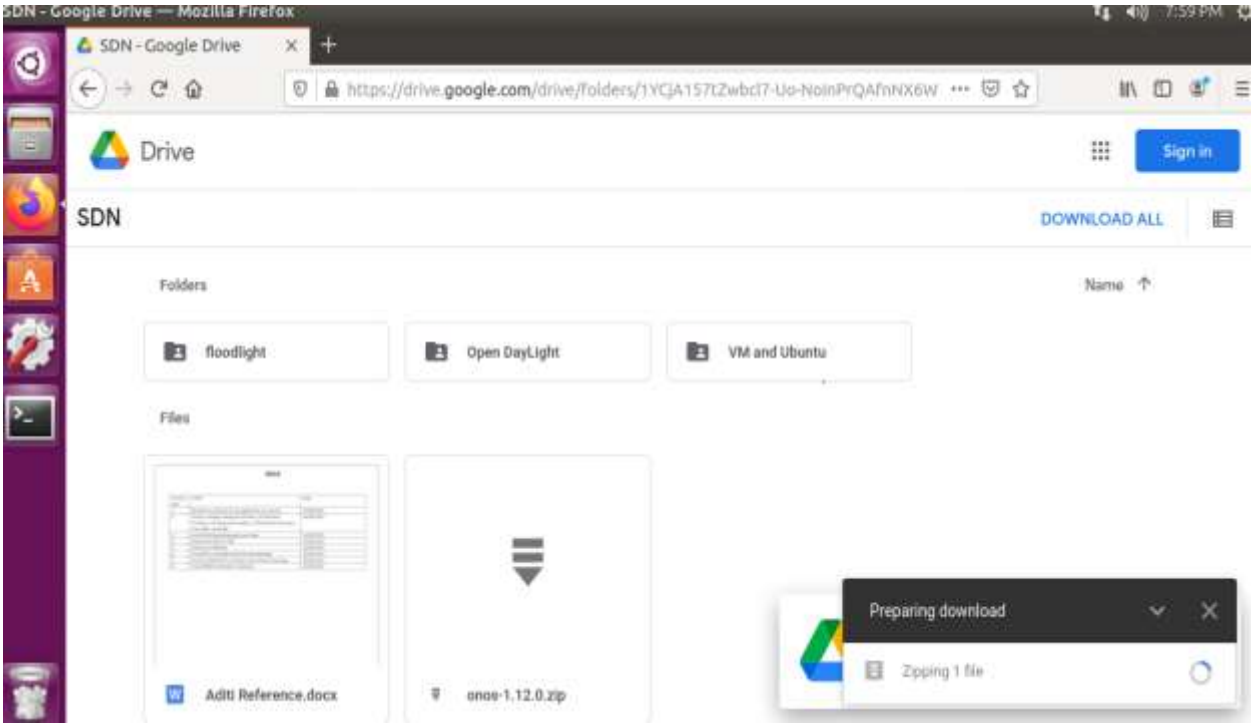
Step 1 : sudo apt-get install build-essential ant maven python-dev

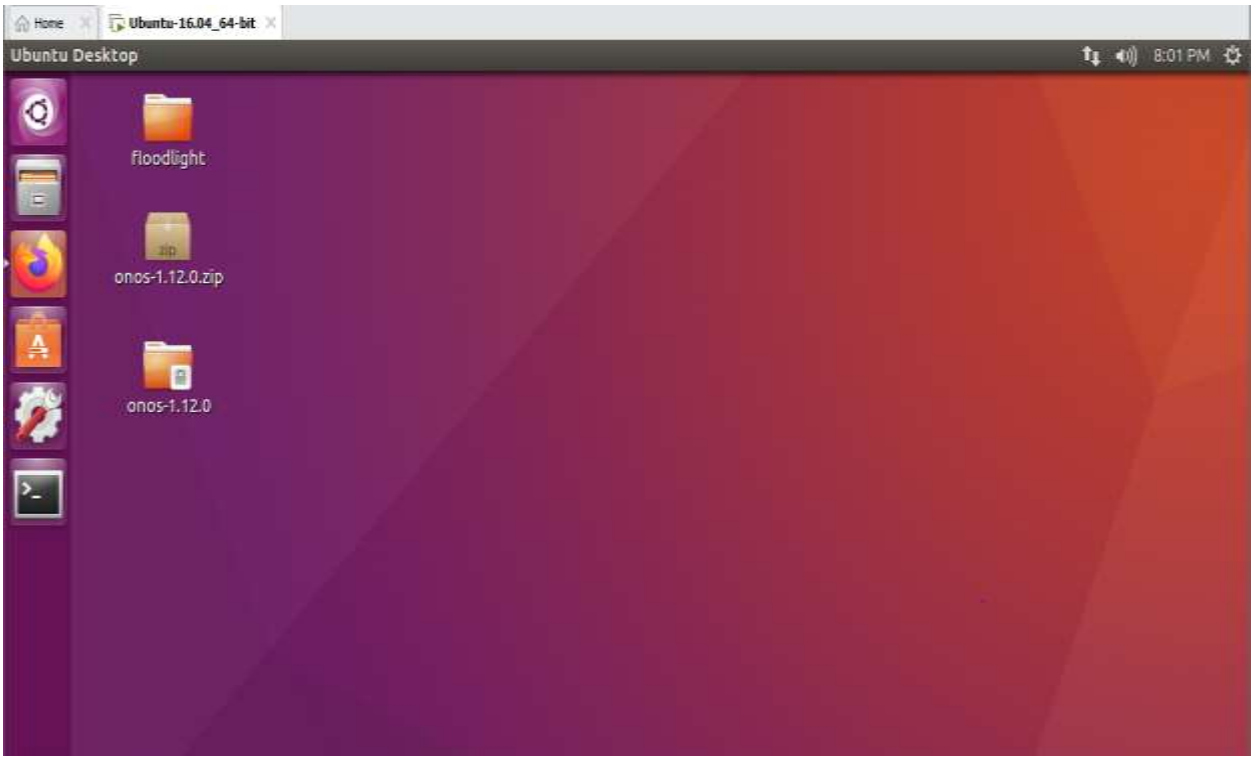




Step 2 : Download the setup file from google drive link and paste it in the desktop.

https://drive.google.com/drive/folders/1YCjA157tZwbcl7-Uo-NoInPrQAfnNX6W


SDN - Google Drive — Mozilla Firefox
SDN - Google Drive
https://drive.google.com/drive/folders/1YCjA157tZwbcl7-Uo-NoinPrQAfnNX6W
Drive
Sign in
SDN
DOWNLOAD ALL
Folders
Name
floodlight
Open DayLight
VM and Ubuntu
Files
Aditi Reference.docx
onos-1.12.0.zip
Preparing download
Zipping 1 file


Home
Ubuntu-16.04_64-bit
Ubuntu Desktop
8:01 PM
floodlight
onos-1.12.0.zip
onos-1.12.0

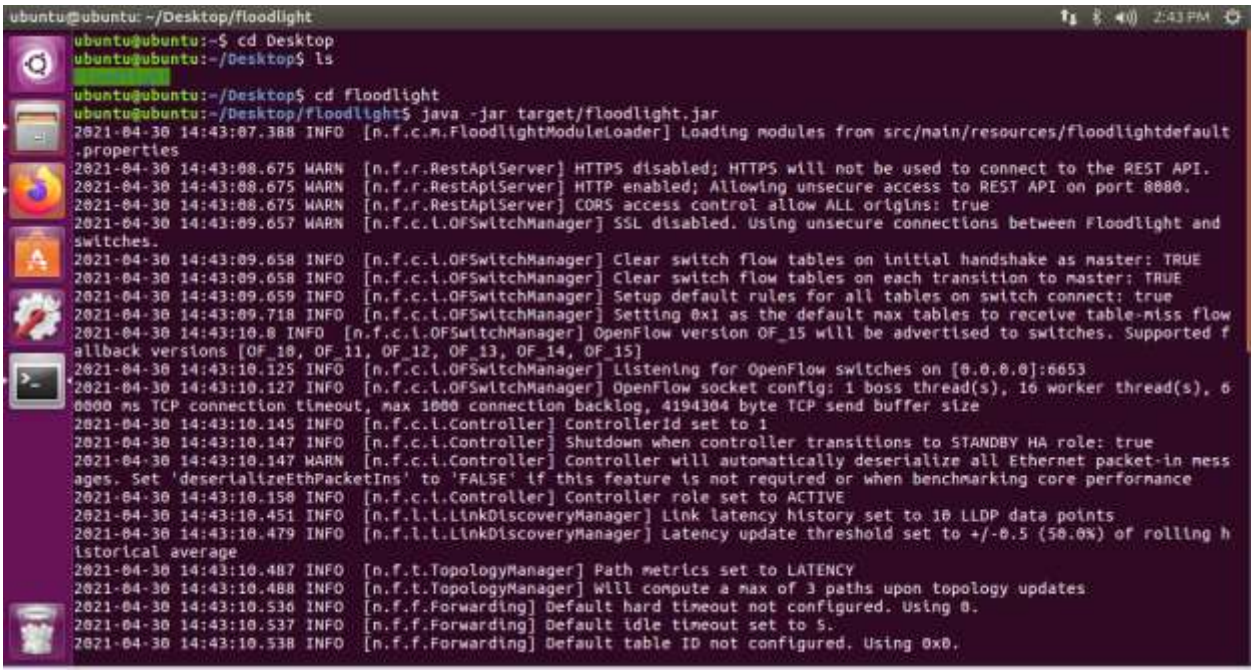
After installation complete follow the steps to build floodlight controller.

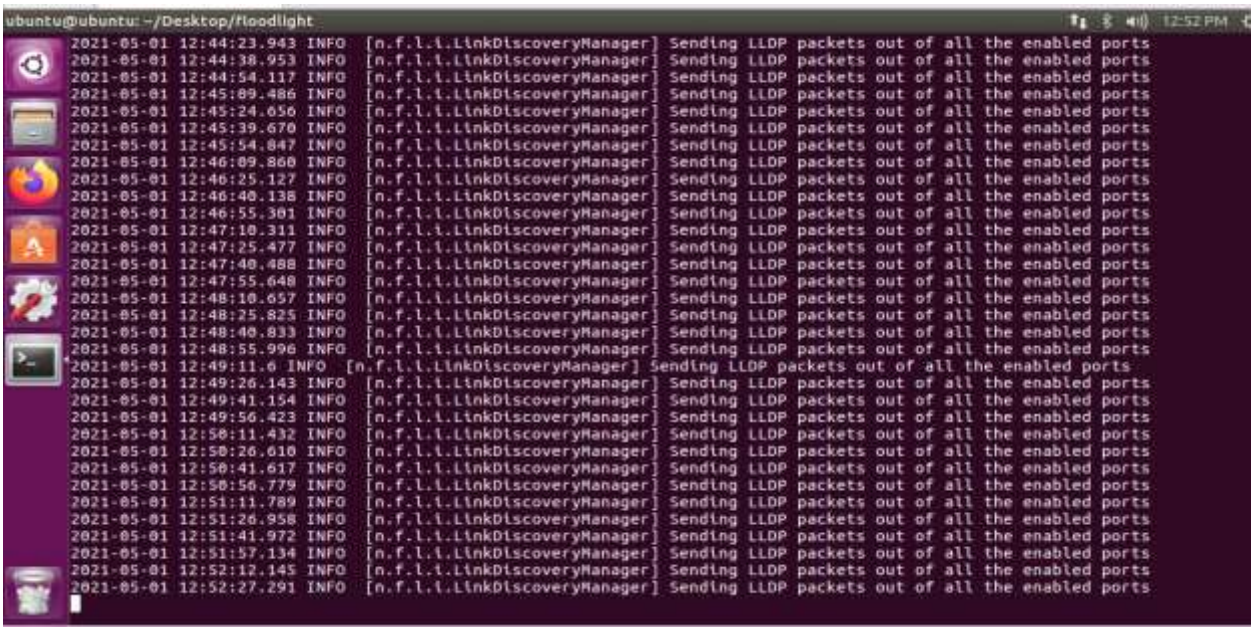
Step 3 : cd Desktop

Step 4 : cd floodlight

Assuming java is in your path, you can directly run the floodlight.jar file produced by ant from within the floodlight directory

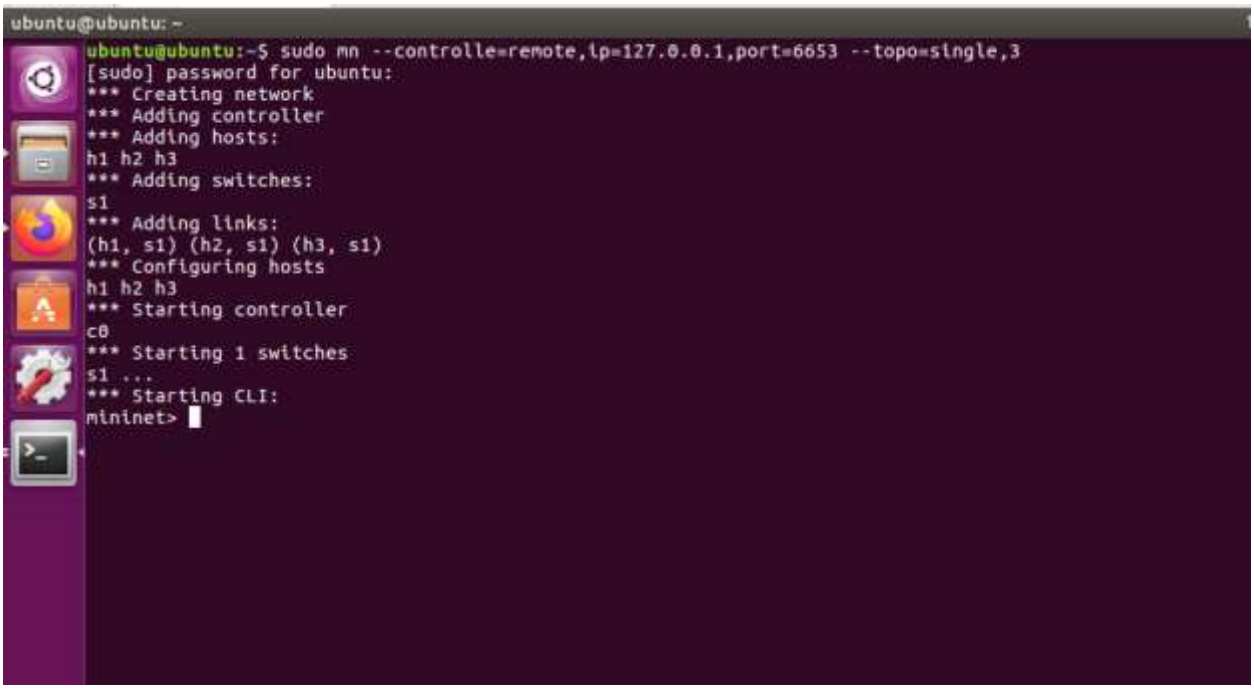
Step 5 : java -jar target/floodlight.jar





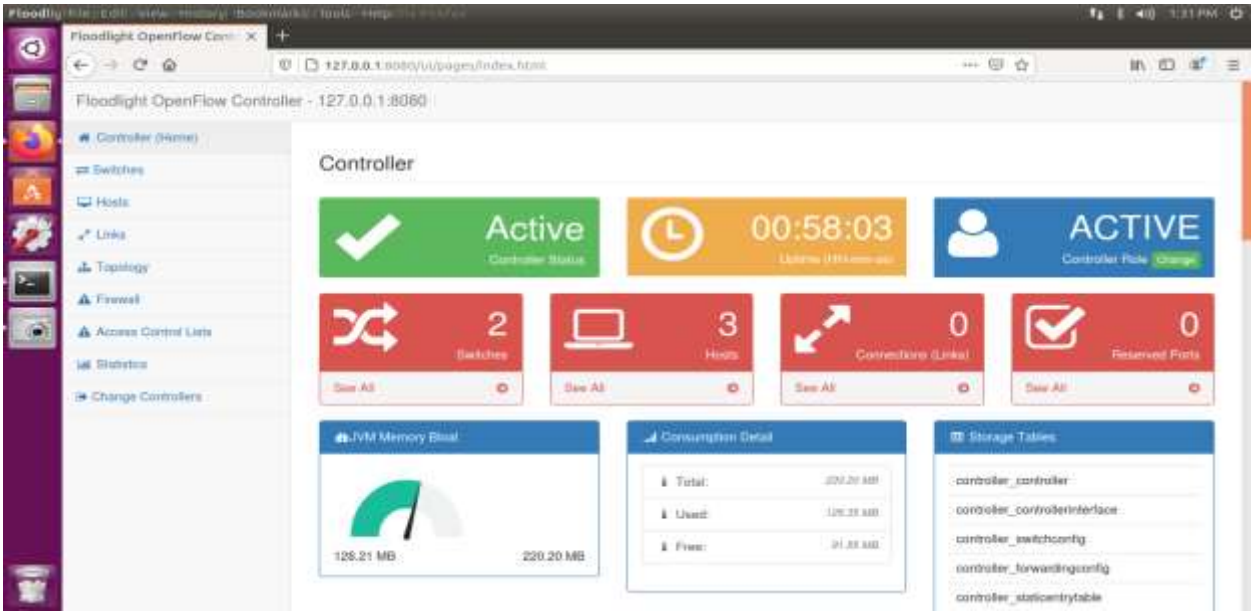
To Create mininet topology with floodlight remote controller

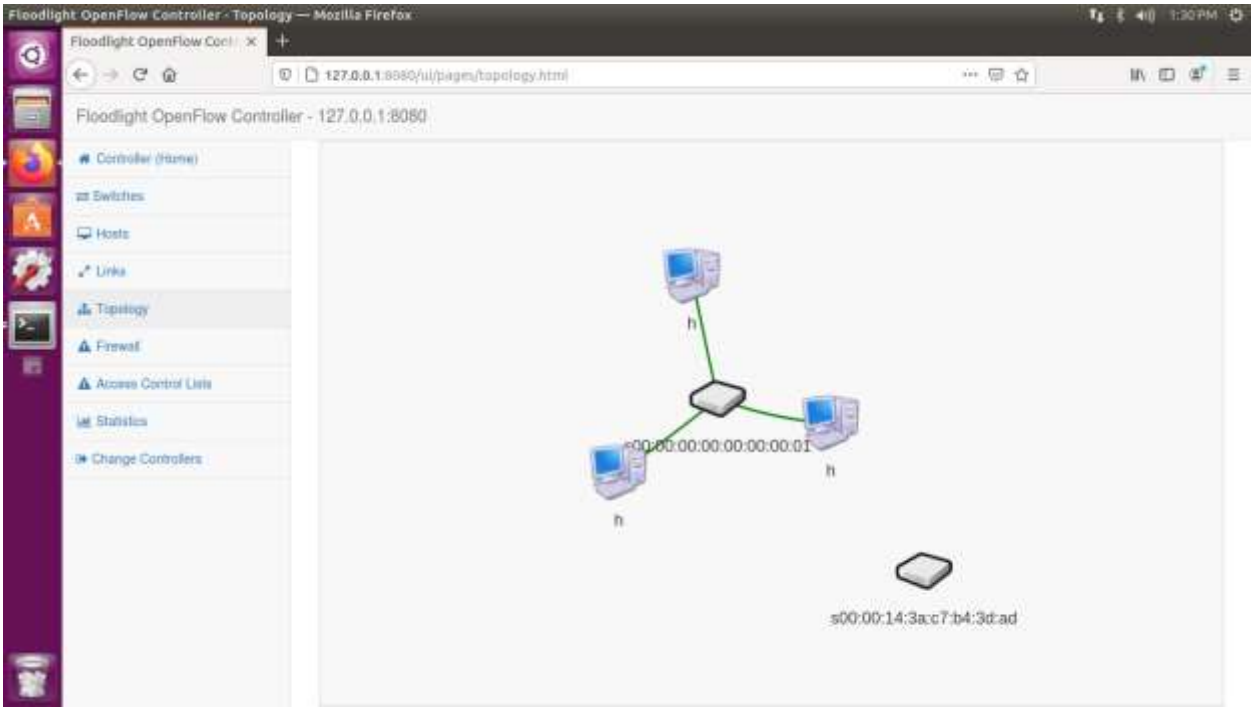
Step 5 : sudo mn --controlle=remote,ip=127.0.0.1,port=6653 --topo=single,3



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,

Step 6 : http://127.0.0.1:8080/ui/index.html




Floodlight OpenFlow Controller - Topology — Mozilla Firefox
Floodlight OpenFlow Controller - 127.0.0.1:8080
Controller (Home)
Switches
Hosts
Links
Topology
Firewall
Access Control Lists
Statistics
Change Controllers
h
h
h
s00:00:14:3a:c7:b4:3d:ad

PRACTICAL NO: 08

Install ONOS controller on Ubuntu

Onos provides the control planes for software defined network managing network computer such as switches and l,inks and running software programs or modules to provide communication services to end hosts and neighbour networks

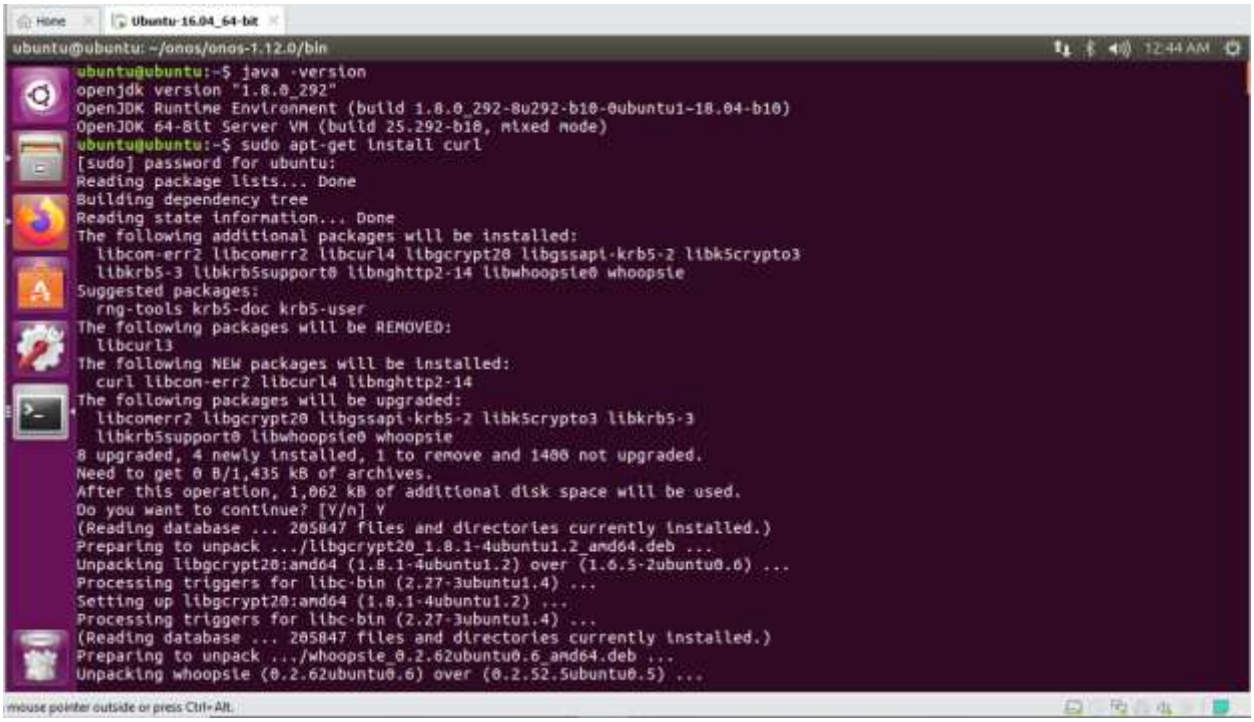
The ONOS platform includes:

- 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.

Check java version and set it to java 8

Step 1 : java -version

Step 2 : sudo apt-get install curl



Download the onos file

Step 3 : wget https://repo1.maven.org/maven2/org/onosproject/onos-releases/onos-1.12.0/onos-1.12.0.tar.gz

```
ubuntu@ubuntu:~$ wget https://repo1.maven.org/maven2/org/onosproject/onos-releases/onos-1.12.0/onos-1.12.0.tar.gz
--2021-04-29 23:05:20--  https://repo1.maven.org/maven2/org/onosproject/onos-releases/onos-1.12.0/onos-1.12.0.tar.gz
Resolving repo1.maven.org (repo1.maven.org)... 151.101.196.209
Connecting to repo1.maven.org (repo1.maven.org)|151.101.196.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 225129387 (215M) [application/x-gzip]
Saving to: 'onos-1.12.0.tar.gz'

onos-1.12.0.tar.gz  100%[===================>] 214.70M  4.83MB/s    in 61s

2021-04-29 23:06:28 (3.50 MB/s) - 'onos-1.12.0.tar.gz' saved [225129387/225129387]

ubuntu@ubuntu:~$ ls
```

Step 4 : ls

Step 5 : mv onos-1.12.0 onos

Step 6 : cd onos/

```
ubuntu@ubuntu:~$ ls
Desktop    examples.desktop  onos-1.12.0        Public
Documents  [redacted]        onos-1.12.0.tar.gz  Templates
Downloads  Music             Pictures           Videos
ubuntu@ubuntu:~$ mv onos-1.12.0 onos
ubuntu@ubuntu:~$ cd onos/
ubuntu@ubuntu:~/onos$ ls
```

Step 7 : sudo unzip /home/ubuntu/onos/onos-1.12.0.zip -d /home/ubuntu/onos/

```
ubuntu@ubuntu: ~/onos/onos-1.12.0/bin
ubuntu@ubuntu:~/onos$ sudo unzip /home/ubuntu/onos/onos-1.12.0.zip -d /home/ubuntu/onos/
Archive:  /home/ubuntu/onos/onos-1.12.0.zip
   creating: /home/ubuntu/onos/onos-1.12.0/
   creating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/
   creating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/client
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/client.bat
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/instance
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/instance.bat
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/karaf
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/karaf.bat
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/setenv
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/setenv.bat
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/shell
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/shell.bat
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/start
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/start.bat
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/status
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/status.bat
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/stop
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/bin/stop.bat
   creating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/data/
   creating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/data/tmp/
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/data/tmp/README
   creating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/deploy/
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/deploy/README
   creating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/all.policy
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/config.properties
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/custom.properties
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/distribution.info
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/equinox-debug.properties
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/java.util.logging.properties
  inflating: /home/ubuntu/onos/onos-1.12.0/apache-karaf-3.0.8/etc/jmx.acl.cfg
```

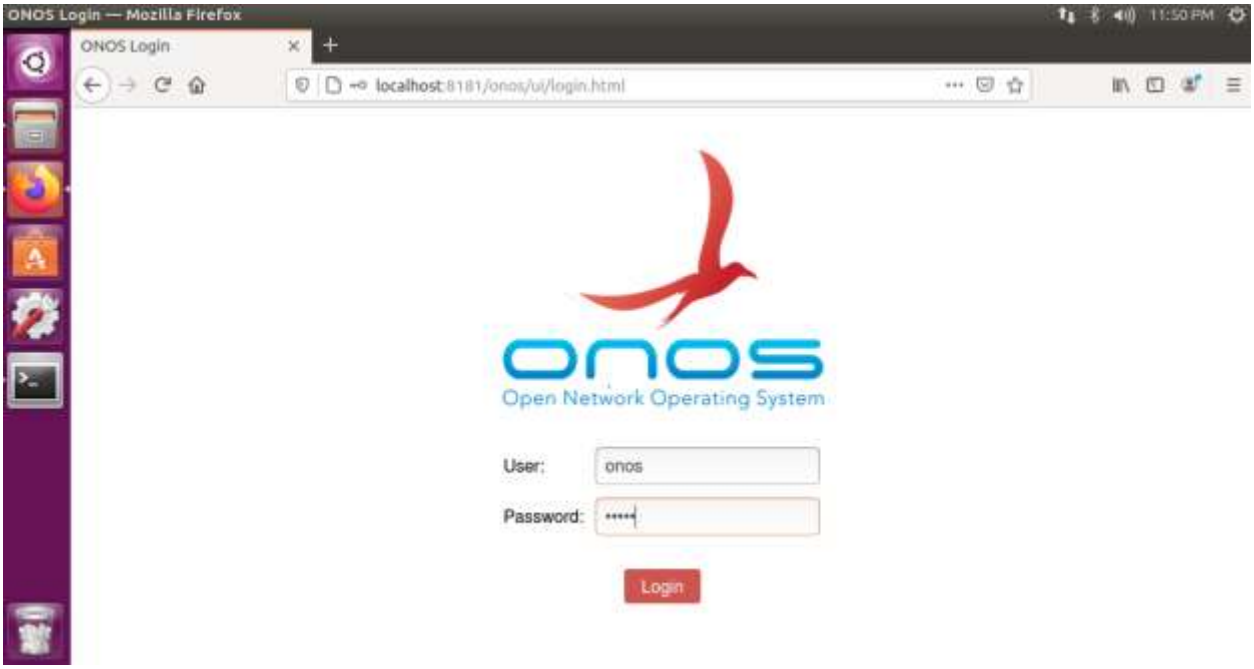
Run onos-service file in bin placed directory of onos

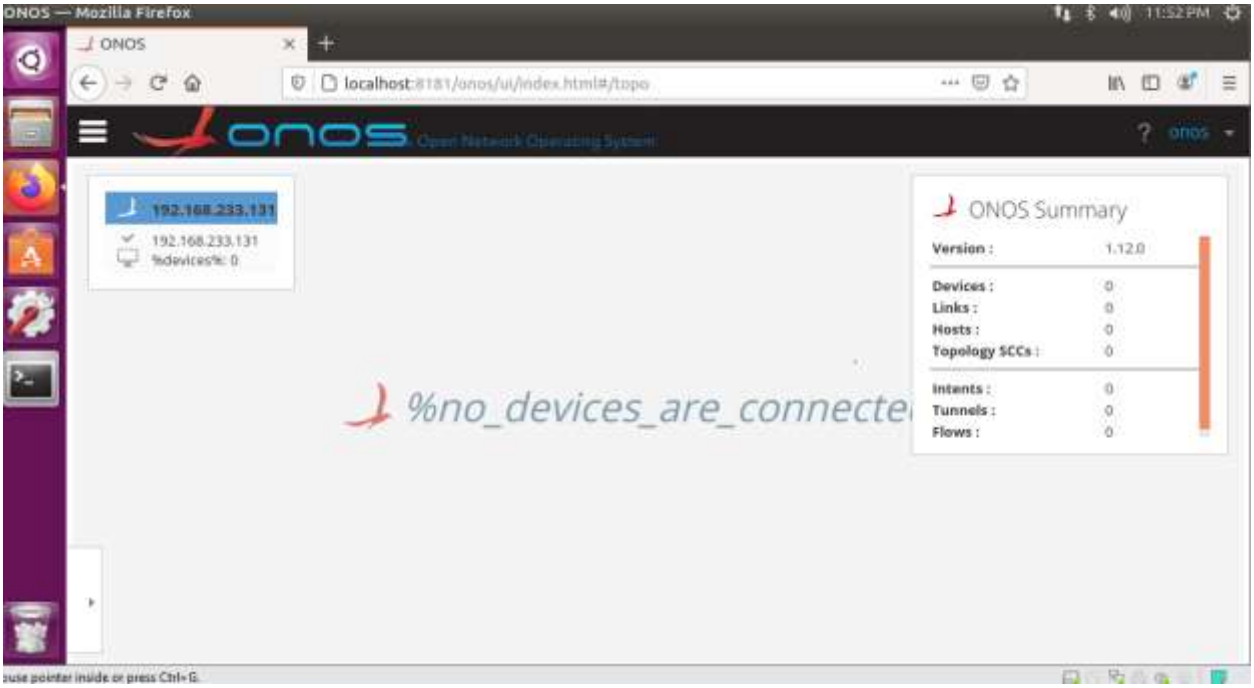
Step 8 : cd onos-1.12.0/bin

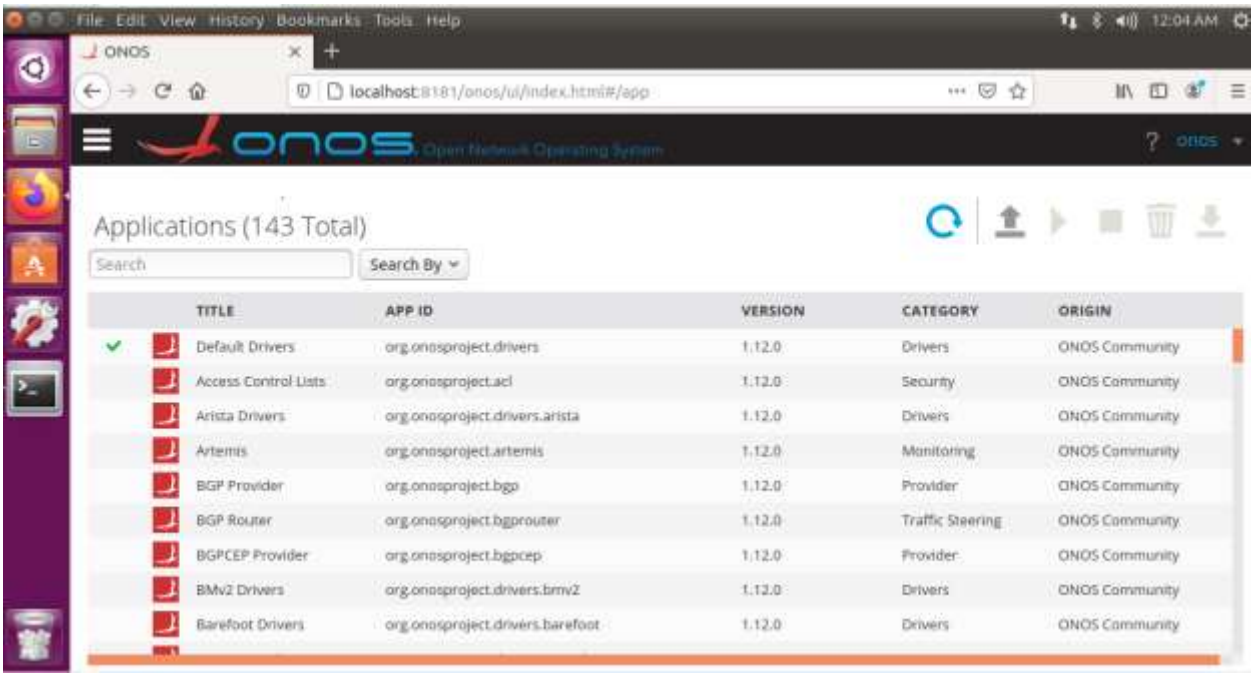
Step 9 : sudo ./onos-service

```
ubuntu@ubuntu:~/onos$ cd onos-1.12.0/bin
ubuntu@ubuntu:~/onos/onos-1.12.0/bin$ ls
_check-json  onos-app     onos-compile-yang  onos-netcfg   onos-user-key
_find-node   onos-backup  onos-form-cluster  onos-restore  onos-user-password
onos         onos-cfg     onos-jpenable      onos-service
ubuntu@ubuntu:~/onos/onos-1.12.0/bin$ .onos-service
.onos-service: command not found
ubuntu@ubuntu:~/onos/onos-1.12.0/bin$ ./onos-service
touch: cannot touch '/home/ubuntu/onos/onos-1.12.0/apps/org.onosproject.drivers/active': Permission denied
ubuntu@ubuntu:~/onos/onos-1.12.0/bin$ sudo ./onos-service
Welcome to Open Network Operating System (ONOS)!

Documentation: wiki.onosproject.org
Tutorials:     tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown ONOS.
```

Login on browser username : onos & password : rocks




ONOS — Mozilla Firefox
ONOS
localhost:8181/onos/ui/index.html#/topo
Open Network Operating System
192.168.233.131
192.168.233.131
%devices%: 0
ONOS Summary
Version : 1.12.0
Devices : 0
Links : 0
Hosts : 0
Topology SCCs : 0
Intents : 0
Tunnels : 0
Flows : 0
%no_devices_are_connecte


File Edit View History Bookmarks Tools Help
ONOS
localhost:8181/onos/ui/index.html#/app
Open Network Operating System
Applications (143 Total)
Search
Search By
TITLE
APP ID
VERSION
CATEGORY
ORIGIN
Default Drivers org.onosproject.drivers 1.12.0 Drivers ONOS Community
Access Control Lists org.onosproject.acl 1.12.0 Security ONOS Community
Arista Drivers org.onosproject.drivers.arista 1.12.0 Drivers ONOS Community
Artemis org.onosproject.artemis 1.12.0 Monitoring ONOS Community
BGP Provider org.onosproject.bgp 1.12.0 Provider ONOS Community
BGP Router org.onosproject.bgprouter 1.12.0 Traffic Steering ONOS Community
BGPCEP Provider org.onosproject.bgpcep 1.12.0 Provider ONOS Community
BMv2 Drivers org.onosproject.drivers.bmv2 1.12.0 Drivers ONOS Community
Barefoot Drivers org.onosproject.drivers.barefoot 1.12.0 Drivers ONOS Community