Practical - 1
Define a simple services like Converting Rs into Dollar and Call it from different platform like JAVA and .NET

1] create a new project and create a new webservice

2] now right clickon the code and clickon insert code > add new web service operation

3] after give name to operation **FtoC**and add one parameter **a** of double data type and click ok the code willauto generate .

4] now do the followingchanges in that auto generated code:

```
 */
@WebMethod(operationName = "FtoC")
public String FtoC(@WebParam(name = "a") double a) {
    //TODO write your implementation code here:
    return "The Fahrenheit Temperature "+a+" in Celsius is "+((a-32)*5/9);
}
```

Now deploy the model.

5] After deploying the model just right clickon your web service and clickon test web service . and you willget redirected to a browser page where you can check that your web service is working or not.

Creating java Clientusing jsp

5] now right clickon web pages > new > jsp . and give name as input jsp.

Againright clickon web pages > new > jsp . and give name as output jsp.

6]In input.jsp write the followingcode :

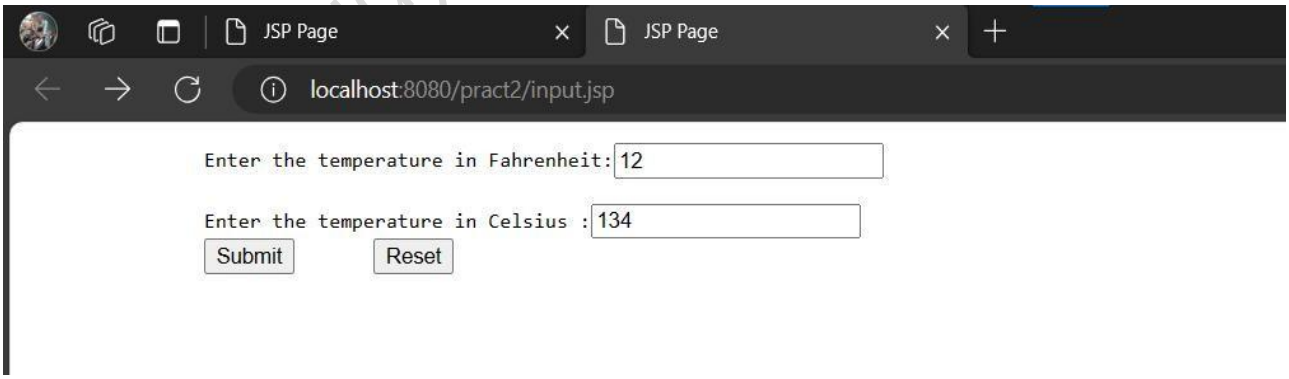
```
<body>
    <form action="output.jsp">
        <pre>
            Enter the temperature in Fahrenheit:<input type="text" name="t1" required>

            Enter the temperature in Celsius :<input type="text" name="t2" required>
            <input type="submit">       <input type="reset">
        </pre>
    </form>
</body>
```

7] create a Web service client and give package name as “Client”. After creating web service client go to web service references and drag and drop the operations in output.jsp in body tag as shown

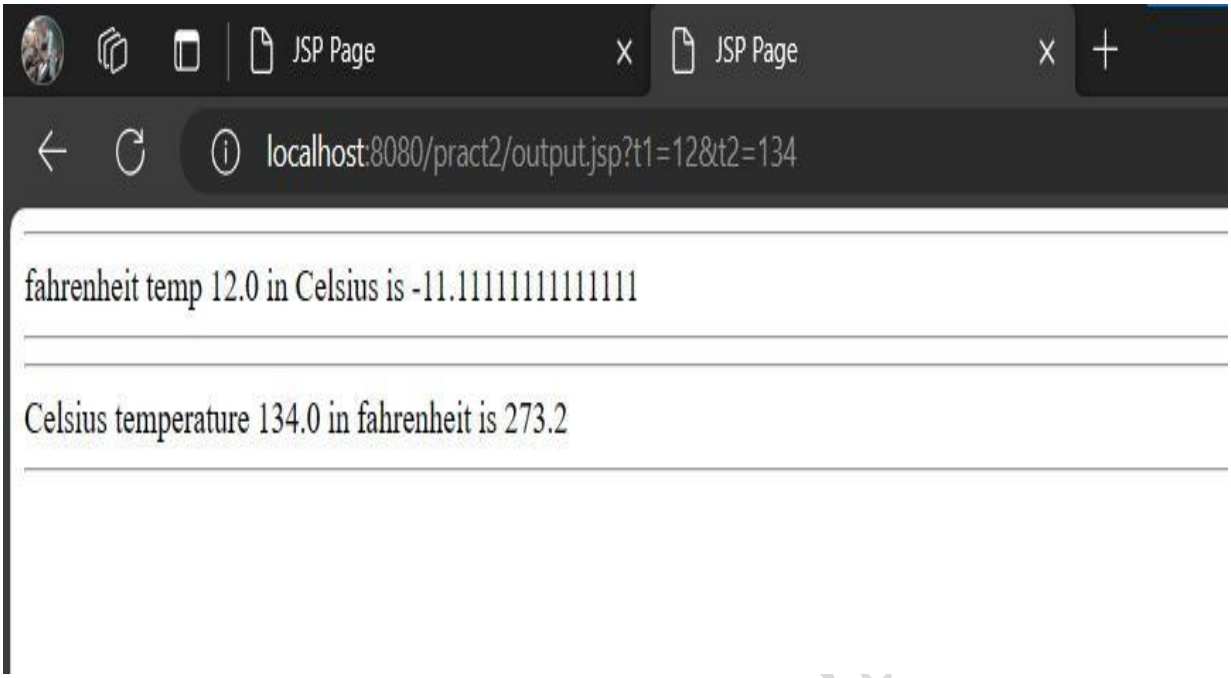
below:

```
        <%-- start web service invocation --%><hr/>
    <%
    try {
        client.Conv_Service service = new client.Conv_Service();
        client.Conv port = service.getConvPort();
         // TODO initialize WS operation arguments here
        double a = Double.parseDouble(request.getParameter("t1"));
        // TODO process result here
        java.lang.String result = port. FtoC(a);
        out.println(result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
```

8] now deploy the model again and run the input.jsp file you willget redirected to the browser page as shown:



After submitting you willget the followingoutput:



Python Client

1] Deploythe web serviceand write the followingcode :

Code:

```
from zeep import Client
wsdl_url='http://localhost:8080/Practical_2/TempConverter?wsdl'
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter temperaure in Fahrenheit :"))
result = client.service.FtoC(d)
print(result)
```

2] replace your_wsdl_url with the url which was copied at the time of web service client creation .

In step no.16 . and replace the operation_name with your method or operation name. and pass parameter to it

3] final code should look like following:

https://study.cscornersunita...

2.py - C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2.py (3.11.4)

File Edit Format Run Options Window Help

```
from zeep import Client
wsdl_url='http://localhost:8080/Practical_2/TempConverter?wsdl'
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter temperaure in Fahrenheit :"))
result = client.service.FtoC(d)
print(result)
```

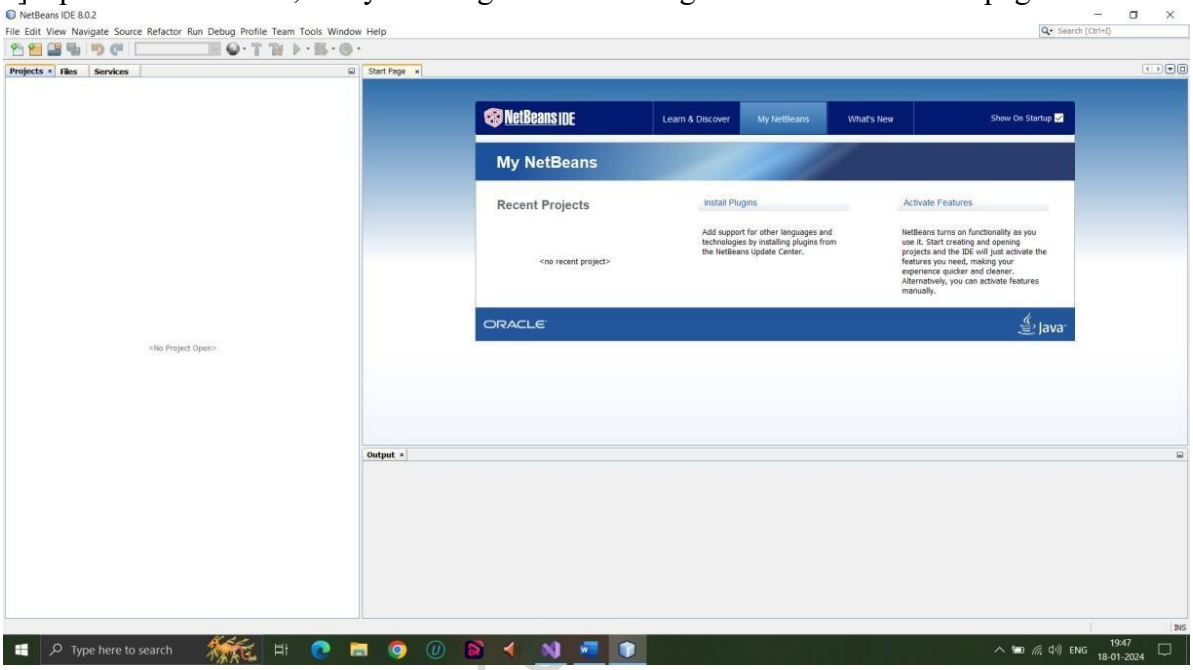
4] afterrunningthe abovecodeyoushouldget the followingutput:

```
===== RESTART: C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2
Enter temperaure in Fahrenheit :1200
The Fahrenheit Temperature 1200.0 in Celsius is 648.8888888888889
```

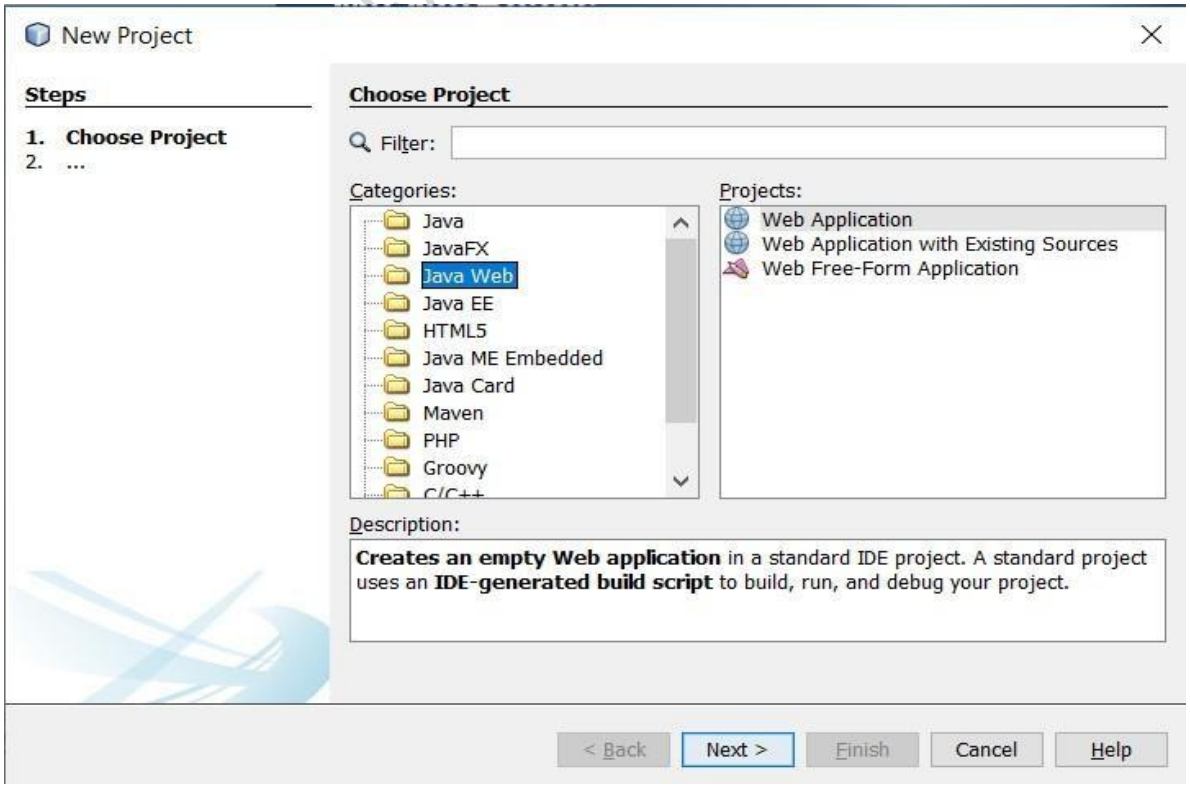
Practical - 2
Create a Simple SOAP service.

Steps:

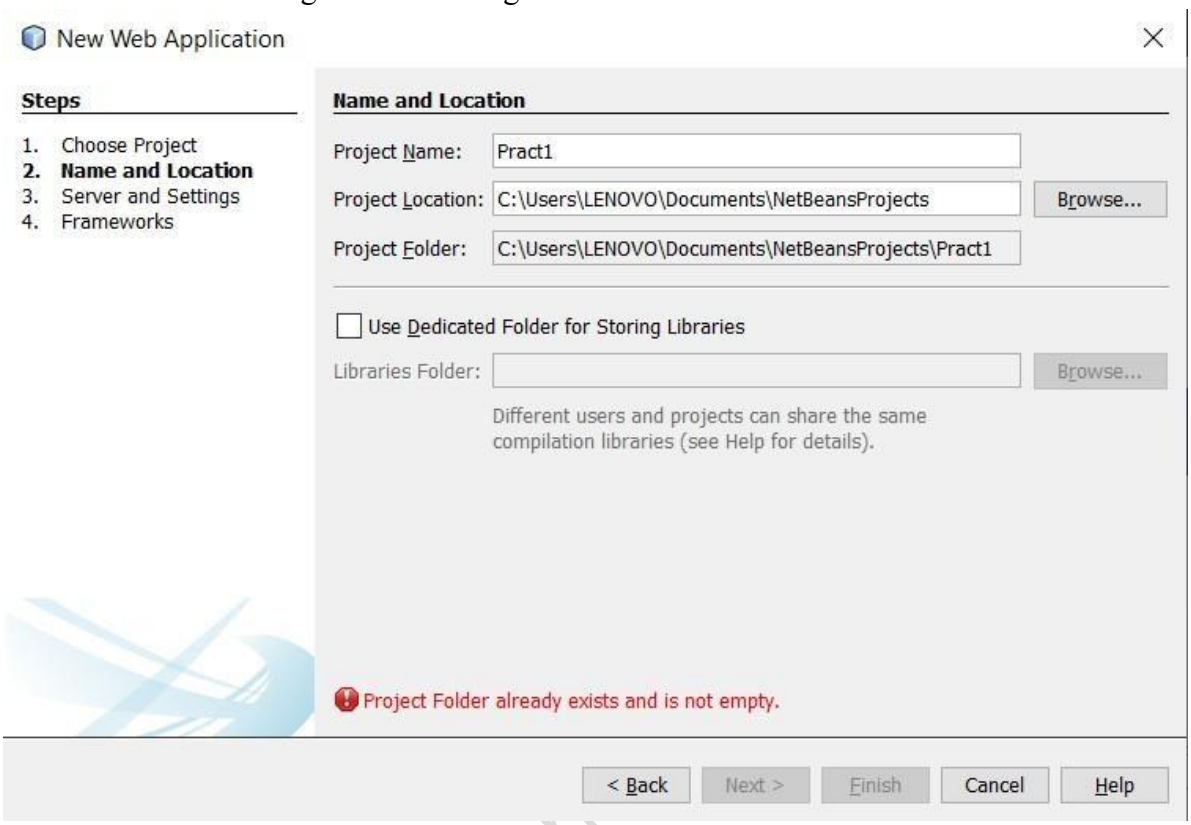
1] Open the NetBeans , and you will get the following screen. Close the startpage.



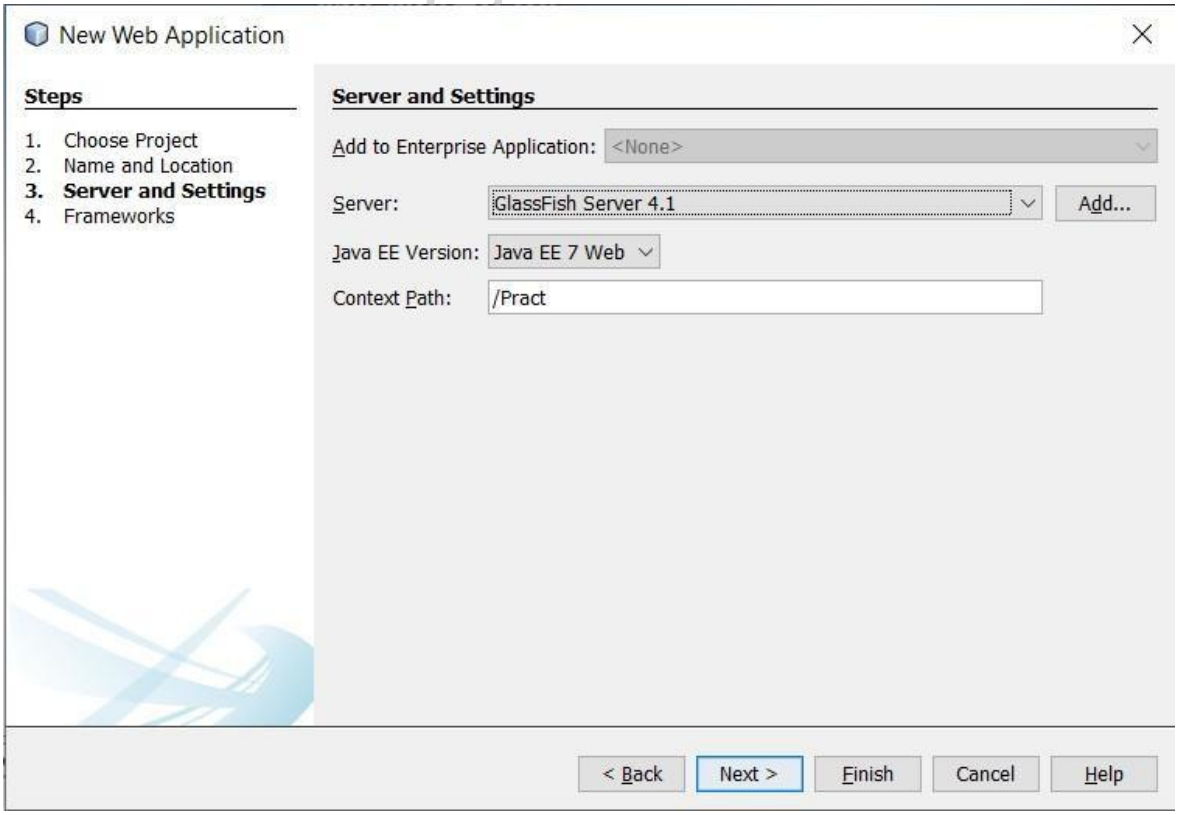
2] Now clickon the file tab and clickon new project you willget the followingscreen :



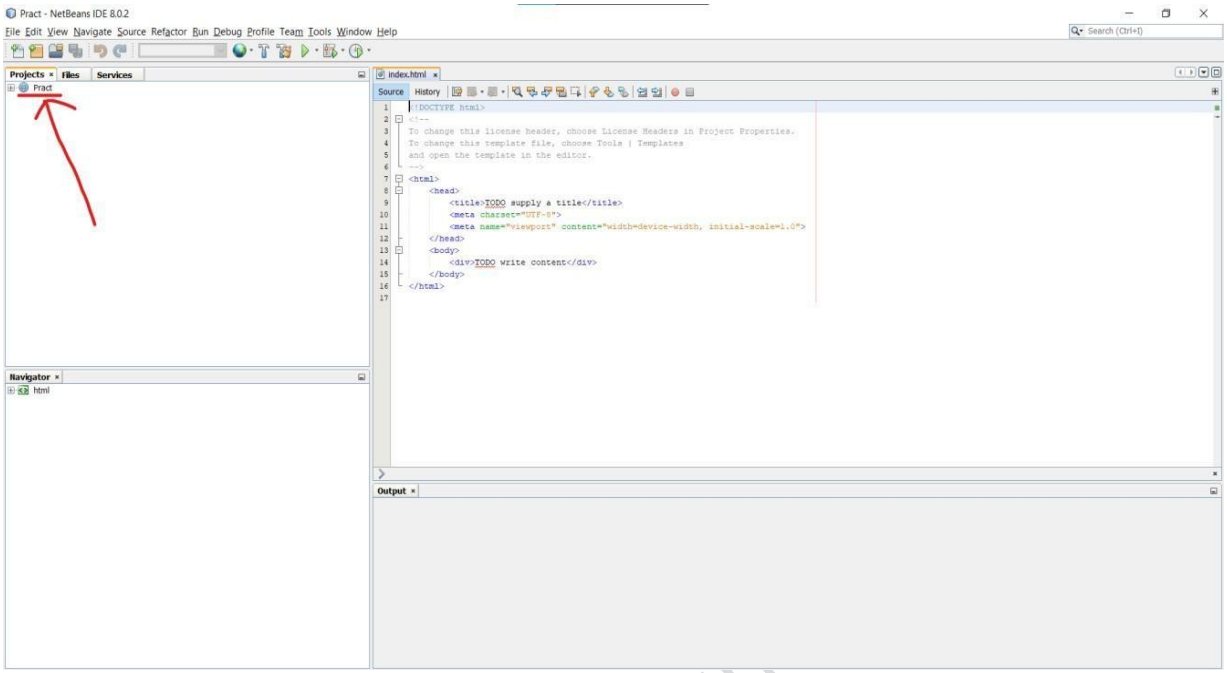
3] In Categories select Java Web and in Projects , Select Web Application .After selecting click onnext . You will get the following window:



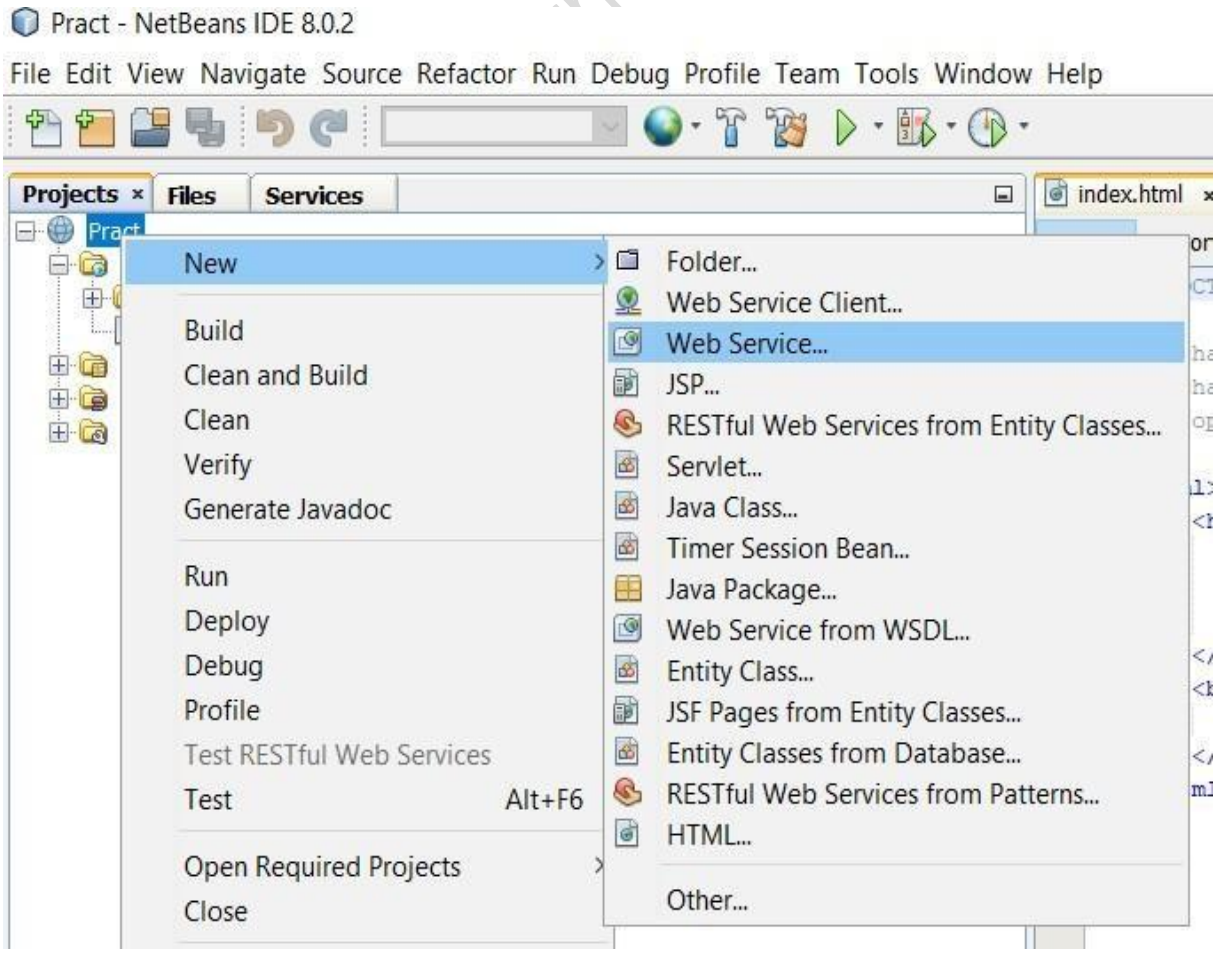
4] NowGivename to the ProjectName: , and clickon next . you willget the followingwindowthen . againclickon finish:



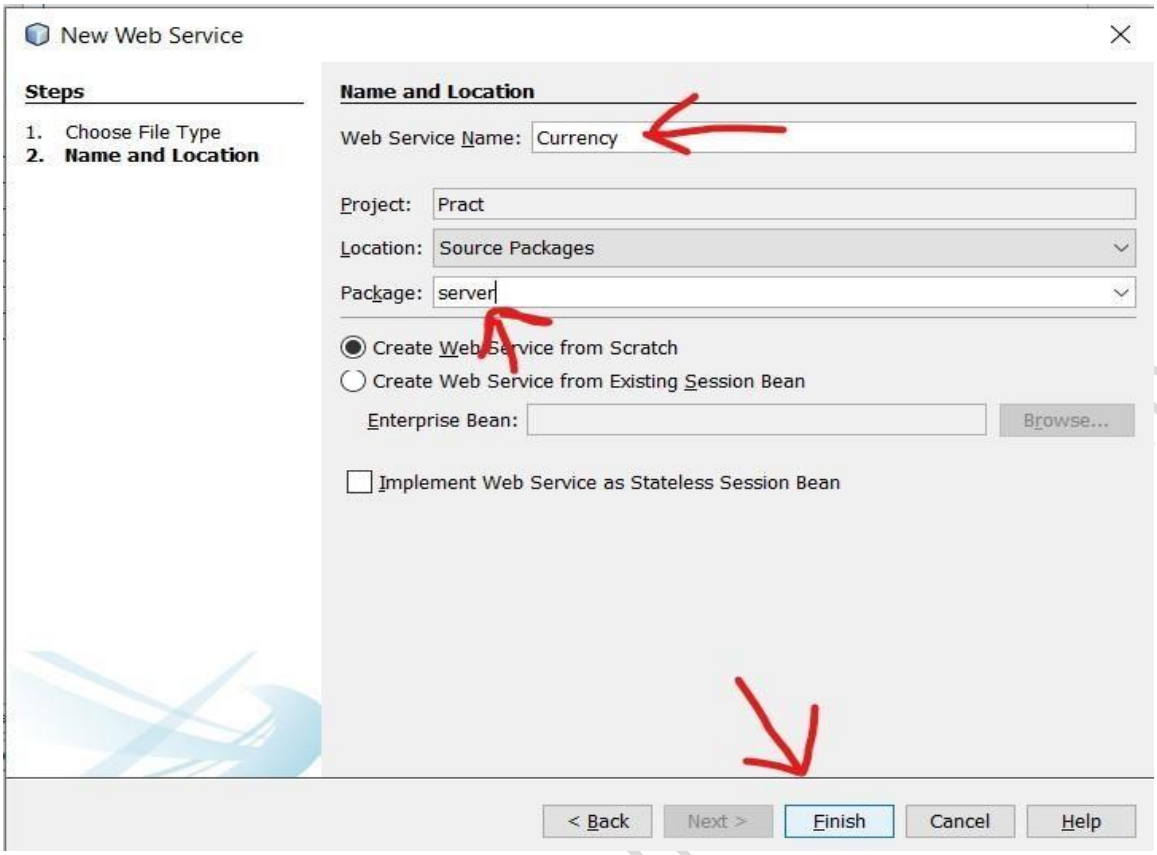
5] You will get the following screen now carefully see in projects section your recently created project appears double click to expand it:



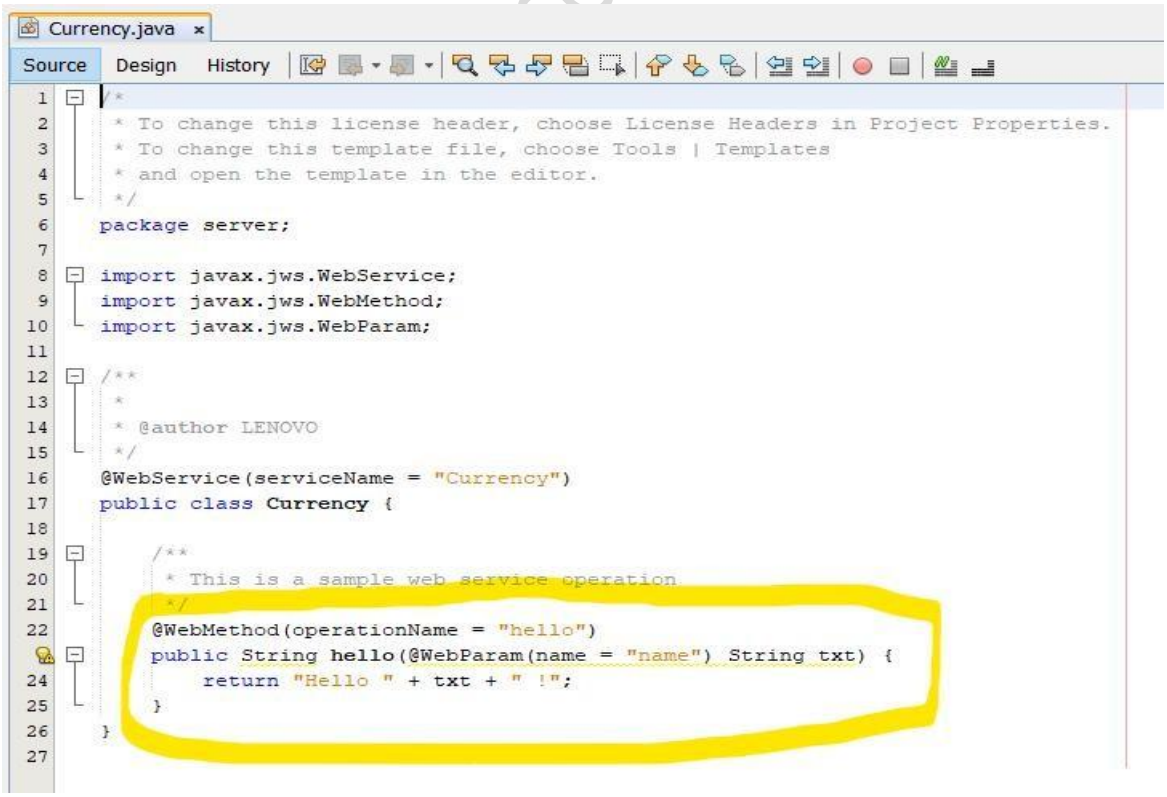
6] Now Rightclickon the project and select new and then select Web Service,Asshown Below:



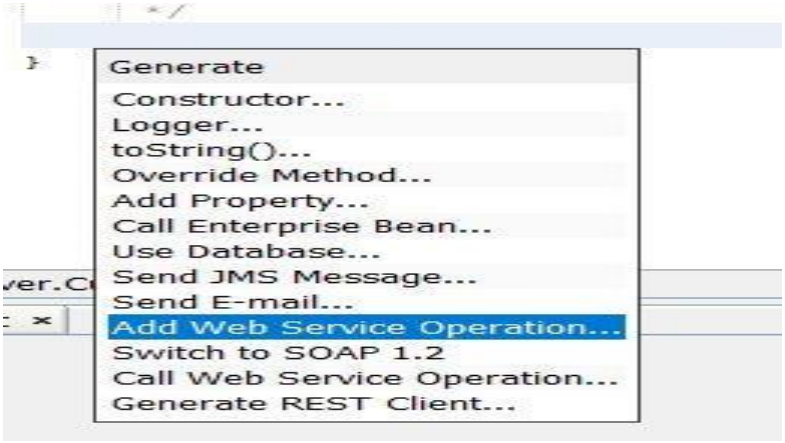
7] after clicking Web Service following window should appear , now give name to web serviceand
give package as “server ”. As shown in the image:



After clicking finish you should get the following window erase the mentioned code :

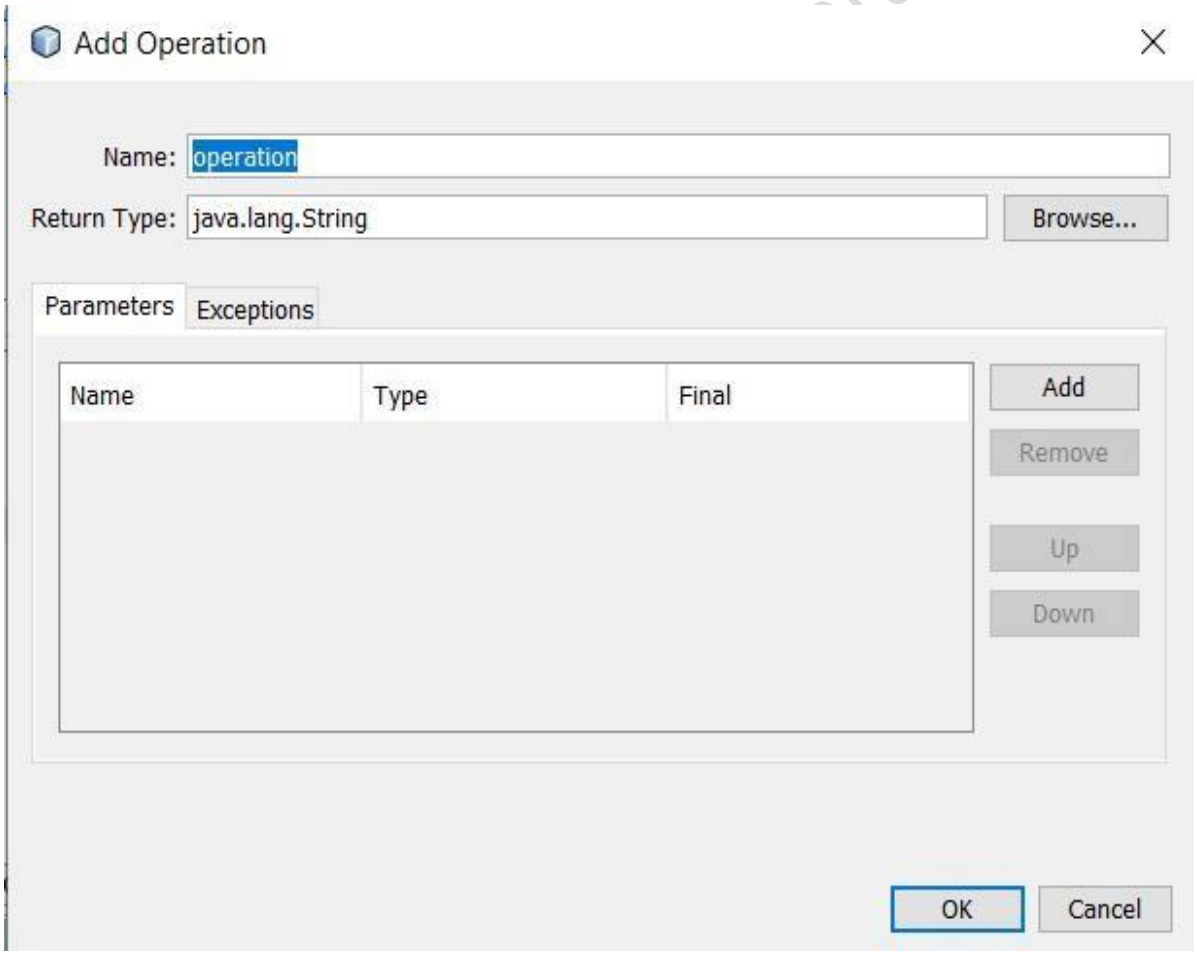


8] Now right click any where and click on insert code and select Add Web Service Operation :

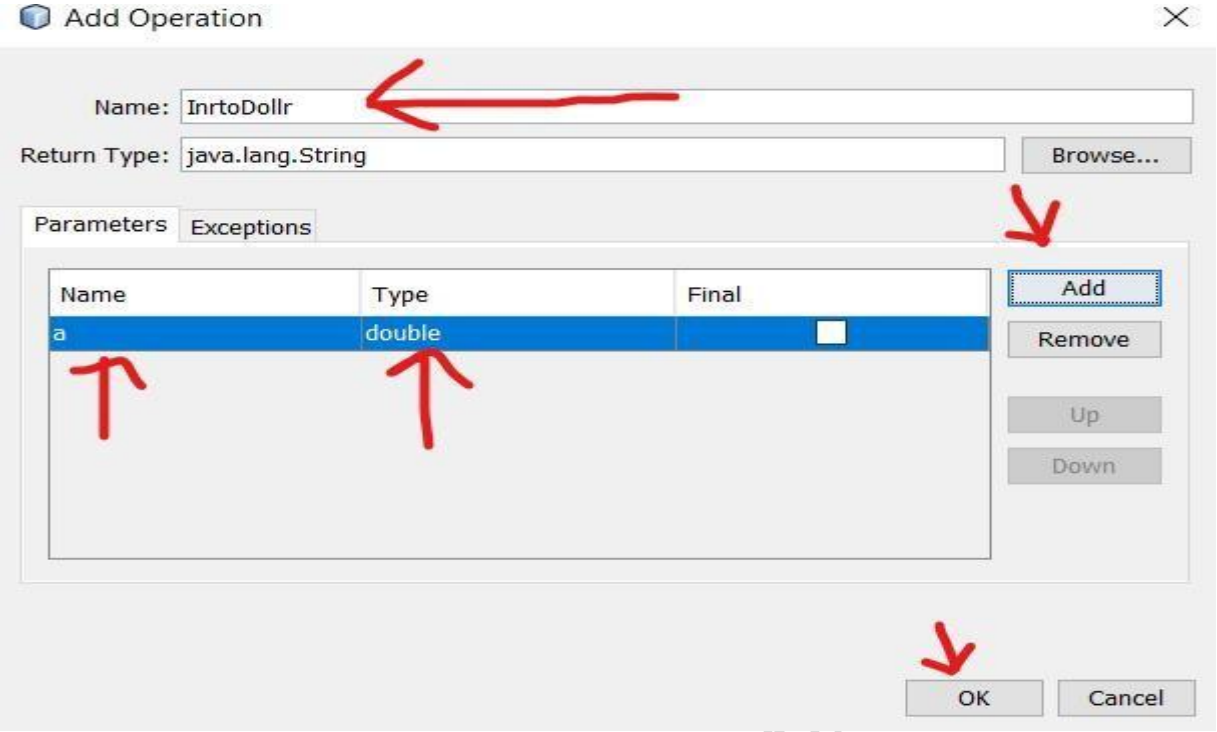


9] the followingwindow would appear :

Just givename to the method or operation and clickon add button to add parameters to the method as here we are converting dollar to rupees we should need only one parameter .



10] give the name to the variable select data type as double and click on ok as shownbelow:

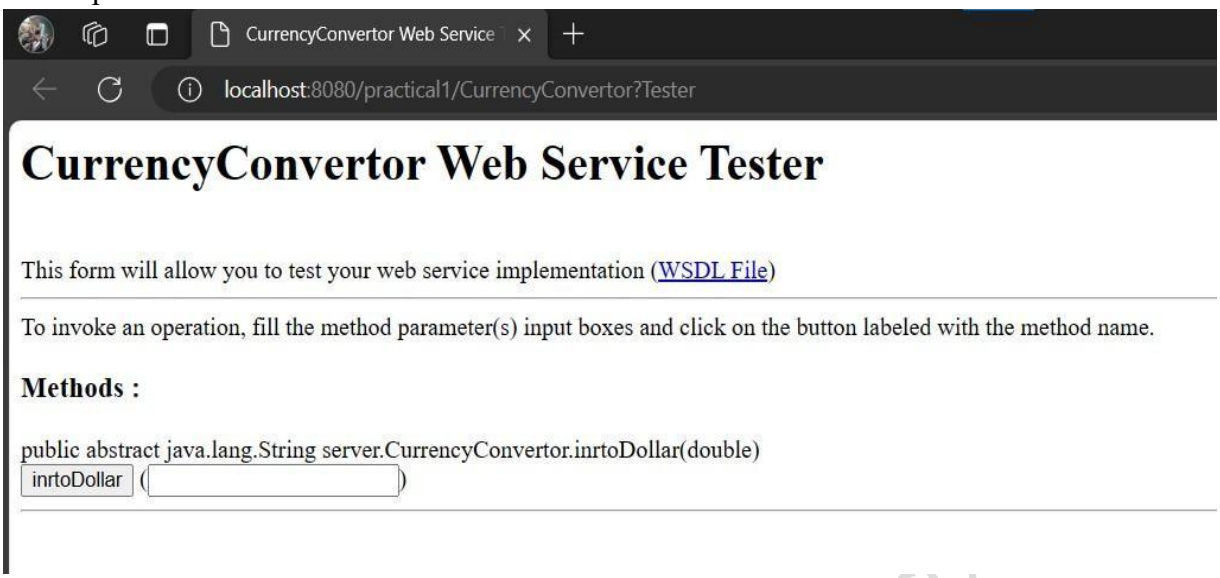


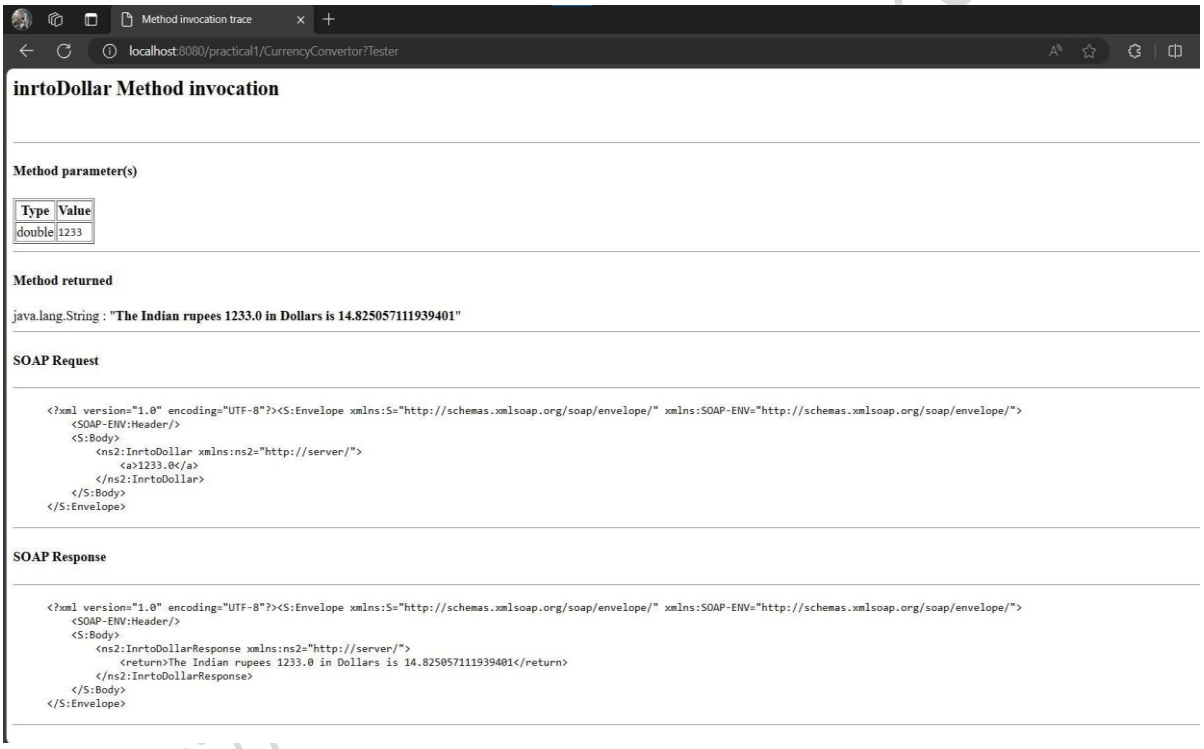
11] Afterclickingok code willautogenerate , make changes In that code as mentioned below:

```
@WebMethod(operationName = "InrtoDollar")
public String InrtoDollar(@WebParam(name = "a") double a) {
    //TODO write your implementation code here:
    return "The Indian rupees "+a+" in Dollars is "+(a/83.17);
}
```

12] now our web service is ready now right clickon project and clickon deploy:

13] now right click on webservice and click on test web service you will get the following output:

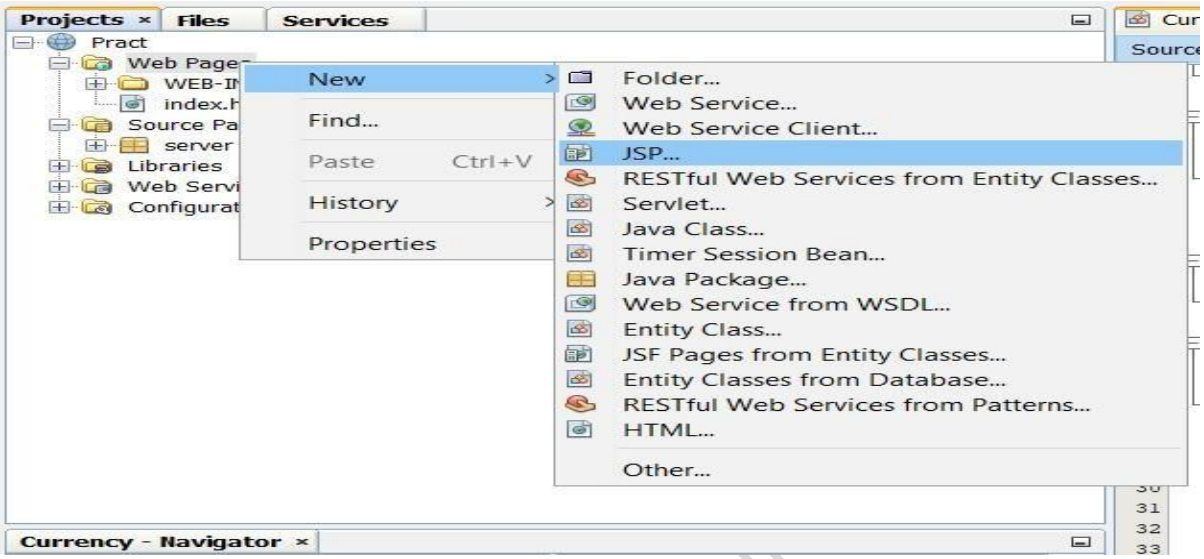




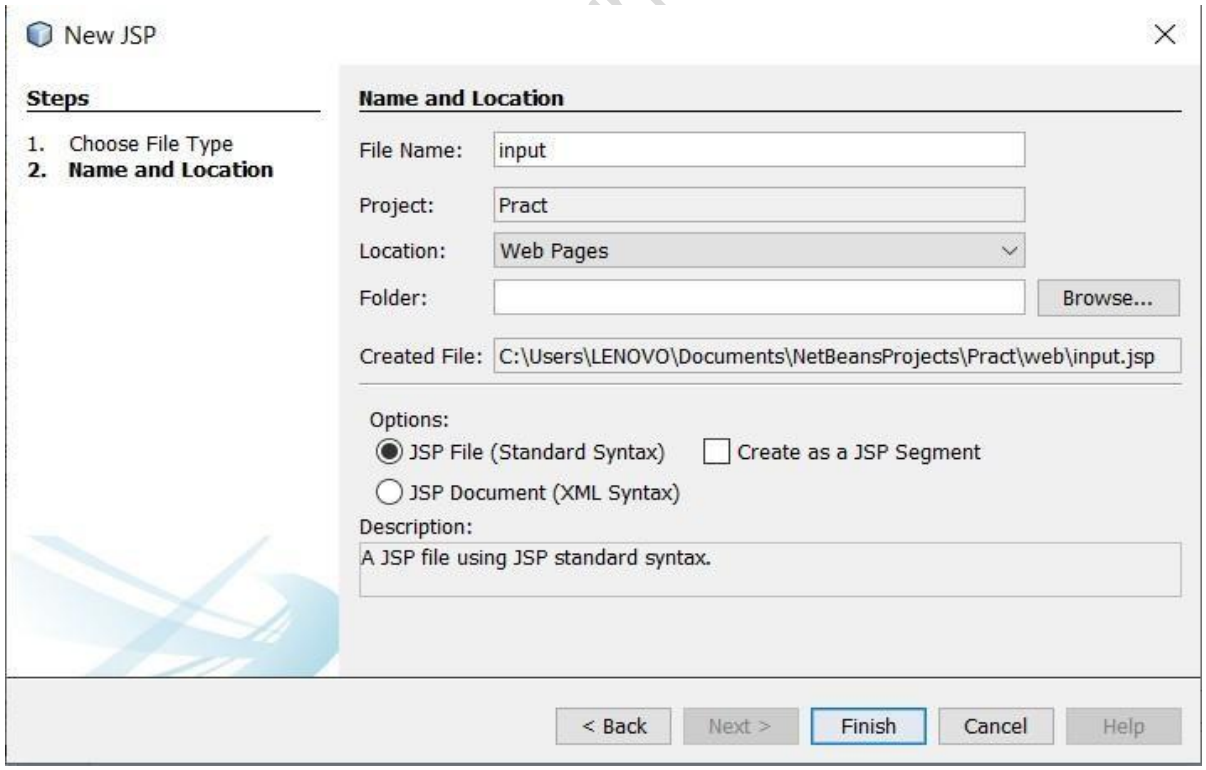
So this is how we created our web service and deployed it.

Creating a java Client using jsp

14] now our web service is successfully deployed. Right click on web pages and select new and select jsp as shown below:



15] give name and click on finish as shown below do it 2 times on for input and one for output:

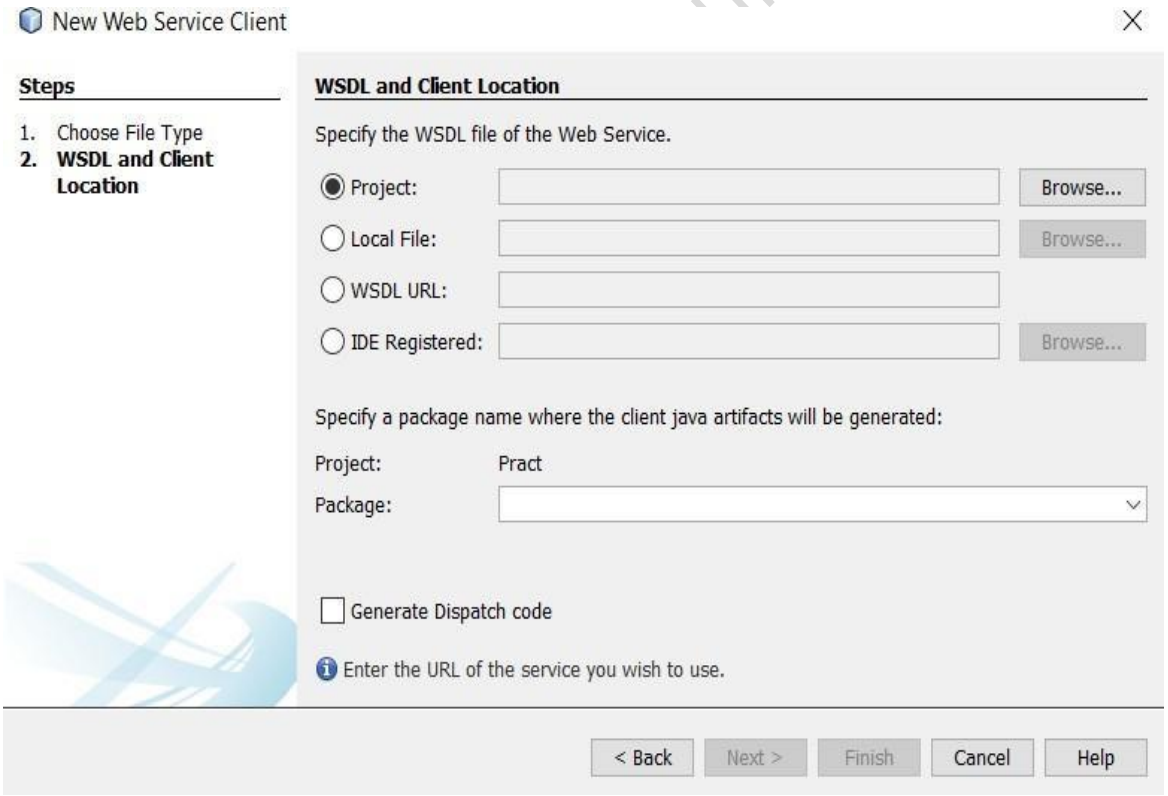


16] In input.jsp create a form for taking user input as shown below:

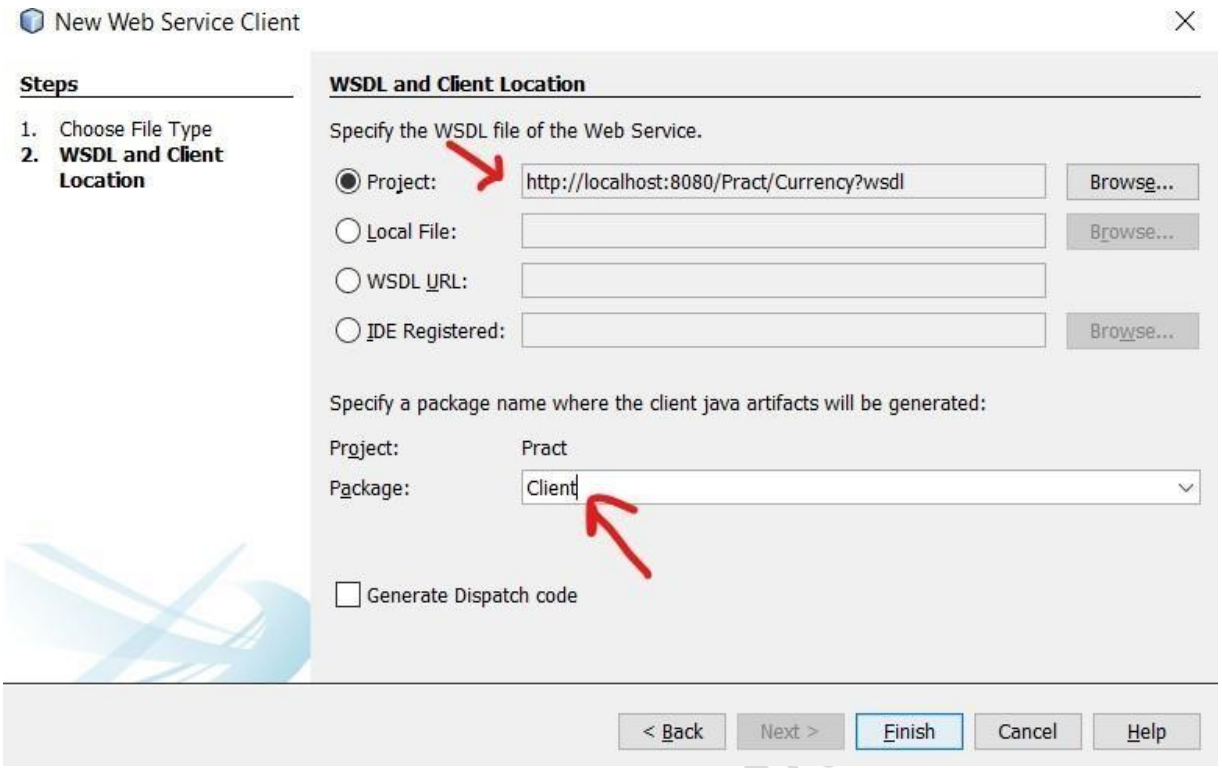
```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="output.jsp">
            <pre>
                Enter the currency in rupees : <input type="text" name="t1">
                <input type="submit">       <input type="reset">
            </pre>
        </form>
    </body>
</html>
```

In this code set action = the jsp file where u want output and give name to textbox input.

17] now we have to create web service client, for same right clickon project and select new then select web service client you willget the followingscreen:



At this step click on browse and select your project and click ok after clicking ok you will get wsdl url as shown below:

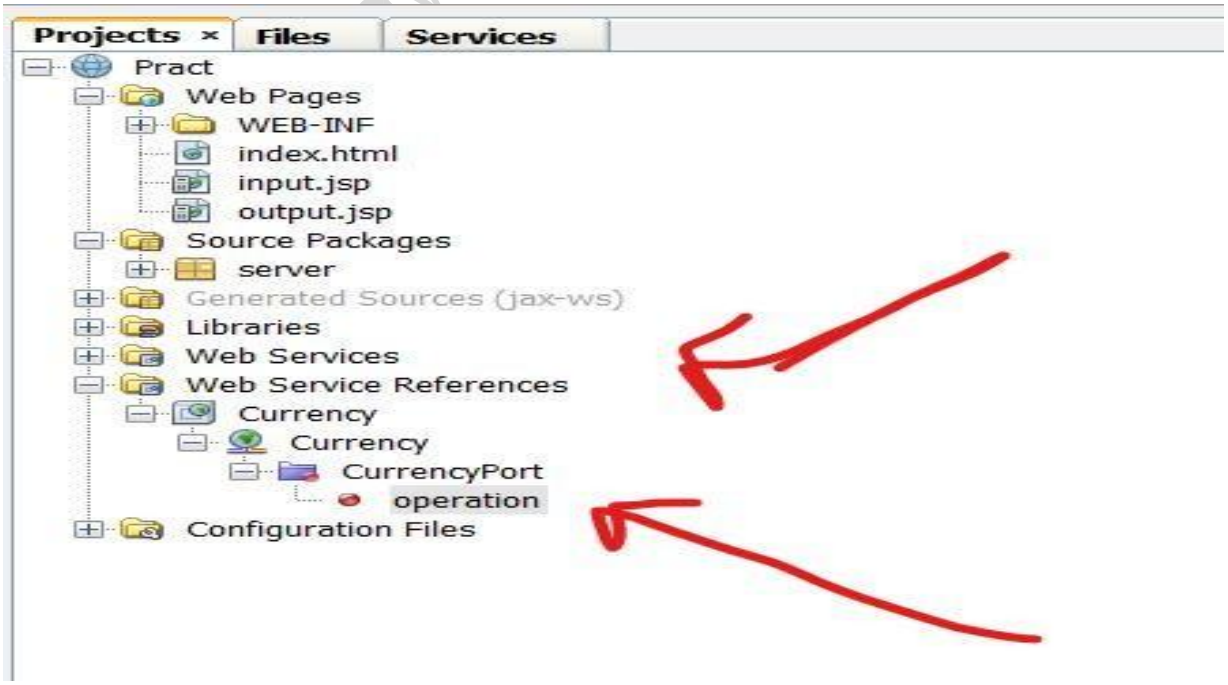


At this stage copy the url and paste it in notepad for future.

Give package name as "Client".

Now click on finish.

18] now in project section you can se a new folder is been created named "web service refrence". double click to expand it you should get the following:



Hold the operation or your operation name and drag it into the output.jsp in body tag as shown

: You will get the following auto generated code:

```
<body>
        <%-- start web service invocation --%><hr/>
<%
try {
    Client.Currency_Service service = new Client.Currency_Service();
    Client.Currency port = service.getCurrencyPort();
     // TODO initialize WS operation arguments here
    double a = 0.0d;
    // TODO process result here
    java.lang.String result = port.operation(a);
    out.println("Result = "+result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>

</body>
```

19] now make changes as shown below in the code:

```
        <%-- start web service invocation --%><hr/>
<%
try {
    Client.Currency_Service service = new Client.Currency_Service();
    Client.Currency port = service.getCurrencyPort();
     // TODO initialize WS operation arguments here
    double a = Double.parseDouble(request.getParameter("t1"));
    // TODO process result here
    java.lang.String result = port.operation(a);
    out.println(result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>
```

20] now Deploy our project again and right click on input.jsp and click run file you will redirect to a browser page as shown :

Enter any numerical value in text box and click on submit you will get the following output:

Python client:

1] for consuming java web services in python we should repeat the above steps till step np.19 .

Note: for getting output the project or web service should be deployed .

After doing all of the steps write the following code in python idle:

Code:

```
from zeep import Client
wsdl_url='http://localhost:8080/pract/Currency?wsd
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter Currency in Indian Rupees:"))
result = client.service.InrtoDollar(d)
print(result)
```

2] replace wsdl_url with the url which was copied at the time of web service client creation .

In step no.17 . and replace the InrtoDollar with your method or operation name. and pass parameter to it

3] final code should look like following:

```
*2.py - C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2.py (3.11.4)*
File Edit Format Run Options Window Help
from zeep import Client
wsdl_url='http://localhost:8080/pract/Currency?wsdl'
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter Currency in Indian Rupees:"))
result = client.service.InrtoDollar(d)
print(result)
```

4] after running the above code you should get the followingoutput:

```
= RESTART: C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2.py
Enter Currency in Indian Rupees:12000
The Indian rupees 12000.0 in Dollars is 144.28279427678248
```

.Net Client

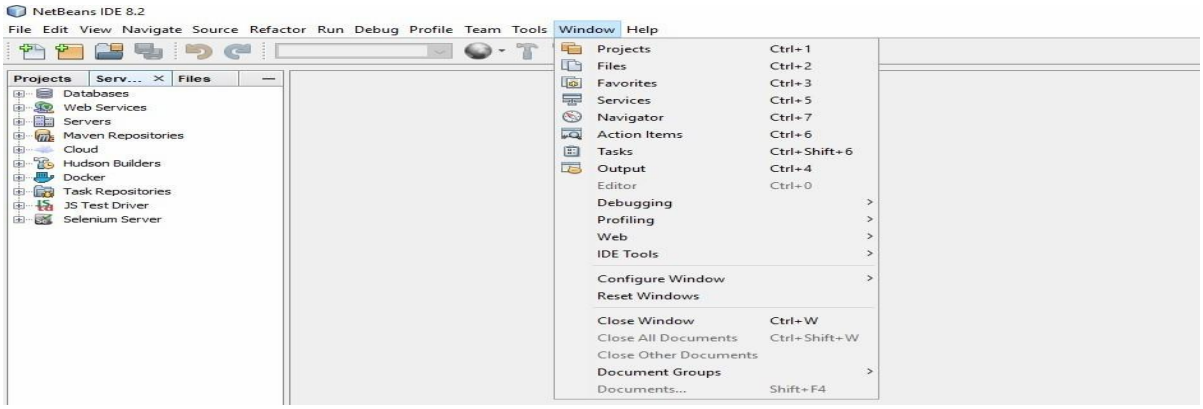
www.scornersunitaral.com/

Practical - 3

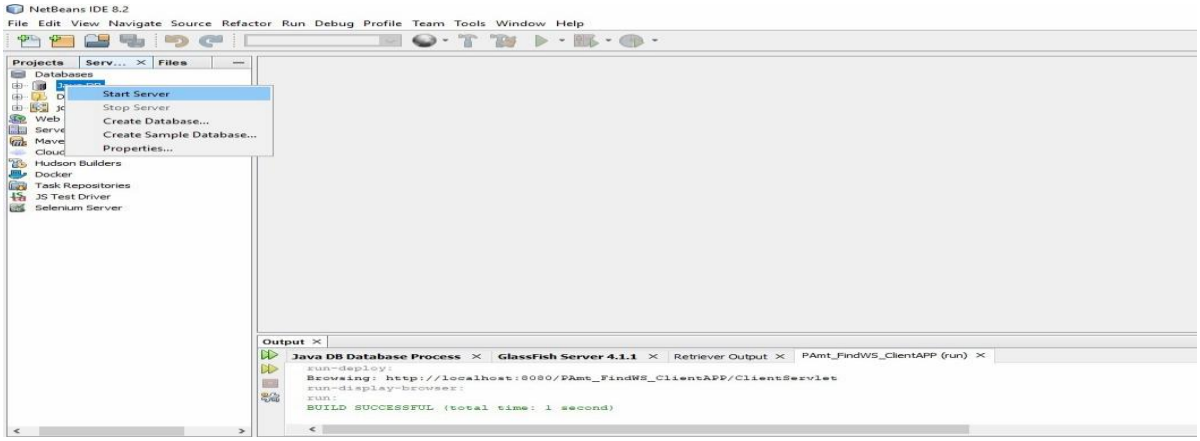
Create a Simple REST Service to demonstrate CRUD operations with "Student" database

Steps:

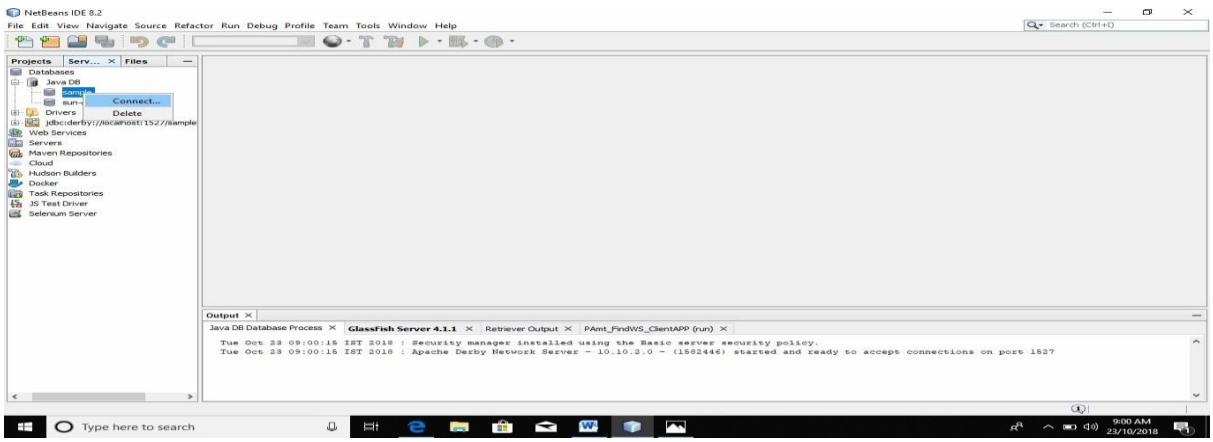
1. **Click on Window menu and click on Projects, Files & Services to open it.**



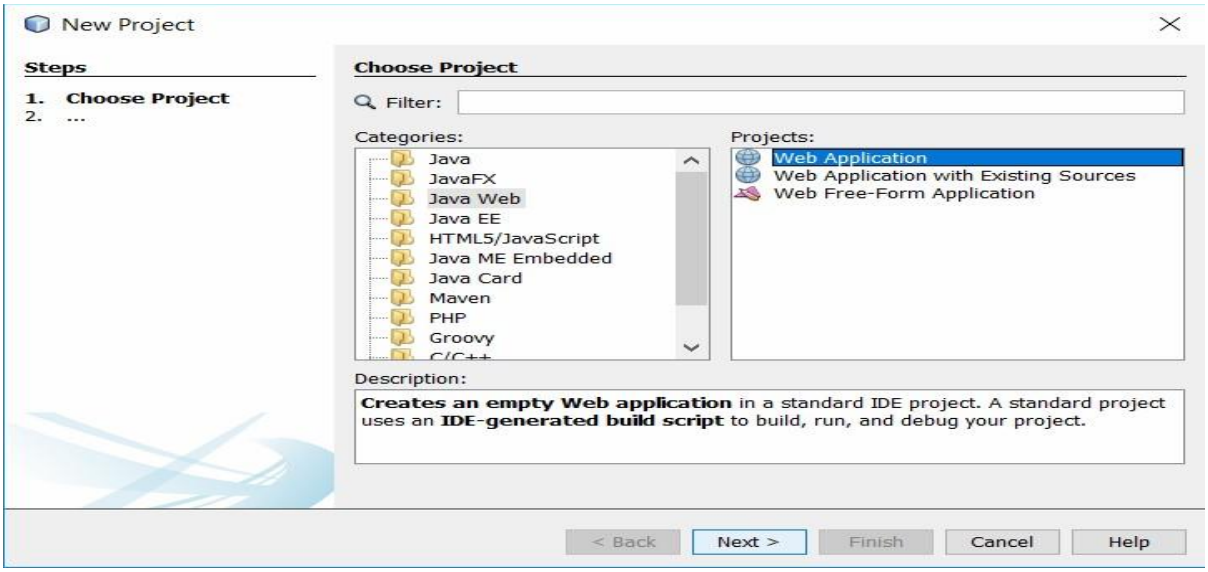
2. **Right click on Java DB and then click on Start Server to start the server .**

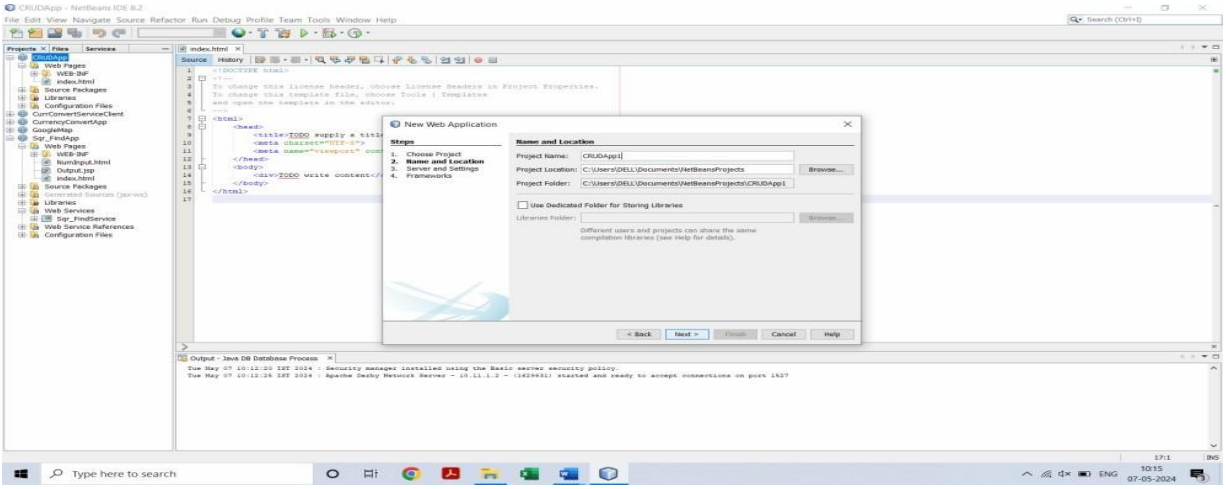


3. **Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.**

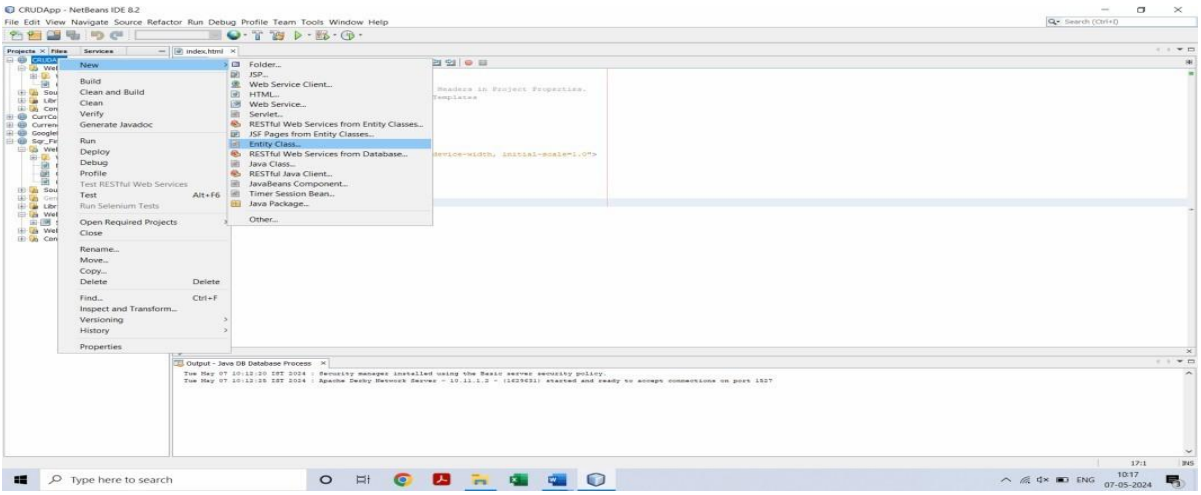


4. Now create a web application with the name CRUD_Operation. A window will open like following pic.



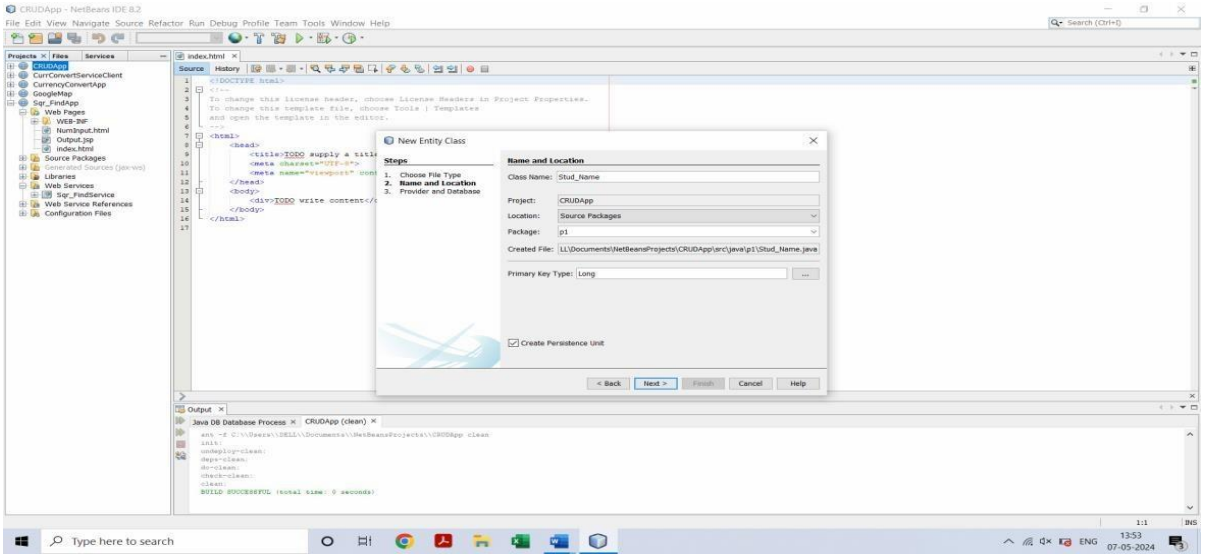


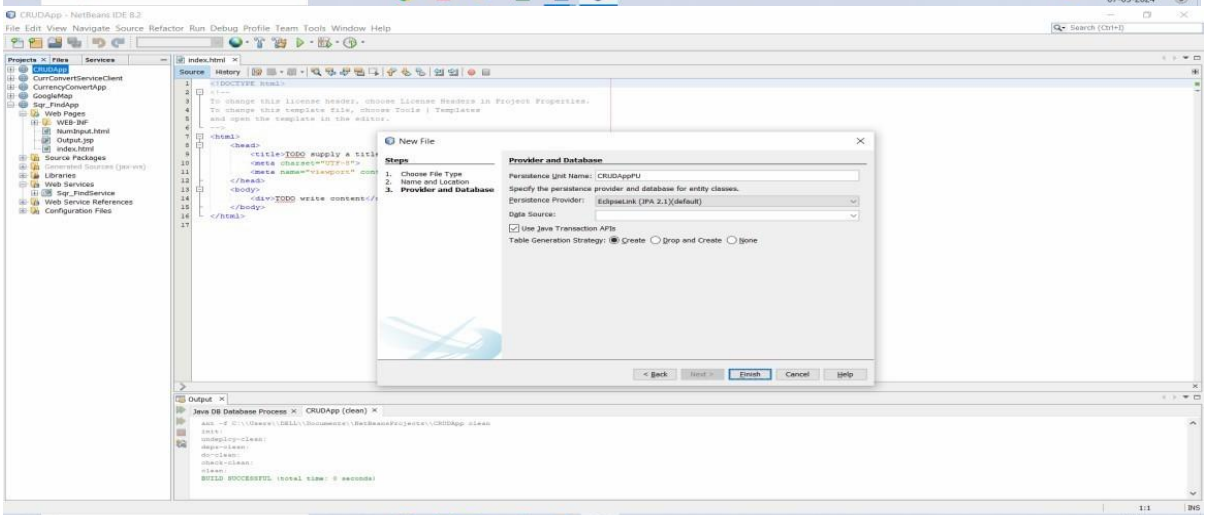
5. Create an entity class. Right click on project name -> New -> Entity Class.

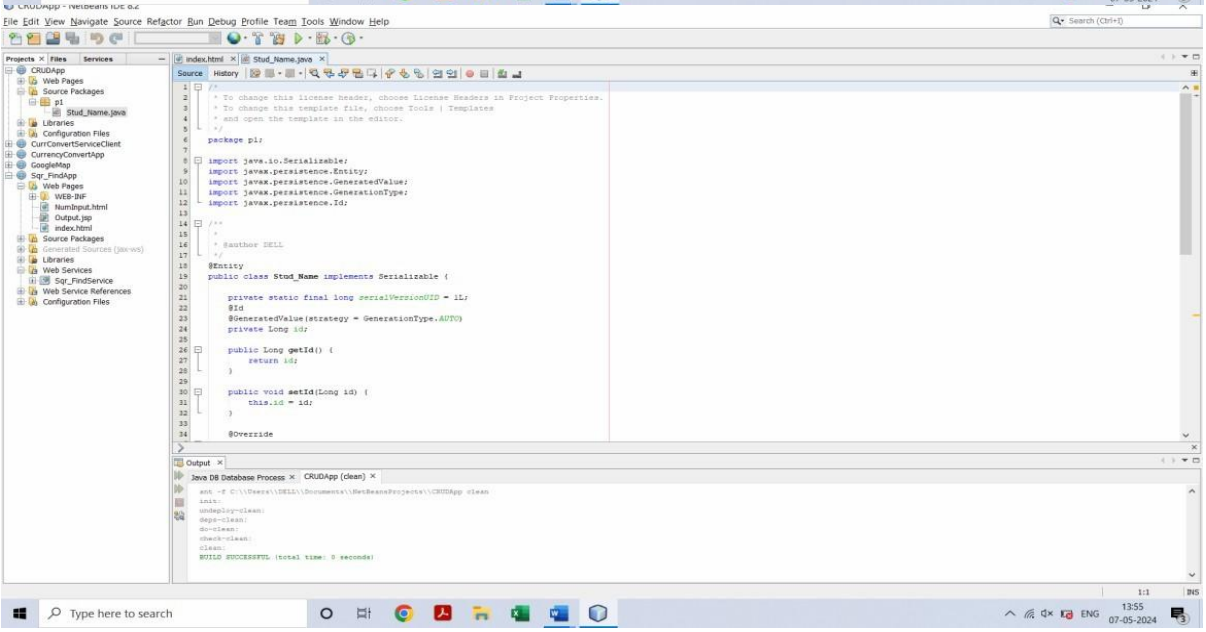


6. A window will appear like bellow pic. Enter following data and click on Next ….
Class Name ->Stud_Name Package Name -> p1

7. Click on finish.

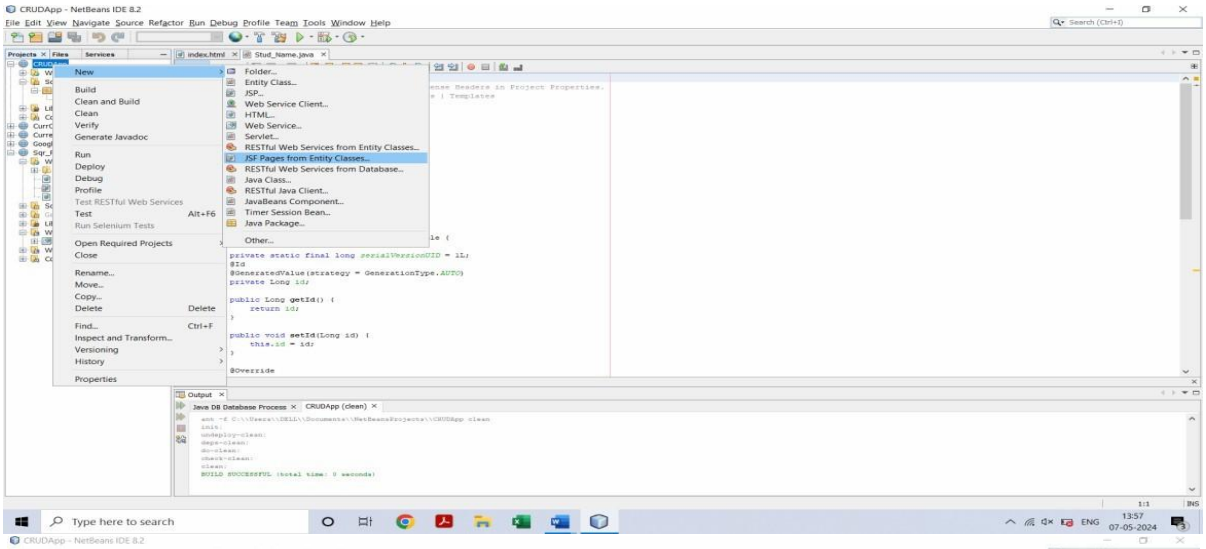


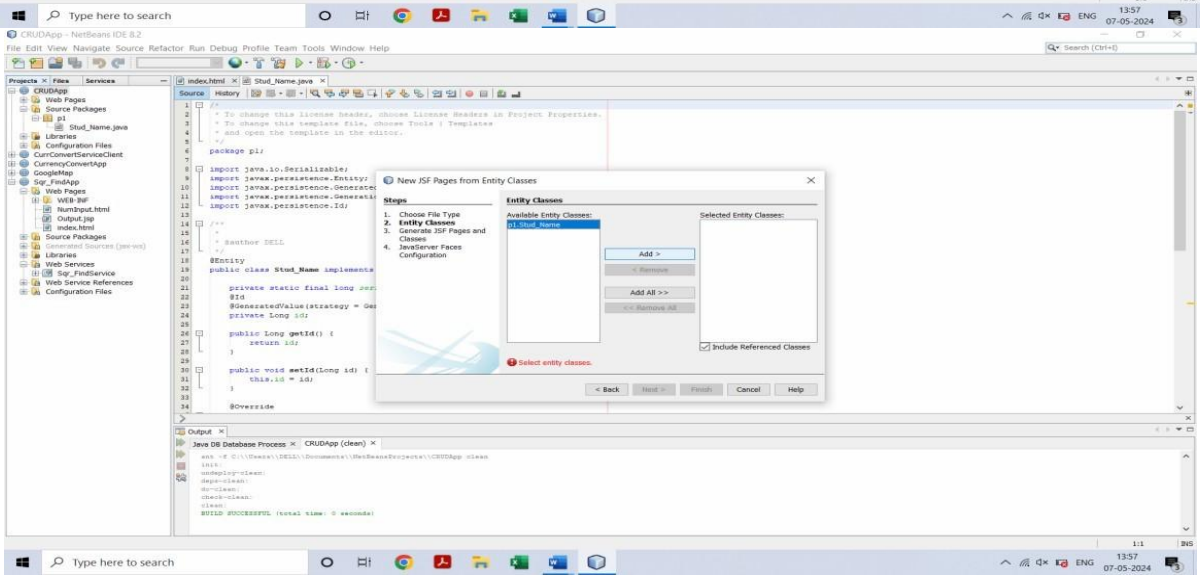


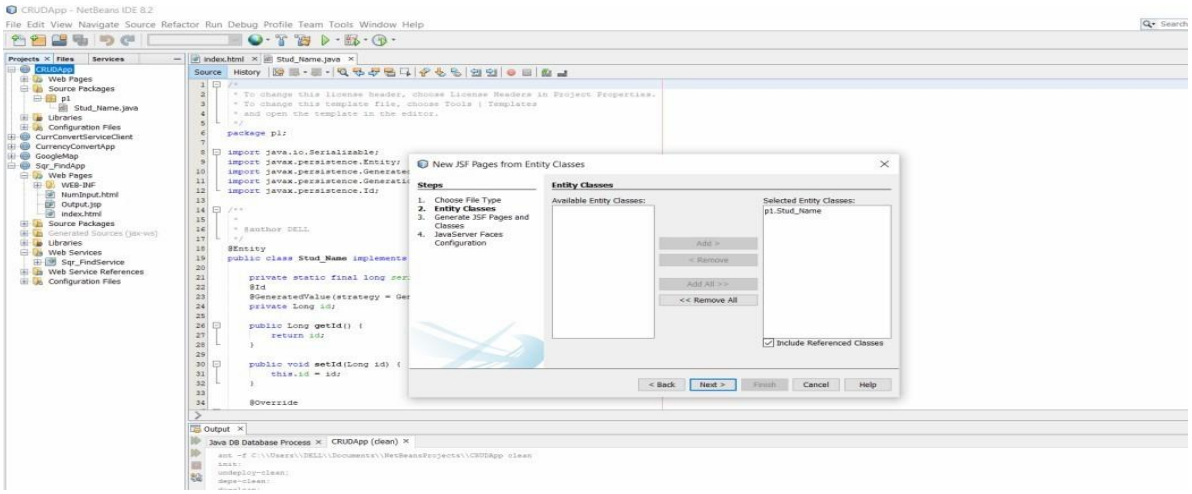


8. Right click on project name and create JSF Pages from Entity Classes.

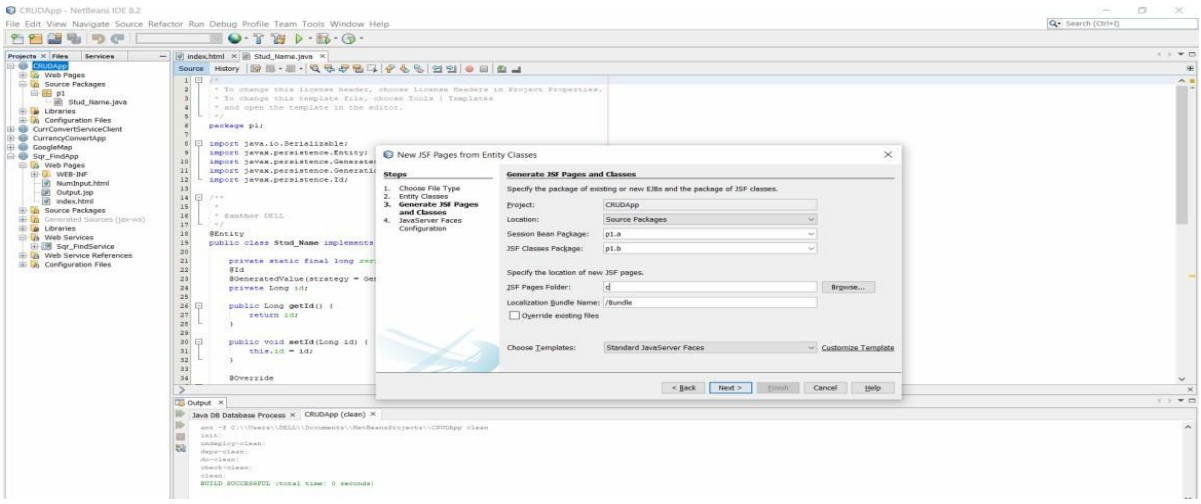
9. Select and click on Add button and then Next button on below



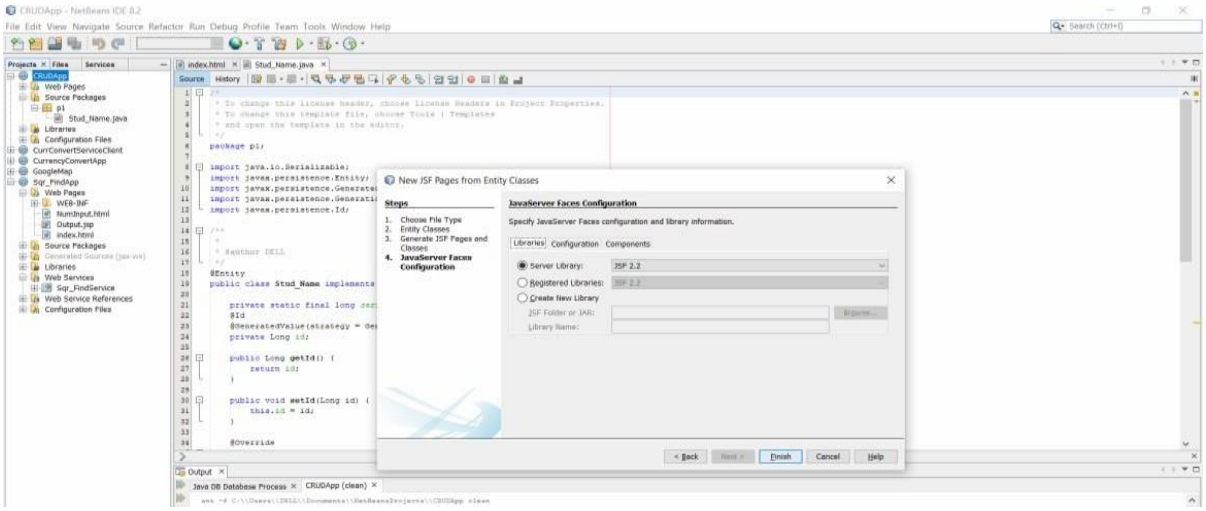




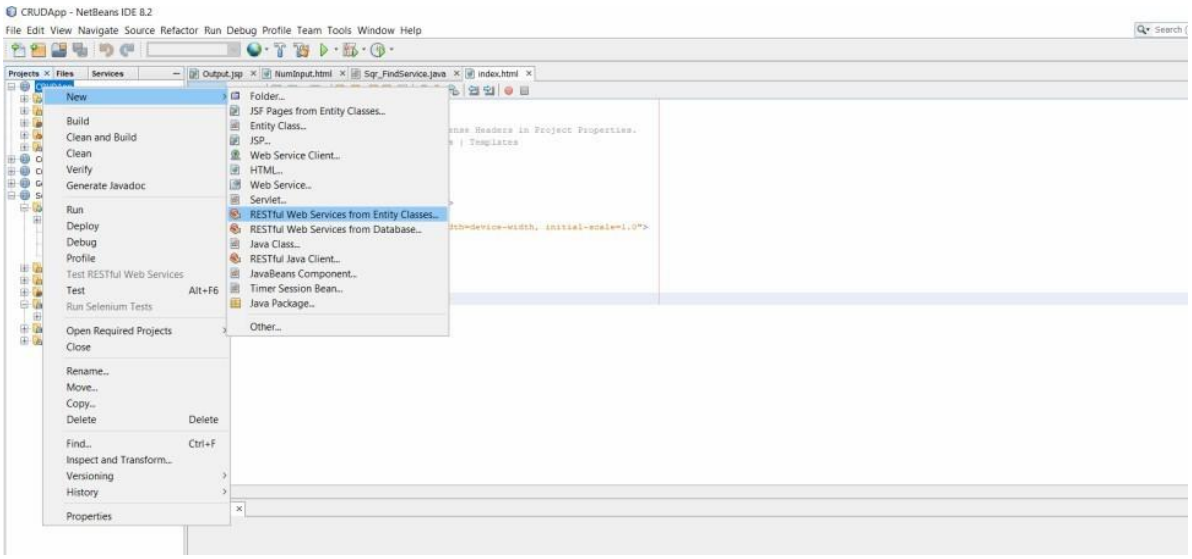
10. A window like below will appear on the screen. Enter the data into that window as entered in below pic and click on Next button.

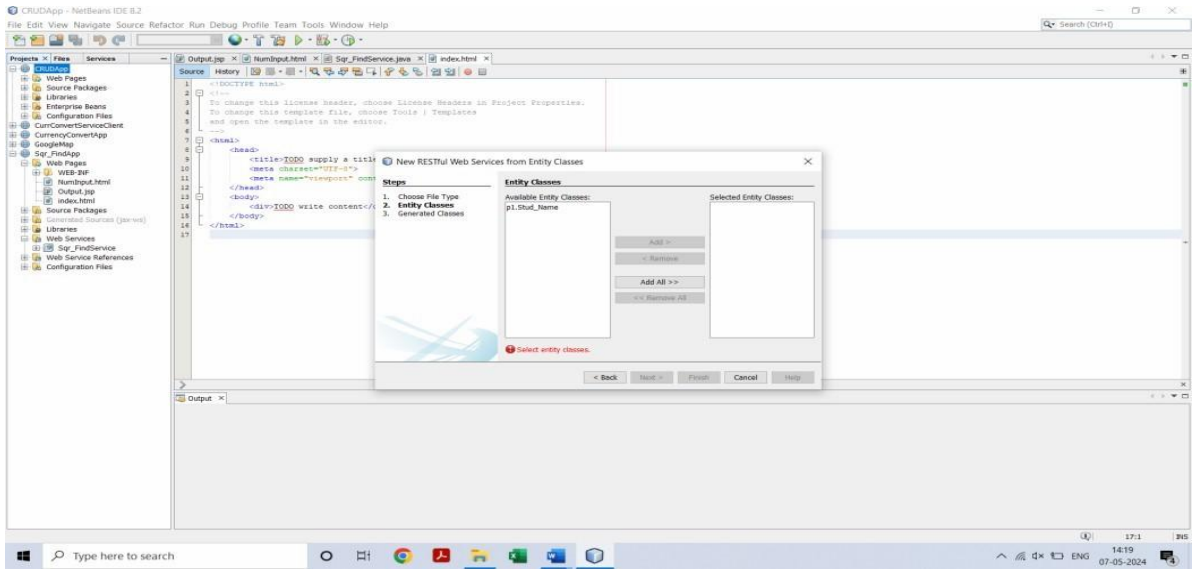


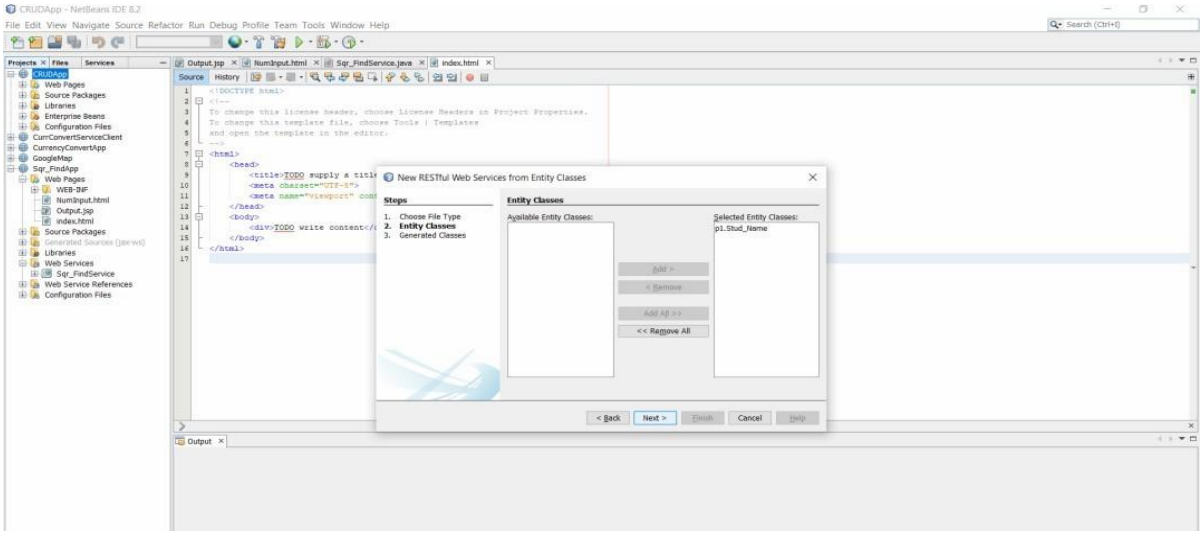
11. Now click on Finish.



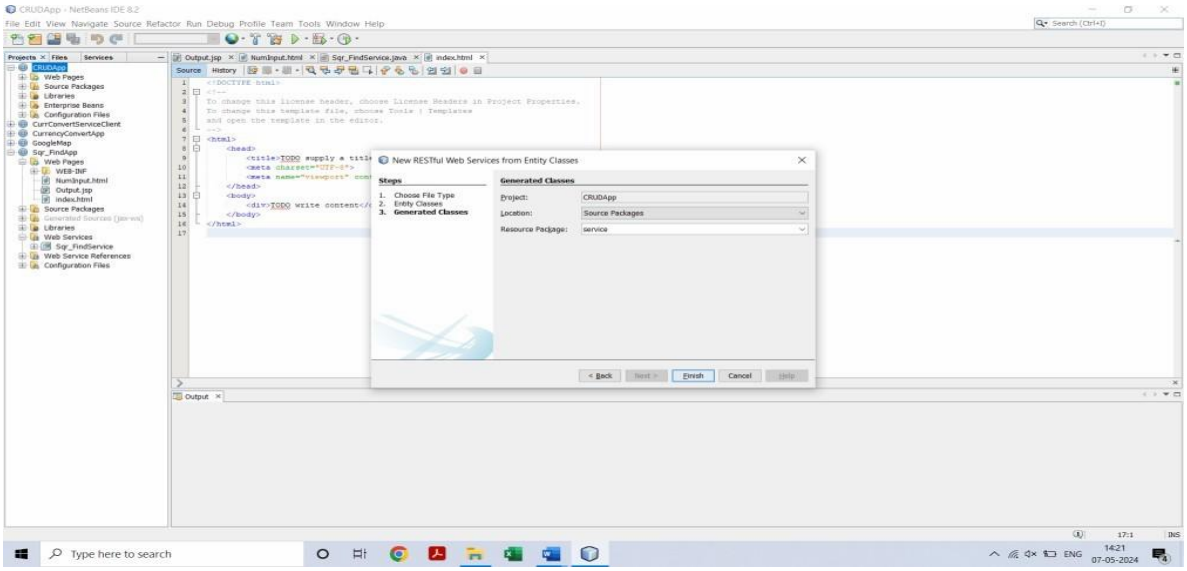
12. Right click on project name and create RESTful Web Services from Entity Classes.







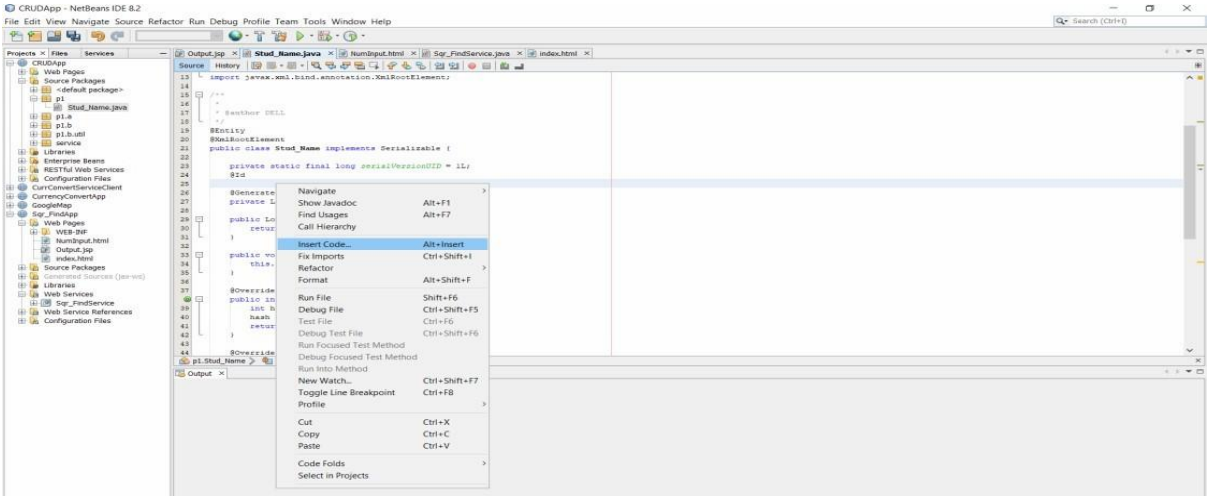
13. Repeat step 9 and then it will go on next page. Then enter the p1.service in Resource Package and then click on Finish button.

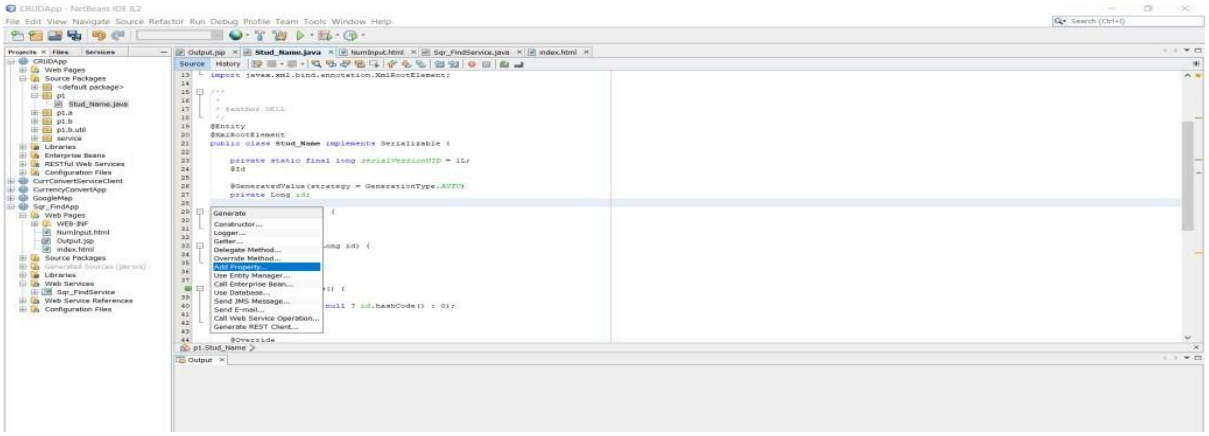


14.Now open Stud_Name.java file under p1 package.

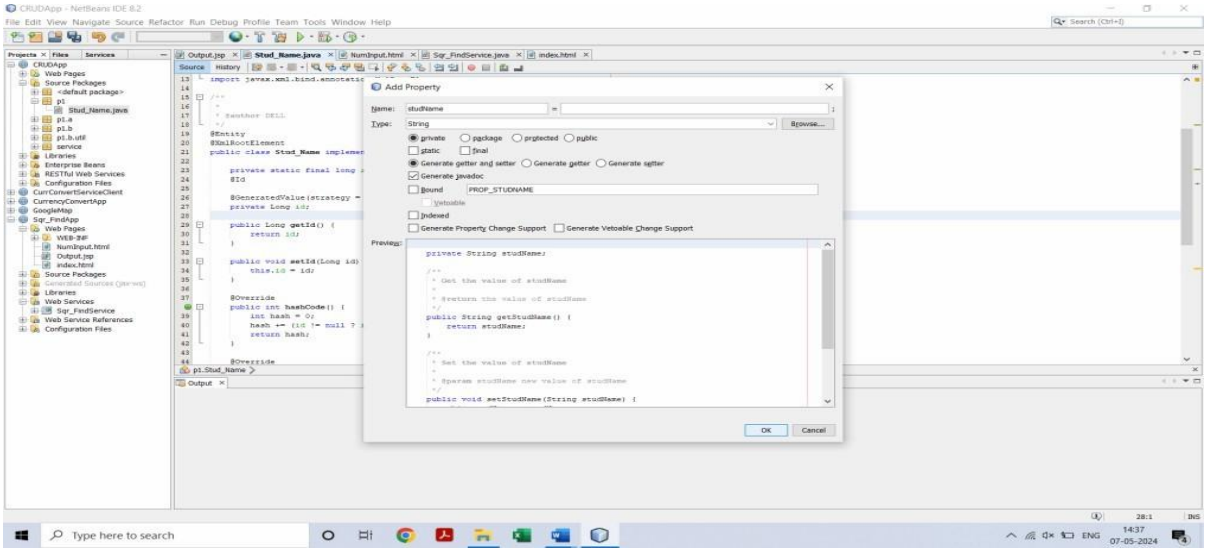
15. In this file at line number 24, do the right click and select Insert Code.

16. A new list will appear. Click on Add Property

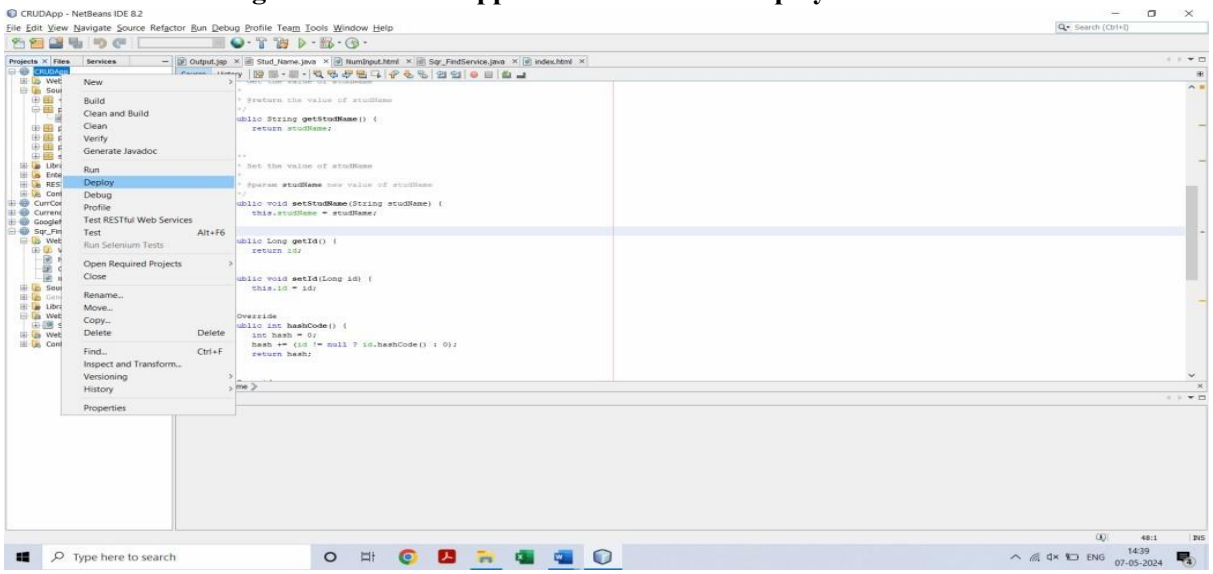




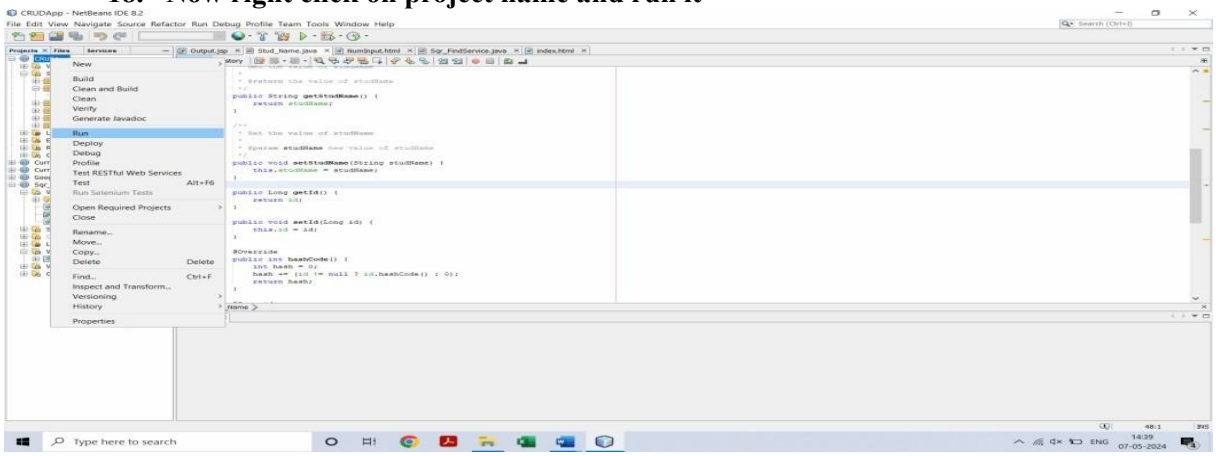
A new window will open. Enter name as studName. Make sure name should be exact same as of mine and then click on OK button. Actually we are setting getter and setter method for studName.



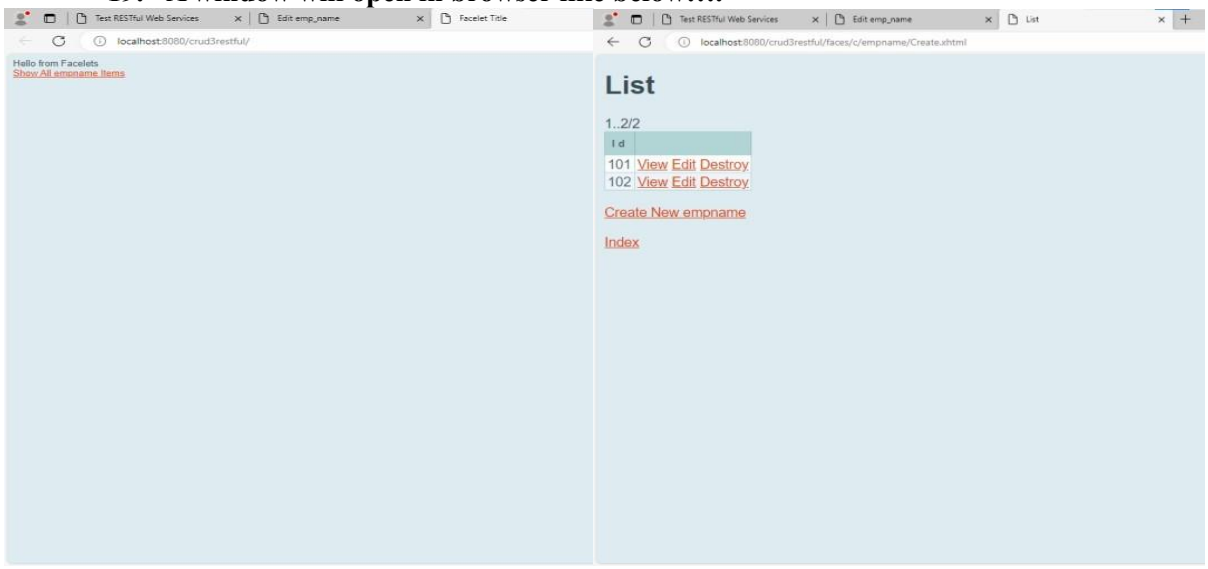
17. Now right click on web application name and Deploy it.



18. Now right click on project name and run it



19. A window will open in browser like below….

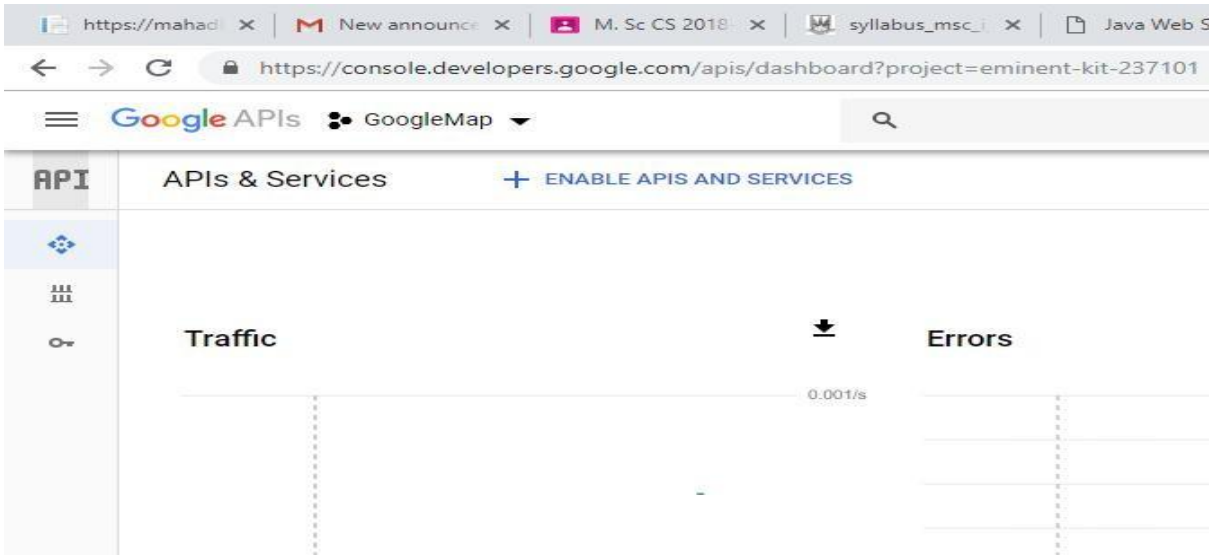


Practical - 4
Develop application to consume Google's search / Google's Map RESTful Web service.

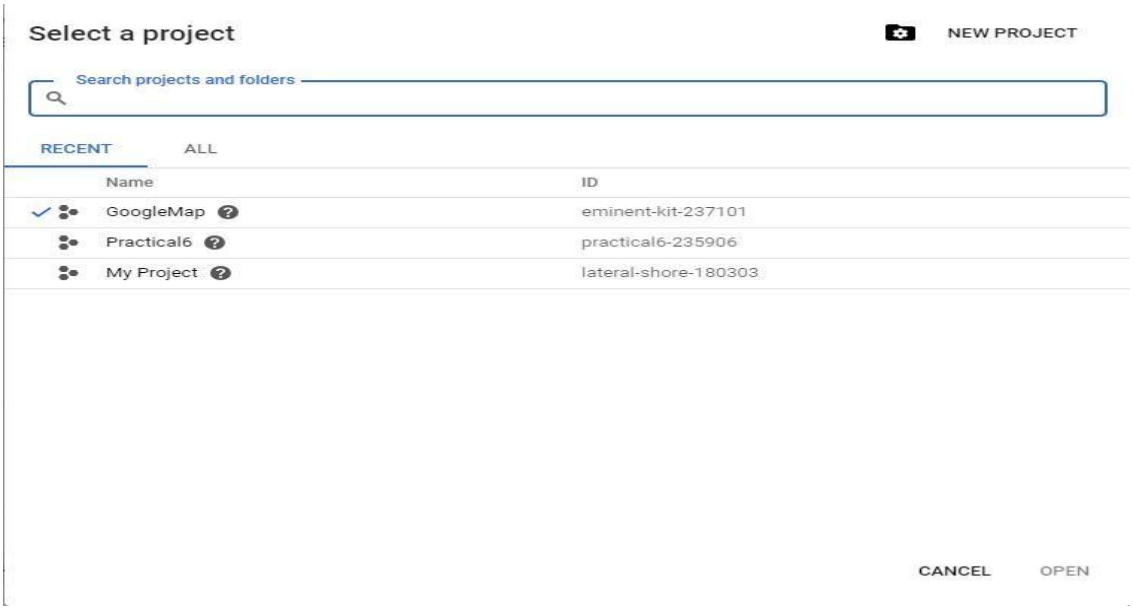
- Develop application to consume Google's search / Google's Map RESTful Web service.
- 1.First of all we need to create an Java Web Application with any name, let it be GoogleMap here using Netbeans IDE.
- 2.The code inside the input.jsp will be similar to this input.jsp Input.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
     </head>
    <body>
        <form action="index.jsp">
            <pre>
                Enter latitude:<input type="text" name="t1" />
                Enter longitude:<input type="text" name="t2" />
                <input type="submit"
            value="Show" /> </pre>
        </form>
    </body>
</html>
```

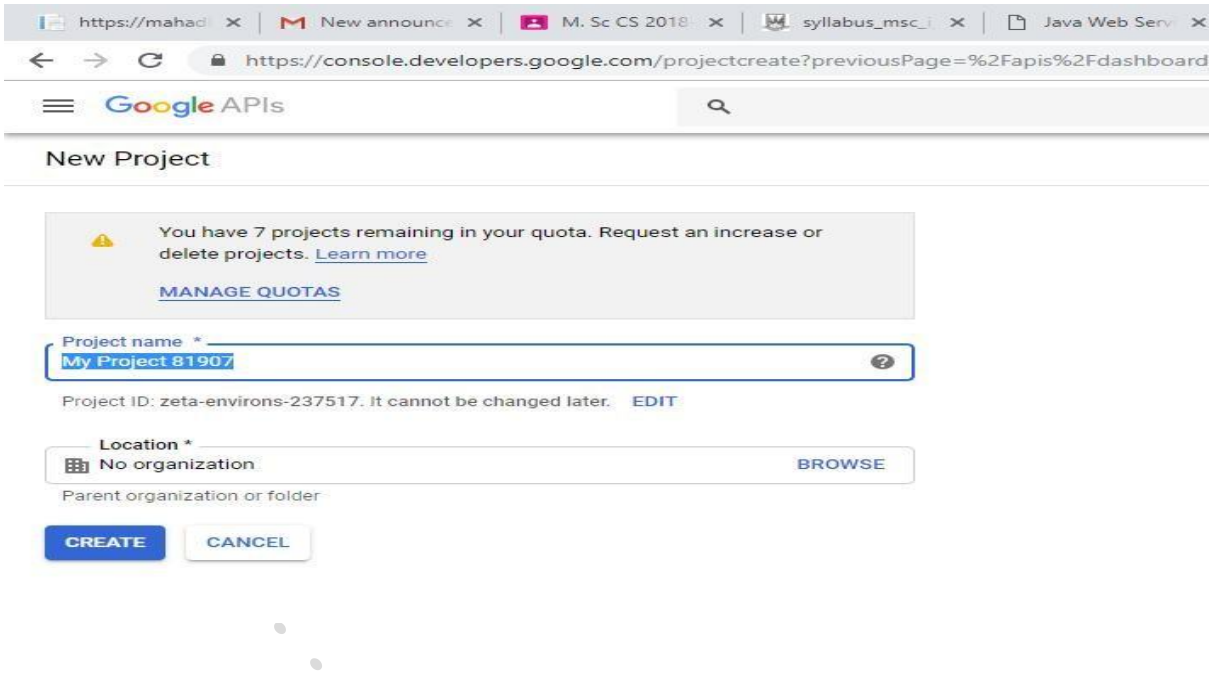
3. Before running the application we need the Google API key. The steps are shown here: - Visit Google APIs Console (https://console.developers.google.com, you have to login with your Google account).



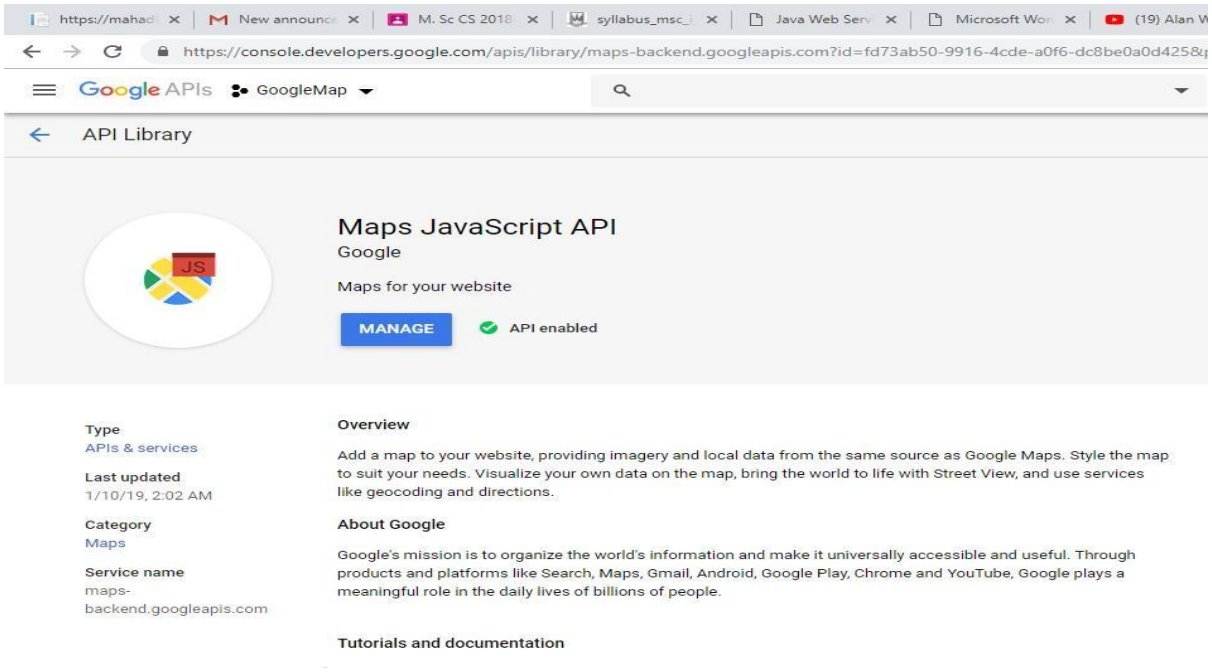
- Create a new API Project.



- Enter the name to your project.



- Enable the Google Maps API V3.



4. Create another file index.jsp

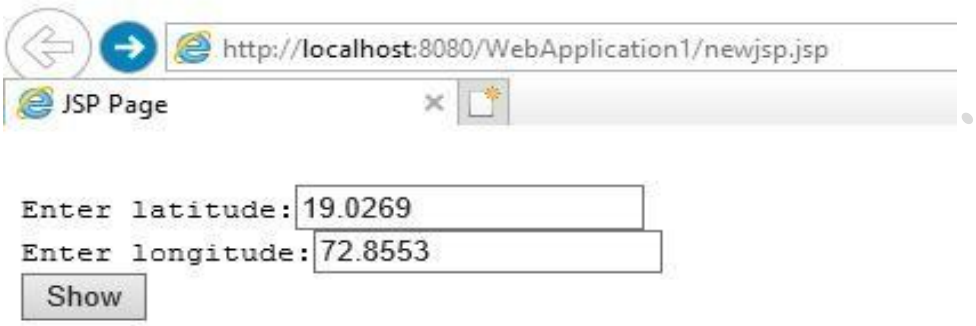
Index.jsp

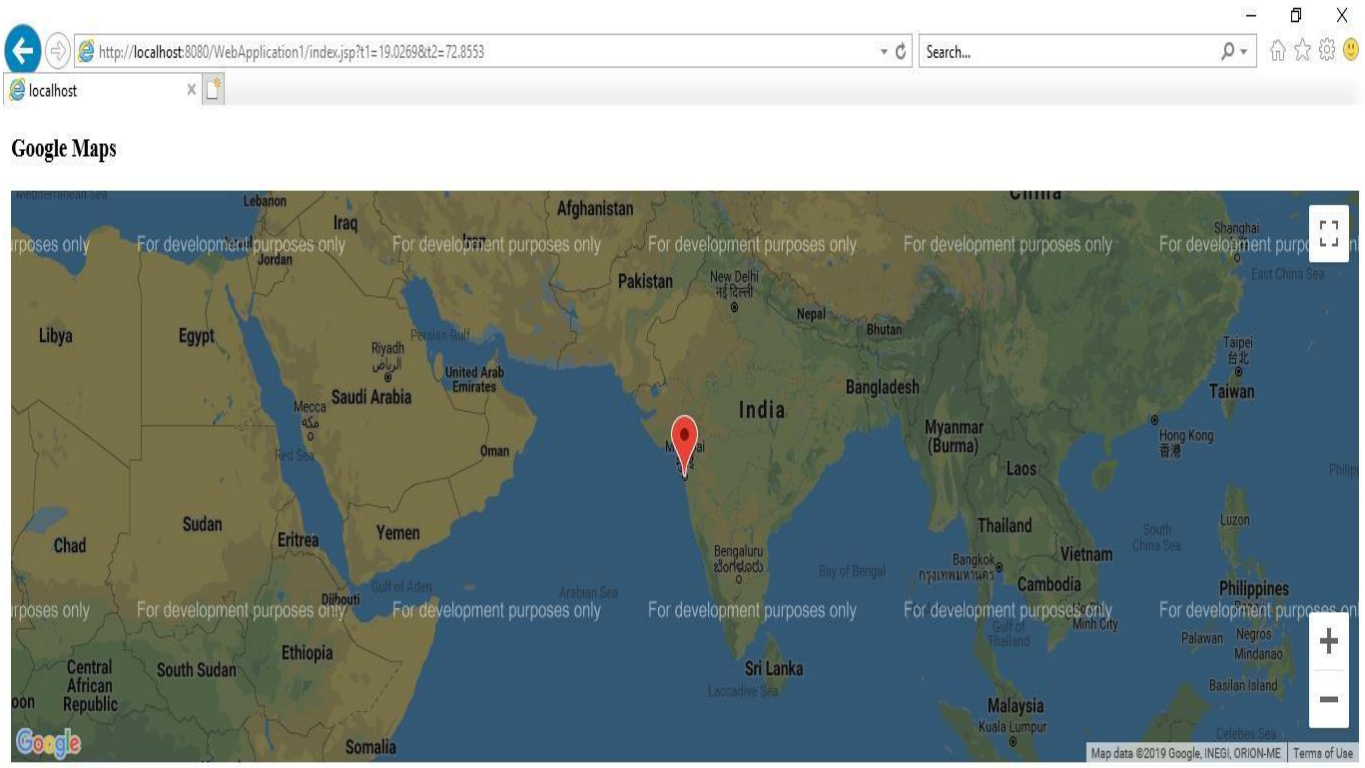
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

   <meta charset="UTF-8">

   <title>Google Maps</title>
```

```
    <style>
        #map {
            height: 400px;
            width: 100%;
        }
    </style>
</head>
<body>
    <%
        double lati = Double.parseDouble(request.getParameter("t1"));
        double longi = Double.parseDouble(request.getParameter("t2"));
    %>

    <h3>Google Maps</h3>
    <div id="map"></div>

    <script lang="javascript">
        function initMap() {
            var info = { lat: <%= lati %>, lng: <%= longi %> };

            var map = new google.maps.Map(document.getElementById('map'), {
                zoom: 4,
                center: info
            });

            var marker = new google.maps.Marker({
                position: info,
                map: map
            });
        }
    </script>
```

```
    <script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"></script>

</body>
```

Output :-






```
</html>
```

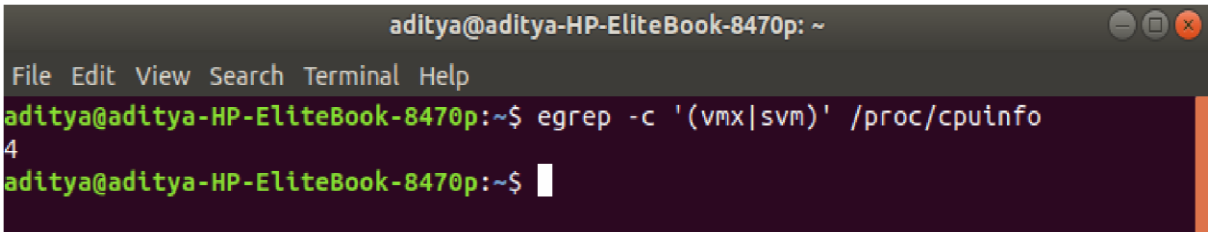
Practical - 5
Installation and Configuration of virtualization using KVM.

Steps to install Virtualization using KVM

To create a virtual machine first we need to ensure that virtualization is enabled on our system. It is mandatory to create virtual machines. There are multiple ways to check if virtualization is enabled,

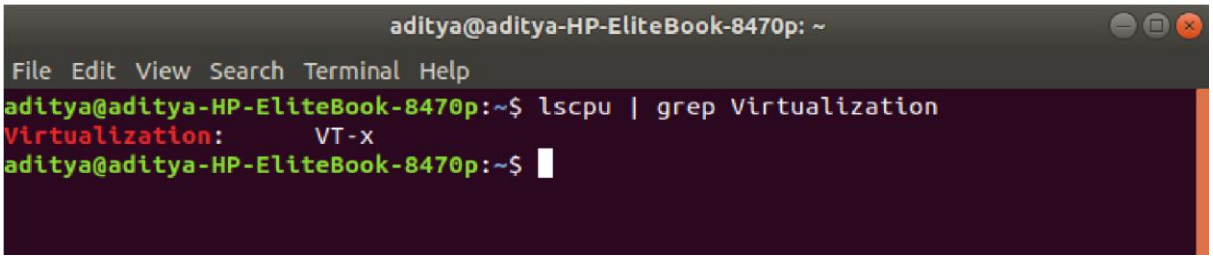
$ egrep -c '(vmx|svm)' /proc/cpuinfo

The above command /proc/cpuinfo gives information about the processor. The output of the command will be a number. The output number 1 or more than that represents that virtualization is enabled, output 0 says you need to enable virtualization on your system.



Virtualization Inability $ lscpu | grep Virtualization

This command is used to check which type of virtualization your processor supports. If the system contains a CPU with Intel VT support, the above command will provide the following output



Virtualization Type

Installing KVM on Ubuntu

$ sudo apt-get install virt-manager

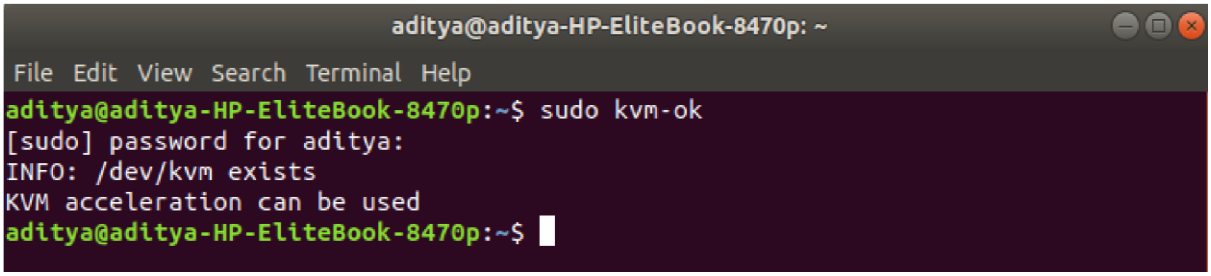
Now we know our system is capable of creating and running virtual machines, it's time to install tools which will create our virtual machines. To install KVM and other KVM dependencies such as virt-manager, bridge-utils enter command:

$ sudo apt -y install bridge-utils cpu-checker libvirt-clients libvirt-daemon qemu qemu-kvm

1. bridge-utils: The bridge-utils package contains a utility needed to create and manage bridge devices. This is useful in setting up networks for a hosted virtual machine (VM).
2. cpu-checker: Outputs the specifications of CPU

3. libvirt-clients: a toolkit to manage virtualization platforms/clients and hypervisors
4. qemu: A program that can run the operating system of the machine on different machines
5. qemu-kvm: Runs process using KVM module

All dependencies are installed now run command its time to check if KVM ins installed successfully: *$ sudo kvm-ok*

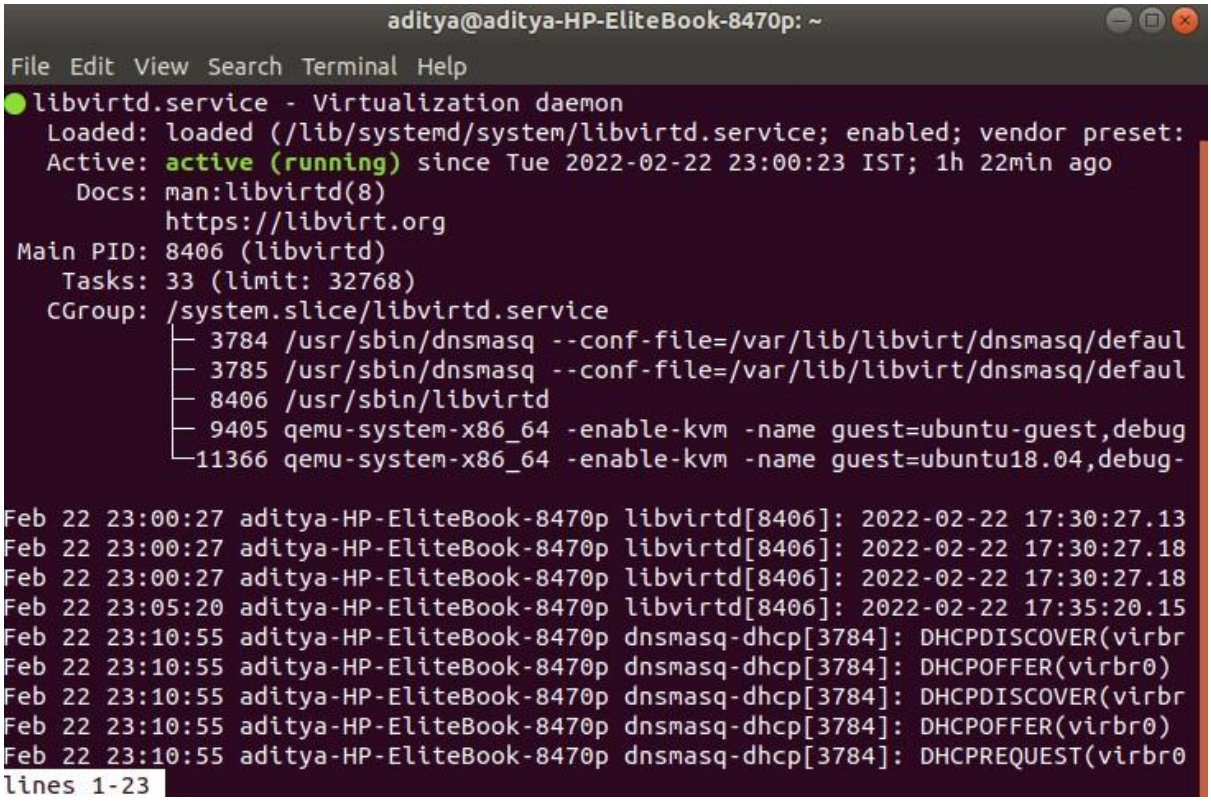


Check if KVM is Installed properly

Also, we need to confirm if the virtualization daemon – libvritd-daemon – is running, to do so enter the command.

$ sudo systemctl status libvirtd

If the output is not active: running you need to start daemon thread



If the daemon thread is not running enter the following command to start the thread,

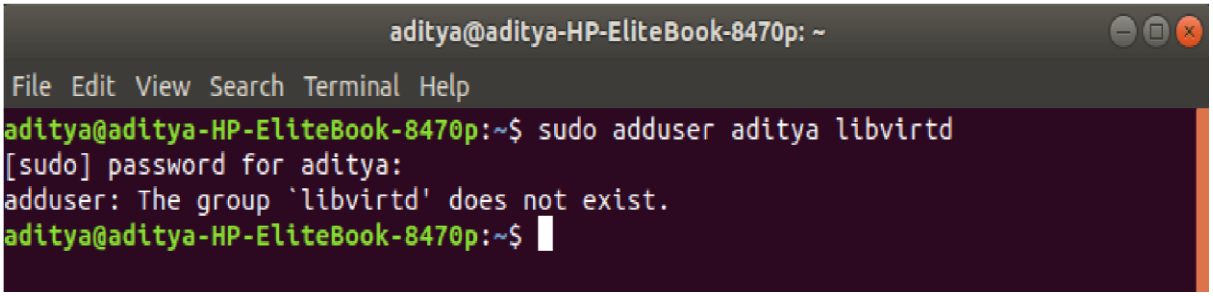
$ sudo systemctl enable –now libvirtd

Adding a user to KVM

In this part, we are going to create a user for KVM. To prevent root user from using KVM and root user is only available when root user is a part of libvirt/libvirtd group. To add a new user to KVM, use the following command,

sudo adduser [username] libvirtd

[username] enter the username of your choice if the output is



Adding User to KVM

If your KVM is already a member of the non-root user and serves the same purpose as libvirtd group then you don't need to add yourself to the group.

Creating Virtual Machine

There are two ways to create a virtual maclsne

1. Using the command line
2. Using the graphical interface

- **Create a Virtual Machine via Command Line**

virt-install is a command which is used to create virtual machines in Linux, following is the command which creates a VM.

sudo virt-install –name=ubuntu-guest –os-variant=ubuntu20.04 –vcpu=2 –ram=2048 –graphics none –location=[local path to iso file] –network bridge:vibr0

The above command creates a Ubuntu virtual machine with version 20.04 and the name ubuntu-guest.

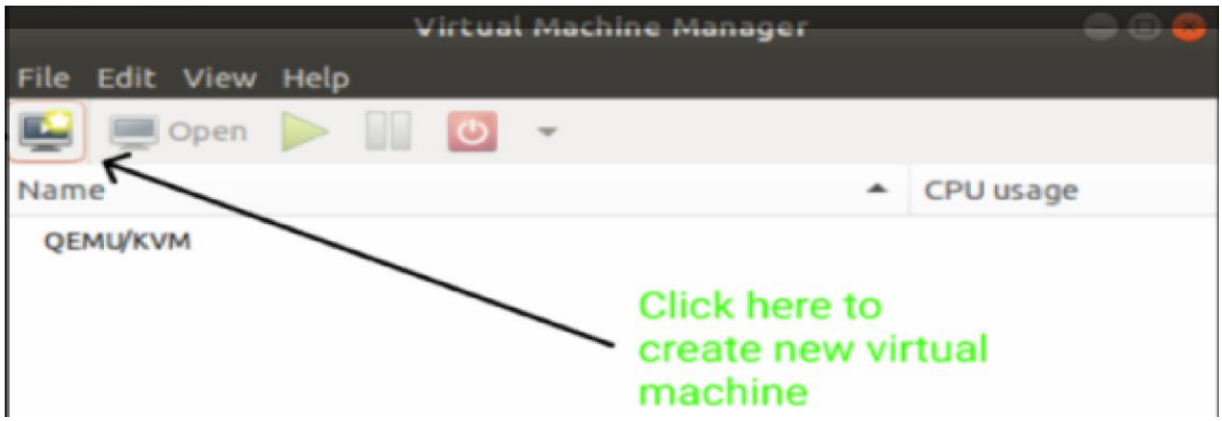
1. Name: specify the name of the virtual machine being created
2. vcpu: Number of virtual CPUs to configure for the guest.
3. Ram: Memory to allocate for guest instance in megabytes. According to your machine, you can specify the given memory of the VM.
4. Graphics spice: If no graphics option is specified, "virt-install" will default to '–graphics vnc' if the display environment variable is set, otherwise '–graphics none' is used.
5. Location: location of iso file on which virtual machine will be built. It can be the path to an ISO image, or to a CDROM device. It can also be a URL from which to fetch/access a minimal boot ISO image.
6. Network-bridge: Connect the guest to the host network, Connect to a bridge device in the host called "BRIDGE".

Create a Virtual machine using a graphical interface

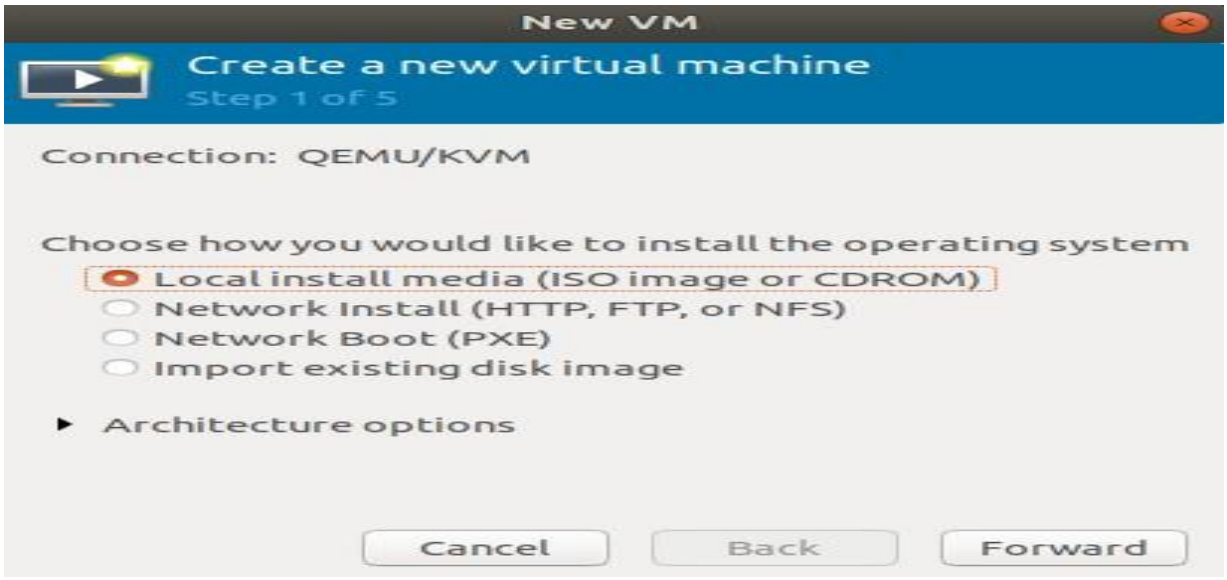
If you are not much familiar with the command line don't worry there's another way to create a virtual machine using a tool called virt-manager you can easily create virtual machines.

Steps to create VM using virt-manager,

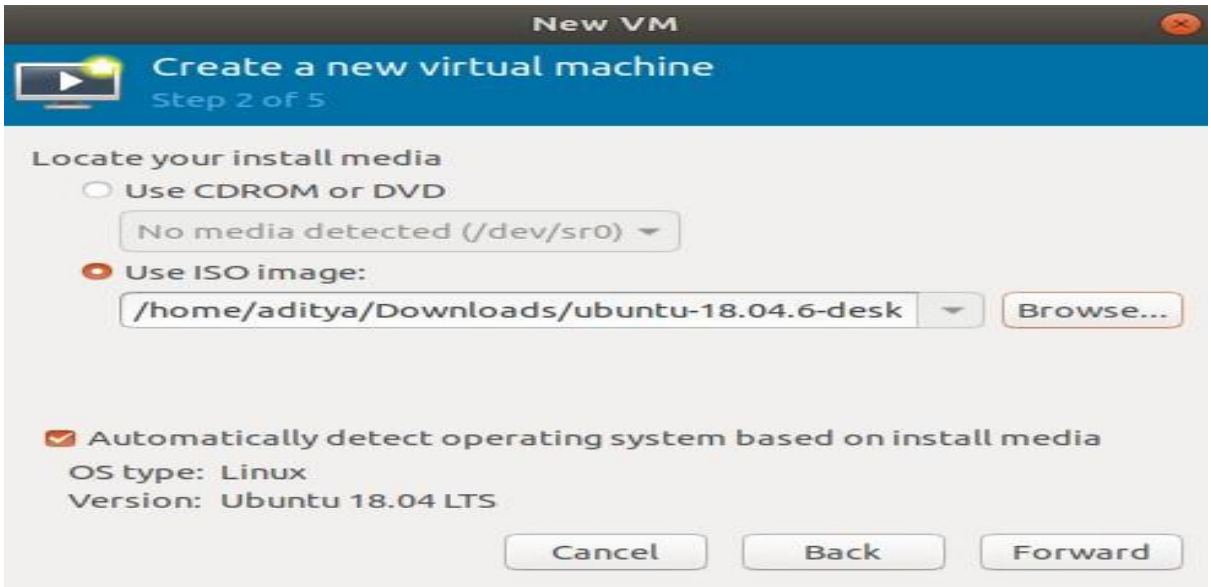
Step 1: Launch virt-manager



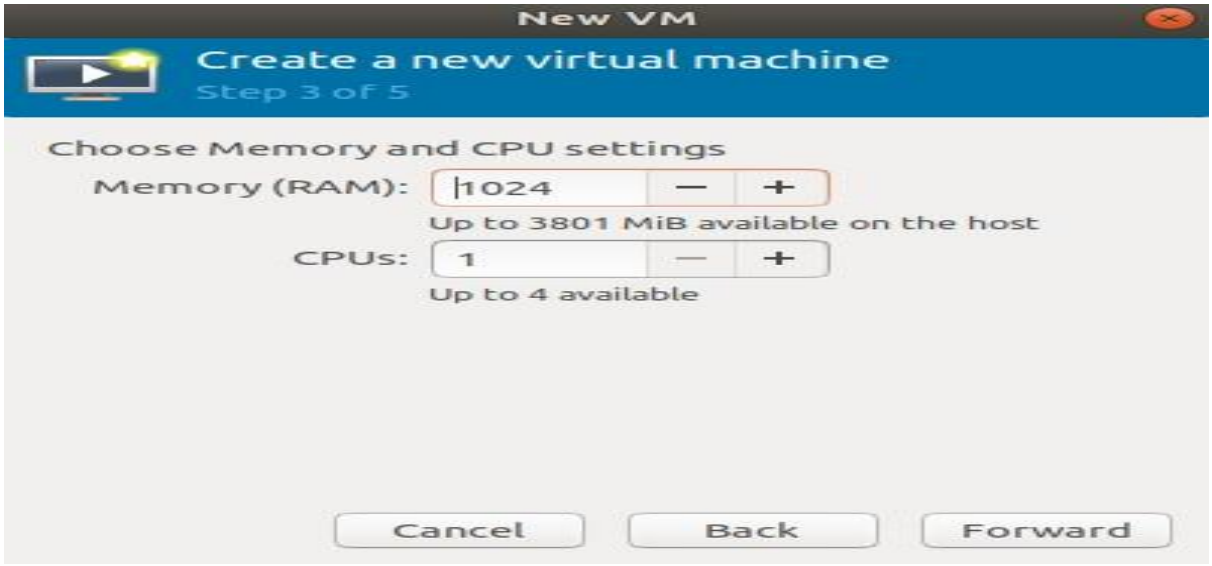
Step 2: Choose installation media it can be an iso file of OS, you can install from the network or can be a disk copy



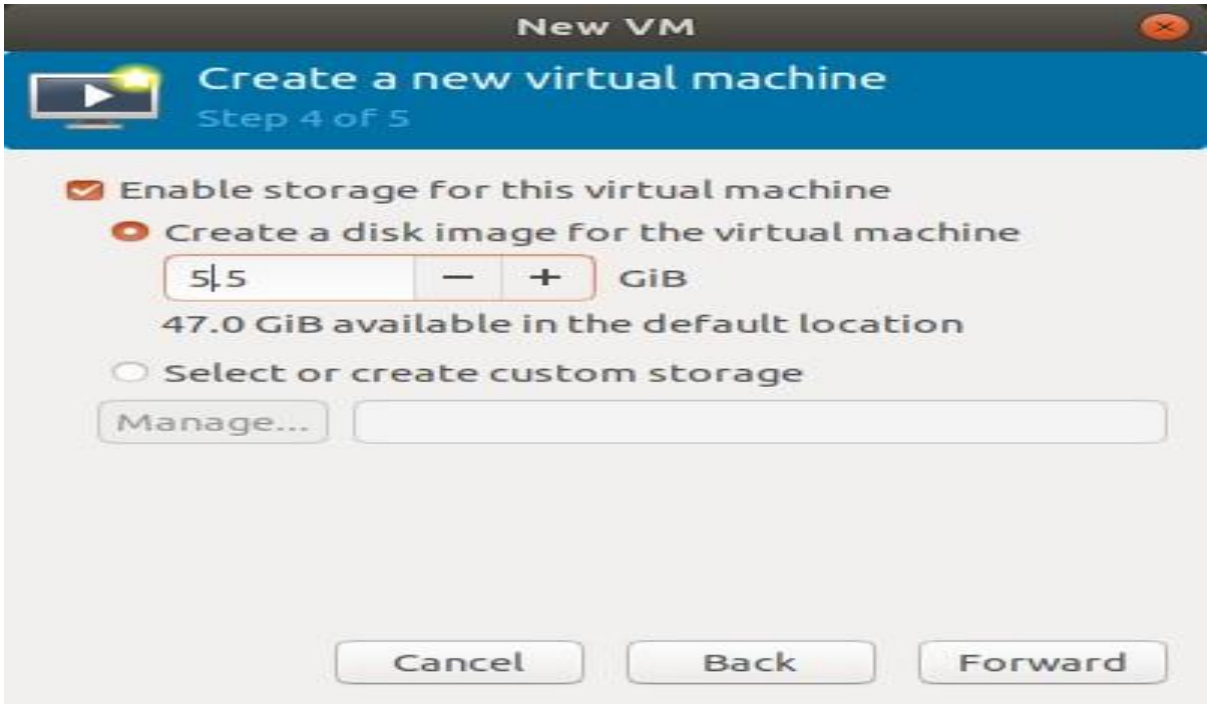
Step 3: Enter the path/URL of your file click next



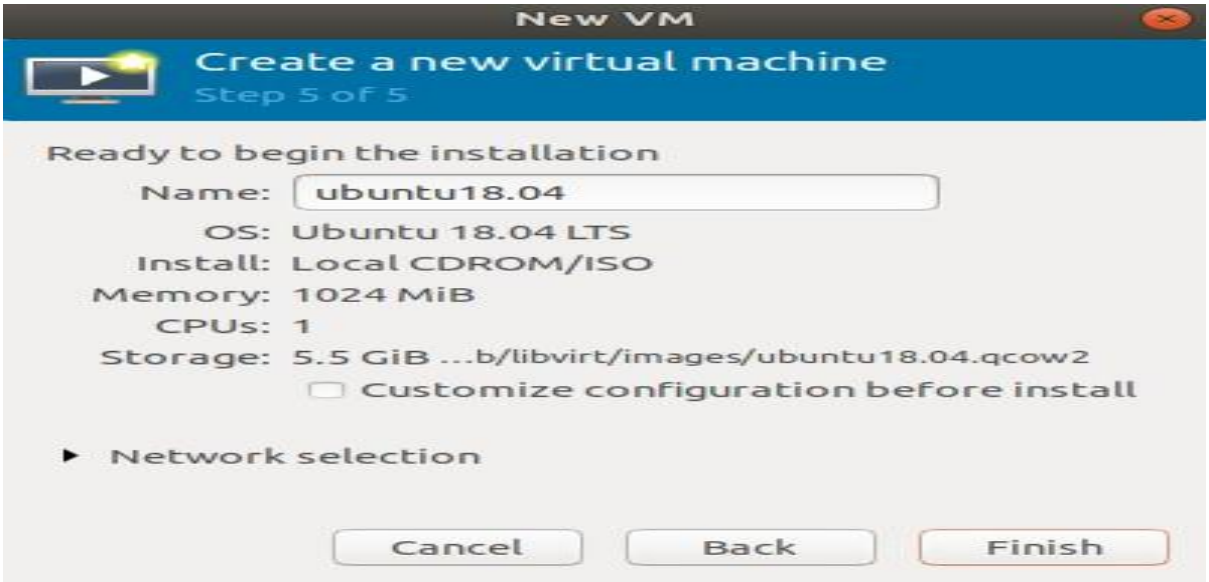
Step 4: Enter memory and CPU requirements



Step 5: Enter required disk space



Step 6: Enter the name of VM and check specifications and hit the finish



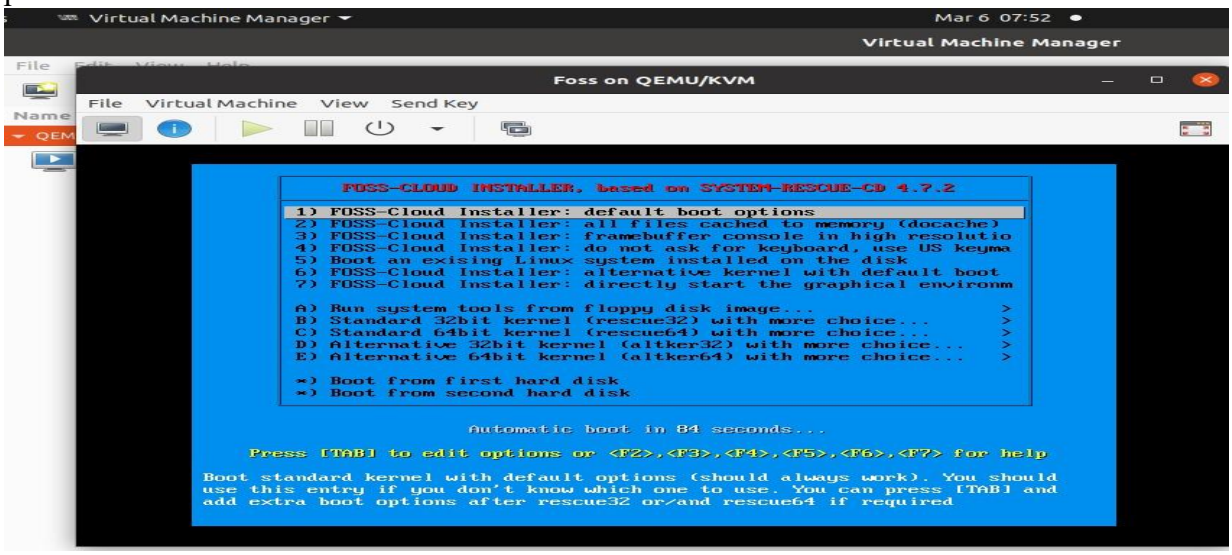
Now the installations should be completed and the virtual machine should start running.

Conclusion: Hence we have successfully installed and Configured virtualization using KVM.

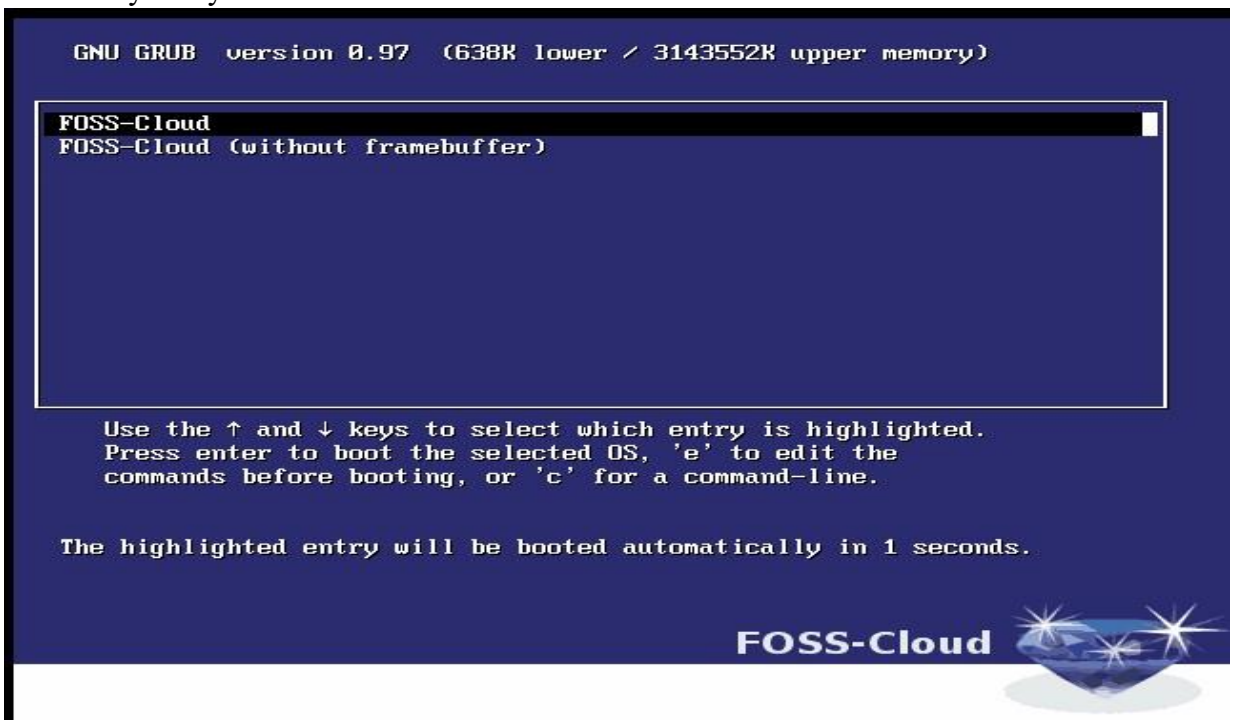
Practical - 6
Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Creating Virtual Machine or Storage

Steps:

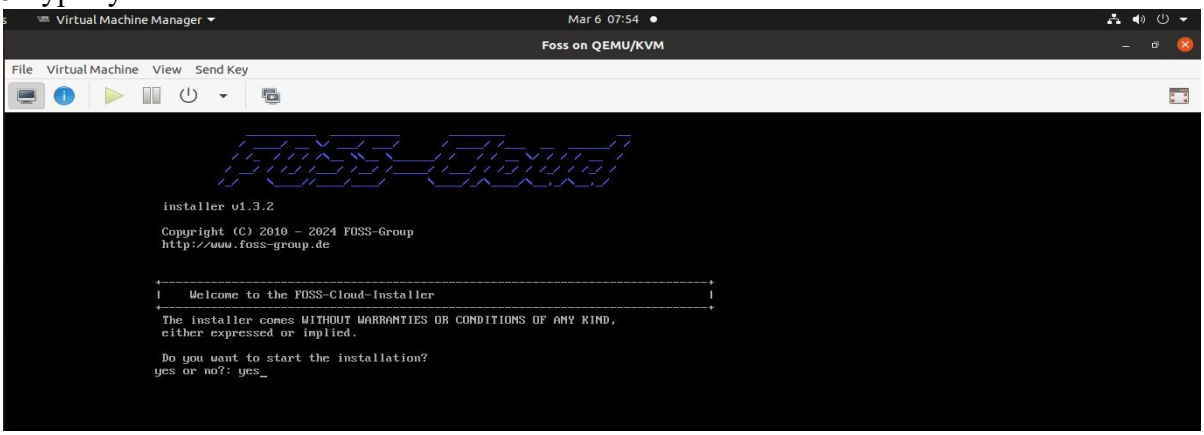
1. Start VMWARE workstation player 15 → select Open virtual machine then→ Browse a foss cloud iso file→ then select Operating System: Other Version: Other, Name: FOSS.
2. Then click on edit virtual machine and select Processor: 2 or 3, Memory: expand to 165 GM.
3. Once the foss Cloud is launched select “Foss-Cloud installer :default boot options” and press Enter



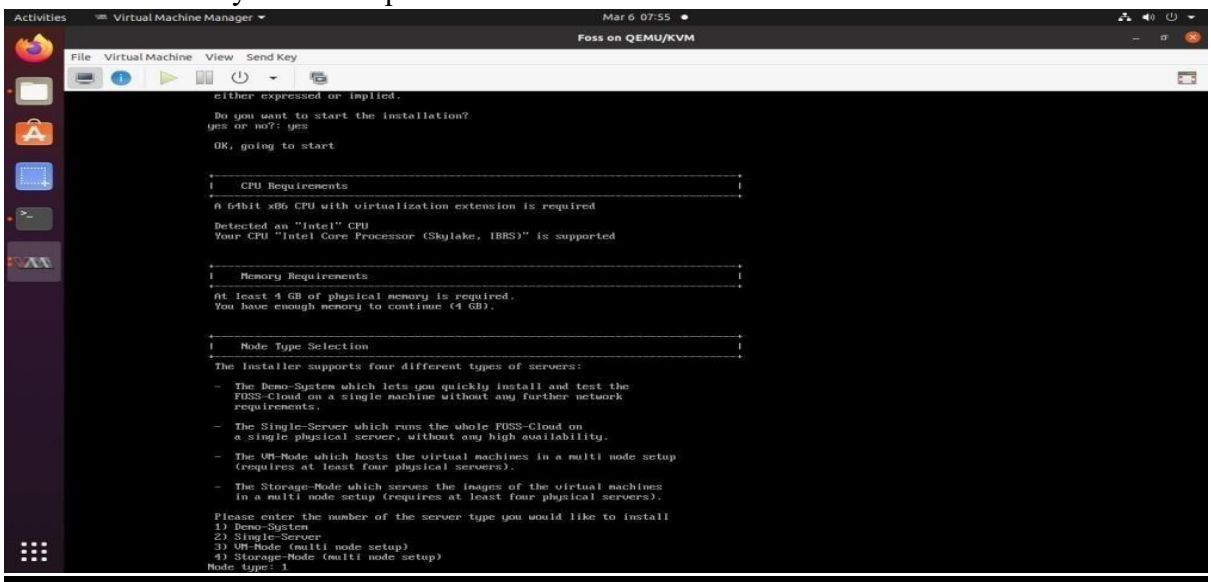
4.Reboot your system and select foss cloud

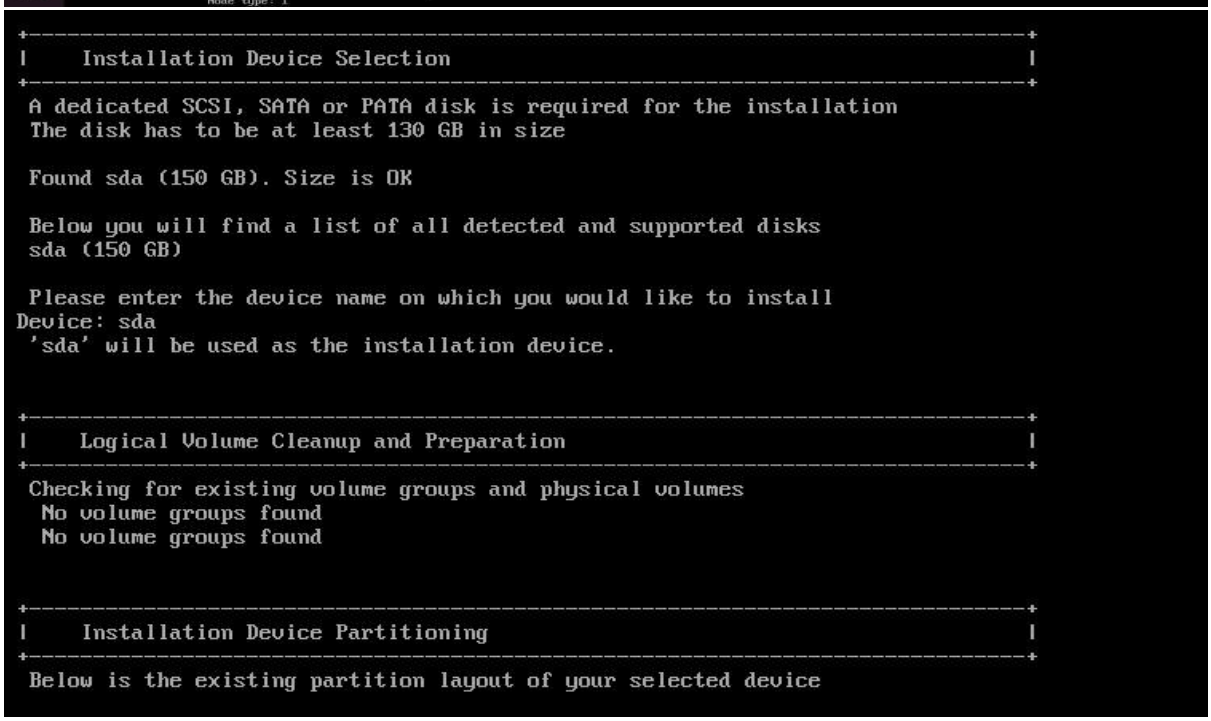


5.Types yes to start the installation



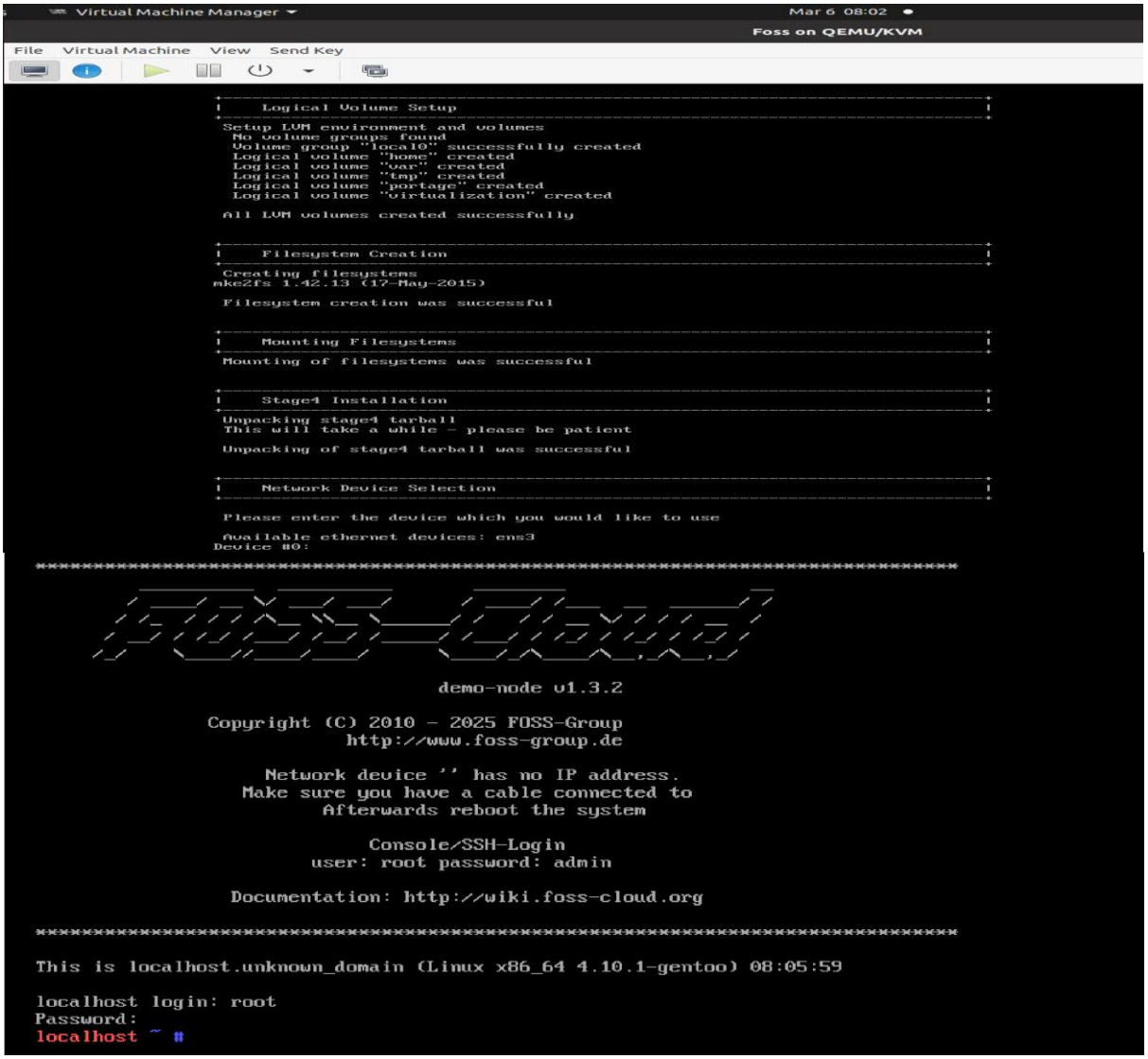
6.Select “Demo System” and press Enter



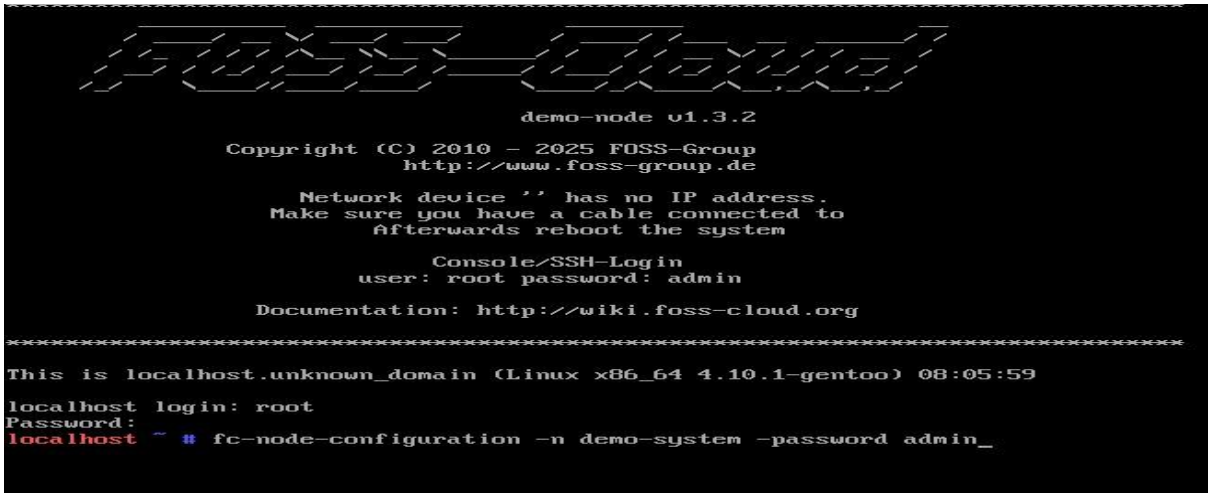


7.Enter the Device name as “sda” and press enter

8.Put credential of foss cloud and start foss cloud



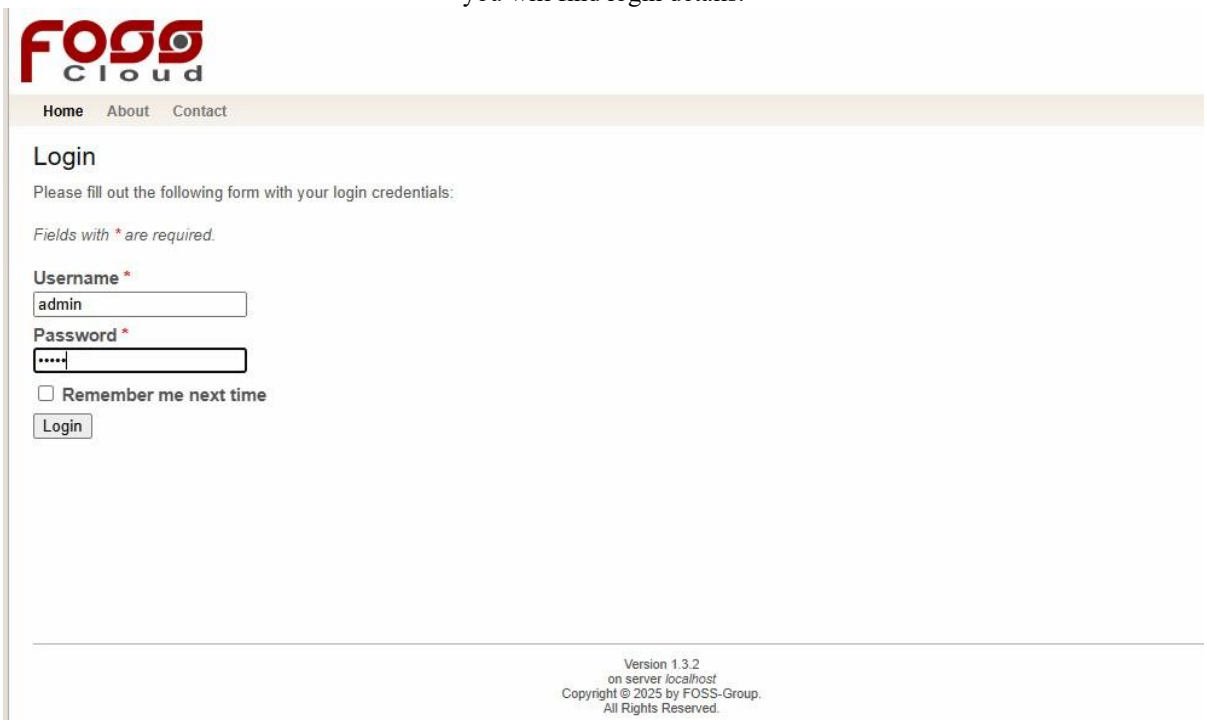
9. Then Execute below mention command:- Fc-node-configuration –n demo-system –password admin



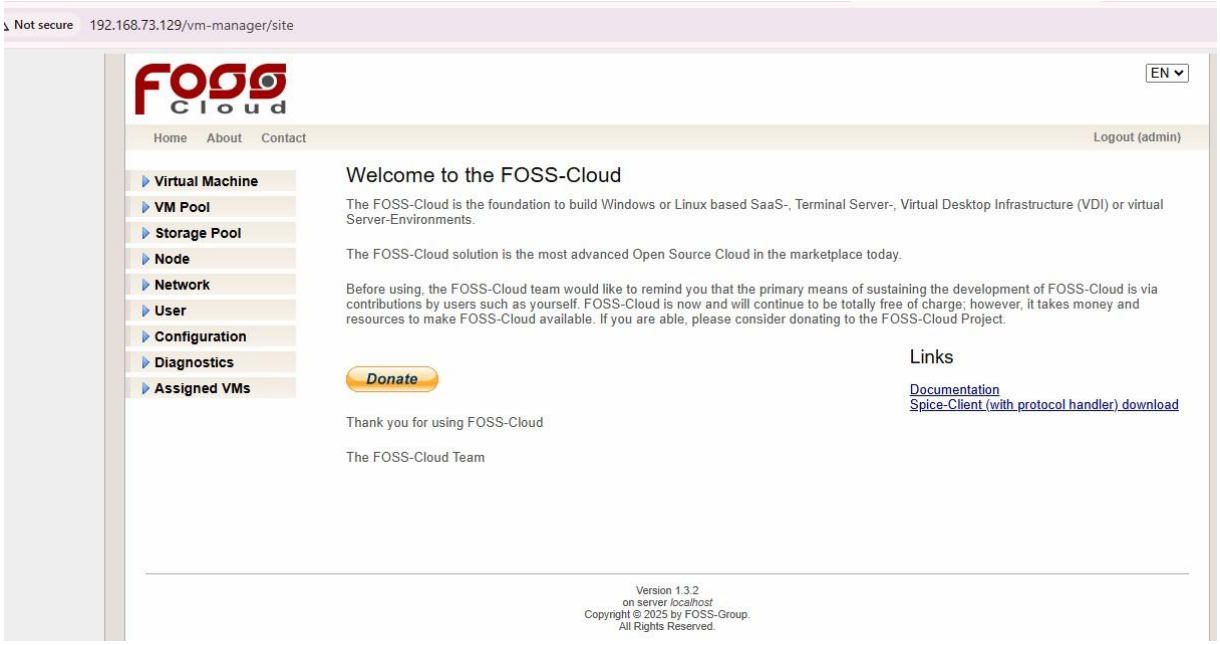
10.Then type ifconfig command to get the ip address

```
localhost ~ # ifconfig
eno16777728: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.73.129  netmask 255.255.255.0  broadcast 192.168.73.255
        ether 00:0c:29:8d:a1:dc  txqueuelen 1000  (Ethernet)
        RX packets 199  bytes 17394 (16.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 132  bytes 11637 (11.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 18  base 0x2000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1134  bytes 291057 (284.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1134  bytes 291057 (284.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vmbr0: flags=4419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST>  mtu 1500
        inet 172.31.255.1  netmask 255.255.255.0  broadcast 172.31.255.255
        ether 9a:bb:f6:59:fe:e0  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

localhost ~ #
```

11.Then go to any web browser in windows OS and type the IP Address → FOSS cloud web page will open and you will find login details.



12. After putting the login details you will get the below mention page

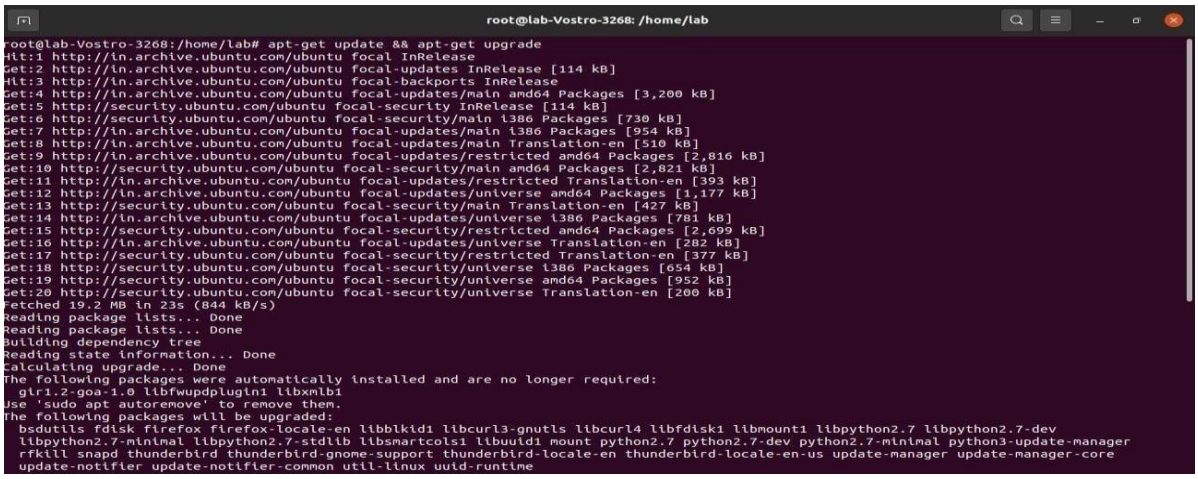


Practical - 7

Using AWS Flow Framework develop application that includes a simple workflow. Workflow calls an activity to print hello world to the console. It must define the basic usage of AWS Flow Framework, including defining contracts, implementation of activities and workflow coordination logic and worker programs to host them.

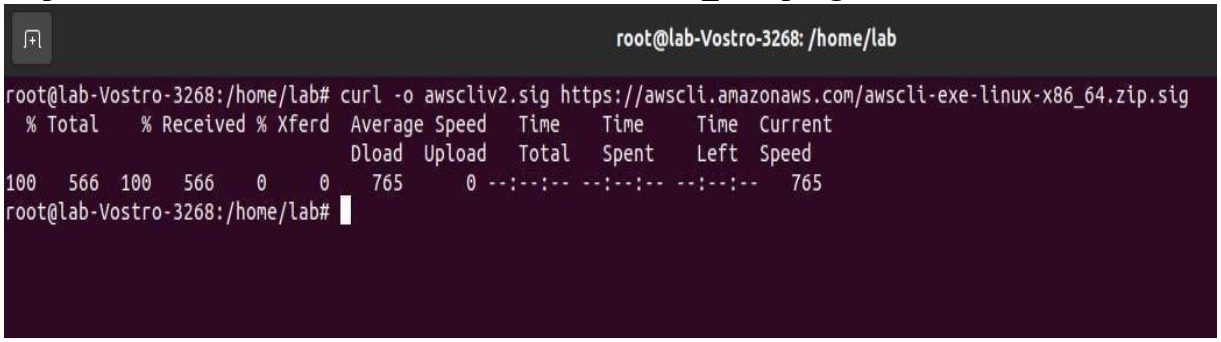
Step 1: Open Terminal and Update and Upgrade your system by

command sudo apt-get update && sudo apt-get upgrade



Step 2: Download awscliv2.zip with command
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

Step 3: Download awscliv2.sig file with command curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip.sig



Step 4: unzip awscliv2.zip with command unzip awscliv2.zip

```
root@lab-Vostro-3268:/home/lab# unzip awscliv2.zip
Archive:  awscliv2.zip
   creating: aws/
   creating: aws/dist/
  inflating: aws/THIRD_PARTY_LICENSES
  inflating: aws/install
  inflating: aws/README.md
   creating: aws/dist/awscli/
   creating: aws/dist/cryptography/
   creating: aws/dist/docutils/
   creating: aws/dist/lib-dynload/
  inflating: aws/dist/aws
  inflating: aws/dist/aws_completer
  inflating: aws/dist/libpython3.11.so.1.0
  inflating: aws/dist/_awscrt.abi3.so
  inflating: aws/dist/_cffi_backend.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/_ruamel_yaml.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/libz.so.1
  inflating: aws/dist/liblzma.so.0
  inflating: aws/dist/libbz2.so.1
  inflating: aws/dist/libffi.so.5
  inflating: aws/dist/libsqlite3.so.0
  inflating: aws/dist/base_library.zip
  inflating: aws/dist/lib-dynload/_pickle.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_hashlib.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha3.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_blake2.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha256.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_md5.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha1.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha512.cpython-311-x86_64-linux-gnu.so
```

Step 5: Run command sudo ./aws/install

```
root@lab-Vostro-3268:/home/lab#  sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

Step 6: Type command pip3 install aws-sam-cli

```
root@lab-Vostro-3268:/home/lab# pip3 install aws-sam-cli
Collecting aws-sam-cli
  Downloading aws_sam_cli-1.113.0-py3-none-any.whl (5.9 MB)
     |████████████████████████████████| 5.9 MB 21 kB/s
Collecting aws-lambda-builders==1.47.0
  Downloading aws_lambda_builders-1.47.0-py3-none-any.whl (130 kB)
     |████████████████████████████████| 130 kB 547 kB/s
Collecting tzlocal==5.2
  Downloading tzlocal-5.2-py3-none-any.whl (17 kB)
Collecting dateparser~=1.2
  Downloading dateparser-1.2.0-py2.py3-none-any.whl (294 kB)
     |████████████████████████████████| 294 kB 2.4 MB/s
Collecting Flask<3.1
  Downloading flask-3.0.2-py3-none-any.whl (101 kB)
     |████████████████████████████████| 101 kB 447 kB/s
Collecting boto3<2,>=1.29.2
  Downloading boto3-1.34.76-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 558 kB/s
Collecting pyopenssl~=24.1.0
  Downloading pyOpenSSL-24.1.0-py3-none-any.whl (56 kB)
     |████████████████████████████████| 56 kB 580 kB/s
Collecting requests~=2.31.0
  Using cached requests-2.31.0-py3-none-any.whl (62 kB)
Collecting click~=8.1
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 377 kB/s
Collecting watchdog==4.0.0
  Downloading watchdog-4.0.0-py3-none-manylinux2014_x86_64.whl (82 kB)
     |████████████████████████████████| 82 kB 113 kB/s
Collecting chevron~=0.12
  Downloading chevron-0.14.0-py3-none-any.whl (11 kB)
Collecting ruamel-yaml~=0.18.6
  Downloading ruamel.yaml-0.18.6-py3-none-any.whl (117 kB)
     |████████████████████████████████| 117 kB 363 kB/s
Collecting aws-sam-translator==1.86.0
```

Step 7: Type command sam init in terminal to launch Sam

CLISelect 1st option to use AWS Quick Ttart Templates

```
root@lab-Vostro-3268:/home/lab# sam init

        SAM CLI now collects telemetry to better understand customer needs.

        You can OPT OUT and disable telemetry collection by setting the
        environment variable SAM_CLI_TELEMETRY=0 in your shell.
        Thanks for your help!

        Learn More: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-telemetry.html

/usr/lib/python3/dist-packages/paramiko/transport.py:220: CryptographyDeprecationWarning: Blowfish has been deprecated and will be removed in
a future release
  "class": algorithms.Blowfish,

You can preselect a particular runtime or package type when using the `sam init` experience.
Call `sam init --help` to learn more.

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1
```

Step 8: Select Template no.1 Hello World Example

```
Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Data processing
        3 - Hello World Example with Powertools for AWS Lambda
        4 - Multi-step workflow
        5 - Scheduled task
        6 - Standalone function
        7 - Serverless API
        8 - Infrastructure event management
        9 - Lambda Response Streaming
        10 - Serverless Connector Hello World Example
        11 - Multi-step workflow with Connectors
        12 - GraphQLApi Hello World Example
        13 - Full Stack
        14 - Lambda EFS example
        15 - Hello World Example With Powertools for AWS Lambda
        16 - DynamoDB Example
        17 - Machine Learning
Template: 1
```

Step 9: Type “N” if it ask to use most popular runtime and package type

Open new terminal by pressing ctrl+shift+T and check for python version by command python –version

Select the option according to your python version in my case its option 19- python 3.11

```
Use the most popular runtime and package type? (Python and zip) [y/N]: n

Which runtime would you like to use?
        1 - aot.dotnet7 (provided.al2)
        2 - dotnet8
        3 - dotnet6
        4 - go1.x
        5 - go (provided.al2)
        6 - go (provided.al2023)
        7 - graalvm.java11 (provided.al2)
        8 - graalvm.java17 (provided.al2)
        9 - java21
        10 - java17
        11 - java11
        12 - java8.al2
        13 - nodejs20.x
        14 - nodejs18.x
        15 - nodejs16.x
        16 - python3.9
        17 - python3.8
        18 - python3.12
        19 - python3.11
        20 - python3.10
        21 - ruby3.2
        22 - rust (provided.al2)
        23 - rust (provided.al2023)
Runtime: 17
```

Step 10: Select package type as <u>Zip</u>

```
What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 1
```

Step 11: Now choose Yes option everytime it ask .

Give project name as per your preference in my case its <u>sam-app-test</u>

```
Based on your selections, the only dependency manager available is pip.
We will proceed copying the template using pip.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/N]: y
X-Ray will incur an additional cost. View https://aws.amazon.com/xray/pricing/ for more details

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch-application-insights.html [y/N]: y
AppInsights monitoring may incur additional cost. View https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/appinsights-what-is.html
#appinsights-pricing for more details

Would you like to set Structured Logging in JSON format on your Lambda functions?  [y/N]: y
Structured Logging in JSON format might incur an additional cost. View https://docs.aws.amazon.com/lambda/latest/dg/monitoring-cloudwatchlogs.
html#monitoring-cloudwatchlogs-pricing for more details

Project name [san-app]: san-app-test

Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a moment)

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app-test
    Runtime: python3.8
    Architectures: x86_64
    Dependency Manager: pip
    Application Template: hello-world
    Output Directory: .
    Configuration file: san-app-test/samconfig.toml

    Next steps can be found in the README file at sam-app-test/README.md
```

Step 12: Now one folder will be created by your provided project name go into that folder by command cd (folder name)

After entering the project folder we will invoke the HelloWorldFunction by using commandsam local invoke „HelloWorldFunction"

```
root@lab-Vostro-3268:/home/lab/sam-app-test# sam local invoke 'HelloWorldFunction'
/usr/lib/python3/dist-packages/paramiko/transport.py:220: CryptographyDeprecationWarning: Blowfish has been deprecated and will be removed in
a future release
  "class": algorithms.Blowfish,
Invoking app.lambda_handler (python3.8)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-python3.8
Building image.....
.....
.....
.....
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /home/lab/sam-app-test/hello_world as /var/task:ro,delegated, inside runtime container
START RequestId: 065fefd3-5d41-4b87-bb31-8d820fb635e6 Version: $LATEST
END RequestId: 4b9baa5c-2523-443d-b340-f86e2b139f6a
REPORT RequestId: 4b9baa5c-2523-443d-b340-f86e2b139f6a  Init Duration: 0.06 ms  Duration: 99.11 ms      Billed Duration: 100 ms Memory Size: 1
28 MB   Max Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}
root@lab-Vostro-3268:/home/lab/sam-app-test#
```

It should give you StatusCode:200

Use command sudo snap install docker if docker error occurs.

Step 13: Type command sam local start-api this will give you URL open it in any browser.

```
root@lab-Vostro-3268:/home/lab/sam-app-test# sam local start-api
/usr/lib/python3/dist-packages/paramiko/transport.py:220: CryptographyDeprecationWarning: Blowfish has been deprecated and will be removed in
a future release
  "class": algorithms.Blowfish,
Initializing the lambda functions containers.
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-python3.11
Building image.....
.....
Using local image: public.ecr.aws/lambda/python:3.11-rapid-x86_64.

Mounting /home/lab/sam-app-test/hello_world as /var/task:ro,delegated, inside runtime container
Containers Initialization is done.
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
You can now browse to the above endpoints to invoke your functions. You do not need to restart/reload SAM CLI while working on your functions,
changes will be reflected instantly/automatically. If you used sam build before running local commands, you will need to re-run sam build for
the changes to be picked up. You only need to restart SAM CLI if you update your AWS SAM template
2024-04-03 09:26:18 WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
2024-04-03 09:26:18 Press CTRL+C to quit
Invoking app.lambda_handler (python3.11)
Reuse the created warm container for Lambda function 'HelloWorldFunction'
Lambda function 'HelloWorldFunction' is already running
START RequestId: 0652f01c-89a7-43d9-85fb-3844c8f1a32b Version: $LATEST
END RequestId: b229bc70-c5b6-4bf2-b9a3-97b2cf9340a1
REPORT RequestId: b229bc70-c5b6-4bf2-b9a3-97b2cf9340a1  Init Duration: 0.04 ms  Duration: 1153.99 ms    Billed Duration: 1154 ms        Memory
 Size: 128 MB   Max Memory Used: 128 MB

No Content-Type given. Defaulting to 'application/json'.
2024-04-03 09:26:38 127.0.0.1 - - [03/Apr/2024 09:26:38] "GET /hello HTTP/1.1" 200 -
2024-04-03 09:26:39 127.0.0.1 - - [03/Apr/2024 09:26:39] "GET /favicon.ico HTTP/1.1" 403 -
```

Output:

Practical - 8
Implementation of Openstack with user and private network creation.

THEORY: OpenStack is a cloud OS that is used to control the large pools of computing, storage, and networking resources within a data center. Open Stack is open-source and free software. This is basically used for cloud computing and deployed as an IaaS.

Components of OpenStack

Major components of OpenStack are given below:

Compute (Nova): Compute is a controller that is used to manage resources in virtualized environments. It handles several virtual machines and other instances that perform computing tasks.

Object Storage (Swift): To store and retrieve arbitrary data in the cloud, object storage is used. In Swift, it is possible to store the files, objects, backups, images, videos, virtual machines, and other unstructured data. Developers may use a special identifier for referring the file and objects in place of the path, which directly points to a file and allows OpenStack to manage where to store the files.

Block Storage (Cinder): This works in the traditional way of attaching and detaching an external hard drive to the OS for its local use. Cinder manages to add, remove, create new disk space in the server. This component provides the virtual storage for the virtual machines in the system.

Networking (Neutron): This component is used for networking in OpenStack. Neutron manages all the network-related queries, such as IP address management, routers, subnets, firewalls, VPNs, etc. It confirms that all the other components are well connected with OpenStack.

Dashboard (Horizon): This is the first component that the user sees in the OpenStack. Horizon is the web UI (user interface) component used to access the other back-end services.

Through individual API (Application programming interface), developers can access the OpenStack's components, but through the dashboard, system administrators can look at what is going on in the cloud and manage it as per their need.

Identity Service (Keystone): It is the central repository of all the users and their permissions for the OpenStack services they use. This component is used to manage identity services like authorization, authentication, AWS Styles (Amazon Web Services) logins, token-based systems, and checking the other credentials (username & password).

Image Service (Glance): The glance component is used to provide the image services to OpenStack. Here, image service means the images or virtual copies of hard disks. When we plan to deploy a new virtual machine instance, then glance allows us to use these images as templates. Glance allows virtual box (VDI), VMware (VMDK, OVF), Raw, Hyper-V (VHD) and KVM (qcow2) virtual images.

STEPS:

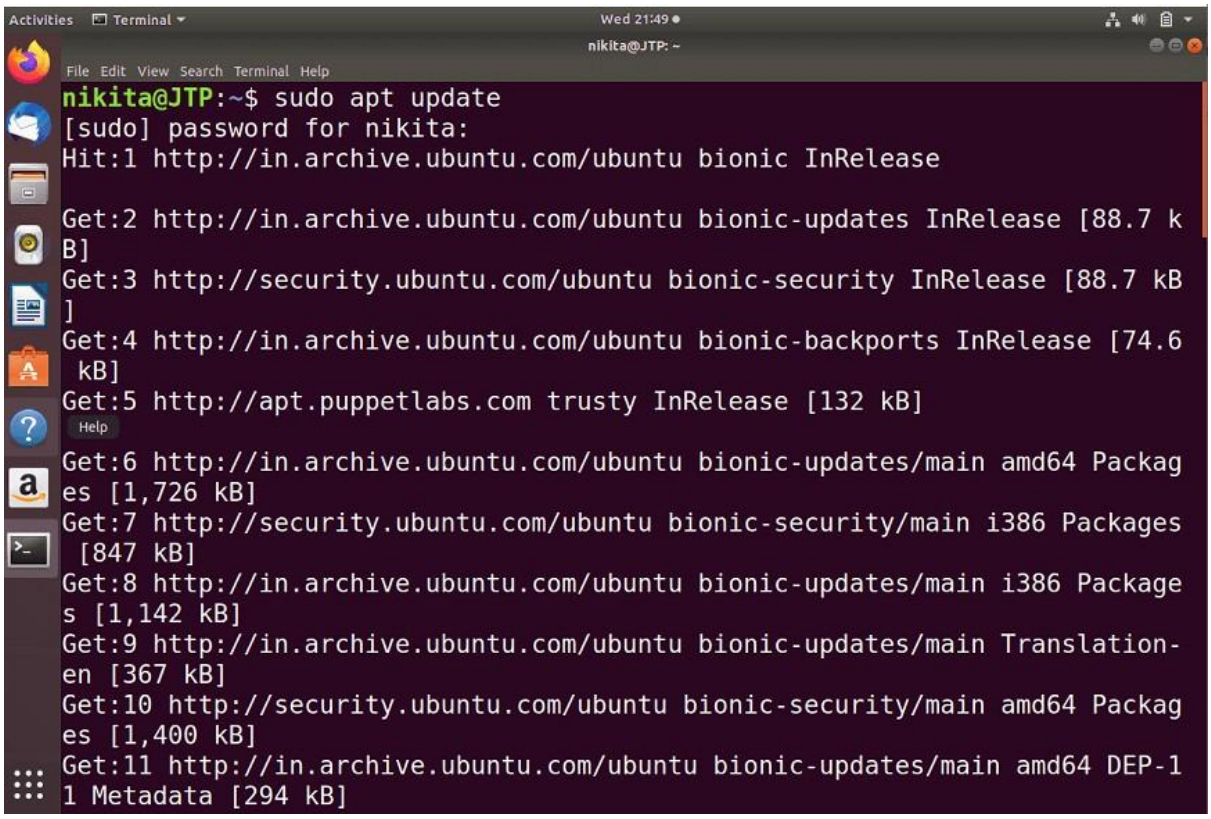
Step 1: Update Ubuntu System

Open the terminal and run the following command to ensure that the system is up to date :

$ sudo apt update **OR**

$ sudo apt -y upgrade **Sample**

Output :



Reboot the system after running the above command. To reboot the system, run the following command :

$ sudo reboot or

$ init 6

Step 2: Create Stack User

It is important that the devstack must run as a regular user (non-root user) with the sudo enabled.

To keep this note in mind, let's create a new user with the name "stack" and assign the sudo permissions or privileges. To create a stack user, run the following command in your terminal:

$ sudo useradd -s /bin/bash -d /opt/stack -m stack Output :

```
nikita@JTP:~$ sudo useradd -s /bin/bash -d /opt/stack -m stack
nikita@JTP:~$
```

Now, to assign the sudo privileges to the stack user, run the following command :

$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack Output :

```
nikita@JTP:~$ echo "stack ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/stack
stack ALL=(ALL) NOPASSWD:ALL
nikita@JTP:~$
```

You can switch to the 'stack' user by running the following command: $ sudo su - stack Output :

```
nikita@JTP:~$ sudo su - stack
stack@JTP:~$
```

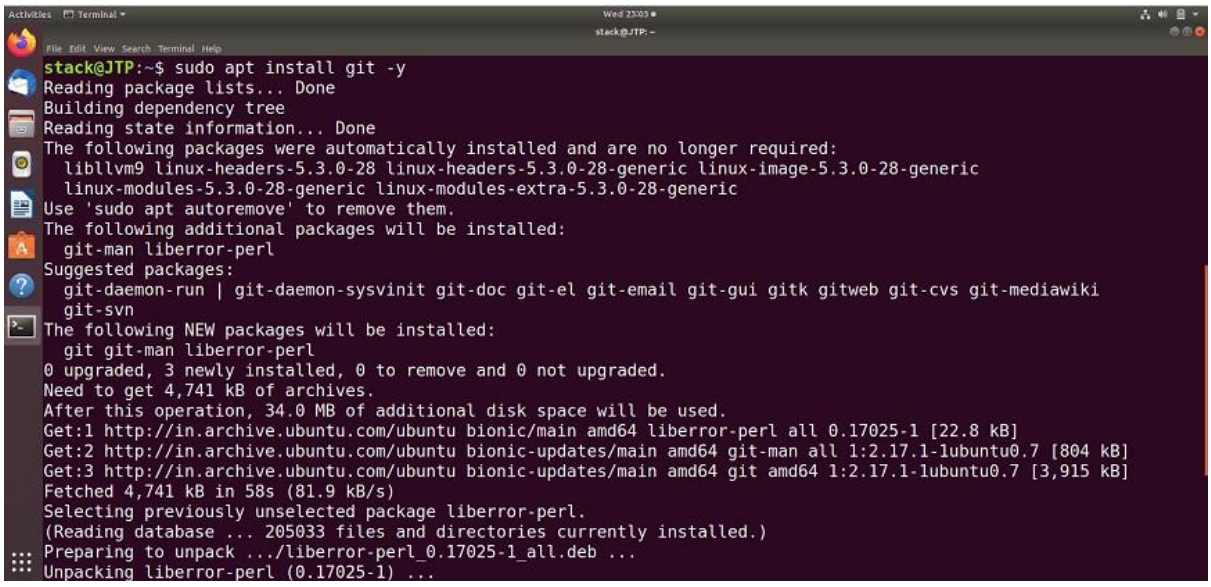
```
nmfc@nmfc-virtual-machine:~/Desktop$ echo "stack1 AL=(ALL) NOPASSWD:ALL" | sudo
 tee /etc/sudoers.d/stack
[sudo] password for nmfc:
stack1 AL=(ALL) NOPASSWD:ALL
nmfc@nmfc-virtual-machine:~/Desktop$ sudo apt install git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 563 not upgraded.
```

Step 3: Install the Git

In Most of the Ubuntu systems, git comes by default. But if git is missing on your system, then install it by running the following command:

$ sudo apt install git -y Sample

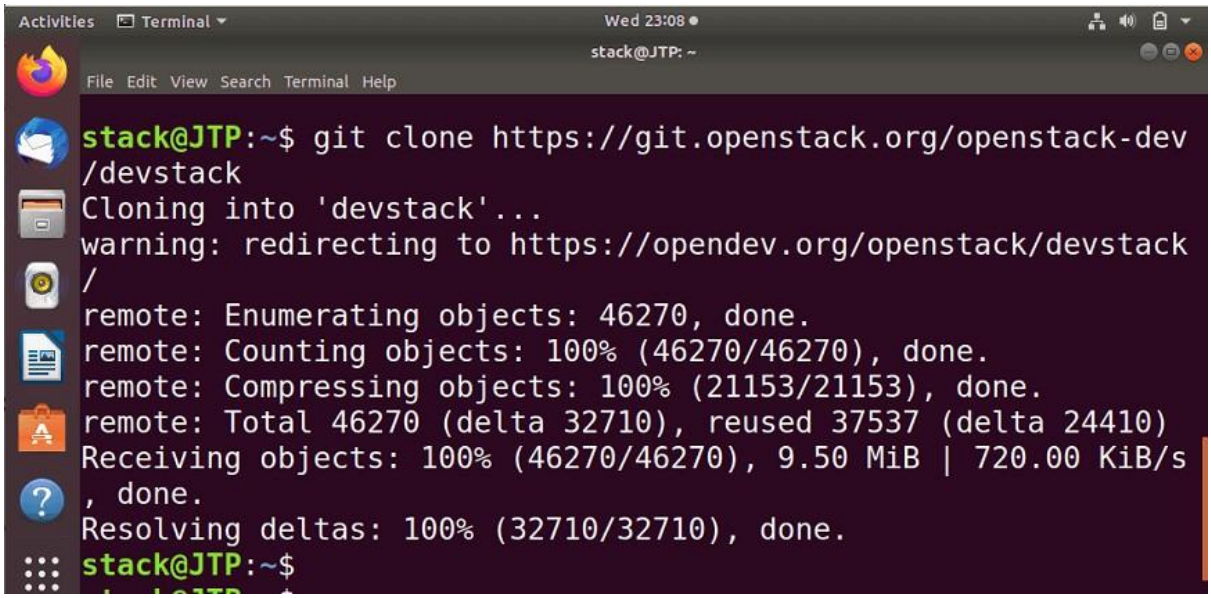
Output :



Step 4: Download OpenStack

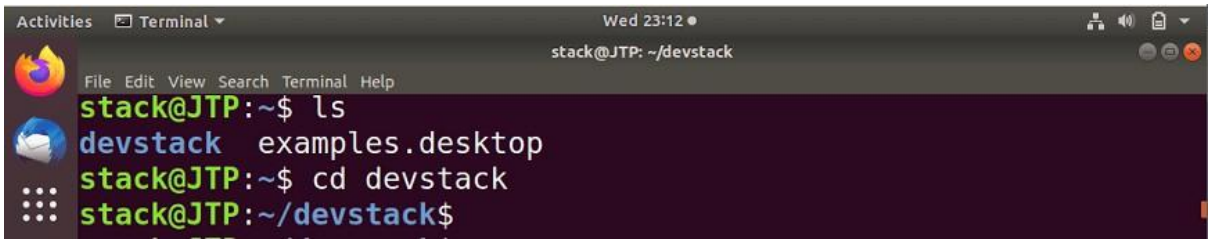
Once you install the git, use the git command to download the DevStack from Github.

$ git clone https://git.openstack.org/openstack-dev/devstack Output :

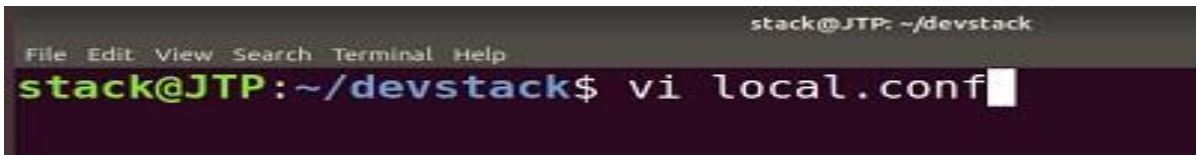


Step 5: Create a DevStack Configuration File

First of all, go to the devstack directory by running the following command : $ cd devstack Output :



Now, create a local.conf file in which you have to enter the four passwords and the host IP address : Output :



Copy the following line of content in the file :

[[local|localrc]]

# Password for KeyStone, Database, RabbitMQ and Service
ADMIN_PASSWORD=StrongAdminSecret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# Host IP - To get your Server or VM IP, run the 'ip addr' or 'ifconfig' command
HOST_IP=192.168.56.103 **Output :**

Press the ESC, then wq to save and then exit from the local.conf file. Here, ADMIN_PASSWORD is the password that we will use to log into the OpenStack login page. The default username for an OpenStack is 'admin'.

And HOST_IP is the IP address of your system. To get your Server or VM IP, run the 'ifconfig' or 'ip addr' command.

Step 6 : Install OpenStack with DevStack

To install and run the openstack, execute the following command : $

./stack.sh

DevStack will install the following components:

Compute Service (Nova)
Image Service- Glance
Identity Service-Keystone,
Block Storage Service - Cinder
OpenStack Dashboard - Horizon
Network Service - Neutron
Placement API - Placement
Object Storage - Swift

The installation will take about 10-20 minutes, mostly depends on your internet speed.

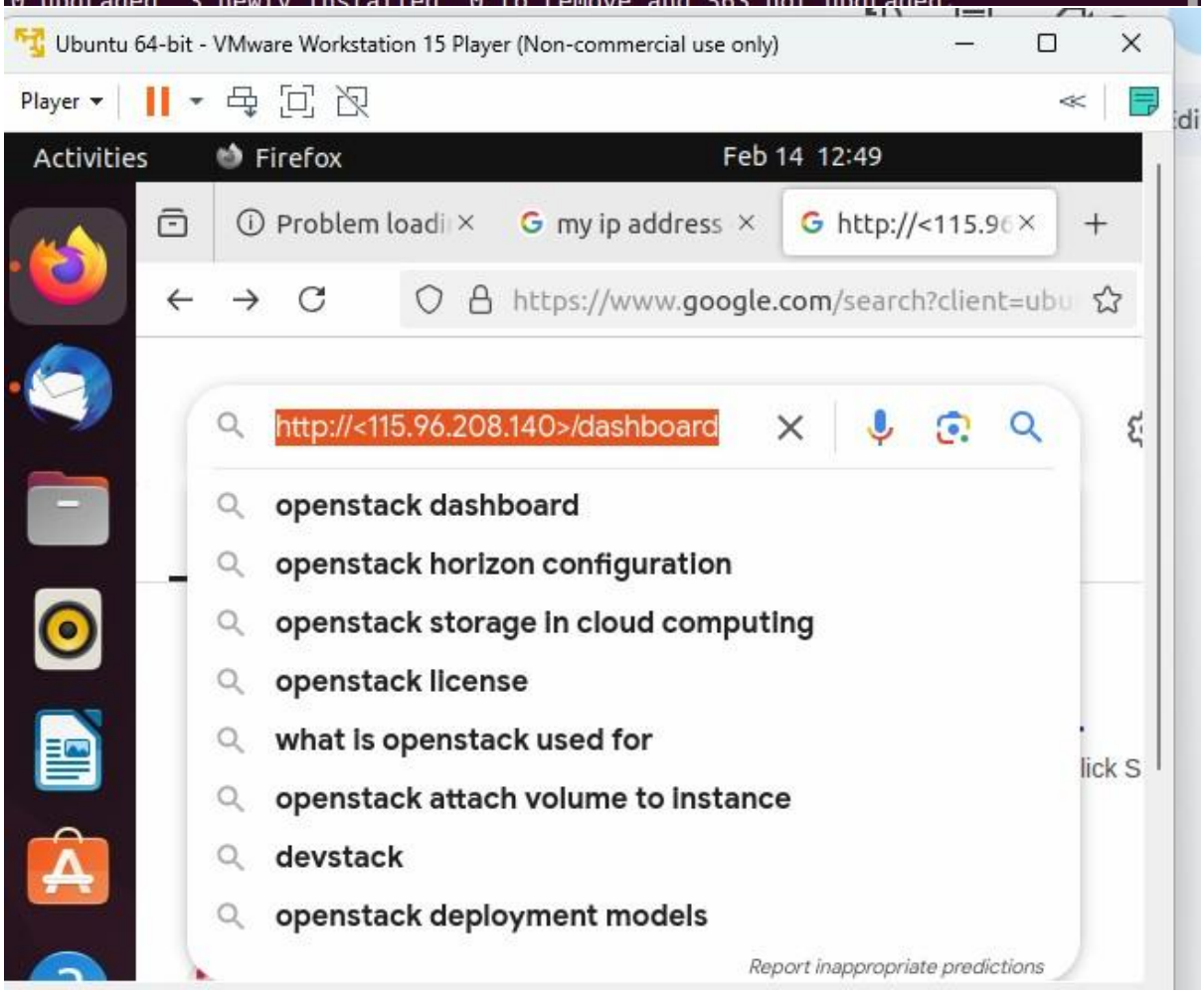
At the very end of the installation, you will get the host's IP address, URL for managing it and the username and password to handle the administrative task.

Step 7: Accessing OpenStack on a browser

Copy the horizon URL given in the installation output and paste it into your browser :

http://<IP Address>/dashboard

```
nmfc@nmfc-virtual-machine:~/Desktop$ echo "stack1 AL=(ALL) NOPASSWD:ALL" | sudo
 tee /etc/sudoers.d/stack
[sudo] password for nmfc:
stack1 AL=(ALL) NOPASSWD:ALL
nmfc@nmfc-virtual-machine:~/Desktop$ sudo apt install git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 563 not upgraded.
```



Conclusion: Hence we have successfully implemented Openstack with user and private network creation.