1)Currency Converter

Currency_converted.py

```python
from flask import Flask, request, jsonify

app = Flask(__name__)
@app.route('/convert')
def convert():
    try:
        inr = float(request.args['inr'])
        return jsonify(INR=inr, USD=round(inr * 0.012, 4))
    except:
        return jsonify(error="Invalid INR amount"), 400

if __name__ == '__main__':
    app.run(debug=True)
```

CurrencyConverterClient.java

```java
import java.net.URI;
import java.net.http.*;
import java.util.Scanner;

public class CurrencyConverterClient {
    public static void main(String[] args) throws Exception {
        System.out.print("Enter INR: ");
        String inr = new Scanner(System.in).nextLine();
        String url = "http://localhost:5000/convert?inr=" + inr;
        var response = HttpClient.newHttpClient()
            .send(HttpRequest.newBuilder().uri(URI.create(url)).build(),
                HttpResponse.BodyHandlers.ofString());
        System.out.println(response.body());
    }
}
```

2)check whether a number is prime or not

Here's a more concise version of the **Prime Number Check** implementation, both for the **Python Flask API** and the **Java Client**.

✅ **Python Flask API for Prime Check (prime_checker.py)**

```
from flask import Flask, request, jsonify

app = Flask(__name__)

def is_prime(n):
    if n < 2: return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0: return False
    return True

@app.route('/check_prime', methods=['GET'])
def check_prime():
    try:
        number = int(request.args.get('number'))
        return jsonify({"number": number, "is_prime": is_prime(number)})
    except (ValueError, TypeError):
        return jsonify({"error": "Invalid or missing number"}), 400

if __name__ == '__main__':
    app.run(debug=True)
```

✅ **Java Client to Check Prime Number (PrimeCheckerClient.java)**

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
```

```java
import java.net.URL;
import java.util.Scanner;

public class PrimeCheckerClient {
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
            System.out.print("Enter number to check if it's prime: ");
            String number = scanner.nextLine();
            URL url = new URL("http://127.0.0.1:5000/check_prime?number=" +
number);

            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
            conn.setRequestMethod("GET");
            if (conn.getResponseCode() != 200) throw new
RuntimeException("Failed");

            try (BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()))) {
                br.lines().forEach(System.out::println);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

🔧 **How to Run:**

1. **Run the Python Flask API (prime_checker.py):**

Install Flask:

pip install flask

Start the Flask app:

python prime_checker.py

2. **Compile and Run the Java Client (PrimeCheckerClient.java):**

Compile:

javac PrimeCheckerClient.java

Run:

java PrimeCheckerClient

3) Faranite to Celcius converter

Here's a more concise version of the code for the temperature converter functionality:

✅ **TemperatureConverter.py (Flask API)**

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/convert', methods=['GET'])
def convert():
    scale, value = request.args.get('scale'), request.args.get('value')
    if not scale or not value: return jsonify({"error": "Missing params"}), 400
    try:
        value = float(value)
        if scale.lower() == 'celsius':
            return jsonify({"Celsius": value, "Fahrenheit": value * 9/5 + 32})
        if scale.lower() == 'fahrenheit':
            return jsonify({"Fahrenheit": value, "Celsius": (value - 32) * 5/9})
        return jsonify({"error": "Invalid scale"}), 400
    except ValueError:
        return jsonify({"error": "Invalid value"}), 400
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

✅ **TemperatureConverterClient.java (Java Client)**

```
import java.io.*;
import java.net.*;

public class TemperatureConverterClient {
    public static void main(String[] args) {
        try (Scanner sc = new Scanner(System.in)) {
            System.out.print("Scale (Celsius/Fahrenheit): ");
            String scale = sc.nextLine();
            System.out.print("Temperature: ");
            String value = sc.nextLine();
            URL url = new URL("http://127.0.0.1:5000/convert?scale=" + scale +
"&value=" + value);
            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
            conn.setRequestMethod("GET");
            if (conn.getResponseCode() == 200) {
                new BufferedReader(new
InputStreamReader(conn.getInputStream())).lines().forEach(System.out::pri
ntln);
            }
            conn.disconnect();
        } catch (Exception e) { e.printStackTrace(); }
    }
}
```

---

4)SOAP ALL Practical's: [Java Based]

Here's a **combined Java SOAP Web Service** that handles all the requested operations:

✅ **CalculatorService.java (Interface)**

```
import javax.jws.WebService;

@WebService
public interface CalculatorService {
   int add(int a, int b);
   int multiply(int a, int b);
   int largestOfTwo(int a, int b);
   int largestOfThree(int a, int b, int c);
   int square(int a);
   int cube(int a);
   int addThree(int a, int b, int c);
}
```

✅ **CalculatorServiceImpl.java (Implementation)**

```
import javax.jws.WebService;

@WebService(endpointInterface = "CalculatorService")
public class CalculatorServiceImpl implements CalculatorService {

   public int add(int a, int b) {
      return a + b;
   }

   public int multiply(int a, int b) {
      return a * b;
   }

   public int largestOfTwo(int a, int b) {
```

```
    return Math.max(a, b);
  }

  public int largestOfThree(int a, int b, int c) {
    return Math.max(a, Math.max(b, c));
  }

  public int square(int a) {
    return a * a;
  }

  public int cube(int a) {
    return a * a * a;
  }

  public int addThree(int a, int b, int c) {
    return a + b + c;
  }
}
```

✅ **CalculatorPublisher.java (Server Publisher)**

```
import javax.xml.ws.Endpoint;

public class CalculatorPublisher {
  public static void main(String[] args) {
    Endpoint.publish("http://localhost:8080/calculator", new
CalculatorServiceImpl());
    System.out.println("Calculator Web Service is running...");
  }
}
```

✅ CalculatorClient.java (Client)

```
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import java.net.URL;

public class CalculatorClient {
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://localhost:8080/calculator?wsdl");
        QName qname = new QName("http://soap.example.com/",
"CalculatorServiceImplService");

        CalculatorService service = Service.create(url,
qname).getPort(CalculatorService.class);

        System.out.println("Add: " + service.add(10, 5));
        System.out.println("Multiply: " + service.multiply(10, 5));
        System.out.println("Largest of Two: " + service.largestOfTwo(10, 5));
        System.out.println("Largest of Three: " + service.largestOfThree(10, 20,
15));
        System.out.println("Square: " + service.square(6));
        System.out.println("Cube: " + service.cube(3));
        System.out.println("Add Three Numbers: " + service.addThree(1, 2, 3));
    }
}
```

✅ Output Example

Add: 15

Multiply: 50

Largest of Two: 10

Largest of Three: 20

Square: 36

Cube: 27

Add Three Numbers: 6

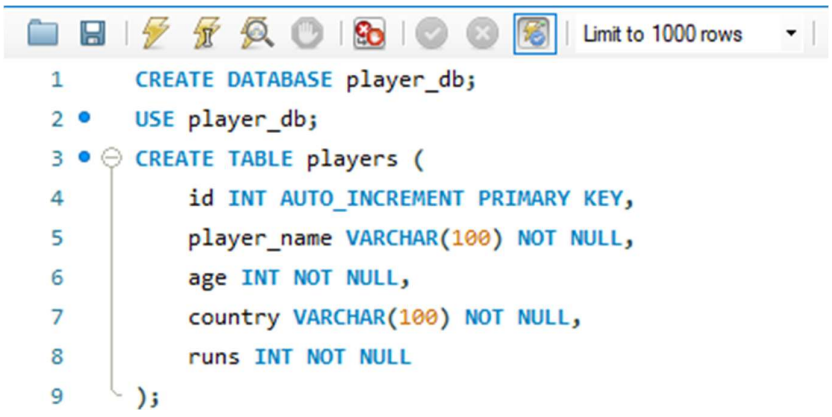
5) Create a simple rest service to demonstrate CRUD operations.

Step 1 : Install THE FOLLOWING module via pip in THE command prompt :

Command: pip install Flask Flask-SQLAlchemy mysqlclient

Step 2 : Open mysql workbench And create a New database 'player_db' And a table 'PLAYERS' as follows:

Command:



Step 3 : Write the following codes

app.py

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

# Configure MySQL database connection (no password required if MySQL is set up like that)
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:aniket@localhost/player_db'
```

```
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# Define the Player model
class Player(db.Model):
    __tablename__ = 'players'
    id = db.Column(db.Integer, primary_key=True)
    player_name = db.Column(db.String(100), nullable=False)

    age = db.Column(db.Integer, nullable=False)
    country = db.Column(db.String(100), nullable=False)
    runs = db.Column(db.Integer, nullable=False)

# Create tables when app starts
with app.app_context():
    db.create_all()

# Create a player
@app.route('/player', methods=['POST'])
def create_player():
    data = request.json
    player = Player(**data)
    db.session.add(player)
    db.session.commit()
```

```
    return jsonify({"message": "Player created successfully"}), 201

# Get all players
@app.route('/players', methods=['GET'])
def get_players():
    players = Player.query.all()
    return jsonify([{
        "id": p.id, "player_name": p.player_name, "age": p.age,
        "country": p.country, "runs": p.runs
    } for p in players])

# Update a player
@app.route('/player/<int:id>', methods=['PUT'])
def update_player(id):
    player = Player.query.get(id)
    if not player:
        return jsonify({"message": "Player not found"}), 404

    data = request.json
    for key, value in data.items():
        setattr(player, key, value)

    db.session.commit()
    return jsonify({"message": "Player updated successfully"})
```

```
# Delete a player
@app.route('/player/<int:id>', methods=['DELETE'])
def delete_player(id):
    player = Player.query.get(id)
    if not player:
        return jsonify({"message": "Player not found"}), 404

    db.session.delete(player)
    db.session.commit()
    return jsonify({"message": "Player deleted successfully"})

if __name__ == '__main__':
    app.run(debug=True)
```

test_app.py

```
from app import db, Player, app
def create_player():
    new_player = Player(player_name="MS Dhoni", age=35, country="India",
runs=120)
    db.session.add(new_player)
    db.session.commit()
    print(f"Player created: {new_player.player_name}")

def get_all_players():
    players = Player.query.all()
    print("All players in DB:")
```

```
    for player in players:
        print(f"{player.id}: {player.player_name}, {player.age}, {player.country},
{player.runs}")


def update_player(player_id):
    player = db.session.get(Player, player_id)
    if player:
        player.player_name = "Rohit Sharma"
        db.session.commit()
        print(f"Updated player {player.id} ")
    else:
        print("Player not found!")


def delete_player(player_id):
    player = db.session.get(Player, player_id)
    if player:
        db.session.delete(player)
        db.session.commit()
        print(f"Deleted player {player.player_name}")
    else:
        print("Player not found!")


if __name__ == "__main__":
    with app.app_context():
```

create_player()

get_all_players()

update_player(1)

get_all_players()

Step 4 : First run the app.py file

```
PS H:\Aniket\6th Semester\6th SEM\Cloud Computing\practicals\REST_CRUD> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 454-092-406
```

Step 5 : now run the test_app.py file

```
cals\REST_CRUD> python .\test_app.py
Player created: MS PS H:\Aniket\6th Semester\6th SEM\Cloud Computing\practi: MS: M::::: MS:::::::
: MS Dhoni
All players in DB:
1: MS Dhoni, 35, India, 120
Updated player 1
All players in DB:
1: Rohit Sharma, 35, India, 120
PS H:\Aniket\6th Semester\6th SEM\Cloud Computing\practicals\REST_CRUD>
```

YOU will see that a new Player will Be created And you can Also update , delete or get all the players details

In the Mysql workbench run the select command to check if the changes are getting reflected in the database

command: select * from players;

	id	player_name	age	country	runs
▶	1	Rohit Sharma	35	India	120
*	NULL	NULL	NULL	NULL	NULL

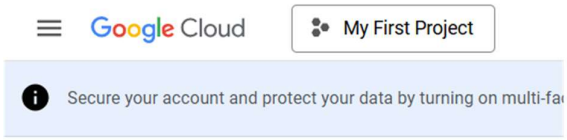
you can also see the details in the browser

6) Develop application to consume Google's search / Google's Map RESTful Web service.

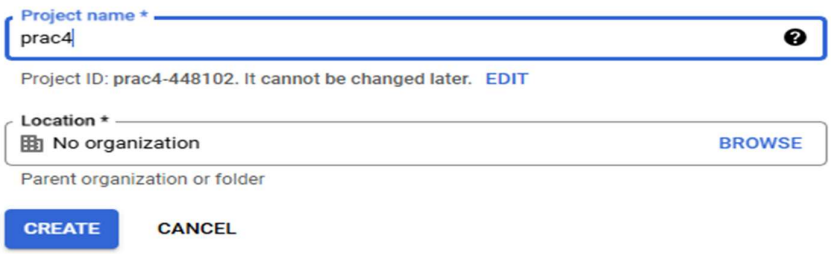
SOLUTION:

Step 1: Search "console cloud google" on chrome and sign up: https://console.cloud.google.com

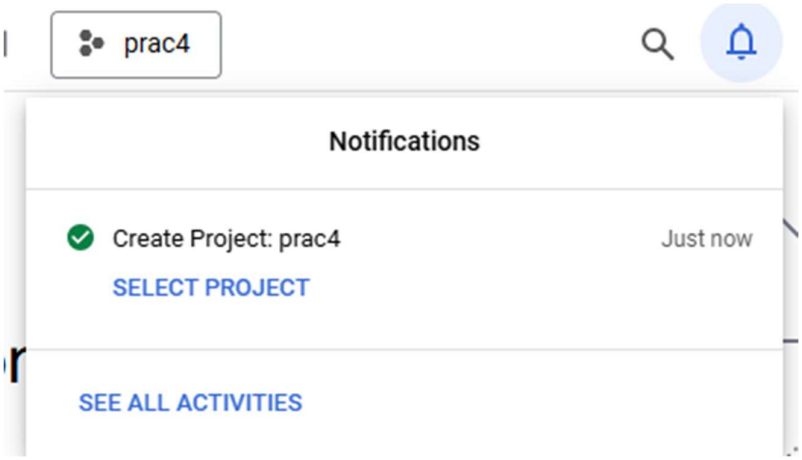
Step 2 : Click on my first project and create new project



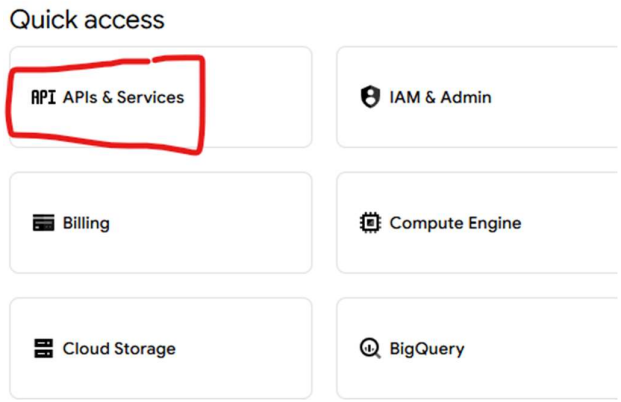
Step 3 : Give project name and then click create.



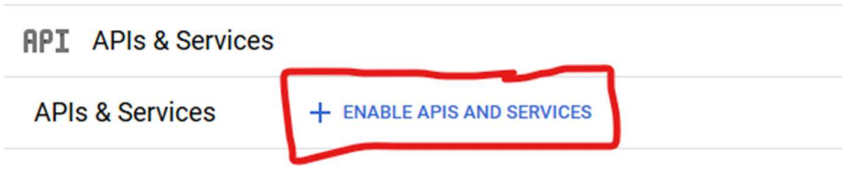
Step 4: You can see your project by clicking on bell icon.



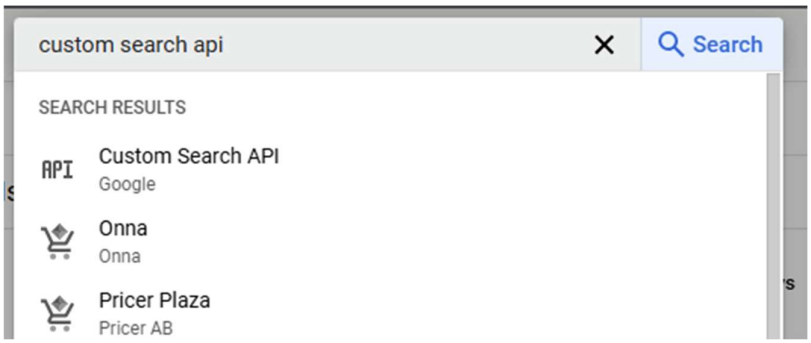
Step 5 : Click on bell icon and click “select project” Once you select the project then click on “APIs and Services” → Enabled APIs and services



Right click on APIs and Services and Enable them.

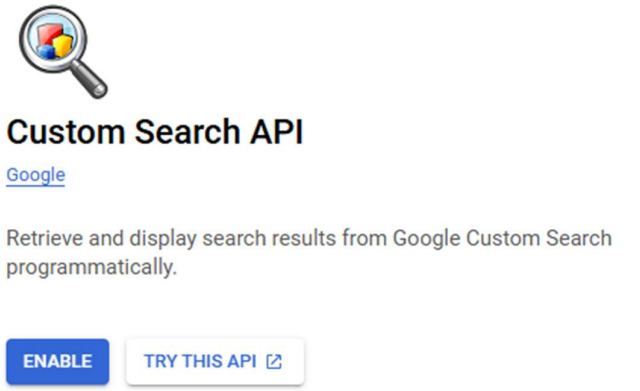


Step 6: Search “Custom search api” on search bar and click on first link.



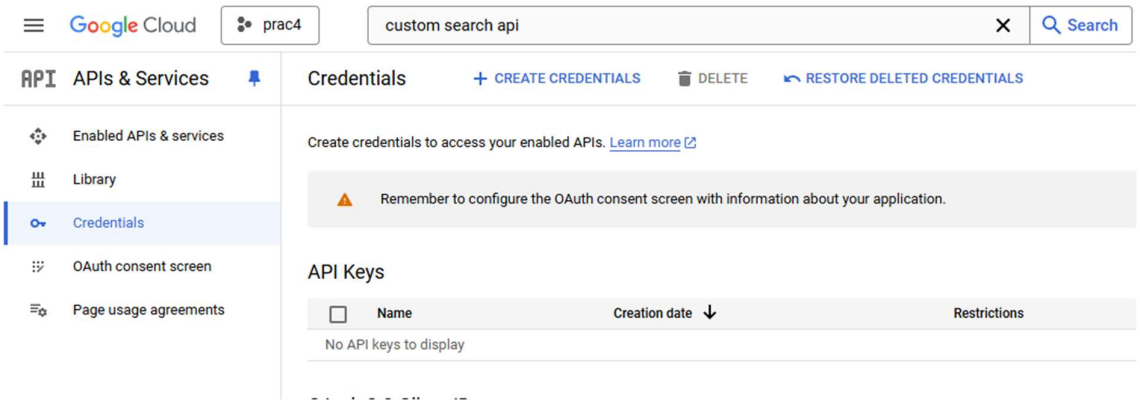
Step 7: In google cloud project you always have to specify which functionality you want to enable.

Click on enable to 'enable' custom search api. This enable the custom search api for this particular project.

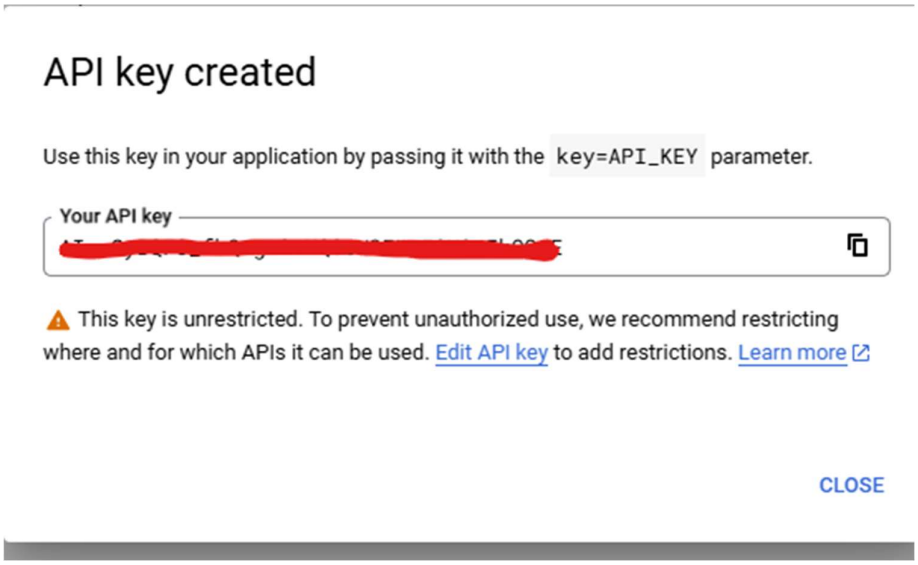


Step 8: Now you need to generate an api key to authenticate yourself from python.

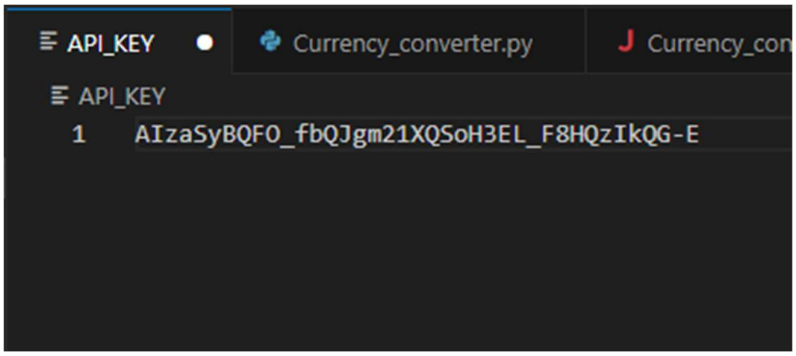
In the Api & Services section click on credentials→ click create credentials → Click API key.



Step 9: API key will be created.



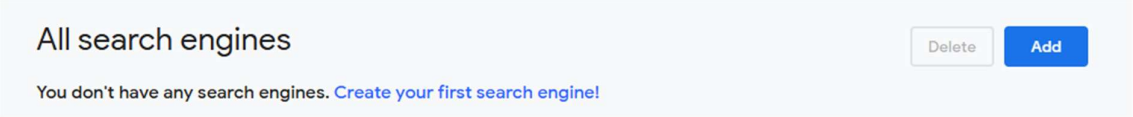
Step 10 : Now create a file named API_KEY inside your project folder and paste your API key value in it.



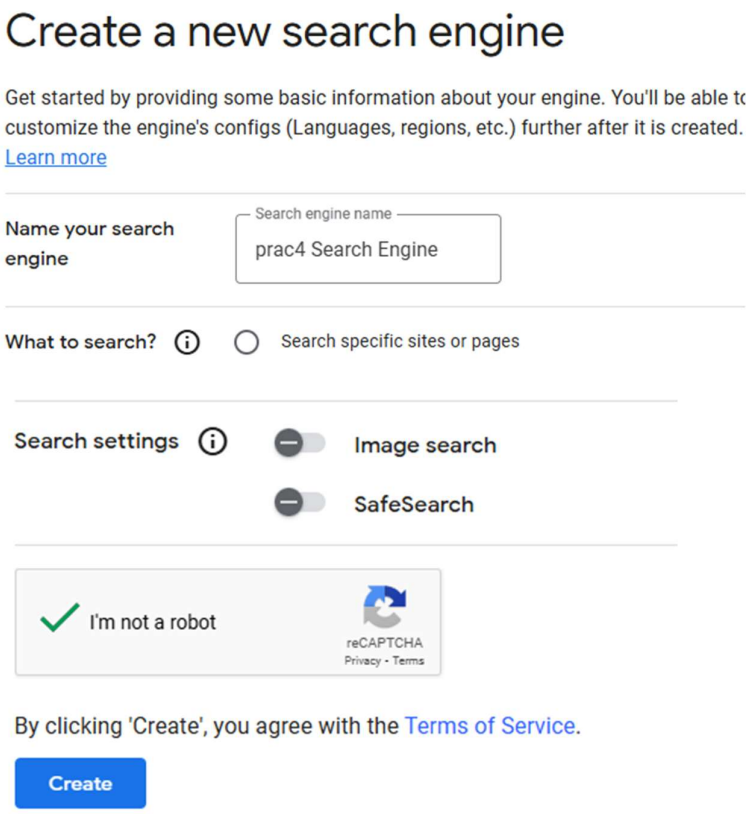
Step 11: We have to Create a search engine. Navigate to the following website:

https://programmablesearchengine.google.com/about/

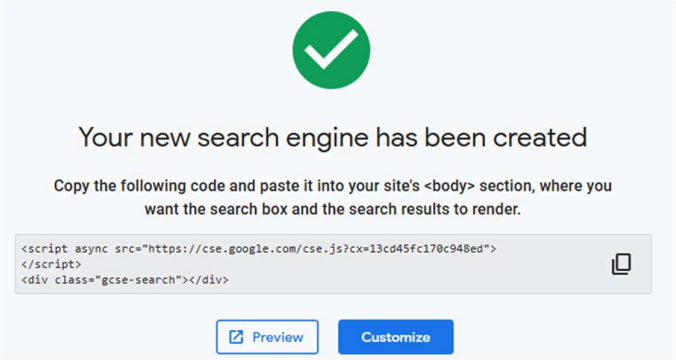
Step 12 : Click on Get started and then click on ‘create your first search engine’.



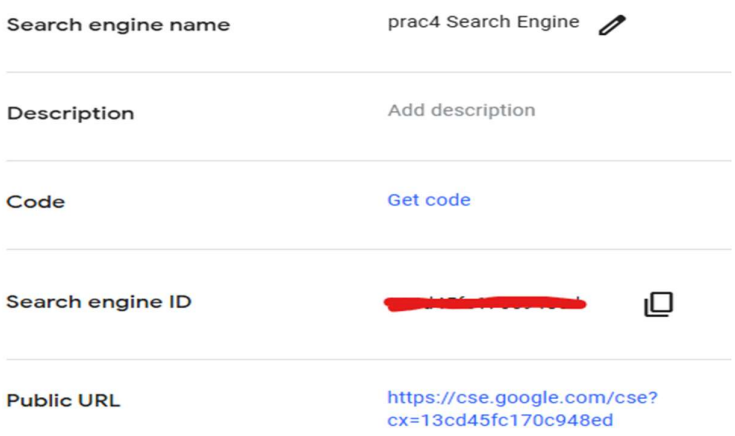
Step 13: Give name to your search engine and select the "search the entire web “and click create



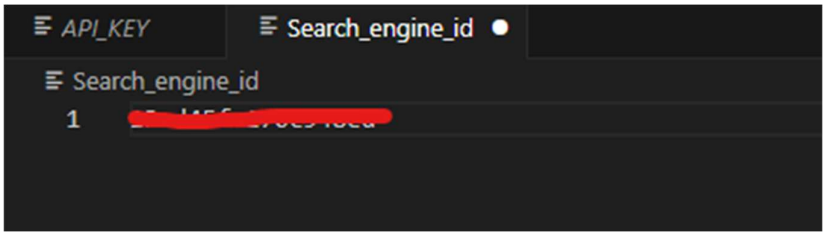
Step 14: A new search engine will be created.



Step 15: Click "go back to all search engine" → click on your search engine → Copy the search engine ID.



Step 16: Create a new file inside your project folder as SEARCH_ENGINE_ID and paste your search engine ID.



Step 17: Open new terminal and Download requests using the command, pip3 install requests.

```
PS H:\Aniket\6th Semester\6th SEM\Cloud Computing\practicals\prac4> pip install requests
Collecting requests
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting charset-normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.1-cp313-cp313-win_amd64.whl.metadata (36 kB)
Collecting idna<4,>=2.5 (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.3.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
```

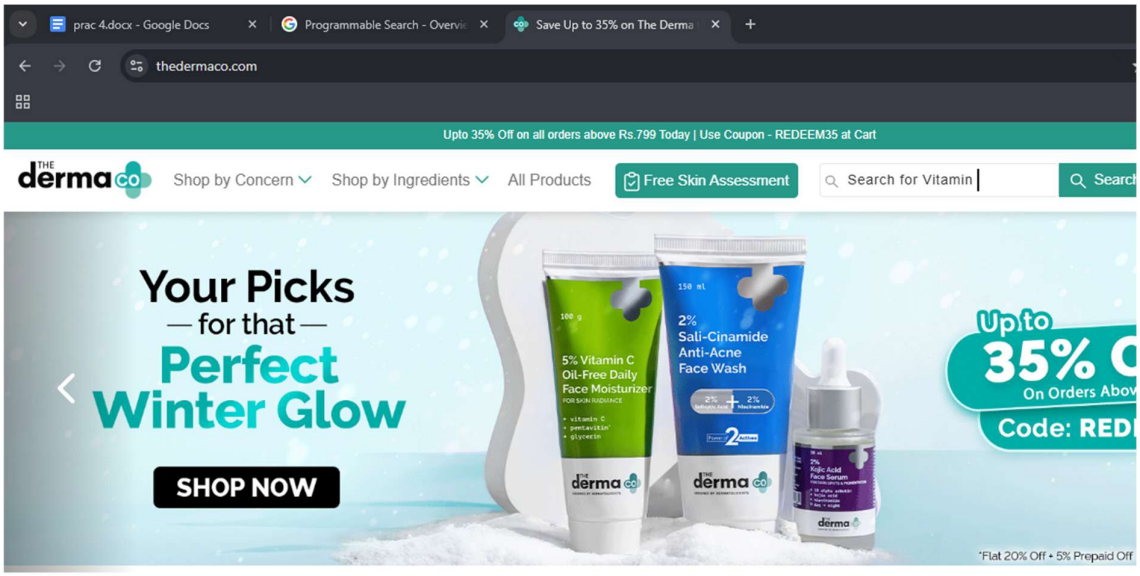
Step 18: Create a file main.py and write a below code in it

CODE:

```python
import requests

# Replace with your actual API key and Search Engine ID
API_KEY = 'AIzaSyBQFO_fbQJgm21XQSbH3EL_F8HQzlkQG-E'
SEARCH_ENGINE_ID = '13cd45fc170c948ed'

search_query = 'the derma co'
url = 'https://www.googleapis.com/customsearch/v1'
params = {
    'q': search_query,
    'key': API_KEY,
    'cx': SEARCH_ENGINE_ID
}

response = requests.get(url, params=params)
results = response.json()

if 'items' in results:
    print(results['items'][0]['link'])
else:
    print("No results found.")
```

Output:

```
PS H:\Aniket\6th Semester\6th SEM\Cloud Computing\practicals\prac4> python main.py
https://thedermaco.com/
```



7) Installation and Configuration of virtualization using KVM.

Step 1:

Sudo apt-get update

Step 2: kvm

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

```
prathamesh@prathamesh-VirtualBox:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
10
```

Step 3 :

kvm-ok

```
prathamesh@prathamesh-VirtualBox:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Step 4:

Sudo apt install -y cpu-checker

```
prathamesh@prathamesh-VirtualBox:~$ sudo apt install -y cpu-checker
[sudo] password for prathamesh:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cpu-checker is already the newest version (0.7-1.3build2).
0 upgraded, 0 newly installed, 0 to remove and 126 not upgraded.
```

Step 5:

sudo apt-get install -y qemu-kvm virt-manager libvirt-daemon-system virtinstlibvirt-clients bridge-utils

```
prathamesh@prathamesh-VirtualBox:~$ sudo apt install -y qemu-kvm virt-manager libvirt-daemon-system virtinst libvirt-cli
ents bridge-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
qemu-system-x86 is already the newest version (1:8.2.2+ds-0ubuntu1.4).
virt-manager is already the newest version (1:4.1.0-3ubuntu0.1).
libvirt-daemon-system is already the newest version (10.0.0-2ubuntu8.5).
virtinst is already the newest version (1:4.1.0-3ubuntu0.1).
libvirt-clients is already the newest version (10.0.0-2ubuntu8.5).
bridge-utils is already the newest version (1.7.1-1ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 126 not upgraded.
```

Step 6:

sudosystemctl enable –now libvirtd

```
prathamesh@prathamesh-VirtualBox:~$ sudo systemctl enable --now libvirtd
```

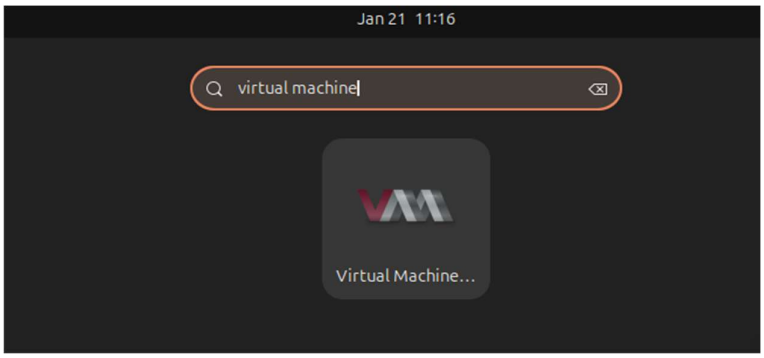
sudosystemctl start libvirtd

```
prathamesh@prathamesh-VirtualBox:~$ sudo systemctl start libvirtd
```

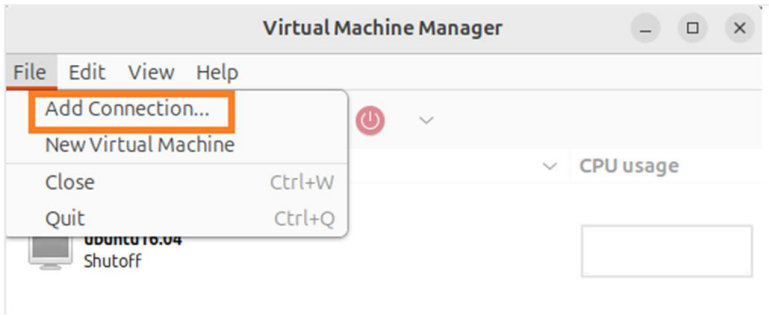
sudosystemctl status linvirtd

```
prathamesh@prathamesh-VirtualBox:~$ sudo systemctl status libvirtd
● libvirtd.service - libvirt legacy monolithic daemon
     Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-01-21 11:08:58 IST; 1min 32s ago
TriggeredBy: ● libvirtd.socket
             ● libvirtd-admin.socket
             ● libvirtd-ro.socket
       Docs: man:libvirtd(8)
             https://libvirt.org/
   Main PID: 5691 (libvirtd)
      Tasks: 22 (limit: 32768)
     Memory: 24.7M (peak: 26.7M)
        CPU: 215ms
```

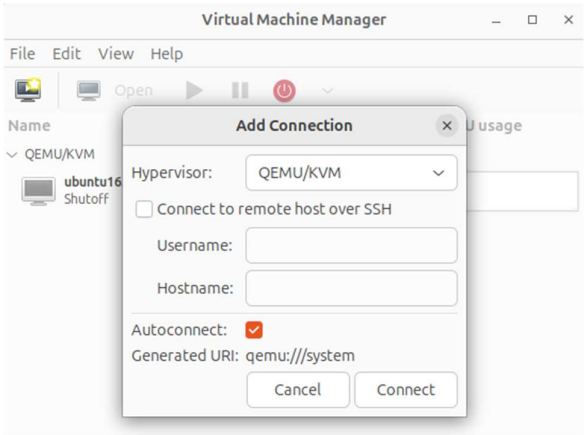
Step 7: In this step search virtual machine and start



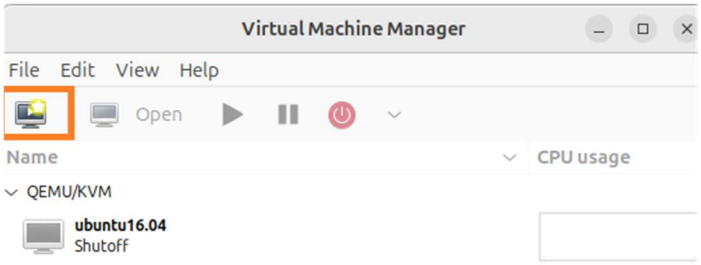
Step 8: Click add connection



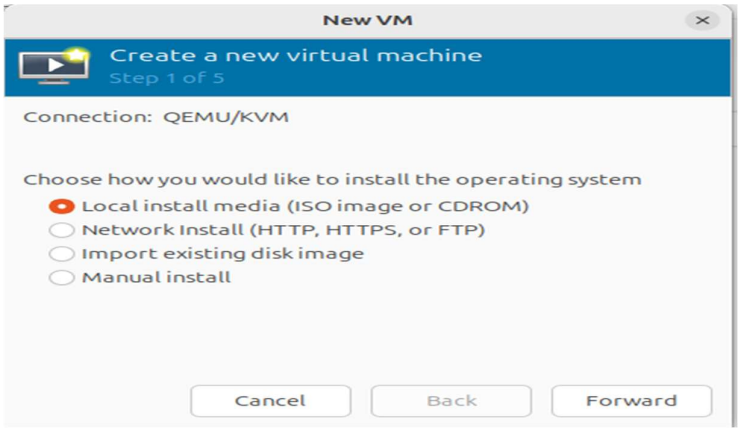
Step 9: Now click connect



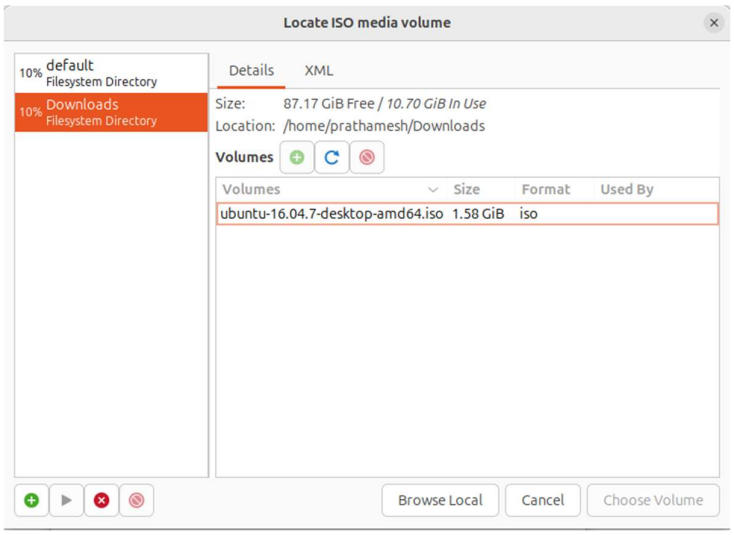
Step 10: Select the computer icon as shown in mage



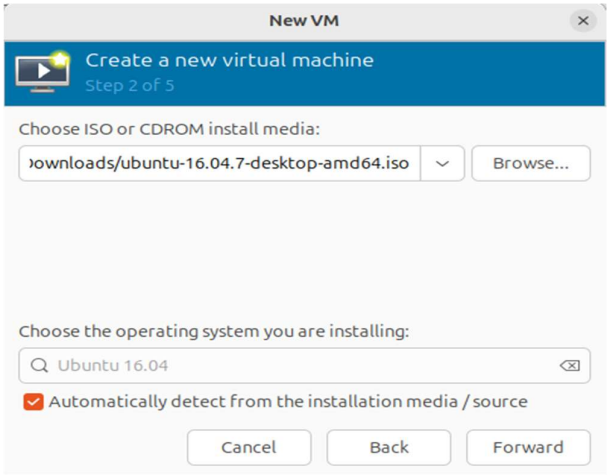
Step 11: *Select 'Local install media' and Click on* ***Forward***



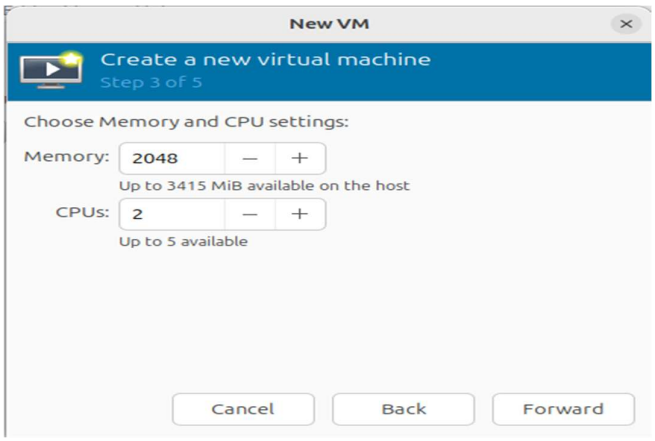
Step 12: *Select the .iso from Downloads and click on 'Choose volume'*



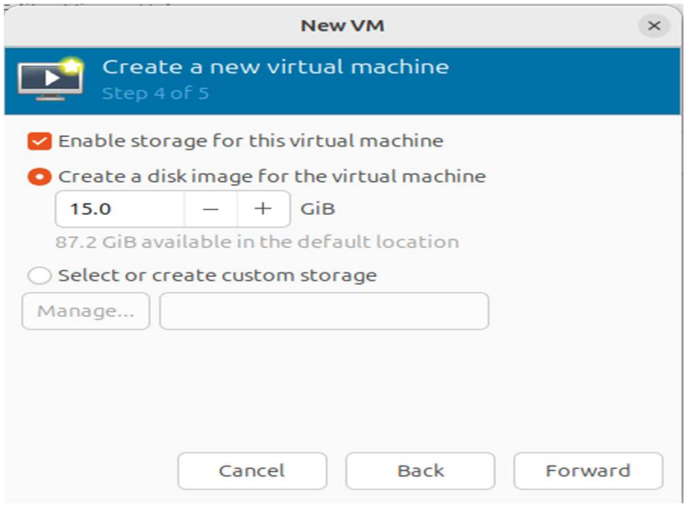
Step 13: *Click on 'Forward'*



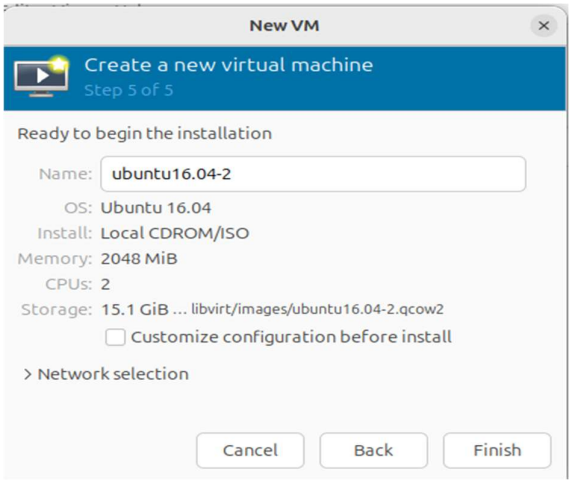
Step 14: *Select Memory= 2048 and CPUs = 2 and Click on 'Forward'*



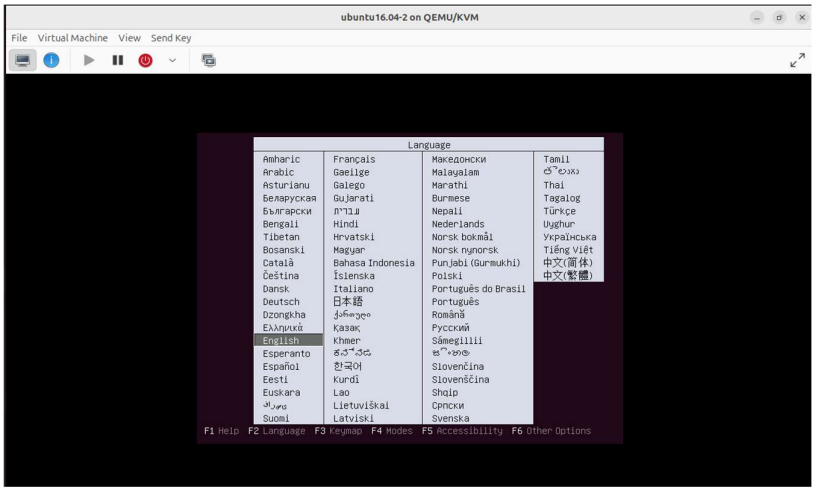
Step 15: *Click on Radio button and config. The disk image to 15.0 and click on Forward*



Step 16: *Click on 'Finish'*



The installation has been successfully completed. Select the Language i.e English



Select second option from arrow key 'Install Ubuntu'

All set and ready to go!!!!!

8) Develop application to download image/video from server or upload image/video to server using MTOM techniques.

SOLUTION:

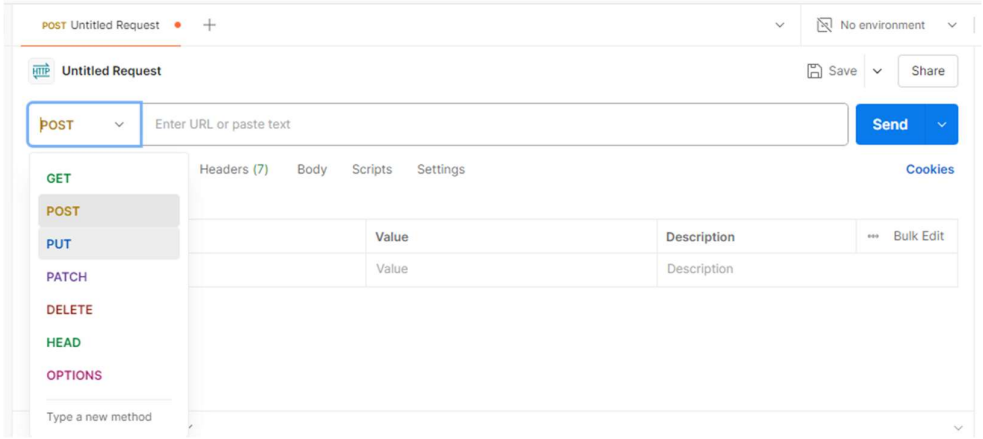
Step 1: Inside the project folder create a new file named app.py and write the following code

```
from flask import Flask, request, send_from_directory
import os

app = Flask(__name__)
FOLDER = 'mtom'
os.makedirs(FOLDER, exist_ok=True)

@app.route('/upload', methods=['POST'])
def upload():
    f = request.files.get('file')
    if f:
        f.save(os.path.join(FOLDER, f.filename))
        return {"message": f"{f.filename} uploaded"}
    return {"error": "No file uploaded"}, 400

@app.route('/download/<name>')
def download(name):
    try:
        return send_from_directory(FOLDER, name, as_attachment=True)
    except:
        return {"error": "File not found"}, 404

if __name__ == '__main__':
    app.run(debug=True)
```

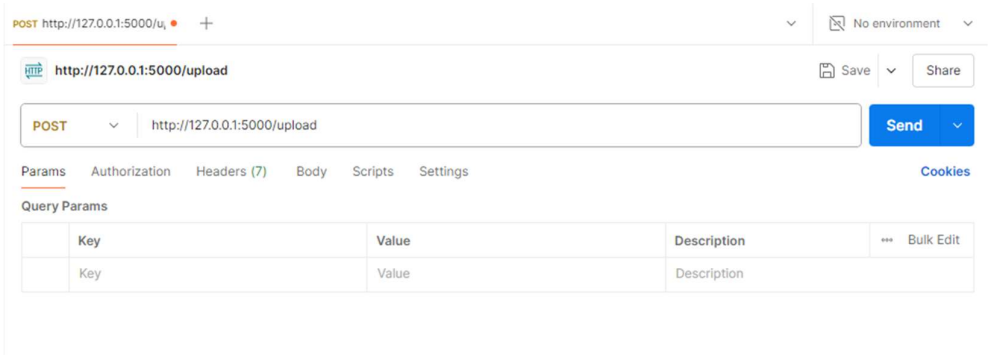
Step 2 : Run the app.py code.

```
n.exe "e:/Aniket/TYBSC.CS/6th Semester/6th SEM/Cloud Computing/prac6/app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 454-092-406
```

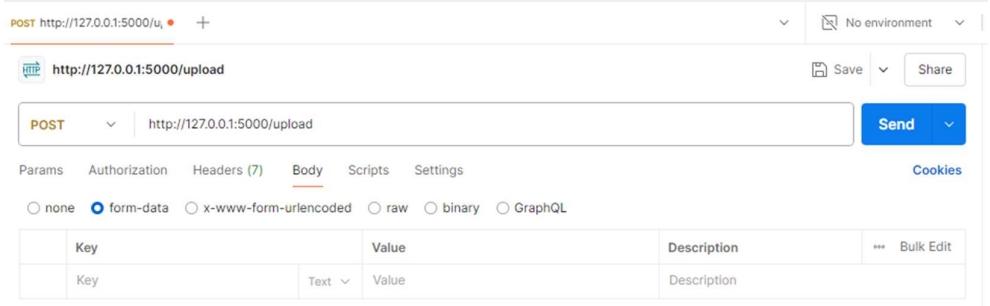
Step 3: Open Postman API and and select POST request



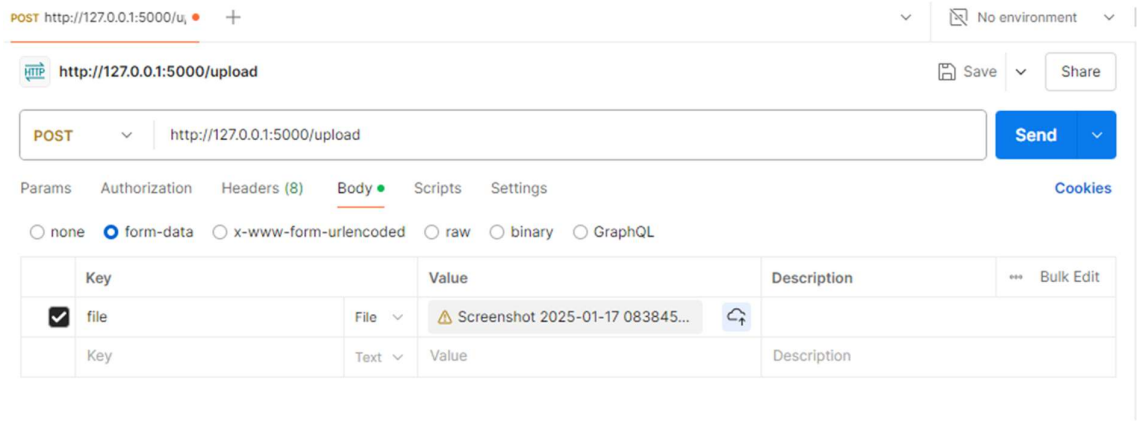
Step 4: Enter the following url :http://127.0.0.1:5000/upload.



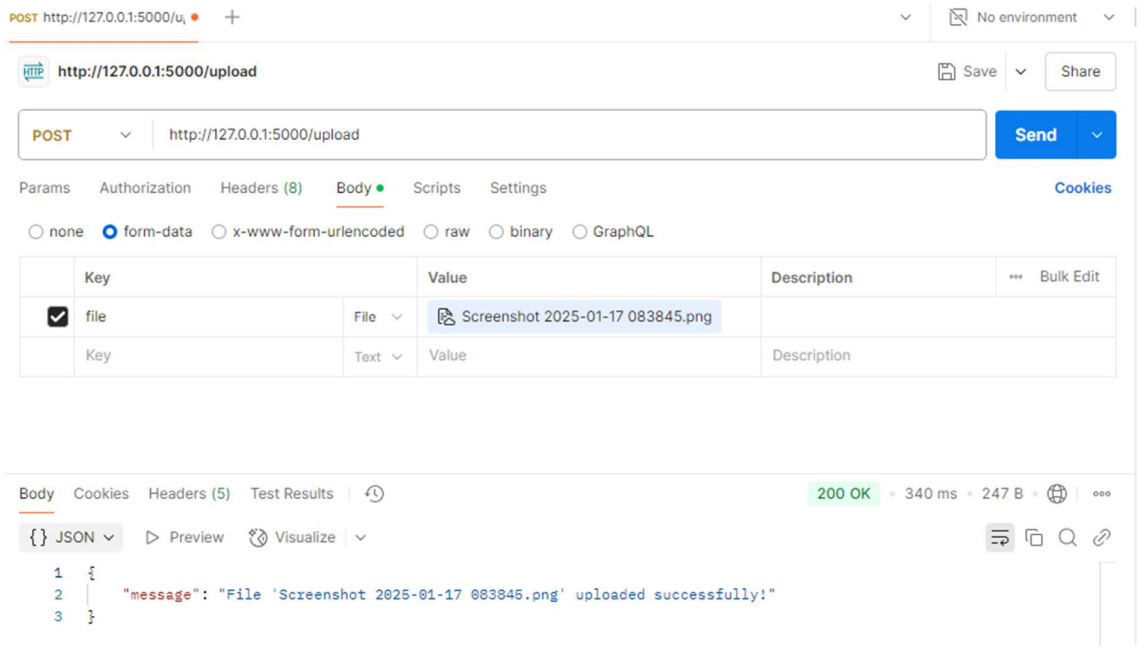
Step 5: Click on Body and select form-data.



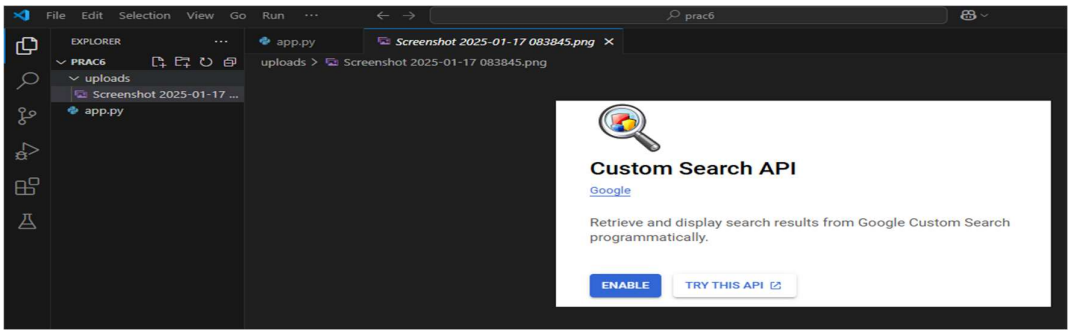
Step 6: Select file and name it as ‘file’.



Step 7: Click on Send and You will get to see the following output.



Step 8: Now Go to Vscode and inside the upload folder you will see the image you uploaded in Postman.



```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 454-092-406
127.0.0.1 - - [18/Jan/2025 09:21:55] "POST /upload HTTP/1.1" 200 -
```

9)Implementation of Openstack with user and private network creation.

Steps to perform:

1. *sudo apt update*
2. *Sudo apt upgrade*

Step 1: *sudo snap install microstack --beta*

```
prathamesh@prathamesh-VirtualBox:~$ sudo snap install microstack --beta
[sudo] password for prathamesh:
snap "microstack" is already installed, see 'snap help refresh'
```

Step 2: *snap list microstack*

```
prathamesh@prathamesh-VirtualBox:~$ snap list microstack
Name        Version  Rev  Tracking     Publisher   Notes
microstack  ussuri   245  latest/beta  canonical✓  -
```

Step 3: *sudo microstack init --auto --control*

```
prathamesh@prathamesh-VirtualBox:~$ sudo microstack init --auto --control
2025-02-15 10:39:10,460 - microstack_init - INFO - Configuring clustering ...
2025-02-15 10:39:10,565 - microstack_init - INFO - Setting up as a control node.
2025-02-15 10:39:12,787 - microstack_init - INFO - Generating TLS Certificate an
d Key
```

Step 4: *microstack.openstack --version*

```
prathamesh@prathamesh-VirtualBox:~$ microstack.openstack --version
openstack 5.2.0
```

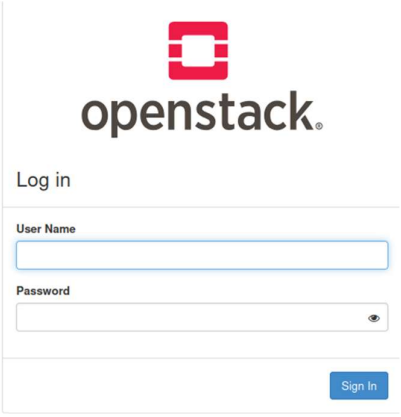
Step 5: *sudo snap get microstack config.credentials.keystone-password*

```
thamesh@prathamesh-VirtualBox:~$ sudo snap get microstack config.credentials.keystone-password
p0S6lRoN3LQh[redacted]
```

Step 6: ip a

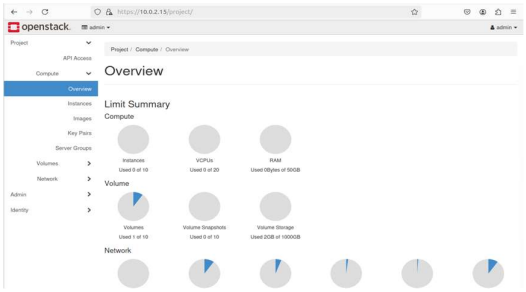
```
prathamesh@prathamesh-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3e:2e:92 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 84342sec preferred_lft 84342sec
    inet6 fe80::a00:27ff:fe3e:2e92/64 scope link
       valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:d7:ac:12 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
```

Now paste inet : 10.0.2.15 in browser it will open the following login page of Openstack



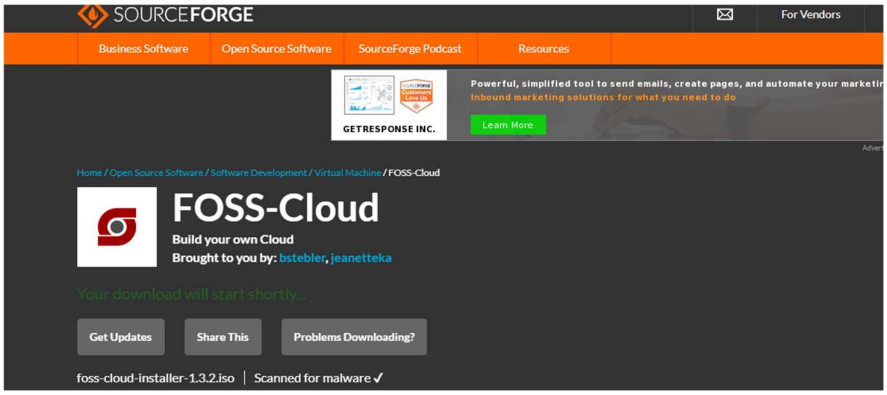
Fill the username **as *'admin'*** *and password which is provided in* ***step 5***

Congratulation you logged in OpenStack



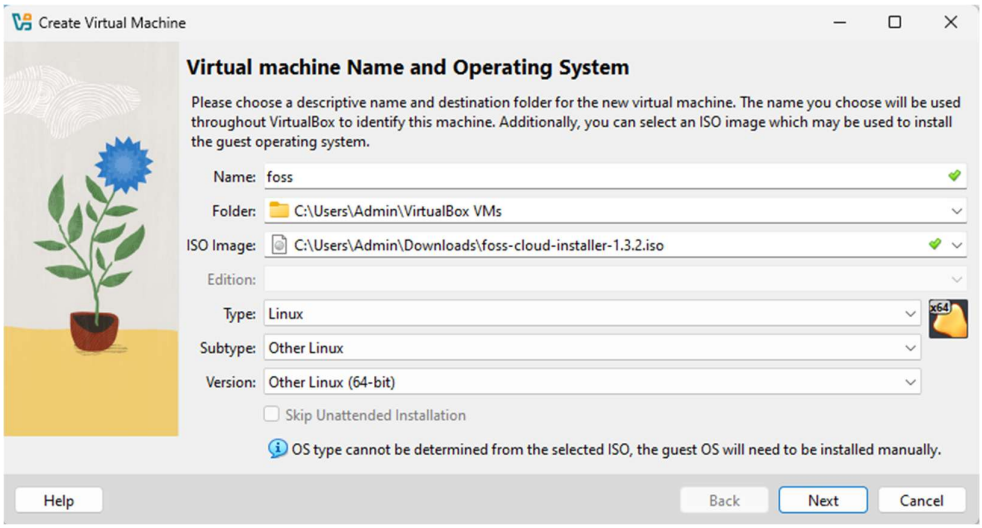
10) Aim: Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Storage & Platform as a Service (PaaS).

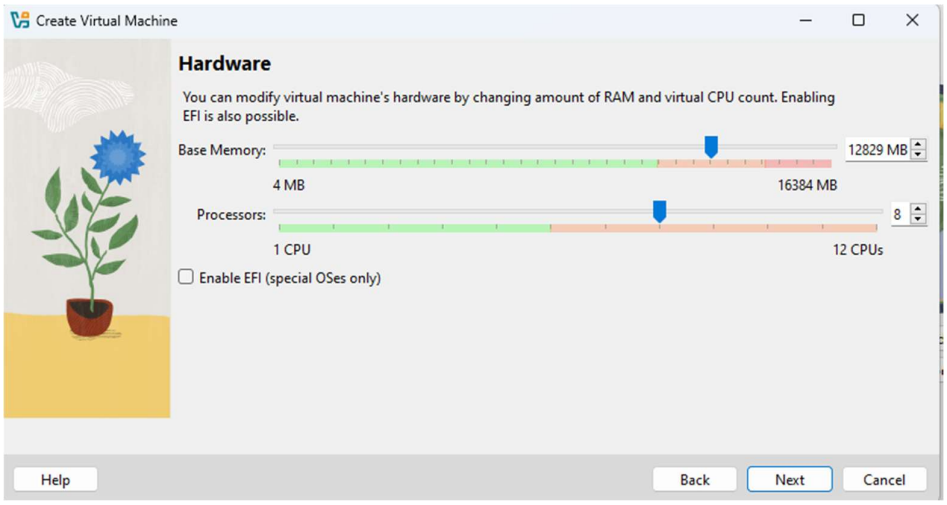
Step 1: Download and install FOSS-Cloud from its official website.



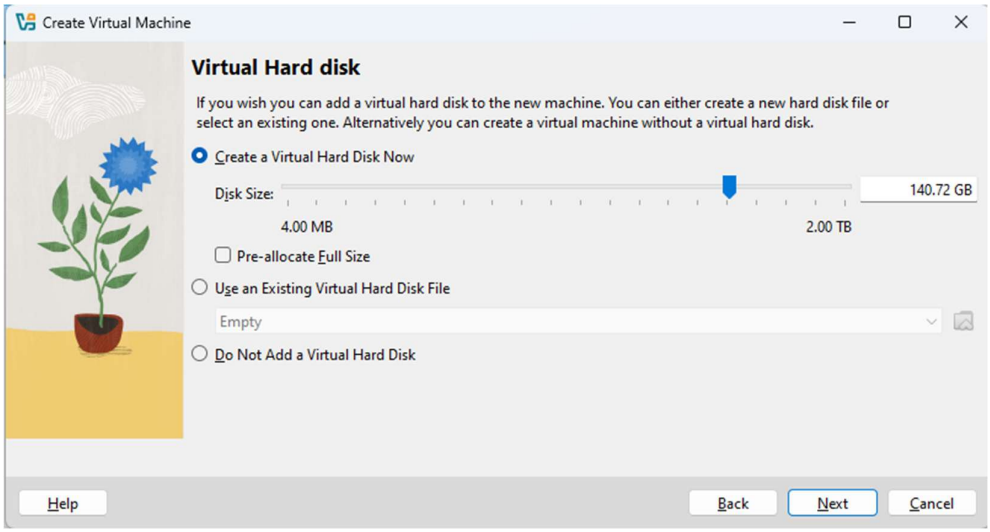
Step 2: Open Virtual Machine and create new Virtual Machine. Add your foss cloud iso file path at ISO Image.

Step 3: At the time of setup allocate Base memory.

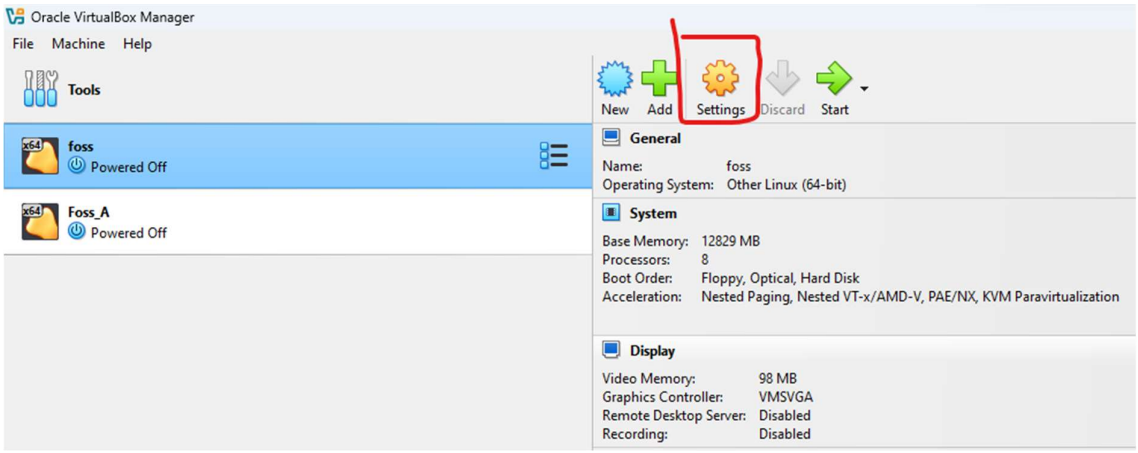




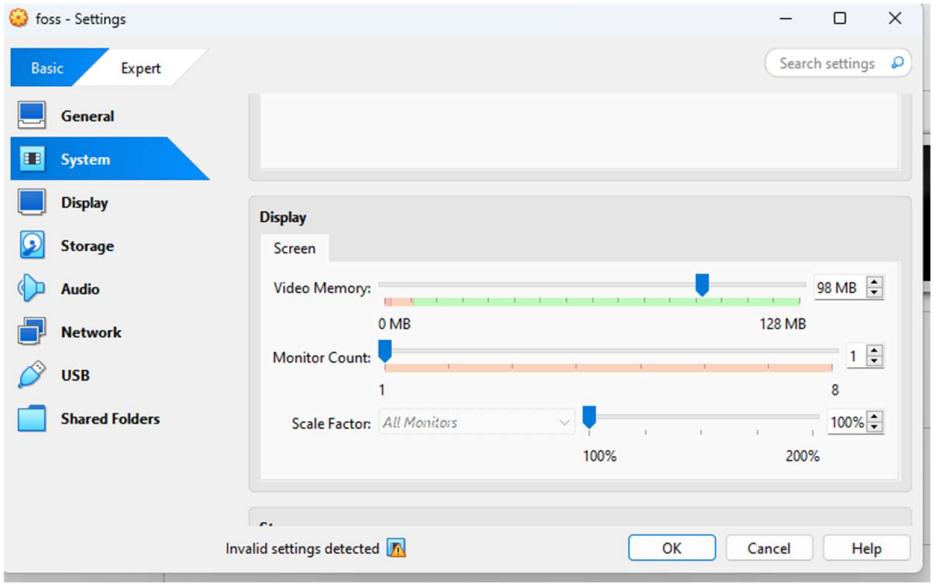
Step 4: Create Virtual Hard Disk and give disk size.



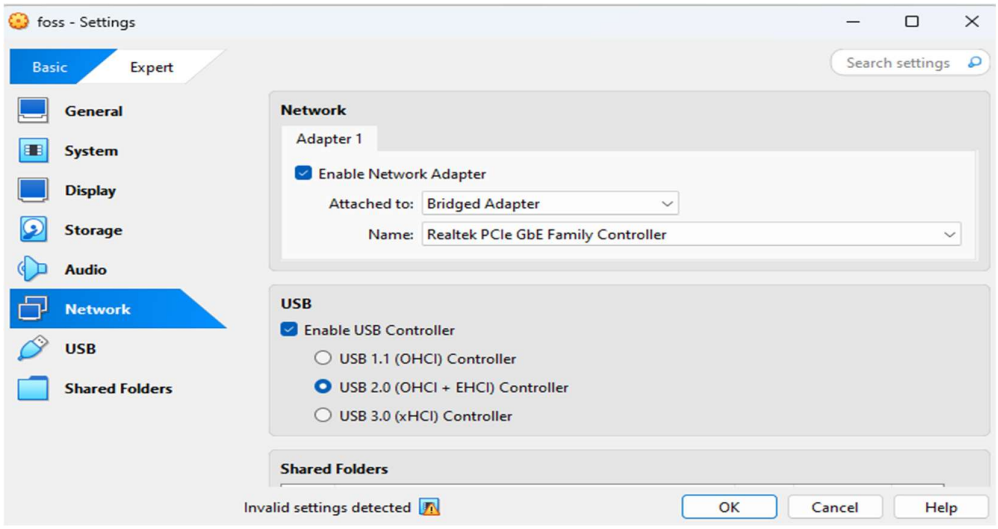
Step 5: Click on settings.



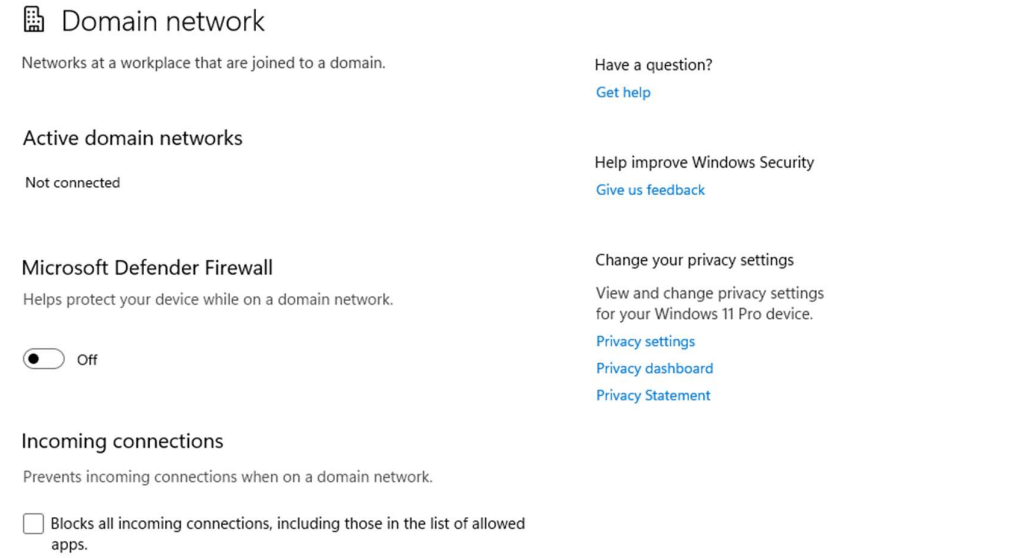
Step 6: Select System → Display & add video memory.

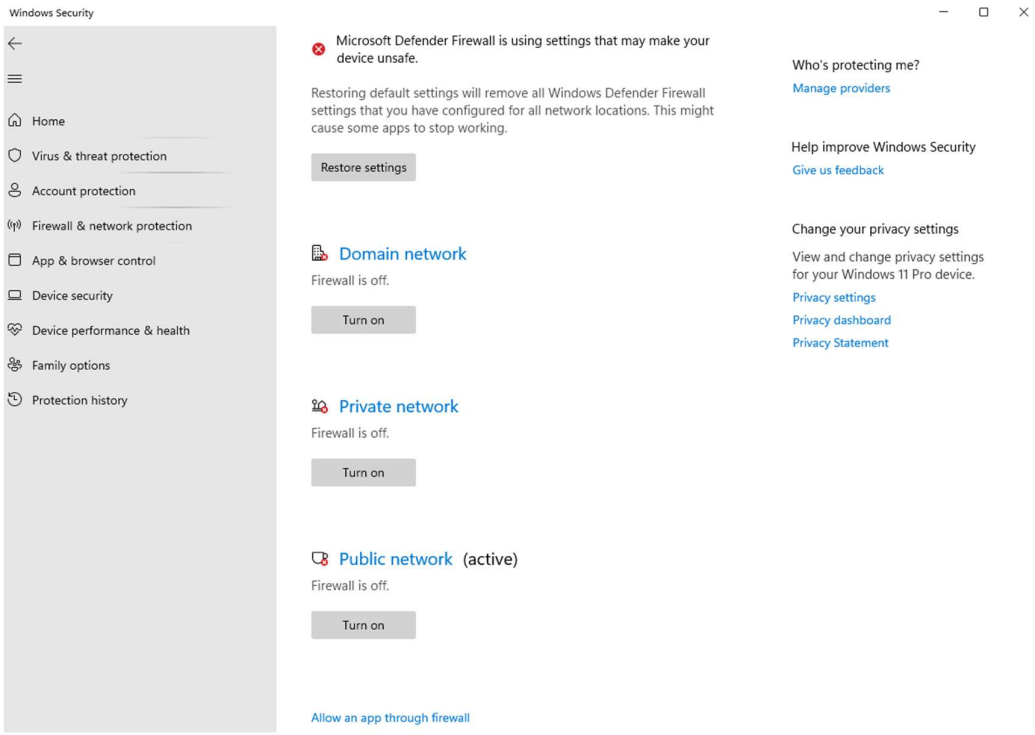


Step 7: Select Network & add Bridge Adapter. Then USB & SELECT USB 2.0 (OHCI + EHCI) Controller.

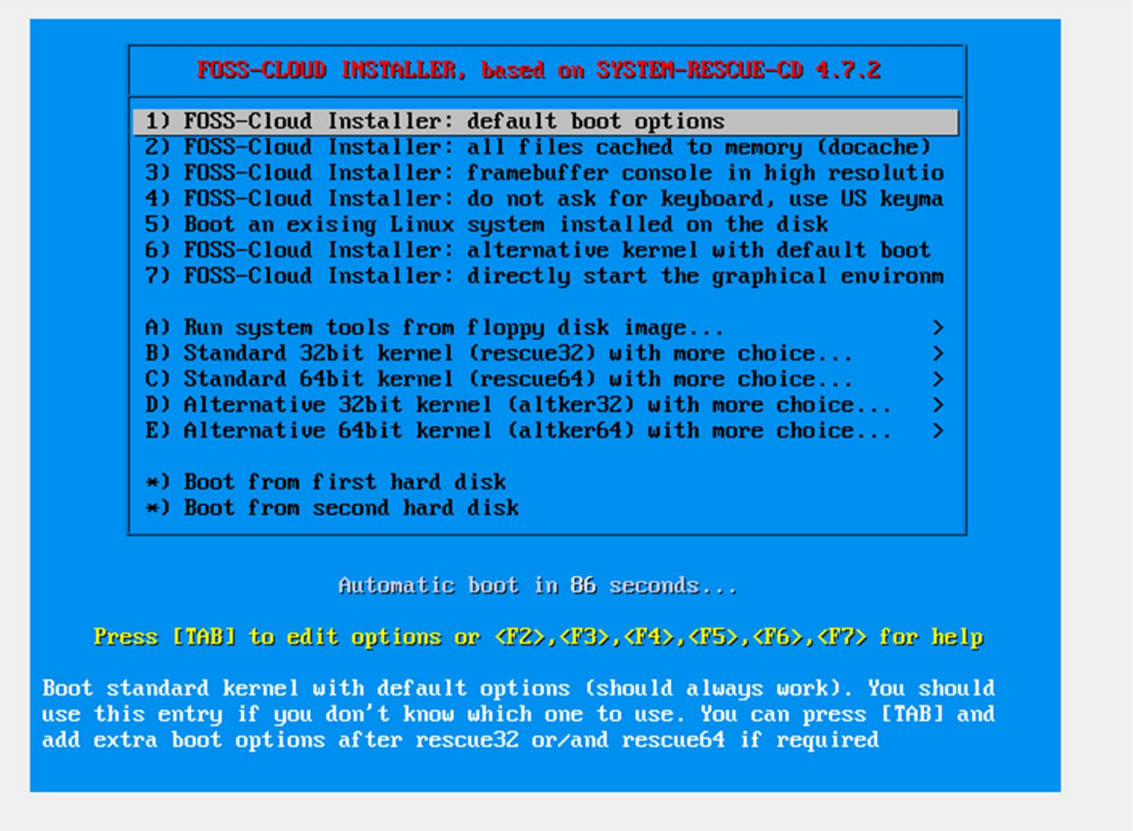


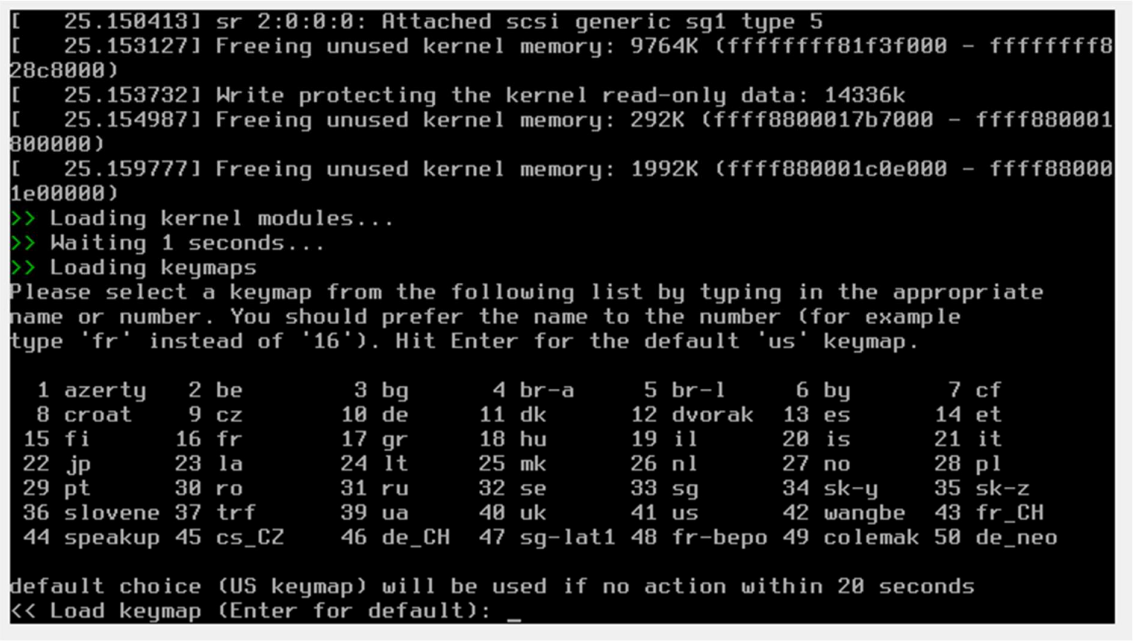
Step 8: Block/Off the firewall from settings.



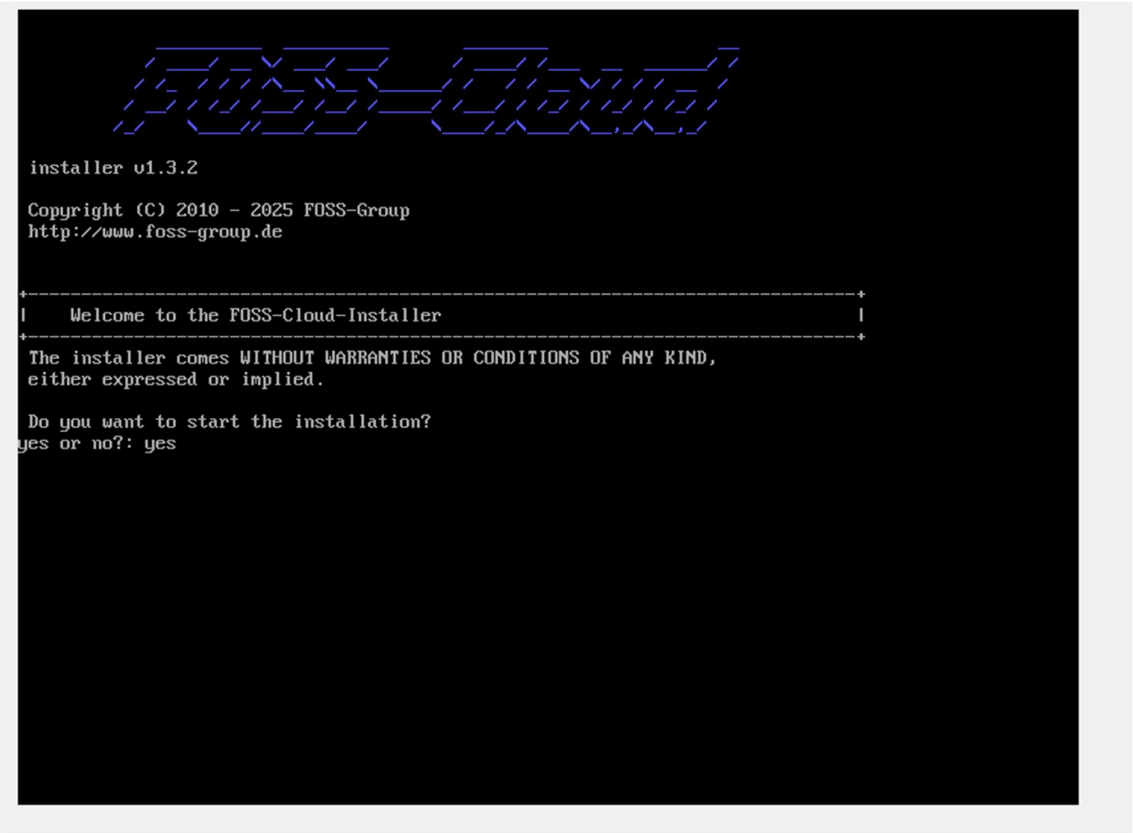


Step 9: Select 1st option here and Enter.





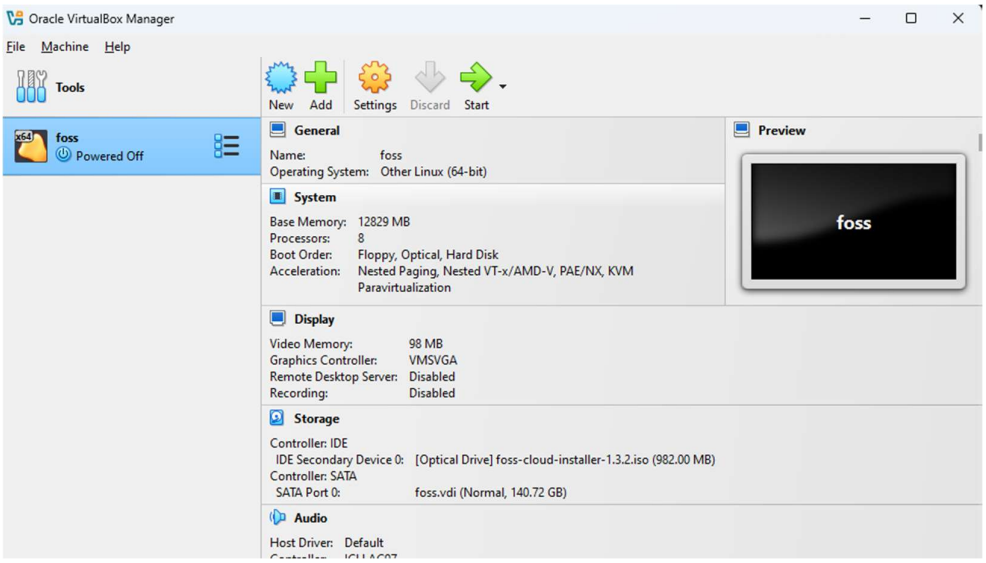
Step 10: Type yes to start the installation.



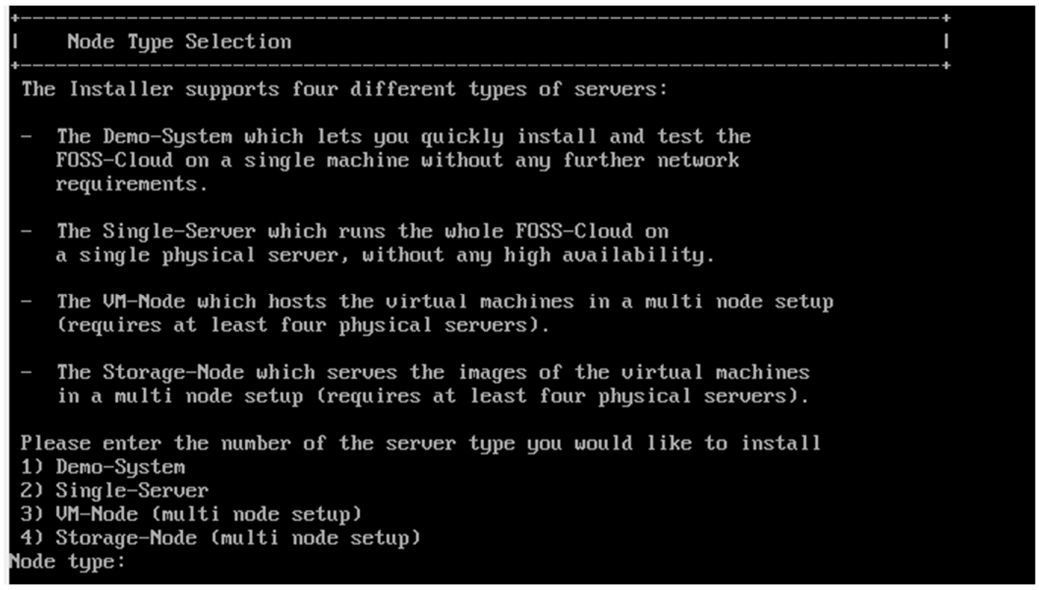
Step 11: Open cmd and type following command and enter.

```
C:\Windows\System32>cd C:\Program Files\Oracle\VirtualBox

C:\Program Files\Oracle\VirtualBox>VBoxManage modifyvm "foss" --nested-hw-virt on
```

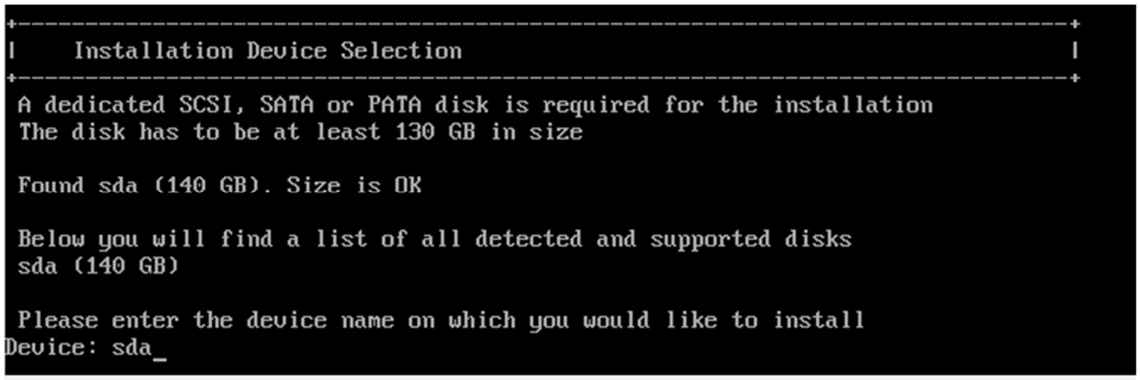
Step 12: Now start with the FOSS virtual machine.



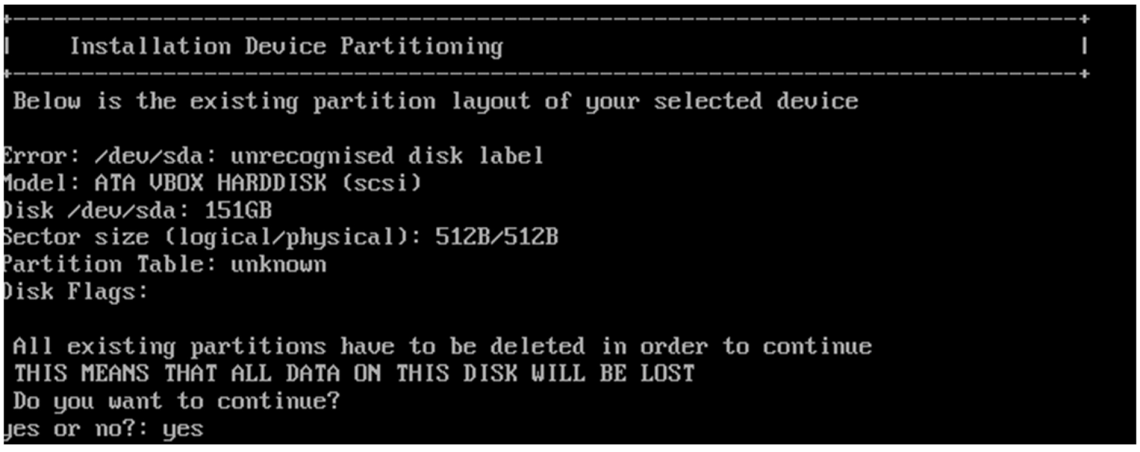
Step 13: Enter Number 1 to install Demo-System server.



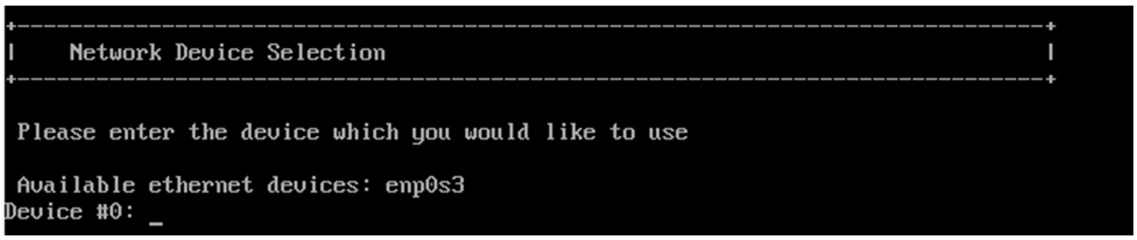
Step 14: Add device name sda.



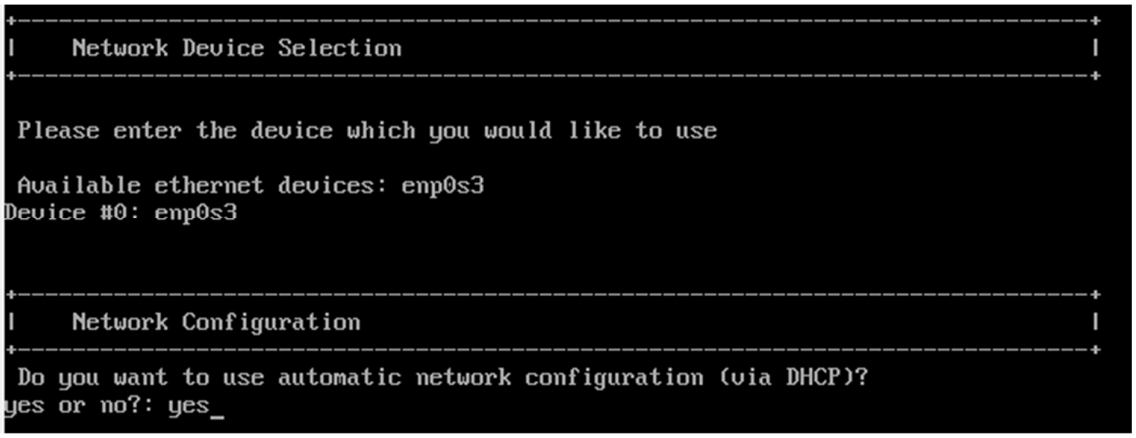
Step 15: Enter yes to continue for installation of Device Partitioning.

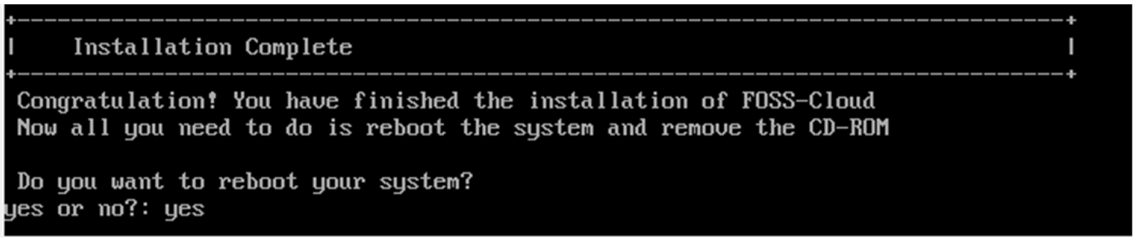


Step 16: Add available Ethernet device name eg: enp0s3

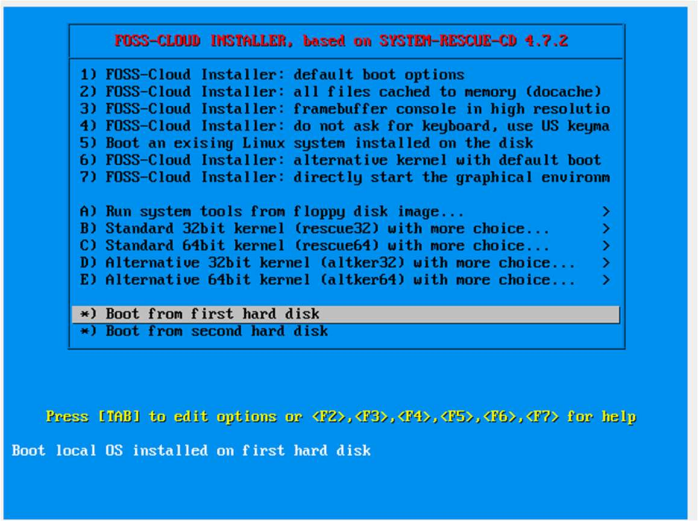


Step 17: Enter yes to use automatic network configuration and enter yes.

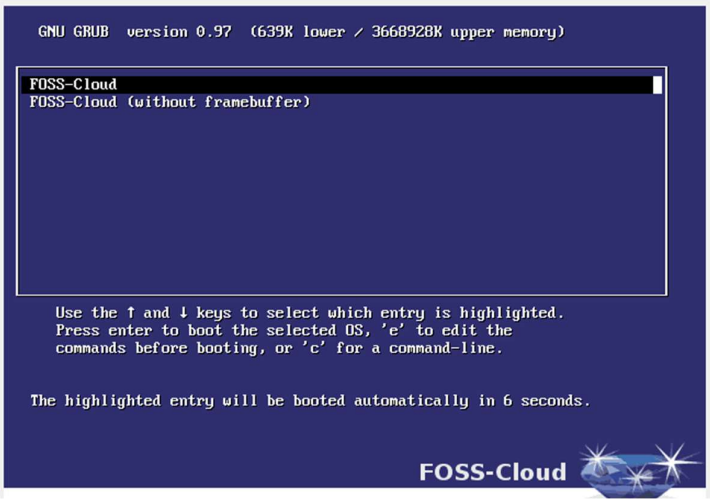




Step 18: Now select Boot from first hard disk and enter.

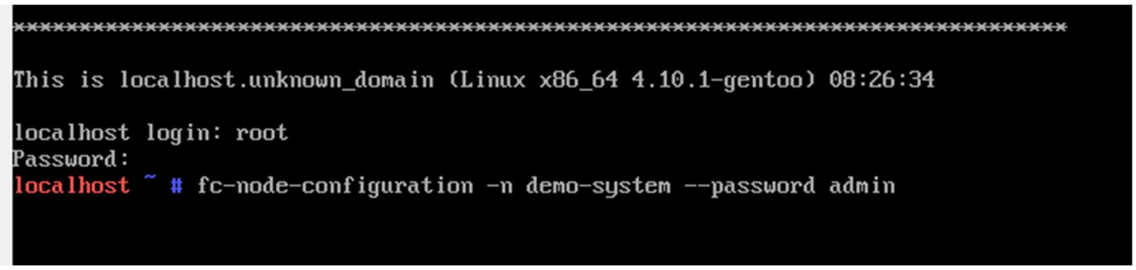


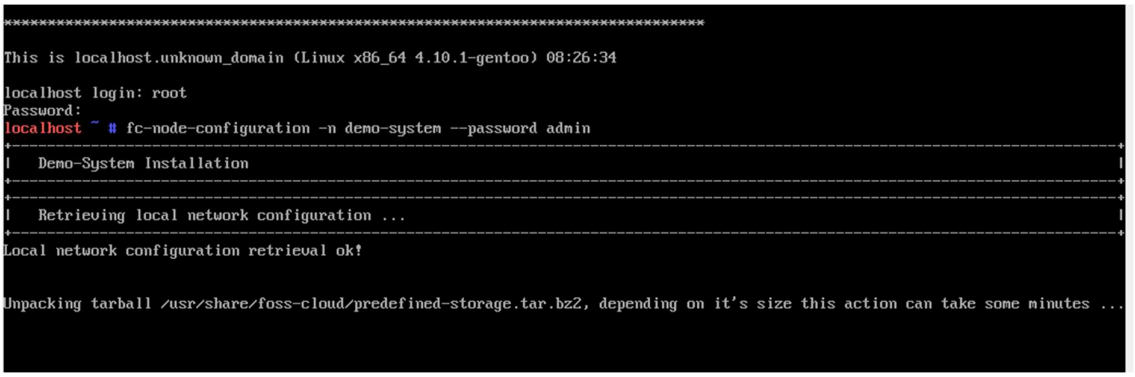
Step 19: Select FOSS-Cloud and press Enter.



Step 20: Now, login with user credentials given here.

Step 21: Add following command in front of localhost fc-node-configuration –n demo-system –password admin. Press Enter and the installation process will started.



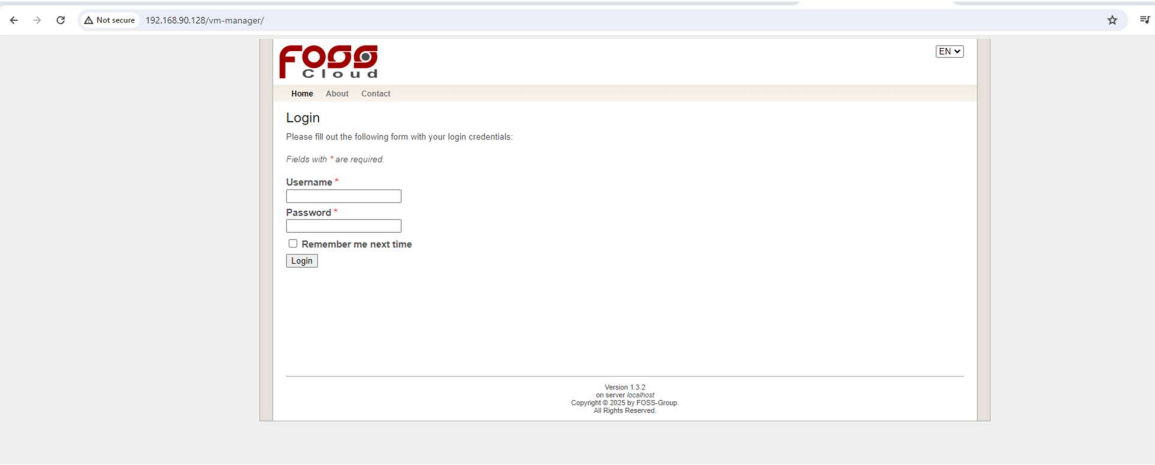


```
Starting the daemon dhcpd ...
Executing: /etc/init.d/dhcpd start...
 * /var/run/dhcp: creating directory
 * /var/run/dhcp: correcting owner
 * /var/lib/dhcp: correcting owner
 * /var/lib/dhcp/dhcpd.leases: creating file
 * /var/lib/dhcp/dhcpd.leases: correcting mode
 * /var/lib/dhcp/dhcpd.leases: correcting owner
 * Starting dhcpd ...                                                    [ ok ]
Started dhcpd successfully!

Adding the daemon dhcpd to the runlevel default ...
Executing: rc-update add dhcpd default ...
Added dhcpd successfully to runlevel default!

Directory /var/virtualization/backup created


Created backup directory symlink: /var/backup --> /var/virtualization/backup, if you want to change the backup directory you may
 do so by modifing this symlink.
File /etc/local.d/50-foss-cloud-firstrun.start deleted


Congratulations, you have finished the installation and configuration of this Node!

localhost ~ #
```

Step 22: Type ifconfig command to get inet of the device.

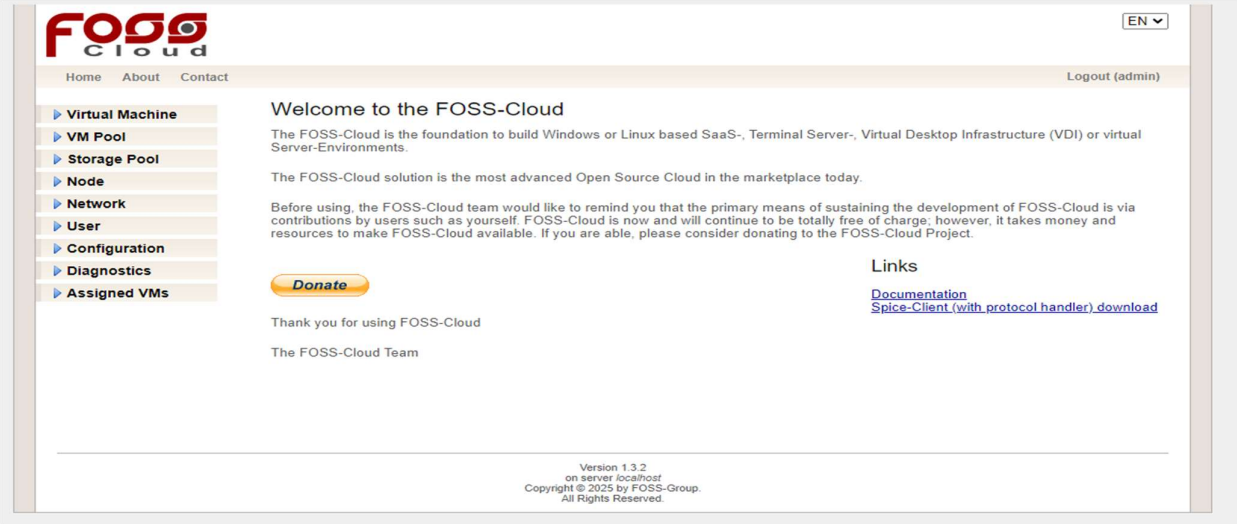
```
localhost ~ # ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.90.128  netmask 255.255.255.0  broadcast 192.168.90.255
        ether 08:00:27:8b:a5:e6  txqueuelen 1000  (Ethernet)
        RX packets 567  bytes 45035 (43.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 126  bytes 11421 (11.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

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 1510  bytes 378169 (369.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1510  bytes 378169 (369.3 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 52:bb:52:d5:2c:4b  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
```

Step 23: Open browser add ip address & press enter



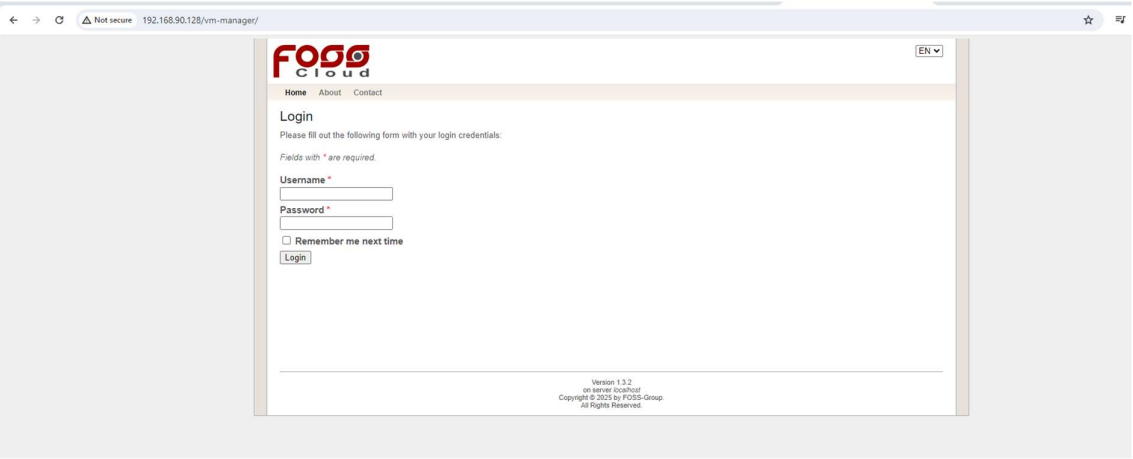
Step 24: Login with the credentials.

Step 25: You are done with the FOSS Cloud installation & setup.

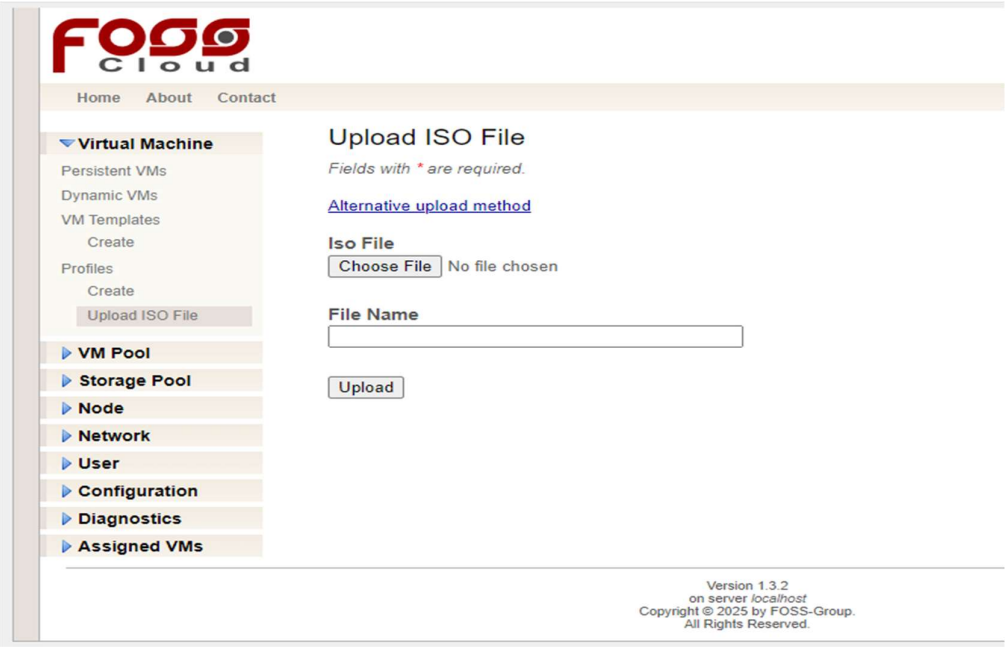


Platform as a Service (PaaS).

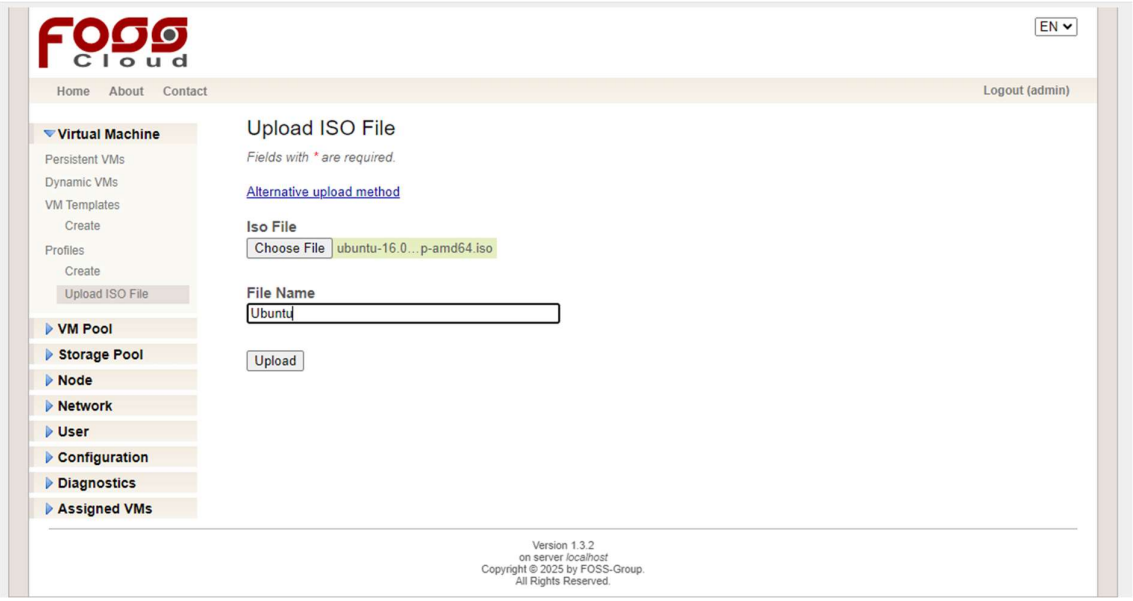
Step 1: After installation, it will show you an IP Address. Put it in your browser to access your administration page.



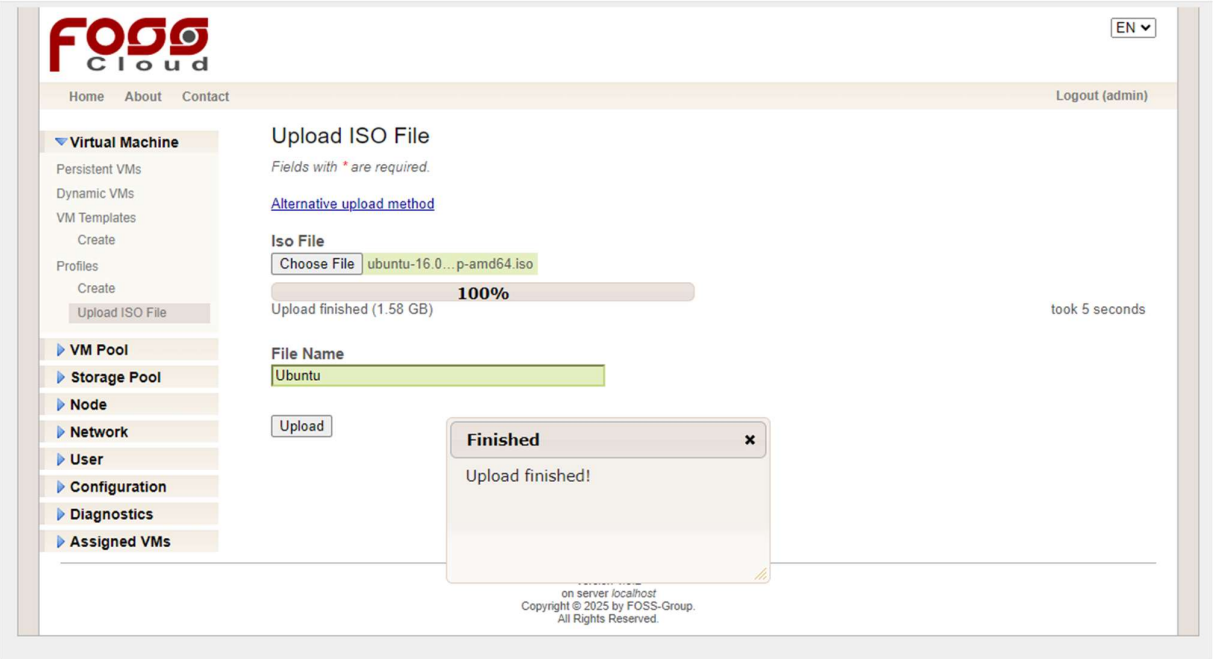
Step 2: Now select virtual machine → Upload ISO File & choose iso file to upload. You can upload either Ubuntu iso file or foss iso file or any other.



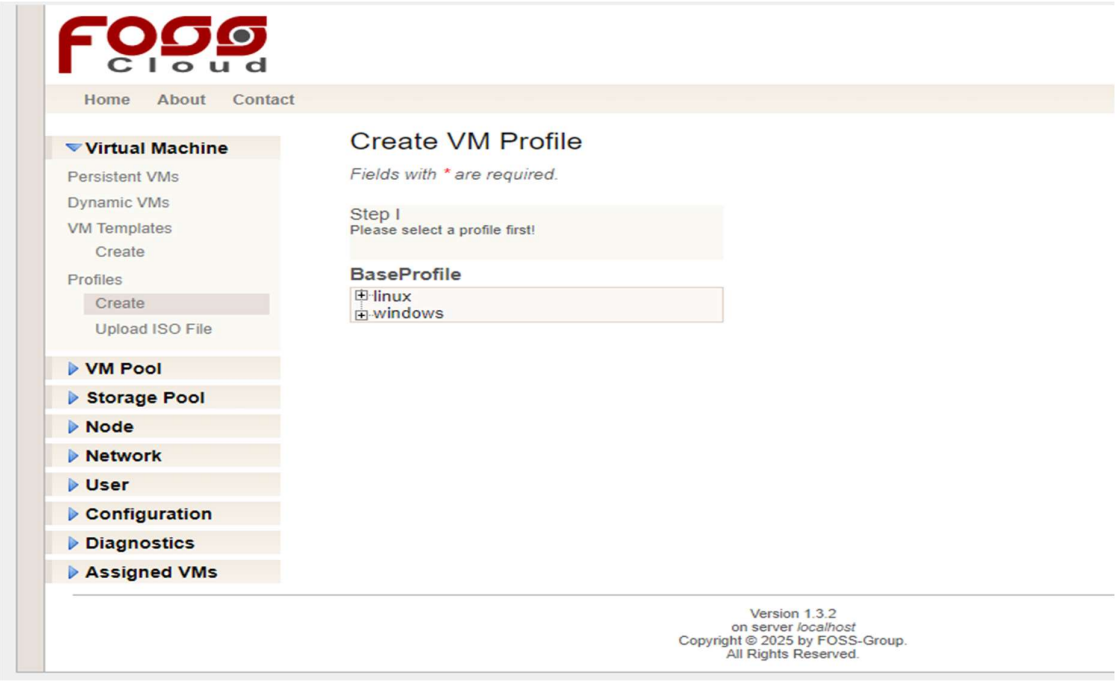
Step 3: Add file name after choosing the iso file.



Step 4: Now click on upload button.

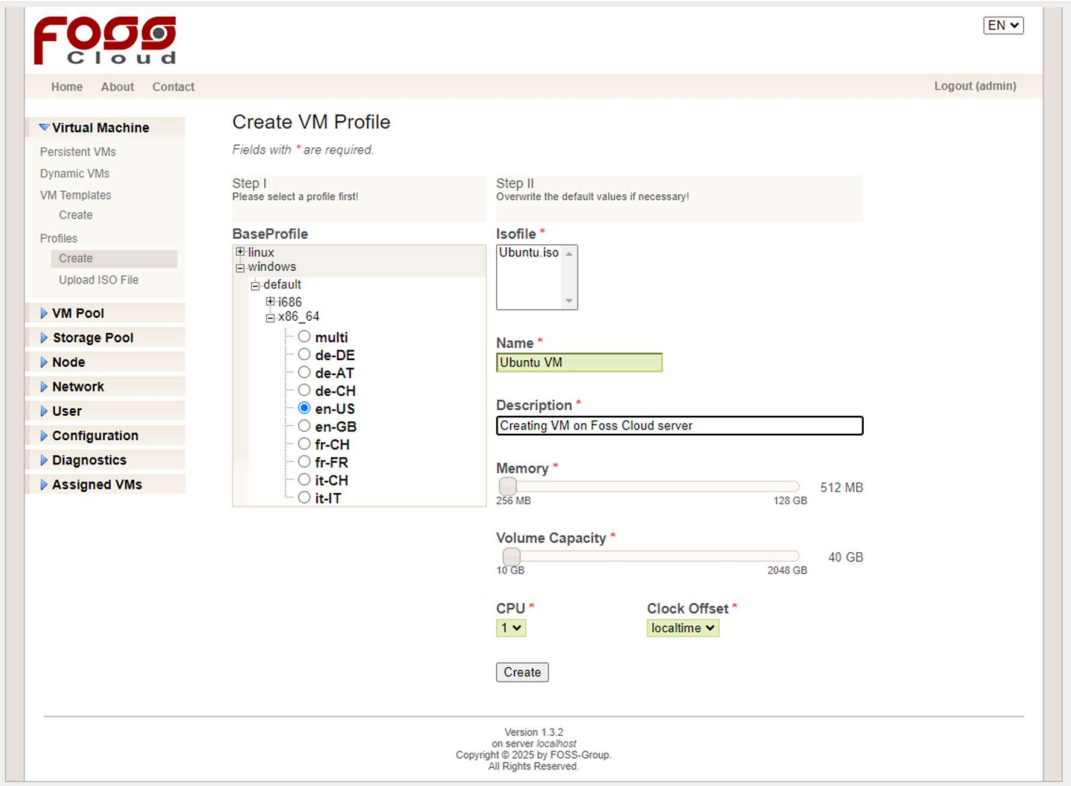


Step 5: Now after uploading file. Select create under the profile section.

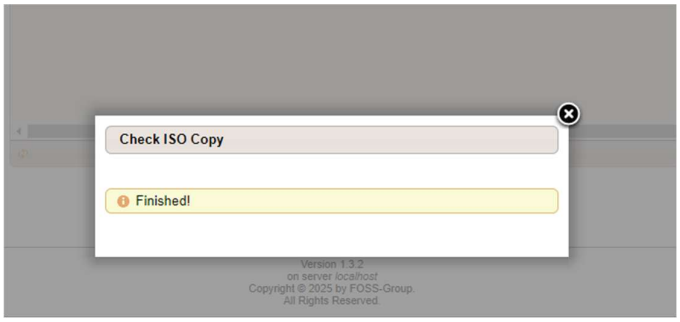


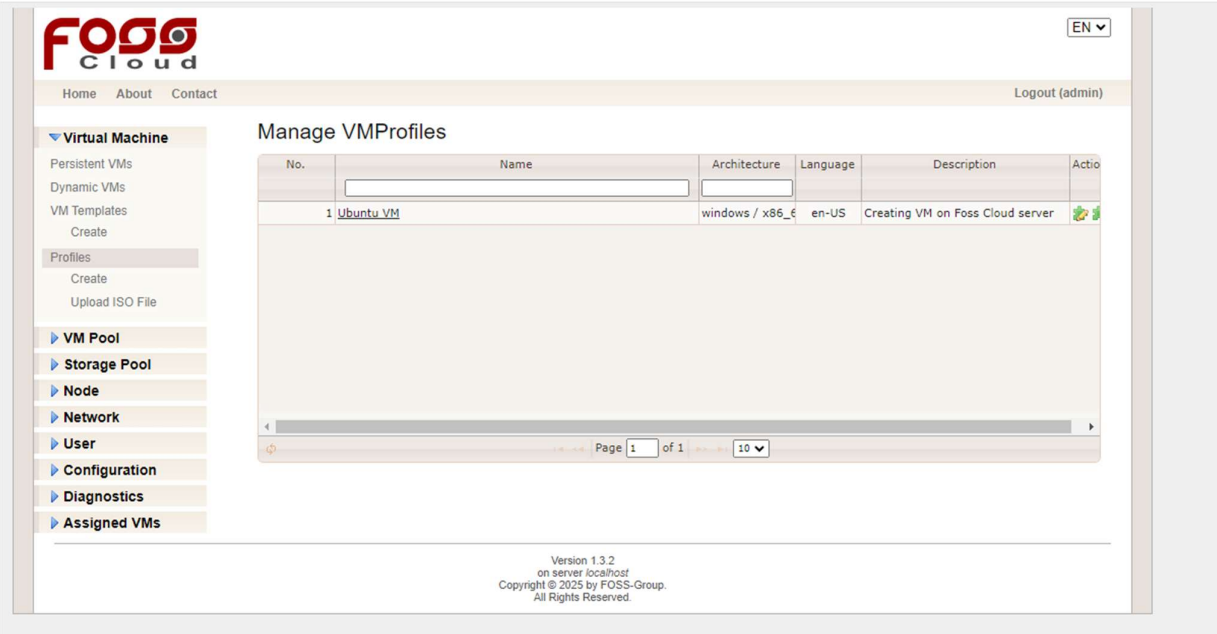
Step 6: Select in Step 1: select windows → en-US.

Step 2: Select iso file. Give Name to your Vm Profile. Add Description. Add memory and remaining details as given or provided & click on Create button given below.

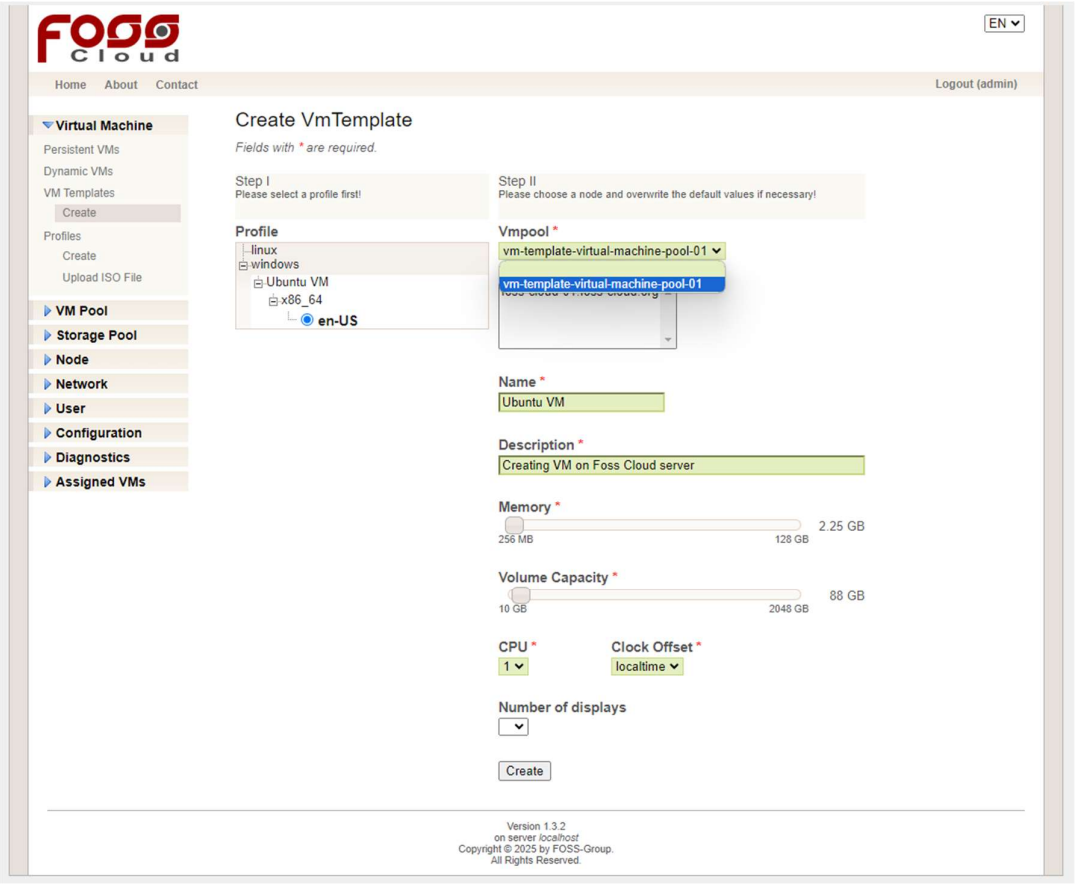


Step 7: After creating following page will open.

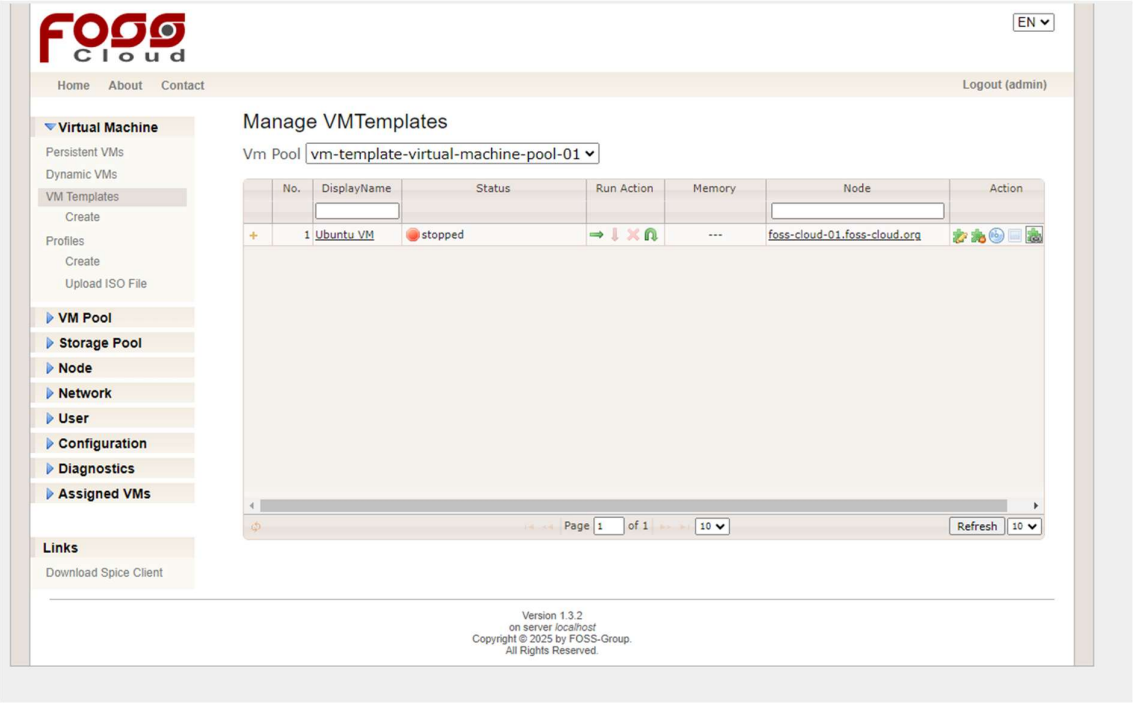




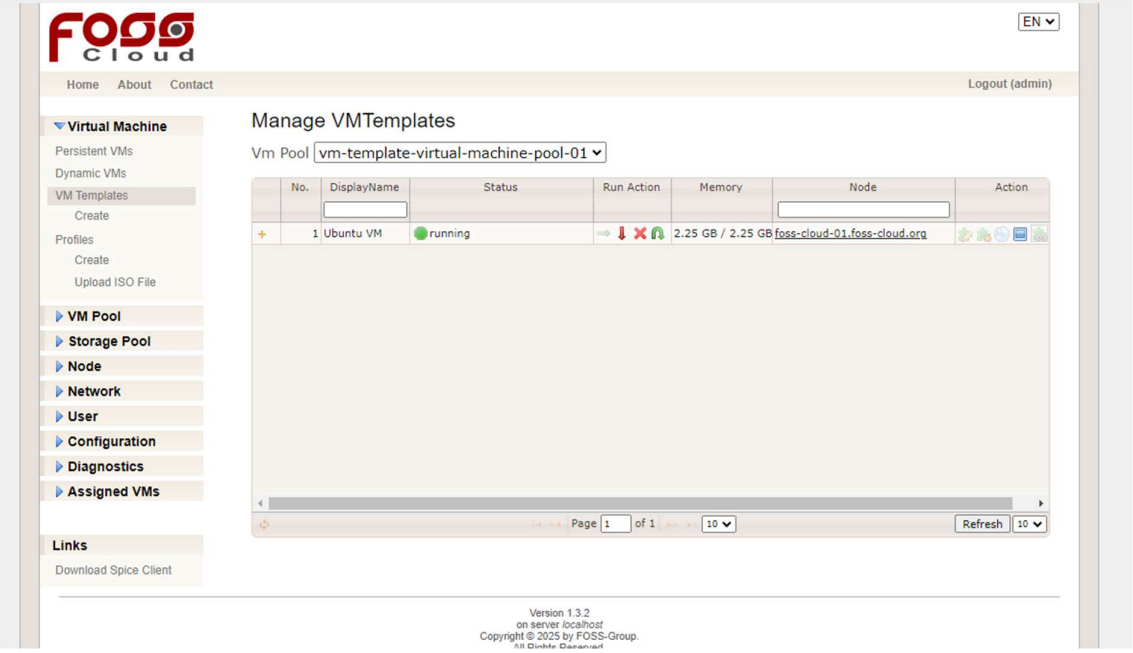
Step 8: Now, create VmTemplate.

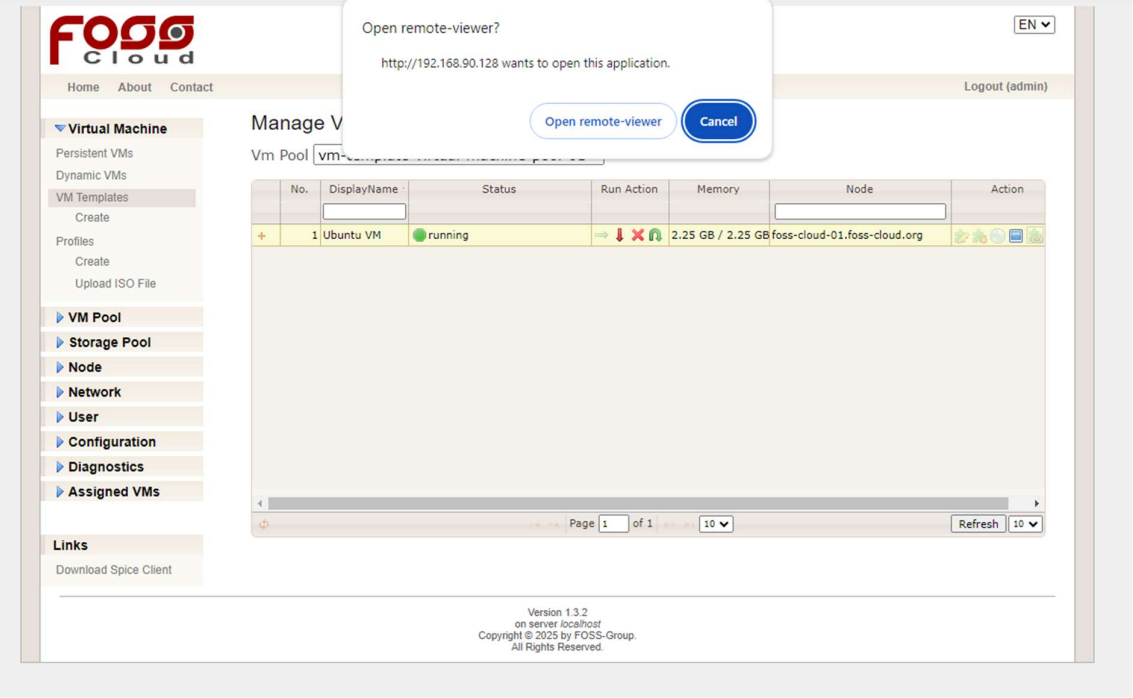


Step 9: After creating template Manage VMTemplate page will be displayed. Status is stopped to start it click on green Arrow in Run Action to start the status.



Step 10: Here, you can see the status is running.





11) Create a simple factorial web service using Python (with Flask), and call it from Java and .NET platforms.

Factorial_service.py

```
from flask import Flask, request, jsonify
app = Flask(__name__)

@app.route('/factorial')
def factorial():
    try:
        n = int(request.args.get('n'))
        f = 1
        for i in range(2, n+1): f *= i
        return jsonify({"factorial": f})
    except:
        return jsonify({"error": "Invalid input"}), 400

if __name__ == '__main__':
    app.run(debug=True)
```

FactorialClient.java

```
import java.net.*; import java.util.*; import java.io.*;

public class FactorialClient {
   public static void main(String[] args) throws Exception {
      System.out.print("Enter number: ");
      String n = new Scanner(System.in).nextLine();
      HttpURLConnection con = (HttpURLConnection) new
URL("http://localhost:5000/factorial?n=" + n).openConnection();
      new BufferedReader(new
InputStreamReader(con.getInputStream())).lines().forEach(System.out::prin
tln);
   }
}
```