

## 1)Currency Converter

Currency\_converted.py

```
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

```
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;
```

```
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

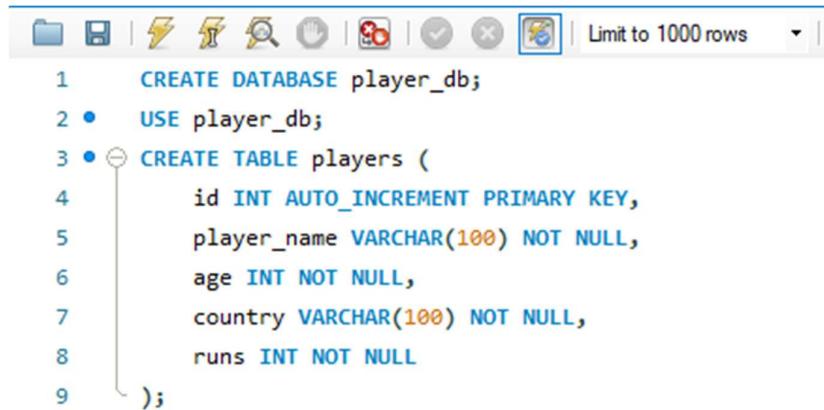
## 5) Create a simple rest service to demonstrate CRUD operations.

**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:



The screenshot shows a MySQL Workbench interface with a query editor window. The code entered is:

```
1 CREATE DATABASE player_db;
2 USE player_db;
3 CREATE TABLE players (
4     id INT AUTO_INCREMENT PRIMARY KEY,
5     player_name VARCHAR(100) NOT NULL,
6     age INT NOT NULL,
7     country VARCHAR(100) NOT NULL,
8     runs INT NOT NULL
9 );
```

**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\practicals\REST_CRUD> 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> []
```

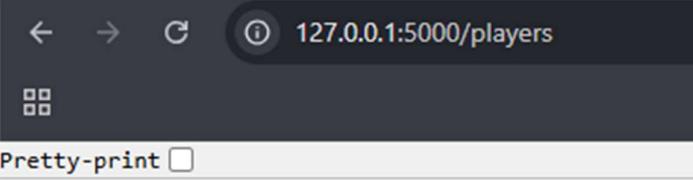
YOU will see that a new Player will Be created And you can Also update ,  
delete or get all the players details

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



```
[  
  {  
    "age": 35,  
    "country": "India",  
    "id": 1,  
    "player_name": "Rohit Sharma",  
    "runs": 120  
  }  
]
```

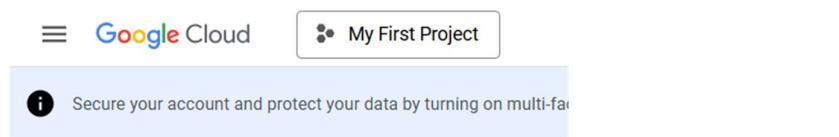
## 6) Develop application to consume Google's search / Google's Map RESTful Web service.

### SOLUTION:

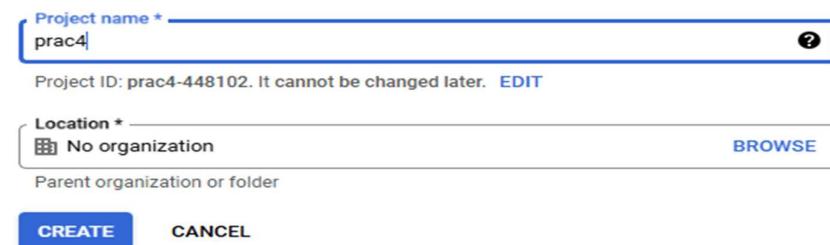
**Step 1:** Search “console cloud google” on chrome and sign up:

<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.



The screenshot shows the "Create New Project" dialog box. It has two main input fields: "Project name \*" containing "prac4" and "Location \*" containing "No organization". Below the location field is a "BROWSE" button. At the bottom of the dialog are two buttons: "CREATE" and "CANCEL".

Project name \*  ?

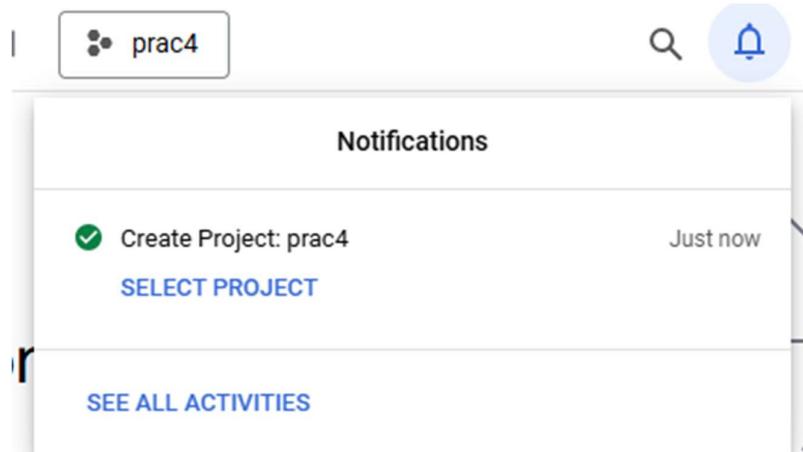
Project ID: prac4-448102. It cannot be changed later. [EDIT](#)

Location \*  BROWSE

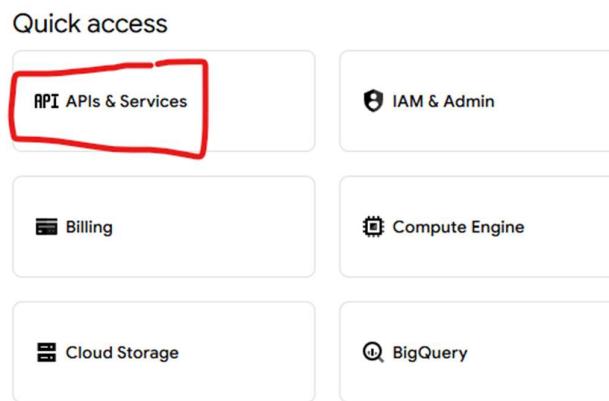
Parent organization or folder

[CREATE](#) [CANCEL](#)

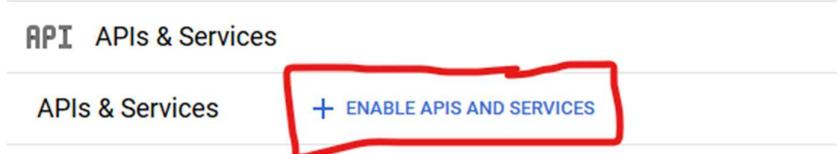
**Step 4:** You can see your project by clicking on bell icon.



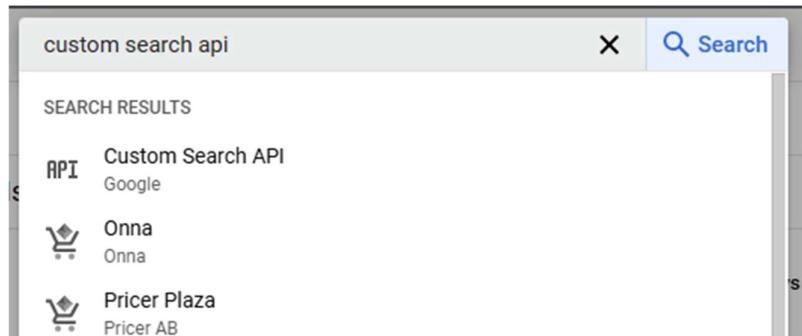
**Step 5 :** Click on bell icon and click “select project” Once you select the project then click on “APIs and Services” → Enabled APIs and services



Right click on APIs and Services and Enable them.

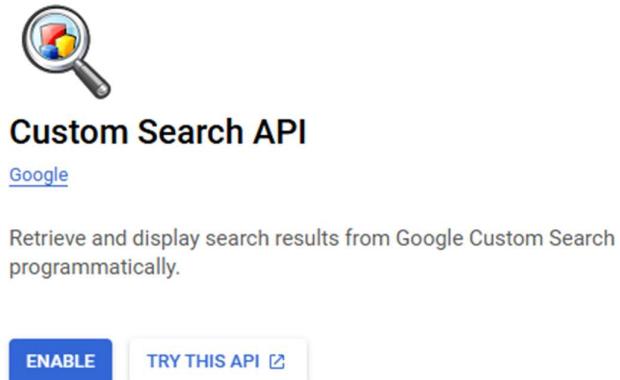


**Step 6:** Search “Custom search api” on search bar and click on first link.



**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.

The screenshot shows the Google Cloud Platform interface for managing APIs & Services. The left sidebar has 'APIs & Services' selected. Under 'Credentials', there is a sub-section titled 'API Keys'. A note says 'Create credentials to access your enabled APIs.' and 'Remember to configure the OAuth consent screen with information about your application.' Below this, a table lists 'No API keys to display'.

**Step 9:** API key will be created.

## API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

Your API key  
[REDACTED]  
[REDACTED]

**⚠** This key is unrestricted. To prevent unauthorized use, we recommend restricting where and for which APIs it can be used. [Edit API key](#) to add restrictions. [Learn more](#)

CLOSE

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

A screenshot of a code editor showing a file named `API_KEY`. The content of the file is a single line of text: `AIzaSyBQFO_fbQJgm21XQSoH3EL_F8HQzIkQG-E`.

**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’.

The screenshot shows a user interface for managing search engines. At the top, it says "All search engines". Below that, there is a message: "You don't have any search engines. [Create your first search engine!](#)". On the right side of the message are two buttons: "Delete" and "Add".

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

### Create a new search engine

Get started by providing some basic information about your engine. You'll be able to customize the engine's configs (Languages, regions, etc.) further after it is created.

[Learn more](#)

The screenshot shows a form for creating a new search engine. It includes fields for the search engine name ("prac4 Search Engine"), options for searching ("Search specific sites or pages" selected), search settings ("Image search" and "SafeSearch" toggled off), and a reCAPTCHA verification ("I'm not a robot" checked). At the bottom, there is a note about agreeing to terms of service and a "Create" button.

Name your search engine

What to search?   Search specific sites or pages

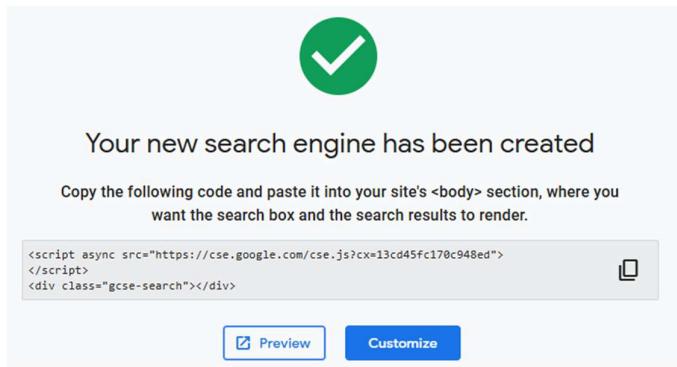
Search settings   Image search  
 SafeSearch

I'm not a robot  Privacy + Terms

By clicking 'Create', you agree with the [Terms of Service](#).

**Create**

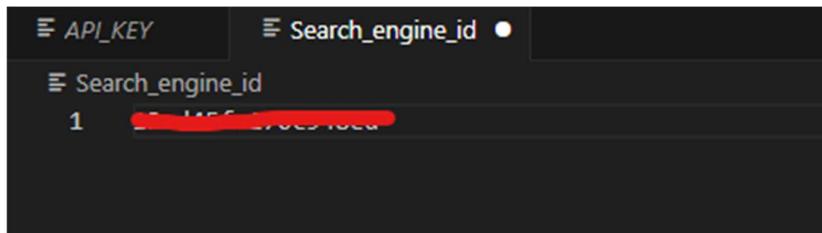
**Step 14:** A new search engine will be created.



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

Search engine name	prac4 Search Engine 
Description	Add description
Code	<a href="#">Get code</a>
Search engine ID	 
Public URL	<a href="https://cse.google.com/cse?cx=13cd45fc170c948ed">https://cse.google.com/cse? cx=13cd45fc170c948ed</a>

**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:**

```
import requests

# Replace with your actual API key and Search Engine ID
API_KEY = 'AlzaSyBQFO_fbQjgm21XQSsH3EL_F8HQzIkQG-E'
SEARCH_ENGINE_ID = '13cd45fc170c948ed'

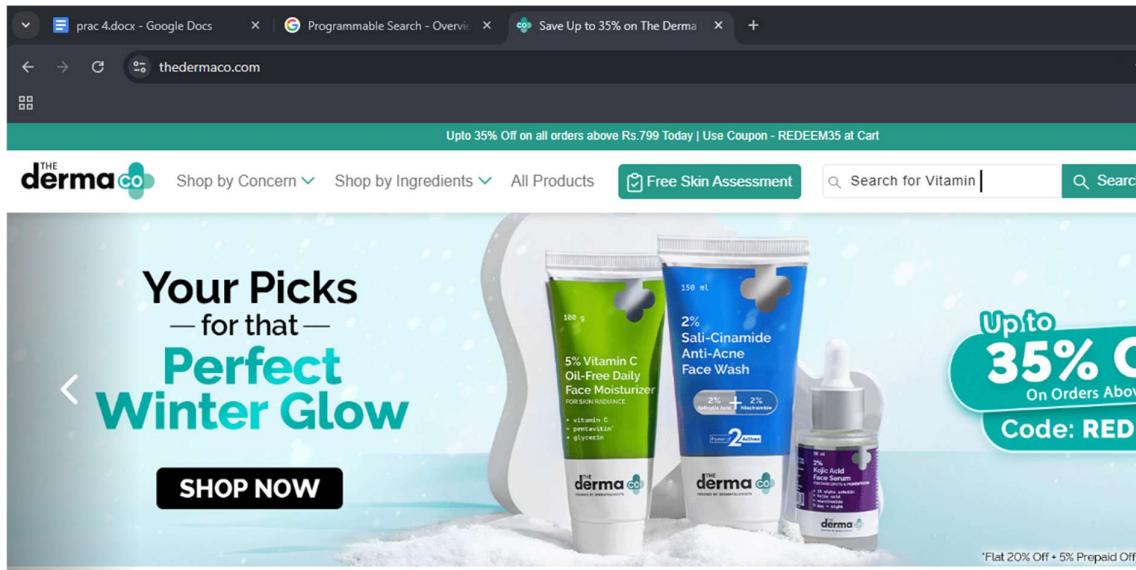
search_query = 'the derma co'
url = 'https://www.googleapis.com/customsearch/v1'
params = {
    'q': search_query,
    'key': API_KEY,
    'cx': SEARCH_ENGINE_ID
}

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  
virtinst libvirt-clients bridge-utils*

```
prathamesh@prathamesh-VirtualBox:~$ sudo apt install -y qemu-kvm virt-manager libvirt-daemon-system virtinst libvirt-clients 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*

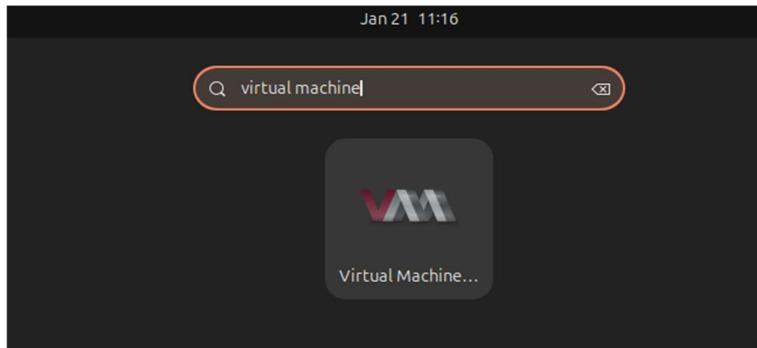
```
prathamesh@prathamesh-VirtualBox:~$ sudo systemctl enable --now libvirtd
sudosystemctl start libvirtd
```

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

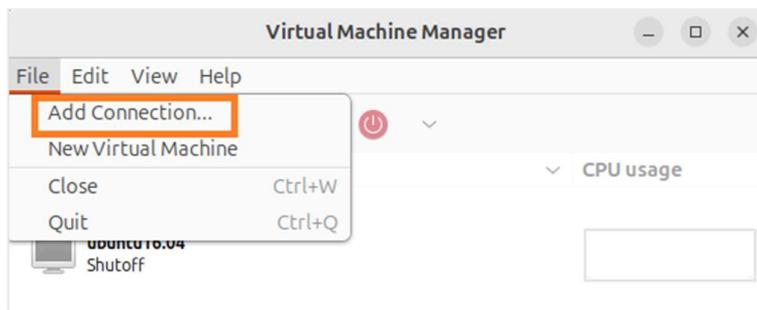
*sudosystemctl status libvirtd*

```
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
```

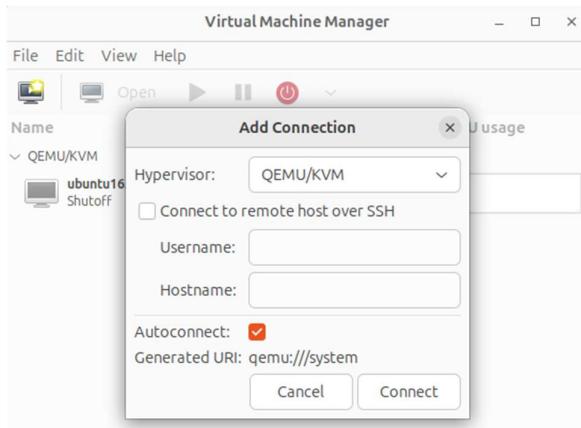
**Step 7:** In this step search virtual machine and start



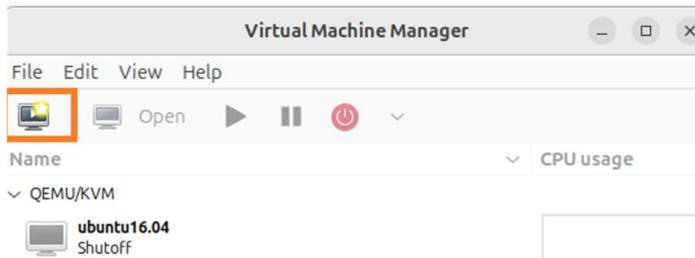
**Step 8:** Click add connection



**Step 9:** Now click connect



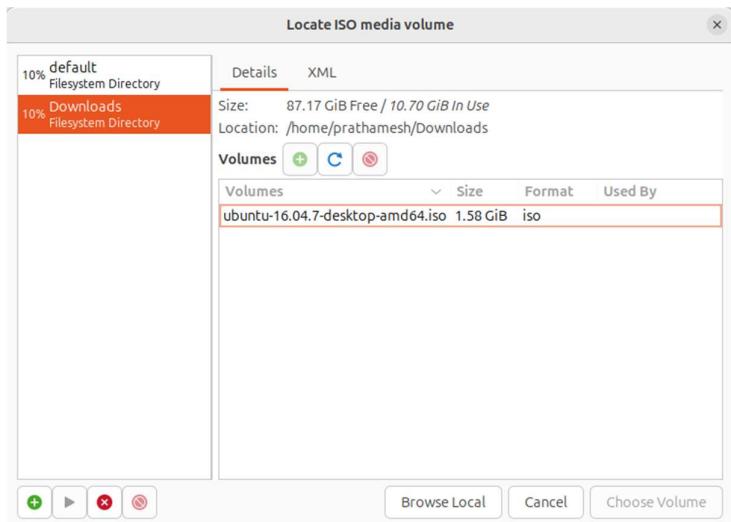
**Step 10:** Select the computer icon as shown in mage



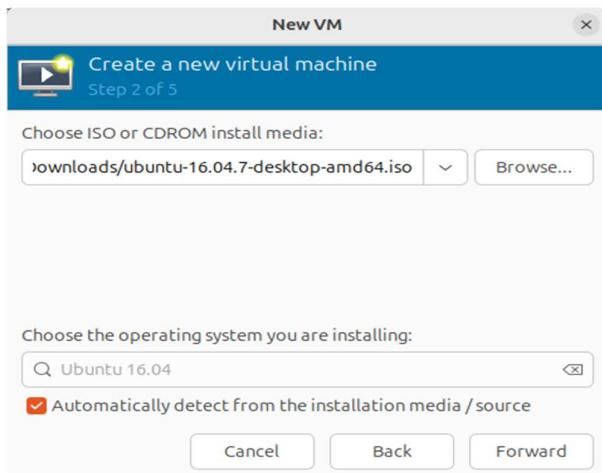
**Step 11: Select 'Local install media' and Click on **Forward****



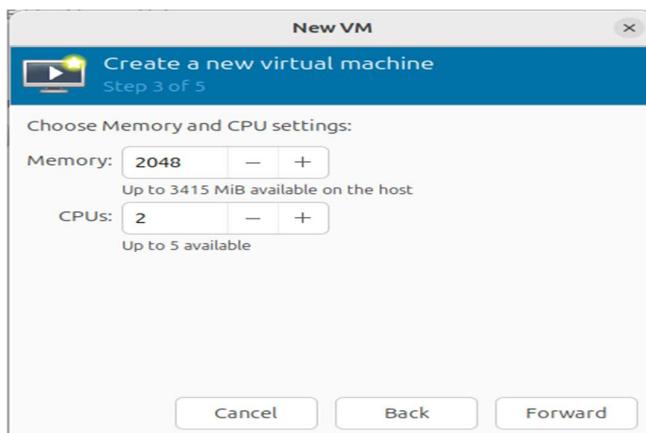
**Step 12: Select the .iso from Downloads and click on 'Choose volume'**



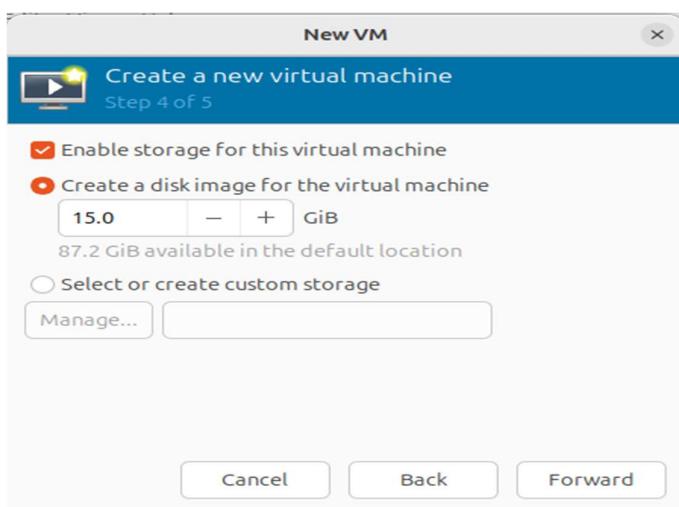
**Step 13: Click on 'Forward'**



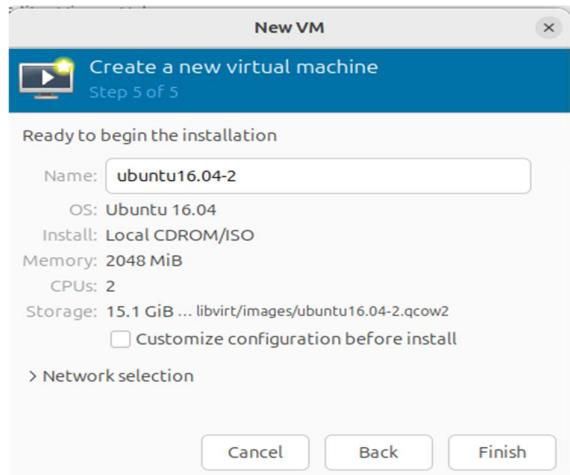
**Step 14: Select Memory= 2048 and CPUs = 2 and Click on ‘Forward’**



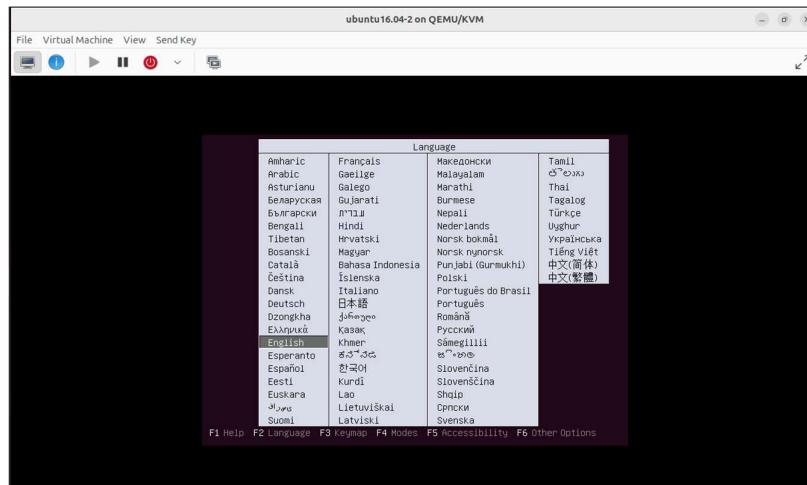
**Step 15: Click on Radio button and config. The disk image to 15.0 and click on Forward**



## 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

The screenshot shows the Postman interface with a new request being created. The method dropdown is set to POST. The URL field contains 'Enter URL or paste text'. On the left, a sidebar lists various HTTP methods: GET, POST (selected), PUT, PATCH, DELETE, HEAD, and OPTIONS. The main panel shows a table for managing headers, body, and settings.

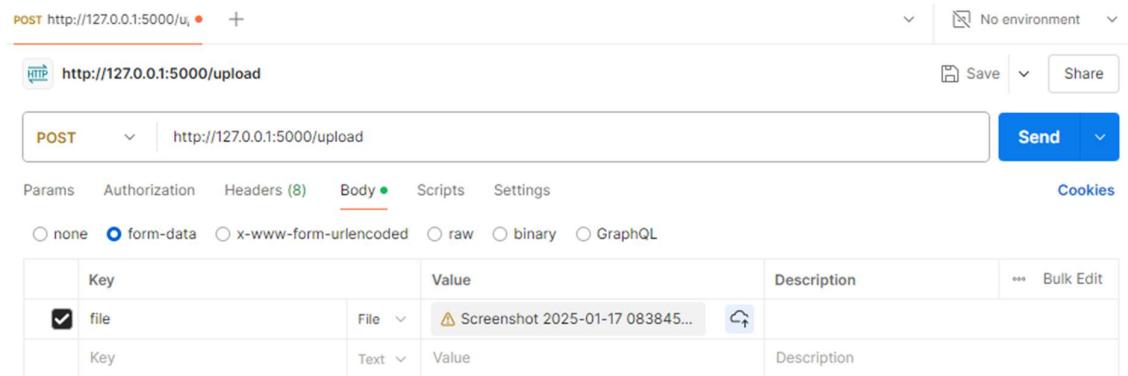
## Step 4: Enter the following url :<http://127.0.0.1:5000/upload>.

The screenshot shows the Postman interface with the URL 'http://127.0.0.1:5000/upload' entered in the address bar. The Params tab is selected in the navigation bar. The main panel shows a table for managing query parameters.

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

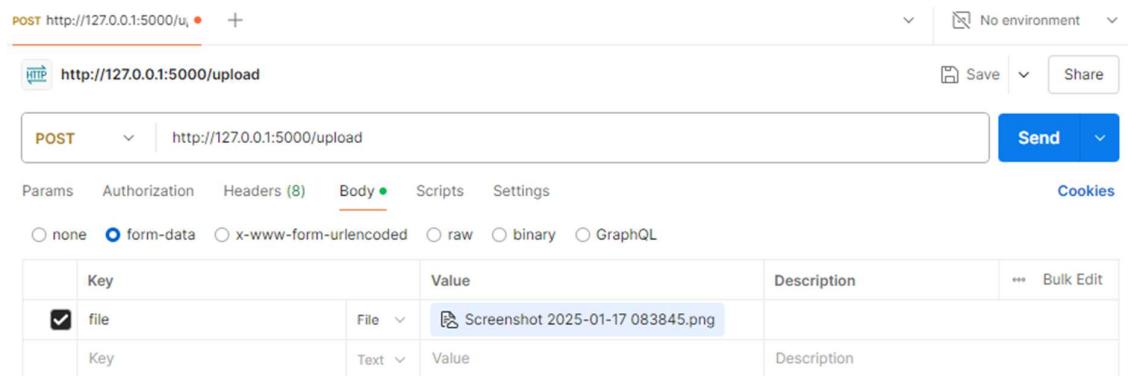
The screenshot shows the Postman interface with the Body tab selected in the navigation bar. The 'form-data' option is selected under the body type dropdown. The main panel shows a table for managing form-data fields.

## Step 6: Select file and name it as 'file'.



The screenshot shows the Postman interface for a POST request to `http://127.0.0.1:5000/upload`. The 'Body' tab is selected, showing a table with one row. The 'Key' column contains 'file', the 'File' dropdown is set to 'File', and the 'Value' column contains a file named 'Screenshot 2025-01-17 083845...'. There are other columns for 'Description' and 'Bulk Edit'.

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

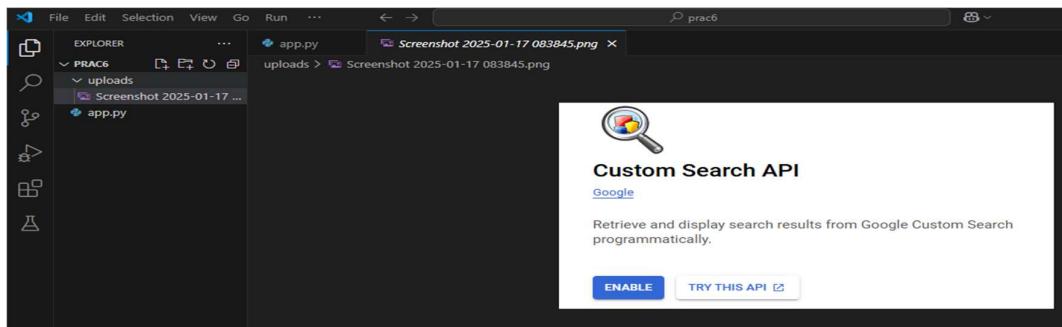


The screenshot shows the Postman interface after sending the request. The 'Body' tab is selected, showing a table with one row. The 'Key' column contains 'file', the 'File' dropdown is set to 'File', and the 'Value' column contains a file named 'Screenshot 2025-01-17 083845.png'. There are other columns for 'Description' and 'Bulk Edit'.



The screenshot shows the Postman interface displaying the API response. The status is '200 OK' with a response time of '340 ms' and a size of '247 B'. The response body is a JSON object with a single key-value pair: 'message': 'File 'Screenshot 2025-01-17 083845.png' uploaded successfully!'.

## 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  ussurri 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 and 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**

```
prathamesh@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



openstack.

Log in

User Name

Password

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



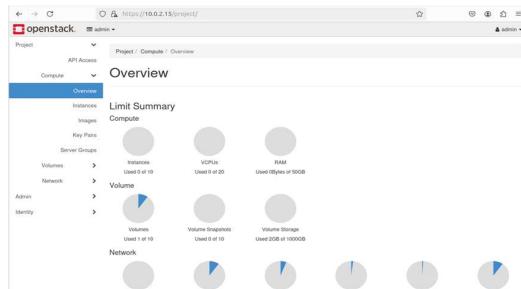
openstack.

Log in

User Name

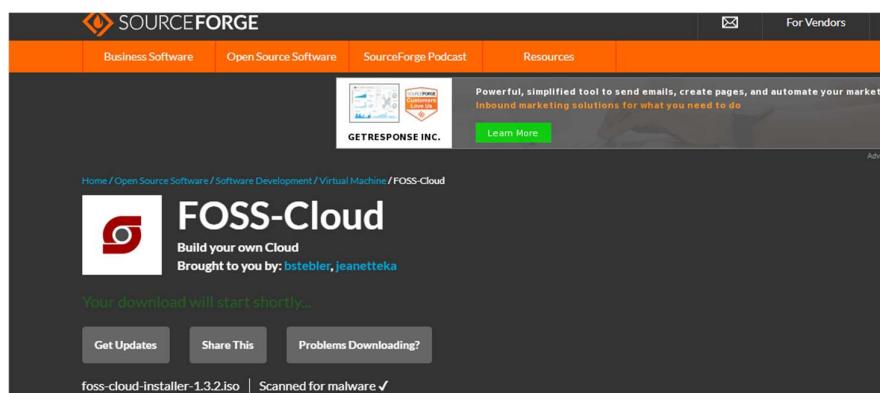
Password

**Congratulation you logged in OpenStack**



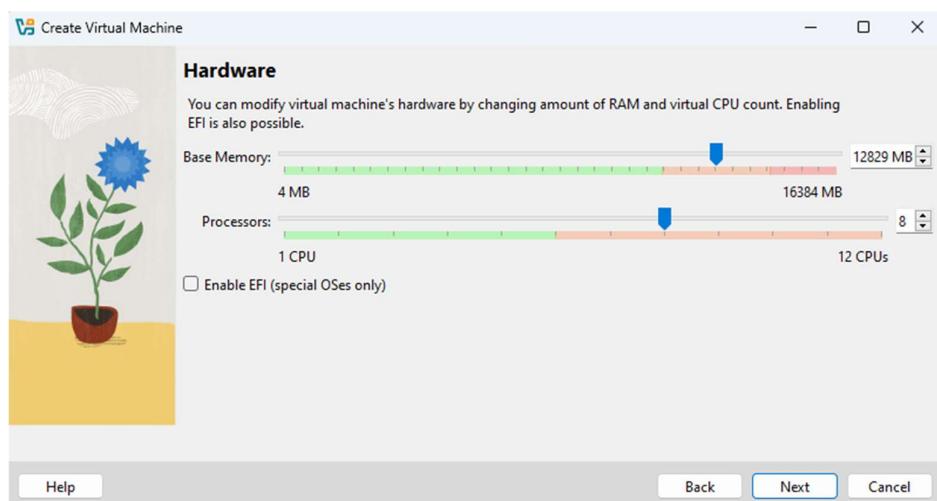
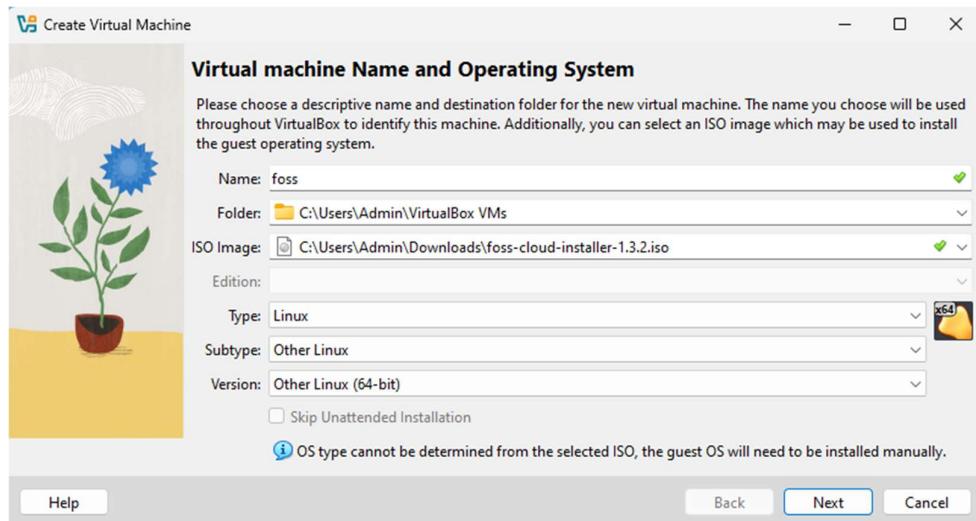
## **10) Aim: Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Storage & Platform as a Service (PaaS).**

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

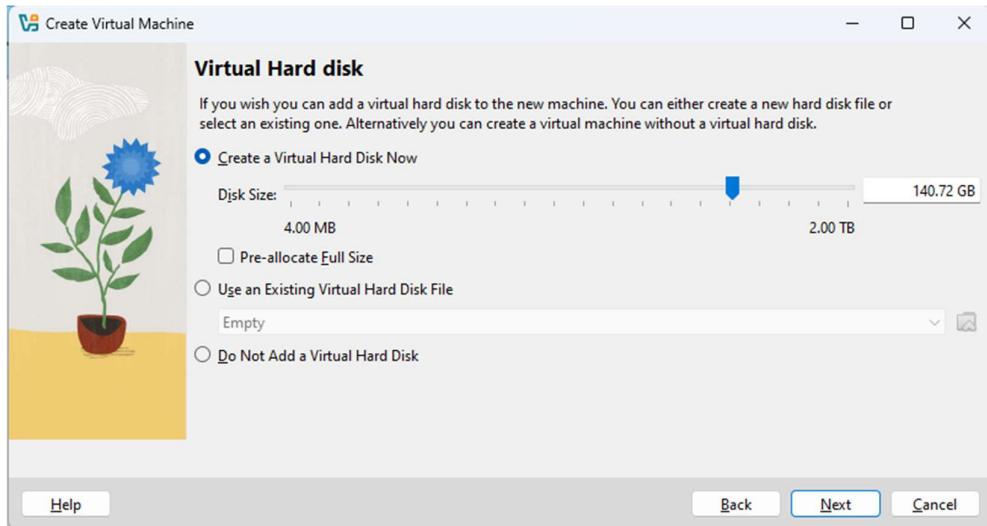


**Step 2:** Open Virtual Machine and create new Virtual Machine. Add your foss cloud iso file path at ISO Image.

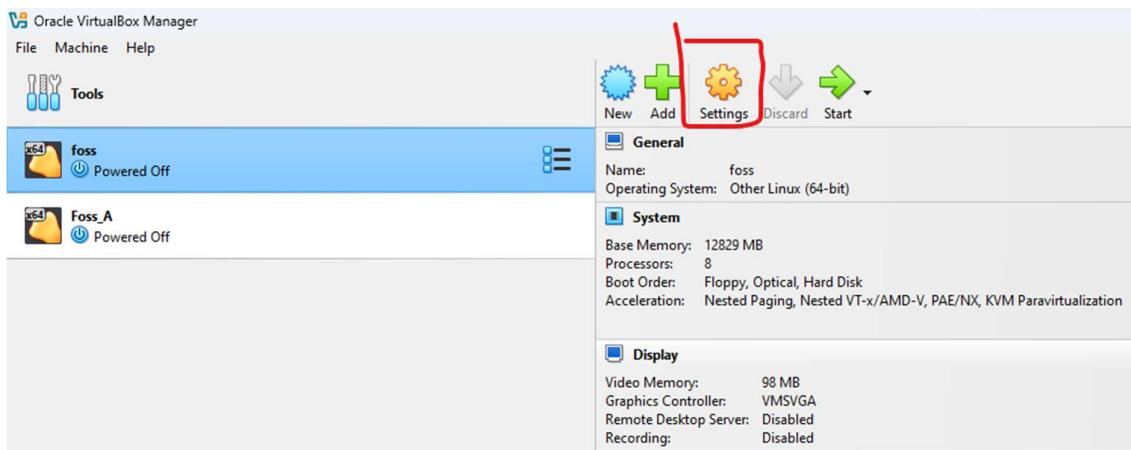
### Step 3: At the time of setup allocate Base memory.



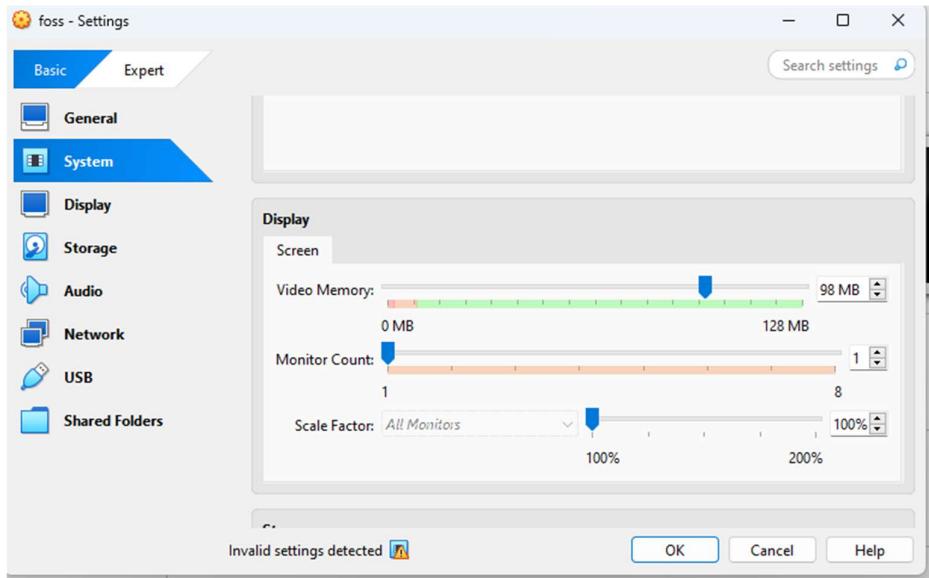
### Step 4: Create Virtual Hard Disk and give disk size.



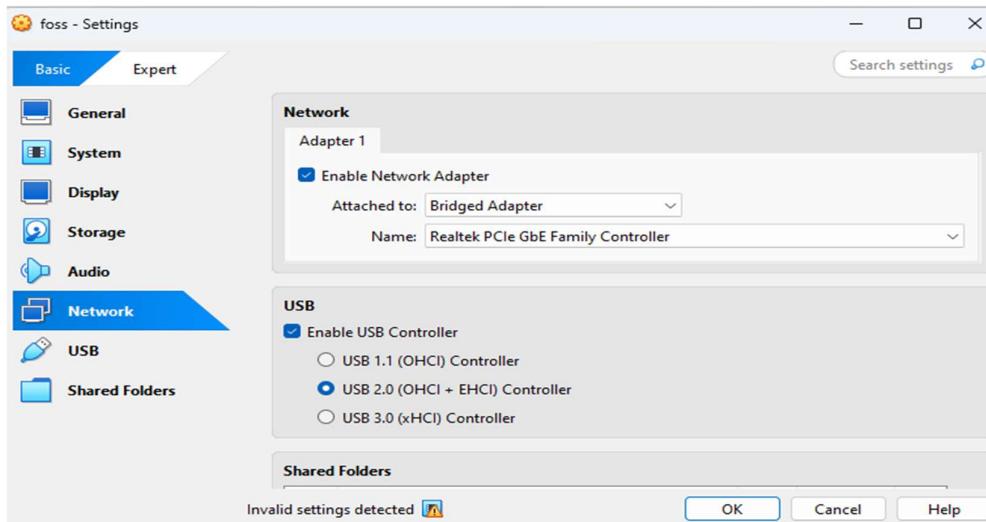
## Step 5: Click on settings.



## 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.

## Domain network

Networks at a workplace that are joined to a domain.

Have a question?

[Get help](#)

### Active domain networks

Not connected

Help improve Windows Security

[Give us feedback](#)

### Microsoft Defender Firewall

Helps protect your device while on a domain network.



Off

Change your privacy settings

View and change privacy settings  
for your Windows 11 Pro device.

[Privacy settings](#)

[Privacy dashboard](#)

[Privacy Statement](#)

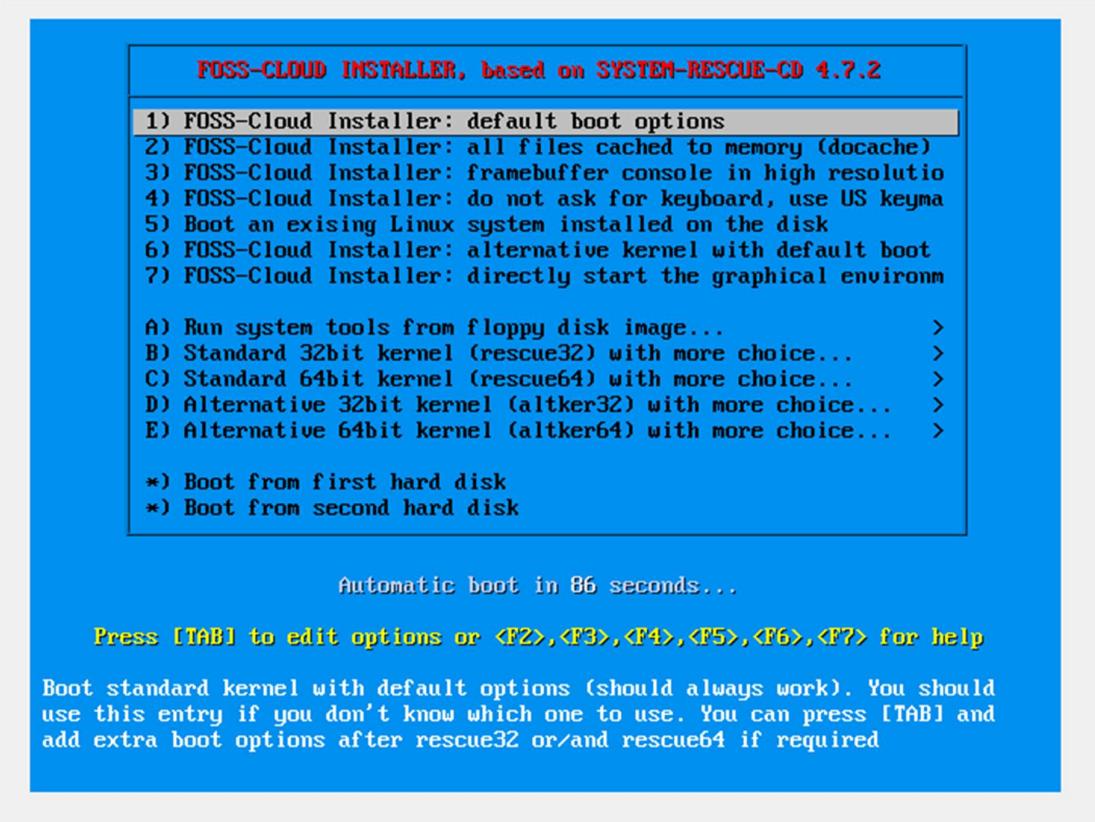
### Incoming connections

Prevents incoming connections when on a domain network.

Blocks all incoming connections, including those in the list of allowed apps.

The screenshot shows the Windows Security interface. On the left, there's a sidebar with options like Home, Virus & threat protection, Account protection, Firewall & network protection (which is selected), App & browser control, Device security, Device performance & health, Family options, and Protection history. The main area has a title 'Domain network' with a note that 'Firewall is off.' Below it is a 'Turn on' button. To the right, there's a sidebar with links for Microsoft Defender Firewall settings, privacy settings, and help options. At the bottom, there's a link to 'Allow an app through firewall'.

**Step 9:** Select 1<sup>st</sup> option here and Enter.

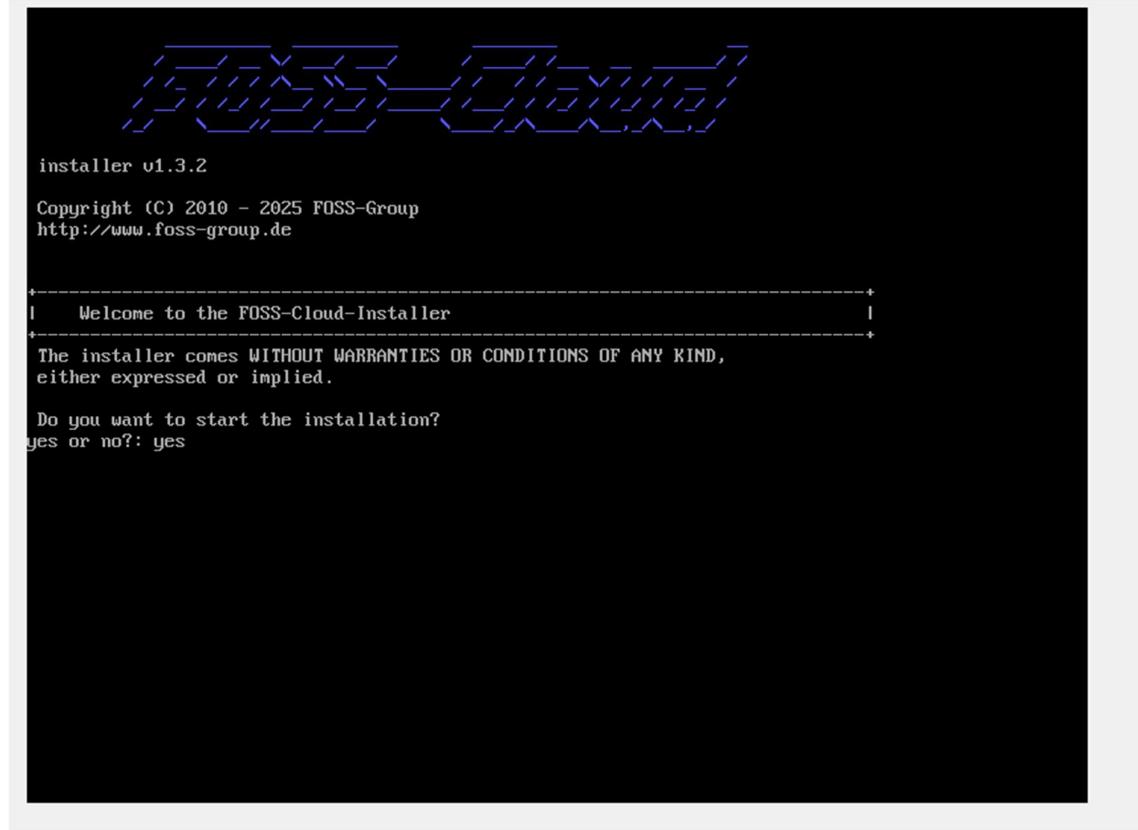


```
[ 25.150413] sr 2:0:0:0: Attached scsi generic sg1 type 5
[ 25.153127] Freeing unused kernel memory: 9764K (ffffffffff81f3f000 - ffffffff8
28c8000)
[ 25.153732] Write protecting the kernel read-only data: 14336k
[ 25.154987] Freeing unused kernel memory: 292K (fffff8800017b7000 - fffff880001
800000)
[ 25.159777] Freeing unused kernel memory: 1992K (fffff880001c0e000 - ffff88000
1e00000)
>> Loading kernel modules...
>> Waiting 1 seconds...
>> Loading keymaps
Please select a keymap from the following list by typing in the appropriate
name or number. You should prefer the name to the number (for example
type 'fr' instead of '16'). Hit Enter for the default 'us' keymap.

  1 azerty    2 be      3 bg      4 br-a     5 br-l     6 by      7 cf
  8 croat    9 cz      10 de     11 dk      12 dvorak   13 es      14 et
 15 fi       16 fr      17 gr     18 hu      19 il      20 is      21 it
 22 jp       23 la      24 lt     25 mk      26 nl      27 no      28 pl
 29 pt       30 ro      31 ru     32 se      33 sg      34 sk-y     35 sk-z
 36 slovene  37 trf     39 ua     40 uk      41 us      42 wangbe   43 fr_CH
 44 speakup  45 cs_CZ   46 de_CH  47 sg-lat1  48 fr-bepo  49 colemak 50 de_neo

default choice (US keymap) will be used if no action within 20 seconds
<< Load keymap (Enter for default): _
```

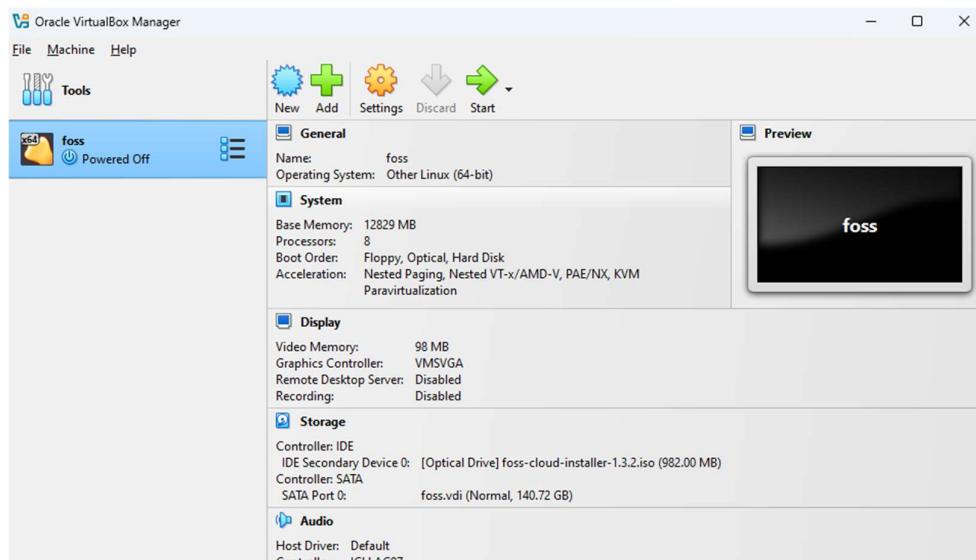
**Step 10:** Type yes to start the installation.



**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.

```
+-----+  
| Node Type Selection |  
+-----+  
The Installer supports four different types of servers:  
- The Demo-System which lets you quickly install and test the  
FOSS-Cloud on a single machine without any further network  
requirements.  
- The Single-Server which runs the whole FOSS-Cloud on  
a single physical server, without any high availability.  
- The VM-Node which hosts the virtual machines in a multi node setup  
(requires at least four physical servers).  
- The Storage-Node which serves the images of the virtual machines  
in a multi node setup (requires at least four physical servers).  
Please enter the number of the server type you would like to install  
1) Demo-System  
2) Single-Server  
3) VM-Node (multi node setup)  
4) Storage-Node (multi node setup)  
Node type:
```

## Step 14: Add device name sda.

```
+-----+  
| Installation Device Selection |  
+-----+  
A dedicated SCSI, SATA or PATA disk is required for the installation  
The disk has to be at least 130 GB in size  
  
Found sda (140 GB). Size is OK  
  
Below you will find a list of all detected and supported disks  
sda (140 GB)  
  
Please enter the device name on which you would like to install  
Device: sda_
```

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

```
+-----+  
| Installation Device Partitioning |  
+-----+  
Below is the existing partition layout of your selected device  
  
Error: /dev/sda: unrecognised disk label  
Model: ATA VBOX HARDDISK (scsi)  
Disk /dev/sda: 151GB  
Sector size (logical/physical): 512B/512B  
Partition Table: unknown  
Disk Flags:  
  
All existing partitions have to be deleted in order to continue  
THIS MEANS THAT ALL DATA ON THIS DISK WILL BE LOST  
Do you want to continue?  
yes or no?: yes
```

## Step 16: Add available Ethernet device name eg: enp0s3

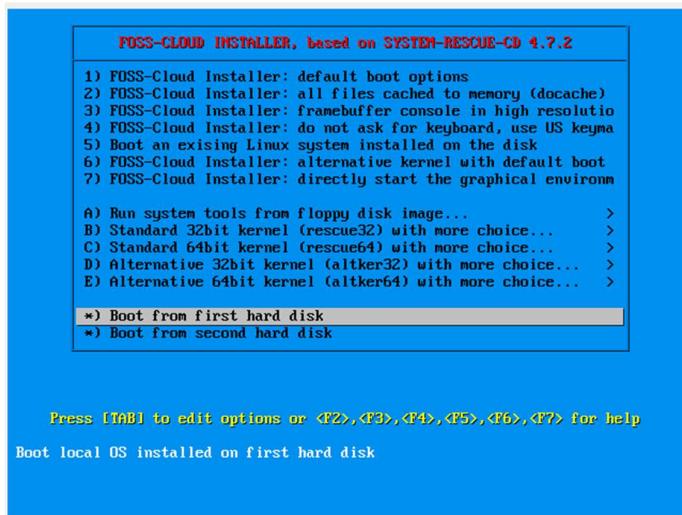
```
+-----+  
| Network Device Selection |  
+-----+  
  
Please enter the device which you would like to use  
  
Available ethernet devices: enp0s3  
Device #0: _
```

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

```
+-----+  
| Network Device Selection |  
+-----+  
  
Please enter the device which you would like to use  
  
Available ethernet devices: enp0s3  
Device #0: enp0s3  
  
+-----+  
| Network Configuration |  
+-----+  
  
Do you want to use automatic network configuration (via DHCP)?  
yes or no?: yes_
```

```
+-----+  
| Installation Complete |  
+-----+  
  
Congratulation! You have finished the installation of FOSS-Cloud  
Now all you need to do is reboot the system and remove the CD-ROM  
  
Do you want to reboot your system?  
yes or no?: 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.

```
*****  
This is localhost.unknown_domain (Linux x86_64 4.10.1-gentoo) 08:26:34  
localhost login: root  
Password:  
localhost ~ # fc-node-configuration -n demo-system --password admin
```

```
*****  
This is localhost.unknown_domain (Linux x86_64 4.10.1-gentoo) 08:26:34  
  
localhost login: root  
Password:  
localhost ~ # fc-node-configuration -n demo-system --password admin  
+-----+  
| Demo-System Installation  
+-----+  
+-----+  
| Retrieving local network configuration ...  
+-----+  
Local network configuration retrieval ok!  
  
Unpacking tarball /usr/share/foss-cloud/predefined-storage.tar.bz2, depending on it's size this action can take some minutes ...
```

```

Starting the daemon dhcpcd ...
Executing: /etc/init.d/dhcpcd start...
* /var/run/dhcpc: creating directory
* /var/run/dhcpc: correcting owner
* /var/lib/dhcpc: correcting owner
* /var/lib/dhcpc/dhcpcd.leases: creating file
* /var/lib/dhcpc/dhcpcd.leases: correcting mode
* /var/lib/dhcpc/dhcpcd.leases: correcting owner
Starting dhcpcd ...
Started dhcpcd successfully!

Adding the daemon dhcpcd to the runlevel default ...
Executing: rc-update add dhcpcd default ...
Added dhcpcd 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 modifying 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.

```

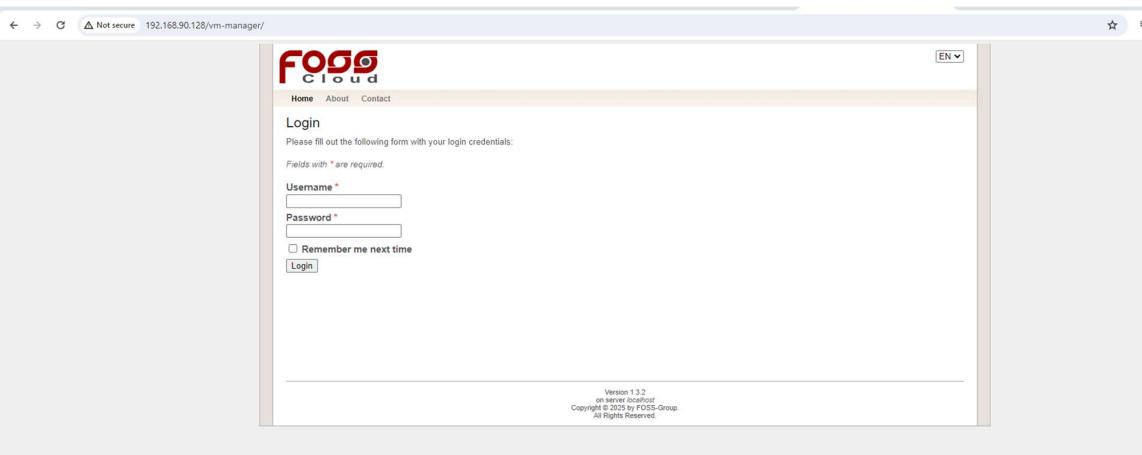
localhost ~ # ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.90.128 netmask 255.255.255.0 broadcast 192.168.90.255
                ether 08:00:27:8b:a5:e6 txqueuelen 1000 (Ethernet)
                RX packets 567 bytes 45035 (43.9 KiB)
                RX errors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

umbr0: 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.

Please fill out the following form with your login credentials.

Fields with \* are required.

Username \*

Password \*

Remember me next time

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

Welcome to the FOSS-Cloud

The FOSS-Cloud is the foundation to build Windows or Linux based SaaS-, Terminal Server-, Virtual Desktop Infrastructure (VDI) or virtual Server-Environments.

The FOSS-Cloud solution is the most advanced Open Source Cloud in the marketplace today.

Before using, the FOSS-Cloud team would like to remind you that the primary means of sustaining the development of FOSS-Cloud is via contributions by users such as yourself. FOSS-Cloud is now and will continue to be totally free of charge; however, it takes money and resources to make FOSS-Cloud available. If you are able, please consider donating to the FOSS-Cloud Project.

Links

[Documentation](#)  
[Spice-Client \(with protocol handler\) download](#)

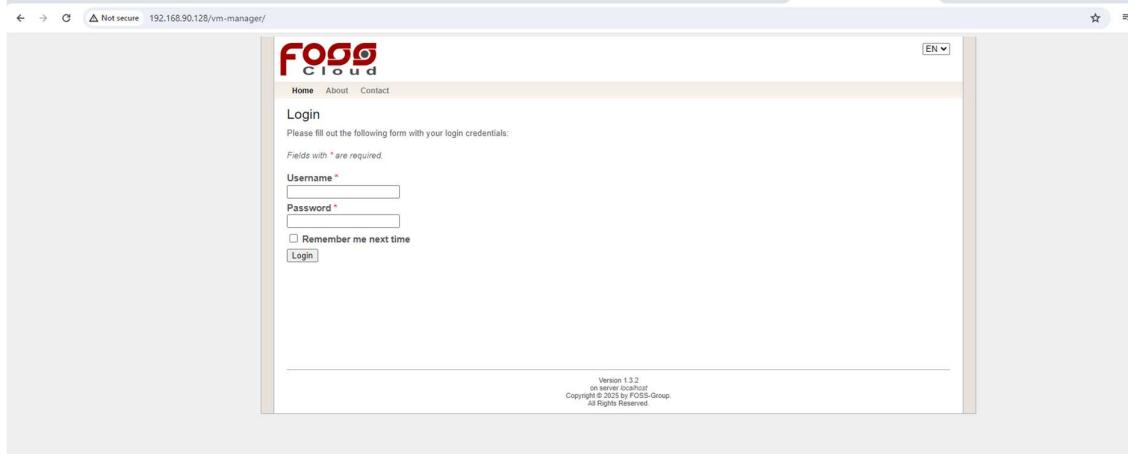
Thank you for using FOSS-Cloud

The FOSS-Cloud Team

Version 1.3.2  
on server *localhost*  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

## 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.

A screenshot of the FOSS Cloud VM Manager interface. The left sidebar shows a navigation menu with sections like Virtual Machine, VM Pool, Storage Pool, Node, Network, User, Configuration, Diagnostics, and Assigned VMs. The "Virtual Machine" section is expanded, and "Upload ISO File" is selected. The main content area is titled "Upload ISO File" and includes a note: "Fields with \* are required." Below this is a link "Alternative upload method". There are two input fields: "Iso File" with a "Choose File" button and a message "No file chosen", and "File Name" with a text input field. Below these is a "Upload" button. The footer at the bottom right is identical to the one in the previous screenshot: "Version 1.3.2 on server localhost Copyright © 2025 by FOSS-Group. All Rights Reserved."

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

**FOSS**  
Cloud

EN ▾

Logout (admin)

Home About Contact

Virtual Machine

- Persistent VMs
- Dynamic VMs
- VM Templates
  - Create
- Profiles
  - Create
- Upload ISO File

VM Pool

Storage Pool

Node

Network

User

Configuration

Diagnostics

Assigned VMs

Upload ISO File

Fields with \* are required.

Alternative upload method

Iso File  ubuntu-16.0...p-amd64.iso

File Name

Upload

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**Step 4:** Now click on upload button.

**FOSS**  
Cloud

EN ▾

Logout (admin)

Home About Contact

Virtual Machine

- Persistent VMs
- Dynamic VMs
- VM Templates
  - Create
- Profiles
  - Create
- Upload ISO File

VM Pool

Storage Pool

Node

Network

User

Configuration

Diagnostics

Assigned VMs

Upload ISO File

Fields with \* are required.

Alternative upload method

Iso File  ubuntu-16.0...p-amd64.iso

100%  
Upload finished (1.58 GB) took 5 seconds

File Name

Upload

Finished

Upload finished!

on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

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

The screenshot shows the FOSS Cloud web interface. The top navigation bar includes links for Home, About, and Contact. On the left, a sidebar menu lists categories like Virtual Machine, VM Pool, Storage Pool, Node, Network, User, Configuration, Diagnostics, and Assigned VMs. Under the Virtual Machine category, options for Persistent VMs, Dynamic VMs, VM Templates, Create, Profiles, and Upload ISO File are shown. The 'Create' option under Profiles is highlighted. The main content area is titled 'Create VM Profile' and contains a note: 'Fields with \* are required.' A step 1 message says 'Please select a profile first!' Below this, a 'BaseProfile' section shows two options: 'linux' and 'windows'. At the bottom of the page, a footer provides version information: 'Version 1.3.2 on server localhost Copyright © 2025 by FOSS-Group. All Rights Reserved.'

**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.

**Foss Cloud**

Home About Contact Logout (admin)

**Create VM Profile**

Fields with \* are required.

Step I  
Please select a profile first!

Step II  
Overwrite the default values if necessary!

**BaseProfile**

- linux
- windows
- default
  - i686
  - x86\_64
    - multi
    - de-DE
    - de-AT
    - de-CH
    - en-US
    - en-GB
    - fr-CH
    - fr-FR
    - it-CH
    - it-IT

**Isofile \***  
Ubuntu.iso

**Name \***  
Ubuntu VM

**Description \***  
Creating VM on Foss Cloud server

**Memory \***  
256 MB 512 MB

**Volume Capacity \***  
10 GB 2048 GB 40 GB

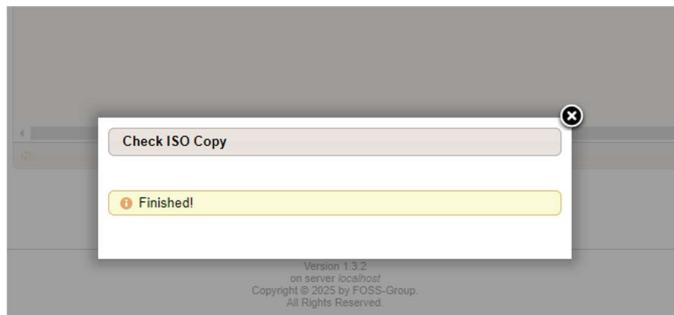
**CPU \*** 1

**Clock Offset \*** localtime

**Create**

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**Step 7:** After creating following page will open.



Manage VMProfiles

No.	Name	Architecture	Language	Description	Action
1	Ubuntu VM	windows / x86_6	en-US	Creating VM on Foss Cloud server	

Page 1 of 1 10

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

## Step 8: Now, create VmTemplate.

Create VmTemplate

Fields with \* are required.

Step I  
Please select a profile first!

Step II  
Please choose a node and overwrite the default values if necessary!

**Profile**

- linux
- windows
- Ubuntu VM
  - x86\_64
  - en-US

**Vmpool \***

- vm-template-virtual-machine-pool-01
- vm-template-virtual-machine-pool-01

**Name \***  
Ubuntu VM

**Description \***  
Creating VM on Foss Cloud server

**Memory \***  
2.25 GB

**Volume Capacity \***  
88 GB

**CPU \*** 1      **Clock Offset \*** localtime

**Number of displays**

**Create**

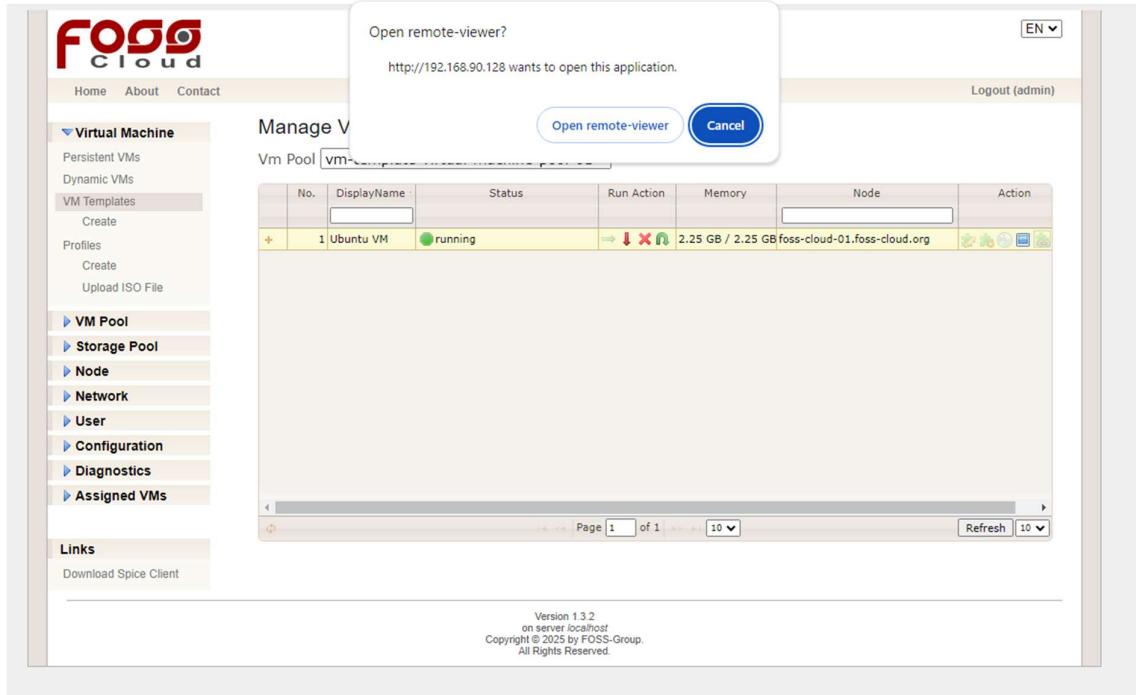
Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**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.

The screenshot shows the FOSS Cloud management interface. The left sidebar has a 'Virtual Machine' section with 'VM Templates' selected. The main area is titled 'Manage VMTemplates' and shows a table for 'Vm Pool: vm-template-virtual-machine-pool-01'. The table has columns: No., DisplayName, Status, Run Action, Memory, Node, and Action. One row is listed: No. 1, DisplayName 'Ubuntu VM', Status 'stopped', Run Action with a green arrow, Memory '2.25 GB / 2.25 GB', Node 'foss-cloud-01.foss-cloud.org', and Action with several icons. At the bottom, there's a footer with version information: 'Version 1.3.2 on server localhost Copyright © 2025 by FOSS-Group. All Rights Reserved.'

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

This screenshot is identical to the one above, showing the FOSS Cloud management interface. The 'Virtual Machine' section in the sidebar has 'VM Templates' selected. The main 'Manage VMTemplates' page shows the same table for 'Vm Pool: vm-template-virtual-machine-pool-01'. The difference is in the 'Status' column for the single row, which now shows 'running' instead of 'stopped'. The rest of the interface, including the table structure and footer, remains the same.



## 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);  
    }  
}
```