**VCC-Assignment1**

**M23AID053-Bhuvaneswari J**

**Installation of Virtual box:**

Downloaded windows host version from below link and installed Virtual box.

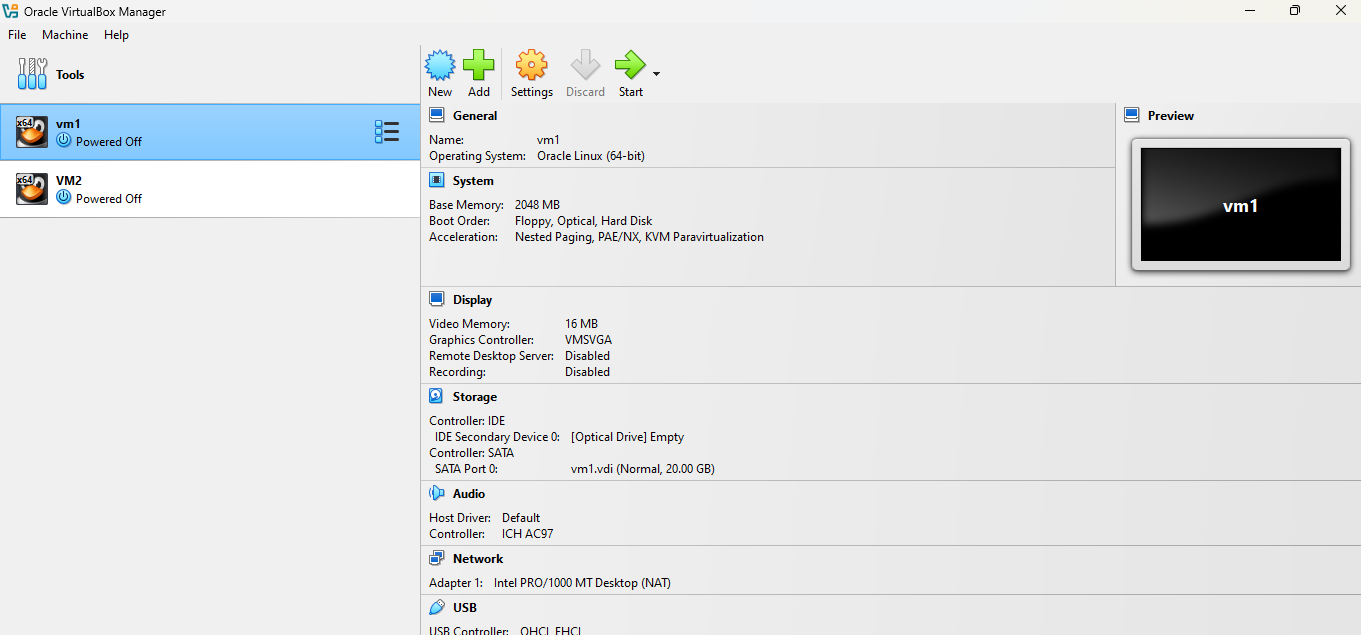
[**https://www.virtualbox.org/wiki/Downloads**](https://www.virtualbox.org/wiki/Downloads)

**Creation of VMs:**

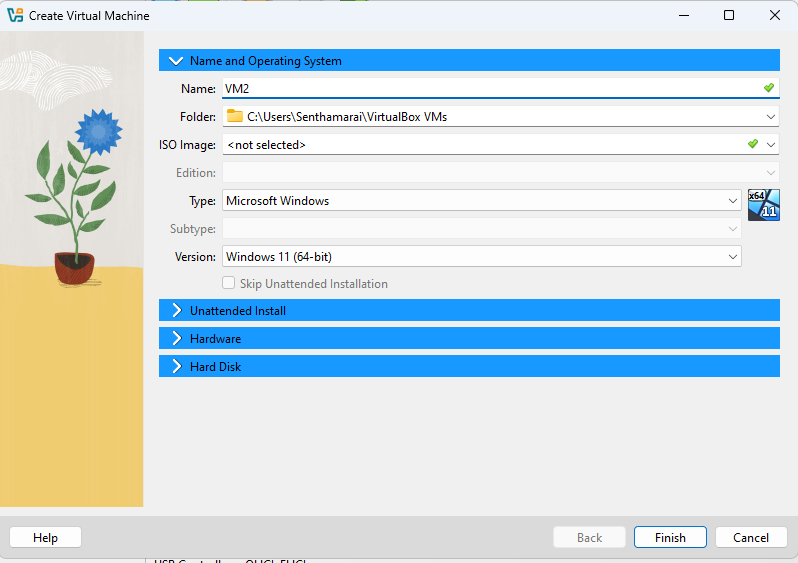
Downloaded puppy linux iso file (20.04 - [F96-CE](https://f96.puppylinux.com/) (64 bit)) from below link before VM creation.

[**https://forum.puppylinux.com/puppy-linux-collection**](https://forum.puppylinux.com/puppy-linux-collection)

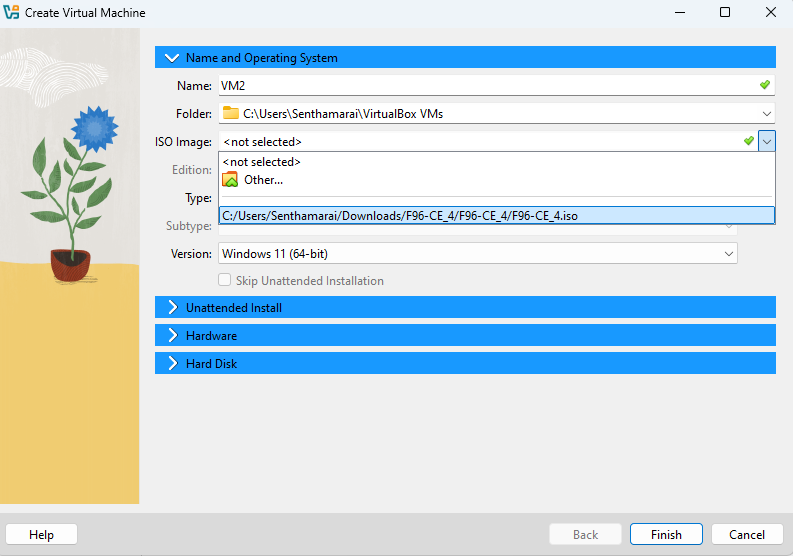
Click New:

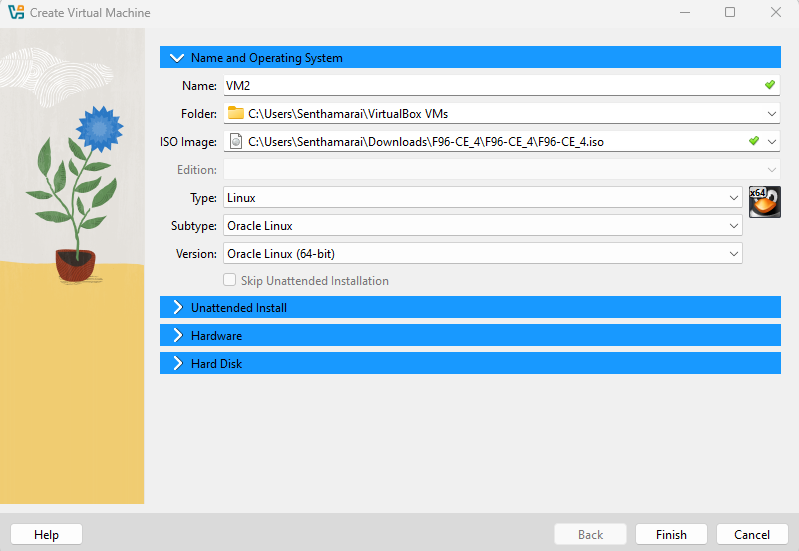
****

Add VM name

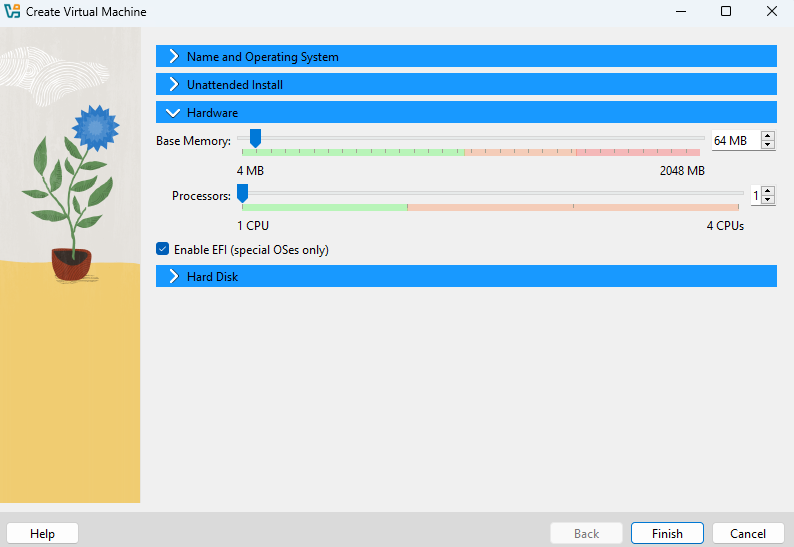
****

Upload puppy linux ISO file

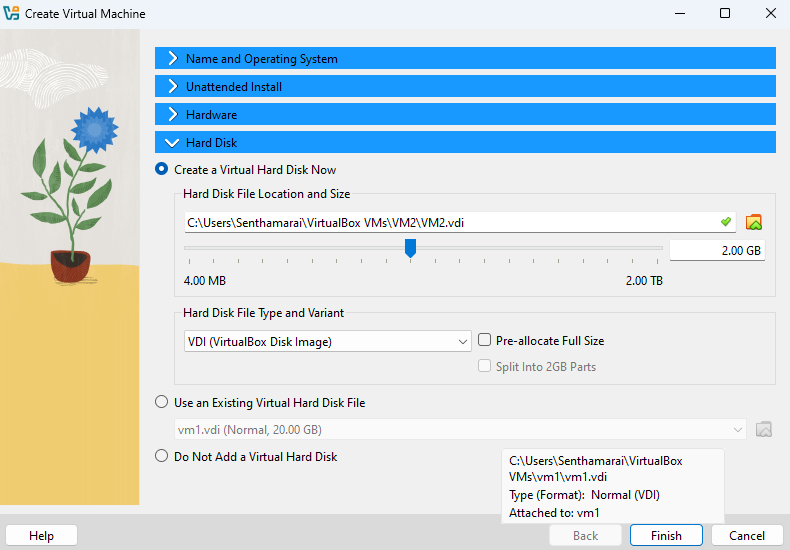
****

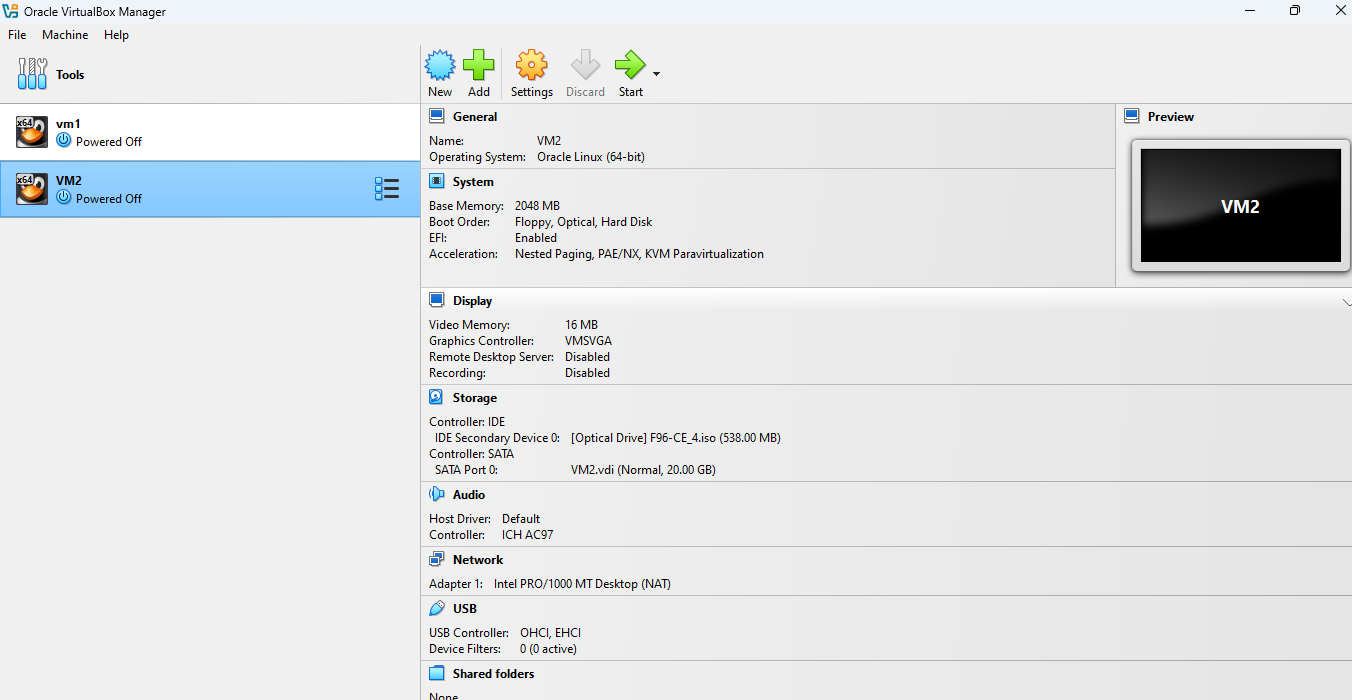
****

Choose memory size

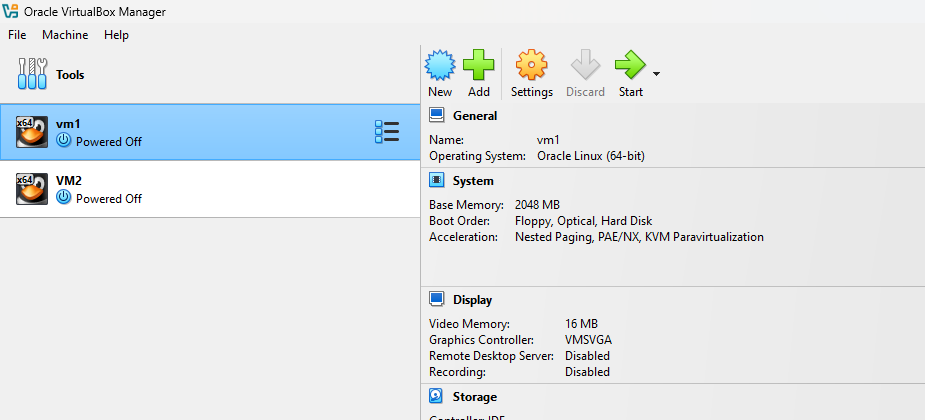
****

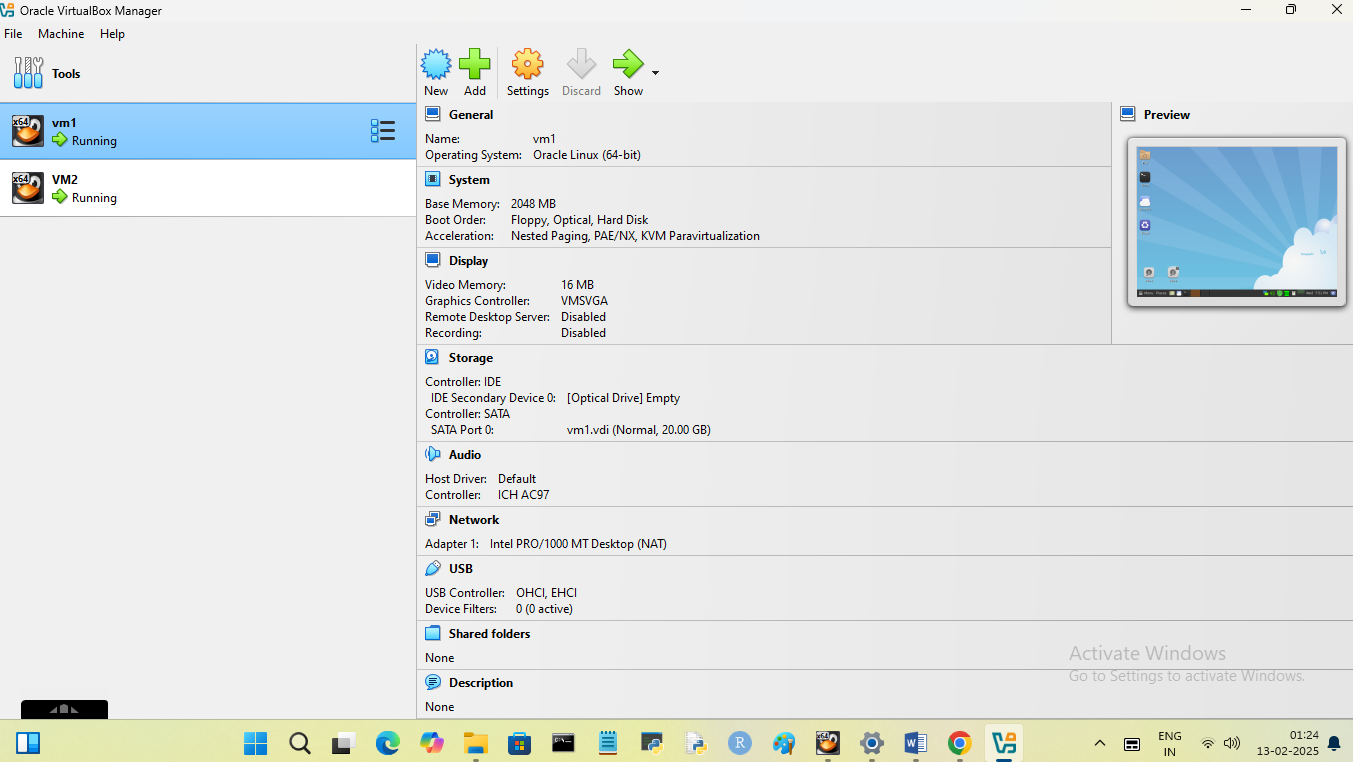
Click finish to create VM

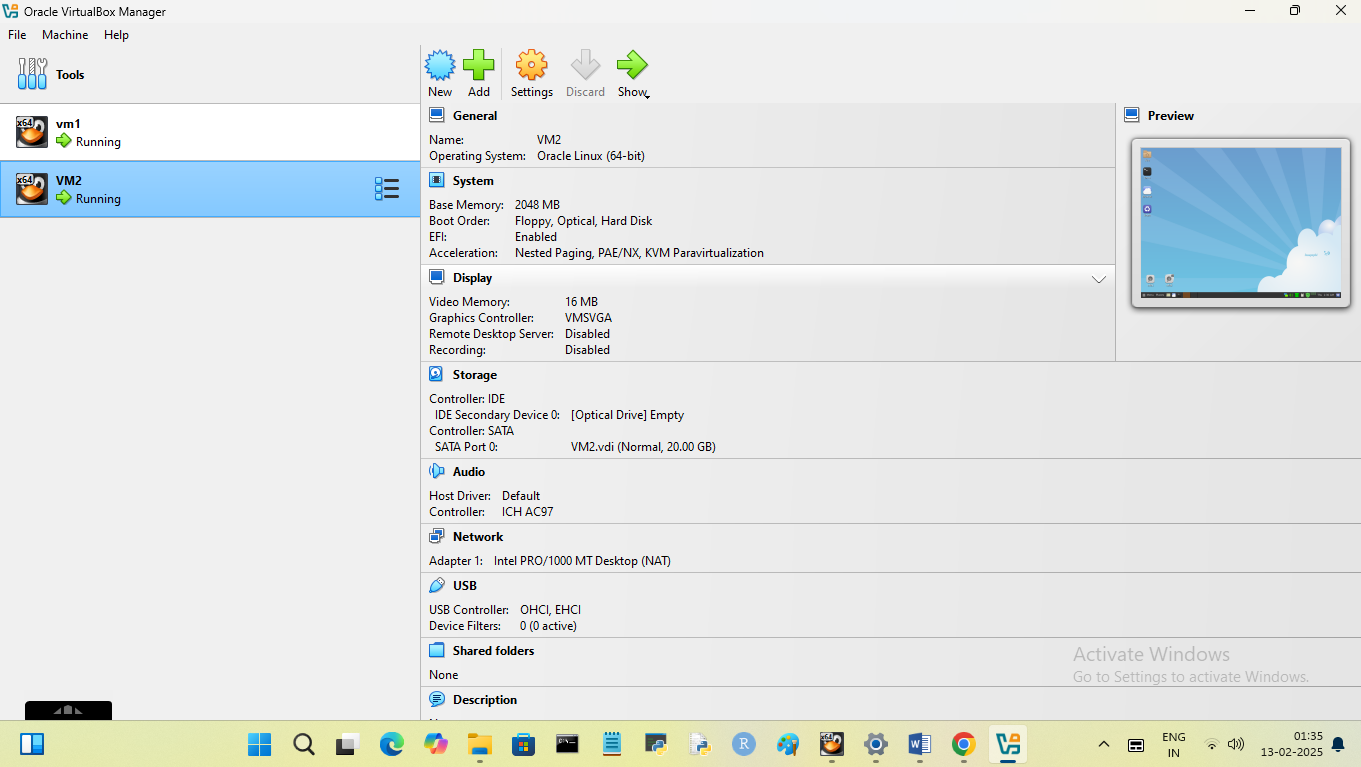
****

****

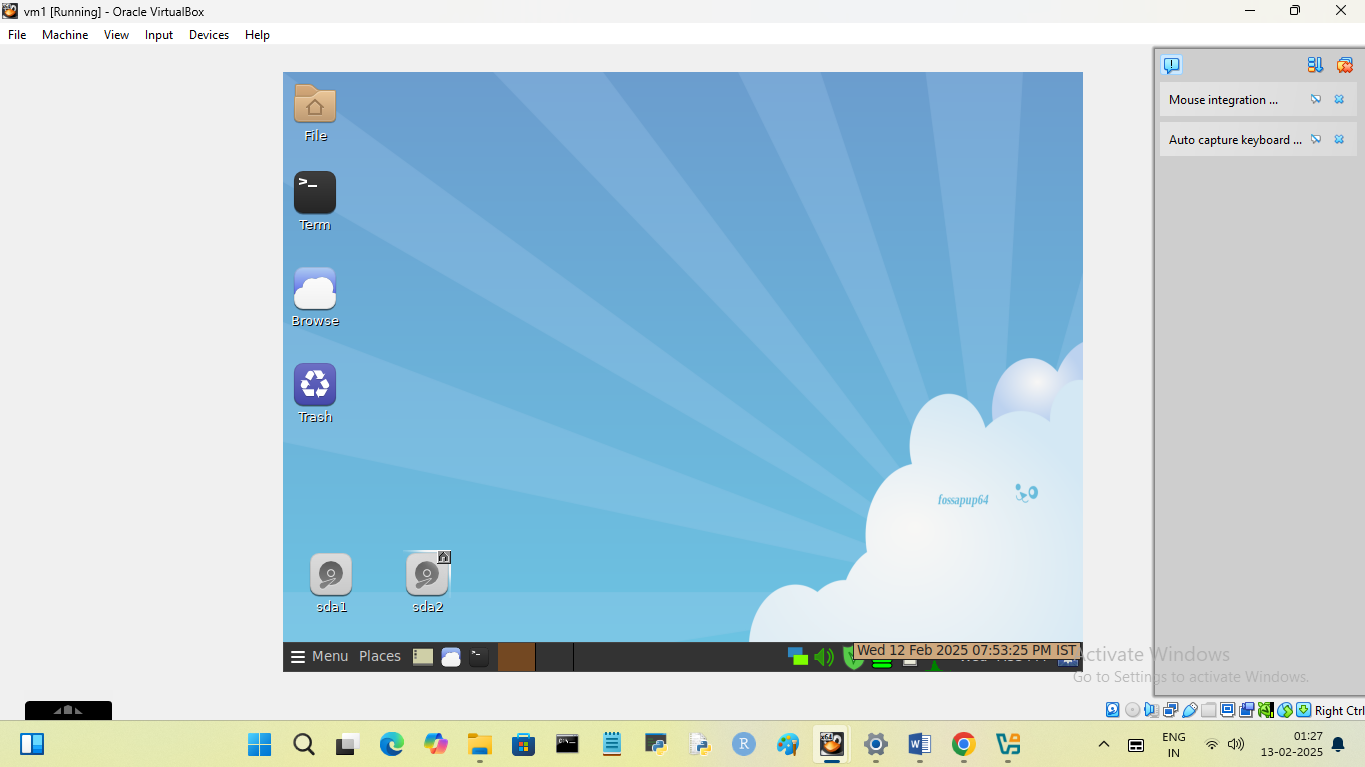
Click start to run VM

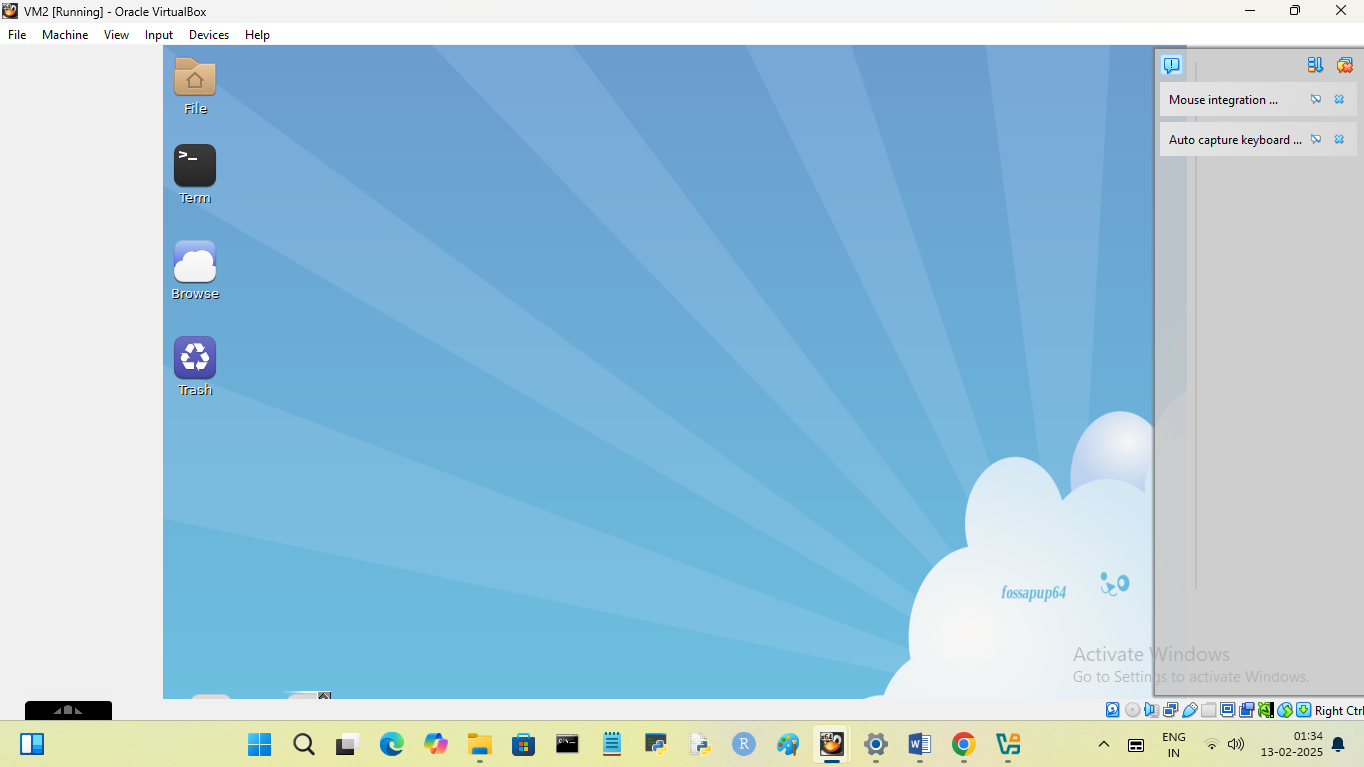
****

****

****

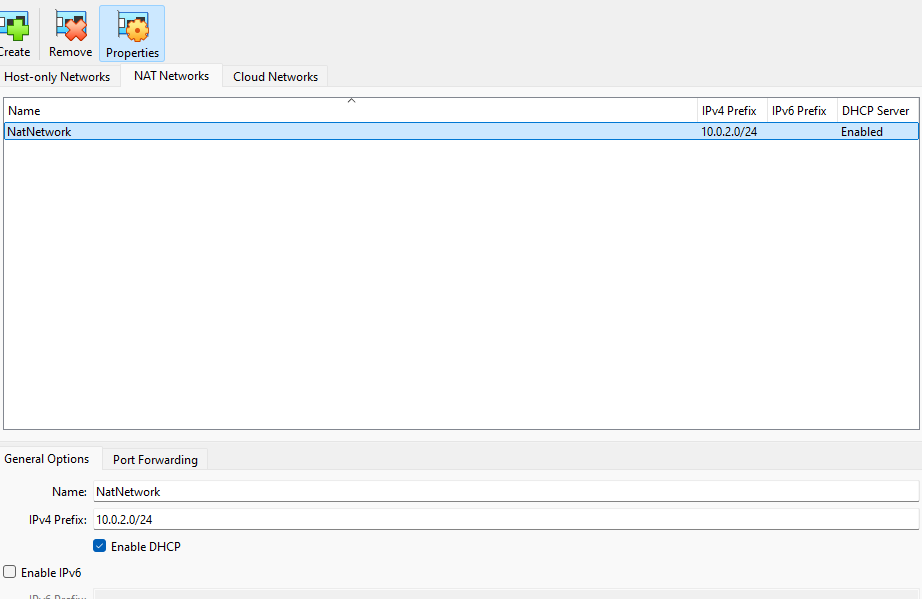
Change Desktop background and icon settings as required

****

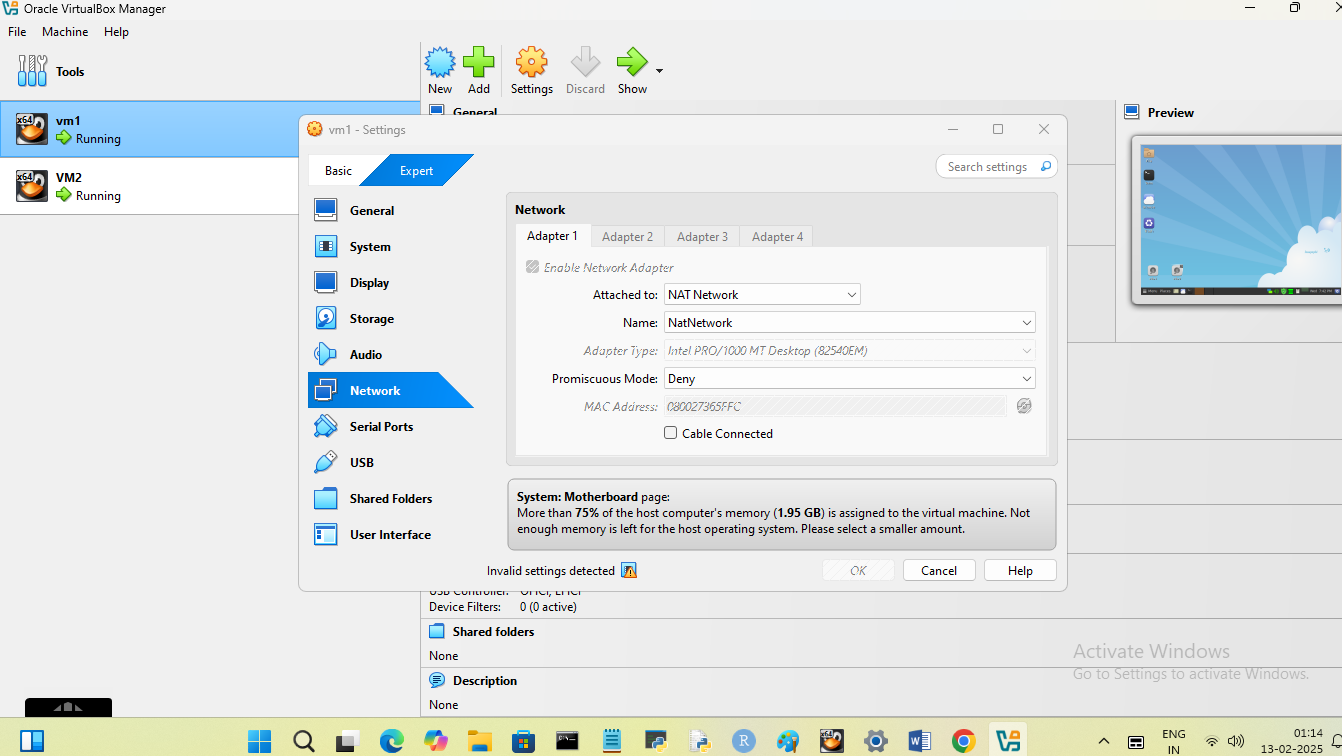
****

**Configuration of Network settings to connect the VMs:**

Create Nat network (Tools->Network->NAT Networks->Create)

****

**Connect VMS to NAT network (VM Settings->Network)**

****

Unable to connect VMs due to system memory error.

**Deployment of a simple microservice application across VMs:**

**Install Required Dependencies on VMS.**

Python

SQLITE3

**Create two microservices**

**User Service(User\_service.py)-on VM1:**

**from flask import Flask, request, jsonify**

**from flask\_sqlalchemy import SQLAlchemy**

**from werkzeug.security import generate\_password\_hash, check\_password\_hash**

**import os**

**app = Flask(\_\_name\_\_)**

**# Configure the database URI, ensure absolute path for consistency**

**db\_path = os.path.join(os.getcwd(), 'users.db')**

**app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///' + db\_path**

**app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False # Silence deprecation warning**

**# Initialize the database connection**

**db = SQLAlchemy(app)**

**# Define the User model**

**class User(db.Model):**

**id = db.Column(db.Integer, primary\_key=True)**

**username = db.Column(db.String(80), unique=True, nullable=False)**

**password = db.Column(db.String(120), nullable=False)**

**# Root route**

**@app.route('/')**

**def index():**

**return jsonify({'message': 'Welcome to the User API!'}), 200**

**# Endpoint to register a new user**

**@app.route('/register', methods=['POST'])**

**def register():**

**data = request.get\_json()**

**hashed\_password = generate\_password\_hash(data['password'], method='pbkdf2:sha256')**

**new\_user = User(username=data['username'], password=hashed\_password)**

**db.session.add(new\_user)**

**db.session.commit()**

**return jsonify({'message': 'User created'}), 201**

**# Endpoint to log in a user**

**@app.route('/login', methods=['POST'])**

**def login():**

**data = request.get\_json()**

**user = User.query.filter\_by(username=data['username']).first()**

**if not user or not check\_password\_hash(user.password, data['password']):**

**return jsonify({'message': 'Login failed'}), 401**

**return jsonify({'message': 'Logged in successfully'}), 200**

**if \_\_name\_\_ == '\_\_main\_\_':**

**try:**

**# Ensure tables are created within app context**

**with app.app\_context():**

**print(f"Creating database tables in: {db\_path}")**

**db.create\_all()**

**print("Database tables created successfully in users.db")**

**except Exception as e:**

**print(f"Error creating tables: {e}")**

**# Run the Flask application**

**app.run(debug=True, host='0.0.0.0')**

**Run:**

python3 User\_service.py

**Question Service(Question\_Service.py)- on VM2:**

**from flask import Flask, request, jsonify**

**from flask\_sqlalchemy import SQLAlchemy**

**import os**

**app = Flask(\_\_name\_\_)**

**# Configure the database URI, ensure absolute path for consistency**

**db\_path = os.path.join(os.getcwd(), 'questions.db')**

**app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///' + db\_path**

**app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False # Silence deprecation warning**

**# Initialize the database connection**

**db = SQLAlchemy(app)**

**# Define the Question model**

**class Question(db.Model):**

**id = db.Column(db.Integer, primary\_key=True)**

**text = db.Column(db.String(200), nullable=False)**

**exam\_year\_id = db.Column(db.Integer, nullable=False)**

**exam\_type\_id = db.Column(db.Integer, nullable=False)**

**# Root route**

**@app.route('/')**

**def index():**

**return jsonify({'message': 'Welcome to the Question API!'}), 200**

**# Endpoint to add a question**

**@app.route('/question', methods=['POST'])**

**def add\_question():**

**data = request.get\_json()**

**new\_question = Question(**

**text=data['text'],**

**exam\_year\_id=data['exam\_year\_id'],**

**exam\_type\_id=data['exam\_type\_id']**

**)**

**db.session.add(new\_question)**

**db.session.commit()**

**return jsonify({'message': 'Question added'}), 201**

**if \_\_name\_\_ == '\_\_main\_\_':**

**try:**

**# Ensure tables are created within app context**

**with app.app\_context():**

**print(f"Creating database tables in: {db\_path}")**

**db.create\_all()**

**print("Database tables created successfully in questions.db")**

**except Exception as e:**

**print(f"Error creating tables: {e}")**

**# Run the Flask application**

**app.run(debug=True, host='0.0.0.0')**

**Run:**

python3 Question\_service.py

Test the Microservice:

From **any VM** or **Host Machine**, run:

curl http://192.168.56.101:5000/users

curl http://192.168.56.102:5001/question