To implement the cloud simulation practical in MATLAB, follow these steps:

**1. Open MATLAB**

First, open the MATLAB software on your computer. You will use MATLAB’s integrated development environment (IDE) to enter and run the code.

**2. Create a New Script**

* Click on the **New Script** button in MATLAB to create a new file. This will allow you to write the code and save it.
* Alternatively, you can go to the **Home** tab, click **New**, and select **Script**.

**3. Write the Code**

* Copy and paste the code provided into the script editor.
* Make sure you include the entire code that models the cloud computing environment and the task allocation process.

Here is the summary of the code sections you need to include:

1. **Modeling Resources (VMs and Tasks)**:
   * This section defines the number of virtual machines (VMs) and the number of tasks with their resource requirements.
2. **Task Allocation (Cloud Resource Scheduling)**:
   * This is where tasks are assigned to VMs based on resource availability using the First-Fit algorithm.
3. **Visualizing the Allocation**:
   * This part visualizes how tasks are distributed across VMs.

**4. Save the Script**

* Once you've entered the code, save the script. You can save it with a name like cloud\_simulation.m.
* In MATLAB, save the script by going to **File > Save**, or use the keyboard shortcut Ctrl + S.

**5. Run the Script**

* To run the script, click the **Run** button in the script editor, or type the name of the script in the command window and press Enter.
* For example, if you saved the script as cloud\_simulation.m, type cloud\_simulation in the command window and press Enter.

**6. Observe the Output**

* The output of the script will show:
  + The number of tasks allocated to each VM.
  + A bar chart displaying the VM load distribution.
* The task allocation will be shown in the MATLAB figure window.

**7. Debugging (If Needed)**

If you encounter any errors or unexpected behavior:

* Check the code for any syntax errors or typos.
* Use disp or fprintf functions to display intermediate values (such as VM resources or task allocation) to help you debug.

**8. Enhancements and Experimentation**

After successfully running the simulation, you can experiment with:

* Adjusting the number of VMs (num\_vms) or tasks (num\_tasks).
* Changing the task resource requirements and VM capabilities.
* Trying different scheduling algorithms (e.g., **Best-Fit**, **Round-Robin**, etc.).
* Adding more visualizations to monitor resource usage over time.

**Example of Code Steps in MATLAB:**

**1. Modeling the Cloud Resources**

% Number of VMs (Cloud Resources)

num\_vms = 5;

% Resources available in each VM (e.g., CPU and Memory)

vm\_resources = [10, 20; % VM1: 10 CPU cores, 20 GB memory

15, 30; % VM2

8, 15; % VM3

20, 25; % VM4

12, 18]; % VM5

% Number of Tasks (Jobs)

num\_tasks = 10;

% Task resource requirements (e.g., CPU and Memory for each task)

task\_requirements = [4, 5; % Task 1: 4 CPU cores, 5 GB memory

8, 10; % Task 2

6, 8; % Task 3

7, 6; % Task 4

5, 4; % Task 5

12, 10; % Task 6

4, 7; % Task 7

3, 6; % Task 8

10, 15; % Task 9

5, 3]; % Task 10

**2. Simulating Task Allocation**

% Initialize VM allocation (1 for allocated, 0 for not)

vm\_allocation = zeros(num\_tasks, num\_vms);

% Initialize VM load (how many resources are used by each VM)

vm\_load = zeros(num\_vms, 1);

% Simple First-Fit Task Allocation

for task = 1:num\_tasks

for vm = 1:num\_vms

if vm\_resources(vm, 1) >= task\_requirements(task, 1) && vm\_resources(vm, 2) >= task\_requirements(task, 2)

% If VM has enough resources, allocate the task to the VM

vm\_allocation(task, vm) = 1;

vm\_load(vm) = vm\_load(vm) + 1; % Increase VM load

% Decrease VM available resources

vm\_resources(vm, 1) = vm\_resources(vm, 1) - task\_requirements(task, 1); % CPU usage

vm\_resources(vm, 2) = vm\_resources(vm, 2) - task\_requirements(task, 2); % Memory usage

break;

end

end

end

**3. Visualizing the Task Allocation**

% Visualize the allocation

figure;

bar(vm\_load);

title('VM Load Distribution');

xlabel('Virtual Machines');

ylabel('Number of Tasks Allocated');

xticks(1:num\_vms);

xticklabels({'VM1', 'VM2', 'VM3', 'VM4', 'VM5'});

grid on;

**9. Optional: Make It More Interactive**

You can add user input sections to allow users to customize the number of VMs, tasks, or resources.

num\_vms = input('Enter the number of virtual machines: ');

num\_tasks = input('Enter the number of tasks: ');

**Final Notes:**

* **Experimentation**: You can experiment by changing task/resource parameters or scheduling algorithms.
* **Visualization**: You can add more visualizations, such as a dynamic display of resource usage over time or more detailed plots for each VM.
* **Enhancements**: Explore cloud concepts like load balancing and resource optimization algorithms to enhance the simulation.