

Cloud Computing Experiment 1

Understanding Virtualization by installing Virtual Box and creating VM(Linux) for a React Application

1. Objective:

The goal of this task is to explore the concept of virtualization by installing VirtualBox and setting up a Virtual Machine (VM) with a Linux operating system. The VM will be utilized to host and deploy a React application, offering practical experience in creating and managing virtual environments for software development and deployment.

2. Background:

Virtualization enables multiple operating systems to operate on a single physical machine by creating virtual versions of resources such as servers. A Virtual Machine (VM) is a software-based emulation of a computer, allowing it to run an operating system and applications independently. Oracle VM VirtualBox, a type 2 hypervisor, manages these VMs. In this experiment, a Linux-based VM will be configured to host a React application. React, a JavaScript library, is used for building dynamic user interfaces. Networking within the VM facilitates internet access and hosting of the React app.

3. Tools and Services:

- VM VirtualBox
- Ubuntu ISO image
- Node.js and npm
- React application code

4. Experiment Setup:

1. Install virtualbox
 - a. Download Oracle VM VirtualBox from the [official website](#)
 - b. Follow the installation steps to set up VirtualBox on your host machine.
2. Download Linux ISO
 - a. Download the Ubuntu ISO image from the [official website](#).

5. Execution:

1. Create a new Ubuntu VM
 - a. Open the installed virtualBox and click on "New".

- b. Name the virtual machine and select the ISO file downloaded.

Create Virtual Machine

Virtual machine Name and Operating System

Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.

Name: ✓

Folder:

ISO Image:

Edition:

Type:

Version:

☐ Skip Unattended Installation

Detected OS type: Ubuntu (64-bit). This OS type can be installed unattendedly. The install will start after this wizard is closed.

Help Expert Mode Back Next Cancel

- c. The type and version will be automatically detected.
- d. Click on next.
- e. Provide a username and password for VM.

Create Virtual Machine

Unattended Guest OS Install Setup

You can configure the unattended guest OS install by modifying username, password, and hostname. Additionally you can enable guest additions install. For Microsoft Windows guests it is possible to provide a product key.

Username and Password

Username: ✓

Password:

Repeat Password:

Additional Options

Product Key:

Hostname: ✓

Domain Name:

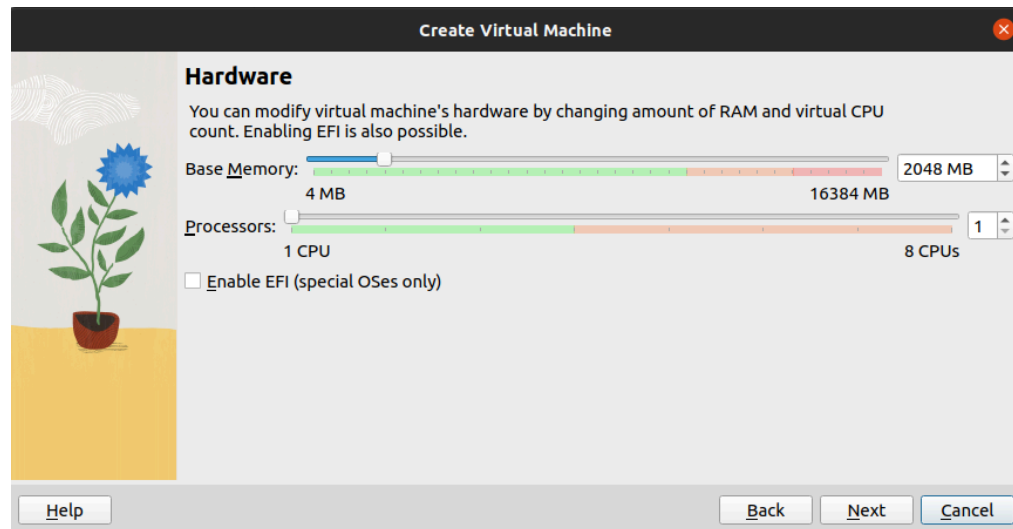
☐ Install in Background

☐ Guest Additions

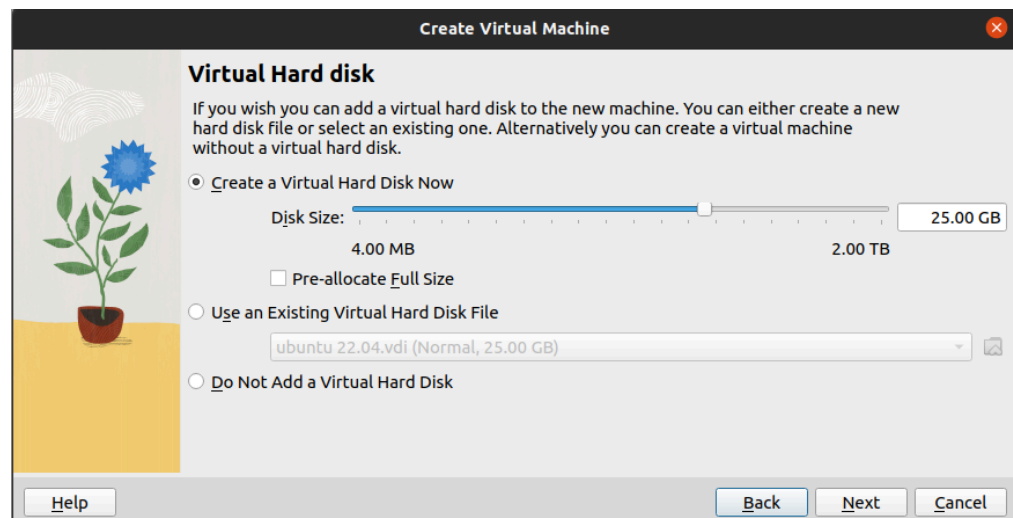
Guest Additions ISO:

Help Back Next Cancel

- f. Allocate the required memory and processor for the VM.

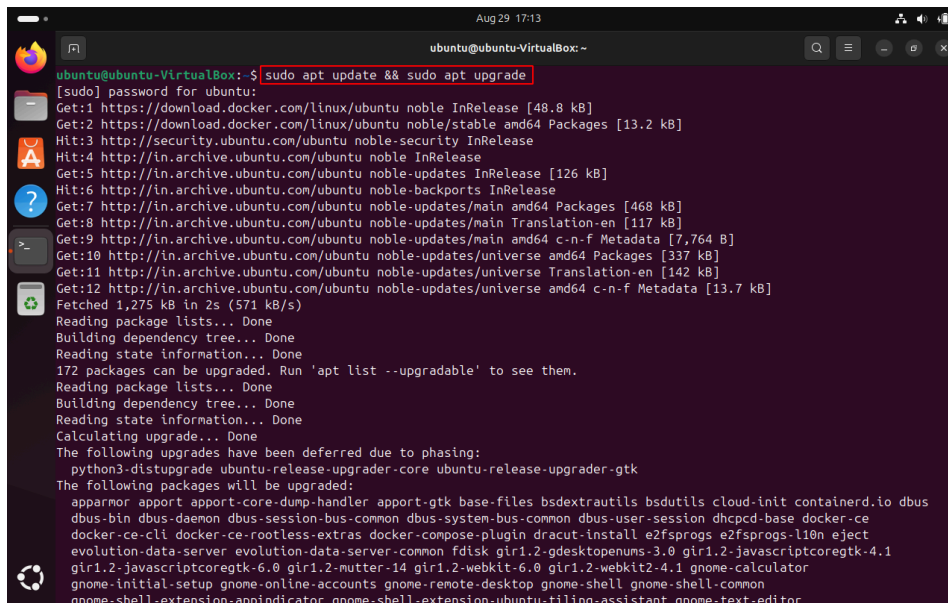


- g. Provide a suitable hard disk size for the VM



- h. Click Next and then Finish button to create the Ubuntu VM.
2. Setup Linux environment and install the dependencies.
- a. Once the VM is created, boot into the VM.

- b. Open the terminal and update the package manager.

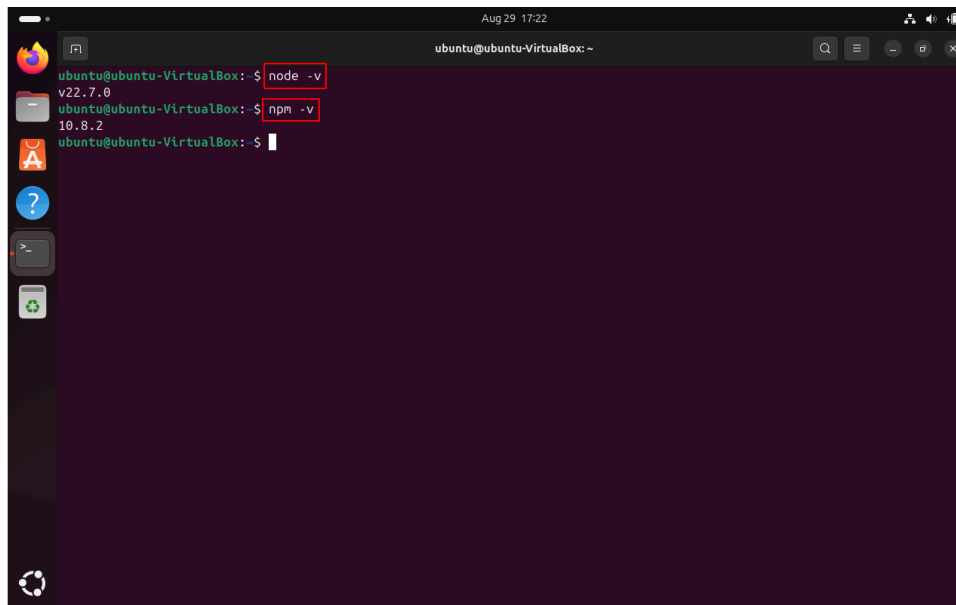


```
ubuntu@ubuntu-VirtualBox: ~  
ubuntu@ubuntu-VirtualBox:~$ sudo apt update && sudo apt upgrade  
[sudo] password for ubuntu:  
Get:1 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]  
Get:2 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [13.2 kB]  
Hit:3 http://security.ubuntu.com/ubuntu noble-security InRelease  
Hit:4 http://in.archive.ubuntu.com/ubuntu noble InRelease  
Get:5 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
Hit:6 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease  
Get:7 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [460 kB]  
Get:8 http://in.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [117 kB]  
Get:9 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [7,764 B]  
Get:10 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [337 kB]  
Get:11 http://in.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [142 kB]  
Get:12 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [13.7 kB]  
Fetched 1,275 kB in 2s (571 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
172 packages can be upgraded. Run 'apt list --upgradable' to see them.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Calculating upgrade... Done  
The following upgrades have been deferred due to phasing:  
python3-distupgrade ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk  
The following packages will be upgraded:  
apparmor apport apport-core-dump-handler apport-gtk base-files bsdxtrutils bsdutils cloud-init containerd.io dbus  
dbus-bin dbus-daemon dbus-session-bus-common dbus-system-bus-common dbus-user-session dhcpcd-base docker-ce  
docker-ce-cli docker-ce-rootless-extras docker-compose-plugin dracut-install e2fsprogs e2fsprogs-l10n eject  
evolution-data-server evolution-data-server-common fdisk gir1.2-gdesktopenums-3.0 gir1.2-javascriptcoregtk-4.1  
gir1.2-javascriptcoregtk-6.0 gir1.2-mutter-14 gir1.2-webkit-6.0 gir1.2-webkit2-4.1 gnome-calculator  
gnome-initial-setup gnome-online-accounts gnome-remote-desktop gnome-shell gnome-shell-common  
gnome-shell-extension-appindicator gnome-shell-extension-ubuntu-tiling-assistant gnome-text-editor
```

- c. Install npm and Node.js.

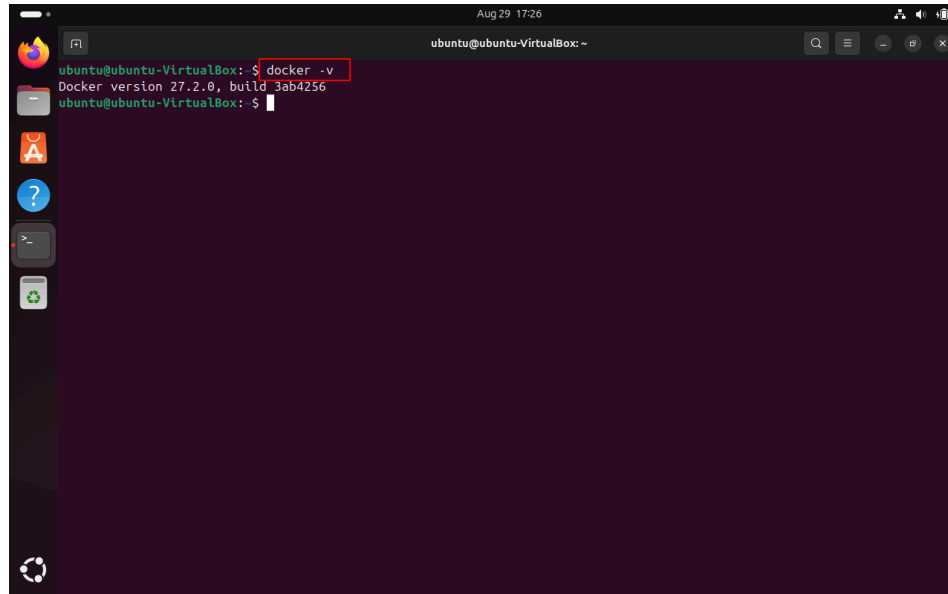
<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

- d. Verify the installation using these commands:



```
ubuntu@ubuntu-VirtualBox: ~  
ubuntu@ubuntu-VirtualBox:~$ node -v  
v22.7.0  
ubuntu@ubuntu-VirtualBox:~$ npm -v  
10.8.2  
ubuntu@ubuntu-VirtualBox:~$
```

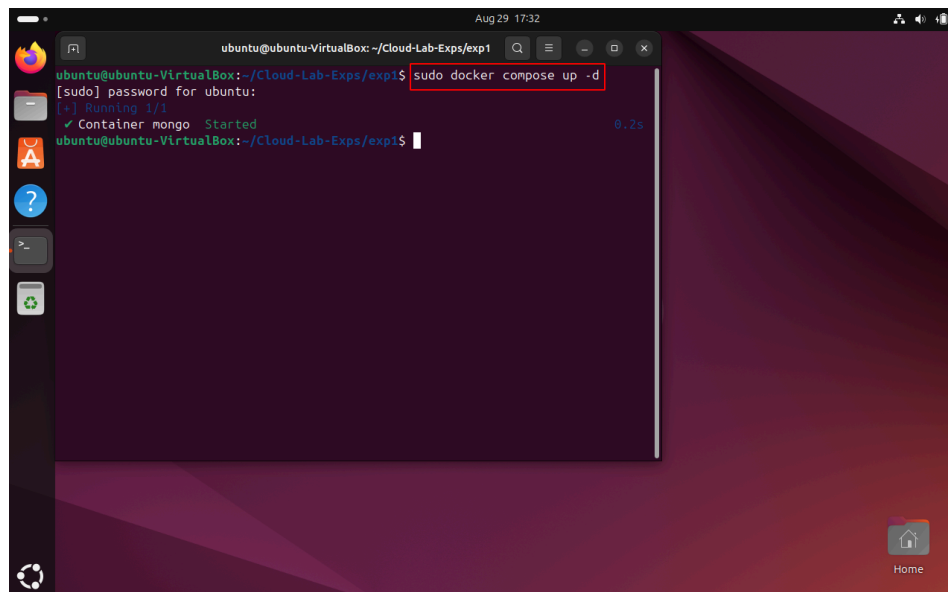
- e. Install docker desktop from official site [documentation](#) and verify the installation using this command

A terminal window titled 'ubuntu@ubuntu-VirtualBox: ~' with a dark purple background. The command 'docker -v' is entered and highlighted with a red box. The output is 'Docker version 27.2.0, build 3ab4256'.

```
ubuntu@ubuntu-VirtualBox: ~  
$ docker -v  
Docker version 27.2.0, build 3ab4256  
ubuntu@ubuntu-VirtualBox: ~
```

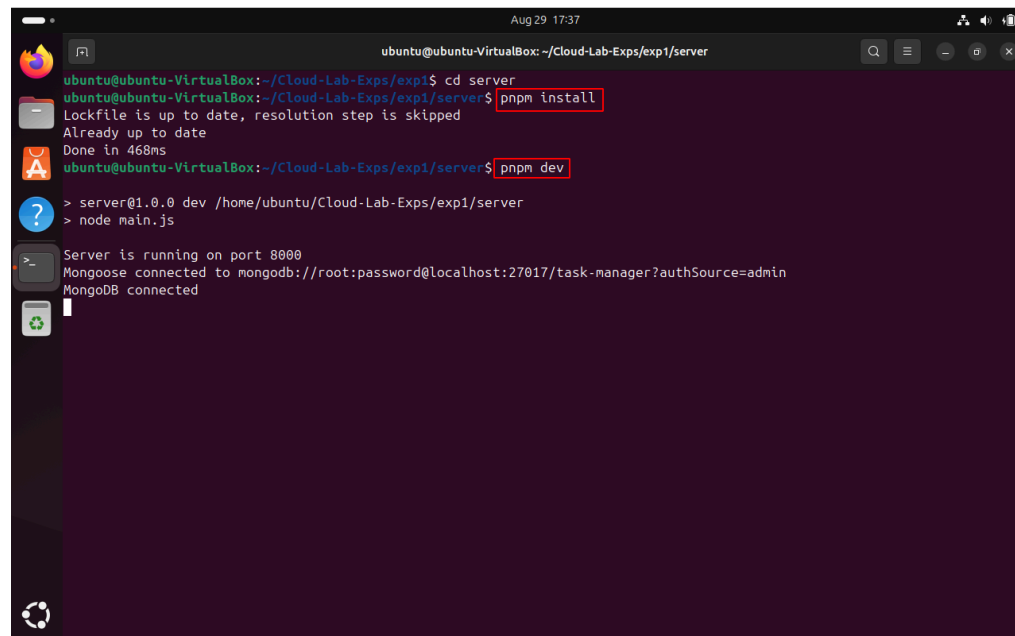
3. Setup the React project

- Clone this [repository](#) from Github using the git clone command.
- Initialize the mongoDB using the docker command.

A terminal window titled 'ubuntu@ubuntu-VirtualBox: ~/Cloud-Lab-Exps/exp1' with a dark purple background. The command 'sudo docker compose up -d' is entered and highlighted with a red box. The output shows '[sudo] password for ubuntu:', '[+] Running 1/1', and 'Container mongo Started' with a duration of '0.2s'.

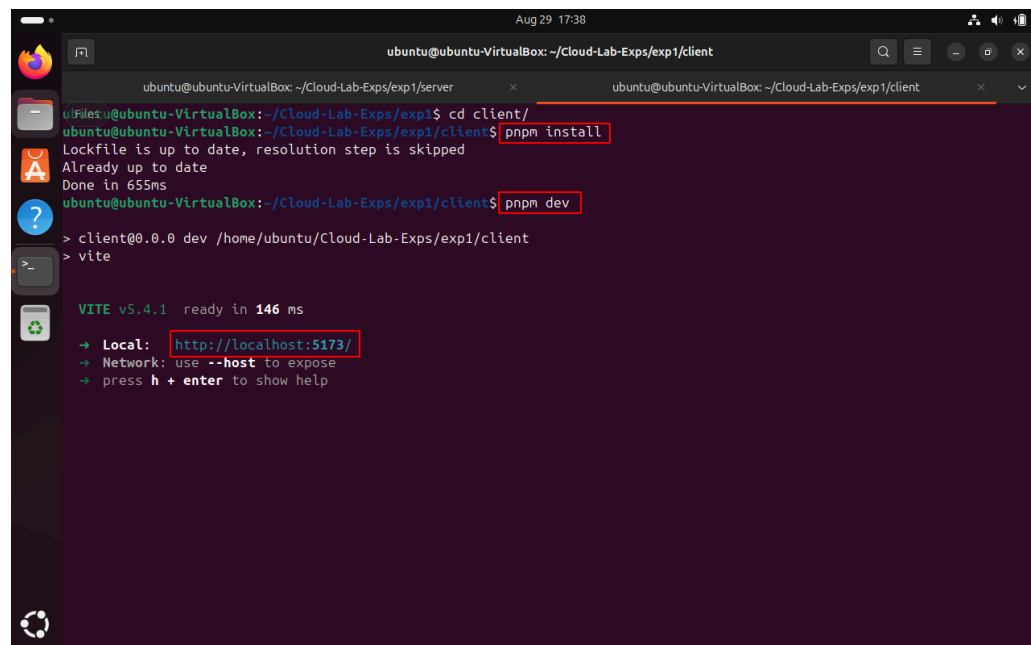
```
ubuntu@ubuntu-VirtualBox: ~/Cloud-Lab-Exps/exp1  
$ sudo docker compose up -d  
[sudo] password for ubuntu:  
[+] Running 1/1  
Container mongo Started 0.2s  
ubuntu@ubuntu-VirtualBox: ~/Cloud-Lab-Exps/exp1$
```

- c. Navigate to the server folder and install the dependencies and start the server.

A terminal window titled 'ubuntu@ubuntu-VirtualBox: ~/Cloud-Lab-Exps/exp1/server' showing the process of installing dependencies and starting a server. The user navigates to the 'server' directory and runs 'pnpm install', which completes successfully. Then, they run 'pnpm dev', which starts the server on port 8000 and connects to MongoDB. The terminal output includes: 'ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1\$ cd server', 'ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/server\$ pnpm install', 'Lockfile is up to date, resolution step is skipped', 'Already up to date', 'Done in 468ms', 'ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/server\$ pnpm dev', '> server@1.0.0 dev /home/ubuntu/Cloud-Lab-Exps/exp1/server', '> node main.js', 'Server is running on port 8000', 'Mongoose connected to mongodb://root:password@localhost:27017/task-manager?authSource=admin', and 'MongoDB connected'.

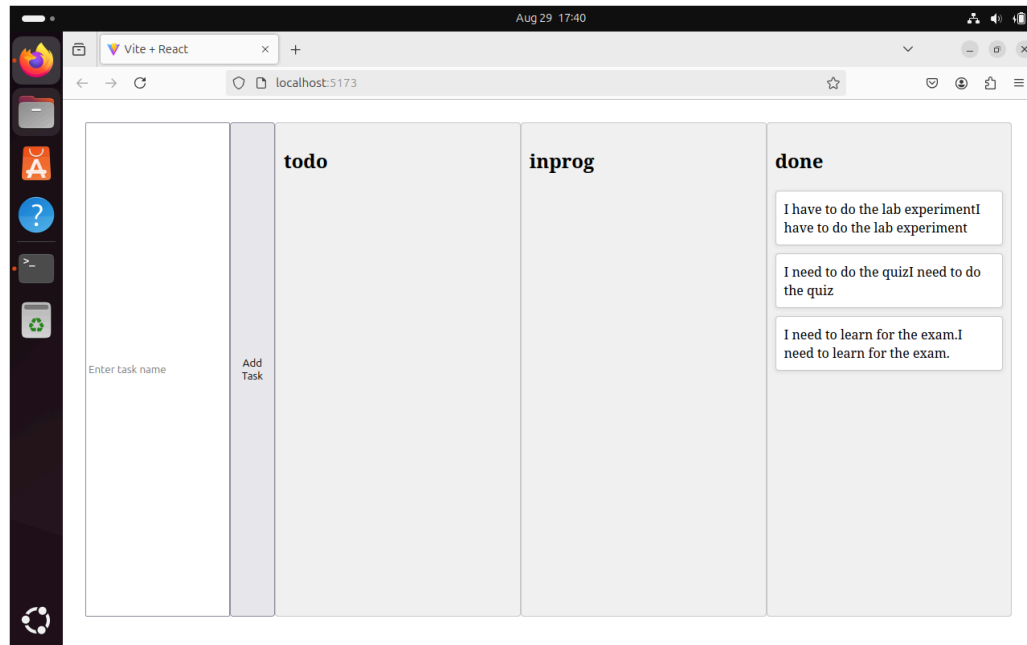
```
ubuntu@ubuntu-VirtualBox: ~/Cloud-Lab-Exps/exp1/server
ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1$ cd server
ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/server$ pnpm install
Lockfile is up to date, resolution step is skipped
Already up to date
Done in 468ms
ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/server$ pnpm dev
> server@1.0.0 dev /home/ubuntu/Cloud-Lab-Exps/exp1/server
> node main.js
Server is running on port 8000
Mongoose connected to mongodb://root:password@localhost:27017/task-manager?authSource=admin
MongoDB connected
```

- d. Navigate to the client folder and install dependencies and start the client application.

A terminal window titled 'ubuntu@ubuntu-VirtualBox: ~/Cloud-Lab-Exps/exp1/client' showing the process of installing dependencies and starting a client application. The user navigates to the 'client' directory and runs 'pnpm install', which completes successfully. Then, they run 'pnpm dev', which starts the client application using Vite. The terminal output includes: 'ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1\$ cd client/', 'ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/client\$ pnpm install', 'Lockfile is up to date, resolution step is skipped', 'Already up to date', 'Done in 655ms', 'ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/client\$ pnpm dev', '> client@0.0.0 dev /home/ubuntu/Cloud-Lab-Exps/exp1/client', '> vite', 'VITE v5.4.1 ready in 146 ms', '→ Local: http://localhost:5173/', '→ Network: use --host to expose', and '→ press h + enter to show help'.

```
ubuntu@ubuntu-VirtualBox: ~/Cloud-Lab-Exps/exp1/client
ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1$ cd client/
ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/client$ pnpm install
Lockfile is up to date, resolution step is skipped
Already up to date
Done in 655ms
ubuntu@ubuntu-VirtualBox:~/Cloud-Lab-Exps/exp1/client$ pnpm dev
> client@0.0.0 dev /home/ubuntu/Cloud-Lab-Exps/exp1/client
> vite
VITE v5.4.1 ready in 146 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

- e. Now the react application is running on port 5173 of the VM.



4. Once done, save the VM state or shut it down through virtualbox.

6. Observations:

Observed that VirtualBox and Linux VM are set up, Node.js and npm work, the React application runs with code changes, and the app is accessible from the browser, confirming proper networking.

7. Results:

Virtualization was understood by installing Virtual Box and running a React Application.