

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

Objective:

The objective of this task is to understand the concept of virtualization by installing VirtualBox and creating a Virtual Machine (VM) with a Linux operating system. The VM will then be used to host and deploy a React application, providing hands-on experience with setting up and managing virtual environments for software development and deployment.

Prerequisites:

- Basic understanding of virtualization.
- Familiarity with Linux command line.
- Knowledge of React application development.

Theory:

Virtualization allows multiple operating systems to run on a single physical machine by creating virtual versions of resources like servers. A Virtual Machine (VM) is a software emulation of a computer, running an OS and applications in isolation. Oracle VM VirtualBox is a type 2 hypervisor that manages VMs. In this experiment, a Linux VM will be set up to host a React application. React is a JavaScript library for building dynamic user interfaces. Networking within the VM enables internet access and hosting the React app.

Materials and Equipment:

- VM VirtualBox
- Linux ISO image (e.g., Ubuntu)
- Node.js and npm
- React application code

Procedure:

1. Install VirtualBox :

- Have installed VirtualBox from the official website.



- Follow the installation prompts to set up VirtualBox on your host machine.

→ . Download Linux ISO :

Choose a Linux distribution (e.g., Ubuntu) and download the ISO image from the official website.



Select an image

Ubuntu is distributed on three types of images described below.

Desktop image

The desktop image allows you to try Ubuntu without changing your computer at all, and at your option to install it permanently later. This type of image is what most people will want to use. You will need at least 1024MiB of RAM to install from this image.

64-bit PC (AMD64) desktop image

Choose this if you have a computer based on the AMD64 or EM64T architecture (e.g., Athlon64, Opteron, EM64T Xeon, Core 2). Choose this if you are at all unsure.

Server install image

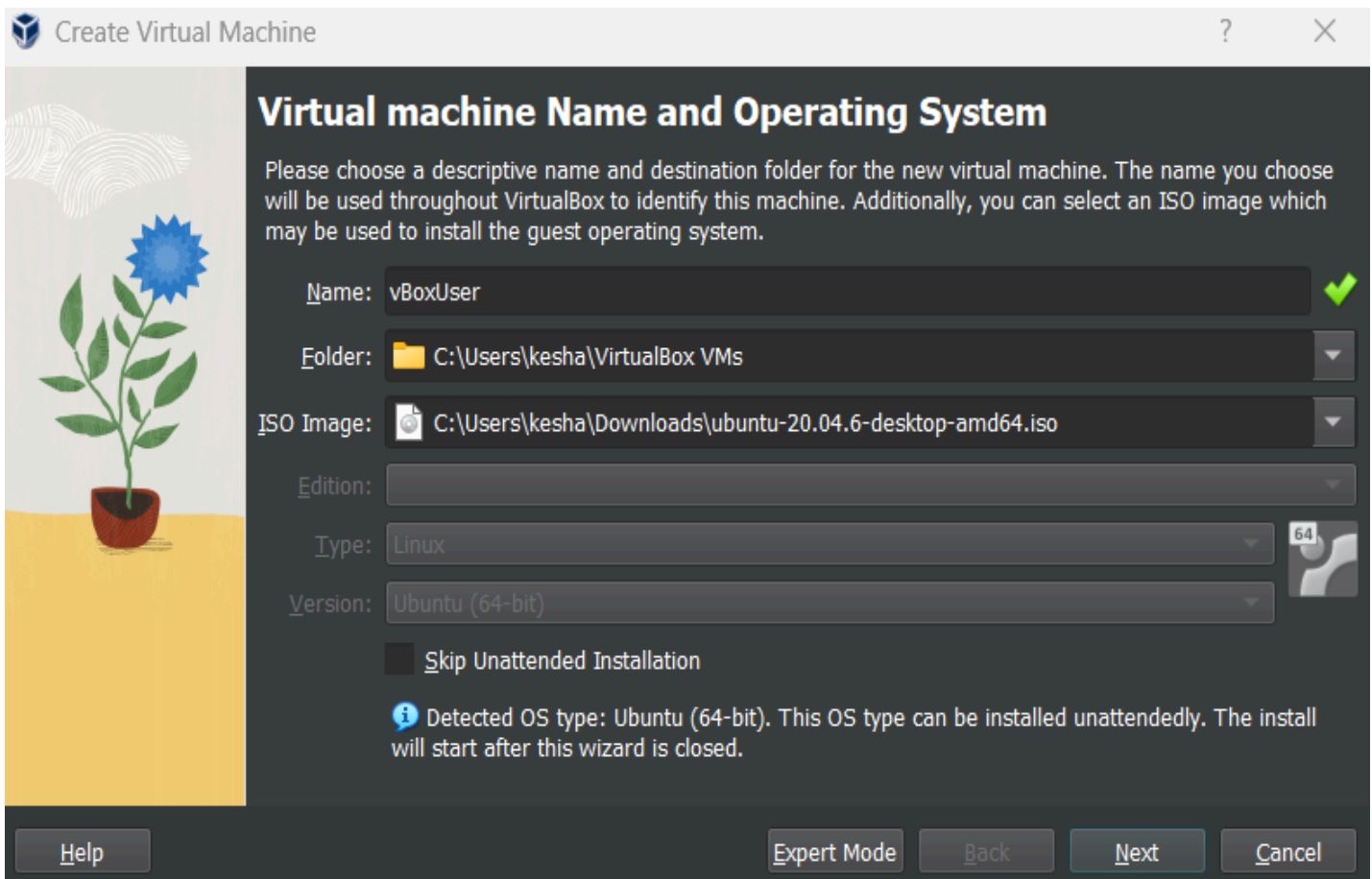
The server install image allows you to install Ubuntu permanently on a computer for use as a server. It will not install a graphical user interface.

64-bit PC (AMD64) server install image

Choose this if you have a computer based on the AMD64 or EM64T architecture (e.g., Athlon64, Opteron, EM64T Xeon, Core 2). Choose this if you are at all unsure.

→ 3 Create a New VM :

- Open VirtualBox and click on "New."
- Name the VM, select "Linux" as the type, and choose the appropriate version (e.g., Ubuntu 64 bit).
- Allocate memory (at least 2GB) and create a virtual hard disk (20GB or more).



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:

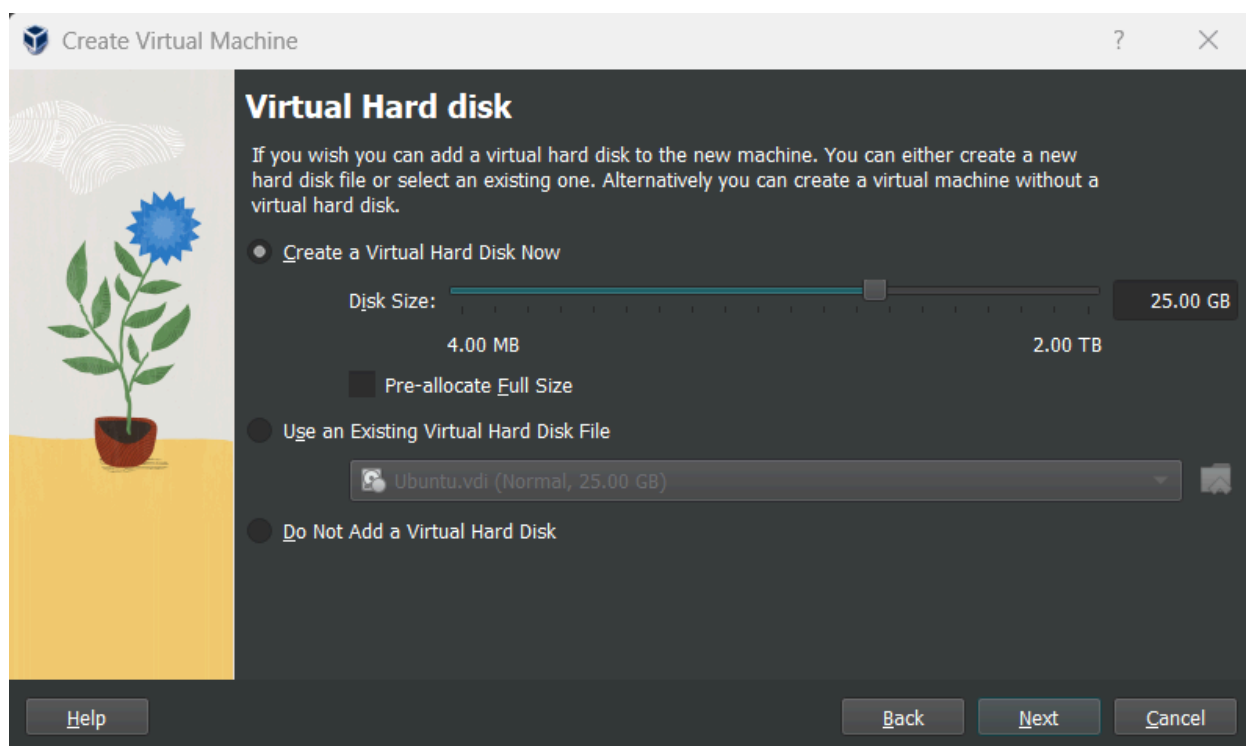
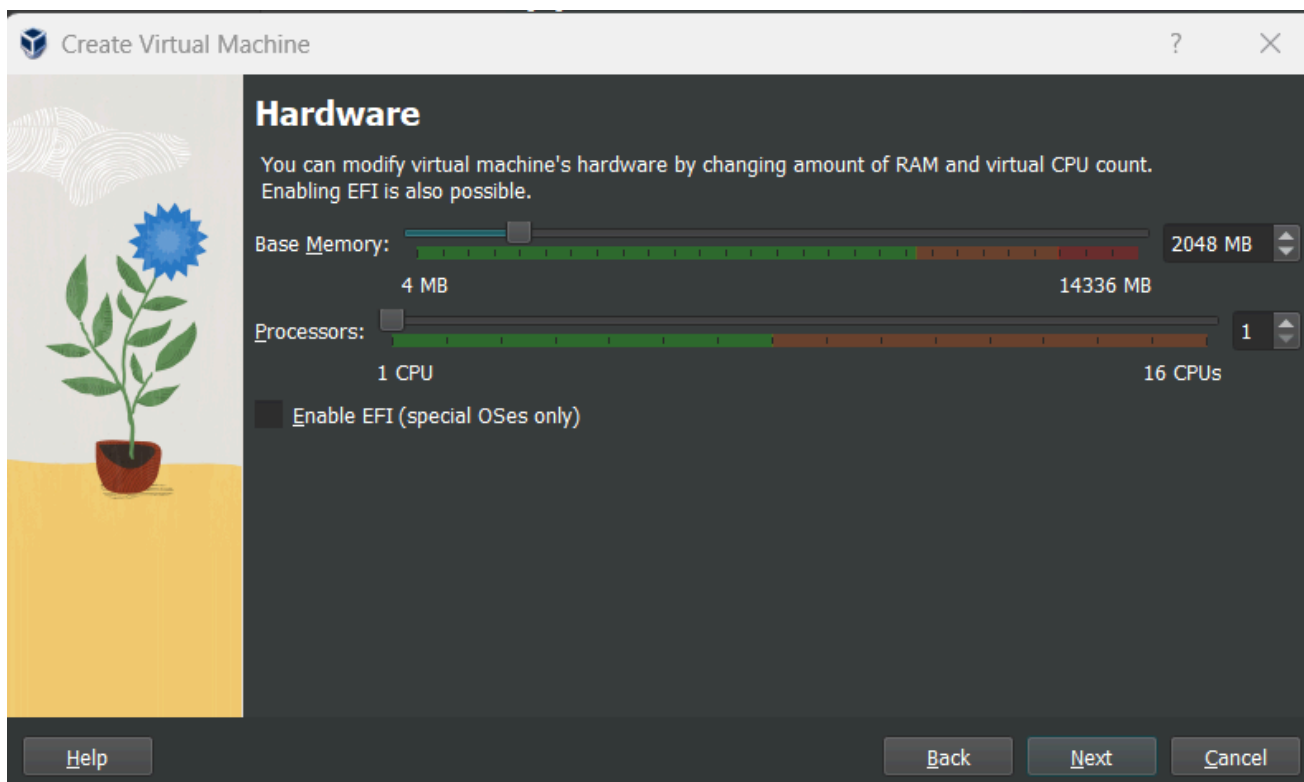
Type: 64


Version:

☐ Skip Unattended Installation

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

[Help](#) [Expert Mode](#) [Back](#) [Next](#) [Cancel](#)





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: vboxuser ✓

Password: •••••• ••••••

Repeat Password: •••••• ••••••

Additional Options


Product Key: #####-#####-#####-#####

Hostname: vBoxUser ✓

Domain Name: myguest.virtualbox.org


☐ Install in Background

Guest Additions

Guest Additions ISO:  C:\Program Files\Oracle\VirtualBox\VBBoxGuestAdditions.iso

Help

BackNextCancel



Create Virtual Machine

Summary

The following table summarizes the configuration you have chosen for the new virtual machine. When you are happy with the configuration press Finish to create the virtual machine. Alternatively you can go back and modify the configuration.

Machine Name and OS Type

Machine NamevBoxUser

Machine FolderC:/Users/kesha/VirtualBox VMs/vBoxUser

ISO ImageC:/Users/kesha/Downloads/ubuntu-20.04.6-desktop-amd64.iso

Guest OS TypeUbuntu (64-bit)

Skip Unattended Installfalse

Unattended Install

Usernamevboxuser

Product Keyfalse

Hostname/Domain NamevBoxUser.myguest.virtualbox.org

Install in Backgroundfalse

Install Guest Additionsfalse

Hardware

Base Memory2048

Processor(s)1

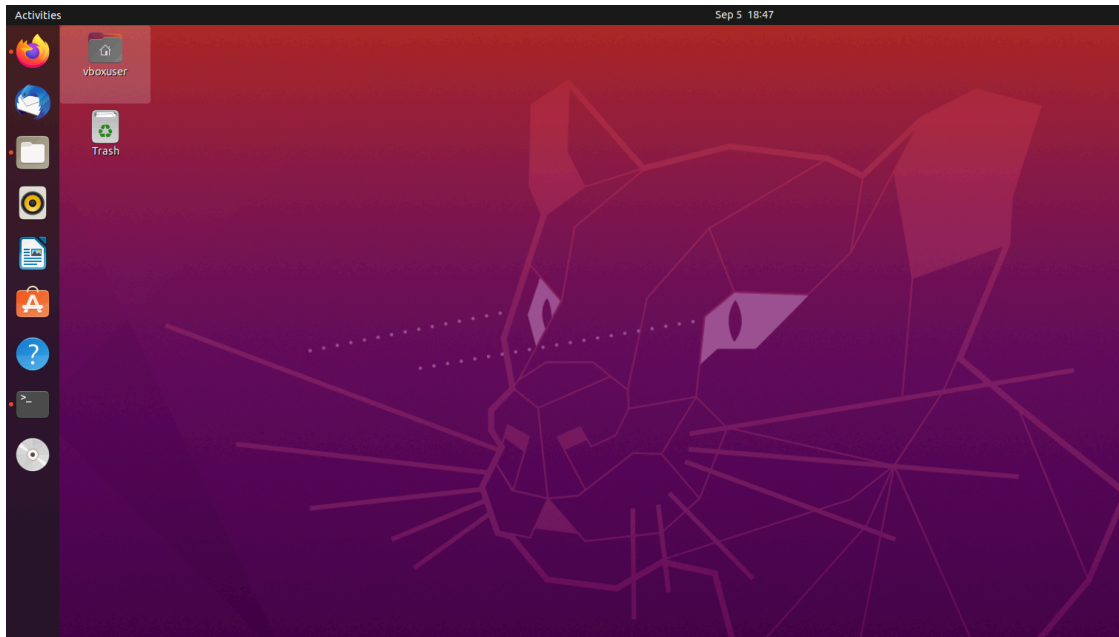
EFI Enablefalse

Help

BackFinishCancel

2. Set Up Linux Environment :

- Once Linux is installed, boot into the VM.
- Update the package manager (``sudo apt update``) and install any necessary updates (``sudo apt upgrade``).



3. Install Node.js and npm :

- In the Linux terminal, install Node.js and npm by running the following commands:

bash

sudo apt install nodejs

sudo apt install npm

- Verify the installation with ``node -v`` and ``npm -v``.

```
vboxuser@Ubuntu:~$ node -v
v20.17.0
vboxuser@Ubuntu:~$ npm -v
10.8.2
vboxuser@Ubuntu:~$
```

4. .Navigate to the project directory in the terminal
5. After installing all the docker containers and other dependencies , change the mongo version to 4.4 using any editor and save the changes. (in case any error is encountered due to CPU limitations)

```
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps$ cat exp1/docker-compose.yml
services:
  mongo:
    image: mongo:4.4
    container_name: mongo
    ports:
      - "27017:27017"
    volumes:
      - ./data:/data/db
      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=password
```

6. Start the docker container
- > first up the compose file in the place where the compose file is present using the command - **sudo docker compose up -d --build**
 - > **navigate to client folder** >> **npm i**
 - >> **npm run dev**

```

vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1$ cd client
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1/client$ npm run dev

> client@0.0.0 dev
> vite --host

VITE v5.4.1 ready in 765 ms

  → Local:   http://localhost:5173/
  → Network: http://10.0.2.15:5173/
  → press h + enter to show help
^C
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1/client$ cd ..
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1$ cd server

```

7. Follow the same steps for the server folder also

→ Navigate to server folder and follow the same steps

--> first up the compose file in the place where the compose file is present using the command - sudo docker compose up -d --build

--> navigate to server folder >> npm i

>> npm run dev

```

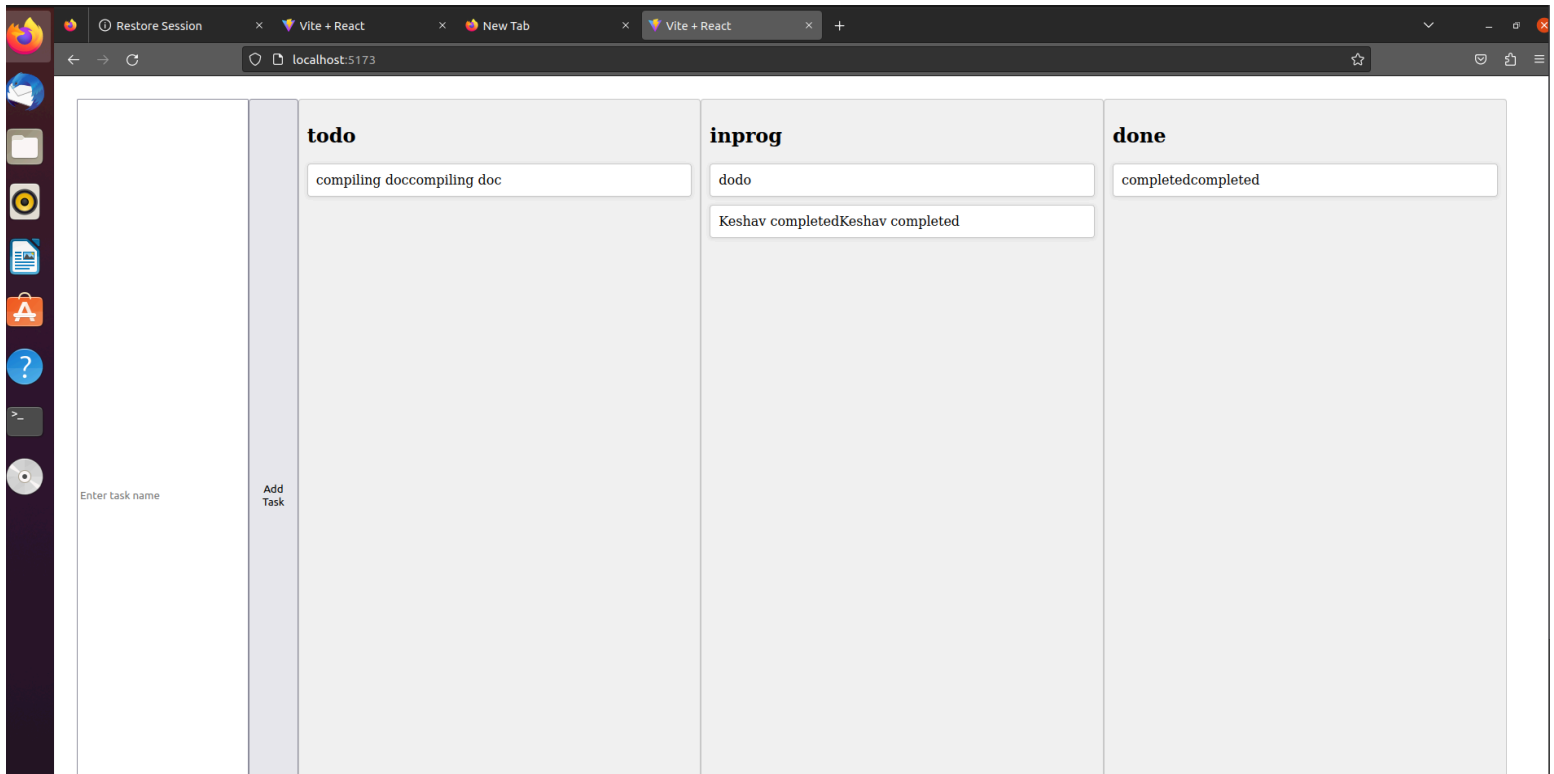
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1/server$ sudo docker compose up -d --build
empty compose file
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1/server$ cd ..
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1$ sudo docker compose up -d --build
[+] Running 2/2
  ✓ Network exp1_default    Created                                0.3s
  ✓ Container mongo         Started                               1.9s
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1$ cd server
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp1/server$ npm run dev

> server@1.0.0 dev
> node main.js

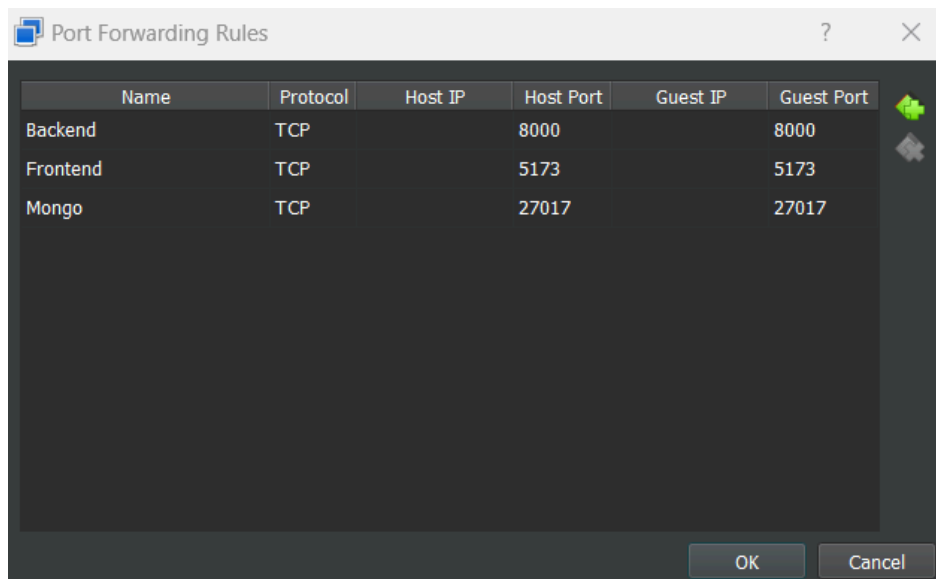
Server is running on port 8000
Mongoose connected to mongodb://root:password@localhost:27017/task-manager?authSource=admin
MongoDB connected

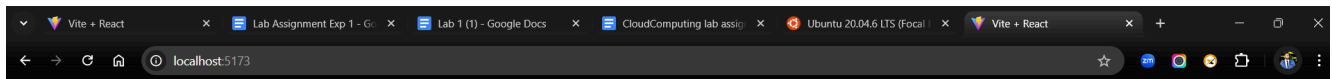
```

8. Open the application in your browser at the address



9. Enable port forwarding in advanced settings in the VM to open the app in the host machine .





	todo	inprog	done
Enter task name	<input type="text" value="compiling doccompiling doc"/>	<input type="text" value="dodo"/>	<input type="text" value="completedcompleted"/>
		<input type="text" value="Keshav completedKeshav completed"/>	

Observations:

It is seen that VirtualBox and Linux VM are set up, Node.js and npm work, the application runs, and the app is accessible from the browser and host machine, confirming proper networking and port forwarding.

Result:

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