Cloud Computing Laboratory (20CP322P)

Lab Report Submitted to

Pandit Deendayal Energy University, Gandhinagar

for

Bachelor of Technology

in

Computer Science & Engineering Department

Submitted by

Harsh Shah – 21BCP359
Semester: VI
Division: 6, Group: G11

Course Faculty

Dr. ABHINAV SHARMA

PANDIT DEENDAYAL ENERGY UNIVERSITY, GANDHINAGAR - 382007, India

May-2024

List of Practical

Exp. No.	Experiment Title	Date	Signature
1	Study of Various Virtualization Software such as VMware, QEMU, KVM, XEN Hypervisor and Oracle Virtual Box.	11-01-24	
2	Exploring Oracle VM VirtualBox to create the virtual Machines	18-01-24	
3	To Install and configure VMware Workstation Pro or Oracle Virtual Box for creating Virtual Machines, cloning, and deleting VMs.	01-02-24	
4	To Explore Docker Container	15-02-24	
5	Create two dockers with front-end and back-end applications and connect these containers.	14-03-24	
6	Create a customized docker image and perform the given instructions	28-03-44	
7	Create three containers and access these containers from each other by using IP address and container name.	04-04-24	
8	Re-Do the problem of Lab 5 using docker compose. Also do assign a static IP address to WordPress container using subnetting.	25-04-24	

PRACTICAL 1

Aim: Study of Various Virtualization Software such as VMware, QEMU, KVM, XEN Hypervisor and Oracle Virtual Box.

1. Compare their Virtualization type and methodologies.

Virtualization Types

a. Full Virtualization
- This technique emulates the complete hardware environment
- Allows any guest OS to run on the VM
- VMware and VirtualBox utilize Full Virtualization

b. Para Virtualization
- This technique modifies the guest OS to interact with the Hypervisor
- Offers better performance than full virtualization but requires compatibility between guest OS and Hypervisor
- Xen and KVM utilize Para Virtualization

c. Hardware assisted Virtualization
- This technique leverages special features built into modern processors to improve virtualization performance and security
- Features like translating guest OS memory addresses to physical memory addresses.
- VMware ESXi and Xen utilize hardware assisted virtualization.

Methodologies

a. VMware
- Uses a technique called hardware-assisted virtualization where it leverages features provided by modern CPUs (Intel VT-x or AMD-V) to improve the efficiency of virtualization

b. QEMU
- QEMU utilizes dynamic binary translation to emulate the guest CPU instructions
- Dynamic binary translation is particularly useful in scenarios where direct execution of guest instructions on the host is not feasible due to architectural differences. This process occurs dynamically at runtime, translating the binary code of guest instructions to equivalent instructions that can be executed on the host system

c. KVM
- KVM leverages hardware virtualization extensions (Intel VT-x or AMD-V) to provide a direct interface to the host's CPU for virtualization. It works as a kernel module and relies on the Linux kernel's scheduler and memory management.

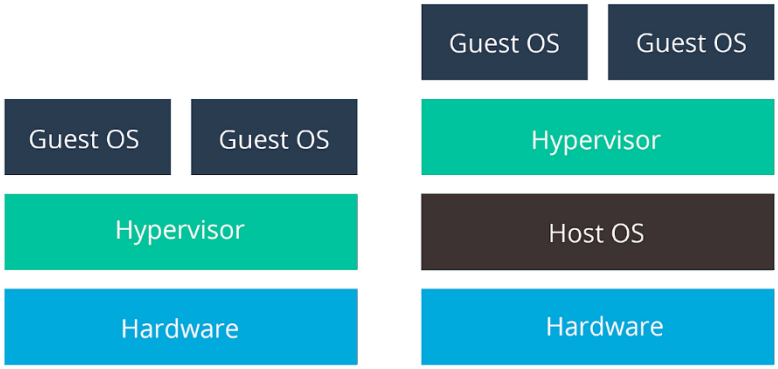
d. XEN Hypervisor
- Xen uses a microkernel design where the hypervisor runs as a separate, lightweight kernel (the "Xen hypervisor") above the hardware. It allows multiple operating systems to run

concurrently in isolated domains (known as Xen domains or VMs). Each domain has its own kernel, and the hypervisor manages their access to hardware resources.

e. Oracle VirtualBox

- VirtualBox runs as a user-space application on the host operating system. It uses a combination of dynamic binary translation and various device drivers to emulate hardware and provide a virtualized environment for guest operating systems

2. Identify architectural difference.



Software	**VMware**	**QEMU**	**KVM**	**Xen**	**Oracle VirtualBox**
Type	Type 1 (bare-metal) and Type 2 (hosted)	Type 1 (bare-metal) and Type 2 (hosted)	Type 2 (hosted)	Type 1 (bare-metal) and Type 2 (hosted)	Type 2 (hosted)
Focus	Enterprise virtualization, desktop virtualization, cloud computing	Machine emulation, basic virtualization	Open-source virtualization for Linux	Open-source cloud infrastructure, server virtualization	Free personal virtualization, educational tool
Pros	Robust features, high performance, advanced security	Open-source, highly flexible, good for emulation	Free, open-source, efficient for Linux	Open-source, widely used in cloud, robust features	Free, cross-platform, easy to use
Cons	Can be expensive, complex to manage	Limited features in hosted version, not ideal for high-performance workloads	Requires Linux kernel, not as user-friendly as some options	Less mature than some Type 1 hypervisors, may not be suitable for all workloads	Limited features compared to some enterprise solutions

3. List out various advantages, disadvantages and applications.

a. VM Ware

Advantages:

- Management Tools: Comprehensive management tools like vSphere, advanced features such as vMotion for live migration.
- Compatibility: Wide compatibility with various operating systems and Widely used in enterprise environments

Disadvantages:

- Cost Concerns: VMware can be expensive due to licensing fees, which might be challenging for smaller businesses.
- Limited Flexibility: VMware's proprietary nature can limit customization options compared to open-source alternatives.

Applications:

- Server Virtualization: VMware is extensively used for server virtualization, allowing multiple virtual machines (VMs) to run on a single physical server.
- Desktop Virtualization: VMware provides solutions for desktop virtualization, enabling organizations to centralize desktop management and delivery.
- Cloud Computing: VMware's virtualization technology is used in building and managing cloud infrastructures.

b. QEMU

Advantages:

- Emulation: Can emulate various hardware devices beyond virtualization, ideal for testing and development.
- Lightweight: Less resource-intensive than some other options, suitable for lower-powered environments.

Disadvantages:

- Speed Issues: QEMU may not be as fast as other options, especially in situations where a lot of resources are needed.
- Not User-Friendly: Setting up QEMU can be complex, especially for those who prefer graphical interfaces over command lines.

Applications:

- Emulation: QEMU is often used for hardware emulation, allowing software to run on platforms for which it was not originally designed.
- Cross-Platform Development: Developers use QEMU to test software on different architectures without the need for physical hardware.
- System Recovery: QEMU can be employed for system recovery and backup purposes.

c. KVM

Advantages:

- Built-in: Included in the Linux kernel, providing seamless integration and native performance.
- Open-source: Free and community-driven, making it accessible and customizable.

Disadvantages:

- Linux Focus: KVM works best with Linux, which may be a drawback for environments where Linux isn't the main operating system.
- Hard to Set Up: Getting KVM up and running can be complicated, especially for users unfamiliar with Linux.

Applications:

- Performance: KVM can provide high-performance virtualization by leveraging hardware support.
- Cloud Computing: KVM is widely used in cloud computing environments, providing a robust and efficient virtualization solution.
- Flexibility: Supports a wide range of guest operating systems.

d. XEN Hypervisor

Advantages:

- Type-1 hypervisor: Offers **high performance** and isolation for critical workloads.
- Security focus: Designed with security in mind, incorporating features like **memory paging** and security domains.

Disadvantages:

- Limited Graphics Support: Xen traditionally has **limited support** for **graphics-intensive applications**.
- Lack of USB Passthrough Support: Xen has faced **challenges** with **USB passthrough** support.

Applications:

- Server Consolidation: Xen is used for server consolidation, allowing **multiple virtual servers** to run on a single physical server.
- Cloud Hosting: Xen is utilized in many **cloud hosting platforms** to provide virtualized instances to users.
- Isolation: Xen's architecture provides **strong isolation** between virtual machines, enhancing security.

e. Oracle VirtualBox

Advantages:

- Free and easy to use: Ideal for individual users and smaller deployments.
- User-friendly interface: Simple and intuitive for beginners to set up and manage VMs.

Disadvantages:

- Performance Drawbacks: VirtualBox might not handle high-demand situations as well as some other options.
- Lacking Advanced Features: VirtualBox may lack some advanced features found in solutions like VMware, which could be a drawback for enterprise users.

Applications:

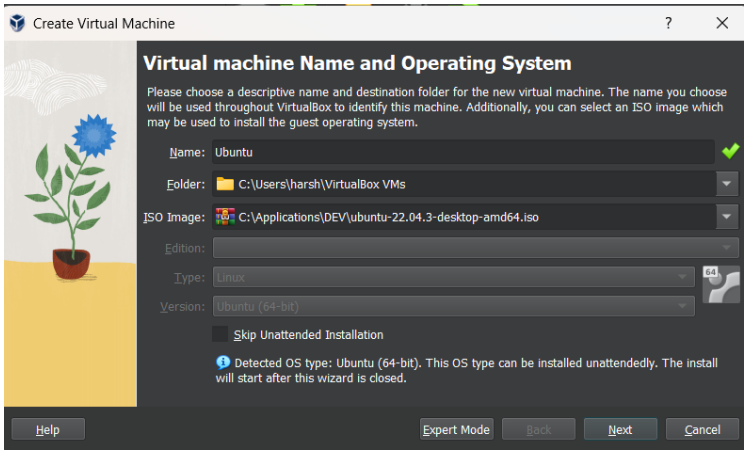
- Desktop Virtualization: VirtualBox is commonly used for desktop virtualization, enabling users to run multiple operating systems on a single physical machine.
- Development and Testing: VirtualBox is popular for development and testing environments, allowing developers to create and test applications on various operating systems.
- Education and Training: VirtualBox is used in educational settings to create virtual labs and facilitate hands-on training in different operating systems.

PRACTICAL 2

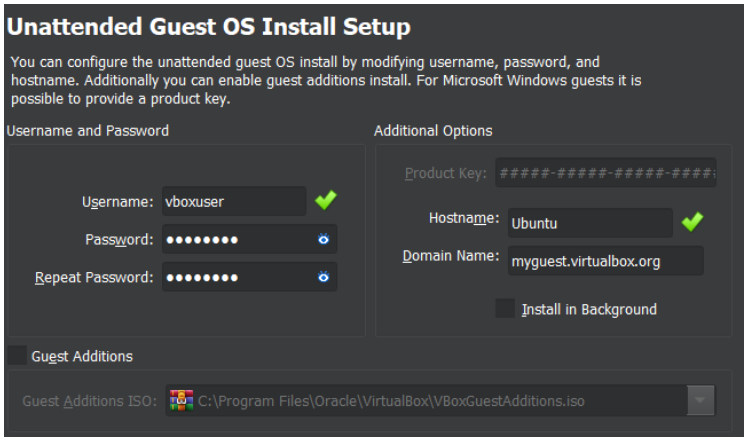
Aim: Exploring Oracle VM VirtualBox to create the virtual Machines

1. Create Two or more Virtual Machine and assign resources.

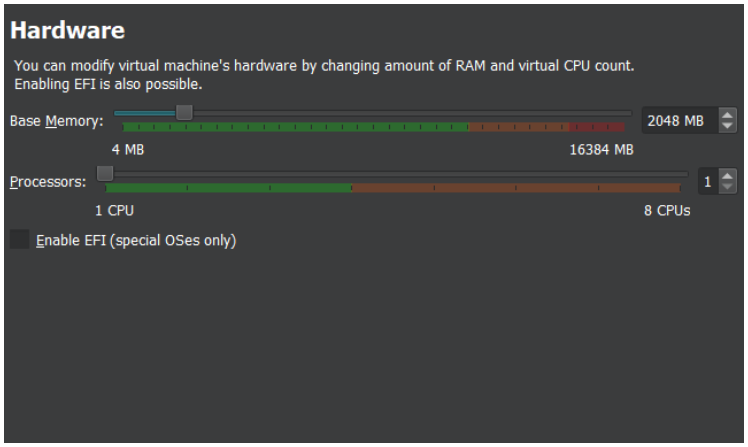
- Click on New button on Home Page. Choose file path of OS.



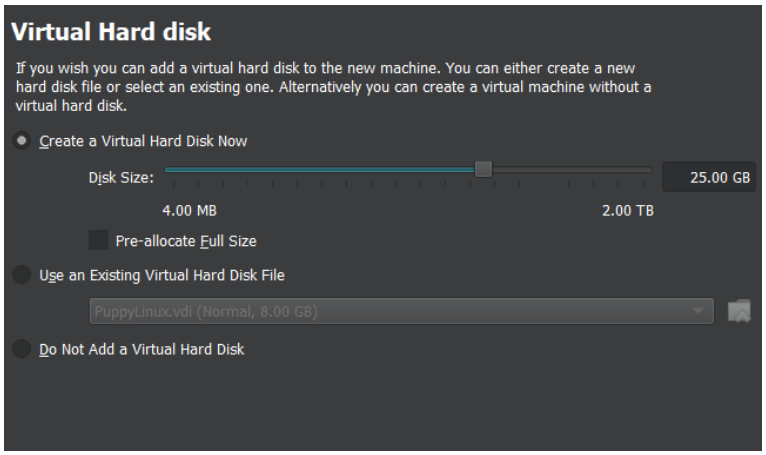
- Give User details



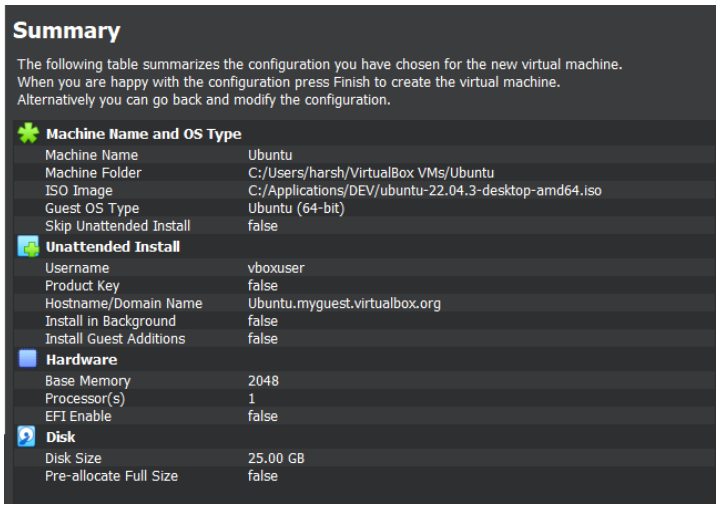
- Allocate Memory and Processors



- Allocate Disk Space

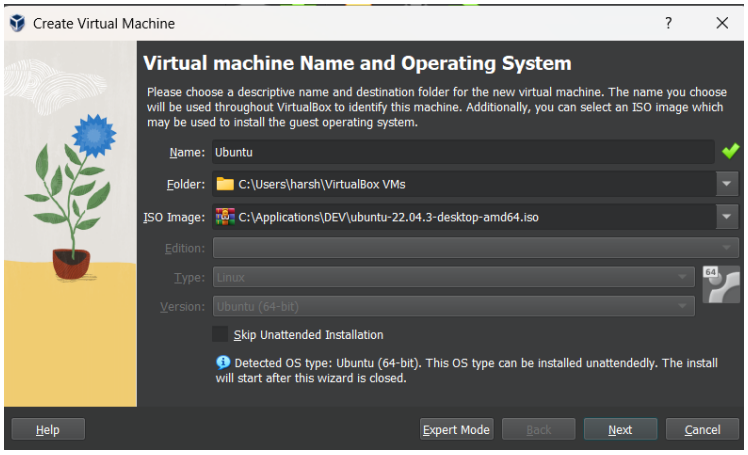


- Check Summary and click Finish. Do the same for other VM.

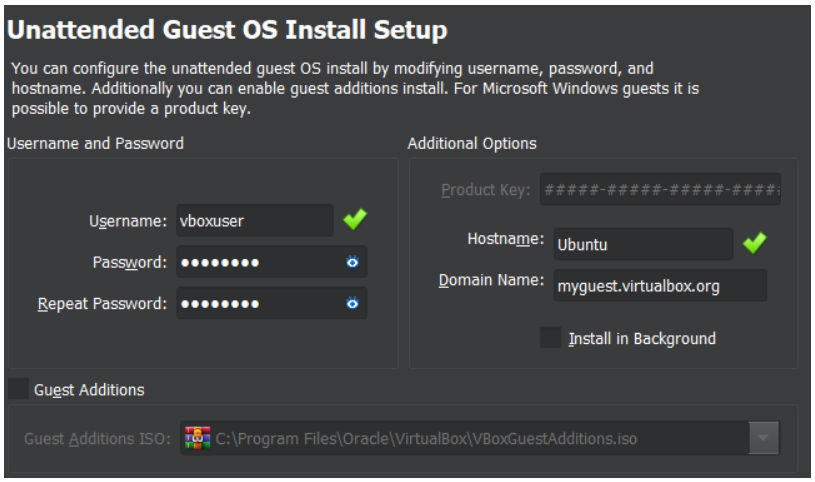


2. Install two or more Guest Operating Systems on all the VMs

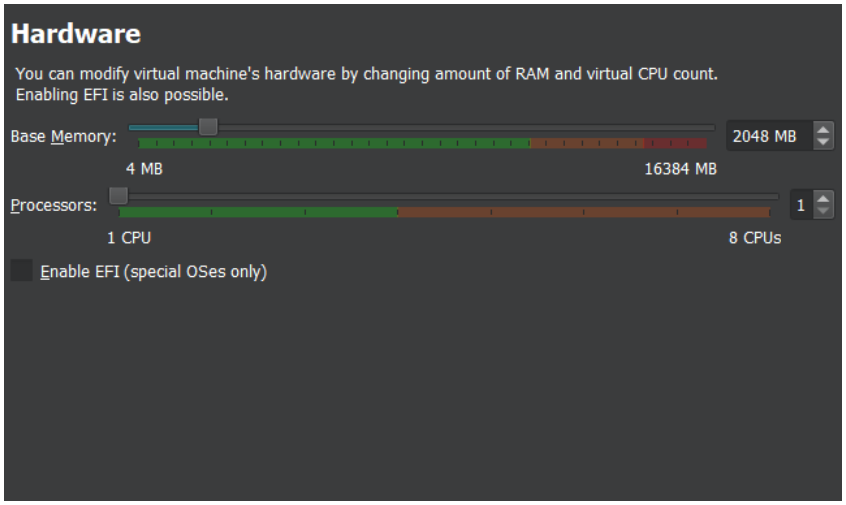
- Click on New button on Home Page. Choose file path of OS.



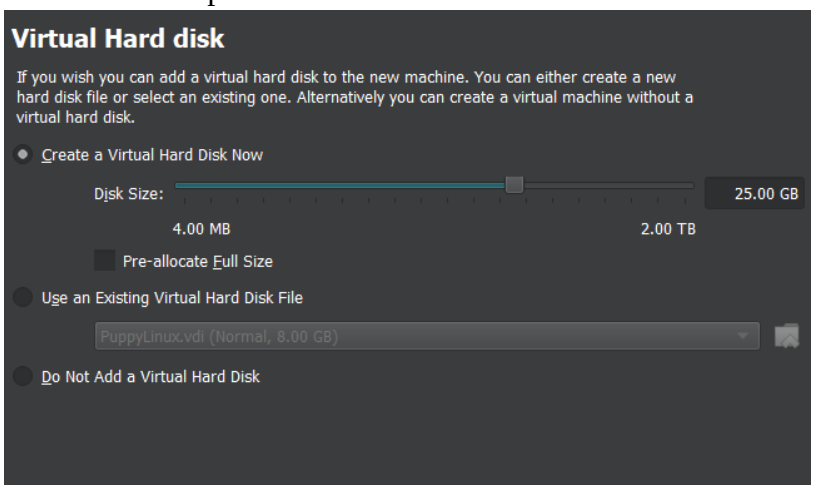
- Give User details



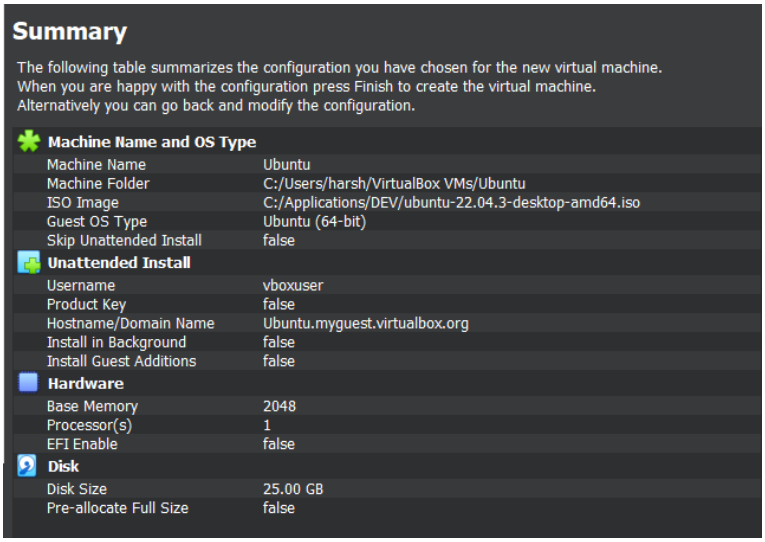
- Allocate Memory and Processors



- Allocate Disk Space



- Check Summary and click Finish. Do the same for other VM.



3. Run simple applications or programs on all the VMs.

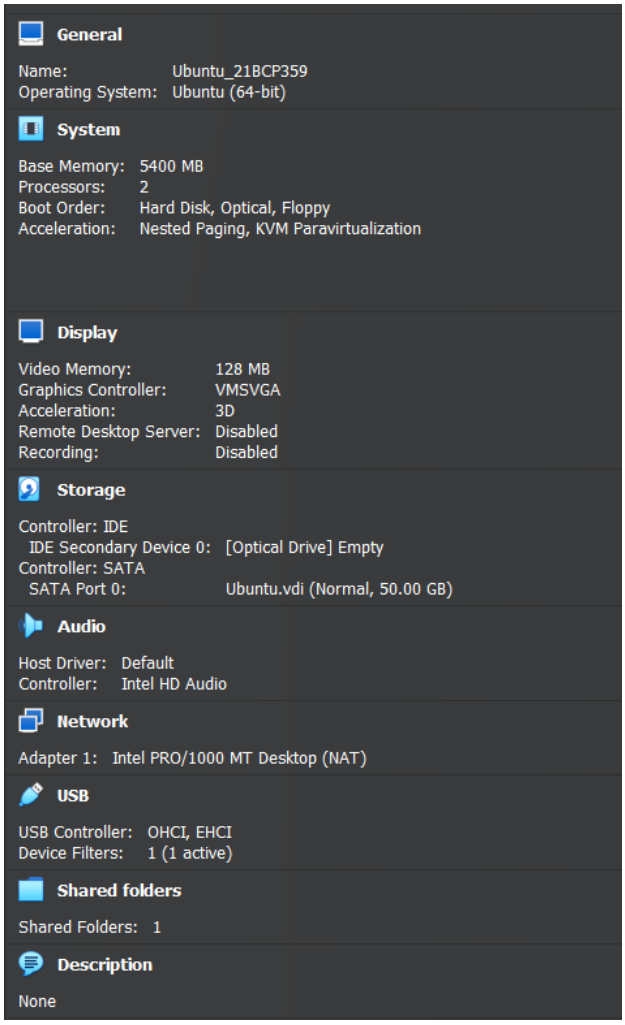
a. Run a simple Python script printing even numbers in VM-1 at 5 sec intervals.

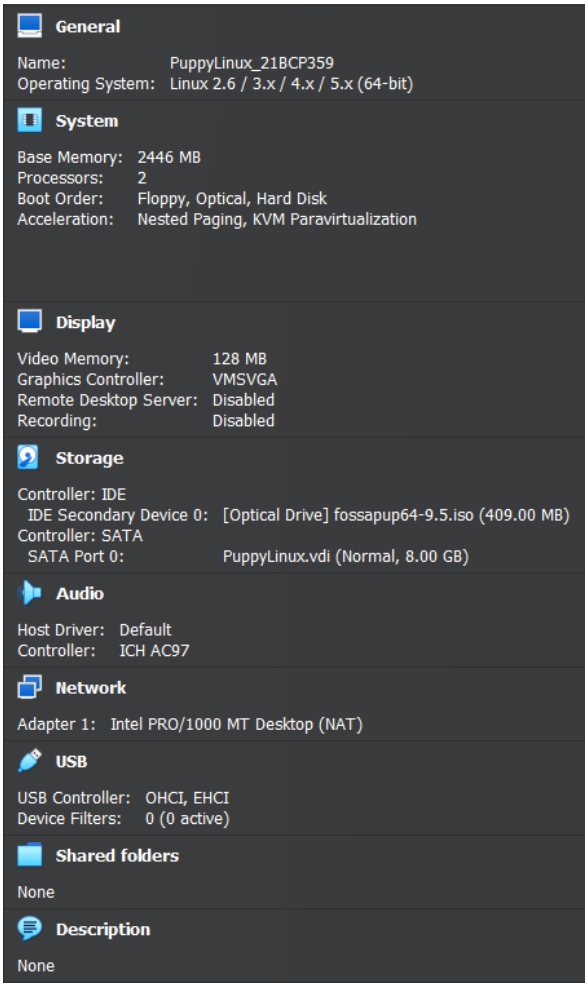
```
harsh@Ubuntu:~/Desktop/PDEU/Cloud_Computing$ python3 odd_even.py
0
2
4
6
```

b. Run a simple Python script printing odd numbers in VM-2 at 5 sec intervals.

```
harsh@Ubuntu:~/Desktop/PDEU/Cloud_Computing$ python3 odd_even.py
1
3
5
7
```

4. View system configurations of each VM. Check whether it is same different from what you have created.



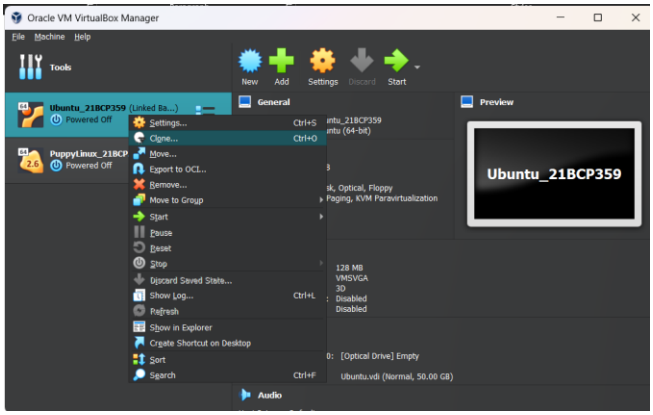


PRACTICAL 3

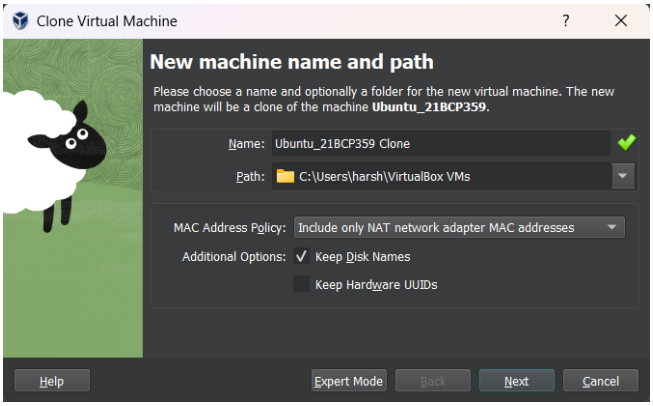
Aim: To Install and configure VMware Workstation Pro or Oracle Virtual Box for creating Virtual Machines, cloning, and deleting VMs.

1. **Cloning of Virtual Machines:**
   a. **Linked Cloning**

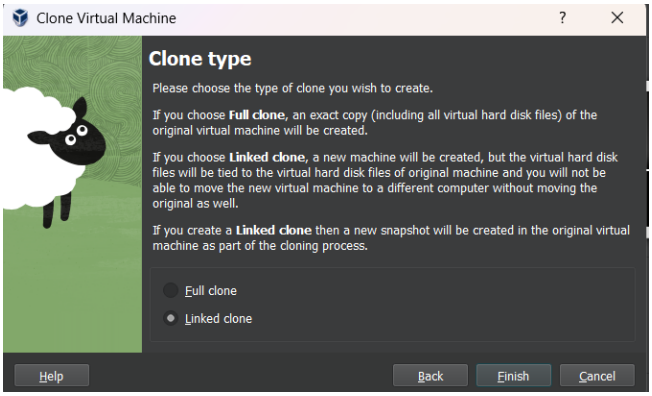
- Right Click on VM and select clone



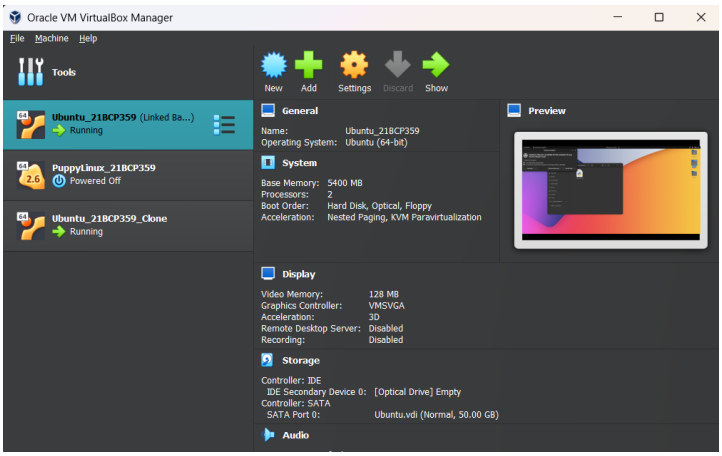
- Give necessary details



- Select Linked Clone

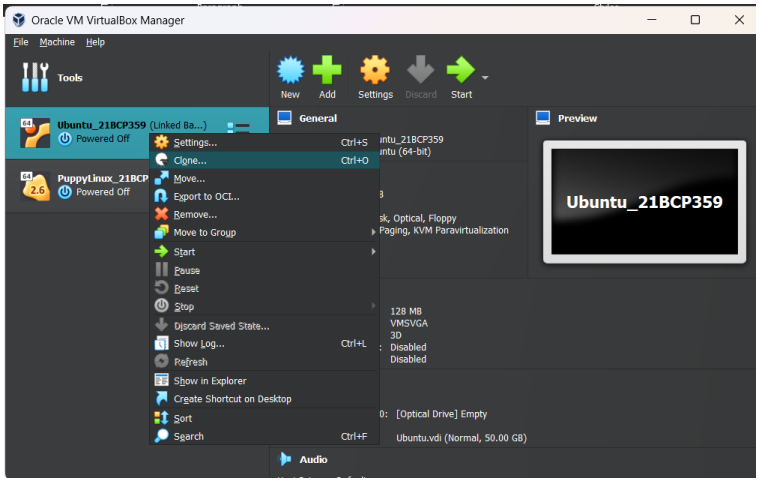


- The Linked Clone is Ready

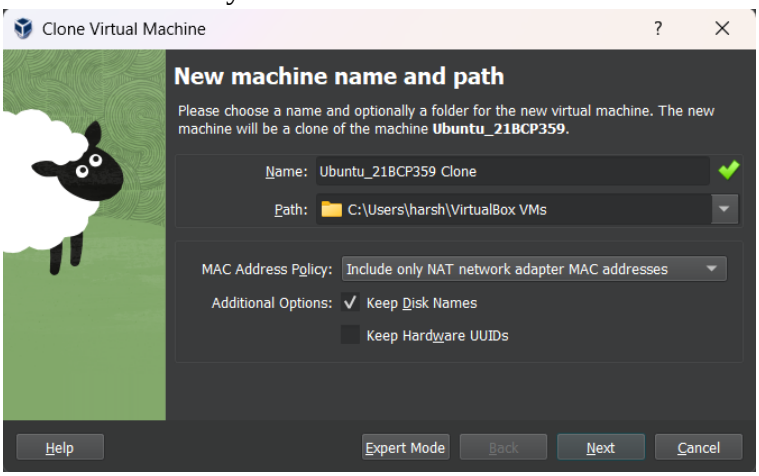


b. Full Cloning

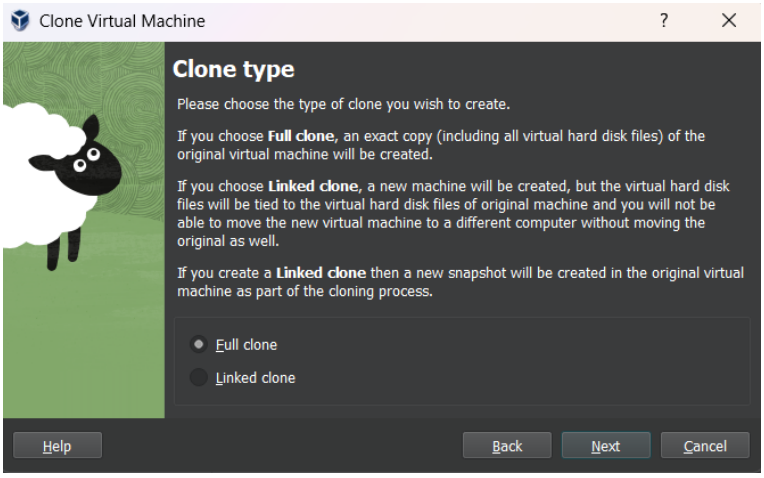
- Right Click on VM and select Clone



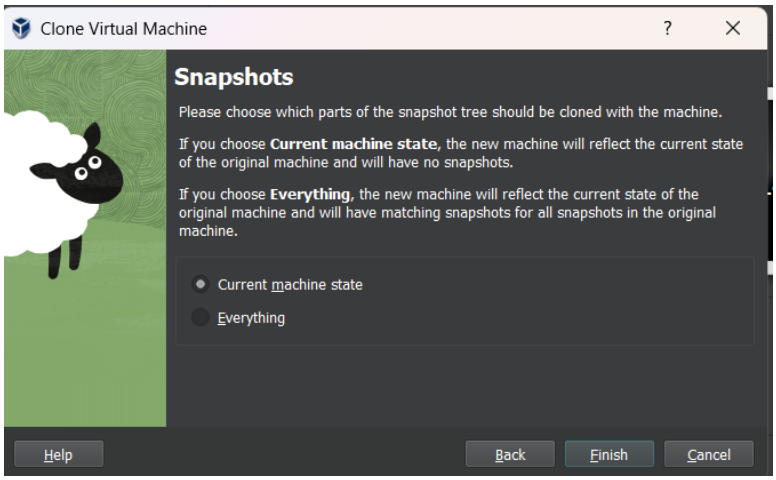
- Give Necessary details



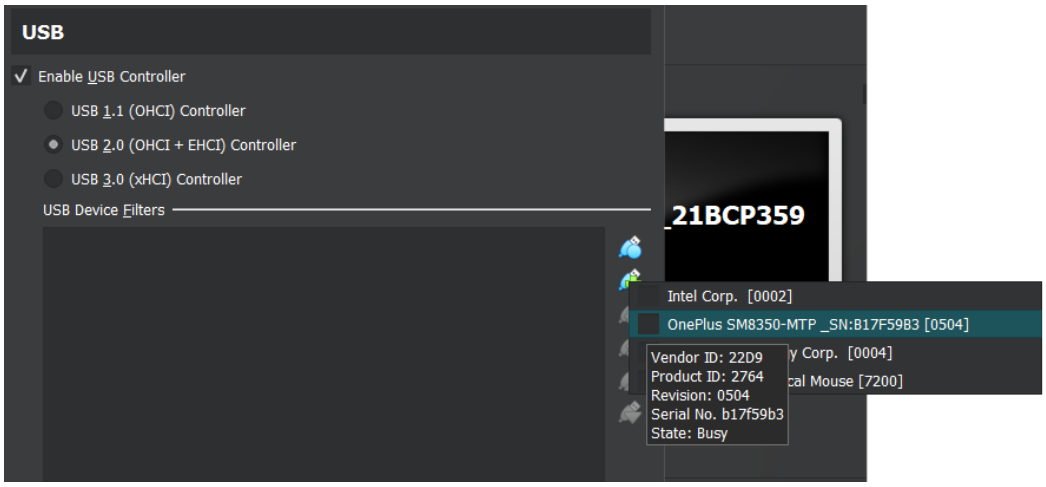
- Select Full Clone

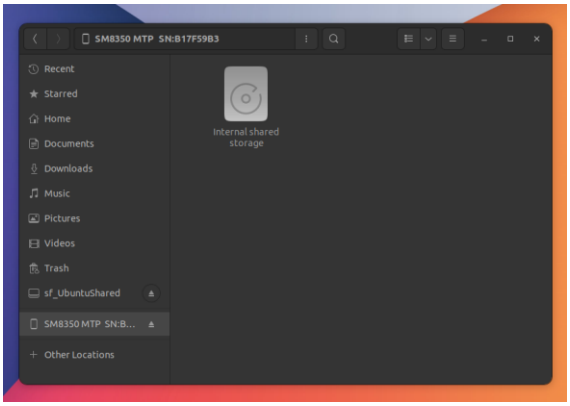


- Snapshots: Select Current Machine State



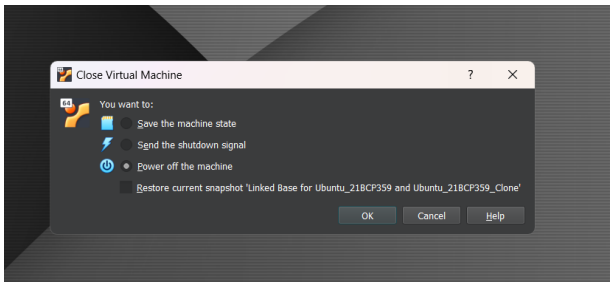
2. Add a USB Controller to Virtual Machine



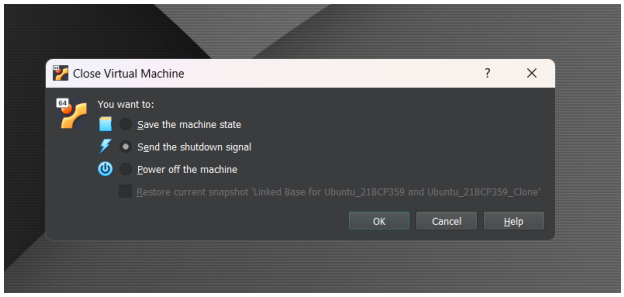


3. Closing Virtual Machines & Exiting VMM

a. Hard Power Off

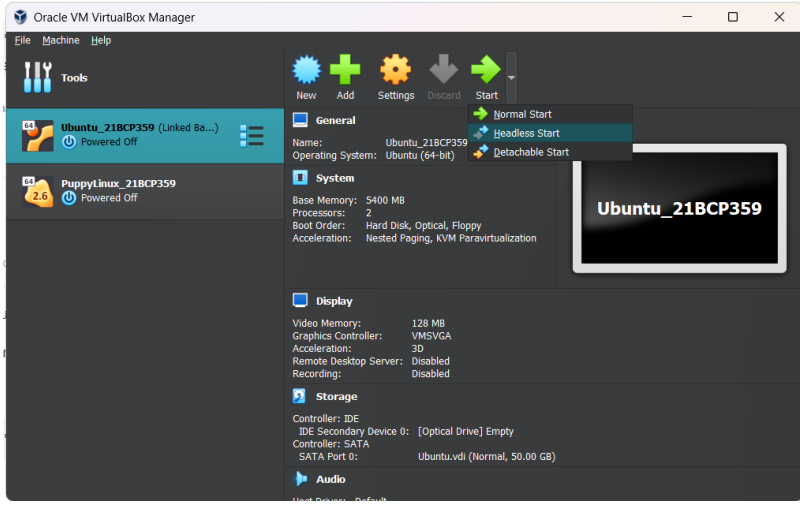


b. Soft Power Off



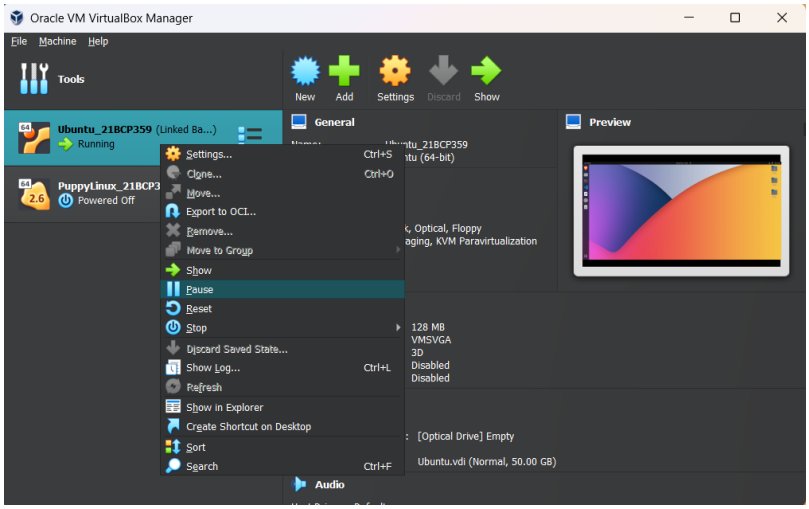
c. Configure Virtual Machine to Run in the Background

- In Start menu of VM click Headless Start

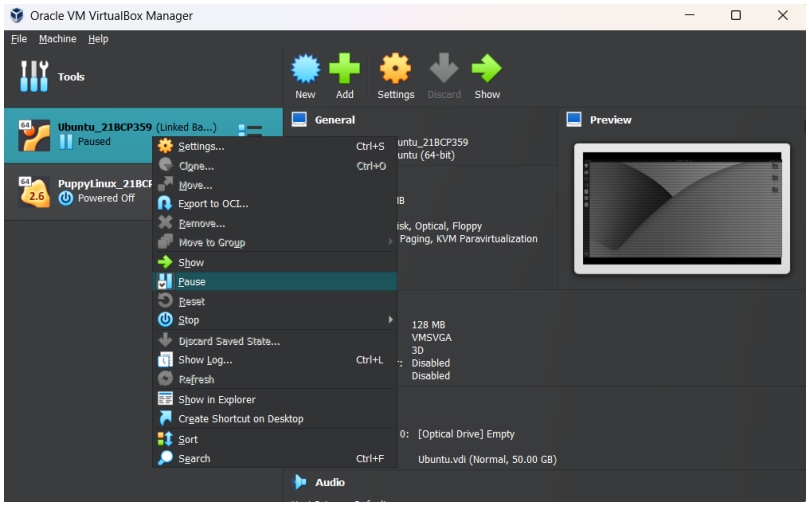


d. Pause and Un-pause a Virtual Machine

Right click on VM and click Pause to Pause

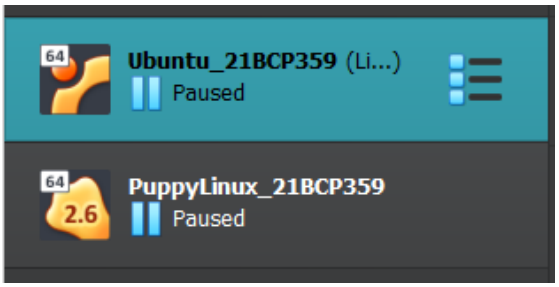


Again click Pause to Resume



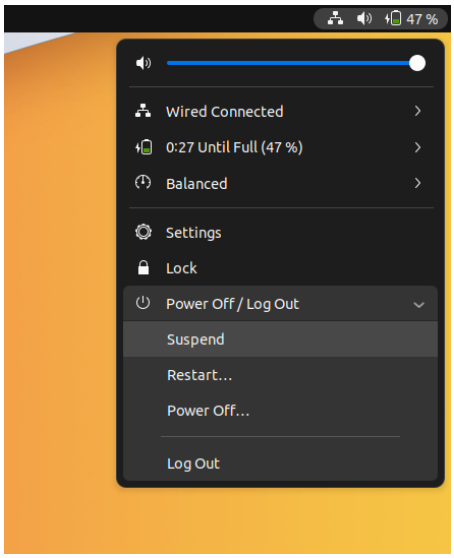
e. Pause all Virtual Machine without interacting Virtual Box

```
C:\Users\harsh>VBoxManage controlvm "PuppyLinux_21BCP359" pause

C:\Users\harsh>VBoxManage controlvm "Ubuntu_21BCP359" pause

C:\Users\harsh>
```



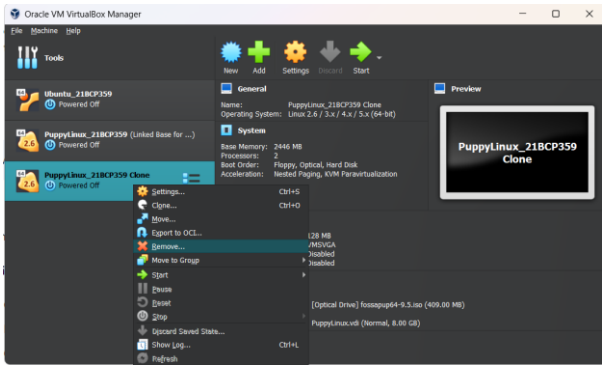
f. Suspend and Resume Virtual Machines

In power menu of Guest OS click Suspend

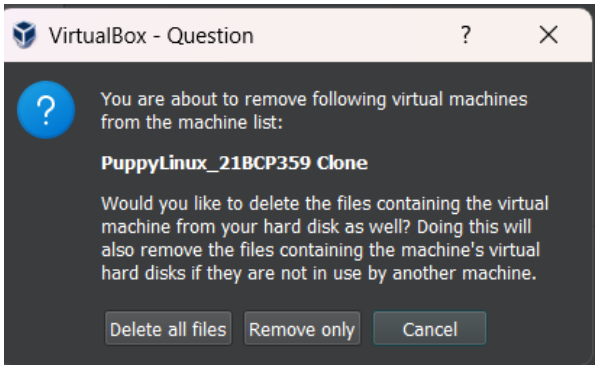


4. Delete a Virtual Machine from the disk

Right click on VM and select Delete.

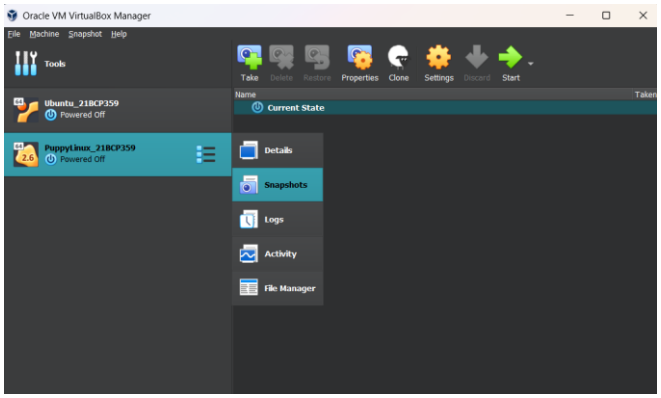


Then click Delete all files

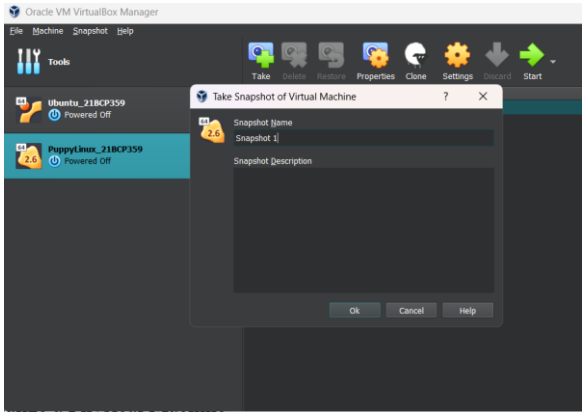


5. To take snapshot of Virtual Machines

Click on Tools icon on VM and select Snapshots.



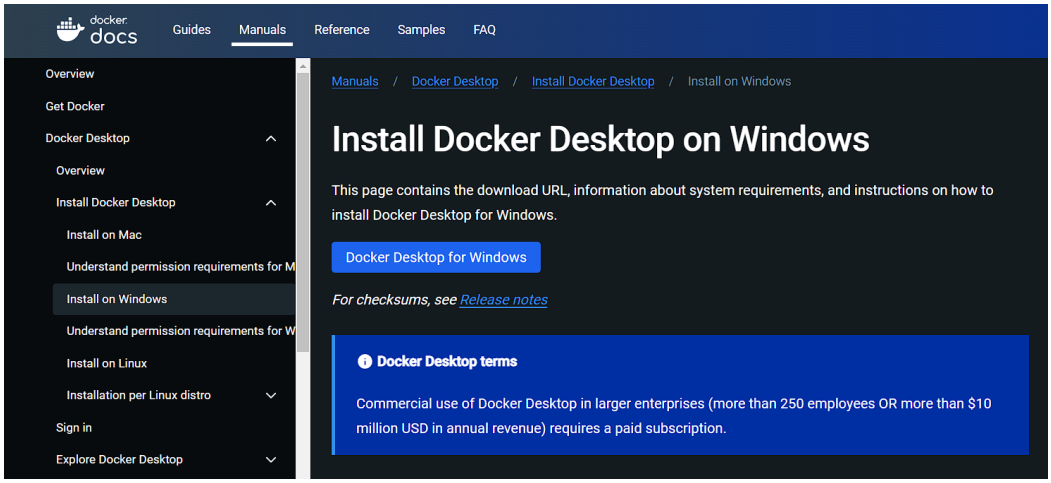
Give a name and click OK

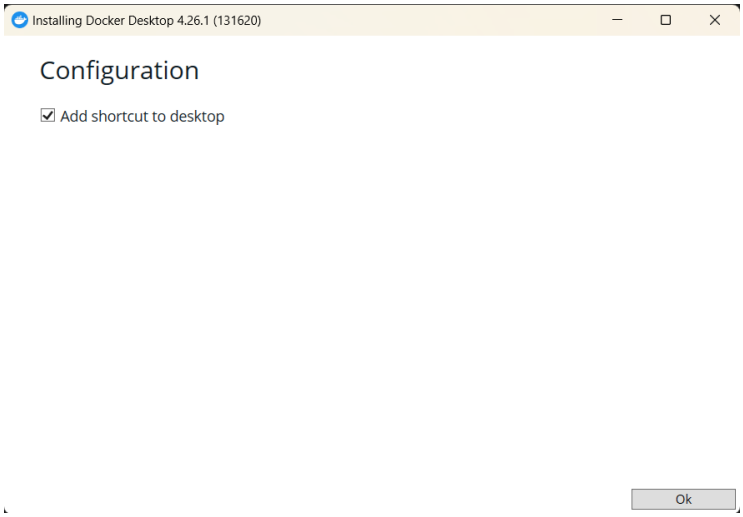


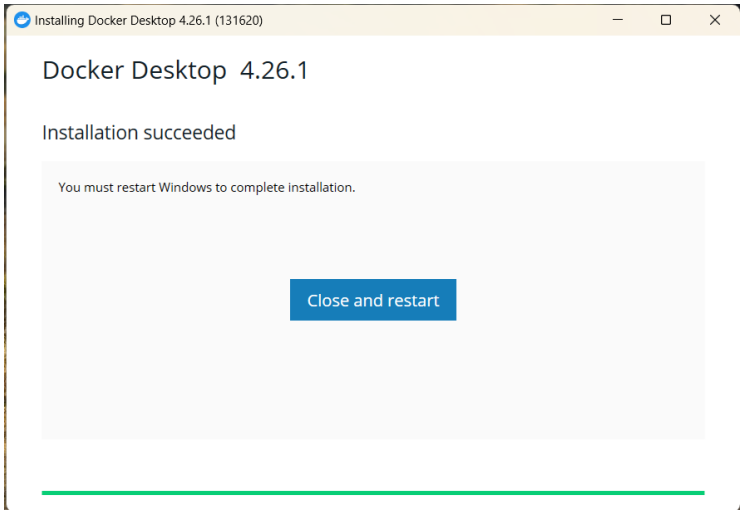
PRACTICAL 4

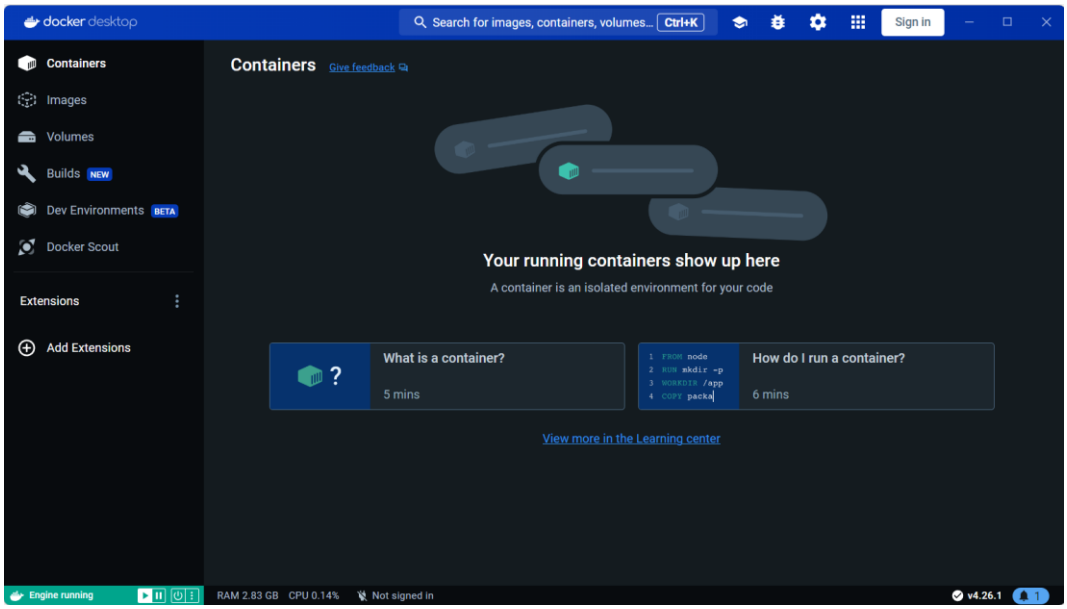
Aim: To Explore Docker Container

1. Install Docker in your environment









2. Verify your Docker by looking up the version

```
C:\Users\harsh>docker version
Client:
 Cloud integration: v1.0.35+desktop.5
 Version:           24.0.6
 API version:       1.43
 Go version:        go1.20.7
 Git commit:        ed223bc
 Built:             Mon Sep  4 12:32:48 2023
 OS/Arch:           windows/amd64
 Context:           default

Server: Docker Desktop 4.24.2 (124339)
 Engine:
  Version:          24.0.6
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.7
  Git commit:       1a79695
  Built:            Mon Sep  4 12:32:16 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.22
  GitCommit:        8165feabfdfe38c65b599c4993d227328c231fca
 runc:
  Version:          1.1.8
  GitCommit:        v1.1.8-0-g82f18fe
 docker-init:
  Version:          0.19.0
```

3. Run a hello-world container

```
(base) PS C:\Users\harsh> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2e6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

4. Verify the hello-world image

```
(base) PS C:\Users\harsh> docker images
REPOSITORY                       TAG                                       IMAGE ID       CREATED         SIZE
shaharsh624/test_app_video       latest                                    5c0f51298cff   3 months ago    1.1GB
<none>                           <none>                                    793300de7692   5 months ago    918MB
docker/welcome-to-docker         latest                                    912b66cfd46e   7 months ago    13.4MB
nginx                            1.23                                      a7be6198544f   8 months ago    142MB
hello-world                      latest                                    d2c94e258dcb   9 months ago    13.3kB
nginx/docker-extension           0.0.3                                     41d3d0d7d940   9 months ago    7.53MB
alpine                           3.16.3                                    bfe296a52501   14 months ago   5.54MB
docker/dev-environments-default  stable-1                                  7eb85f44c229   16 months ago   610MB
crazymax/linguist                7.20.0                                    dc5f9ae3be6a   20 months ago   72.2MB
docker/desktop-git-helper        5a4fca126aadcd3f6cc3a011aa991de982ae7000  efe2d67c403b   2 years ago     44.2MB
```

5. Get the sample application

```
C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing
\LAB\Practical_5>git clone https://github.com/docker/getting-started-a
pp.git
Cloning into 'getting-started-app'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 68 (delta 10), reused 9 (delta 9), pack-reused 37Receivi
ng objects:  92% (63/68), 1.58 MiB | 1.00 MiB/s
Receiving objects: 100% (68/68), 1.75 MiB | 900.00 KiB/s, done.
Resolving deltas: 100% (11/11), done.
```

6. Run the application on the Docker

a. Build the Docker Image

```
C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing
\LAB\Practical_5>cd getting-started-app

C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing
\LAB\Practical_5\getting-started-app>type nul > Dockerfile

C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing
\LAB\Practical_5\getting-started-app>code .

C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing
\LAB\Practical_5\getting-started-app>docker build -t getting-started .
```

```
C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_5\getting-started-app>docker build -t getting-started .
[+] Building 58.1s (10/10) FINISHED                                                                                    docker:default
 => [internal] load .dockerignore                                                                                              0.0s
 => => transferring context: 2B                                                                                                0.0s
 => [internal] load build definition from Dockerfile                                                                           0.1s
 => => transferring dockerfile: 154B                                                                                           0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                                              3.7s
 => [auth] library/node:pull token for registry-1.docker.io                                                                    0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:0085670310d2879621f96a4216c893f92e2ded827e9e6ef8437672e1bd72f437       28.4s
 => => resolve docker.io/library/node:18-alpine@sha256:0085670310d2879621f96a4216c893f92e2ded827e9e6ef8437672e1bd72f437        0.0s
 => => sha256:c8eb770fbfacf54104162cc9035c478ddb7d8dc15dca5298af028257f1dbdb3f 7.14kB / 7.14kB                                 0.0s
 => => sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8 3.41MB / 3.41MB                                 1.5s
 => => sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 40.24MB / 40.24MB                              25.7s
 => => sha256:3d4a65156edf0208c8421995310d9e662e7ee63e2bcae660efb02f6c4ddef6a9 2.34MB / 2.34MB                                2.3s
 => => sha256:0085670310d2879621f96a4216c893f92e2ded827e9e6ef8437672e1bd72f437 1.43kB / 1.43kB                                0.0s
 => => sha256:aacbcec05180c1dd8c33dba8a9c42b75dbfdd659aa57617497f1ce2c5d83d889 1.16kB / 1.16kB                                0.0s
 => => extracting sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8                                      0.2s
 => => sha256:5bdb6c27eb32087b71a9dde411c1f1eeb87563c0445f89db4eb7639d2cf50f45 450B / 450B                                    2.0s
 => => extracting sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97                                      2.2s
 => => extracting sha256:3d4a65156edf0208c8421995310d9e662e7ee63e2bcae660efb02f6c4ddef6a9                                      0.1s
 => => extracting sha256:5bdb6c27eb32087b71a9dde411c1f1eeb87563c0445f89db4eb7639d2cf50f45                                      0.0s
 => [internal] load build context                                                                                              0.5s
 => => transferring context: 6.50MB                                                                                            0.4s
 => [2/4] WORKDIR /app                                                                                                         0.3s
 => [3/4] COPY . .                                                                                                             0.1s
 => [4/4] RUN yarn install --production                                                                                       24.3s
 => exporting to image                                                                                                         1.4s
 => => exporting layers                                                                                                        1.4s
 => => writing image sha256:a55ffbe50aeb5c4a9ba45c199f86f65183b2a8f0e04fd2453809f4ed64ebcd9a                                   0.0s
 => => naming to docker.io/library/getting-started                                                                             0.0s
```

b. Verify the newly build image

```
C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_5\getting-started-app>docker ps -a
CONTAINER ID   IMAGE                                    COMMAND                  CREATED          STATUS                      PORTS
      NAMES
3c1136c87876   hello-world                              "/hello"                 20 minutes ago   Exited (0) 20 minutes ago
      sweet_khayyam
```

7. Run the Docker container

a. Run the container in the detached mode. Name the container “Test Sample”.

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud
 Computing\LAB\Practical_5> docker run --detach --name TestSample gett
ing-started
239b6d77a42e1138802bd6e8f3e1b1ea7a49095ceff26fc314ecdcd423bab62a
```

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud
 Computing\LAB\Practical_5> docker run -dp 127.0.0.1:3000:3000 getting
-started
393ec96fa61aaafbcd4c8aa3cb8896190b50a506c8c47823f577eb4019e61af2
```

b. List all the running containers.

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_5> docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS          PORTS      NAMES
239b6d77a42e   getting-started   "docker-entrypoint.s…"   53 seconds ago   Up 52 seconds   3000/tcp   TestSample
```

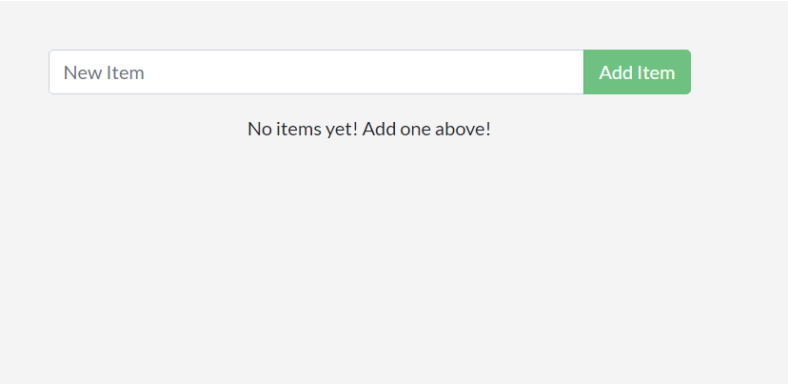
c. Inspect the running containers

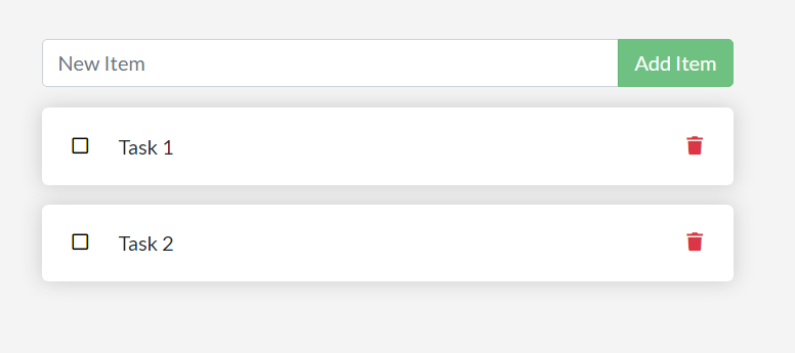

```
{
    "Id": "239b6d77a42e1138802bd6e8f3e1b1ea7a49095ceff26fc314ecdcd423bab62a",
    "Created": "2024-02-01T11:32:00.805207824Z",
    "Path": "docker-entrypoint.sh",
    "Args": [
        "node",
        "src/index.js"
    ],
    "State": {
        "Status": "running",
        "Running": true,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 3247,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2024-02-01T11:32:01.278241859Z",
        "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:a55ffbe50aeb5c4a9ba45c199f86f65183b2a8f0e04fd2453809f4ed64ebcd9a",
    "ResolvConfPath": "/var/lib/docker/containers/239b6d77a42e1138802bd6e8f3e1b1ea7a49095ceff26fc314ecdcd423bab62a/resolv.conf
    "HostnamePath": "/var/lib/docker/containers/239b6d77a42e1138802bd6e8f3e1b1ea7a49095ceff26fc314ecdcd423bab62a/hostname",
    "HostsPath": "/var/lib/docker/containers/239b6d77a42e1138802bd6e8f3e1b1ea7a49095ceff26fc314ecdcd423bab62a/hosts",
    "LogPath": "/var/lib/docker/containers/239b6d77a42e1138802bd6e8f3e1b1ea7a49095ceff26fc314ecdcd423bab62a/239b6d77a42e113880
    "Name": "/TestSample",
```


d. Get the logs of “Test Sample” containers

```
2024-02-01 17:07:23 Using sqlite database at /etc/todos/todo.db
2024-02-01 17:07:23 Listening on port 3000
2024-02-01 17:11:55 Using sqlite database at /etc/todos/todo.db
2024-02-01 17:11:55 Listening on port 3000
```

8. Access the application



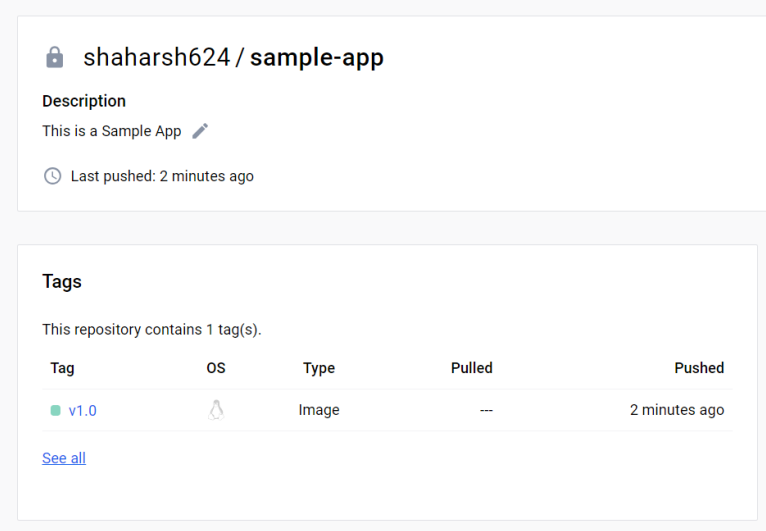


9. Pushing and pulling an image to and from repository

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud
 Computing\LAB\Practical_5> docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
29202e855b20: Pull complete
Digest: sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c
7dd060a74
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud
 Computing\LAB\Practical_5> docker tag getting-started shaharsh624/sam
ple-app:v1.0
```

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud
 Computing\LAB\Practical_5> docker push shaharsh624/sample-app:v1.0
The push refers to repository [docker.io/shaharsh624/sample-app]
b685fe182fe3: Pushed
0627f971a9d6: Pushed
6c547c29af99: Pushed
b325b33b9813: Mounted from library/node
4a0d315ad53e: Mounted from library/node
29e213bad130: Mounted from library/node
d4fc045c9e3a: Mounted from library/node
v1.0: digest: sha256:e202253e711d5bc4e358bda354aae2b0795e6ef9b2f63bc3e
04871821dc85edf size: 1787
```



10. Clean Up

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud
 Computing\LAB\Practical_5> docker stop 239b6d77a42e1138802bd6e8f3e1b1
ea7a49095ceff26fc314ecdcd423bab62a
239b6d77a42e1138802bd6e8f3e1b1ea7a49095ceff26fc314ecdcd423bab62a
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud
 Computing\LAB\Practical_5> docker stop 393ec96fa61aaafbcd4c8aa3cb8896
190b50a506c8c47823f577eb4019e61af2
393ec96fa61aaafbcd4c8aa3cb8896190b50a506c8c47823f577eb4019e61af2
```

PRACTICAL 5

Working with DOCKER

Aim: Create two dockers with front-end and back-end applications and connect these containers. The details of the containers are given below:

Web Application - Front-End + Back-End Application

- MariaDB - Back-End (one container)
- WordPress - Front-End (different container)

1. Execute MariaDB Container First:

- Make the Container name Static like – some-mariadb
- Set the DB Root password using Env Variable - MARIADB_ROOT_PASSWORD
- Set the DB Username using Env Variable - MARIADB_USER
- Set the DB password using Env Variable - MARIADB_PASSWORD
- Set the DB Name using Env Variable - MARIADB_DATABASE

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Clou
d Computing\LAB\Practical_6> docker run --detach --name some-mariadb
--env MARIADB_ROOT_PASSWORD=my-secret-pw  mariadb:latest
d9915a7c8502c42be1dfec5b2380e7155ca9b4e5ec3d3c4249acfe05f900bd84
```

```
d Computing\LAB\Practical_6> docker exec -it some-mariadb bash
root@d9915a7c8502:/# ls
bin   dev                        etc   lib    lib64   media  opt   r
oot  sbin  sys  usr
boot  docker-entrypoint-initdb.d  home  lib32  libx32  mnt    proc  r
un   srv   tmp  var
```

```
root@d9915a7c8502:/# mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 11.3.2-MariaDB-1:11.3.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.001 sec)

MariaDB [(none)]>
```

2. Create a network

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Com
uting\LAB\Practical_6> docker network create some-network
a02cfac328665426c7e69d9a217527d116c566dcad8d0fa4a1c6bd31b539726d
```

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Com
uting\LAB\Practical_6> docker network ls
NETWORK ID     NAME           DRIVER    SCOPE
3a5b407659b5   bridge         bridge    local
dbf9317492ed   host           host      local
20eca24ea466   none           null      local
a02cfac32866   some-network   bridge    local
```

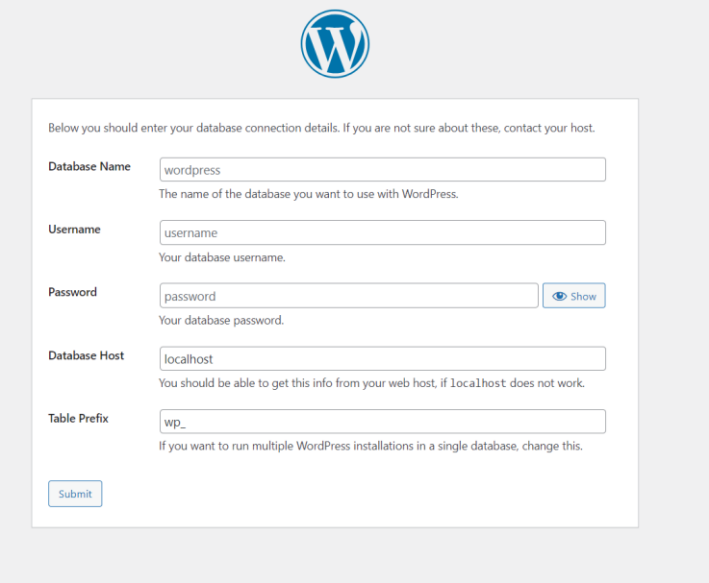
3. Execute WordPress Container -

- Make the Container name Static like - wordpress-container
- Set the DB Container Name in Env Variable - WORDPRESS_DB_HOST
- Set the DB Name in Env Variable - WORDPRESS_DB_NAME
- Set the DB User in Env Variable - WORDPRESS_DB_USER
- Set the DB password in Env Variable - WORDPRESS_DB_PASSWORD
- Expose Front-End Container on port 8080/80

```
uting\LAB\Practical_6> docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
8a1e25ce7c4f: Already exists
5de14226e170: Pull complete
d5aaf617d1d2: Pull complete
d3ba065e262f: Pull complete
142ecae067f5: Pull complete
```

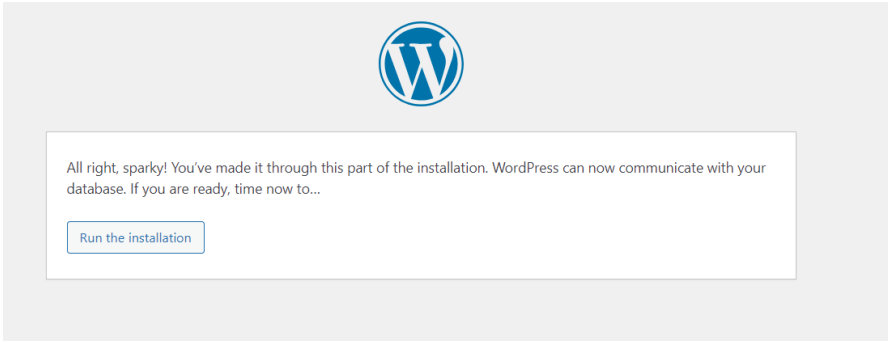
```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Comp
uting\LAB\Practical_6> docker run --network some-network -dp 127.0.0.1:3300
:80 wordpress
439de9458e4d3fdfe89dfed50824f3212874163541caf3b261c142e5dda6fb45
```

- Complete WordPress setup




```
MariaDB [(none)]> create database wordpressdb;
Query OK, 1 row affected (0.001 sec)
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wordpressdb        |
+--------------------+
5 rows in set (0.001 sec)
```



Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title Sample Site

Username harsh

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password harsh123 Hide

Very weak

Important: You will need this password to log in. Please store it in a secure location.

Confirm Password Confirm use of weak password

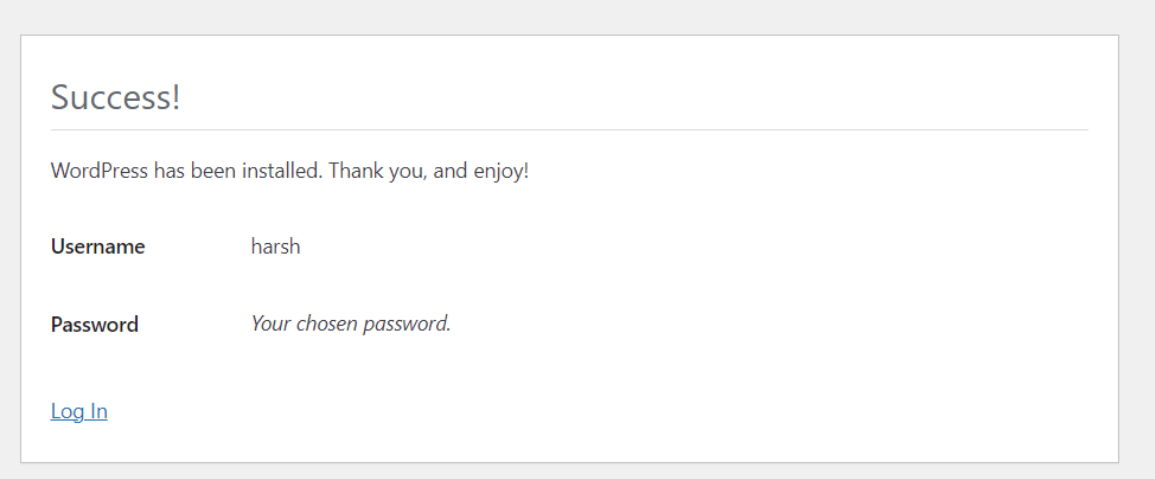
Your Email harsh.sce21@sot.pdpu.ac.in

Double-check your email address before continuing.

Search engine visibility Discourage search engines from indexing this site

It is up to search engines to honor this request.

Install WordPress



4. Access Website on HostIP:PORT

http://127.0.0.1:3300/

PRACTICAL 6

Working with DOCKER

Aim: Create a customized docker image and perform the following

1. **Create Docker File by own.**

```
ARG PYTHON_VERSION=3.11.7
FROM python:${PYTHON_VERSION}-slim as base

ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

WORKDIR /app

ARG UID=10001
RUN adduser \
    --disabled-password \
    --gecos "" \
    --home "/nonexistent" \
    --shell "/sbin/nologin" \
    --no-create-home \
    --uid "${UID}" \
    appuser

RUN --mount=type=cache,target=/root/.cache/pip \
    --mount=type=bind,source=requirements.txt,target=requirements.txt \
    python -m pip install -r requirements.txt

USER appuser
COPY . .
EXPOSE 8000
CMD python app.py
```

2. **Get the latest python image from Docker Repository.**

```
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Comp
uting\LAB\Practical_7> docker pull python
Using default tag: latest
latest: Pulling from library/python
71215d55680c: Pull complete
3cb8f9c23302: Pull complete
5f899db30843: Pull complete
567db630df8d: Pull complete
d68cd2123173: Pull complete
63941d09e532: Pull complete
097431623722: Pull complete
09527fa4de8d: Pull complete
Digest: sha256:19973e1796237522ed1fcc1357c766770b47dc15854eafdda055b65953fe
5ec1
Status: Downloaded newer image for python:latest
docker.io/library/python:latest
```

3. Write down a simple python program or get it from internet.

```
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
fibonacci(10)
```

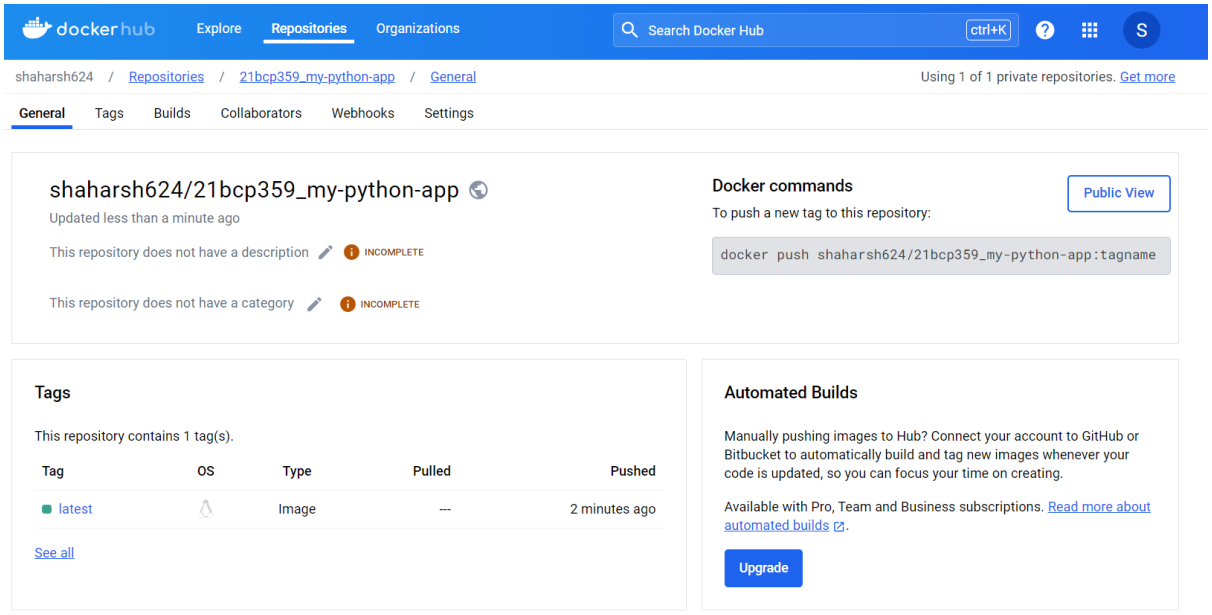
4. Run Python Program inside the Container.

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7> docker build -t 21bcp359_my-python-app .
[+] Building 16.7s (14/14) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                           0.0s
 => => transferring dockerfile: 1.66kB                                                                         0.0s
 => resolve image config for docker.io/docker/dockerfile:1                                                     8.3s
 => [auth] docker/dockerfile:pull token for registry-1.docker.io                                               0.0s
 => docker-image://docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b  5.7s
 => => resolve docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cb  0.0s
 => => sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56 8.40kB / 8.40kB                  0.0s
 => => sha256:4611ea7b7d89ce41ec5c63df83076ccec3fe8daa32a2d9c96e5decb72e9a8d67 482B / 482B                      0.0s
 => => sha256:ab56f6885c985024a40925d2fa322df997655db5f361ad9f221861f9c665a290 1.26kB / 1.26kB                  0.0s
 => => sha256:ccef65a67ab38a038c615e74c797b11a43d36505710abe93c87b021401bebd81 11.98MB / 11.98MB                5.3s
 => => extracting sha256:ccef65a67ab38a038c615e74c797b11a43d36505710abe93c87b021401bebd81                       0.2s
 => [internal] load metadata for docker.io/library/python:3.11.7-slim                                          2.3s
 => [auth] library/python:pull token for registry-1.docker.io                                                  0.0s
 => [internal] load .dockerignore                                                                              0.0s
 => => transferring context: 667B                                                                              0.0s
 => [base 1/5] FROM docker.io/library/python:3.11.7-slim@sha256:53d6284a40eae6b625f22870f5faba6c54f2a28db9027408  0.0s
 => [internal] load build context                                                                              0.0s
 => => transferring context: 310B                                                                              0.0s
 => CACHED [base 2/5] WORKDIR /app                                                                             0.0s
 => CACHED [base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "  0.0s
 => CACHED [base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.t  0.0s
 => [base 5/5] COPY . .                                                                                        0.1s
 => exporting to image                                                                                         0.1s
 => => exporting layers                                                                                        0.0s
 => => writing image sha256:0afdddc1a0bbf9e2db5b6803ab7c3f809050ab0840134cb91e2ef3b519f8e94e                   0.0s
 => => naming to docker.io/library/21bcp359_my-python-app                                                      0.0s
```

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7> docker run -it --rm --name 2
1bcp359_my-running-app 21bcp359_my-python-app
0 1 1 2 3 5 8 13 21 34
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7>
```

5. Tag and Push the Image to Docker Hub.

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7> docker tag 21bcp359_my-pytho
n-app shaharsh624/21bcp359_my-python-app
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7> docker push shaharsh624/21bc
p359_my-python-app
Using default tag: latest
The push refers to repository [docker.io/shaharsh624/21bcp359_my-python-app]
c6308a6674ea: Pushed
2a437f0cebed: Pushed
5a34a09fe985: Pushed
f723b1d28ad8: Pushed
d369fa6642a9: Mounted from library/python
ee1179522d3c: Mounted from library/python
ee03a3ee4c2e: Mounted from library/python
da5d55102092: Mounted from library/python
fb1bd2fc5282: Mounted from library/python
latest: digest: sha256:5a55ebdf8b166b83231d63ecef4e9c784d89fc9ecf7bf725316ce190a099a749 size: 2201
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7>
```



6. Remove Image from Local and Again execute it from Hub.

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7> docker rmi -f shaharsh624/21b
cp359_my-python-app
Untagged: shaharsh624/21bcp359_my-python-app:latest
Untagged: shaharsh624/21bcp359_my-python-app@sha256:5a55ebdf8b166b83231d63ecef4e9c784d89fc9ecf7bf725316ce190a099a749
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7> docker rmi -f 21bcp359_my-pyt
hon-app
Untagged: 21bcp359_my-python-app:latest
Deleted: sha256:0afdddc1a0bbf9e2db5b6803ab7c3f809050ab0840134cb91e2ef3b519f8e94e
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7>
```

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7> docker run -it --rm --name 21
bcp359_my-running-app shaharsh624/21bcp359_my-python-app
Unable to find image 'shaharsh624/21bcp359_my-python-app:latest' locally
latest: Pulling from shaharsh624/21bcp359_my-python-app
c57ee5000d61: Already exists
be0f2e005f57: Already exists
eab129fe7d73: Already exists
dd24933c9a93: Already exists
3fcbdacf3969: Already exists
61951b604ec9: Already exists
d959509106f6: Already exists
983ca7509753: Already exists
ccdc9b8bbf88: Already exists
Digest: sha256:5a55ebdf8b166b83231d63ecef4e9c784d89fc9ecf7bf725316ce190a099a749
Status: Downloaded newer image for shaharsh624/21bcp359_my-python-app:latest
0 1 1 2 3 5 8 13 21 34
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_7>
```

PRACTICAL 7

Working with DOCKER

Aim: Create three containers and access these containers from each other by using IP address and container name.

7. Create three containers.

Pull *busybox* Image

```
l_8>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
7b2699543f22: Pull complete
Digest: sha256:c3839dd800b9eb7603340509769c43e146a74c63dca3045a8e7dc8ee07e53966
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview busybo
x
```

Create and run three containers

```
l_8> docker run -d -it --name busybox1 busybox
9c04bc262c4a01882a1133f5b8d39c8088885c823cddd307f2ad1cb9d0c9cfa8
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practica
l_8> docker run -d -it --name busybox2 busybox
54c35e6da5f5a209ccfb4a8c5c9ee8e0018cf72c842a6555b16fd0e8419e374d
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practica
l_8> docker run -d -it --name busybox3 busybox
92ed77a43a1456ab12ffe18dd019c3989c617b198acf192b307b323d05631edc
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practica
l_8>
```

Create a network and connect with the containers

```
l_8> docker network create 21BCP359_net
255808b191e8423c3ecb99d6d7c10d11605e7ab05fae00d70fe68616fe69a2e9
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practica
l_8>
```

```
AB\Practical_8> docker network connect 21BCP359_net busybox1
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\L
AB\Practical_8> docker network connect 21BCP359_net busybox2
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\L
AB\Practical_8> docker network connect 21BCP359_net busybox3
(base) PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\L
AB\Practical_8>
```

8. Access these containers using IP address

Find IP Address of containers

```
_8> docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' busybox1
172.18.0.2172.17.0.2
```

```
_8> docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' busybox2
172.18.0.3172.17.0.3
```

```
_8> docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' busybox3
172.18.0.4172.17.0.4
```

Ping from one container to another

```
_8> docker exec busybox1 ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3): 56 data bytes
64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.160 ms
64 bytes from 172.17.0.3: seq=1 ttl=64 time=0.109 ms
64 bytes from 172.17.0.3: seq=2 ttl=64 time=0.142 ms
```

```
_8> docker exec busybox1 ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4): 56 data bytes
64 bytes from 172.17.0.4: seq=0 ttl=64 time=0.185 ms
64 bytes from 172.17.0.4: seq=1 ttl=64 time=0.164 ms
64 bytes from 172.17.0.4: seq=2 ttl=64 time=0.496 ms
```

```
_8> docker exec busybox2 ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4): 56 data bytes
64 bytes from 172.17.0.4: seq=0 ttl=64 time=0.132 ms
64 bytes from 172.17.0.4: seq=1 ttl=64 time=0.073 ms
64 bytes from 172.17.0.4: seq=2 ttl=64 time=0.088 ms
64 bytes from 172.17.0.4: seq=3 ttl=64 time=0.104 ms
```

9. Access these containers using container name.

```
AB\Practical_8> docker exec busybox1 ping busybox2
PING busybox2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.078 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.080 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.055 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.066 ms
```

```
AB\Practical_8> docker exec busybox1 ping busybox3
PING busybox3 (172.18.0.4): 56 data bytes
64 bytes from 172.18.0.4: seq=0 ttl=64 time=0.099 ms
64 bytes from 172.18.0.4: seq=1 ttl=64 time=0.082 ms
64 bytes from 172.18.0.4: seq=2 ttl=64 time=0.077 ms
64 bytes from 172.18.0.4: seq=3 ttl=64 time=0.082 ms
```

```
AB\Practical_8> docker exec busybox2 ping busybox3
PING busybox3 (172.18.0.4): 56 data bytes
64 bytes from 172.18.0.4: seq=0 ttl=64 time=0.067 ms
64 bytes from 172.18.0.4: seq=1 ttl=64 time=0.056 ms
64 bytes from 172.18.0.4: seq=2 ttl=64 time=0.171 ms
```

PRACTICAL 8

Working with DOCKER

Aim: Re-Do the problem of Lab 5 using docker compose. Also do assign a static IP address to WordPress container using subnetting.

Create two docker containers with front-end and back-end applications and connect these containers using docker compose.

1. **Running WordPress container normally**

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker run --name wp1 -dp 127.0.0.1:3000:80 wordpress
ce430c434d09cd9ac7ec5d6892f1d79d64e1bd35f9a66f2953bbd0c0c0888fa1
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
```

2. **Creating *compose.yaml* file**

```
services:
  website1:
    image: wordpress
    ports:
      - "3000:80"
```

3. **Execute Docker Compose**

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker-compose up -d
[+] Running 1/2
 - Network practical_9_default       Created                                    0.4s
 ✔Container practical_9-website1-1  Started                                    0.4s
```

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker compose ls
NAME                STATUS              CONFIG FILES
practical_9         running(1)          C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\S
EM 6\Cloud Computing\LAB\Practical_9\compose.yaml
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
```

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker compose stop
[+] Stopping 1/1
 ✔Container practical_9-website1-1  Stopped                                    1.3s
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker compose start
[+] Running 1/1
 ✔Container practical_9-website1-1  Started                                    0.3s
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
```

4. Running multiple containers

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker-compose up -d
[+] Running 2/2
 ✔Container practical_9-website2-1  Started                                    0.8s
 ✔Container practical_9-website1-1  Started                                    0.8s
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
```

5. Configuring network

```
services:
  website1:
    image: wordpress
    ports:
      - "3100:80"
  website2:
    image: wordpress
    ports:
      - "3200:80"
    networks:
      wordpress_net:
        ipv4_address: 192.168.99.20
networks:
  wordpress_net:
    ipam:
      driver: default
      config:
        - subnet: "192.168.99.0/24"
```

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker-compose up -d
[+] Running 2/4
 - Network practical_9_default        Created                                  0.8s
 - Network practical_9_wordpress_net  Created                                  0.8s
 ✔Container practical_9-website1-1   Started                                  0.7s
 ✔Container practical_9-website2-1   Started                                  0.7s
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
```

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9>
 docker inspect 0dd72c58403219edd0bed0b0c0b7791e6fd6b4ab35553b2ef7f5212b6e274e6b
[
    {
        "Id": "0dd72c58403219edd0bed0b0c0b7791e6fd6b4ab35553b2ef7f5212b6e274e6b",
        "Created": "2024-04-25T09:53:37.4892287Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "apache2-foreground"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
```

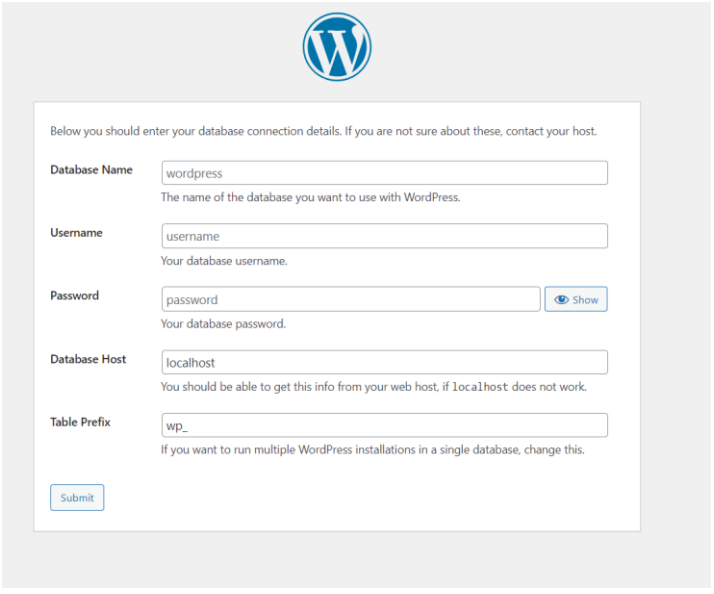
```
                "Gateway": "192.168.99.1",
                "IPAddress": "192.168.99.20",
                "IPPrefixLen": 24,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "DriverOpts": null,
                "DNSNames": [
                    "practical_9-website2-1",
                    "website2",
                    "0dd72c584032"
                ]
```

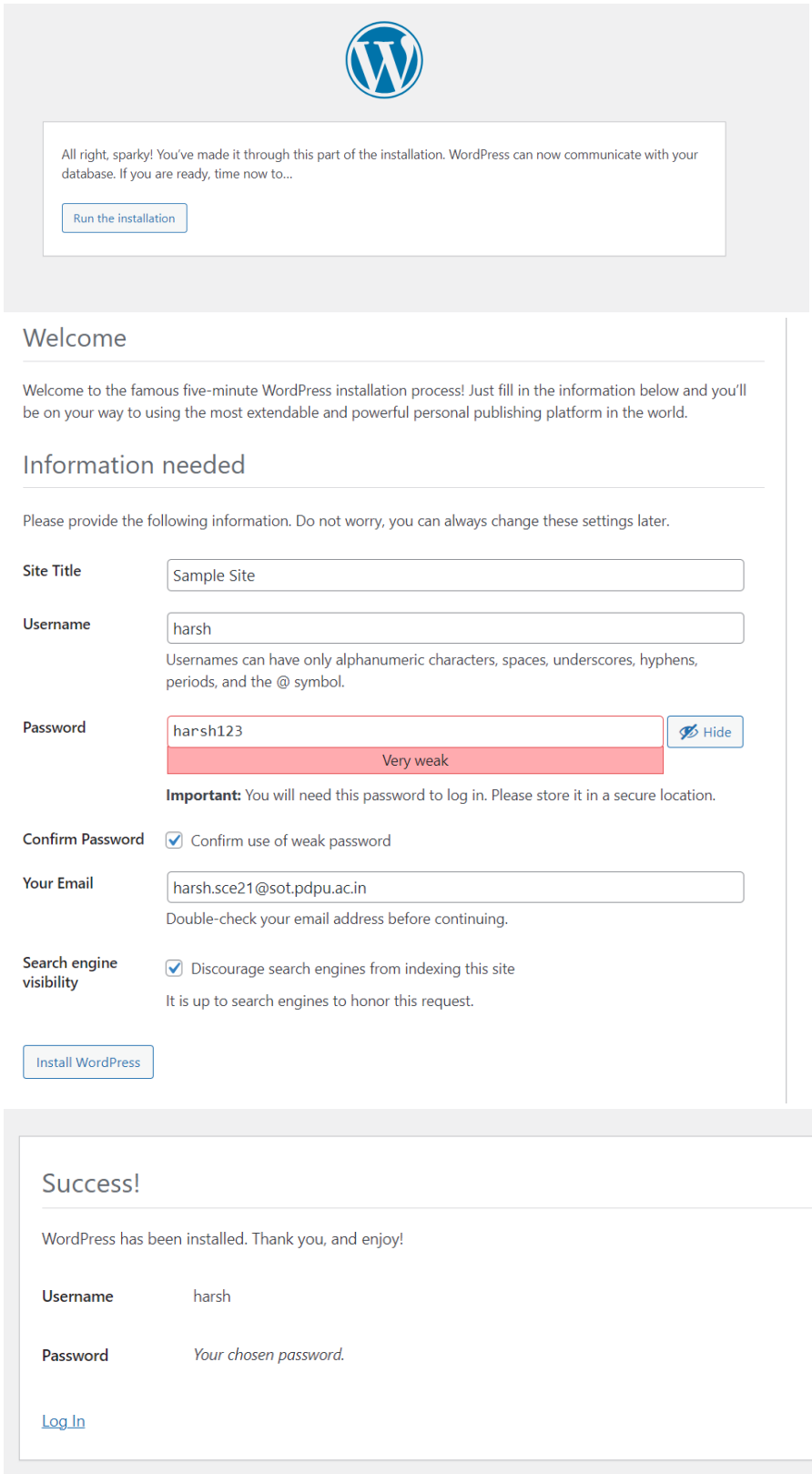
6. Connection Database with Container

```
version: "3"
services:
  mariadb_db:
    container_name: mariadb_container
    image: mariadb
    restart: always
    environment:
      MARIADB_ROOT_PASSWORD: password
      MARIADB_DATABASE: testdb
      MARIADB_USER: harsh
      MARIADB_PASSWORD: wp_password
    volumes:
    - mariadb_vol:/var/lib/mysql

  wordpress:
    depends_on:
```

```
      - mariadb_db
    image: wordpress:latest
    restart: always
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: mariadb_db:3306
      WORDPRESS_DB_NAME: testdb
      WORDPRESS_DB_USER: harsh
      WORDPRESS_DB_PASSWORD: wp_password
    volumes:
      ["./:/var/www/html"]

volumes:
 mariadb_vol: {}
```

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 6\Cloud Computing\LAB\Practical_9> docker-compose up -d
[+] Running 12/12
 ✔db 11 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                                    216.0s
   ✔bd37f6d99203 Pull complete                                                              162.3s
   ✔d2433cba0951 Pull complete                                                                4.8s
   ✔13702d9fe3c3 Pull complete                                                                7.0s
   ✔83bcc87284a1 Pull complete                                                               66.1s
   ✔c38d8660e1fa Pull complete                                                                8.7s
   ✔7e1bc321f421 Pull complete                                                               15.3s
   ✔bddd54b9c549 Pull complete                                                              202.2s
   ✔4eaae1e844ac Pull complete                                                               68.1s
   ✔5196e1e87d8f Pull complete                                                              158.7s
   ✔6586d096303c Pull complete                                                              160.1s
   ✔cf55ff1c80af Pull complete                                                              161.6s
[+] Running 2/5
 - Network website_default        Created                                                     1.5s
 - Volume "website_db"            Created                                                     1.4s
 - Volume "website_wordpress"     Created                                                     1.4s
 ✔Container website-db-1          Started                                                     1.1s
 ✔Container website-wordpress-1   Started                                                     0.8s
```





7. **Access Website on HostIP:PORT**

   http://127.0.0.1:3300/