

IT PLATFORM FINAL EXAM PROJECT

Virtual Machine and Docker Integration

Presented by Alvaro, Batool,
Bibhushan, Omar, Roza

Overview of Project

Objective:

Implement and integrate network configurations, Linux systems, and Docker technology.

Scope:

Virtual Machine setups.

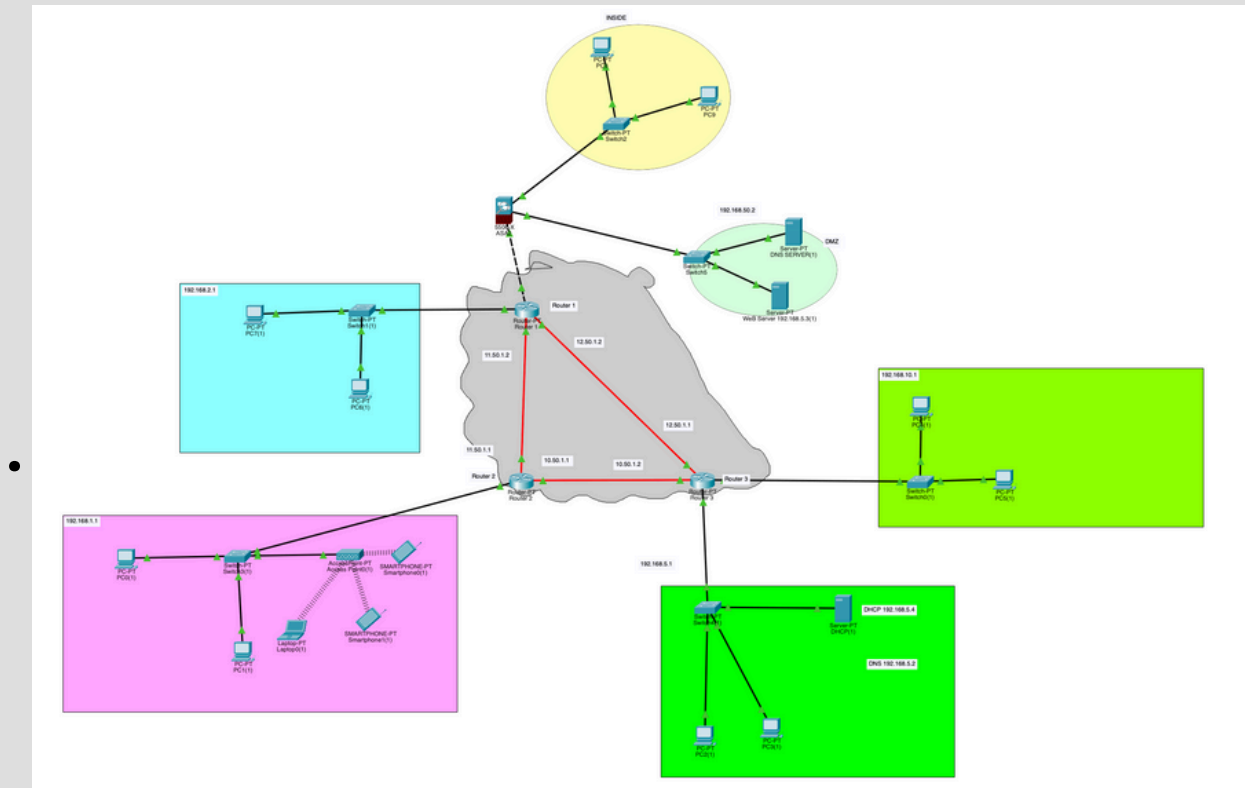
Network security and configurations.

Dockerized applications.

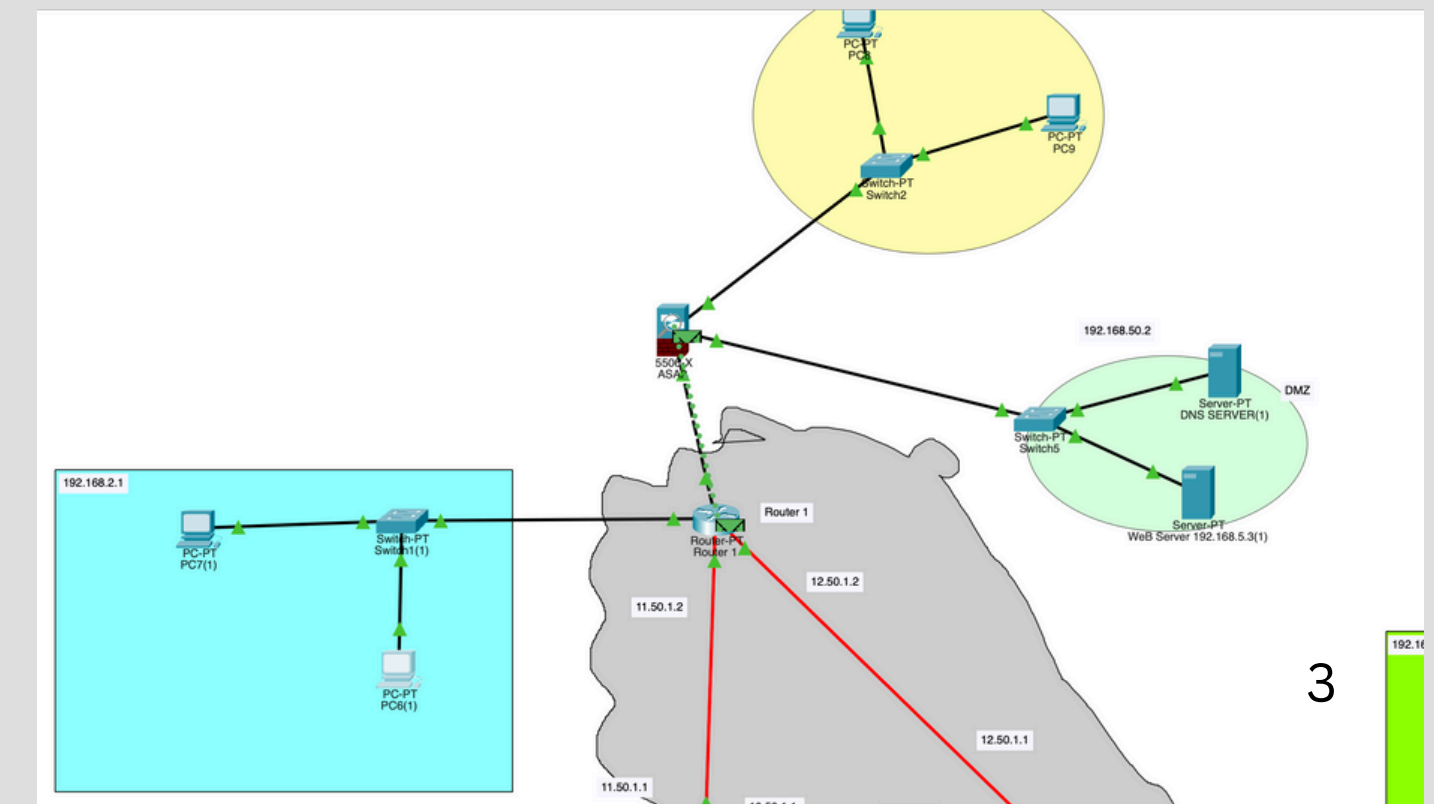
Task-specific achievements in IT environments.

Task 1 - Add Firewall

- Steps Taken:
 - Verified and corrected network configurations.
 - Changed the server pool network to DMZ.
 - Installed and configured firewall(s).
 -

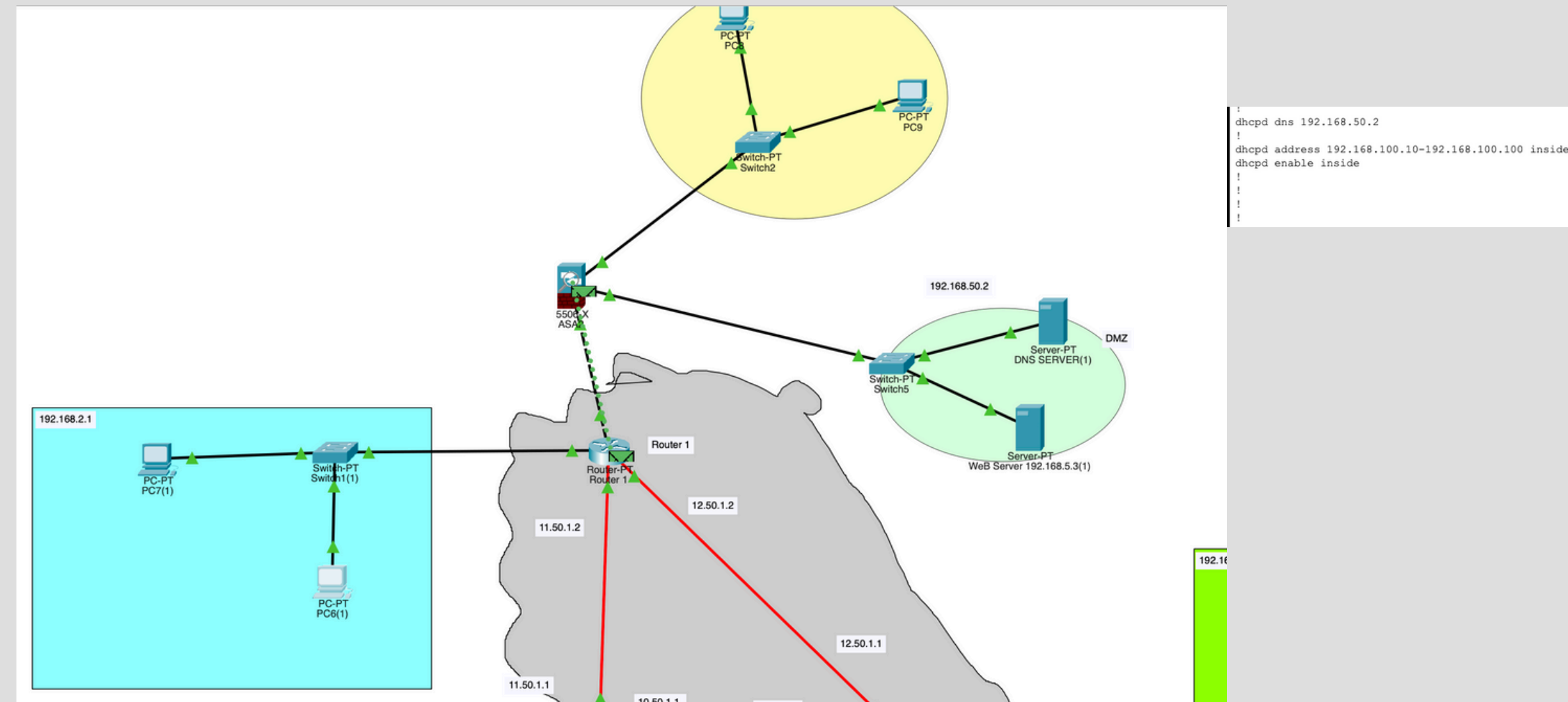


```
!
dhcpcd dns 192.168.50.2
!
dhcpcd address 192.168.100.10-192.168.100.100 inside
dhcpcd enable inside
!
!
!
```



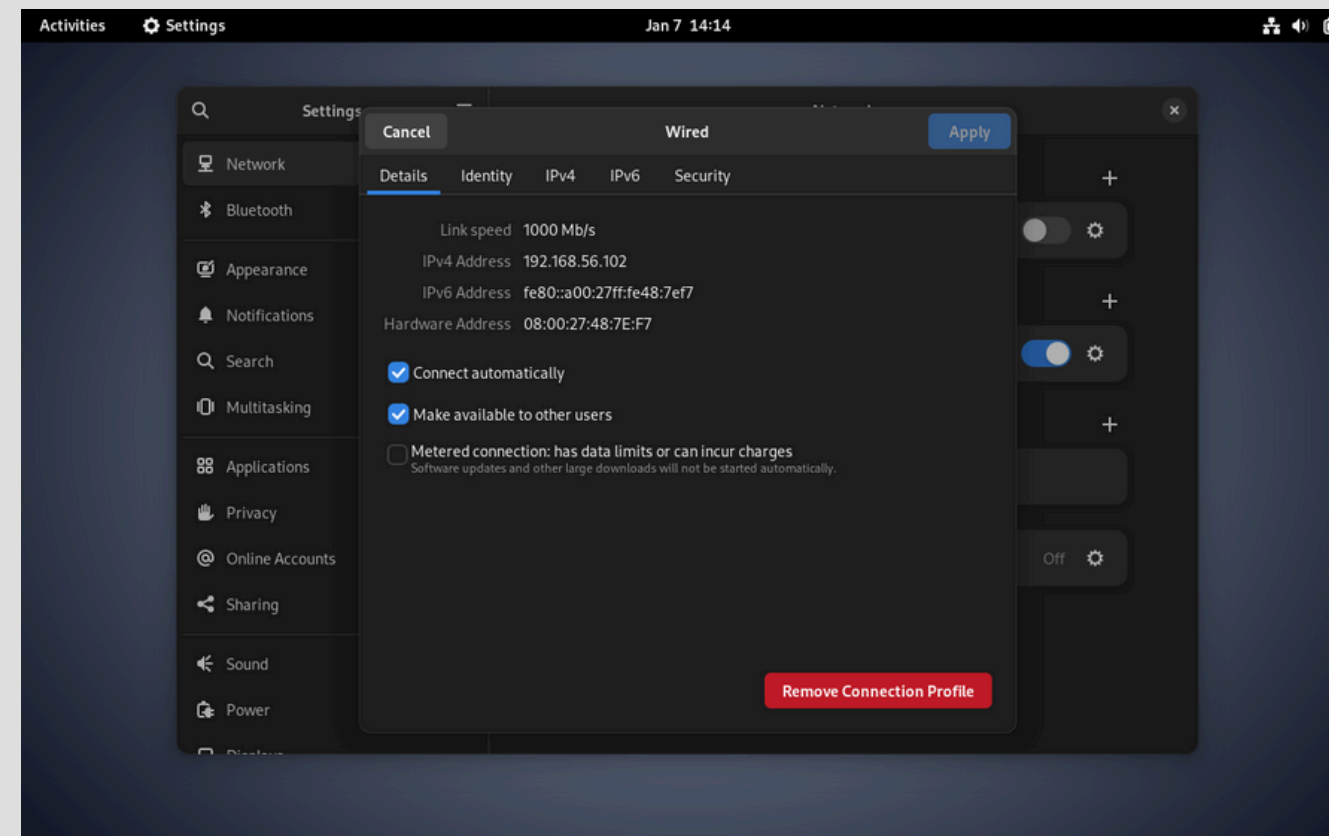
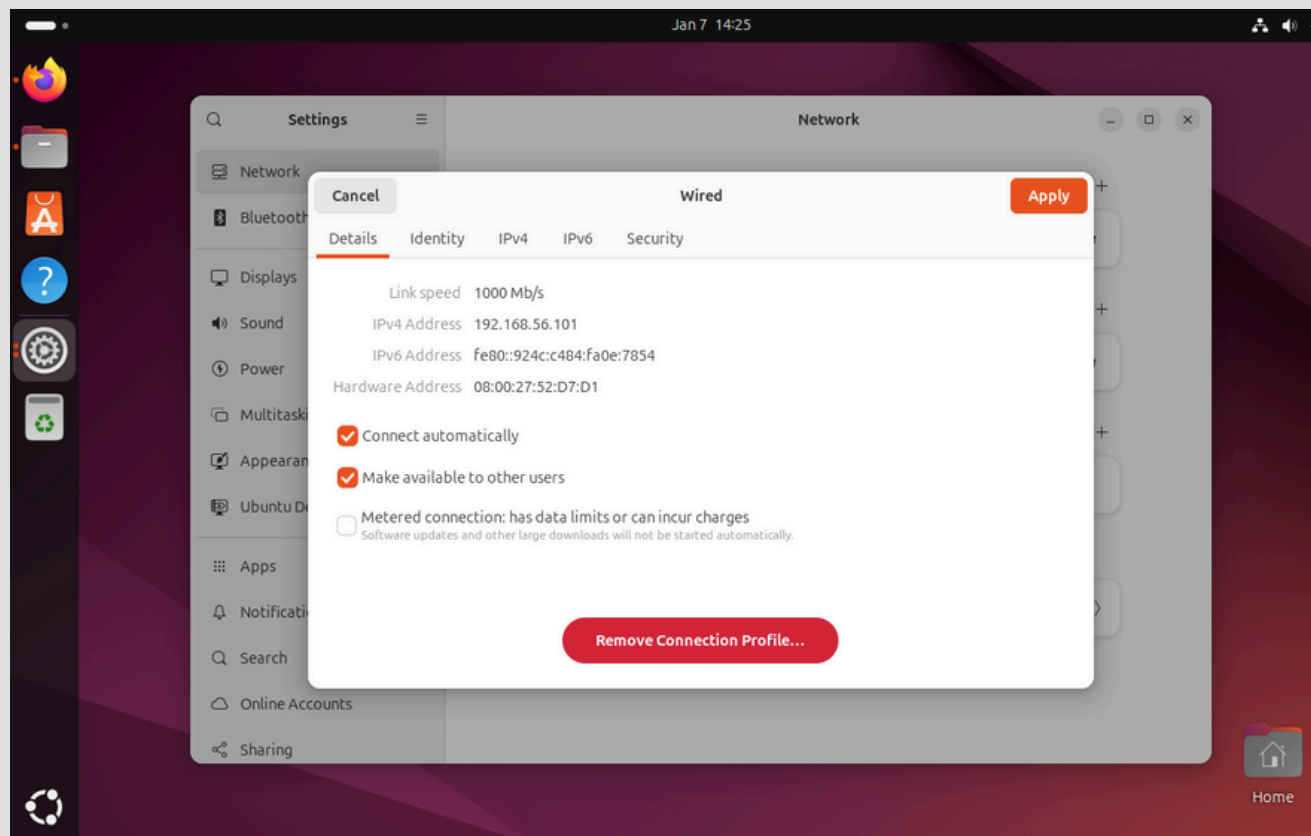
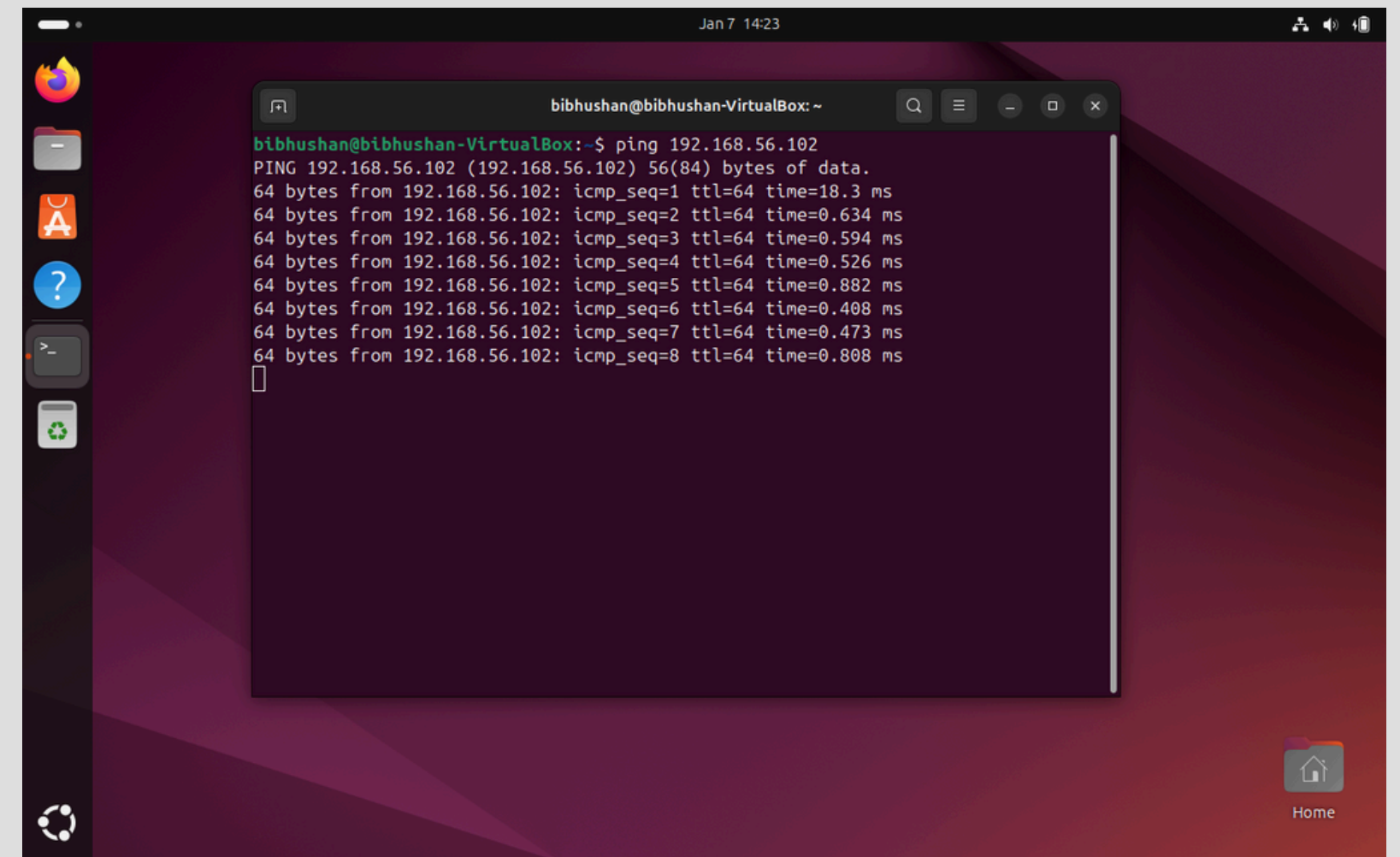
Task 1 - Add Firewall

- Set up firewall as the DHCP server.
- Configured intrusion protection for external networks.
- Tools Used: Firewall configuration software, existing network designs.



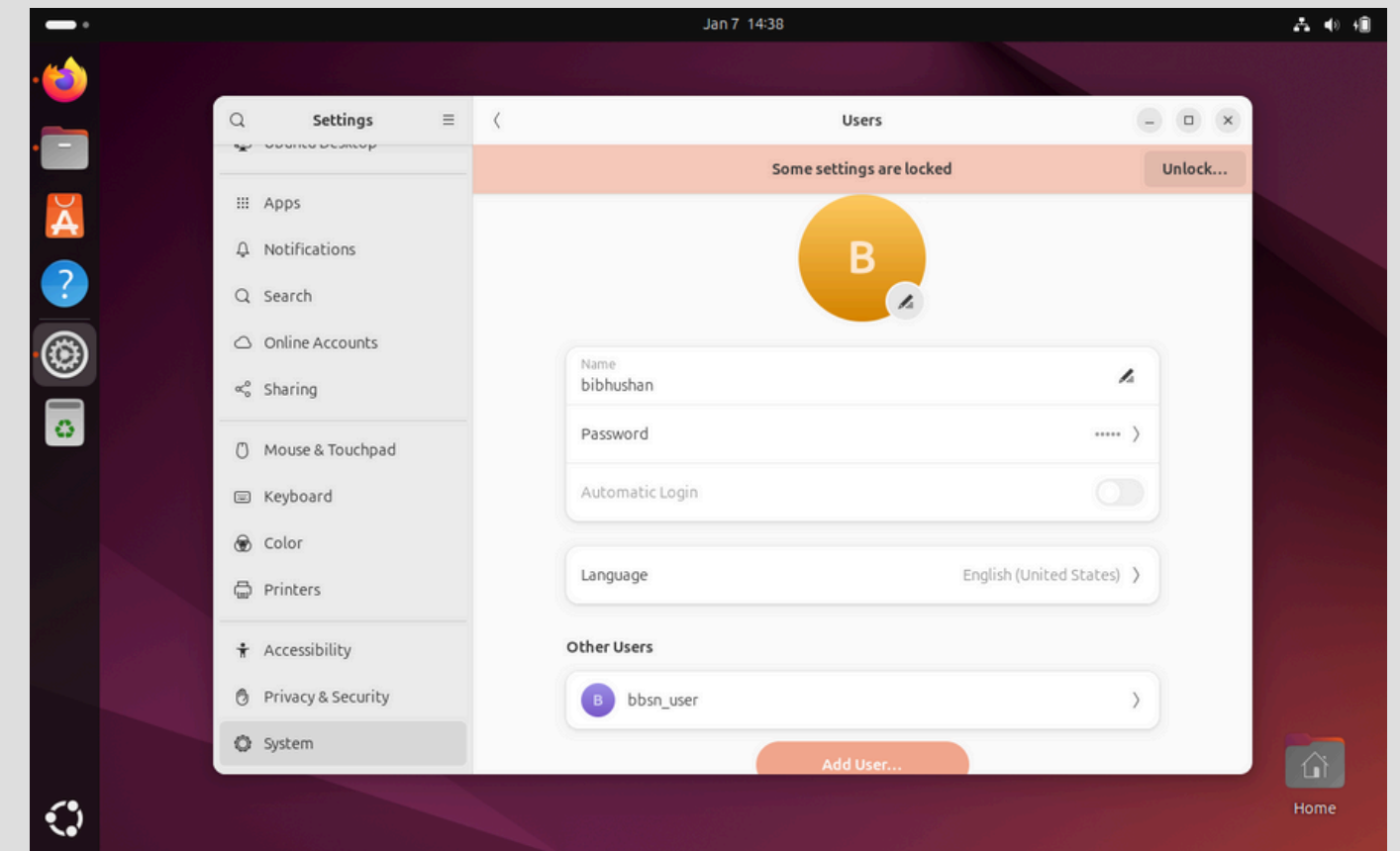
Task 2 - Virtual Machine Setup

- Installation Process:
 - Installed Oracle VirtualBox.
 - Created two virtual machines:
 - Ubuntu OS.
 - Debian OS.
 - Configured network links between the VMs.



Task 2 - Virtual Machine Setup

- Ubuntu OS Achievements:
 - Create admin and normal users.
 - Installed Java and Python.
 - *`sudo apt install open-jdk-17-jdk`*
 - Ran sample Java programs.
 - use `touch helloworld.java` to create file and `nano helloworld.java` to open code editor to write java code.
 - To compile java file use `javac helloworld.java` and to run `java helloworld`.



```
GNU nano 7.2 helloworld.java
public class helloworld{
    public static void main(String[] args){
        System.out.println("Hello World!!");
    }
}
```

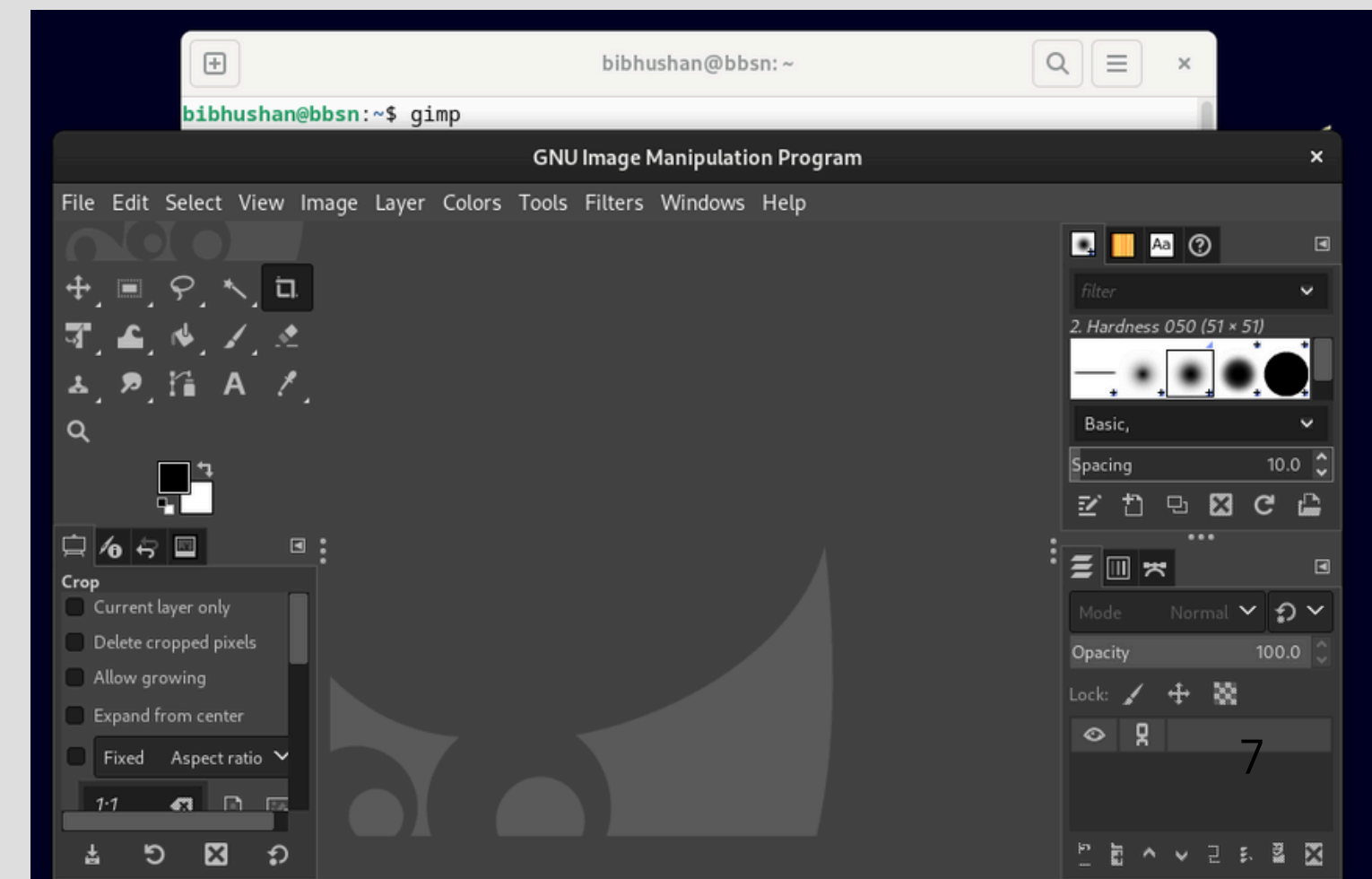
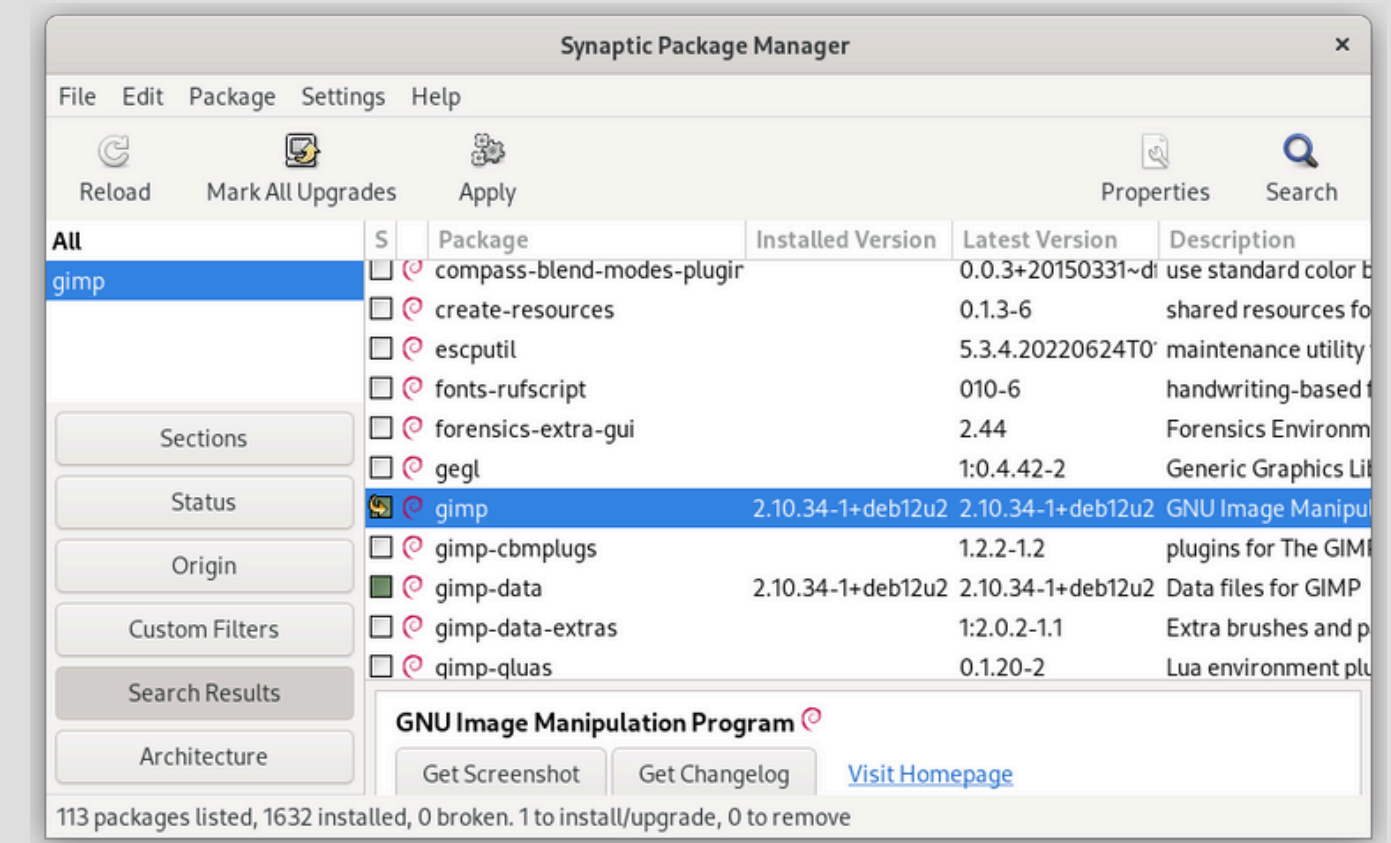
^G Help ^O Write Out ^W Where Is ^K Cut
^X Exit ^R Read File ^\ Replace ^U Paste

```
bibhushan@bibhushan-VirtualBox:~/Documents$ java helloworld
Hello World!!
bibhushan@bibhushan-VirtualBox:~/Documents$
```


Task 2 - Virtual Machine Setup

Debian OS Achievements:

- **Configured Sudo user**
 - `su -` (If already have sudo user)
 - `su - username` (replace username to create new sudo user)
- **Installed Synaptic Package Manager.**
 - `sudo apt install synaptic -y`
 - `sudo synaptic` (To open Synaptic)
- **Installed photo editor software using Synaptic .**
 - In Synaptic search bar search photo editor software eg:gimp.
 - Right-click on gimp and select Mark for Installation.
 - Click on the Apply button to start the installation.
- **Open Gimp photo editor software**
 - Type `gimp` in cmd



Task 3 - Docker Installation and Use

- Steps Completed:

1. Installed Docker on Ubuntu VM.

- `sudo apt install docker.io`
- `sudo systemctl enable docker`
- `sudo systemctl status docker` (Check its running or not)
- `sudo systemctl start docker.`

2. Created Docker Hub account from website

- `sudo docker login`. After this cmd you have to type your login details.

3. Pulled and ran the "Hello World" Docker image.

- `sudo docker pull hello-world` / `sudo docker image pull hello-world`
- `sudo docker run hello-world`

```
bibhushan@bibhushan-VirtualBox: ~/Documents
bibhushan@bibhushan-VirtualBox:~/Documents$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-01-07 14:22:45 CET; 26min ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
      Main PID: 1303 (dockerd)
        Tasks: 17
       Memory: 109.0M (peak: 113.4M)
          CPU: 3.014s
      CGroup: /system.slice/docker.service
              └─1303 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jan 07 14:22:44 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:44.081565704+01:00" level=info msg="Starting Docker daemon"
Jan 07 14:22:44 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:44.086919207+01:00" level=info msg="detected"
Jan 07 14:22:44 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:44.314574017+01:00" level=info msg="[graphdriver] using overlay2"
Jan 07 14:22:44 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:44.409712658+01:00" level=info msg="Loading containers"
Jan 07 14:22:45 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:45.082447983+01:00" level=info msg="Default daemon"
Jan 07 14:22:45 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:45.175607376+01:00" level=info msg="Loading containers"
Jan 07 14:22:45 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:45.237450801+01:00" level=info msg="Docker daemon"
Jan 07 14:22:45 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:45.238784251+01:00" level=info msg="Daemon has started"
Jan 07 14:22:45 bibhushan-VirtualBox systemd[1]: Started docker.service - Docker Application Container Engine.
Jan 07 14:22:45 bibhushan-VirtualBox dockerd[1303]: time="2025-01-07T14:22:45.377757431+01:00" level=info msg="API list"
lines 1-22/22 (END)
```

```
bibhushan@bibhushan-VirtualBox:~/Documents$ sudo docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
bibhushan@bibhushan-VirtualBox:~/Documents$ sudo docker run hello-world

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/

bibhushan@bibhushan-VirtualBox:~/Documents$
```


Task 3 - Docker Installation and Use

4. Pulled and ran Ubuntu Docker image.

- `sudo docker pull ubuntu`
- `sudo docker run -it ubuntu bash`
 - when you run this cmd , now you are running Ubuntu root from Docker.
 - Has limited command access.

```
bibhushan@bibhushan-VirtualBox:~/Documents$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
bibhushan@bibhushan-VirtualBox:~/Documents$ sudo docker run -it ubuntu bash
root@7ac576add13b:/#
```

5. Check running container.

- `sudo docker ps`

```
bibhushan@bibhushan-VirtualBox:~/Documents/myproject$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
```

6. Create your own ls in docker ubuntu

- `ls bin` (list the content of /bin)
- `cp /bin/ls /bin/my-ls` (Copy the ls binary to create a new command my-ls)
- `/bin/my-ls`(Execute the custom my-ls command:)

```
root@0863e22f020a: /
bibhushan@bibhushan-VirtualBox:~$ sudo docker run -it ubuntu bash
root@0863e22f020a:/# cp bin/ls /bin/my-ls
root@0863e22f020a:/# /bin/my-ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@0863e22f020a:/#
```

Task 4 - Java/Python Docker Container

- Setup Process:

- a. **Created directory in Ubuntu and Java file .**

- `mkdir JavaCode`
 - `cd JavaCode`
 - create new java file inside this new directory like we have created sample java file in Task 2.

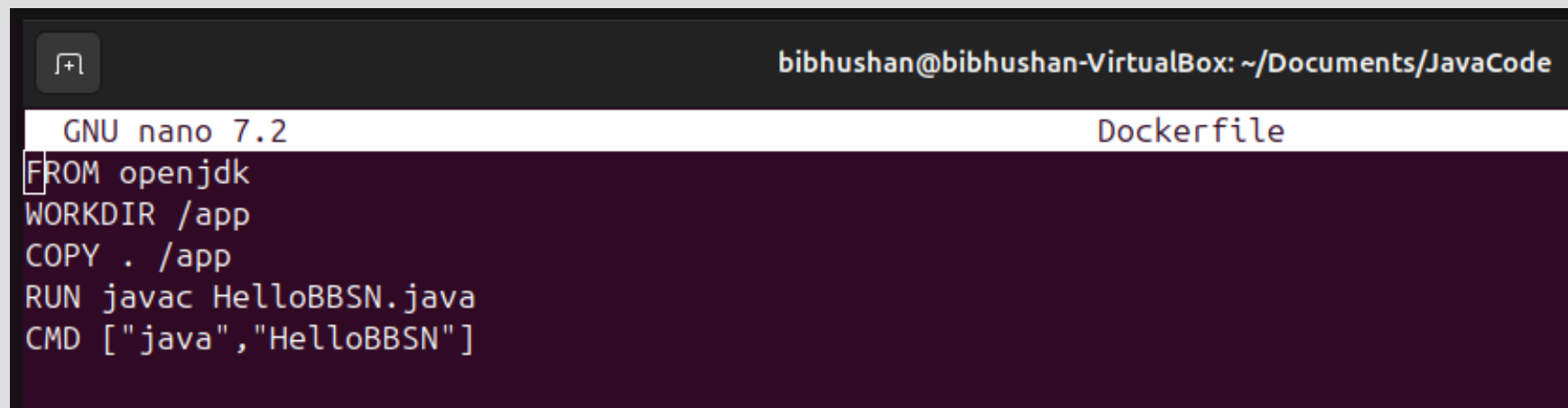
- b. **Create Dockerfile.**

- `cd Documents/JavaCode`
 - `touch Dockerfile`
 - `nano Dockerfile` (type cmd for Docker inside this file)



A screenshot of a terminal window showing the nano editor editing a file named `HelloBBSN.java`. The terminal title bar indicates the user is `bibhushan` on a `VirtualBox` machine, in the directory `~/Documents/JavaCode`. The editor shows the following Java code:

```
class HelloBBSN {  
    public static void main(String[] args) {  
        System.out.println("Hello!, Everyone Welcome to Docker Container from Java XD :) :)");  
    }  
}
```



A screenshot of a terminal window showing the nano editor editing a file named `Dockerfile`. The terminal title bar indicates the user is `bibhushan` on a `VirtualBox` machine, in the directory `~/Documents/JavaCode`. The editor shows the following Dockerfile content:

```
FROM openjdk  
WORKDIR /app  
COPY . /app  
RUN javac HelloBBSN.java  
CMD ["java","HelloBBSN"]
```

Task 4 - Java/Python Docker Container

c. Create Java Container

- `sudo docker build -t javaprogram °`
- `sudo docker image ls` (to check weather it is created or not)

d. Run Java Container

- `sudo docker run --name java1 javaprogram`
- `sudo docker ps -a` (to check all container id in Docker)
- `sudo docker run javaprogram`

• Outcome:

- Successfully executed programs within the container.

```
bibhushan@bibhushan-VirtualBox: ~/Documents
bibhushan@bibhushan-VirtualBox:~/Documents$ sudo docker image ls
REPOSITORY      TAG       IMAGE ID       CREATED        SIZE
my-flask-app    latest    d92e90e95f51   3 weeks ago    1.01GB
javaprogram     latest    afcd85d7bce0   3 weeks ago    470MB
ubuntu          latest    b1d9df8ab815   6 weeks ago    78.1MB
hello-world     latest    d2c94e258dcb   20 months ago  13.3kB
bibhushan@bibhushan-VirtualBox:~/Documents$
```

```
bibhushan@bibhushan-VirtualBox: ~/Documents
bibhushan@bibhushan-VirtualBox:~/Documents$ sudo docker run javaprogram
Hello!, Everyone Welcome to Docker Container from Java XD :) :)
bibhushan@bibhushan-VirtualBox:~/Documents$
```

Task 5 - Python Flask Application in Docker

a. Created myproject directory containing Python code and HTML template.

- Create myproject directory

- `mkdir myproject`
- `cd myproject/`

- Create python file

- `touch app.py`
- `nano app.py` (add code from instruction pdf)

- Create templates dir in myproject and HTML file inside templates

- `mkdir templates`
- `cd templates`
- `touch index.html`
- `nano index.html` (add code from pdf)

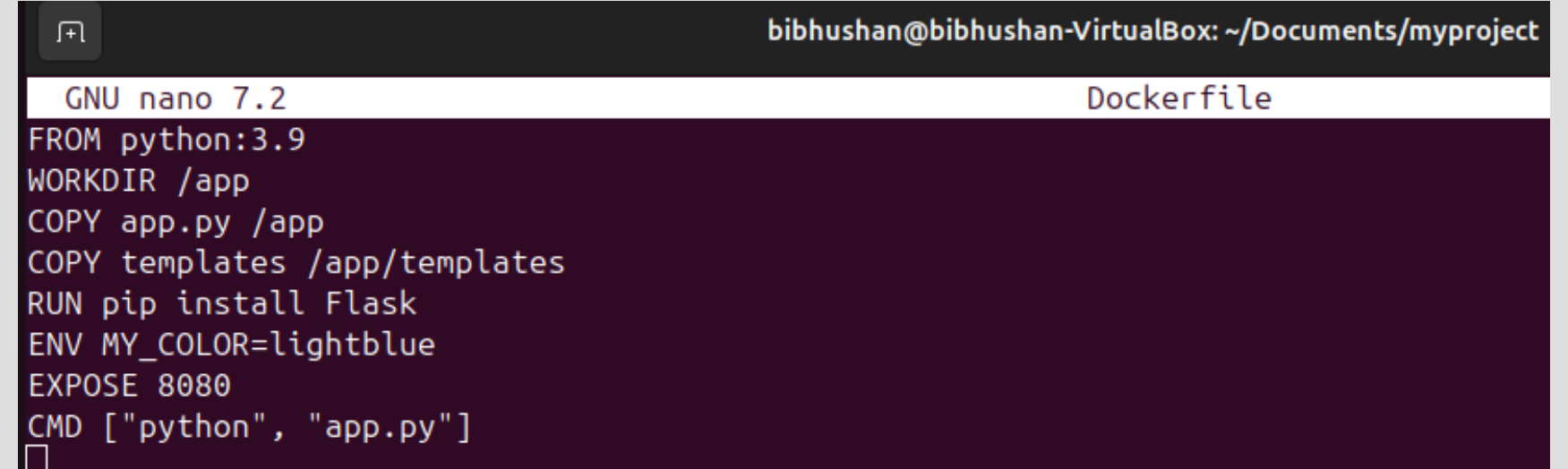
```
bibhushan@bibhushan-VirtualBox: ~/Documents/myproject
bibhushan@bibhushan-VirtualBox:~/Documents/myproject$ ls
app.py  Dockerfile  templates
bibhushan@bibhushan-VirtualBox:~/Documents/myproject$
```

```
bibhushan@bibhushan-VirtualBox: ~/Documents/myproject/templates
bibhushan@bibhushan-VirtualBox:~/Documents/myproject$ cd templates/
bibhushan@bibhushan-VirtualBox:~/Documents/myproject/templates$ ls
index.html
bibhushan@bibhushan-VirtualBox:~/Documents/myproject/templates$
```


Task 5 - Python Flask Application in Docker

b. Configured Dockerfile with Flask dependencies.

- Create docker file in myproject directory
 - *touch dockerfile*
 - *nano dockerfile*
 - Inside dockerfile write commands and Flask dependencies

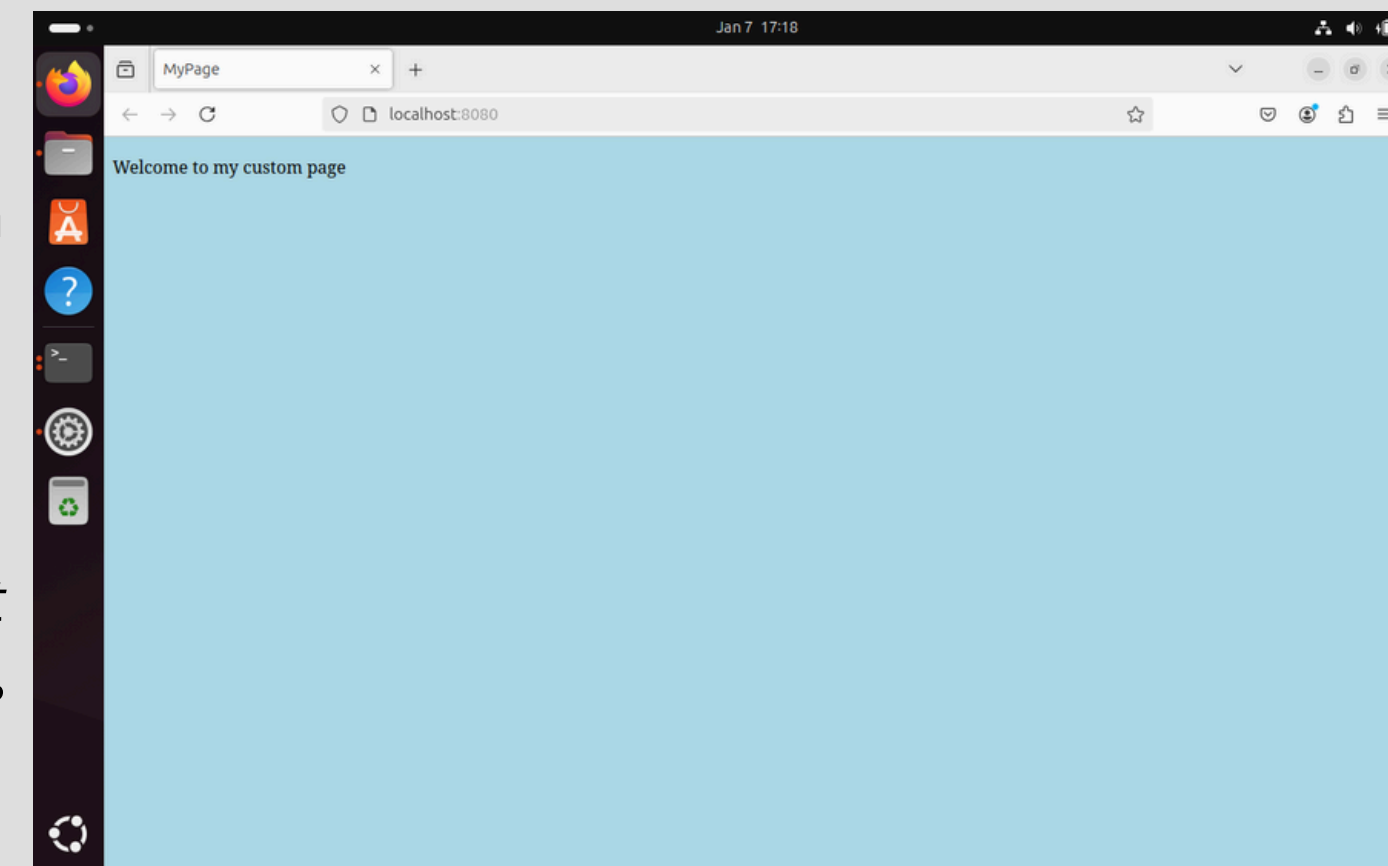


The screenshot shows a terminal window with the title bar 'bibhushan@bibhushan-VirtualBox: ~/Documents/myproject'. The editor is GNU nano 7.2, editing a file named 'Dockerfile'. The content of the file is as follows:

```
FROM python:3.9
WORKDIR /app
COPY app.py /app
COPY templates /app/templates
RUN pip install Flask
ENV MY_COLOR=lightblue
EXPOSE 8080
CMD ["python", "app.py"]
```

c. Built and ran the container.

- Build Python container
 - *sudo docker build -t my-flask-app .* (my-flask-app is repo name you can modify of your choice)
- Run container
 - *sudo docker run -d -p 8080:8080 --name my-python-app my-flask-app*
 - **where** *-d* : runs the container indetached mode , *-p 8080:8080* : maps port 8080 on host to port 8080 in container and *-- name my-python-app* : name of the running container
- Now open <http://localhost:8080> in your Virtual OS



Conclusion

- Successfully implemented all tasks, from firewall configuration to Docker deployment.
- Enhanced network security with a DMZ and firewall as a DHCP server.
- Installed and configured Ubuntu and Debian virtual machines, ensuring seamless communication.
- Developed and ran Java and Python programs via the command line.
- Integrated Docker to deploy and run containers for both Java and Python applications.
- Demonstrated practical skills in Python and Flask by deploying a lightweight web app.
- Achieved project goals with a focus on scalability, security, and modern IT practices.

Q&A

- Thank you for your attention!
- Questions and discussions are welcome.