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.

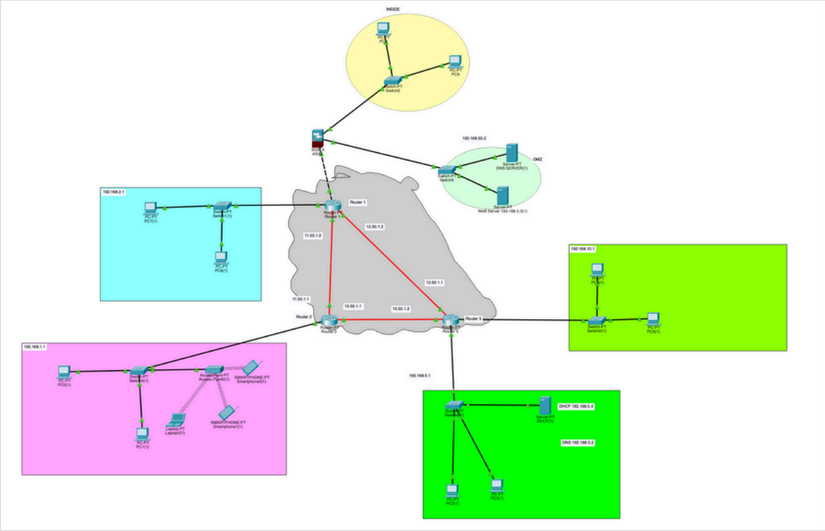
Network security and configurations.

Dockerized applications.

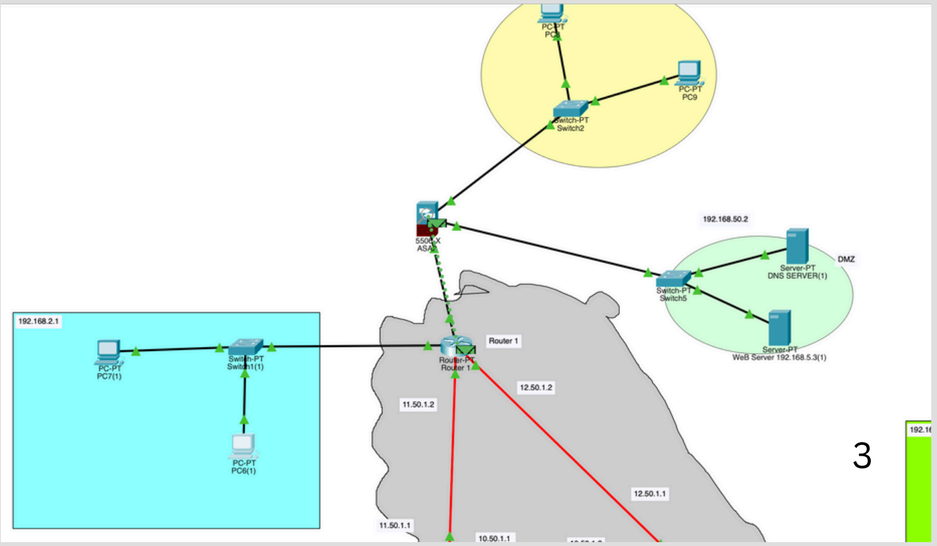
Task-specific achievements in IT environments.

Task 1 - Add Firewall

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

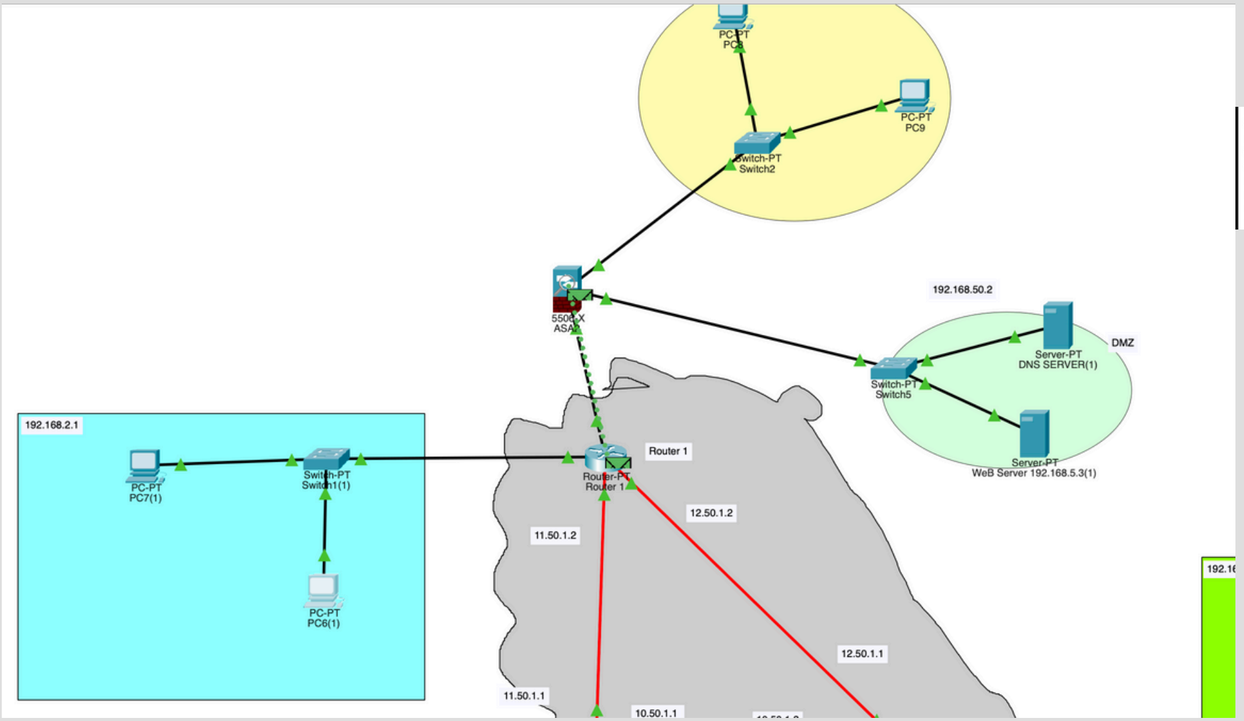


```
!
dhcpd dns 192.168.50.2
!
dhcpd address 192.168.100.10-192.168.100.100 inside
dhcpd enable inside
!
!
!
!
```



Task 1 - Add Firewall

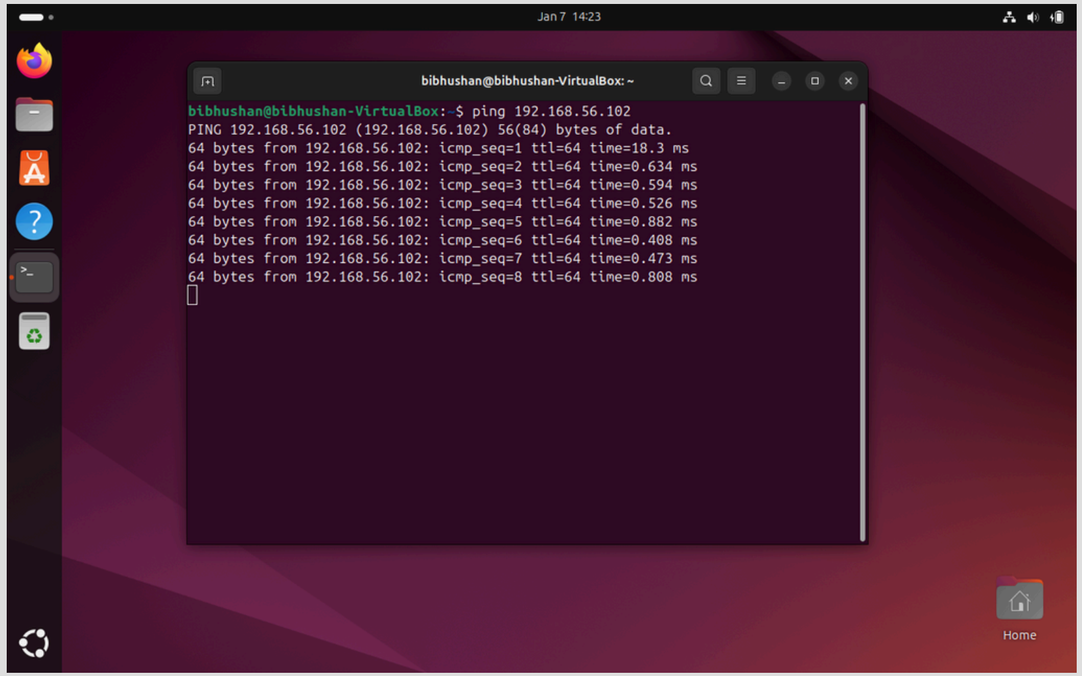
- Set up firewall as the DHCP server.
- Configured intrusion protection for external networks.
- Tools Used: Firewall configuration software, existing network designs.

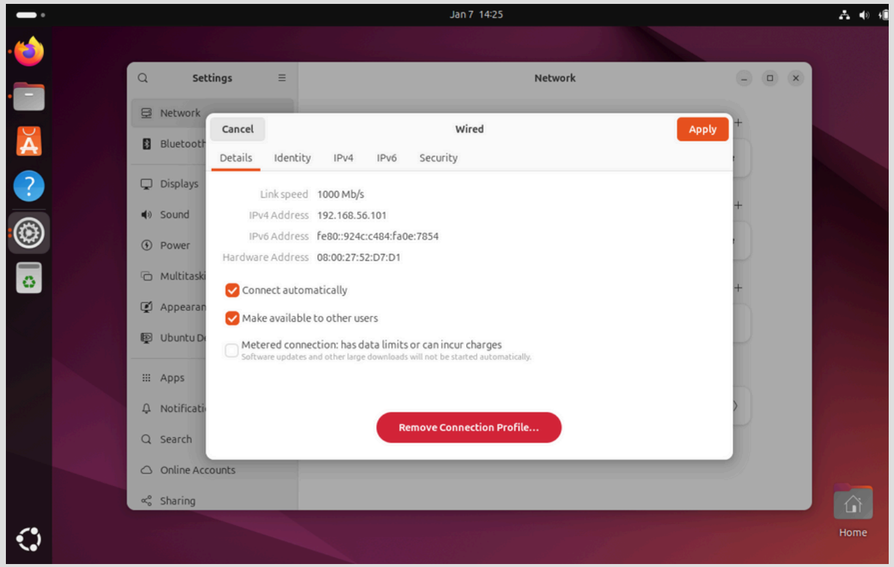


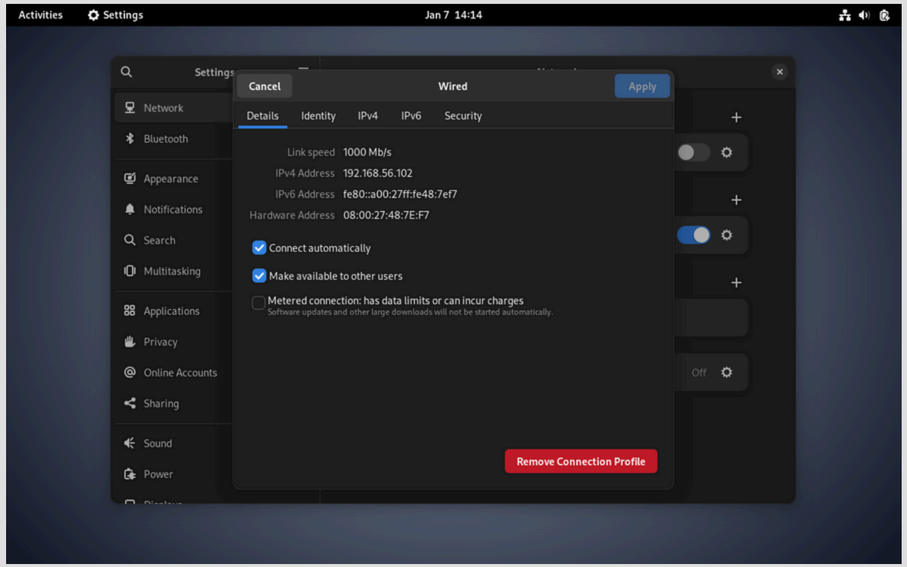

```
!
dhcpd dns 192.168.50.2
!
dhcpd address 192.168.100.10-192.168.100.100 inside
dhcpd enable inside
!
!
!
!
```

Task 2 - Virtual Machine Setup

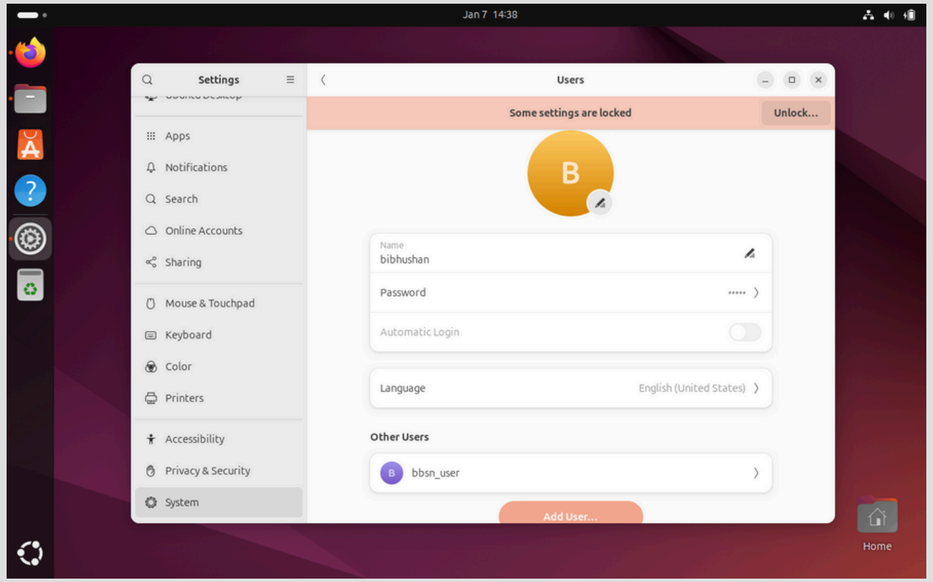
- Installation Process:
  - Installed Oracle VirtualBox.
  - Created two virtual machines:
    - Ubuntu OS.
    - Debian OS.
  - Configured network links between the VMs.



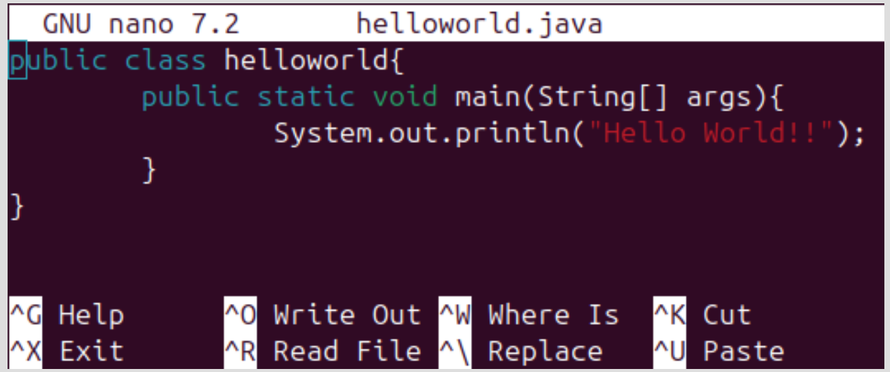


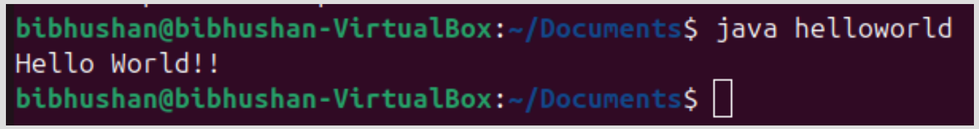


Task 2 - Virtual Machine Setup



- 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.

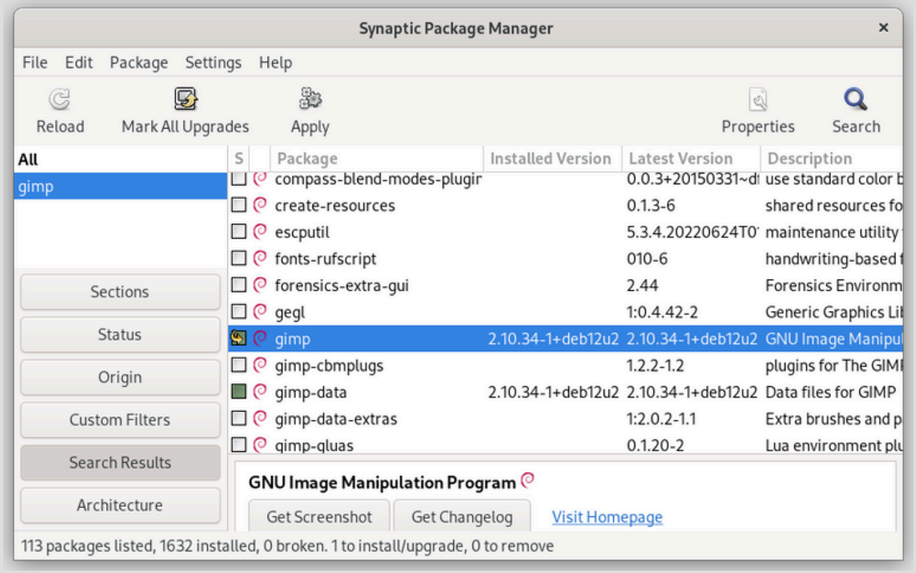




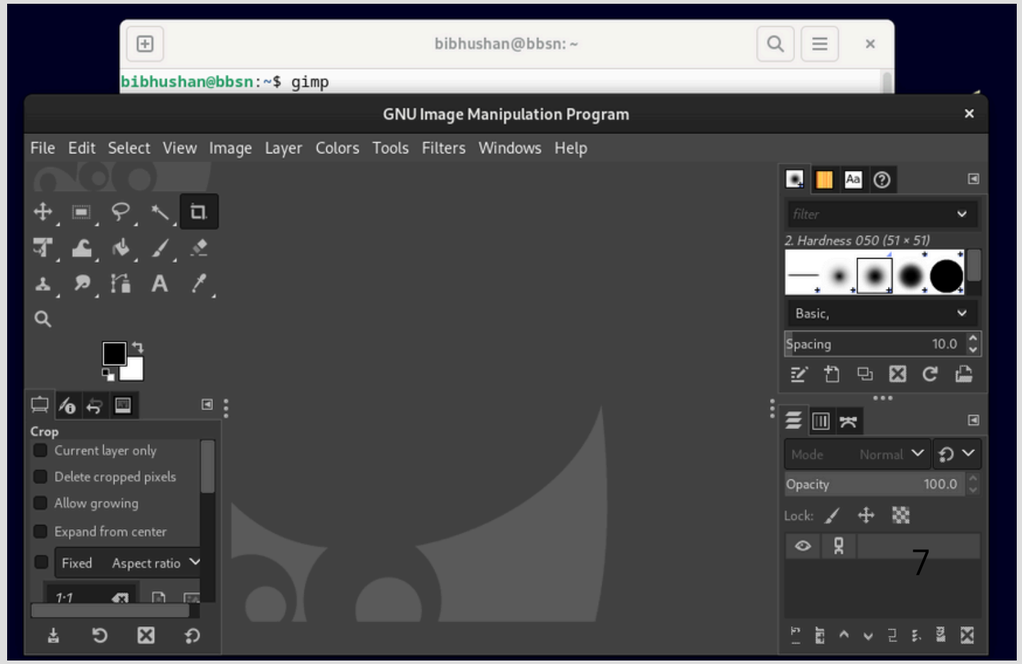
Task 2 - Virtual Machine Setup

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**)



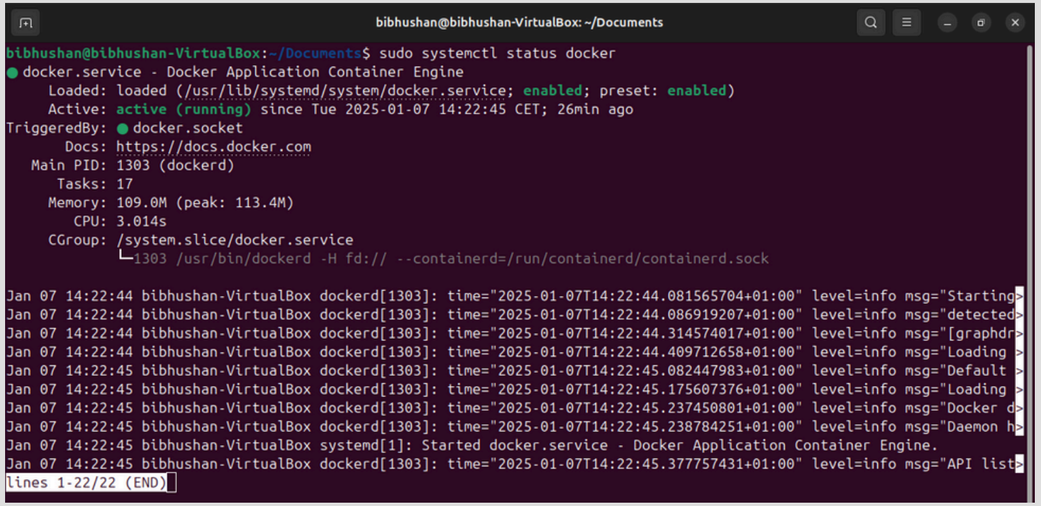
- **Installed photo editor software using Synaptic .**
  - In Synaptic search bar search photo editor software eg:gimp.
  - Right-click on gimp and select Mark for Installation.
  - Click on the Apply button to start the installation.
- **Open Gimp photo editor software**
  - Type *gimp* in cmd

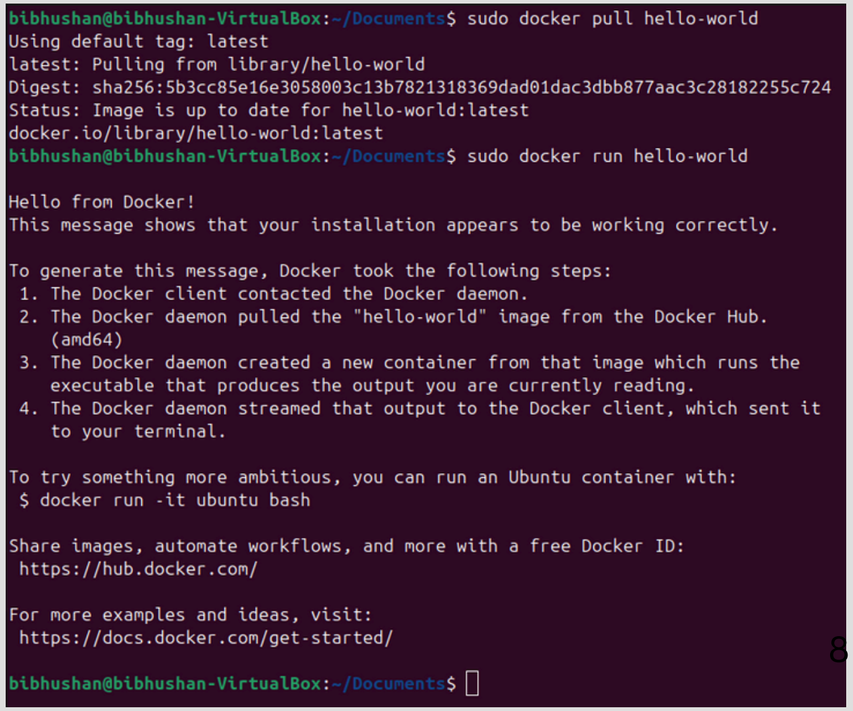


Task 3 - Docker Installation and Use

- **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*





Task 3 - Docker Installation and Use

```
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:/#
```

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.
5. **Check running container.**
   - *sudo docker ps*

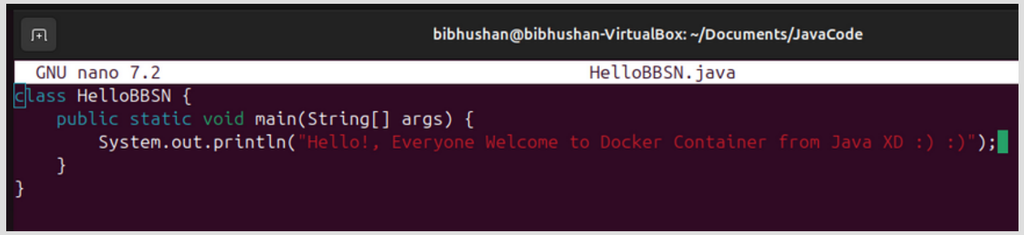
```
bibhushan@bibhushan-VirtualBox:~/Documents/myproject$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

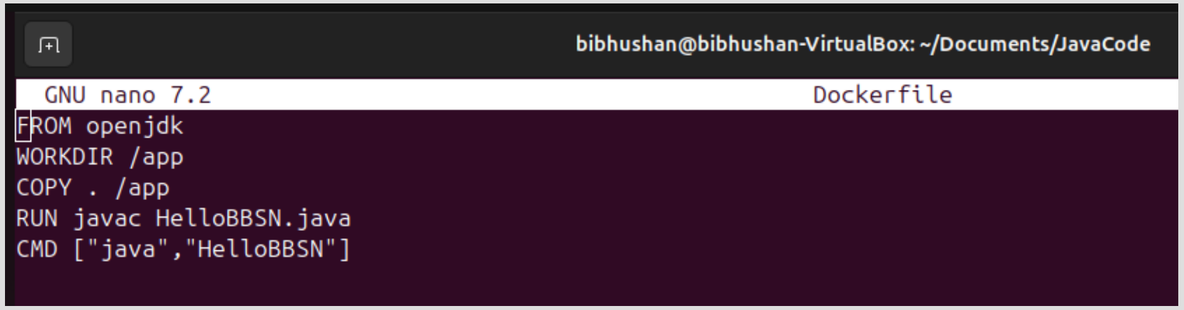
6. **Create your own ls in docker ubuntu**
   - *ls bin* (list the content of /bin)
   - *cp /bin/ls /bin/my-ls* (Copy the ls binary to create a new command my-ls)
   - */bin/my-ls*(Execute the custom my-ls command:)

```
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)





Task 4 - Java/Python Docker Container

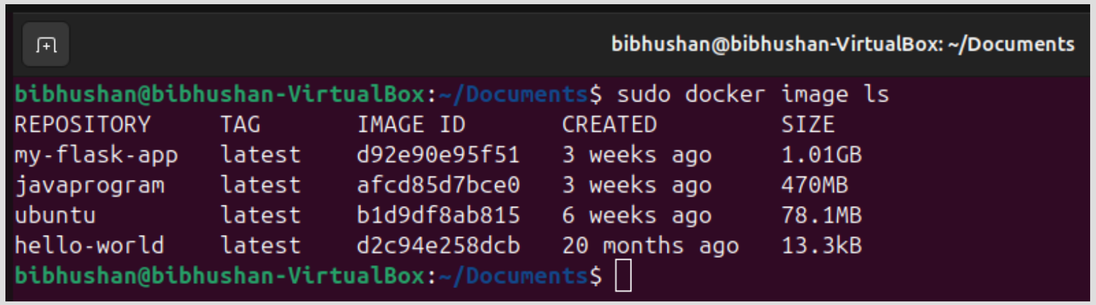
c. **Create Java Container**

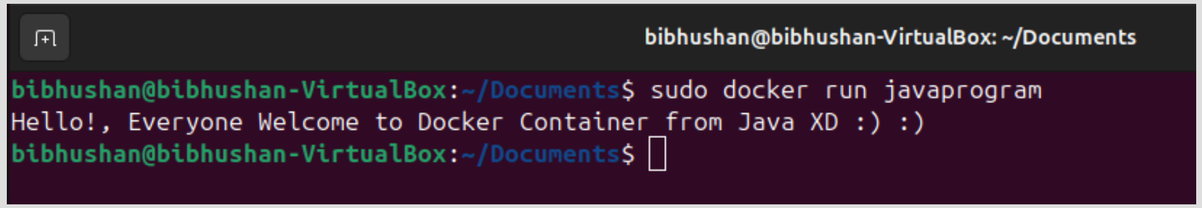
- *sudo docker build -t javaprogram °*
- *sudo docker image ls* (to check weather it is created or not)

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.

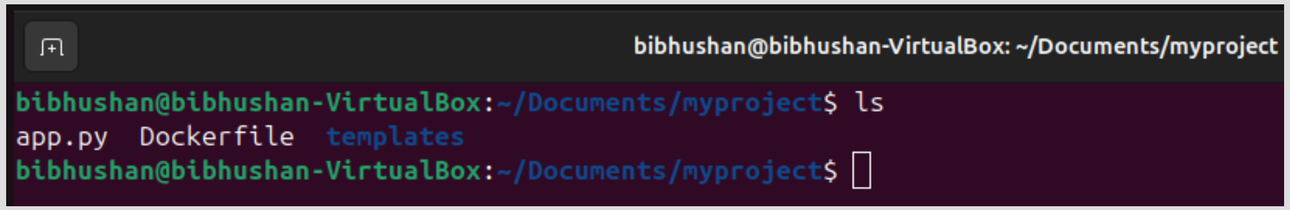




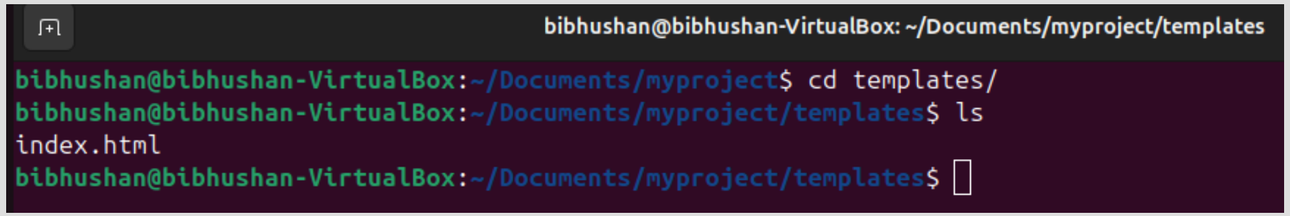
Task 5 - Python Flask Application in Docker

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 templetes dir in myproject and HTML file inside templetes**
  - *mkdir templetes*
  - *cd templetes*
  - *touch index.html*
  - *nano index.html*(add code from pdf)



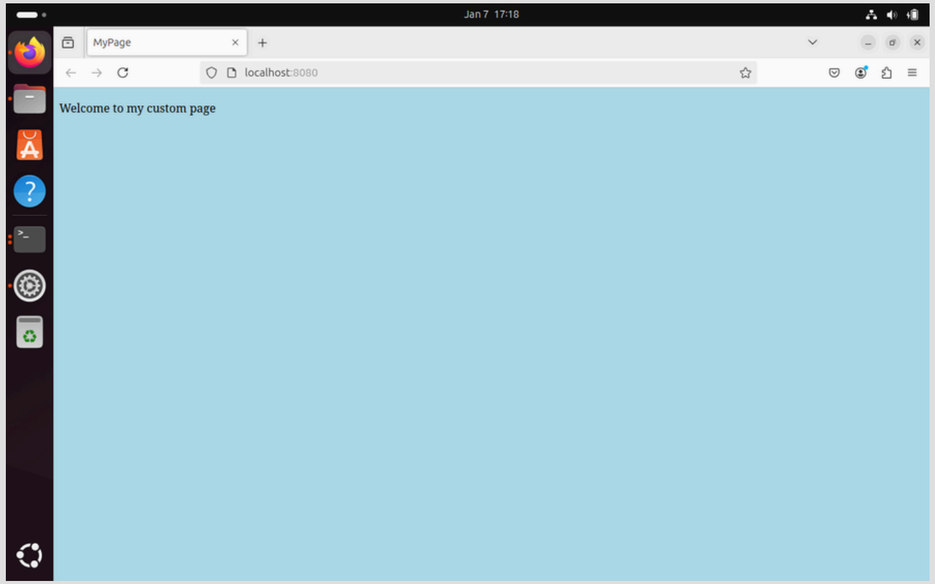
Task 5 - Python Flask Application in Docker

b. **Configured Dockerfile with Flask dependencies.**

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

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.