IT Platforms final project report

Kateryna Sadovska

Nazar Zhanabergenov

Karim Saliev

Mark Shafarenko

Ravshanbek Musaev

Task 1: Install and Uses Linux Operating system.

Initial operating system: Windows 11.

1. **To install Linux Operating system on Windows, we need a Virtual Machine.**
2. **Downloaded and installed Oracle Virtual Machine from https://www.virtualbox.org/ website.**

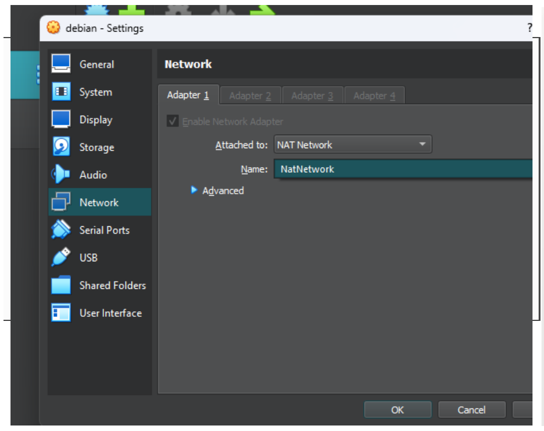
a) In Oracle VM created virtual machine “Ubuntu”. Downloaded Ubuntu OS version 22.04.3 and installed it virtual machine “Ubuntu” inside Oracle VM.

b) In Oracle VM created virtual machine “Debian”. Downloaded Debian OS version 12.4 and installed it virtual machine “Debian” inside Oracle VM.

c) To run the virtual machine, we select necessary VM in Oracle VM and “Run” button is pressed. We ran both “Ubuntu” and “Debian” virtual machines at the same time.

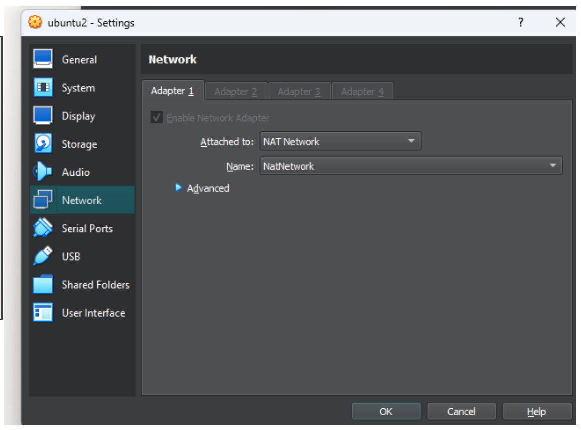
d) To configure and connect both virtual operating systems, we need to configure NAT Network in Oracle VM tools.
Properties of NAT Network configuration:
Name: NatNetwork
IPv4 Prefix: 10.0.2.0 /24
DHCP enabled

After that we need to configure NAT Network for each virtual operating system. To do that in the Settings -> Network. There we set Attached to: NAT Network; with Name of our NAT Network “NatNetwork”. For both



Attached to: NAT Network

Name: NatNetwork

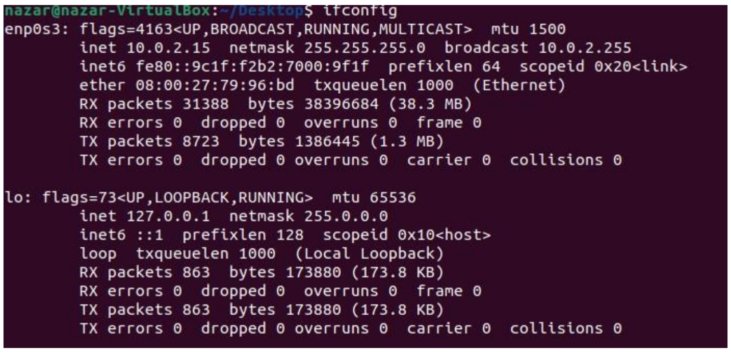


Attached to: NAT Network

Name: NatNetwork

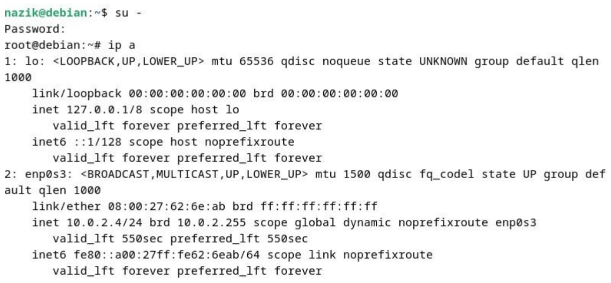
virtual operating systems this setting is the same.

To check the link between virtual operating systems we need to ping them.



```
nazar@nazar-VirtualBox:~/Desktop$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::9c1f:f2b2:7000:9f1f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:79:96:bd  txqueuelen 1000  (Ethernet)
        RX packets 31388  bytes 38396684 (38.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8723  bytes 1386445 (1.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 863  bytes 173880 (173.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 863  bytes 173880 (173.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

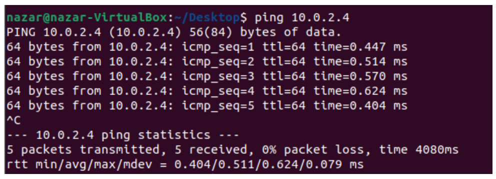
In Ubuntu virtual operating system with help of command ‘*ifconfig*’ we found that IP address is: 10.0.2.15



```
nazik@debian:~$ su -
Password:
root@debian:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group def
ault qlen 1000
    link/ether 08:00:27:62:6e:ab brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 550sec preferred_lft 550sec
    inet6 fe80::a00:27ff:fe62:6eab/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

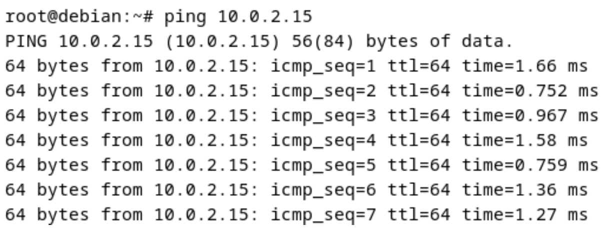
In Debian virtual operating system with help of command *‘ip a’* we found that IP address is: 10.0.2.4

Now to check connection, we ping one virtual operating system from another one.



```
nazar@nazar-VirtualBox:~/Desktop$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.447 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.514 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.570 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.624 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.404 ms
^C
--- 10.0.2.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4080ms
rtt min/avg/max/mdev = 0.404/0.511/0.624/0.079 ms
```

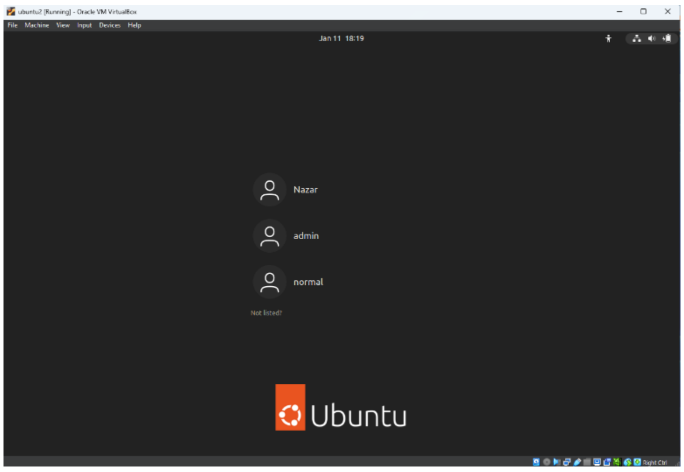
Successful ping test from Ubuntu to Debian virtual operating system.



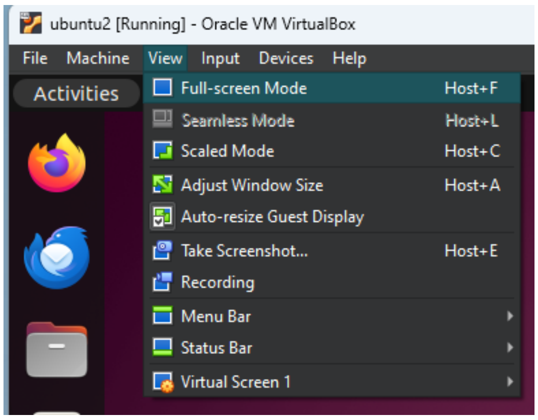
```
root@debian:~# ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=1.66 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.752 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.967 ms
64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=1.58 ms
64 bytes from 10.0.2.15: icmp_seq=5 ttl=64 time=0.759 ms
64 bytes from 10.0.2.15: icmp_seq=6 ttl=64 time=1.36 ms
64 bytes from 10.0.2.15: icmp_seq=7 ttl=64 time=1.27 ms
```

Successful ping test from Debian to Ubuntu virtual operating system.

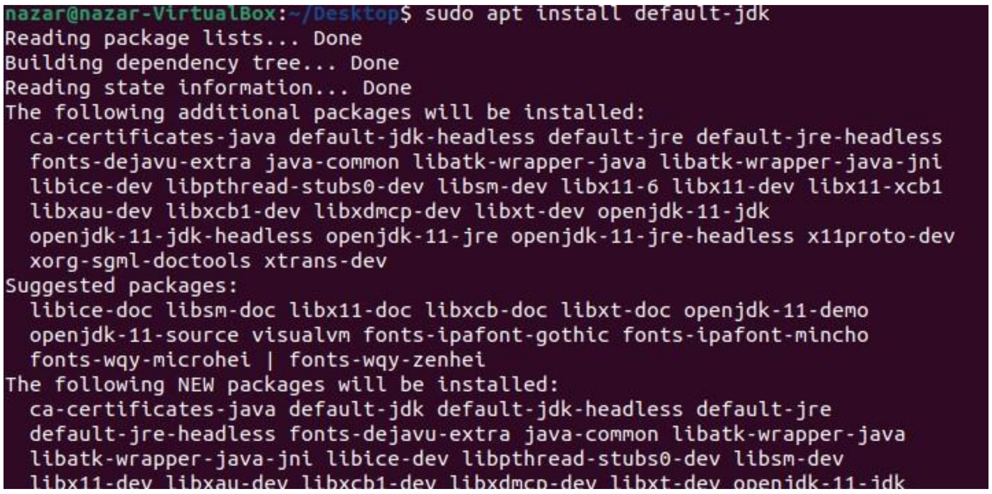
3. **In Ubuntu virtual OS:**
   a) We created 2 users: admin and normal user.



b) To turn on the full screen mode in Ubuntu virtual OS, we need to press "*Host key + F*", where Host key in our configuration is "Right Ctrl".

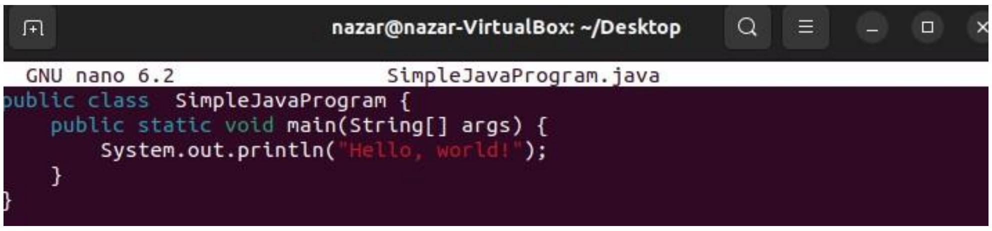

ubuntu2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Full-screen Mode Host+F
Seamless Mode Host+L
Scaled Mode Host+C
Adjust Window Size Host+A
Auto-resize Guest Display
Take Screenshot... Host+E
Recording
Menu Bar
Status Bar
Virtual Screen 1

c) To install Java VM in Ubuntu virtual OS from command line, we need to open the command line and enter the command "*sudo apt install default-jdk*". After that installation process will be done automatically.



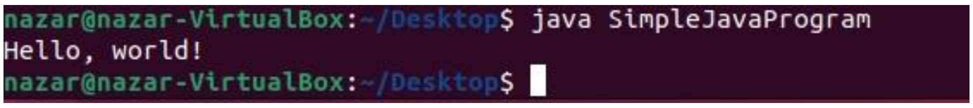
```
nazar@nazar-VirtualBox:~/Desktop$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jdk-headless default-jre default-jre-headless
  fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni
  libice-dev libpthread-stubs0-dev libsm-dev libx11-6 libx11-dev libx11-xcb1
  libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11proto-dev
  xorg-sgml-doctools xtrans-dev
Suggested packages:
  libice-doc libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-11-demo
  openjdk-11-source visualvm fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei | fonts-wqy-zenhei
The following NEW packages will be installed:
  ca-certificates-java default-jdk default-jdk-headless default-jre
  default-jre-headless fonts-dejavu-extra java-common libatk-wrapper-java
  libatk-wrapper-java-jni libice-dev libpthread-stubs0-dev libsm-dev
  libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk
```

d) We created small Java program called SimpleJavaProgram, which prints "Hello, World!".



```
nazar@nazar-VirtualBox: ~/Desktop
  GNU nano 6.2                    SimpleJavaProgram.java
public class  SimpleJavaProgram {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

e) Now we ran this small Java program from the command line of Ubuntu.

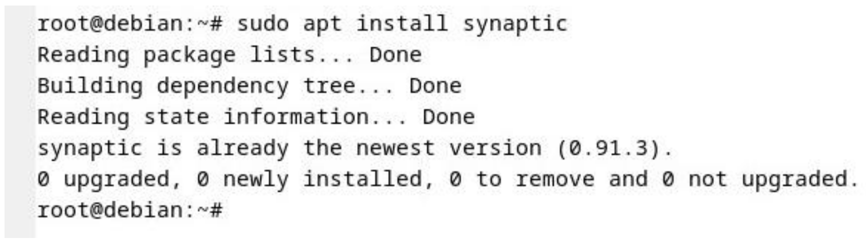


```
nazar@nazar-VirtualBox:~/Desktop$ java SimpleJavaProgram
Hello, world!
nazar@nazar-VirtualBox:~/Desktop$ 
```

4. In Debian virtual OS:

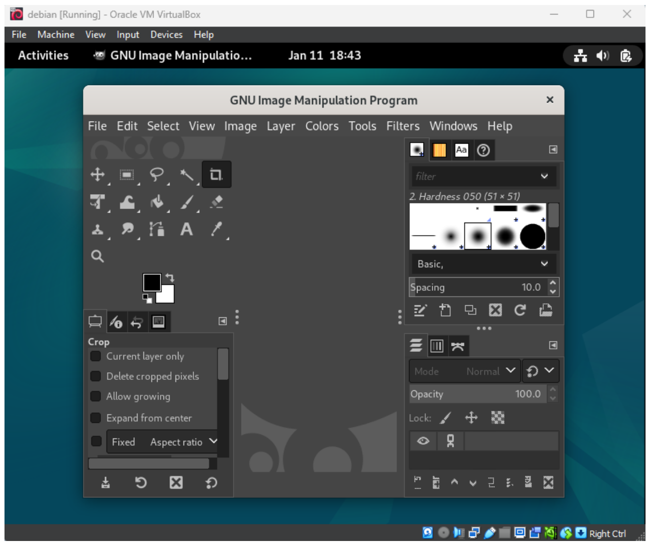
a) To configure Sudo user, first we created a new user in Debian using command "*adduser nazik*", added the password. Then with command "*usermod -aG sudo nazik*" we added this new user to the Sudo group. To switch to this user, we used command "*su nazik*".

b) To install Synaptic Package Manager in Debian virtual OS, we used the command “*sudo apt install synaptic*”.



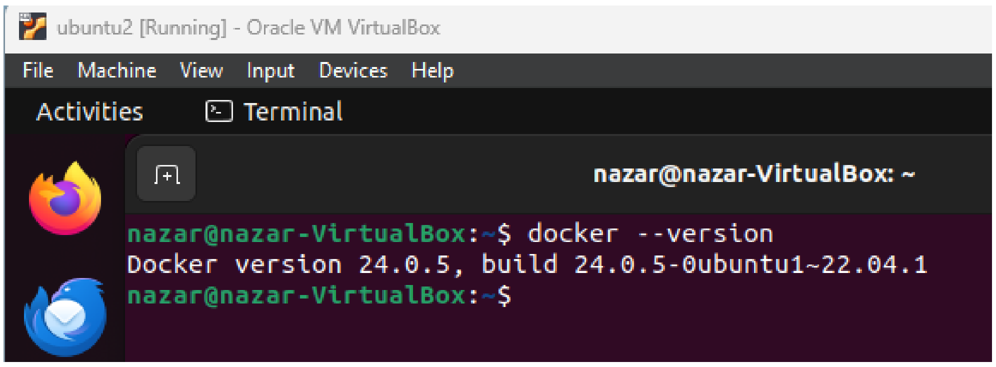
```
root@debian:~# sudo apt install synaptic
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
synaptic is already the newest version (0.91.3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@debian:~#
```

c) Using the Synaptic Package Manager, we installed “GNU Image manipulation program” photo editor. To do that, we searched this photo editor, selected it for installation and finally pressed “Apply” button.


debian [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
GNU Image Manipulatio...
Jan 11 18:43
GNU Image Manipulation Program
File Edit Select View Image Layer Colors Tools Filters Windows Help
2. Hardness 050 (51 × 51)
Basic,
Spacing 10.0
Crop
Current layer only
Delete cropped pixels
Allow growing
Expand from center
Fixed Aspect ratio
Mode Normal
Opacity 100.0
Lock:
Right Ctrl

Task 2: Install and Uses Docker in Ubuntu operating system.

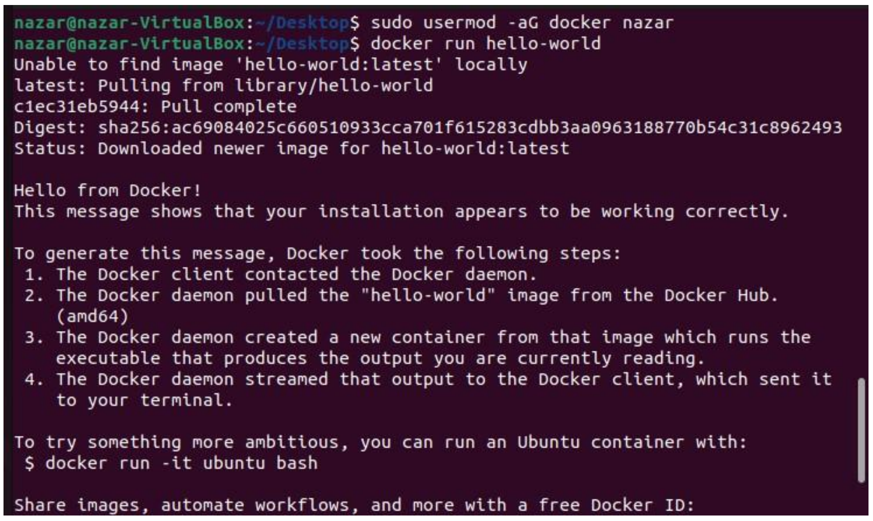
1. **To install Docker in Ubuntu virtual OS we used command line:**
   sudo apt-get update – to check that system is up-to-date
   sudo apt install docker.io
   sudo snap install docker – at this point Docker must be installed
   docker -version – check version and confirm that Docker is installed


ubuntu2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
nazar@nazar-VirtualBox: ~

```
nazar@nazar-VirtualBox:~$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
nazar@nazar-VirtualBox:~$
```

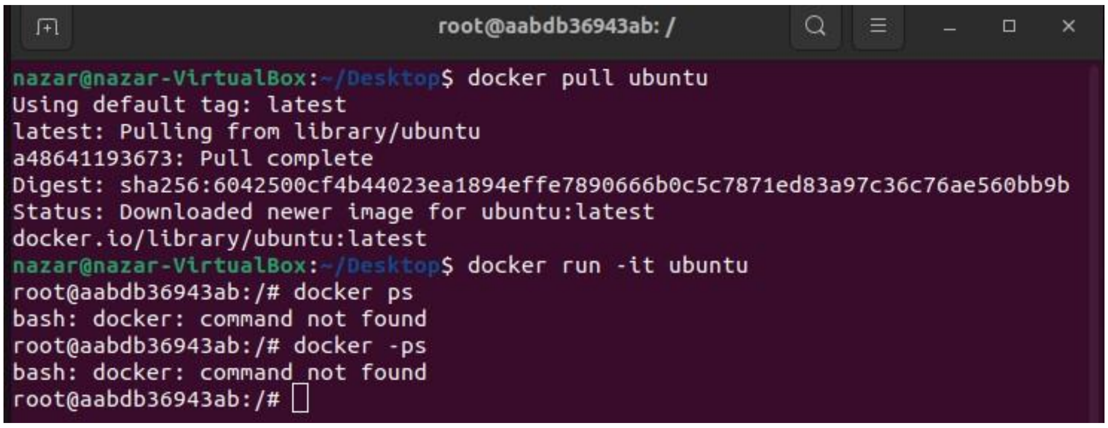
2. **We created a Docker account in Docker hub “jolygolden” for our project.**

3. **To pull “hello world” image from Docker Hub**, firstly we ran Docker with command “*sudo systemctl start docker*”. Then we set our user with command “*sudo usermod -aG docker nazar*”. Finally with command “*docker run hello-world*” we pulled the “hello world” image from Docker hub.



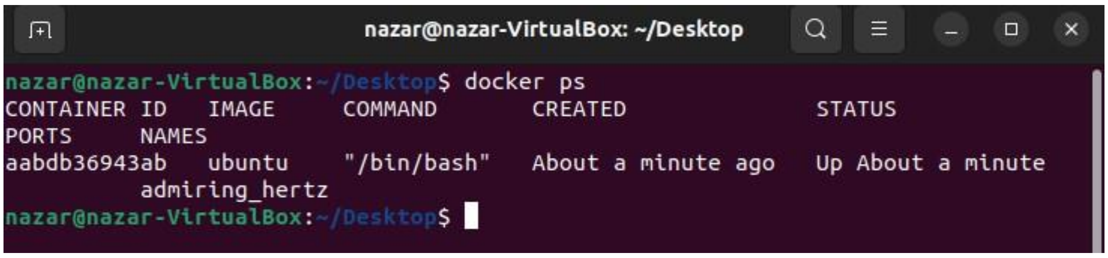
```
nazar@nazar-VirtualBox:~/Desktop$ sudo usermod -aG docker nazar
nazar@nazar-VirtualBox:~/Desktop$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:ac69084025c660510933cca701f615283cdbb3aa0963188770b54c31c8962493
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:
```

4. **To pull Ubuntu Docker image from Docker hub**, we used command “*docker pull ubuntu*”. Then with command “*docker run -it ubuntu*” we ran the image.



```
nazar@nazar-VirtualBox:~/Desktop$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
a48641193673: Pull complete
Digest: sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
nazar@nazar-VirtualBox:~/Desktop$ docker run -it ubuntu
root@aabdb36943ab:/# docker ps
bash: docker: command not found
root@aabdb36943ab:/# docker -ps
bash: docker: command not found
root@aabdb36943ab:/# 
```

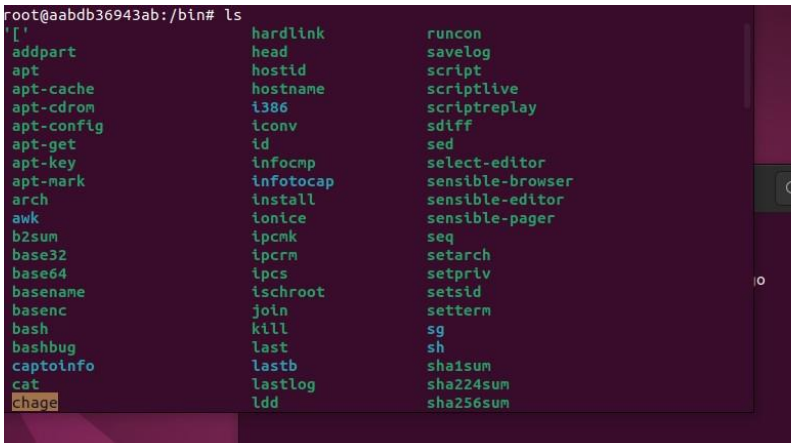
5. **We checked running containers in Docker** using command “*docker ps*”. It gives the list of running containers.



```
nazar@nazar-VirtualBox:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED              STATUS
PORTS     NAMES
aabdb36943ab   ubuntu    "/bin/bash"   About a minute ago   Up About a minute
          admiring_hertz
nazar@nazar-VirtualBox:~/Desktop$ 
```

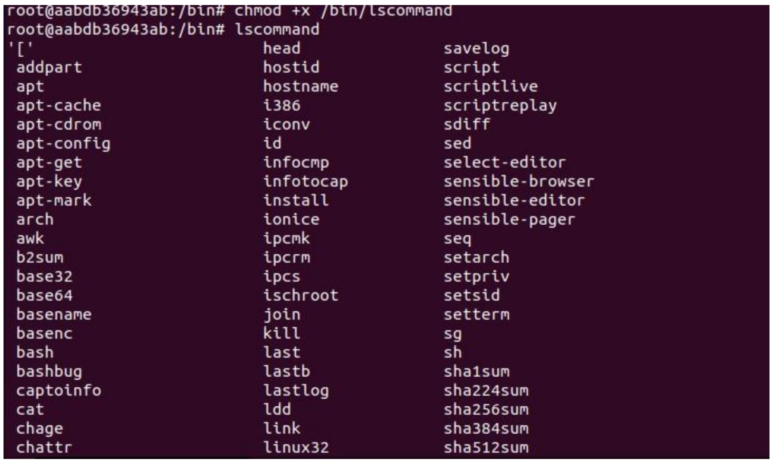
6. **To run the batch in the Ubuntu container**, firstly we created the script with command “lscommand.sh”, made it executable by command “chmod +x lscommand.sh”. Then, to run this batch inside Ubuntu container we pull Ubuntu image “*docker pull ubuntu*”, run the container “docker run -it ubuntu /bin/bash”, copy script into container “docker cp lscommand.sh aabdb36943ab:/root/lscommand.sh”, and finally execute script in the container “docker exec -it aabdb36943ab/bin/bash -c "/root/lscommand.sh"”.

To see the bin folder, we use command first we navigate bin folder with command "cd /bin" and the list the contents of bin folder with command "ls".



```
root@aabdb36943ab:/bin# ls
'['                       hardlink          runcon
 addpart                  head              savelog
 apt                      hostid            script
 apt-cache                hostname          scriptlive
 apt-cdrom                i386              scriptreplay
 apt-config               iconv             sdiff
 apt-get                  id                sed
 apt-key                  infocmp           select-editor
 apt-mark                 infotocap         sensible-browser
 arch                     install           sensible-editor
 awk                      ionice            sensible-pager
 b2sum                    ipcmk             seq
 base32                   ipcrm             setarch
 base64                   ipcs              setpriv
 basename                 ischroot          setsid
 basenc                   join              setterm
 bash                     kill              sg
 bashbug                  last              sh
 captoinfo                lastb             sha1sum
 cat                      lastlog           sha224sum
 chage                    ldd               sha256sum
```

To create new ls command, we navigate bin folder "cd /bin" and execute command "chmod +x /bin/lscommand".

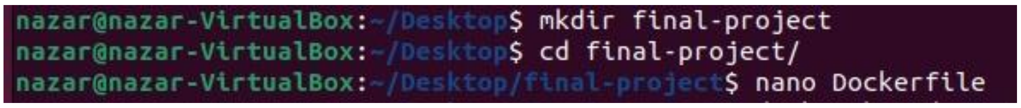


```
root@aabdb36943ab:/bin# chmod +x /bin/lscommand
root@aabdb36943ab:/bin# lscommand
'['                       head              savelog
 addpart                  hostid            script
 apt                      hostname          scriptlive
 apt-cache                i386              scriptreplay
 apt-cdrom                iconv             sdiff
 apt-config               id                sed
 apt-get                  infocmp           select-editor
 apt-key                  infotocap         sensible-browser
 apt-mark                 install           sensible-editor
 arch                     ionice            sensible-pager
 awk                      ipcmk             seq
 b2sum                    ipcrm             setarch
 base32                   ipcs              setpriv
 base64                   ischroot          setsid
 basename                 join              setterm
 basenc                   kill              sg
 bash                     last              sh
 bashbug                  lastb             sha1sum
 captoinfo                lastlog           sha224sum
 cat                      ldd               sha256sum
 chage                    link              sha384sum
 chattr                   linux32           sha512sum
```

Task 3: Create a Java Docker or Python container.

a. In Ubuntu virtual OS we created new directory 'final-project' using the command *"mkdir final-project"*.

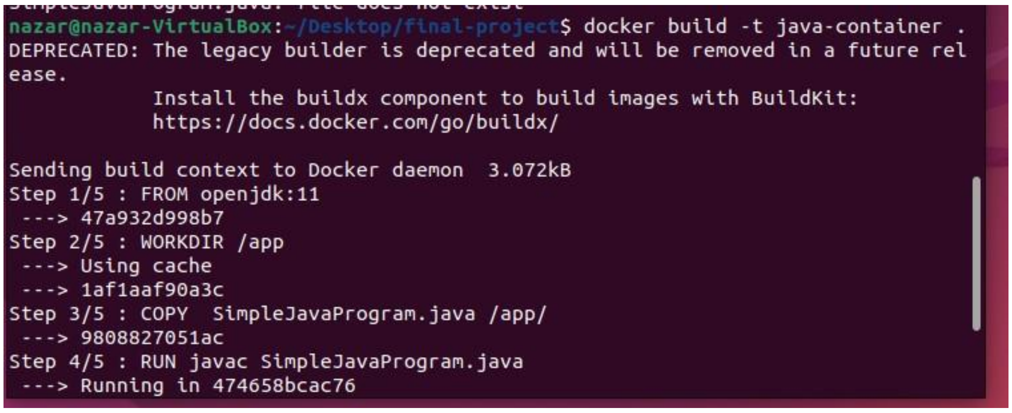
b. To create new a Docker file, firstly we selected our new directory with command *"cd final-project/"*, and then created a Docker file with command *"nano Dockerfile"*.



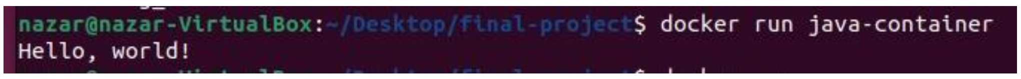
```
nazar@nazar-VirtualBox:~/Desktop$ mkdir final-project
nazar@nazar-VirtualBox:~/Desktop$ cd final-project/
nazar@nazar-VirtualBox:~/Desktop/final-project$ nano Dockerfile
```

c. We ran Docker file to create Java Container with help of *"docker build -t java-container"*.

d. We copied our 'SimpleJavaProgram' into this Java container. Then with command *"docker run java-container"* we launched our 'SimpleJavaProgram' in this container. As an output, we see program printed "Hello, World!".



```
nazar@nazar-VirtualBox:~/Desktop/final-project$ docker build -t java-container .
DEPRECATED: The legacy builder is deprecated and will be removed in a future rel
ease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM openjdk:11
 ---> 47a932d998b7
Step 2/5 : WORKDIR /app
 ---> Using cache
 ---> 1af1aaf90a3c
Step 3/5 : COPY  SimpleJavaProgram.java /app/
 ---> 9808827051ac
Step 4/5 : RUN javac SimpleJavaProgram.java
 ---> Running in 474658bcac76
```



```
nazar@nazar-VirtualBox:~/Desktop/final-project$ docker run java-container
Hello, world!
```

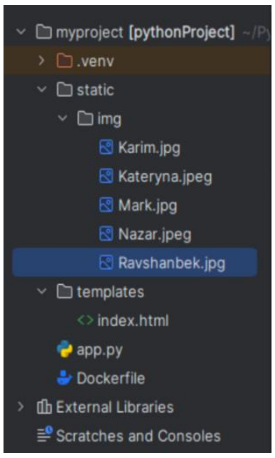
Task 4: Change the system configuration using Python Docker container.

a. We created a new directory 'myproject' with command *"mkdir myproject"*.
As we used Java for previous tasks, to be able to do this task, we firstly must install Python to our Ubuntu virtual OS.

To install Python we used commands:
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppt:deadsnakes/ppa
sudo apt update
sudo apt install python3.11

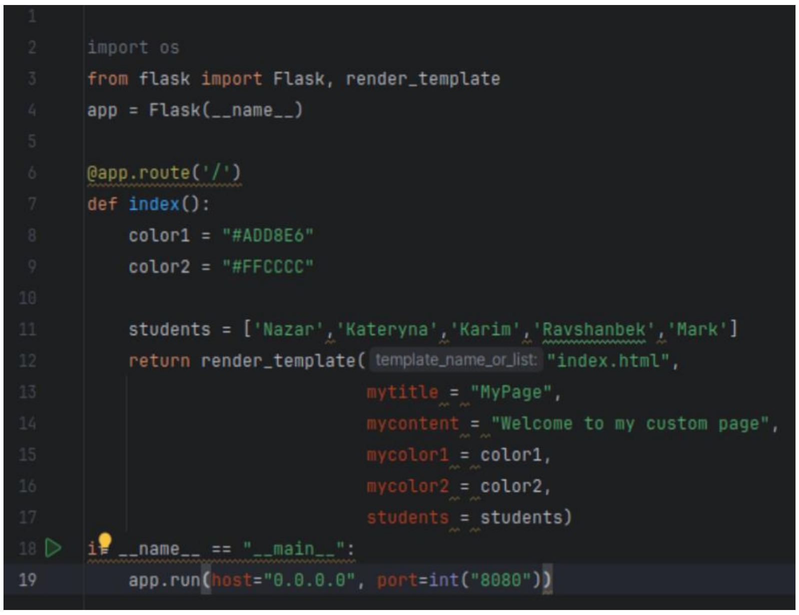
Then we needed to created Python Directories with commands:
cd PycharmProjects/
mkdir myproject
cd myproject
mkdir static
cd static
mkdir img
cd ..
mkdir templates
mkdir venv
python3 -m venv venv
source venv/bin/activate

b. Myproject folder must have the python code, templates directory with html index.



```
myproject [pythonProject] ~/Py
  .venv
  static
    img
      Karim.jpg
      Kateryna.jpeg
      Mark.jpg
      Nazar.jpeg
      Ravshanbek.jpg
  templates
    index.html
  app.py
  Dockerfile
External Libraries
Scratches and Consoles
```

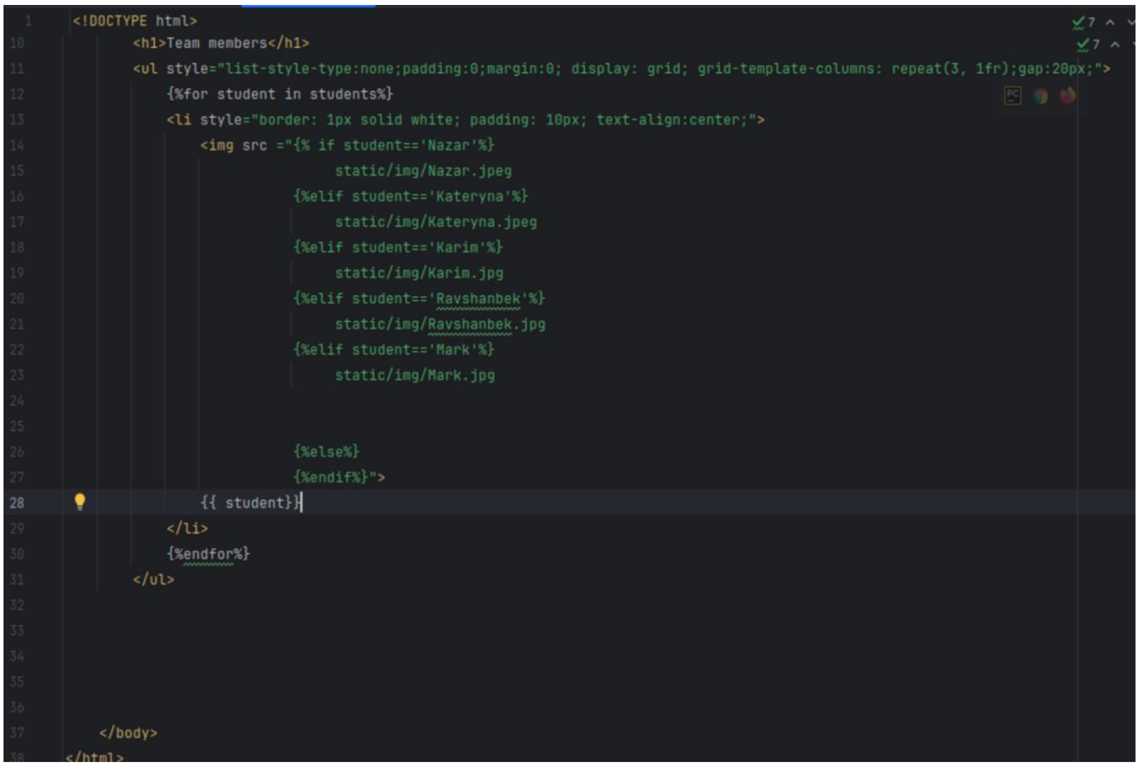
This is the root directory tree, representing the structure. Static dictionary contains dictionary img with images for display. Myproject directory has a directory templates for index.html file.



```python

import os
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def index():
    color1 = "#ADD8E6"
    color2 = "#FFCCCC"

    students = ['Nazar','Kateryna','Karim','Ravshanbek','Mark']
    return render_template( template_name_or_list: "index.html",
                           mytitle = "MyPage",
                           mycontent = "Welcome to my custom page",
                           mycolor1 = color1,
                           mycolor2 = color2,
                           students = students)
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=int("8080"))
```

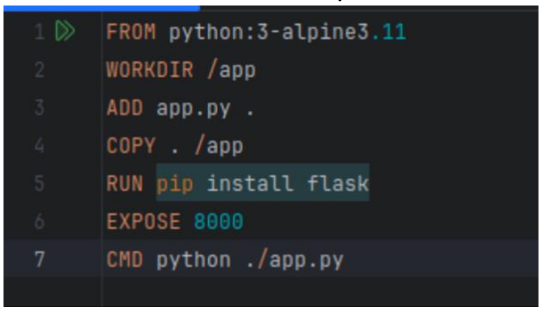
Python code with the main backend functionality for this project.

Html code for project (we changed the Html code and made it custom):



```html
<!DOCTYPE html>
        <h1>Team members</h1>
        <ul style="list-style-type:none;padding:0;margin:0; display: grid; grid-template-columns: repeat(3, 1fr);gap:20px;">
            {%for student in students%}
            <li style="border: 1px solid white; padding: 10px; text-align:center;">
                <img src ="{% if student=='Nazar'%}
                                static/img/Nazar.jpeg
                           {%elif student=='Kateryna'%}
                                static/img/Kateryna.jpeg
                           {%elif student=='Karim'%}
                                static/img/Karim.jpg
                           {%elif student=='Ravshanbek'%}
                                static/img/Ravshanbek.jpg
                           {%elif student=='Mark'%}
                                static/img/Mark.jpg


                           {%else%}
                           {%endif%}">
                {{ student}}
            </li>
            {%endfor%}
        </ul>





    </body>
</html>
```

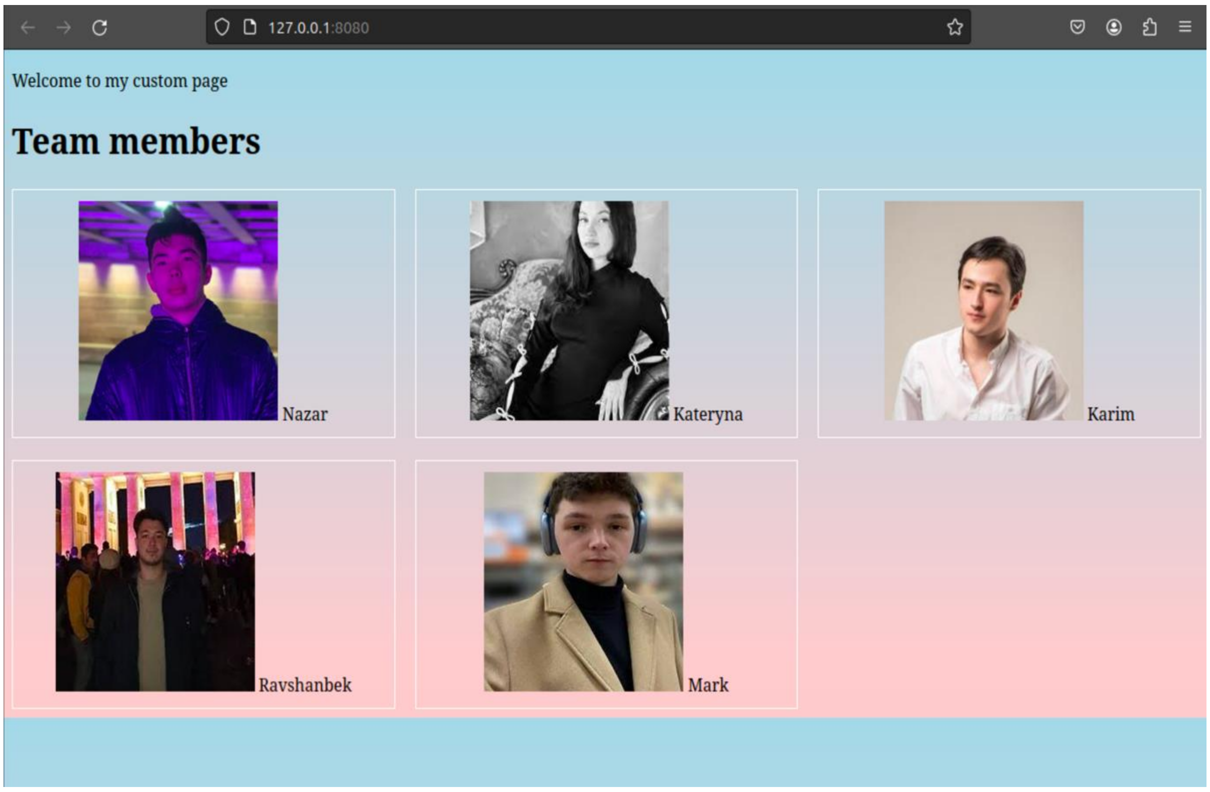
c. **We created a Docker file**. Python was installed earlier, and Flask we installed using command *"pip install flask"*.



```
FROM python:3-alpine3.11
WORKDIR /app
ADD app.py .
COPY . /app
RUN pip install flask
EXPOSE 8000
CMD python ./app.py
```

d. We ran the docker file to create Python Container:
Docker login to "jolygolden" account.
docker build -t jolygolden/finalproject:0.0.1.RELEASE .
docker push jolygolden /finalproject:0.0.1.RELEASE

e. We used Following commands to run the container:
docker pull jolygolden /finalproject:0.0.1.RELEASE
docker run -it --rm -p 8080:8080 jolygolden /finalproject:0.0.1.RELEASE



Welcome to my custom page

Team members

The image was successfully pulled and launched on the 8080 port.