Before We Get Started

The goal of this runbook is to safeguard the user from a deep hatred of technology and early hair loss. We will attempt to set up and configure our system to run the project before we encounter issues. Hopefully, this preemptive approach will provide the necessary guidance to simplify the process.

Introduction

This project is built solely for use in a Linux environment, and will only be explained in that capacity. It is not impossible to implement something similar in a different operating system, but we will not detail steps to do such. As a Linux based project, there are certain libraries and dependencies that must be installed for the program to work. We will cover all those details (and commands) very soon.

This project was developed with python in the PyCharm IDE using Flask and SQLAlchemy. It implements posting and user account microservices, such as creating a post or user and deleting a post or user. Additional functionalities of our implemented microservices will be discussed later.

Early development testing was done using Postman, but we have since moved to automated testing through curl commands in bash scripts. We are using Foreman to start all the processes associated with our project to better accommodate testing within our WSGI container, Gunicorn.

Getting Started

The following section outlines very necessary steps to ensure that our project will function properly. From our desktop, we should be able to press ALT+F2 to open a command line. Type gnome-terminal and press enter (Figure 1). This will open the terminal. We may now begin.

Step 1

First and foremost, we should be concerned with the system's default Python version. The command python --version will return the current version of python the system is using. If the current version is 3.x.x then we can move to *Step 2*. Otherwise, we must update to a stable version of Python 3 or tell the system to point to the correct path.

It is possible that we have multiple versions of python installed. As a test, we can run the command python3 --version. If the terminal echoes "Python 3.x.x", we have python3 installed, but the system's default path is pointing somewhere else. If the terminal does not echo, we must install Python 3 and can do so with the command sudo apt-get install python3.

Next, we must tell our system to point to the correct path. Using the command sudo nano ~/.bash_aliases we open a text file where we assign an alias to tell the system where we want the default python version to be. Type alias python=/usr/bin/python3, then press CTRL+S and CTRL+X. We have just saved an alias for python and may continue to *Step 2*.

**Note: You may want to revert that change at some point which will require you to reopen the .bash_aliases file and delete the alias.*

Step 2

The second thing we should make sure our system has is pip. If you installed the latest Python 3.x.x in *Step 1*, pip should have been included. Run the command pip3 --version to check if it is installed. If the terminal says that pip3 is not installed, run the command sudo apt-get install pip3. We may refer to Figure 2 to see example outputs. We are ready to install all the libraries and dependencies to run the project.

Libraries and Dependencies

This project uses a few different libraries to run. For simplicity, they've been alphabetically arranged in a table. Ensure that the following libraries and their dependencies are installed on your Linux system:

Library/Dependency	Command
Flask	sudo pip3 install flask
Flask-SQLAlchemy	sudo pip3 install flask_sqlalchemy
Flask-Marshmallow	sudo pip3 install flask_marshmallow
Foreman	gem install foreman
Gunicorn3	sudo apt install --yes gunicorn3
Marshmallow-SQLAlchemy	sudo pip3 install marshmallow-sqlalchemy
Pytz	sudo pip3 install pytz

Setup and Run

To keep things clean and less confusing, it would be wise to create a new directory for our project. Using the command mkdir CPSC449_DDM_1, we can create a directory named

CPSC449_DDM_1. After we've created our directory, we will need to navigate to that directory. Use the command cd CPSC449_DDM_1 to enter our newly created directory (Figure 3).

We are ready to grab the project and can do so by running the command (Figure 4):

git clone https://github.com/daytonschuh/CPSC449Project1

This will create another directory with all the files from the project. Again, we will need to navigate into that directory by using the command cd CPSC449Project1 (Figure 5). We are now in our working directory where all the magic will happen.

**Note: You may name the directory whatever you wish by changing CPSC449_DDM_1 to something different. However, this will affect all following commands that use the directory name.*

Files

If we run the command ls, we will see all the files that are included in our project (Figure 6). The files that we should see in our project folder are:

- app.py
- Caddyfile
- posts.db
- Procfile
- README.md
- Runbook.pdf
- testpost.json
- testpost.sh
- testuser.json
- testuser.sh
- wsgi.py

If there are any missing or additional files before we proceed, there may have been edits to the github and this runbook is outdated. We will continue to our microservices and testing section to see if things are still working.

Microservices and Testing

Now we are ready to run a test. Press CTRL+SHIFT+N to open a second terminal in the same directory. Make sure that the directories are the same. If this does not work, we will have to open another terminal and navigate to the working directory.

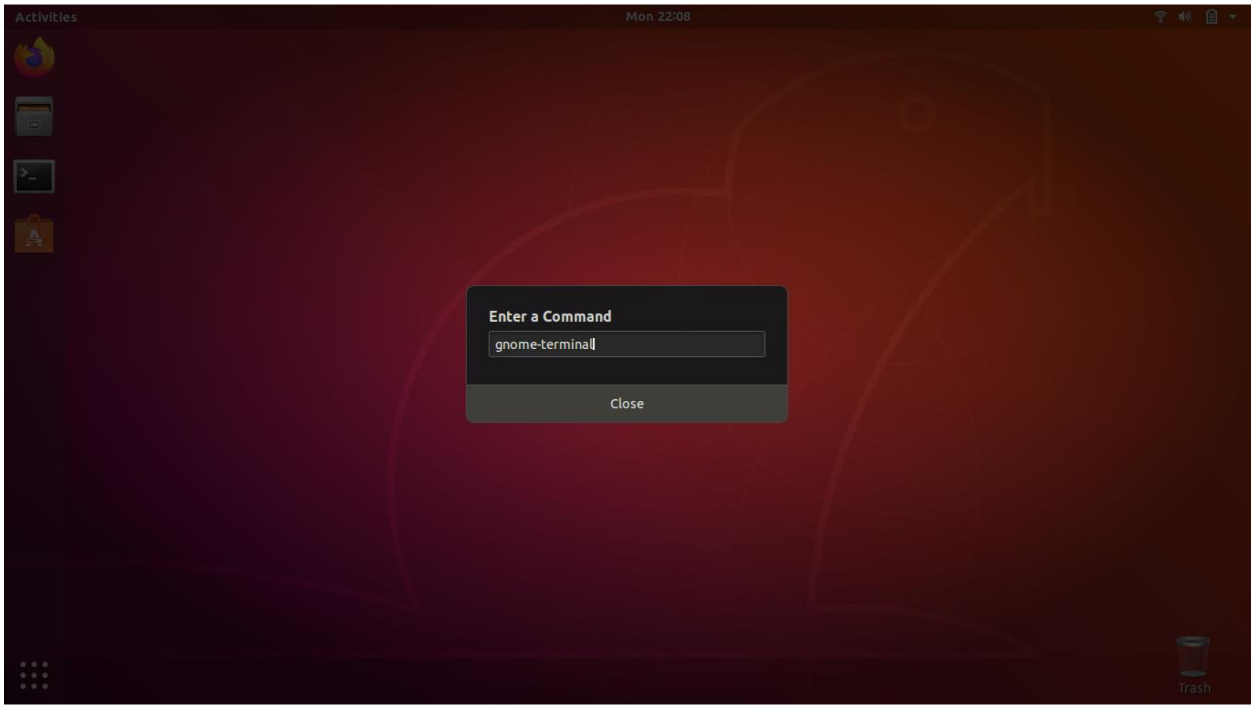
It may be easiest to resize the terminals and place them side by side to watch the testing process. In the first terminal, run the command foreman start -c. This reads all the commands from our Procfile and names those instances according to their associated labels.

In the second terminal, we can test our user management API with the command bash testuser.sh (Figure 7). It will attempt to create a user, update the user's email, increment the user's karma, decrement the user's karma and delete the user. It uses our testuser.json file to supply the appropriate data for the test. Since testuser.sh is a bunch of curl commands, we will have to carefully read over the output. We should be able to find a success message for each test.

After confirming that each test for testuser.sh has run properly, we can use the command bash testpost.sh to test our post management API (Figure 8). It will attempt to create a post, retrieve a post, delete a post, get a specified number of most recent posts from a specified community and get a specified number of most recent posts from any community. It uses our testpost.json file to supply the appropriate data for the test. Again, we will have to carefully review the output for each success message.

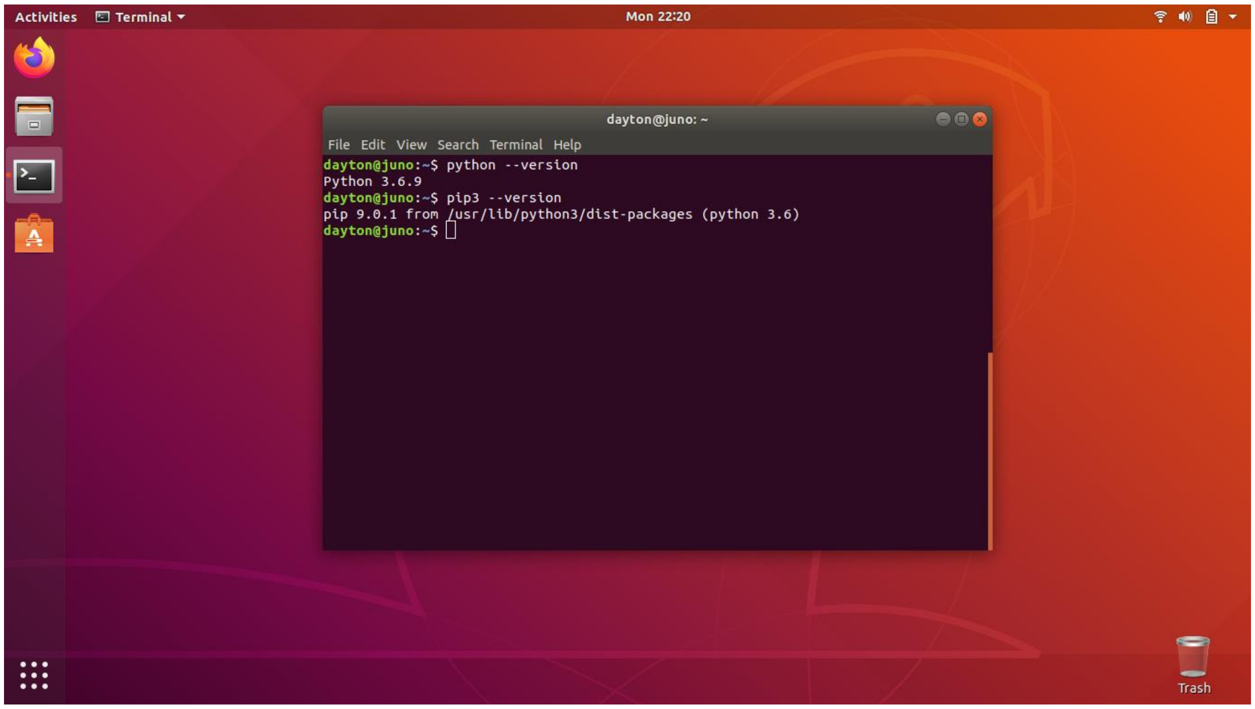
Figures

Figure 1



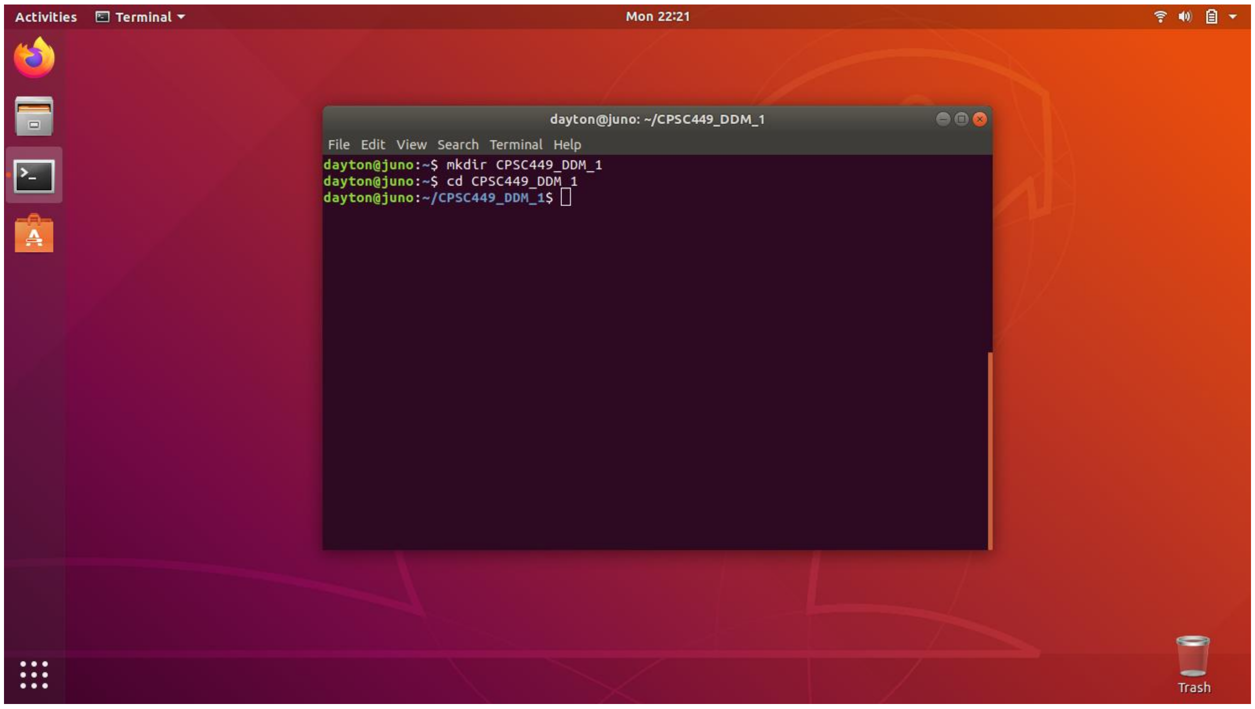
```
gnome-terminal
```

Figure 2



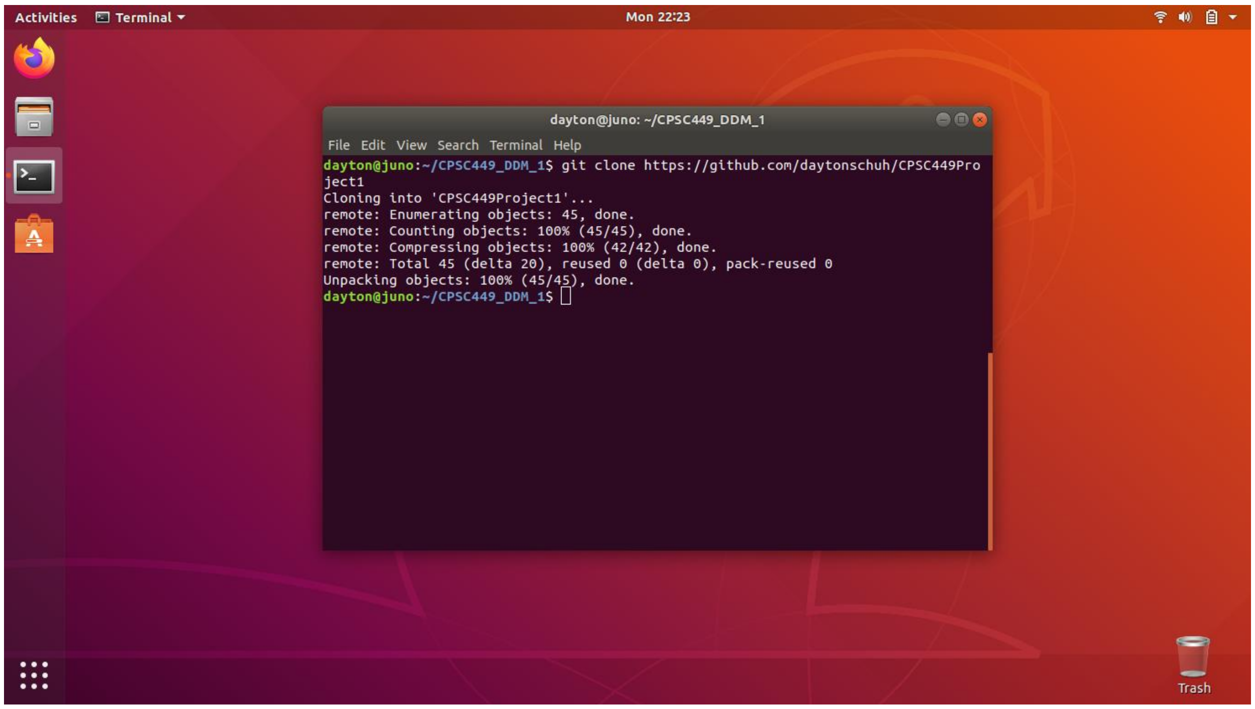
```
dayton@juno:~$ python --version
Python 3.6.9
dayton@juno:~$ pip3 --version
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)
dayton@juno:~$
```

Figure 3


Activities
Terminal
Mon 22:21
dayton@juno: ~/CPSC449_DDM_1
File Edit View Search Terminal Help
Trash

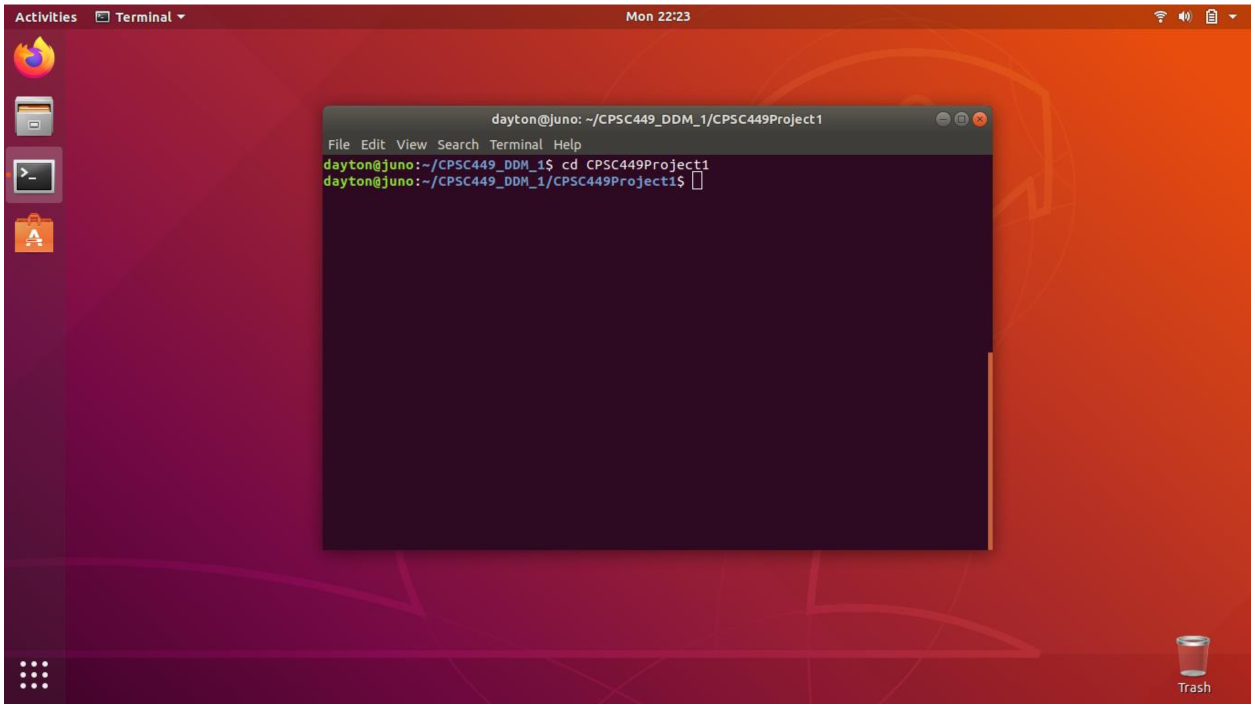
```
dayton@juno:~$ mkdir CPSC449_DDM_1
dayton@juno:~$ cd CPSC449_DDM_1
dayton@juno:~/CPSC449_DDM_1$
```

Figure 4


Activities
Terminal
Mon 22:23
dayton@juno: ~/CPSC449_DDM_1
File Edit View Search Terminal Help
Trash

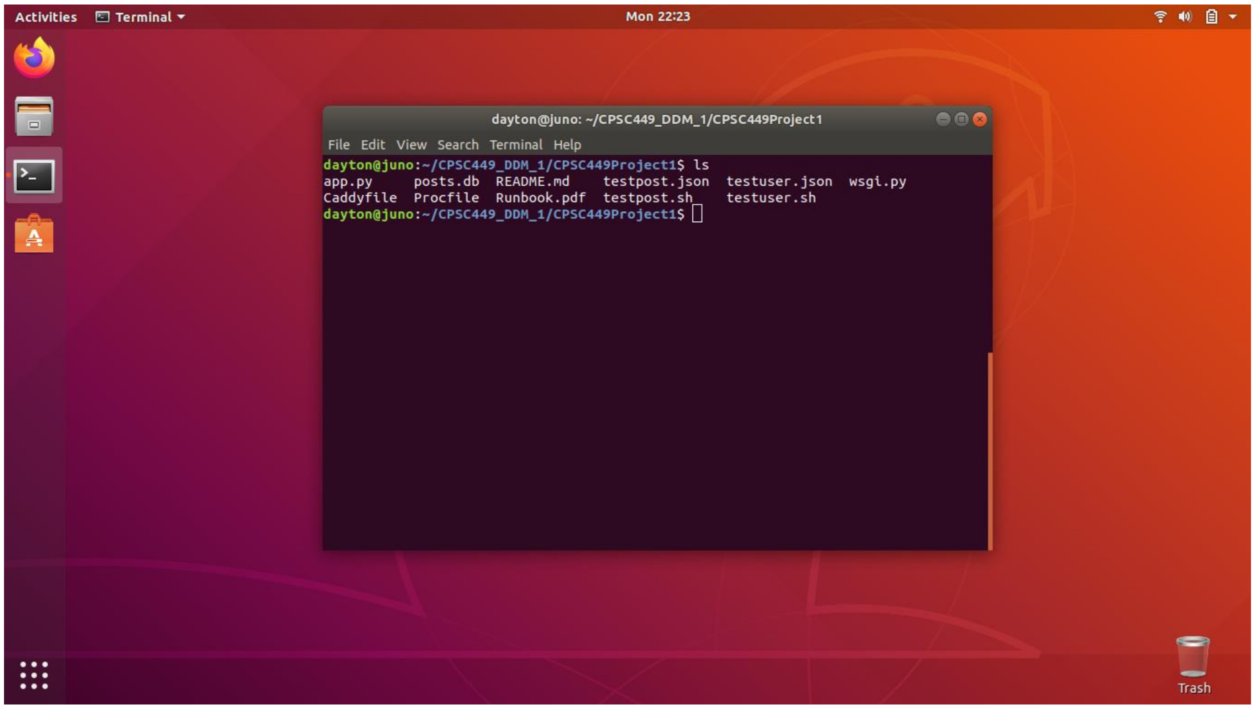
```
dayton@juno:~/CPSC449_DDM_1$ git clone https://github.com/daytonschuh/CPSC449Pro
ject1
Cloning into 'CPSC449Project1'...
remote: Enumerating objects: 45, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 45 (delta 20), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (45/45), done.
dayton@juno:~/CPSC449_DDM_1$
```

Figure 5


Activities
Terminal
Mon 22:23
dayton@juno: ~/CPSC449_DDM_1/CPSC449Project1
File Edit View Search Terminal Help
Trash

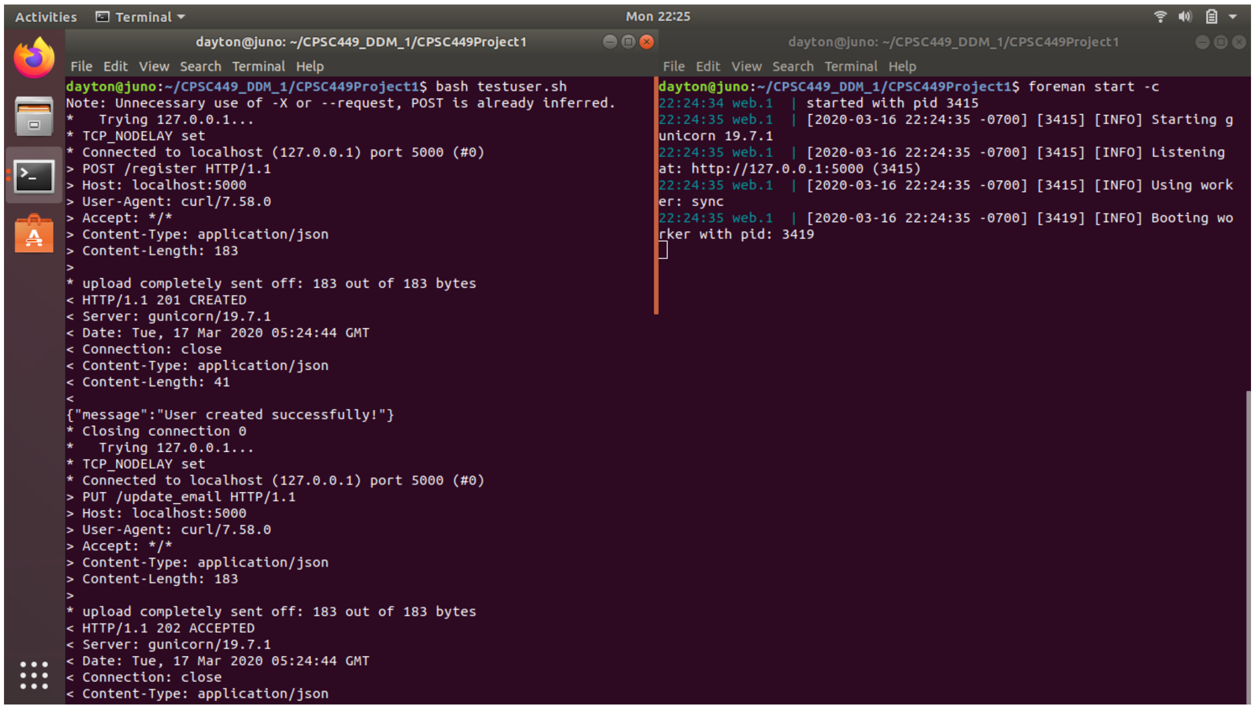
```
dayton@juno:~/CPSC449_DDM_1$ cd CPSC449Project1
dayton@juno:~/CPSC449_DDM_1/CPSC449Project1$
```

Figure 6


Activities
Terminal
Mon 22:23
dayton@juno: ~/CPSC449_DDM_1/CPSC449Project1
File Edit View Search Terminal Help
Trash

```
dayton@juno:~/CPSC449_DDM_1/CPSC449Project1$ ls
app.py     posts.db  README.md    testpost.json  testuser.json  wsgi.py
Caddyfile  Procfile  Runbook.pdf  testpost.sh    testuser.sh
dayton@juno:~/CPSC449_DDM_1/CPSC449Project1$
```

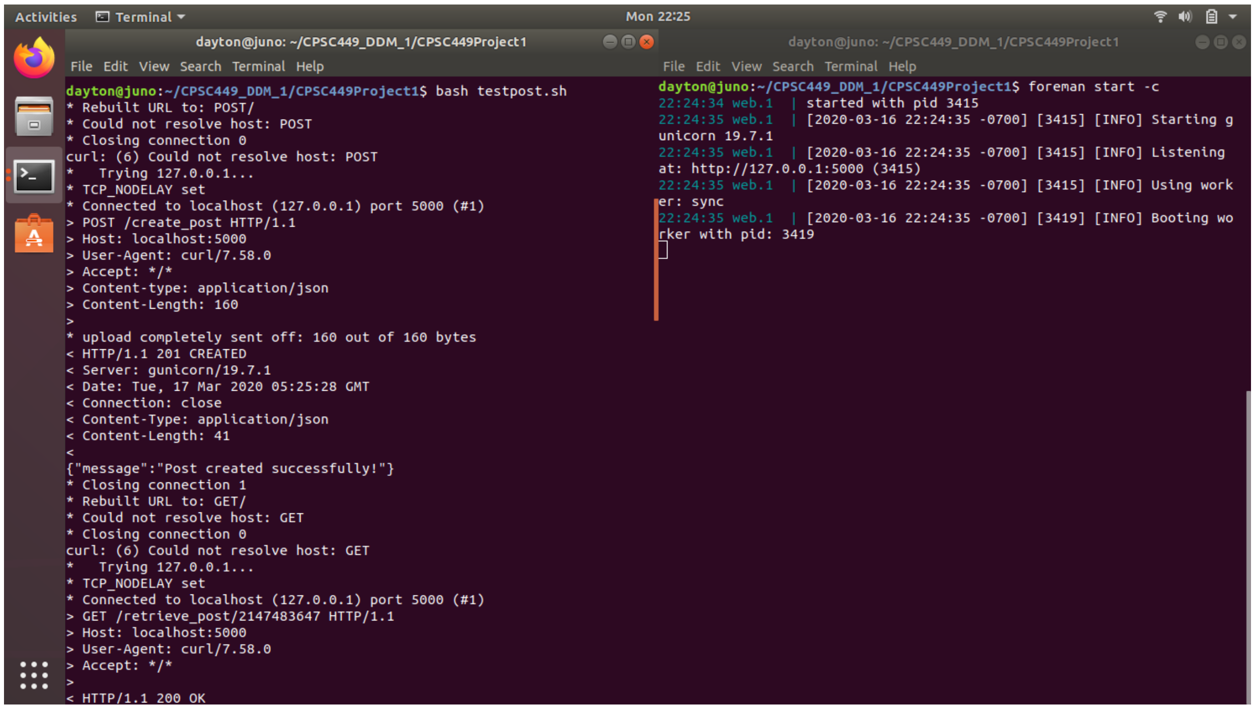
Figure 7


Activities
Terminal
Mon 22:25
dayton@juno: ~/CPSC449_DDM_1/CPSC449Project1
File Edit View Search Terminal Help
dayton@juno: ~/CPSC449_DDM_1/CPSC449Project1
File Edit View Search Terminal Help

```
dayton@juno:~/CPSC449_DDM_1/CPSC449Project1$ bash testuser.sh
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#0)
> POST /register HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 183
>
* upload completely sent off: 183 out of 183 bytes
< HTTP/1.1 201 CREATED
< Server: gunicorn/19.7.1
< Date: Tue, 17 Mar 2020 05:24:44 GMT
< Connection: close
< Content-Type: application/json
< Content-Length: 41
<
{"message":"User created successfully!"}
* Closing connection 0
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#0)
> PUT /update_email HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 183
>
* upload completely sent off: 183 out of 183 bytes
< HTTP/1.1 202 ACCEPTED
< Server: gunicorn/19.7.1
< Date: Tue, 17 Mar 2020 05:24:44 GMT
< Connection: close
< Content-Type: application/json
```

```
dayton@juno:~/CPSC449_DDM_1/CPSC449Project1$ foreman start -c
22:24:34 web.1  | started with pid 3415
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3415] [INFO] Starting g
unicorn 19.7.1
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3415] [INFO] Listening
at: http://127.0.0.1:5000 (3415)
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3415] [INFO] Using work
er: sync
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3419] [INFO] Booting wo
rker with pid: 3419
```

Figure 8



Activities
Terminal
Mon 22:25
dayton@juno: ~/CPSC449_DDM_1/CPSC449Project1
File Edit View Search Terminal Help
dayton@juno: ~/CPSC449_DDM_1/CPSC449Project1
File Edit View Search Terminal Help

```
dayton@juno:~/CPSC449_DDM_1/CPSC449Project1$ bash testpost.sh
* Rebuilt URL to: POST/
* Could not resolve host: POST
* Closing connection 0
curl: (6) Could not resolve host: POST
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#1)
> POST /create_post HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-type: application/json
> Content-Length: 160
>
* upload completely sent off: 160 out of 160 bytes
< HTTP/1.1 201 CREATED
< Server: gunicorn/19.7.1
< Date: Tue, 17 Mar 2020 05:25:28 GMT
< Connection: close
< Content-Type: application/json
< Content-Length: 41
<
{"message":"Post created successfully!"}
* Closing connection 1
* Rebuilt URL to: GET/
* Could not resolve host: GET
* Closing connection 0
curl: (6) Could not resolve host: GET
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#1)
> GET /retrieve_post/2147483647 HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
```

```
dayton@juno:~/CPSC449_DDM_1/CPSC449Project1$ foreman start -c
22:24:34 web.1  | started with pid 3415
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3415] [INFO] Starting g
unicorn 19.7.1
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3415] [INFO] Listening
at: http://127.0.0.1:5000 (3415)
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3415] [INFO] Using work
er: sync
22:24:35 web.1  | [2020-03-16 22:24:35 -0700] [3419] [INFO] Booting wo
rker with pid: 3419
```