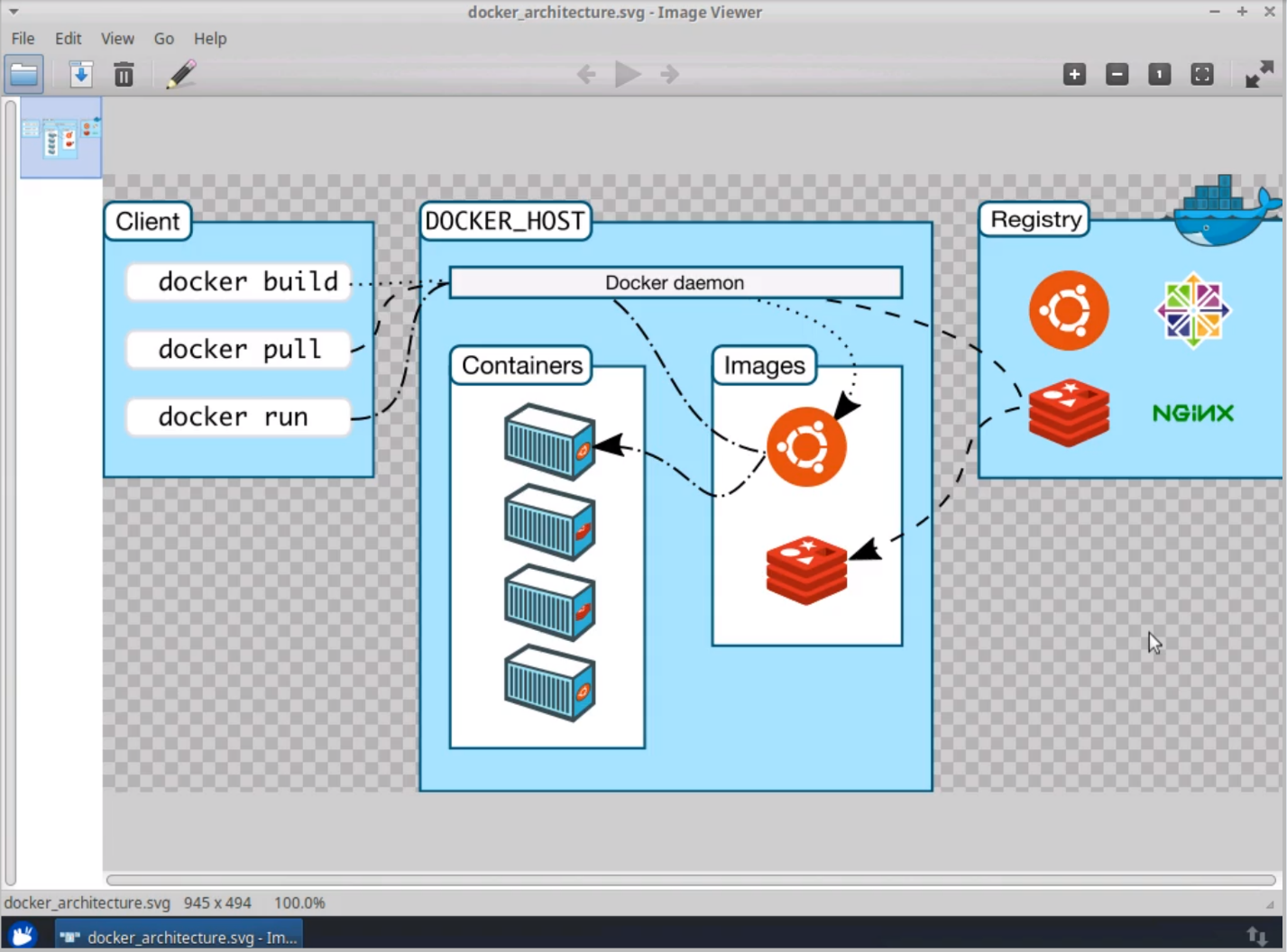
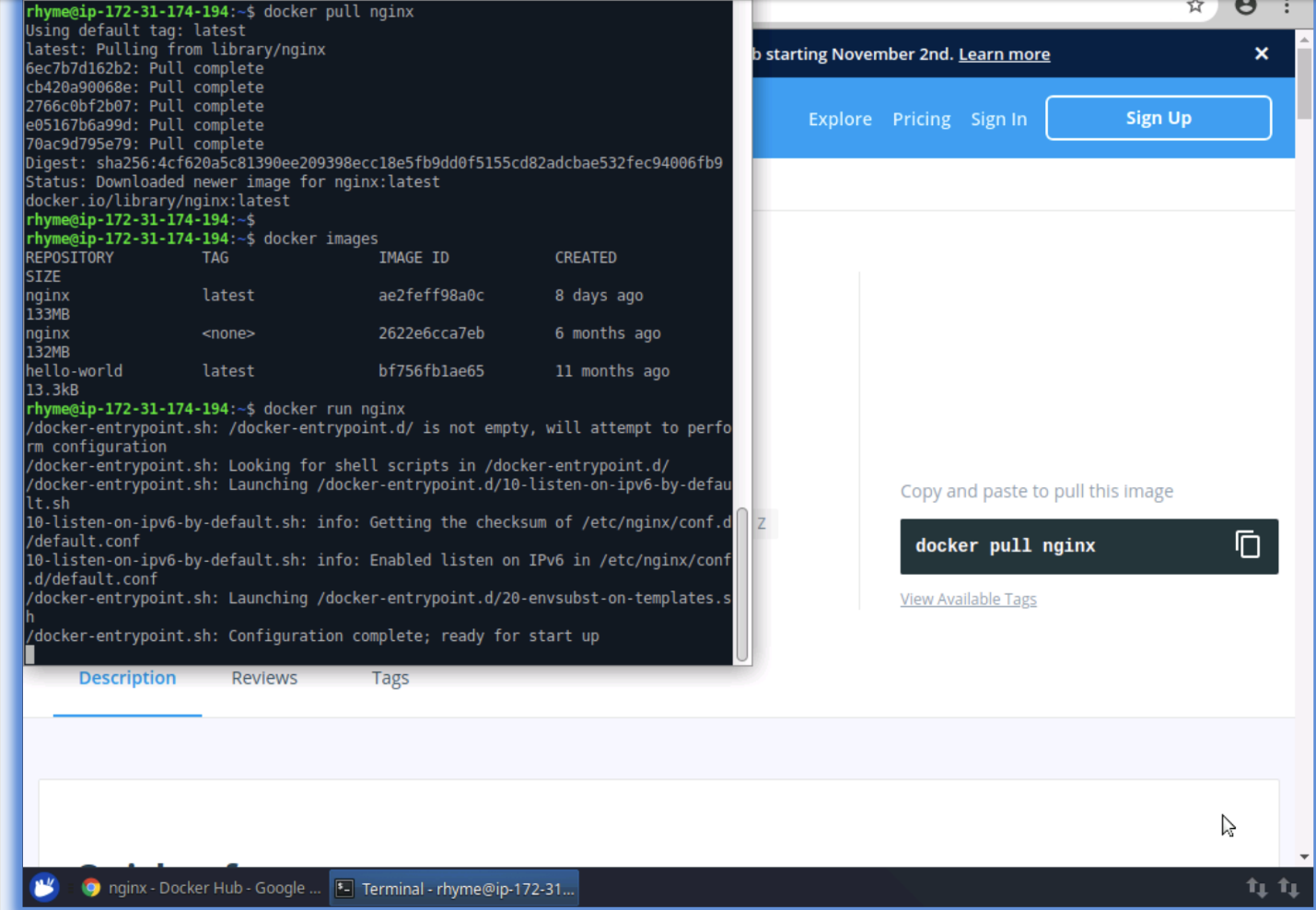
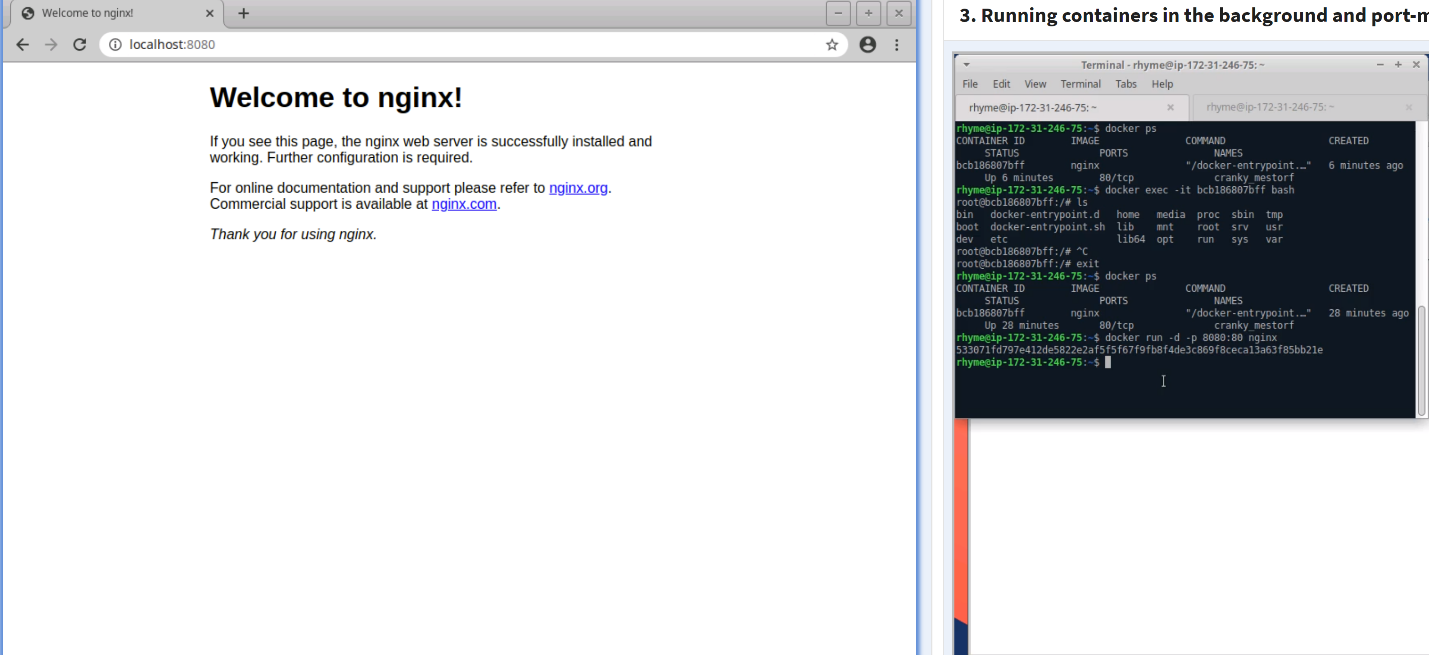
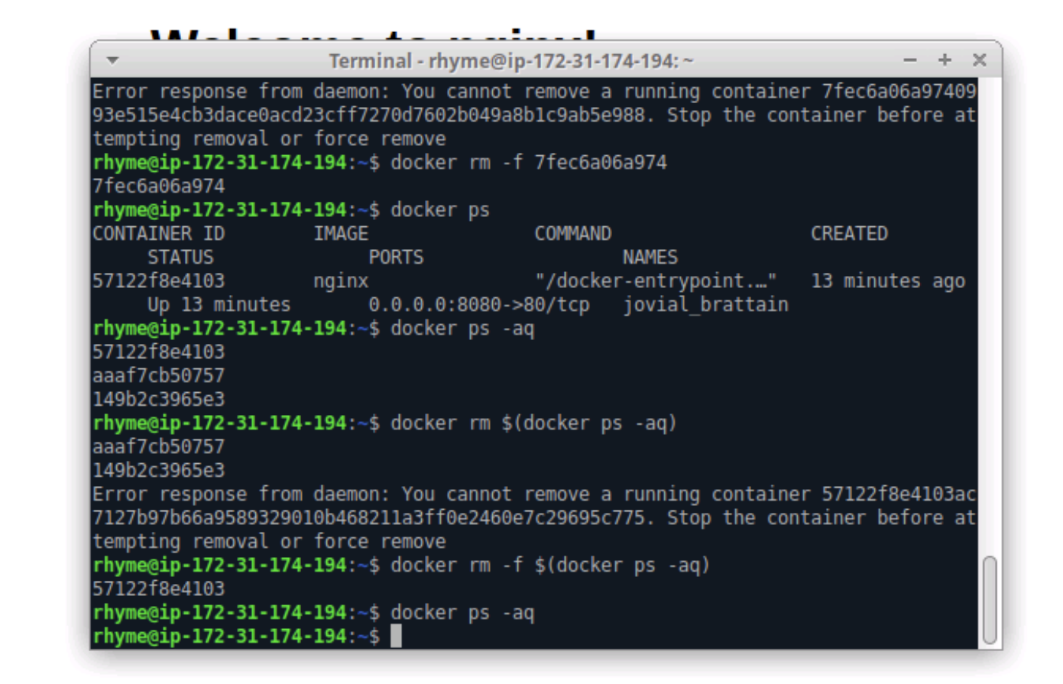
Build Your Own Portfolio Site in Docker

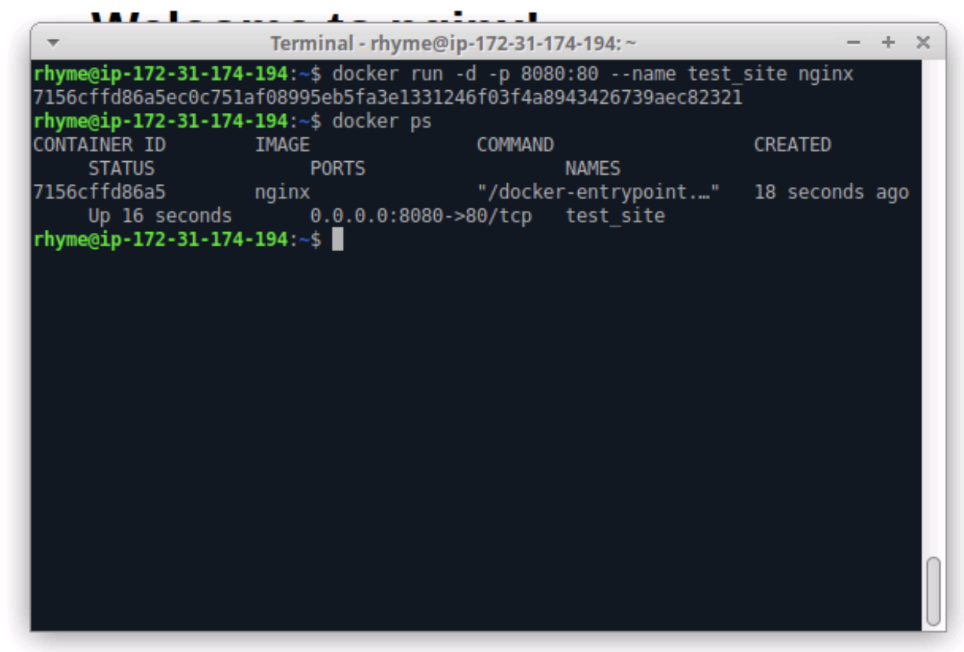
# Architecture of Docker



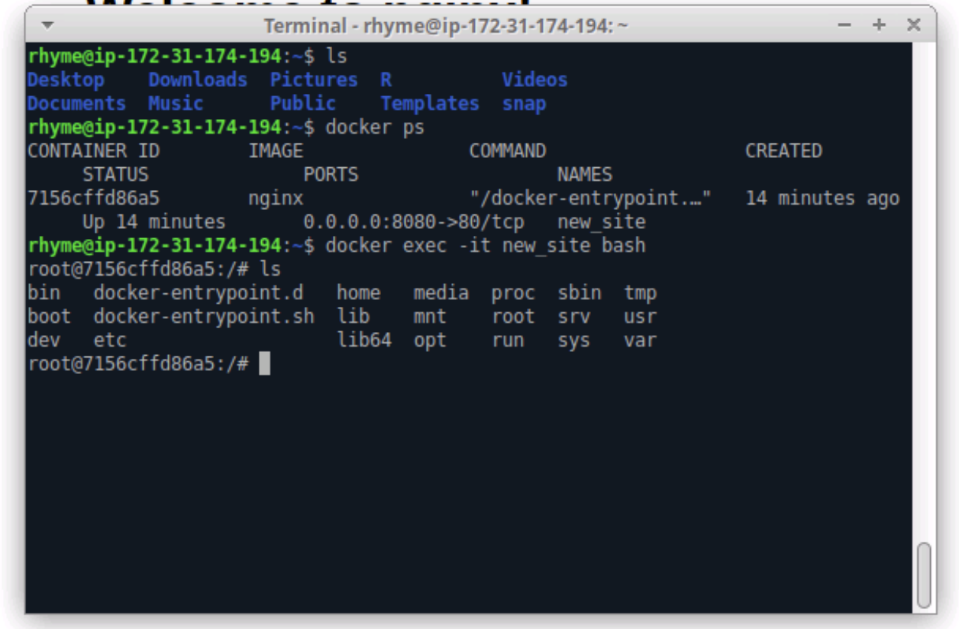
* Docker Engine is what lives on the host machine and is made of three things:
  + Client – where we issue commands from (on the CLI)
  + Server (Docker Host)
  + Registry (Remote)
* What the commands mean:
  + docker pull to pull an image down to the local machine and then use it to create a container to go and build upon.
  + What we did:
    - We downloaded an image of nginx software on our system using the terminal.
      * We used command: docker pull nginx
      * To check installation, we simply wrote docker images
      * Then run: docker run nginx



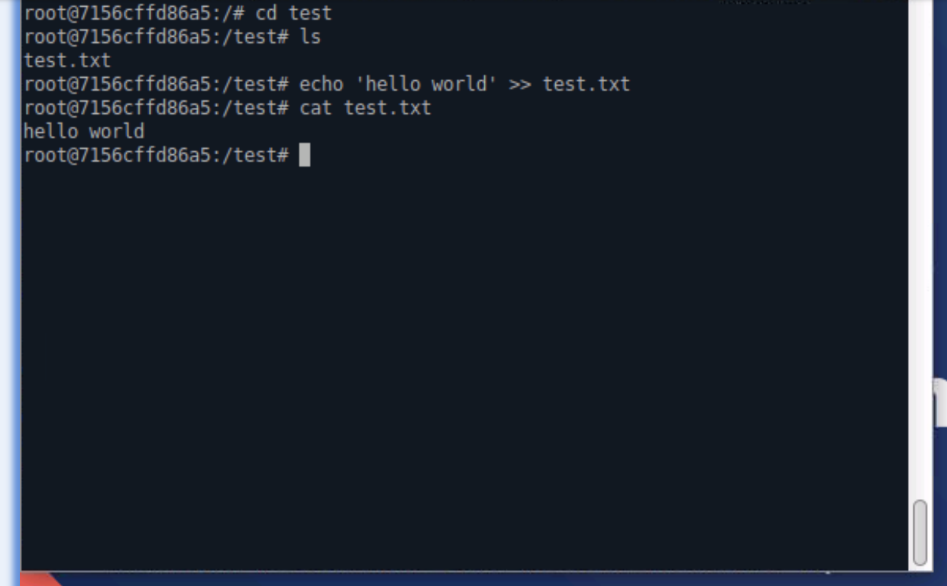
* To check if the nginx is running/started:
  + - Open a new terminal and type: docker ps (ps means process)
* Chapter 2: Running Containers in the background and port mapping
* How to enter a docker container and then work in it:
  + - Type: docker ps and press enter
    - From the details displayed, copy the container ID
    - Now type: docker exec -it 7fec6a06a974 bash (the value after it is the container id)
    - Now you have successfully entered the container
* You can exit the container by pressing ctrl + d
* Now, for port mapping part,
  + - Again, run command docker ps
    - Check your docker port (displayed as 80/tcp) where 80 is the port
    - Now run: docker run -d -p 8080:80 nginx
    - Now go to localhost:8080 from your browser and check if the mapping is done
    - A successful mapping looks like this:
    - Chapter 3: How to start/stop/delete containers and other important docker commands
      * To stop a docker container, you need the container’s ID which again you can get by running: docker ps
      * Copy the ID
      * Run: docker stop \*container ID\*
      * Similarly, to restart it, run: docket start \*container ID\*
      * To have a glimpse of all the processes we started and exited, run:
        + docker ps -a
      * To delete/remove a running container, simply run:
        + docker rm -f \*container ID\*
        + -f stands for force as we’re force-removing the currently running container. If we don’t mention this, the command line gives us an error saying we can’t remove a running container
      * To get a list of existing containers, (running or stopped), simply run:
        + docker ps -aq
        + To delete all of these at once, run: docker rm -f $(docker ps -aq)
        + This is what you see
        + Running docker ps -aq again will then display nothing
* Now to create and rename a fresh docker container:
  + Run: docker run -d -p 8080:80 –name (read it as two hyphens before name) test\_site (i.e. the name you give it) nginx
  + See image below:



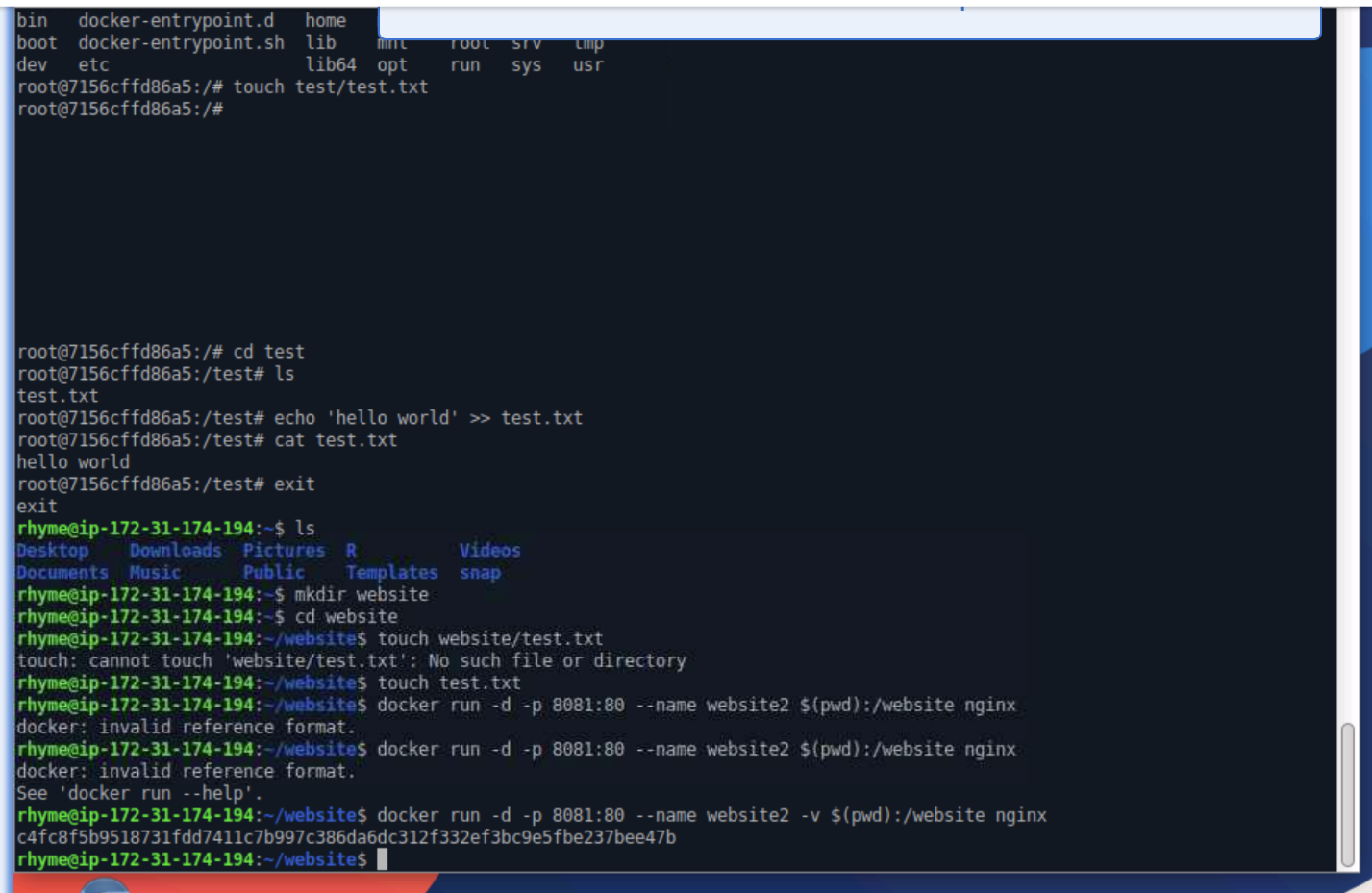
* To rename this container, run: docker rename test\_site new\_site
* Check the changes by running: docker ps
* Chapter 4: Link development files with some files inside the docker container using docker volumes
* A problem: when working on your local machine or some code editor in your regular local working environment, whatever changes you make will reflect on the website connected to your localhost port but won’t reflect in the file system of the docker container
* Meaning, the changes will only be limited to the local environment and won’t reflect in the docker environment.
* To get around this, we will have to create a whole new docker image and rebuild our image each time
* Now, start one of the existing docker containers by running: docker exec -it new\_site bash



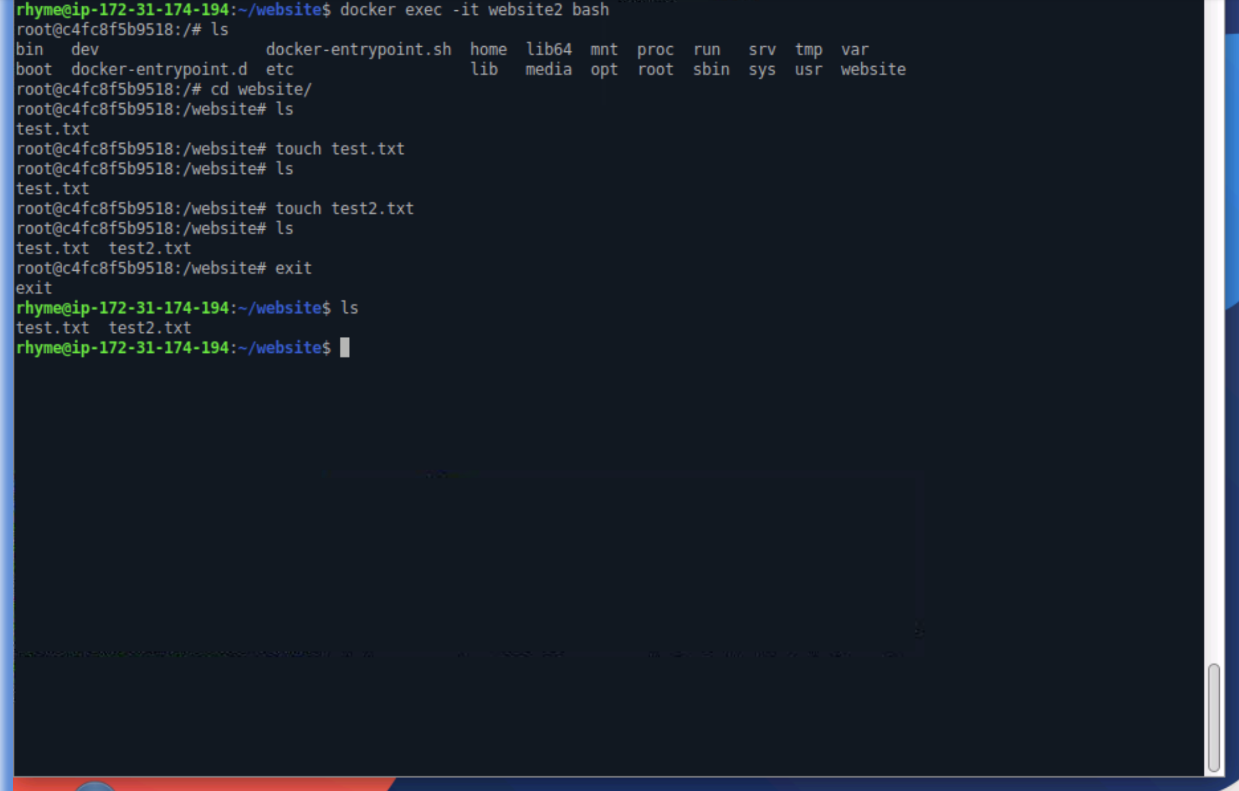
* + - * Now,
        + Make a new folder inside the running container: mkdir test
        + Inside test, create a file by running: touch test/test.txt
        + Now, the file has been created and added to the test folder
        + See change directory to the folder and make changes to the created file test.txt



* + - * Now,



* + - * Now, for linking the local and docker file systems,
        + cd into the website folder created in the docker file system
        + using touch test2.txt, create another file in the website folder
        + if you run ls, you will see both test.txt and test2.txt
        + run: exit
        + Now, again if you do run ls and see the two files listed, it means your local machine has them too, since we already run exit and came out of the docker container
        + See below:

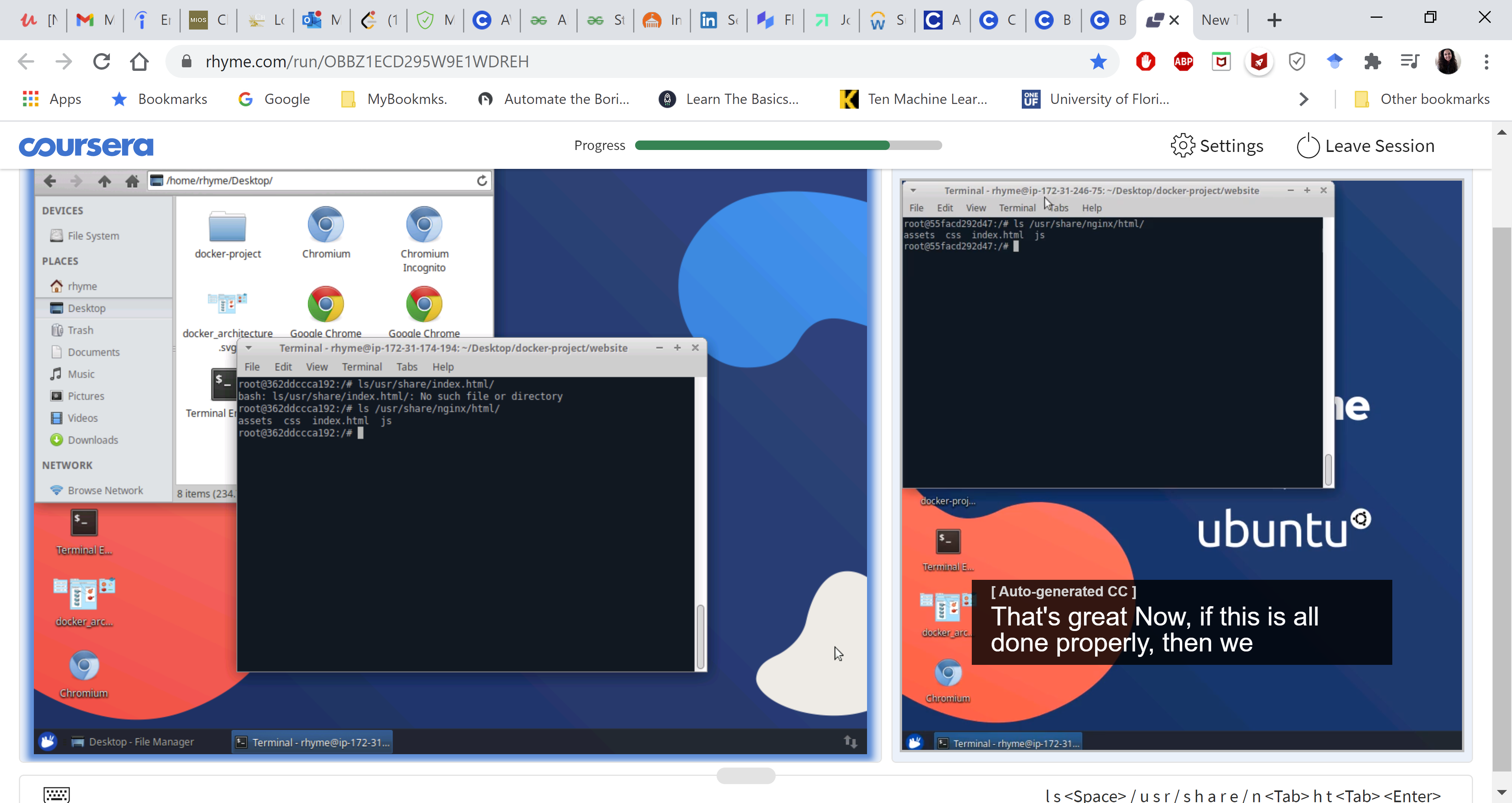


* + - * Chapter 5:



Now run: docker exec -it website2 bash to go inside the container

Now, got the folder to see if all website files are present:



Now, if you go to localhost:8080, it’ll open up to the created website/portfolio