**Assignment**

**NOTE: Do not forget to see manual page using "--help" option in command when searching for options/commands for a particular task.**

**1. Install Docker, either on your native OS or on a VM. Make sure it runs. type "docker -v" to check if it's installed.**

**try below commands for help**

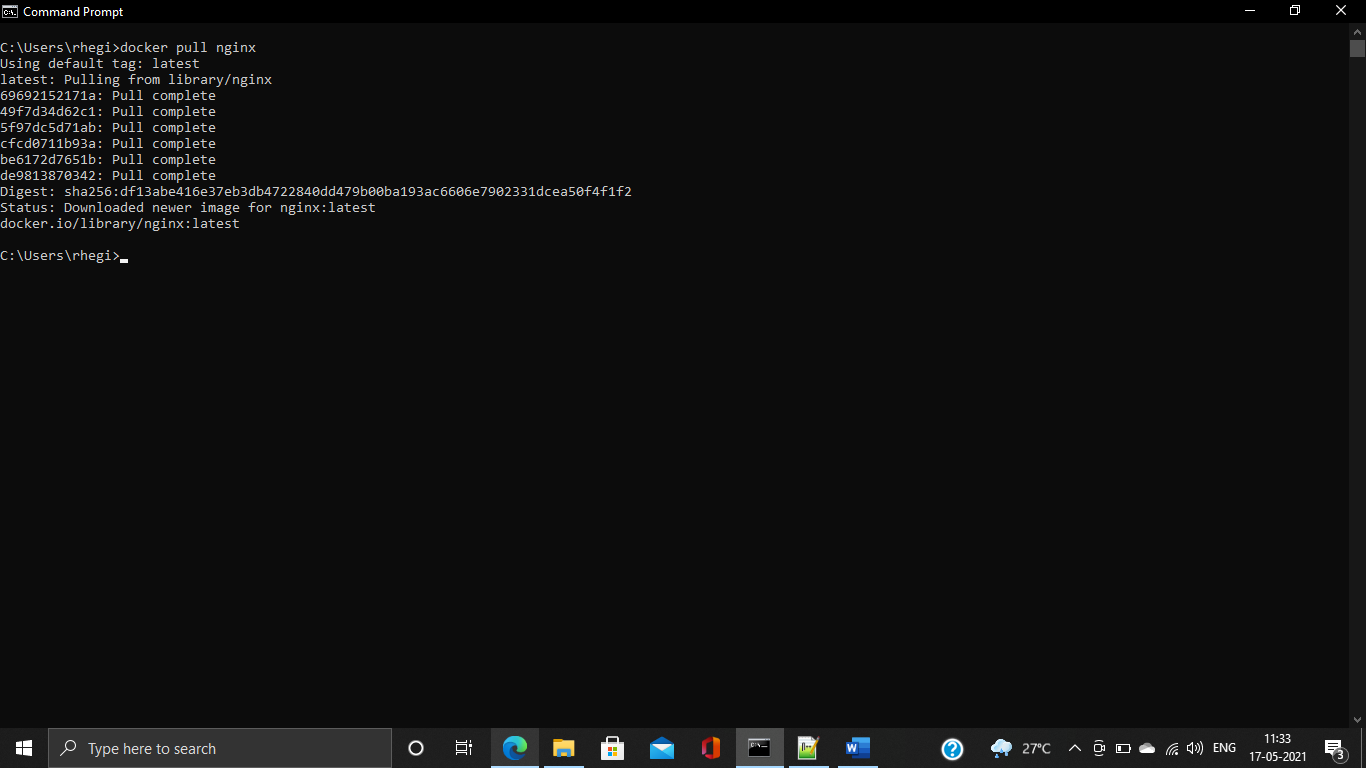
**docker --help ---> This command shows all available options and commands to work with images and containers**

**docker images --help ---> This command shows all the avaialble options and commands to work with docker images**

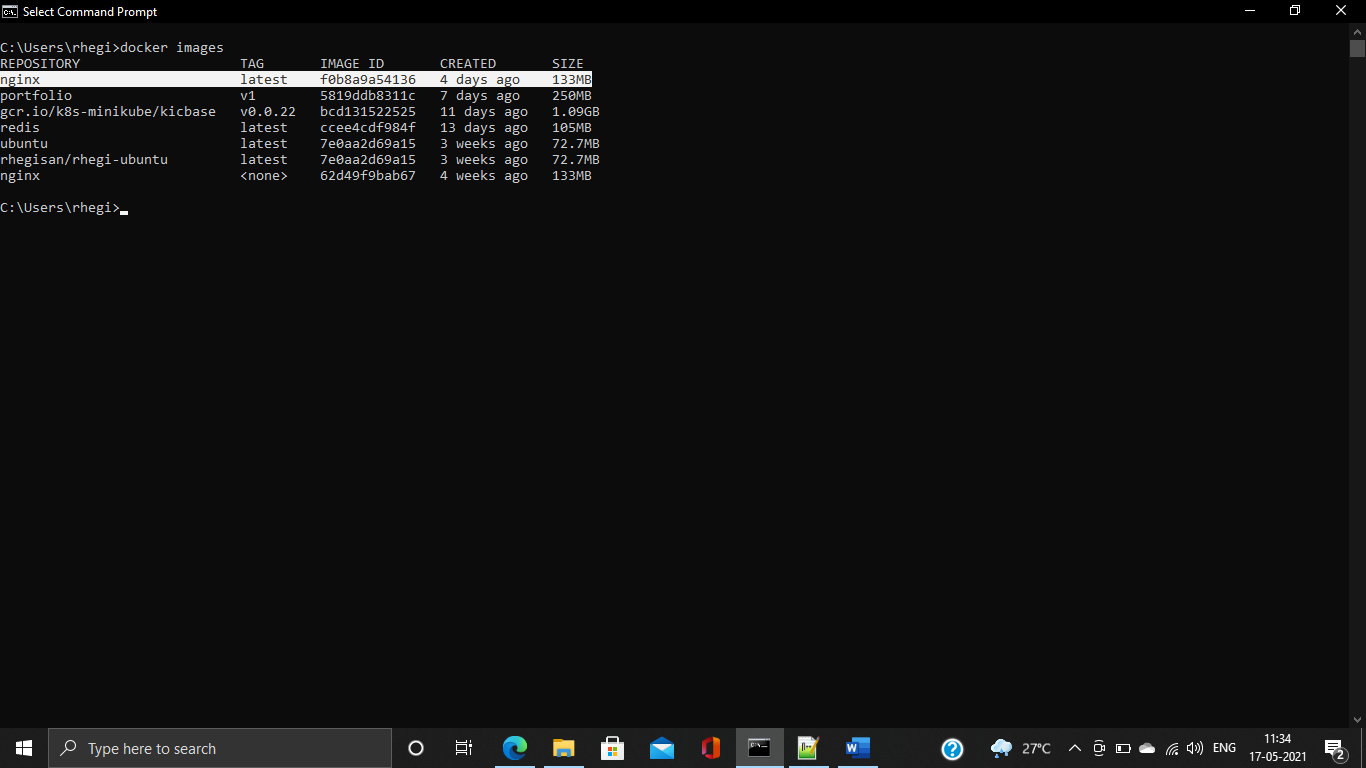
**docker ps --help ---> This command shows all the avaialble options and commands to work with docker containers**



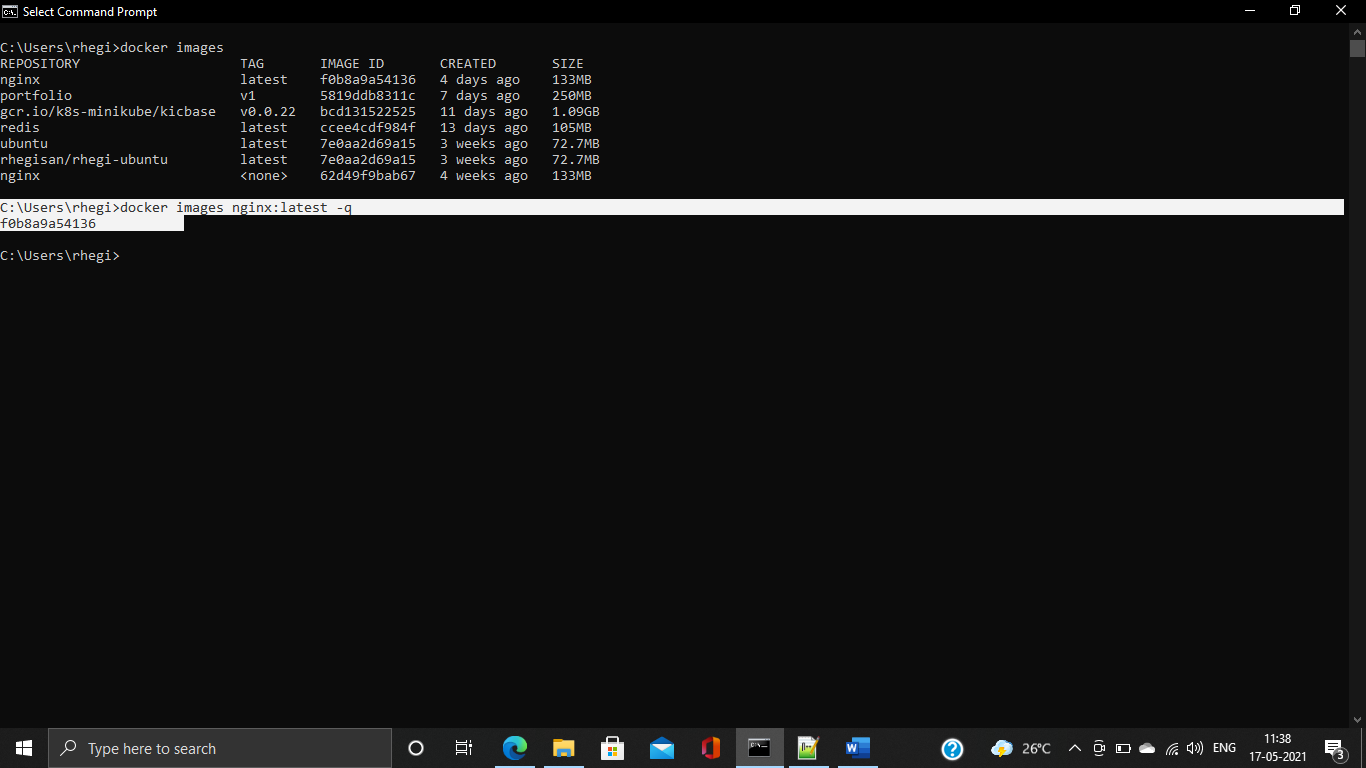
**2. Find a image from dockerhub of your choice(recommeded: nginx), don't use browser, pull the official image from dockerhub**



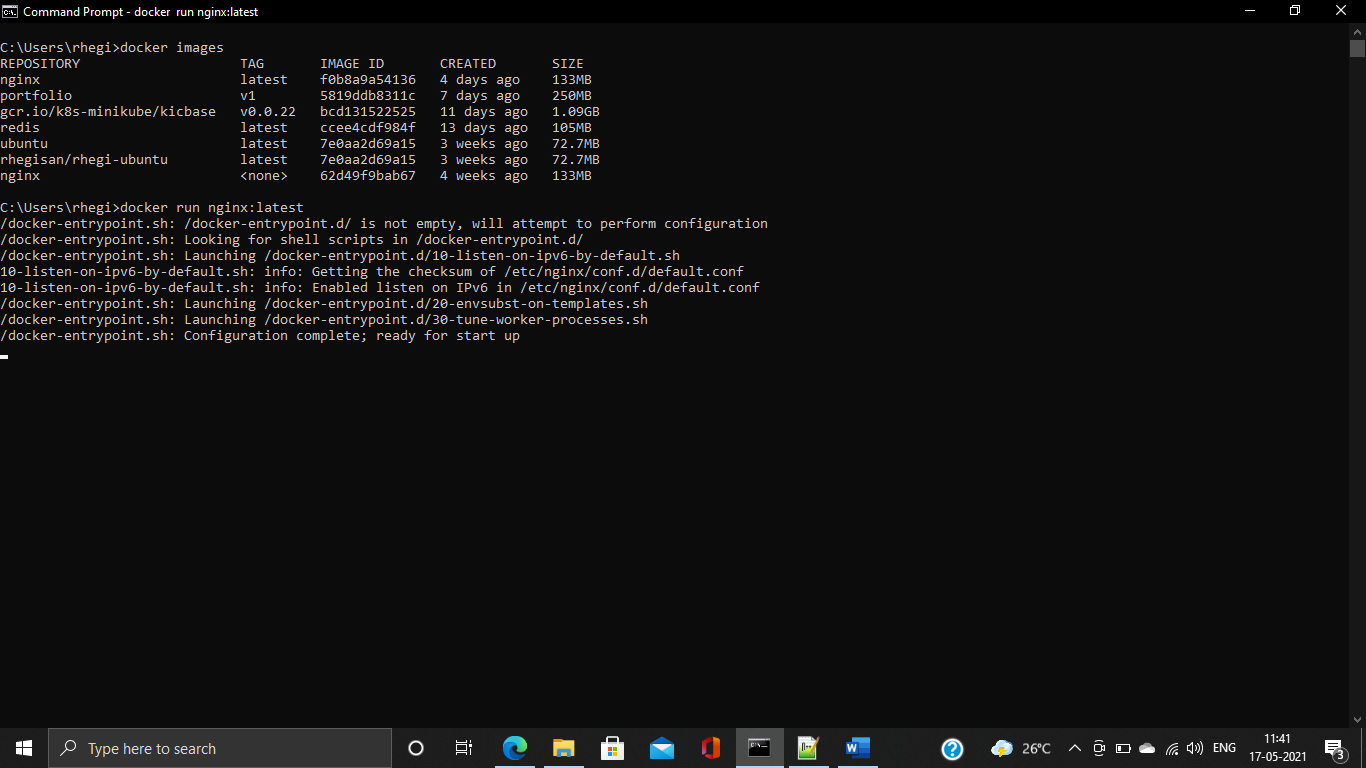
**3. List all the available images in your machine/vm, make sure you see recently pulled image in the list.**



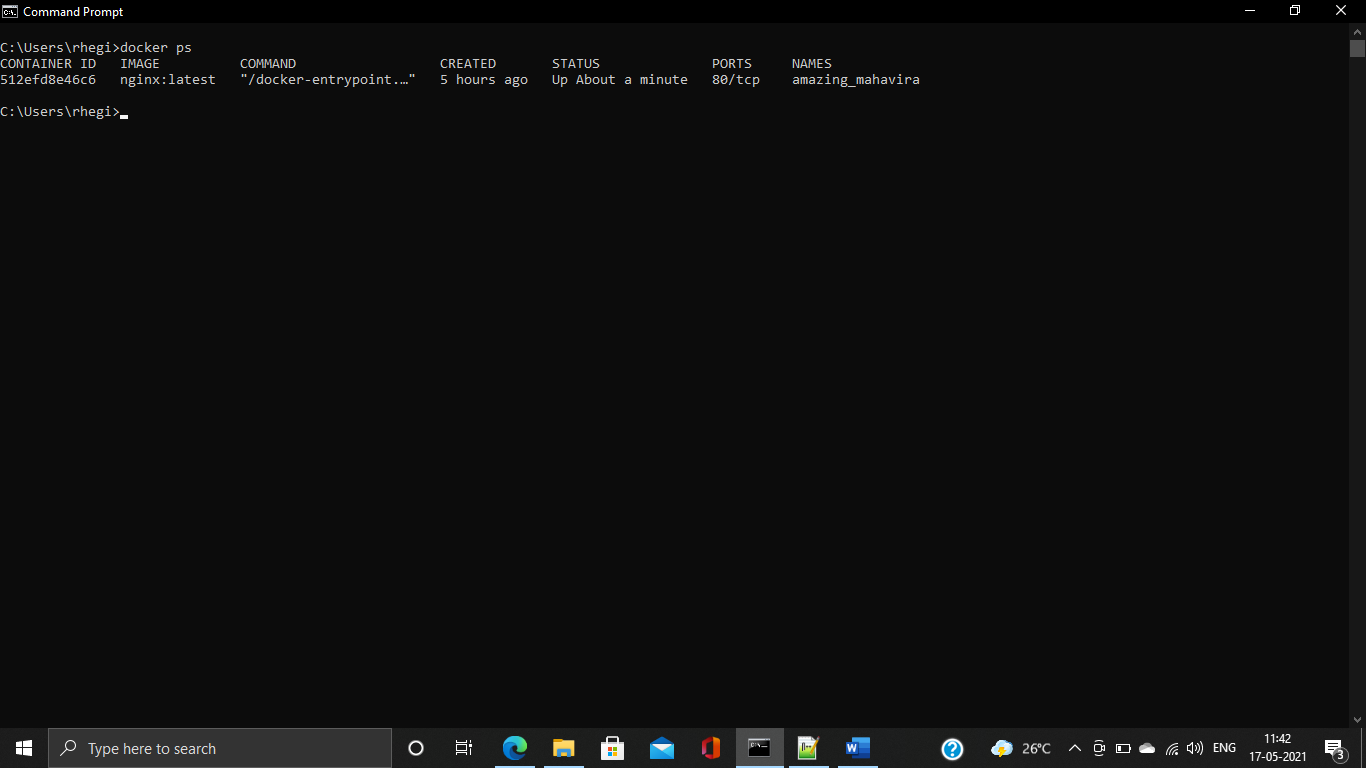
**4. Find out the "Full" ImageId of the image that you pulled and write it below.**



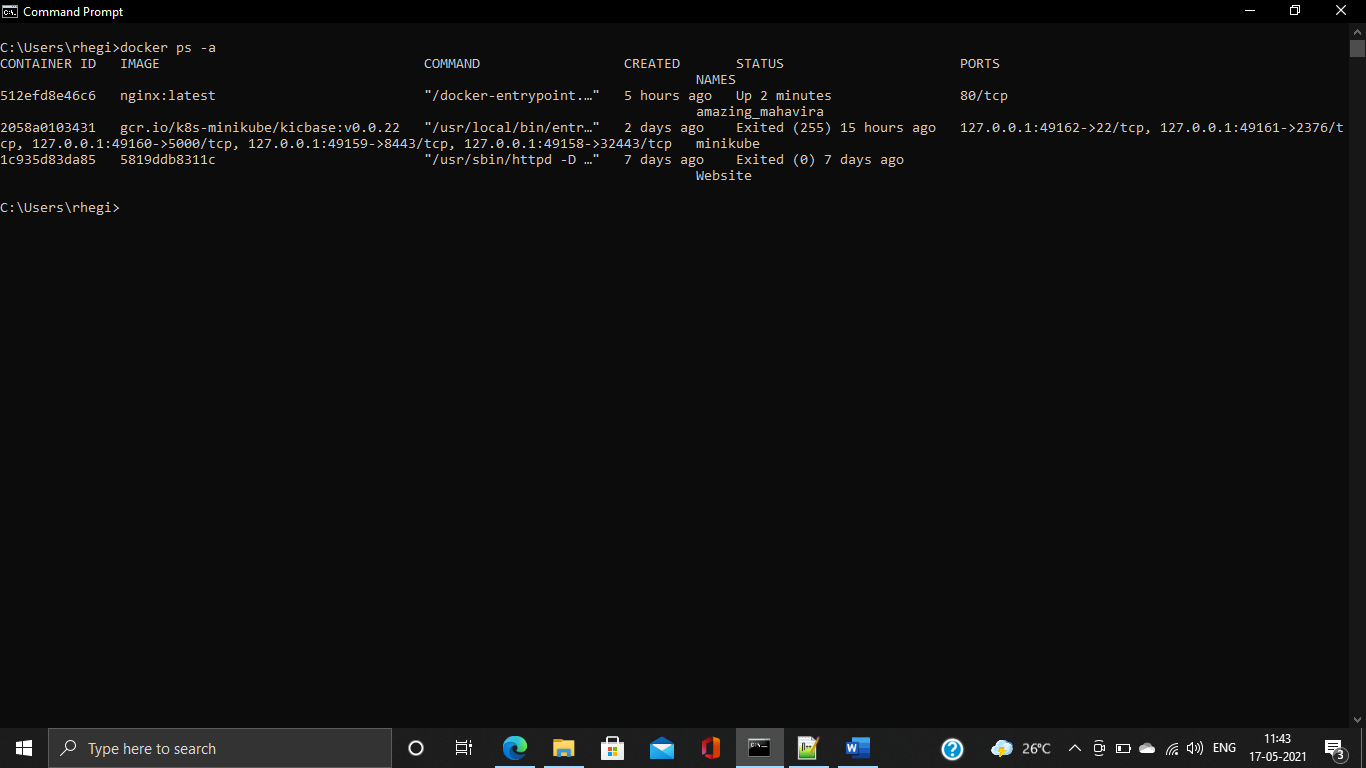
**5. Create a container of your image**



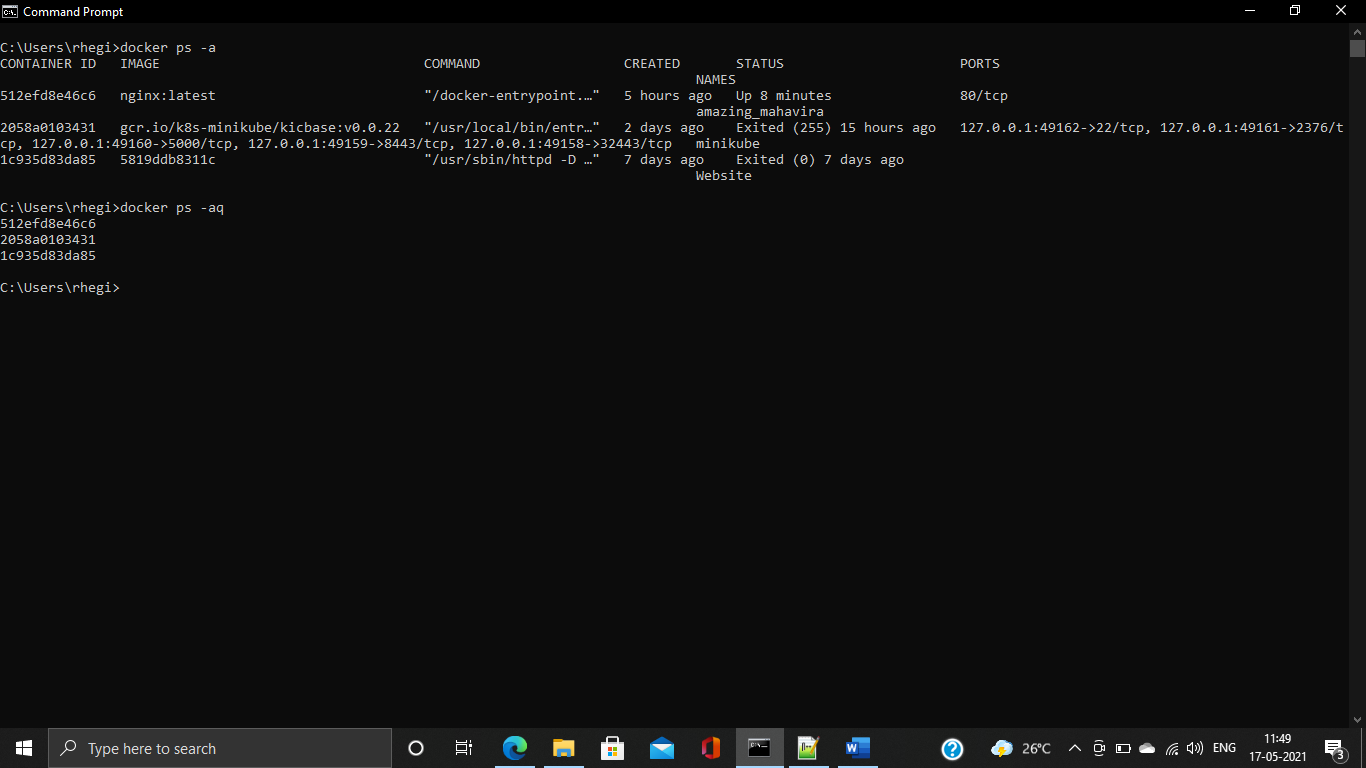
**6. List all the running containers**



**7. List all the running and stopped containers**

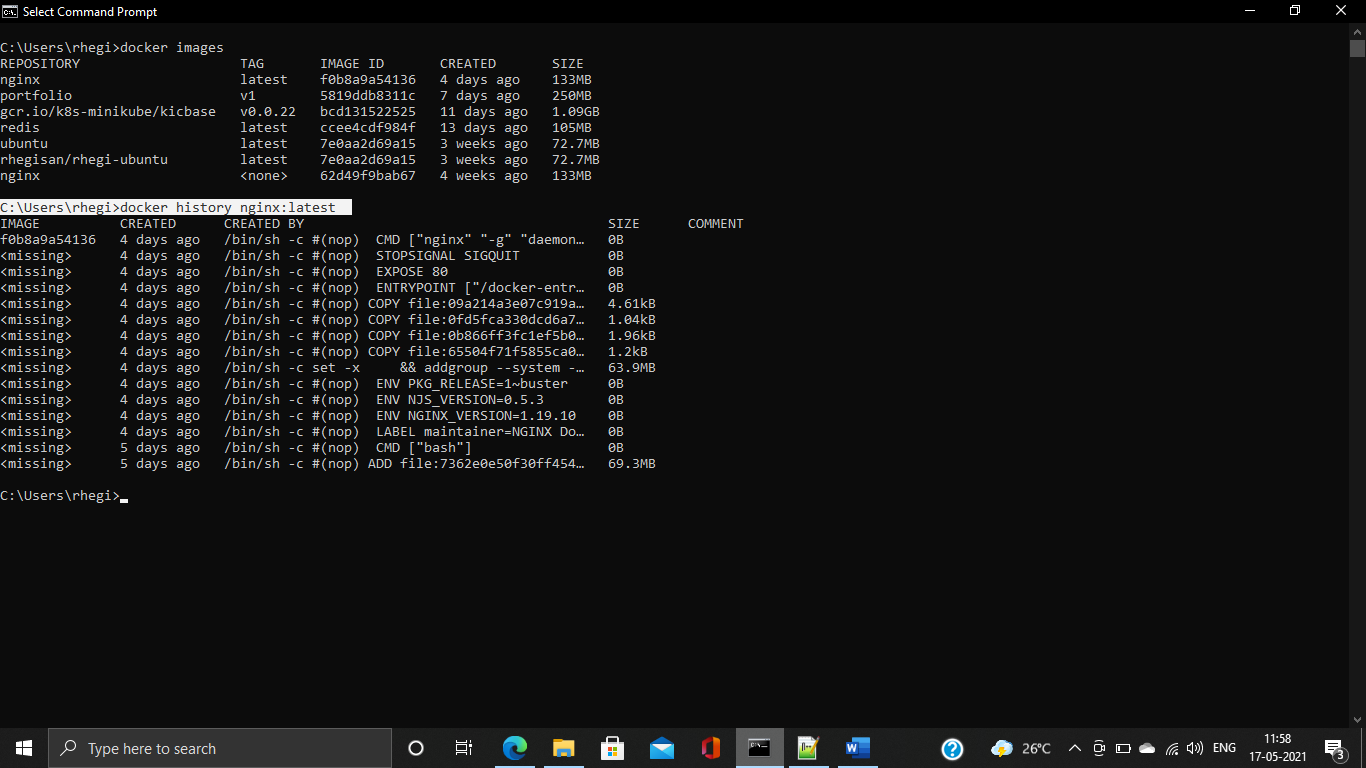


**8. Find out the "Full" containerId of the container and write it below.**



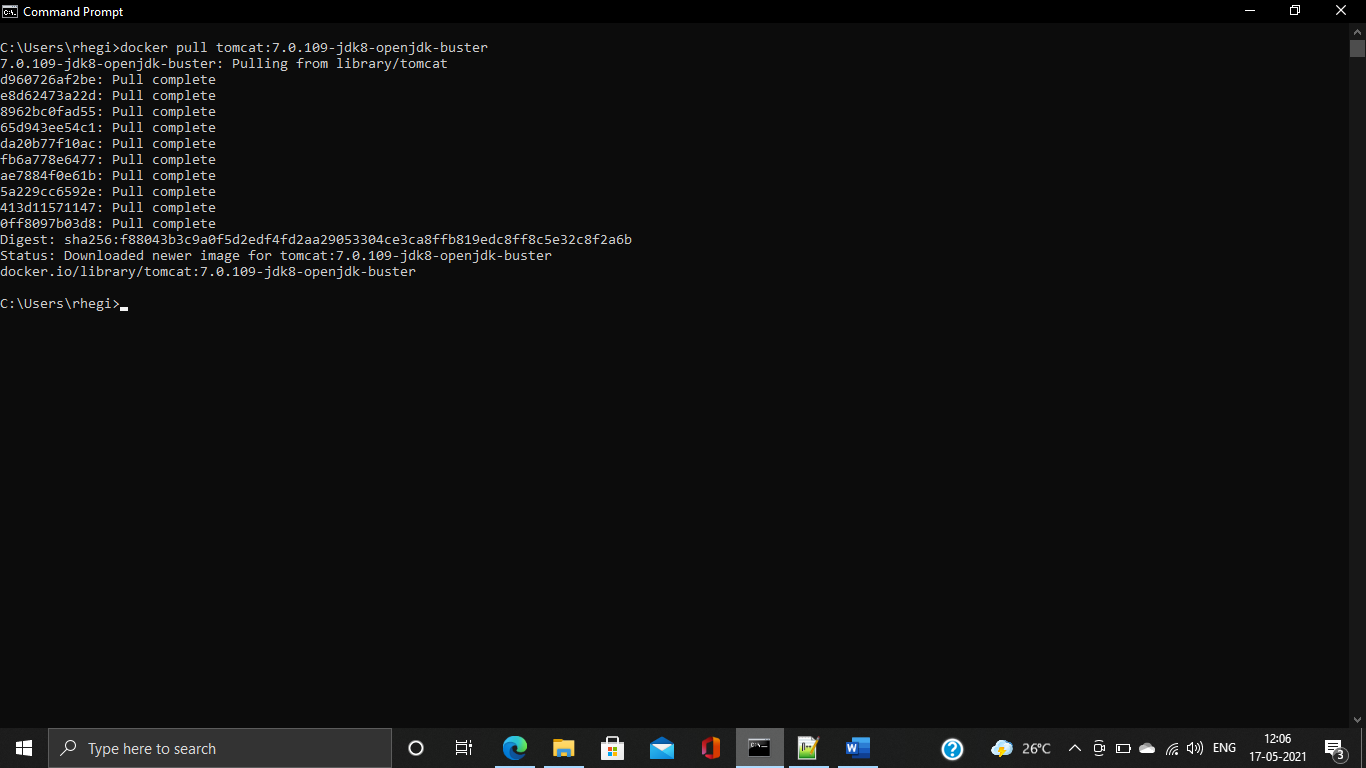
**9. Find out how many image layers are used to build this image.**

Using docker history <image-name>

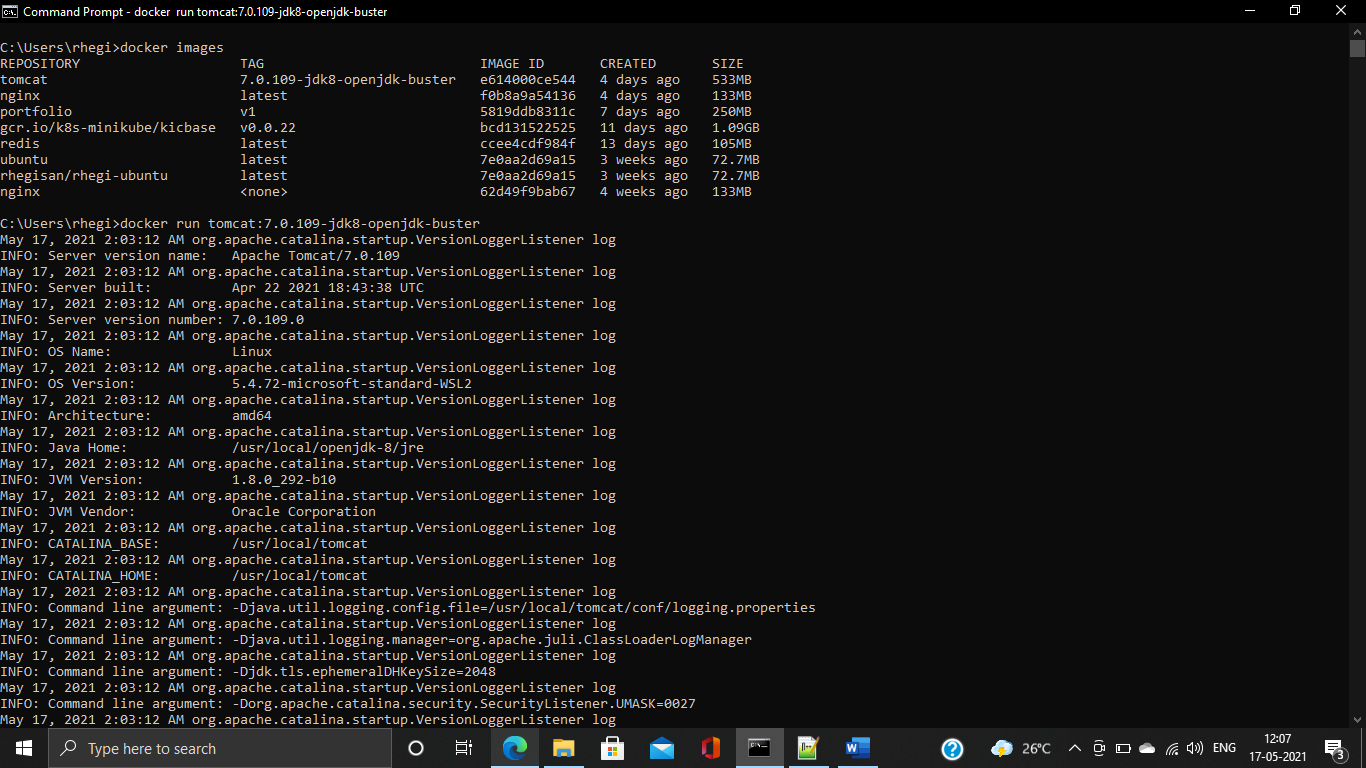


Using docker inspect <image-id>

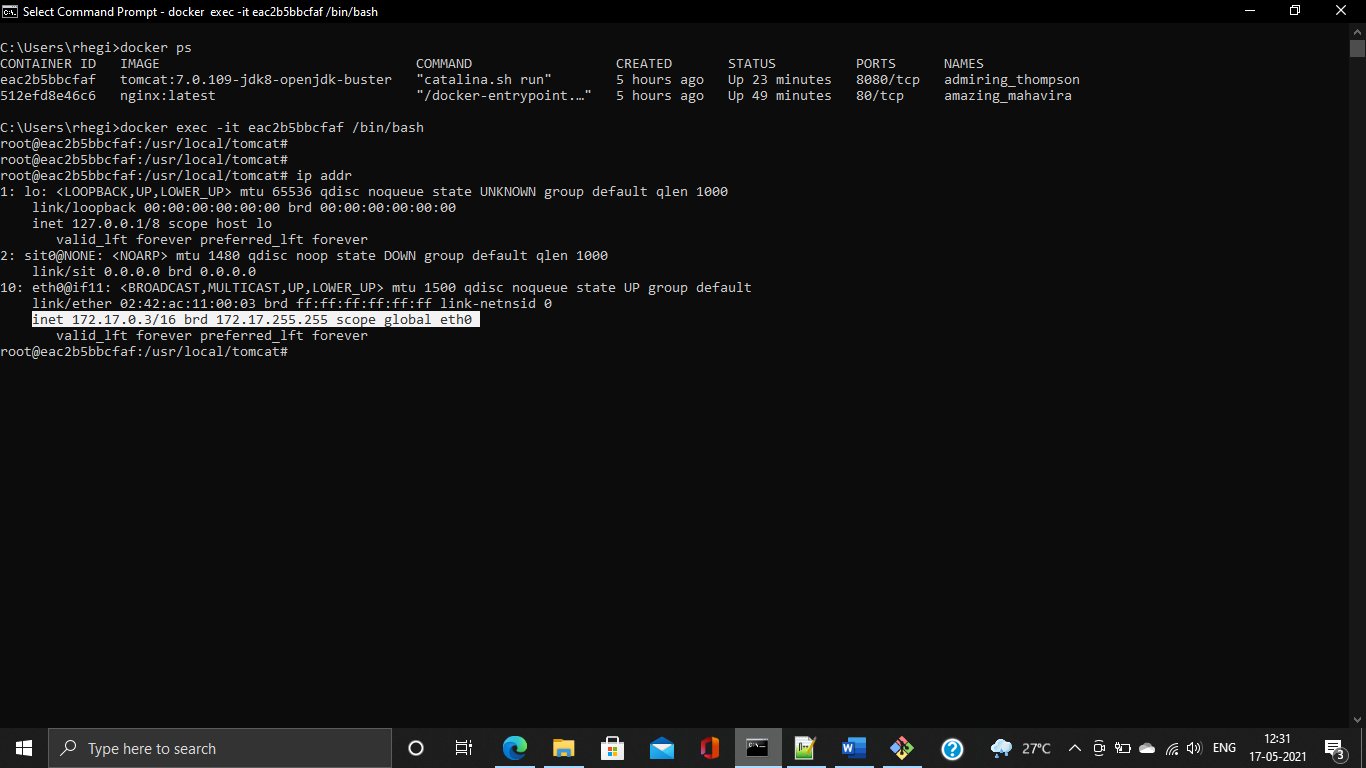
**10. Get the Apache Tomcat 7 server image from the docker hub.**

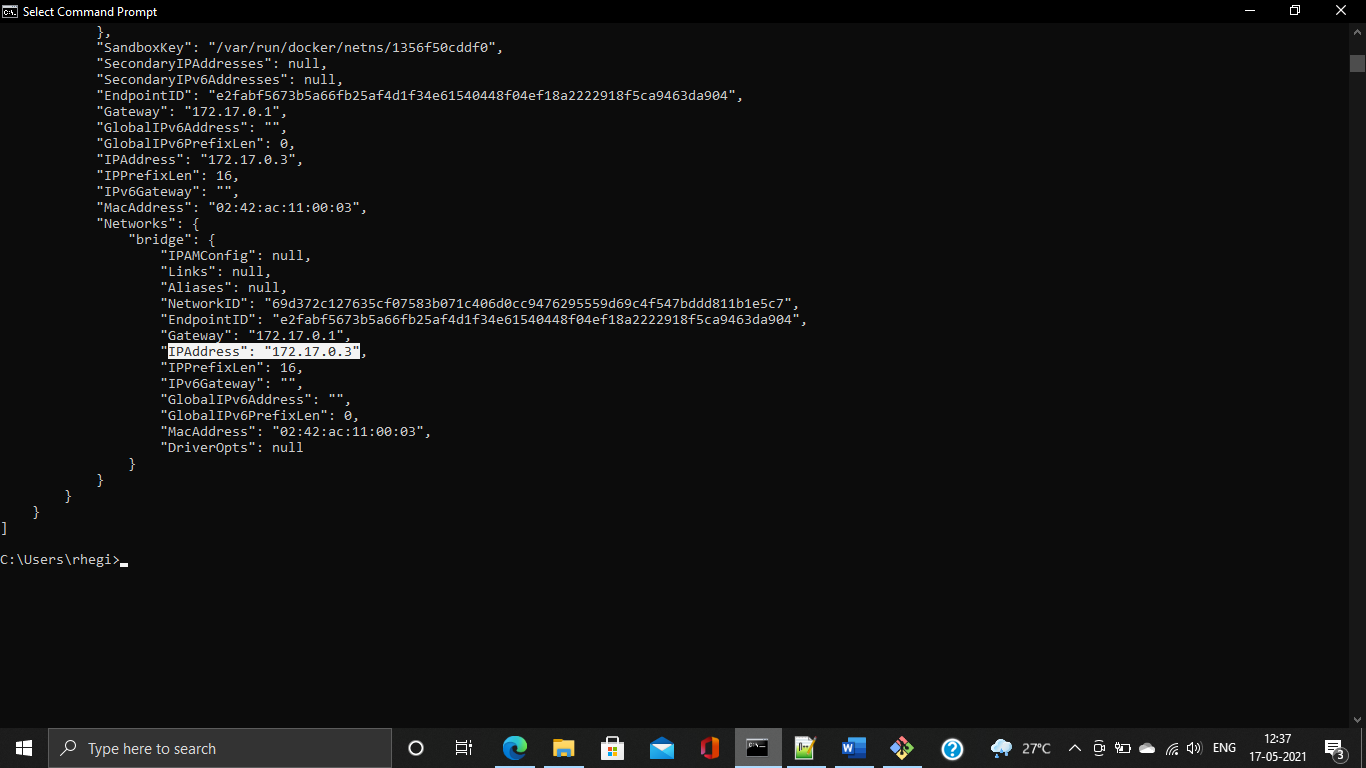


**11. Run the Apache Tomcat 7, I mean create a container of Apache Tomcat.**

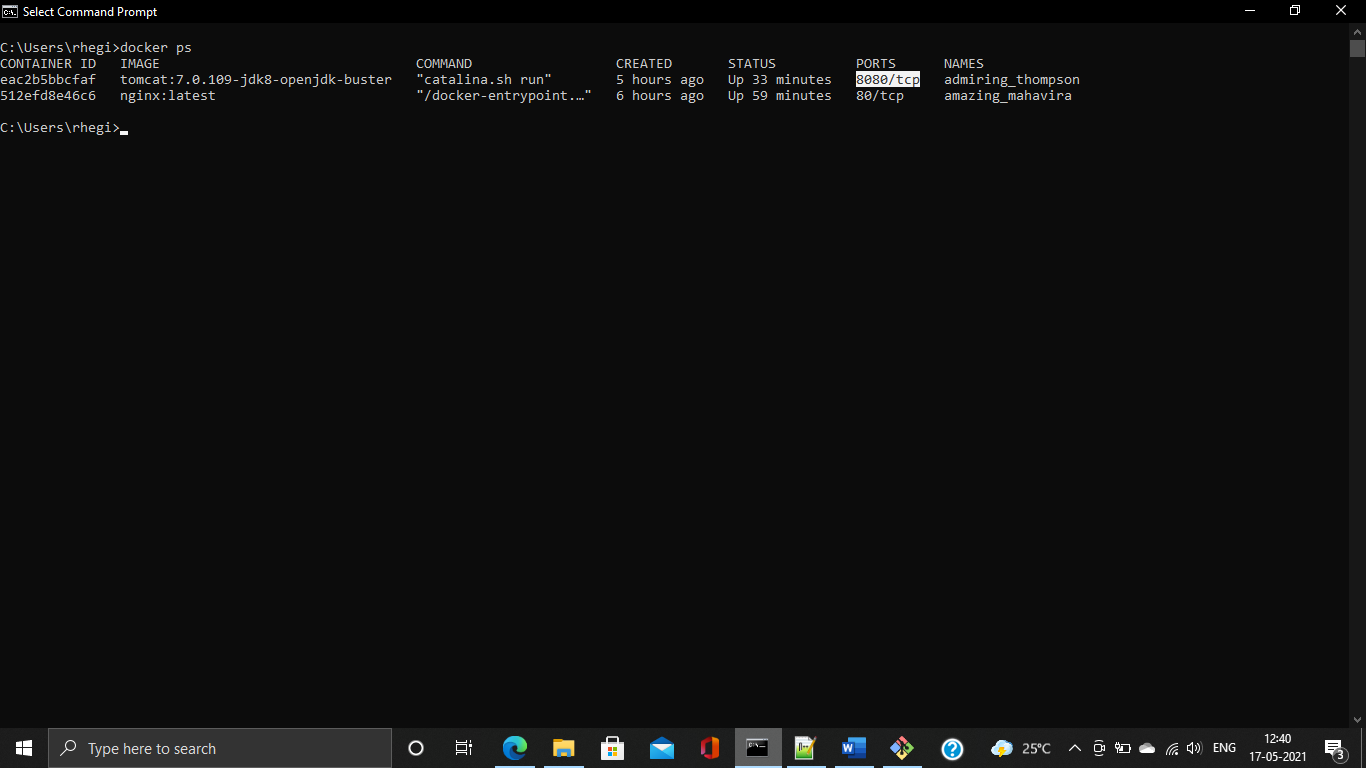


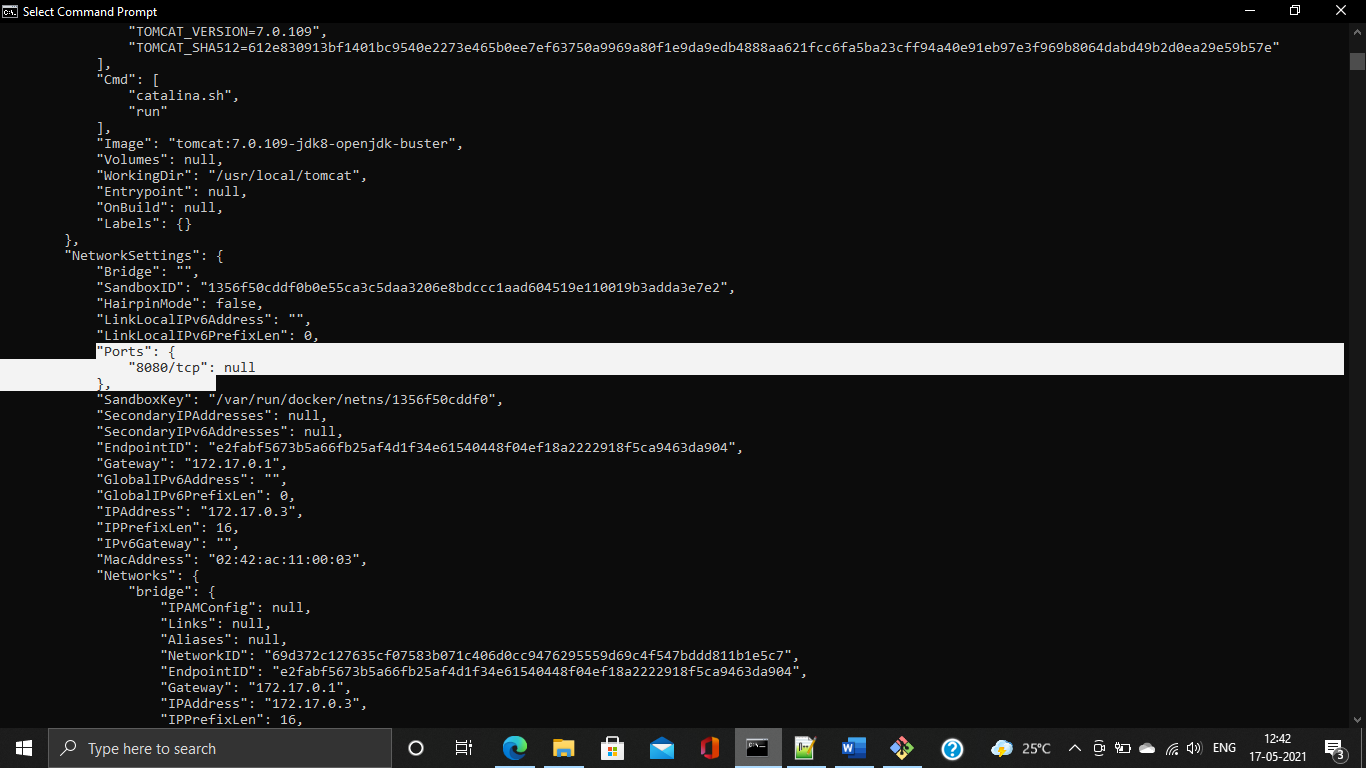
**12. Find out what is the IP Address of the Apache Tomcat Container that it is running on**

Using ip addr 

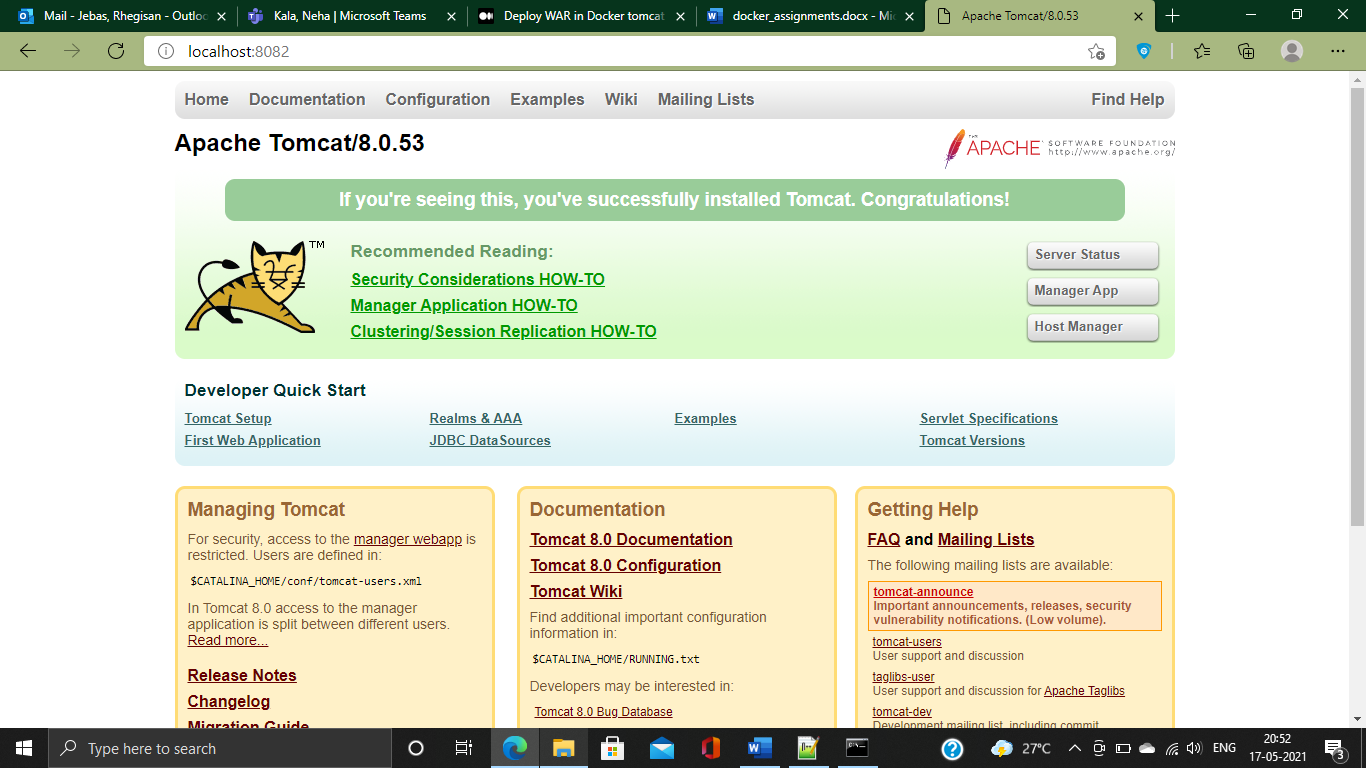
Using docker container inspect <container-id> 

**13. Which Port it is using?**

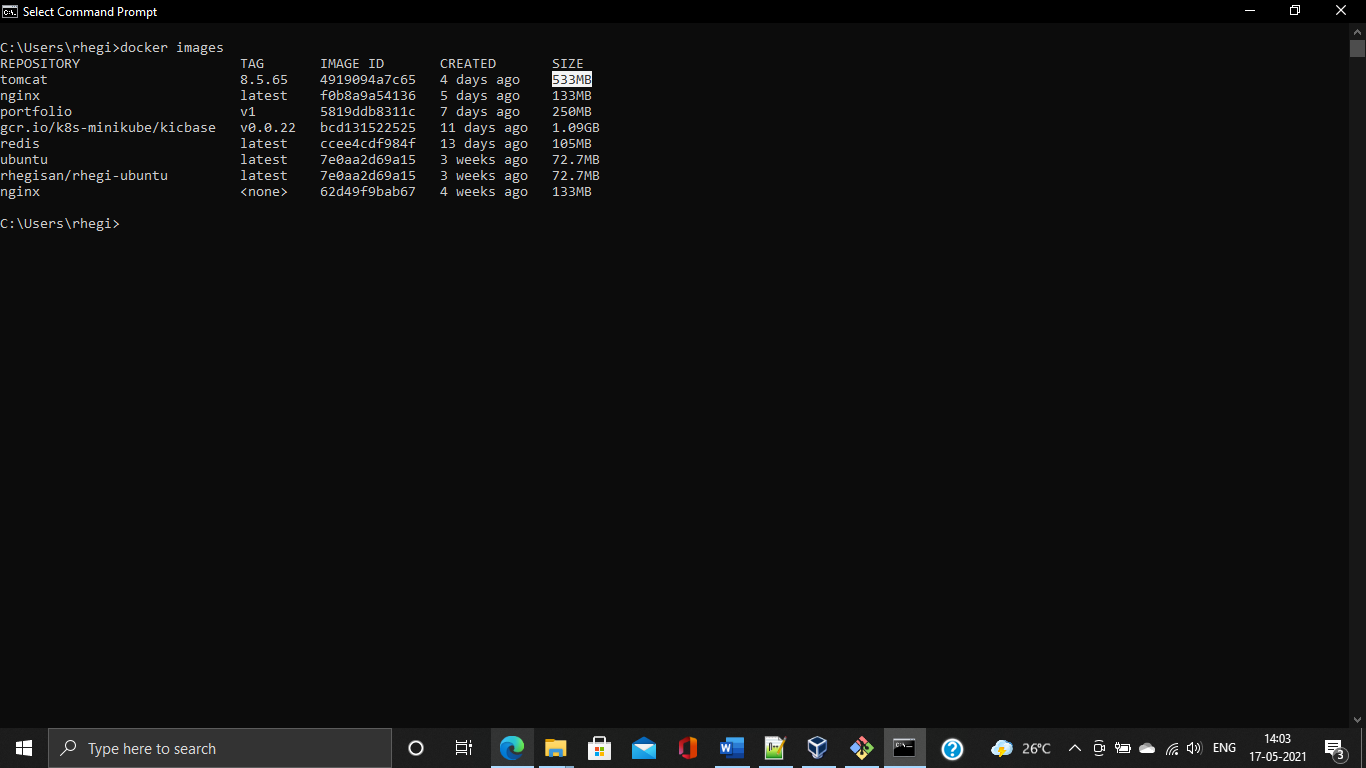


Using docker container inspect <container-id>

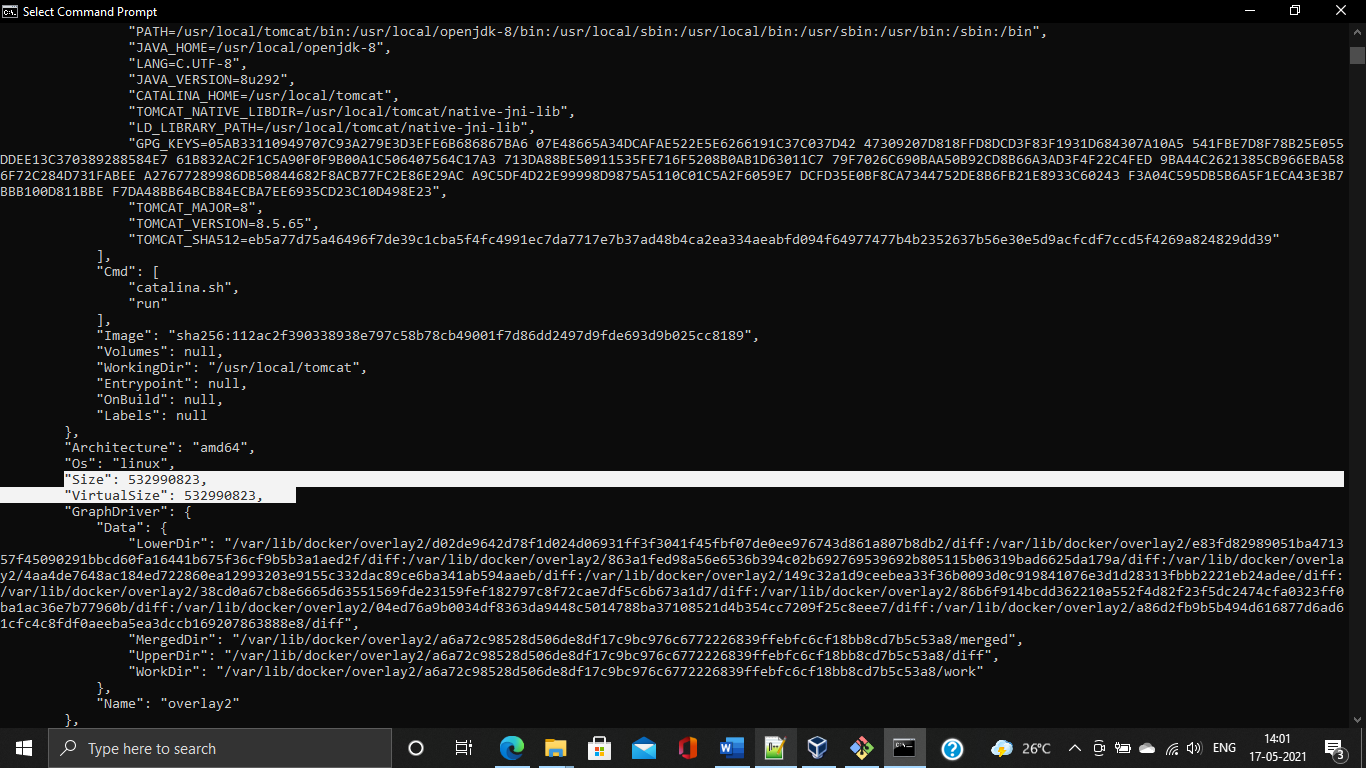
**14. Try to access the Tomcat's home page from your machine/vm.**



**15. What is the disk size of Apache Tomcat image?**

Using docker images

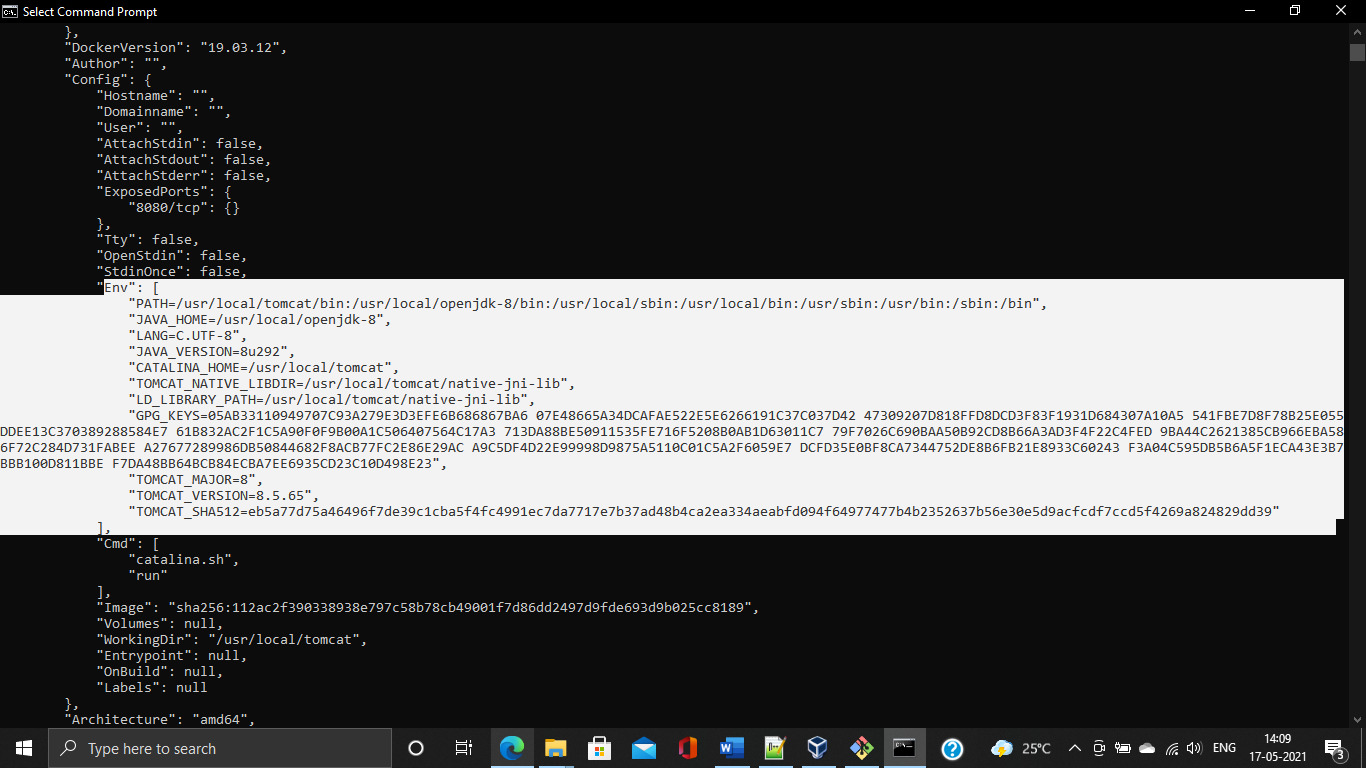
Using docker images inspect 4919094a7c65



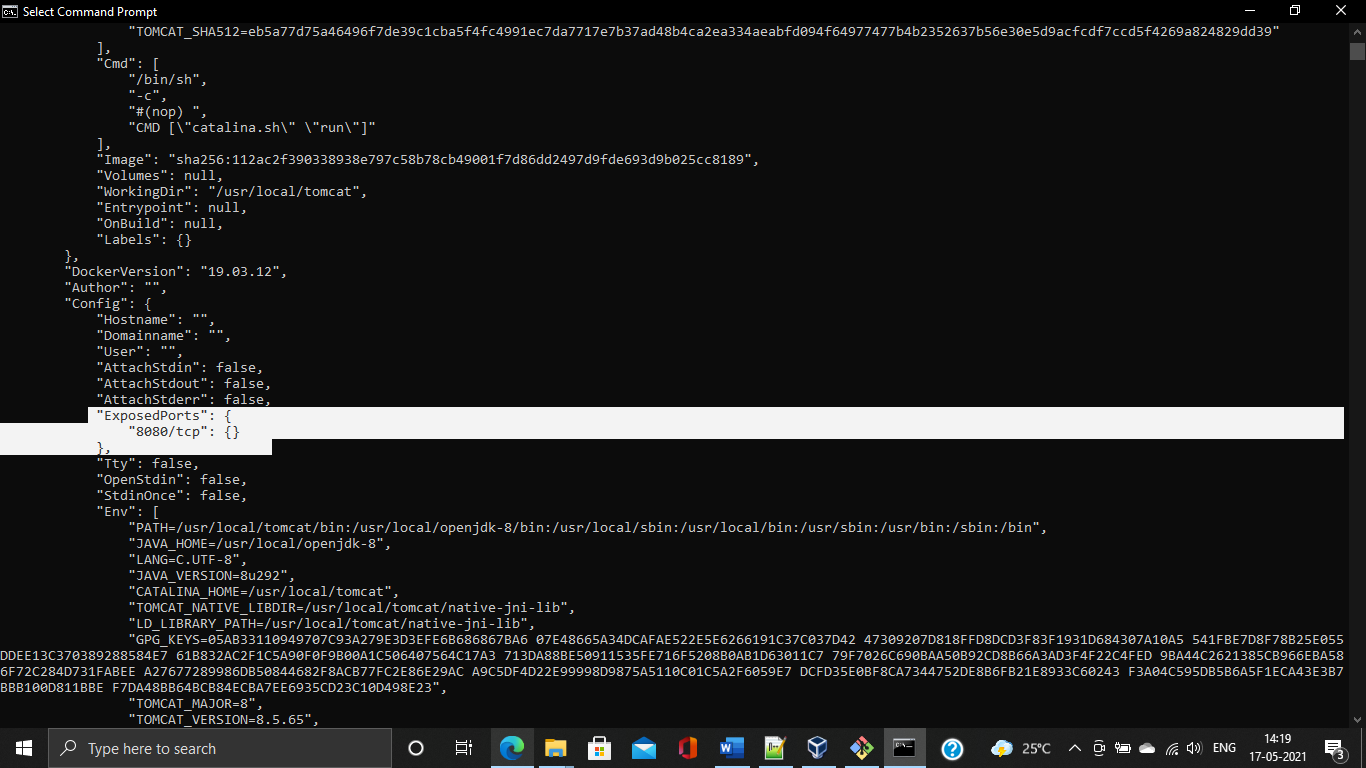
**16. Find out list of all environment variables that is configured for tomcat image, can you see JAVA\_HOME and CATALINA\_HOME? What did you notice about it?**

CATALINA\_HOME :- It is the folder where you unzip Tomcat

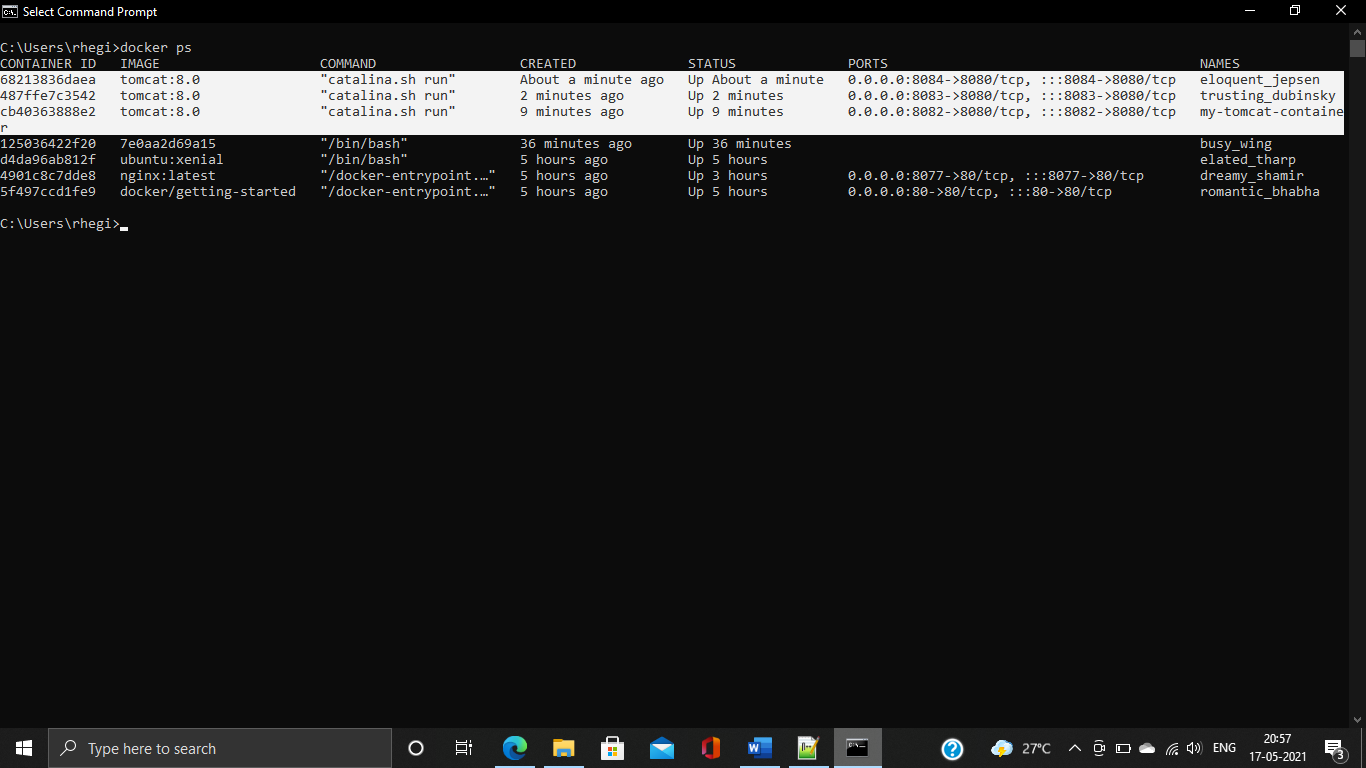
JAVA\_HOME :- Apache Tomcat expects this environment variable to be set to the installation directory of the JDK

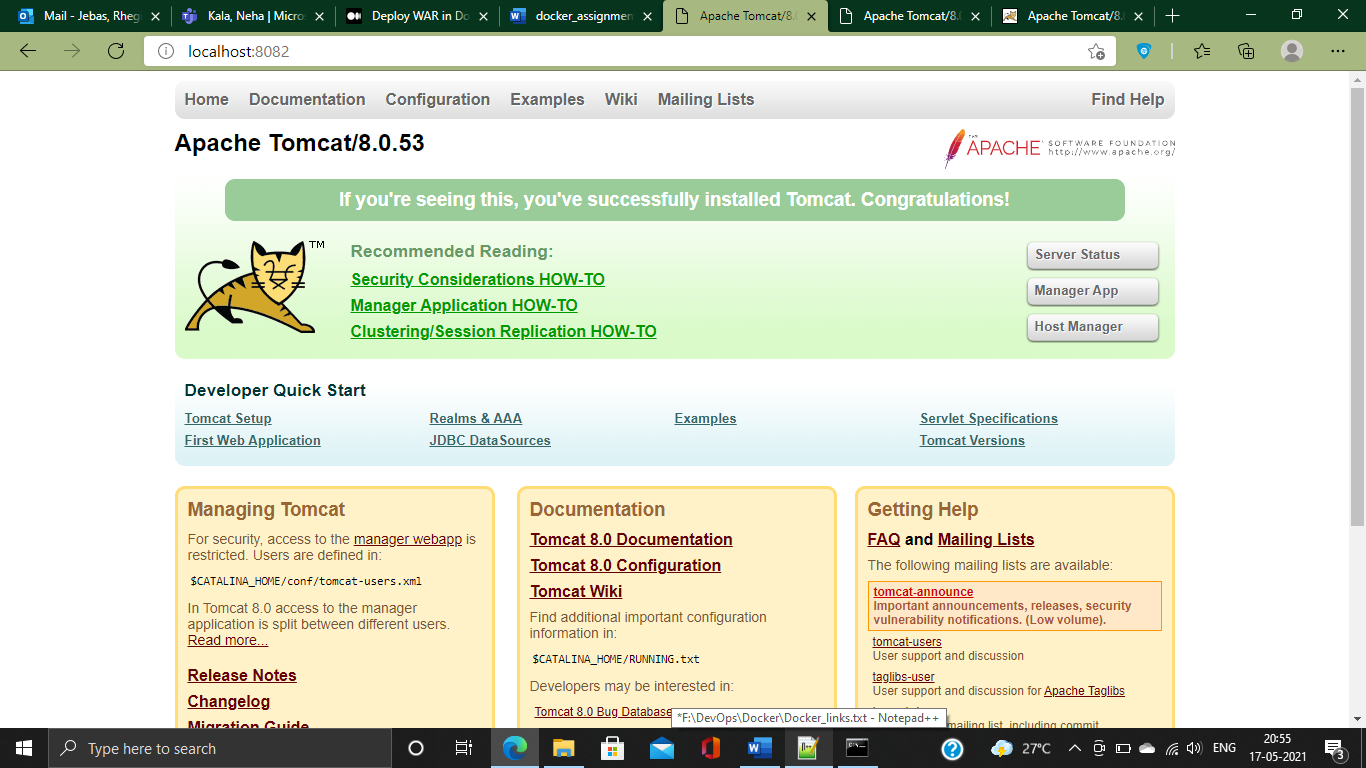


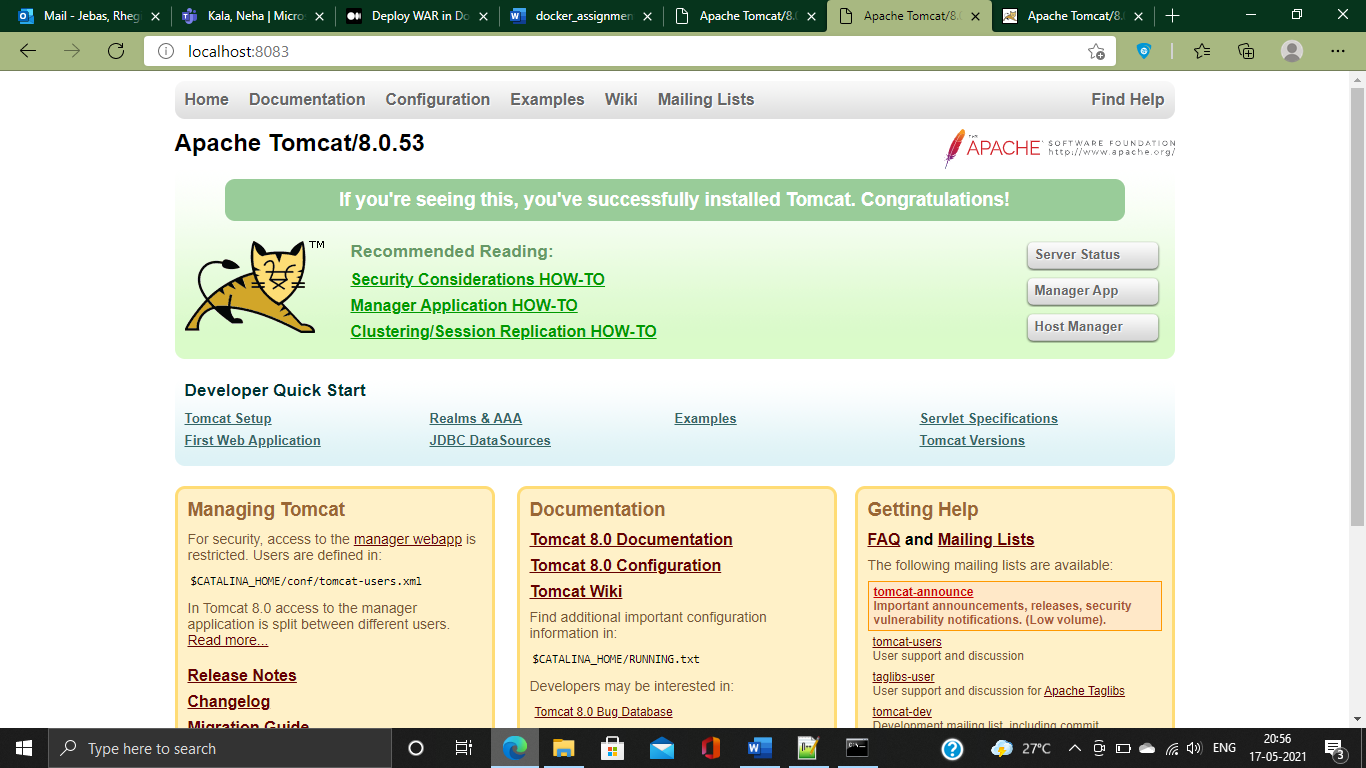
**17. Find out which port is exposed for tomcat?**

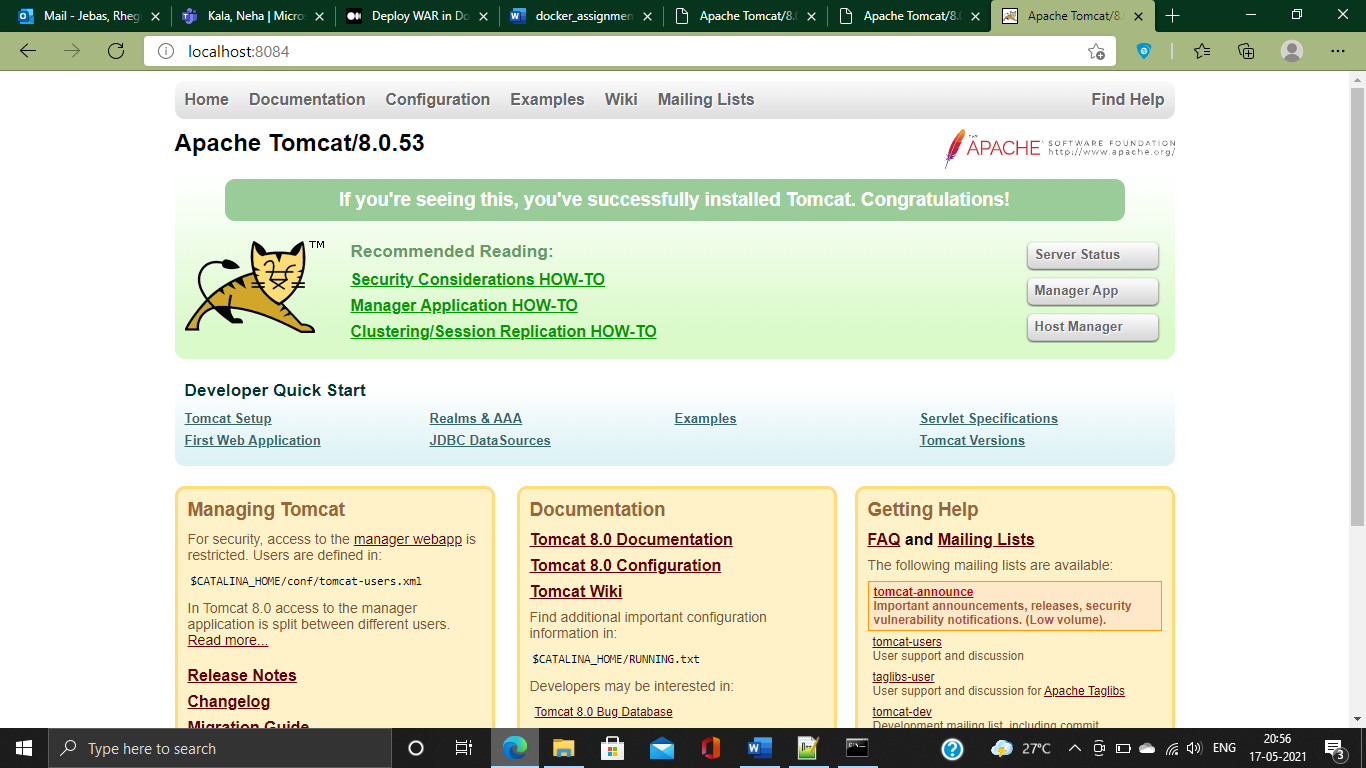


**18. Run multiple containers of tomcat on different port and access it's home page.**

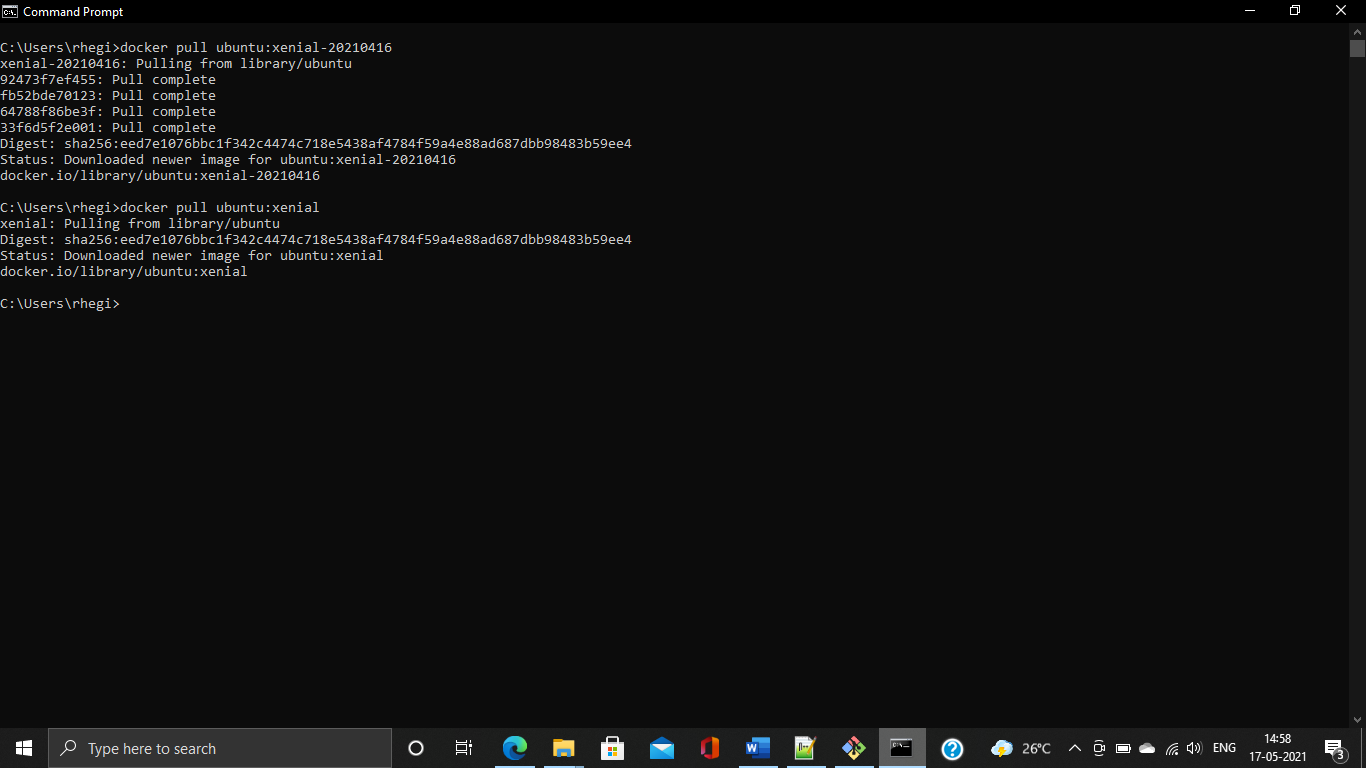




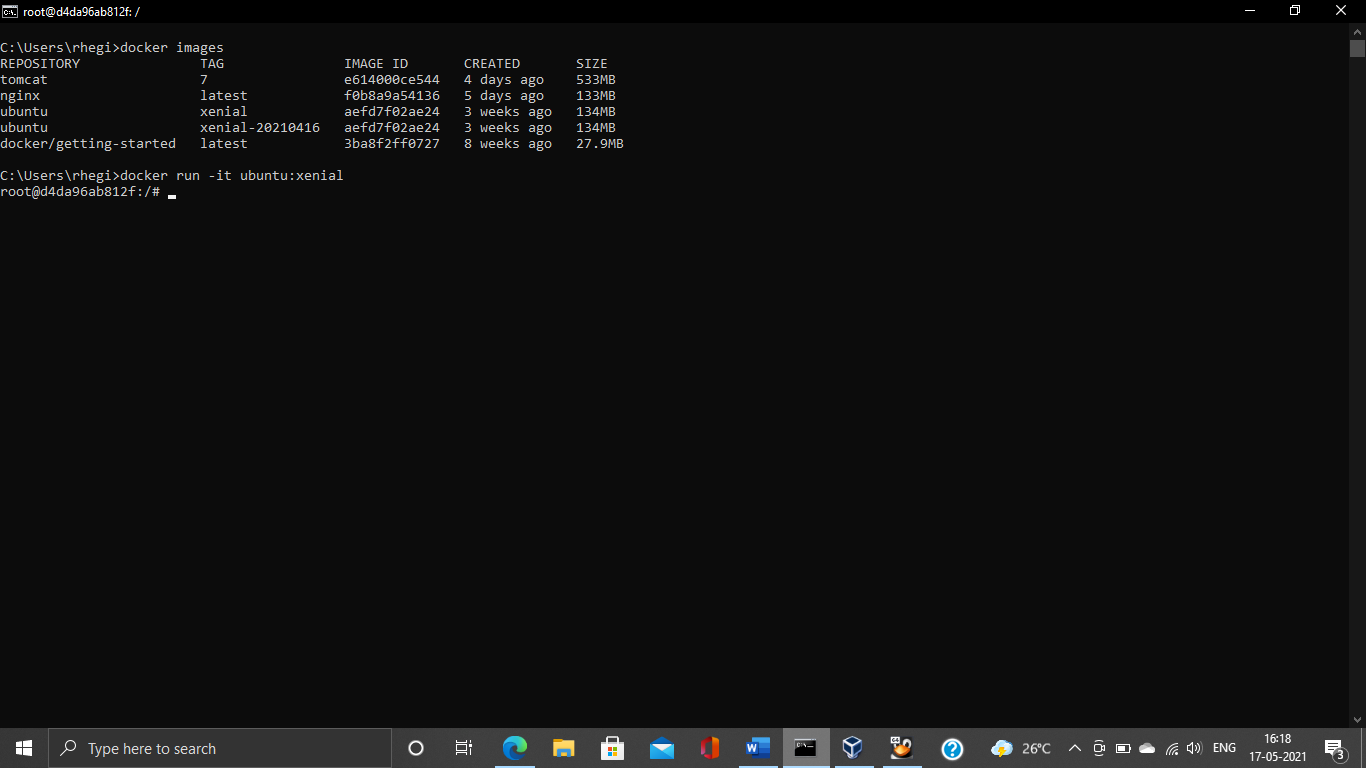




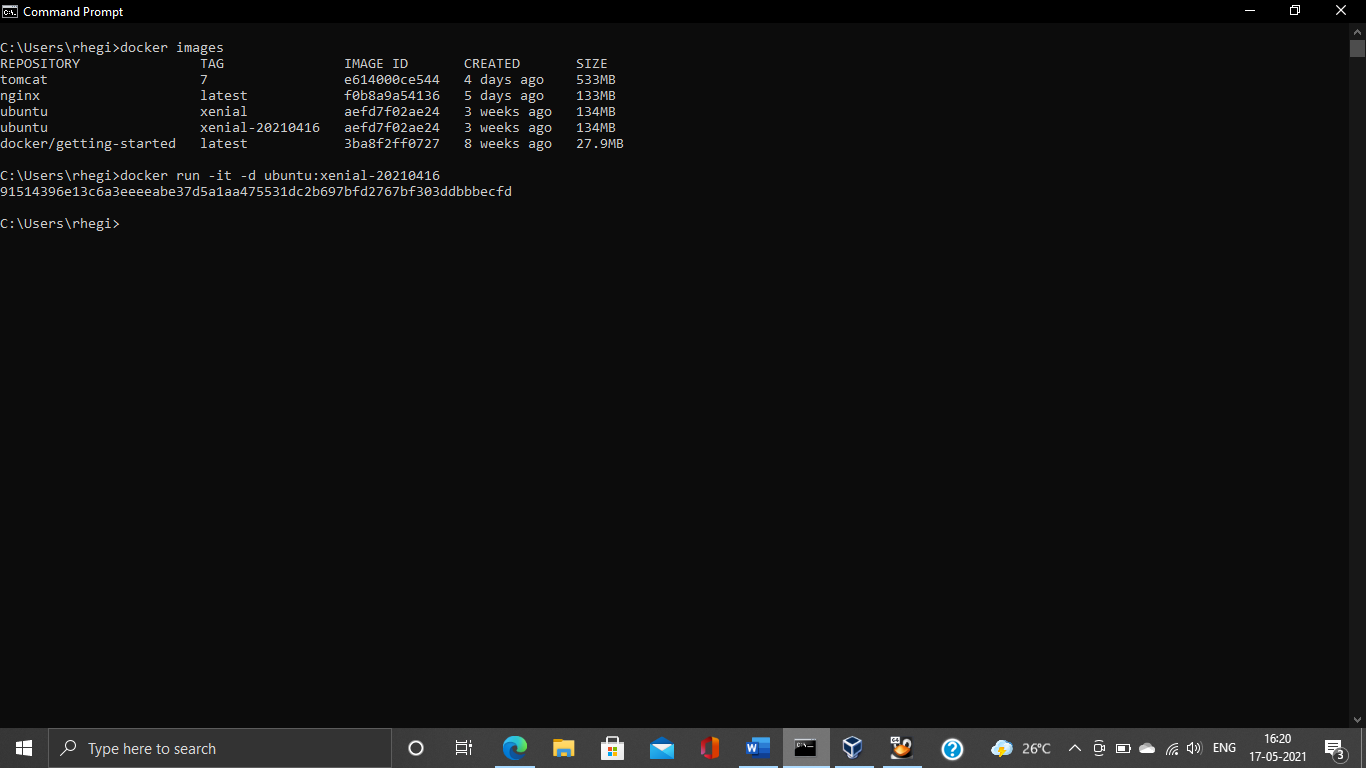
**19. Pull ubuntu os from dockerhub, try to pull 2 images of ubuntu, Except the latest one.**



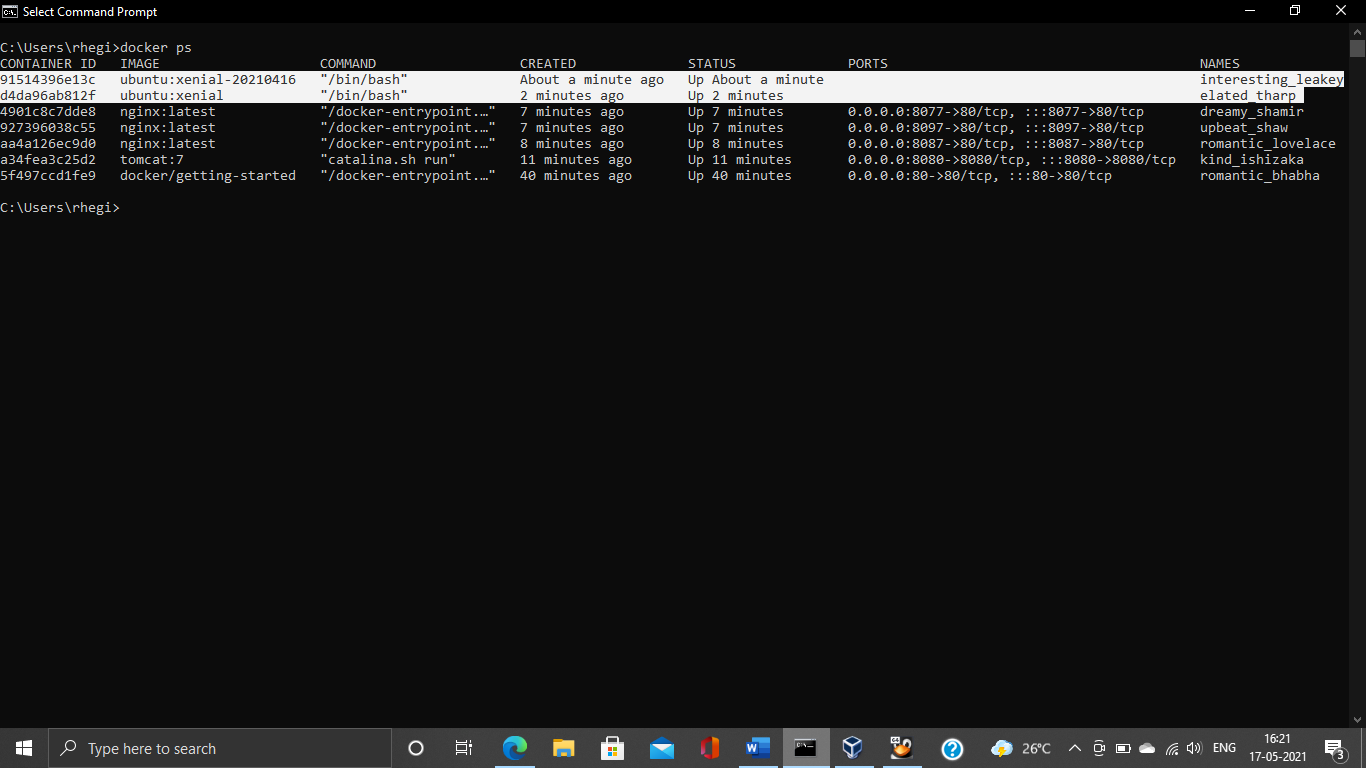
**20. Run the container of ubuntu in attached mode.**



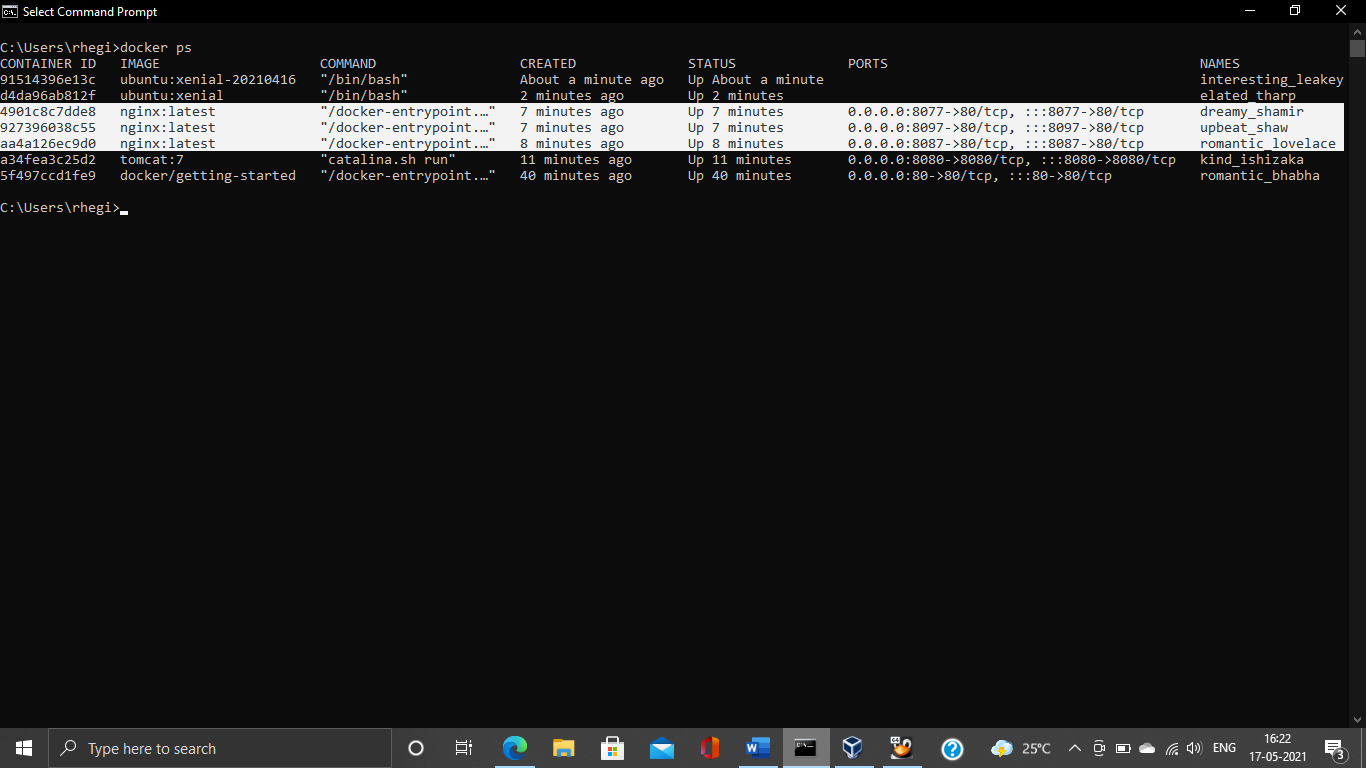
**21. Run the container of another ubuntu in detached mode.**



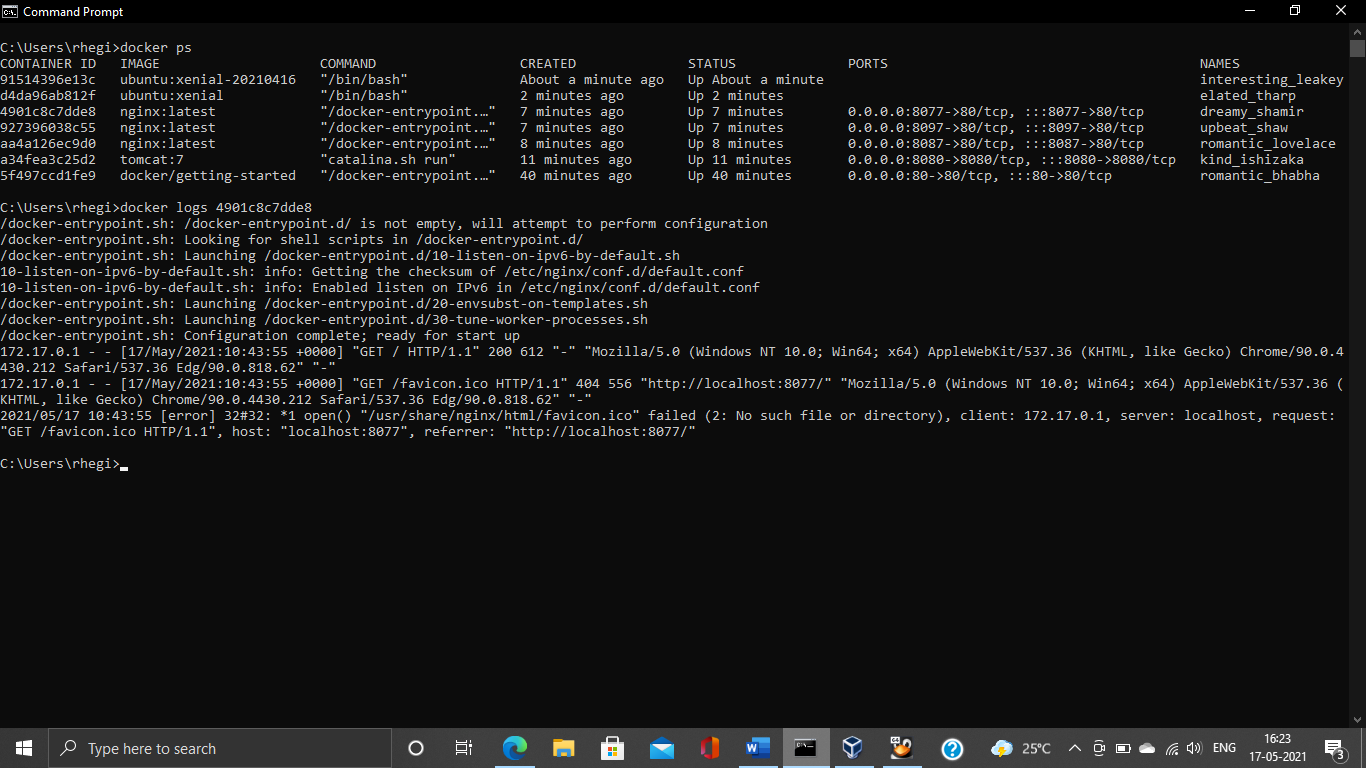
**22. Check how many ubuntu containers are running and stopped**



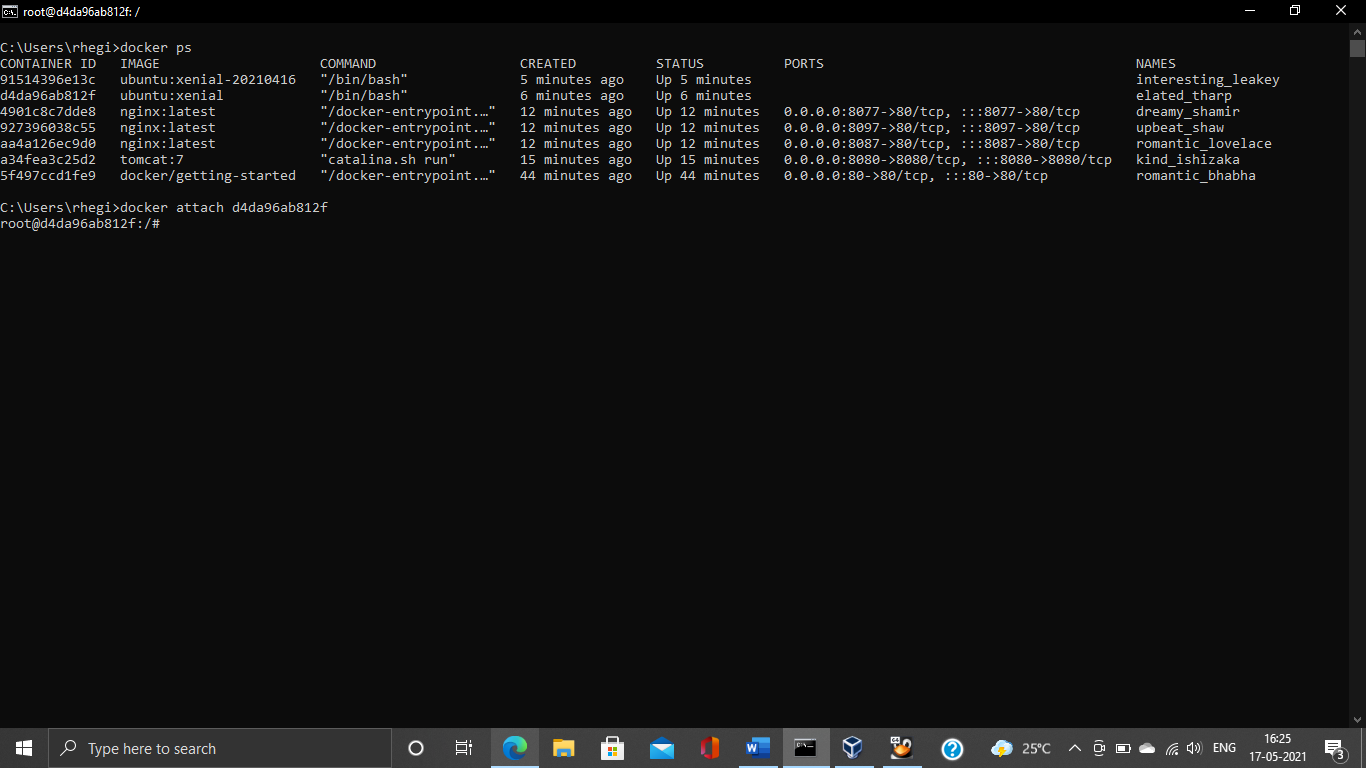
**23. Is the tomcat container running? If no, start one.**



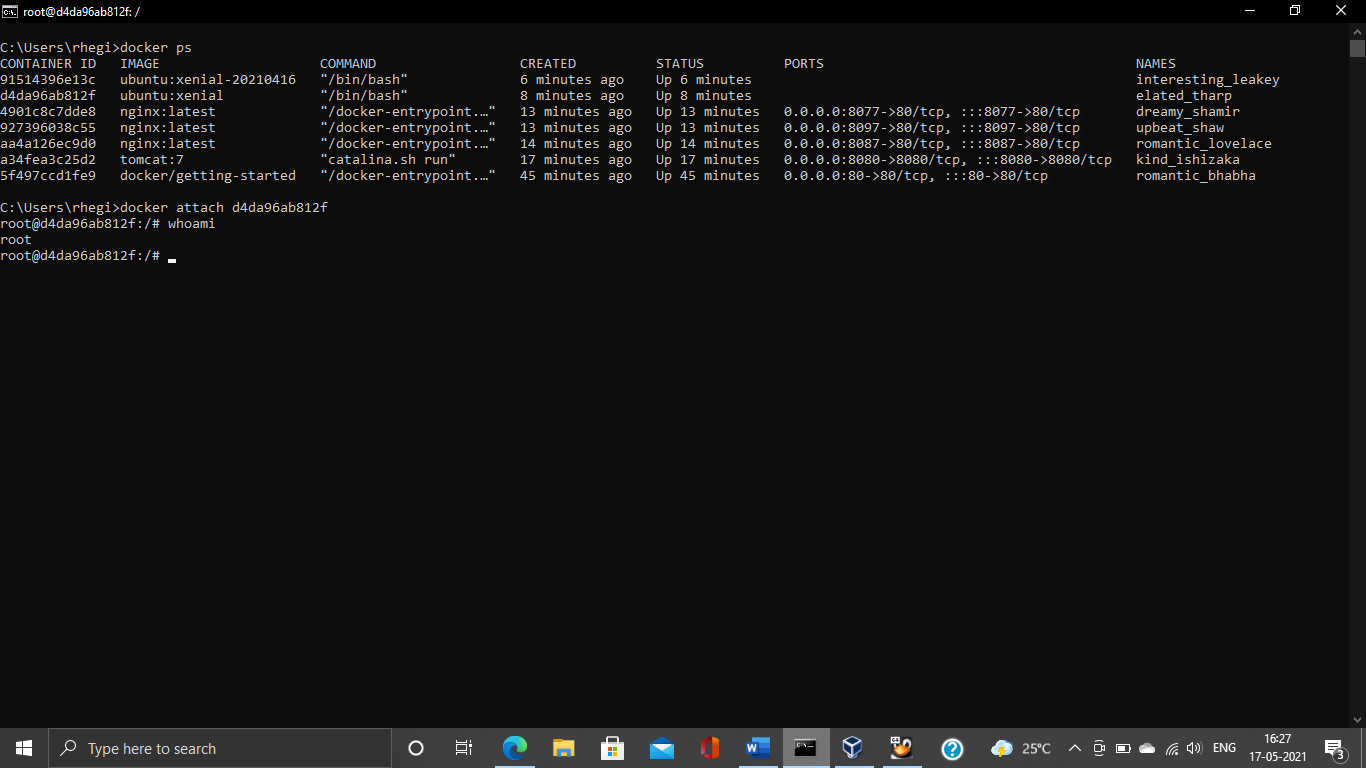
**24. Check the logs, generated by tomcat container(don't forget to make request to tomcat's home page to see the log).**



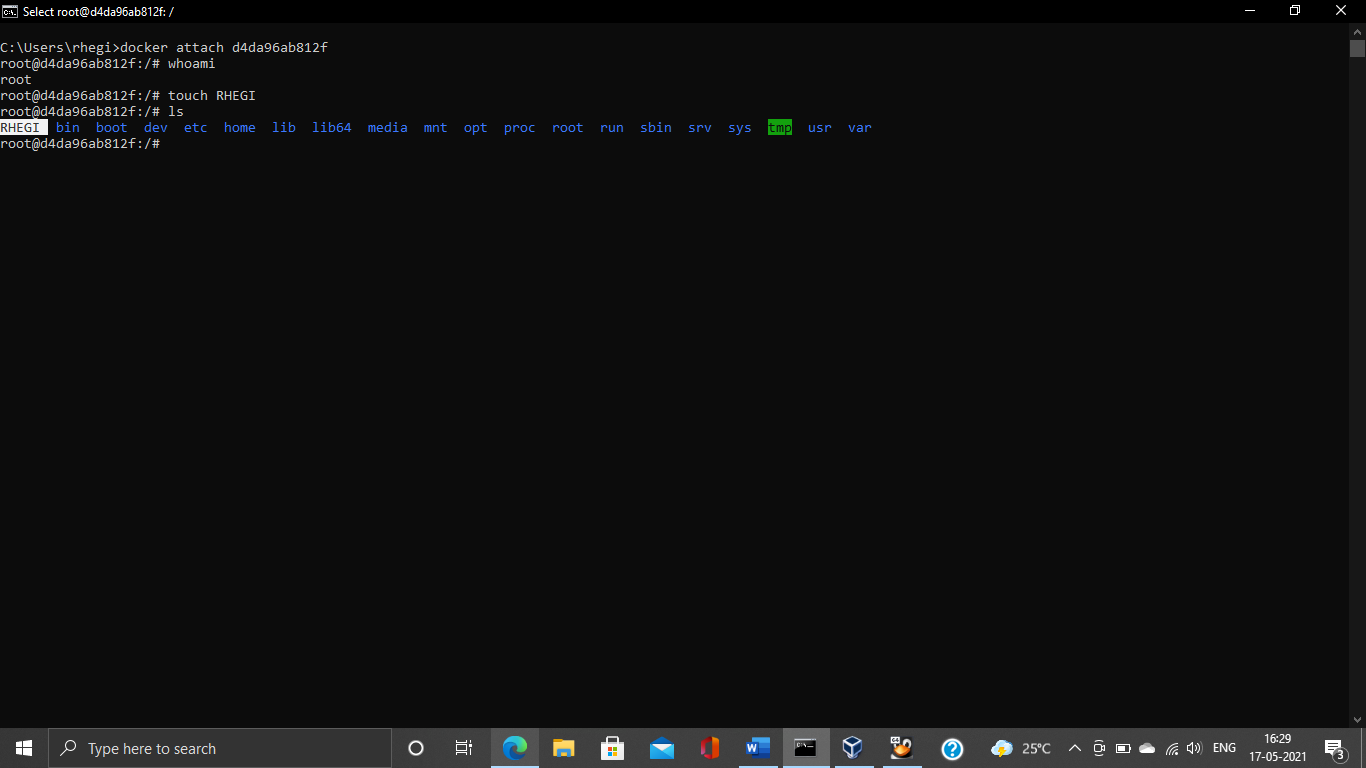
**25. Check if ubuntu conatiner is running? If no, start one in attached mode to the terminal.**



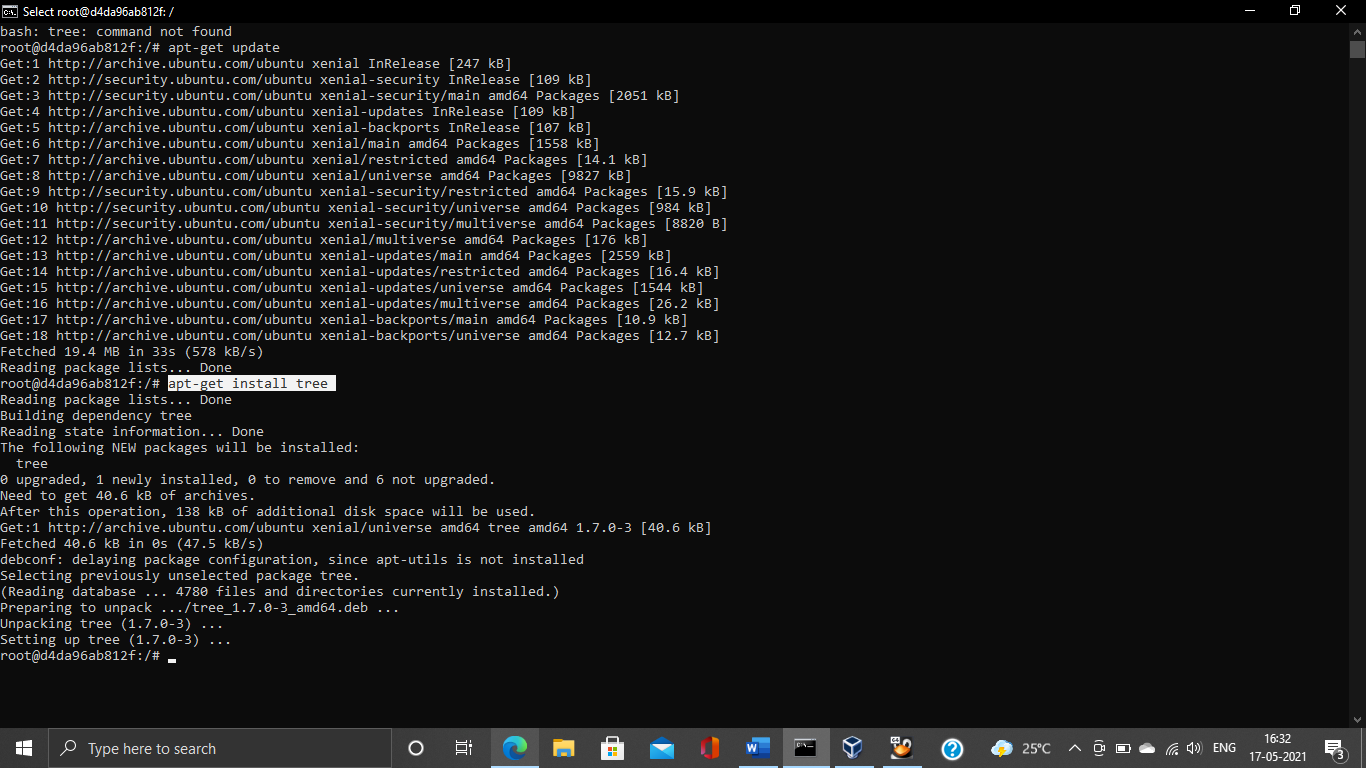
**26. Login as root user in ubuntu container**



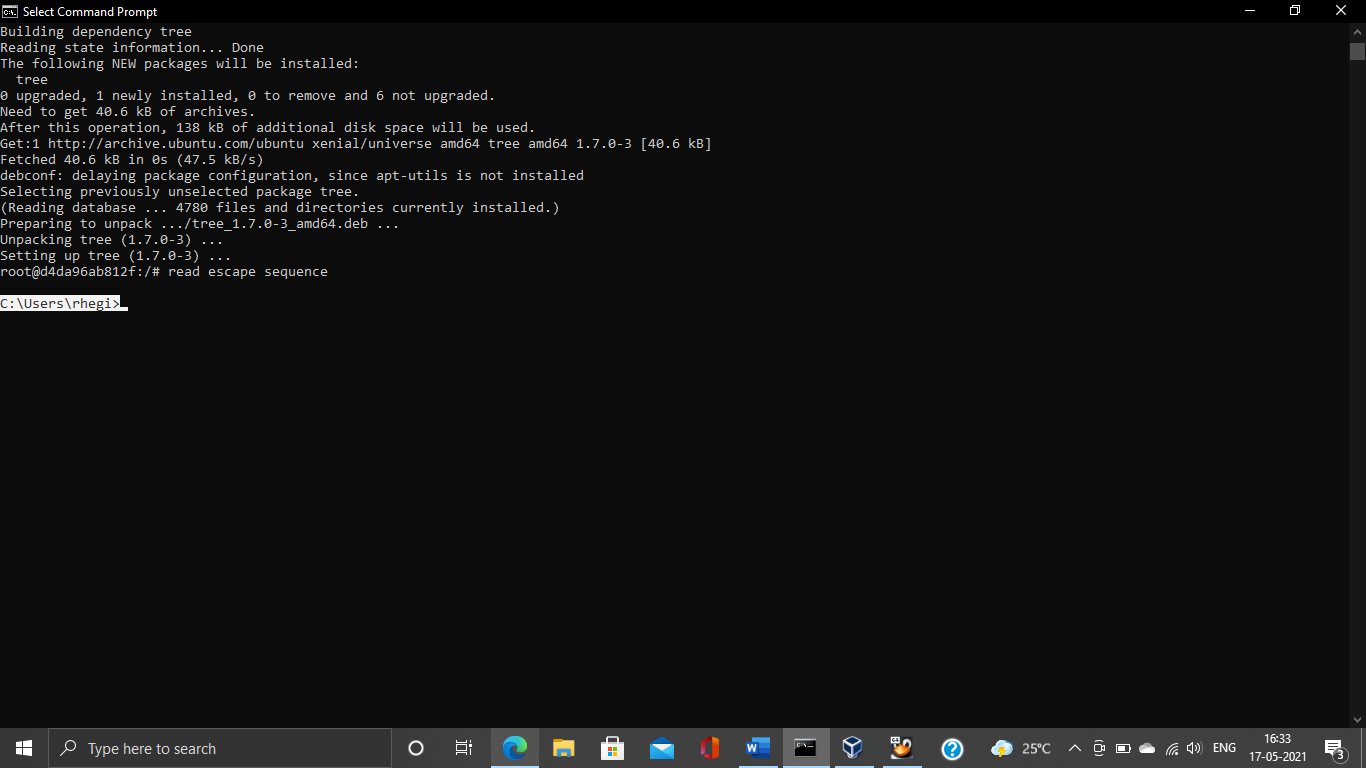
**27. Create a file with any name in root directory**



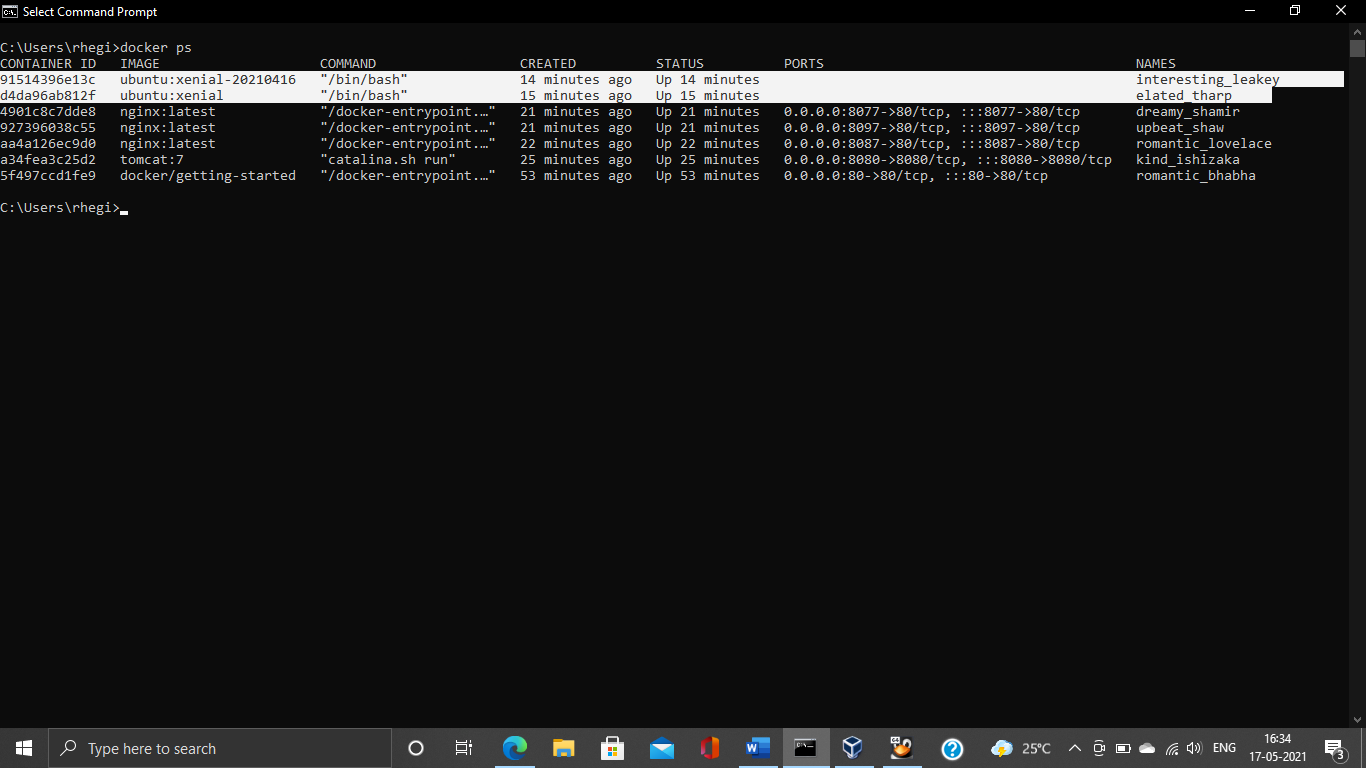
**28. Install software of your choice in ubuntu container using "apt-get install"**



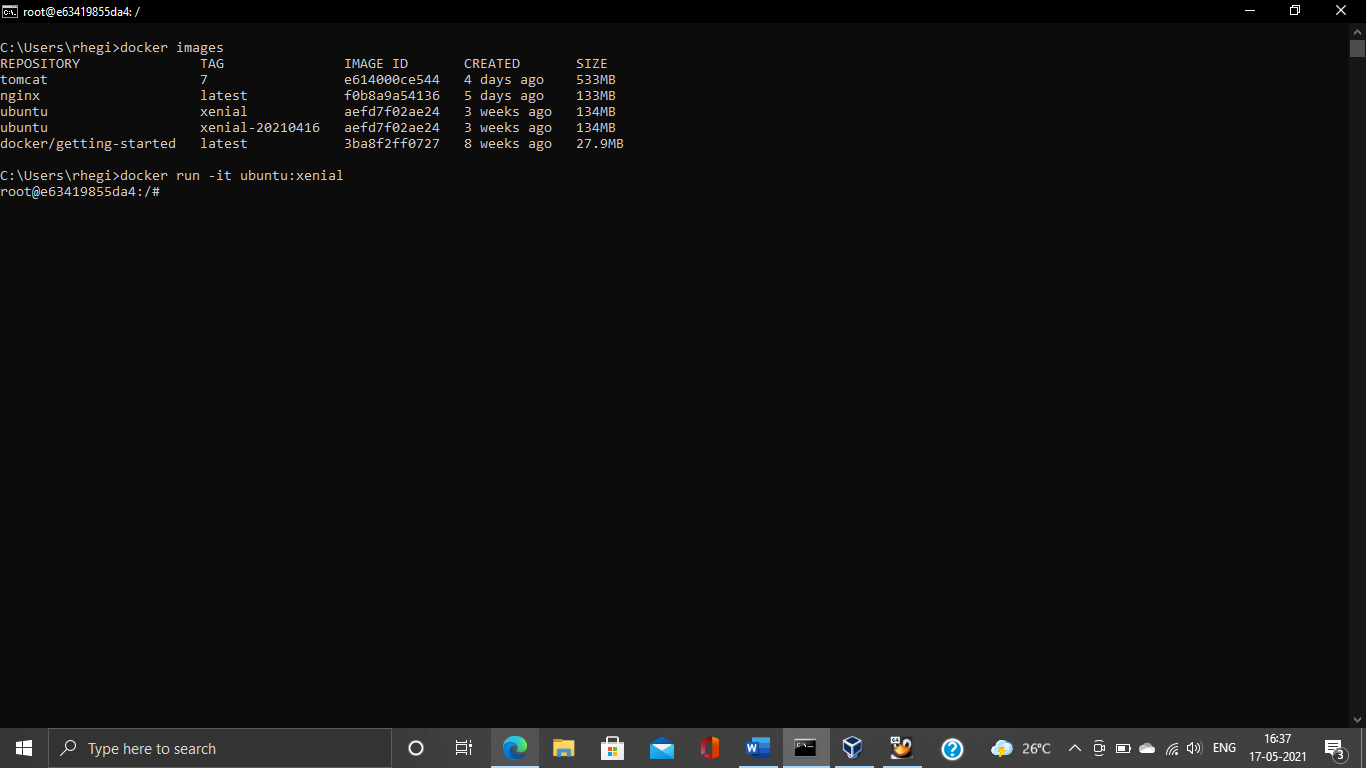
**29. Now exit the ubuntu shell, are you back to your host machine, if not, come back to the host machine.**



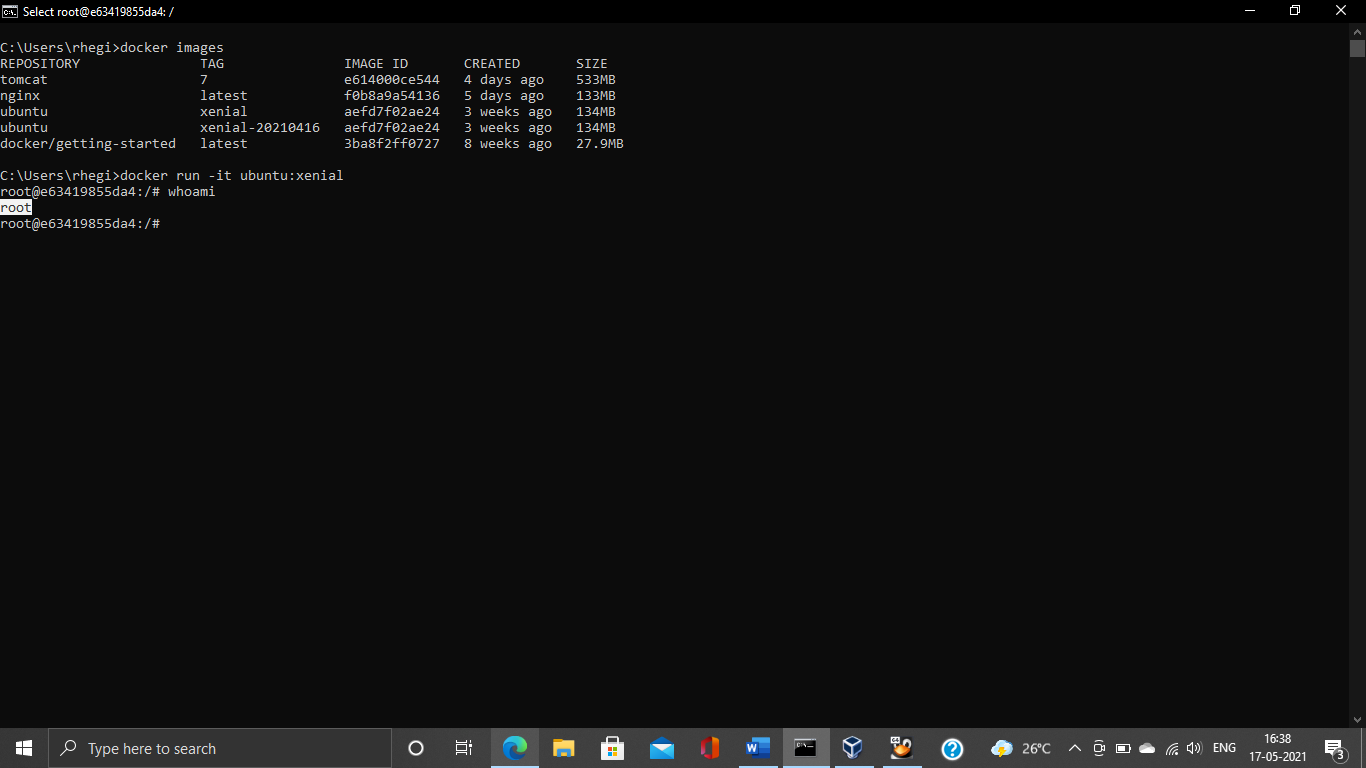
**30. Check if the ubuntu container is running.**



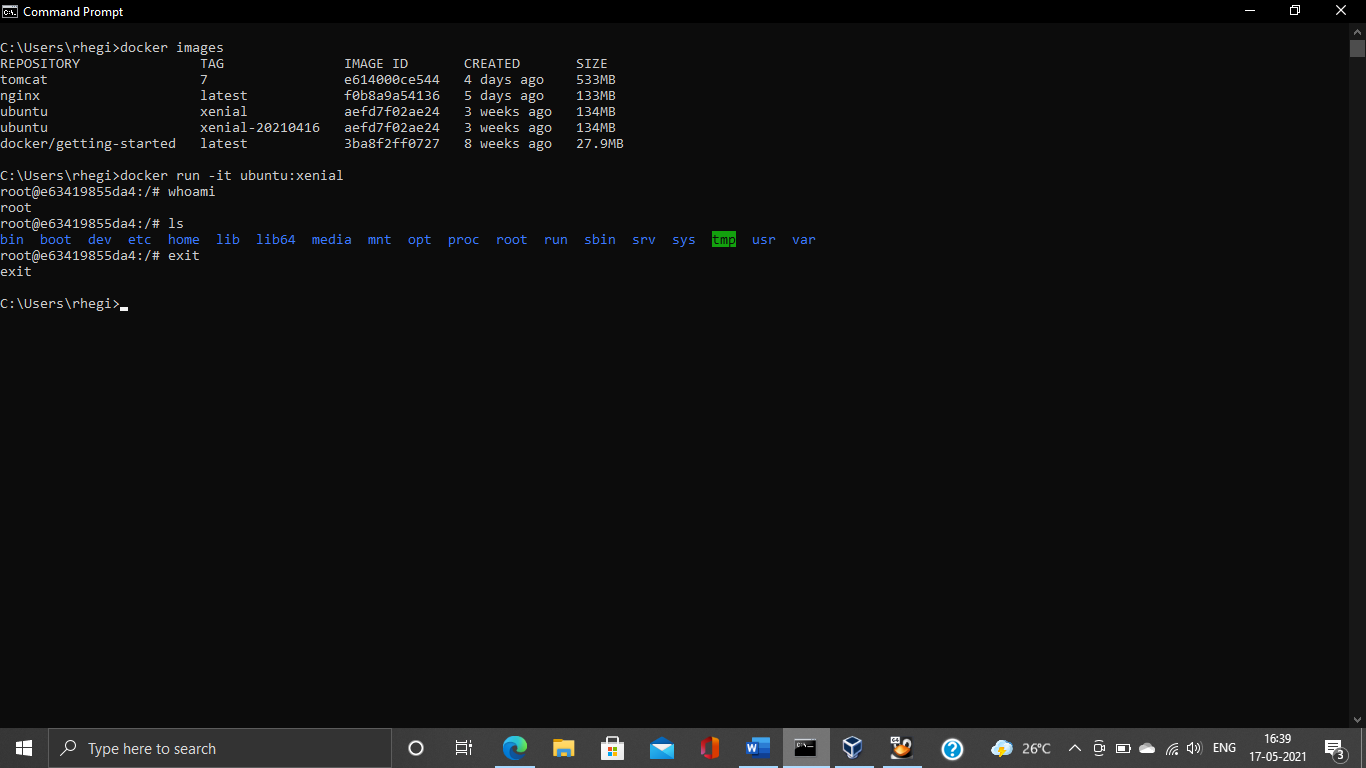
**31. Create a new ubuntu container out of the same image as that previous container in attached mode.**



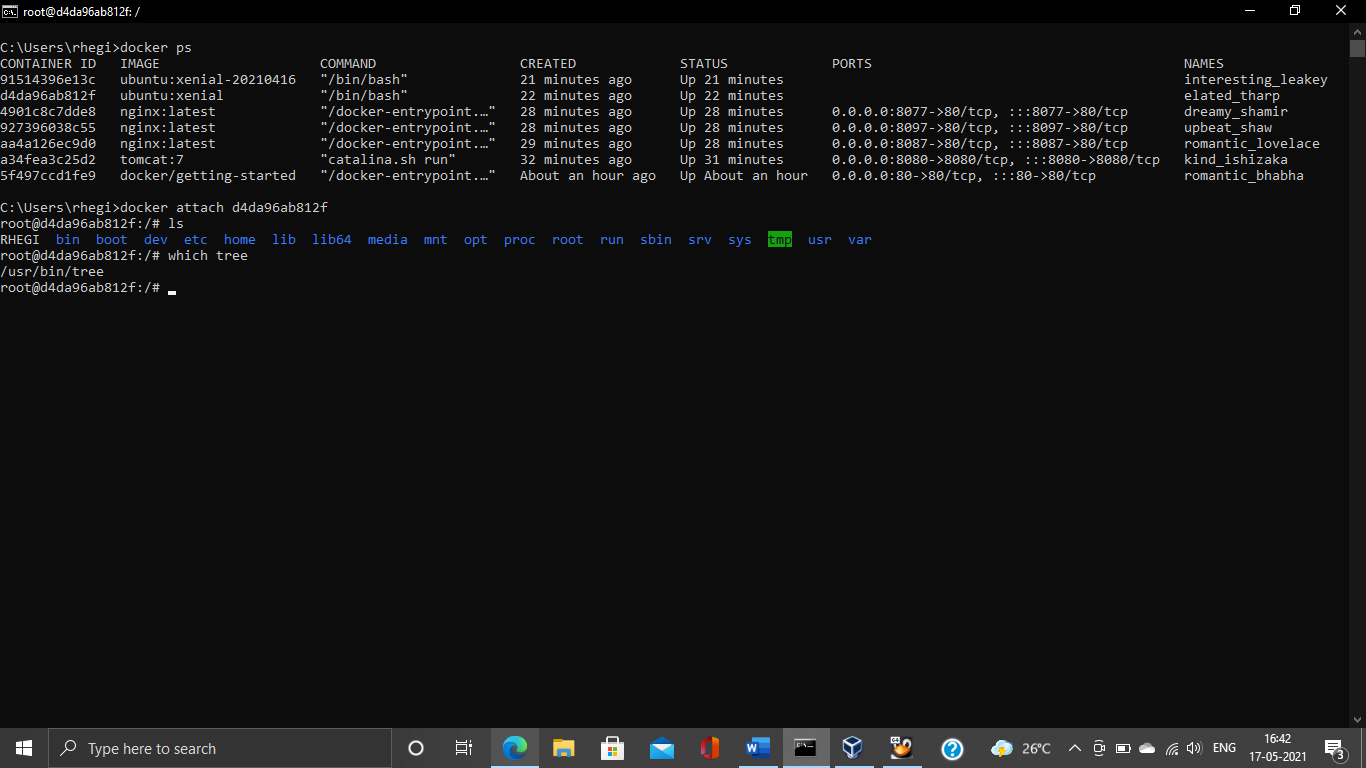
**32. Login as a root user**



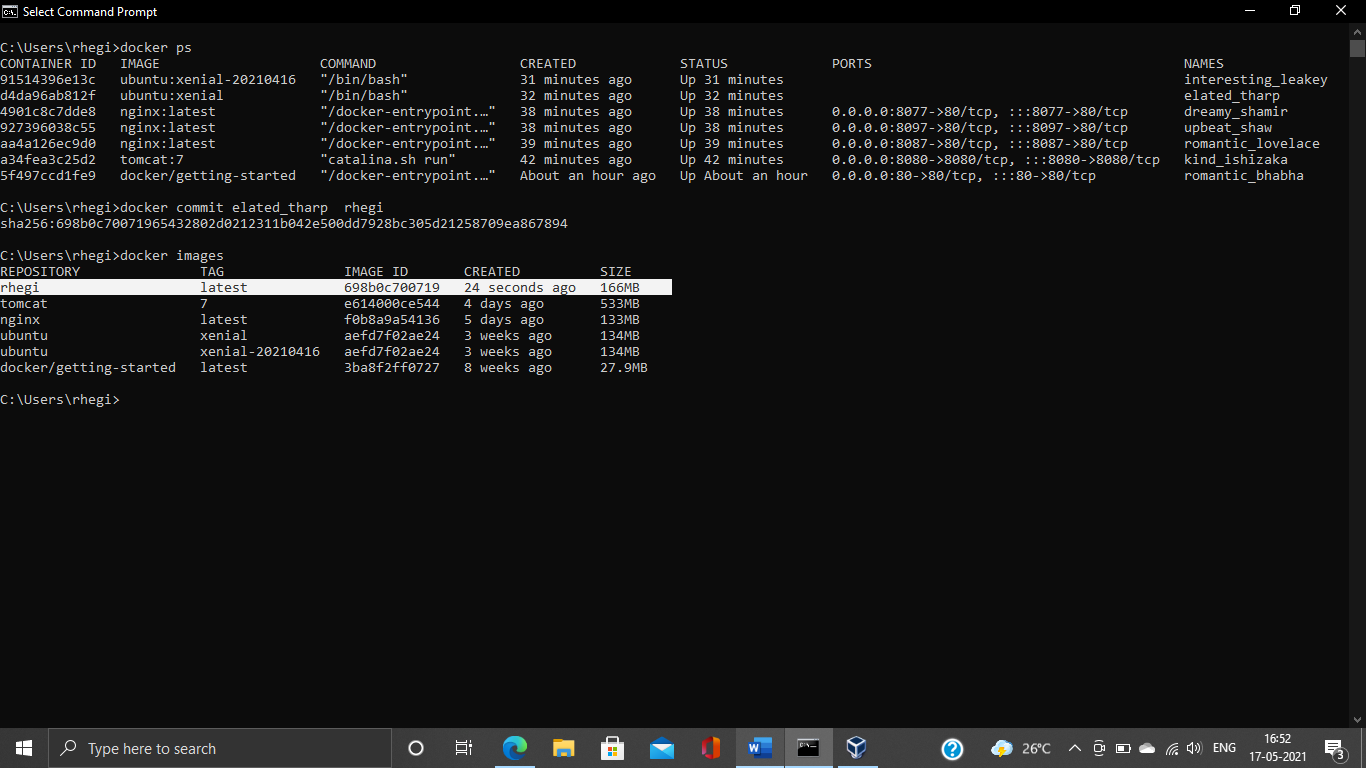
**33. Check if you can see the file created in previous container, you will not see the file as well as software that you installed in the previous container. Now kill this Container.**



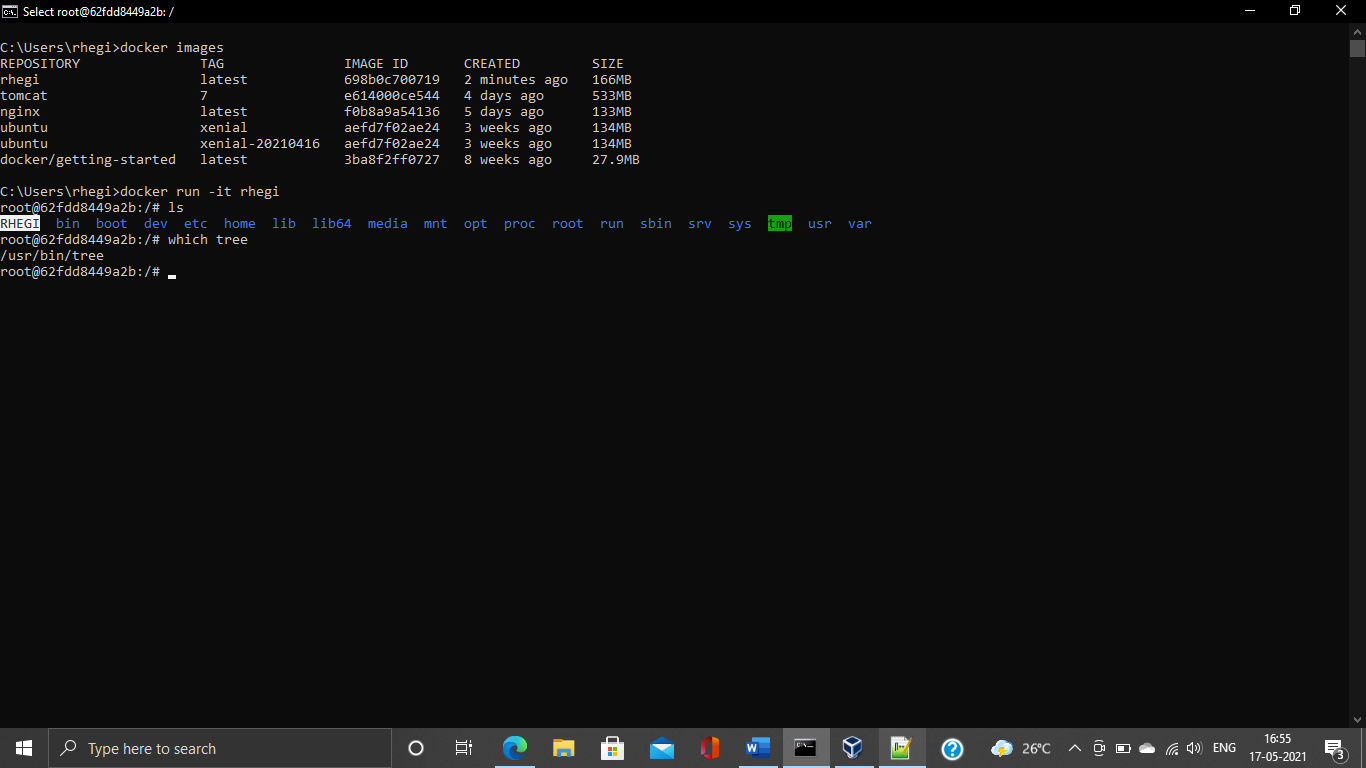
**34. Do you have the previous ubuntu container where you created the file and installed the software? If no repeat step 25 to 29.**



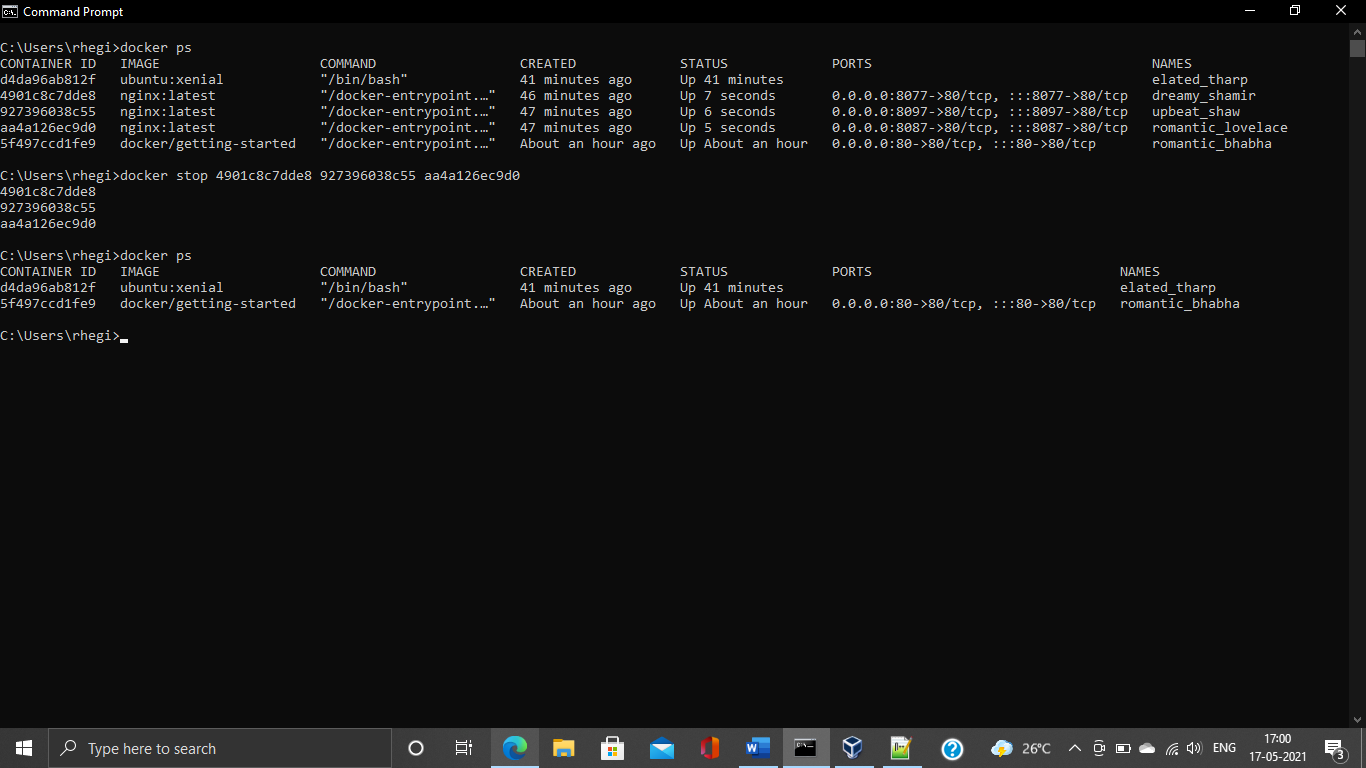
**35. Create an Image out of the existing container.**



**36. Now Create a Container out of this image and login into it to see if you can see the file and software installed by you in the previous container.**



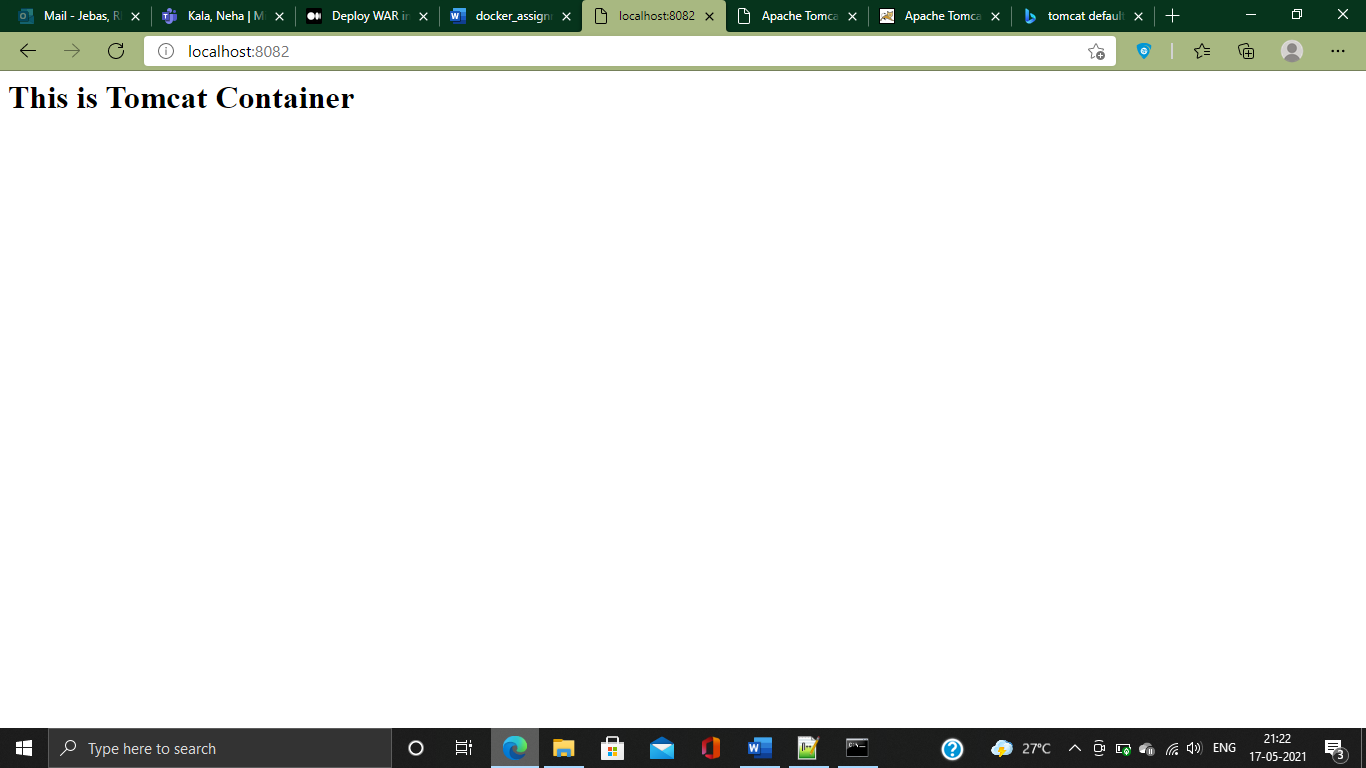
**37. Do you have running tomcat container? If yes, Stop it and kill all tomcat container.**



**38. Create an index.html file with following code in it:-**

**<h1>This is Tomcat Container</h1>**

**Now, Start a tomcat container in such a way that on hitting its URL for home page it should show the above html page.**



**39. type below command:-**

**docker images --help**

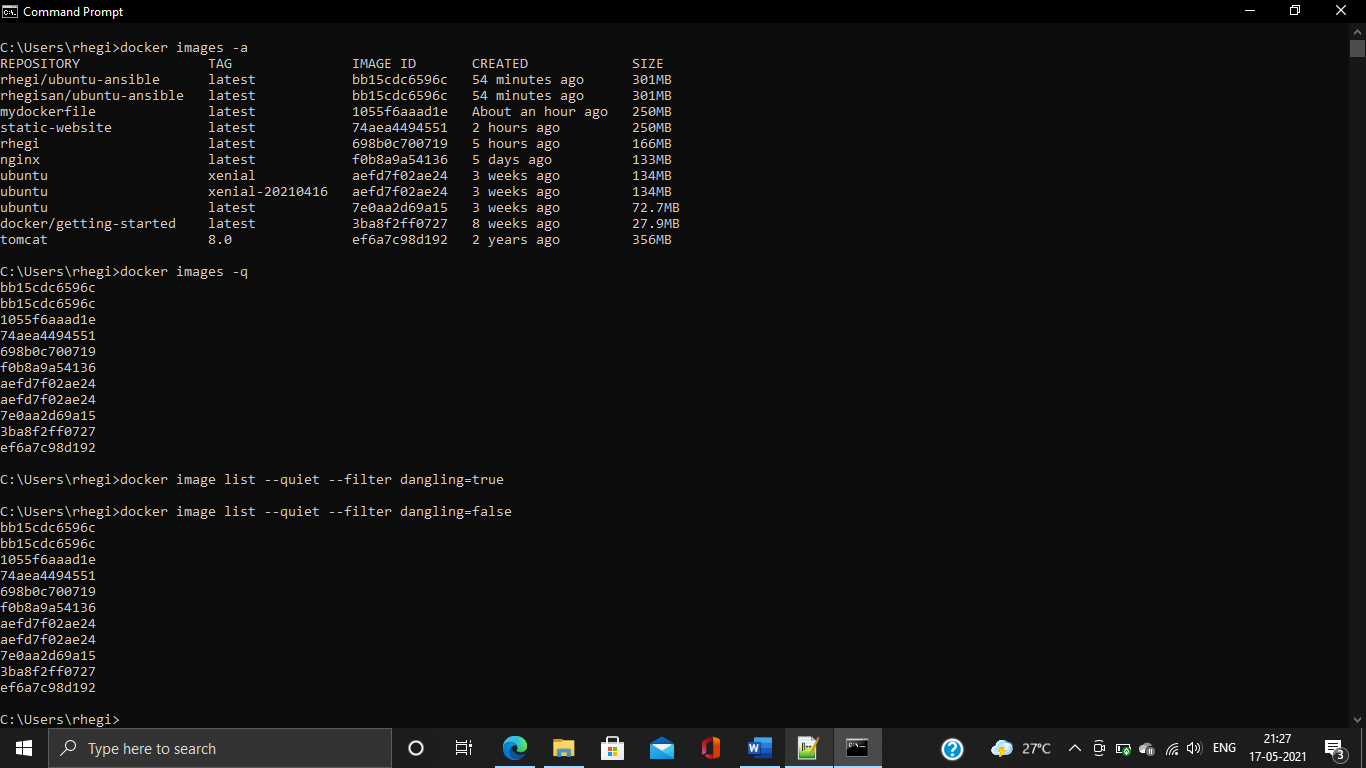
**Now, try to run command that proves the concept of following three options:-**

**1. -a**

**2. -f**

**3. -q**

**write atleast 1 command using each option above and prove their concepts as described in the --help.**



**40. type below command:-**

**docker ps --help**

**Now, try to run command that proves the concept of following six options:-**

**1. -a**

**2. -f**

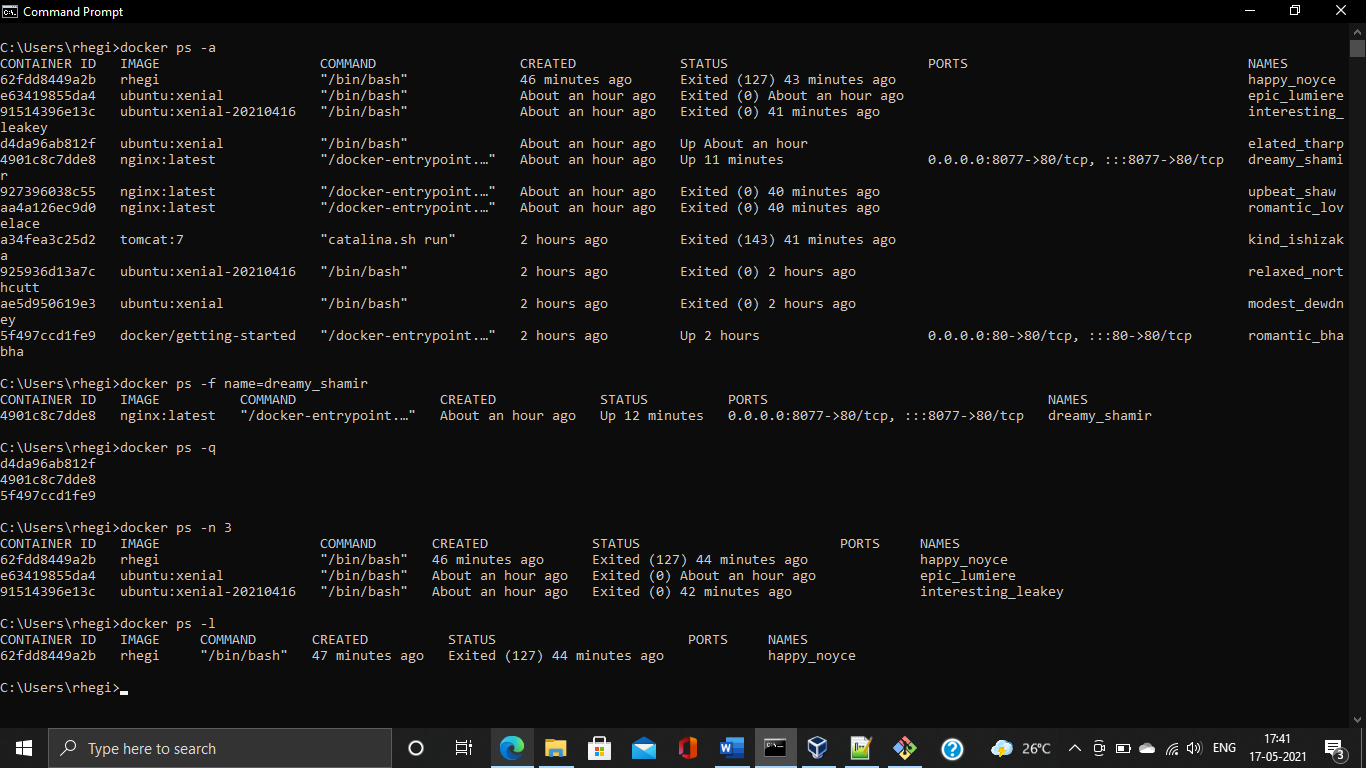
**3. -q**

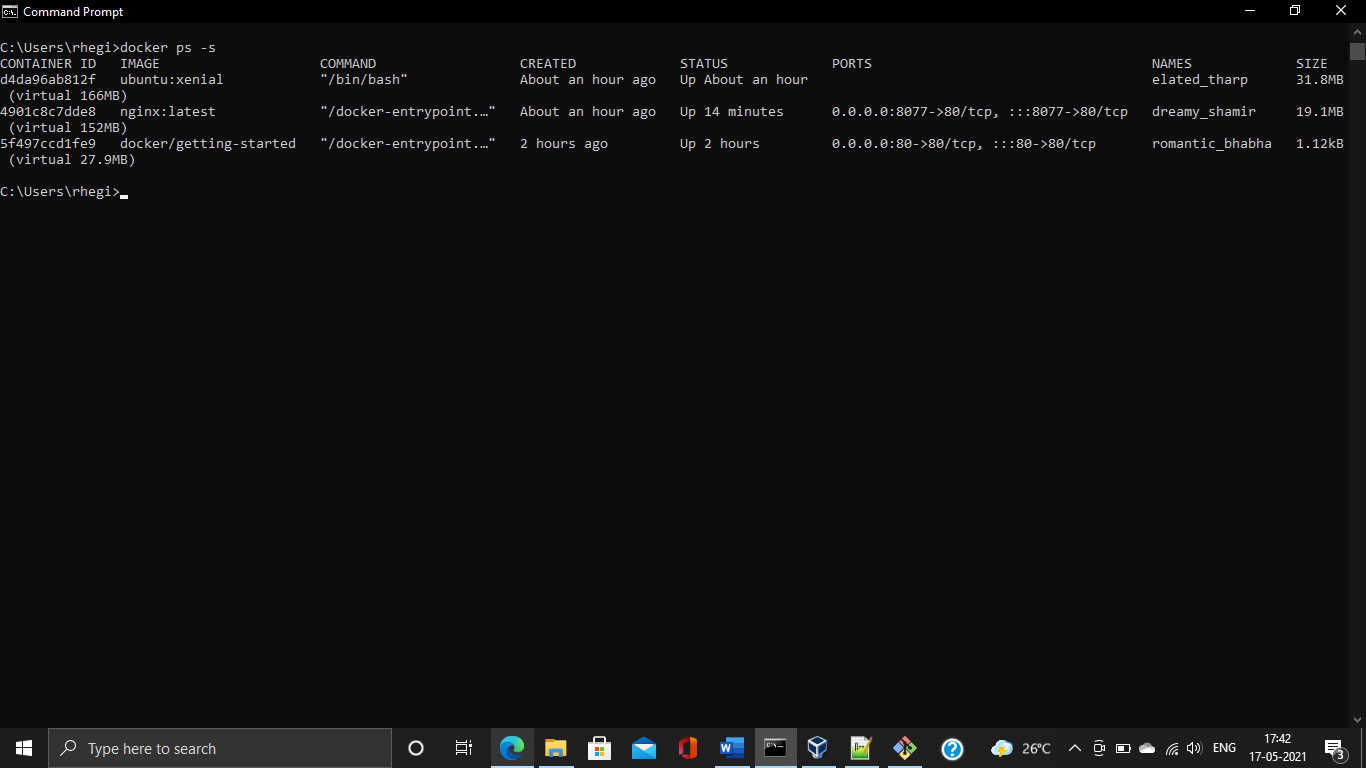
**4. -n**

**5. -l**

**6. -s**

**write atleast 1 command using each option above and prove their concepts as described in the --help.**





**41. Type below command:-**

**docker --help**

**you will various sections of commands apart from options like "Management Commands" and "Commands".**

**Write some texts below describing the use of "Management Commands".**

1. Docker Swarm is used as an orchestration tool it Initializes a swarm, Join a swarm as a node and/or manager etc.
2. Docker config manages the docker configs
3. Docker container, image and context manages the containers, images and the contexts
4. Trust manages trust on Docker images
5. Volume deals with the volumes which is shared between the containers and the host.
6. Many other commands like builds, plugins and service are also in this section and can me very handy while managing the activities.

**Use each command mentioned below and prove its concepts as described in the --help desription. write what you have understood from the output of the command after its successful execution.**

**1. cp –** It Copy files/folders between a container and the local filesystem

**2. create –** It Creates a new container but does not run it, we need to start it separately.

**3. export -** Export a container's filesystem as a tar archive, docker export <container-name> > example.tar

**4. history -** Shows the history of an image

**5. info –** It displays system-wide information

**6. login –** It is used to log in to a Docker registry

**7. logout -** It is used to log out from a Docker registry

**8. rename -** Renames a container

**9. save - S**ave one or more images to a tar archive,docker save <container-name> > example.tar

**10. stats –** It displays a live stream of container(s) resource usage statistics

**11. top -** Display the running processes of a container

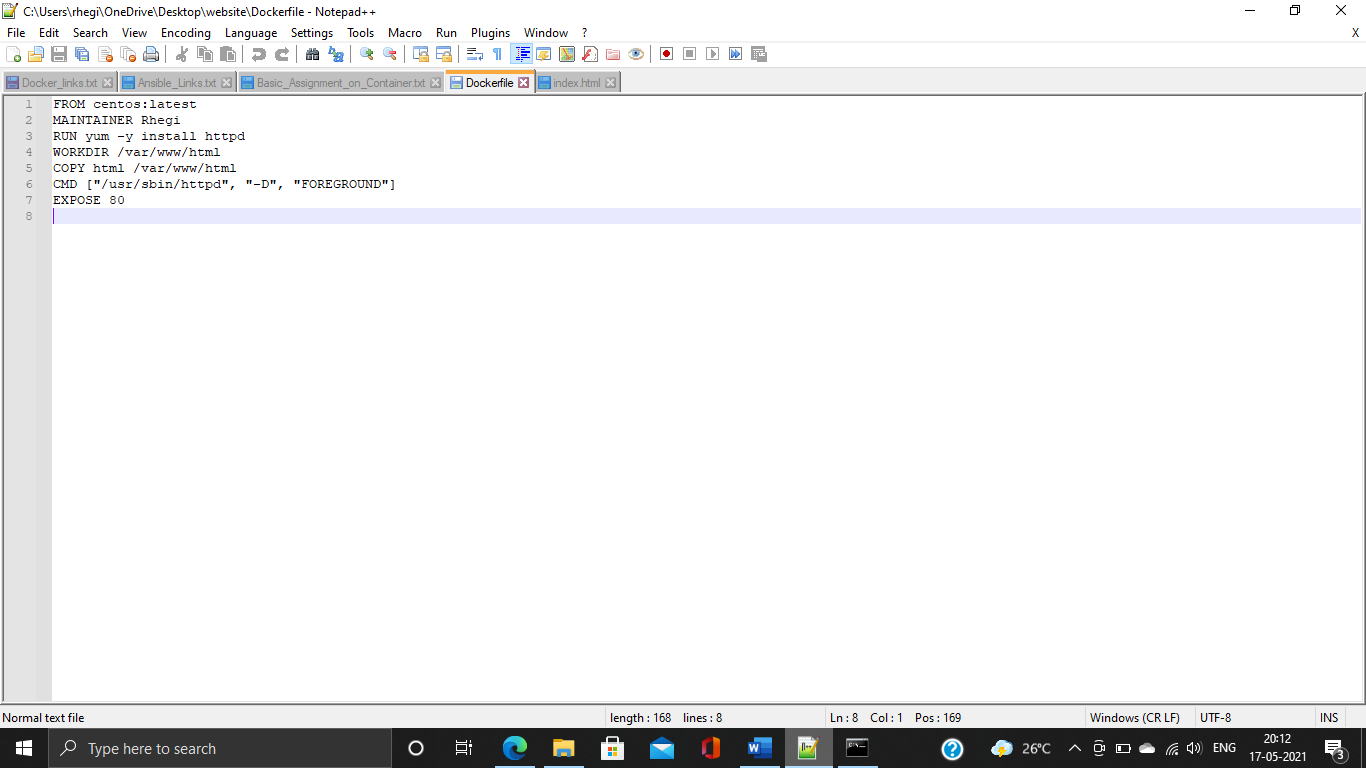
**42. Kill all running container in one liner command.**

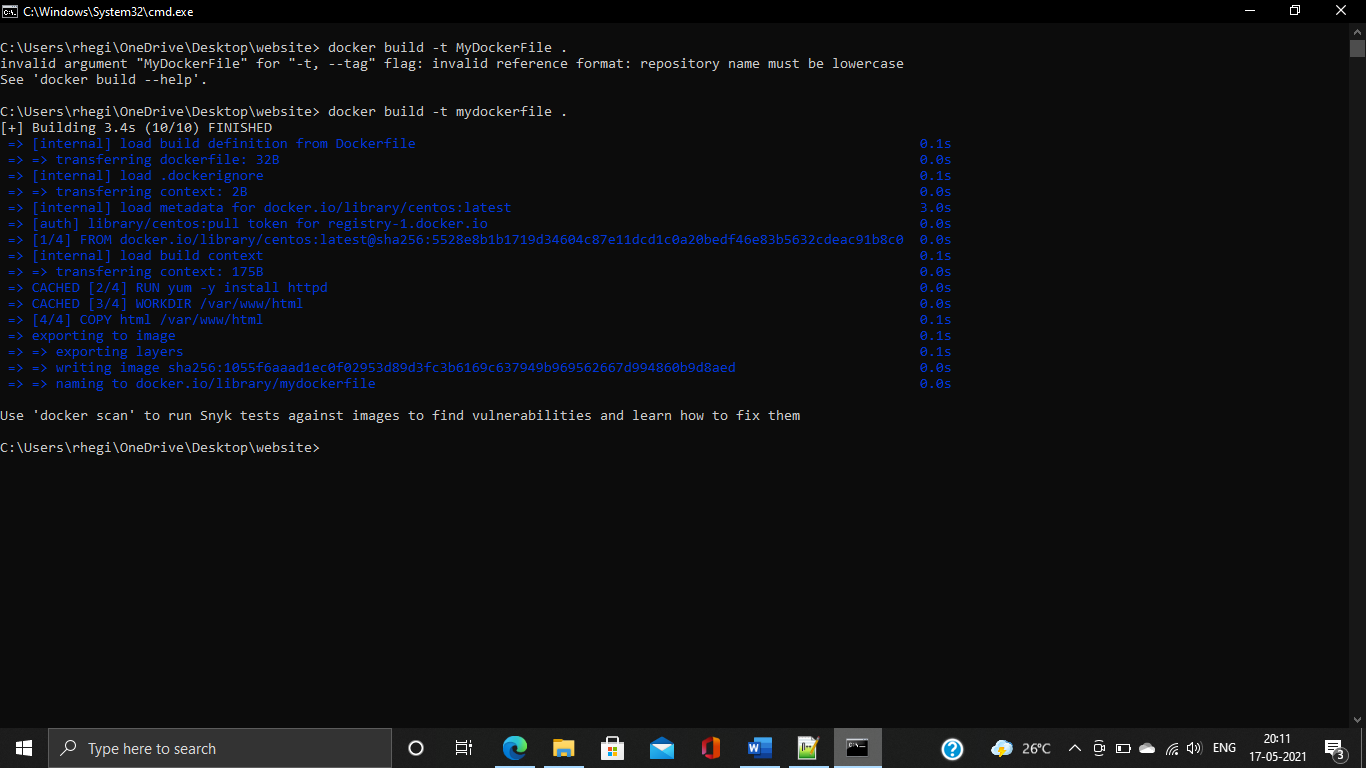
docker rm –f $(docker ps –qa) --> Forcefully removes all the containers

**43. Delete all images in one liner command.**

docker images -a -q | xargs docker rmi –f --> Forcefully removes all the images

**44. Create a simple Dockerfile, build it and run it.**





**45. Create one or more Dockerfile that demostrate the following commands in Dockerfile (Write a PoC for each in one or more Dockerfile)**

**1. USER**

**2. RUN**

**3. ENV**

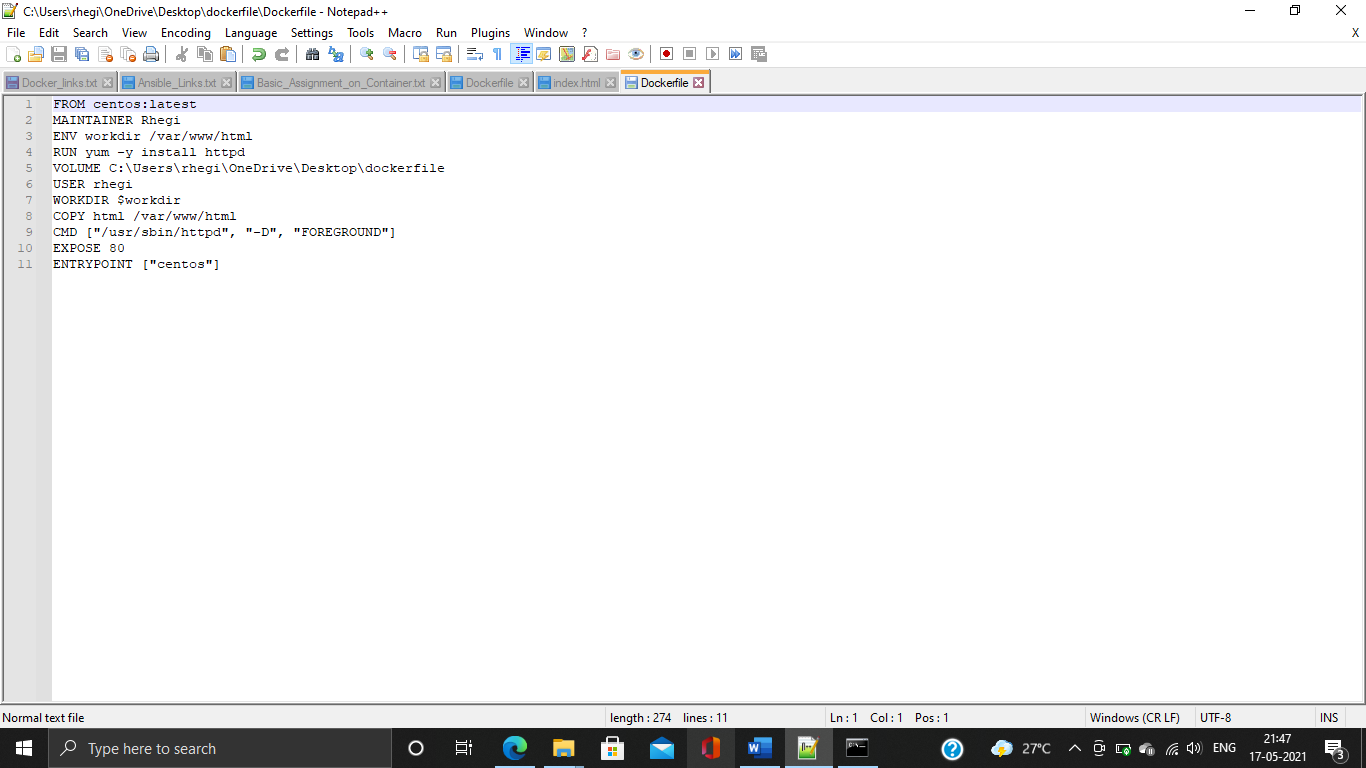
**4. CMD**

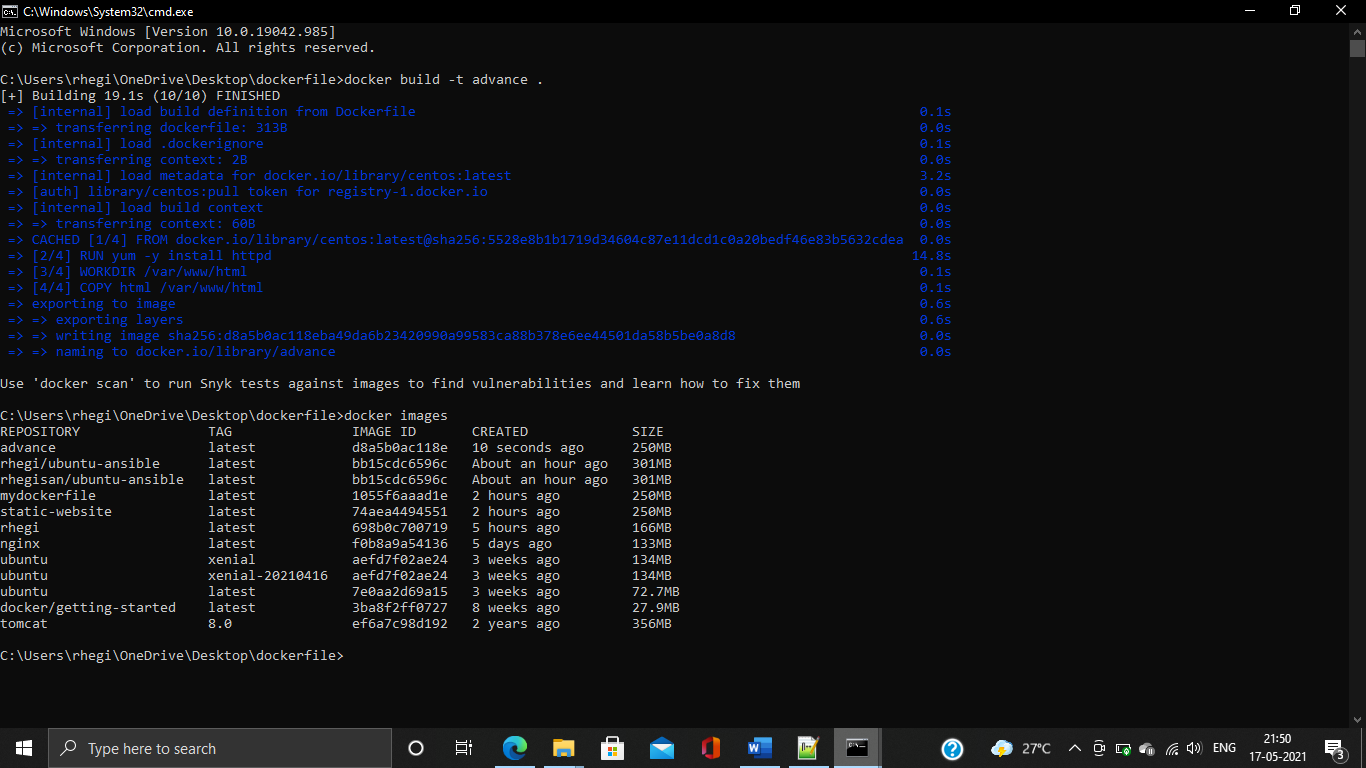
**5. RUN**

**6. ENTRYPOINT**

**7. EXPOSE**

**8. VOLUME**





**46. Docker hub: -**

**Find an application you care about on docker hub.**

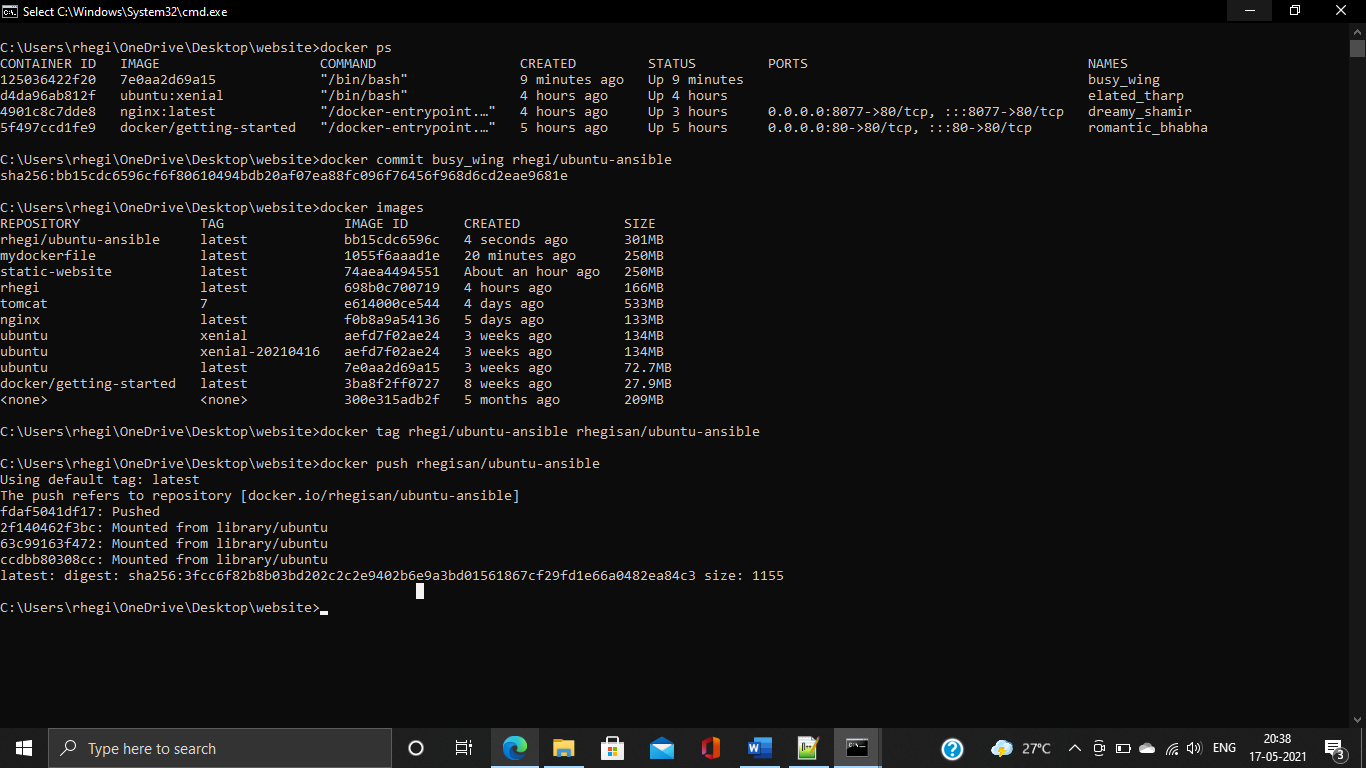
**Launch the container.**

**Install another application in it.**

**Save (commit) the image.**

**Upload that to docker hub in your account and share it with a colleague, ask them to use your image and run the container out of it in their machine/vm.**

1. Pulled an Ubuntu latest image from Docker Hub
2. Launched the container with Ubuntu image
3. Installed ansible inside the container
4. Converted the container (Ubuntu + Ansible installed) to docker image
5. Uploaded the image to my docker hub account.



Link of my Docker hub account :-[rhegisan/ubuntu-ansible (docker.com)](https://hub.docker.com/r/rhegisan/ubuntu-ansible)