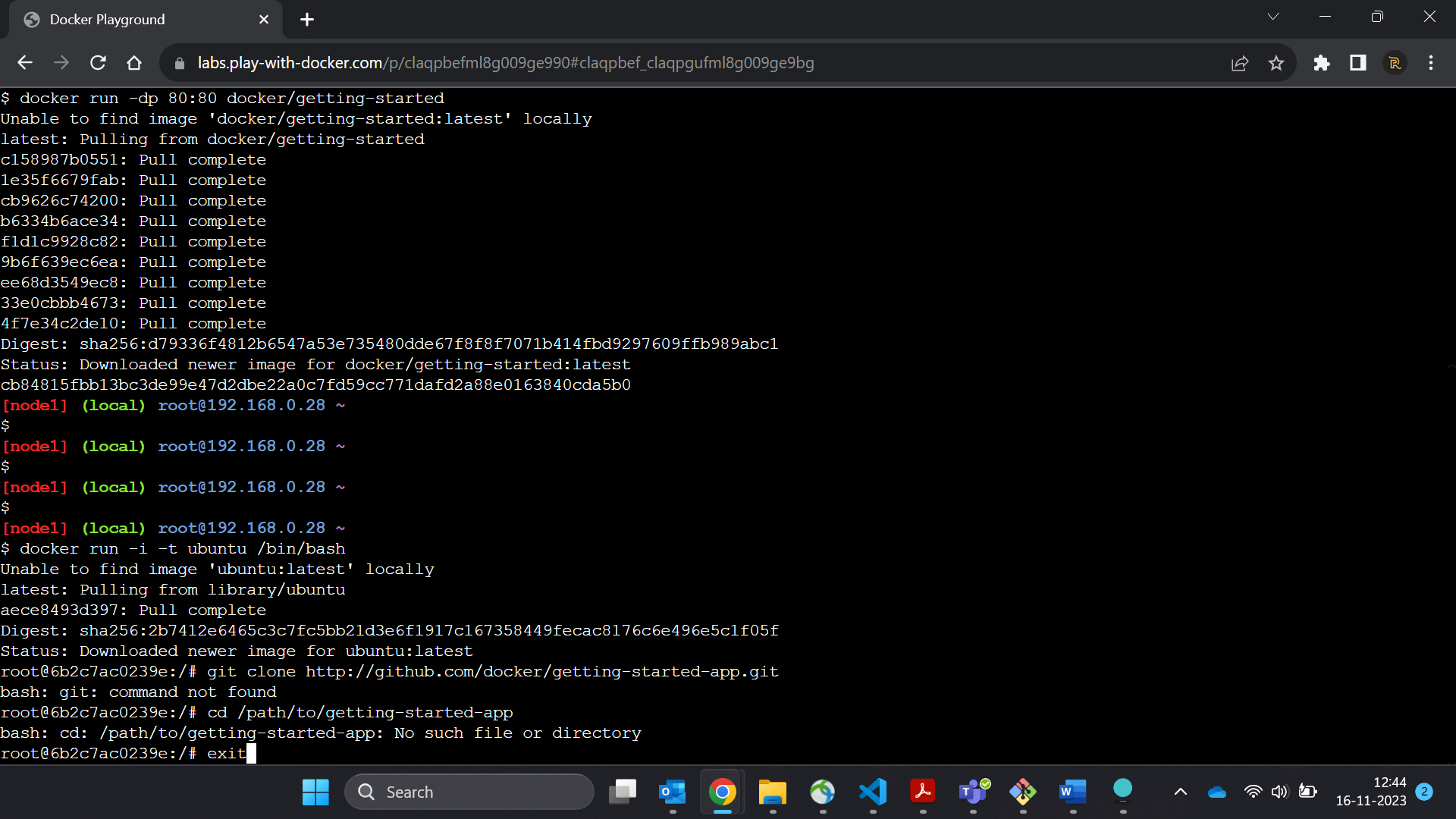
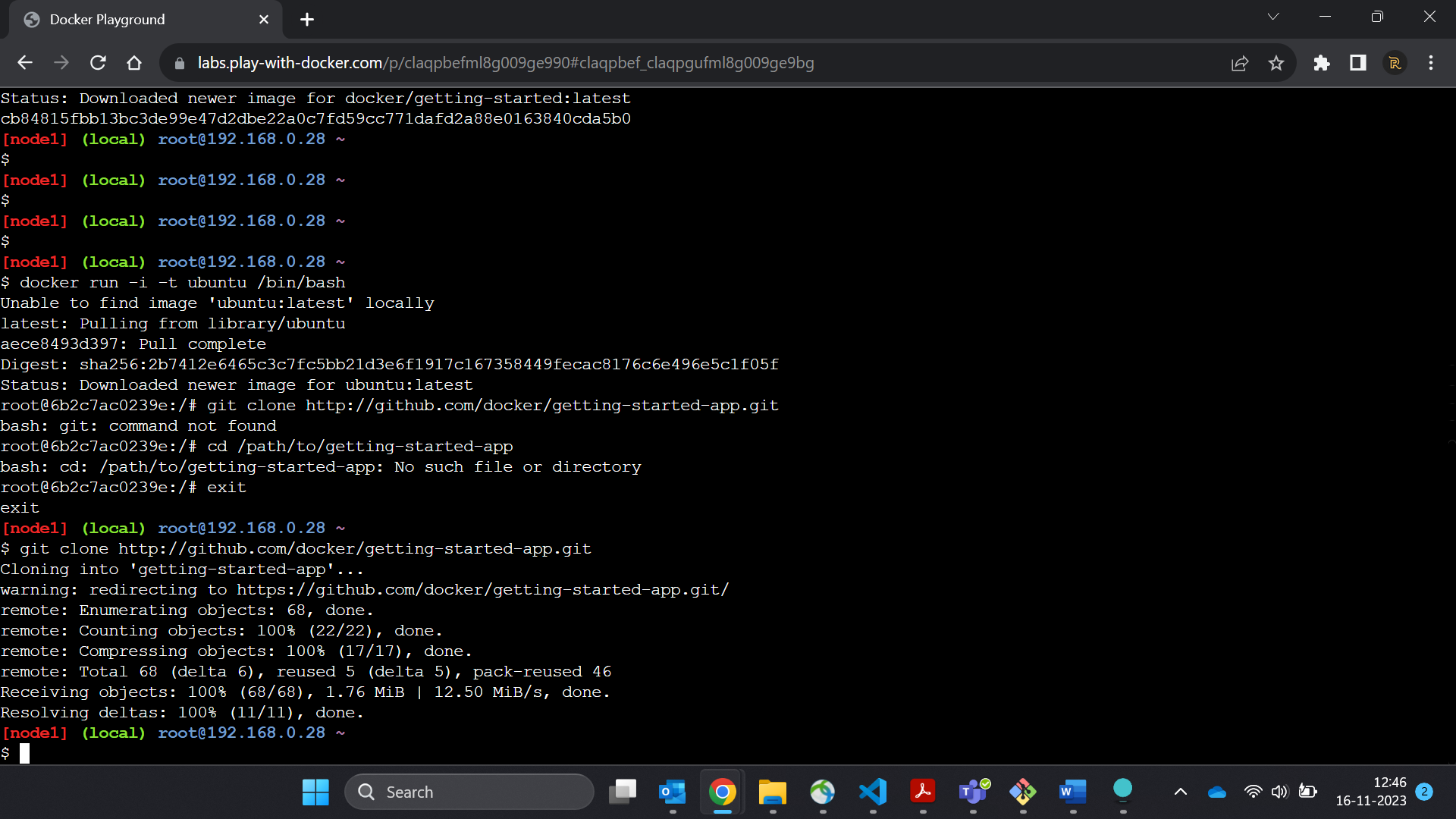
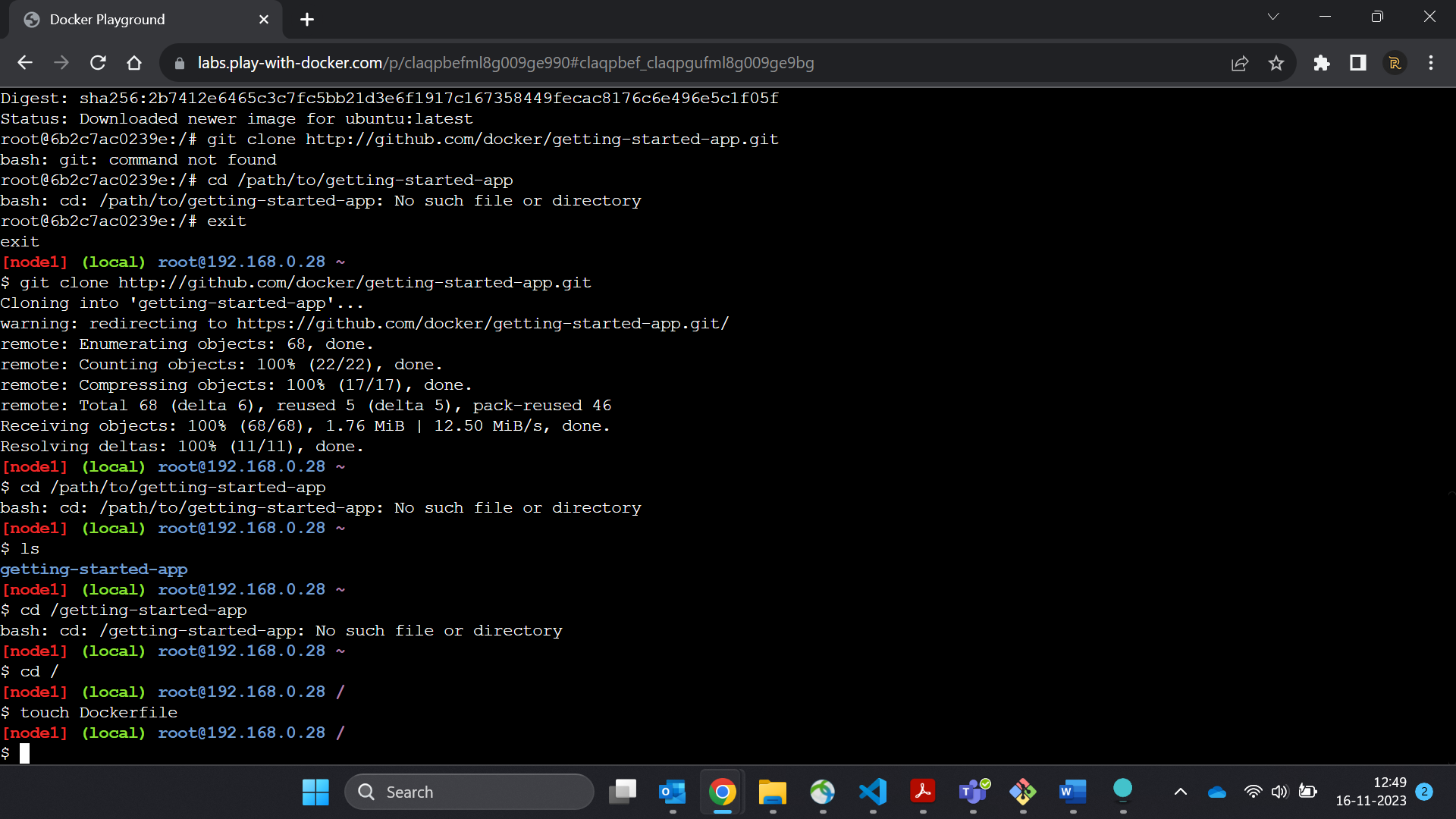
1……pulled and create the container from ubuntu image

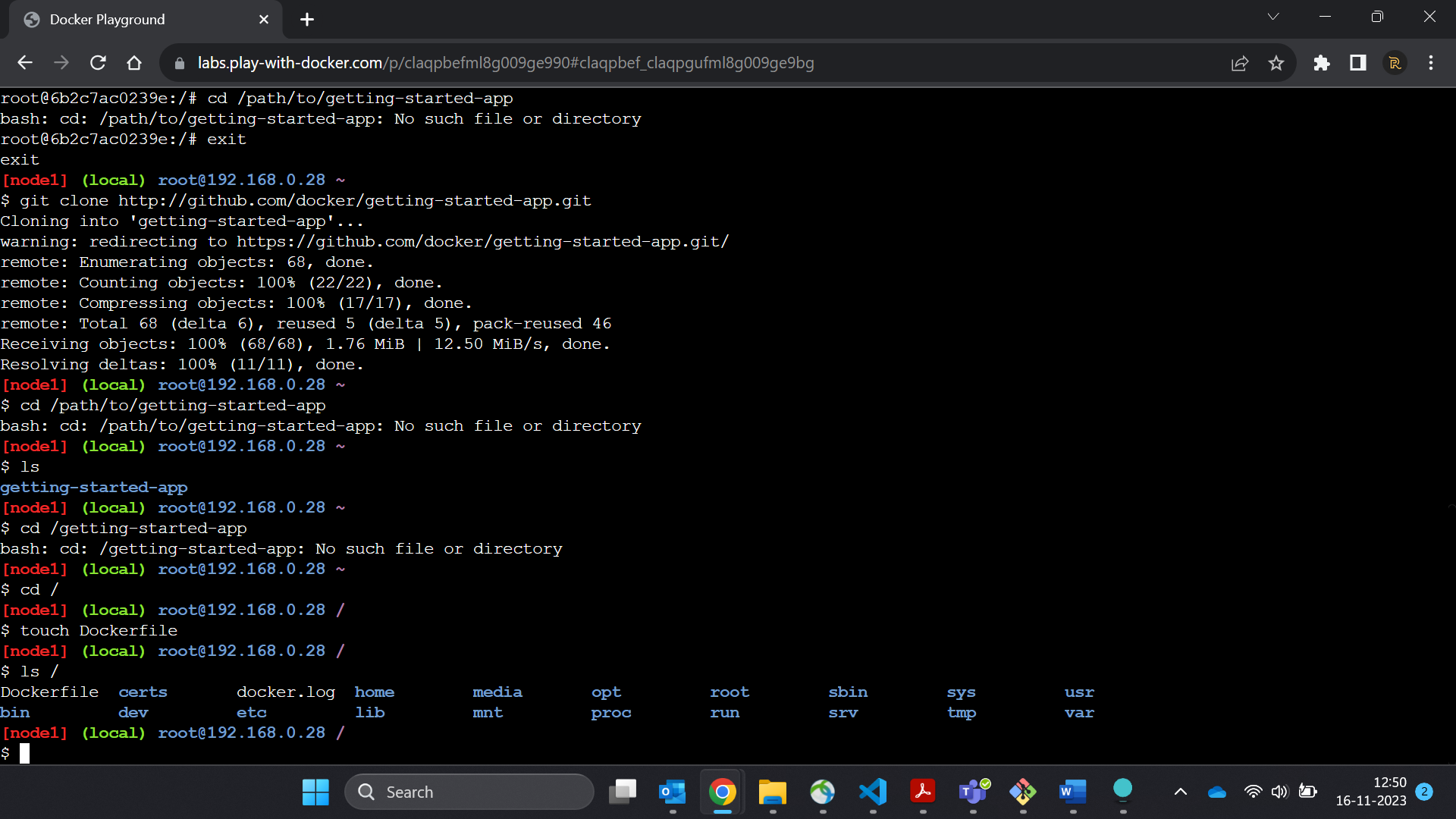


2…Before you can run the application, you need to get the application source code onto your machine.



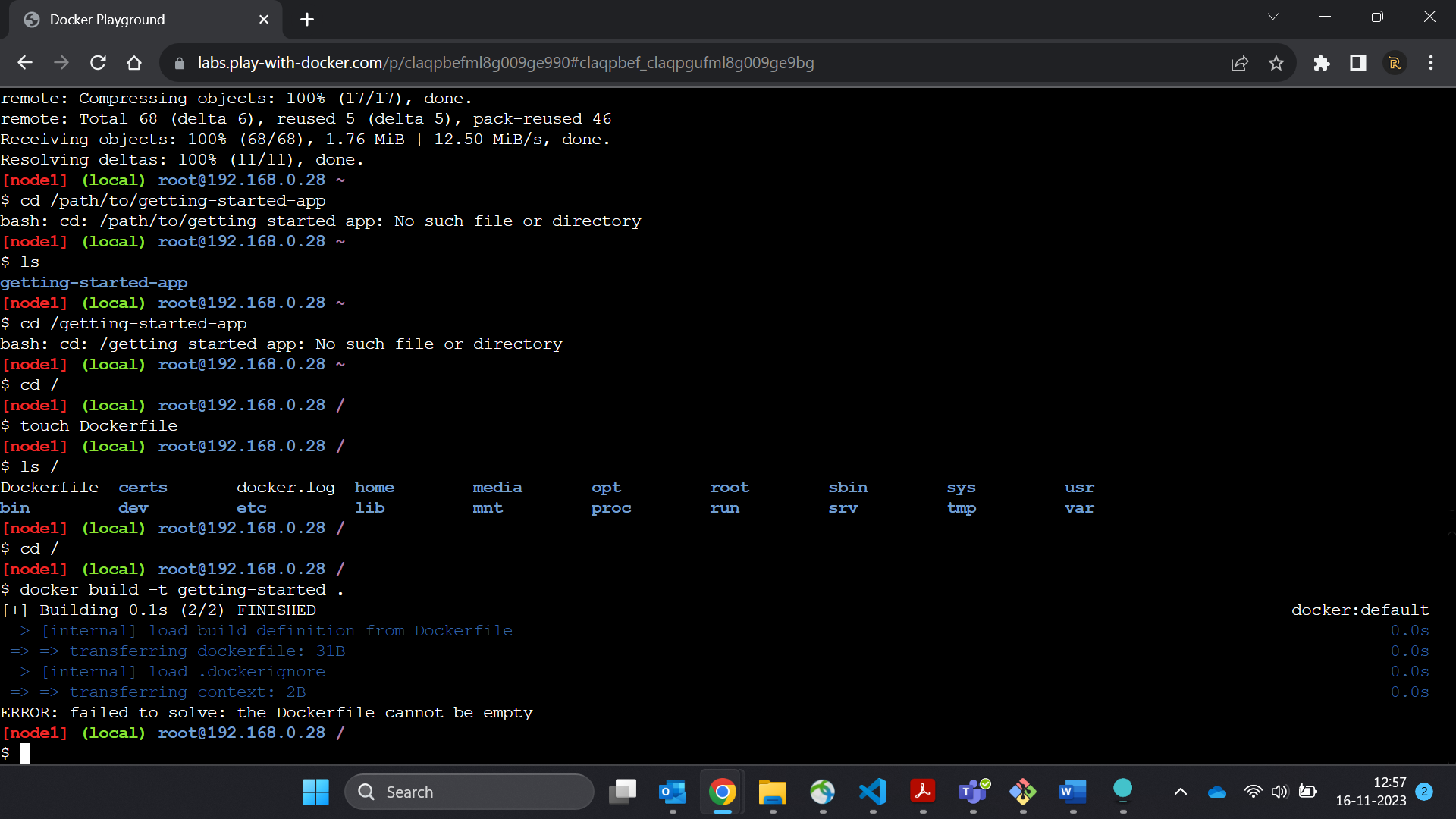
3………Create an empty file named Dockerfile.





4…..

Build the image.

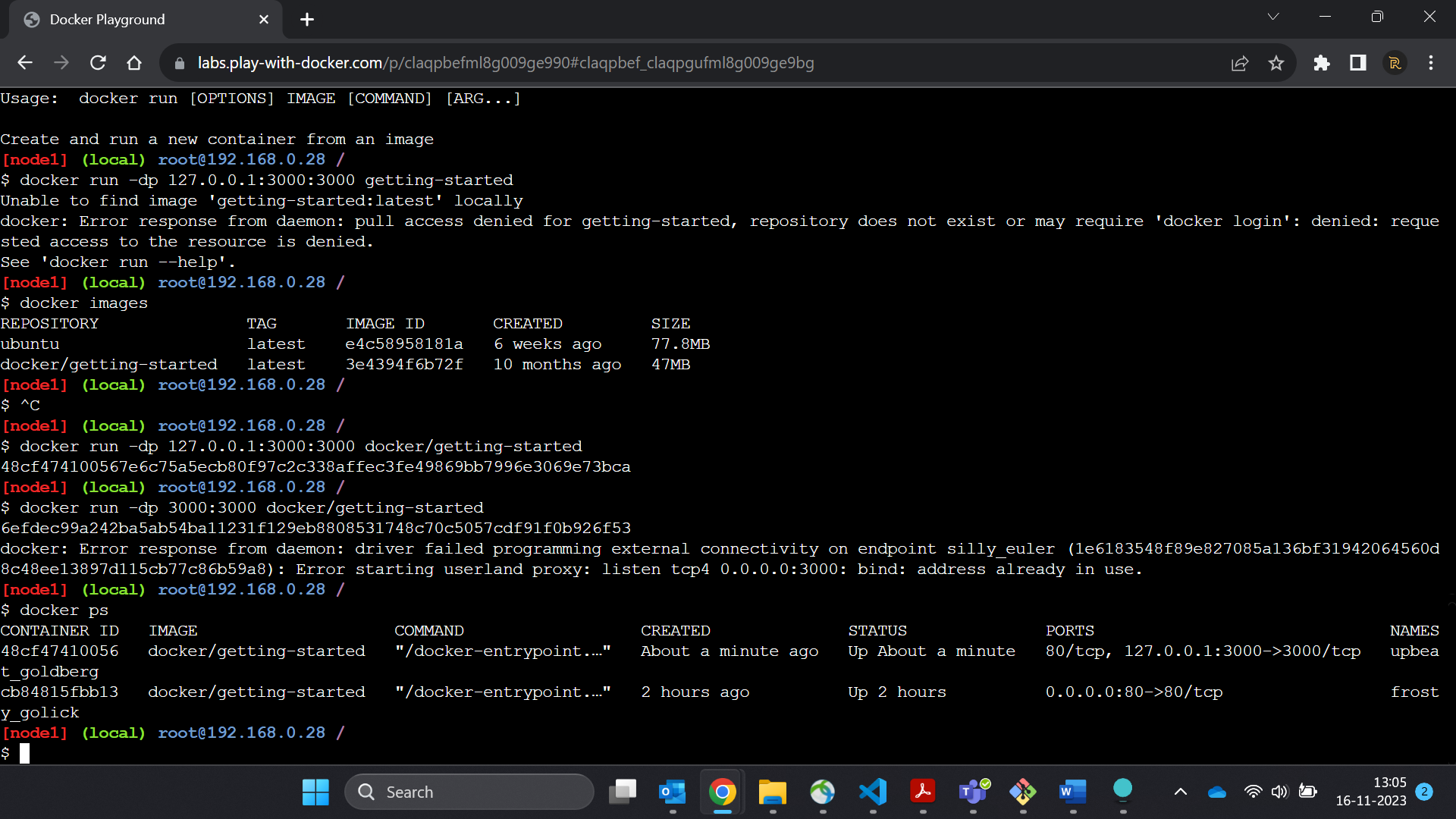


It will build the image of your project files from the current directory…

4. Run your container using the docker run command and specify the name of the image you just created:

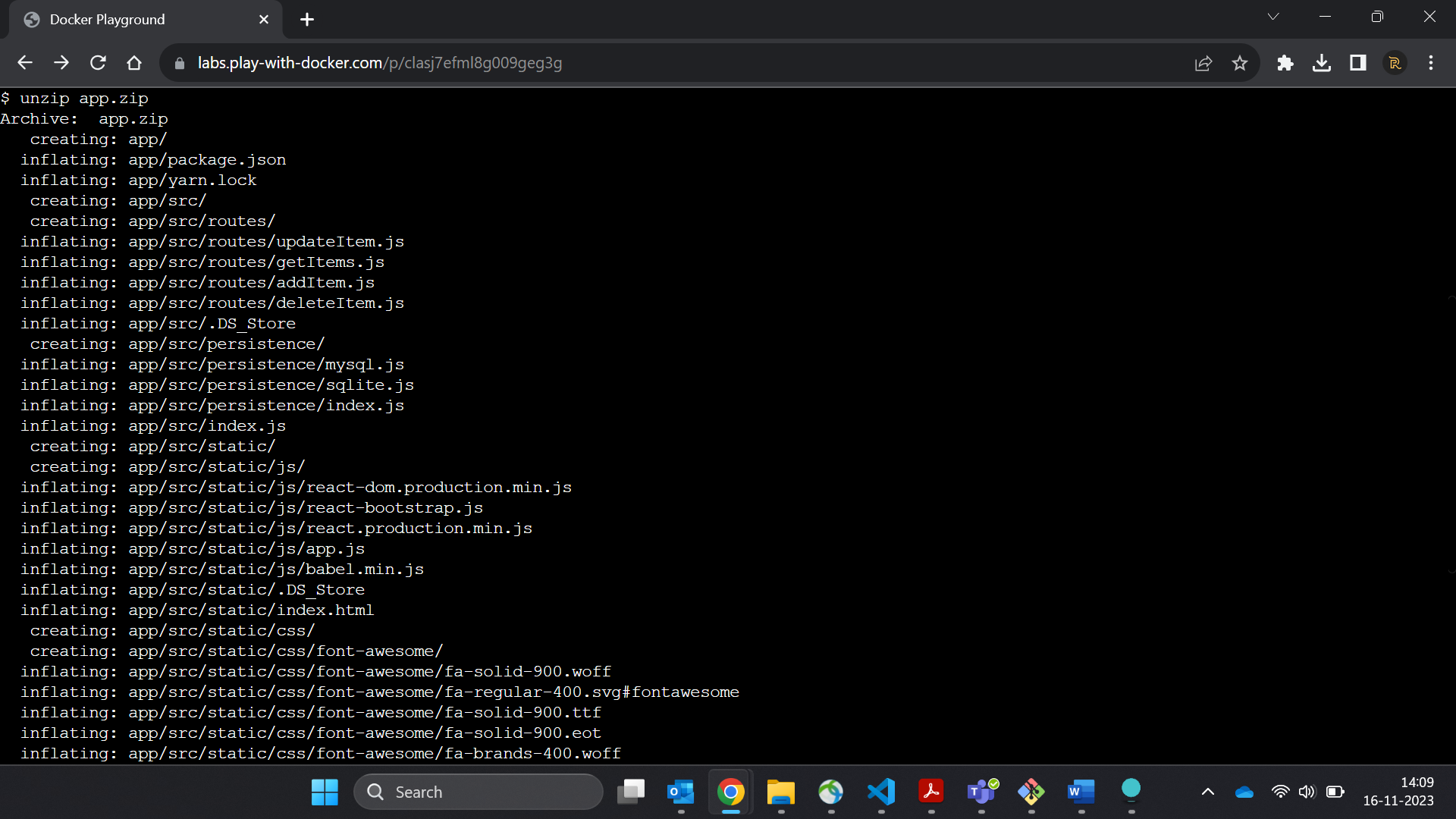


5..List all the container



6..

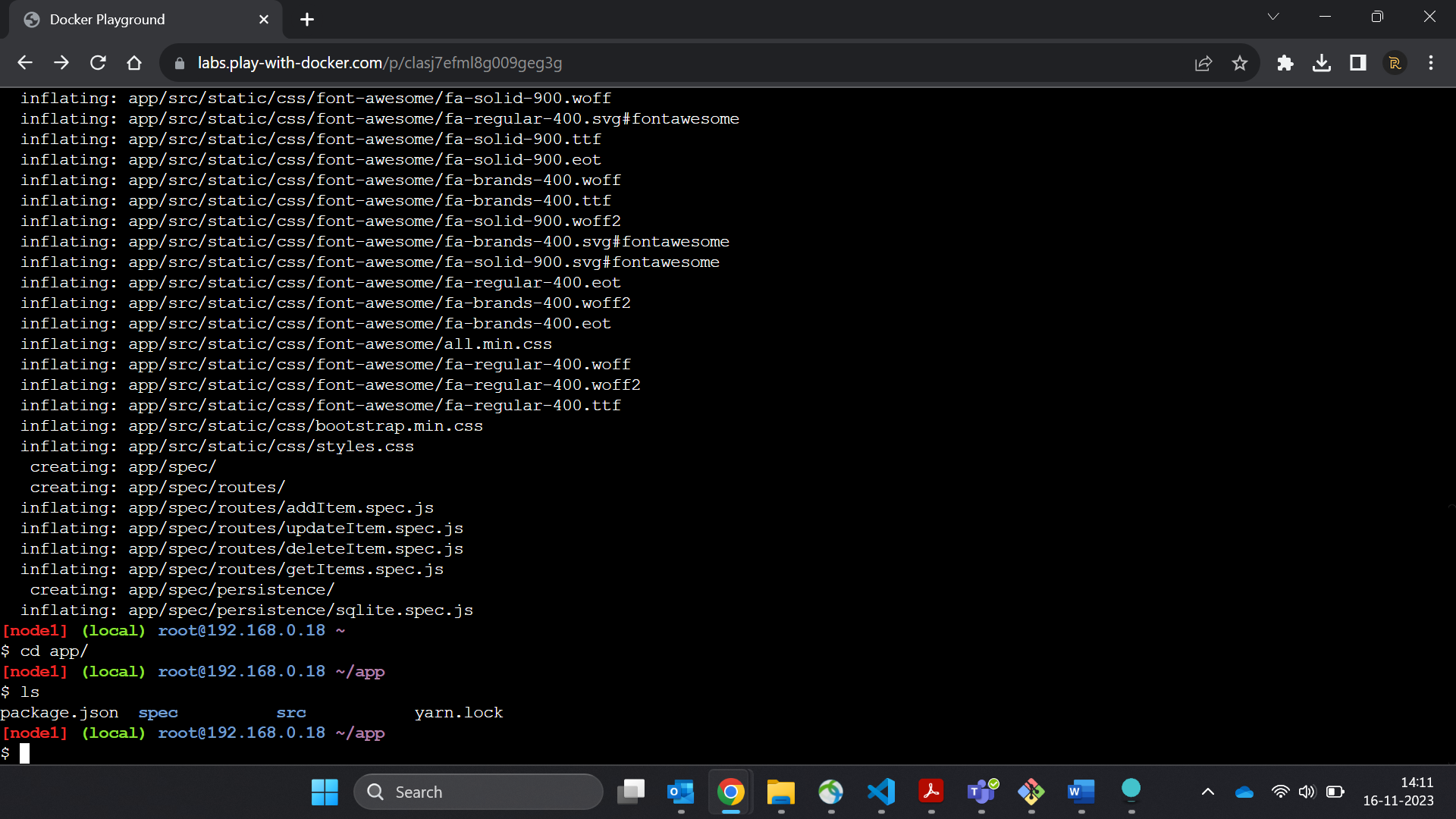
# **Our Application**

1…In the PWD terminal, extract the zip file.

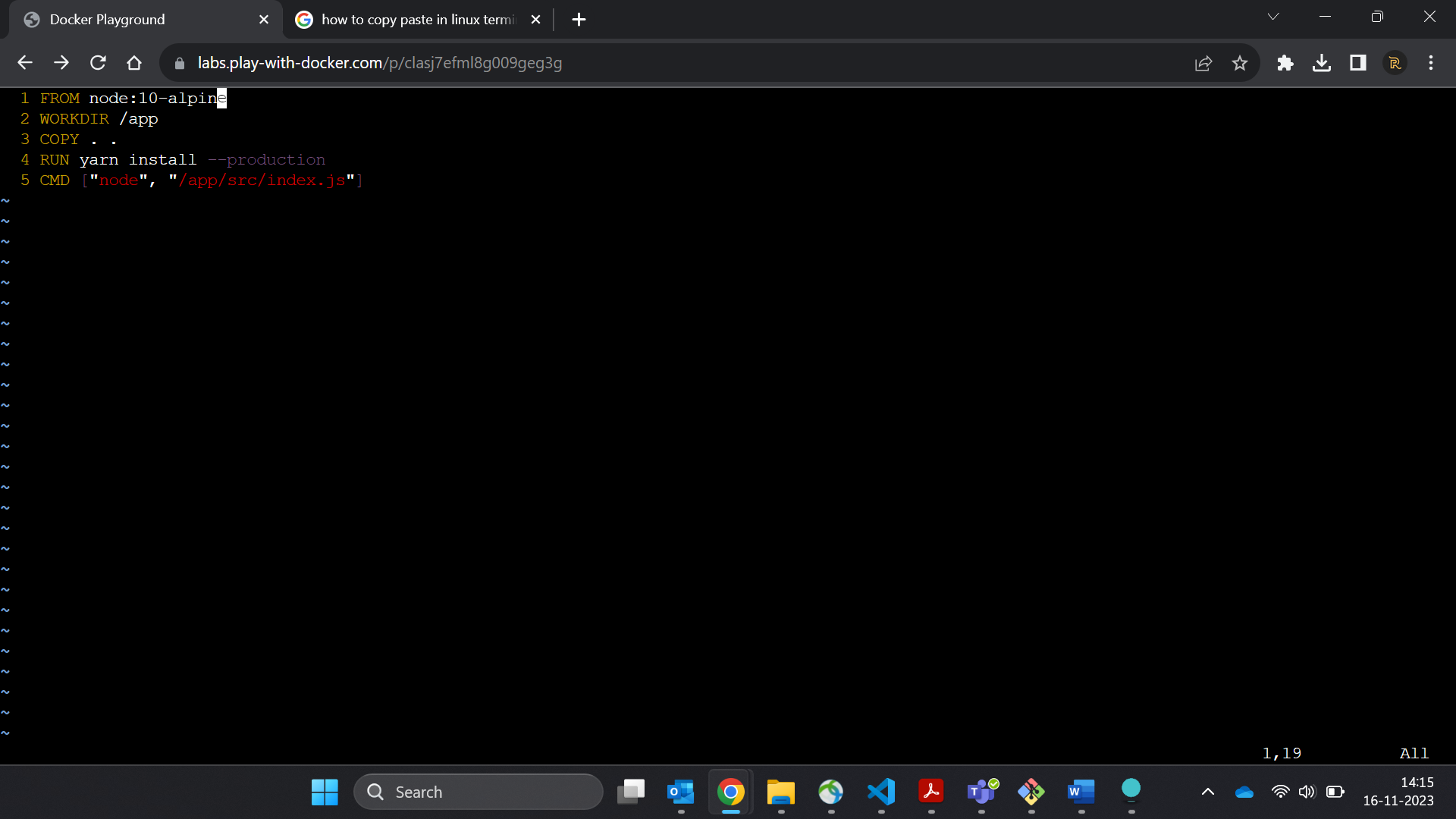
2. Change your current working directory into the new 'app' folder.

3. In this directory, you should see a simple Node-based application.

ls



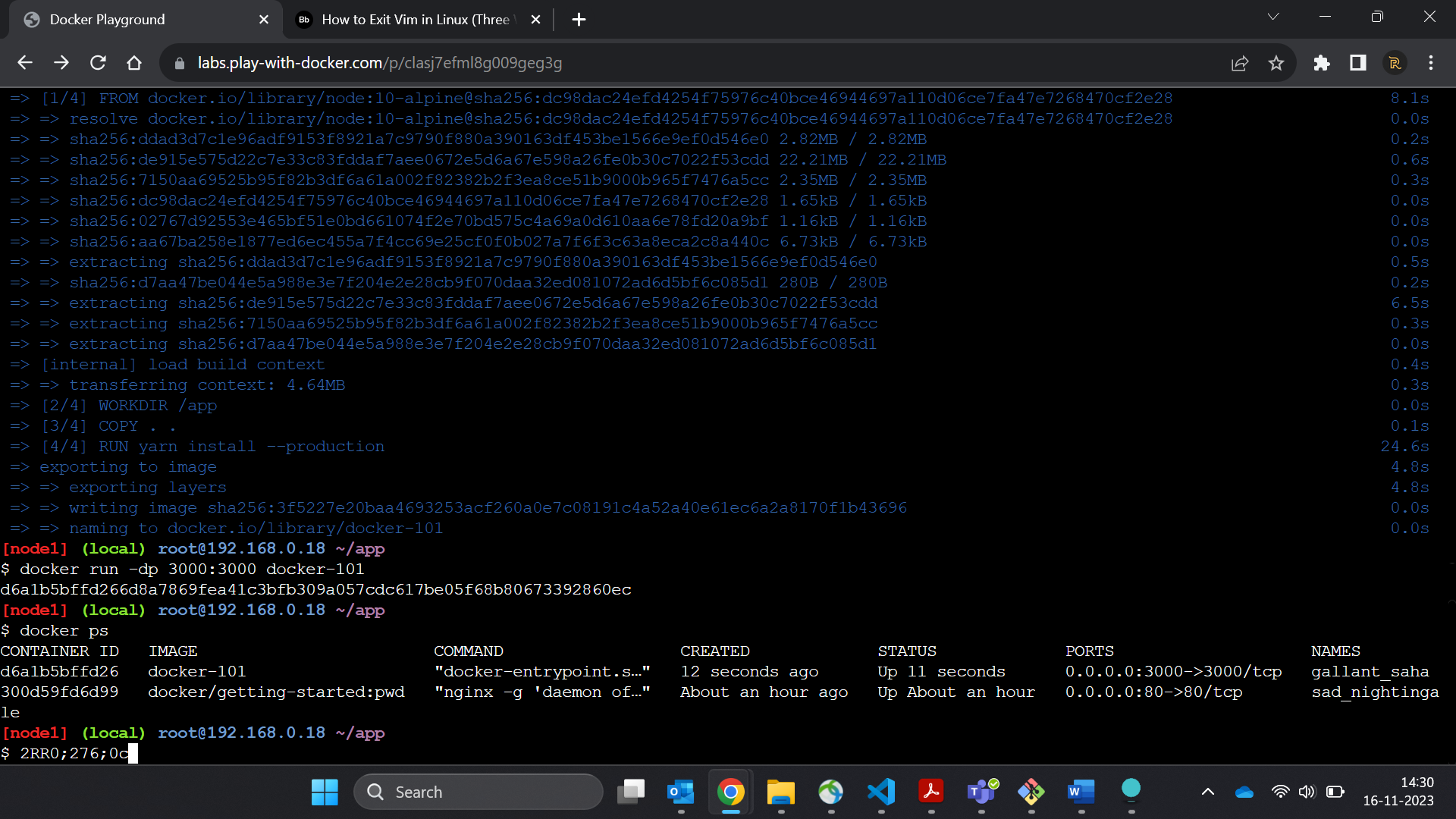
4. Create a file named Dockerfile with the following contents.



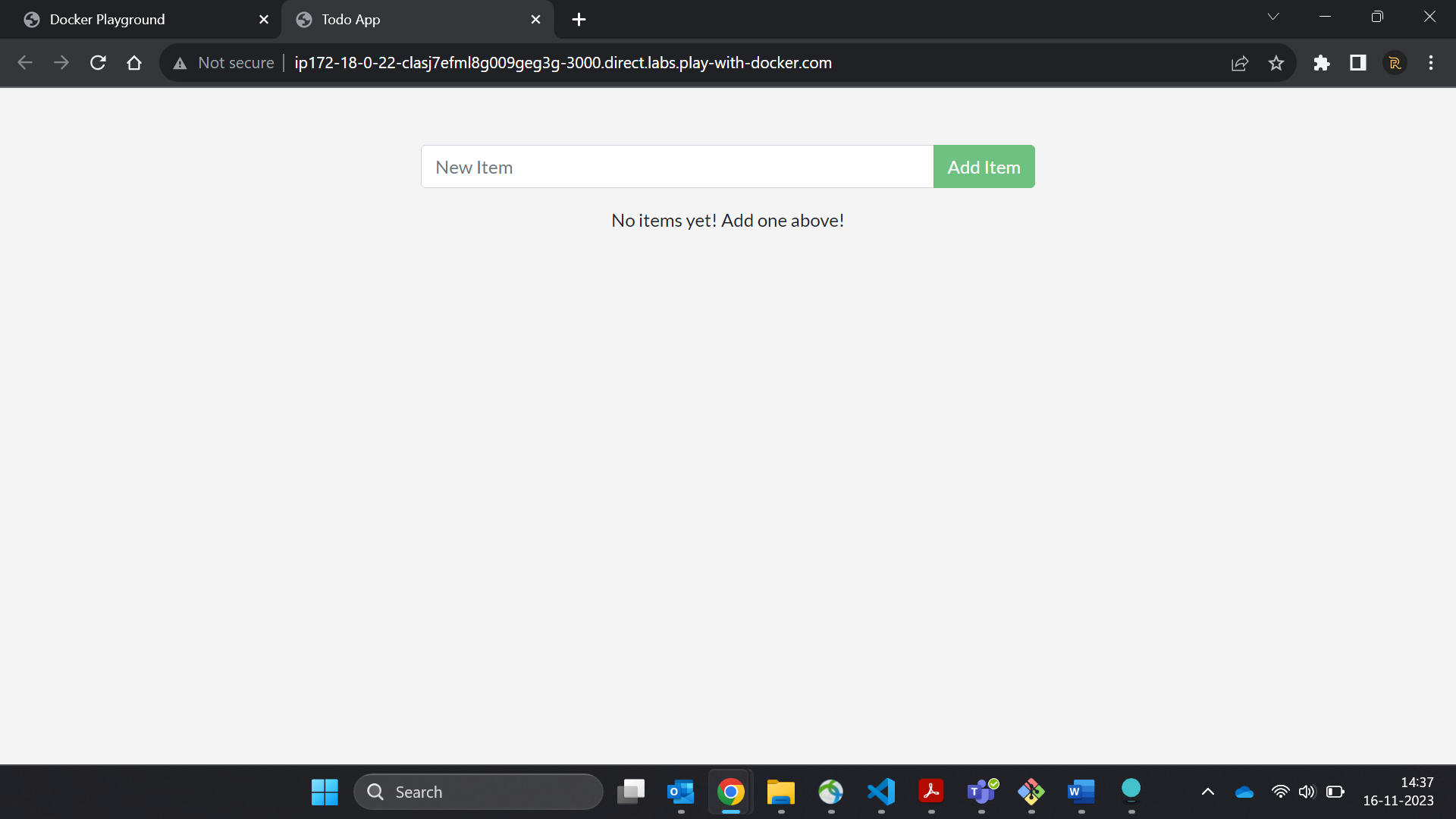
5. Build the container image using the docker build command



6…Start your container using the docker run command:



7….Open the application by clicking on the "3000" badge at the top of the PWD interface. Once open, you should have an empty todo list!

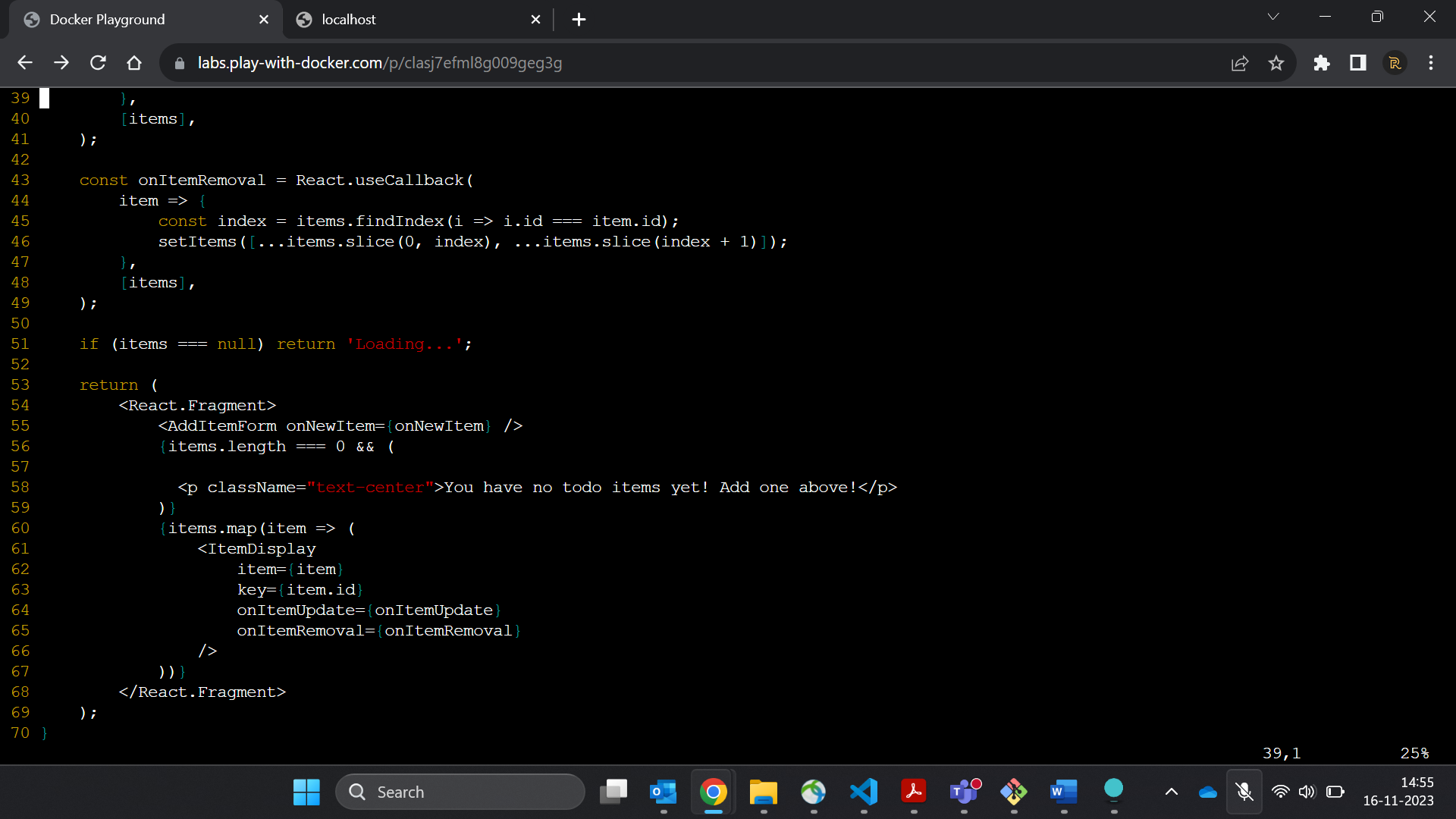


8…Go ahead and add an item or two and see that it works as you expect. You can mark items as complete and remove items

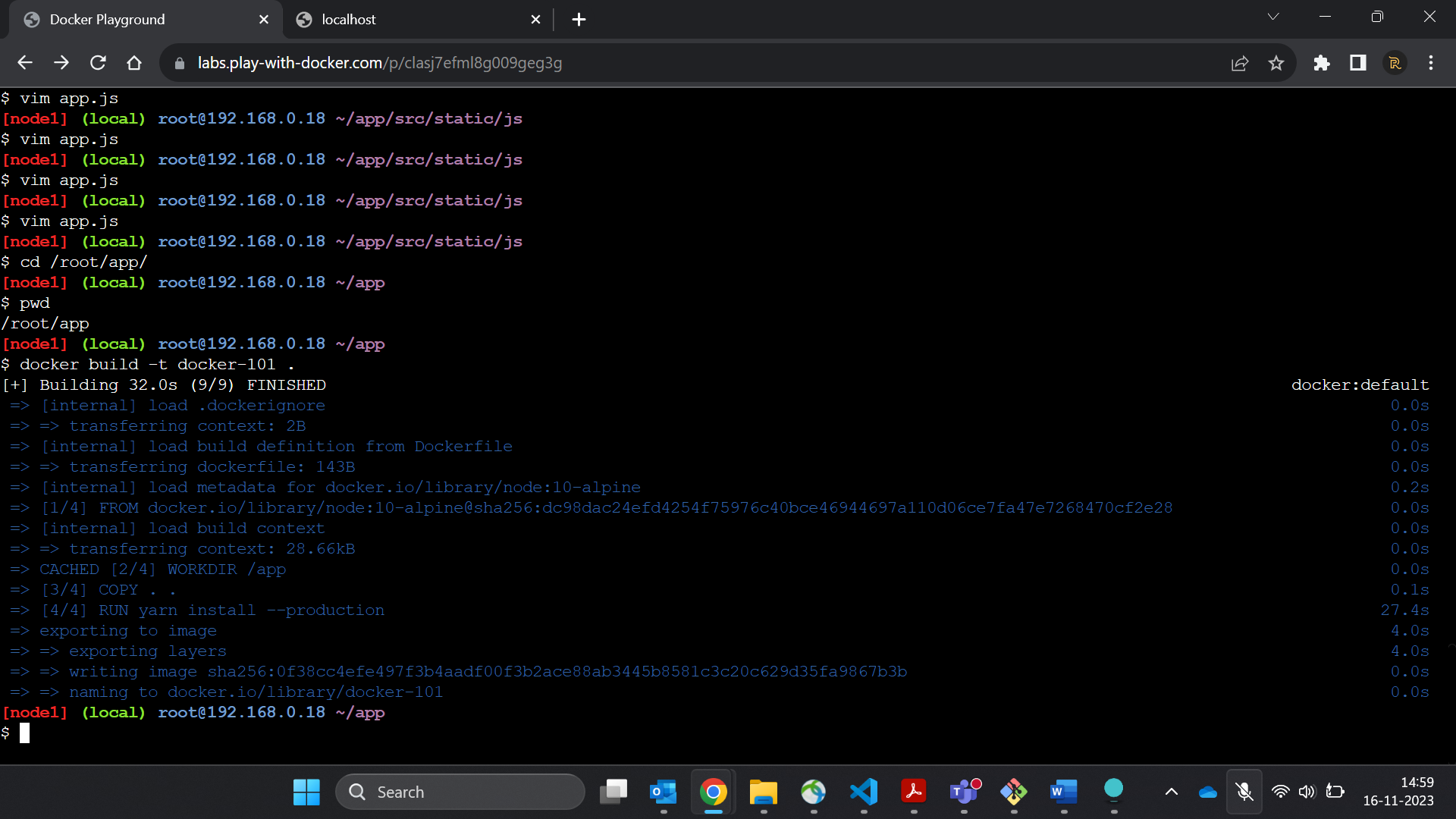


# 2……..**Updating our App**

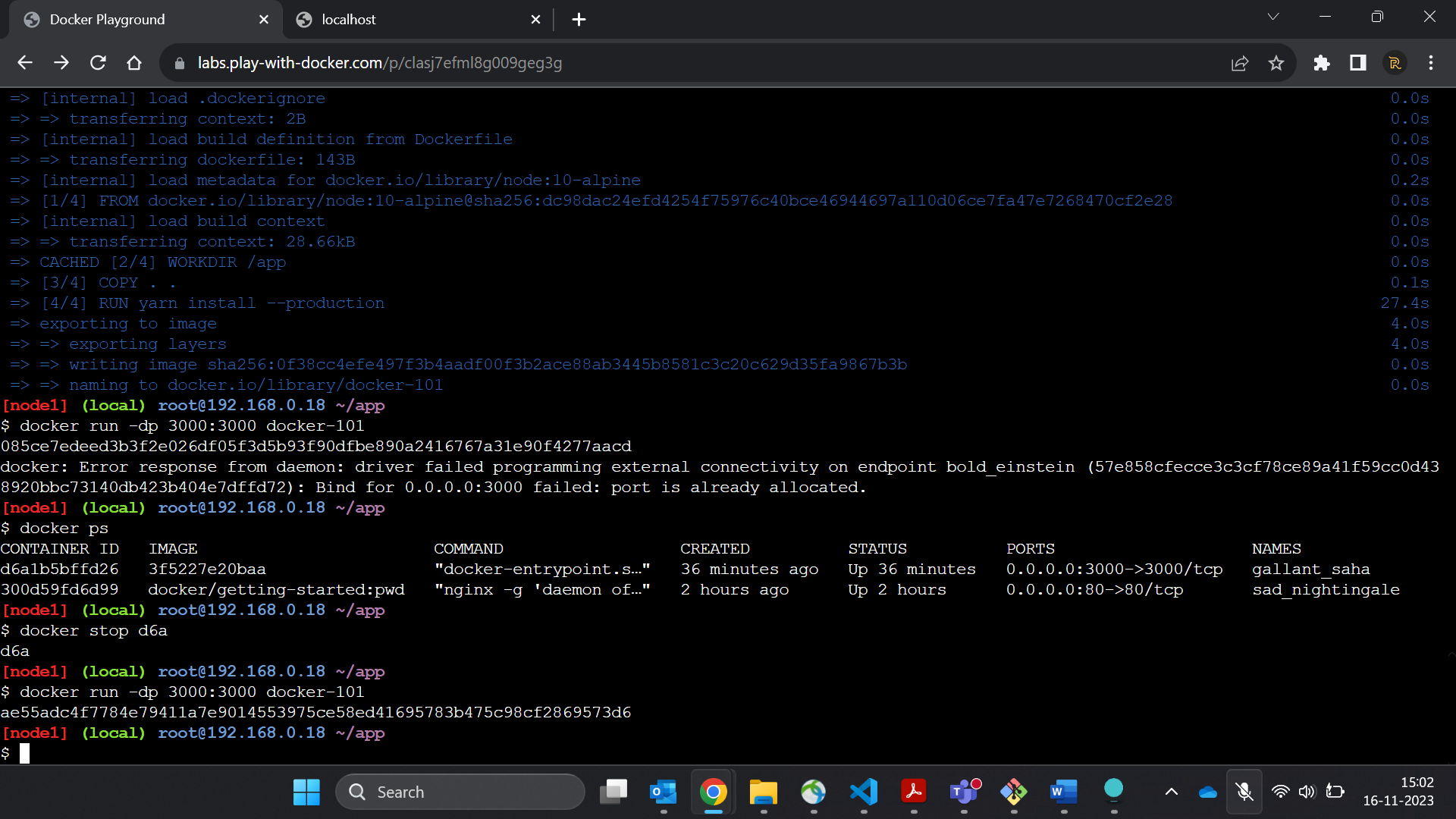
1. In the ~/app/src/static/js/app.js file, update line 56 to use the new empty text. ([Editing files in PWD tips here](http://ip172-18-0-22-clasj7efml8g009geg3g-80.direct.labs.play-with-docker.com/pwd-tips#editing-files))



2.Let's build our updated version of the image, using the same command we used before.



3. Let's start a new container using the updated code.



4.Get the ID of the container by using the docker ps command.

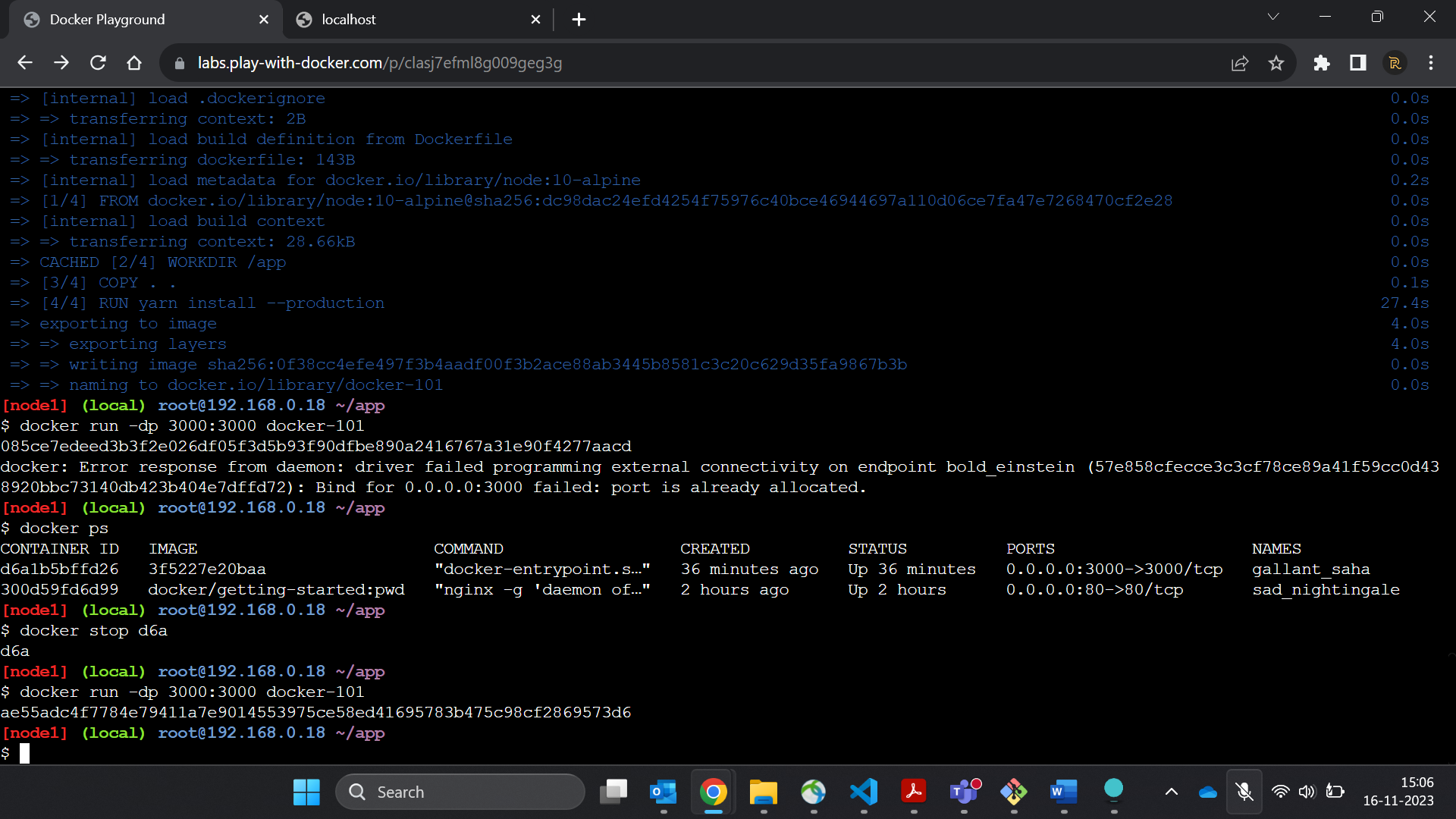
5.Use the docker stop command to stop the container.

1. docker stop <the-container-id>

6.Once the container has stopped, you can remove it by using the docker rm command.

7.Now, start your updated app.

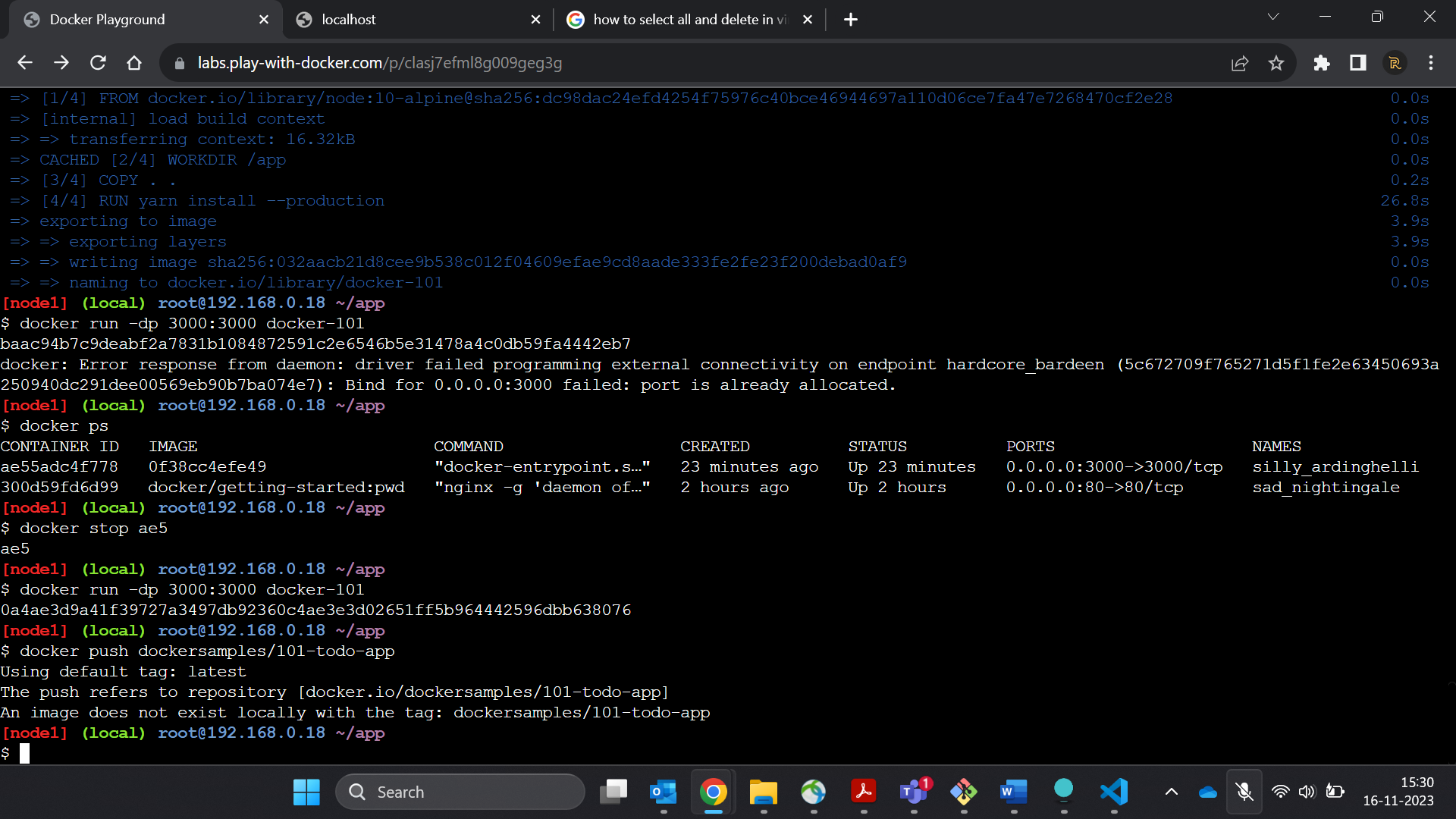
8.Open the app and you should see your updated help text!



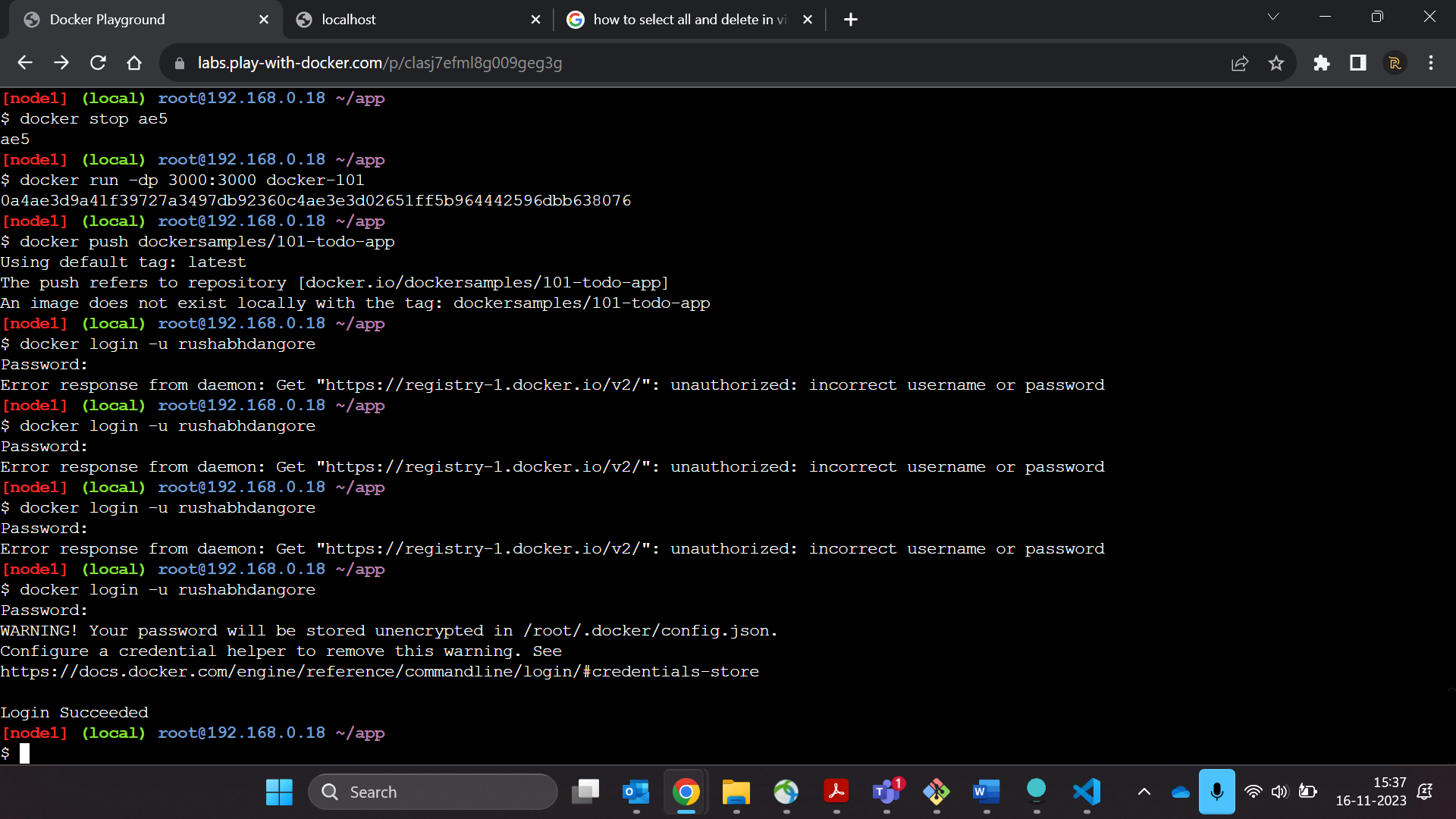
**3….Sharing the app**

## Pushing our Image[¶](http://ip172-18-0-22-clasj7efml8g009geg3g-80.direct.labs.play-with-docker.com/tutorial/sharing-our-app/#pushing-our-image)

1. Back in your PWD instance, try running the command. You should get an error that looks something like this:

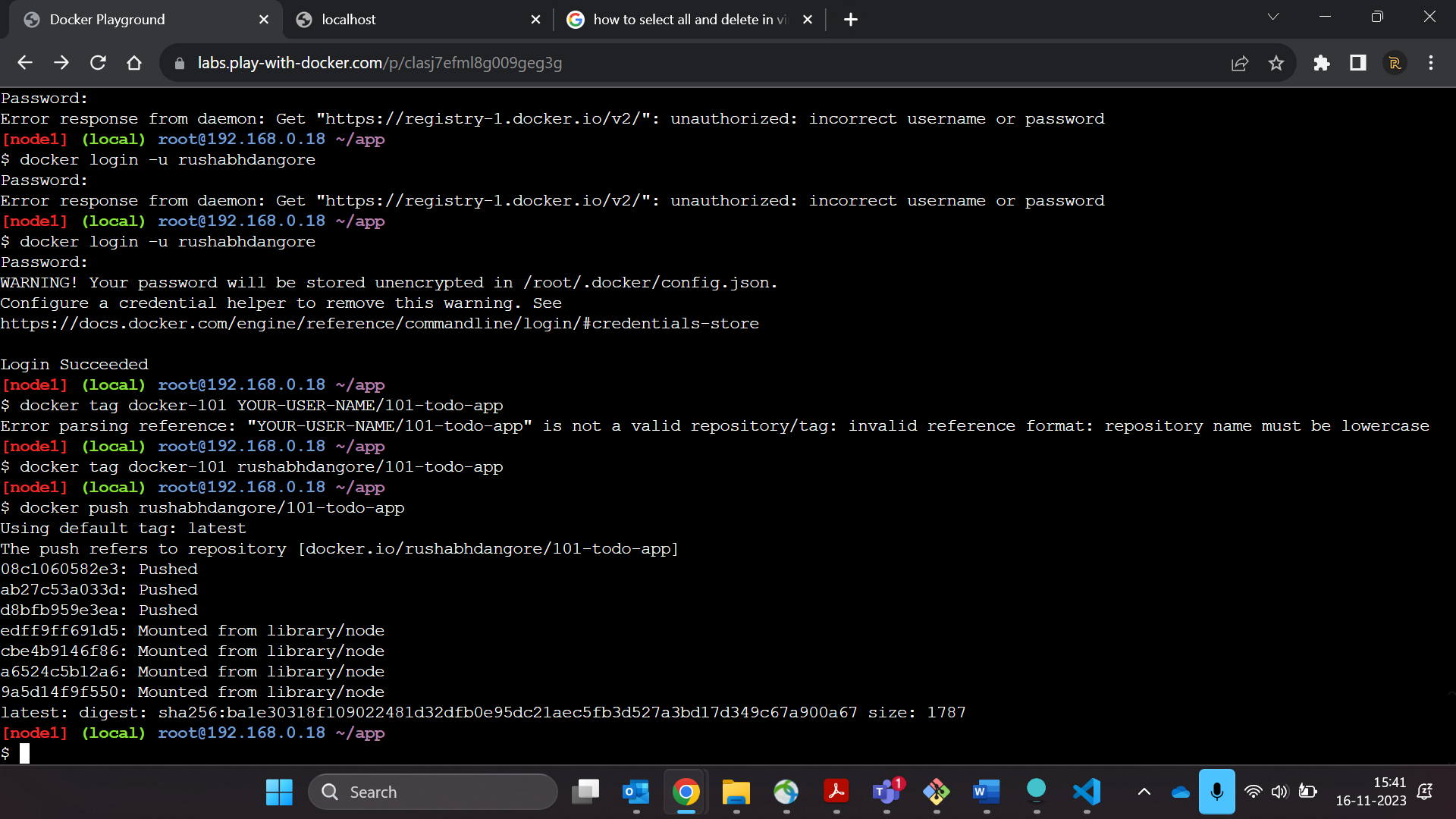


2.Login to the Docker Hub using the command docker login -u YOUR-USER-NAME.



3.Use the docker tag command to give the docker-101 image a new name. Be sure to swap out YOUR-USER-NAME with your Docker ID.

4. Now try your push command again. If you're copying the value from Docker Hub, you can drop the tagname portion, as we didn't add a tag to the image name.

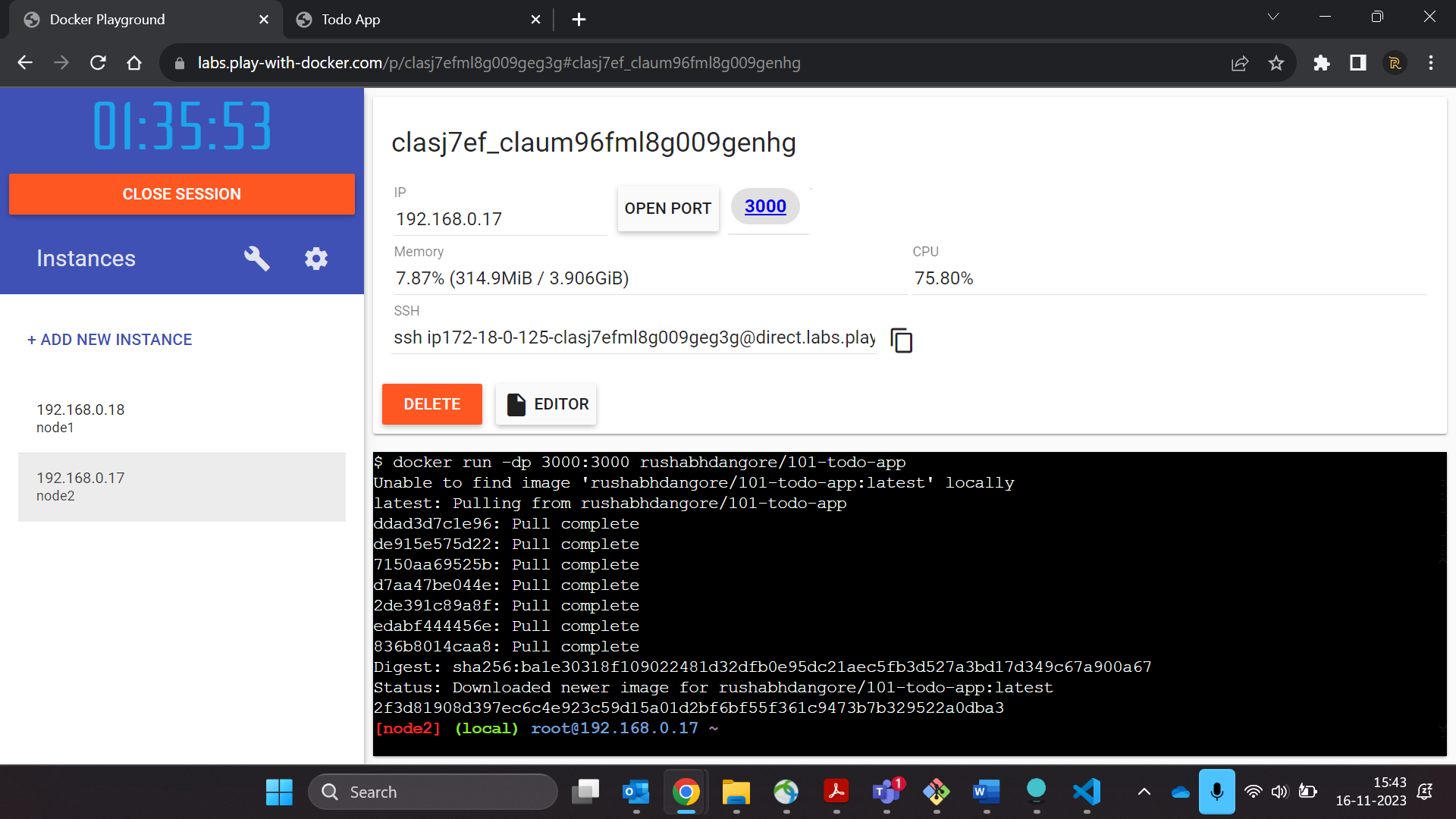


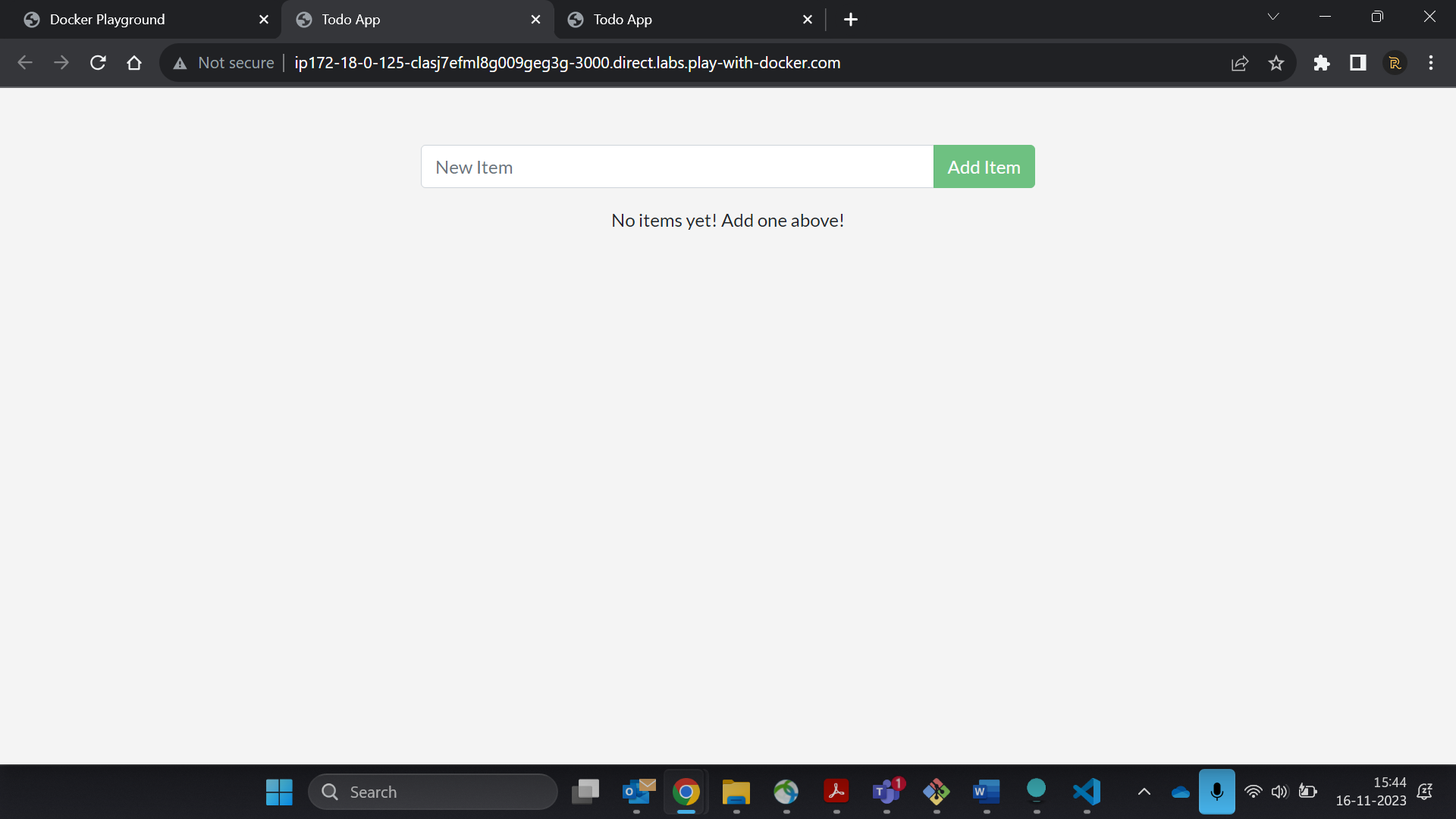
## Running our Image on a New Instance[¶](http://ip172-18-0-22-clasj7efml8g009geg3g-80.direct.labs.play-with-docker.com/tutorial/sharing-our-app/#running-our-image-on-a-new-instance)

Now that our image has been built and pushed into a registry, let's try running our app on a brand instance that has never seen this container!

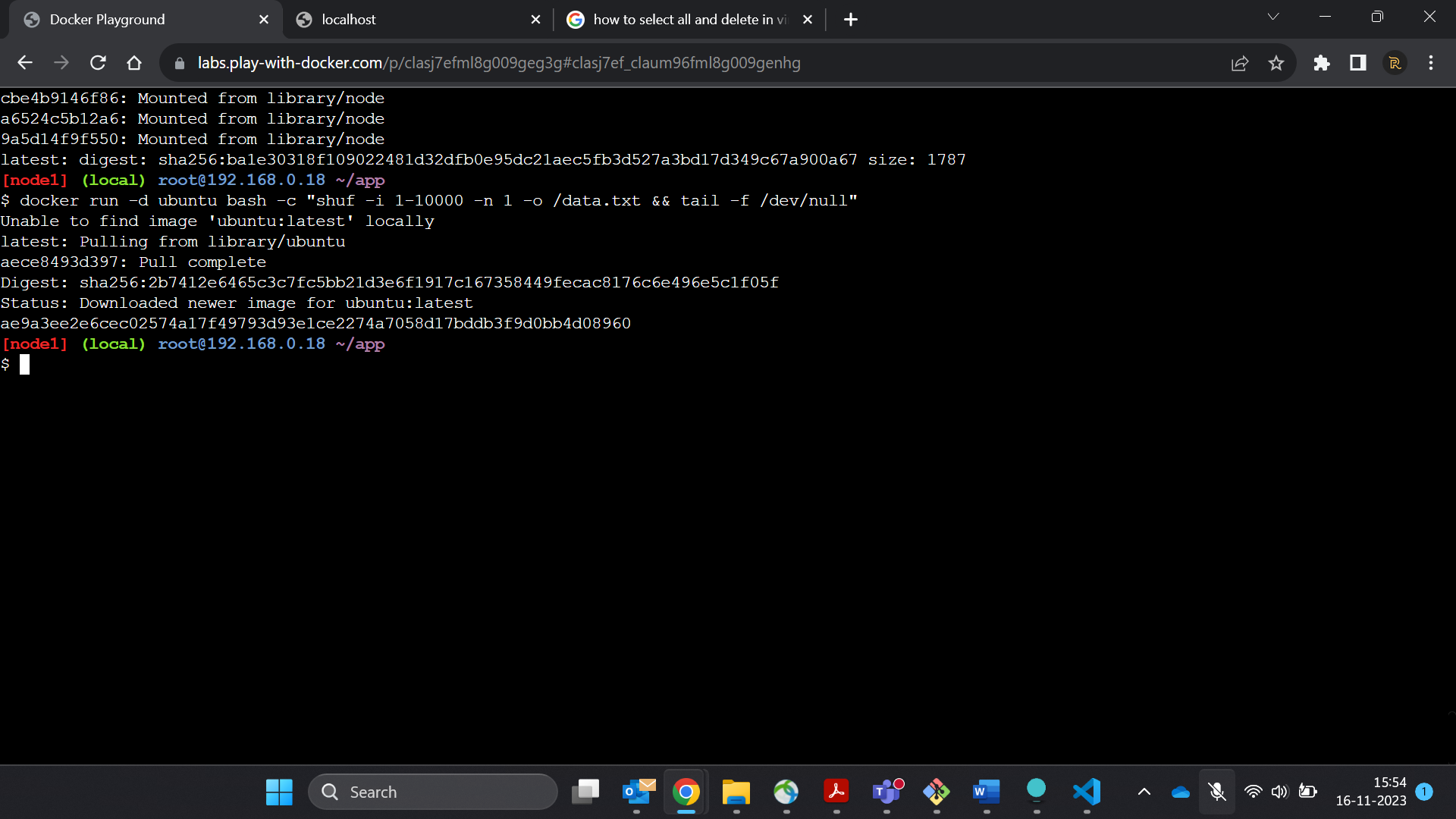
1. Back in PWD, click on **Add New Instance** to create a new instance.
2. In the new instance, start your freshly pushed app.

You should see the image get pulled down and eventually start up!

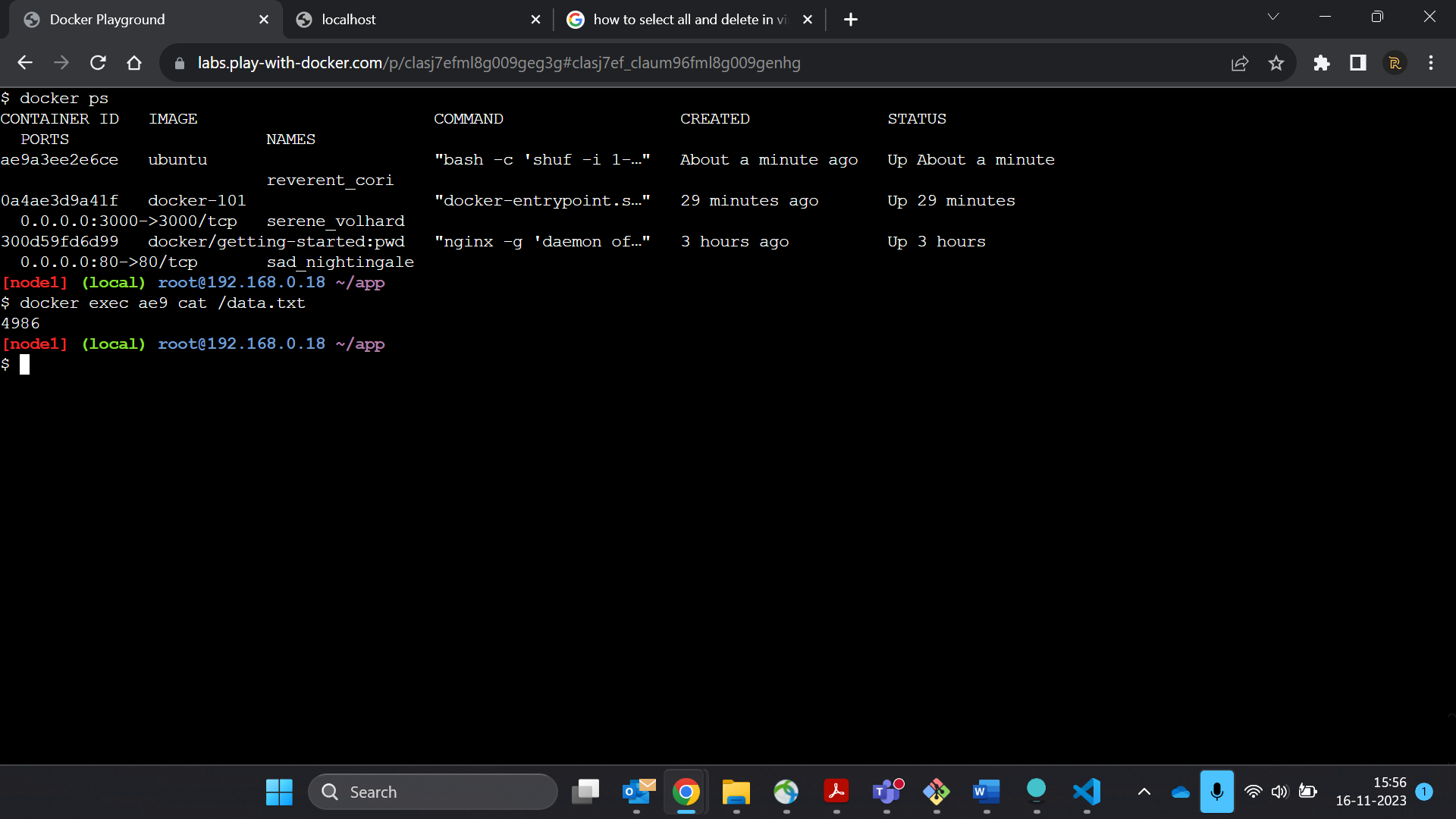




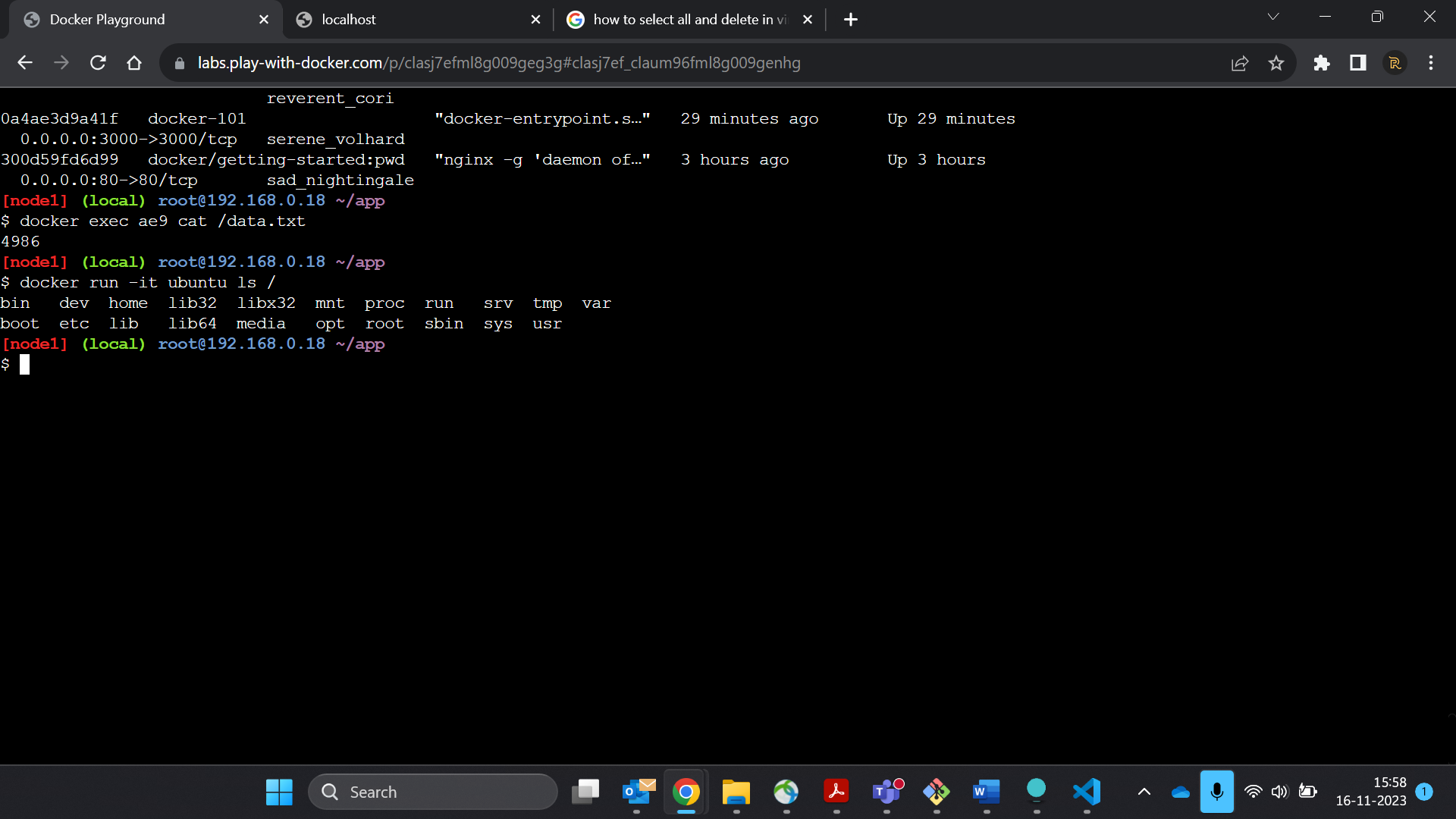
# **Persisting our DB**



2. Validate we can see the output by exec'ing into the container. To do so, you need to get the container's ID (use docker ps to get it).

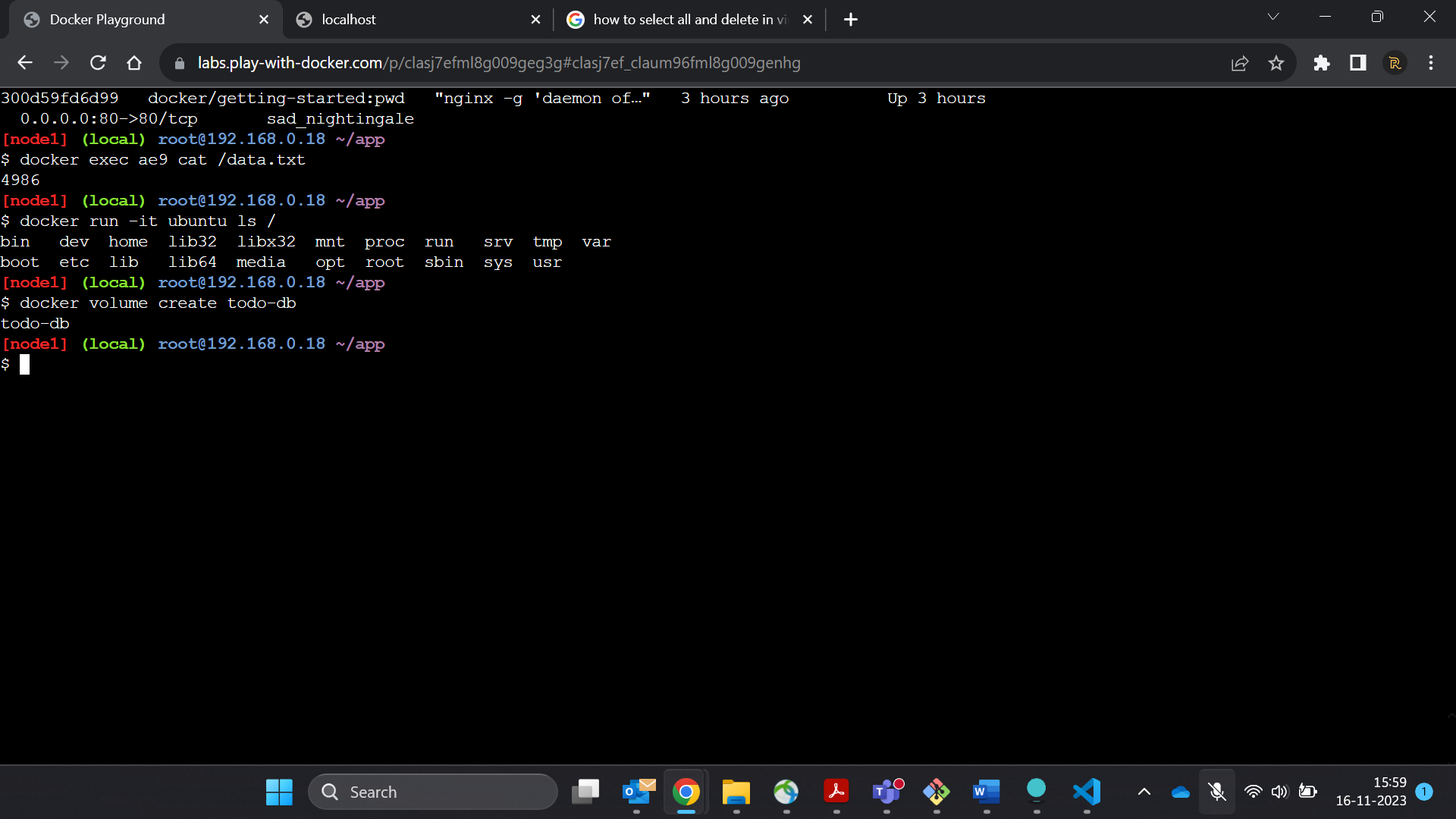


1. Now, let's start another ubuntu container (the same image) and we'll see we don't have the same file.



## Persisting our Todo Data

1.Create a volume by using the docker volume create command.



2..Start the todo container, but add the -v flag to specify a volume mount. We will use the named volume and mount it to /etc/todos, which will capture all files created at the path.

