**Experiment 9:** Docker Ignore and OnBuild Optimization Implementation

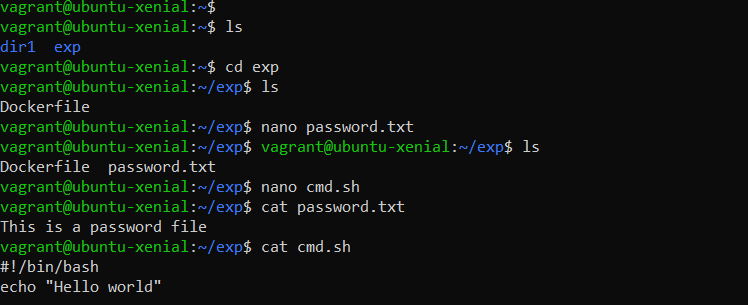
1. Docker Ignore Implementation
2. Logon to the virtual machine and create a new directory ‘exp’, enter into the directory and create some files in it.
3. One of the files is the password.txt storing a text and other file is cmd.sh storing the shell script to print ‘Hello world’ in terminal. Now create a Dockerfile and the code inside of Dockerfile should be:

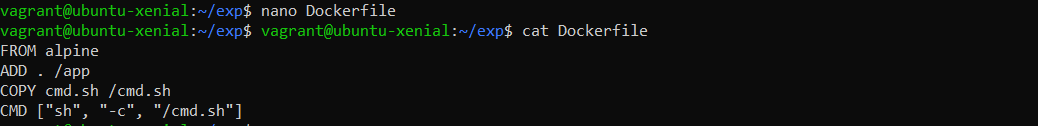
FROM alpine

ADD . /app

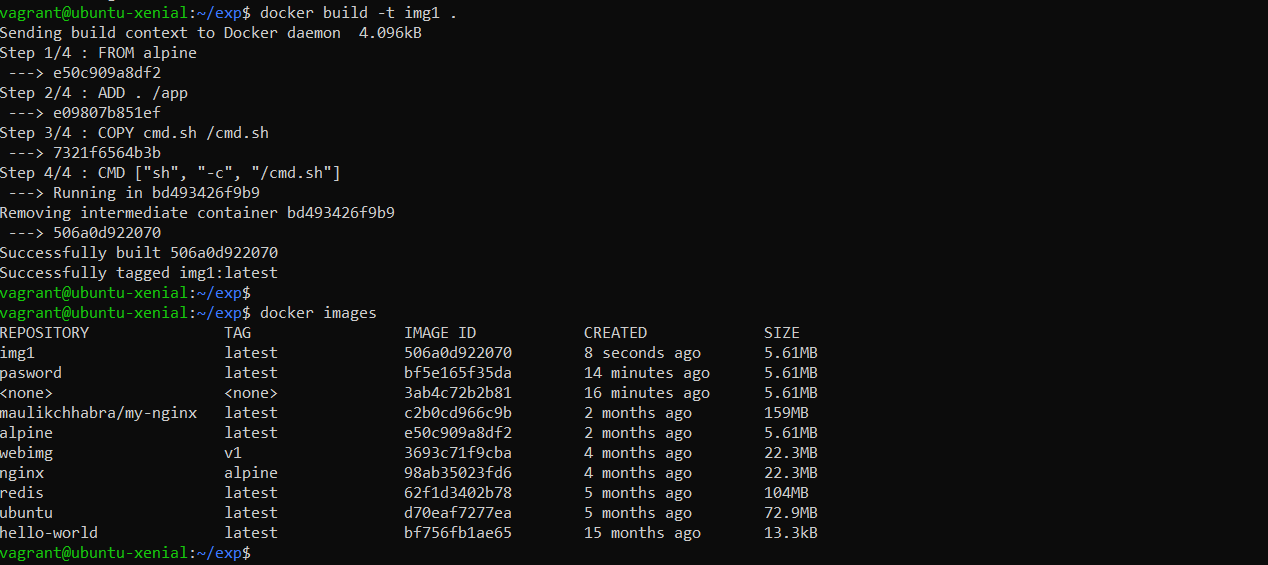
COPY cmd.sh /cmd.sh

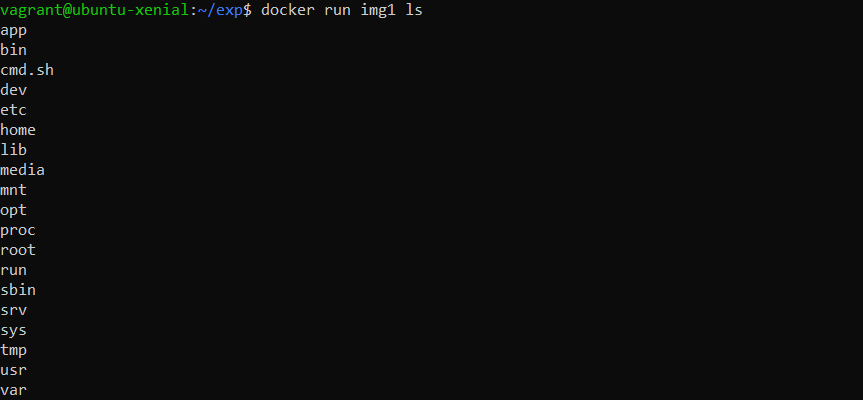
CMD ["sh", "-c", "/cmd.sh"]



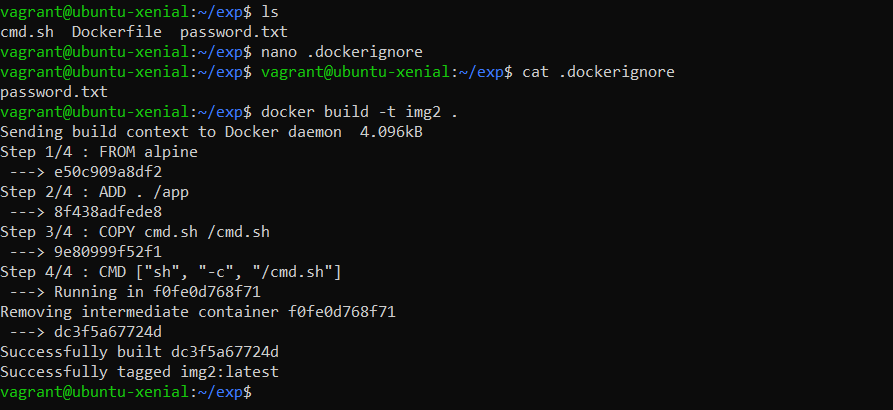


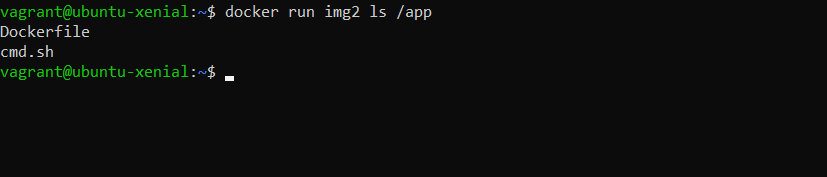
1. Now to build the image of Dockerfile, use the command ‘docker build -t <image-name>.’. Here the name of the image is img1.
2. When we execute the command, each layer in the Dockerfile will get execute and the files we created in the exp directory will be available in the app folder in the image.
3. To check the listing of the new image, run the command ‘docker images’.
4. Now we have to create a container of the same, for that, run the command ‘docker run img1’ and to list the content use command ls, for app directory use ‘ls app’.





1. Now for implementing docker ignore, create a file ‘.dockerignore’ in the same exp directory and in that file, enter the name of the file you want to ignore while building the container, in this case we are writing password.txt in the .dockerignore file.
2. Again build the image with name img2 and create a container of the same with the command ‘docker run img2 ls /app’. The command will create a container and list the files in the app directory of the alpine container. Here we can see that password.txt did not appear in the app directory as it is listed in the .dockerignore file.





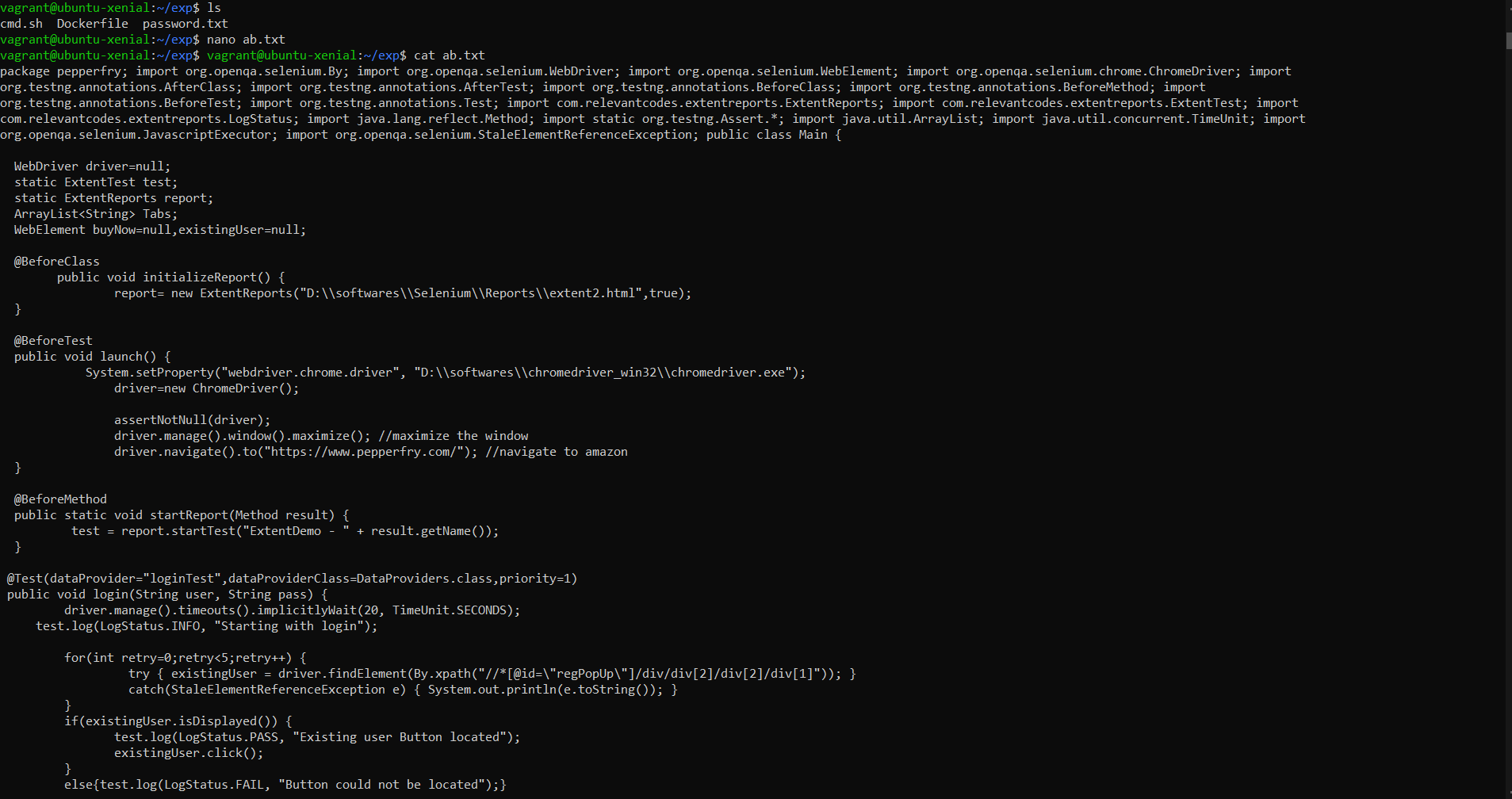
1. Now in the same exp directory create a new file ab.txt and input a big code in it.
2. Build its image as img3, we can view that the size of the image is 14.85 kB.
3. Run a container of the same and view the list of files in the app folder.
4. Now edit the .dockerignore file and in the file write:

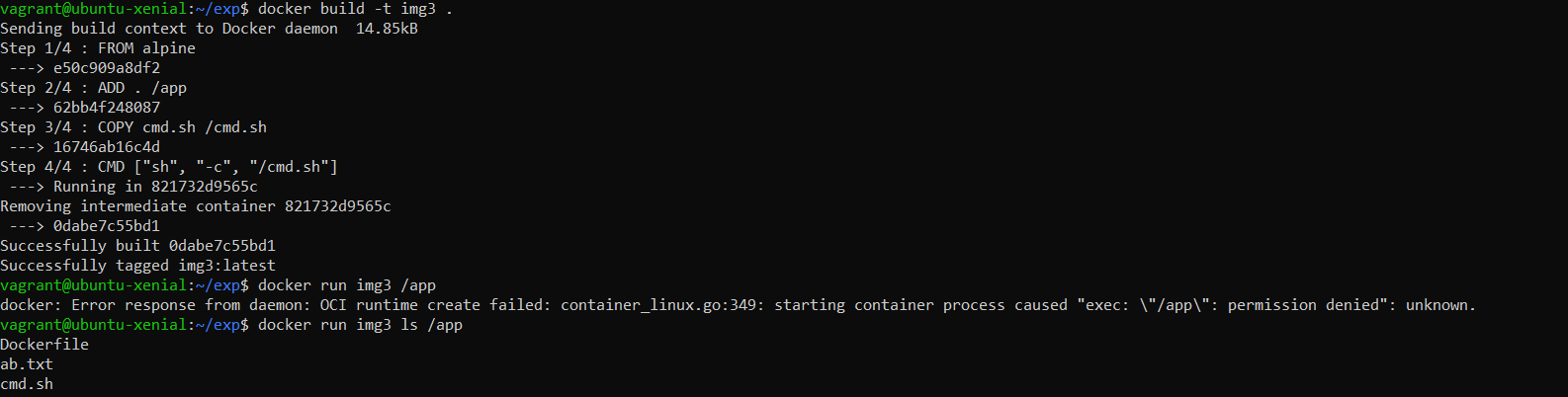
\*.txt

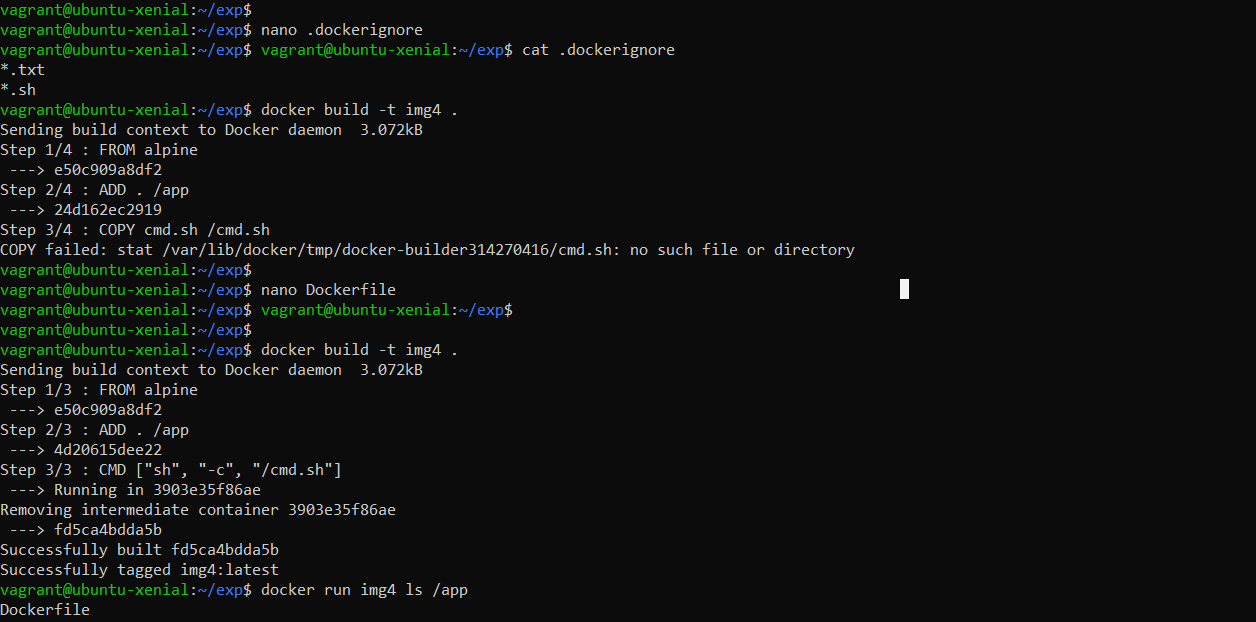
\*.sh

This will ignore all the files of .txt and .sh file type while building the image and the hence these will not be listed in the app folder of the container.

1. Build the image, we can see the size of image reduced to 3.072 kB.
2. Now run a container and list the files in app directory. Only Dockerfile is present in the app directory.
3. In the same case we can ignore large files of 100-200 mB sizes or the files not really important, this will also reduce the image size and increase its building, downloading and uploading speed.







1. OnBuild Optimization Implementation
2. First create a Dockerfile using the touch command.
3. Open the Editor and write the following code in the Dockerfile:

FROM node:7

RUN mkdir -p /usr/src/app

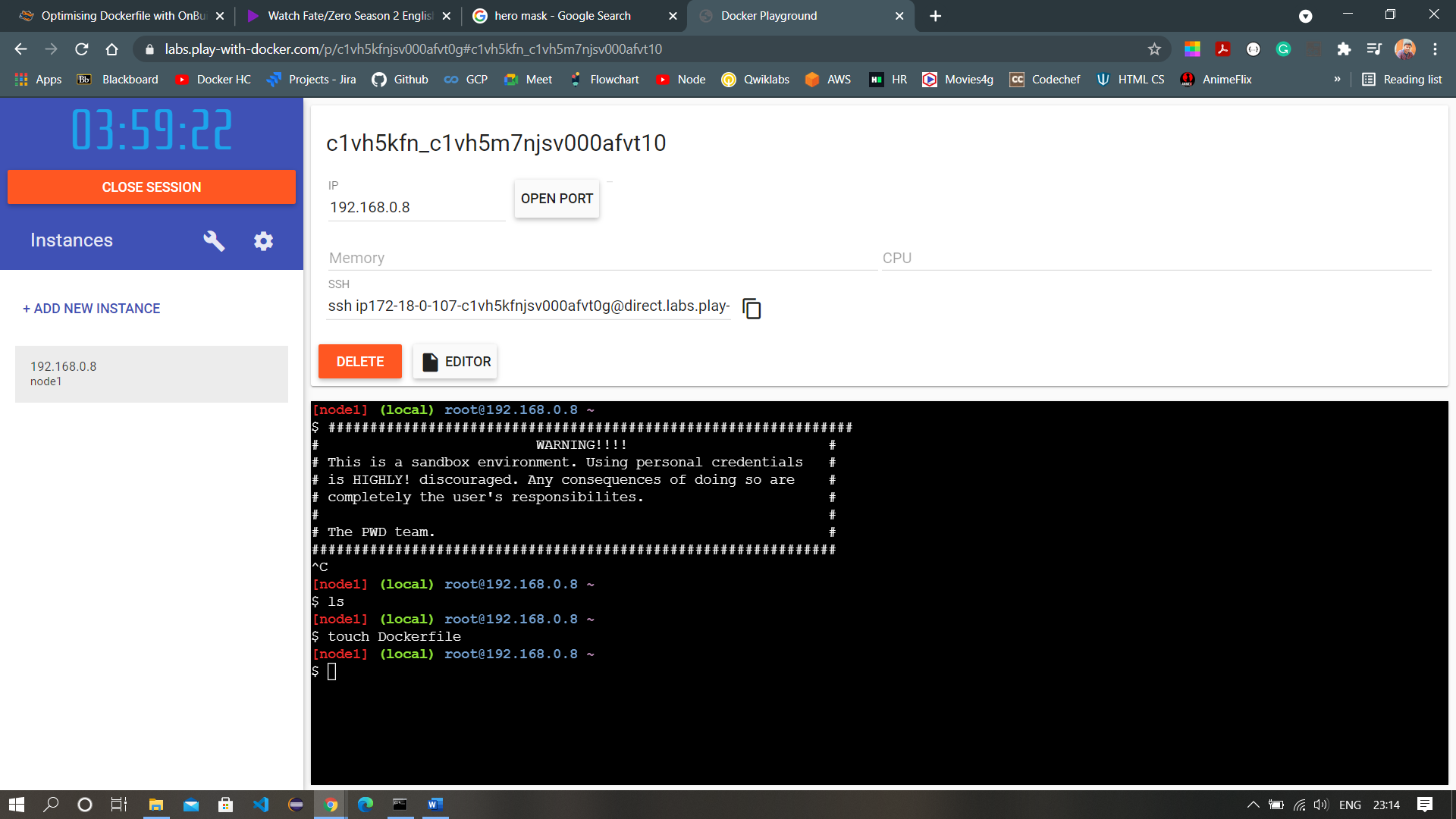
WORKDIR /usr/src/app

ONBUILD COPY package.json /usr/src/app/

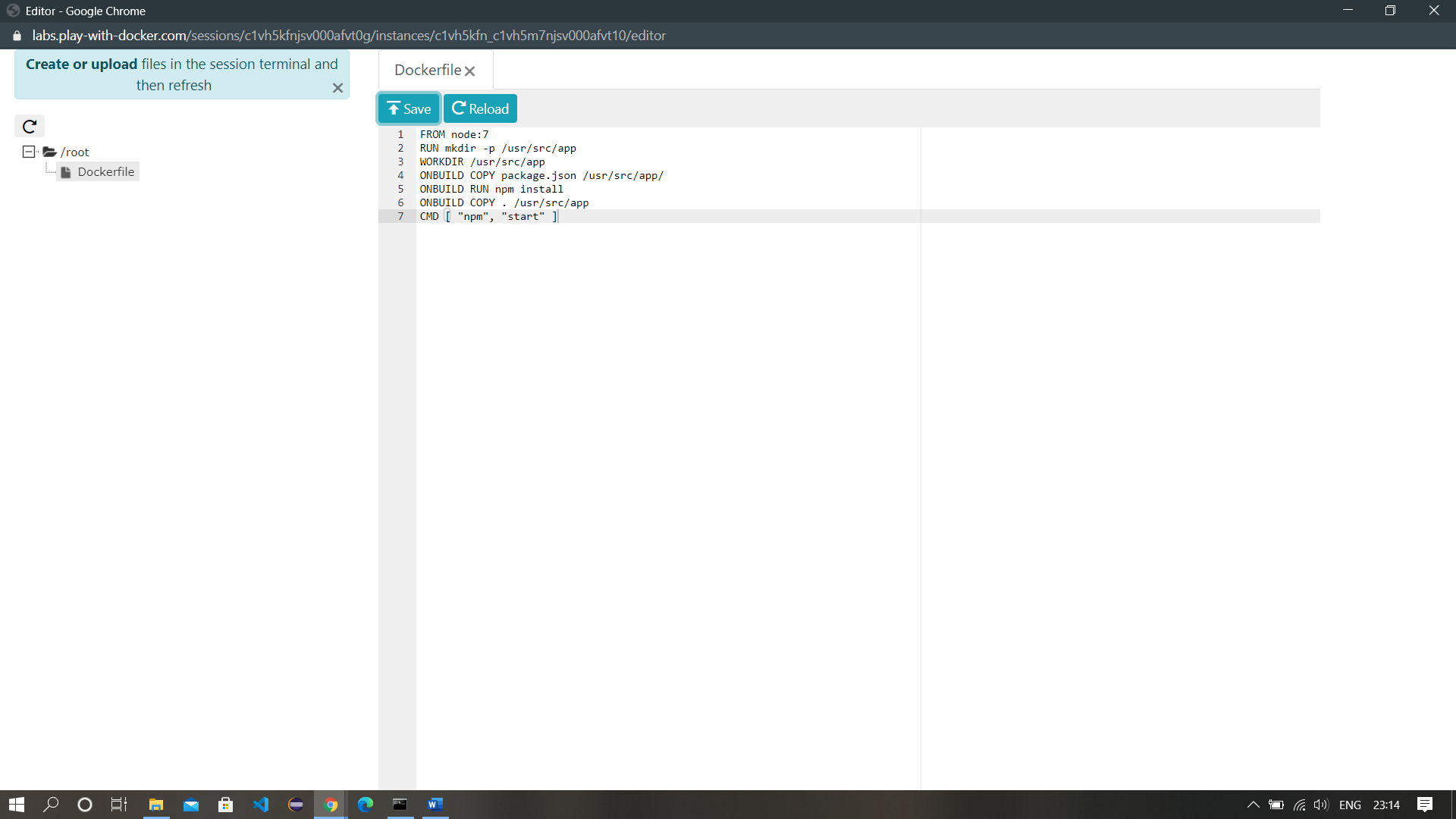
ONBUILD RUN npm install

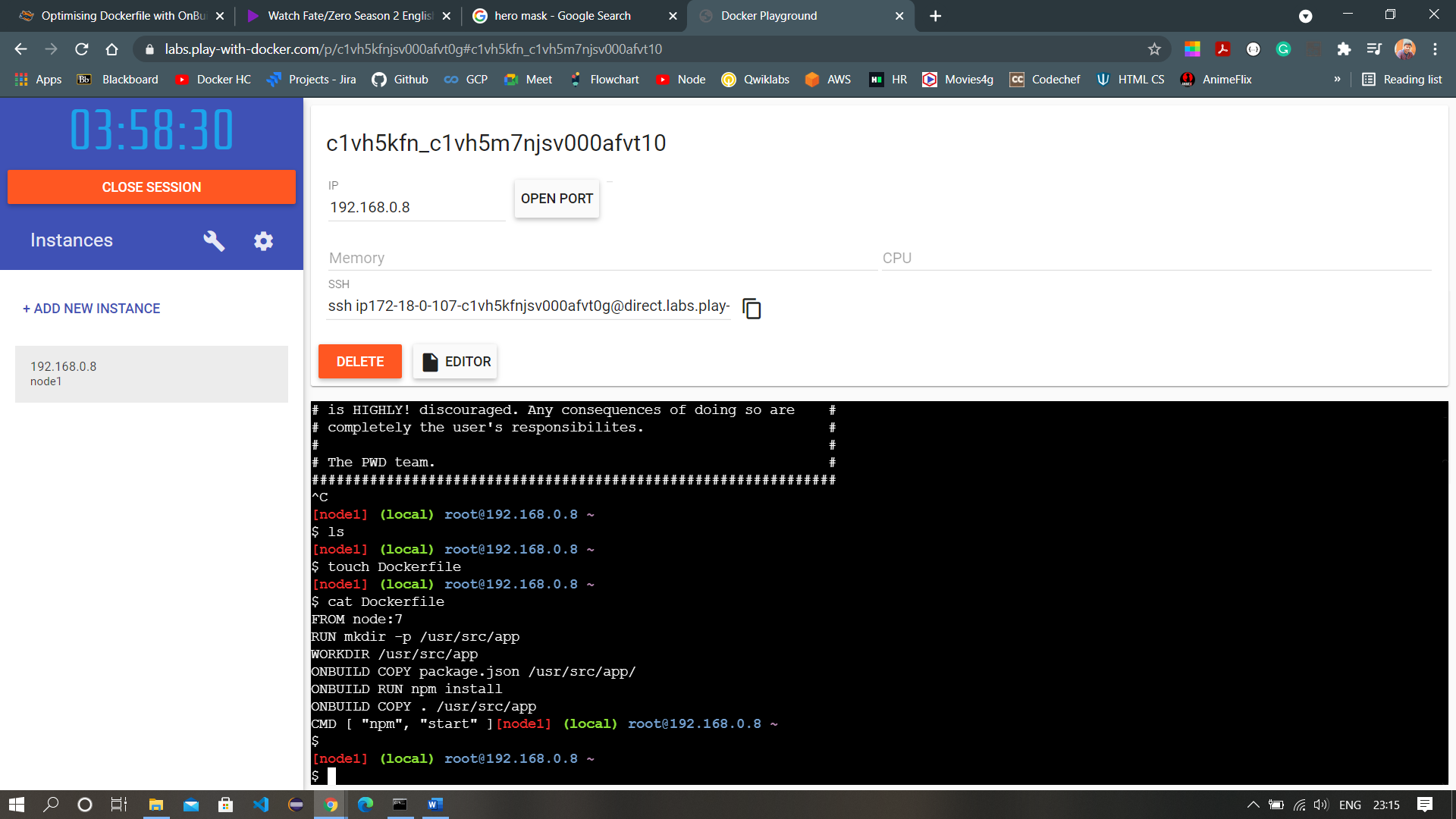
ONBUILD COPY . /usr/src/app

CMD [ "npm", "start" ]

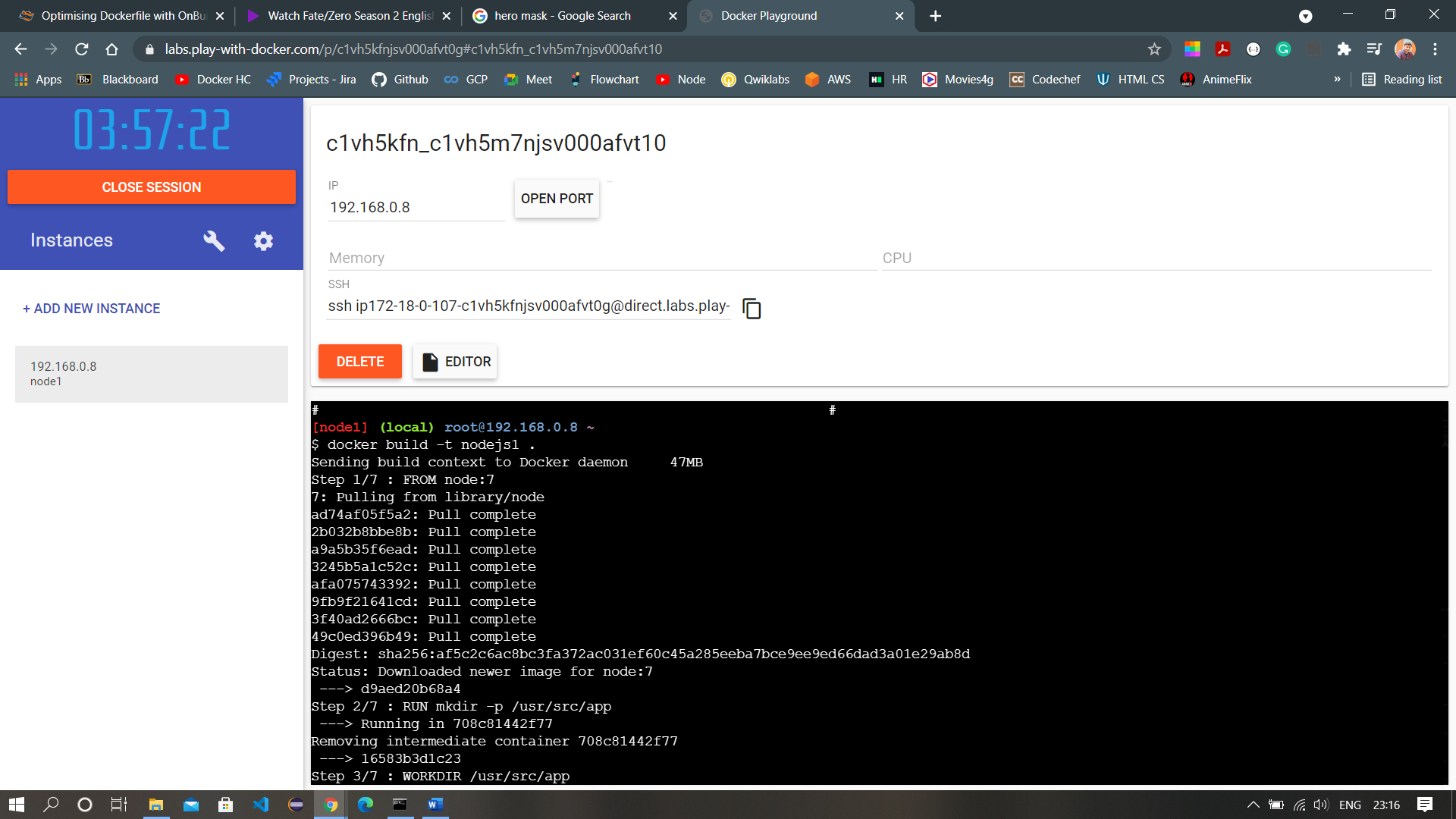


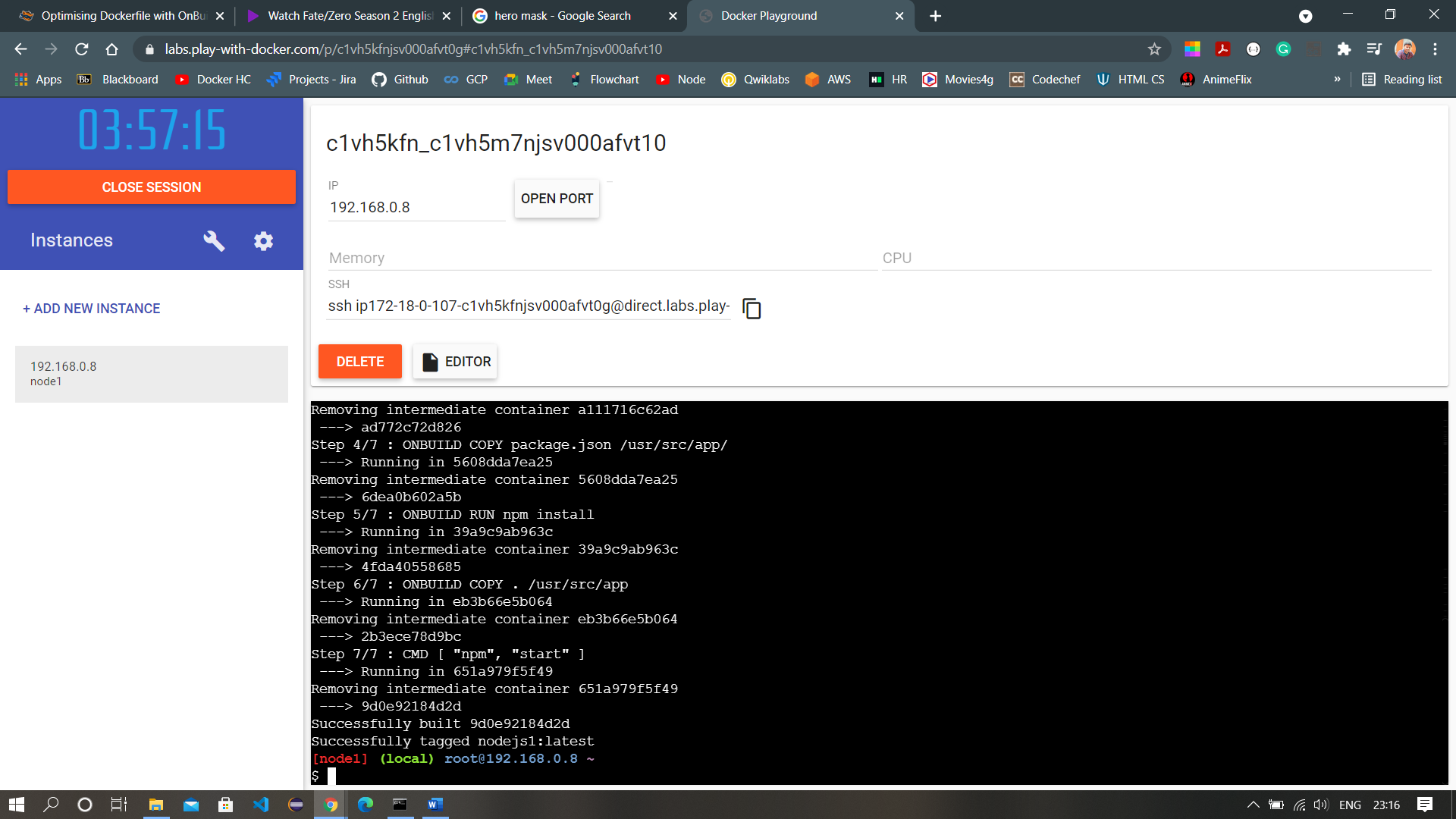
The code is to install NodeJS in the system. In the code, we can see a keyword ‘ONBUILD’ at the start of some instructions. These instructions will not run when the Dockerfile will be run, rather these commands run when a new image is to be made taking the image in this Dockerfile as a base image.





1. Build the image using the command ‘docker build -t <image-name> .’. Here image-name is nodejs1.





1. When the image build is complete, check its listing in docker images and check if any running containers are there or not using the command ‘docker ps’.
2. Now, create a new container of this image with the command ‘docker run -d –name myapp -p 3000:3000 nodejs1’.

