Scenario3:

**Building container image:**

**Docker images:**

**\*** A Dockerfile defines all the steps required to create a Docker image with your application configured and ready to be run as a container.

**Step 1: Base images:**

**\*** Base image is the same images from the Docker Registry which are used to start containers.

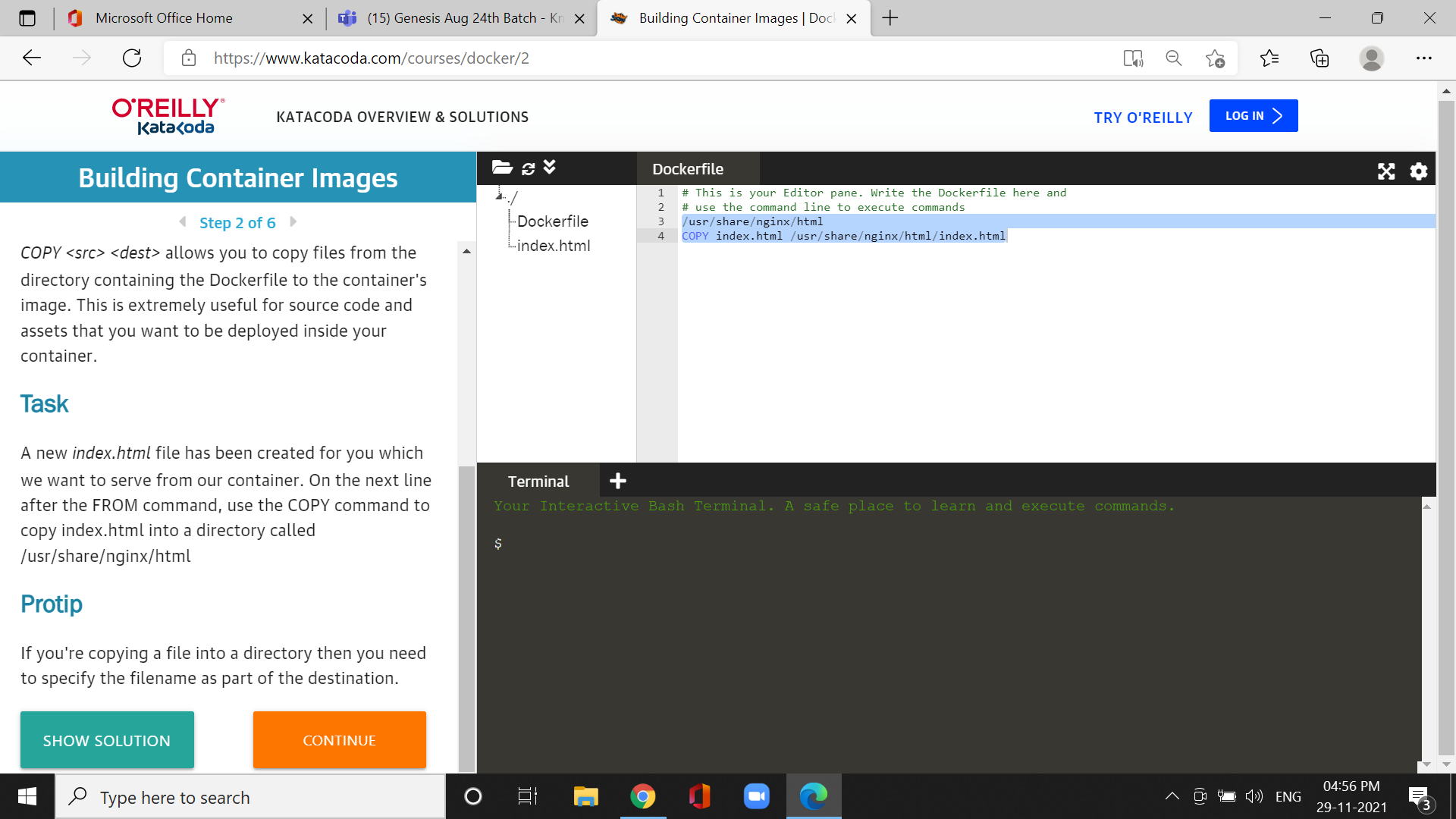
**Step 2: Running commands:**

**\****RUN <command>* allows you to execute any command as you would at a command prompt.

\**COPY <src> <dest>* allows you to copy files from the directory containing the Dockerfile to the container's image.

Cmd : /usr/share/nginx/html

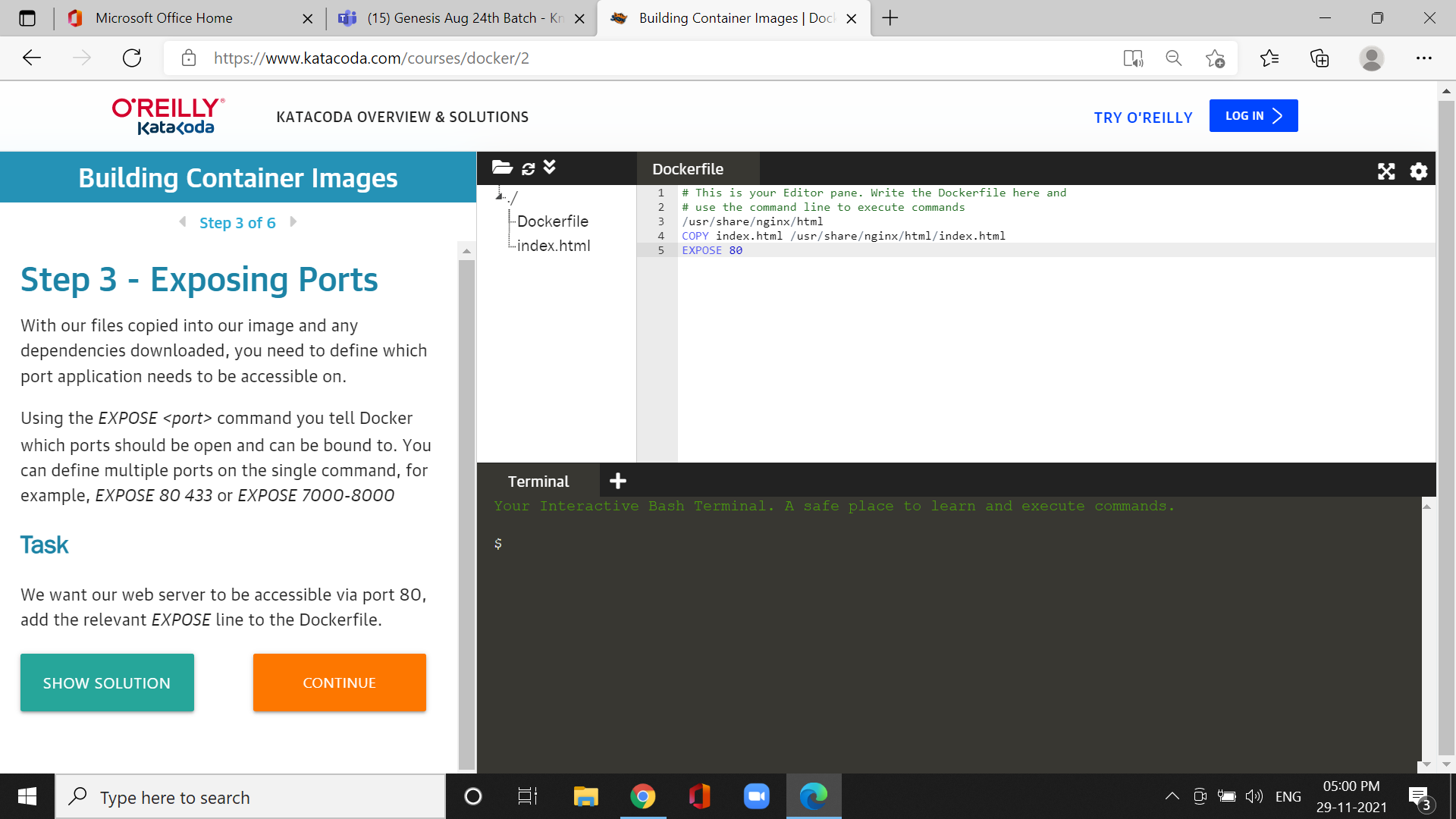
COPY index.html /usr/share/nginx/html/index.html



\*On the next line after the FROM command, use the COPY command to copy index.html into a directory

Step3: Exporting images

cmd:EXPOSE 80



\*multiple ports on the single command.

**Step4:default commands**

**\*** *CMD* line in a Dockerfile defines the default command to run when a container is launched.

\*["cmd", "-a", "arga value", "-b", "argb-value"], which will be combined together and the command.

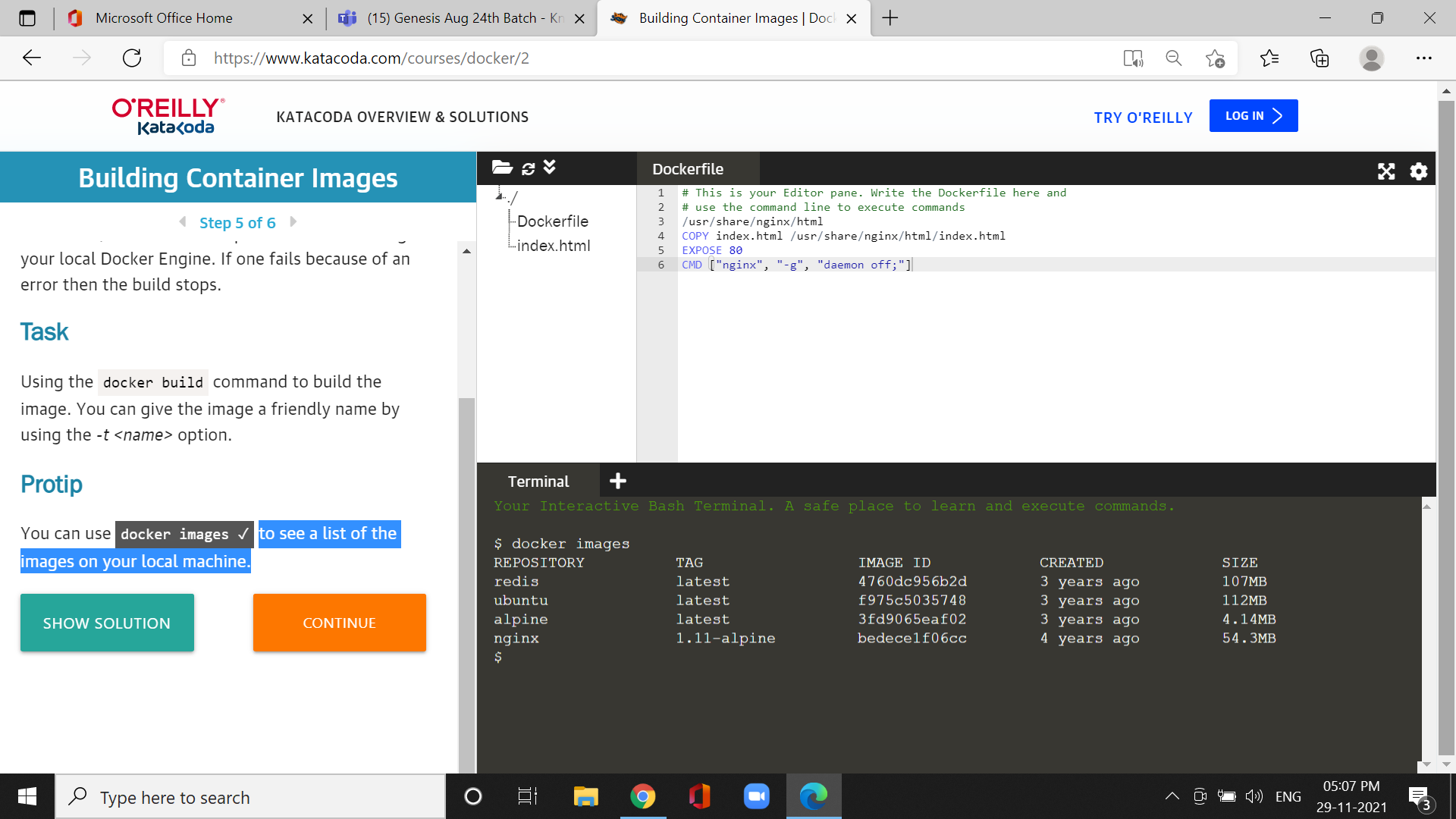
\*The command to run NGINX is nginx -g daemon off;. Set this as the default command in the Dockerfile.

cmd:CMD ["nginx", "-g", "daemon off;"]

**Step5:Building containers**

\*docker build command to build the image. You can give the image a friendly name by using the *-t <name>* option.

Cmd :docker image

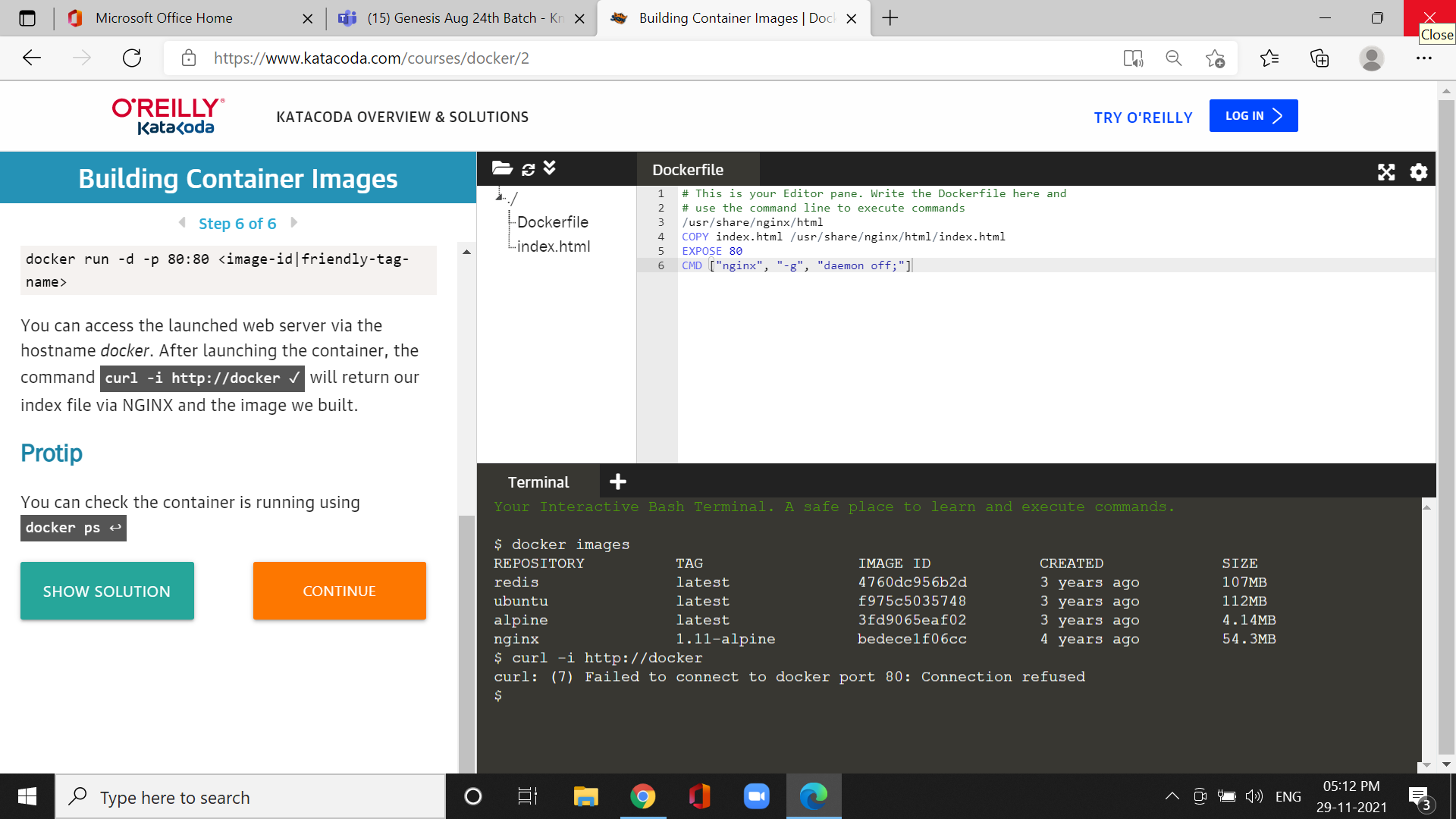


\* it is used to see a list of the images on your local machine.

**Step 6:Launching new image**

\*NGINX is designed to run as a background service so you should include the option *-d.*

*Cmd:curl -i* [*http://docker*](http://docker)

**

\*it will return our index file via NGINX and the image we built.