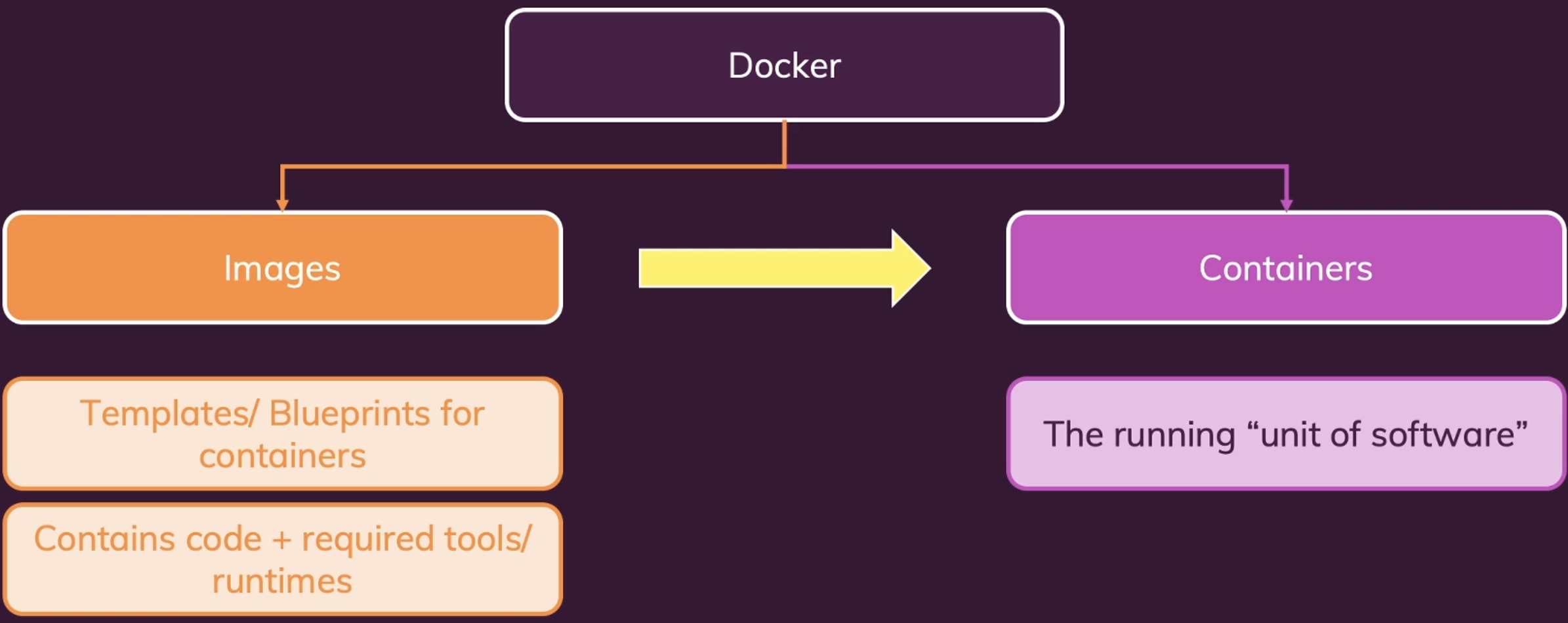
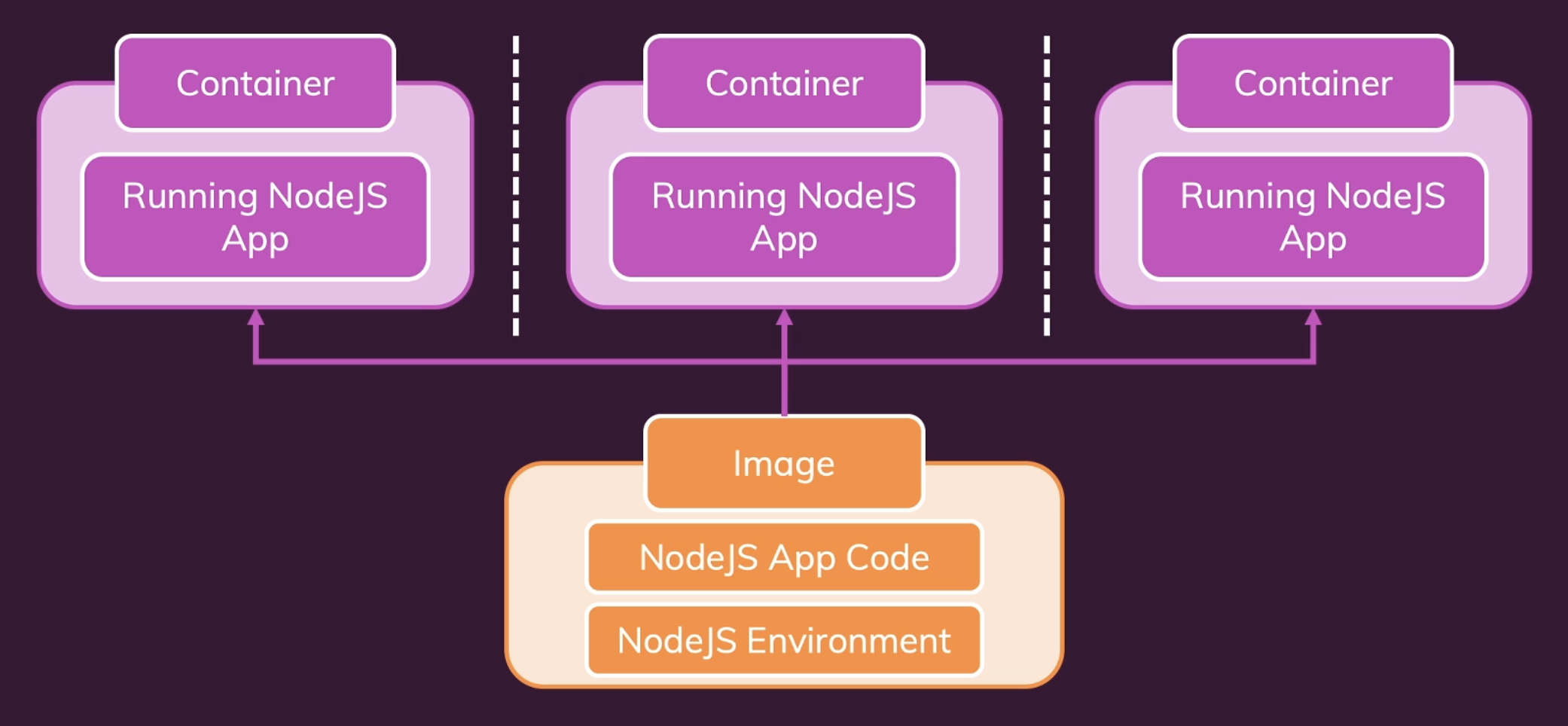
**Images and Containers**

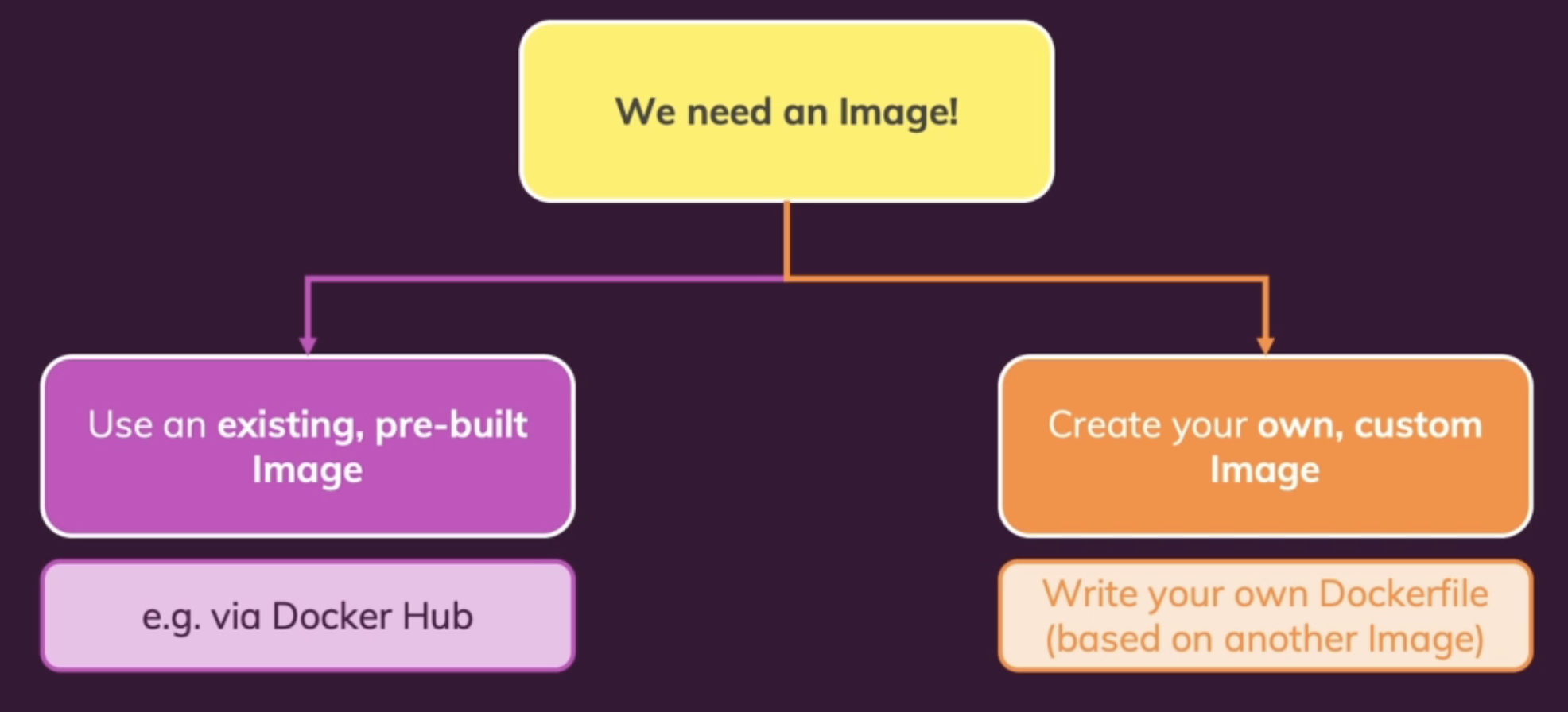


We can use image to create multiple containers based on the image.

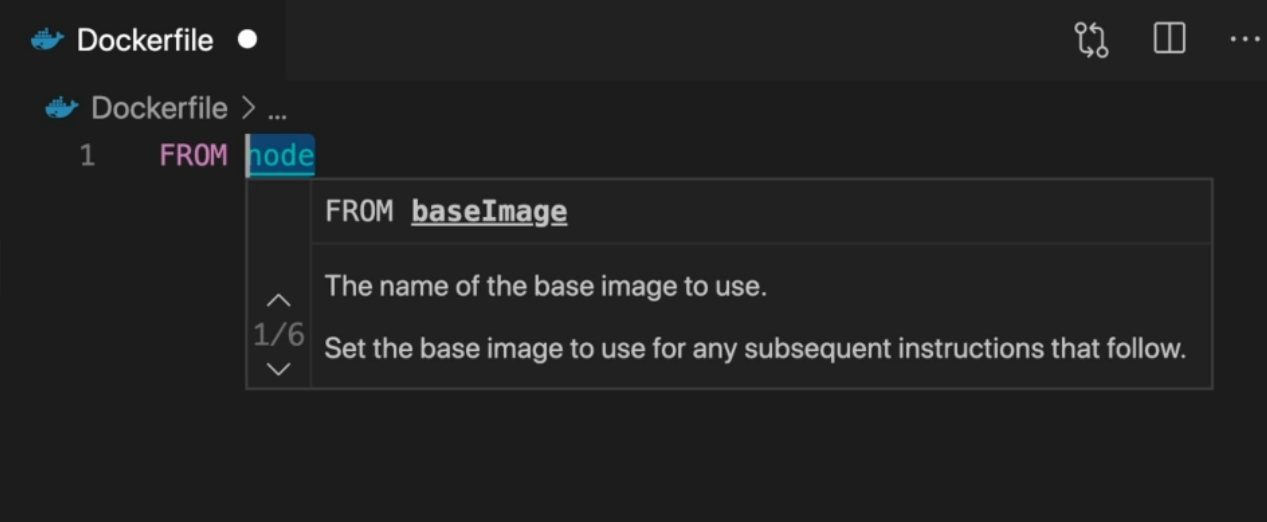


**Using and Running External Pre-Build Images**

Docker Hub is a service provided by Docker for finding and sharing container images.

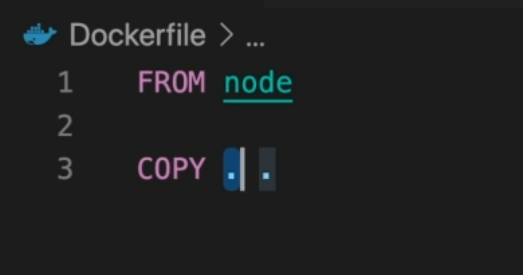


**Dockerfile**



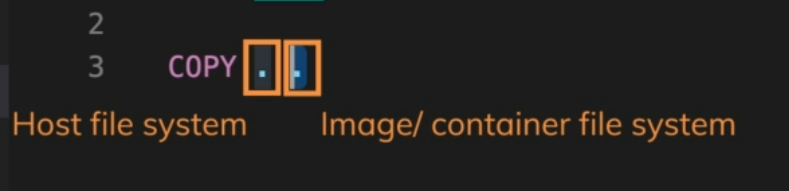
This allows you to build your image up on another base image. And this is what you typically do. Just entering the image name of an image which either exists on your system, or under that name on Docker hub. And this image now exists on Docker hub.

This image was downloaded and cached locally.



And now the second dot is the path inside of the image where those files should be stored. Every image and for also every container created based on an image, has its own internal file system which is totally detached from your file system on your machine. It's hidden away inside of the Docker container. And actually, here it is a good idea to not use the root folder, the root entry in your Docker container but some sub folder which is totally up to you.

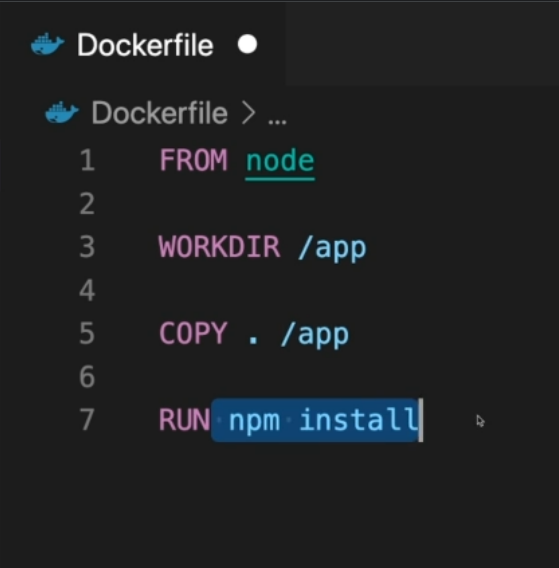
This first dot would tell Docker that all the folders, sub folders and files here in this project should be copied into the Image.



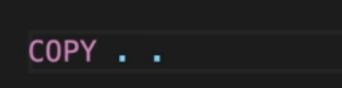
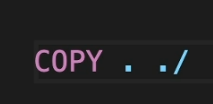
Now all the files here in the same folder as the Dockerfile and all the sub folders there as well, will be copied into an app folder inside of the container. And this folder will simply be created in the image and container if it doesn't exist yet.



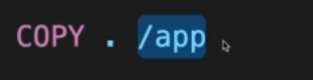
By default, that working directory is the root folder in that container file system. All commands should be executed in that folder, is that you set another instruction here before you copy everything. And that's the workdir instruction.

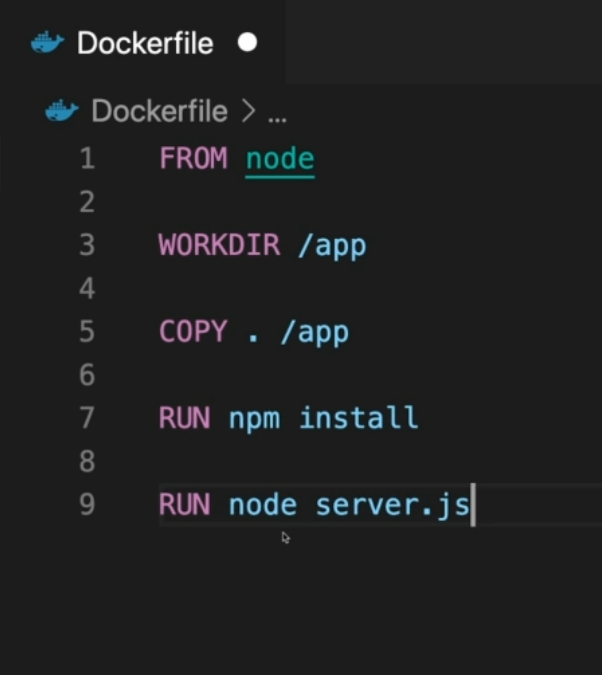


And now as a side node, given the fact that we now did set the working directory to slash app, we could also change copy to copy everything from the path to Docker file is in, to just dot or dot slash which basically means due to current working directory inside of our Docker container.

 or 

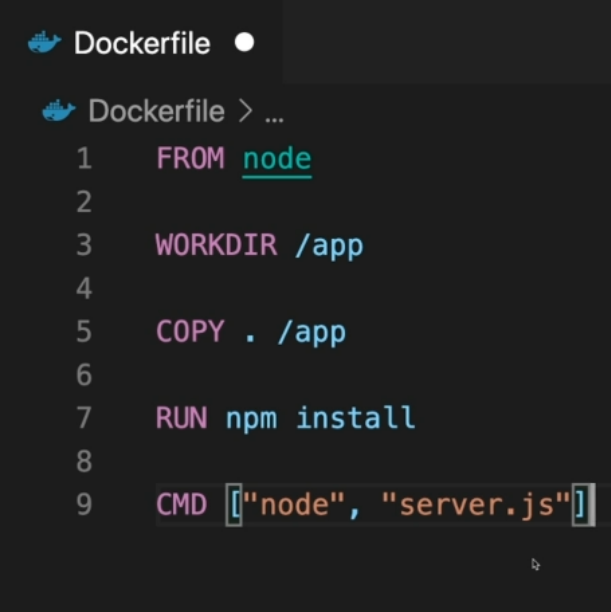
But we can also be more explicit here and set this to the absolute slash app path like this.





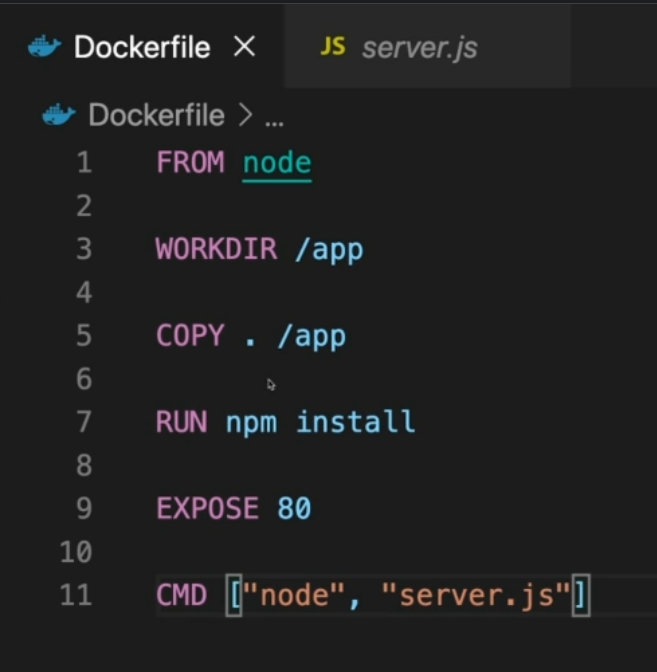
All these here are instructions to Docker for setting up the image. Now keep in mind the image should be the template. The image is not what you run in the end, you run a container based on an image.

for the container.



The difference to run is that this will now not be executed when the image is created, but when a container is started based on the image.

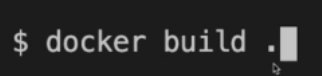
**Note**: A Docker container is isolated. It's isolated from our local environment. And as a result, it also has its own internal network. And when we listen to port 80 in the node application inside of our container, the container does not expose that port to our local machine. So, we won't be able to listen on the port just because something's listening inside of a container.



Therefore, in the Docker file after setting everything up, before specifying the command, which should always be the last instruction in your Dockerfile, we can add to the expose instruction to let Docker know that when this container is started, we want to expose a certain port to your local system. So, to our machine here which will run this container. And then we'll be able to run the container (indistinct) we listen on this port.

**Building Docker Image**

It tells Docker to build a new custom image based on a Dockerfile. if we just type a dot here, we tell Docker that the Dockerfile will be in the same folder

****

**Output::**

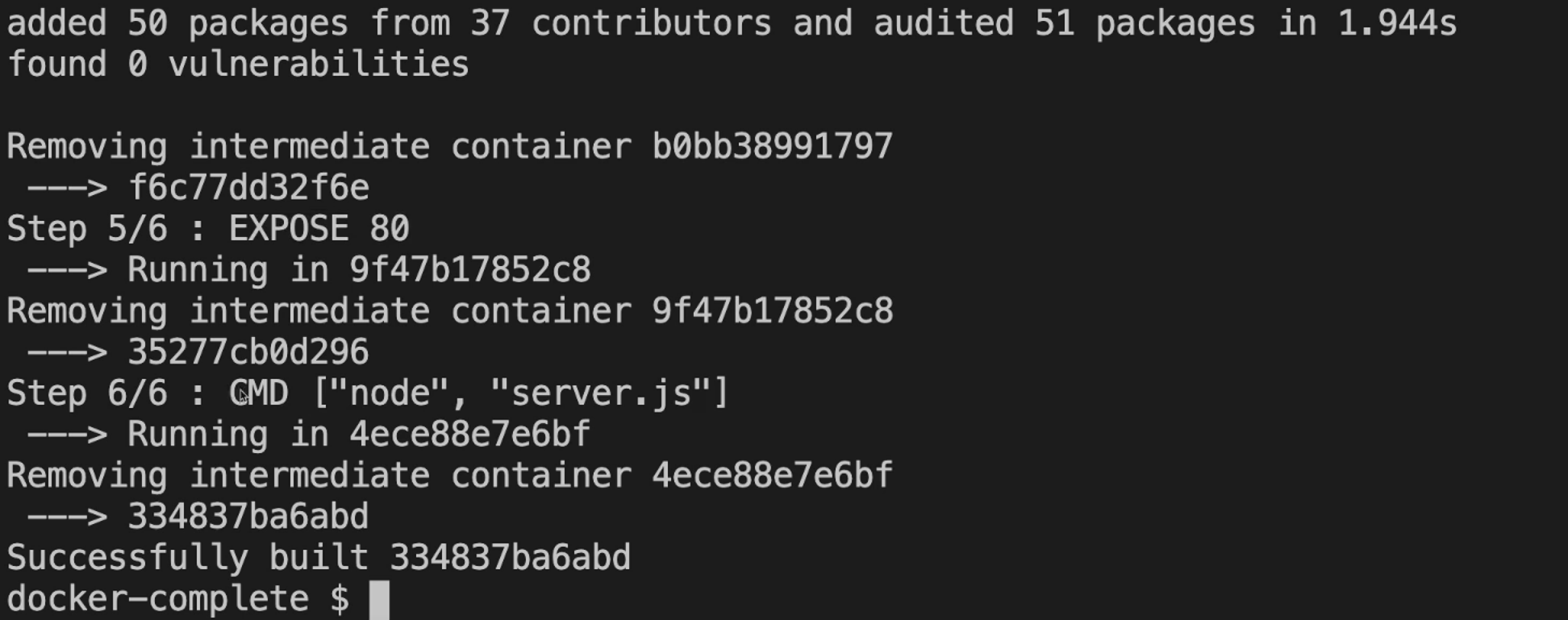
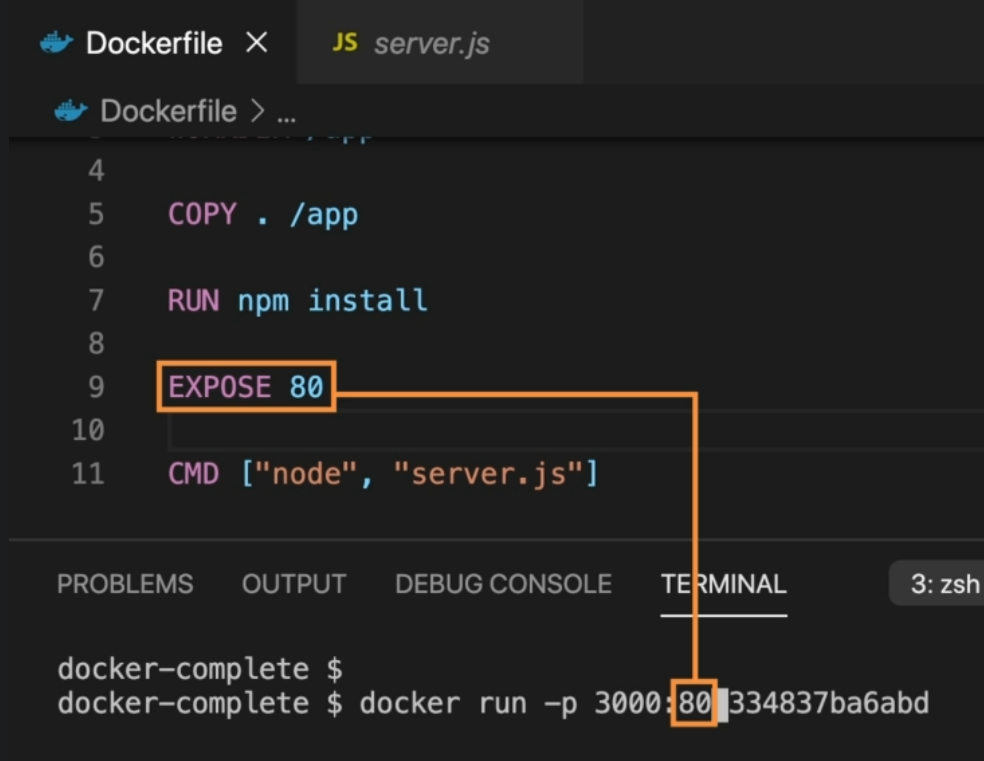
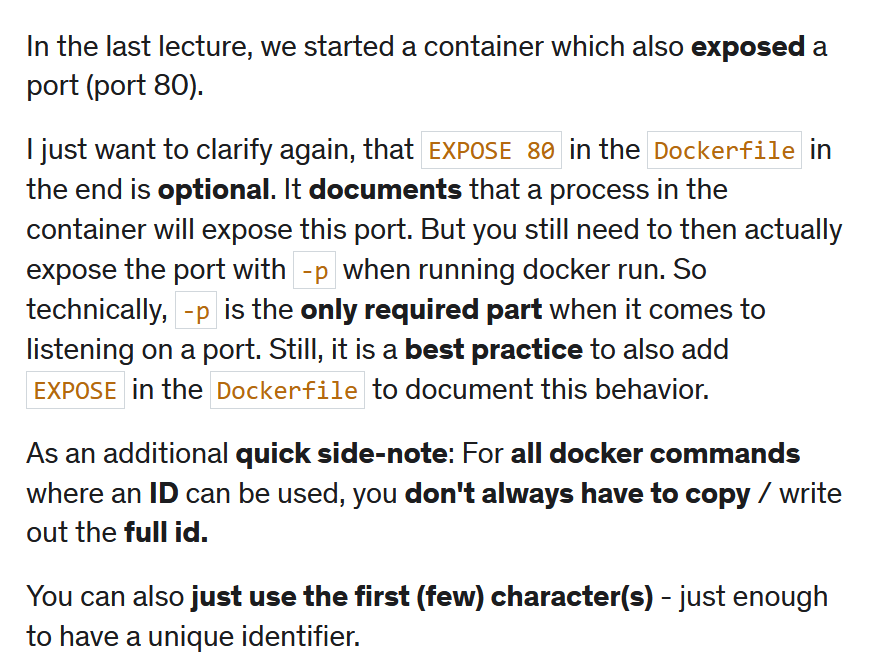
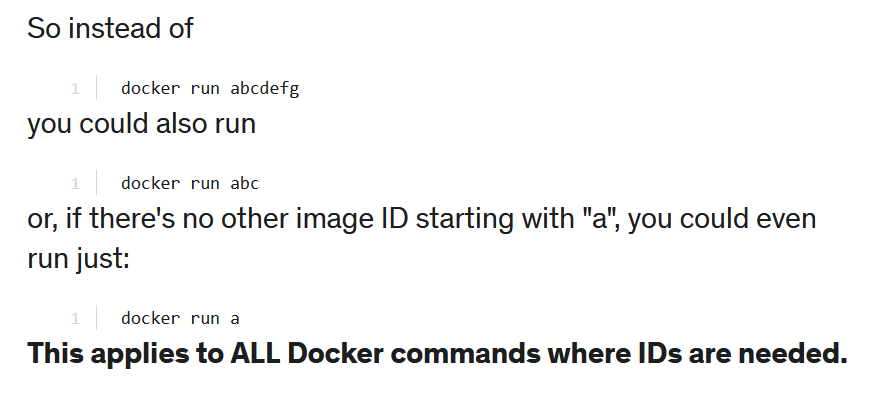
****

Image ID

**Run Docker**

****

****

****

**Note:**

Everything in the images is read only then, and you can't edit it from the outside by simply updating your code just because you copied that code in the past. You need to rebuild to pick up external changes