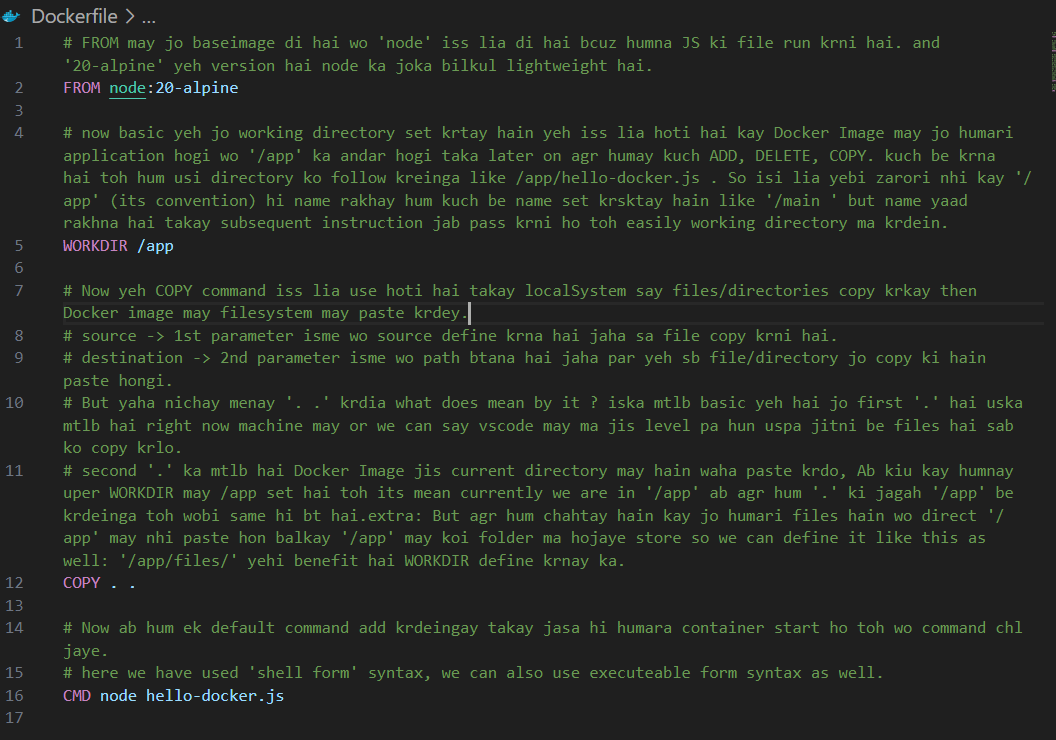
1. Go in your project where you want to create Docker Image.
2. Then, create Docker File inside it. With the name **Dockerfile** (it is necessary) . At that point may be vscode ask for installing Docker extension, you can install it .
3. Go in Docker and write instruction , 1st instruction should define base Image , then further moving forward.

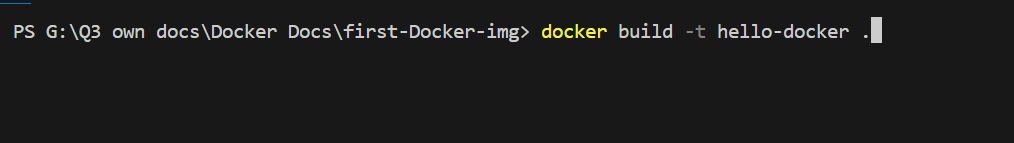


1. Then after creating Dockerfile, Now for creating Docker image we have to create build using Dockerfile.
2. Open terminal, navigate to the path or directory where project is created and then write : **docker build -t nameOfImage writePathWhereDockerFileIsCreated**

Where,

**-t** -> ka mtlb hai **tag** **,** so its optional but agar **-t** flag laga deingay toh humari image ko ek flag assign hojayega.

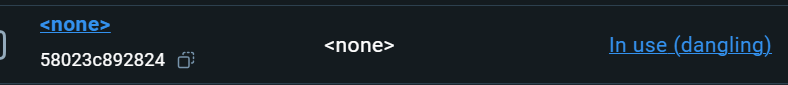
1. Like in our case jaha humari JS ki file hai at the same level DockerFile be hai toh humay path define krnay ki zarorat nhi just “.” Dot likhay gay or uska mtlb hai current dockerFile isi directory may same level pa hai.



1. Now in this way we have created the Image successfully now how to run this image and create container.
2. So simply cmd ma jakay we will write **docker run -it ImageName**

Remember one thing agar image apnay create ki hai or pehla be usi same name say image hai toh Docker jo pehlay wali image hai uski overwrite krdega or uska name and tag **None** hojyaga. But agr ap chahtay hain kay same name say hi image create krni hai so then **-t** ka through dono ko different tag dedein toh by the help of tag wo differentiate krna possible hoga.

Now the SS is below when you create another image with same name and tag as it exist before also:



Jo previous image hogi uska Name and tag dono **None** hojayegay or status mabi **dangling** likha ajyega .

When you create a new Docker image with the same name as an existing one, the previous image might end up with `<none>` as its name and tag, and its status might be listed as "In use (dangling)." Here's what's happening:

**Why the Previous Image Name Becomes `<none>`:**

1. \*\*Image Overwriting\*\*:

- When you build a Docker image with the same name and tag as an existing image (e.g., `myimage:latest`), Docker will overwrite the existing image with the new one.

- The old image is not deleted immediately; instead, it loses its name and tag because those are now associated with the new image. As a result, the previous image is left without a name or tag and is marked as `<none>`.

2. \*\*Dangling Image\*\*:

- An image without a name or tag is called a "dangling" image. These are images that are not associated with any tags, meaning they're not referenced by any name and are essentially orphaned.

\*\*What Does "In use (dangling)" Mean?\*\*

1. \*\*Dangling\*\*:

- "Dangling" means the image is not tagged and isn’t associated with any named reference. It's not linked to any tags, making it difficult to reference or reuse.

2. \*\*In Use\*\*:

- "In use" indicates that this image is still being used by some containers. Even though it's dangling, it might be the base for running containers, so Docker is keeping it around until those containers are stopped or removed.

If you no longer need the old, dangling image, you can remove it using Docker commands like `docker image prune` to clean up unused images.

**Use and Purpose of the -t Flag:**

1. **Tagging the Image**:
   * The -t flag allows you to specify a name and optionally a tag (version) for the Docker image you are building.
   * The format for the -t flag is usually name:tag. If you don't specify a tag, Docker assigns latest as the default tag.

Example:

**docker build -t myapp:1.0 .**

* + In this example, myapp is the name of the image, and 1.0 is the tag (version) of that image.

1. **Multiple Tags**:
   * You can use the -t flag multiple times in the same docker build command to assign multiple tags to the same image.

Example:

**docker build -t myapp:1.0 -t myapp:latest .**

* + Here, the same image is tagged both as myapp:1.0 and myapp:latest

--------------------------------------------------------------------------------------------------------------

Now if we want kay jab hum container run krein toh direct CMD wali command na execute ho blka humara pass interactive mode may ek shell open hojaye and waha hum easily jis be file ko execute krwana hai hum krwa sktay hain.But remember jo shell open hoga wo WORKDIR may hoga like in our case it is **app**.

Shell run krnay kay liya: **docker run -it imageName sh** , jo last ma ‘sh’ it will open shell.

below you can see kay basic humara host OS linux hai bcuz jo Docker Desktop hai wo WSL 2 ka through jo VM create hui hai with linux OS toh Docker desktop usko as a host OS use kr rha hai iss lia we have to run linux command. Like jab **ls** kia toh sab directories and file agyi then humna jo baseImage may runtime Rakha tha which is Node uskay through **main.js** execute krwa dia.

