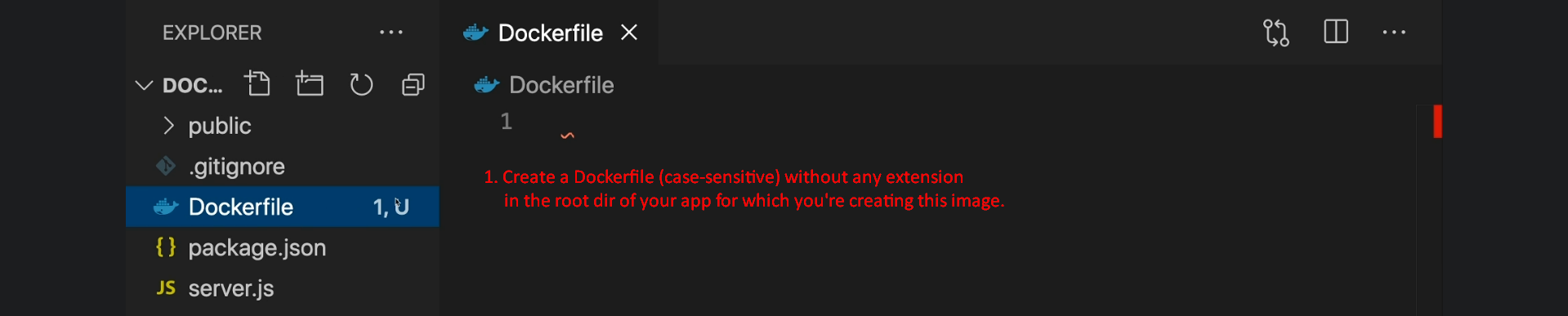
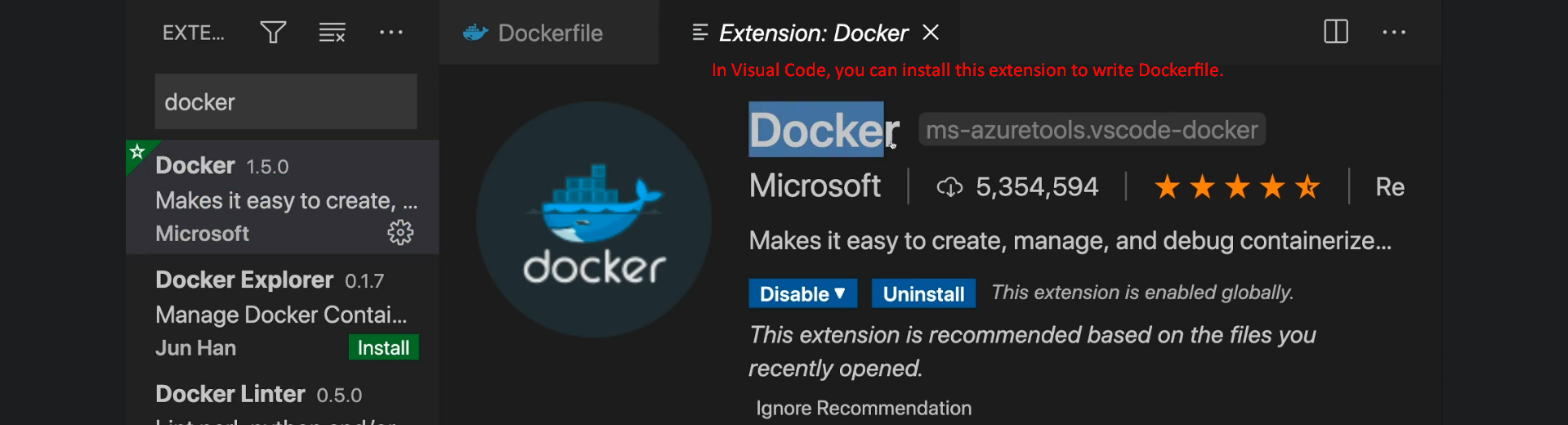
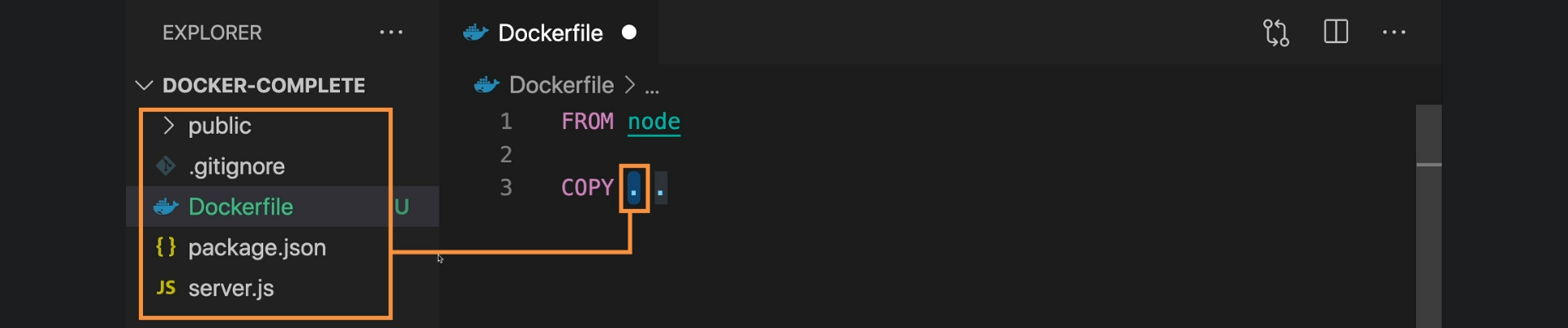
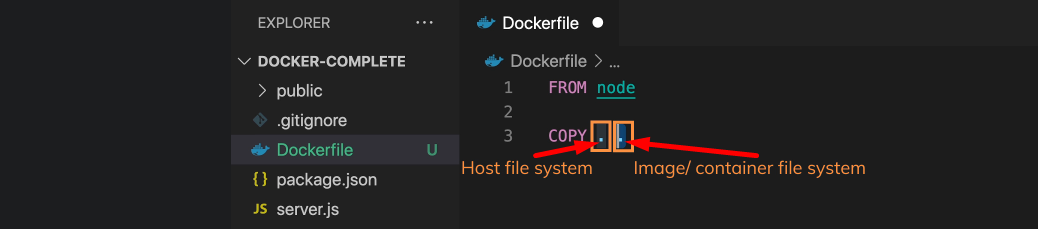
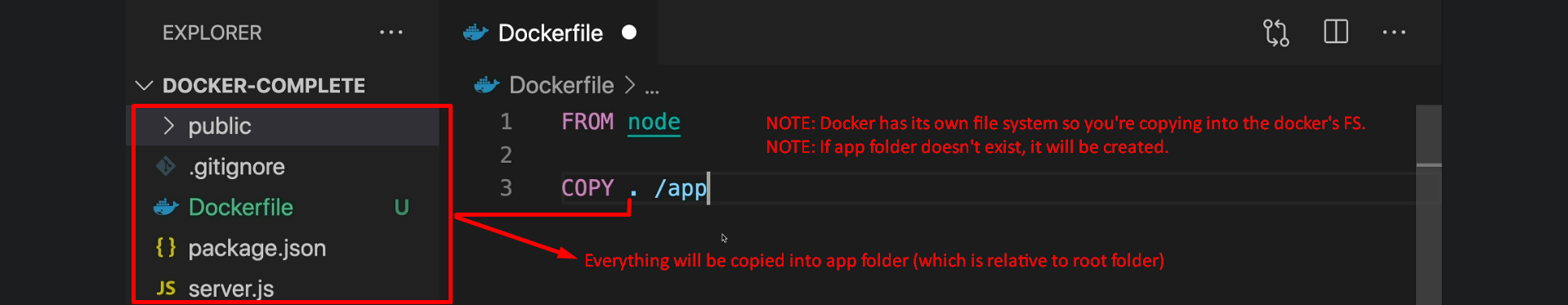
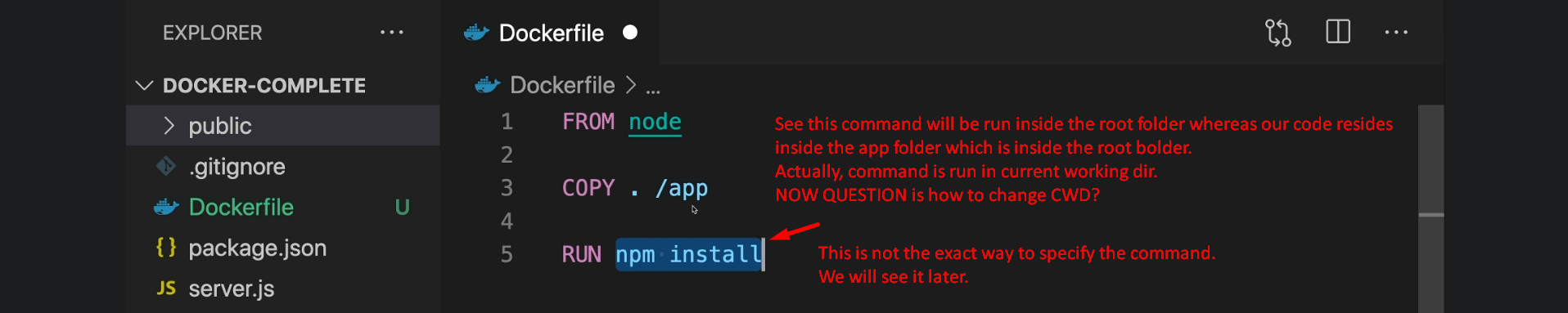
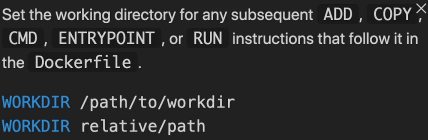
1. 
2. 
3. **Dockerfile**: Contains a set of instructions for Docker to build our own image.
4. **Commands**
   1. **FROM**: To specify the base image.  
      Maybe this image contains some OS layer only or also some other tools/dependencies/installed S/W like node.  
      Theoretically, we can build our own image from the scratch but of course, in any custom image you want some kind of OS Layer along with some software installed and of course for this, we have image already based on which we make our own custom image. The **base image will be cached** on the local system.
   2. **Copy:** To tell Docker which files/folders (present on our local system) should go into the image.  
         
      **The First Dot**:
      1. Specifies the path on our local system (outside of image/container).
      2. The path will be the folder containing Dockerfile. In our case, it is the folder -> **DOCKER-COMPLETE**.
      3. So, all the files/folders excluding Dockerfile will be copied into the image.

**The Second Dot:**

1. First note that an image/container has its **own internal file system which is hidden** away from us that is totally detached from our **host file system**.
2. The 2nd dot represents the root Dir of the internal file system of the image.
3. **NOTE**: Better not to use root Dir. So, in place of 2nd dot we will use /app wherein we will copy our codes.
   * + 1. Let’s understand the path.
          1. First keep in mind that by default, working directory is the root directory and root directory is the topmost directory (not inside any directory). So, dot represents both working directory and root directory unless we don’t change the working directory with WORKDIR Docker Command.  
             With this /app represents the app directory inside the root directory as it is starting with /.



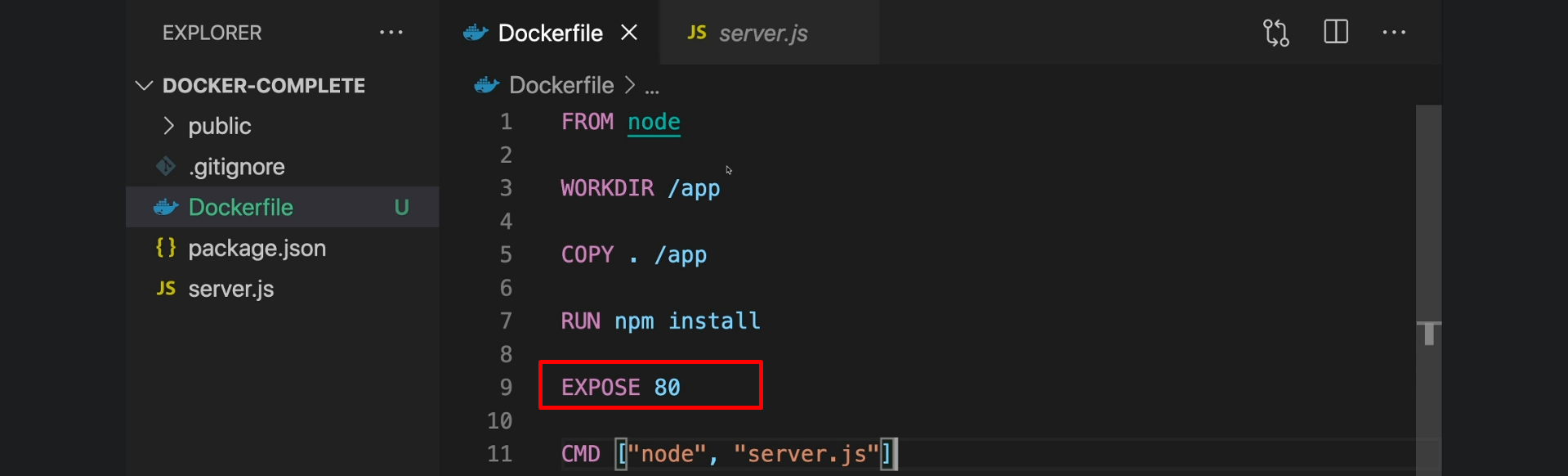
* 1. 
  2. **RUN:** 
     1. To run a command.   
        Such as we want to run “npm install” command in the folder app (in image) as app folder contains package.json.
     2. How to run a command?  
        Answer 🡺 “RUN npm install”.
     3. But the command will be run in Docker file system’s root Dir (as by default, root directory is working directory) but we want to run it in app folder of the Docker’s file system.  
        **Solution**: Use Docker command WORKDIR.
     4. 
  3. **WORKDIR:**
     1. Text

        Description automatically generated
     2. As we did set the working directory to /app, so change the “copy . /app” to “copy . ./” or “copy . .”  
        Even “copy . /app” is fine as /app says starting from the root and then inside app. All the same.   
        The **.**/ is relative app where as /app is absolute path starting from root.
  4. **CMD:**
     1. To setup an initial instruction which will be executed on the container as soon as a container will be created out of an image.  
        Text

        Description automatically generated
     2. Command:
        1.   
           Note that syntax is different as here we’re passing array of strings.
     3. 

|  |  |
| --- | --- |
| **RUN** | **CMD** |
| This instruction is run during image creation. | This instruction is executed on container as soon as it is created. |
| Syntax is different from CMD command. |  |

* 1. **EXPOSE:** To expose a certain port to our local system (To our machine which will run this container).
     1. **Text

        Description automatically generated**
     2. This instruction is to let Docker know that we want to expose this port from our container to the host machine (on which this container is running).  
        

1. Text

   Description automatically generated