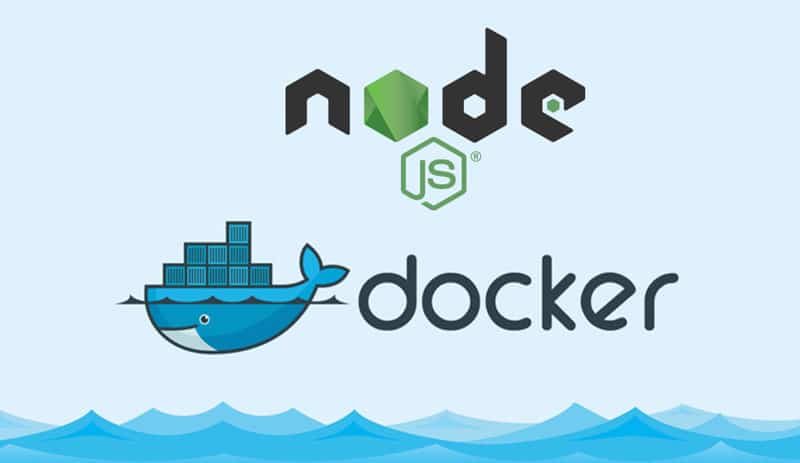
**Building and Deploying a Node.js Application with Docker on Ubuntu**

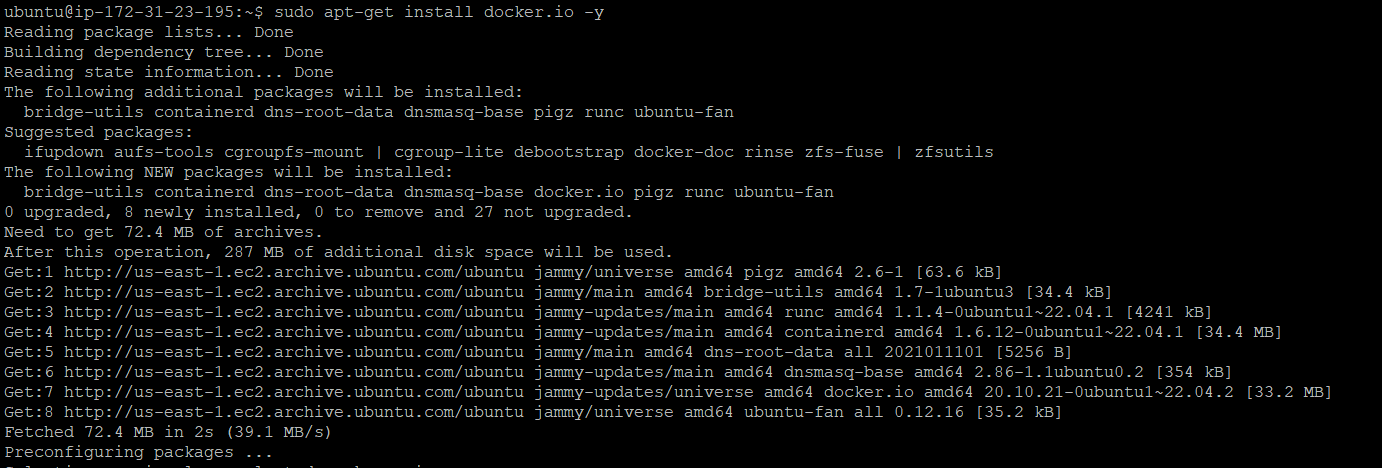


**Step 1 - Installation of docker on Ubuntu**

The first step is to update the package manager. The second step is to install Docker by using the command "sudo apt-get install docker.io -y" on the terminal. After installation, the Docker version can be checked by typing "sudo docker –version" in the terminal.

Command-

1. apt-get update
2. apt-get install docker.io -y
3. docker --version

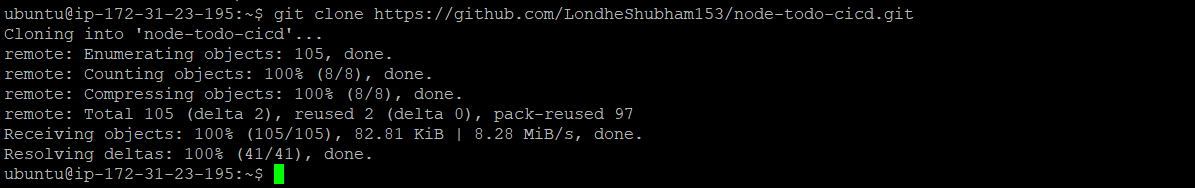




**Step 2 : Clone the code from Github**

The code for the project is cloned from GitHub using the command 'git clone <repository\_url>'. In this case, the repository URL is

<https://github.com/LondheShubham153/node-todo-cicd.git>



(Here I’m cloning the code from Shubham londhe’s account and even you can forks that Repositorie to your github account).

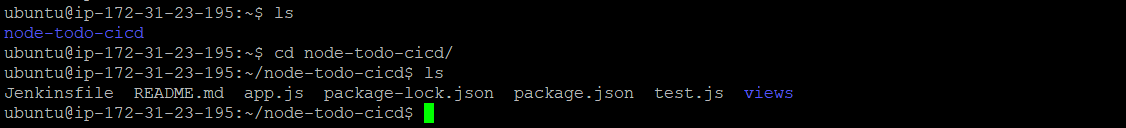
**Step 3: Check the files and understand the requirements**

Check the files and understand the requirements Once the code has been cloned, we navigate to the directory using the command 'cd react\_django\_demo1\_app' and use the 'ls' command to list the files in the directory. This helps us to understand the requirements of the project.

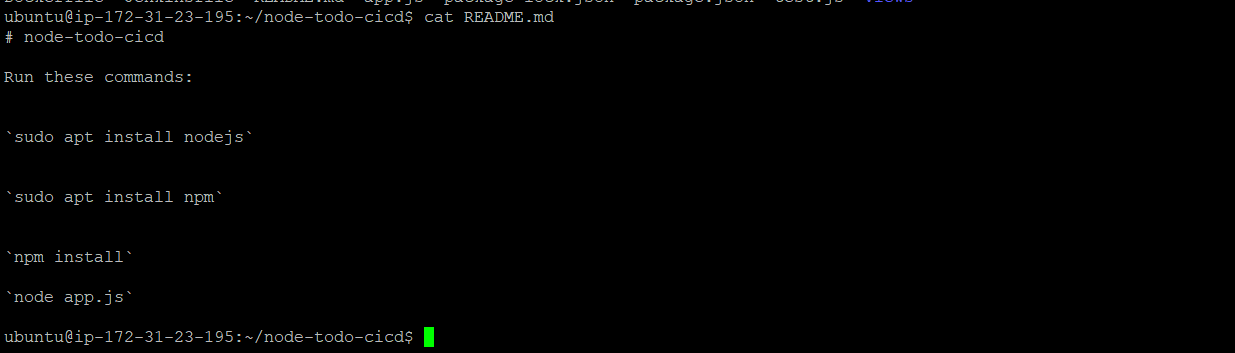
Commands

- Cd react\_django\_demo1\_app

- ls



In the README.md file, the user can find the requirements for the project. After understanding 'README.md' file the requirements for the project are documented, including the need for Node.js, Java, and npm.

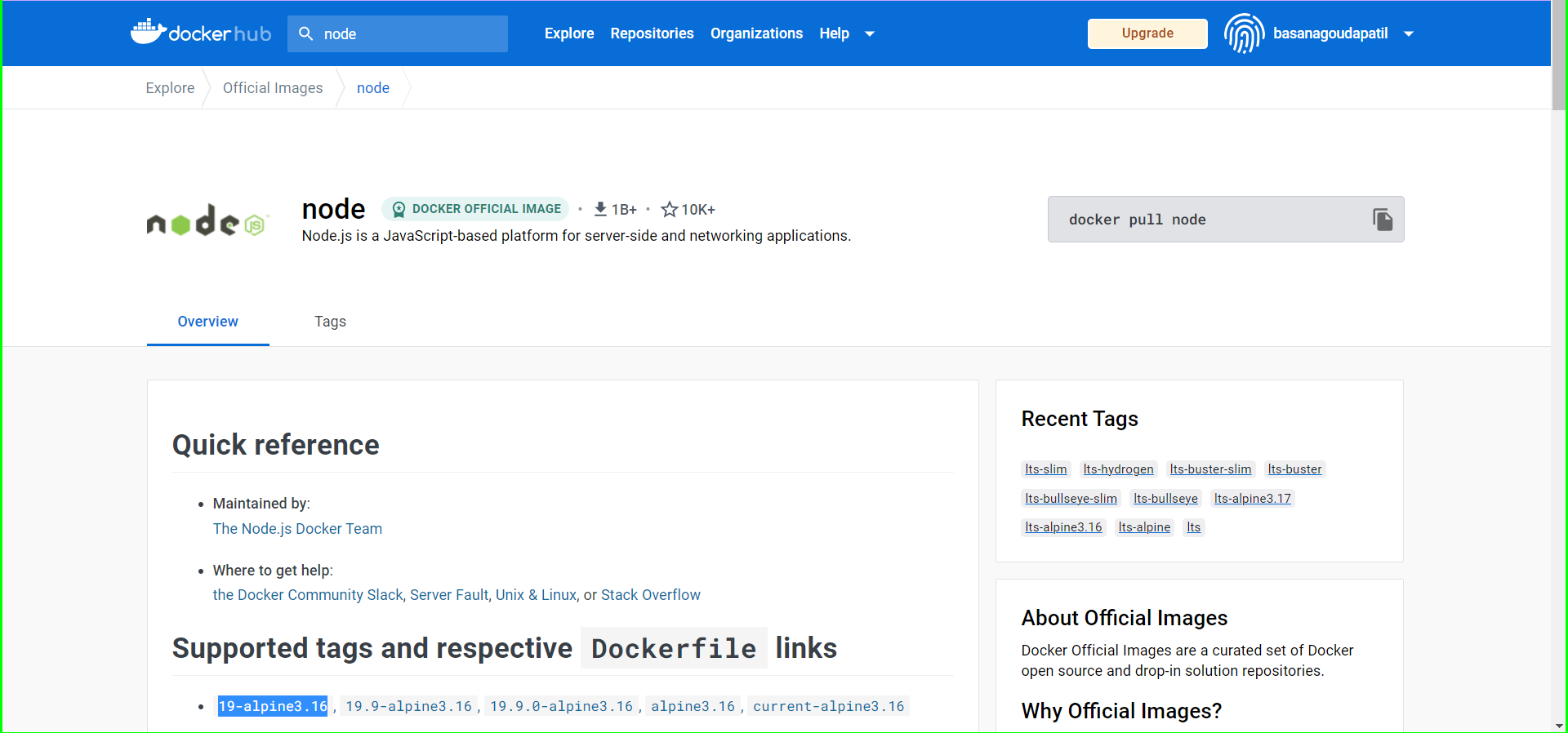


**Step 4: Create a docker image using a dockerfile**

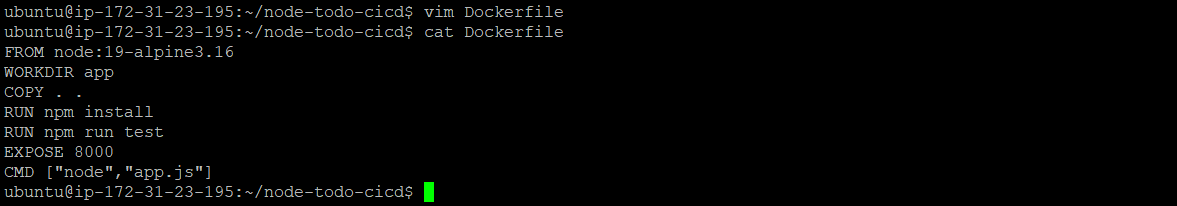
To Create a Docker image using a Dockerfile A Dockerfile is used to define the configuration of the Docker image. In this step, a Dockerfile is created to build the Docker image for the project.

**I suggested to use base image the node image version 19-alpine3.16 from the Docker Hub.**

**The Alpine Linux distribution is a lightweight Linux distribution that is commonly used in Docker images because of its small size. The 19-alpine3.16 tag refers to a specific version of the image.**



**Create a Dockerfile to define the environment for the application. The Dockerfile specifies the base image to use, the packages to install, and any other configuration required for the application.**

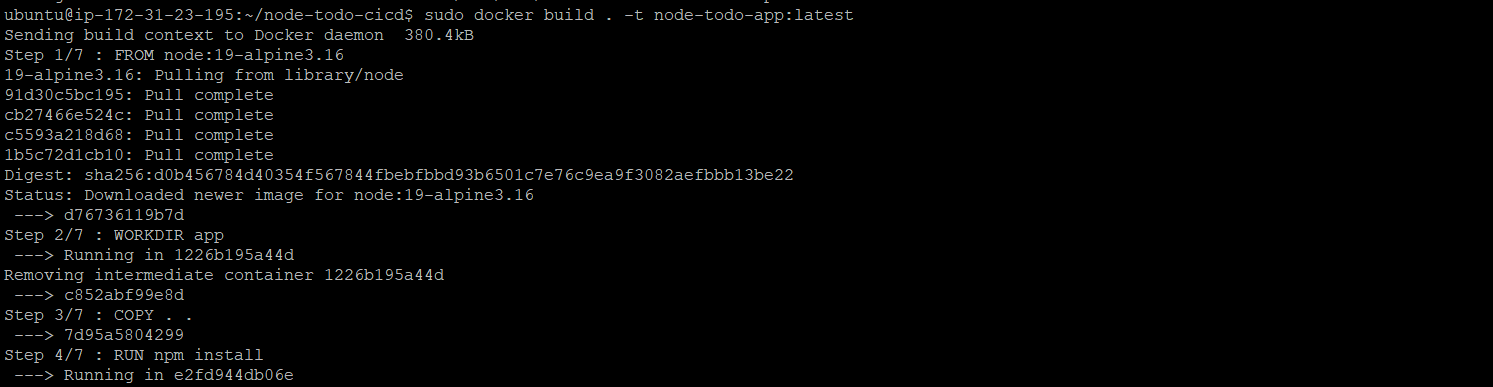


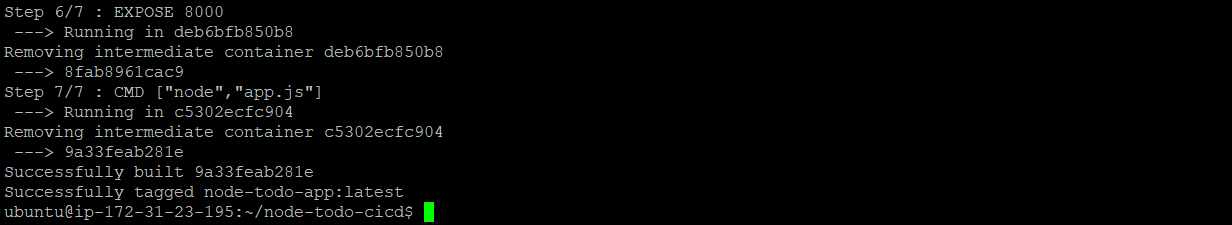
**Step 4: Build the docker image**

Build the Docker image Once the Dockerfile has been created, the Docker image is built using the 'docker build' command. The '-t' flag is used to specify the name and tag of the Docker image.

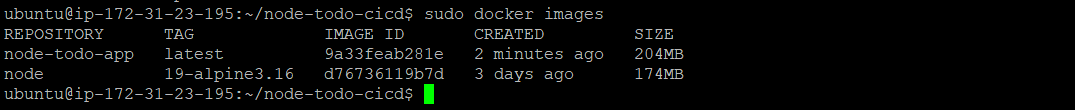
Command

docker build . -t node-todo-app:latest





To check the images, use the command "docker images".

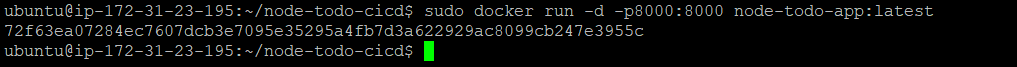


**Step 5: Create a docker container**

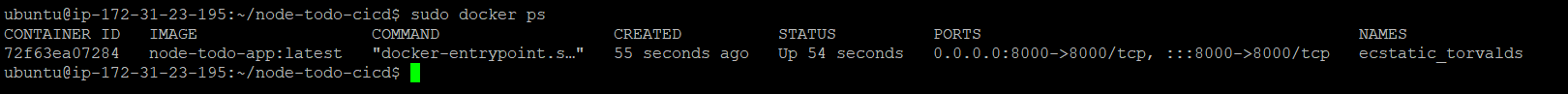
Create a Docker container A Docker container is created using the Docker image that was built in the previous step. The '-d' flag is used to run the container in detached mode, and the '-p' flag is used to map port 8000 of the container to port 8000 of the host machine.

Commands

docker run -d -p8000:8000 node-todo-app:latest

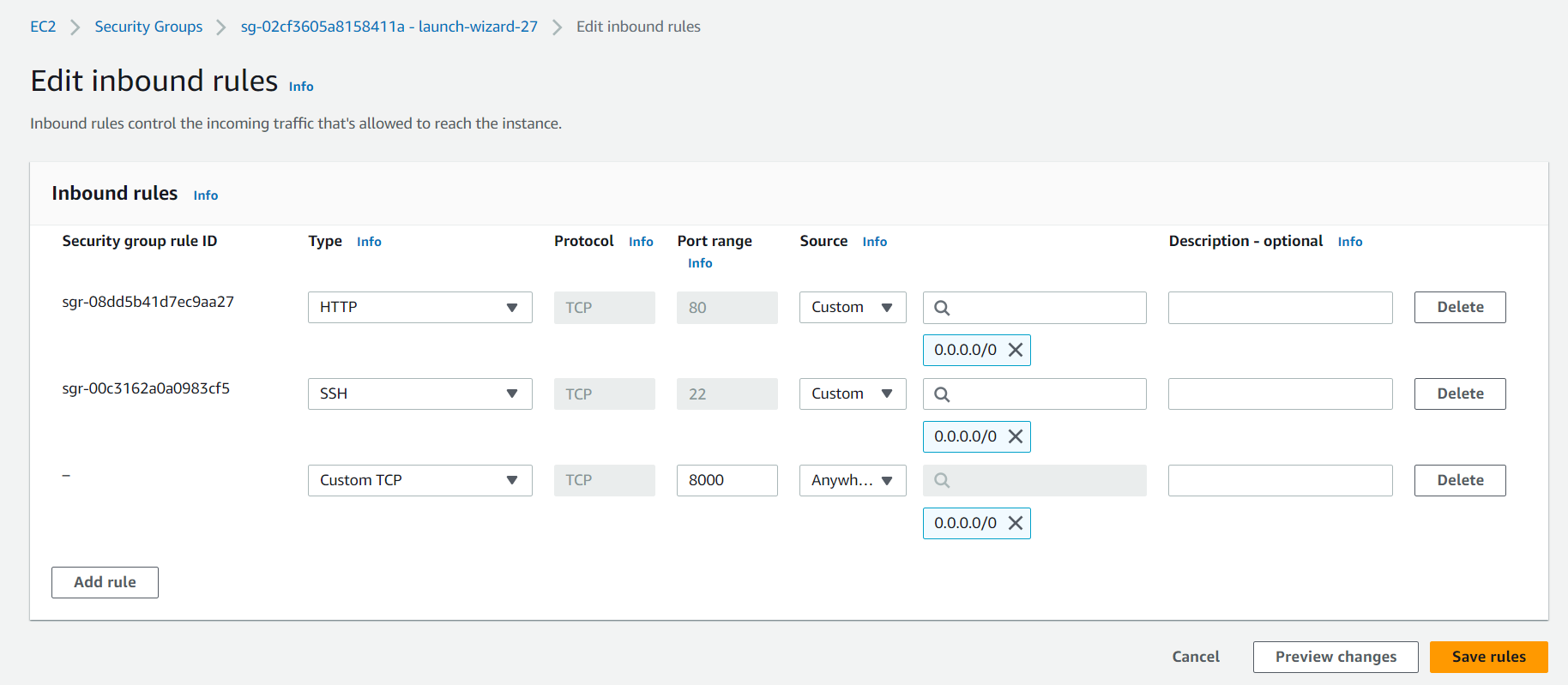


The user can check the running containers by using the command "sudo docker ps".



**Step 6: Open port 8000 in the instance’s security inbound rule**

Open port 8000 in the instance’s security inbound rule To allow traffic to access the application, port 8000 needs to be opened in the instance’s security inbound rule.



**Step 7: Check if the application is running**

Check if the application is running Finally, we can check if the application is running by opening a browser and typing the URL 'http://<host\_ip\_address>:8000' in a web browser.

s