

Assessment

I have done the assessment using AWS ill be putting the screenshots below:-

The screenshot shows the AWS CloudShell interface. On the left, there is a sidebar with various AWS services: EC2 Dashboard, EC2 Global View, Events, Console-to-Code (selected), Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (selected), AMIs, and AMI Catalog. The main area displays the 'Instances (1/1) Info' page for one instance named 'docker-node-a...'. The instance is in a 'Running' state (t2.micro, 2/2 checks passed). Below this, the 'Instance: i-062081eccd3f6338c (docker-node-app)' details are shown, including its public IPv4 address (54.91.233.143) and private IP (172.31.25.5). The bottom of the screen shows the Windows taskbar with icons for File Explorer, Mail, and other applications.

I made two file that is index.js and package.json and pushed it to my github

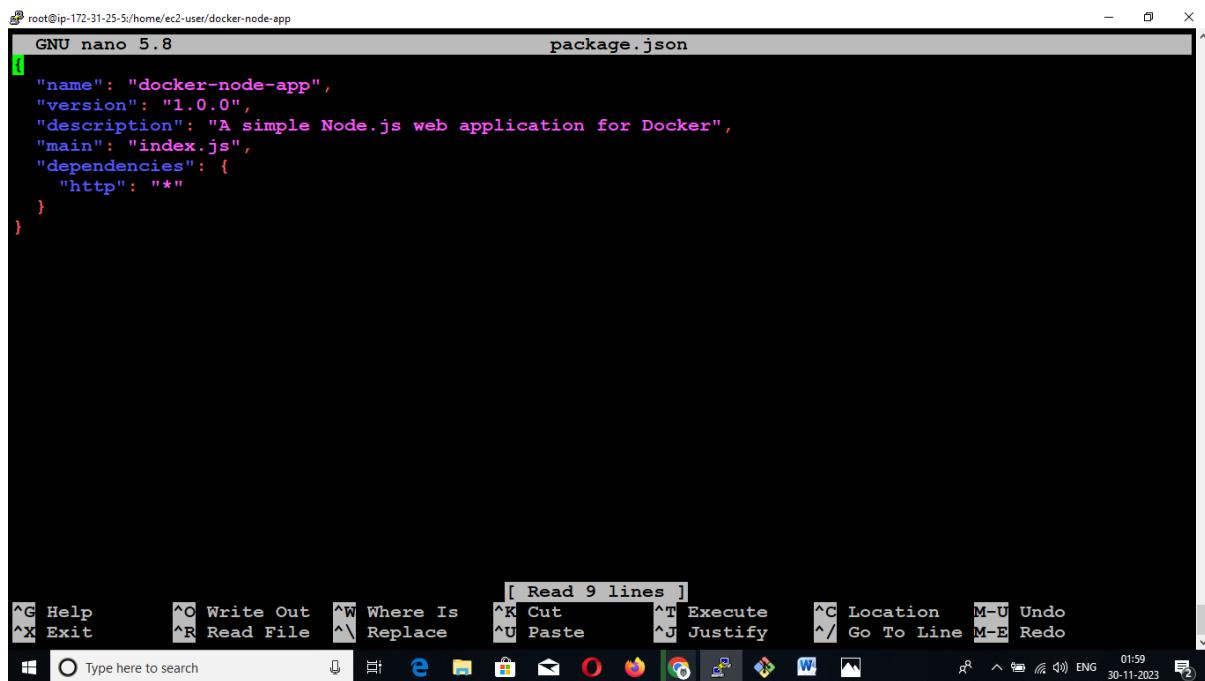
index.js

```
root@ip-172-31-25-5:/home/ec2-user/docker-node-app
GNU nano 5.8                               index.js
const http = require('http');
const port = 3000;
const hostname = '0.0.0.0'; // Set the hostname to listen on all available network interfaces

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, Docker!\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Package.json:-



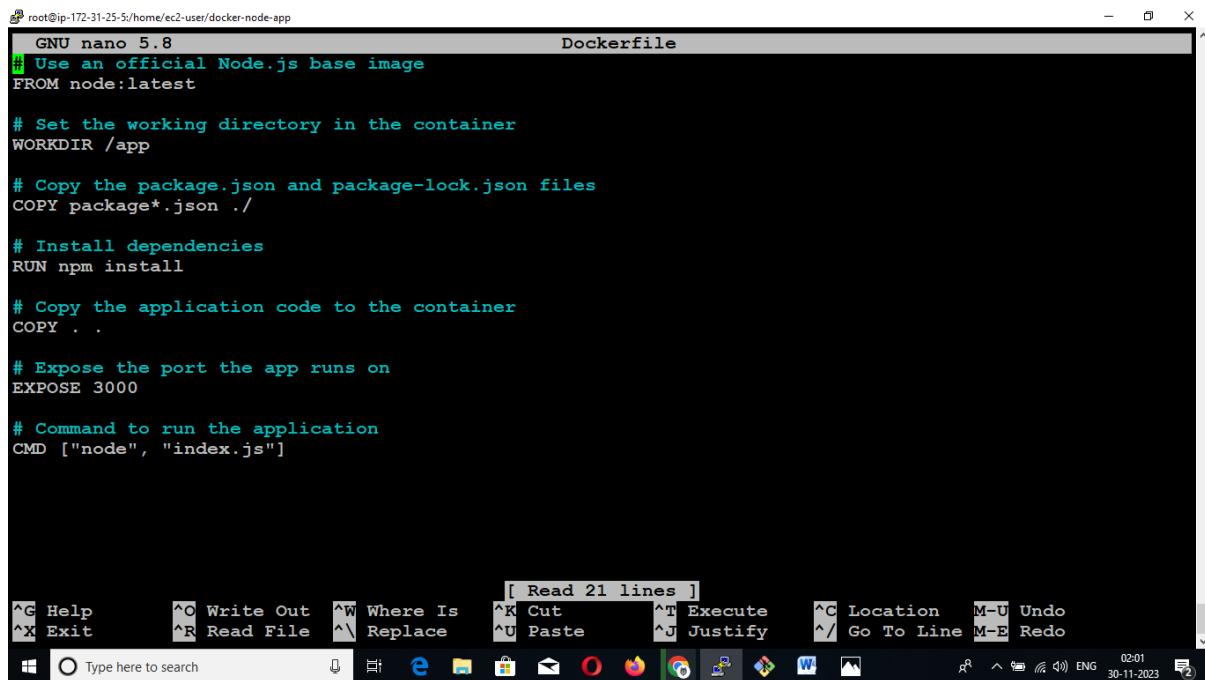
```
root@ip-172-31-25-5:/home/ec2-user/docker-node-app
GNU nano 5.8                               package.json

{
  "name": "docker-node-app",
  "version": "1.0.0",
  "description": "A simple Node.js web application for Docker",
  "main": "index.js",
  "dependencies": {
    "http": "*"
  }
}
```

The screenshot shows a terminal window titled "package.json" containing the JSON configuration file. The file defines a project named "docker-node-app" with version "1.0.0". It includes a description and sets "index.js" as the main entry point. The "dependencies" section is present but empty. The terminal is running on a Linux system with a standard desktop environment at the bottom.

then I clowned it on my aws instance as you can see in the above screenshots

after clowning In the docker-node-app directory, I created a file named Dockerfile (without any file extension) using a text editor and add the following content



```
root@ip-172-31-25-5:/home/ec2-user/docker-node-app
GNU nano 5.8                               Dockerfile

# Use an official Node.js base image
FROM node:latest

# Set the working directory in the container
WORKDIR /app

# Copy the package.json and package-lock.json files
COPY package*.json .

# Install dependencies
RUN npm install

# Copy the application code to the container
COPY .

# Expose the port the app runs on
EXPOSE 3000

# Command to run the application
CMD ["node", "index.js"]
```

The screenshot shows a terminal window titled "Dockerfile" containing the Dockerfile. It specifies an official Node.js base image and sets the working directory to "/app". It copies the package files and installs dependencies. It then copies the application code, exposes port 3000, and runs the application with "node index.js". The terminal is running on a Linux system with a standard desktop environment at the bottom.

Build the Docker image using the following command:
docker build -t my-node-app .

Now to make user running the docker build command has the necessary permissions:
sudo docker build -t my-node-app-image .

confirm the successful creation of the Docker image by running the following command:
sudo docker images

To run a Docker container from the newly created image, use the docker run command:

sudo docker run -d -p 3000:3000 my-node-app-image

After running the container, your Node.js application should be accessible at
<http://54.91.233.143:3000>

the output I got at the last as successful running of the container and access to node.js application :-

