

## A) Create a Instance



```
apt install nginx -y
```

```
apt install nodejs -y
```

→ Install NPM (node package manager)

**apt install npm -y**

```
root@ip-172-31-33-75:~# npm --version
8.5.1
root@ip-172-31-33-75:~#
```

→ Install Pm2 (process manager)

**npm install -g pm2**

```
[PM2] Starting the daemon pm2
[PM2] PM2 Successfully daemonized
5.4.3
```

### c) Creating a Nodejs Application

---

→ First, using `nano` or your favorite text editor, create a sample application called `hello.js` inside the home directory

→ Go to the home directory

**cd /home**

→ open a `hello.js` file and paste the code

**nano hello.js**

**const http = require('http');**

**const hostname = '0.0.0.0';**

**const port = 3000;**

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

**server.listen(port, hostname, () => {**

```
console.log(`Server running at http://${hostname}:${port}/`);
});
```

```
const http = require('http');

const hostname = 'localhost';
const port = 3000;

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

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

Save the file and exit from the editor

Ctrl + o = saving

Ctrl + x = exit

→ This Node.js application listens on the specified address (**0.0.0.0**) and **port (3000)**, and returns “**Hello World!**” with a **200 HTTP** success code. Since we’re listening on localhost, remote clients won’t be able to connect to our application.

To test your application, type:

**node hello.js**

```
root@ip-172-31-33-75:/home# node hello.js
Server running at http://localhost:3000/
```

→ Going to start node application in pm2

**pm2 start hello.js --name app**

```
root@ip-172-31-33-75:/home# pm2 start hello.js --name app
[PM2] Starting /home/hello.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	🔄	status	cpu	memory
0	app	fork	0	online	0%	11.5mb

→ Setting up node application in nginx as a reverse proxy

To let Your application is **running and listening on localhost**, but you need to set up a way for your users to access it. We will set up the **Nginx web server as a reverse proxy** for this purpose.

In the prerequisite tutorial, you set up your **Nginx** configuration in the **/etc/nginx/sites-available/example.com** file. Open this file for editing

```
server {
    listen 80;
    server_name 13.208.246.123;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

```
server {
    listen 80;
    server_name 13.208.246.123;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
```

```
proxy_set_header Connection 'upgrade';
proxy_set_header Host $host;
proxy_cache_bypass $http_upgrade;
}
```

```
}
```

(replace with your server ip)

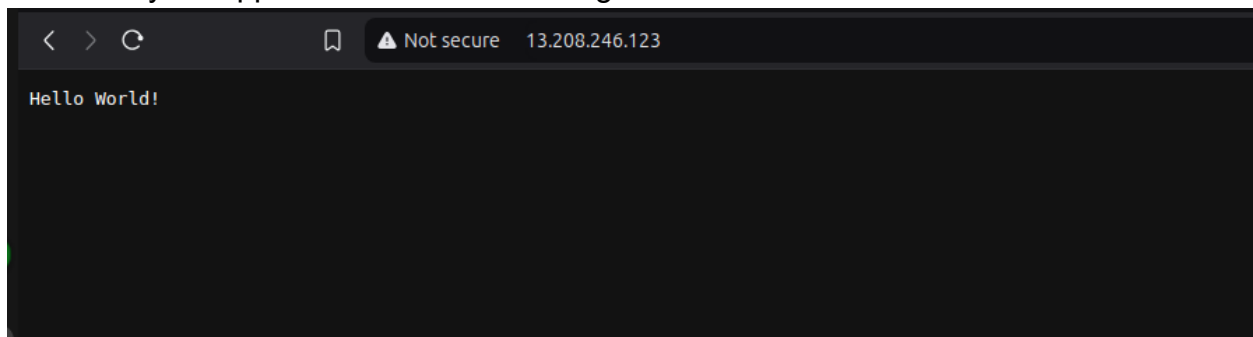
→ Create a symlink for this example.com

**In -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/**

→ Restart your nginx webserver

**systemctl restart nginx**

→ Check your application in browser using IP



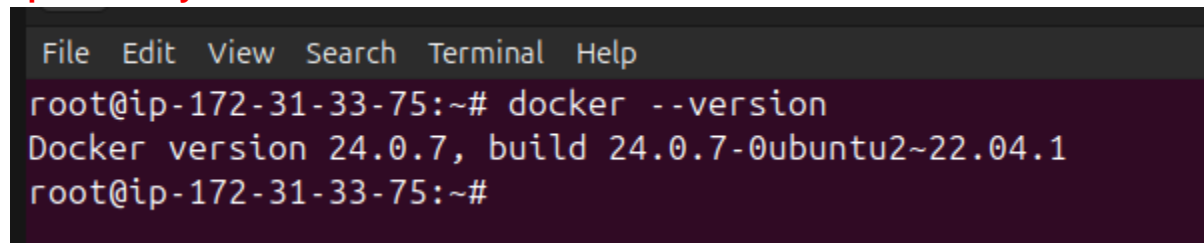
Nodejs application is Running on Ec2 instance

## D) Setting up in Docker

-----

→ Install docker

**apt install -y docker.io**



→ Install Docker Compose

**apt install -y docker-compose**

```
root@ip-172-31-33-75:~# docker-compose --version
docker-compose version 1.29.2, build unknown
root@ip-172-31-33-75:~#
```

→ Navigate to the application directory

**cd /home/node**

→ Create a Dockerfile in the /home/node directory

```
## Use Node.js base image
FROM node:12

# Set the working directory inside the container
WORKDIR /app

# Install dependencies
RUN npm install

# Copy the application code
COPY . .

# Expose port 3000
EXPOSE 3000

# Run the application
CMD ["node", "hello.js"]
```

```
# Use Node.js base image
FROM node:12
```

```
# Set the working directory inside the container
WORKDIR /app
```

```
# Install dependencies
RUN npm install
```

```
# Copy the application code
COPY . .
```

```
# Expose port 3000
EXPOSE 3000
```

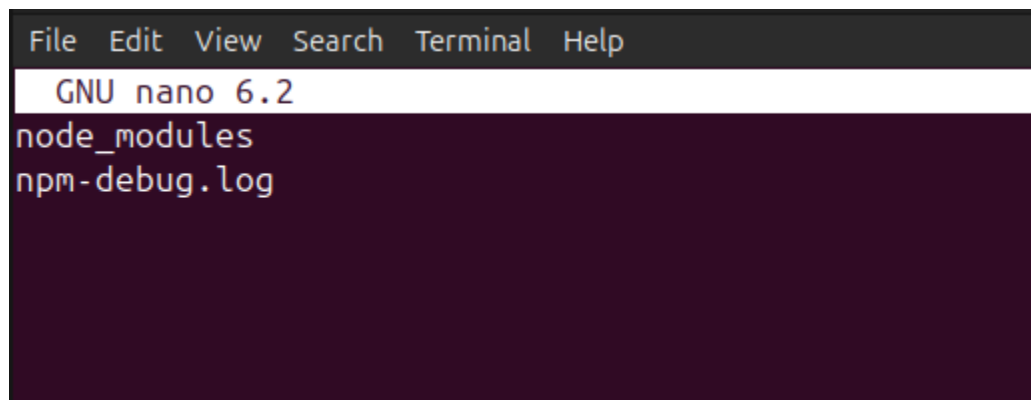
```
# Run the application
CMD ["node", "hello.js"]
```

Save it and exit from editor

→ Create a .dockerignore File

Create a .dockerignore file to exclude unnecessary files from the image

```
node_modules
npm-debug.log
```

A screenshot of a terminal window showing the nano text editor. The menu bar at the top includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The title bar indicates 'GNU nano 6.2'. The main text area shows the contents of the .dockerignore file: 'node\_modules' and 'npm-debug.log' on separate lines.

Save it and exit from editor

→ Build the docker image

## **docker build -t arunhub01/node-app:latest .**

(Here arunhub01 is my dockerhub username. Replace with your dockerhub username)

```
root@ip-172-31-33-75:/home/node# docker build -t arunhub01/node-app:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
             Install the buildx component to build images with BuildKit:
             https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  4.096kB
Step 1/7 : FROM node:12
12: Pulling from library/node
f5196cdf2518: Pull complete
9bed1e86f01e: Pull complete
f44e4bdb3a6c: Pull complete
2f75d131f406: Pull complete
07dff4ad21eb: Pull complete
e0ac4f13b766: Pull complete
df2c3b2eb7cc: Pull complete
efe636eac583: Pull complete
fe17849545bb: Pull complete
Digest: sha256:01627afeb110b3054ba4a1405541ca095c8bfca1cb6f2be9479c767a2711879e
Status: Downloaded newer image for node:12
--> 6c8de432fc7f
Step 2/7 : WORKDIR /app
--> Running in 86ae60c28fde
Removing intermediate container 86ae60c28fde
--> 16d9690918a5
Step 3/7 : COPY package*.json ./
COPY failed: no source files were specified
```

Docker build the image is completed

→ Verify the image is build or not

## **docker images**

```
Successfully built 321bd5bc4878
Successfully tagged arunhub01/node-app:latest
root@ip-172-31-33-75:/home/node# docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
arunhub01/node-app  latest         321bd5bc4878   6 seconds ago  918MB
node                12            6c8de432fc7f   2 years ago    918MB
root@ip-172-31-33-75:/home/node#
```

## **E) Push the image to Docker Hub**

→ Log in to Docker Hub

## **docker login**



```
root@ip-172-31-33-75:/home/node# docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, go to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security. For more information, see https://docs.docker.com/go/access-tokens/
required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: arunhub01
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-33-75:/home/node#
```

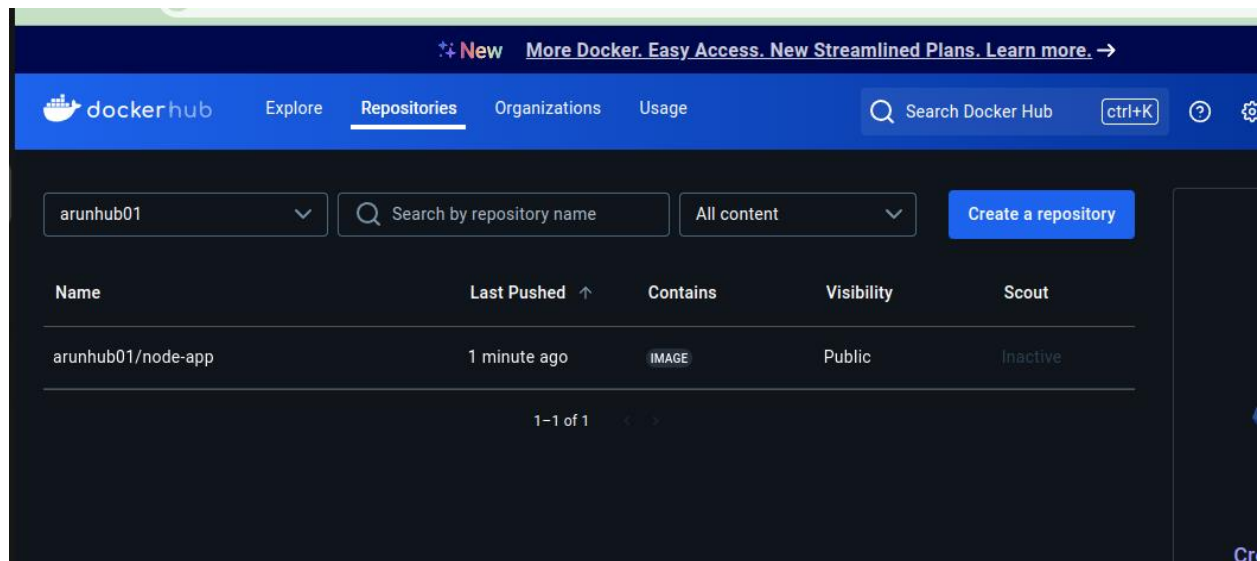
When you use docker login command, you were asked by username and password. Please enter correct username and password of your dockerhub.

→ Push the image to docker hub

### **docker push arunhub01/node-app:latest**

```
root@ip-172-31-33-75:/home/node# docker push arunhub01/node-app:latest
The push refers to repository [docker.io/arunhub01/node-app]
bf5b153cc95f: Pushed
c6d9246fb892: Pushed
47df0936ac6a: Pushed
586c0b414da7: Mounted from library/node
0bfd290f2c17: Mounted from library/node
6d75cd01c26c: Mounted from library/node
95904c181913: Mounted from library/node
df69bfa94785: Mounted from library/node
f35deb8d96fc: Mounted from library/node
f6c2459e2059: Mounted from library/node
f8323fb3a55c: Mounted from library/node
2f4dc9775f33: Mounted from library/node
latest: digest: sha256:414e7d55f202c3cb4117af826ebf47a7c4844e88cbc2cfa3bdd7134108aef603 size: 2835
root@ip-172-31-33-75:/home/node#
```

→ Go to the Docker hub and check repo is there or not



Completed

---