

# Deployment of Node.js Application Using Docker on AWS EC2

This guide explains the steps to deploy a **Node.js** application using **Docker** on **AWS EC2**.

## Step 1: Create Dockerfile

A **Dockerfile** is a script that contains instructions to build a Docker image. Here's the content for the **Dockerfile**:

A screenshot of a code editor showing a Dockerfile. The editor has a dark background with light-colored text. The Dockerfile content is as follows:

```
Dockerfile > ...
1  FROM node:18
2
3  WORKDIR /app
4
5  COPY . .
6
7  RUN npm install
8
9  EXPOSE 3000
10
11  CMD ["node", "app.js"]
12
```

### Explanation:

- **FROM node:18**: This pulls the official Node.js image of version 18.
  - **WORKDIR /app**: Set **/app** as the working directory inside the container.
  - **COPY . .**: Copies the application code into the container.
  - **RUN npm install**: Installs dependencies defined in **package.json**.
  - **EXPOSE 3000**: Exposes port 3000, the port your app will listen on.
  - **CMD ["node", "app.js"]**: Runs the application using Node.js.
-

## Step 2: Build the Docker Image Locally

Once you have the **Dockerfile** set up, use the following command to build the Docker image:

```
docker buildx build --platform linux/amd64 -t nodeimage .
```

```
bishalranjitkar@Bishals-MacBook-Air node-app % docker buildx build --platform linux/amd64 -t nodeimage .
[+] Building 103.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 348B
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:18@sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf
=> resolve docker.io/library/node:18@sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf
=> sha256:8031108f3cda87bb32f090262d0109c8a0db99168050967becfad502e9a681b 24.06MB / 24.06MB
=> sha256:1d281e50d3e435595c266df06531a7e8c2ebb0c185622c8ab2eed8d760e6576b 64.39MB / 64.39MB
=> sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf 6.41kB / 6.41kB
=> sha256:40d3794e81923d1f62d6c8f4c023701def3c07b615f9de1a0c443c714f107ffb 2.49kB / 2.49kB
=> sha256:512bc7f93b1cd61f26c94e4f2b809fcfa90ef577db27c906c71bf1068c1afa5d 6.39kB / 6.39kB
=> sha256:155ad54a8b2812a0ec559ff82c0c6f0f0dddb337a226b11879f09e15f67b69fc 48.48MB / 48.48MB
=> extracting sha256:155ad54a8b2812a0ec559ff82c0c6f0f0dddb337a226b11879f09e15f67b69fc
```

This will build the image and tag it as **nodeimage**.

### Explanation:

- **docker buildx build**: Uses Docker Buildx to create the image with multi-platform support.
- **--platform linux/amd64**: Specifies the platform (useful for cross-platform compatibility).
- **-t nodeimage**: Tags the image as **nodeimage**.
- **.**: Specifies the current directory (which contains the Dockerfile).

## Step 3: Run the Docker Image Locally

To test the image on your local machine, run the following command:

```
docker run -d -p 3000:3000 nodeimage
```

```
bishalranjitkar@Bishals-MacBook-Air node-app % docker run -d \
> -p 3000:3000 \
> nodeimage
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
19d80e2a7e4b822c41e31293c20533fe236ead871dbdf5b929a9b12d6ca8f287
bishalranjitkar@Bishals-MacBook-Air node-app % docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
19d80e2a7e4b	nodeimage	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:3000->3000/tcp	gifted_robinson

This command starts the container in **detached mode** and maps port **3000** of the container to port **3000** on your local machine.

### Explanation:

- `docker run -d`: Runs the container in detached mode (background).
- `-p 3000:3000`: Maps port 3000 from the container to port 3000 on your local machine.
- `nodeimage`: Specifies the image to run (built in Step 2).

## Step 4: Push the Docker Image to Docker Hub

After building the image, you need to push it to **Docker Hub** so it can be accessed from EC2.

### Login to Docker Hub:

```
docker login
```

### Tag the Image:

```
docker tag nodeimage bishalranjit0606/mynodeapp
```

### Push the Image:

```
docker push bishalranjit0606/mynodeapp
```

```
bishalranjitkar@Bishals-MacBook-Air node-app % docker login
Authenticating with existing credentials...
Login Succeeded
bishalranjitkar@Bishals-MacBook-Air node-app % docker tag nodeimage bishalranjit0606/mynodeapp
bishalranjitkar@Bishals-MacBook-Air node-app % docker push bishalranjit0606/mynodeapp
Using default tag: latest
The push refers to repository [docker.io/bishalranjit0606/mynodeapp]
d55f9cf5818a: Pushed
6a8774fa14ba: Pushed
e3487a8c4629: Pushed
36ef49e04de8: Pushed
f270deee6385: Mounted from library/node
e129ba4f1574: Mounted from library/node
780658909072: Mounted from library/node
eae0a84b6ca2: Mounted from library/node
4b017a36fd9c: Mounted from library/node
20a9b386e10e: Mounted from library/node
```

### Explanation:

- `docker login`: Logs you into Docker Hub.
  - `docker tag nodeimage bishalranjit0606/mynodeapp`: Tags your local image as `bishalranjit0606/mynodeapp` for uploading to Docker Hub.
  - `docker push bishalranjit0606/mynodeapp`: Pushes the image to your Docker Hub repository.
- 

## Step 5: Launch an EC2 Instance and SSH Login

### 1. Launch an EC2 Instance:

- Use **Amazon Linux 2** for the instance type.
- Open **port 80 (HTTP)** in the **Security Group** settings.
- Open **port 22 (SSH)** for SSH access.



## Step 6: Install Docker on EC2

### Update Packages:

```
sudo yum update -y
```

### Install Docker:

```
sudo yum install -y docker
```

### Enable and Start Docker:

```
sudo systemctl enable docker
```

```
sudo systemctl start docker
```

```
[ec2-user@ip-172-31-1-197 ~]$ sudo su
[root@ip-172-31-1-197 ec2-user]# yum update -y
Amazon Linux 2023 Kernel Livepatch repository    120 kB/s | 14 kB    00:00
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-1-197 ec2-user]# yum install docker
Last metadata expiration check: 0:00:10 ago on Wed Feb 26 06:38:19 2025.
Dependencies resolved.
=====
Package                                Arch      Version                                Repository    Size
=====
Installing:
docker                                x86_64    25.0.8-1.amzn2023.0.1                amazonlinux    44 M
Installing dependencies:
containerd                            x86_64    1.7.25-1.amzn2023.0.1                amazonlinux    36 M
iptables-libs                         x86_64    1.8.8-3.amzn2023.0.2                amazonlinux    401 k
iptables-nft                          x86_64    1.8.8-3.amzn2023.0.2                amazonlinux    183 k
libcgroup                             x86_64    3.0-1.amzn2023.0.1                  amazonlinux    75 k
libnetfilter_conntrack                x86_64    1.0.8-2.amzn2023.0.2                amazonlinux    58 k
libnftnl                              x86_64    1.0.1-19.amzn2023.0.2               amazonlinux    30 k
libnftnl                              x86_64    1.2.2-2.amzn2023.0.2               amazonlinux    84 k
pigz                                  x86_64    2.5-1.amzn2023.0.3                  amazonlinux    83 k
runc                                   x86_64    1.2.4-1.amzn2023.0.1                amazonlinux    3.4 M
=====
```

---

## Step 7: Pull the Docker Image from Docker Hub

On your EC2 instance, pull the image from Docker Hub:

```
docker pull bishalranjit0606/mynodeapp
```

### Explanation:

- This command pulls the `mynodeapp` image from your Docker Hub account onto the EC2 instance.

---

## Step 8: Run the Docker Container on EC2

Run the Docker container on your EC2 instance, exposing port 80 for public access:

```
docker run -d -p 80:3000 bishalranjit0606/mynodeapp
```

```
[root@ip-172-31-1-197 ec2-user]# docker pull bishalranjit0606/mynodeapp
Using default tag: latest
latest: Pulling from bishalranjit0606/mynodeapp
155ad54a8b28: Pull complete
8031108f3cda: Pull complete
1d281e50d3e4: Pull complete
447713e77b4f: Pull complete
f87facc2c491: Pull complete
2f9475d0583b: Pull complete
396e9c5702ad: Pull complete
c8728cb69dce: Pull complete
b40fb75709f2: Pull complete
c8e52e499854: Pull complete
9cec0a816327: Pull complete
83871cbcd87c: Pull complete
Digest: sha256:da09161fe5ffbc53a9c7150c72084ce65d6f28661582bf0e2676b24e56b30da6
Status: Downloaded newer image for bishalranjit0606/mynodeapp:latest
docker.io/bishalranjit0606/mynodeapp:latest
[root@ip-172-31-1-197 ec2-user]# docker run -d \
> -p 80:3000 \
> bishalranjit0606/mynodeapp
aedf881d6f203dd7ea52fe59895b59d222714f2289676516b56e4036440cf2cf
[root@ip-172-31-1-197 ec2-user]#
```

### Explanation:

- `-p 80:3000`: Maps port 80 on the EC2 instance to port 3000 inside the container.
- `bishalranjit0606/mynodeapp`: Specifies the image to use.

---

## Step 9: Access the Application

Now, open a browser and navigate to your EC2 instance's public IP address:

```
http://your-ec2-public-ip
```

---

## Conclusion

You've successfully deployed a **Node.js application** using **Docker** on **AWS EC2**, and you can now access your app using the EC2 instance's public IP.