Installing & Running a Node.js Application on AWS EC2 (Amazon Linux)

This guide provides a step-by-step walkthrough for installing Node.js on an Amazon EC2 instance, creating a simple test application, and running it using two different methods.

Step 1: Connect to Your EC2 Instance and Install Node.js

First, connect to your EC2 instance using SSH. Once connected, you'll install Node.js and its package manager, npm.

1. **Update your instance's packages:** It's always a good practice to start by updating the package manager.

   ```
   sudo yum update -y
   ```

2. **Install Node.js:** Use the `yum` package manager to install Node.js. The `y` flag automatically confirms the installation.

   ```
   sudo yum install nodejs -y
   ```

3. **Verify the Installation:** Check the versions of Node.js and npm to ensure they were installed correctly.

   ```
   # Check Node.js version
   node -v

   # Check npm version
   npm -v
   ```

Step 2: Create a Sample Node.js Application

Next, you'll create a directory for your application and set up a basic Express server.

1. **Create a Project Directory:**

   ```
   mkdir nodeapp
   cd nodeapp
   ```

2. **Create a `package.json` File:** This file manages your project's metadata and dependencies. Create the file using a text editor like `nano`.

   ```
   nano package.json
   ```

   Paste the following JSON content into the file. This defines your app and lists `express` as a dependency.

   ```
   {
     "name": "node-app",
     "description": "A simple Node.js test application.",
     "version": "1.0.0",
     "private": true,
     "dependencies": {
       "express": "4.17.1"
     }
   }
   ```

   Save the file and exit `nano` (press `CTRL + X`, then `Y`, then `Enter`).

3. **Create the Main Application File:** This file will contain the code for your web server.

   ```
   nano index.js
   ```

   Paste the following JavaScript code. This code creates a simple web server that responds with "Hello from your Node.js App!"

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 3000;

// Respond with a message for requests to the root URL "/"
app.get('/', (req, res) => {
  res.send('Hello from your Node.js App!');
});

// Start the server and listen on the specified port
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});

module.exports = app;
```

Save and exit the file.

4. **Install Dependencies:** Run `npm install` to download the `express` package defined in your `package.json` file.

```
npm install
```

You should now see a `node_modules` directory in your `nodeapp` folder.

Step 3: Configure AWS Security Group

To access your application from the internet, you must open the port it runs on (in this case, port 3000) in your EC2 instance's security group.

1. Navigate to the **EC2 Dashboard** in your AWS Console.
2. Go to **Instances** and select your instance.
3. In the **Security** tab below, click on the security group name.
4. Click **Edit inbound rules**.
5. Click **Add rule** and configure it as follows:
   - **Type:** `Custom TCP`

- **Port Range:** `3000`
- **Source:** `Anywhere` (or `0.0.0.0/0`) for testing. For better security, you can restrict this to your IP address.

6. Click **Save rules**.

Step 4: Run Your Node.js Application

Here are two methods to run your server.

Method A: Running in the Foreground (for Testing)

This method is simple and useful for testing, but the application will stop as soon as you close your terminal session.

1. **Start the Server:**

   ```
   node index.js
   ```

   You should see the message: `Server is running on http://localhost:3000`.

2. **Access in Browser:** Open your web browser and navigate to: `http://<Your-EC2-Public-IP-Address>:3000`

   You should see the message "Hello from your Node.js App!".

3. **To stop the server**, go back to your terminal and press `CTRL + C`.

Method B: Running in the Background with PM2 (Recommended)

PM2 is a process manager that will keep your application running in the background and automatically restart it if it crashes. This is the preferred method for production.

1. **Install PM2 Globally:**

   ```
   sudo npm install -g pm2
   ```

2. **Start Your Application with PM2:**

```
pm2 start index.js
```

Your app is now running as a background service.

3. **Useful PM2 Commands:**
    - `pm2 list`: View the status of all running applications.
    - `pm2 stop index`: Stop the application.
    - `pm2 restart index`: Restart the application.
    - `pm2 logs`: View the application logs in real-time.
    - `pm2 delete index`: Stop and remove the application from PM2's list.

(Optional) Advanced: Using Nginx as a Reverse Proxy

To run your application on the standard web port (80) without running Node.js as the root user, you can use Nginx as a reverse proxy. This setup forwards traffic from port 80 to your Node.js app running on port 3000.

1. **Install Nginx:**

```
sudo amazon-linux-extras install nginx1 -y
```

2. **Start and Enable Nginx:**

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

3. **Configure Nginx:** Edit the main configuration file.

```
sudo nano /etc/nginx/nginx.conf
```

Find the `location / { ... }` block inside the `server { ... }` section and replace it with the following to proxy requests to your Node app.

```
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;
}
```

4. **Restart Nginx:**

```
sudo systemctl restart nginx
```

Now, you can access your application directly via your EC2's public IP address (`http://<Your-EC2-Public-IP-Address>`) without specifying the port. Ensure port 80 is open in your security group.