

# Local Setup

## Step 1: Clone the Repository

Clone your project repository to your local machine using Git:

```
```bash
git clone <repository-url>
cd your-project-directory
```
```

## Step 2: Configure Environment Variables

Create a `.env` file in your project root directory and set the environment variables:

```
```env
DATABASE_HOST=localhost
DATABASE_USERNAME=your-db-username
DATABASE_NAME=your-db-name
DATABASE_PASSWORD=your-db-password
```
```

## Step 3: Install Dependencies

Install project dependencies using npm:

```
```bash
npm install
```
```

## Step 4: Start the Server

Start the Node.js server:

```
```bash
npm start
```
```

Your local server should now be running and listening on a specified port (port 3000).

# AWS EC2 Deployment

## Step 1: Connect to Your EC2 Instance

Connect to your AWS EC2 instance using SSH. Replace `your-key.pem` and `your-instance-ip` with your own values:

```
```bash
ssh -i your-key.pem ec2-user@your-instance-ip
```
```

## Step 2: Clone the Repository

Clone your project repository on your EC2 instance:

```
```bash
git clone <repository-url>
cd your-project-directory
```
```

## Step 3: Configure Environment Variables

Create a `.env` file in your project directory on the EC2 instance and set the environment variables:

```
```env
DATABASE_HOST=your-rds-hostname
DATABASE_USERNAME=your-db-username
DATABASE_NAME=your-db-name
DATABASE_PASSWORD=your-db-password
```
```

## Step 4: Install Dependencies

Install project dependencies using npm:

```
```bash
npm install
```
```

## Step 5: Start the Server as a Background Process

You can use a process manager like `pm2` to run your Node.js application as a background service:

```
```bash
npm install -g pm2
pm2 start server.js
```
```

This will keep your Node.js application running even if you log out of the SSH session.

#### Step 6: Configure Security Groups

Ensure that your AWS EC2 instance's security group allows incoming traffic on the port your Node.js application is running (default is port 3000).

#### Step 7: Access Your Application

Project should now be able to access your deployed application in a web browser by using your EC2 instance's public IP address and the port you've specified in your Node.js application.