Build and Deploy JavaScript Application using Node.js, npm, and Nginx

This document describes the step-by-step procedure to build and deploy a JavaScript application using two servers: one for building the code (Build Server) and another for deployment (Deploy Server).

1. Prerequisites

Ensure you have two Ubuntu-based servers:
- Build Server: Used to install Node.js and npm, and build the JS project.
- Deploy Server: Used to host the built files using Nginx.

2. Connect to Both Servers

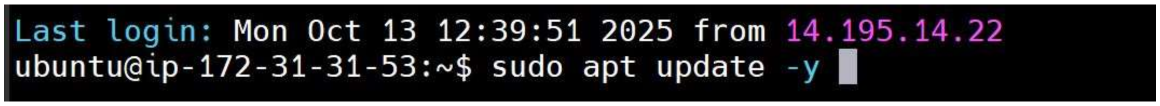
Use SSH to connect to the servers:

ssh ubuntu@3.101.124.71

ssh ubuntu@54.219.222.163

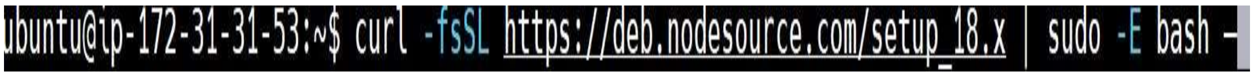
3. Install Node.js and npm on the Build Server

Run the following commands on the Build Server to install Node.js and npm:
sudo apt update



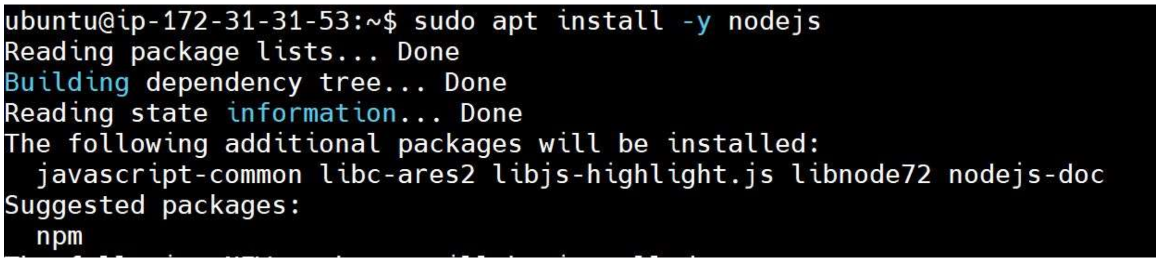
```
Last login: Mon Oct 13 12:39:51 2025 from 14.195.14.22
ubuntu@ip-172-31-31-53:~$ sudo apt update -y
```

curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash –



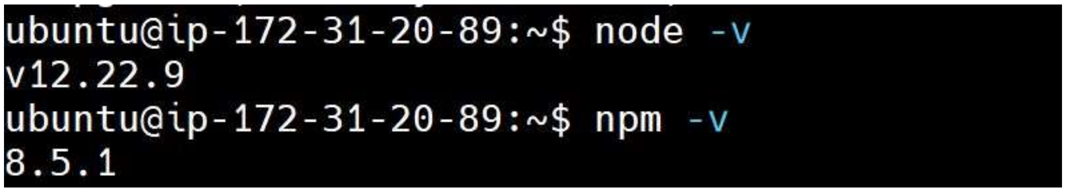
```
ubuntu@ip-172-31-31-53:~$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

The above command ensures node.js to be installed to the latest version
sudo apt install -y nodejs



```
ubuntu@ip-172-31-31-53:~$ sudo apt install -y nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  javascript-common libc-ares2 libjs-highlight.js libnode72 nodejs-doc
Suggested packages:
  npm
```

node -v
npm -v

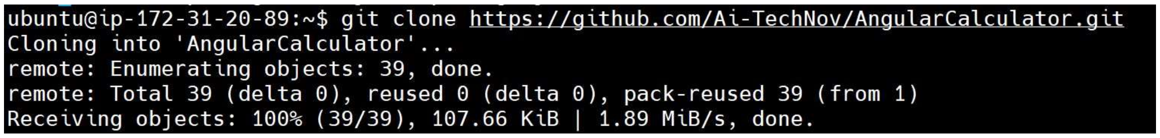


```
ubuntu@ip-172-31-20-89:~$ node -v
v12.22.9
ubuntu@ip-172-31-20-89:~$ npm -v
8.5.1
```

This ensures both Node.js and npm are properly installed and compatible.

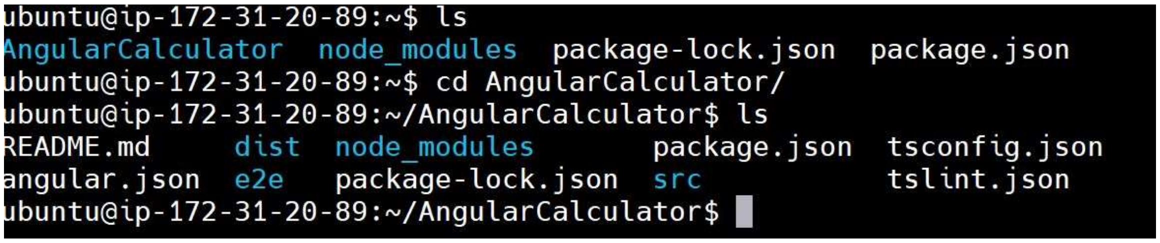
4. Clone and Build the JavaScript Project

Clone your JavaScript or Node.js project from the repository and build it:



```
ubuntu@ip-172-31-20-89:~$ git clone https://github.com/Ai-TechNov/AngularCalculator.git
Cloning into 'AngularCalculator'...
remote: Enumerating objects: 39, done.
remote: Total 39 (delta 0), reused 0 (delta 0), pack-reused 39 (from 1)
Receiving objects: 100% (39/39), 107.66 KiB | 1.89 MiB/s, done.
```

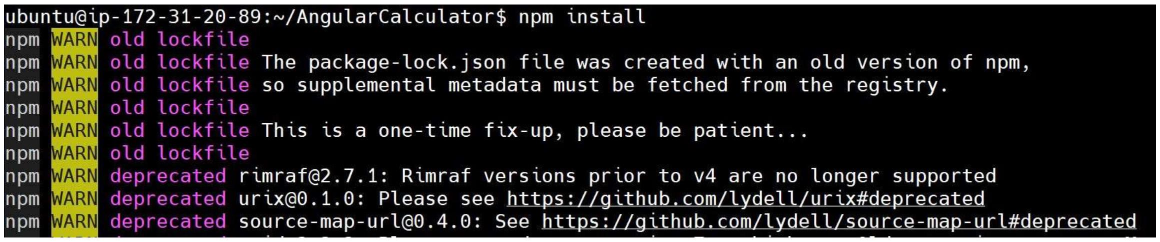
git clone https://github.com/Ai-TechNov/AngularCalculator.git



```
ubuntu@ip-172-31-20-89:~$ ls
AngularCalculator  node_modules  package-lock.json  package.json
ubuntu@ip-172-31-20-89:~$ cd AngularCalculator/
ubuntu@ip-172-31-20-89:~/AngularCalculator$ ls
README.md     dist  node_modules       package.json  tsconfig.json
angular.json  e2e   package-lock.json  src           tslint.json
ubuntu@ip-172-31-20-89:~/AngularCalculator$
```

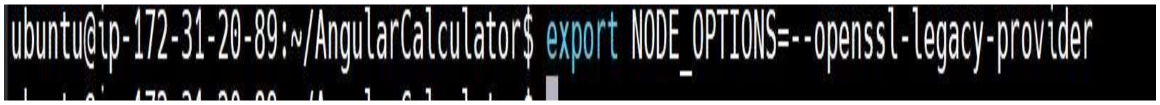
cd AngularCalculator

npm install



```
ubuntu@ip-172-31-20-89:~/AngularCalculator$ npm install
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile
npm WARN deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated source-map-url@0.4.0: See https://github.com/lydell/source-map-url#deprecated
```

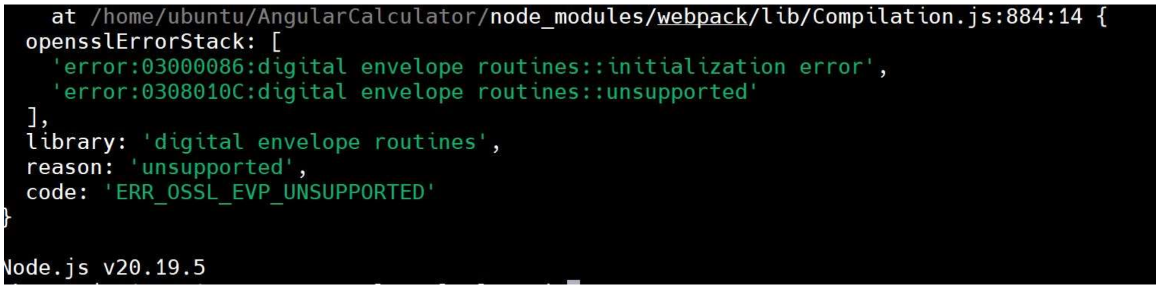
export NODE_OPTIONS=--openssl-legacy-provider



```
ubuntu@ip-172-31-20-89:~/AngularCalculator$ export NODE_OPTIONS=--openssl-legacy-provider
```

It enables Node.js to use legacy OpenSSL algorithms, fixing build errors in older projects on Node.js ≥17.

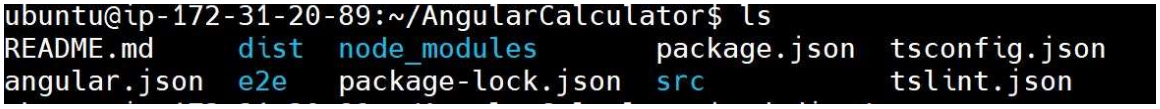
ng build build



```
    at /home/ubuntu/AngularCalculator/node_modules/webpack/lib/Compilation.js:884:14 {
  opensslErrorStack: [
    'error:03000086:digital envelope routines::initialization error',
    'error:0308010C:digital envelope routines::unsupported'
  ],
  library: 'digital envelope routines',
  reason: 'unsupported',
  code: 'ERR_OSSL_EVP_UNSUPPORTED'
}

Node.js v20.19.5
```

The build process generates static files (HTML, CSS, JS) inside the 'build' or 'dist' folder.

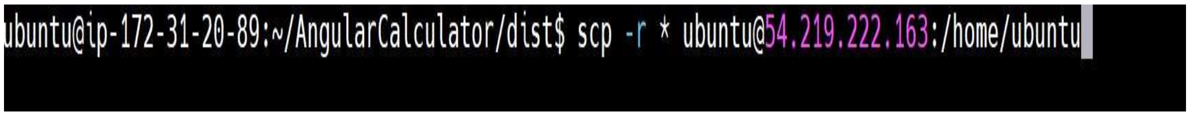


```
ubuntu@ip-172-31-20-89:~/AngularCalculator$ ls
README.md     dist  node_modules       package.json  tsconfig.json
angular.json  e2e   package-lock.json  src           tslint.json
```

5. Transfer Build Files to the Deploy Server

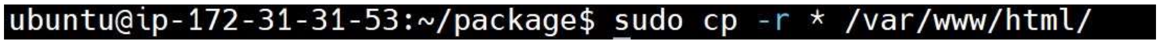
Use the 'scp' command to securely copy the build output to the Deploy Server:

scp -r build/* ubuntu@ 54.219.222.163:/home/ubuntu



```
ubuntu@ip-172-31-20-89:~/AngularCalculator/dist$ scp -r * ubuntu@54.219.222.163:/home/ubuntu
```

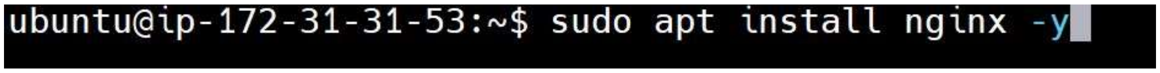
Here we cannot directly copy the files to /var/www/html as it is owned by root. So, first we need to copy the files to home folder of the deploy server and the we can copy to /var/www/html



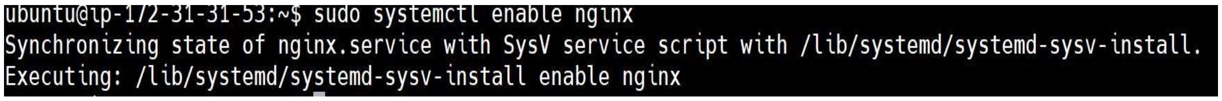
```
ubuntu@ip-172-31-31-53:~/package$ sudo cp -r * /var/www/html/
```

6. Install and Configure Nginx on the Deploy Server

Run the following commands on the Deploy Server to install and configure Nginx:
sudo apt update
sudo apt install nginx -y

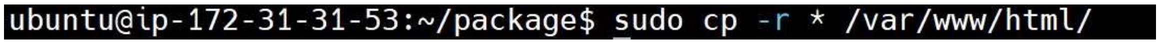

```
ubuntu@ip-172-31-31-53:~$ sudo apt install nginx -y
```

sudo systemctl enable nginx


```
ubuntu@ip-172-31-31-53:~$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
```

sudo systemctl start nginx

Place your build files inside the Nginx web root directory (/var/www/html/).

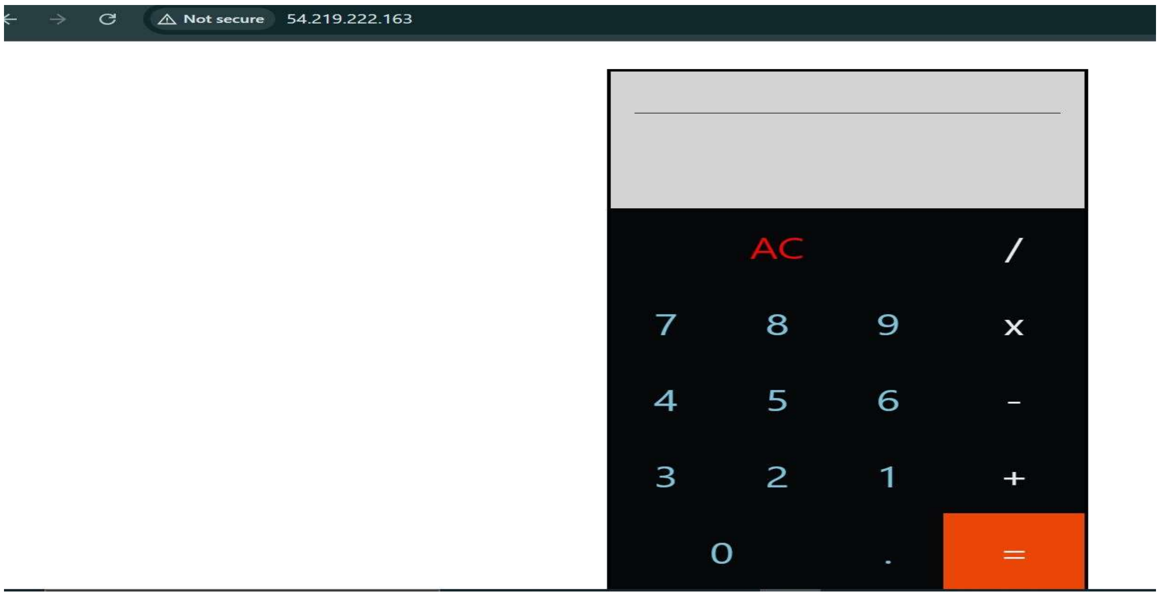

```
ubuntu@ip-172-31-31-53:~/package$ sudo cp -r * /var/www/html/
```

7. Verify Deployment

Check if the web application is accessible by visiting the Deploy Server's public IP in a browser:

http://54.219.222.163/

You should see your JavaScript application running successfully.



9. Summary

You have successfully:
1. Installed Node.js and npm on the Build Server.
2. Built the JavaScript application.
3. Deployed the static build files to the Deploy Server.
4. Configured Nginx to serve the application.

Your web application should now be live and accessible via the Deploy Server's public IP.