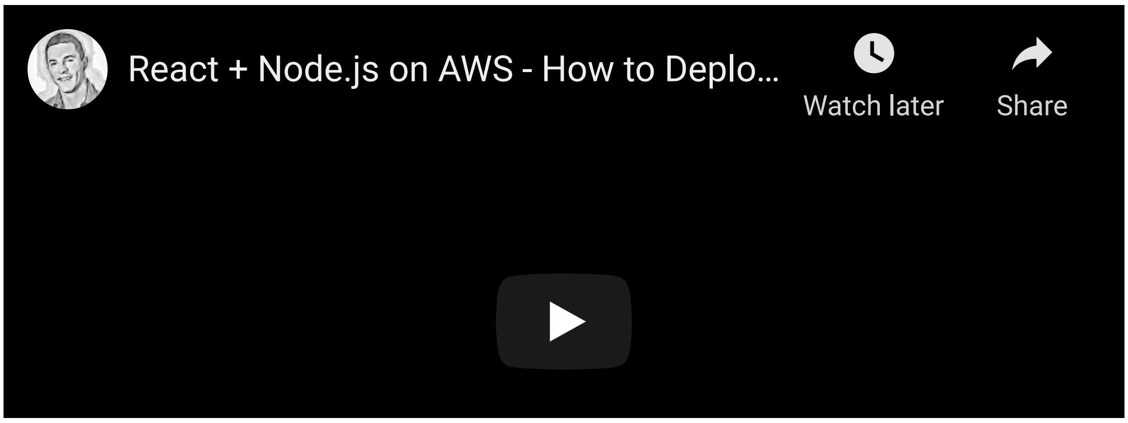
PUBLISHED: NOVEMBER 18 2019
LAST UPDATED: JANUARY 22 2020

React + Node.js on AWS - How to Deploy a MERN Stack App to Amazon EC2

In this tutorial we're going to setup a production ready web server from scratch on the Amazon EC2 (Elastic Compute Cloud) service, then deploy a custom MERN Stack application to it that supports user registration and authentication.



What is a MERN Stack App?

A MERN Stack application is made up of a front-end app built with React that connects to a back-end api built with Node.js + Express + MongoDB, hence the name MERN Stack (Mongo, Express, React, Node). Other variations of the stack include the MEAN Stack that has an Angular front-end, and the MEVN Stack that has a Vue.js front-end.

Scope of this tutorial

This tutorial will be focused on setting up the server on AWS EC2, then deploying and configuring the front-end and back-end pieces of the MERN stack app to work together. For more in-depth information about the React app or Node.js api used in this post, check out the following tutorials which cover them in detail:

- React + Redux - User Registration and Login Tutorial & Example
- NodeJS + MongoDB - Simple API for Authentication, Registration and User Management

Tutorial Contents

Create a new Ubuntu Server on AWS EC2

Watch this step on YouTube

Before doing anything we need a server that we can work on, follow these steps to spin up a new Ubuntu 18.04 server instance on AWS EC2.

1. Sign into the AWS Management Console at https://aws.amazon.com/console/. If you don't have an account yet click the "Create a Free Account" button and follow the prompts.
2. Go to the EC2 Service section.
3. Click the "Launch Instance" button.
4. **Choose AMI** - Check the "Free tier only" checkbox, enter "Ubuntu" in search box and press enter, then select the "Ubuntu Server 18.04" Amazon Machine Image (AMI).
5. **Choose Instance Type** - Select the "t2.micro" (Free tier eligible) instance type and click "Configure Security Group" in the top menu.
6. **Configure Security Group** - Add a new rule to allow HTTP traffic then click "Review and Launch".
7. **Review** - Click Launch
8. Select "Create a new key pair", enter a name for the key pair (e.g. "my-aws-key") and click "Download Key Pair" to download the private key, you will use this to connect to the server via SSH.
9. Click "Launch Instances", then scroll to the bottom of the page and click "View Instances" to see details of the new Ubuntu EC2 instance that is launching.

Connect to Ubuntu EC2 Instance via SSH

Watch this step on YouTube

Once the EC2 instance reaches a running state you can connect to it via SSH using the private key downloaded in the previous step.

1. Open a terminal window and update the permissions of the private key file with the command `chmod 400 <path-to-key-file>` e.g. `chmod 400 ~/Downloads/my-aws-key.pem`, the key must not be publicly viewable for SSH to work.
2. Copy the "Public DNS (IPv4)" property from the instance description tab in the AWS Console, then connect to the instance from the terminal window with the command `ssh -i <path-to-key-file> ubuntu@<domain name>` e.g. `ssh -i ~/Downloads/my-aws-key.pem ubuntu@ec2-52-221-185-40.ap-southeast-2.compute.amazonaws.com`
3. Enter `yes` to the prompt "Are you sure you want to continue connecting (yes/no)?" to add the url to your list of known hosts.

NOTE: If you're using Windows you can connect to your instance via SSH using the PuTTY SSH client, for instructions see Connect Using PuTTY in the AWS docs.

Setup Web Server with Node.js + MongoDB + NGINX

Watch this step on YouTube

The below command executes a script to automatically setup and configure a production ready MERN Stack web server on Ubuntu that includes Node.js, MongoDB, PM2, NGINX and UFW.

For more details about how the script works see Setup Node.js + MongoDB Production Server on Ubuntu.

While connected to the new AWS EC2 instance in the terminal window, run the following command:
`curl https://gist.githubusercontent.com/cornflourblue/f0abd30f47d96d6ff127fe8a9e5bbd9f/raw/e3047c9dc3ce8b796e7354c92d2c47ce61981d2f/setup-nodejs-mongodb-production-server-on-ubuntu-1804.sh | sudo bash`

For instructions on how to securely connect to the remote MongoDB server from your local machine using Mongo Shell or MongoDB Compass see Connect to remote MongoDB on AWS EC2 simply and securely via SSH tunnel.

Deploy Node.js + MongoDB Back-end API

Watch this step on YouTube

Follow these steps to setup the Node.js API on the server and configure NGINX to enable access to it.

1. Clone the Node.js + MongoDB API project into the `/opt/back-end` directory with the command `sudo git clone https://github.com/cornflourblue/node-mongo-registration-login-api /opt/back-end`
2. Navigate into the back-end directory and install all required npm packages with the command `cd /opt/back-end && sudo npm install`
3. Start the API using the PM2 process manager with command `sudo pm2 start server.js`

The API is now running on Node.js under the PM2 process manager and listening on port 4000. Only port 80 (HTTP) is publicly accessible on the server so we can't hit the API yet, this will be possible after we've configured NGINX as a reverse proxy to pass through HTTP traffic to the api (more on this shortly).

Deploy React + Redux Front-end app

Watch this step on YouTube

Follow these steps to setup the React application on the server.

1. Clone the React + Redux project into the `/opt/front-end` directory with the command `sudo git clone https://github.com/cornflourblue/react-redux-registration-login-example.git /opt/front-end`
2. Navigate into the front-end directory and install all required npm packages with the command `cd /opt/front-end && sudo npm install`
3. Update the app to use real backend API:
   - Run the command `sudo nano /opt/front-end/src/index.jsx` to open the main react entry file in the nano text editor.
   - Delete the following lines from the file to remove the fake backend that the react app uses by default:

     ```
     // setup fake backend
     import { configureFakeBackend } from './_helpers';
     configureFakeBackend();
     ```

   - Save the file by pressing `ctrl` + `x` and selecting `Yes` to save.
4. Configure the path to API:
   - Run the command `sudo nano /opt/front-end/webpack.config.js` to open the webpack config file in the nano text editor.
   - Change the `apiUrl` config property to `'/api'` like below so it points to the public path we configured in NGINX in the previous section:

     ```
     // global app config object
     config: JSON.stringify({
         apiUrl: '/api'
     })
     ```

   - Save the file by pressing `ctrl` + `x` and selecting `Yes` to save.
5. Build the front end app with the command `sudo npm run build`

The React app is now built and ready to be served from the `/opt/front-end/dist` directory, in the next step we'll configure our NGINX web server to enable access to it.

Configure NGINX to serve the Node.js API and React front-end

Watch this step on YouTube

Since our MERN Stack application is made up of two separate projects that both need to be accessed via the same port (HTTP on port 80), we're going to use NGINX as our public facing web server to receive requests for both the front-end and back-end, and decide where to send each request based on its path. Requests beginning with the path `/api/*` will be proxied through to the Node.js api running on port 4000, while other requests will serve the React front-end app and associated files (js/css/images).

Follow these steps to configure NGINX for the MERN stack app.

1. Delete the default NGINX site config file with the command `sudo rm /etc/nginx/sites-available/default`
2. Launch the nano text editor to create an new default site config file with `sudo nano /etc/nginx/sites-available/default`
3. Paste in the following config:

```
server {
  listen 80 default_server;
  server_name _;

  # react app & front-end files
  location / {
    root /opt/front-end/dist;
    try_files $uri /index.html;
  }

  # node api reverse proxy
  location /api/ {
    proxy_pass http://localhost:4000/;
    }
}
```

4. Save the file by pressing `ctrl` + `x` and selecting `Yes` to save.
5. Restart NGINX with the command `sudo systemctl restart nginx`

NGINX Config Reference

`server { ... }` defines a server block which contains the configuration for a virtual server within NGINX.

`listen 80 default_server;` uses the listen directive to configure the virtual server to accept requests on port `80` and sets it as the default virtual server on this NGINX server.

`server_name _;` uses the server_name directive to set the server name to an underscore (`_`) to make this server block a catch-all block that matches any domain name that doesn't match another more specific server block. Since this example has only one server block it will match all domain names.

`location / { ... }` defines a location block which contains the configuration for requests that have a URI beginning with a forward slash (`/`), unless the request URI matches another more specific location block.

`root /opt/front-end/dist;` uses the root directive to set the root directory to the front end dist folder (`/opt/front-end/dist`) for requests matching this location block.

`try_files $uri /index.html;` uses the try_files directive to first check for the existence of a file matching the request URI (`$uri`) and returning it if there is one. If no file matches the request URI then it defaults to returning `/index.html`.

`location /api/ { ... }` defines a location block which contains the configuration for requests that have a URI beginning with `/api/`.

`proxy_pass http://localhost:4000/;` uses the proxy_pass directive to proxy requests beginning with `/api/` through to the Node.js API running at `http://localhost:4000/`.

Test your new MERN Stack application running on AWS

Watch this step on YouTube

Enter the hostname of your AWS EC2 instance in a browser to access and test your new MERN stack application.

The hostname is the "Public DNS (IPv4)" property located on the instance description tab in the AWS Console.

Subscribe or Follow Me For Updates

Subscribe to my YouTube channel or follow me on Twitter or GitHub to be notified when I post new content.

- Subscribe on YouTube at https://www.youtube.com/JasonWatmore
- Follow me on Twitter at https://twitter.com/jason_watmore
- Follow me on GitHub at https://github.com/cornflourblue
- Feed formats available: RSS, Atom, JSON

My Motorcycle Adventure Vlog

Other than coding I'm currently attempting to travel around Australia by motorcycle with my wife Tina on a pair of Royal Enfield Himalayans. You can follow our adventure on YouTube at https://www.youtube.com/TinaAndJason.

Tags: React, NodeJS, MERN Stack, AWS, NGINX, Deployment

Share:

More React Posts

- Next.js 11 - JWT Authentication Tutorial with Example App
- Next.js - NavLink Component Example with Active CSS Class
- Next.js - Make the Link component work like React Router Link
- React Hook Form 7 - Required Checkbox Example
- React + Axios - HTTP DELETE Request Examples
- React + Axios - HTTP PUT Request Examples
- React Hook Form 7 - Form Validation Example
- Next.js 10 - CRUD Example with React Hook Form
- React + Fetch - HTTP DELETE Request Examples
- React + Fetch - HTTP PUT Request Examples
- React + Facebook - How to use the Facebook SDK in a React App
- React - Facebook Login Tutorial & Example
- React Router v5 - Fix for redirects not rendering when using custom history
- React Hook Form - Combined Add/Edit (Create/Update) Form Example
- React - CRUD Example with React Hook Form
- React - Required Checkbox Example with React Hook Form
- React - Form Validation Example with React Hook Form
- React - Dynamic Form Example with React Hook Form
- React + Formik Dynamic Form Example
- React - Display a list of items
- React + Axios - HTTP POST Request Examples
- React + Axios - HTTP GET Request Examples
- React - Setup Development Environment
- React Boilerplate - Email Sign Up with Verification, Authentication & Forgot Password
- React Hooks + RxJS - Communicating Between Components with Observable &

Subject

- React + Formik - Combined Add/Edit (Create/Update) Form Example
- Fetch API - A Lightweight Fetch Wrapper to Simplify HTTP Requests
- React + Formik - Master Details CRUD Example
- React Hooks + Bootstrap - Alert Notifications
- React Router - Relative Links Example
- React Router - Remove Trailing Slash from URLs
- React + Fetch - Fake Backend Example for Backendless Development
- React Hooks + Redux - User Registration and Login Tutorial & Example
- React - Alert (Toaster) Notifications
- React - How to add Global CSS / LESS styles to React with webpack
- React + Formik 2 - Form Validation Example
- React + Formik - Required Checkbox Example
- React + Fetch - HTTP POST Request Examples
- React + Fetch - HTTP GET Request Examples
- React + ASP.NET Core on Azure with SQL Server - How to Deploy a Full Stack App to Microsoft Azure
- React + Node - Server Side Pagination Tutorial & Example
- React + Formik - Form Validation Example
- React (without Redux) - JWT Authentication Tutorial & Example
- React + RxJS - Communicating Between Components with Observable & Subject
- React - Role Based Authorization Tutorial with Example
- React - Basic HTTP Authentication Tutorial & Example
- React + npm - How to Publish a React Component to npm
- npm - JW React Pagination Component
- npm - JW React Modal Dialog
- React - Custom Modal Window / Dialog Box
- React + Redux - JWT Authentication Tutorial & Example
- React + Redux - User Registration and Login Tutorial & Example
- React - Pagination Example with Logic like Google

What do you think?

23 Responses

Upvote Funny Love Surprised Angry Sad

9 Comments Jason Watmore's Blog Disqus' Privacy Policy 1 Login

Recommend 2 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Deepak Suthar • a year ago

Hello. Thank for this tutorial. I followed your all instructions, but when I hit server ip in browser it is not loading. What mistake I have made? Do you have any idea?

see more

1 ^ | ˅ • Reply • Share ›

Janelle ↪ Deepak Suthar • 7 months ago

I am experiencing this as well! :(

^ | ˅ • Reply • Share ›

Casey Craig Kawamura • a year ago

Hey Jason, there isn't any directory for "/etc/nginx/sites-available" or "sites-enabled" when I install NGINX. What should I do?

1 ^ | ˅ • Reply • Share ›

Christopher Regner • a year ago

Incredibly simple tutorial for a very simple task. Very succinct too! I rarely comment on articles but you deserve it! I'm still working on deploying an app, I'll post here how it goes. Cheers!

see more

1 ^ | ˅ • Reply • Share ›

Dmitry • 7 months ago

Hi. I have just deployed my react and express project to AWS. front end and backend is working well on my local but isn't working on live server.
ex: 1. CSS isn't working well.
2. login and sign up isn't working well on my app of live server.
I am going to fix these problems.
Would you please help me someone? Thank you.

see more

^ | ˅ • Reply • Share ›

Emmanuel Martinez • 7 months ago

Hi Jason, do you think the steps still work to this day?

^ | ˅ • Reply • Share ›

Garg Vaibhav • a year ago

Hi @Jason Watmore,
My MERN app doesn't have a seperate webpackconfig.js
The server is as ./bin/www
And it uses two ports 4000 for http and 4443 for https.
The server is configured with Cors , corswithOptions(one with white list).
How should I proceed -
1.to intergrate these server ports with client on NGINX
2.configure a new secutiy group on amazon aws ec2 instance so https requests can be used with http

see more

^ | ˅ • Reply • Share ›

Adeola Oni • a year ago

Please anybody with a tutorial link onhow to deploy a MERN stack app to a cpanel on private virtual vps server?

^ | ˅ • Reply • Share ›

Vihar Desu • a year ago • edited

Hey, I love this tutorial! Thanks for making this.

I'm new to AWS and how all this works.

How can I edit and customize the front-end and back-end scripts after the website is up and running on AWS?

Is there a simple way to test this code locally and then push edits to SSH using some sort of GUI?

see more

^ | ˅ • Reply • Share ›

ABOUT

I'm a web developer in Sydney Australia and the technical lead at Point Blank Development, I've been building websites and web applications in Sydney since 1998.

Find me on:

Subscribe to Feed: RSS, Atom, JSON

Code Snippets

MONTHS

2021
- August (1)
- July (4)
- June (7)
- May (8)
- April (6)

2020
2019

2018
2017
2016
2015
2014
2013
2012
2011

TAGS

Alerts, Angular, Angular 10, Angular 11, Angular 2, Angular 4, Angular 5, Angular 6, Angular 7, Angular 8, Angular 9, Angular Directive, Angular UI Router, AngularJS, Animation, ASP.NET, ASP.NET Core, ASP.NET Web API, Authentication and Authorization, AWS, Axios, Azure, Basic Authentication, Blazor, Bootstrap, C#, Chai, CKEditor, CSS3, DDD, Deployment, Design Patterns, Dynamic LINQ, EF Core, ELMAH, ES6, Exchange, Facebook, Fetch, Fluent NHibernate, Formik, Google Analytics, Google API, Google Maps API, Google Plus, Heroku, HTML5, HTTP, IIS, Insecure Content, Instagram API, Ionic Framework, iOS, iPhone, JavaScript, jQuery, JWT, LinkedIn, LINQ, Login, MEAN Stack, MERN Stack, MEVN Stack, Mocha, Modal, MongoDB, Moq, MVC, MVC5, MySQL, .NET, NextJS, NGINX, ngMock, NHibernate, Ninject, NodeJS, npm, Pagination, Pinterest, Razor Pages, React, React Hooks, Redmine, Redux, Registration, Repository, RxJS, Security, Sequelize, Shell Scripting, Sinon, SinonJS, SSH, TDD, Terraform, Twitter, TypeScript, Ubuntu, Umbraco, Unit of Work, Unit Testing, URL Rewrite, Validation, VS Code, Vue, Vue 3, Vuex, Webpack, Windows Server 2008