**Deploying a Next.js App manually on AWS EC2: A Step-by-Step Guide**

[[](https://medium.com/@mudasirhaji?source=post_page---byline--58b266ff1c52---------------------------------------)](https://medium.com/@mudasirhaji?source=post_page---byline--58b266ff1c52---------------------------------------)

[Mudasir](https://medium.com/@mudasirhaji?source=post_page---byline--58b266ff1c52---------------------------------------)

·

Following

7 min read

·

Aug 18, 2023

460

7

***Introduction***

In the ever-evolving landscape of web development, modern frameworks like Next.js have revolutionized how we build user interfaces. Next.js, a React framework, offers a seamless experience for building server-rendered and statically-generated React applications. Its ability to enhance performance, SEO, and user experience makes it a developer's go-to choice. In this guide, we will walk you through deploying a Next.js app on an AWS EC2 instance, leveraging the power of the cloud to deliver your application to the world.

**Benefits of Deploying Next.js on AWS Cloud**

Before diving into the deployment process, let’s highlight some key benefits of hosting your Next.js app on the AWS cloud:

1. Scalability: AWS provides the flexibility to scale your EC2 instances based on demand. This means your Next.js app can handle traffic spikes without compromising performance.
2. Reliability: AWS offers a highly reliable infrastructure with a Service Level Agreement (SLA) that ensures your application is available and operational when your users need it.
3. Global Reach: Deploying on AWS allows you to choose data centers around the world, reducing latency and providing a faster experience for users across different geographical locations.
4. Security: AWS provides robust security measures, including firewalls, encryption, and identity management, ensuring that your Next.js app and user data are well-protected.
5. Cost Efficiency: AWS EC2 instances offer a pay-as-you-go model, allowing you to control costs by selecting the appropriate instance type and scaling resources as needed.

**Step-by-Step Guide to Deploying a Next.js App on AWS EC2**

Now, let’s dive into the deployment process. We’ll break it down into manageable steps to ensure a smooth experience:

**Prerequisites**

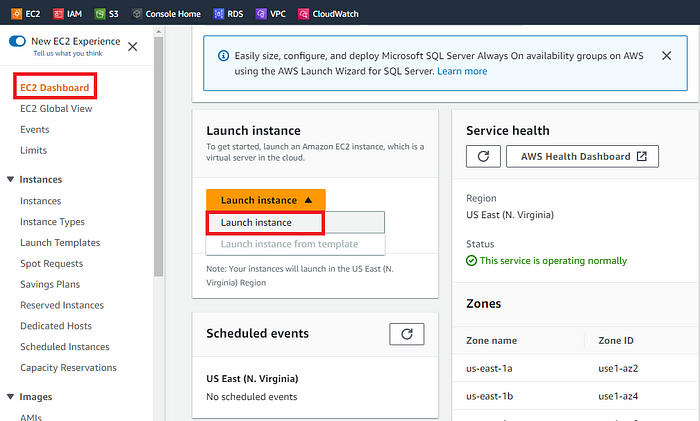
* Github account
* AWS Account
* Next.js web application

**Step 1: Set Up AWS Account**

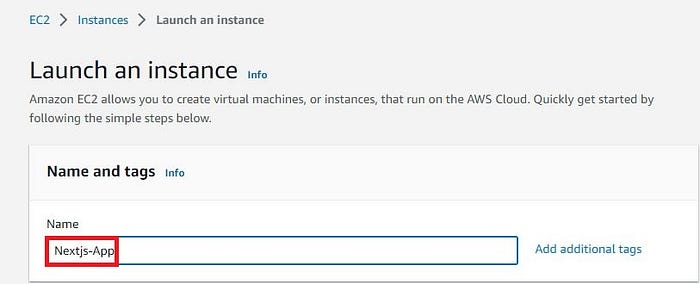
1. Sign in to your AWS Management Console or create a new account if you don’t have one.
2. Navigate to the EC2 dashboard to create and manage instances.

**Step 2: Launch an EC2 Instance**

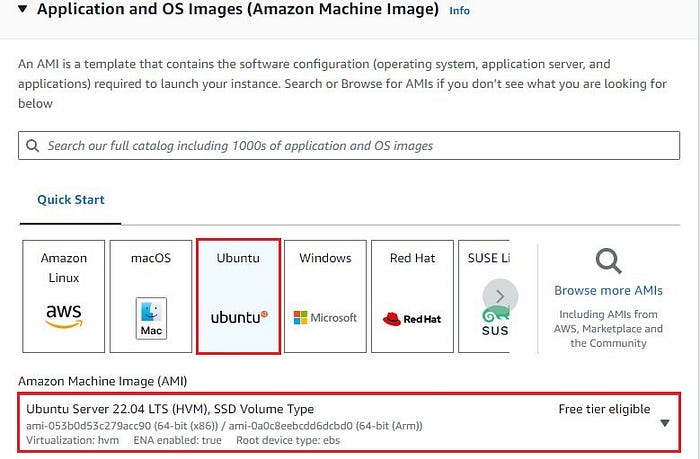
Log in to the Amazon management console, open EC2 Dashboard, click on the Launch Instance drop-down list, and click on Launch Instance as shown below:



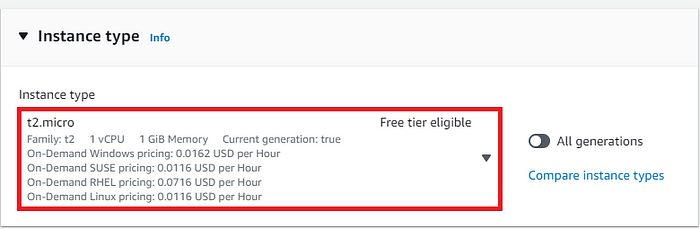
Once the **Launch an instance** window opens, provide the name of your EC2 Instance:



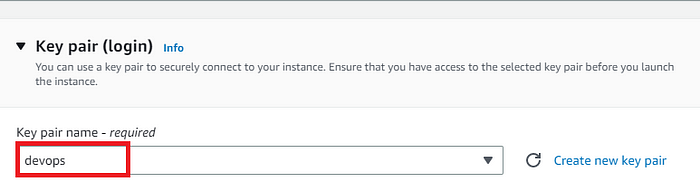
For this demo, we will select Ubuntu 22.04 LTS, which is also free tier eligible.



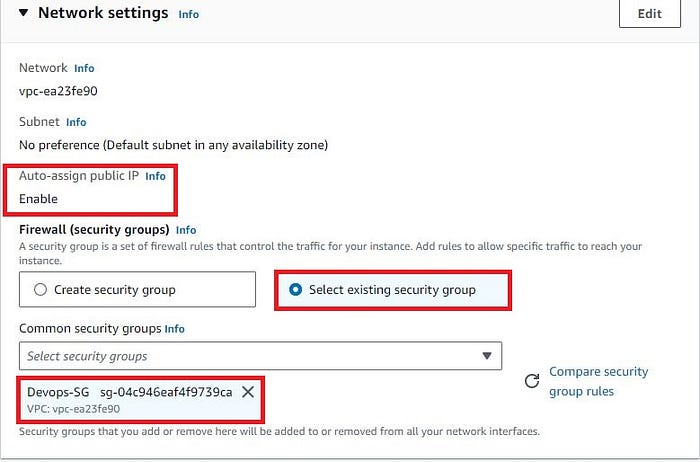
Choose an Instance Type. Here you can select the type of machine, number of vCPUs, and memory that you want to have. Select **t2.micro** which is free-tier eligible.



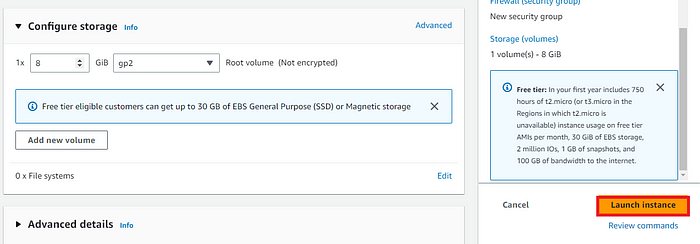
For this demo, we will select an already existing key pair. You can create new key pair if you don’t have:



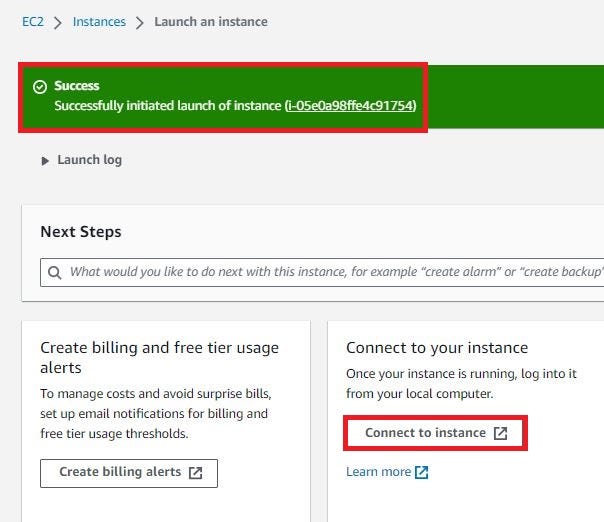
Now under **Network Settings**, Choose the default VPC with Auto-assign public IP in enable mode. For this demo, I will select an existing security group and will make sure under the Inbound rules of my Devops-SG, HTTP, and HTTPS ports are open. Click on Save rules to proceed.



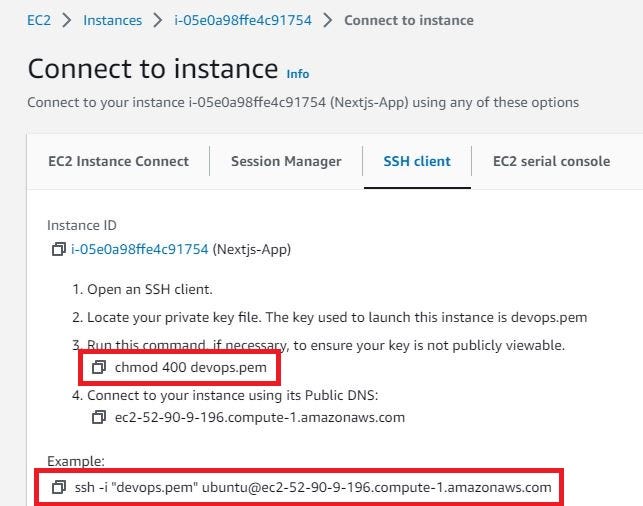
Rest of the settings we will keep them at default and go ahead and click on **Launch Instance**



On the next screen you can see a success message after the successful creation of the EC2 instance, click on Connect to instance button:



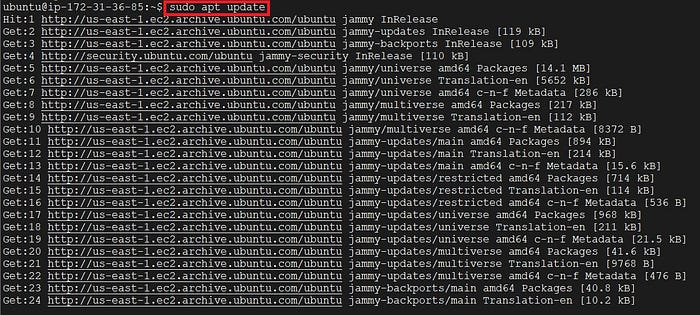
Now **connect to instance** wizard will open, go to SSH client tab and copy the provided chmod and SSH command:



Open any SSH Client in your local machine, take the public IP of your EC2 Instance, and add the pem key and you will be able to access your EC2 machine.

**Step 3: Prepare the EC2 Instance and Install other Dependencies**

Update the system packages: Run sudo apt update to ensure your instance has the latest updates.



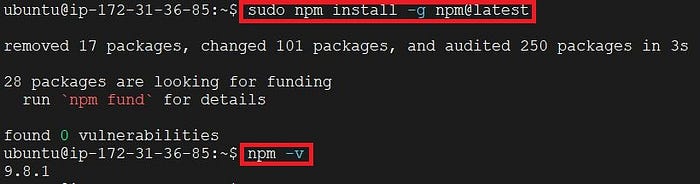
Install Node.js and npm. The Node js version used in this tutorial is **16 LTS.**Execute the below commands:

curl -sL https://deb.nodesource.com/setup\_16.x | sudo -E bash -  
sudo apt-get install -y nodejs  
node -v

https://miro.medium.com/v2/resize:fit:381/1*UgeXgVTrvuYVNw8Itv1FHg.jpeg

Ensure that the most current version of NPM is installed:

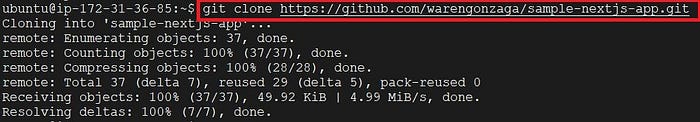
sudo npm install -g npm@latest  
npm -v



**Step 4: Clone your Nextjs App from GitHub.**

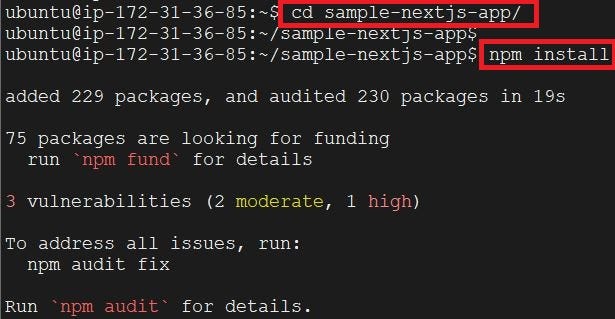
In this demo, we will use a publicly available Nextjs Project on GitHub.Clone the repo on the EC2 Instance.

https://github.com/warengonzaga/sample-nextjs-app.git



**Step 5: npm install**

Navigate to the project directory/folder and execute the command, to install the dependencies required to run the Next.js web application



**Step 6: npm run build**

To make the application ready for the production stage, we need to bundle the javascript files. This process is done by Next.js by executing the following command:

The output of the npm run build command is given for your reference.

ubuntu@ip-172-31-36-85:~/sample-nextjs-app$ npm run build  
  
> sample-next-app@0.1.0 build  
> next build  
  
info - SWC minify release candidate enabled. https://nextjs.link/swcmin  
Attention: Next.js now collects completely anonymous telemetry regarding usage.  
This information is used to shape Next.js' roadmap and prioritize features.  
You can learn more, including how to opt-out if you'd not like to participate in this anonymous program, by visiting the following URL:  
https://nextjs.org/telemetry  
  
info - Linting and checking validity of types  
Browserslist: caniuse-lite is outdated. Please run:  
 npx browserslist@latest --update-db  
 Why you should do it regularly: https://github.com/browserslist/browserslist#browsers-data-updating  
Browserslist: caniuse-lite is outdated. Please run:  
 npx browserslist@latest --update-db  
 Why you should do it regularly: https://github.com/browserslist/browserslist#browsers-data-updating  
info - Creating an optimized production build  
info - Compiled successfully  
info - Collecting page data  
info - Generating static pages (3/3)  
info - Finalizing page optimization  
  
Route (pages) Size First Load JS  
┌ ○ / 689 B 78.6 kB  
├ └ css/ae0e3e027412e072.css 707 B  
├ /\_app 0 B 77.9 kB  
├ ○ /404 186 B 78.1 kB  
└ λ /api/hello 0 B 77.9 kB  
+ First Load JS shared by all 78.1 kB  
 ├ chunks/framework-db825bd0b4ae01ef.js 45.7 kB  
 ├ chunks/main-3123a443c688934f.js 30.9 kB  
 ├ chunks/pages/\_app-0e6b46beaaa55ac1.js 498 B  
 ├ chunks/webpack-7ee66019f7f6d30f.js 755 B  
 └ css/ab44ce7add5c3d11.css 247 B  
  
λ (Server) server-side renders at runtime (uses getInitialProps or getServerSideProps)  
○ (Static) automatically rendered as static HTML (uses no initial props)

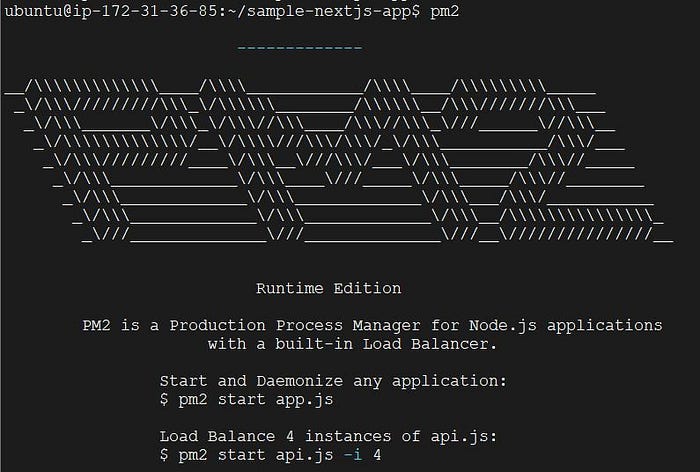
**Step 6: Install PM2**

We require a solution for handling the Next.js process, ensuring it continues running in the background even after closing the terminal. To fulfill this need, PM2 serves as the ideal tool for managing the process.

* Install PM2 using the below command:

sudo npm install pm2 -g

* To verify PM2 installation, execute the following command pm2, you will get a response similar to the screenshot given below.

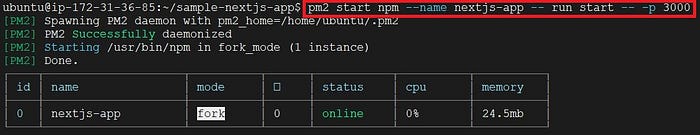


**Step 7: Run Next.js via PM2 in the background**

We need to run, stop and restart the Next.js application even after the terminal is closed. This can be achieved using the PM2 tool.

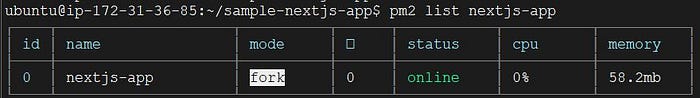
Execute the below code to run Next.js with PM2:

pm2 start npm --name nextjs-app -- run start -- -p 3000

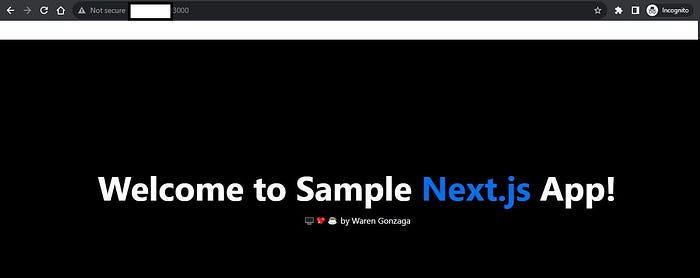


We can also check the status of our nextjs-app by using the below command:

pm2 list nextjs-app

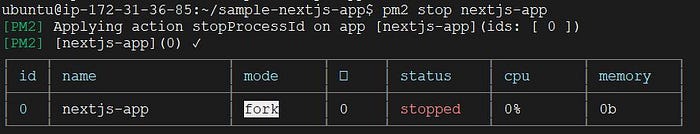


To verify the functionality of the Next.js application, simply enter the EC2 Instance’s public IP along with port number 3000 in a web browser. For instance, use the format 0.0.0.0:3000



To **stop**the nextjs-app process, use the below command:

pm2 stop nextjs-app



To start the nextjs-app process:

pm2 start nextjs-app

To restart the nextjs-app process:

pm2 restart nextjs-app

To Delete the nextjs-app process:

pm2 delete nextjs-app

**Conclusion**

In conclusion, this blog has demonstrated the successful deployment of a Next.js app on an AWS EC2 instance using manual steps. While these steps provide a solid foundation for deploying applications, it’s important to note that a more efficient and streamlined approach can be achieved through containerization and Continuous Integration/Continuous Deployment (CI/CD) pipelines.

In our upcoming posts, we will delve into the world of containerization and explore how leveraging tools like Docker and Kubernetes, along with CI/CD pipelines, can automate the deployment process, enhance scalability, and further optimize the management of Next.js applications. Stay tuned for our future discussions on these advanced deployment strategies.

If you found this blog post helpful and insightful, I invite you to show your appreciation by giving it a clap! Your support fuels my motivation to continue sharing valuable content. Don’t forget to hit that follow button as well, so you can stay connected and receive updates on upcoming posts. Let’s embark on this journey together and explore even more exciting insights into the world of DevOps. Your engagement is greatly appreciated! 👏🔗

<http://13.203.154.123:3000>

<http://public_ip:3000>

Your Next.js app is running on an AWS EC2 instance with the **public IP: 13.203.154.123** and **private IP: 172.31.14.202**. To access it, follow these steps:

### 1️⃣ **Check if the App is Running on the Correct Port**

By default, Next.js runs on port **3000**. Try accessing it in your browser:

cpp

CopyEdit

http://13.203.154.123:3000