AWS & DevOps Report

THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU

(An Autonomous Institute under VTU, Belagavi)

In partial fulfilment of the requirements for the award of degree of

Bachelor of Engineering in

Computer Science and Engineering

Submitted by

M N R SAI MAMATHA (4NI19CS062)

Under the guidance of

Thamotharan N
DevOps Lead Architect
Solvendo

B R Vatsala
Assistant Professor
Dept. of CS&E
NIE, Mysuru

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

Mysore-570 008

2022-2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

CERTIFICATE

This is to certify that the work carried out by **M N R SAI MAMATHA (4NI19CS062)** in partial fulfilment of the requirements for the completion of the course "AWS & DevOps" in the semester VII, Department of Computer Science and Engineering , as per the academic regulations of The National Institute of Engineering, Mysuru , during the academic year 2022-23. It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the department library.

Signature of the guide

_____

B R Vatsala
Assistant Professor
Dept of CS&E
NIE, Mysuru

Table of Contents

List of Figures

Introduction

Node JS:

Node.js is an open-source, cross-platform JavaScript runtime environment used for executing JavaScript code outside of a web browser. Node JS is a runtime for JavaScript, that is built on top of google chrome V8 engine which is writing in C++.

Node.js is a great web framework for beginners because it works great for **data-intensive applications**, like streaming and real-time apps, and Node.js makes it easy to start **building the back-end**.

Node.js allows us to **use JavaScript everywhere** and on any browser, including MacOS, Linux, and Windows. When we say everywhere, we mean the front-end, the middle-ware, and the back-end. Node.js is, therefore, part of some very popular web development stacks, such as the MERN stack, MEVN stack, and MEAN stack.

There are a number of characteristics that make Node.js what it is:

- **Google Chrome V8 JavaScript Engine:** This runtime environment is built on the Google Chrome V8 JavaScript runtime engine. In the same way a Java Virtual Machine translates bytecode, the Chrome V8 JavaScript engine takes JavaScript and makes it readable.
- **Modules/Packages:** Node.js has npm, a node package manager, with a library of over 350,000 packages to help get your project or application off the ground with efficiency and ease.
- **Event Driven, Single-Threaded I/O Model:** JavaScript relies on user interactions or events to run. In most cases, code is run synchronously. Server requests and other such asynchronous tasks rely on a system of promises or async/await functions to handle these inputs and outputs.

AWS:

Amazon Web Services is very popular and reliable platform to host our application on top of their servers. There are many types of amazon's services that we can use to host our app, but we will use EC2 (Elastic Compute Service). It is free and provides a very customized configuration during the creation process. We create an instance: a set amount of resources initialized (OS, disk space, Ram, etc).

EC2:

It's free and provide very customizable configurations to the end user. We can select our favourite Linux based operating system and install all the required utilities and software on that OS. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Deploying a Node.js application on AWS EC2 server

Steps to be followed:

1. **Create account on AWS**:

   Go to "https://www.aws.amazon.com" and create a new account on AWS. If you have already registered on AWS then go to AWS console and sign-in as root user and then enter root email and password.

2. **Launch an EC2 instance**:

   Once you are logged in you can see all the aws services under the **Services** section as shown in Figure 1.

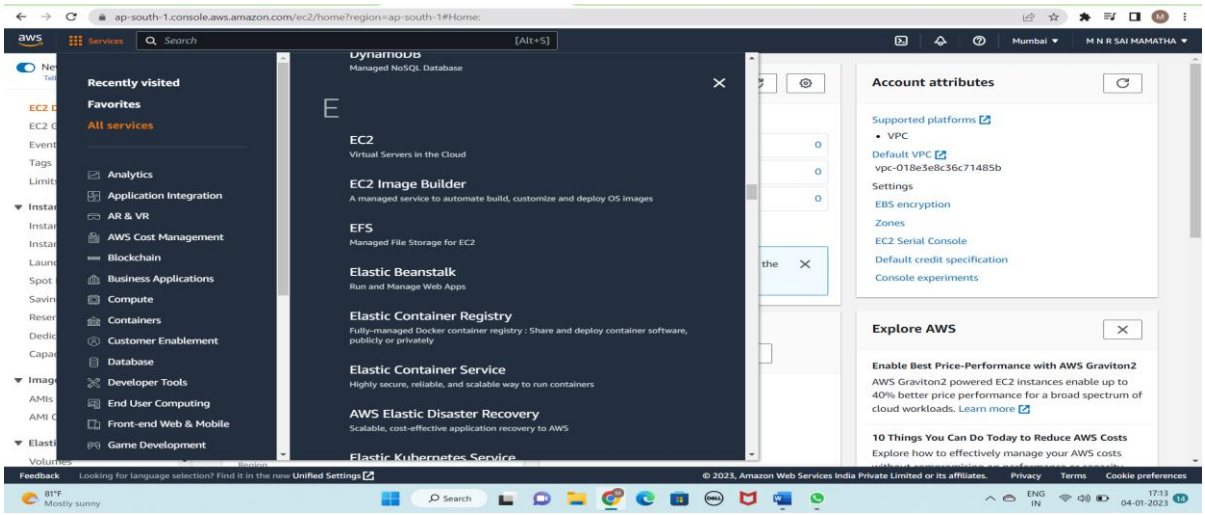


Figure 1: AWS Services

Click on the ***EC2*** Service, it will show us EC2 Dashboard page, as shown in Figure 2, click on ***Launch Instance***.

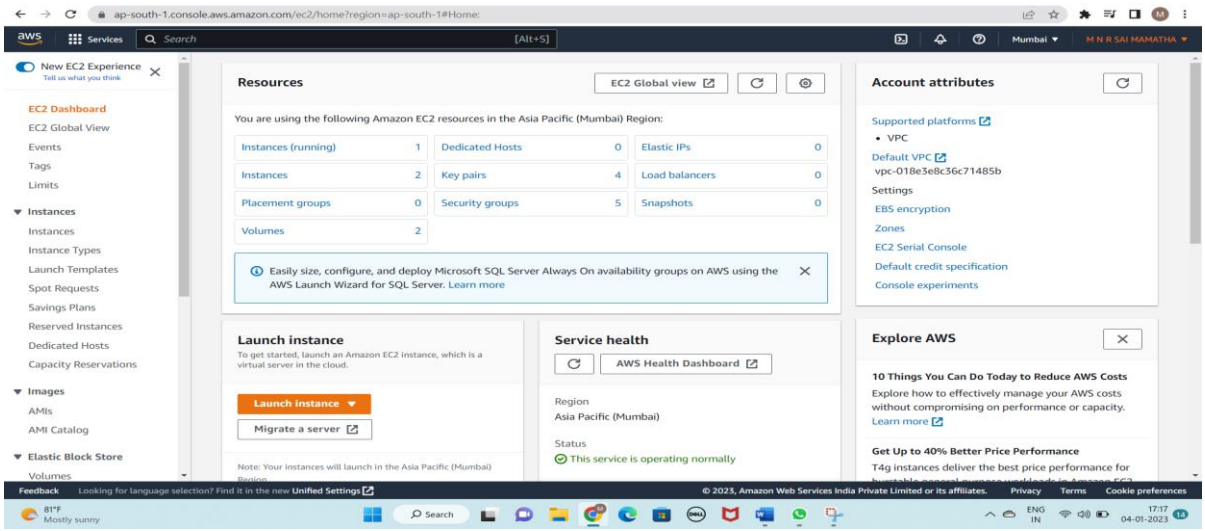


Figure 2: EC2 Dashboard

In ***Launch an instance*** page ,as shown in Figure 3, enter name as “nodejswebapp” under ***Name and tags***. We have to choose ***Amazon Machine Image*** for launching our instance. Amazon provides various types of images.

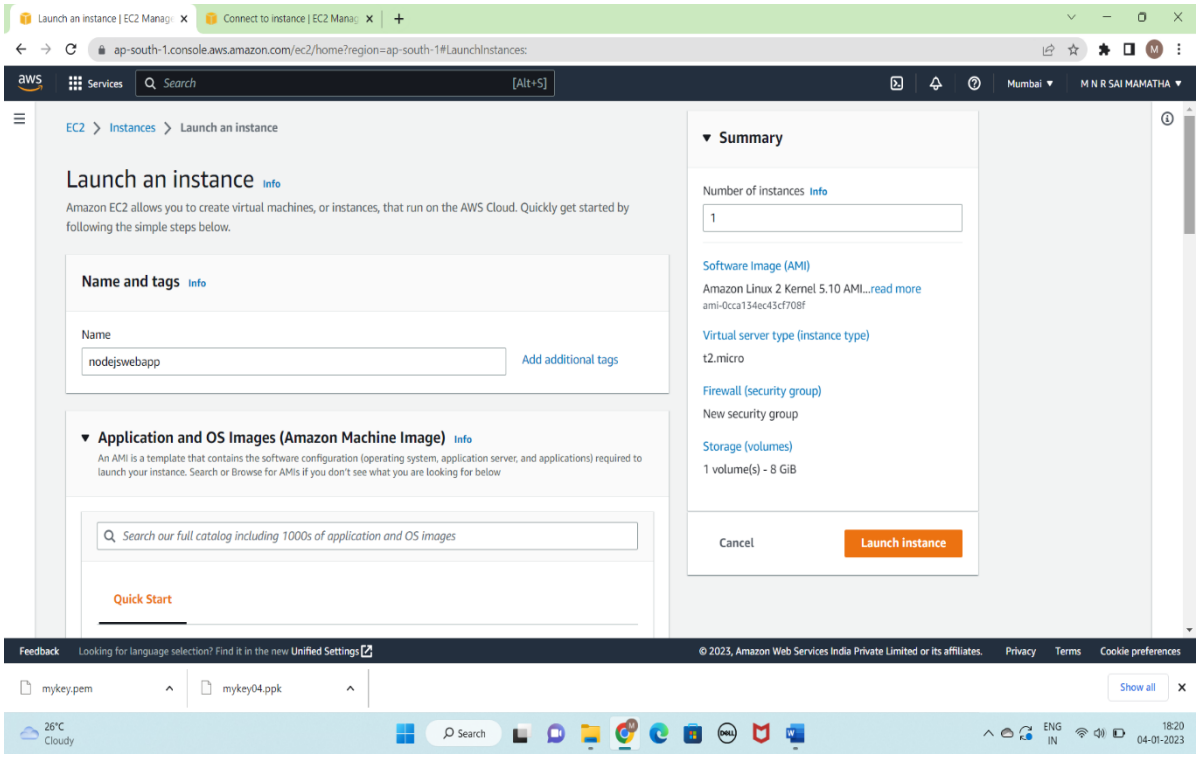


Figure 3: Launch an instance

Create a new key pair and then click on ***Launch Instance*** button.

3. **SSH into your instance**:

So we have successfully launched our EC2 server. Now we will access the server from our local machine using putty on Windows. If you are using Windows you need to change your **.pem** key to **.ppk** format.

To convert .pem file to .ppk file, download puttygen.exe. Once downloaded open the PuTTY Key Generator and load the downloaded the .pem file and then click on save private key. Now open the putty and enter the public IP of the instance created. Select SSH, then click on Auth and then browse the saved .ppk file and click on open. Login as ec2-user.

4. **Install Node.js:**

Install Node.js on putty using the following commands-

- sudo -i
- yum install gcc-c++ make
- curl -sL https://rpm.nodesource.com/setup_16.x | sudo -E bash –
- yum install nodejs

5. **Install Git and clone repository from GitHub:**

Use the following commands to install git and for cloning repository from GitHub-

- yum install git
- git clone https://github.com/johnpapa/node-hello.git

After cloning the repository, run cd node-hello and see all the content of the folder using the ls command as shown in Figure 4.

```
[root@ip-172-31-44-226 ~]# cd node-hello
[root@ip-172-31-44-226 node-hello]# ls
index.js  package.json  package-lock.json  README.md
```

Figure 4: Contents of node-hello directory

6. **Start the node.js app:**

We have successfully cloned the app on our server. Run the commands as shown in Figure 5, inside our project directory:

- npm install

It will install all the required packages for the app. There are several ways to start our app but we will use simple command:

- node index.js

```
[root@ip-172-31-44-226 node-hello]# npm install
npm WARN ancient lockfile
npm WARN ancient lockfile The package-lock.json file was created with an old ver
sion of npm,
npm WARN ancient lockfile so supplemental metadata must be fetched from the regi
stry.
npm WARN ancient lockfile
npm WARN ancient lockfile This is a one-time fix-up, please be patient...
npm WARN ancient lockfile

up to date, audited 1 package in 215ms

found 0 vulnerabilities
[root@ip-172-31-44-226 node-hello]# node index.js
Server running on http://localhost:3000/
```

Figure 5: Starting the node.js app

Copy public DNS from your instances page – "http://ec2-13-234-112-221.ap-south-1.compute.amazonaws.com", paste it on your web browser, and append port :3000 at the end of the DNS address. As you will see this will not work and there is nothing to show on browser and the browser connection will timeout as shown in Figure 6.

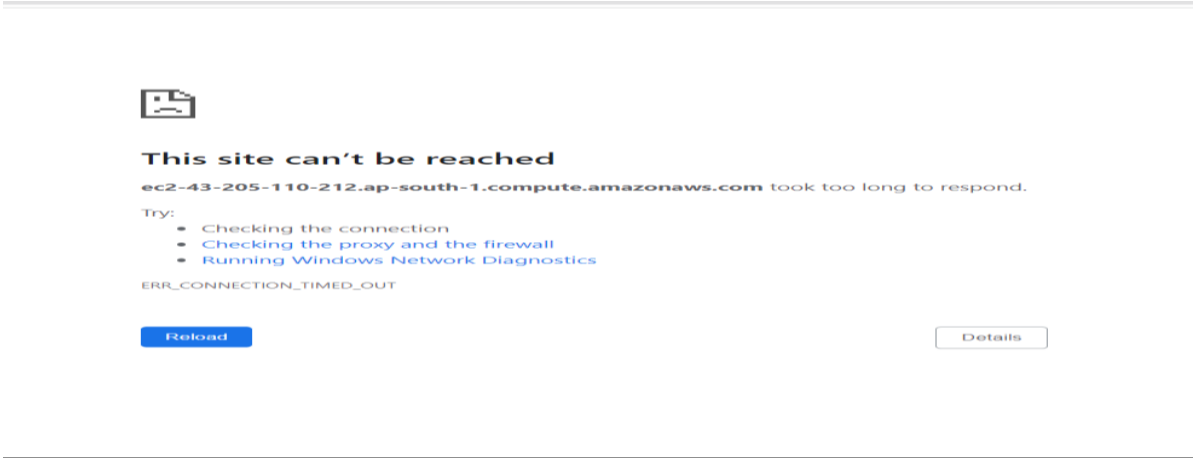


Figure 6: Browser connection timeout

This is where the Security Group comes in. In the AWS management console, go to ***Security Groups***, select the one not named 'default' (launch-wizard-1 or your security group name) and edit the ***Inbound*** rules as shown in Figure 7.

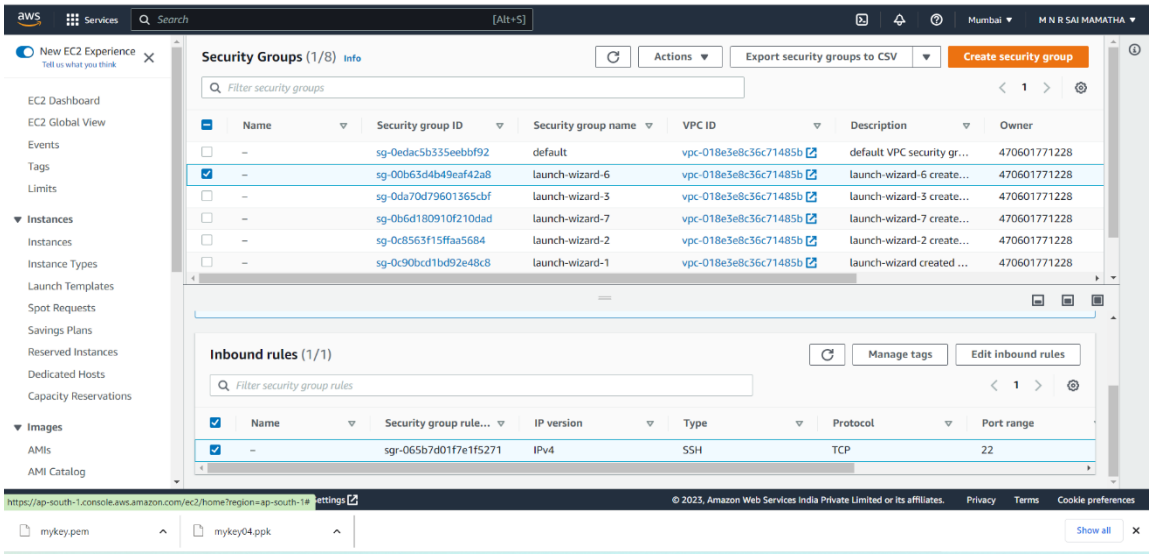


Figure 7: Security Groups

Access the URL again and you should be able to see the app content now.

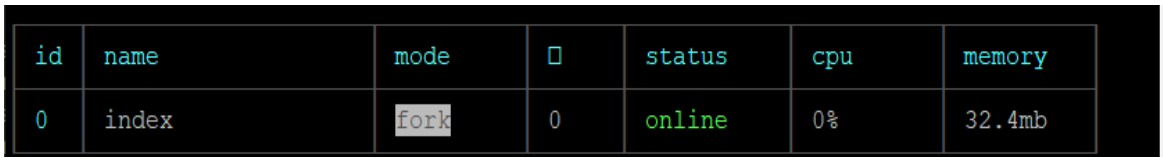
7. **Keep App running using Pm2:**

The app is running as soon as you open the terminal and it will terminate as you will close the terminal. We will install PM2 (Production manager 2) to keep live our app after closing our terminal or disconnect from remote server. Run the following command:

- npm install pm2 -g

It will install PM2 package globally on server. Switch to our app directory and run:

- sudo pm2 start index.js



id	name	mode	□	status	cpu	memory
0	index	fork	0	online	0%	32.4mb

Figure 8: List of running applications

Now even if you close the terminal and check the url in the browser, the app will be running. For automatically running PM2 when the server restarts, issue the following command:

- sudo pm2 startup

Now you can reboot the instance using sudo reboot and connect after 1 min. You can still see the app is running on port 3000 using:

- pm2 list
- pm2 show app

Source file: index.js

```
const http = require('http');
const port = process.env.PORT || 3000;
const server = http.createServer((req, res) => {
res.statusCode = 200;
const msg = 'Hello Node!\n'
res.end(msg);
});
server.listen(port, () => {
console.log(`Server running on http://localhost:${port}/`);
});
```

Output : Hello Node!

Screenshot of output

Enter “http://ec2-13-234-112-221.ap-south-1.compute.amazonaws.com:3000/” in the web browser and the output is as shown in Figure – 9.

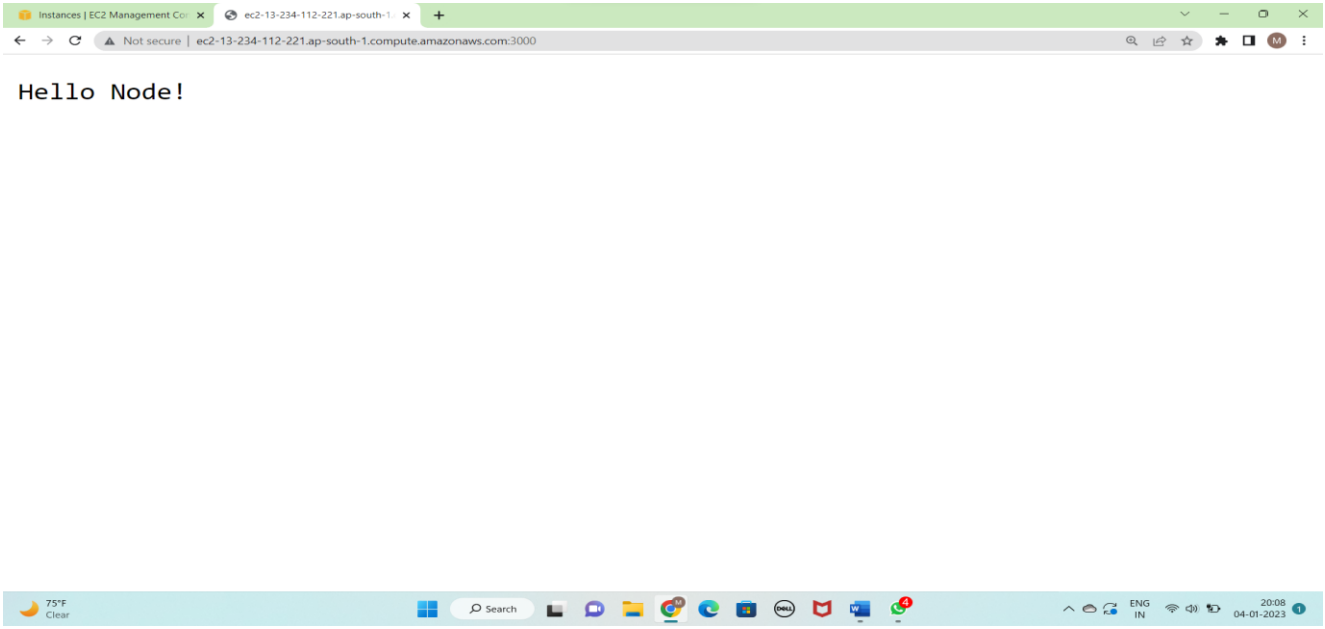


Figure 9 : Output of deployed node.js web application