MEAN Stack Deployment to Ubuntu in AWS

Introduction

This project focuses on deploying a MEAN (MongoDB, Express.js, Angular, Node.js) stack on an Ubuntu server in AWS, enabling a scalable and secure cloud-based web application. The MEAN stack is a widely adopted JavaScript-based framework that provides a seamless development experience by using a single programming language across the entire application. This full-stack architecture is well-suited for building dynamic, data-driven applications, offering high flexibility, real-time capabilities, and efficient performance.

By leveraging AWS cloud infrastructure, this deployment ensures high availability, scalability, and security while optimizing server resources. The combination of MongoDB's NoSQL database, Express.js for backend development, Angular for frontend interactivity, and Node.js for server-side execution makes the MEAN stack a powerful choice for modern web applications.

Understanding the MEAN Stack

MongoDB – A NoSQL database for storing data in JSON-like documents.

Express.js – A lightweight backend framework for building web applications and APIs.

Angular – A frontend framework for creating dynamic and interactive web applications.

Node.js – A JavaScript runtime for executing server-side applications efficiently.

Project Objectives

i. To deploy a Fully Functional MEAN Stack Application on AWS.
ii. To install and configure MEAN Stack Components.
iii. To develop a RESTful API for CRUD Operations (POST, GET, PUT and DELETE).
iv. To test and deploy the Application for Public Access.

Project Steps

Step 1: Spin up an Ubuntu server and SSH into the server

Step 2: Install Node.js

Step 3: Install MongoDB

Step 4: Install Express and set up routes to the server

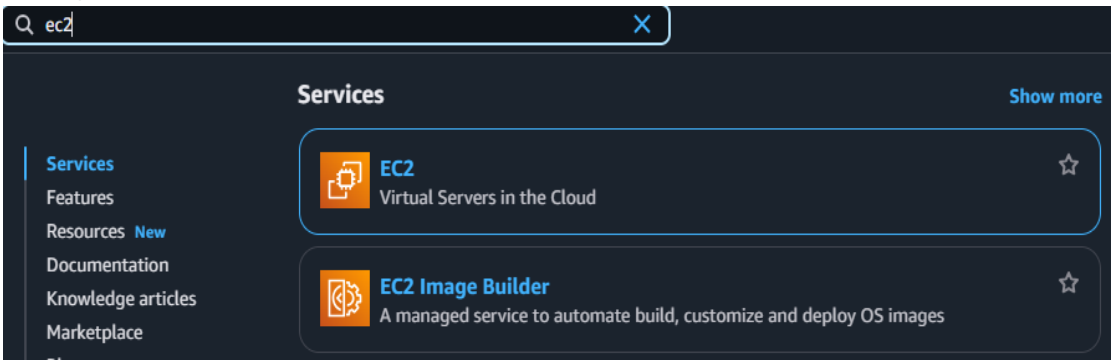
Step 5: Access the routes with Angular.js

Project Implementation

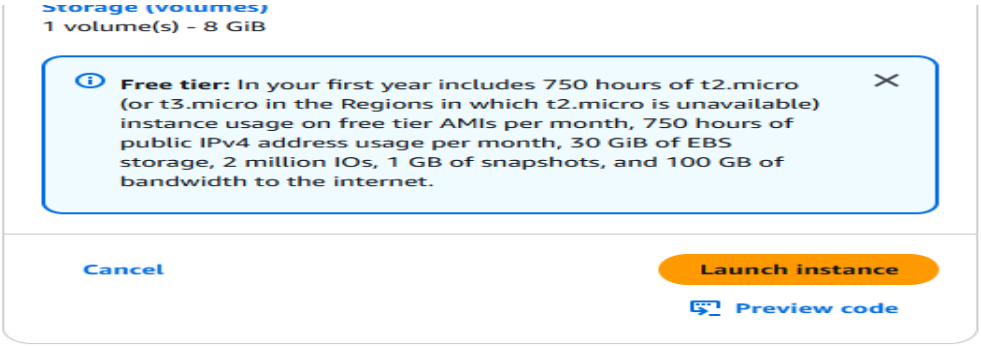
Step 1: Spin up an Ubuntu server and SSH into the server

Ubuntu Server is an open-source Linux operating system based on Debian, designed specifically for server environments. It is widely used for web servers, database servers, cloud services, and more.

- Go to your AWS Console and search for ec2 instance and click on it



- Click Launch instance -> Give Instance a name -> Select ubuntu as your Application and OS image -> select t2.micro as your instance type -> create a new key pair -> click launch instance



Next, we would need to SSH (Secure Shell) into our newly created instance from our local host. To achieve this;

- Open your git bash and run this command to go to the directory where you save your key pair

cd Downloads

- Next we would need to change mode of our key pair. We always do this to enhance security. We would use 400 with means

  - ✓ 4 (read permission for the owner)
  - ✓ 0 (no permissions for the group)
  - ✓ 0 (no permissions for others)

  To change the mode of our keypair, run the command
  chmod 400 "Devops.pem"

```
MINGW64:/c/Users/Lenovo/Downloads

Lenovo@General MINGW64 ~
$ cd Downloads

Lenovo@General MINGW64 ~/Downloads
$ chmod 400 "Devops.pem"

Lenovo@General MINGW64 ~/Downloads
$ |
```

- To SSH into our instance, run command in this format
  ssh -i {your_key.pem} ubuntu@{your_instance_public_ip}
  The command would look like this
  ssh -i Devops.pem ubuntu@54.159.121.163

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-84-39:~$ |
```

Step 2: Install Node.js

Node.js is a powerful, open-source JavaScript runtime built on Chrome's V8 engine. It's primarily used for building scalable network applications, especially web servers and real-time applications like chat apps, APIs, and more.

- Update your terminal. Run;
  sudo apt update

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2079 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2316 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [387 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2944 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [515 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1187 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [291 kB]
```

- Install curl if not already installed. Curl is a command-line tool used to transfer data to or from a server using various protocols like HTTP, HTTPS, FTP, etc. Run;
  sudo apt install curl

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.20).
curl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
ubuntu@ip-172-31-88-81:~$
```

- Use the following command to fetch and set up the NodeSource repository. NodeSource repository is a trusted external package repository that allows you to install and update Node.js on your system more easily and ensures you get the latest stable versions of Node.js
  curl -sL https://deb.nodesource.com/setup_lts.x | sudo -E bash -

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ curl -sL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
2025-02-16 20:10:43 - Installing pre-requisites
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.20).
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
```

- Install Node.js
  sudo apt install nodejs

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 35 not upgraded.
Need to get 36.4 MB of archives.
After this operation, 223 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_22.x nodistro/main amd64 nodejs amd64 22.14.0-1nodesource1 [36.4 MB]
Fetched 36.4 MB in 1s (63.8 MB/s)
Selecting previously unselected package nodejs.
(Reading database ... 65857 files and directories currently installed.)
Preparing to unpack .../nodejs_22.14.0-1nodesource1_amd64.deb ...
Unpacking nodejs (22.14.0-1nodesource1) ...
Setting up nodejs (22.14.0-1nodesource1) ...
```

- Verify node.js is installed correctly. Run;
  node -v

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ node -v
v22.14.0
ubuntu@ip-172-31-88-81:~$
```

Step 2: Install MongoDB

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents, allowing for dynamic schemas and scalable, high-performance data storage and retrieval.

For our example application, we are adding book records to MongoDB that contain book name, isbn number, author, and number of pages.

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F3730359A14518585931BC711F9BA15703C6

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F3730359A1451858
5931BC711F9BA15703C6
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
Executing: /tmp/apt-key-gpghome.qea1du9lXM/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F373035
9A14518585931BC711F9BA15703C6
gpg: key BC711F9BA15703C6: public key "MongoDB 3.4 Release Signing Key <packaging@mongodb.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

echo "deb [arch=amd64] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4 mu
ltiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4 multiverse
ubuntu@ip-172-31-88-81:~$
```

- Install MongoDB. Run;
  sudo apt-get install -y mongodb-org

```
ubuntu@ip-172-31-88-81:~$ sudo apt-get install -y mongodb-org
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  mongodb-database-tools mongodb-mongosh mongodb-org-database mongodb-org-database-tools-extra
  mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
The following NEW packages will be installed:
  mongodb-database-tools mongodb-mongosh mongodb-org mongodb-org-database mongodb-org-database-tools-extra
  mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
0 upgraded, 9 newly installed, 0 to remove and 35 not upgraded.
Need to get 164 MB of archives.
After this operation, 513 MB of additional disk space will be used.
Get:1 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0/multiverse amd64 mongodb-database-tools amd64 10
0.11.0 [55.7 MB]
Get:2 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0/multiverse amd64 mongodb-mongosh amd64 2.3.9 [54
.1 MB]
Get:3 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0/multiverse amd64 mongodb-org-shell amd64 6.0.20
[2984 B]
Get:4 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0/multiverse amd64 mongodb-org-server amd64 6.0.20
 [31.7 MB]
Get:5 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0/multiverse amd64 mongodb-org-mongos amd64 6.0.20
```

- Start, enable and check the status of MongoDB

  sudo systemctl start mongod
  sudo systemctl enable mongod
  sudo systemctl status mongod

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ sudo systemctl start mongod
ubuntu@ip-172-31-88-81:~$ sudo systemctl enable mongod
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service → /lib/systemd/system/mongod.service.
ubuntu@ip-172-31-88-81:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-02-16 20:46:17 UTC; 21s ago
       Docs: https://docs.mongodb.org/manual
   Main PID: 5786 (mongod)
     Memory: 76.7M
        CPU: 852ms
     CGroup: /system.slice/mongod.service
             └─5786 /usr/bin/mongod --config /etc/mongod.conf

Feb 16 20:46:17 ip-172-31-88-81 systemd[1]: Started MongoDB Database Server.
Feb 16 20:46:17 ip-172-31-88-81 mongod[5786]: {"t":{"$date":"2025-02-16T20:46:17.590Z"},"s":"I",  "c":"CONTROL"
lines 1-12/12 (END)
```

- We need 'body-parser' package to help us process JSON files passed in requests to the server.

  sudo npm install body-parser

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ sudo npm install body-parser

added 41 packages in 4s

11 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.9.2 -> 11.1.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.1.0
npm notice To update run: npm install -g npm@11.1.0
npm notice
ubuntu@ip-172-31-88-81:~$
```

- Create a folder named 'Books' and change directory into the Books folder

  mkdir Books
  cd Books

```
ubuntu@ip-172-31-88-81: ~/Books
ubuntu@ip-172-31-88-81:~$ mkdir Books
ubuntu@ip-172-31-88-81:~$ cd Books/
ubuntu@ip-172-31-88-81:~/Books$
```

- Initialize npm project. Run;
  npm init

```
ubuntu@ip-172-31-88-81: ~/Books
ubuntu@ip-172-31-88-81:~/Books$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (books)
```

- Add a file to it named server.js
  touch server.js

- Copy and paste the web server code below into the server.js file using ***sudo vi server.js***

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(express.static(__dirname + '/public'));
app.use(bodyParser.json());
require('./apps/routes')(app);
app.set('port', 3300);
app.listen(app.get('port'), function() {
    console.log('Server up: http://localhost:' + app.get('port'));
});
```

```
ubuntu@ip-172-31-88-81: ~/Books
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(express.static(__dirname + '/public'));
app.use(bodyParser.json());
require('./apps/routes')(app);
app.set('port', 3300);
app.listen(app.get('port'), function() {
    console.log('Server up: http://localhost:' + app.get('port'));
});
~
~
~
~
```

Step 4: Install Express and Set Up Routes to the Server

Express.js is a back-end web application framework for Node.js that simplifies building web servers and APIs. It provides a lightweight and flexible way to handle HTTP requests, manage middleware, and define routes. Express is widely used in full-stack development, where it serves as the server-side component. It allows developers to build RESTful APIs, handle authentication, connect to databases, and serve static files efficiently.

- To install express, run;
  sudo npm install express mongoose

```
ubuntu@ip-172-31-88-81: ~
ubuntu@ip-172-31-88-81:~$ sudo npm install express mongoose

added 48 packages, and audited 90 packages in 3s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ubuntu@ip-172-31-88-81:~$
```

- In the book folder, create a folder named app. In the app folder, create a file named routes.js
  cd Books
  mkdir app
  cd app
  touch routes.js

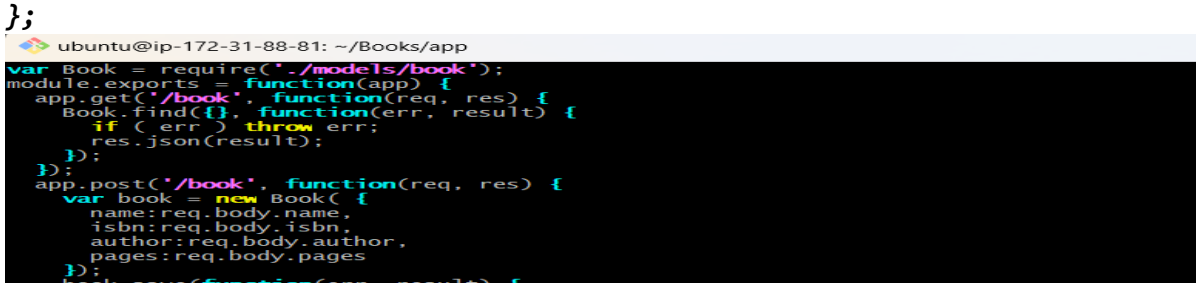
```
ubuntu@ip-172-31-88-81: ~/Books/app
ubuntu@ip-172-31-88-81:~$ cd Books/
ubuntu@ip-172-31-88-81:~/Books$ ls
server.js
ubuntu@ip-172-31-88-81:~/Books$ mkdir app
ubuntu@ip-172-31-88-81:~/Books$ cd app
ubuntu@ip-172-31-88-81:~/Books/app$ touch routes.js
ubuntu@ip-172-31-88-81:~/Books/app$
```

- Open the routes.js file so you can paste some items. Run;
  sudo vi routes.js

```
ubuntu@ip-172-31-88-81: ~/Books/app
~
~
~
~
~
~
~
~
~
~
~
```

- Paste the below in the routes.js file

```
var Book = require('./models/book');
module.exports = function(app) {
  app.get('/book', function(req, res) {
    Book.find({}, function(err, result) {
      if ( err ) throw err;
      res.json(result);
    });
  });
  app.post('/book', function(req, res) {
    var book = new Book( {
      name:req.body.name,
      isbn:req.body.isbn,
      author:req.body.author,
      pages:req.body.pages
    });
    book.save(function(err, result) {
      if ( err ) throw err;
      res.json( {
        message:"Successfully added book",
        book:result
      });
    });
  });
  app.delete("/book/:isbn", function(req, res) {
    Book.findOneAndRemove(req.query, function(err, result) {
      if ( err ) throw err;
      res.json( {
        message: "Successfully deleted the book",
        book: result
      });
    });
  });
  var path = require('path');
  app.get('*', function(req, res) {
    res.sendFile(path.join(__dirname + '/public', 'index.html'));
  });
};
```



- In the app folder, create a folder named models. In the models folder, create a file named book.js

```
mkdir models
cd models
touch book.js
```

```
ubuntu@ip-172-31-88-81: ~/Books/app/models
ubuntu@ip-172-31-88-81:~/Books/app$ sudo vi routes.js
ubuntu@ip-172-31-88-81:~/Books/app$ pwd
/home/ubuntu/Books/app
ubuntu@ip-172-31-88-81:~/Books/app$ ls
routes.js
ubuntu@ip-172-31-88-81:~/Books/app$ mkdir models
ubuntu@ip-172-31-88-81:~/Books/app$ ls
models  routes.js
ubuntu@ip-172-31-88-81:~/Books/app$ cd models/
ubuntu@ip-172-31-88-81:~/Books/app/models$ touch book.js
ubuntu@ip-172-31-88-81:~/Books/app/models$ ls
book.js
ubuntu@ip-172-31-88-81:~/Books/app/models$
```

- Paste the below in the book.js file

```
var mongoose = require('mongoose');
var dbHost = 'mongodb://localhost:27017/test';
mongoose.connect(dbHost);
mongoose.connection;
mongoose.set('debug', true);
var bookSchema = mongoose.Schema( {
  name: String,
  isbn: {type: String, index: true},
  author: String,
  pages: Number
});
var Book = mongoose.model('Book', bookSchema);
module.exports = mongoose.model('Book', bookSchema);
```

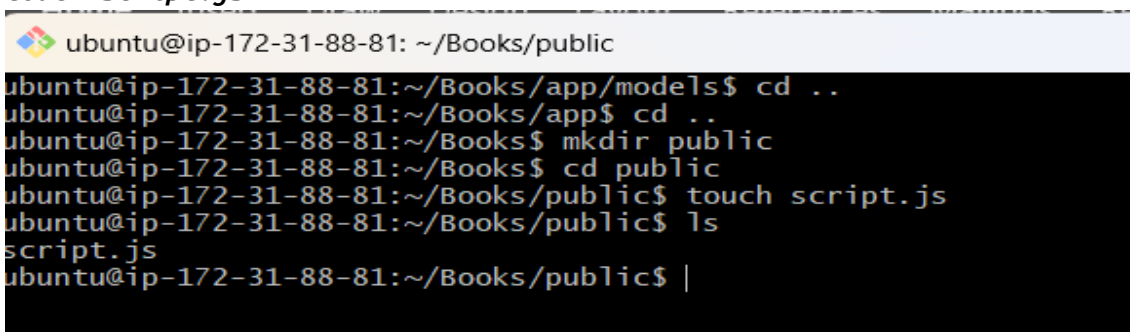
```
ubuntu@ip-172-31-88-81: ~/Books/app/models
var mongoose = require('mongoose');
var dbHost = 'mongodb://localhost:27017/test';
mongoose.connect(dbHost);
mongoose.connection;
mongoose.set('debug', true);
var bookSchema = mongoose.Schema( {
  name: String,
  isbn: {type: String, index: true},
  author: String,
  pages: Number
});
var Book = mongoose.model('Book', bookSchema);
module.exports = mongoose.model('Book', bookSchema);
~
~
```

Step 5: Access the routes with Angular.js

Angular is a web framework for building dynamic, modular, and scalable single-page applications (SPAs) using TypeScript and component-based architecture.

- Go back to the apps Directory. In that directory, create a directory called public. In the public directory, create a file named script.js

```
cd ..
mkdir public
cd public
touch script.js
```



- Paste the below in the script.js file

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
  $http( {
    method: 'GET',
    url: '/book'
  }).then(function successCallback(response) {
    $scope.books = response.data;
  }, function errorCallback(response) {
    console.log('Error: ' + response);
  });
  $scope.del_book = function(book) {
    $http( {
      method: 'DELETE',
      url: '/book/:isbn',
      params: {'isbn': book.isbn}
    }).then(function successCallback(response) {
      console.log(response);
    }, function errorCallback(response) {
      console.log('Error: ' + response);
    });
  };
  $scope.add_book = function() {
    var body = '{ "name": "' + $scope.Name +
    '", "isbn": "' + $scope.Isbn +
    '", "author": "' + $scope.Author +
```

```
    '", "pages": "' + $scope.Pages + '" }';
    $http({
      method: 'POST',
      url: '/book',
      data: body
    }).then(function successCallback(response) {
      console.log(response);
    }, function errorCallback(response) {
      console.log('Error: ' + response);
    });
  };
});
```

ubuntu@ip-172-31-88-81: ~/Books/public

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
  $http( {
    method: 'GET',
    url: '/book'
  }).then(function successCallback(response) {
    $scope.books = response.data;
  }, function errorCallback(response) {
    console.log('Error: ' + response);
  });
  $scope.del_book = function(book) {
    $http( {
      method: 'DELETE',
      url: '/book/:isbn',
      params: {'isbn': book.isbn}
    }).then(function successCallback(response) {
```

- In public folder, create a file named index.html

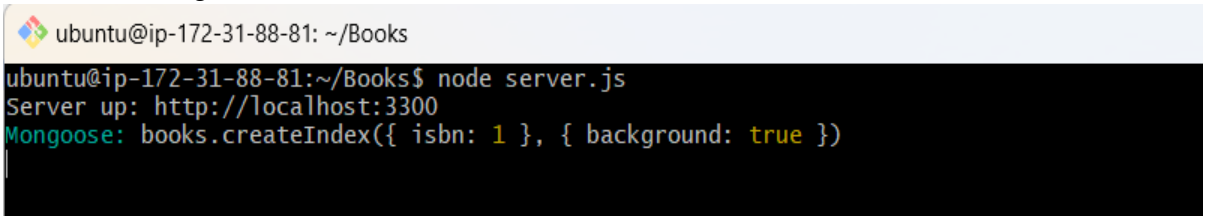
```
vi index.html
Copy and paste the code below into index.html file.
<!doctype html>
<html ng-app="myApp" ng-controller="myCtrl">
  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.
js"></script>
    <script src="script.js"></script>
  </head>
  <body>
    <div>
      <table>
        <tr>
          <td>Name:</td>
          <td><input type="text" ng-model="Name"></td>
        </tr>
        <tr>
          <td>Isbn:</td>
          <td><input type="text" ng-model="Isbn"></td>
        </tr>
```

```
        <tr>
          <td>Author:</td>
          <td><input type="text" ng-model="Author"></td>
        </tr>
        <tr>
          <td>Pages:</td>
          <td><input type="number" ng-model="Pages"></td>
        </tr>
      </table>
      <button ng-click="add_book()">Add</button>
    </div>
    <hr>
    <div>
      <table>
        <tr>
          <th>Name</th>
          <th>Isbn</th>
          <th>Author</th>
          <th>Pages</th>

        </tr>
        <tr ng-repeat="book in books">
          <td>{{book.name}}</td>
          <td>{{book.isbn}}</td>
          <td>{{book.author}}</td>
          <td>{{book.pages}}</td>

          <td>
<input type="button" value="Delete" data-ng-click="del_book(book)"></td>
        </tr>
      </table>
    </div>
  </body>
</html>
```

ubuntu@ip-172-31-88-81: ~/Books/apps/public

```
<!doctype html>
<html ng-app="myApp" ng-controller="myCtrl">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
    <script src="script.js"></script>
  </head>
  <body>
    <div>
      <table>
        <tr>
          <td>Name:</td>
          <td><input type="text" ng-model="Name"></td>
        </tr>
        <tr>
          <td>Isbn:</td>
          <td><input type="text" ng-model="Isbn"></td>
        </tr>
```

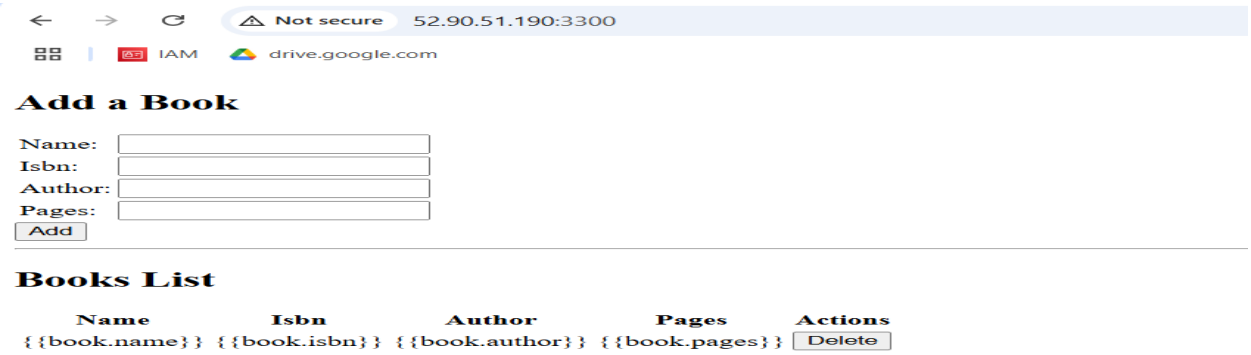
- Change the directory back up to Books Start the server by running this command;
  node server.js



The server is now up and running, we can connect it via port 3300. We can also try and access it from the Internet. For this – you need to open TCP port 3300 in your AWS Web Console for your EC2 Instance.

Access the web page from your web browser with this

http://{ip_address}/3300



Conclusion

This project successfully deployed a MEAN stack application on an Ubuntu server in AWS, leveraging MongoDB, Express.js, Angular, and Node.js to create a scalable, dynamic web app. The setup involved installing and configuring the necessary components, developing a RESTful API for CRUD operations, and making the app publicly accessible. Using AWS ensures high availability and security, making this deployment suitable for real-world applications. Overall, the project achieved its goals of deploying a fully functional MEAN stack app in the cloud, providing a strong foundation for future development.