

MERN STACK PROJECT IMPLEMENTATION ON A SIMPLE TO-DO APPLICATION

The main aim for this project is to explain the DevOps concepts and processes using a MERN web stack on a simple to-do application. Some developers use this set of framework and tools to develop software products. We would be carrying out this project in the AWS platform.

MERN is an acronym of sets of technologies used to develop a technical software product.

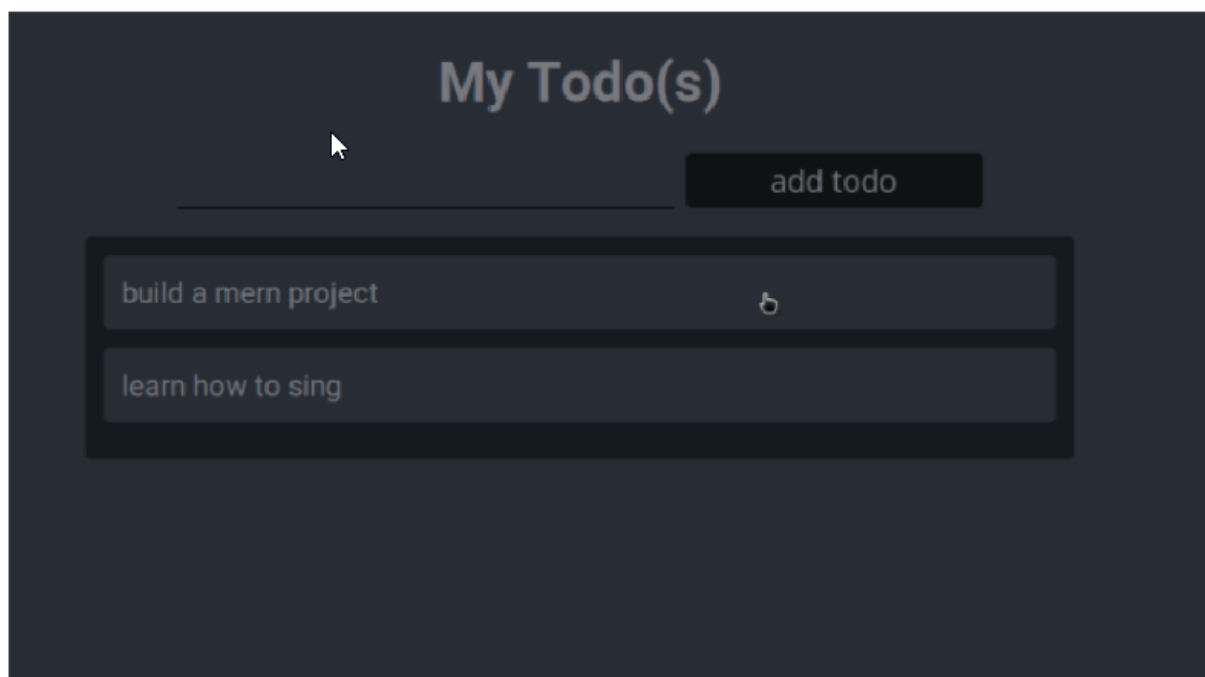
MongoDB

Express

ReactJS

NodeJS

PROJECT:



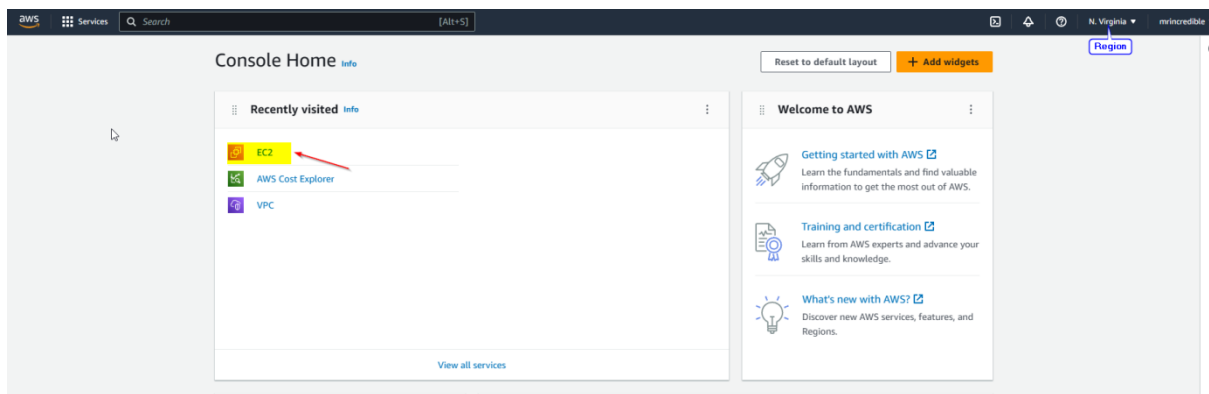
Pre-requisite for the projects is the following.

- 1) Fundamental Knowledge of Installing and downloading software
- 2) Basic Understanding of Linux Commands
- 3) AWS account login with EC2 instance
- 4) Internet connection

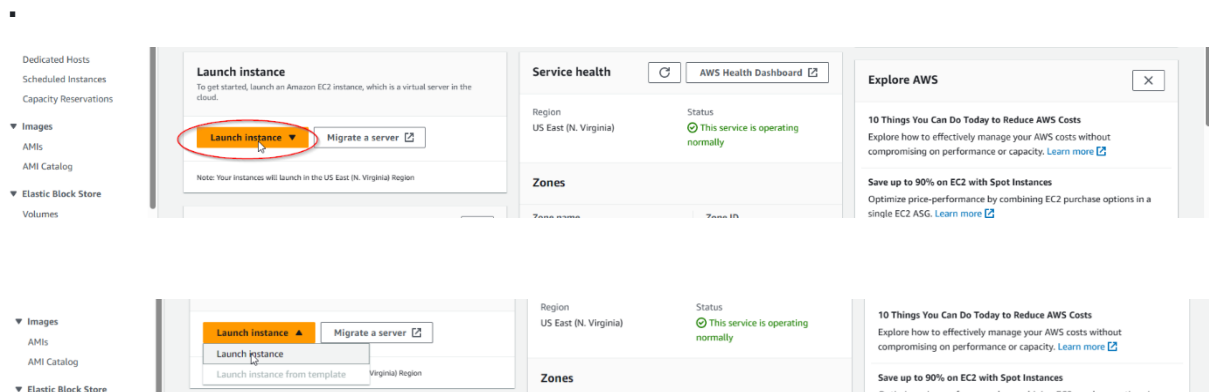
IMPLEMENTATION STEPS:

- i) Ensure you login with your details to your AWS console via the <https://aws.amazon.com>

ii) Click on the EC2 link to create instances.



iii) Click on launch instance dropdown button and select launch instance



iv) Fill in all relevant details to the LEMP project such as:

Type in the name and additional tag to the project (mern). Select ubuntu from the quick start option .Also note that the Amazon machine image selection varies from user to user

Select Ubuntu server 22.04 LTS (HVM),SSD Volume Type (Free Tier)

Name

Mem
Add additional tags

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

S

aws

Mac

ubuntu

Microsoft

Red Hat

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type
ami-0261755bbcb8c4a84 (64-bit (x86)) / ami-097d5b19d4f1a7d1b (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)
Canonical, Ubuntu, 20.04 LTS, ...[read more](#)
ami-0261755bbcb8c4a84

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

v)The instance type selected in the configuration is the t2 micro -free tier.

Click on the “Create new key pair” link.

Ensure the Checkbox remains unchanged on the “Create security group”.

Instance type

t2.micro
Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour

All generations
Compare instance types

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Select
Create new key pair

▼ Network settings [Info](#)

Network [Info](#)
vpc-0c3c371436c0dcd9d

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-18' with the following rules:

Allow SSH traffic from
Helps you connect to your instance

Anywhere
0.0.0.0/0

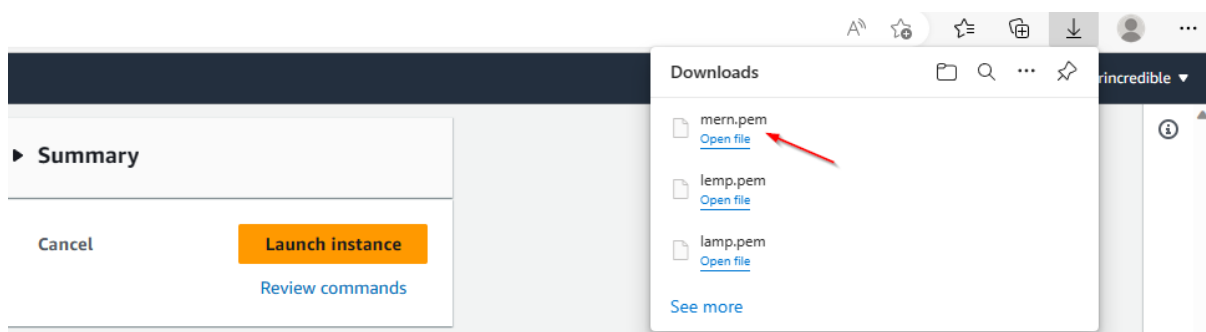
Cancel
Launch instance
Review commands

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

vi) Type in the key pair name, chose the default key pair type and private key file format (rsa and .pem) and clicked the “Create key pair button”

The screenshot shows the 'Create key pair' dialog box. At the top, the title is 'Create key pair' with a close button. Below it, the 'Key pair name' section has a text input field containing 'mern' and a red arrow pointing to it. A note below the field states: 'The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.' The 'Key pair type' section has two options: 'RSA' (selected with a radio button and highlighted in yellow) and 'ED25519'. The 'Private key file format' section has two options: '.pem' (selected with a radio button and highlighted in yellow) and '.ppk'. At the bottom, there is a warning box with a red triangle icon and text: 'When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. Learn more'. Below the warning box are two buttons: 'Cancel' and 'Create key pair' (highlighted in orange with a red arrow pointing to it).

vii) The .pem file was downloaded successfully



viii) I have deliberately chosen default settings to allow SSH traffic from anywhere as well as the storage volume given by AWS.

Then we proceed to launch our instance finally.

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-18' with the following rules:

☒ Allow SSH traffic from

Helps you connect to your instance

Anywhere

0.0.0.0/0

☐ Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

☐ Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

×

▼ Configure storage

Info

Advanced

1 x

8

GiB

gp2

Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

×

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems

Edit

► Summary

Cancel

Launch instance

Review commands

Instance successfully launched and click to view Instance details with the IP address.

EC2 > Instances > Launch an instance

✓

Success

Successfully initiated launch of instance (i-08b30f6a7ffcdd456)

▼ Launch log

Initializing requests

Succeeded

Creating security groups

Succeeded

Creating security group rules

Succeeded

Launch initiation

Succeeded

Instances (1/10) Info

Find instance by attribute or tag (case-sensitive)

Connect Instance state Actions Launch instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
lemp	i-03a0b1f1478866b9a	Stopped	t2.micro	-	No alarms	us-east-1b	-	-	-
Mem	i-08b30f6a7ffcdd456	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-100-24-45-95.com...	100.24.45.95	-
nginx-server	i-09924c00df90fa7a9	Stopped	t2.micro	-	No alarms	us-east-1b	-	-	-
Demo-Apache2	i-01fdd0d21739d518	Stopped	t2.micro	-	No alarms	us-east-1a	-	-	-

Instance: i-08b30f6a7ffcdd456 (Mem)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary Info

Instance ID
i-08b30f6a7ffcdd456 (Mem)

IPv6 address
-

Hostname type
IP name: in-172-31-86-218 ar? Internal

Public IPv4 address
100.24.45.95 | open address

Instance state
Running

Private IP DNS name (IPv4 only)
in-172-31-86-218 ar? Internal

Private IPv4 addresses
172.31.86.218

Public IPv4 DNS
ec2-100-24-45-95.compute-1.amazonaws.com | open address

Click the “Connect” button and copy the ssh client details we would be using on the git bash console.

Instances (1/10) Info

Find instance by attribute or tag (case-sensitive)

Connect Instance state Actions Launch instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
lemp	i-03a0b1f1478866b9a	Stopped	t2.micro	-	No alarms	us-east-1b	-	-	-
Mem	i-08b30f6a7ffcdd456	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-100-24-45-95.com...	100.24.45.95	-

Open git bash on visual studio code or whichever console is convenient to use. We are using git bash here with Visual Studio Code

```
oshor@Oshority MINGW64 ~ (master)
$ cd Downloads
oshor@Oshority MINGW64 ~/Downloads (master)
$ ssh -i "mern.pem" ubuntu@ec2-100-24-45-95.compute-1.amazonaws.com
```

Type YES to connect.

```
$ ssh -i mern.pem ubuntu@ec2-100-24-45-95.compute-1.amazonaws.com
The authenticity of host 'ec2-100-24-45-95.compute-1.amazonaws.com (100.24.45.95)' can't be established.
ED25519 key fingerprint is SHA256:IMiN4aA13sgYvUXUmnlSscEJCcj1usuFhvqmOLX/9aM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

You have successfully connected to the EC2 instance launched on AWS via ssh

Type clear to have a clear console and proceed to updating the lists of packages in the package manager.

```

1
ubuntu@ip-172-31-86-218:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]

```

Then we proceed to upgrade the packages and Type YES to continue.

```

ubuntu@ip-172-31-86-218:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done

```

```

Need to get 89.0 MB of archives.
After this operation, 242 MB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

Still upgrading

```

1
root@kali:~# [#####/etc/kernel/postinst.d/zz-update-grub:#####....]
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/40-force-partuuid.cfg'
Sourcing file '/etc/default/grub.d/50-cloudimg-settings.cfg'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
GRUB_FORCE_PARTUUID is set, will attempt initrdless boot
Found linux image: /boot/vmlinuz-5.15.0-1037-aws
Found initrd image: /boot/microcode.cpio /boot/initrd.img-5.15.0-1037-aws
Found linux image: /boot/vmlinuz-5.15.0-1036-aws
Found initrd image: /boot/microcode.cpio /boot/initrd.img-5.15.0-1036-aws
Found Ubuntu 20.04.6 LTS (20.04) on /dev/xvda1
done
ubuntu@ip-172-31-86-218:~$

```

Now we need to get the location of Node.js software from the ubuntu repositories by typing the command below.

```

ubuntu@ip-172-31-86-218:~$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -

## Installing the NodeSource Node.js 18.x repo...

## Populating apt-get cache...

+ apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease

```

We can now install Node.js on the server and confirm the versions of the node and npm package managers as shown below

```

ubuntu@ip-172-31-86-218:~$ sudo apt-get install -y nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 28.7 MB of archives.
After this operation, 187 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_18.x focal/main amd64 nodejs amd64 18.16.0-deb-1nodesource1 [28.7 MB]
Fetched 28.7 MB in 0s (67.9 MB/s)
Selecting previously unselected package nodejs.
(Reading database ... 90707 files and directories currently installed.)
Preparing to unpack .../nodejs_18.16.0-deb-1nodesource1_amd64.deb ...
Unpacking nodejs (18.16.0-deb-1nodesource1) ...
Setting up nodejs (18.16.0-deb-1nodesource1) ...
Processing triggers for man-db (2.9.1-1) ...
ubuntu@ip-172-31-86-218:~$ node -v
v18.16.0
ubuntu@ip-172-31-86-218:~$ npm -v
9.5.1

```

Once versions are confirmed. We created a directory called To-Do project and verify the directory is present .We then change directory with the `cd` command to the new directory we just created .

After which we initialize the project with the `npm init` command as seen below.

```
ubuntu@ip-172-31-86-218:~$ mkdir Todo
ubuntu@ip-172-31-86-218:~$ ls
Todo
ubuntu@ip-172-31-86-218:~$ cd Todo
ubuntu@ip-172-31-86-218:~/Todo$ npm init
```

The reason for this is to create a new `package.json` file .This files contains the application and its dependencies it needs to run and we need to press the Enter button several times to confirm the details we intend to use for its documentation and finally click “yes” to proceed .

```
Is this OK? (yes) yes
npm notice
npm notice New minor version of npm available! 9.5.1 -> 9.6.7
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.7
npm notice Run npm install -g npm@9.6.7 to update!
npm notice
ubuntu@ip-172-31-86-218:~/Todo$
```

An error appears which states we should install a new minor npm update from 9.51. to version 9.6.7 as seen below

```
ubuntu@ip-172-31-86-218:~/Todo$ sudo npm install -g npm@9.6.7

removed 1 package, and changed 62 packages in 3s

27 packages are looking for funding
  run `npm fund` for details
ubuntu@ip-172-31-86-218:~/Todo$
```

Lets run the `ls` command to confirm the `package .json` file is created.

```
ubuntu@ip-172-31-86-218:~/Todo$ ls
package.json
```

Next, we will install ExpressJS and create the Routes directory

INSTALL EXPRESSJS

Express is a framework for Node.js which further simplifies development and makes things work seamlessly. Express helps to define routes of the application based on HTTP methods and URL's.

We install the npm package modules for express and create an index file. Ensure to verify with an `ls` command to see the newly created `index.js` file.


```
ubuntu@ip-172-31-86-218:~/Todo$ npm install express
added 58 packages, and audited 59 packages in 2s

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

found 0 vulnerabilities
```

```
ubuntu@ip-172-31-86-218:~/Todo$ touch index.js
ubuntu@ip-172-31-86-218:~/Todo$ ls
index.js  node_modules  package-lock.json  package.json
```

We then install the dotenv module

```
ubuntu@ip-172-31-86-218:~/Todo$ npm install dotenv
added 1 package, and audited 60 packages in 714ms

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

found 0 vulnerabilities
```

Edit the index.js file with the vim command and type in the code below. Please take note of the port :5000 that was in the file .This would be required late when we want to get it working on the browser.

```
ubuntu@ip-172-31-86-218:~/Todo$ vim index.js
ubuntu@ip-172-31-86-218:~/Todo$
```

```
const express = require('express');
require('dotenv').config();

const app = express();

const port = process.env.PORT || 5000;

app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
  next();
});

app.use((req, res, next) => {
  res.send('Welcome to Express');
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`)
})
~
~
~
~
~
~
```

Save with the esc :wq Enter. Next step is to start our server to see if it works

```
ubuntu@ip-172-31-86-218:~/Todo$ vim index.js
ubuntu@ip-172-31-86-218:~/Todo$ node index.js
Server running on port 5000
```

Our server is running at port 5000. We need to click Ctrl C to exit from the message caption. We would need to create an inbound rule to open at port 5000

Click on security button

Instances (1/10) Info

Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
stackmean	i-09f1e479ad3e0bdce	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-
stacklemp	i-0862f845ba4a743f2	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-
stacklamp	i-03ac707da751f0f22	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-
mernstackmern	i-011b9e604e64ce6f4	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-

Instance: i-08b30f6a7ffcd456 (Mern)

Details Security Networking Storage Status checks Monitoring Tags

▼ Security details

IAM Role: -

Owner ID: 416591745024

Launch time: Thu Jun 01 2023 16:15:35 GMT+0100 (British Summer Time)

Security groups: sg-04d3aa35fbfeb428a (launch-wizard-18)

And click the security group link.

Security groups

sg-04d3aa35fbfeb428a (launch-wizard-18)

▼ Inbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-0c2da85afe8fb4424	22	TCP	0.0.0.0/0	launch-wizard-18	-

Click on “Edit inbound rules” in order to add a new rule for port 5000

Inbound rules (1/1)

Filter security group rules

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0c2da85afe8fb4424	IPv4	SSH	TCP	22	0.0.0.0/0	-

Edit inbound rules

Add rule

Add a new rule.

Inbound rules Info

Security group rule ID: sgr-0c2da85afe8fb4424

Type: SSH

Protocol: TCP

Port range: 22

Source: Custom 0.0.0.0/0

Description - optional: -

Add rule

Type in the port range and click “Anywhere ipv4”

-

Custom TCP

TCP

5000

Custom 0.0.0.0/0

Anywhere IPv4

Anywhere IPv6

My IP

Add rule

Cancel Preview changes Save rules

Click the “Save rules” Button.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Actions
sgr-0c2da85afe8fb4424	SSH	TCP	22	Custom		Delete
-	Custom TCP	TCP	5000	Anywhere...	Mem Project port 5000	Delete

Cancel Preview changes **Save rules**

Inbound rule successfully modified



**Open any browser of your choice and access the URL
<http://100.24.45.95:5000>**



ExpressJS default page successfully displayed.

From the MERN stack, we have implemented with Linux and now have Express ready

We would need to perform some actions in our simple to-do application which are the following:

- 1) Create a new task.**
- 2) Display all tasks.**
- 3) Delete all tasks.**

Please note that each tasks is associated with some particular endpoints and would use the standard HTTP request methods namely :POST ,GET AND DELETE

Each tasks would require us to create routes that defines the endpoints that the to-do application would depend on .

So we create a folder routes and change directory to the new folder

```
ubuntu@ip-172-31-86-218:~/Todo$ mkdir routes && cd routes
ubuntu@ip-172-31-86-218:~/Todo/routes$
```

We create a file called api.js

```
ubuntu@ip-172-31-86-218:~/Todo/routes$ touch api.js
ubuntu@ip-172-31-86-218:~/Todo/routes$ vim api.js
ubuntu@ip-172-31-86-218:~/Todo/routes$ vim api.js
ubuntu@ip-172-31-86-218:~/Todo/routes$
```

Edit the file and paste some code inside it . press ESC ,save and exit

```
const express = require ('express');
const router = express.Router();

router.get('/todos', (req, res, next) => {

});

router.post('/todos', (req, res, next) => {

});

router.delete('/todos/:id', (req, res, next) => {

})

module.exports = router;
```

with “ :wq

After this has been done, we would need to create models because we would be making use of a NoSQL database called MongoDB. These models function at the helm of JavaScript based application and makes it very interactive. Models are also used to define database schema which is the blueprint of how database are constructed including other data fields that aren't required to be stored in a database known as virtual properties.

To achieve this we need to install mongoose which is a node package.

Next step is to change directory back to Todo folder and install mongoose.

```
ubuntu@ip-172-31-86-218:~/Todo/routes$ cd ..
ubuntu@ip-172-31-86-218:~/Todo$ npm install mongoose
up to date, audited 84 packages in 787ms

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

found 0 vulnerabilities
```

Create models folder and change directory into the new folder

Create a todo.js file and edit the file and paste this code inside it .

press ESC ,save and exit with “ :wq” command

```
ubuntu@ip-172-31-86-218:~/Todo$ mkdir models && cd models
ubuntu@ip-172-31-86-218:~/Todo/models$ touch todo.js
ubuntu@ip-172-31-86-218:~/Todo/models$ vim todo.js
ubuntu@ip-172-31-86-218:~/Todo/models$
```

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

//create schema for todo
const TodoSchema = new Schema({
  action: {
    type: String,
    required: [true, 'The todo text field is required']
  }
})

//create model for todo
const Todo = mongoose.model('todo', TodoSchema);

module.exports = Todo;
~
~
```

Next we need to change the directory to update our routes and edit the api.js in the routes directory to make use of this model as illustrated below.

```
ubuntu@ip-172-31-86-218:~/Todo/models$ cd ..
ubuntu@ip-172-31-86-218:~/Todo$ cd routes/
ubuntu@ip-172-31-86-218:~/Todo/routes$ vim api.js
ubuntu@ip-172-31-86-218:~/Todo/routes$ vim api.js
ubuntu@ip-172-31-86-218:~/Todo/routes$
```

Replace this previous line of codes with the delete command :%d

```
const express = require ('express');
const router = express.Router();

router.get('/todos', (req, res, next) => {

});

router.post('/todos', (req, res, next) => {

});

router.delete('/todos/:id', (req, res, next) => {

})

module.exports = router;
```

With this new set of codes. Press ESC ,save and exit with “ :wq” command

```
router.get('/todos', (req, res, next) => {

//this will return all the data, exposing only the id and action field
to the client
  Todo.find({}, 'action')
    .then(data => res.json(data))
    .catch(next)
});

router.post('/todos', (req, res, next) => {
  if(req.body.action){
    Todo.create(req.body)
    .then(data => res.json(data))
    .catch(next)
  }else {
    res.json({
      error: "The input field is empty"
    })
  }
});

router.delete('/todos/:id', (req, res, next) => {
  Todo.findOneAndDelete({"_id": req.params.id})
    .then(data => res.json(data))
    .catch(next)
})

module.exports = router;
"api.js" 31L, 673C
```

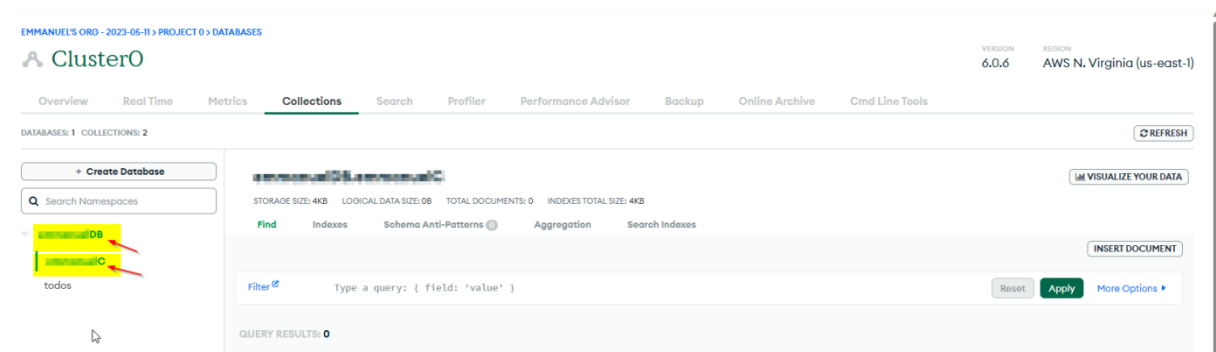
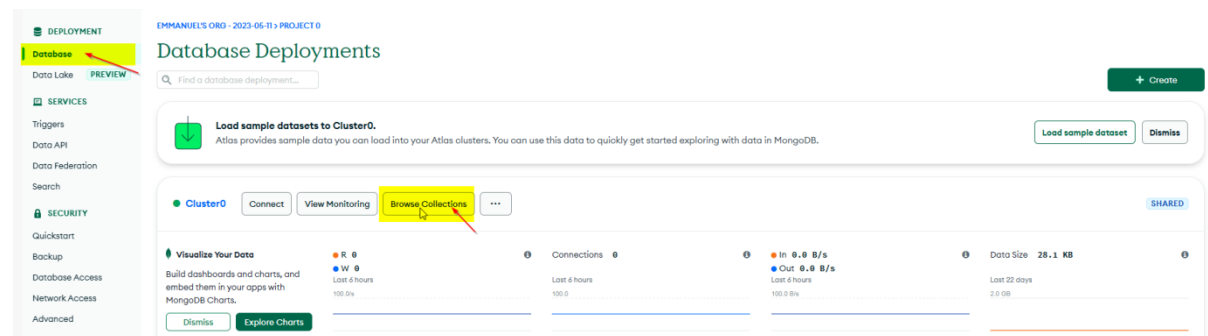
After completion we need to create the MongoDB database

MONGODB DATABASE

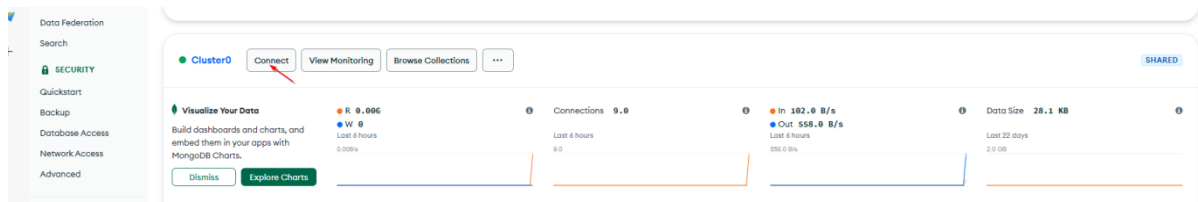
A database is needed to store data and we would be making use of MongoDB database provided by mlab as a service solution. It is expected to have signed up for an account and select AWS as the cloud provider choosing a region close to you .

When we click on browse collections we have access to the database name and collection name created .We would need these details when configuring the .env file .

Please note: When you sign up ensure you change the time of deleting the entry from 6hours to 1 week and for the testing purpose we would allow access to our database from anywhere for study purpose but not secure to do that .



When we click on connect, we connect to our application through the drivers as seen below



Connect to Cluster0



Connect to your application



Drivers

Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)



Access your data through tools



Compass

Explore, modify, and visualize your data with MongoDB's GUI



Shell

Quickly add & update data using MongoDB's Javascript command-line interface



MongoDB for VS Code

Work with your data in MongoDB directly from your VS Code environment



Atlas SQL

Easily connect SQL tools to Atlas for data analysis and visualization



We then cope the connection string and edit the parameter to fit our details ,then input it into our application code in this format

DB = 'mongodb+srv://<username>:<password>@<network-address>/<dbname>?retryWrites=true&w=majority'

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver	Version
<div>Node.js ▾</div>	<div>4.1 or later ▾</div>

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

☐ View full code sample

```
mongodb+srv://[username]:<password>@cluster0.bectm8o.mongodb.net/?  
retryWrites=true&w=majority
```

Create a file in Todo directory called `.env` and edit the file and paste the connection string from the database inside it . press ESC ,save and exit with “ :wq

```
ubuntu@ip-172-31-86-218:~/Todo/routes$ cd ..  
ubuntu@ip-172-31-86-218:~/Todo$ touch .env  
ubuntu@ip-172-31-86-218:~/Todo$ vi .env
```

```
DB = mongodb+srv://[username]:[password]@cluster0.bectm8o.mongo  
db.net/[username]?retryWrites=true&w=majority
```

Then we need to update index.js to show the use of `.env` so that node.js can connect to the database

Simple delete the existing content and update the previous codes with the code below.

Replace this previous line of codes.

```
const express = require('express');
require('dotenv').config();

const app = express();

const port = process.env.PORT || 5000;

app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
  next();
});

app.use((req, res, next) => {
  res.send('Welcome to Express');
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`)
})
```

With this new set of codes below. Press ESC, save and exit with “:wq” command

```
mongoose.connect(process.env.DB, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log(`Database connected successfully`))
  .catch(err => console.log(err));

//since mongoose promise is depreciated, we override it with node's promise
mongoose.Promise = global.Promise;

app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
  next();
});

app.use(bodyParser.json());

app.use('/api', routes);

app.use((err, req, res, next) => {
  console.log(err);
  next();
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`)
})
```

37,3

Bot

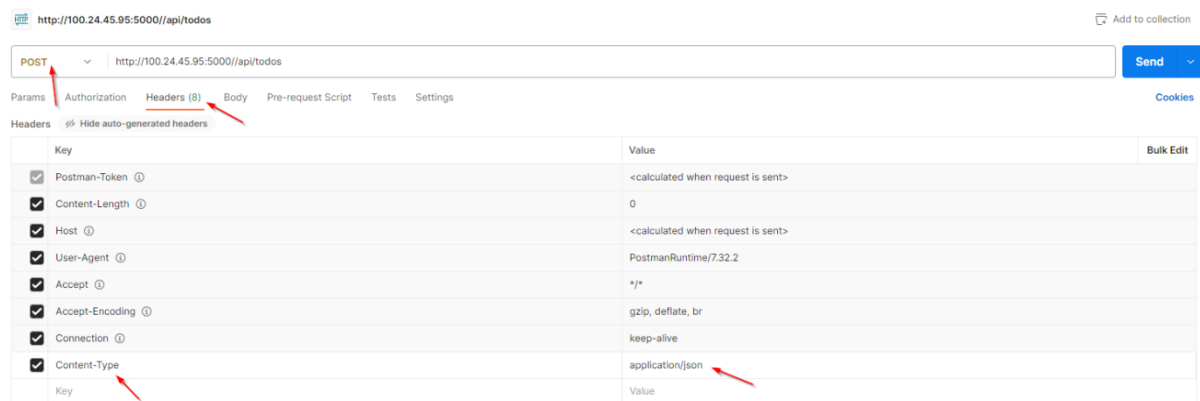
Using environmental variable is very important and most secure and best practice to store sensitive or secrete data from the application.

We would start our server using the command below and our database should be connected successfully.

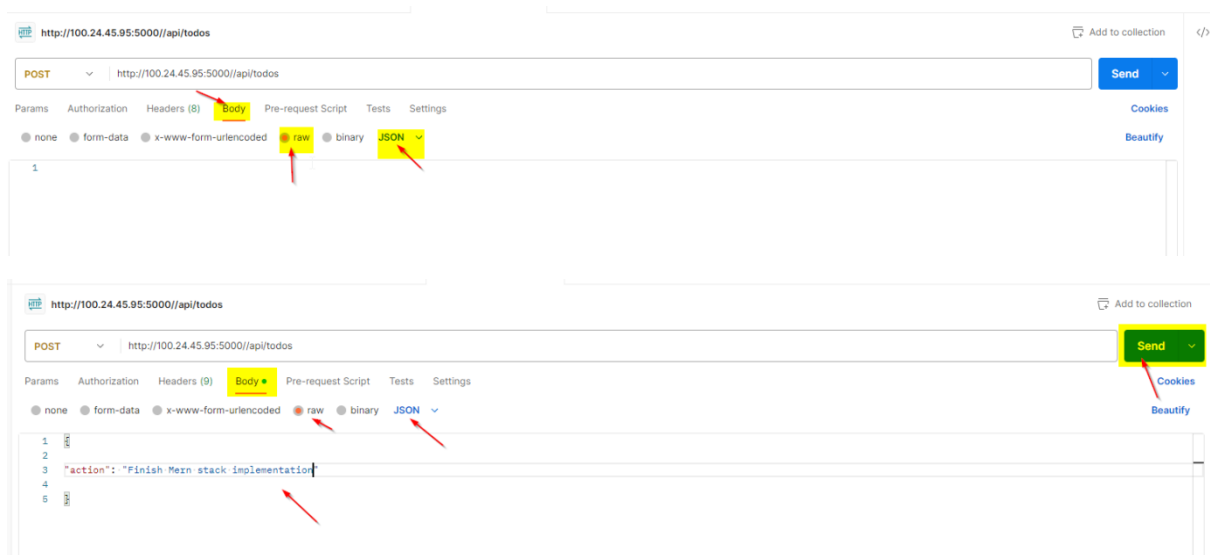
```
ubuntu@ip-172-31-86-218:~/Todo$ node index.js
Server running on port 5000
Database connected successfully
```

Now we open our postman to be able to perform some API request as mentioned earlier. We have to test that all API endpoints are working as expected.

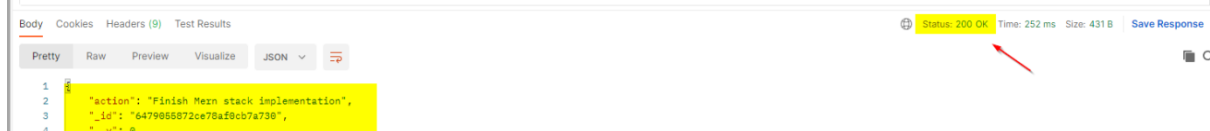
First task would be to create a POST request. Click the header and select the right key value pair as shown below.



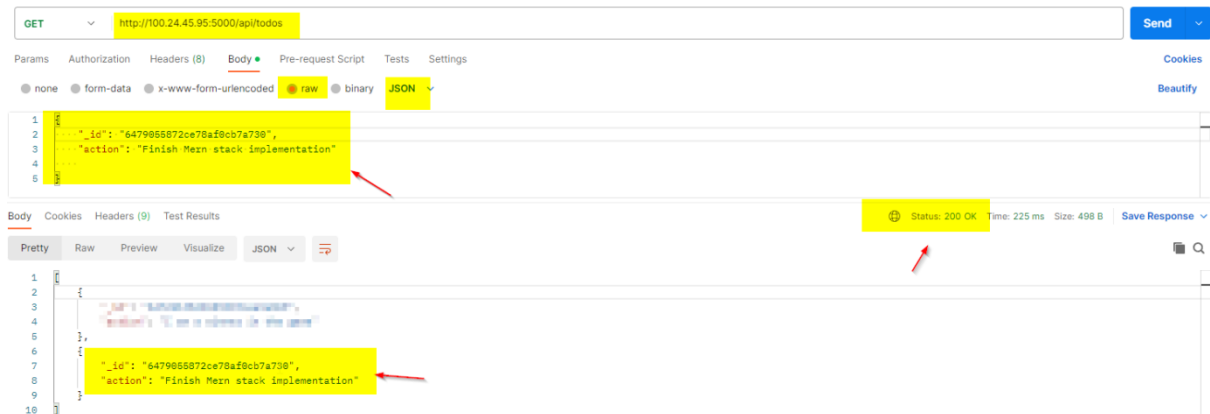
Click on the body and raw, select JSON format. Then type the details as seen below



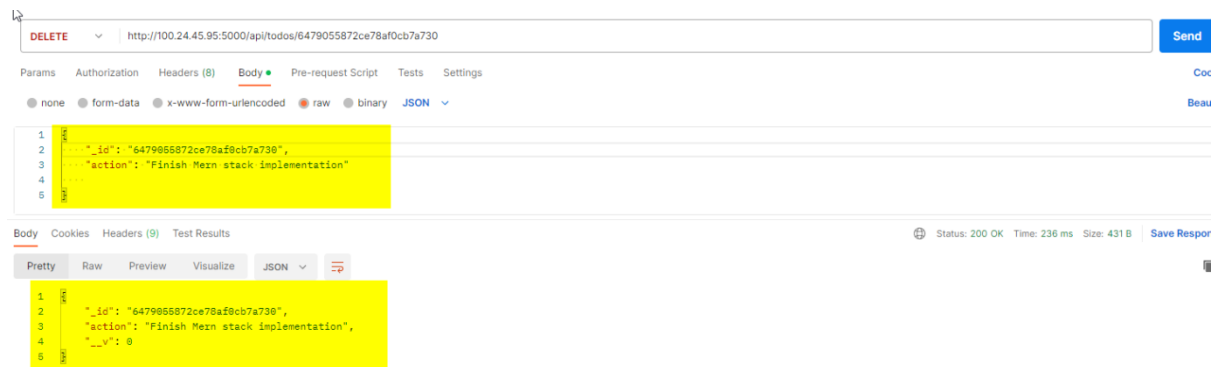
Then 200 OK status is displayed to confirm it is working as expected.



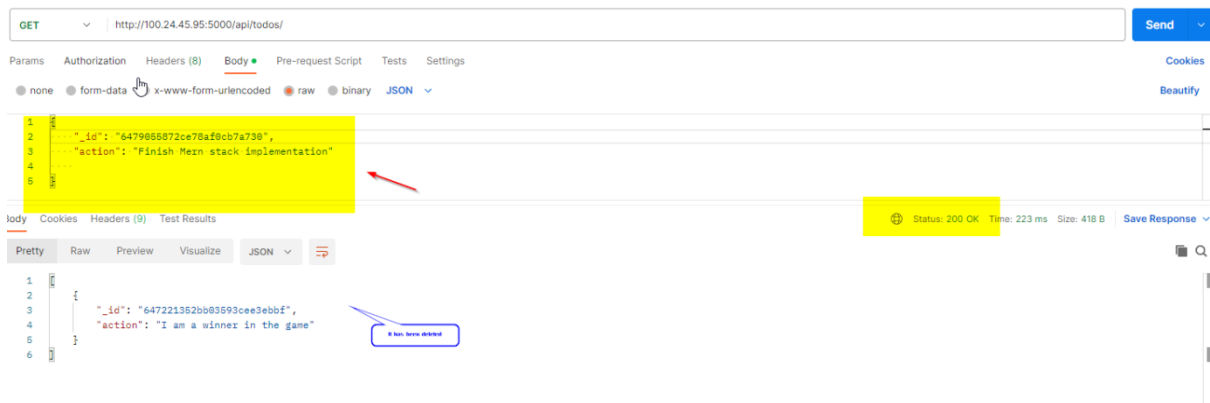
Second task would be to send a GET request and with the relevant details we get the Then 200 OK status is displayed to confirm it is working as expected.



Third task would be to delete any request as seen below.



After deleting, try to get the same data ,You would find out it has been deleted from the database



Now that we are done with the 3 task, we would then proceed to create the front end

FRONTEND CREATION

It is time to create the user interface for a web client to interact with the application via API. To start out we would need to use the command below to scaffold our app.

```
ubuntu@ip-172-31-86-218:~/Todo$ npx create-react-app client
Need to install the following packages:
  create-react-app@5.0.1
Ok to proceed? (y) y
```

This would create a new folder called client in the todo directory and this is where we would add the react code .

```
Success! Created client at /home/ubuntu/Todo/client
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd client
  npm start

Happy hacking!
ubuntu@ip-172-31-86-218:~/Todo$ ls
client  models  package-lock.json  routes
index.js  node_modules  package.json
ubuntu@ip-172-31-86-218:~/Todo$
```

We would now 2 dependencies

1)Install concurrently which is used to run more than one command simultaneously from the same terminal window.

2) Install nodemon which is used to run and monitor the server. If there's any change in the server nodemon would always restart automatically and load the new changes

```
ubuntu@ip-172-31-86-218:~/Todo$ npm install concurrently --save-dev
added 30 packages, and audited 114 packages in 13s

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

found 0 vulnerabilities
ubuntu@ip-172-31-86-218:~/Todo$ npm install nodemon --save-dev
added 32 packages, and audited 146 packages in 2s

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

found 0 vulnerabilities
```

Then we would open the package.json file in the todo folder and change the scrips and test section from the file and replace with the code below

```
"scripts": {
  "start": "node index.js",
  "start-watch": "nodemon index.js",
  "dev": "concurrently \"npm run start-watch\" \"cd client && npm start\""
},
```

```
ubuntu@ip-172-31-86-218:~/Todo$ ls
client index.js models node_modules package-lock.json package.json routes
ubuntu@ip-172-31-86-218:~/Todo$ vim package.json
ubuntu@ip-172-31-86-218:~/Todo$
```

```
{
  "name": "todo",
  "version": "1.0.0",
  "description": "A todo app",
  "main": "index.js",

  "scripts": {
    "start": "node index.js",
    "start-watch": "nodemon index.js",
    "dev": "concurrently \"npm run start-watch\" \"cd client && npm start\""
  },

  "keywords": [
    "todo",
    "application"
  ],
  "author": "Emmanuel Wendy",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^16.1.3",
    "express": "^4.18.2",
    "mongoose": "^7.2.2"
  },
  "devDependencies": {
    "concurrently": "^8.1.0",
    "nodemon": "^2.0.22"
  }
}
-- INSERT --
```

Change directory to client and edit the package.json file and add the key value pair to the file . Press ESC, save and exit with “ :wq” command

```
ubuntu@ip-172-31-86-218:~/Todo$ cd client
ubuntu@ip-172-31-86-218:~/Todo/client$ ls
README.md node_modules package-lock.json package.json public src
ubuntu@ip-172-31-86-218:~/Todo/client$ vi package.json
ubuntu@ip-172-31-86-218:~/Todo/client$
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

{"proxy": "http://localhost:5000",
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

-- INSERT --
```

Now ensure you are inside the todo directory and run the `npm run dev` command. You should see that the application was compiled successfully.

```
ubuntu@ip-172-31-86-218:~/Todo$ npm run dev

> todo@1.0.0 dev
> concurrently "npm run start-watch" "cd client && npm start"

[0] > todo@1.0.0 start-watch
[0] > nodemon index.js
[0]
[1] > client@0.1.0 start
[1] > react-scripts start
[1]
[0] [nodemon] 2.0.22
[0] [nodemon] to restart at any time, enter `rs`
[0] [nodemon] watching path(s): *.*
[0] [nodemon] watching extensions: js,mjs,json
[0] [nodemon] starting `node index.js`
[0] Server running on port 5000
[0] Database connected successfully
[1] (node:28896) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
[1] (Use `node --trace-deprecation ...` to show where the warning was created)
[1] (node:28896) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
[1] Starting the development server...
[1]
[1] Compiled successfully!
[1]
[1] You can now view client in the browser.
[1]
```

Your application should open and start running on localhost:3000. We would need to create an inbound rule to open at port 3000

Click on security button

Instances (1/10) Info

Find instance by attribute or tag (case-sensitive)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input type="checkbox"/>	stackmean	i-09f1e479ad3e0bdce	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-
<input type="checkbox"/>	stacklemp	i-0862f845ba4a743f2	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-
<input type="checkbox"/>	stacklamp	i-03ac707da751f0f22	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-
<input type="checkbox"/>	mernstackmern	i-011b9e604e64ce6f4	Stopped	t2.micro	-	No alarms +	us-east-1a	-	-	-

Instance: i-08b30f6a7ffcdd456 (Mern)

Details Security Networking Storage Status checks Monitoring Tags

▼ Security details

IAM Role -	Owner ID 416591745024	Launch time Thu Jun 01 2023 16:15:35 GMT+0100 (British Summer Time)
---------------	--------------------------	--

Security groups
sg-04d3aa35fbfeb428a (launch-wizard-18)

And click the security group link.

Security groups

sg-04d3aa35fbfeb428a (launch-wizard-18)

▼ Inbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-0c2da85afe8fb4424	22	TCP	0.0.0.0/0	launch-wizard-18	-

Click on “Edit inbound rules “in order to add a new rule for port 5000

Inbound rules (1/1)

Filter security group rules

Manage tags Edit inbound rules

<input checked="" type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input checked="" type="checkbox"/>	-	sgr-0c2da85afe8fb4424	IPv4	SSH	TCP	22	0.0.0.0/0	-

Add a new rule.

Inbound rules Info

Security group rule ID: sgr-0c2da85afe8fb4424

Type: SSH Protocol: TCP Port range: 22 Source: Custom 0.0.0.0/0

Add rule

Type in the port range and click “Anywhere ipv4”

Custom TCP TCP 3000 Custom

Anywhere-IPv4

Cancel Preview changes Save rules

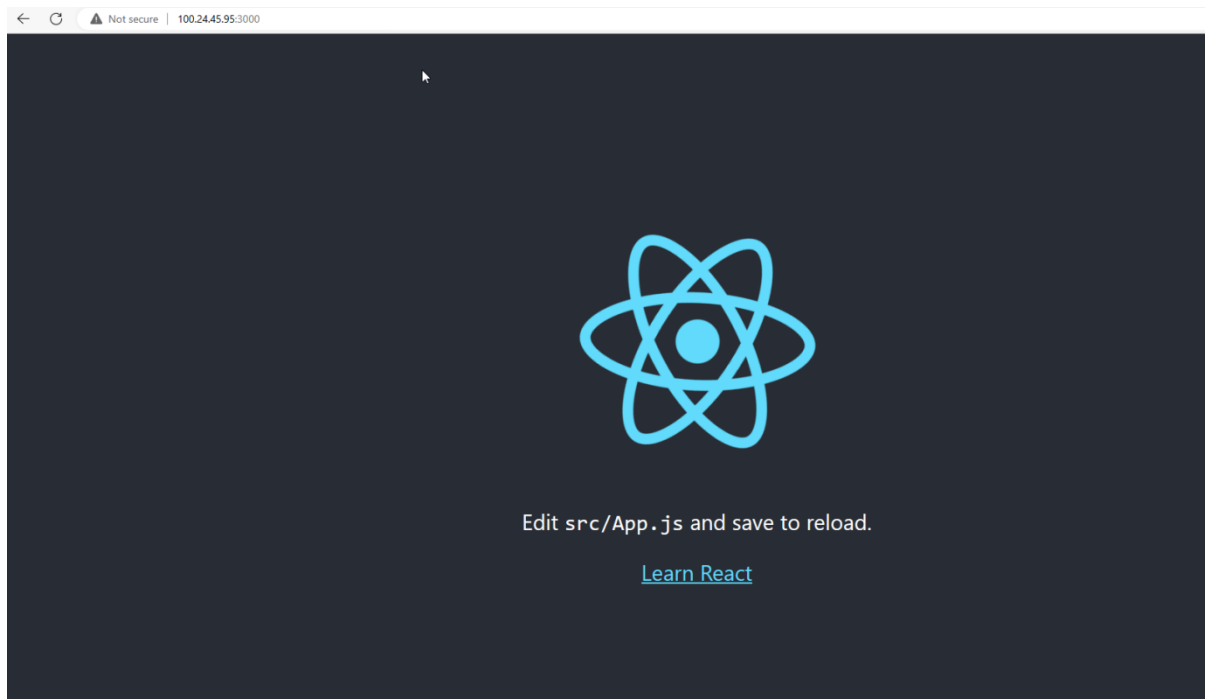
Click the “Save rules” Button

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Info
sg-0b798929a43844065	Custom TCP	TCP	5000	Custom	Mem Project port 5000	Delete
sg-0c2da85afe8fb4424	SSH	TCP	22	Custom		Delete
-	Custom TCP	TCP	3000	Anywhere...	Application launch at 3000	Delete

Inbound rule successfully modified

☒ Inbound security group rules successfully modified on security group (sg-04d3aa35fbfeb428a | launch-wizard-18)

React app launches successfully at port 3000



From our todo app, there would be two stateful component and one stateless component .This is because we want to make the code modular and reusable

Change directory to client and move to the src directory .

Create another folder names components and change directory into the new folder

Create 3 files as earlier explained (be two stateful component and one stateless component) input.js ListTodo.js Todo.js

Open the input.js file and paste the code below.

```
ubuntu@ip-172-31-86-218:~/Todo$ cd client
ubuntu@ip-172-31-86-218:~/Todo/client$ ls
README.md  node_modules  package-lock.json  package.json  public  src
ubuntu@ip-172-31-86-218:~/Todo/client$ cd src
ubuntu@ip-172-31-86-218:~/Todo/client/src$ mkdir components
ubuntu@ip-172-31-86-218:~/Todo/client/src$ cd components/
ubuntu@ip-172-31-86-218:~/Todo/client/src/components$ touch Input.js ListTodo.js Todo.js
ubuntu@ip-172-31-86-218:~/Todo/client/src/components$ vi Input.js
ubuntu@ip-172-31-86-218:~/Todo/client/src/components$
```

```
        this.setState({action: ""})
      }
    }
  }.catch(err => console.log(err))
} else {
  console.log('input field required')
}
}

handleChange = (e) => {
  this.setState({
    action: e.target.value
  })
}

render() {
  let { action } = this.state;
  return (
    <div>
      <input type="text" onChange={this.handleChange} value={action} />
      <button onClick={this.addToDo}>add todo</button>
    </div>
  )
}
}

export default Input
:wd
```

Then we try to install Axios which is a promise-based HTTP client for the browser and node.js.

Move back twice to get to the client folder and install Axios

Move to component directory and edit ListTodo.js

```
ubuntu@ip-172-31-86-218:~/Todo/client$ cd src/components
ubuntu@ip-172-31-86-218:~/Todo/client/src/components$ vi ListTodo.js
```

```
import React from 'react';

const ListTodo = ({ todos, deleteTodo }) => {

  return (
    <ul>
    {
      todos &&
      todos.length > 0 ?
      (
        todos.map(todo => {
          return (
            <li key={todo._id} onClick={() => deleteTodo(todo._id)}>{todo.action}</li>
          )
        })
      )
      :
      (
        <li>No todo(s) left</li>
      )
    }
    </ul>
  )
}

export default ListTodo
-- REPLACE --
```

We would then navigate to the Todo.js file and copy the code below inside it.

```
ubuntu@ip-172-31-86-218:~/Todo/client/src/components$ vi Todo.js
ubuntu@ip-172-31-86-218:~/Todo/client/src/components$
```

```
    axios.delete(`/api/todos/${id}`)
      .then(res => {
        if(res.data){
          this.getTodos()
        }
      })
      .catch(err => console.log(err))
  }

  render() {
    let { todos } = this.state;

    return(
      <div>
        <h1>My Todo(s)</h1>
        <Input getTodos={this.getTodos}/>
        <ListTodo todos={todos} deleteTodo={this.deleteTodo}/>
      </div>
    )
  }
}

export default Todo;
-- INSERT --
```

55,21

Bot

We need to make a little adjustment to our react code .Delete the logo and adjust our App.js to look like this

Move to src folder

```

import React from 'react';

import Todo from '../components/Todo';
import './App.css';

const App = () => {
  return (
    <div className="App">
      <Todo />
    </div>
  );
}

export default App;
~
~
~
~
~
~
~
~
~
~
~

```

14,19

Logo has been deleted and replaced

Next step would be to pass the new code below into the App.css file and exit it

```

input {
  width: 100%;
}

button {
  width: 100%;
  margin-top: 15px;
  margin-left: 0;
}

@media only screen and (min-width: 640px) {
  .App {
    width: 60%;
  }

  input {
    width: 50%;
  }

  button {
    width: 30%;
    margin-left: 10px;
    margin-top: 0;
  }
}

```

87,1

Next step would be to pass the new code below into the index.css file and exit it

```
ubuntu@ip-172-31-86-218:~/Todo/client/src$ vi App.css
ubuntu@ip-172-31-86-218:~/Todo/client/src$ vim index.css
ubuntu@ip-172-31-86-218:~/Todo/client/src$ vim index.css
ubuntu@ip-172-31-86-218:~/Todo/client/src$
```

```
body {
margin: 0;
padding: 0;
font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto",
"Oxygen",
"Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
sans-serif;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
box-sizing: border-box;
background-color: #282c34;
color: #787a80;
}

code {
font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
monospace;
}

~
~
~
~
~
~
~
~

"index.css" 17L, 423C 17,1 All
```

Navigating back to the todo directory

Running the command npm run dev

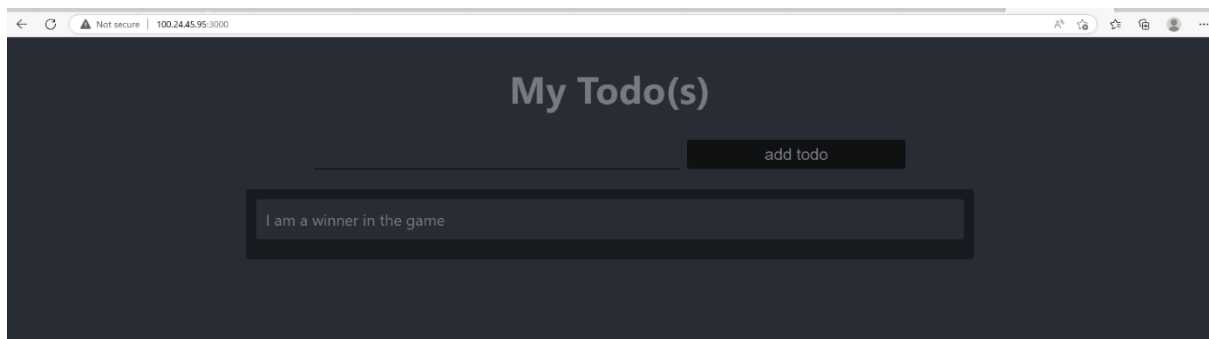
```

[0] [nodemon] watching path(s): *.*
[0] [nodemon] watching extensions: js,mjs,json
[0] [nodemon] starting `node index.js`
[0] Server running on port 5000
[0] Database connected successfully
[1] (node:29976) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE]
  DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated.
  Please use the 'setupMiddlewares' option.
[1] (Use `node --trace-deprecation ...` to show where the warning was
  as created)
[1] (node:29976) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE]
  DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated.
  Please use the 'setupMiddlewares' option.
[1] Starting the development server...
[1]
[1] Compiled successfully!
[1]
[1] You can now view client in the browser.
[1]
[1]   Local:            http://localhost:3000
[1]   On Your Network:  http://172.31.86.218:3000
[1]
[1] Note that the development build is not optimized.
[1] To create a production build, use npm run build.
[1]
[1] webpack compiled successfully

```

Our To-do application should be ready and fully functional with all functionality working perfectly .

Creating a task ,deleting a task and viewing all your tasks.



Simple to-do application deployed in a MERN stack

A Front-end application using React.js that communicates with the backend application written using Express.js.

Created a MongoDB backend for storing tasks in a database

