- **MERN STACK IMPLEMENTATION**

In this project we will be creating a **To-Do** application that creates To-Do lists like.

After creating a new instance lets proceed to the Back end configuration for the To-do application.

1. Update ubuntu `sudo apt update`
2. Upgrade ubuntu `sudo apt upgrade`
3. Let's get the location of Node.js software from Ubuntu repositories by running this command `curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -`
4. Install Node.js with this command `sudo apt-get install -y nodejs`

   Note: The command above installs both nodejs and npm. NPM is a package manager for Node like apt for Ubuntu, it is used to install Node modules & packages and to manage dependency conflicts.

5. Verify the node installation with the command `node -v`
6. Verify the node installation with the command `npm -v`

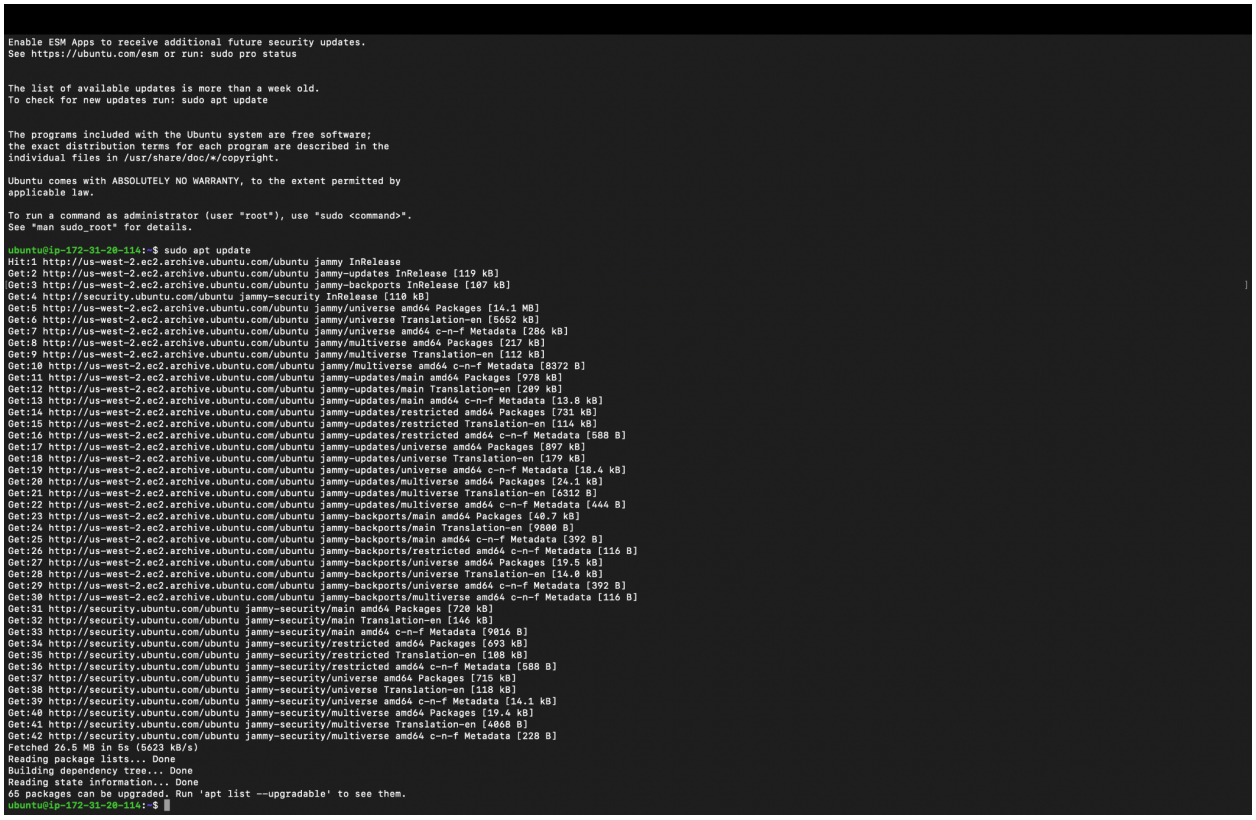
   Application Code Setup

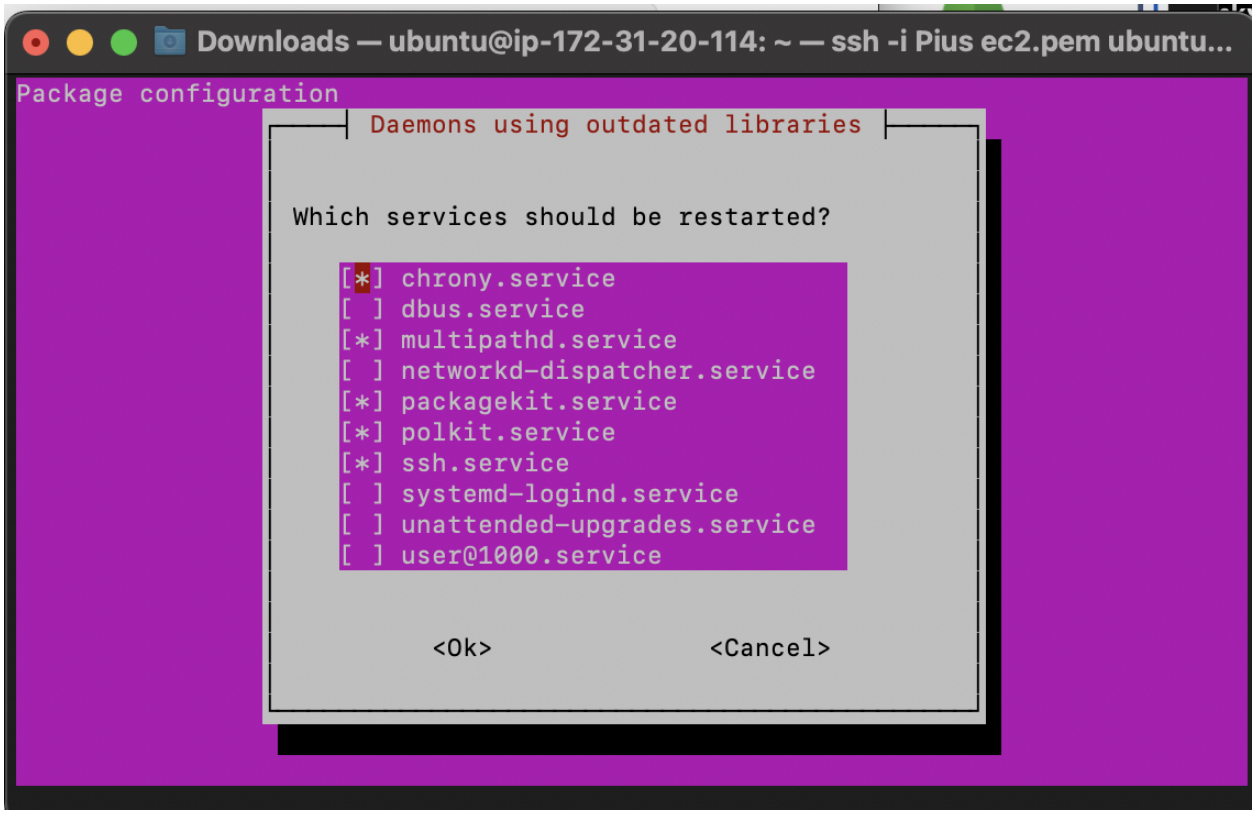
7. Create a new directory for the To-Do project `mkdir Todo`

   TIP: In order to see some more useful information about files and directories, you can use following combination of keys ls -lih – it will show you different properties and size in human readable format. You can learn more about different useful keys for ls command with ls --help.
8. Now change the current directory to the newly created one `cd Todo`

   Next, you will use the command “npm init” to initialize the project, so that a new file named package.json will be created. This file will normally contain information about your application and the dependencies that it needs to run. Follow the prompts after running the command. You can press Enter several times to accept default values, then accept to write out the package.json file by typing yes. Everything you are doing to create the application should be done inside the directory created which is the “Todo”.
   `npm init`

After that run the command Run the command ls to confirm that you have package.json file created.






```
ubuntu@ip-172-31-20-114:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  linux-aws-5.19-headers-5.19.0-1022 linux-headers-5.19.0-1022-aws
  linux-image-5.19.0-1022-aws linux-modules-5.19.0-1022-aws
The following packages have been kept back:
  grub-efi-amd64-bin grub-efi-amd64-signed
The following packages will be upgraded:
  apparmor base-files cloud-init curl fwupd-signed git git-man intel-microcode
  isc-dhcp-client isc-dhcp-common less libapparmor1 libcurl3-gnutls libcurl4
  libgnutls30 libldap-2.5-0 libldap-common libmbim-glib4 libmbim-proxy
  libmm-glib0 libnetplan0 libnss-systemd libnss3 libpam-systemd libpython3.10
  libpython3.10-minimal libpython3.10-stdlib libqmi-glib5 libqmi-proxy
  libsasl2-2 libsasl2-modules libsasl2-modules-db libsystemd0 libudev1
  linux-aws linux-headers-aws linux-image-aws motd-news-config netplan.io
  python-apt-common python3-apt python3-distupgrade
  python3-software-properties python3.10 python3.10-minimal rsync shim-signed
  software-properties-common sudo systemd systemd-hwe-hwdb systemd-sysv tar
  tcpdump ubuntu-advantage-tools ubuntu-release-upgrader-core udev
  update-notifier-common vim vim-common vim-runtime vim-tiny xxd
63 upgraded, 4 newly installed, 0 to remove and 2 not upgraded.
32 standard LTS security updates
Need to get 96.2 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 motd-news-config all 12ubuntu4.3 [4484 B]
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 base-files amd64 12ubuntu4.3 [62.6 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 tar amd64 1.34+dfsg-1ubuntu0.1.22.04.1 [295 kB]
Get:4 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnss-systemd amd64 249.11-0ubuntu3.7 [133 kB]
Get:5 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsystemd0 amd64 249.11-0ubuntu3.7 [317 kB]
Get:6 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 systemd-sysv amd64 249.11-0ubuntu3.7 [10.5 kB]
Get:7 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpam-systemd amd64 249.11-0ubuntu3.7 [203 kB]
Get:8 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 systemd amd64 249.11-0ubuntu3.7 [4580 kB]
Get:9 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 udev amd64 249.11-0ubuntu3.7 [1557 kB]
Get:10 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libudev1 amd64 249.11-0ubuntu3.7 [76.9 kB]
Get:11 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapparmor1 amd64 3.0.4-2ubuntu2.2 [39.2 kB]
Get:12 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgnutls30 amd64 3.7.3-4ubuntu1.2 [968 kB]
Get:13 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpython3.10 amd64 3.10.6-1~22.04.2ubuntu1 [1958 kB]
Get:14 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3.10 amd64 3.10.6-1~22.04.2ubuntu1 [497 kB]
Get:15 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpython3.10-stdlib amd64 3.10.6-1~22.04.2ubuntu1 [1831 kB]
Get:16 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3.10-minimal amd64 3.10.6-1~22.04.2ubuntu1 [2263 kB]
Get:17 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpython3.10-minimal amd64 3.10.6-1~22.04.2ubuntu1 [810 kB]
Get:18 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 rsync amd64 3.2.7-0ubuntu0.22.04.2 [436 kB]
Get:19 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python-apt-common all 2.4.0ubuntu1 [14.3 kB]
Get:20 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-apt amd64 2.4.0ubuntu1 [164 kB]
Get:21 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 update-notifier-common all 3.192.54.6 [185 kB]
Get:22 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 isc-dhcp-client amd64 4.4.1-2.3ubuntu2.4 [235 kB]
Get:23 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 isc-dhcp-common amd64 4.4.1-2.3ubuntu2.4 [45.0 kB]
Get:24 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 less amd64 590-1ubuntu0.22.04.1 [143 kB]
Get:25 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 netplan.io amd64 0.105-0ubuntu2~22.04.3 [92.9 kB]
Get:26 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnetplan0 amd64 0.105-0ubuntu2~22.04.3 [103 kB]
Get:27 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 sudo amd64 1.9.9-1ubuntu2.3 [820 kB]
Get:28 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ubuntu-advantage-tools amd64 27.13.6~22.04.1 [172 kB]
Get:29 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim amd64 2:8.2.3995-1ubuntu2.4 [1728 kB]
Get:30 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-tiny amd64 2:8.2.3995-1ubuntu2.4 [705 kB]
Get:31 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-runtime all 2:8.2.3995-1ubuntu2.4 [6833 kB]
Get:32 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 xxd amd64 2:8.2.3995-1ubuntu2.4 [51.5 kB]
Get:33 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-common all 2:8.2.3995-1ubuntu2.4 [81.5 kB]
Get:34 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apparmor amd64 3.0.4-2ubuntu2.2 [595 kB]
Get:35 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ubuntu-release-upgrader-core all 1:22.04.16 [26.2 kB]
Get:36 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-distupgrade all 1:22.04.16 [107 kB]
Get:37 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 tcpdump amd64 4.99.1-3ubuntu0.1 [501 kB]
Get:38 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsasl2-modules-db amd64 2.1.27+dfsg2-3ubuntu1.2 [20.5 kB]
Get:39 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsasl2-2 amd64 2.1.27+dfsg2-3ubuntu1.2 [53.8 kB]
```

```
ubuntu@ip-172-31-20-114:~$ 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-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Fetched 226 kB in 1s (360 kB/s)
Reading package lists... Done

## Confirming "jammy" is supported...

+ curl -sLf -o /dev/null 'https://deb.nodesource.com/node_18.x/dists/jammy/Release'

## Adding the NodeSource signing key to your keyring...

+ curl -s https://deb.nodesource.com/gpgkey/nodesource.gpg.key | gpg --dearmor | tee /usr/share/keyrings/nodesource.gpg >/dev/null

## Creating apt sources list file for the NodeSource Node.js 18.x repo...

+ echo 'deb [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_18.x jammy main' > /etc/apt/sources.list.d/nodesource.list
+ echo 'deb-src [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_18.x jammy main' >> /etc/apt/sources.list.d/nodesource.list

## Running `apt-get update` for you...

+ apt-get update
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:4 https://deb.nodesource.com/node_18.x jammy InRelease [4563 B]
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:6 https://deb.nodesource.com/node_18.x jammy/main amd64 Packages [776 B]
Fetched 231 kB in 1s (325 kB/s)
Reading package lists... Done

## Run `sudo apt-get install -y nodejs` to install Node.js 18.x and npm
## You may also need development tools to build native addons:
     sudo apt-get install gcc g++ make
## To install the Yarn package manager, run:
     curl -sL https://dl.yarnpkg.com/debian/pubkey.gpg | gpg --dearmor | sudo tee /usr/share/keyrings/yarnkey.gpg >/dev/null
     echo "deb [signed-by=/usr/share/keyrings/yarnkey.gpg] https://dl.yarnpkg.com/debian stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
     sudo apt-get update && sudo apt-get install yarn
```

```
ubuntu@ip-172-31-20-114:~$ sudo apt-get install -y 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 2 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 jammy/main amd64 nodejs amd64 18.15.0-deb-1nodesource1 [28.7 MB]
Fetched 28.7 MB in 1s (52.8 MB/s)
Selecting previously unselected package nodejs.
(Reading database ... 93021 files and directories currently installed.)
Preparing to unpack .../nodejs_18.15.0-deb-1nodesource1_amd64.deb ...
Unpacking nodejs (18.15.0-deb-1nodesource1) ...
Setting up nodejs (18.15.0-deb-1nodesource1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Package configuration

┌──────────────────┤ Pending kernel upgrade ├──────────────────┐
│                                                             │
│ Newer kernel available                                      │
│                                                             │
│ The currently running kernel version is 5.15.0-1028-aws which is not the
│ expected kernel version 5.19.0-1022-aws.                    │
│                                                             │
│ Restarting the system to load the new kernel will not be handled
│ automatically, so you should consider rebooting.            │
│                                                             │
│                             <Ok>                            │
└─────────────────────────────────────────────────────────────┘

Package configuration
```

```
ubuntu@ip-172-31-20-114:~$ 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: (ubuntu) package.json
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/ubuntu/package.json:

{
  "name": "package.json",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
npm notice
npm notice New minor version of npm available! 9.5.0 -> 9.6.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.2
npm notice Run npm install -g npm@9.6.2 to update!
npm notice
ubuntu@ip-172-31-20-114:~$
```

- **INSTALL EXPRESSJS**

Install ExpressJS

Remember that Express is a framework for Node.js, therefore a lot of things developers would have programmed is already taken care of out of the box. Therefore it simplifies development, and abstracts a lot of low level details. For example, Express helps to define routes of your application based on HTTP methods and URLs.

1. To use express, install it using npm `npm install express`

2. Create a file in Todo directory index.js with the command `touch index.js`
3. Run “ls” in your Todo directory to confirm that your index.js file is successfully created
4. Now install the dotenv module in your Todo directory `npm install dotenv`
5. Open the index.js file with the command `vim index.js`
   Type the code below into the file and save.

```
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}`)
});
```

Use :w to save in vim and use :qa to exit vim or use :wa to save and quit.

6. Now it is time to start the server to see if it works. Open your terminal in the same directory as your index.js file which is the “Todo”and type this command `node index.js`

7. Now open a new port in EC2 Security Groups, add an inbound rule to open TCP port 5000.

8. Open up your browser and try to access your server’s Public IP or Public DNS name followed by port 5000: `http://<PublicIP-or-PublicDNS>:5000`

Routes

There are three actions that the To-Do application needs to be able to do:

1. Create a new task
2. Display list of all tasks

3. Delete a completed task

   Each task will be associated with some particular endpoint and will use different standard HTTP request methods: POST, GET, DELETE.

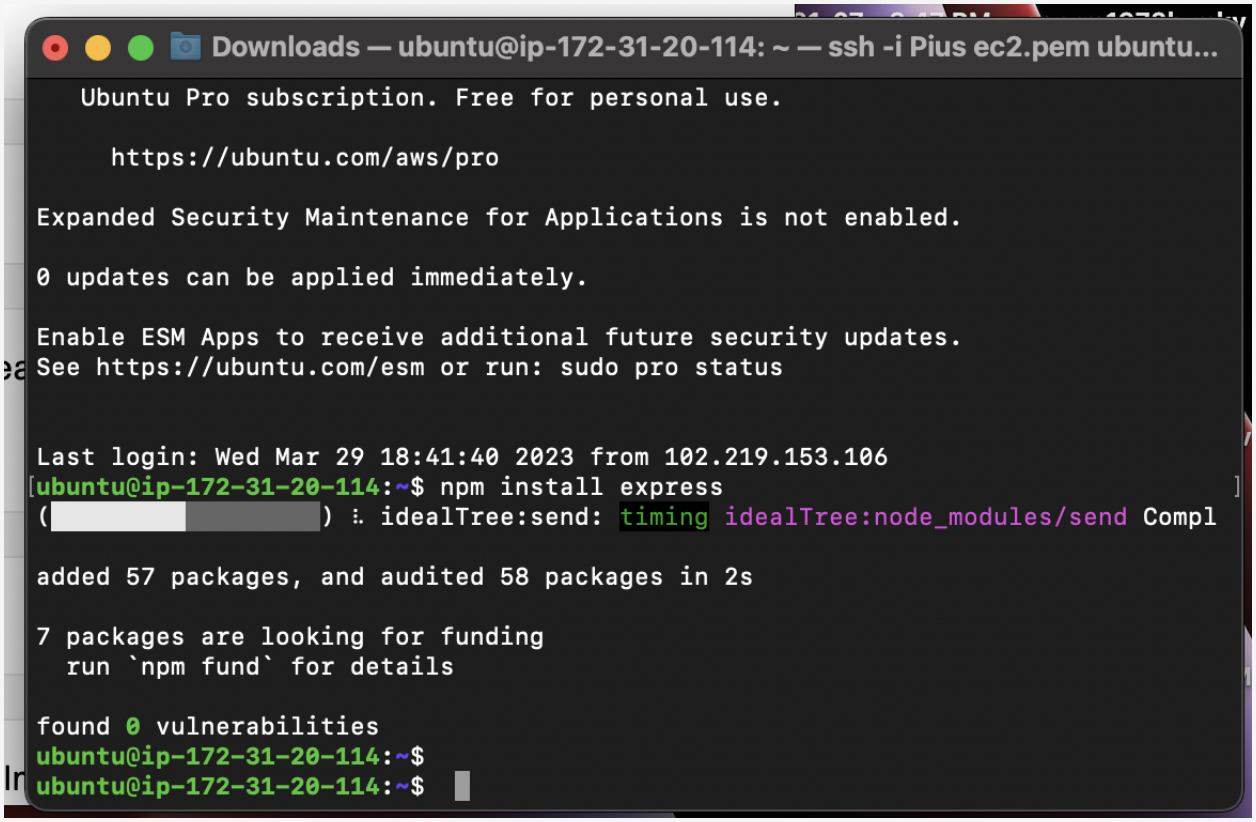
   9. For each task, we need to create “routes” in the Todo directory that will define various endpoints that the To-do app will depend on. So create a folder “routes”

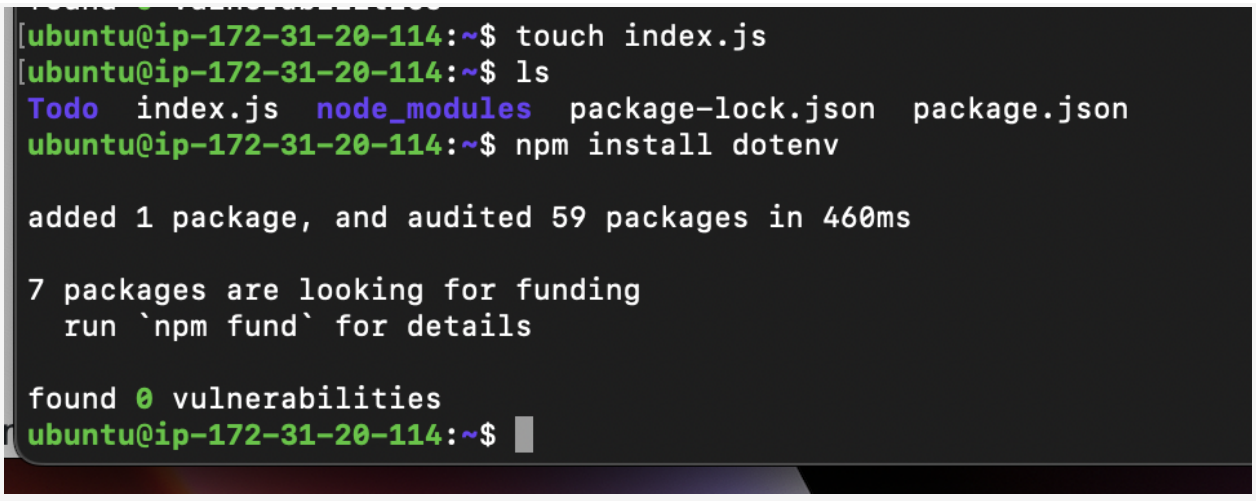
   `mkdir routes`

10. Change directory to routes folder. `cd routes`
11. Now, create a file api.js with the command `touch api.js`
12. Open the file with the command `vim api.js`

Copy below code in the file.

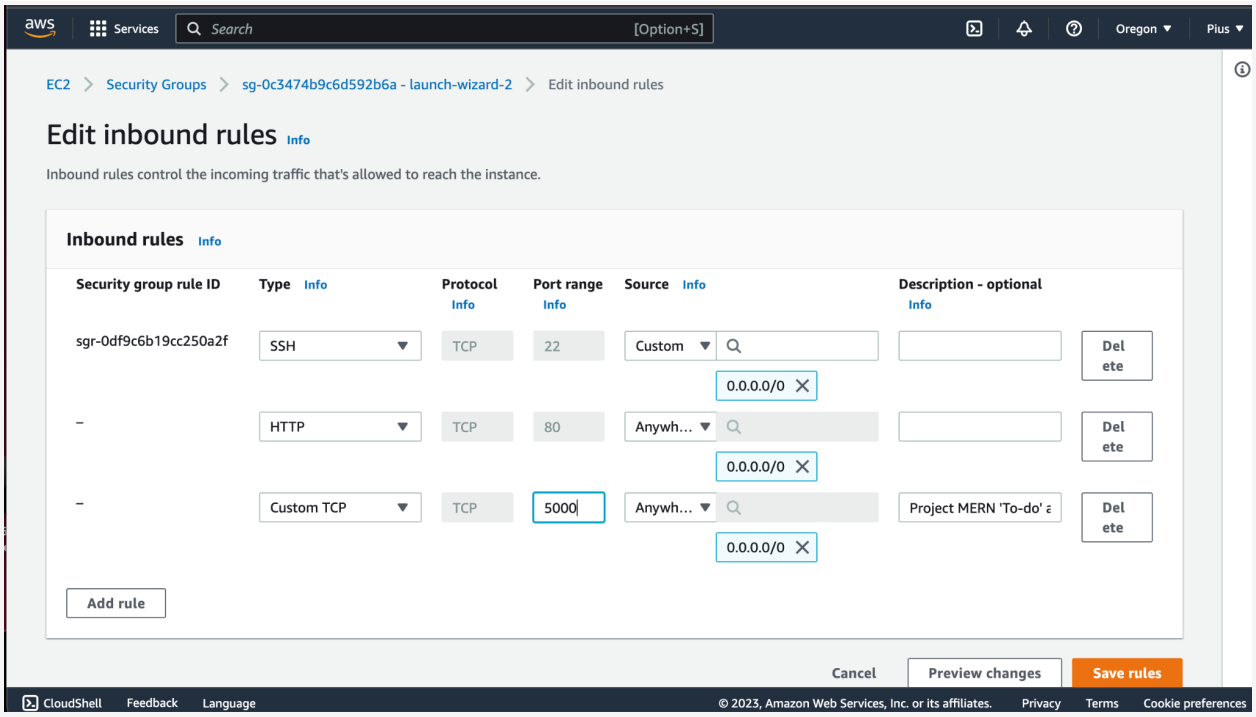
```
 const express = require ('express');
const router = express.Router();

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

});

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

});

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

})

module.exports = router;
```

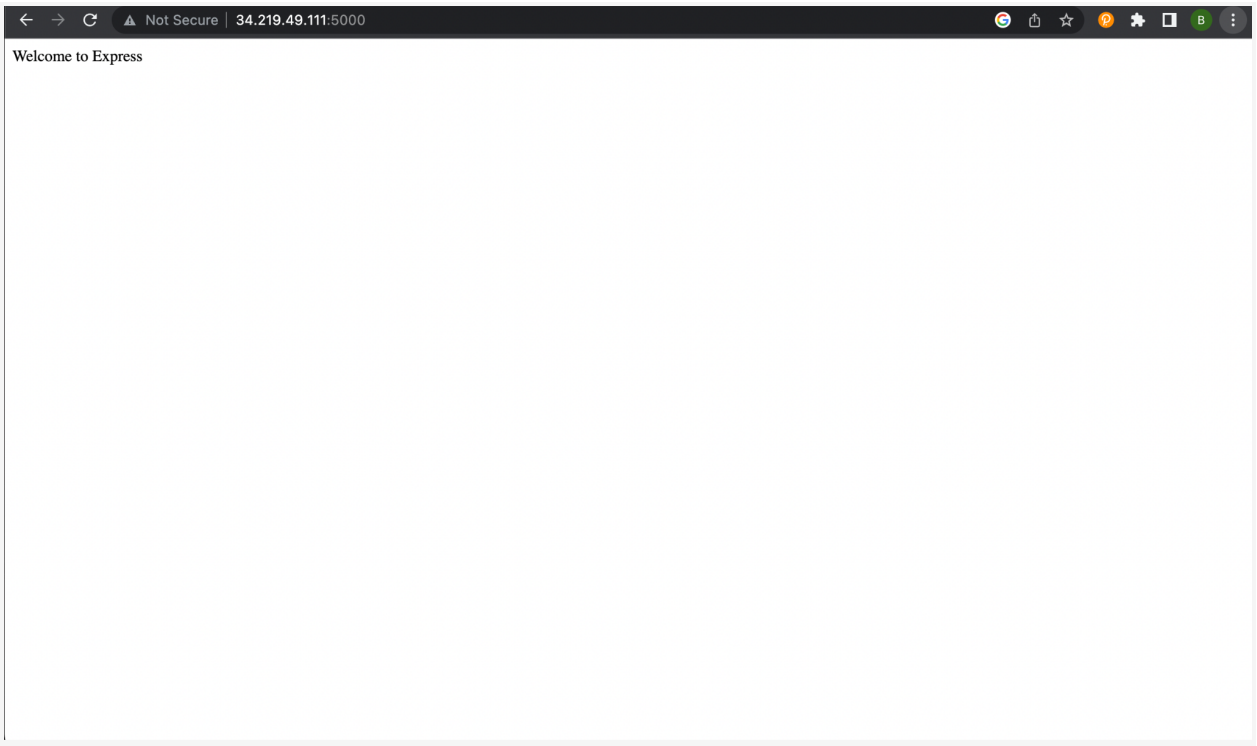

Downloads — ubuntu@ip-172-31-20-114: ~ — ssh -i Pius ec2.pem ubuntu...
Ubuntu Pro subscription. Free for personal use.
https://ubuntu.com/aws/pro
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Last login: Wed Mar 29 18:41:40 2023 from 102.219.153.106
ubuntu@ip-172-31-20-114:~$ npm install express
idealTree:send: timing idealTree:node_modules/send Compl
added 57 packages, and audited 58 packages in 2s
7 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
ubuntu@ip-172-31-20-114:~$
ubuntu@ip-172-31-20-114:~$


ubuntu@ip-172-31-20-114:~$ touch index.js
ubuntu@ip-172-31-20-114:~$ ls
Todo index.js node_modules package-lock.json package.json
ubuntu@ip-172-31-20-114:~$ npm install dotenv
added 1 package, and audited 59 packages in 460ms
7 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
ubuntu@ip-172-31-20-114:~$

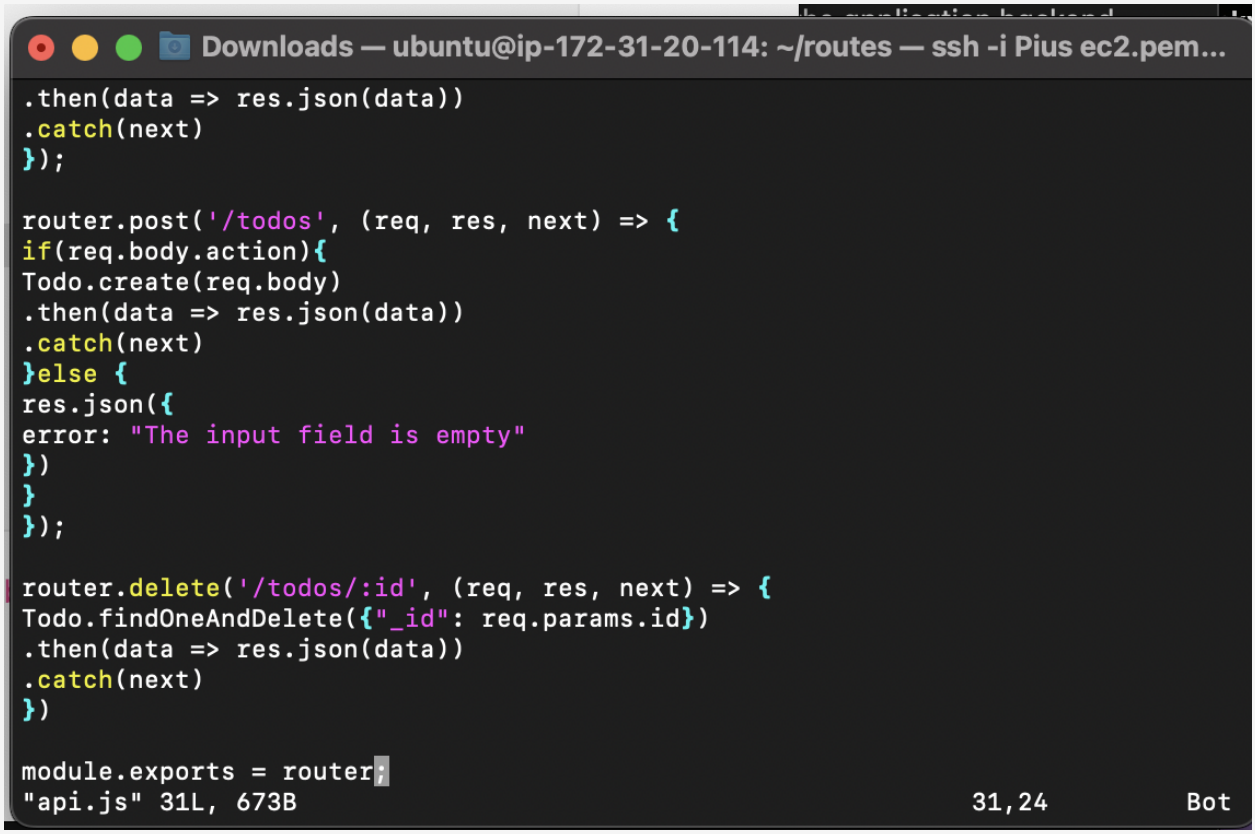
Downloads — ubuntu@ip-172-31-20-114: ~ — ssh -i Pius ec2.pem ubuntu...

```
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-Ty
pe, Accept");
next();
});

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

app.listen(port, () => {
console.log(`Server running on port ${port}`)
});
~
~
"index.js" 20L, 457B                                    20,3          All
```






```
[ubuntu@ip-172-31-20-114:~$ mkdir routes
[ubuntu@ip-172-31-20-114:~$ ls
 Todo  index.js  node_modules  package-lock.json  package.json  routes
[ubuntu@ip-172-31-20-114:~$ cd routes
[ubuntu@ip-172-31-20-114:~/routes$ touch api.js
[ubuntu@ip-172-31-20-114:~/routes$ ls
 api.js
[ubuntu@ip-172-31-20-114:~/routes$ vim api.js
[ubuntu@ip-172-31-20-114:~/routes$ vim api.js
```



- **MODELS**

Since the app is going to make use of Mongodb which is a NoSQL database, we need to create a model. A model is at the heart of JavaScript based applications, and it is what makes it interactive.

We will also use models to define the database schema . This is important so that we will be able to define the fields stored in each Mongodb document. The Schema is a blueprint of how the database will be constructed, including other data fields that may not be required to be stored in the database. These are known as *virtual properties.*

1. To create a Schema and a model, install *mongoose* which is a Node.js package that makes working with mongodb easier, go back to Todo dirctory folder and install Mongoose. `npm install mongoose`
2. Create a new folder “modules” inside Todo directory `mkdir models`

3. Change directory into the newly created ‘models’ folder `cd models`
4. Inside the models folder, create a file and name it todo.js `touch todo.js`

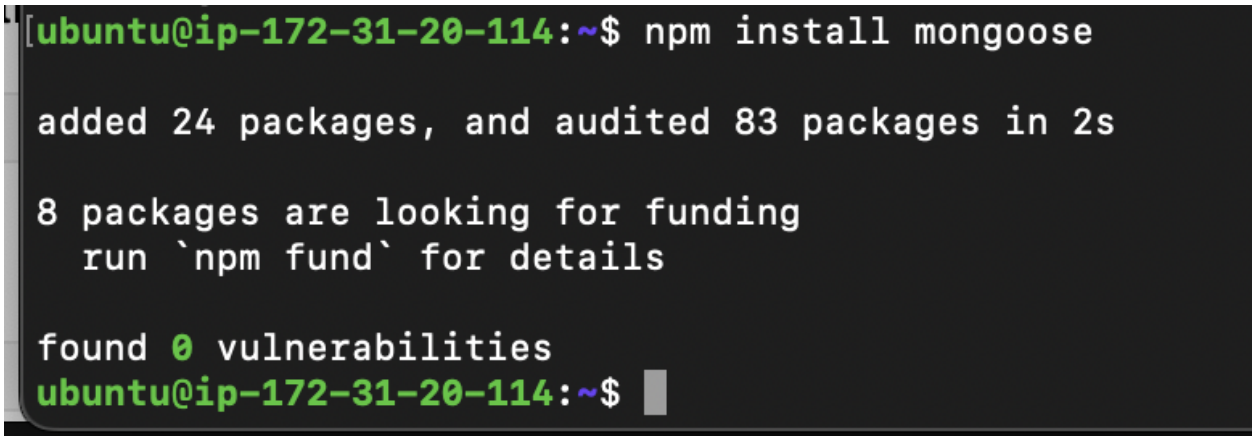
   All three commands above can be defined in one line to be executed consequently with help of && operator, like this: `mkdir models && cd models && touch todo.js`
5. Open the file created with vim todo.js then paste the code below in the file:

```
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;
```

6. Now we need to update our routes from the file api.js in ‘routes’ directory to make use of the new model. In Routes directory, open api.js with vim api.js, delete the code inside with :%d command and paste there code below into it then save and exit with :wa

```
const express = require ('express');
const router = express.Router();
const Todo = require('../models/todo');

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 {
```

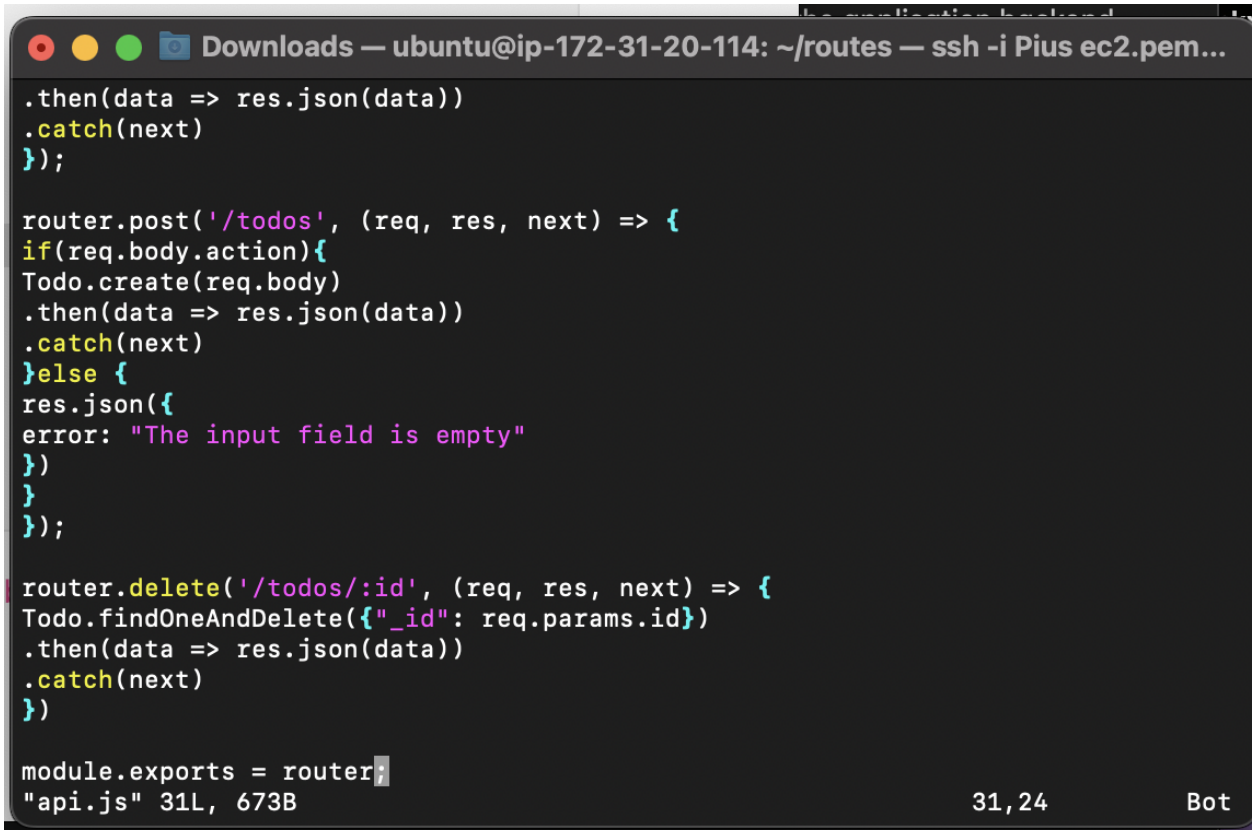
```
        res.json({
        error: "The input field is empty"
        })
        }
        });

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

module.exports = router;
```



```
[ubuntu@ip-172-31-20-114:~$ mkdir models
[ubuntu@ip-172-31-20-114:~$ cd models
[ubuntu@ip-172-31-20-114:~/models$ touch todo.js
[ubuntu@ip-172-31-20-114:~/models$ ls
 todo.js
[ubuntu@ip-172-31-20-114:~/models$ vim todo.js
 ubuntu@ip-172-31-20-114:~/models$
```

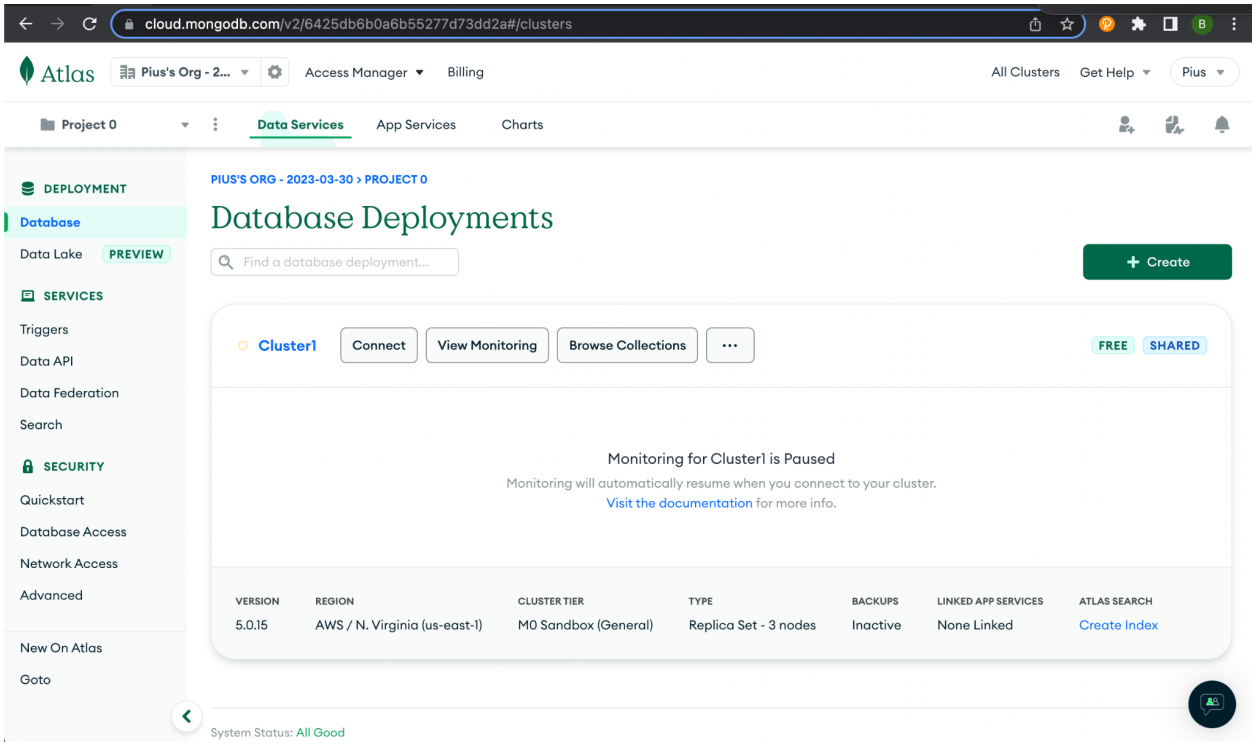
Downloads — ubuntu@ip-172-31-20-114: ~/models — ssh -i Pius ec2.pem...

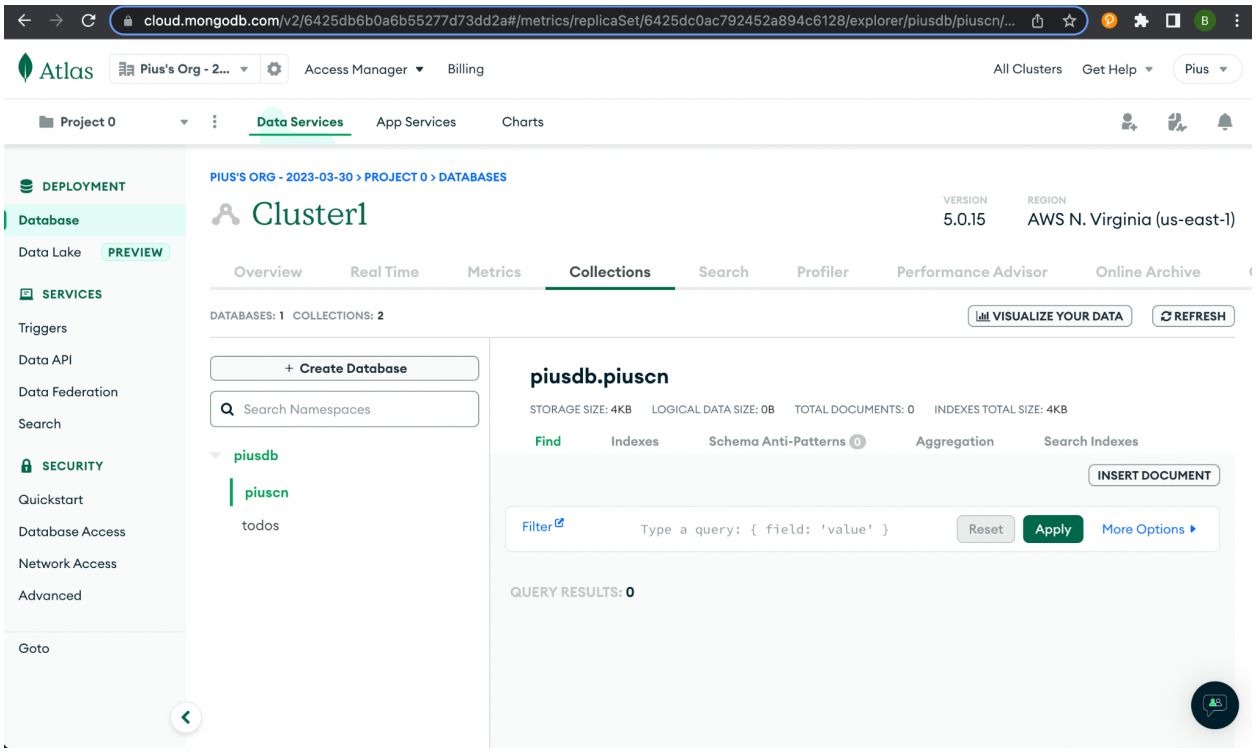
```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

//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;
~
~
~
~
~
~
~
"todo.js" 16L, 308B                                    16,0-1        All
```

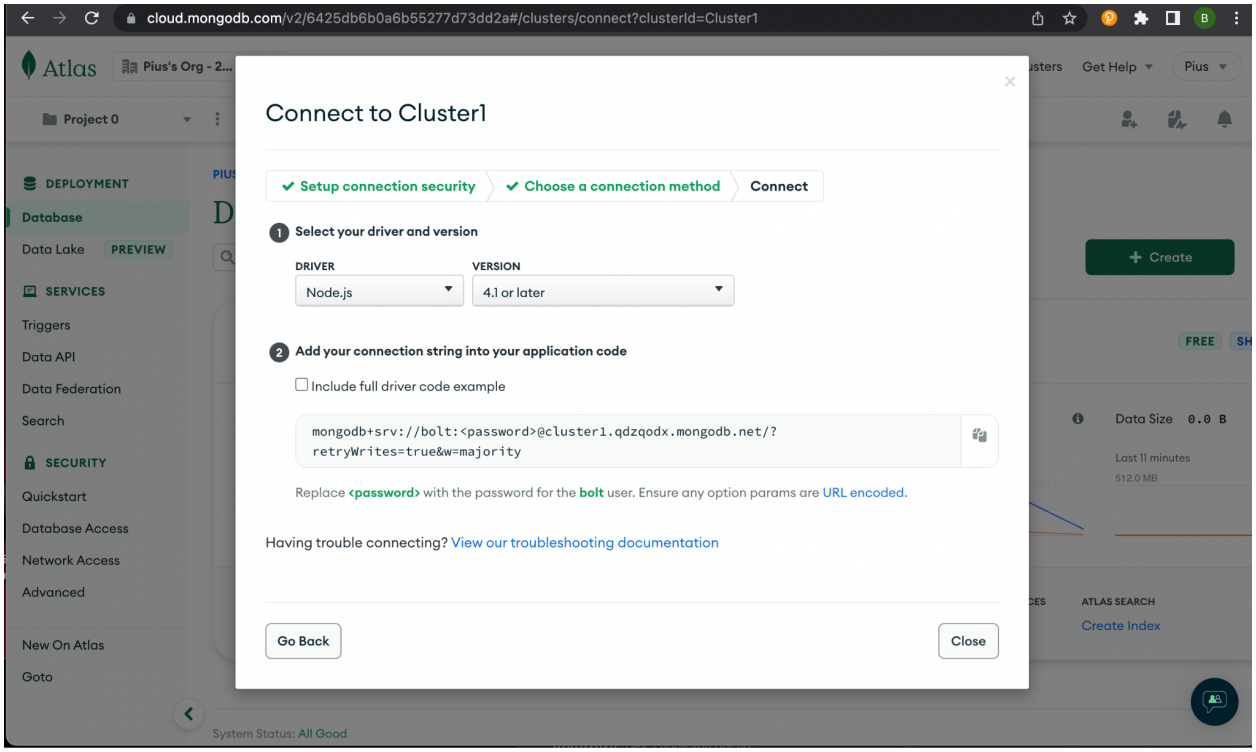


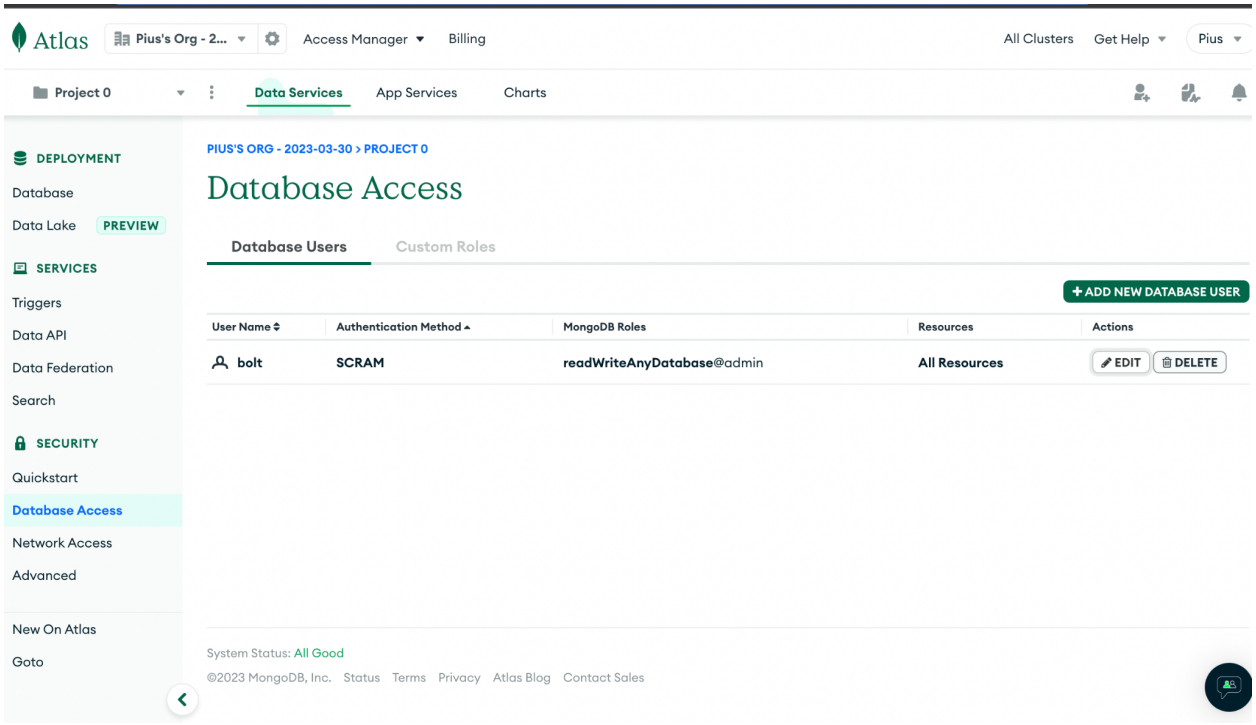
- **MONGODB DATABASE**

We need a database where we will store our data. For this we will make use of mLab. mLab provides MongoDB database as a service solution (DBaaS), so to make life easy, you will need to sign up for a shared clusters free account, which is ideal for our use case. Follow the sign up process, select AWS as the cloud provider, and choose a region near you.


cloud.mongodb.com/v2/6425db6b0a6b55277d73dd2a#/clusters
Atlas
Pius's Org - 2...
Access Manager
Billing
All Clusters
Get Help
Pius
Project 0
Data Services
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
SERVICES
Triggers
Data API
Data Federation
Search
SECURITY
Quickstart
Database Access
Network Access
Advanced
New On Atlas
Goto
PIUS'S ORG - 2023-03-30 > PROJECT 0
Database Deployments
Find a database deployment...
Create
Cluster1
Connect
View Monitoring
Browse Collections
FREE
SHARED
Monitoring for Cluster1 is Paused
Monitoring will automatically resume when you connect to your cluster.
Visit the documentation for more info.
VERSION
REGION
CLUSTER TIER
TYPE
BACKUPS
LINKED APP SERVICES
ATLAS SEARCH
5.0.15
AWS / N. Virginia (us-east-1)
M0 Sandbox (General)
Replica Set - 3 nodes
Inactive
None Linked
Create Index
System Status: All Good


cloud.mongodb.com/v2/6425db6b0a6b55277d73dd2a#/metrics/replicaSet/6425dc0ac792452a894c6128/explorer/piusdb/piuscn/...
Atlas
Pius's Org - 2...
Access Manager
Billing
All Clusters
Get Help
Pius
Project 0
Data Services
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
SERVICES
Triggers
Data API
Data Federation
Search
SECURITY
Quickstart
Database Access
Network Access
Advanced
Goto
PIUS'S ORG - 2023-03-30 > PROJECT 0 > DATABASES
Cluster1
VERSION
5.0.15
REGION
AWS N. Virginia (us-east-1)
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
DATABASES: 1 COLLECTIONS: 2
VISUALIZE YOUR DATA
REFRESH
Create Database
Search Namespaces
piusdb
piuscn
todos
piusdb.piuscn
STORAGE SIZE: 4KB
LOGICAL DATA SIZE: 0B
TOTAL DOCUMENTS: 0
INDEXES TOTAL SIZE: 4KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
More Options
QUERY RESULTS: 0





Make sure you go to data access and go to edit and then change you “built in role” to “read and write to any database” in other to avoid error on your POSTMAN as you proceed.

In the index.js file, we specified process.env to access environment variables, but we have not yet created this file. So we need to do that now.

1. Create a file in your Todo directory and name it .env. `touch .env`
   `vi .env`
   Add the connection string to access the database in it, just as below

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

Ensure to update <username>, <password>, <network-address> and <database> according to your setup

2. Now we need to update the index.js to reflect the use of .env so that Node.js can connect to the database. `vim index.js`
3. Delete the previous codes with :%d and paste the code below

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const routes = require('./routes/api');
const path = require('path');
require('dotenv').config();

const app = express();

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

//connect to the database
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 overide 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}`)
});
```

Using environment variables to store information is considered more secure and best practice to separate configuration and secret data from the application, instead of writing connection strings directly inside the index.js application file.

4. Start your server in your Todo directory using this command `node index.js`
   If you see a message **'Database connected successfully'**, if so – we have our backend configured. Now we are going to test it.

Testing Backend Code without Frontend using RESTful API

So far we have written backend part of the To-Do application, and configured a database, but we do not have a frontend UI yet. We need ReactJS code to achieve that. But during development, we will need a way to test our code using RESTfulL API. Therefore, we will need to make use of some API development client to test our code.

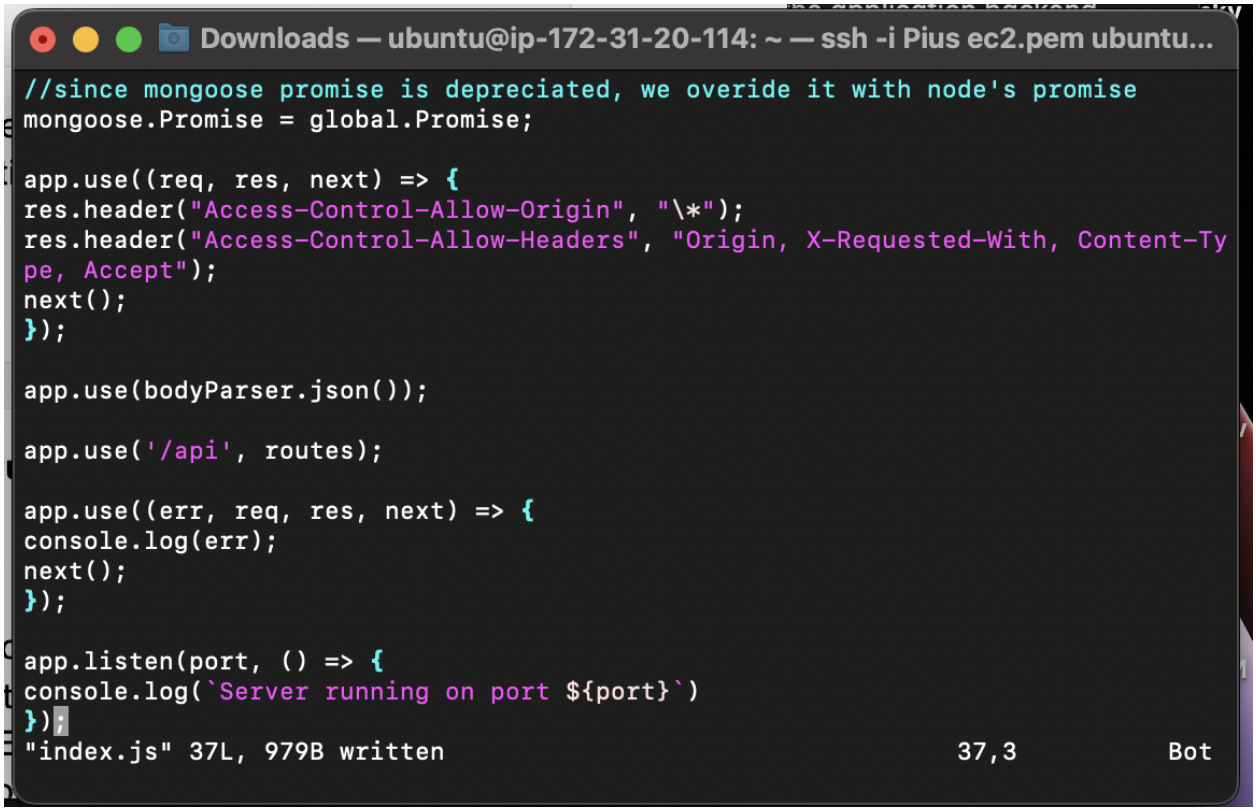
We will use Postman to test our API. You should test all the API endpoints and make sure they are working. For the endpoints that require body, you should send JSON back with the necessary fields since it's what we set up in our code.

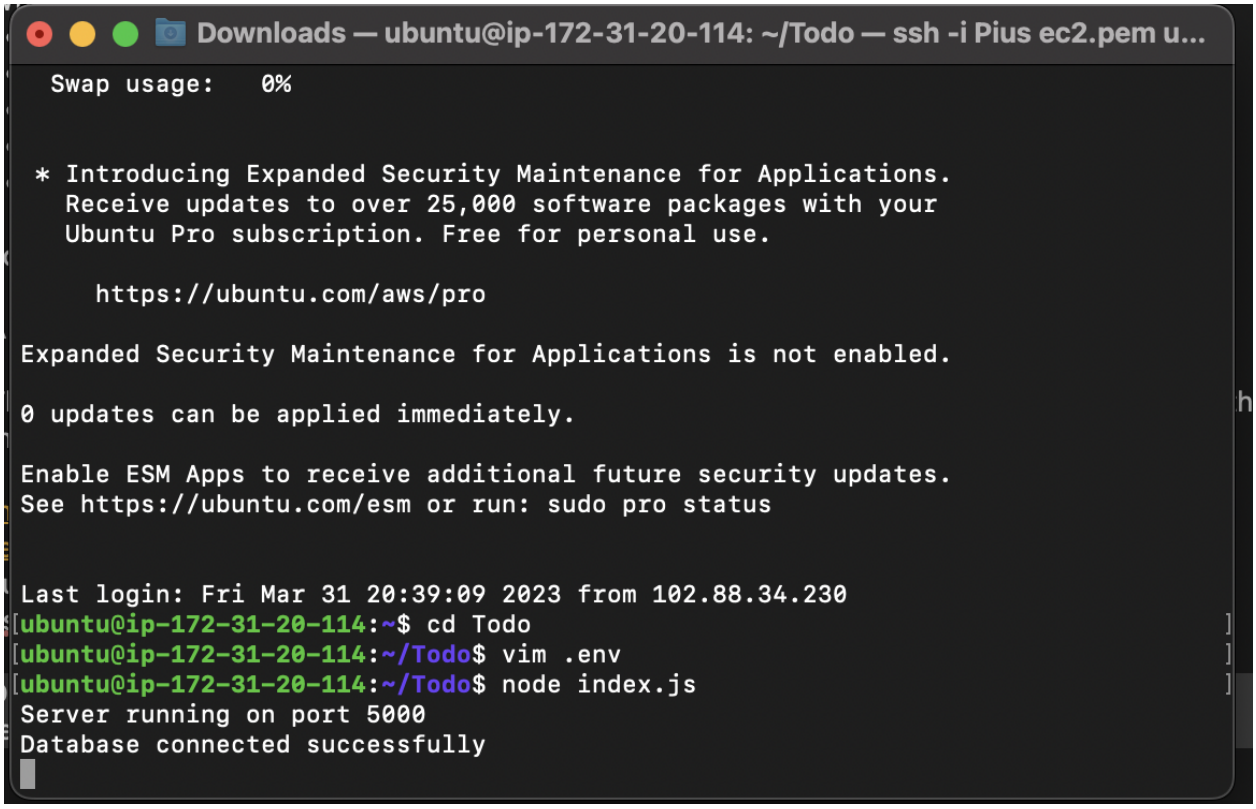
5. Now open your Postman, create a POST request to the API http://<PublicIP-or-PublicDNS>:5000/api/todos. This request sends a new task to our To-Do list so the application could store it in the database.

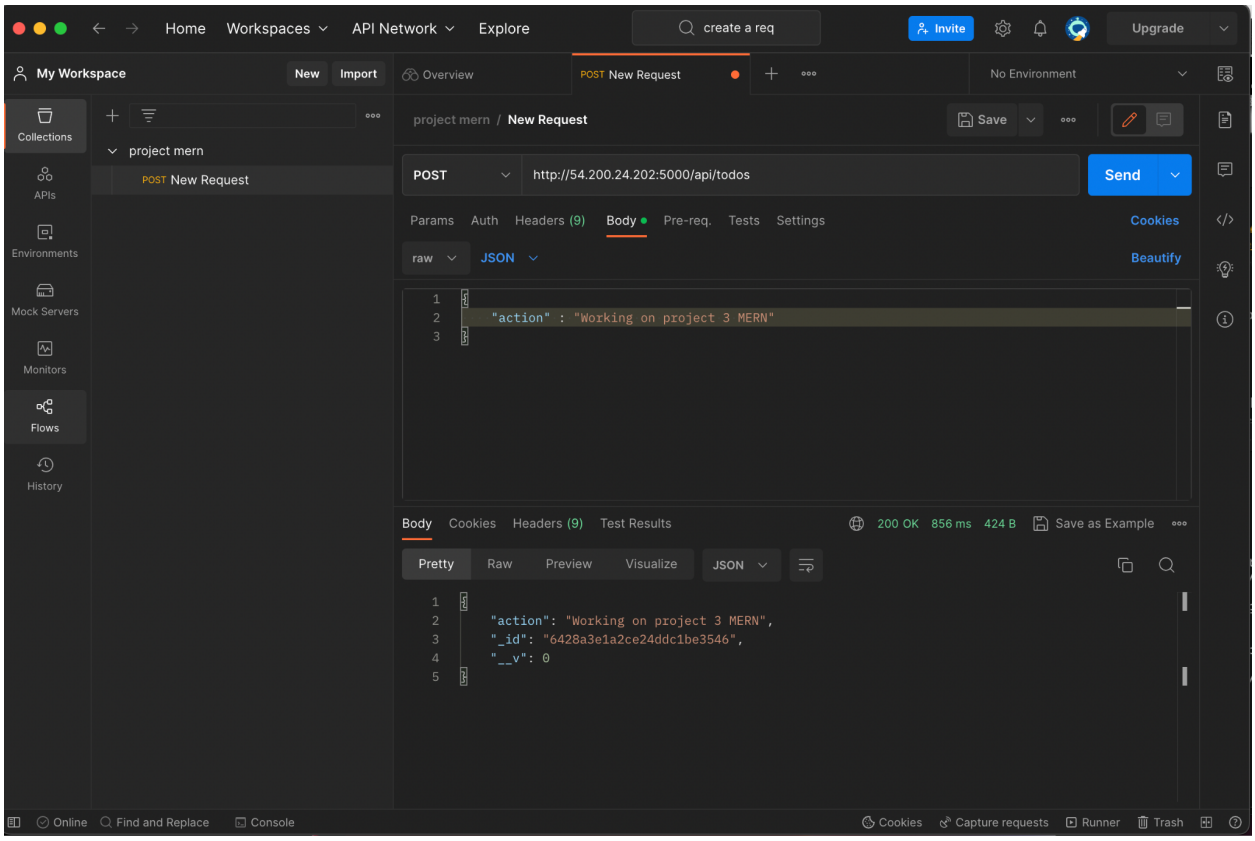
   Note: make sure your set header key Content-Type as application/json

6. Create a GET request to your API on http://<PublicIP-or-PublicDNS>:5000/api/todos. This request retrieves all existing records from our To-do application (backend requests these records from the database and sends it back as a response to GET request).
7. Send a DELETE request to delete a task from our To-Do list, you can do this by adding the id gotten from the GET request http://<PublicIP-or-PublicDNS>:5000/api/todos/id

```
ubuntu@ip-172-31-20-114:~/Todo$ vi .env
[ubuntu@ip-172-31-20-114:~/Todo$
```

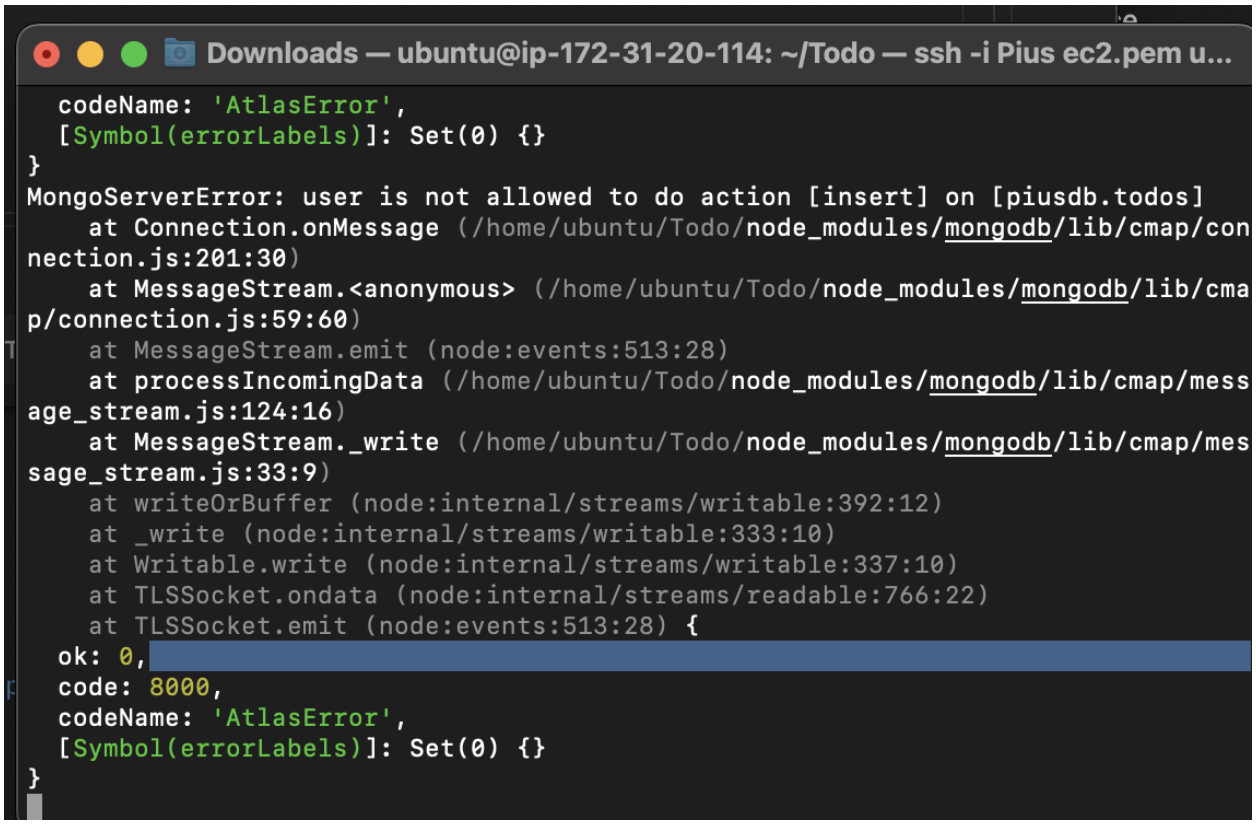



Downloads — ubuntu@ip-172-31-20-114: ~/Todo — ssh -i Pius ec2.pem u...
Swap usage: 0%
* Introducing Expanded Security Maintenance for Applications.
Receive updates to over 25,000 software packages with your
Ubuntu Pro subscription. Free for personal use.
https://ubuntu.com/aws/pro
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Last login: Fri Mar 31 20:39:09 2023 from 102.88.34.230
ubuntu@ip-172-31-20-114:~$ cd Todo
ubuntu@ip-172-31-20-114:~/Todo$ vim .env
ubuntu@ip-172-31-20-114:~/Todo$ node index.js
Server running on port 5000
Database connected successfully


Home
Workspaces
API Network
Explore
create a req
Invite
Upgrade
My Workspace
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
project mern
POST New Request
Overview
POST New Request
No Environment
project mern / New Request
Save
POST
http://54.200.24.202:5000/api/todos
Send
Params
Auth
Headers (9)
Body
Pre-req.
Tests
Settings
Cookies
raw
JSON
Beautify
"action" : "Working on project 3 MERN"
Body
Cookies
Headers (9)
Test Results
200 OK
856 ms
424 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
"action": "Working on project 3 MERN",
"_id": "6428a3e1a2ce24ddc1be3546",
"__v": 0
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash

ISSUES

1. Make sure in your mongodb mlab you change your “built in role” to read and write to any database if not you might get an error on your POSTMAN
2. You might also get an error when running node index.js saying you have a running port, all you just have to do is kill the port by getting the PID using lsof -i tcp:5000 or replace it with any port you have running.



- **FRONTEND CREATION**

Since we are done with the functionality we want from our backend and API, it is time to create a user interface for a Web client (browser) to interact with the application via API. To start out with the frontend of the To-do app, we will use the create-react-app command to scaffold our app.

1. In the same root directory as your backend code, which is the Todo directory, run: `npx create-react-app client`
   This will create a new folder in your Todo directory called client, where you will add all the react code.

Running a React App

Before testing the react app, there are some dependencies that need to be installed.

2. Install concurrently. It is used to run more than one command simultaneously from the same terminal window. `npm install concurrently --save-dev`
3. Install nodemon. It is used to run and monitor the server. If there is any change in the server code, nodemon will restart it automatically and load the new changes `npm install nodemon --save-dev`
4. In Todo folder open the package.json file, replace the 'scripts' part with this new one

```
"scripts": {

"start": "node index.js",

"start-watch": "nodemon index.js",

"dev": "concurrently \"npm run start-watch\" \"cd client && npm start\""

},
```

Configure Proxy in `package.json`

5. Change directory to 'client' `cd client`
6. Open the package.json file `vi package.json`
7. Add the key value pair in the package.json file "proxy": "http://localhost:5000" by typing "i" and pasting the value pair at the top of the existing code.

```
{
  "proxy": "http://localhost:5000",
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
```

The whole purpose of adding the proxy configuration in number 3 above is to make it possible to access the application directly from the browser by simply calling the server url like http://localhost:5000 rather than always including the entire path like http://localhost:5000/api/todos.

8. Now, ensure you are inside the Todo directory, and simply do `npm run dev`

   Thenapp should open and start running on localhost:3000, In order to be able to access the application from the Internet you have to open TCP port 3000 on EC2 by adding a new Security Group rule.

Creating your React Components

One of the advantages of react is that it makes use of components, which are reusable and also makes code modular. For the Todo app, there will be two stateful components and one stateless component.

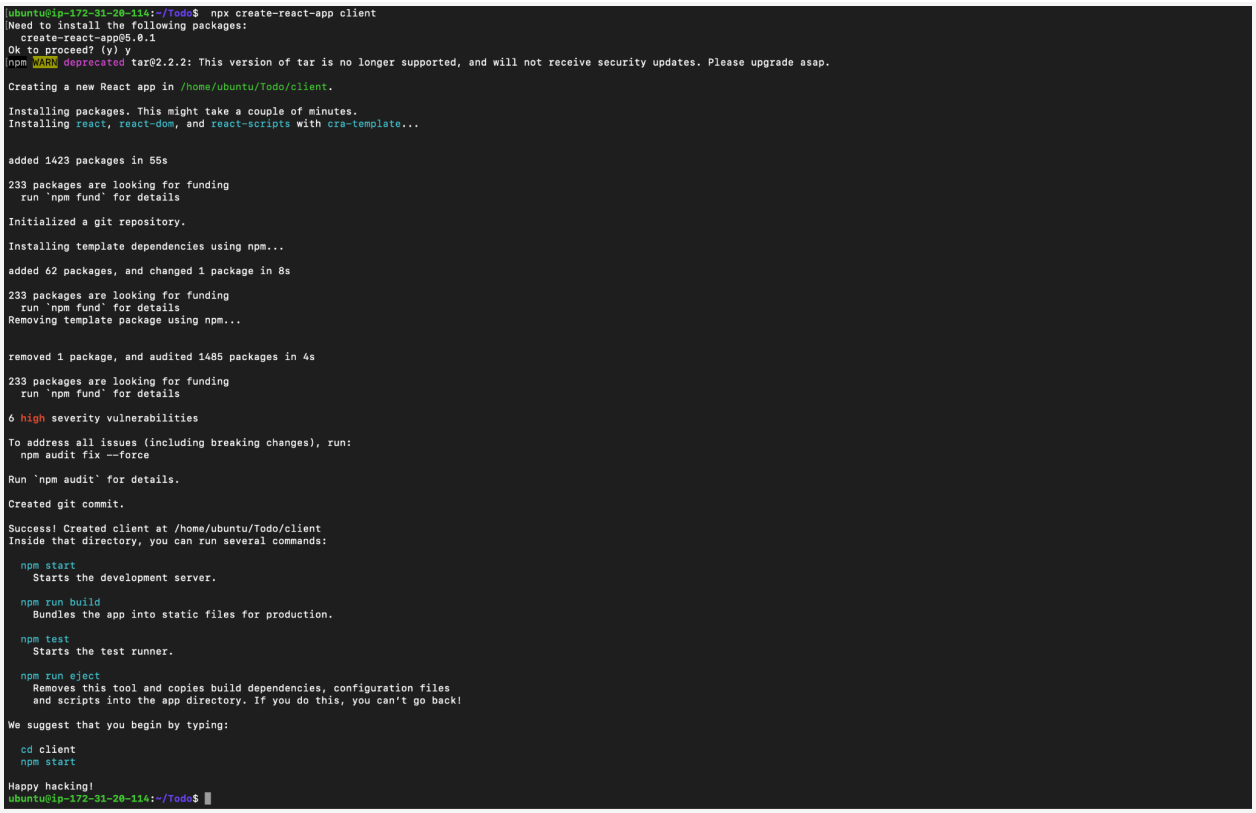
9. From your Todo directory run `cd client`
10. move to the src directory `cd src`
11. Inside your src folder create another folder called components `mkdir components`
12. Move into the components directory with `cd components`
13. Inside 'components' directory create three files Input.js, ListTodo.js and Todo.js. `touch Input.js ListTodo.js Todo.js`
14. Open Input.js file `vi Input.js`
    Copy and paste the following

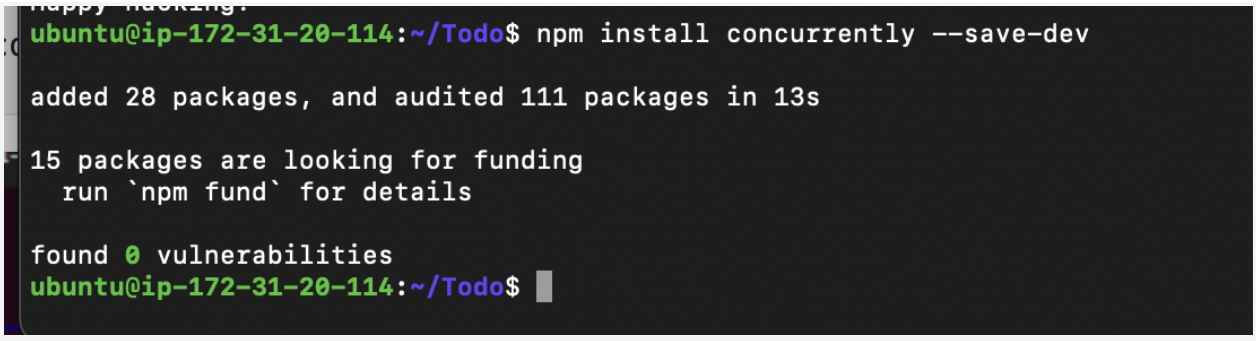
```
import React, { Component } from 'react';
import axios from 'axios';

class Input extends Component {

state = {
```

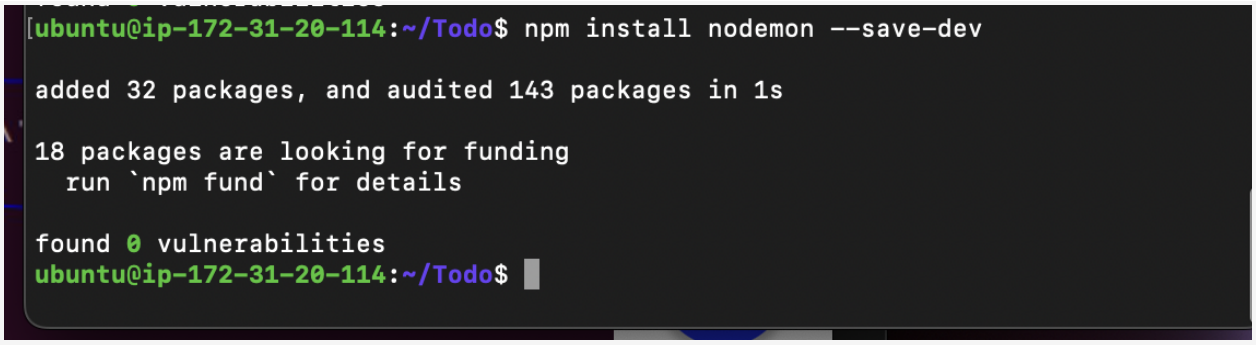
```
action: ""
}

addTodo = () => {
const task = {action: this.state.action}

    if(task.action && task.action.length > 0){
      axios.post('/api/todos', task)
        .then(res => {
          if(res.data){
            this.props.getTodos();
            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
```

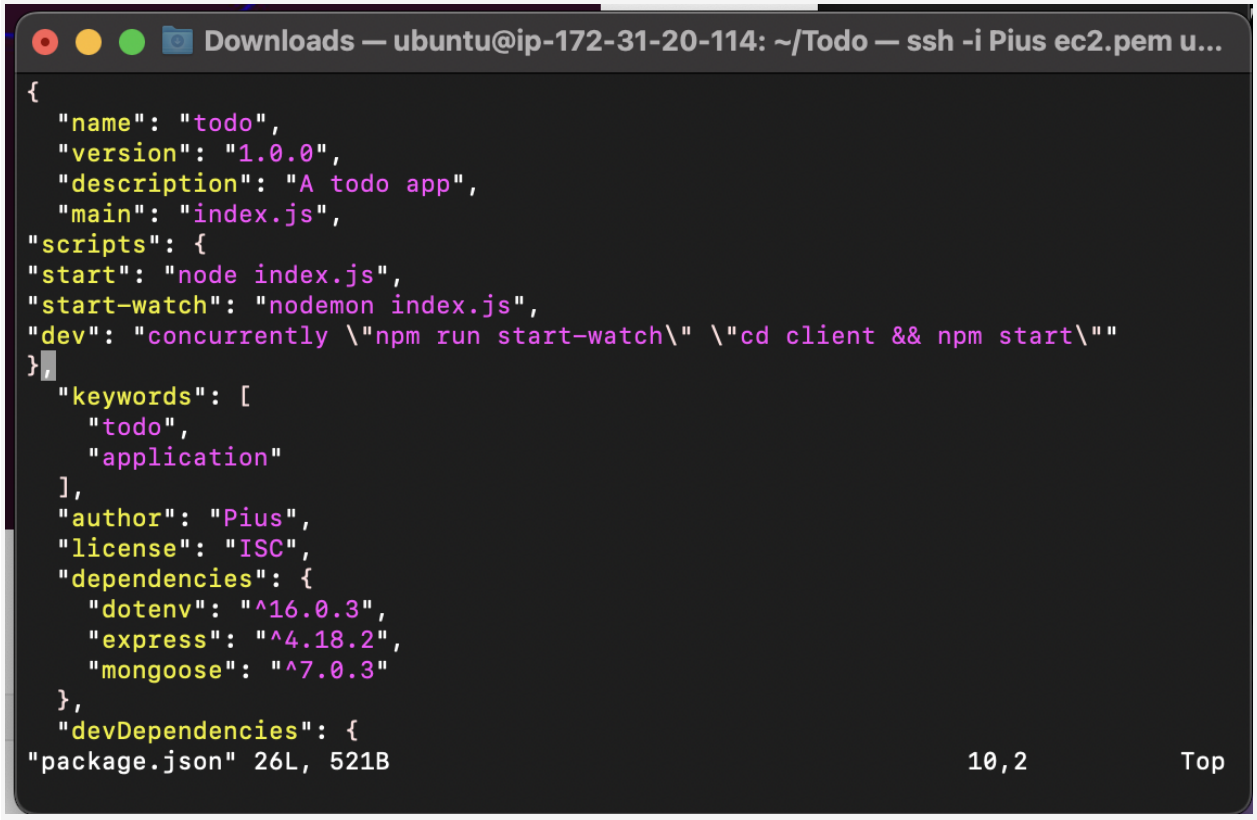
To make use of Axios, which is a Promise based HTTP client for the browser and node.js, you need to cd into your client from your terminal and run yarn add axios or npm install axios.

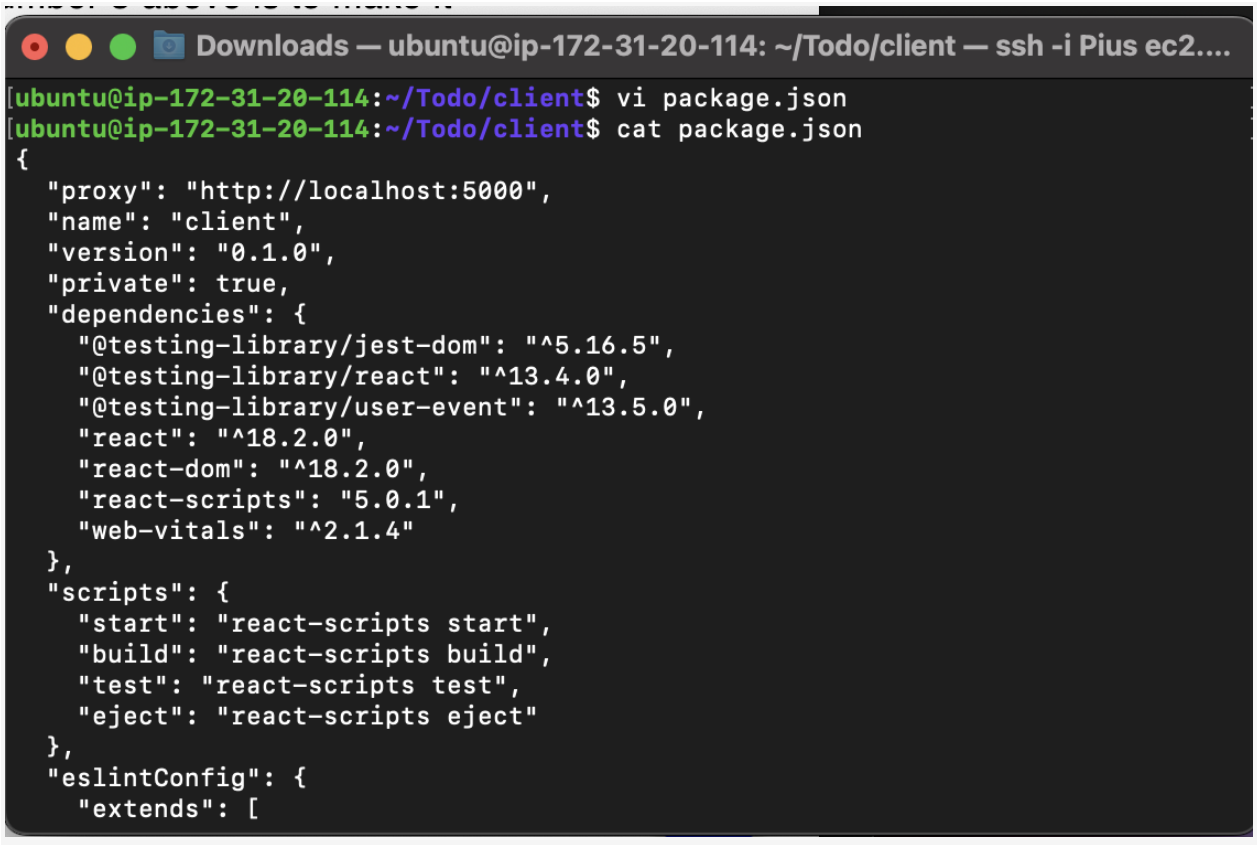
15. Move to the src folder `cd ..`
16. Move to clients folder `cd ..`
17. Install Axios `npm install axios`



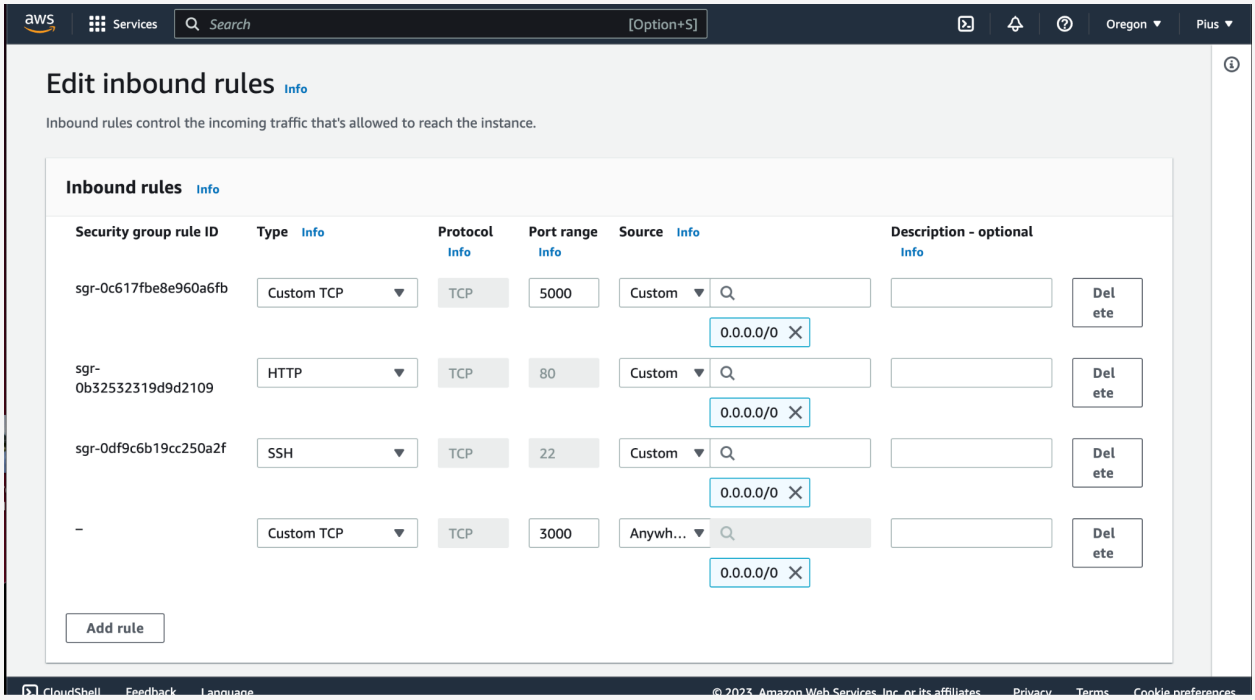




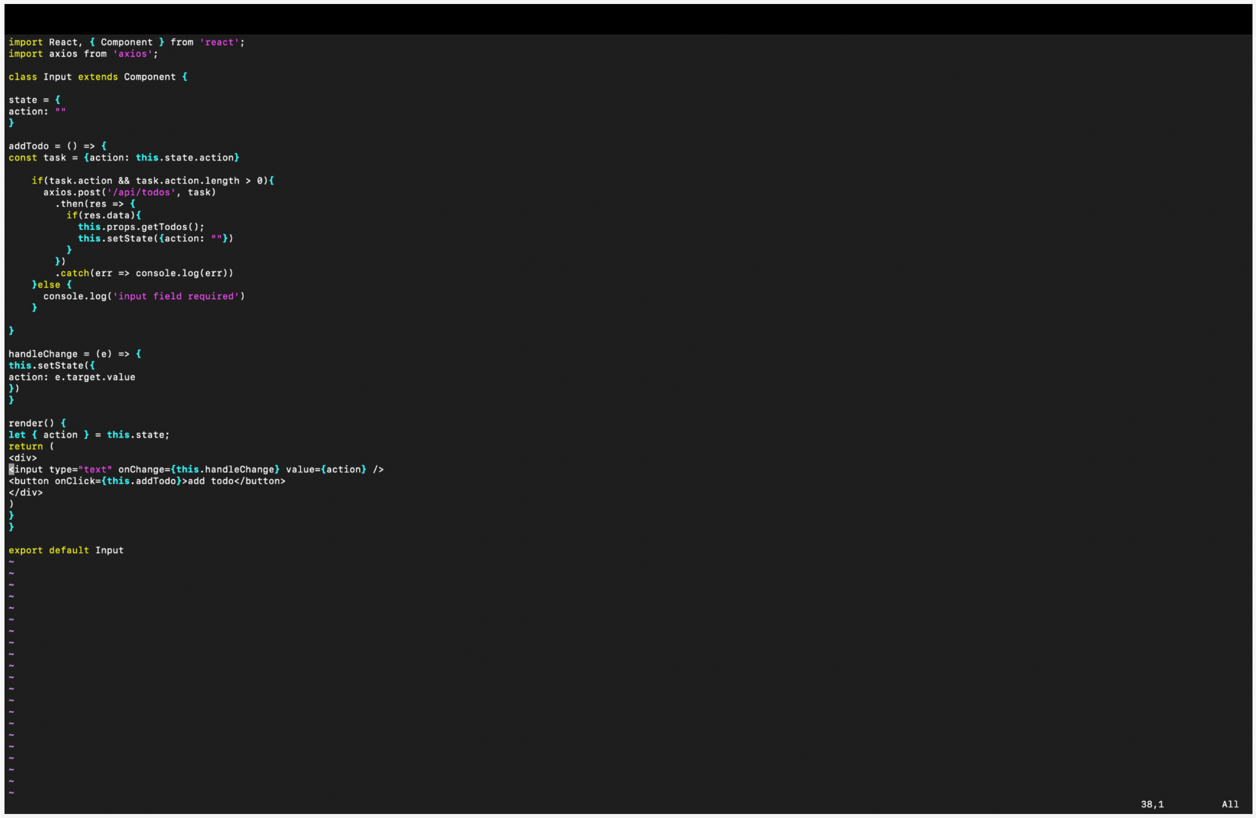

Downloads — ubuntu@ip-172-31-20-114: ~/Todo — ssh -i Pius ec2.pem u...
{
"name": "todo",
"version": "1.0.0",
"description": "A todo app",
"main": "index.js",
"scripts": {
"start": "node index.js",
"start-watch": "nodemon index.js",
"dev": "concurrently \"npm run start-watch\" \"cd client && npm start\""
},
"keywords": [
"todo",
"application"
],
"author": "Pius",
"license": "ISC",
"dependencies": {
"dotenv": "^16.0.3",
"express": "^4.18.2",
"mongoose": "^7.0.3"
},
"devDependencies": {
"package.json" 26L, 521B
10,2
Top


Downloads — ubuntu@ip-172-31-20-114: ~/Todo/client — ssh -i Pius ec2....
[ubuntu@ip-172-31-20-114:~/Todo/client$ vi package.json
[ubuntu@ip-172-31-20-114:~/Todo/client$ cat package.json
{
"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": [

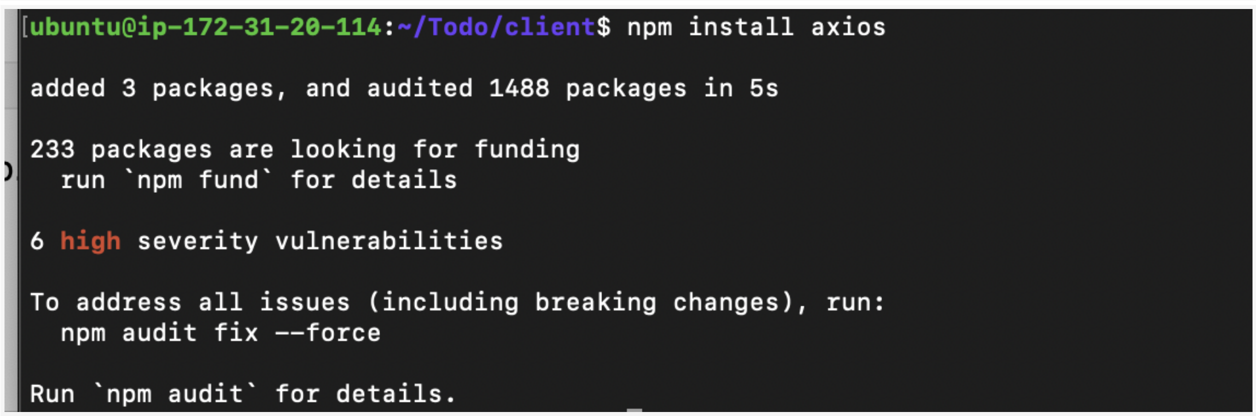
```
[ubuntu@ip-172-31-20-114:~/Todo$ npm run dev
[
> todo@1.0.0 dev
> concurrently "npm run start-watch" "cd client && npm start"

[0]
[0] > todo@1.0.0 start-watch
[0] > nodemon index.js
[0]
[1]
[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:1321) [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:1321) [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://172.31.20.114:3000
[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
```




```
[ubuntu@ip-172-31-20-114:~$ cd Todo
[ubuntu@ip-172-31-20-114:~/Todo$ cd client
[ubuntu@ip-172-31-20-114:~/Todo/client$ cd src
[ubuntu@ip-172-31-20-114:~/Todo/client/src$ mkdir components
[ubuntu@ip-172-31-20-114:~/Todo/client/src$ cd compoents
-bash: cd: compoents: No such file or directory
[ubuntu@ip-172-31-20-114:~/Todo/client/src$ cd components
[ubuntu@ip-172-31-20-114:~/Todo/client/src/components$ touch Input.js ListTodo.js
 Todo.js
ubuntu@ip-172-31-20-114:~/Todo/client/src/components$
```



```
import React, { Component } from 'react';
import axios from 'axios';

class Input extends Component {

state = {
action: ""
}

addTodo = () => {
const task = {action: this.state.action}

    if(task.action && task.action.length > 0){
      axios.post('/api/todos', task)
        .then(res => {
          if(res.data){
            this.props.getTodos();
            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
```



```
ubuntu@ip-172-31-20-114:~/Todo/client$ npm install axios

added 3 packages, and audited 1488 packages in 5s

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

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

- **FRONTEND CREATION (continuation)**

1. Go to ‘components’ directory `cd src/components`
2. After that open your ListTodo.js `vi ListTodo.js`
   in the ListTodo.js copy and paste the following code

```
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
```

3. Then in your Todo.js file you write the following code

```
        import React, {Component} from 'react';

        import axios from 'axios';



        import Input from './Input';

        import ListTodo from './ListTodo';



        class Todo extends Component {



        state = {
```

```
todos: []

}


componentDidMount(){

this.getTodos();

}


getTodos = () => {

axios.get('/api/todos')

.then(res => {

if(res.data){

this.setState({

todos: res.data

})

}

})

.catch(err => console.log(err))

}


deleteTodo = (id) => {
```

```
    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;
```

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

4. Move to the src folder `cd ..`
5. Make sure that you are in the src folder and run `vi App.js`

Copy and paste the code below into it

```
import React from 'react';


import Todo from './components/Todo';

import './App.css';


const App = () => {

return (

<div className="App">

<Todo />

</div>

);

}


export default App;
```

After pasting, exit the editor. :wa

6. In the src directory open the App.css `vi App.css`

   Then paste the following code into App.css

```
.App {
text-align: center;
font-size: calc(10px + 2vmin);
width: 60%;
margin-left: auto;
margin-right: auto;
}

input {
height: 40px;
width: 50%;
border: none;
border-bottom: 2px #101113 solid;
background: none;
font-size: 1.5rem;
color: #787a80;
}
```

```
input:focus {
outline: none;
}

button {
width: 25%;
height: 45px;
border: none;
margin-left: 10px;
font-size: 25px;
background: #101113;
border-radius: 5px;
color: #787a80;
cursor: pointer;
}

button:focus {
outline: none;
}
```

```
ul {
list-style: none;
text-align: left;
padding: 15px;
background: #171a1f;
border-radius: 5px;
}

li {
padding: 15px;
font-size: 1.5rem;
margin-bottom: 15px;
background: #282c34;
border-radius: 5px;
overflow-wrap: break-word;
cursor: pointer;
}

@media only screen and (min-width: 300px) {
.App {
width: 80%;
```

```
}

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;

    }

}
```

Exit

7. In the src directory open the index.css `vim index.css`
   Copy and paste the code below

```
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;
}
```

8. Go to the Todo directory `cd ../..`
9. When you are in the Todo directory run `npm run dev`

```
ubuntu@ip-172-31-20-114:~/Todo/client/src/components$ vi ListTodo.js
ubuntu@ip-172-31-20-114:~/Todo/client/src/components$ cat 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
```

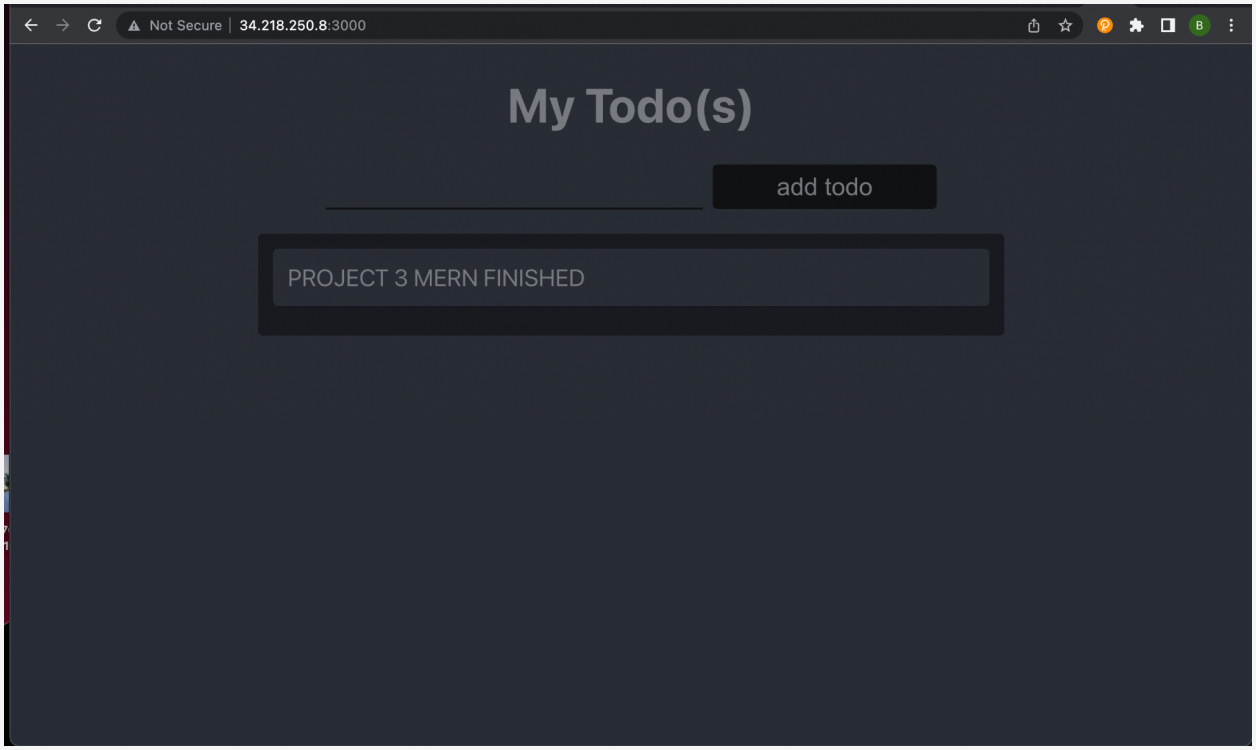
```
ubuntu@ip-172-31-20-114:~/Todo/client/src/components$ vi Todo.js
ubuntu@ip-172-31-20-114:~/Todo/client/src/components$ cat Todo.js
import React, {Component} from 'react';
import axios from 'axios';

import Input from './Input';
import ListTodo from './ListTodo';

class Todo extends Component {

state = {
todos: []
}

componentDidMount(){
this.getTodos();
}

getTodos = () => {
axios.get('/api/todos')
.then(res => {
if(res.data){
this.setState({
todos: res.data
})
}
})
.catch(err => console.log(err))
}

deleteTodo = (id) => {

    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;
```

```
ubuntu@ip-172-31-20-114:~/Todo/client/src$ vi App.js
[ubuntu@ip-172-31-20-114:~/Todo/client/src$ cat App.js
import React from 'react';

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

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

export default App;

ubuntu@ip-172-31-20-114:~/Todo/client/src$
```

```
ubuntu@ip-172-31-20-114:~/Todo/client/src$ vi App.css
ubuntu@ip-172-31-20-114:~/Todo/client/src$ cat App.css
.App {
text-align: center;
font-size: calc(10px + 2vmin);
width: 60%;
margin-left: auto;
margin-right: auto;
}

input {
height: 40px;
width: 50%;
border: none;
border-bottom: 2px #101113 solid;
background: none;
font-size: 1.5rem;
color: #787a80;
}

input:focus {
outline: none;
}

button {
width: 25%;
height: 45px;
border: none;
margin-left: 10px;
font-size: 25px;
background: #101113;
border-radius: 5px;
color: #787a80;
cursor: pointer;
}

button:focus {
outline: none;
}

ul {
list-style: none;
text-align: left;
padding: 15px;
background: #171a1f;
border-radius: 5px;
}

li {
padding: 15px;
font-size: 1.5rem;
margin-bottom: 15px;
background: #282c34;
border-radius: 5px;
overflow-wrap: break-word;
cursor: pointer;
}

@media only screen and (min-width: 300px) {
.App {
width: 80%;
}

input {
width: 100%
}
```

```
ubuntu@ip-172-31-20-114:~/Todo/client/src$ vim index.css
ubuntu@ip-172-31-20-114:~/Todo/client/src$ cat index.css
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;
}
```

```
ubuntu@ip-172-31-20-114:~/Todo/client/src$ cd ../..
ubuntu@ip-172-31-20-114:~/Todo$ npm run dev

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

[0]
[0] > todo@1.0.0 start-watch
[0] > nodemon index.js
[0]
[1]
[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:1794) [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:1794) [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://172.31.20.114:3000
[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
```



Assuming no errors when saving all these files, the To-Do app should be ready and fully functional with the functionality discussed earlier: creating a task, deleting a task and viewing all your tasks.