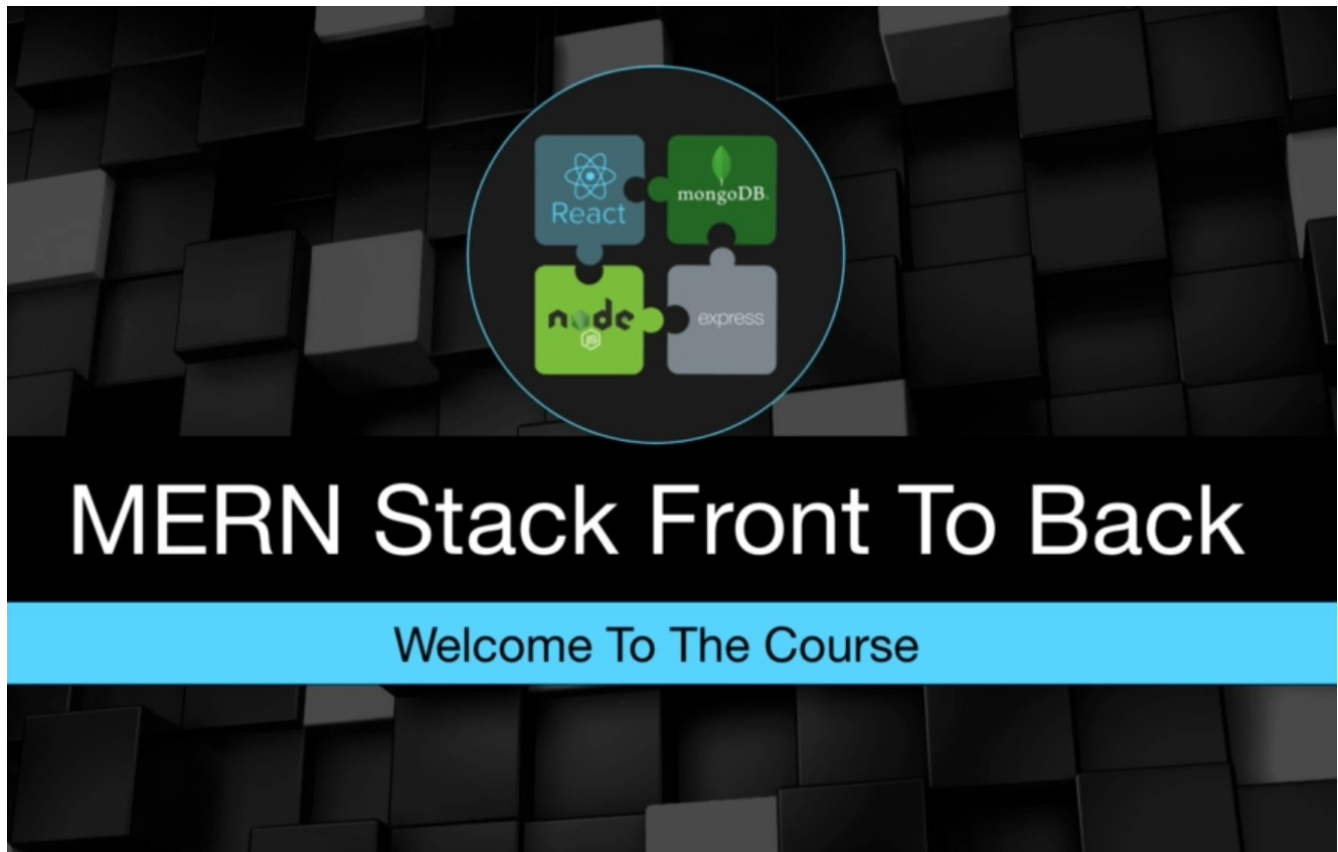


MERN Stack Front To Back: Full Stack React, Redux & Node.js (2020)



<https://www.udemy.com/course/mern-stack-front-to-back/>

by [Brad Traversy](#)

Table of Contents

0. Introduction.....	3
0.1 Environment & Setup.....	3
0.1.1 Node.js.....	3
0.1.2 Visual Studio Code (VSC).....	3
0.1.3 <i>GIT</i>	3
0.1.4 <i>Postman</i>	4
0.1.5 <i>React Developer Tools chrome extension</i>	4
0.1.6 <i>Redux DevTools chrome extension</i>	4
0.1.7 <i>VSC extensions</i>	4
1. Express & MongoDB Setup.....	5
1.1 MongoDB (create a new cluster for project).....	5
1.1 Express (install a package and setup a server).....	6
1.2 Create a connection with mongoDB.....	9
1.3 Create routes.....	11
What was used in project?.....	14

0. Introduction



Modern Technologies Used

- VSCode Editor
- ES6+ Syntax
- Async / Await
- React Hooks
- Redux With DevTools
- JWT (JSON Web Tokens)
- Postman HTTP Client
- Mongoose / MongoDB / Atlas
- Bcrypt Password Hashing
- Heroku & Git Deployment

0.1 Environment & Setup

0.1.1 Node.js

Windows: <https://nodejs.org/en/> - just download and install with GUI.

Linux:

```
CLI=> sudo apt-get update
```

```
CLI=> sudo apt install curl build-essential
```

```
CLI=> curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
```

```
CLI=> sudo apt install -y nodejs
```

```
CLI=> node -v [=> v14.16.1]
```

0.1.2 Visual Studio Code (VSC)

Windows: <https://code.visualstudio.com/> - just download and install with GUI.

Linux:

```
CLI=> sudo apt-get update
```

```
CLI=> sudo apt install code
```

0.1.3 GIT

Windows: <https://git-scm.com/> just download and install with GUI.

Linux: **GIT is a basic component**

0.1.4 Postman

Windows: <https://www.postman.com/> - just download and install with GUI.

Linux: <https://www.postman.com/downloads/> - download archive tar.gz
<https://dl.pstmn.io/download/latest/linux64>

go to download folder and unpack that file with command `tar xvf [PACKAGENAME].tar.gz`

CLI=> `tar xvf Postman-linux-x64-8.6.1.tar.gz`

go to folder and use shortcut

0.1.5 React Developer Tools chrome extension

0.1.6 Redux DevTools chrome extension

0.1.7 VSC extensions

- Bracket Pair Colorizer - <https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer>
- ES7 React/Redux/GraphQL/React-Native snippets - <https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets>
- Prettier - Code formatter - <https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode>

VSC=>Manage=>Settings=> “format on save” should be enable

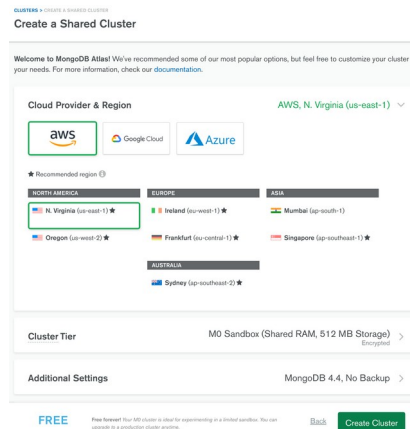
1. Express & MongoDB Setup

1.1 MongoDB (create a new cluster for project)

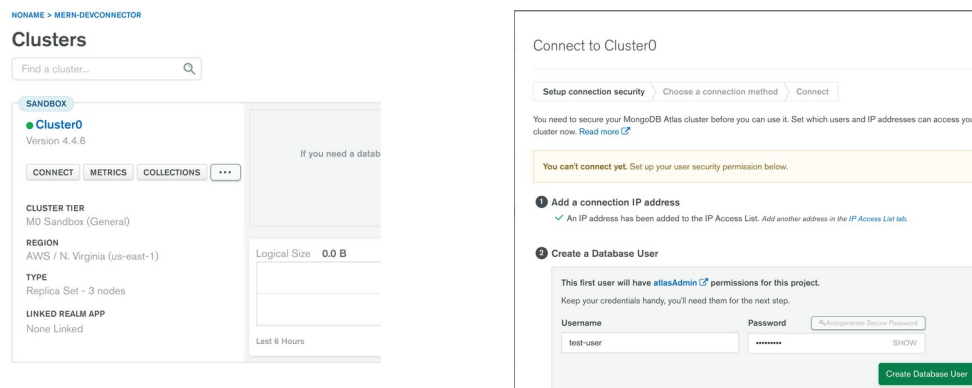
Create an account at <https://www.mongodb.com/>. Sign in to an account.

Create a new project – **MERN-devconnector**

Create a new cluster - **Cluster0**

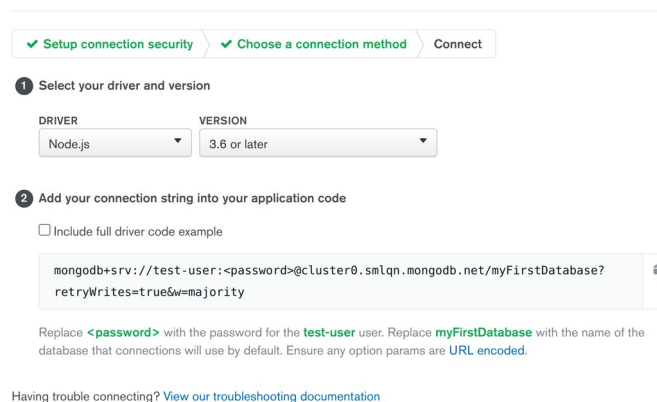


Push button “CONNECT” and add your IP adress and create a new database user to connect to a DB.



Push button “Choose a connection method” and “connect your application” and save that link

Connect to Cluster0



1.1 Express (install a package and setup a server)

1.1.1 Create a .gitignore file

It is a file with list of files/folders which will be ignored by GIT. Add **node_modules** folder

1.1.2 Initialize an NPM project and setup entry point

CLI=> *npm init -y*

```
stslon@stslon-System-Product-Name:/media/stslon/860_2/june2021/git-projects/mern2$ npm init -y
Wrote to /media/stslon/860_2/june2021/git-projects/mern2/package.json:

{
  "name": "mern2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/ILopatenko/mern2.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ILopatenko/mern2/issues"
  },
  "homepage": "https://github.com/ILopatenko/mern2#readme"
}
```

This command creates a new file package.json with basic information about project and all the dependencies. I'm going to change entry point ("main": "server.js") for this project to **server.js**

1.1.3 Install all the packages as regular dependencies

I'm going to use in this project next packages:

- **express** – backend server;
- **express-validator** – for validation data;
- **bcryptjs** – for making hash for passwords;
- **config** – for making global variables;
- **gravatar** – for working with user's avatars;
- **jsonwebtoken** – for working with JWT;
- **mongoose** – for working with MongoDB database;
- **request** – for working with another API based services;

CLI=> *npm install express express bcryptjs config gravatar jsonwebtoken mongoose request*

1.1.4 Install all the packages as DEV dependencies

DEV dependencies will be installed without direct reference to a project (npm will add them to package.json file as a devDependencies)

- **nodemon** – will track all the changes and make a restart server automatically;
- **concurrently** – allows me to run few scripts at the same time with a single command;

CLI=> `npm install -D nodemon concurrently`

```
{ } package.json U X
package.json > ...
1  {
2    "name": "mern2",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "repository": {
10     "type": "git",
11     "url": "git+https://github.com/ILopatenko/mern2.git"
12   },
13   "keywords": [],
14   "author": "",
15   "license": "ISC",
16   "bugs": {
17     "url": "https://github.com/ILopatenko/mern2/issues"
18   },
19   "homepage": "https://github.com/ILopatenko/mern2#readme",
20   "dependencies": {
21     "bcryptjs": "^2.4.3",
22     "config": "^3.3.6",
23     "express": "^4.17.1",
24     "gravatar": "^1.8.1",
25     "jsonwebtoken": "^8.5.1",
26     "mongoose": "^5.12.13",
27     "request": "^2.88.2"
28   },
29   "devDependencies": {
30     "concurrently": "^6.2.0",
31     "nodemon": "^2.0.7"
32   }
33 }
```

1.1.5 Create an express server

```
JS server.js U X
_docs > JS server.js > ...
1  const express = require('express');
2
3  const app = express();
4
5  app.get('/', (req, res) => res.send('API is running'));
6
7  const PORT = process.env.PORT || 5000;
8
9  app.listen(PORT, () => console.log(`Server is started on port ${PORT}`));
10
```

1.1.6 Change run script at package.json

```
"scripts": {  
  "start": "node server.js",  
  "server": "nodemon server.js"  
},
```

Now command “npm run start” will run server.js with node.js and command “npm run server” will run server.js with nodemon

1.1.7 The first run

The image displays three screenshots illustrating the initial setup and execution of a web server.

Terminal Screenshot: Shows the command prompt where the user runs `npm run server`. The output indicates that nodemon is watching for changes and has started the server on port 5000.

```
stslon@stslon-System-Product-Name:/media/stslon/860_2/june2021/git-projects/mern2$ npm run server  
> mern2@1.0.0 server /media/stslon/860_2/june2021/git-projects/mern2  
> nodemon server.js  
  
[nodemon] 2.0.7  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node server.js`  
Server is started on port 5000
```

Browser Screenshot: A web browser window at `localhost:5000` displays the message "API is running".

Postman Screenshot: The Postman application shows a GET request to `localhost:5000`. The response status is 200 OK, and the body contains the text "API is running".

1.2 Create a connection with mongoDB.

Create a new folder **config**. Inside this folder create a new file – **default.json** – with information to connect with a database

```
{ default.json U X
config > {} default.json > ...
1  {}
2  "mongoURI": "mongodbsrv://test-user:test-user@cluster0.smlqn.mongodb.net/myFirstDatabase?retryWrites=true&w=majority"
3  }
```

Create a new file – **db.js** – with connection logic

```
JS db.js U •
config > JS db.js > ...
1  const mongoose = require('mongoose');
2
3  const config = require('config');
4
5  const db = config.get('mongoURI');
6
7  const connectDB = async () => {
8    try {
9      await mongoose.connect(db);
10     console.log('MongoDB is connected!');
11    } catch (error) {
12     console.error(err.message);
13     process.exit(1);
14    }
15  };
16
17  module.exports = connectDB;
```

Add **db.js** to a **server.js**

```
JS server.js M X
JS server.js > ...
1  const express = require('express');
2  const connectToDB = require('./config/db');
3
4  const app = express();
5  connectToDB();
6
7  app.get('/', (req, res) => res.send('API is running'));
8
9  const PORT = process.env.PORT || 5000;
10
11  app.listen(PORT, () => console.log(`Server is started on port ${PORT}`));
12
```

```
stslon@stslon-System-Product-Name:/media/stslon/860_2/june2021/git-projects/mern2$ npm run server
> mern2@1.0.0 server /media/stslon/860_2/june2021/git-projects/mern2
> nodemon server.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
(node:49585) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version.
(Use `node --trace-deprecation ...` to show where the warning was created)
Server is started on port 5000
(node:49585) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:49585) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new MongoClient, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB is connected!
```

Fix all the warnings and run a server again:

```
JS db.js U X
config > JS db.js > ...
1  const mongoose = require('mongoose');
2
3  const config = require('config');
4
5  const db = config.get('mongoURI');
6
7  const connectDB = async () => {
8    try {
9      await mongoose.connect(db, {
10        useNewUrlParser: true,
11        useUnifiedTopology: true,
12      });
13      console.log('MongoDB is connected!');
14    } catch (error) {
15      console.error(err.message);
16      process.exit(1);
17    }
18  };
19
20  module.exports = connectDB;
21
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
stslon@stslon-System-Product-Name:/media/stslon/860_2/june2021/git-projects/mern2$ npm run server
> mern2@1.0.0 server /media/stslon/860_2/june2021/git-projects/mern2
> nodemon server.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server is started on port 5000
MongoDB is connected!
```

1.3 Create routes.

Create a new folder – **routes/api** – to store and work with all the routes.

Inside this folder create new files – **user.js**, **profile.js**, **post.js** and **auth.js**

```
JS user.js U X
routes > api > JS user.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/user
5  //@desc       Test route
6  //@access     Public
7  router.get('/', (req, res) => res.send('User route'));
8
9  module.exports = router;
```

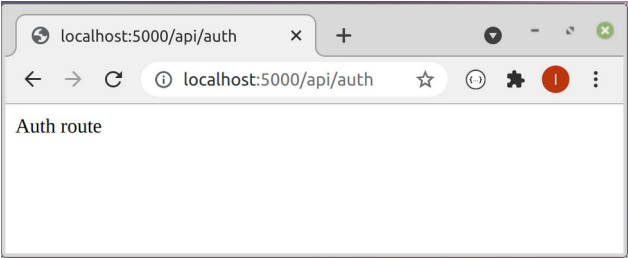
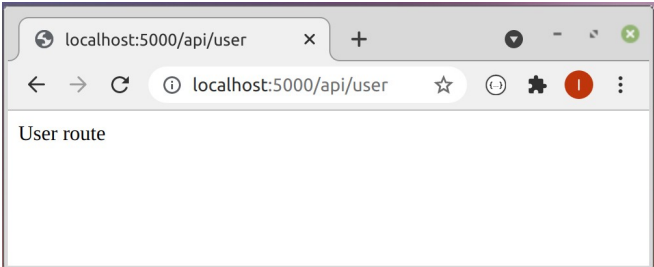
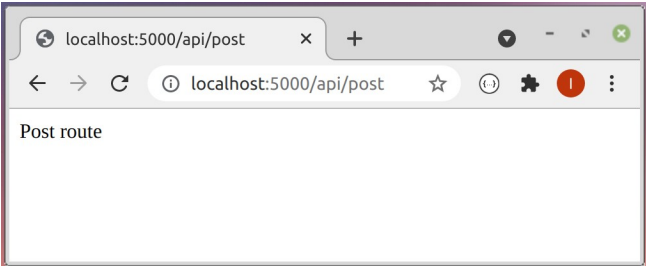
```
JS profile.js U X
routes > api > JS profile.js > router.get('/') callback
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/profile
5  //@desc       Test route
6  //@access     Public
7  router.get('/', [req, res] => res.send('Profile route'));
8
9  module.exports = router;
```

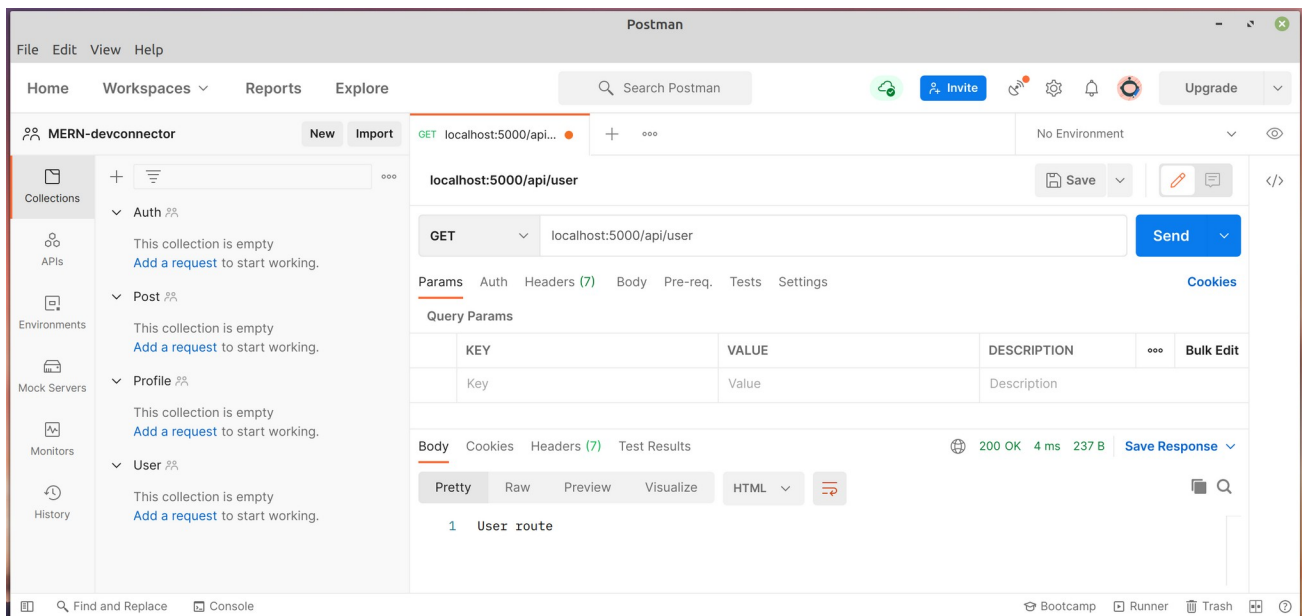
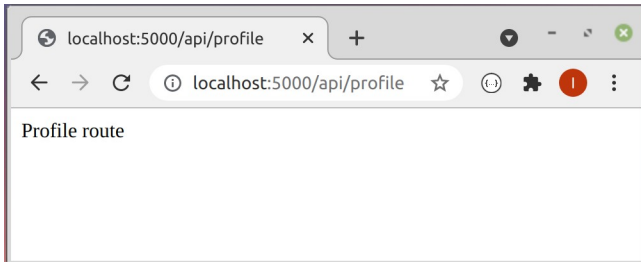
```
JS post.js U X
routes > api > JS post.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/post
5  //@desc       Test route
6  //@access     Public
7  router.get('/', (req, res) => res.send('Post route'));
8
9  module.exports = router;
```

```
JS auth.js U X
routes > api > JS auth.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/auth
5  //@desc       Test route
6  //@access     Public
7  router.get('/', (req, res) => res.send('Auth route'));
8
9  module.exports = router;
```

Add all these routes to **server.js**

```
JS server.js M X
JS server.js > ...
1  const express = require('express');
2
3  const connectToDB = require('./config/db');
4
5  const userRoute = require('./routes/api/user');
6  const profileRoute = require('./routes/api/profile');
7  const postRoute = require('./routes/api/post');
8  const authRoute = require('./routes/api/auth');
9
10 const app = express();
11 connectToDB();
12
13 app.get('/', (req, res) => res.send('API is running'));
14
15 //define all the routes
16 app.use('/api/user', userRoute);
17 app.use('/api/profile', profileRoute);
18 app.use('/api/post', postRoute);
19 app.use('/api/auth', authRoute);
20
21 const PORT = process.env.PORT || 5000;
22
23 app.listen(PORT, () => console.log(`Server is started on port ${PORT}`));
24
```





What was used in project?

Operating Systems: Linux Mint 20.1 and Windows 10

Node.js – javascript runtime enviroment.

NPM

GIT -

MongoDB

Mongoose

nodemon

postman