Create an HTTPS NodeJS Web Service with Express

When hosting NodeJS web applications with external APIs, it's essential to keep communication secure. By setting up a popular NodeJS web framework called Express and configuring API endpoints to communicate via HTTPS NodeJS is a great way to do that.

In this lecture, you will learn how to install and configure the Express NodeJS application framework and set up an encrypted API endpoint to keep communication encrypted.

Let's get started!

Prerequisites

To follow along, be sure you have the following in place:

- A Linux machine – This tutorial will use Ubuntu 20.04.
- NodeJS – NodeJS acts as the runtime environment for all of your Express projects.

Creating a NodeJS Project

Once you have NodeJS installed on your server, it's time to set up Express. The easiest way to install Express is via a NodeJS package. But before you do that, first create a directory for this demo project. This directory will be located in ~/NodejsHTTPSServer

```
mkdir ~/NodejsHTTPSServer
```

Next, initialize a new NodeJS project using `npm init`. This command will create a package.json file in the directory you just created, recording necessary metadata about a project you are building.

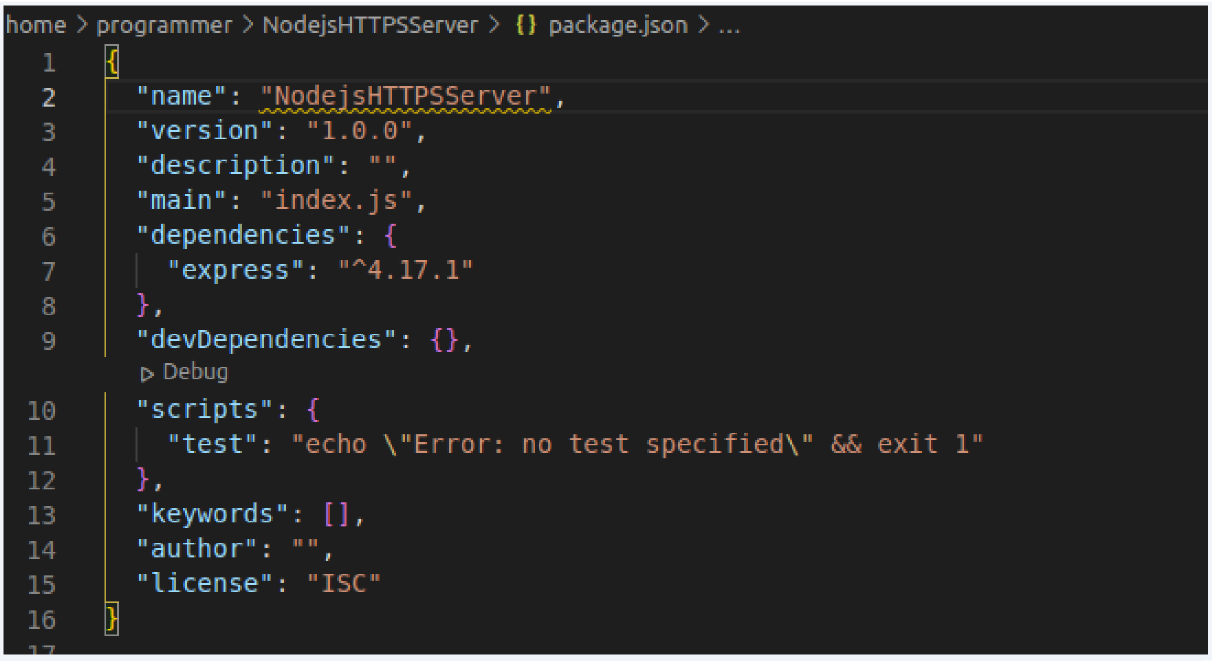
Use the `-y` flag, as shown below to accept the default values. You do not need any unique NodeJS project settings for this tutorial. This action will create a NodeJS project with the same name as the folder (`NodejsHTTPSServer`).

```
npm init -y
```

Installing Express

Now it’s time to install the Express package. To do so, run the command below to install the NodeJS Express package. This command will install the NodeJS Express package save the information in the *package.json* file, as shown below.

```
npm install express
```


home > programmer > NodejsHTTPSServer > {} package.json > ...

```json
{
  "name": "NodejsHTTPSServer",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "dependencies": {
    "express": "^4.17.1"
  },
  "devDependencies": {},
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Creating a Web Service

It’s now time to create a web service/server with Express!

1. To get started, ensure you’re still in the *~/NodejsHTTPSServer* directory and create a blank file called *index.js*. This file will be a Javascript script that will hold all necessary code that NodeJS will execute when launching the web service.

```
touch index.js
```

2. Next, open the *index.js* file and add the following code to it. This file leverages both built-in NodeJS modules and the installed `express` module to bring up a web service listening on port 4000 when executed.

```
// Import builtin NodeJS modules to instantiate the service

const https = require("https");

const fs = request("fs");
```

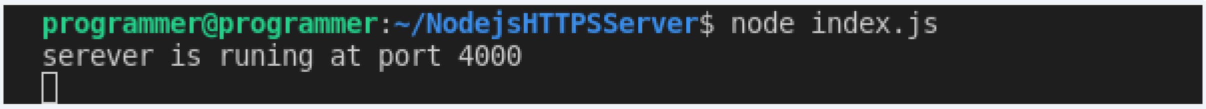
```
// Import the express module

const express = require("express");



// Instantiate an Express application

const app = express();



// Create a NodeJS HTTPS listener on port 4000 that points to the Express
app

// Use a callback function to tell when the server is created.

https

  .createServer(app)

  .listen(4000, ()=>{

    console.log('server is runing at port 4000')

  });



// Create an try point route for the Express app listening on port 4000.

// This code tells the service to listed to any request coming to the /
route.
```

```
// Once the request is received, it will display a message "Hello from
express server."

app.get('/', (req,res)=>{

    res.send("Hello from express server.")

})
```

3. Once you've created the Express project, tell NodeJS to run it to bring up the web service.

```
node index.js
```

If successful, you will see the callback function defined above invoked to return a message letting you know it has started.



```
programmer@programmer:~/NodejsHTTPSServer$ node index.js
serever is runing at port 4000
```

Running express Server

4. Finally, test the service by bringing up a web browser and navigating to *http://localhost:4000/* in your browser. If successful, you should see a message "Hello from express server.".

Creating an SSL Certificate

At this point, your NodeJS app doesn't yet support HTTPS. It's working with unencrypted HTTP, but that's not secure. Let's change that.

To configure an SSL certificate for our NodeJS HTTPS implementation, you can either use a public, trusted certificate or a self-signed certificate. This tutorial will use a self-signed certificate signed locally by the host as it's much easier for proof of concept projects.

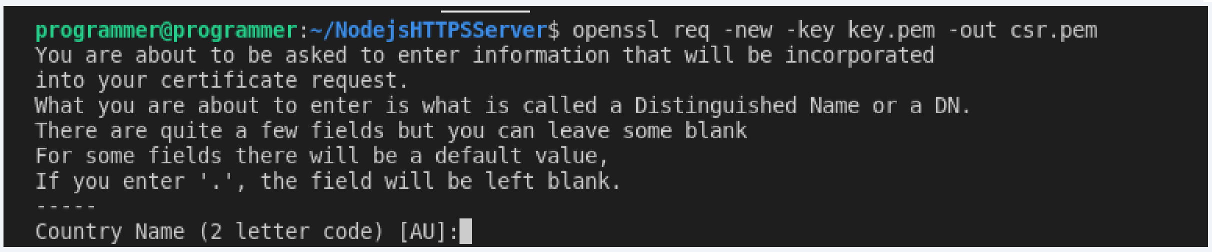
If you're running a NodeJS HTTPS application with Express in a production environment, always be sure to acquire and install a trusted certificate!

1. First, generate a key file used for self-signed certificate generation with the command below. The command will create a private key as a file called *key.pem*.

```
openssl genrsa -out key.pem
```

2. Next, generate a certificate service request (CSR) with the command below. You'll need a CSR to provide all of the input necessary to create the actual certificate.

```
openssl req -new -key key.pem -out csr.pem
```



```
programmer@programmer:~/NodejsHTTPSServer$ openssl req -new -key key.pem -out csr.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
```

3. Finally, generate your certificate by providing the private key created to sign it with the public key created in step two with an expiry date of 9,999 days. This command below will create a certificate called *cert.pem*.

```
openssl x509 -req -days 9999 -in csr.pem -signkey key.pem -out cert.pem
```

Once complete, your project folder should look like below:

```
NodejsHTTPSServer

      /cert.pem

      /csr.pem
      /index.js

      /key.pem

      /package-lock.json

      /package.json
```

Enabling HTTPS Nodejs in Express

At this point, your SSL certificate has been successfully created. Now it's time to enable HTTPS with NodeJS and Express. To enable HTTPS requires slightly modifying the Express app you created earlier.

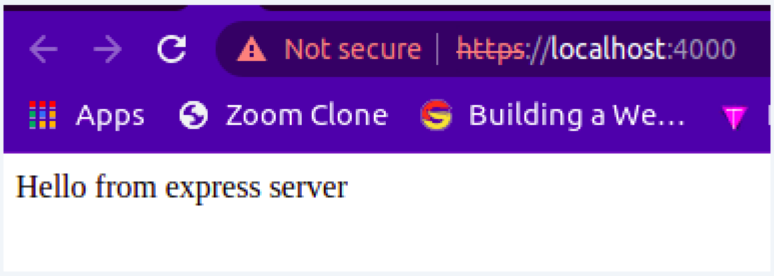
Open the *index.js* file you created earlier and modify the `createServer()` method, as shown below. You'll notice that you must provide the private key (`key.pem`) and the public key/certificate (`cert.pem`) to the `createServer()` method.

The below example reads both private and public keys, which only execute once when the service starts.

```
https

  .createServer(

            // Provide the private and public key to the server by reading
each

            // file's content with the readFileSync() method.

    {

      key: fs.readFileSync("key.pem"),
      cert: fs.readFileSync("cert.pem"),

    },

    app

  )

  .listen(4000, () => {

    console.log("serever is runing at port 4000");

  });
```

With that done, your express HTTPS server has been successfully created. Now test the application in your browser by going to https://localhost:4000.

You should still get the same response of “Hello from express server” on your browser.



Hello from express server

Note that if you can’t initially browse to the web service with HTTPS, you may need to terminate and restart the server.

Conclusion

You should now understand how to set up a NodeJS HTTPS web service using the Express framework. By simply installing the Express NodeJS package and creating a simple configuration script, you can have a secure web service running over HTTPS.