EE4500-CNS-2025-BL2 LAB 4 LoRaWAN Dashboard

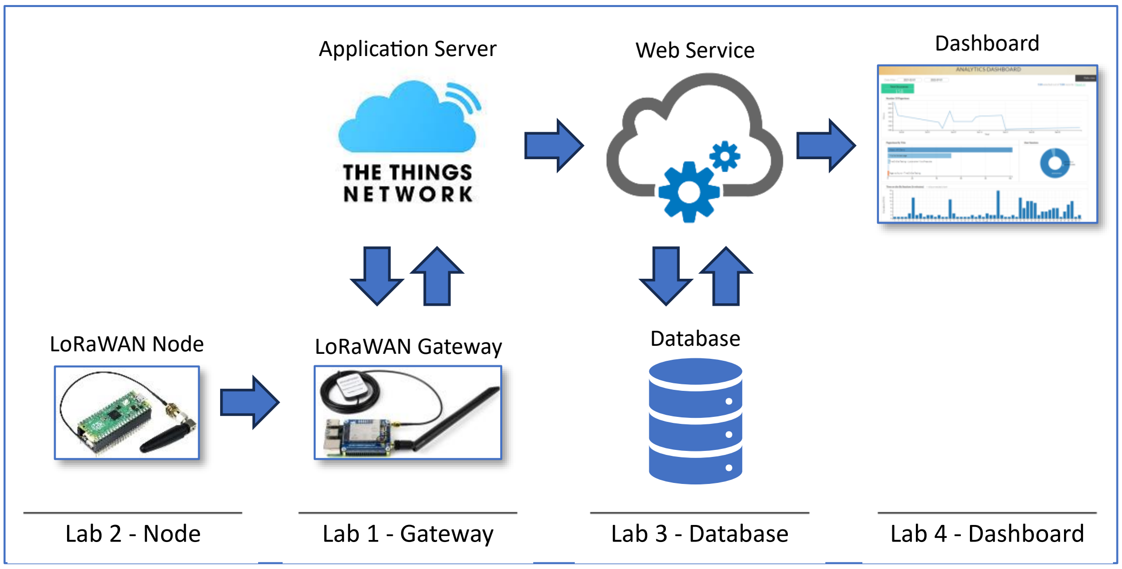
Course	24-CC4950-CNS-INT-SP2
Unit	Lab 4
Title	LoRaWAN Database
Version	1.3
Author(s)	Bruce Belson
Date created	18-Jun-2024
Date updated	8-Mar-2025

Contents

Objectives

The purpose of this lab exercise is to create a Web Dashboard to display data transmitted data from a LoRaWAN Node via The Things Network and stored in a relational database hosted on Amazon Web Services.

i. The node kit from earlier Labs is reconnected to the existing TTN application. The TTN application is connected to a shared AWS-hosted web server & database, by amending the application’s Webhook integration.
ii. A static web site is built, based on a supplied template, which downloads time-series data from the shared web server and displays it graphically.
iii. The web site is enhanced.
iv. The web site is delivered to a AWS EC2 instance.



This lab delivers the remaining portion of the cloud infrastructure for the complete distributed application.

Deliverables

There are two target deliverables for the lab:

i) A web dashboard, built using React, which displays your sensor data, and that of the other members of the class. The dashboard should update in real-time as new data is generated by the Pico sensor boxes. (This code should be delivered as a zip file of your web-site's source code.)

ii) A screenshot of the dashboard functioning.

Deliverable 1: LoRaWAN network dashboard

Zip up all the files in your web server development folder, excluding the node_modules and dist sub-folders.

Deliverable 2: Screenshot

When you have the website deployed on the cloud, view it on a mobile phone and upload a screenshot.

Websites

Raspberry Pi Pico	LearnJCU > Reading Resources > Raspberry Pi Pico
Waveshare product site	https://www.waveshare.com/wiki/Pico-LoRa-SX1262
AWS Academy Portal	As sent to you by AWS Academy
The Things Network console	https://au1.cloud.thethings.network/console
Postman	https://web.postman.co
React	https://react.dev/
react-bootstrap	https://react-bootstrap.netlify.app/
react-apexcharts	https://apexcharts.com/docs/react-charts/
nginx	https://nginx.org/en/
Node JS	https://nodejs.org/en

IDs and Endpoints

Throughout this lab, you will need to note down values from various consoles and screens. Save them here for use later.

Item	Value
EC2 Public DNS	
Web server IP address	

Summary of steps

Connect your node to the shared data server

1. Restart LoRaWAN node
2. Add a webhook from TTN to shared AWS server
3. Inspect the data provided by the shared server

Set up Node.js development on your PC

4. Install Node.js on your PC

Develop your React dashboard

5. Make various improvements to the layout and look of the web site.
6. Limit the amount of data displayed according to the user's preferences.

Start AWS

7. Start AWS Academy Sandbox
8. Save your AWS credentials

Set up AWS EC2 instance as web server

9. Create an EC2 instance
10. Connect to EC2 instance
11. Install nginx
12. Install Node.js and create an Express web site

Deploy web site to AWS EC2 instance

13. Build distribution version
14. Use scp to upload your web site
15. Test the deployed site

Connect your node to the shared data server

Restart LoRaWAN node

TODO

Add a webhook from TTN to shared AWS server

Connect TTN to the webserver, sending a POST each time a message is received.

1. Connect to TTN at https://au1.cloud.thethings.network/console/applications
2. Select the application you built in Lab 2.
3. Select Integrations > Webhooks and delete any existing Webhooks.
4. Select Integrations > Webhooks > Add Webhook
5. Create a Custom Webhook as below:

Webhook ID	Lab4-webhook
Webhook format	JSON
Base URL	http://ec2-3-107-58-161.ap-southeast-2.compute.amazonaws.com:3000/ insert-sensor-record
Filter event data	up.uplink_message.decoded_payload
Enabled event types	Uplink message

Inspect the data provided by the shared server

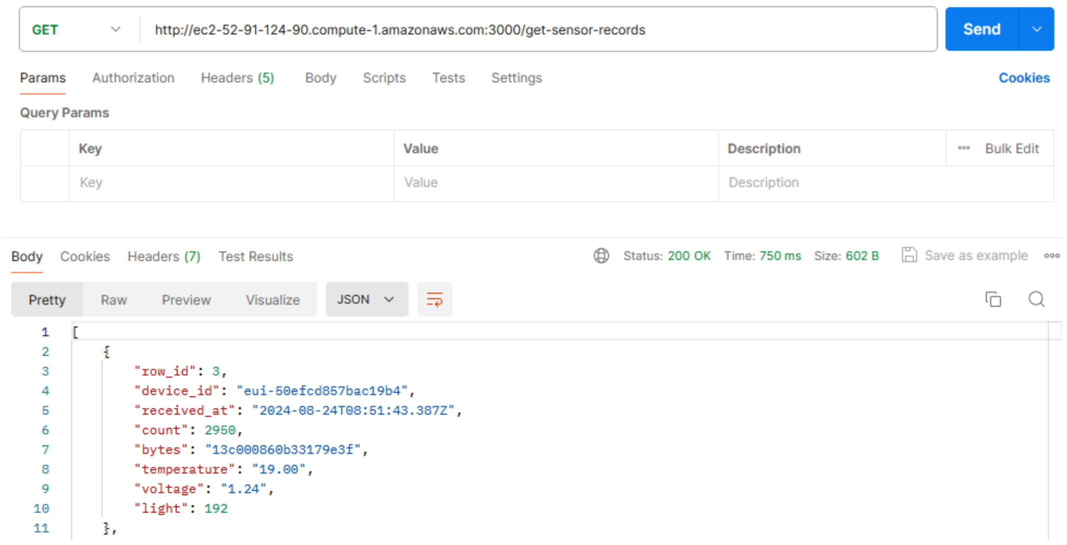
There are now 5 endpoints of interest in the API provided by the shared server.

Endpoint	Description
insert-sensor-record	Inserts a single record into the shared database. (Unchanged from Lab3.)
get-sensor-records	Retrieves all sensor records for all devices. (Unchanged from Lab3.)
get-sensor-records/:id	Retrieves all sensor records for the device whose device_id is specified.
get-sensor-records-since?t=[ts]	Retrieves all sensor records which arrived after the specified timestamp, for all devices. The value of the t parameter, [ts], is in the same format as the received_at field's value.
get-sensor-records-since/:id?t=[ts]	Retrieves all sensor records which arrived after the specified timestamp, for the specified device.

1. Open the Postman app at https://web.postman.co.
2. If you don't have a free account, create one.
3. Set up a connection as follows:

Action	GET
URL	http://ec2-3-107-58-161.ap-southeast-2.compute.amazonaws.com:3000 /get-sensor-records

4. Click ***Send*** and observe a long list of all records for all devices, e.g.:



```
http://ec2-52-91-124-90.compute-1.amazonaws.com:3000/get-sensor-records
```

```
[
    {
        "row_id": 3,
        "device_id": "eui-50efcd857bac19b4",
        "received_at": "2024-08-24T08:51:43.387Z",
        "count": 2950,
        "bytes": "13c000860b33179e3f",
        "temperature": "19.00",
        "voltage": "1.24",
        "light": 192
    },
```

5. Use the GET connection to check that data is reaching the database from your device every 30 seconds.
6. Create a new request to endpoint get-sensor-records-since and use the received_at value of a record near the end of the list just received, e.g.: http://ec2-3-107-58-161.ap-southeast-2.compute.amazonaws.com:3000/get-sensor-records-since?t=2024-08-24T14:31:18.166Z

Set up Node.js development on your PC

First, install Node.js for Windows

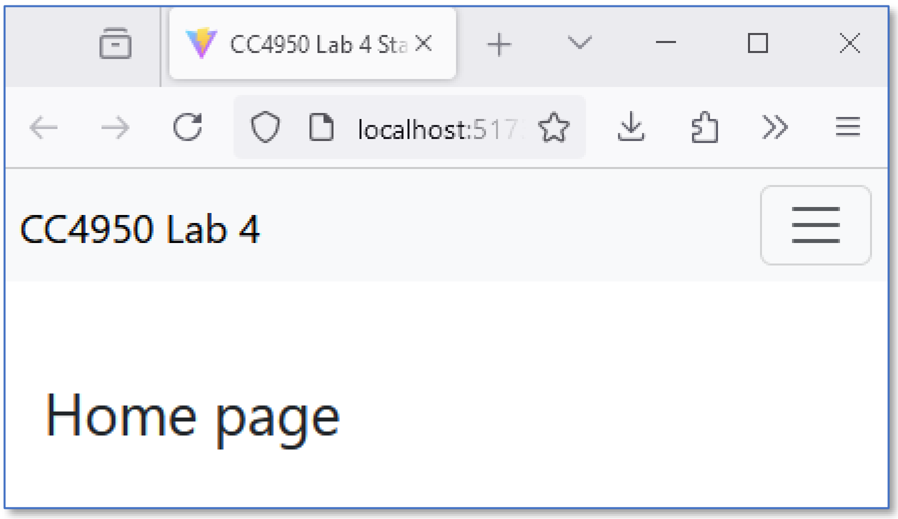
1. Go to https://nodejs.org/en and download Node.js (LTS). This will be version v20.17.0, matching the version on our web server.
2. Install it on your PC, accepting defaults.
3. Open a command prompt on your PC and type:
   ```
   node --version
   ```
   You should see:
   ```
   v20.17.0
   ```

Create the project using template code

4. Create a folder where you will be creating your source code (your *working directory*).
5. Download the skeleton application (the code below should be on a single line):
   ```
   curl https://jcuengineering.com/files/cc4950/labs/lab4/cc4950-lab4-
   react.zip > cc4950-lab4-react.zip
   ```
6. Using Windows Explorer, expand the code, thereby creating a folder named cc4950-lab4-react.
7. In your command prompt, go to the directory and prepare the code for use:
   ```
   cd cc4950-lab4-react
   npm install
   ```
8. Open the editor:
   ```
   code .
   ```

In VS Code, start the project in a local browser:

1. Open a command prompt in the Terminal (Ctrl+`). Make sure that you have Microsoft Windows command prompt session, rather than a Powershell session (where the prompt starts with **PS**.)
2. Type the following to start the development server:
   ```
   npm run dev
   ```
3. If a browser window does not open automatically, open it using the URL in the Terminal window (e.g. http://localhost:5173).
4. The application should now be running locally:



CC4950 Lab 4

Home page

Develop your React dashboard

When you're working on the local copy of the web site:

i) Make sure that your development web site is visible (this was opened automatically when you typed `npm run dev`).

ii) Rearrange you screen so that VS Code and the web page are both visible.

Open the file src/pages/HomePage.jsx and make an arbitrary change. Save the file and you will see the contents of localhost:5173 update automatically.

See how many of the changes below you can implement.

Title bar

Change the web site's title bar from **CC4950 Lab 4 Starter** to **CC4950 Lab 4 - *<Your Name>***.

Home page

Home page content

Add a more informative heading and some useful descriptive text below it. Consider putting these inside Row > Col > Card.

Grids used in Bootstrap have columns possessing a total width of 12. You make a Col stretch all the way across a Row by setting sm={12} within the Col element.

See https://react-bootstrap.netlify.app/docs/layout/grid/ for more details of the Bootstrap grid.

Home page panels

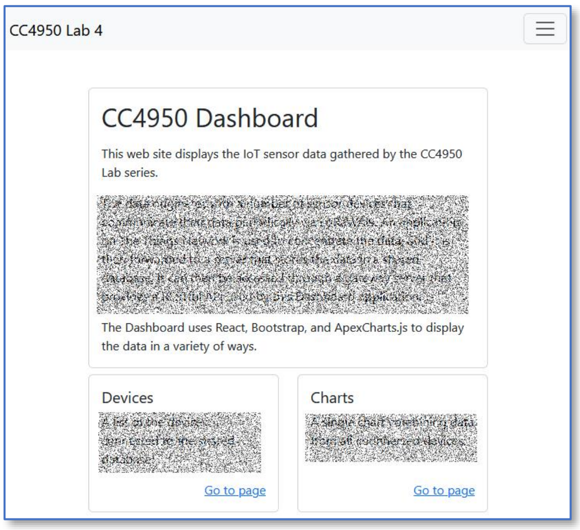
Improve the homepage (see pages/HomePage.jsx) with some panels that describe the Devices and Charts pages and contain links to them.

There is a grid with a single cell already on the page, and this contains a Card container.

Look at pages/ChartsPage.jsx for an example of using a Link element.

Consider adding `className="mt-auto"` or `"ms-auto"` -- or both – to align the Link elements between Cards.

Below is a possible suggested layout:



CC4950 Lab 4

CC4950 Dashboard

This web site displays the IoT sensor data gathered by the CC4950 Lab series.

[redacted]

The Dashboard uses React, Bootstrap, and ApexCharts.js to display the data in a variety of ways.

Devices

[redacted]

Go to page

Charts

[redacted]

Go to page

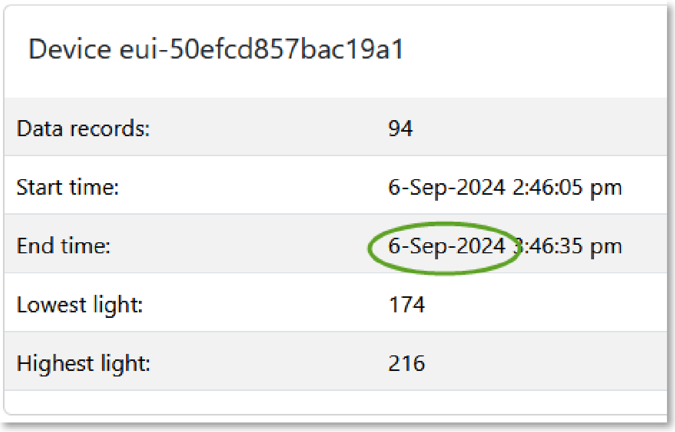
Devices Page

This page shows a list of devices whose data can be found in the shared database. Clicking on a row of the table shows the details in a second panel.

Date format

Change the date/time format in the device details panel to use 3-letter months instead of numbers.

Hint: This is not part of standard JavaScript – you will need to write a function to do this. Consider searching Stack Overflow.

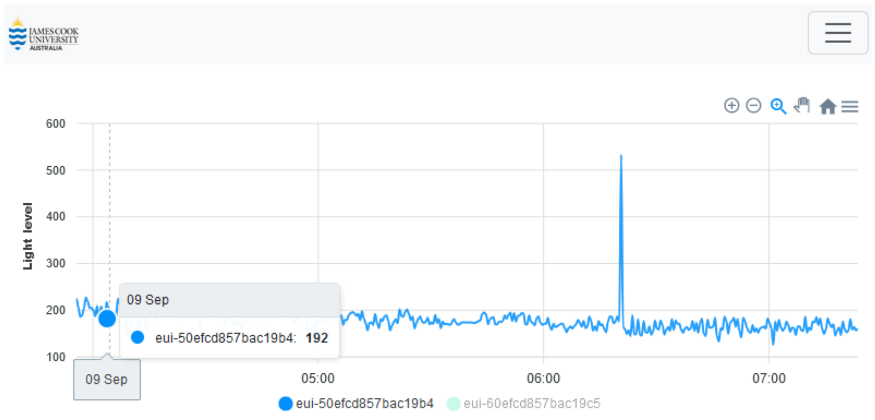


Device eui-50efcd857bac19a1

Data records:	94
Start time:	6-Sep-2024 2:46:05 pm
End time:	6-Sep-2024 3:46:35 pm
Lowest light:	174
Highest light:	216

MultiCharts Page

This page (MultiChartsPage.jsx) shows a line for every device recorded in the database.



Hidden lines

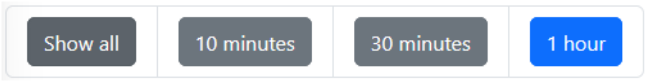
By default, lines for all but the first device are initially hidden. Change the code so that your device is shown by default and all others are initially hidden.

Latest data

Hide the panel displaying latest data.

Limits

The chart currently shows all data available for each device. Implement the limits buttons below so that only the most recent data is shown.



Charts page

This page (ChartsPage.jsx) displays all the data for the device whose link is clicked on the Device List page.

Y axes

Amend both Y axes to show their units.

Limits

Apply the same time limits logic as you applied in the MultiCharts page.

Next- Deployment

The rest of this practical shows you how to deploy your application to the cloud. Take the opportunity to view it on a mobile phone or a tablet and see whether the display is correct.

Start AWS

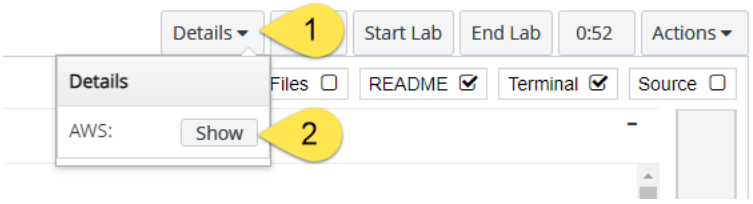
Start AWS Academy Sandbox

1. Log in to AWS Academy using the URL provided to you by Amazon.
2. Select course AWS Academy Cloud Foundations [87157]
3. Select ***Modules***
4. Scroll down and select ***Sandbox Environment***
5. Click ***Load Sandbox Environment in a new window***.
6. In the Lab page, click ***Start Lab*** at the top of the right-hand side.
7. Work elsewhere until Lab status moves from `in creation` to `ready`.

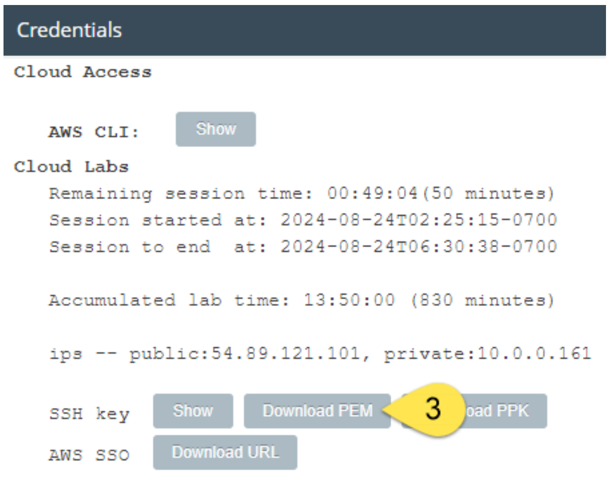
Save your AWS credentials

Once the Lab has started, save your credentials:

1. Return to the *Academy Workbench – Vocareum* window. Close the status popup.
2. Click ***Details > AWS: Show***



3. In the ***Credentials*** window, click ***Download PEM***.



```
Cloud Access

   AWS CLI:

Cloud Labs
   Remaining session time: 00:49:04(50 minutes)
   Session started at: 2024-08-24T02:25:15-0700
   Session to end  at: 2024-08-24T06:30:38-0700

   Accumulated lab time: 13:50:00 (830 minutes)

   ips -- public:54.89.121.101, private:10.0.0.161

   SSH key
   AWS SSO
```

4. Check that the file `labsuser.pem` has been downloaded to the `Downloads` folder on the PC. (Note: If it has an amended name such as `labsuser(1).pem`, delete the older version and rename the latest to `labsuser.pem`.)
5. Click ***AWS*** in the workbench.

Set up AWS EC2 instance as web server

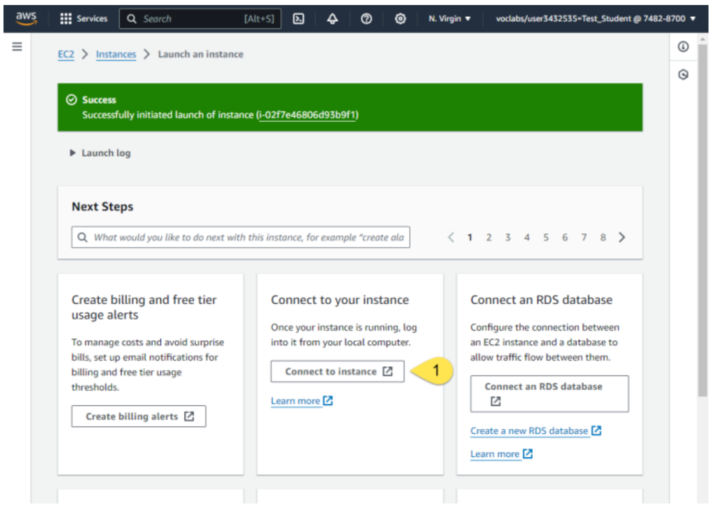
Create an EC2 instance

1. In the AWS console use ***Services*** to open ***EC2***.
2. Click ***Launch instance*** and create an instance as below:

Name	Lab4-webserver
Image	Amazon Linux 2023 AMI
Instance type	t2.micro
Key pair	vockey

Allow SSH traffic from	Anywhere
Allow HTTP traffic from the internet	Checked

3. Click ***Launch instance***.
4. On the instance dashboard, click ***Connect to instance***.



EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-02f7e46806d93b9f1)

Launch log

Next Steps

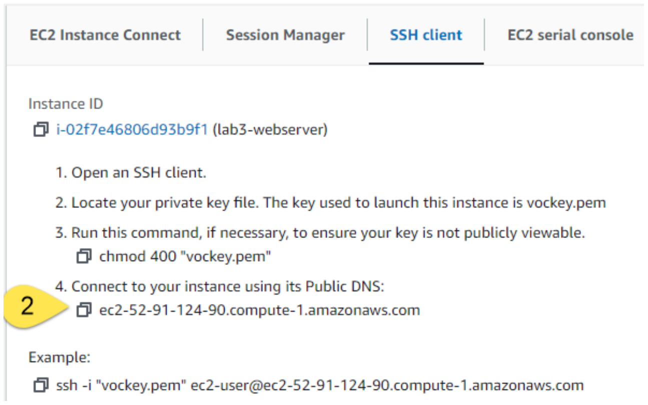
What would you like to do next with this instance, for example "create ala

Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

5. In the new tab, select SSH client.
6. Copy the EC2 Public DNS and save it as ***EC2 Public DNS***.



EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-02f7e46806d93b9f1 (lab3-webserver)

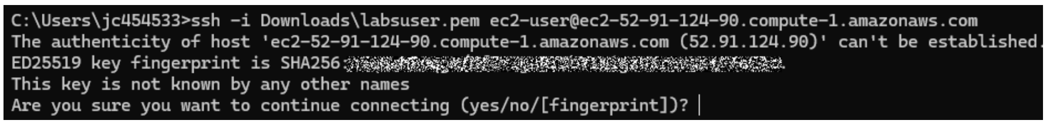
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is vockey.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "vockey.pem"
4. Connect to your instance using its Public DNS:
   ec2-52-91-124-90.compute-1.amazonaws.com

Example:
ssh -i "vockey.pem" ec2-user@ec2-52-91-124-90.compute-1.amazonaws.com

Connect to EC2 instance

Start up an SSH session from the PC to the EC2 instance.

1. On the PC, start a Windows command line session.
2. If necessary, navigate to your home folder, e.g.:
   `cd C:\Users\jc123456`
3. Start an SSH session on the EC2 instance, using the credentials downloaded earlier:
   `ssh -i Downloads\labsuser.pem ec2-user@<EC2 Public DNS>`
4. You may be asked to confirm the connection. Type Yes.



```
C:\Users\jc454533>ssh -i Downloads\labsuser.pem ec2-user@ec2-52-91-124-90.compute-1.amazonaws.com
The authenticity of host 'ec2-52-91-124-90.compute-1.amazonaws.com (52.91.124.90)' can't be established.
ED25519 key fingerprint is SHA256[redacted]
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

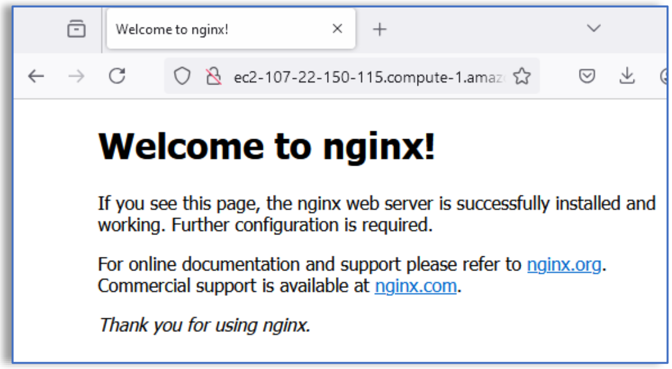
5. You are now logged on to the EC2 Linux virtual machine instance.

Install nginx

nginx is a HTTP and reverse proxy server. We use it to redirect outside HTTP requests (which use port 80) are redirected to our expressjs web server – which runs on port 3000.

In Linux, ports below 1024 (referred to as privileged ports) require root or superuser permissions to bind services to them. It is not secure or conventional to run web-services as a superuser. Thus we run the web server on port 3000 and use nginx to redirect incoming requests from port 80 to port 3000.

1. Using the SSH session connected to your EC2 instance, type the following:
   ```
   sudo yum update -y
   sudo yum install nginx -y
   sudo systemctl start nginx
   sudo systemctl enable nginx
   ```
2. Test the Nginx installation by opening it in a browser:
   `http://<EC2 public DNS>`
   You should see an Nginx web-page, containing **Welcome to nginx!**



Welcome to nginx!

ec2-107-22-150-115.compute-1.amaz

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

   Leave this browser page open.
3. IN the SSH session, open the Nginx configuration file for editing as follows:
   `sudo nano /etc/nginx/nginx.conf`
4. Locate the `server { … }` section and replace it with the code below (inserting your EC2 public DNS where indicated).
   Note: In the `server_name` line be careful to (a) remove the <> characters and (b) terminate the line with a semi-colon.

```
server {
    listen 80;
    server_name <Replace this with your EC2 public DNS>;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

5. Save your changes to the nginx configuration file.

Install Node.js and create an express web site

First, set up **Node.js**, which we will use as the runtime for the web server.

1. For this server, use nvm to install node version v20.17.0
   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh \
   ```

```
    | bash
source ~/.bashrc
nvm install v20.17.0
node -e "console.log('Running Node.js ' + process.version)"
```

2. We should see the following at the command line:
   ```
   Running Node.js v20.17.0
   ```

Install **express.js**

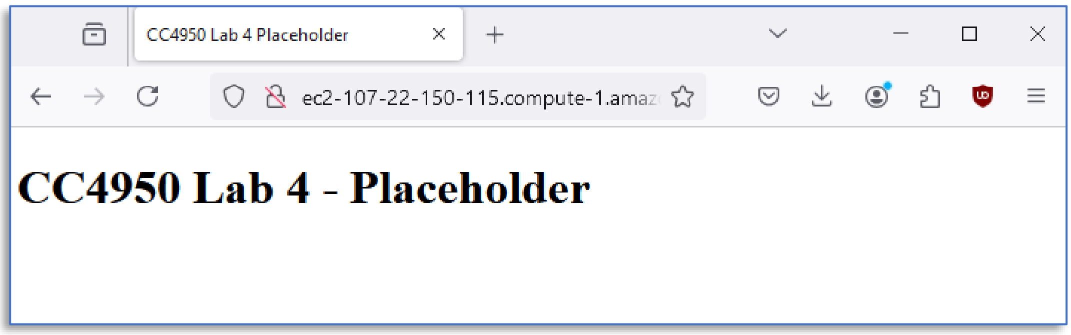
1. Create a directory for your Express server and navigate to it:
   ```
   cd ~
   mkdir express-server
   cd express-server
   ```
2. Initialize a new Node.js project:
   ```
   npm init -y
   ```
3. Install Express
   ```
   npm install express
   ```

Set Up the Express Server to serve static files

Note that if any problems arise running the express web server, full logs can be found in `$HOME/.pm2/logs`.

1. Copy the server code:
   ```
   curl https://jcuengineering.com/files/cc4950/labs/lab4/server.js \
       > server.js
   ```
2. In your express-server directory create a folder named `dist`: and download a place-holder index file.
   ```
   mkdir dist
   cd dist
   curl https://jcuengineering.com/files/cc4950/labs/lab4/placeholder.html \
       > index.html
   ```
3. Start the Express server. We use pm2 to enable automatic restart of the server process.
   ```
   cd ..
   npm install -g pm2
   pm2 start server.js
   ```
4. Check the server log as follows:
   ```
   cat $HOME/.pm2/logs/server-out.log
   ```
   You should see:
   ```
   Server running on port 3000
   ```
5. Restart Nginx to point to the Express server:
   ```
   sudo systemctl reload nginx
   ```
6. Test the express web site by returning to the text web page (`http://<EC2 public DNS>`) and refreshing the page.

7. You should now see a placeholder page, containing only: **CC4950 Lab 4 – Placeholder**.



CC4950 Lab 4 - Placeholder

All the setup work is now complete, and you now ready to start deploying your developed React IoT dashboard.

Deploy web site to AWS EC2 instance

Build distribution version

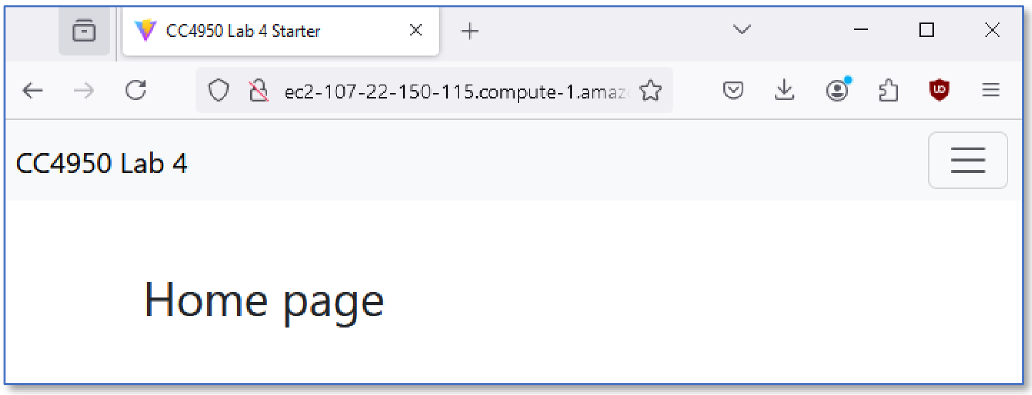
1. Open another terminal window in VS Code (the first is busy running the development server).
2. Type the following to build the distribution version of the web site:
   ```
   npm run build
   ```
3. Look in the `dist` folder and you will see the complete website, ready for deployment.

Use scp to upload your web site

1. Copy the entire `dist` folder to your web server instance, using the credentials stored in your Downloads folder, as shown:
   ```
   scp -i "%HOMEDRIVE%%HOMEPATH%\Downloads\labsuser.pem" \
     -r ./dist \
    ec2-user@<EC2 public DNS>:/home/ec2-user/express-server
   ```
2. Hint – you can save time and typing if you copy the credentials file to your `cc4950-lab4-react` folder.

Test the deployed site

1. Return to the web page that you opened for the deployed web server (`http://<EC2 public DNS>`).
2. Refresh the page to see the equivalent of your development site, e.g.:



Home page

These three steps are what you will need to do to deploy your web site to the live site each time you want to test it in the cloud. (Hint – reserve this terminal window for this purpose and you can use the command history – up-arrow on the keyboard – to re-run them easily.)