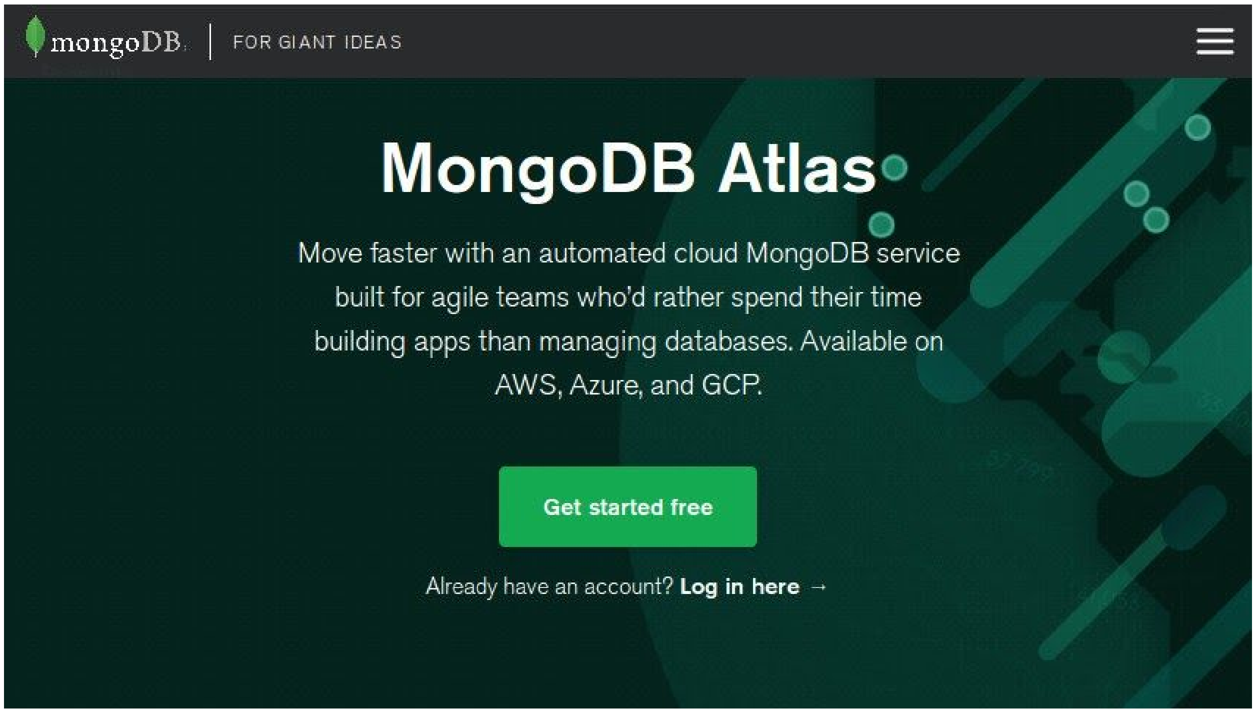
Deployment plan

1. Database deploy

Step 1: Creating a Cluster

Navigate to https://www.mongodb.com/. Then create one, go to its home page and press the **Get started free** button.

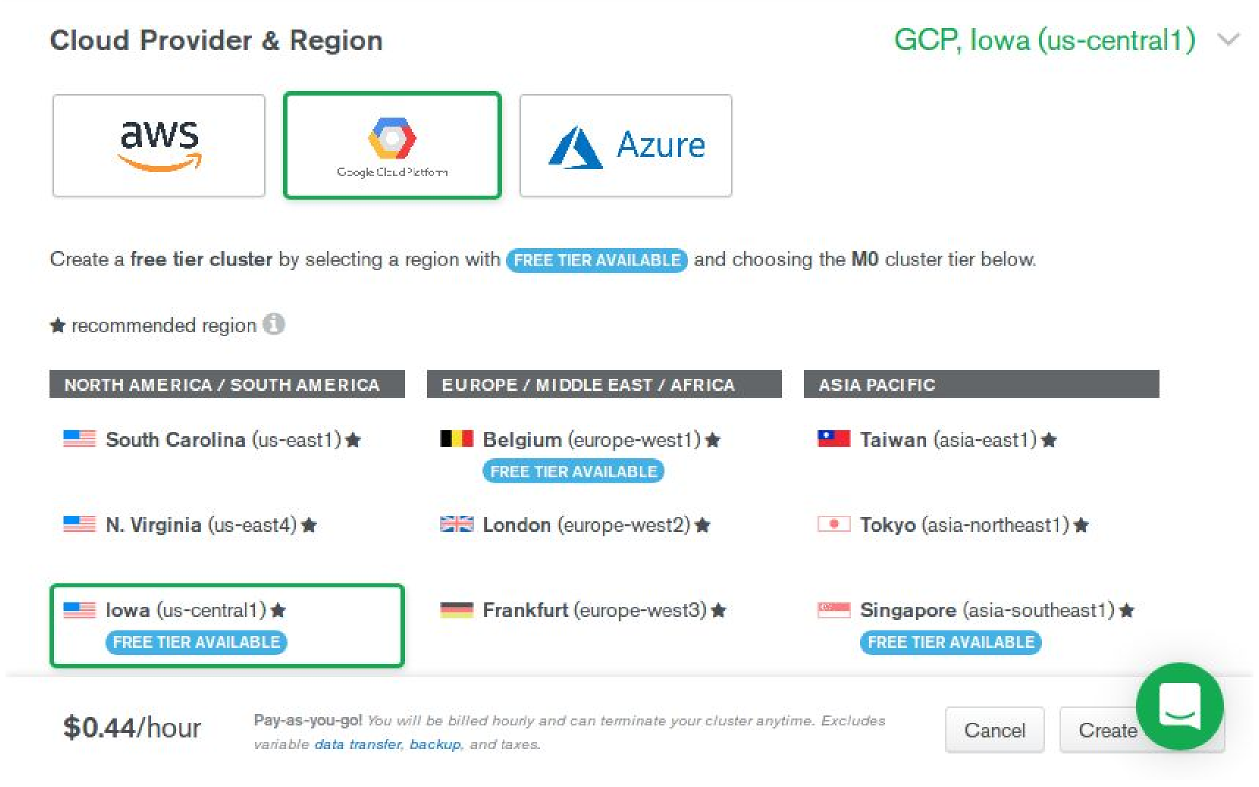


MongoDB Atlas

Move faster with an automated cloud MongoDB service built for agile teams who'd rather spend their time building apps than managing databases. Available on AWS, Azure, and GCP.

After you complete the short sign-up form, you'll be redirected to the cluster creation wizard. In its first section, you'll have to pick the cloud provider and region you prefer.

To minimize network latency, you'd ideally pick a region that's nearest to your computer. For now, however, because we are creating a free tier cluster, make sure the region you select is one that has a free tier available. Additionally, if you are using a Google Cloud VM or an Amazon EC2 instance as your development environment, do select the corresponding cloud provider first.


Cloud Provider & Region

GCP, Iowa (us-central1)

Create a **free tier cluster** by selecting a region with FREE TIER AVAILABLE and choosing the **M0** cluster tier below.

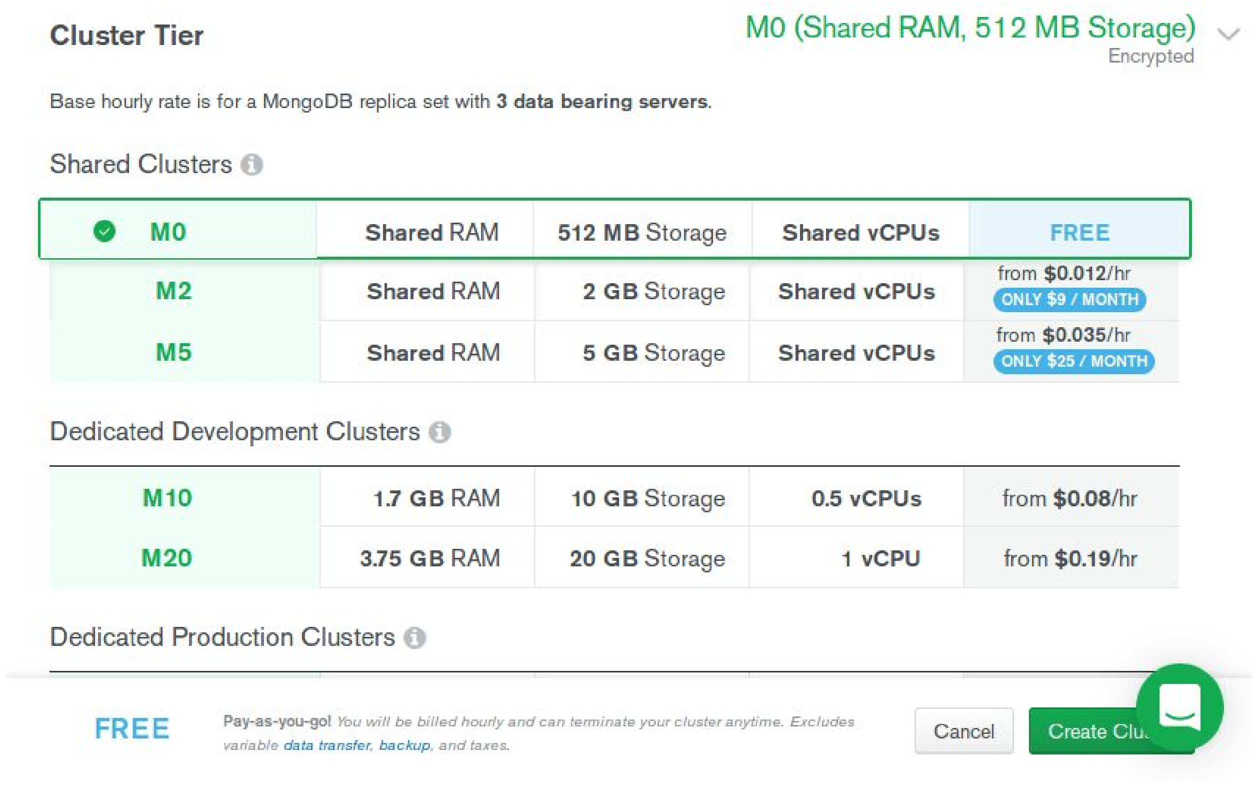
★ recommended region

NORTH AMERICA / SOUTH AMERICA	EUROPE / MIDDLE EAST / AFRICA	ASIA PACIFIC
South Carolina (us-east1) ★	**Belgium** (europe-west1) ★ FREE TIER AVAILABLE	**Taiwan** (asia-east1) ★
N. Virginia (us-east4) ★	**London** (europe-west2) ★	**Tokyo** (asia-northeast1) ★
Iowa (us-central1) ★ FREE TIER AVAILABLE	**Frankfurt** (europe-west3) ★	**Singapore** (asia-southeast1) ★ FREE TIER AVAILABLE

$0.44/hour

Pay-as-you-go! *You will be billed hourly and can terminate your cluster anytime. Excludes variable data transfer, backup, and taxes.*

In the **Cluster Tier** section, select the **M0** option to create your free tier cluster. It offers 512 MB of storage space, a recent version of MongoDB with WiredTiger as the storage engine, a replica set of three nodes, and a generous 10 GB of bandwidth per week.



Cluster Tier

M0 (Shared RAM, 512 MB Storage)
Encrypted

Base hourly rate is for a MongoDB replica set with **3 data bearing servers**.

Shared Clusters

✓ M0	**Shared** RAM	**512 MB** Storage	**Shared vCPUs**	FREE
M2	**Shared** RAM	**2 GB** Storage	**Shared vCPUs**	from **$0.012**/hr ONLY $9 / MONTH
M5	**Shared** RAM	**5 GB** Storage	**Shared vCPUs**	from **$0.035**/hr ONLY $25 / MONTH

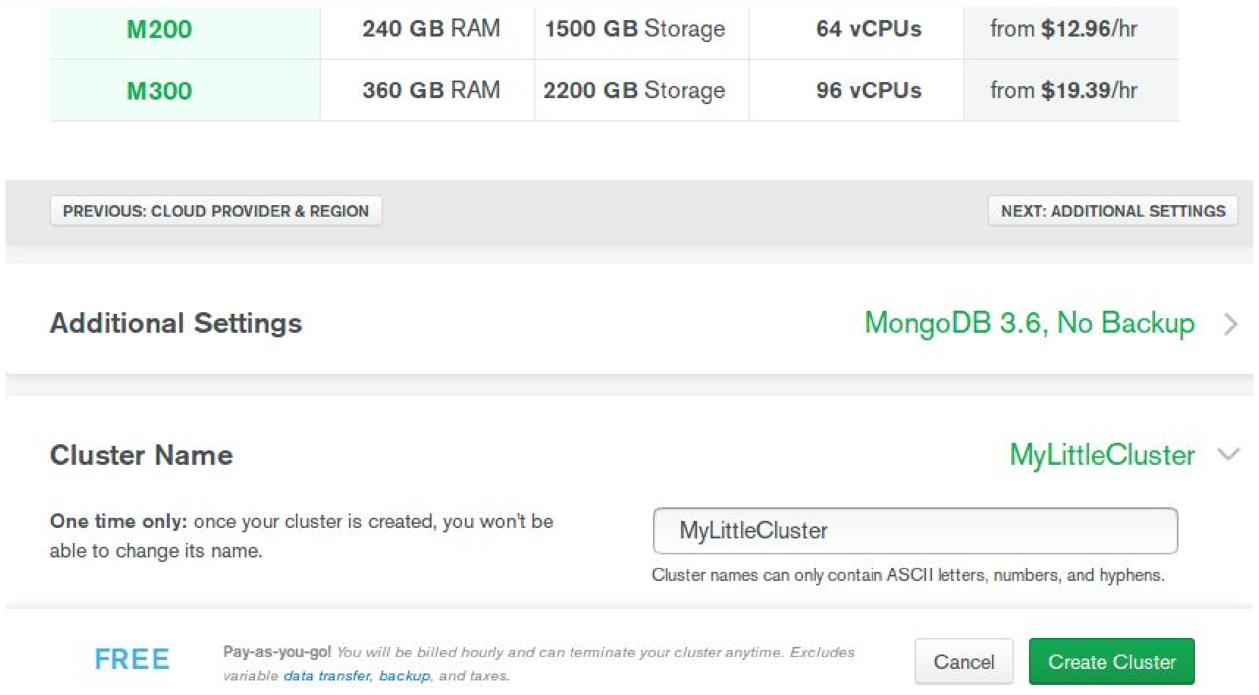
Dedicated Development Clusters

M10	**1.7 GB** RAM	**10 GB** Storage	**0.5 vCPUs**	from **$0.08**/hr
M20	**3.75 GB** RAM	**20 GB** Storage	**1 vCPU**	from **$0.19**/hr

Dedicated Production Clusters

FREE — Pay-as-you-go! *You will be billed hourly and can terminate your cluster anytime. Excludes variable data transfer, backup, and taxes.*

Lastly, give a meaningful name to the cluster and press the **Create Cluster** button.



M200	**240 GB** RAM	**1500 GB** Storage	**64 vCPUs**	from **$12.96**/hr
M300	**360 GB** RAM	**2200 GB** Storage	**96 vCPUs**	from **$19.39**/hr

Additional Settings

MongoDB 3.6, No Backup

Cluster Name

MyLittleCluster

One time only: once your cluster is created, you won't be able to change its name.

MyLittleCluster

Cluster names can only contain ASCII letters, numbers, and hyphens.

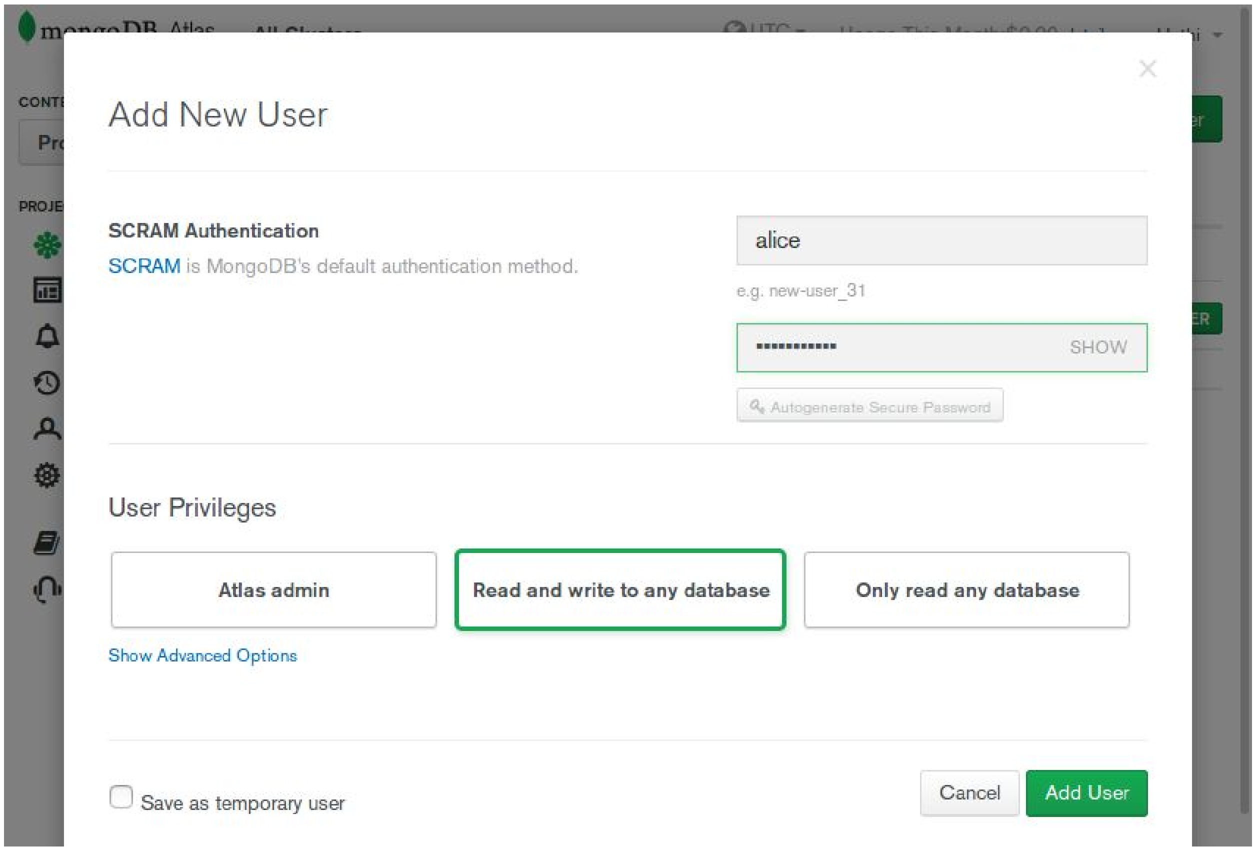
FREE — Pay-as-you-go! *You will be billed hourly and can terminate your cluster anytime. Excludes variable data transfer, backup, and taxes.*

MongoDB Atlas will now take about five minutes to set up your cluster.

Step 2 : Configuring the Cluster

Before you start using the cluster, you'll have to provide a few security-related details, so switch to the **Security** tab.

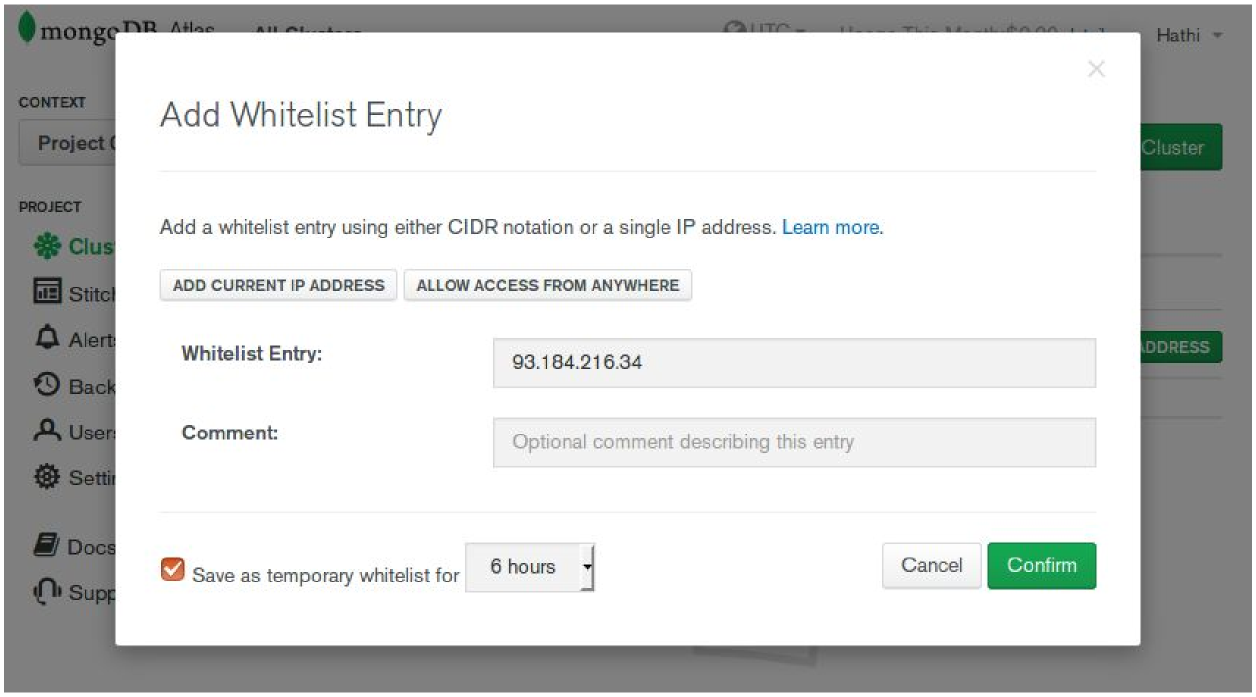
First, in the **MongoDB Users** section, you must create a new user for yourself by pressing the **Add new user** button. In the dialog that pops up, type in your desired username and password, select the **Read and write to any database** privilege, and press the **Add User** button.



Next, in the **IP Whitelist** section, you must provide a list of IP addresses from which you'll be accessing the cluster. For now, providing the current IP address of your computer is sufficient.

Press the **Add IP address** button to create a new IP address entry. In the dialog that pops up, press the **Allow access from anywhere** button to make accessible to every server.

Finally, press **Confirm** to add the entry.


Add Whitelist Entry

Add a whitelist entry using either CIDR notation or a single IP address. Learn more.

Whitelist Entry: 93.184.216.34

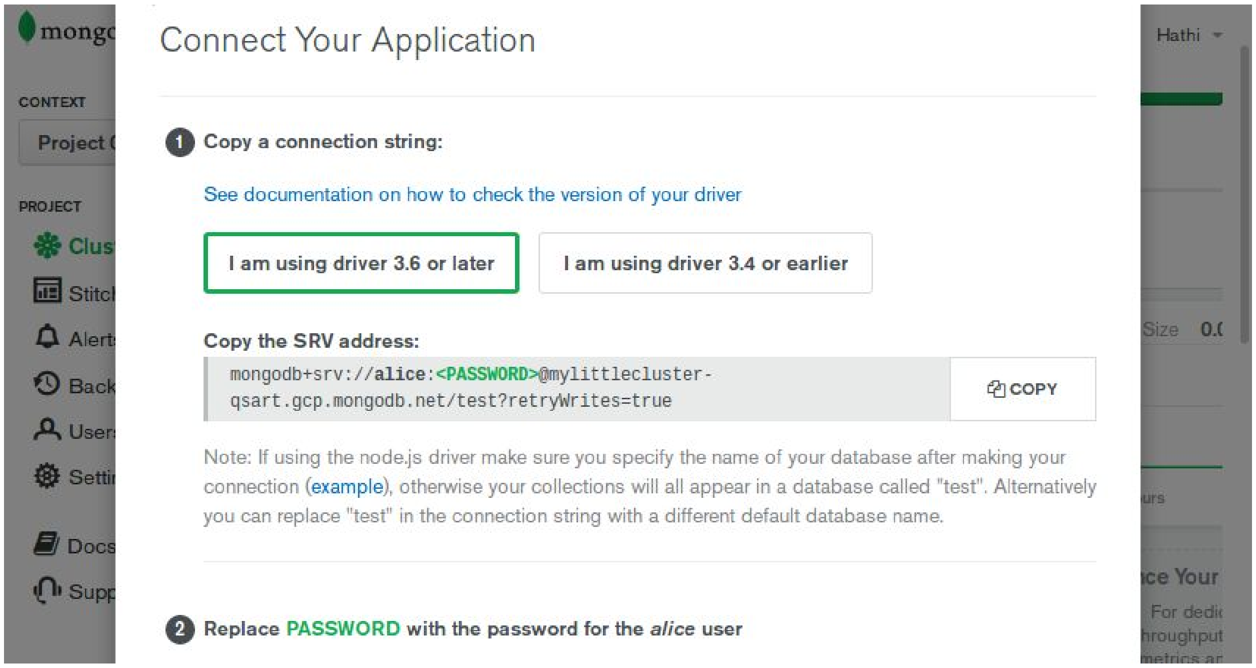
Comment: Optional comment describing this entry

Save as temporary whitelist for 6 hours

Step 3 : Getting the Connection String

You'll need a valid connection string to connect to your cluster from your application. To get it, go to the **Overview** tab and press the **Connect** button.

In the dialog that opens, select the **Connect Your Application** option and press the **I'm using driver 3.6 or later** button. You should now be able to see your connection string. It won't have your actual password, so you'll have to put it in manually. After you do so, make a note of the string so you can use it later.



Connect Your Application

1 Copy a connection string:

See documentation on how to check the version of your driver

[✓ I am using driver 3.6 or later] I am using driver 3.4 or earlier

Copy the SRV address:

```
mongodb+srv://alice:<PASSWORD>@mylittlecluster-
qsart.gcp.mongodb.net/test?retryWrites=true
```

COPY

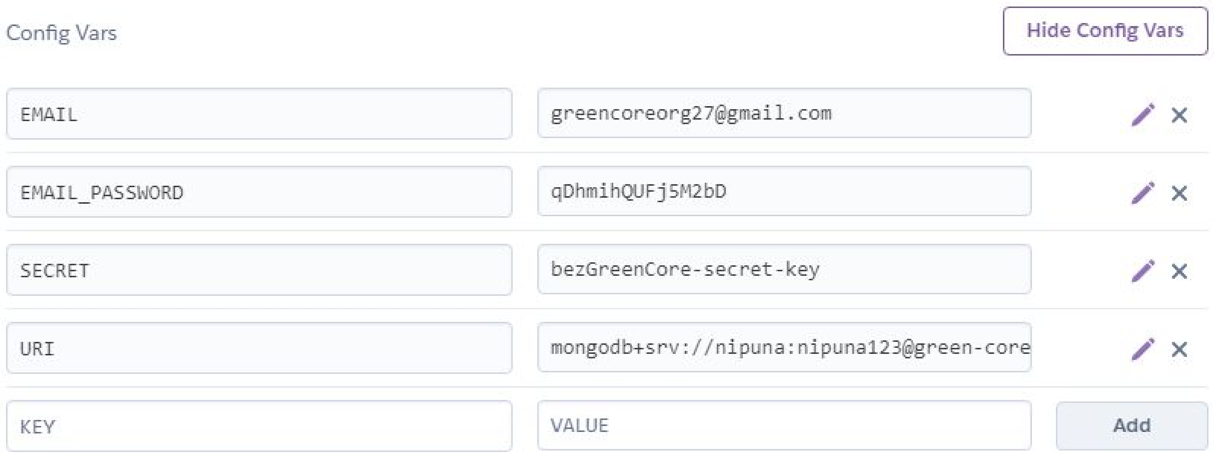
Note: If using the node.js driver make sure you specify the name of your database after making your connection (example), otherwise your collections will all appear in a database called "test". Alternatively you can replace "test" in the connection string with a different default database name.

2 Replace PASSWORD with the password for the *alice* user

Step 4 : Setting up Connection String

Get the connection string .
Go to the heroku app.

Under <app_name>/settings / config vars create a new key value pair.
Key should be “URI” and the value should be the connection string.



EMAIL	greencoreorg27@gmail.com
EMAIL_PASSWORD	qDhmihQUFj5M2bD
SECRET	bezGreenCore-secret-key
URI	mongodb+srv://nipuna:nipuna123@green-core
KEY	VALUE

2. Heroku deploy - backend (IoT, Web, Mobile)

You have to do the following steps for all the backends of the system (IoT Device, Mobile App and the Web App) you can find the github repositories in following URLs

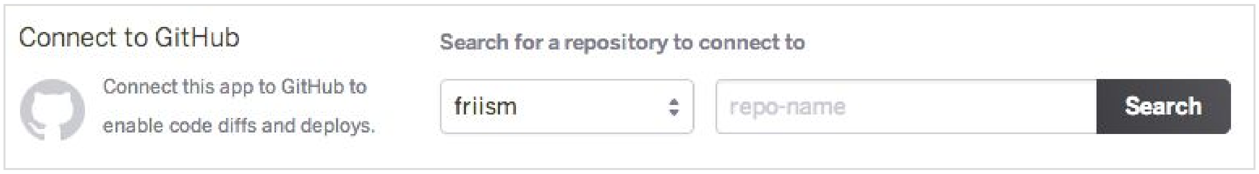
IoT Device
https://github.com/Green-core/IoT-Device

Mobile App
https://github.com/Green-core/Mobile-App-Backend

Web App
https://github.com/Green-core/Web-App-Backend

Step 1 : Enabling GitHub integration

You can configure GitHub integration in the `Deploy` tab of apps in the Heroku Dashboard.


Connect to GitHub

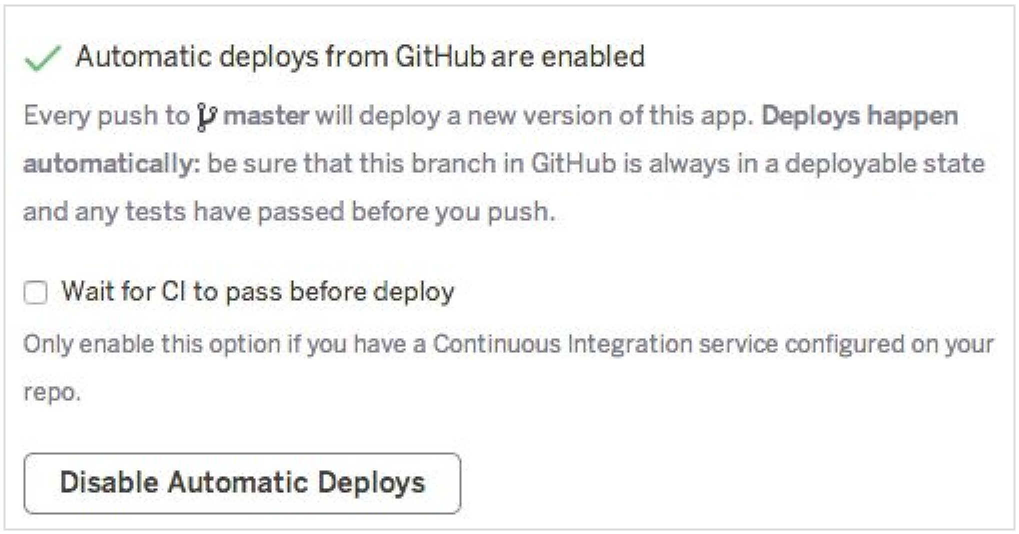
Connect this app to GitHub to enable code diffs and deploys.

To configure GitHub integration, you have to authenticate with GitHub. You only have to do this once per Heroku account.

After you link your Heroku app to a GitHub repo, you can selectively deploy from branches or configure auto-deploys.

Step 2 : Enable automatic deploys

When you enable automatic deploys for a GitHub branch, Heroku builds and deploys all pushes to that branch. If, for example, you have a development app on Heroku, you can configure pushes to your GitHub `development` branch to be automatically built and deployed to that app.


Automatic deploys from GitHub are enabled

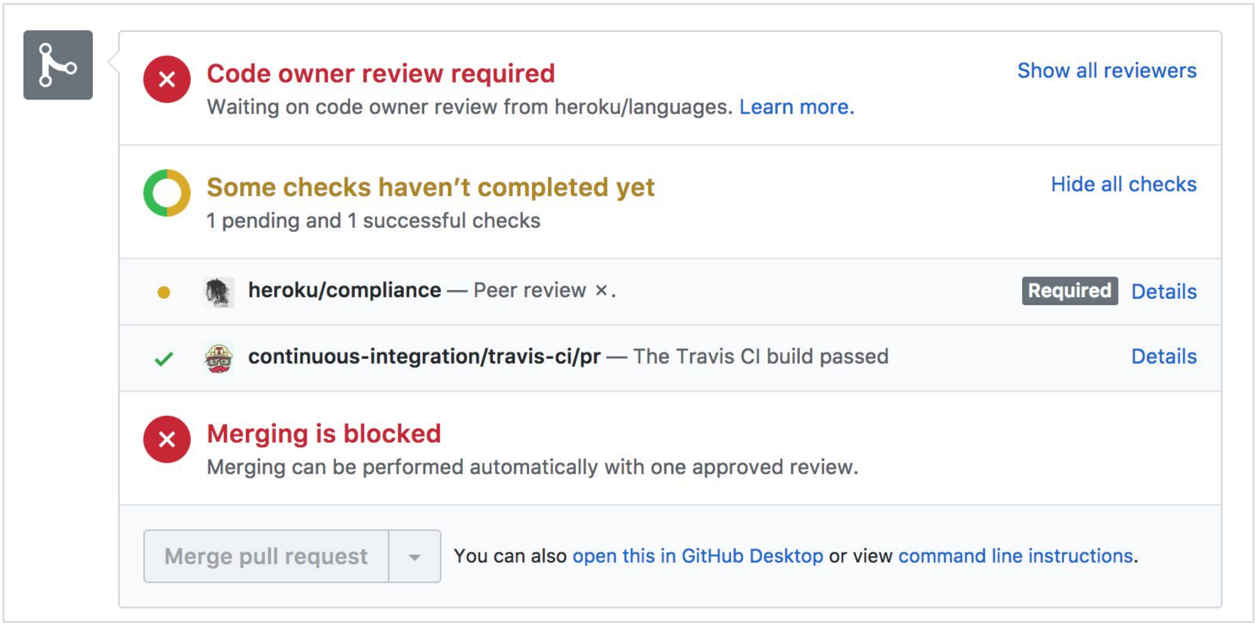
Every push to **master** will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push.

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

If you’ve configured your GitHub repo to use automated Continuous Integration (with Travis CI, for example), you can check the “Wait for CI to pass before deploy" checkbox. When enabled, Heroku will only auto-deploy after all the commit statuses of the relevant commit show `success`.

This commit won't auto-deploy because one of the checks shows a `pending` status:



Code owner review required
Waiting on code owner review from heroku/languages. Learn more.
Show all reviewers

Some checks haven't completed yet
1 pending and 1 successful checks
Hide all checks

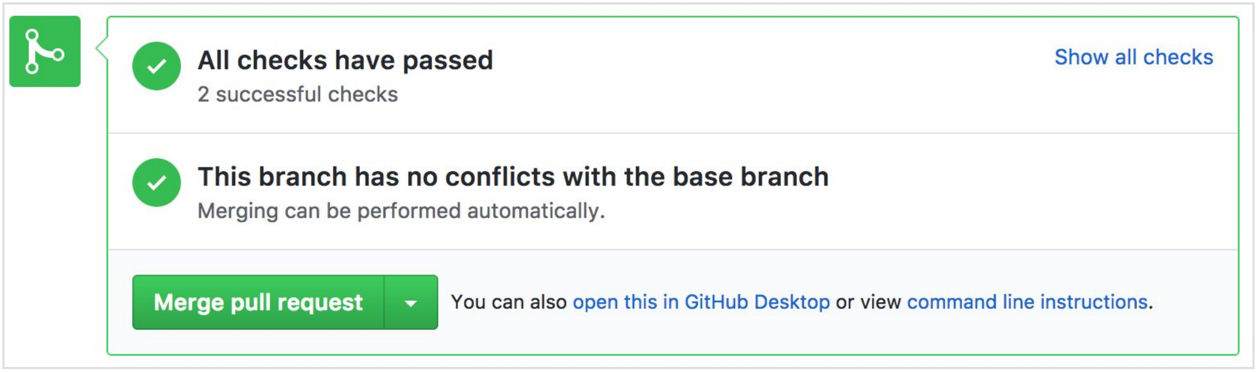
● heroku/compliance — Peer review ×. Required Details

✓ continuous-integration/travis-ci/pr — The Travis CI build passed Details

Merging is blocked
Merging can be performed automatically with one approved review.

You can also open this in GitHub Desktop or view command line instructions.

This commit will auto-deploy because all of the checks show a status of `success`:



All checks have passed
2 successful checks
Show all checks

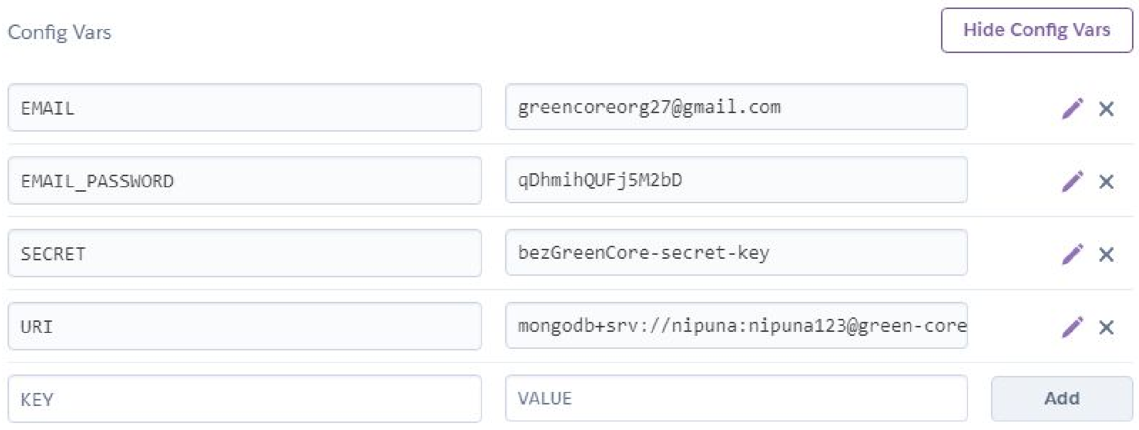
This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Step 3 : Setting up Connection Strings

Get the connection string .
Go to the heroku app.

Under <app_name>/settings / config vars create a new key value pair.
Key should be "URI" and the value should be the connection string.

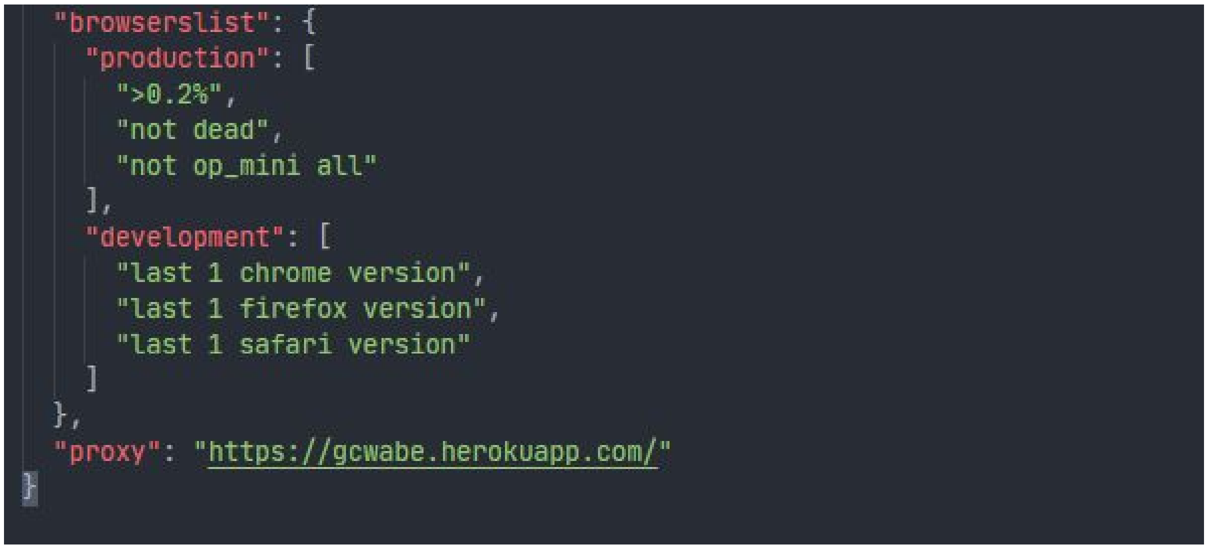


EMAIL	greencoreorg27@gmail.com
EMAIL_PASSWORD	qDhmihQUFj5M2bD
SECRET	bezGreenCore-secret-key
URI	mongodb+srv://nipuna:nipuna123@green-core
KEY	VALUE

3. Heroku deploy - frontend

Download the source code from the GitHub repository and then change the proxy of the package.json file with the URL of the deployed backend

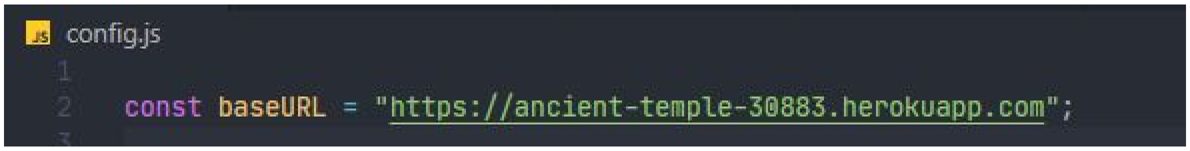
Then continue the steps 1 and 2 as in backend deployment.


```
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "proxy": "https://gcwabe.herokuapp.com/"
}
```

4. Mobile app

Step 1. Set server url

Open config.js and set the server url as BaseURL



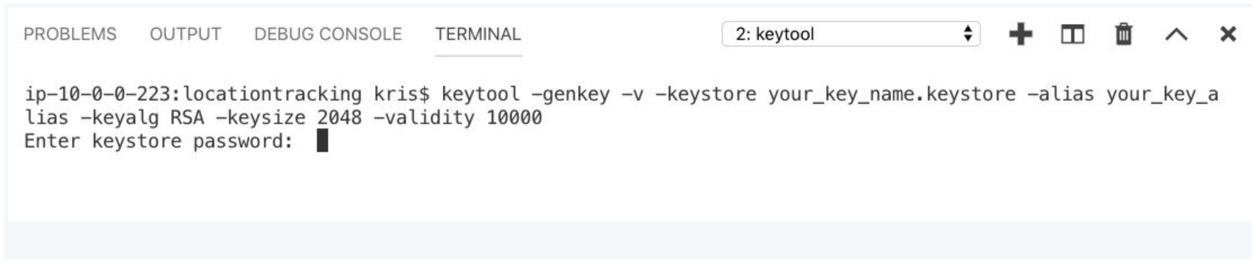
```
const baseURL = "https://ancient-temple-30883.herokuapp.com";
```

Step 2. Generate a keystore

You will need a Java generated signing key which is a keystore file used to generate a React Native executable binary for Android. You can create one using the keytool in the terminal with the following command

```
keytool -genkey -v -keystore your_key_name.keystore -alias your_key_alias
-keyalg RSA -keysize 2048 -validity 10000
```

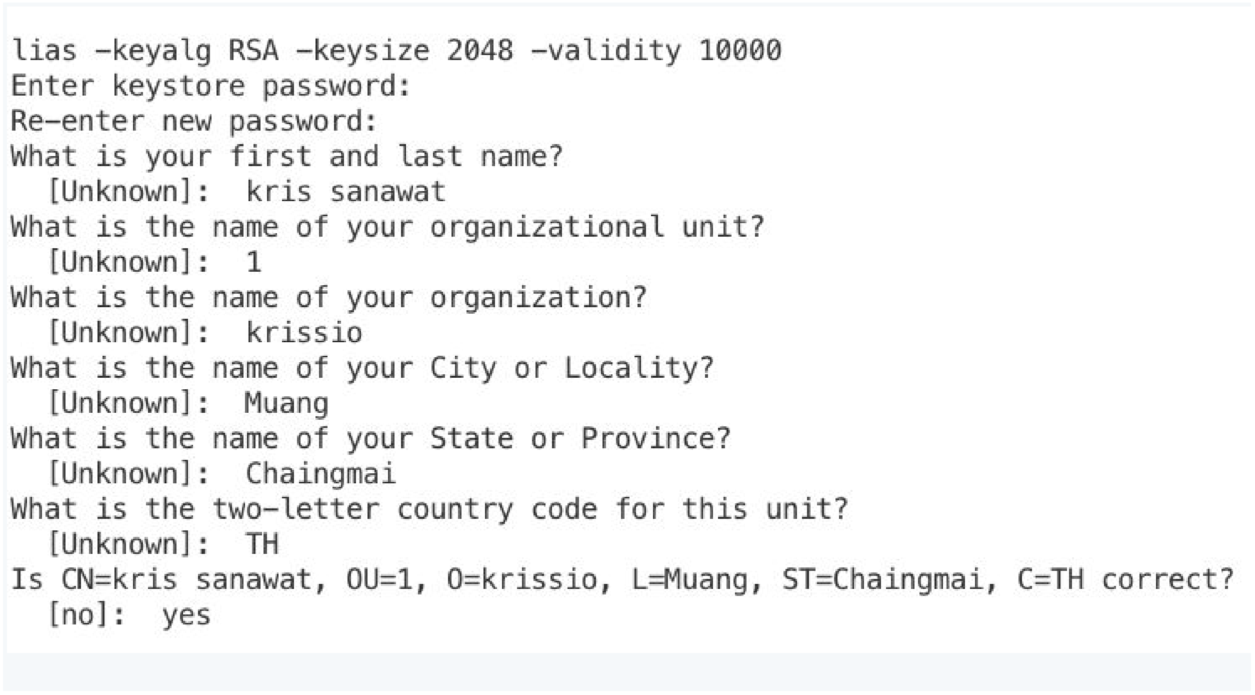
Once you run the keytool utility, you'll be prompted to type in a password. Make sure you remember the password:



```
ip-10-0-0-223:locationtracking kris$ keytool -genkey -v -keystore your_key_name.keystore -alias your_key_a
lias -keyalg RSA -keysize 2048 -validity 10000
Enter keystore password:
```

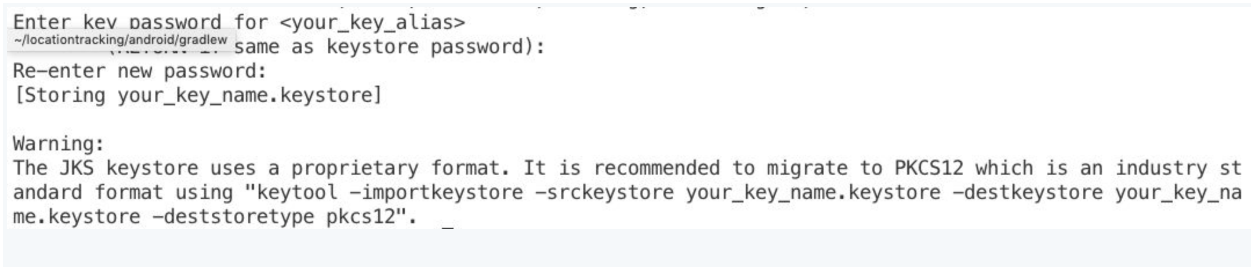
You can change **your_key_name** with any name you want, as well as **your_key_alias**. This key uses key-size 2048, instead of default 1024 for security reason.
Thus, this command prompts you for the password of the keystore, the actual key, and the distinguished name fields for your key. Hence, everything should be entered manually and carefully.

Enter your keystore password: password123
Re-enter new password: password123
What is your first and last name? [unknown]: Dani Williams
What is the name of your organizational unit? [unknown]: Sample Company
What is the name of your organization? [unknown]: Sample
What is the name of your city or Locality? [unknown]: XYZ
What is the name of your State or Province? [unknown]: ABC
What is the two-letter country code for this unit? [unknown]: XX
Your terminal output will look similar to this:



```
lias -keyalg RSA -keysize 2048 -validity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  kris sanawat
What is the name of your organizational unit?
  [Unknown]:  1
What is the name of your organization?
  [Unknown]:  krissio
What is the name of your City or Locality?
  [Unknown]:  Muang
What is the name of your State or Province?
  [Unknown]:  Chaingmai
What is the two-letter country code for this unit?
  [Unknown]:  TH
Is CN=kris sanawat, OU=1, O=krissio, L=Muang, ST=Chaingmai, C=TH correct?
  [no]:  yes
```

Press Enter when you're prompted to enter the password for <my-key-alias>.
Note: If you need to have a new key password, then type it in.



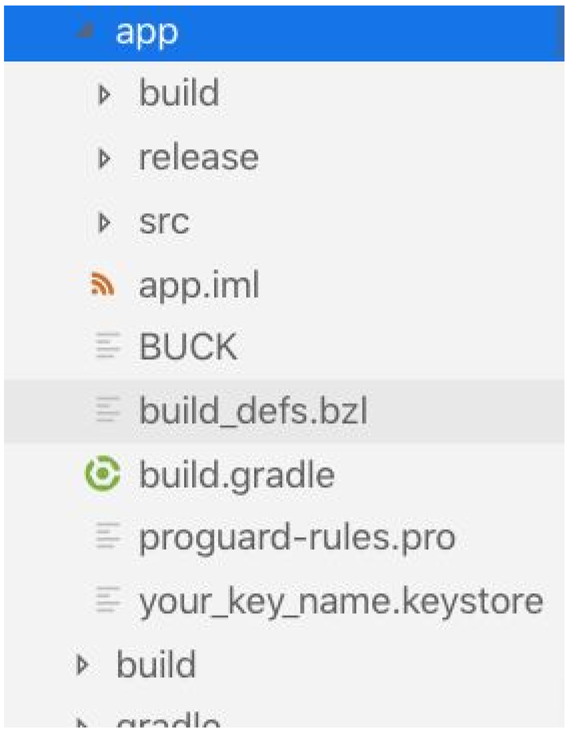
```
Enter key password for <your_key_alias>
~/locationtracking/android/gradlew
        (RETURN if same as keystore password):
Re-enter new password:
[Storing your_key_name.keystore]

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry st
andard format using "keytool -importkeystore -srckeystore your_key_name.keystore -destkeystore your_key_na
me.keystore -deststoretype pkcs12".  _
```

As a result, it generates a key-store file on your project directory named my-release-key.keystore valid for 10000 days. Most importantly, **back up this keystore file and its credentials** (store password, alias, and alias password) which will be required later.

Step 3. Adding Keystore to your project

Firstly, you need to copy the file your_key_name.keystore and paste it under the **android/app** directory in your React Native project folder.
On Terminal,

```
mv my-release-key.keystore /android/app
```



You need to open your android\app\build.gradle file and add the keystore configuration. There are two ways of configuring the project with keystore. First, the common and unsecured way:

```
android {
```

```
....
signingConfigs {
  release {
    storeFile file('your_key_name.keystore')
    storePassword 'your_key_store_password'
    keyAlias 'your_key_alias'
    keyPassword 'your_key_file_alias_password'
  }
 }

 buildTypes {
   release {
     ....
     signingConfig signingConfigs.release

   }
 }
}
```

This is not a good security practice since you store the password in plain text. Instead of storing your keystore password in .gradle file, you can stipulate the build process to prompt you for these passwords if you are building from the command line.

To prompt for password with the Gradle build file, change the above config as

```
signingConfigs {
 release {
   storeFile file('your_key_name.keystore')
   storePassword System.console().readLine("\nKeystore password:")
   keyAlias System.console().readLine("\nAlias: ")
   keyPassword System.console().readLine("\Alias password: ")
  }
}
```

Therefore, you should make sure the signingConfigs block appears before buildTypes block to avoid unnecessary errors. Moreover, before going any further, make sure you have an assets folder under android/app/src/main/assets. If it's not there, create one. Then run the following command to build the bundle.

```
react-native bundle --platform android --dev false --entry-file index.js
--bundle-output android/app/src/main/assets/index.android.bundle
--assets-dest android/app/src/main/res/
```

Note: If you have a different entry file name, like **index.android.js**, change it within the command.

Step 4. Release APK Generation

Place your terminal directory to android using:

```
cd android
```

Then run the following command

For Windows,

```
gradlew assembleRelease
```

For Linux and Mac OS X:

```
./gradlew assembleRelease
```

As a result, the APK creation process is done. You can find the generated APK at android/app/build/outputs/apk/app-release.apk. This is the actual app, which you can send to your phone or upload to the Google Play Store. Congratulations, you've just generated a React Native Release Build APK for Android.

There are frequent errors that show up in this process sometimes, which is typical to a React Native app, given React Native is continuously evolving. We are laying out here the most frequent React Native build errors that we ran into, to save you time and headaches.

If your build fails with the following error:

```
:app:processReleaseResources FAILED
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':app:processReleaseResources'.
> com.android.ide.common.process.ProcessException: Failed to execute aapt
```

You can always solve this React Native Android build error by running:

```
cd ..

rm -rf android/app/src/main/res/drawable-*

react-native bundle --platform android --dev false \
 --entry-file index.js \

 --bundle-output android/app/src/main/assets/index.android.bundle \

   --assets-dest android/app/build/intermediates/res/merged/release/

  cd android && ./gradlew assembleRelease
```

These are steps to generate a Release Signed APK in React Native for Android applications, that can be published to the Google Play Console.

5. IoT Device

Step 0 - Installing Arduino IDE
You can download the Arduino IDE from the below URL according to the Operating System of your computer

https://www.arduino.cc/en/Main/Software

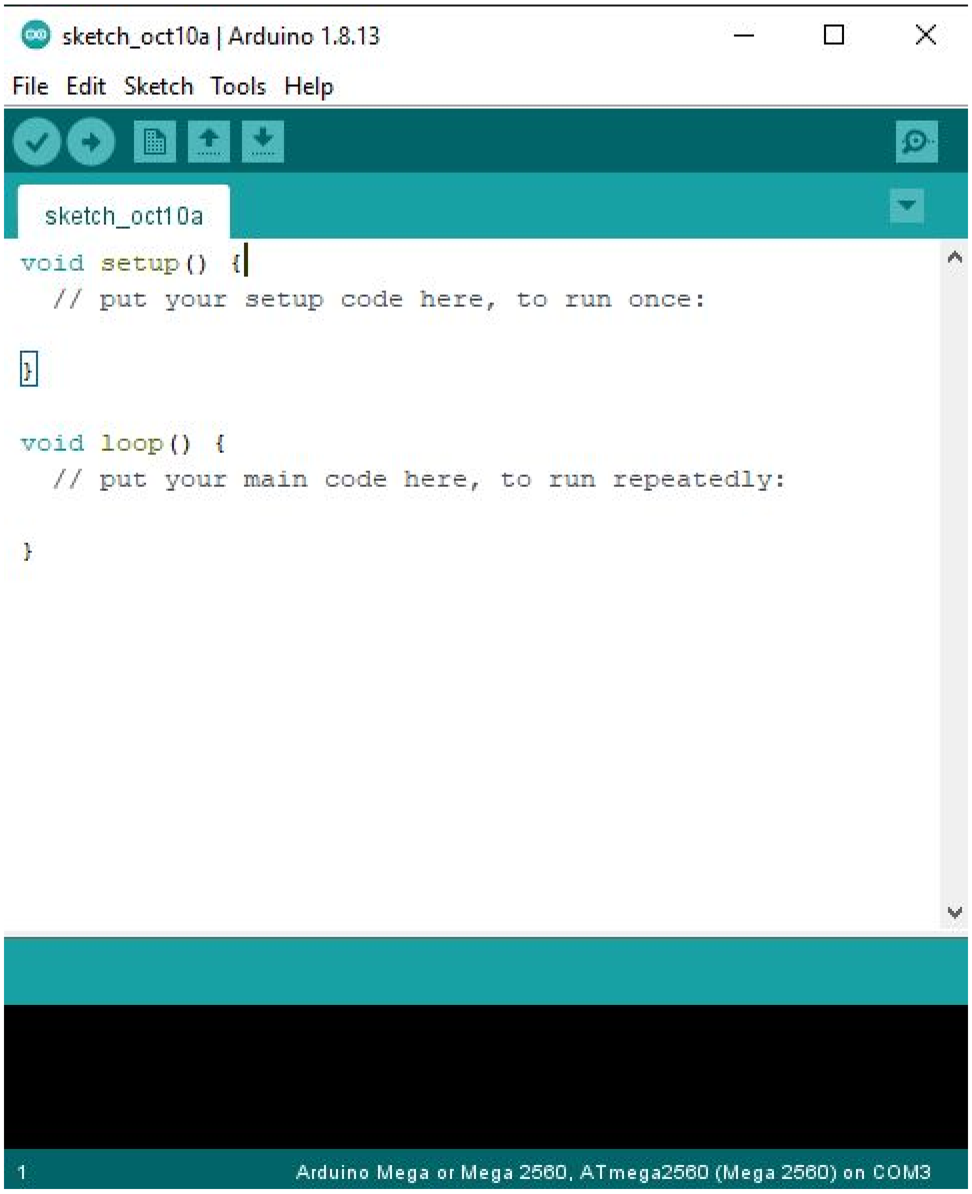
After receiving the assembled IoT Device you have to carry out the following steps to upload the code to the NodeMCU and the Arduino Mega.

Note: Please make sure that both the two switches used to connect the NodeMCU and the Arduino Mega are disconnected (switched off) or else you'll see an error.

1. Setting up the NodeMCU

Step 1 - Installing ESP8266 library to Arduino IDE (Used for the NodeMCU)

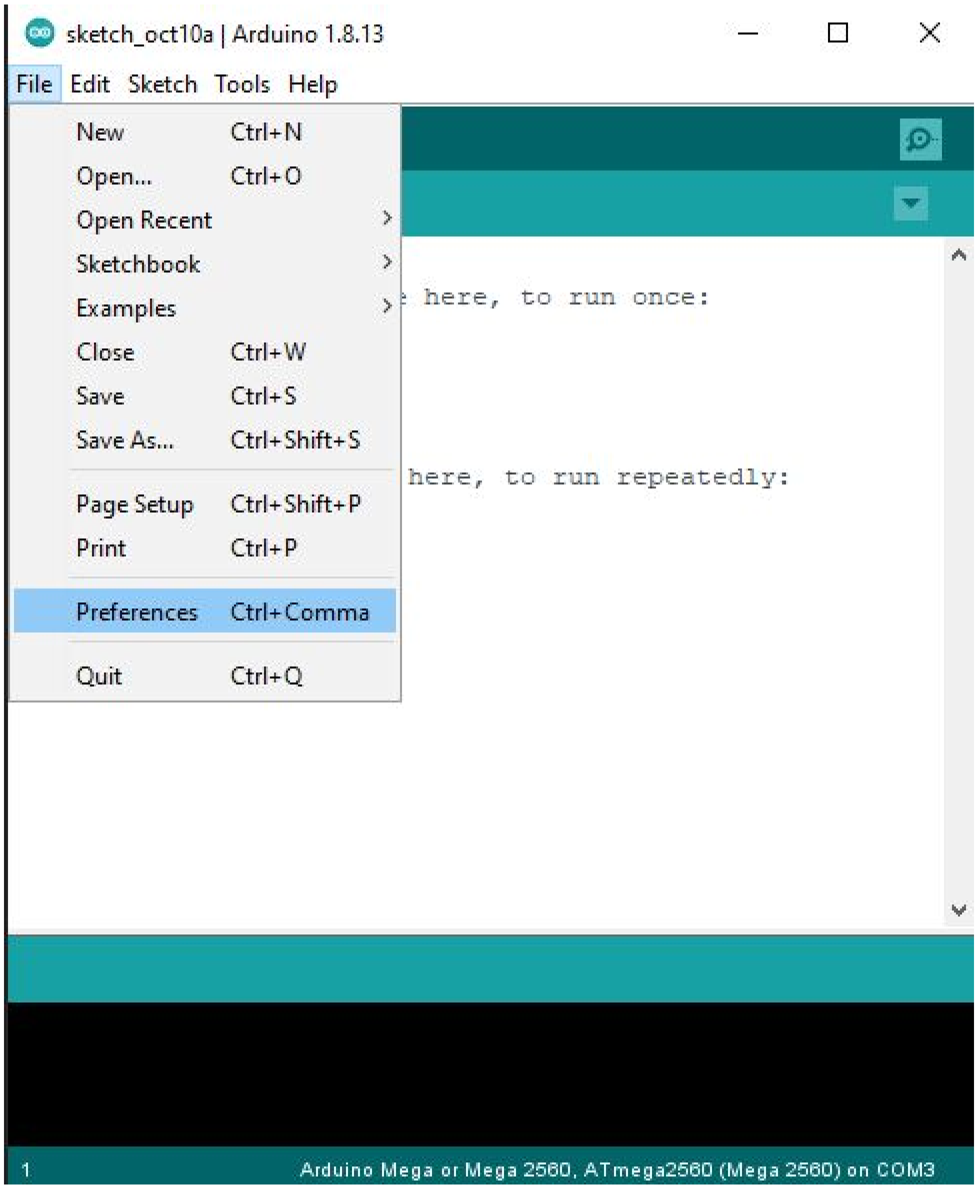
After starting the app you installed in Step 0, you'll see the following screen



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
1        Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3
```

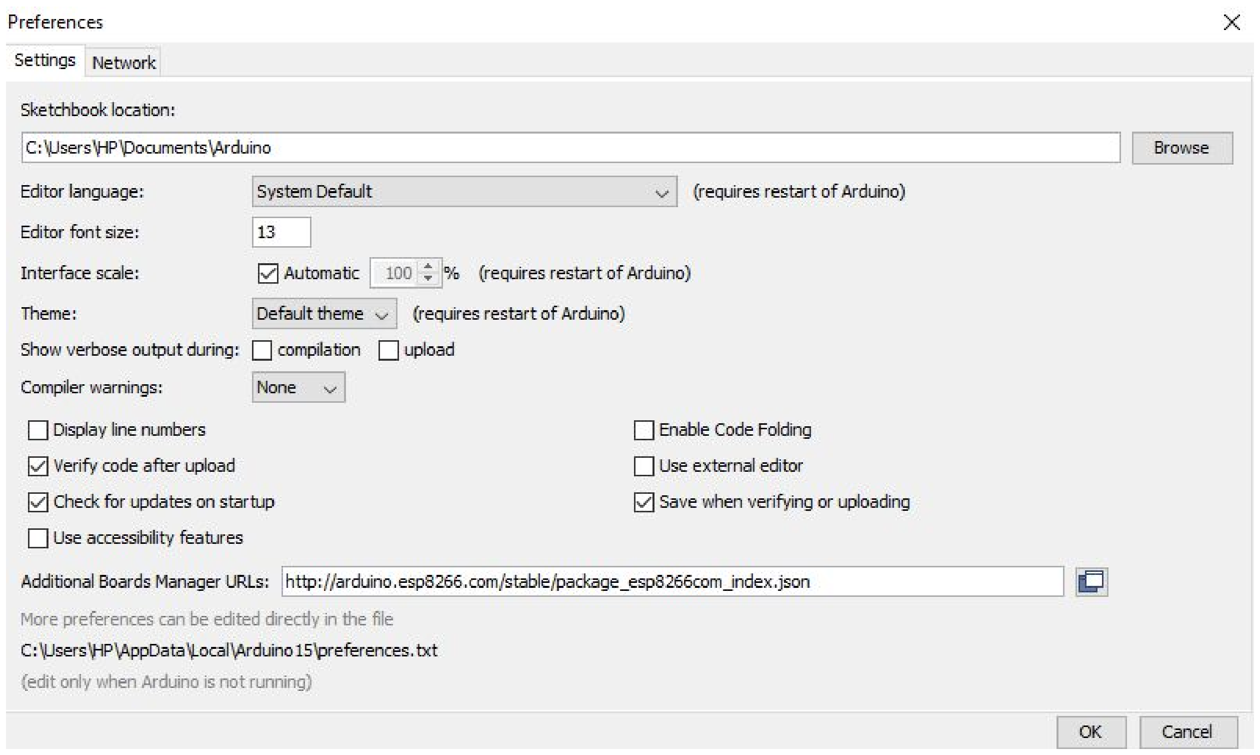
Then Click on the File tab in the top left corner and then select preferences tab in it



```
here, to run once:

here, to run repeatedly:
```

Then enter the following URL into the “Additional Boards Manager URLs” field as shown in the below image

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Preferences

Settings | Network

Sketchbook location: C:\Users\HP\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 13

Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers
☑ Verify code after upload
☑ Check for updates on startup
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
☑ Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

C:\Users\HP\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

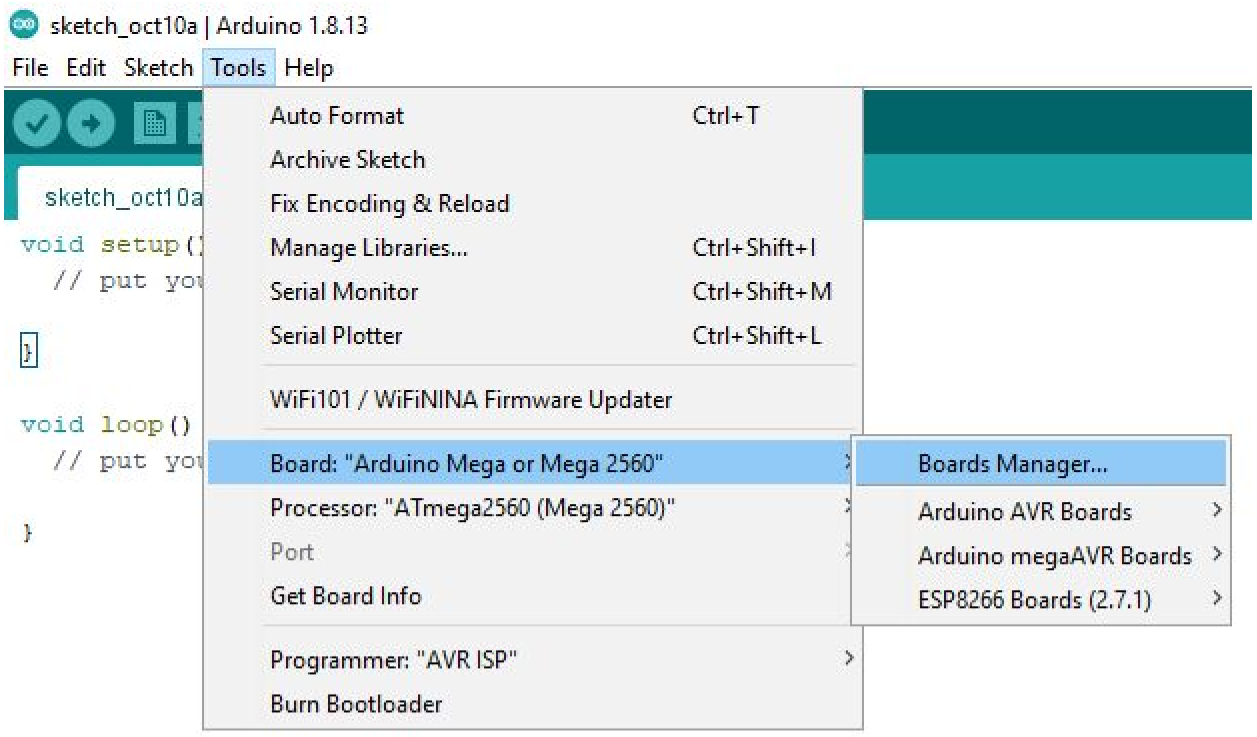
Note: If you already have the ESP32 boards URL, you can separate the URLs with a comma as follows.

```
https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_inde
x.json
```

Then click on OK.

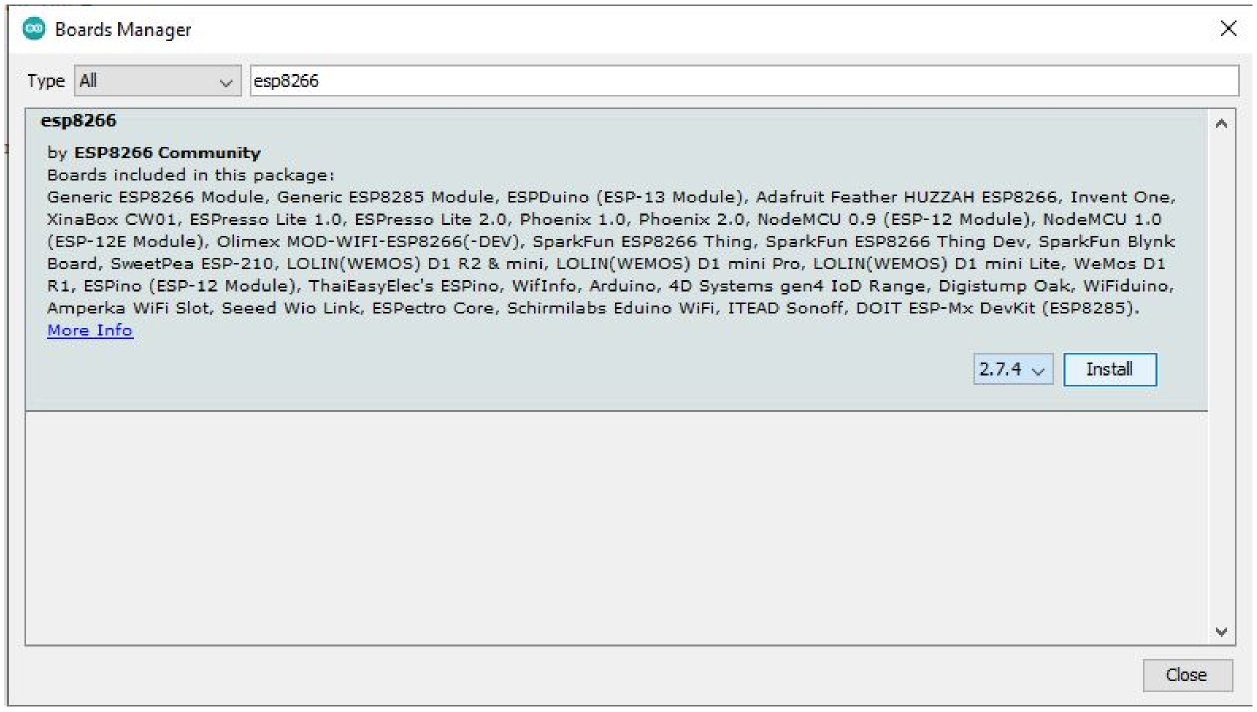
Step 2 - Installing the board

To open the boards manager go to **Tools -> Board -> Boards Manager** as shown in the image below


sketch_oct10a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_oct10a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port
Get Board Info
Programmer: "AVR ISP"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino megaAVR Boards
ESP8266 Boards (2.7.1)

```
void setup()
  // put you
}

void loop()
  // put you

}
```

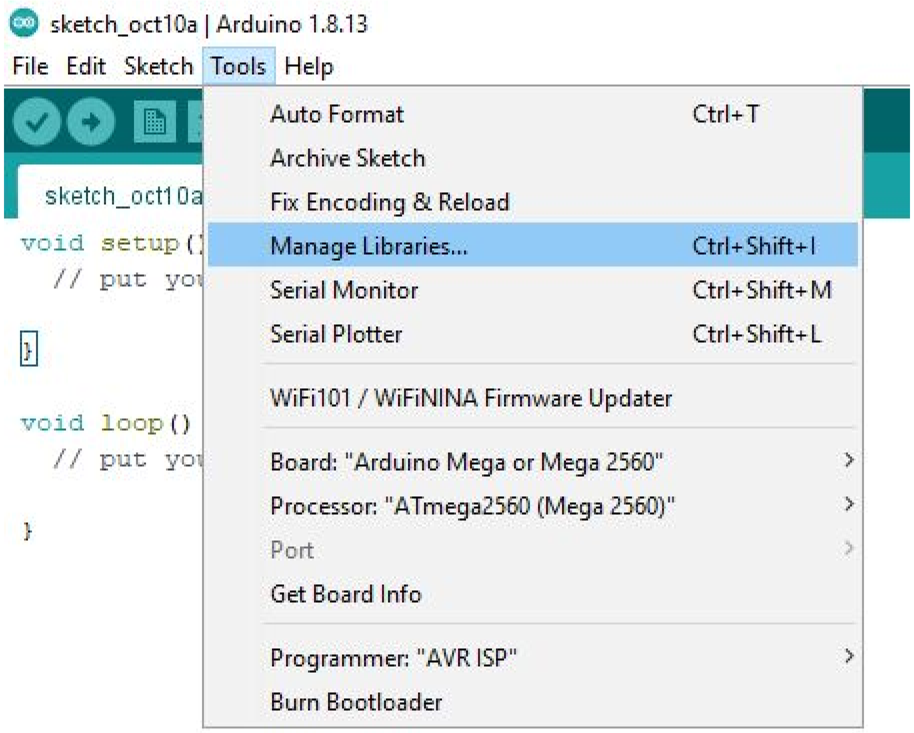
In the Boards Manager search for ESP8266 and install the version 2.7.4



esp8266
by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).

Step 3 - Installing Libraries

Go to **Tools -> Manage Libraries**


sketch_oct10a | Arduino 1.8.13
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port
Get Board Info
Programmer: "AVR ISP"
Burn Bootloader

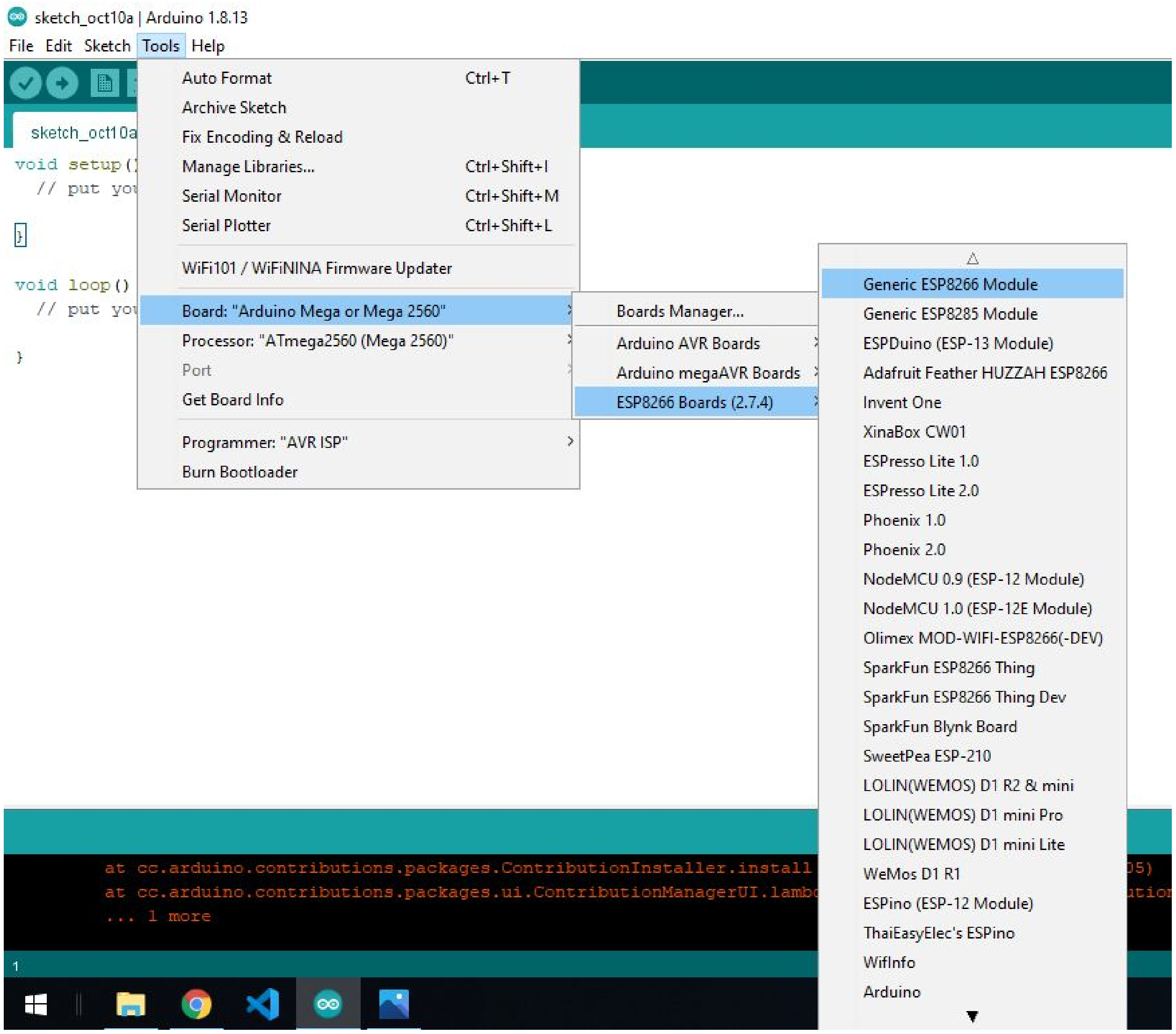
```
void setup(){
  // put you
}

void loop()
  // put you

}
```

Then search for the following libraries and install them

- ArduinoJson - version 6.15.2
- ESP8266RestClient - version 1.0.0

Step 4 - Selecting the board

After installing the libraries mentioned in step 4 to select the board go to **Tools -> Boards -> ESP8266 Boards(2.74)** and select the **Generic ESP8266 Module** (Most probably the topmost one)

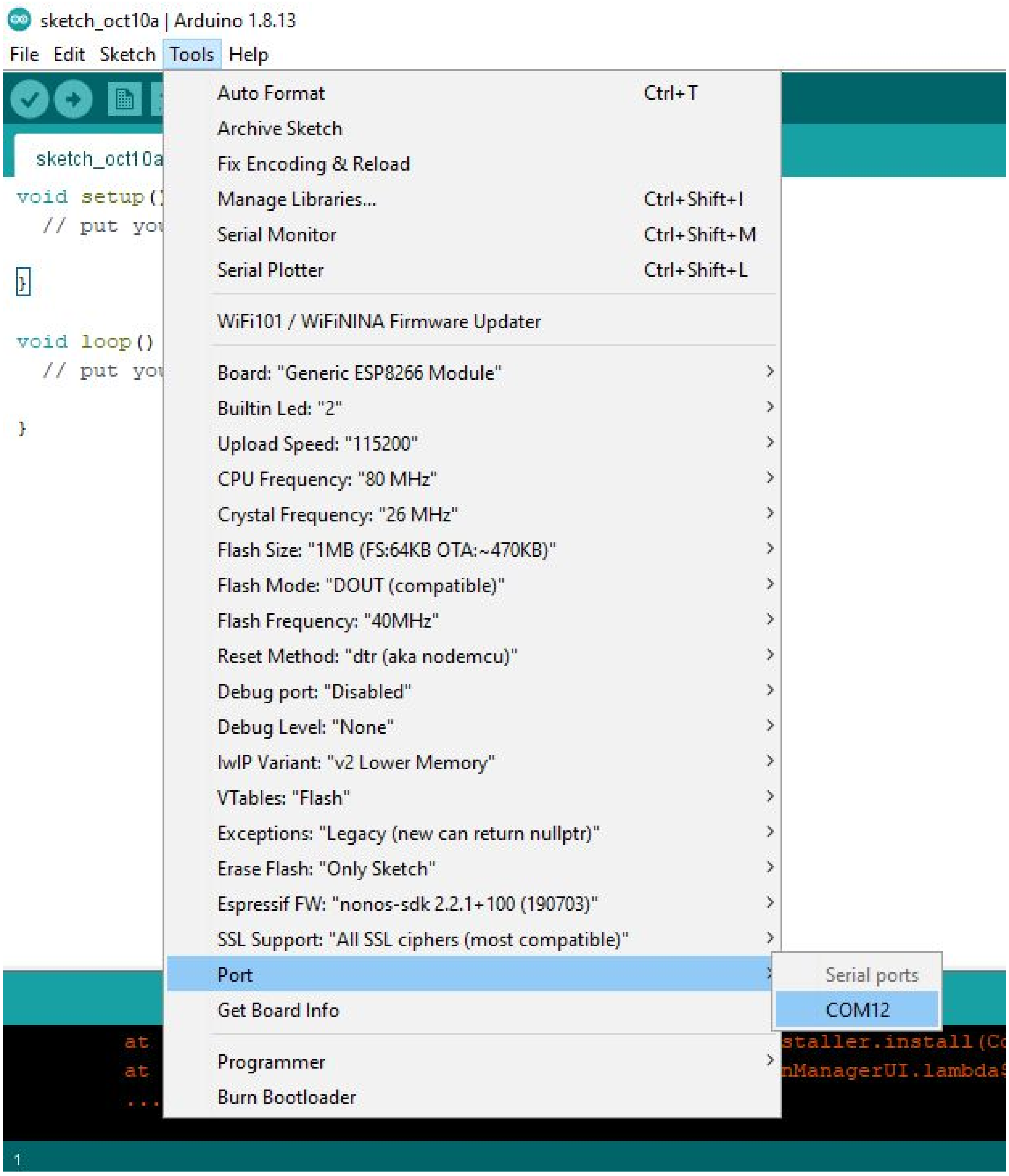


```
void setup()
  // put yo

}

void loop()
  // put yo

}
```

```
      at cc.arduino.contributions.packages.ContributionInstaller.install
      at cc.arduino.contributions.packages.ui.ContributionManagerUI.lamb
      ... 1 more
```

Step 5 - Selecting the Port

After selecting the board in Step 5, connect the ESP8266 module to the computer using the micro USB cable. Then go to **Tools -> Port** and select the port to which the NodeMCU is connected.



sketch_oct10a | Arduino 1.8.13

File Edit Sketch Tools Help

sketch_oct10a

```
void setup()
  // put you

}

void loop()
  // put you

}
```

Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries...	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater	
Board: "Generic ESP8266 Module"	
Builtin Led: "2"	
Upload Speed: "115200"	
CPU Frequency: "80 MHz"	
Crystal Frequency: "26 MHz"	
Flash Size: "1MB (FS:64KB OTA:~470KB)"	
Flash Mode: "DOUT (compatible)"	
Flash Frequency: "40MHz"	
Reset Method: "dtr (aka nodemcu)"	
Debug port: "Disabled"	
Debug Level: "None"	
lwIP Variant: "v2 Lower Memory"	
VTables: "Flash"	
Exceptions: "Legacy (new can return nullptr)"	
Erase Flash: "Only Sketch"	
Espressif FW: "nonos-sdk 2.2.1+100 (190703)"	
SSL Support: "All SSL ciphers (most compatible)"	
Port	Serial ports / COM12
Get Board Info	
Programmer	
Burn Bootloader	

```
        at                              staller.install(Co
        at                              nManagerUI.lambda$
        ...
```

1

Step 6 - Updating the code

After completing the above steps copy the code for the NodeMCU from the below GitHub link and paste it in the editor.

https://github.com/Green-core/Arduino/blob/master/green-core/nodemcu/nodemcu.ino

Then update the WiFi Username, Password and the backend URL (use the one you created earlier for the IoT device) and replace them in the following fields

- ssid
- password
- url

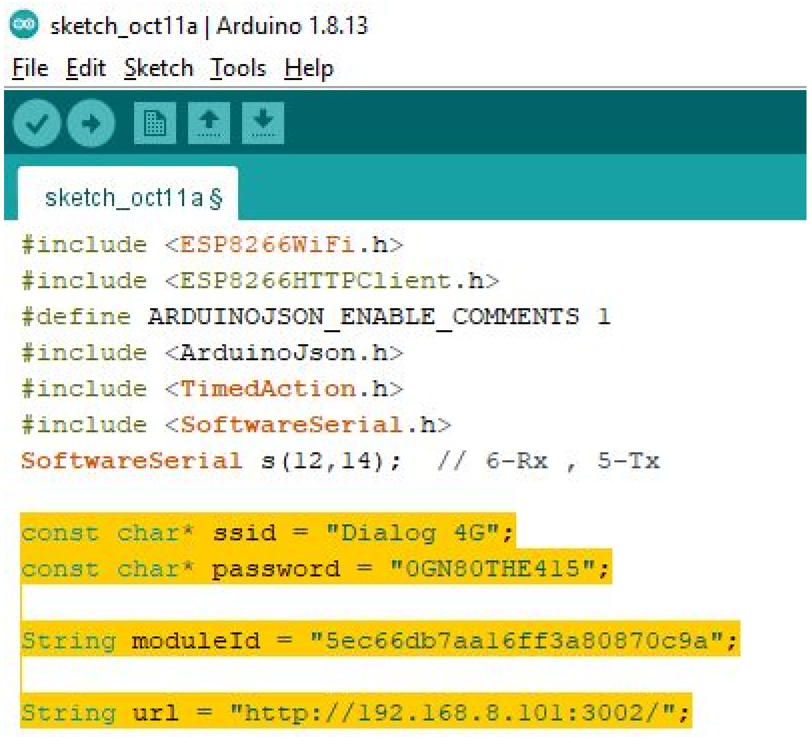
To get the module Id send a API request to {IoT backend URL}/units/connect-unit with the user's id and the user's name, then it will return a module id.

Get that module id and paste it in the 'moduleId' field.

Also make sure that the following libraries are imported (included) into the code

- ESP8266WiFi.h
- ESP8266HTTPClient.h
- ArduinoJson.h
- TimedAction.h
- SoftwareSerial.h

Then save the code.

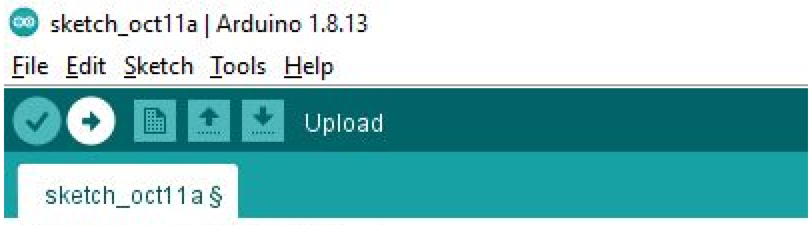


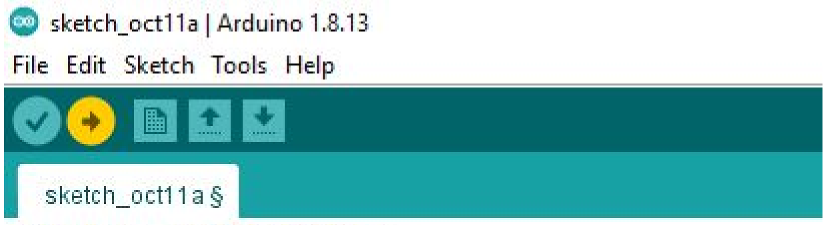
```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#define ARDUINOJSON_ENABLE_COMMENTS 1
#include <ArduinoJson.h>
#include <TimedAction.h>
#include <SoftwareSerial.h>
SoftwareSerial s(12,14);  // 6-Rx , 5-Tx

const char* ssid = "Dialog 4G";
const char* password = "0GN80THE415";

String moduleId = "5ec66db7aa16ff3a80870c9a";

String url = "http://192.168.8.101:3002/";
```

Step 7 - Uploading the code

After updating the code with valid WiFi credentials, module id and the url, you can either compile it and then upload or directly do both at the same time. Click on the upload button in the toolbar (second icon from left) then it will compile the code and upload it into the board.

Note: Please don’t remove the board until the code is uploaded completely

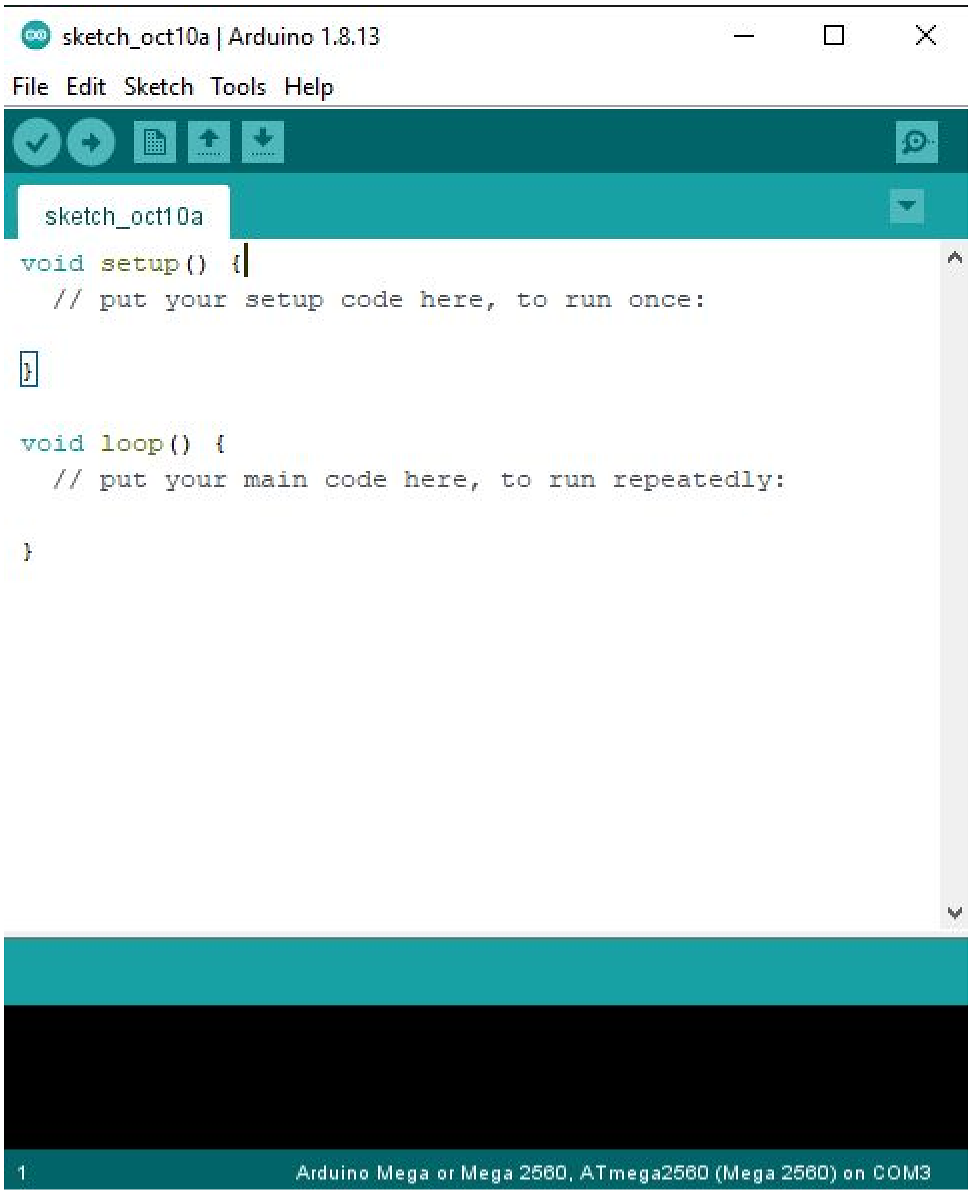




2. Setting up the Arduino Mega

Step 1 -

After starting the app you installed in step 0, you'll see the following screen

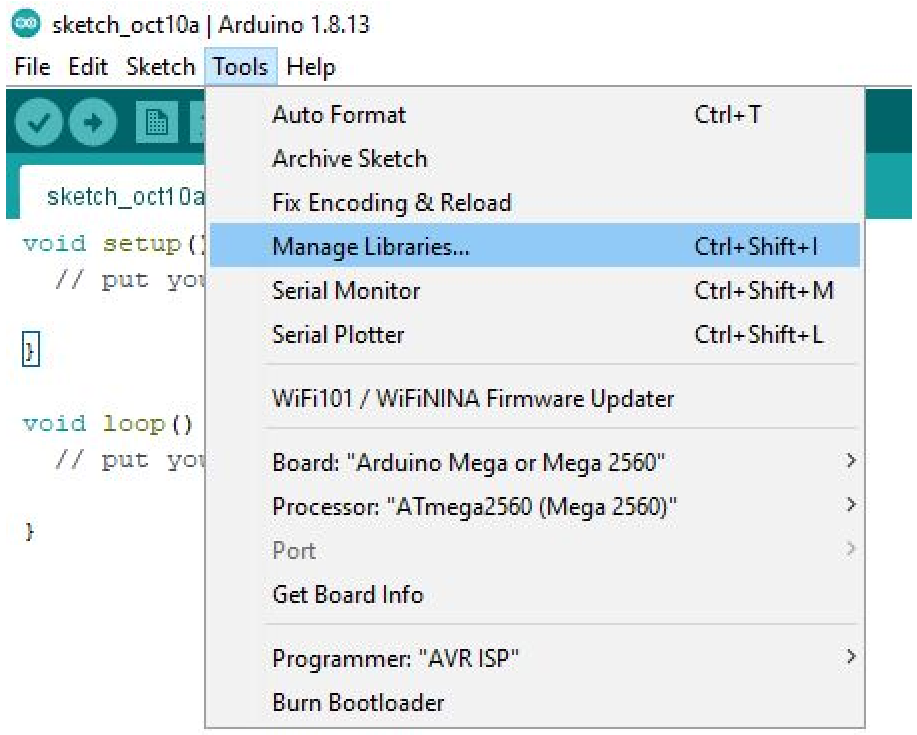


```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
1                    Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3
```

Step 2- Installing Libraries

Since the arduino mega board is inbuilt in the Arduino IDE you don’t need to install it. Go to **Tools -> Manage Libraries** to install libraries.


sketch_oct10a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_oct10a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port
Get Board Info
Programmer: "AVR ISP"
Burn Bootloader

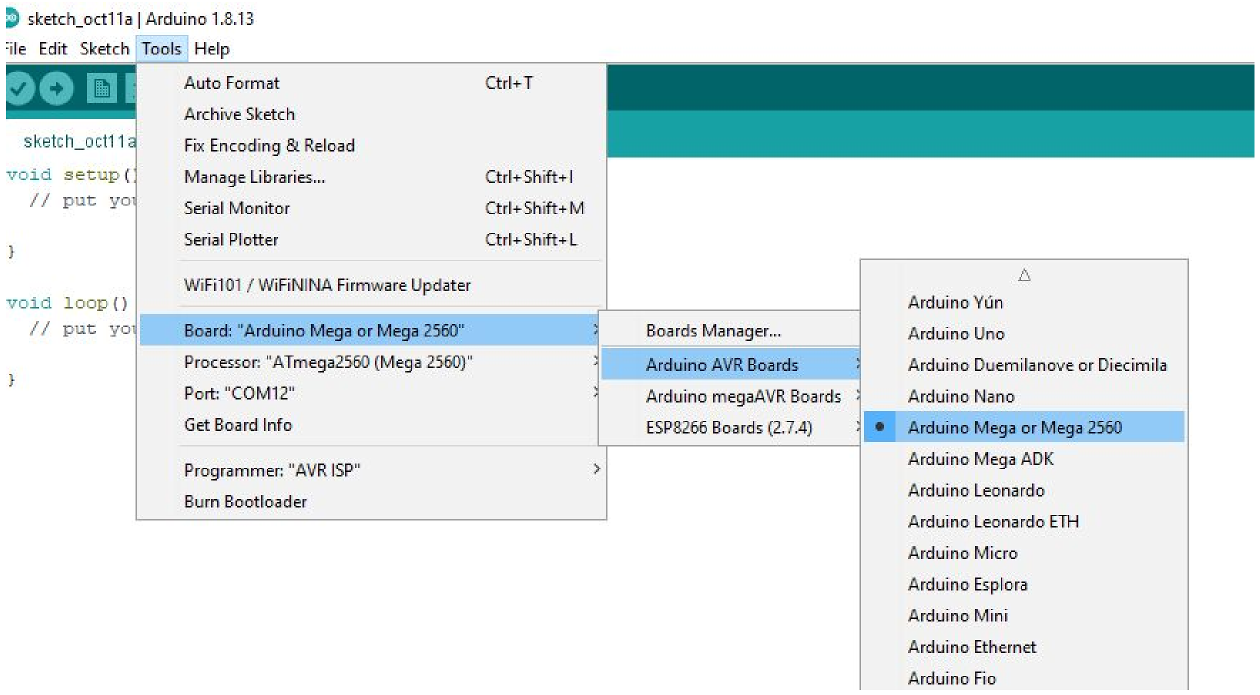
```
void setup(
  // put yo

}

void loop()
  // put yo

}
```

Then search for the following libraries and install them

- ArduinoJson - version 6.15.2 (ignore if already installed in previous setup)
- Adafruit_TSL2561_U - version 1.1.0
- DHT Sensor Library - version 1.3.10
- StreamUtils - version 1.5.0
- Adafruit Unified Sensor - version 1.1.4

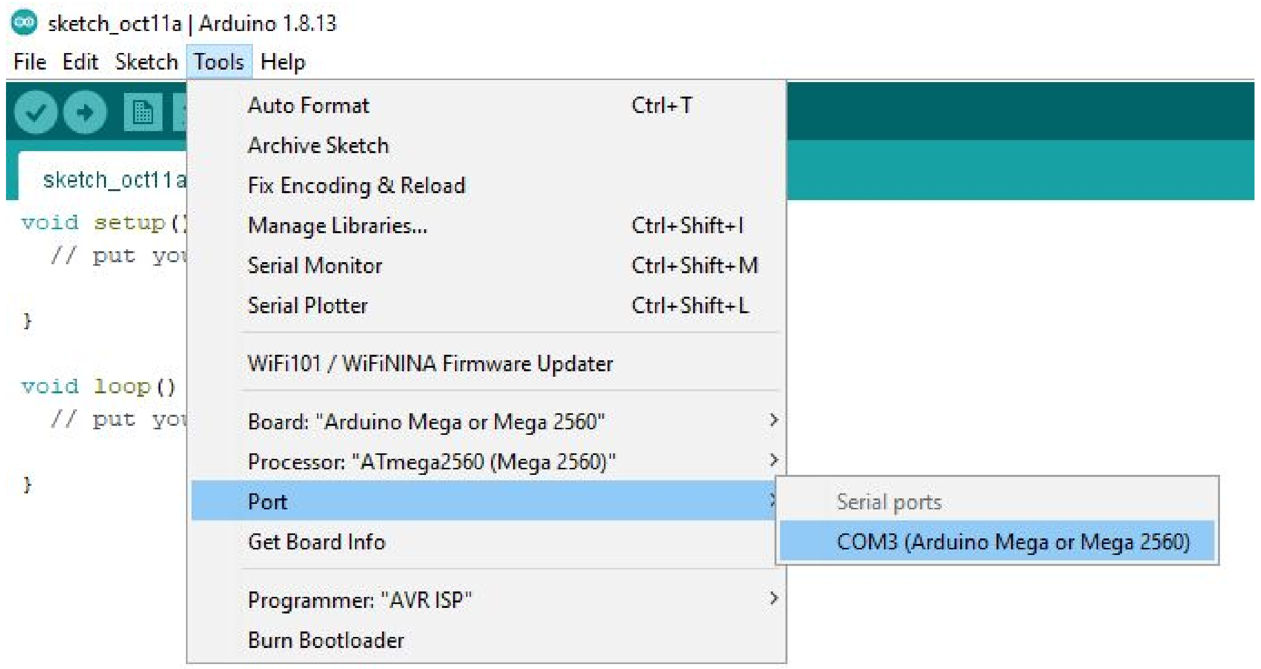
Step 3 - Selecting the board

After installing the libraries mentioned in step 4 to select the board go to **Tools -> Boards -> Arduino AVR Boards** and select the **Arduino Mega or Mega 2560**(Most probably the 5th one)



Step 4 - Selecting the Port

After selecting the board in Step 5, connect the Arduino Mega to the computer using the A to B USB cable (similar to a printer cable). Then go to **Tools -> Port** and select the port to which the Arduino Mega is connected.


sketch_oct11a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_oct11a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port
Get Board Info
Programmer: "AVR ISP"
Burn Bootloader
Serial ports
COM3 (Arduino Mega or Mega 2560)

```
void setup()
  // put yo

}

void loop()
  // put yo

}
```

Step 5 - Updating the code

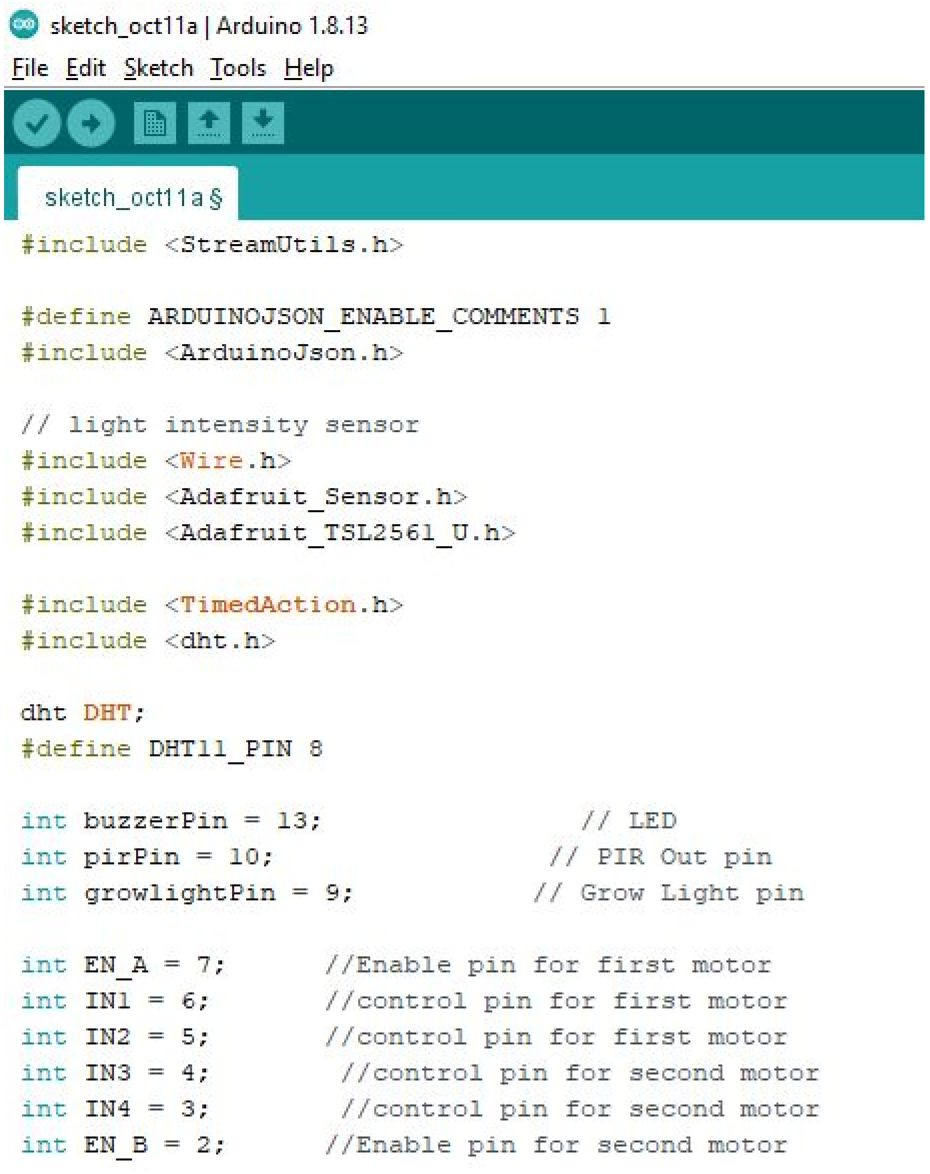
After connecting the device go to following URL and copy the code into the IDE

https://github.com/Green-core/Arduino/blob/master/green-core/arduino/arduino.ino

Then make sure that the following libraries are imported (included) into the code

- StreamUtils.h
- ArduinoJson.h
- Wire.h
- Adafruit_Sensor.h
- Adafruit_TSL2561_U.h
- TimedAction.h
- dht.h

And then save the code

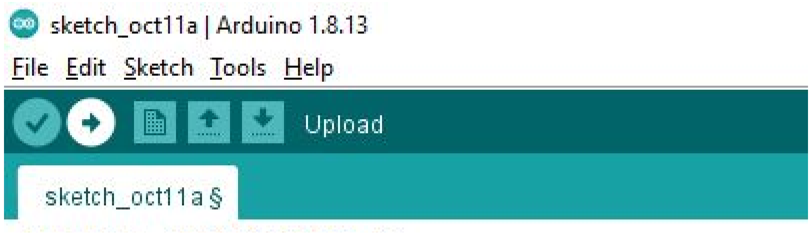


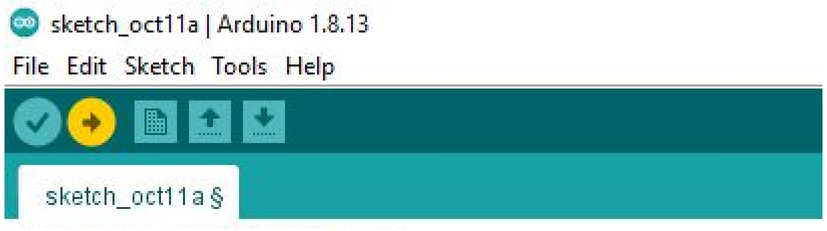
```
#include <StreamUtils.h>

#define ARDUINOJSON_ENABLE_COMMENTS 1
#include <ArduinoJson.h>

// light intensity sensor
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>

#include <TimedAction.h>
#include <dht.h>

dht DHT;
#define DHT11_PIN 8

int buzzerPin = 13;                 // LED
int pirPin = 10;                   // PIR Out pin
int growlightPin = 9;             // Grow Light pin

int EN_A = 7;      //Enable pin for first motor
int IN1 = 6;       //control pin for first motor
int IN2 = 5;       //control pin for first motor
int IN3 = 4;        //control pin for second motor
int IN4 = 3;        //control pin for second motor
int EN_B = 2;      //Enable pin for second motor
```

Step 6 - Uploading the code

After saving the code, you can either compile it and then upload or directly do both at the same time. Click on the upload button in the toolbar (second icon from left) then it will compile the code and upload it into the board.

Note: Please don’t remove the board until the code is uploaded completely





After uploading the code for both the NodeMCU and the Arduino Mega connect them to the micro USB cable and the AC - DC converter cable provided and power it on.

Note: Make sure to switch on the two switches you switched off earlier before setting up the two devices