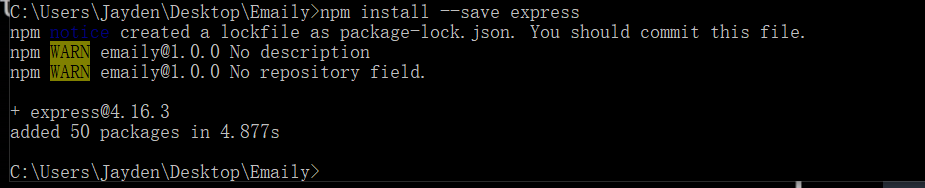
# Initial Project:

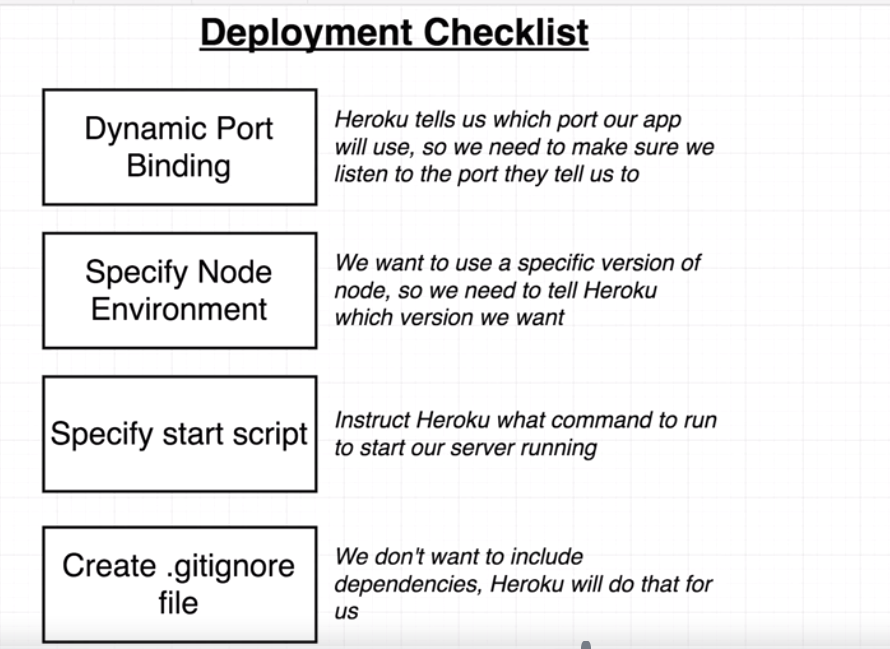


# Install express

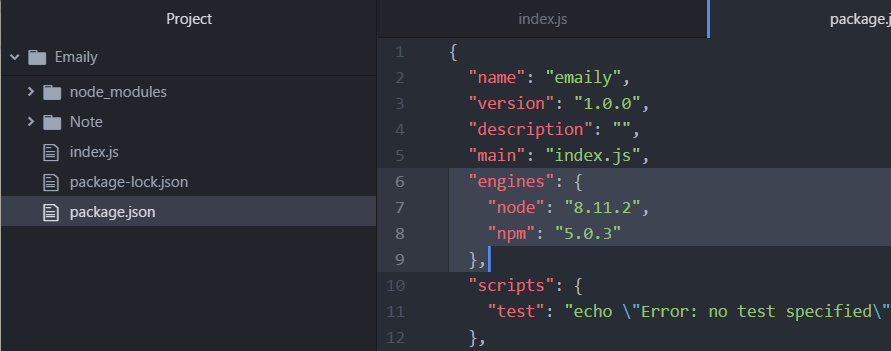


# Development Project

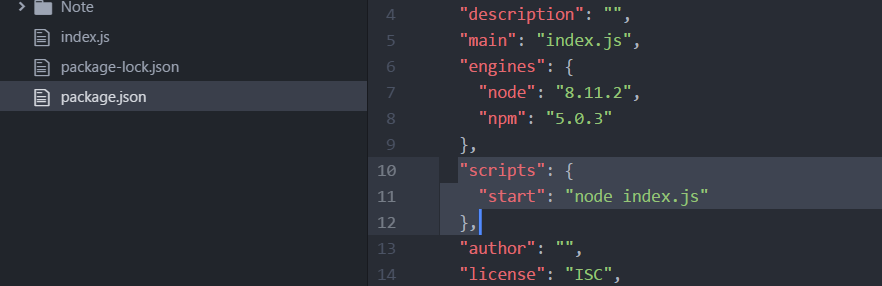
## Heroku



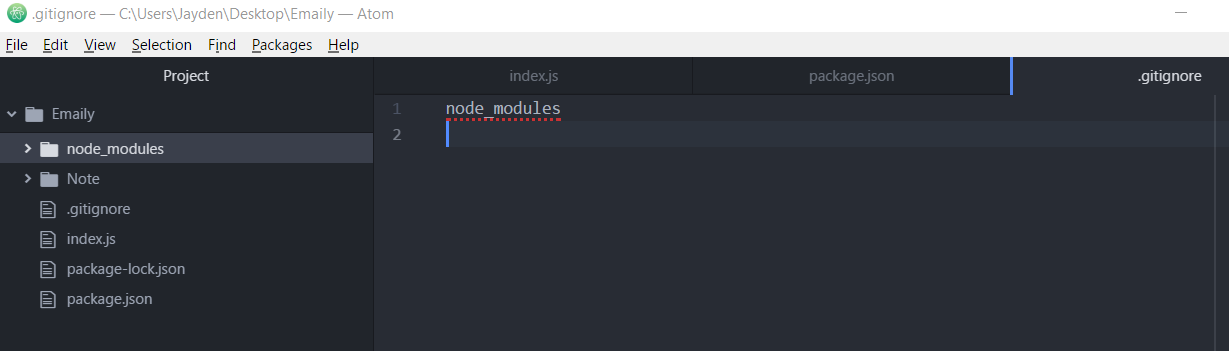
### Let heroku knows which node version we are using.



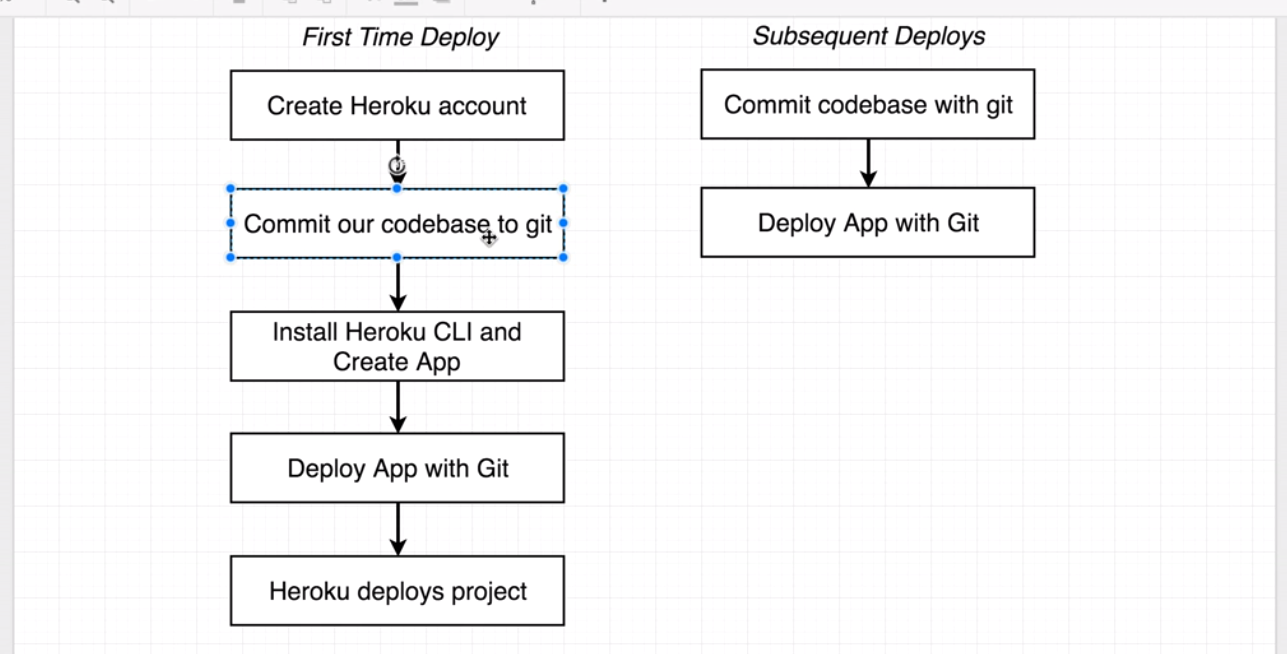
### Let Heroku knows which command to run our app.



### Ignore dependencies when push to git

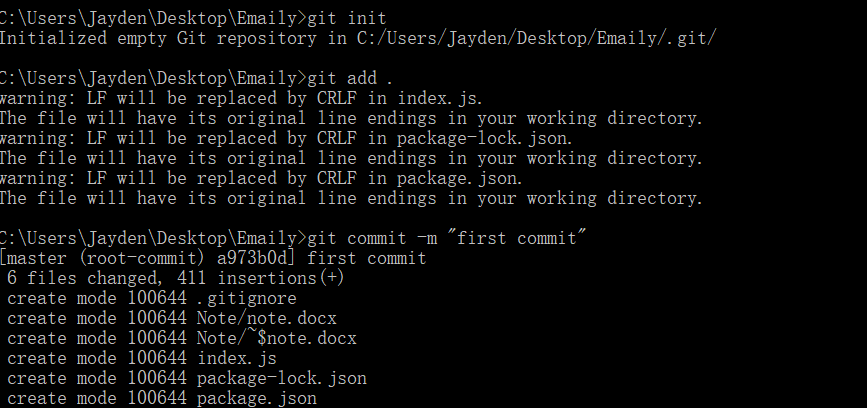


# Deploy app to heroku



## Commit code to git

<https://github.com/wzbjmgs/emaily>





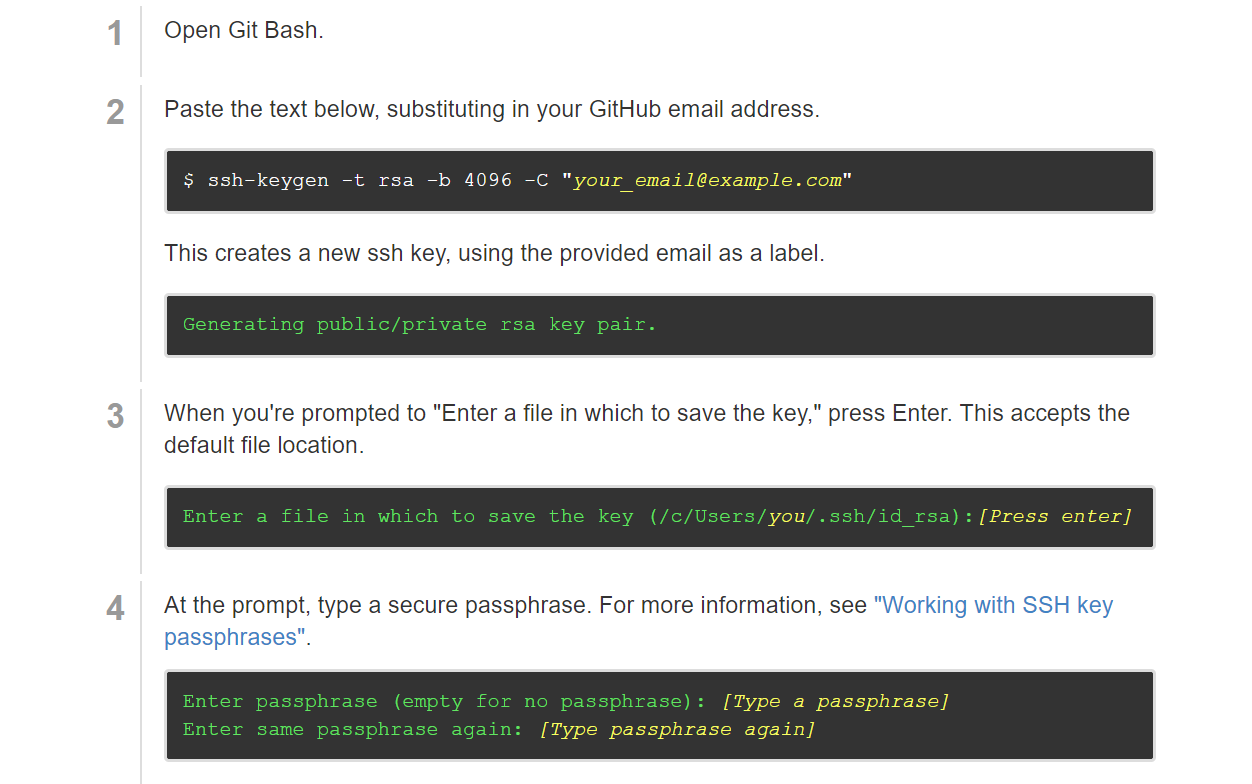
Need one more command after git init

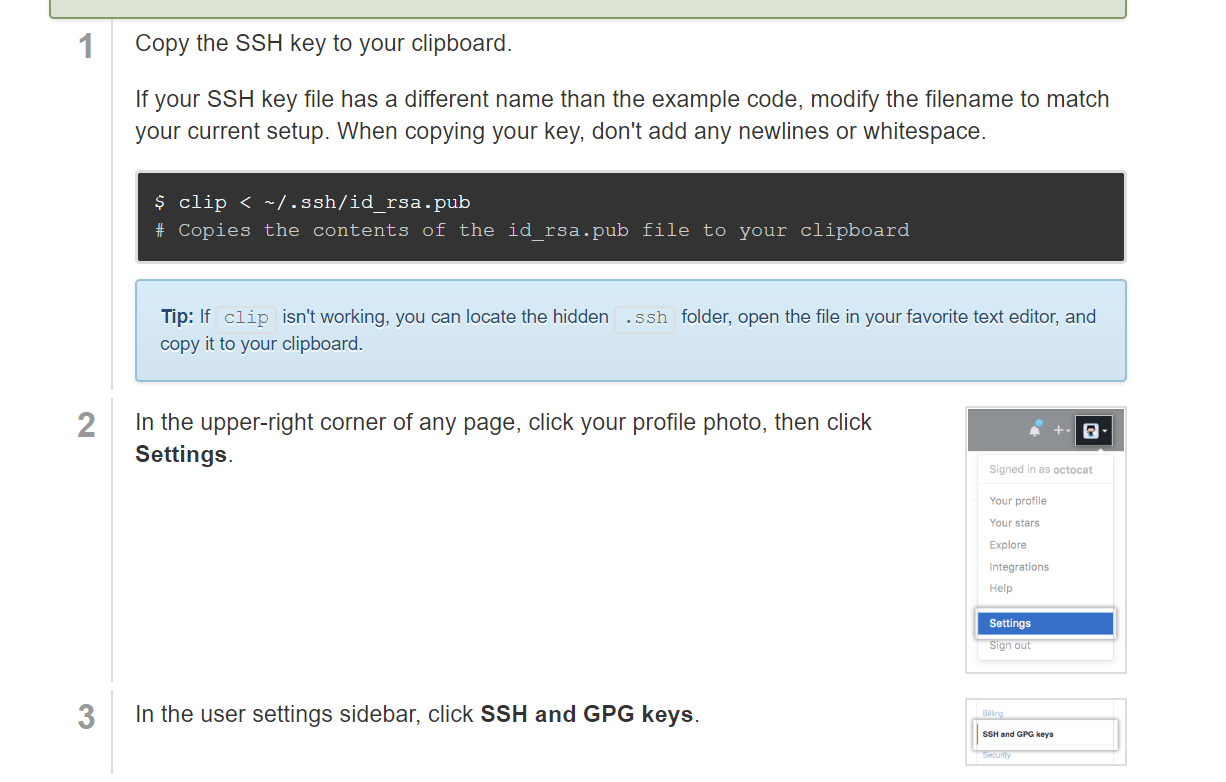
Git add .

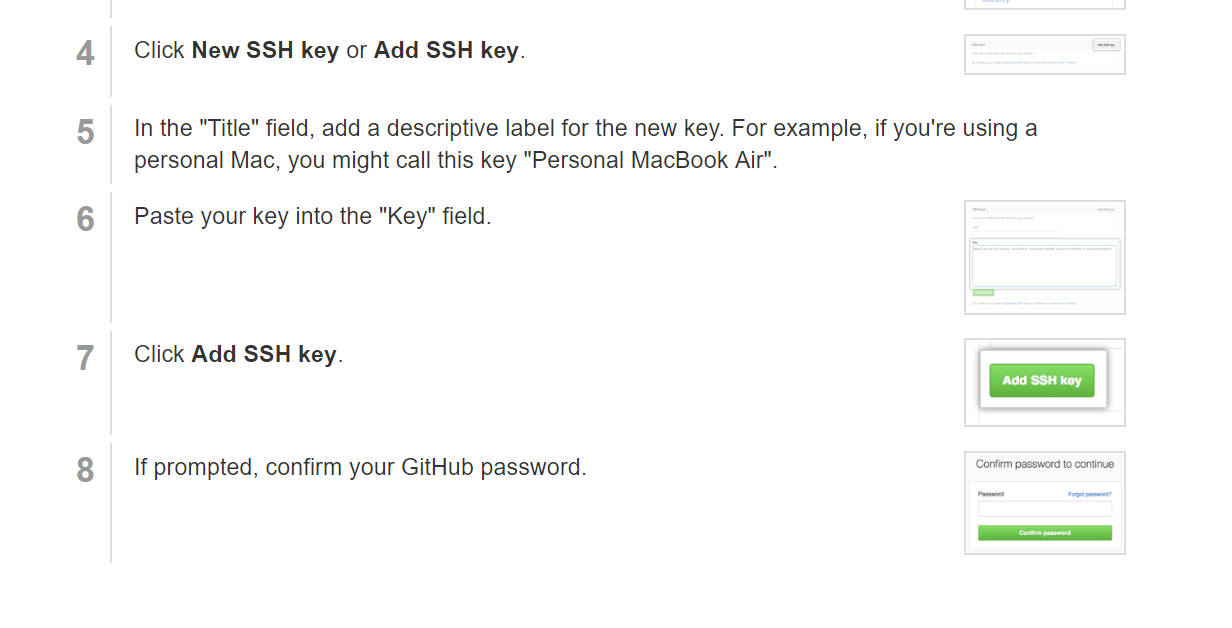
### Generate git SSH

<https://help.github.com/articles/checking-for-existing-ssh-keys/>

<https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/>

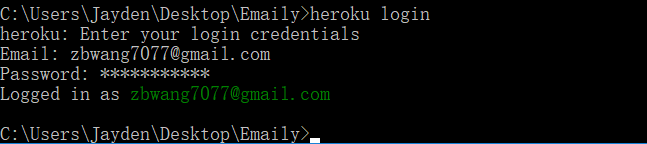






## Create Heroku new app

### Login



### Create



### First link: url to access our application in heroku

<https://agile-beyond-57534.herokuapp.com/>

### Second link: git repo

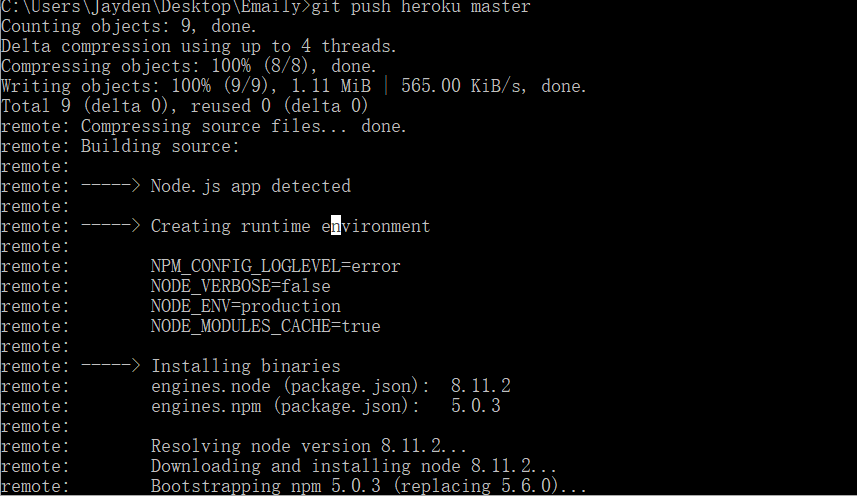
<https://git.heroku.com/agile-beyond-57534.git>

We need to push our local repo to this remote repo.

## Deploy app with git



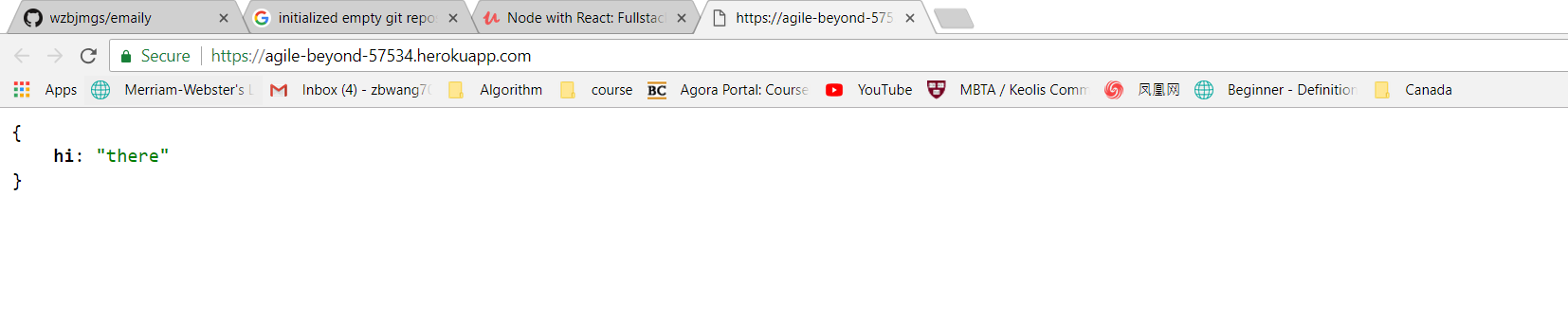
Heroku is the name of the repo



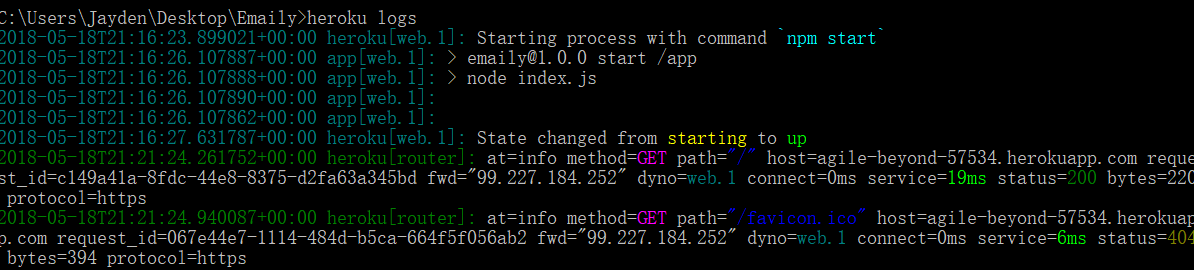
## Test our application



Type this command then a new page opened automatically

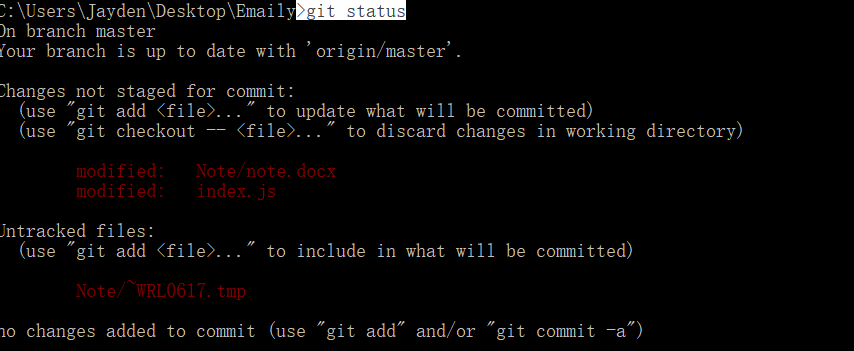


## Check log files

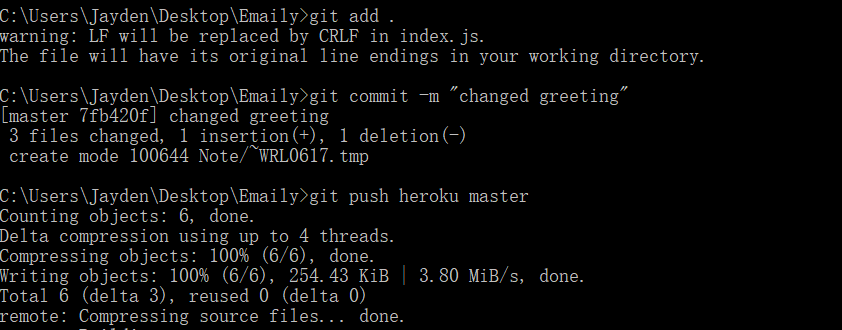


## Commit code when changes happen

First check status



Then commit

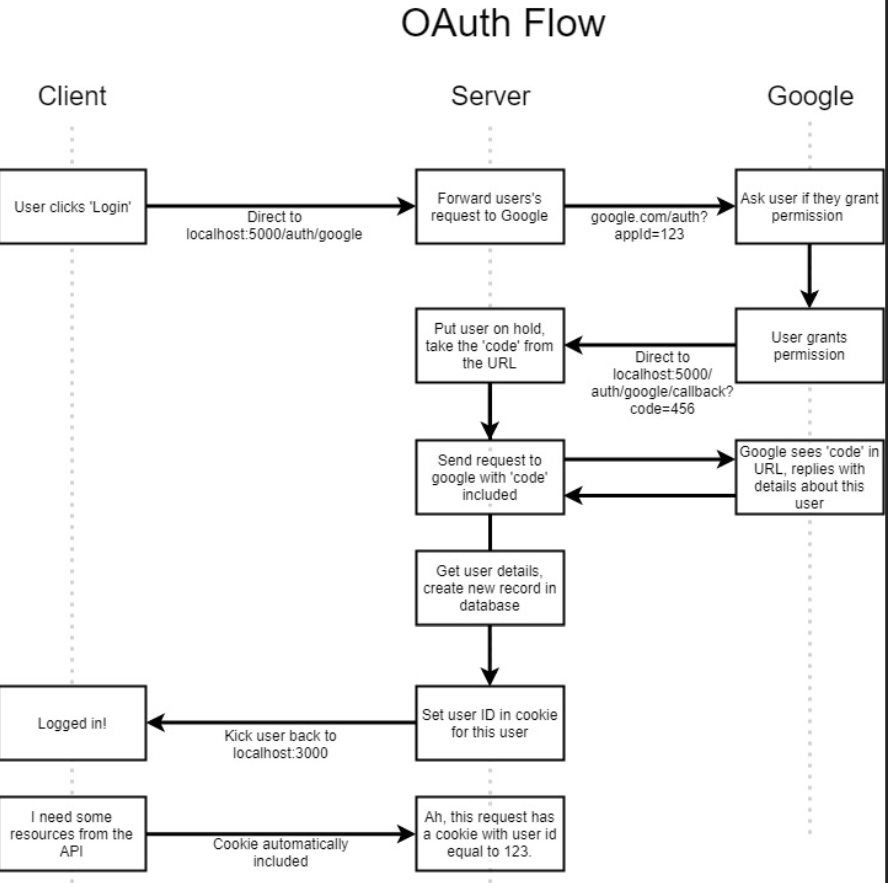


Run Heroku open again to check our application.

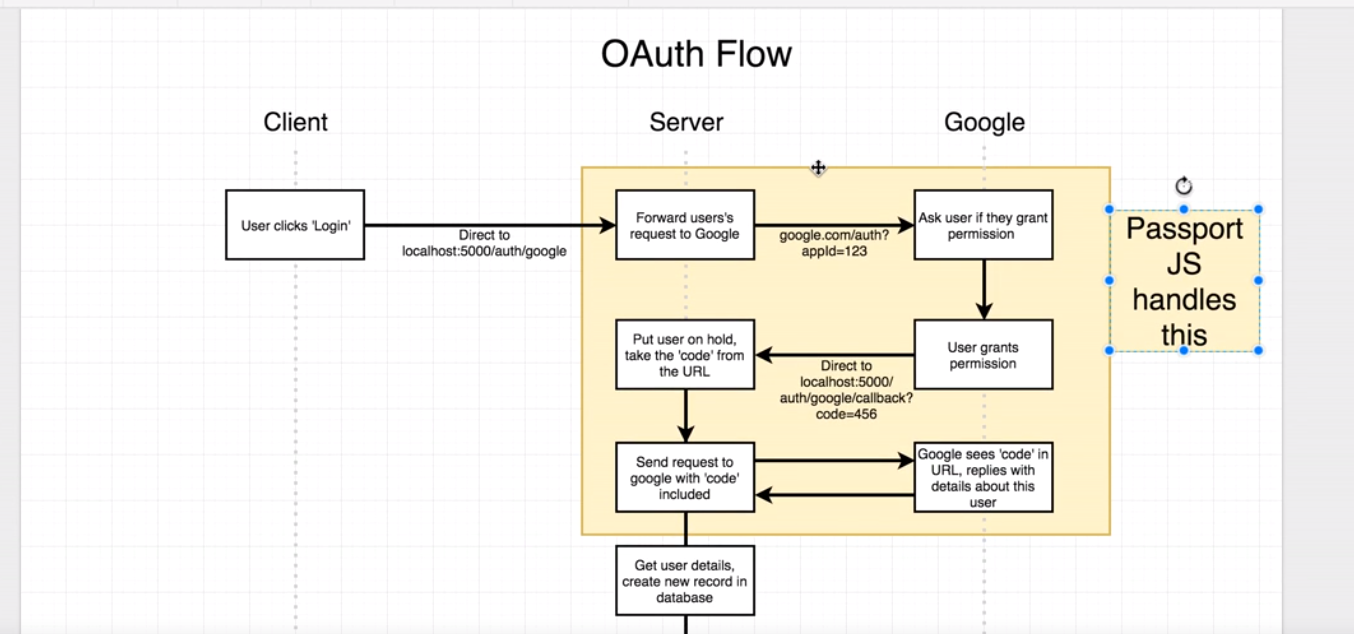
# Google OAuth



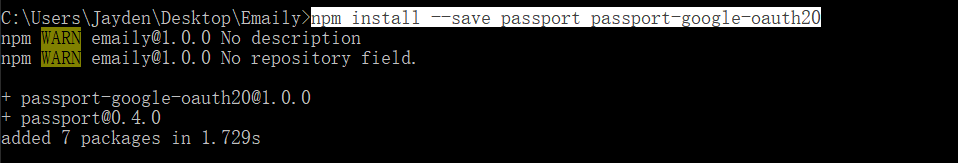
## OAuth Flows



## Passport JS

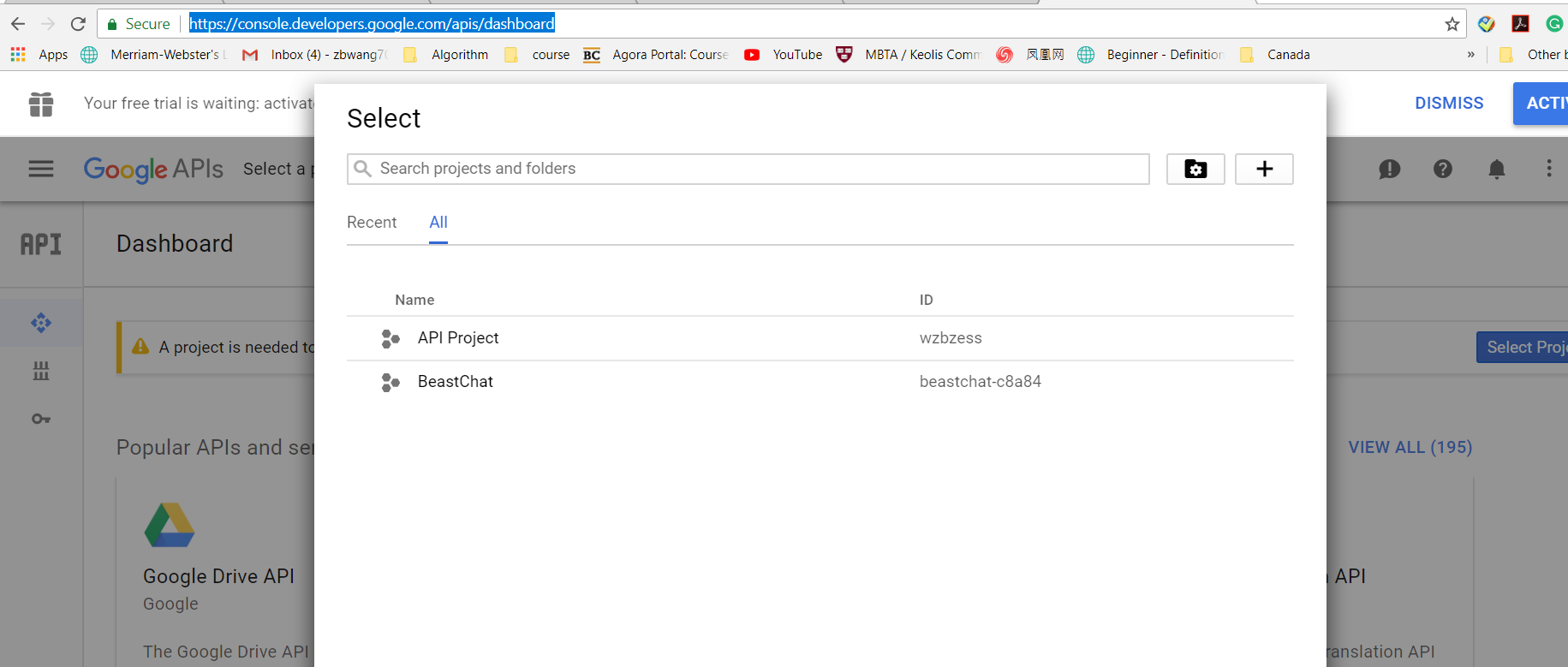


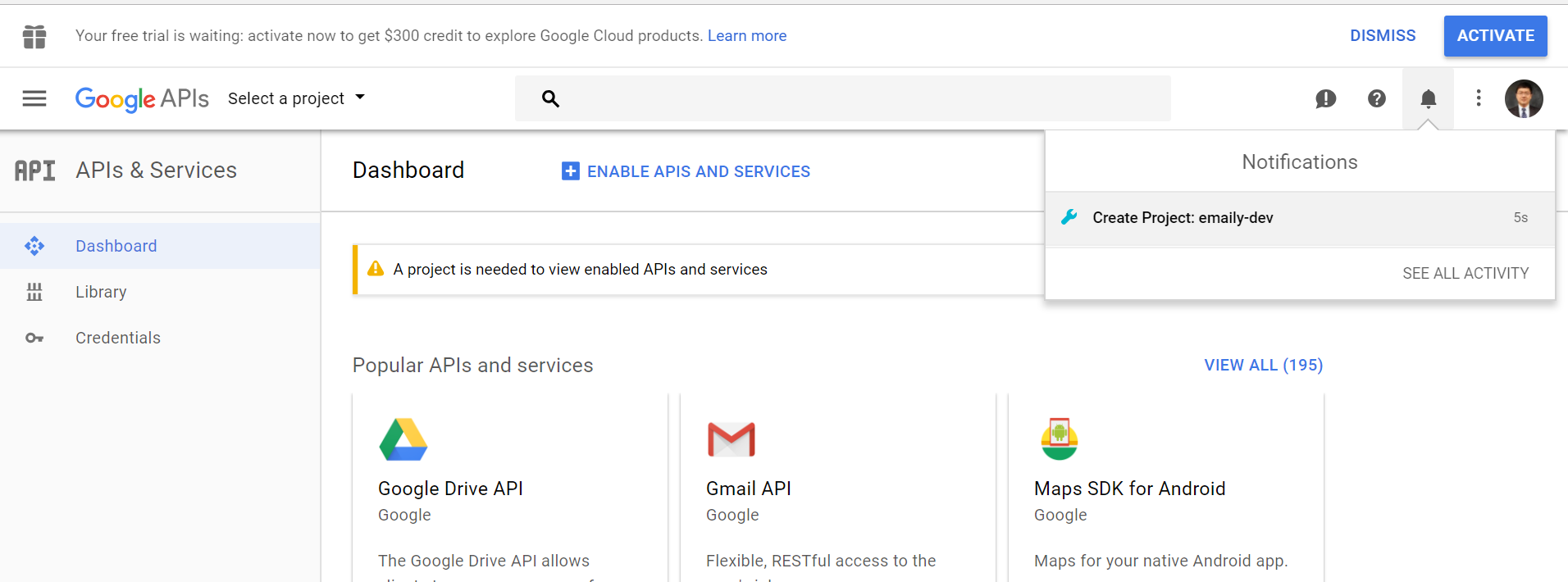
## Install passport



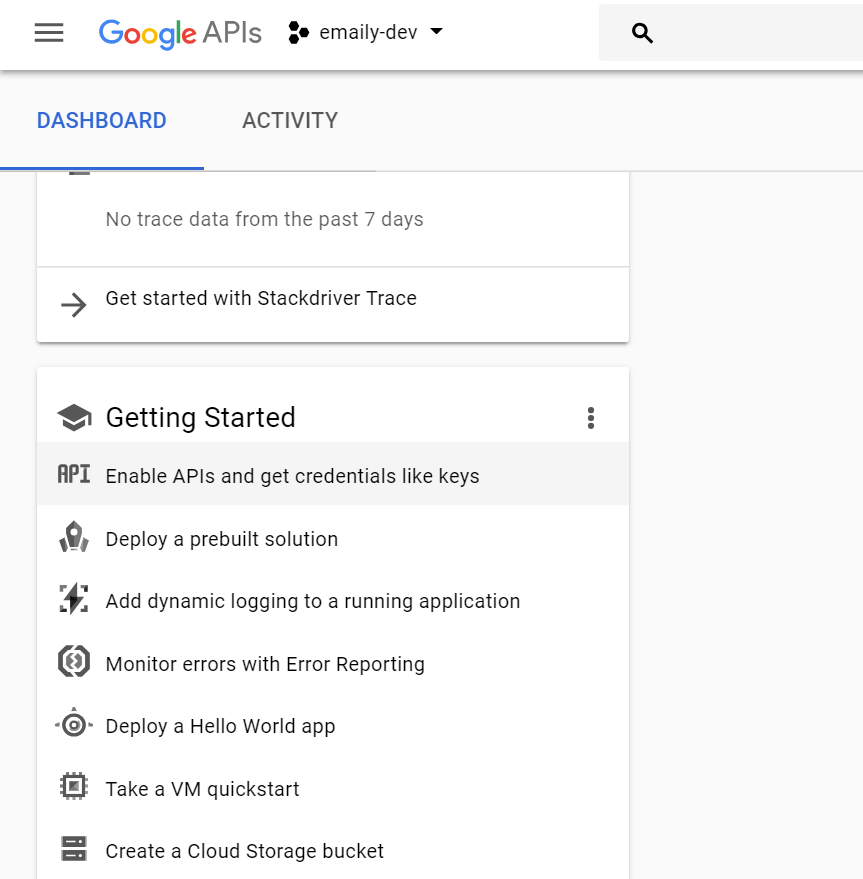
## Generate a new project to use Google oauth

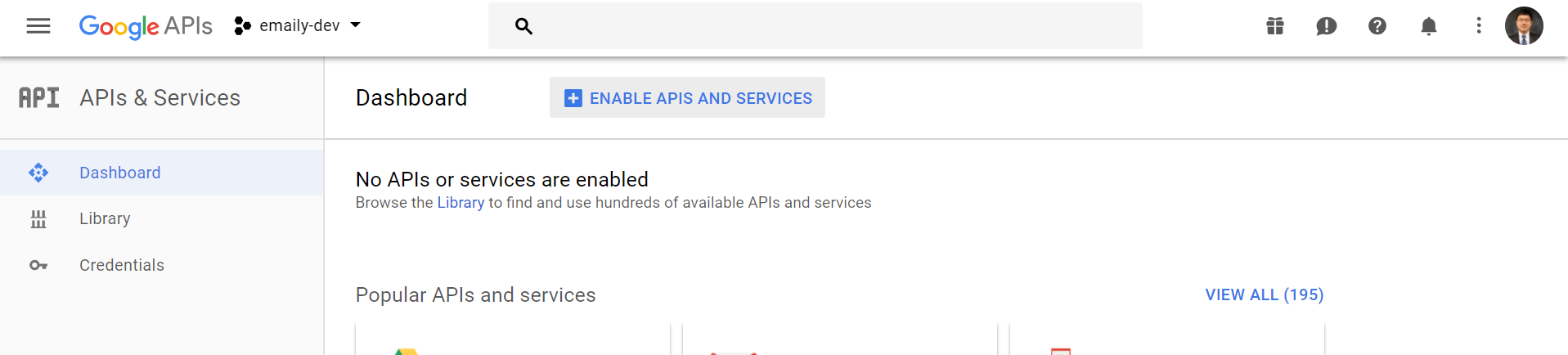
<https://console.developers.google.com/apis/dashboard>





## Enable API

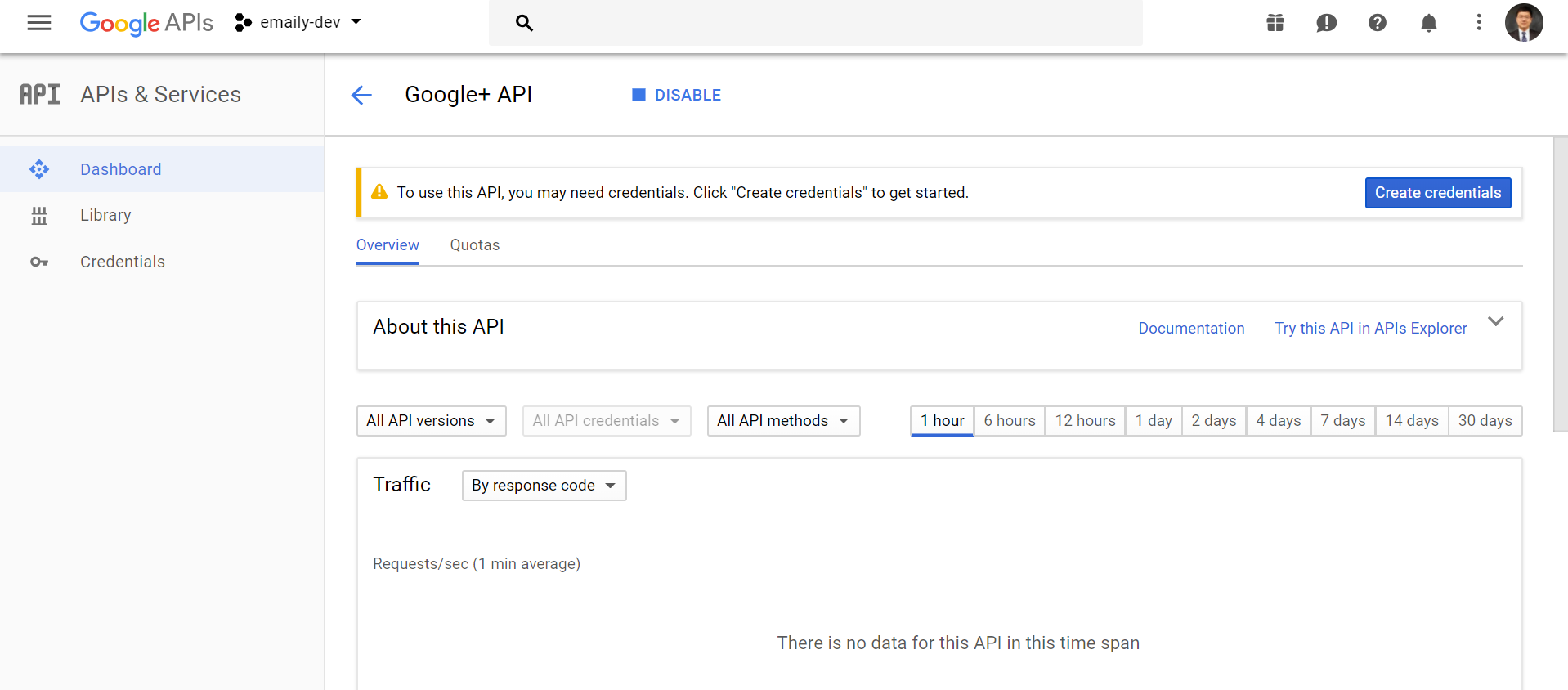




## Enable Specific Google+

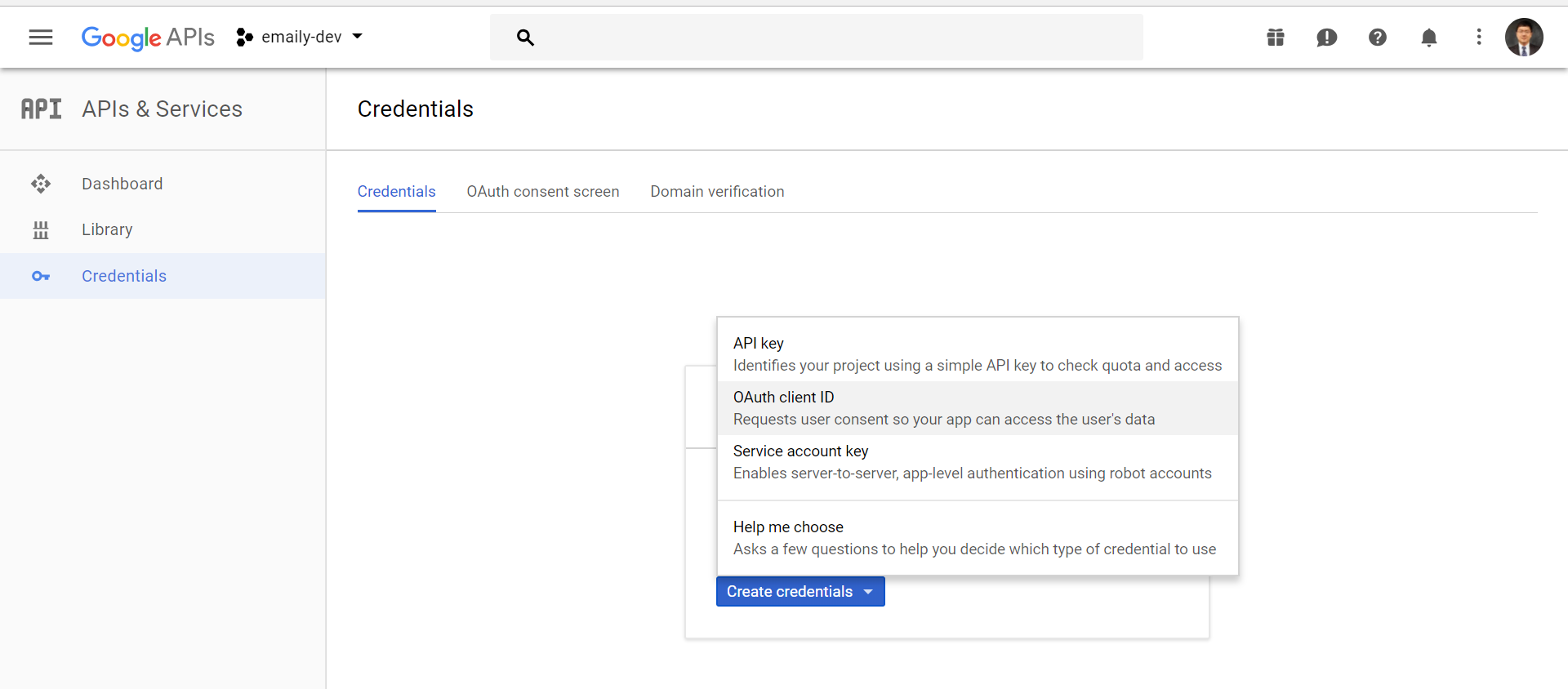


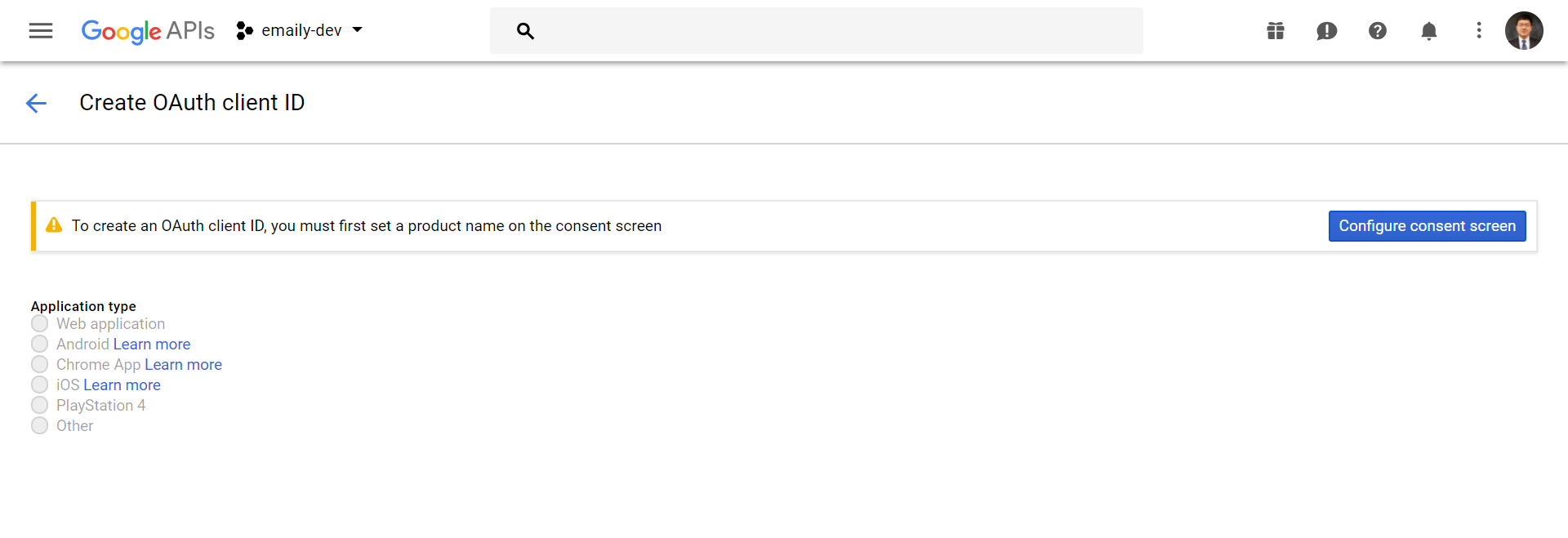
## Enable this API



## Generate credentials to access this API

OAuth client ID

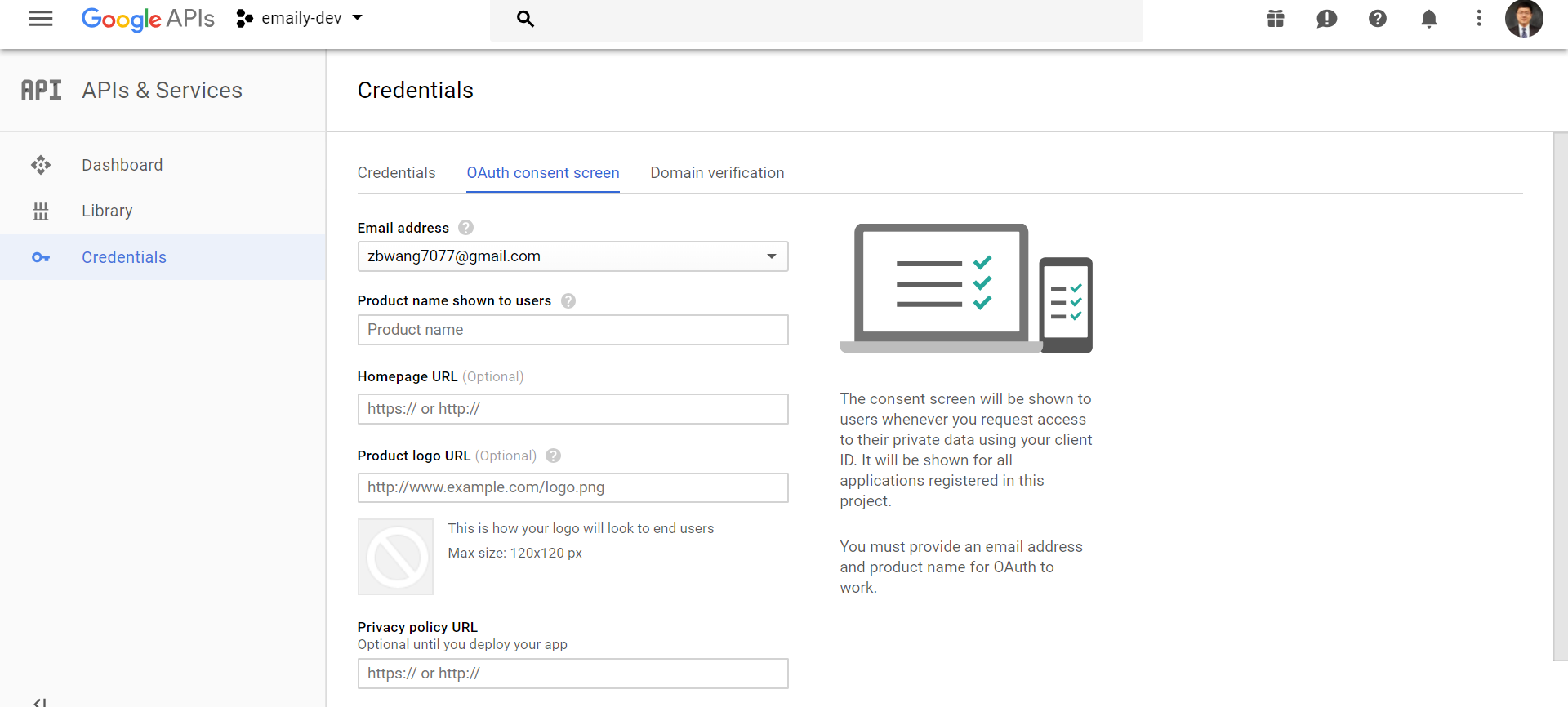




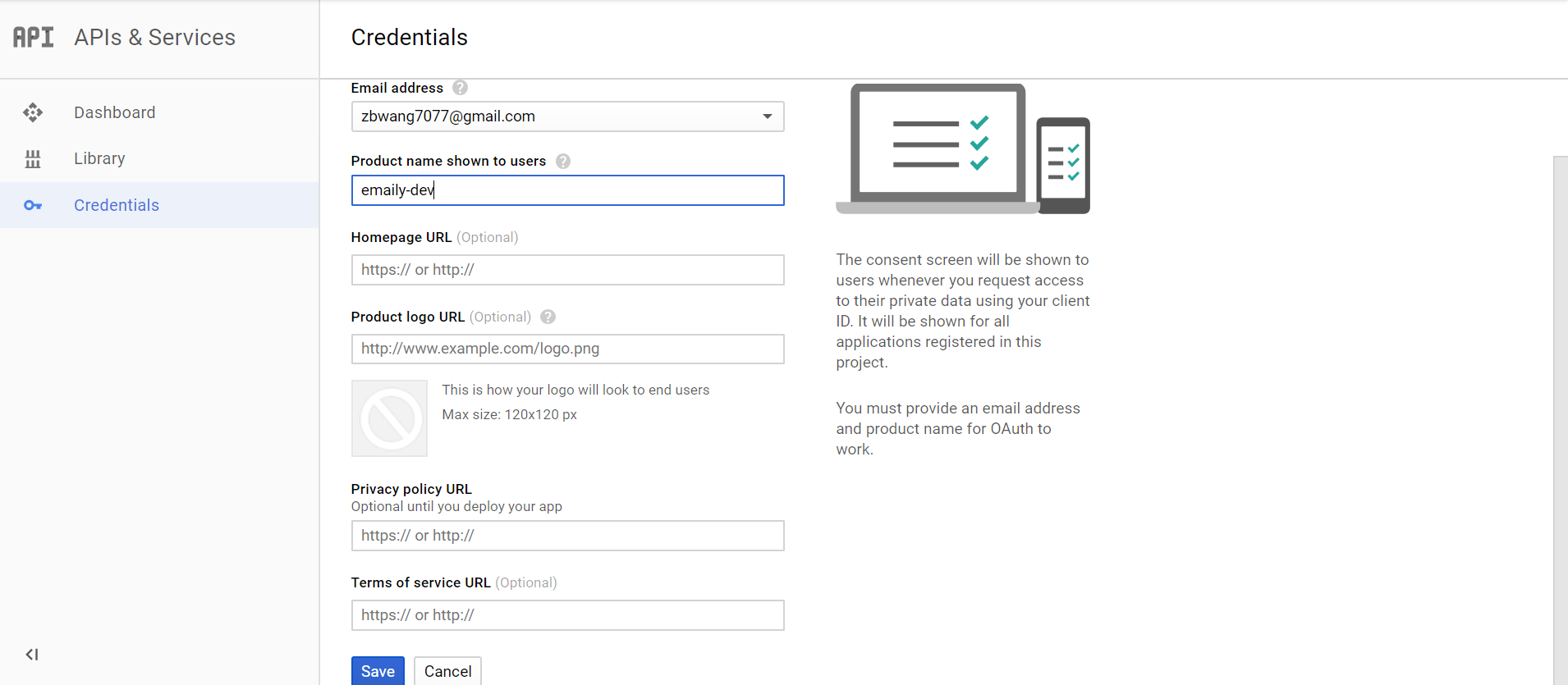
## Consent screen

The screen to ask user grant the permission

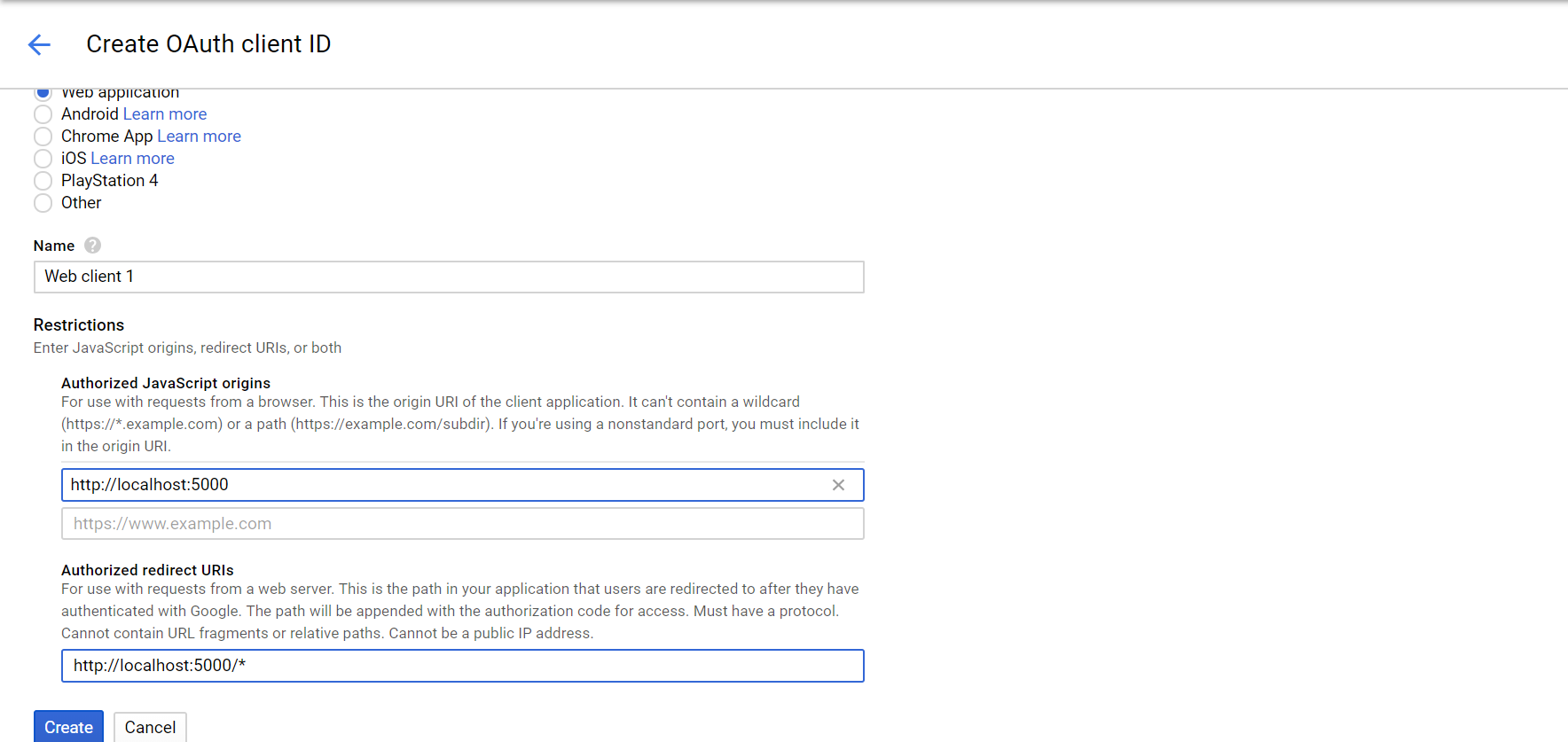
Hit configure consent screen



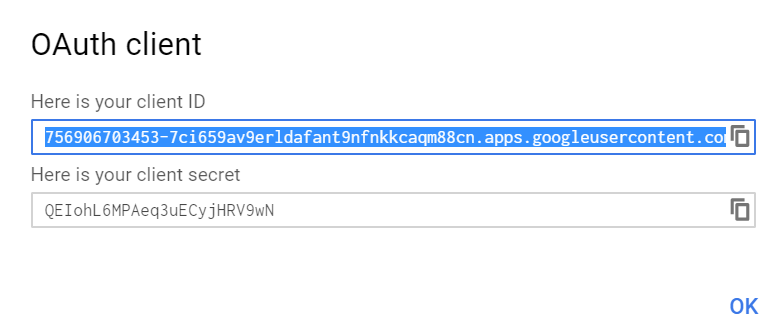
## Add product name



## Choose web application



Then hit create button.

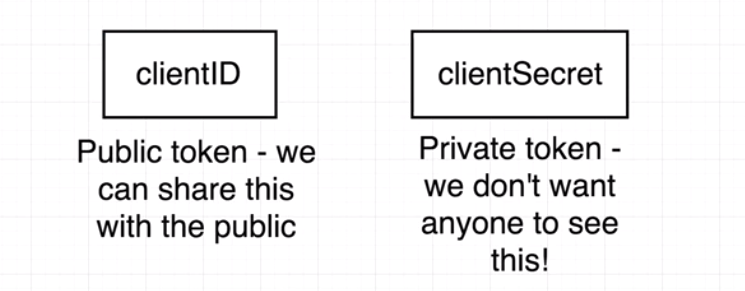


### Client id:

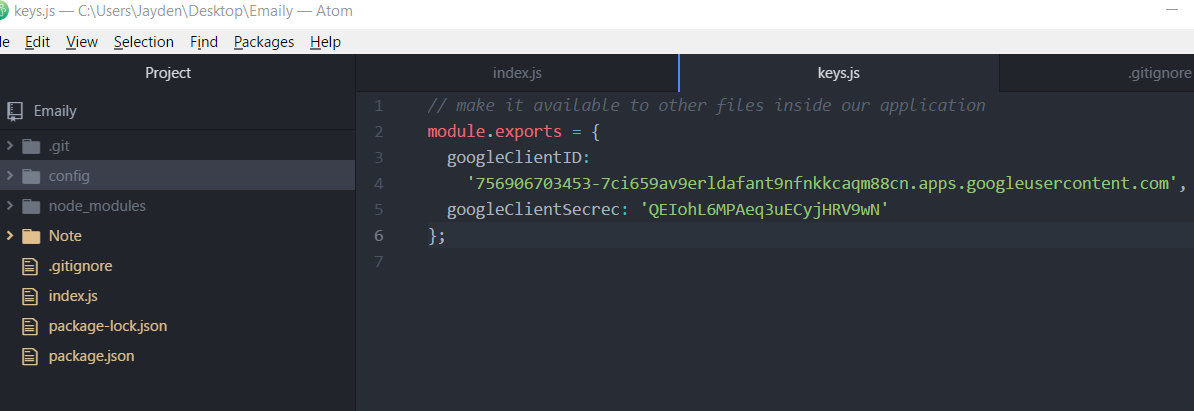
756906703453-7ci659av9erldafant9nfnkkcaqm88cn.apps.googleusercontent.com 703453-

### client Secret:

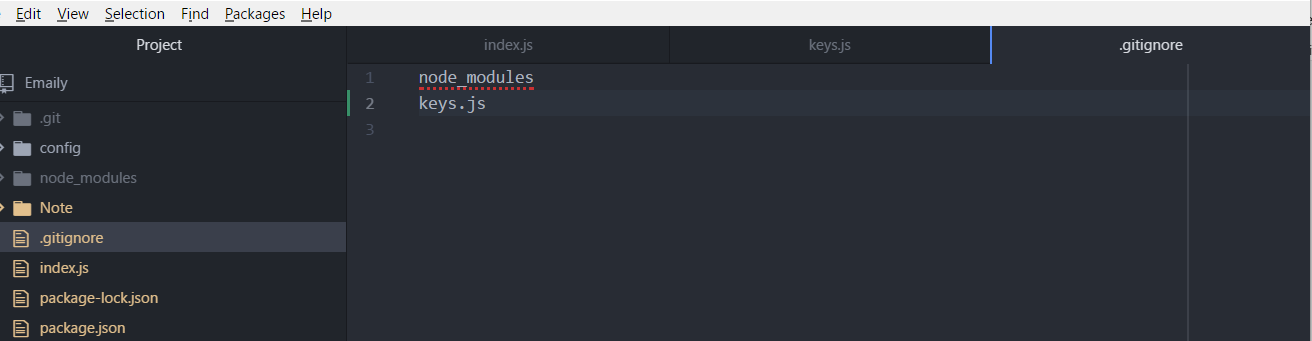
QEIohL6MPAeq3uECyjHRV9wN



## Safely save key



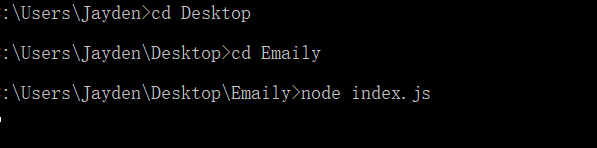
Do not commit into github

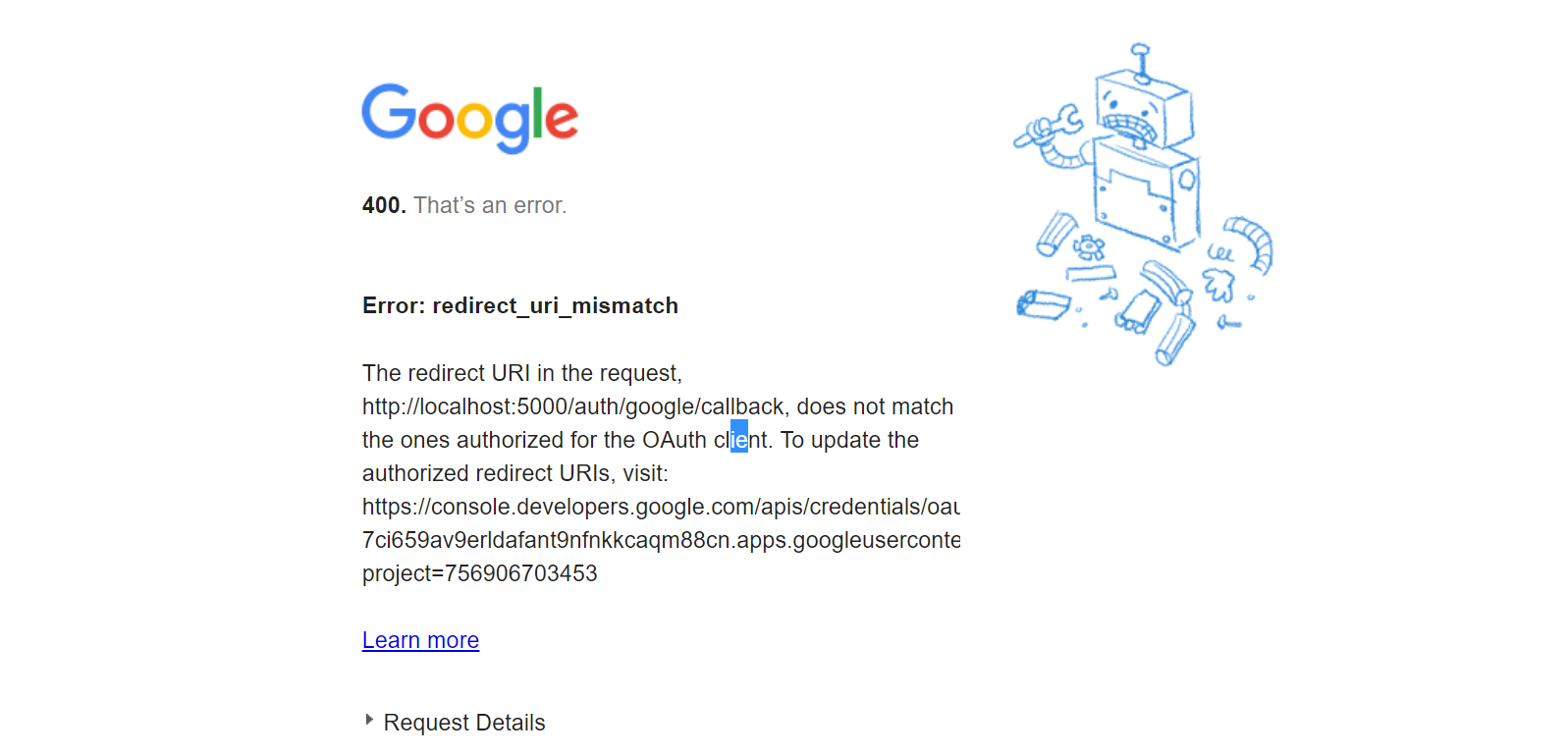


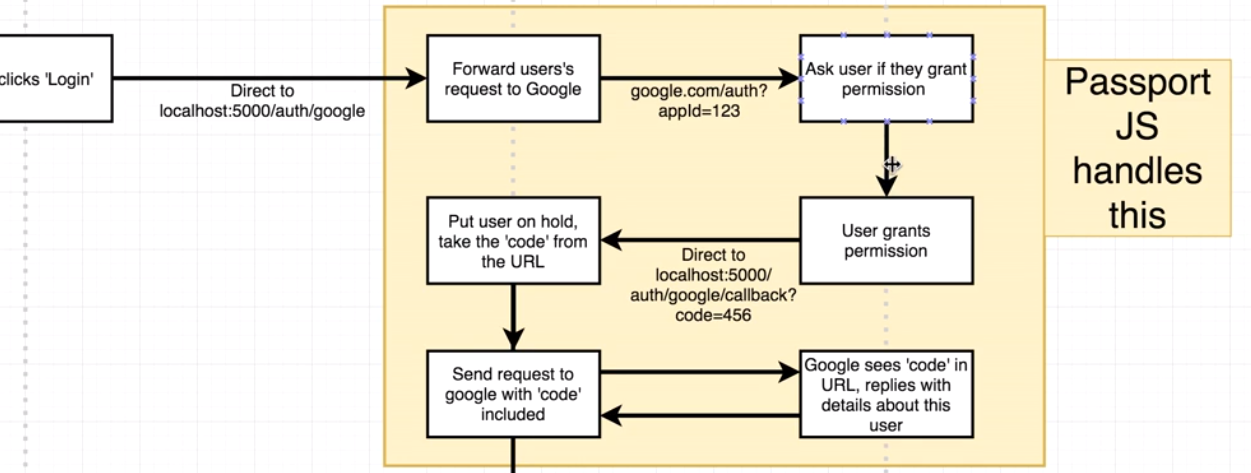
Cmd 进入emaily folder.

node index.js run our application

open browser, type localhost:5000/auth/google, get error page







https://accounts.google.com/o/oauth2/v2/auth? (forward to google)

response\_type=code& (hope get response code which is going to ask user profile)

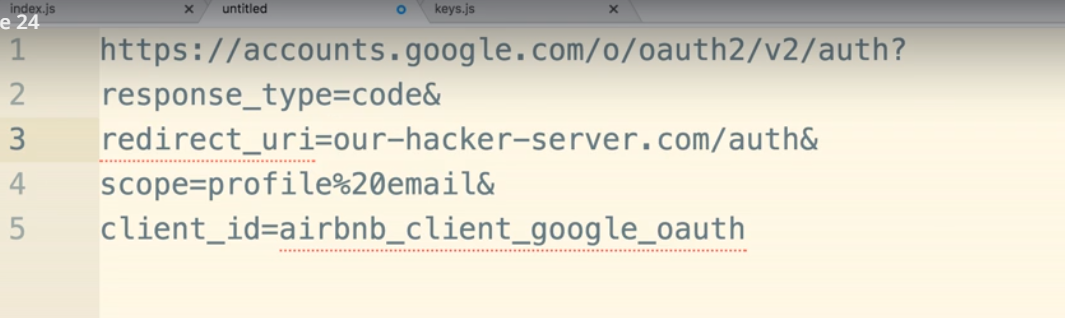
redirect\_uri=http%3A%2F%2Flocalhost%3A5000%2Fauth%2Fgoogle%2Fcallback& (call back link)

scope=profile%20email& (user info scope)

client\_id=756906703453-7ci659av9erldafant9nfnkkcaqm88cn.apps.googleusercontent.com (send client id to google to identify our application) The user whoever on this client ID want to get access to your account

### One bad way is to hack this auth.

Pretent to be Airbnb auth, and send call back to my own server to access user account

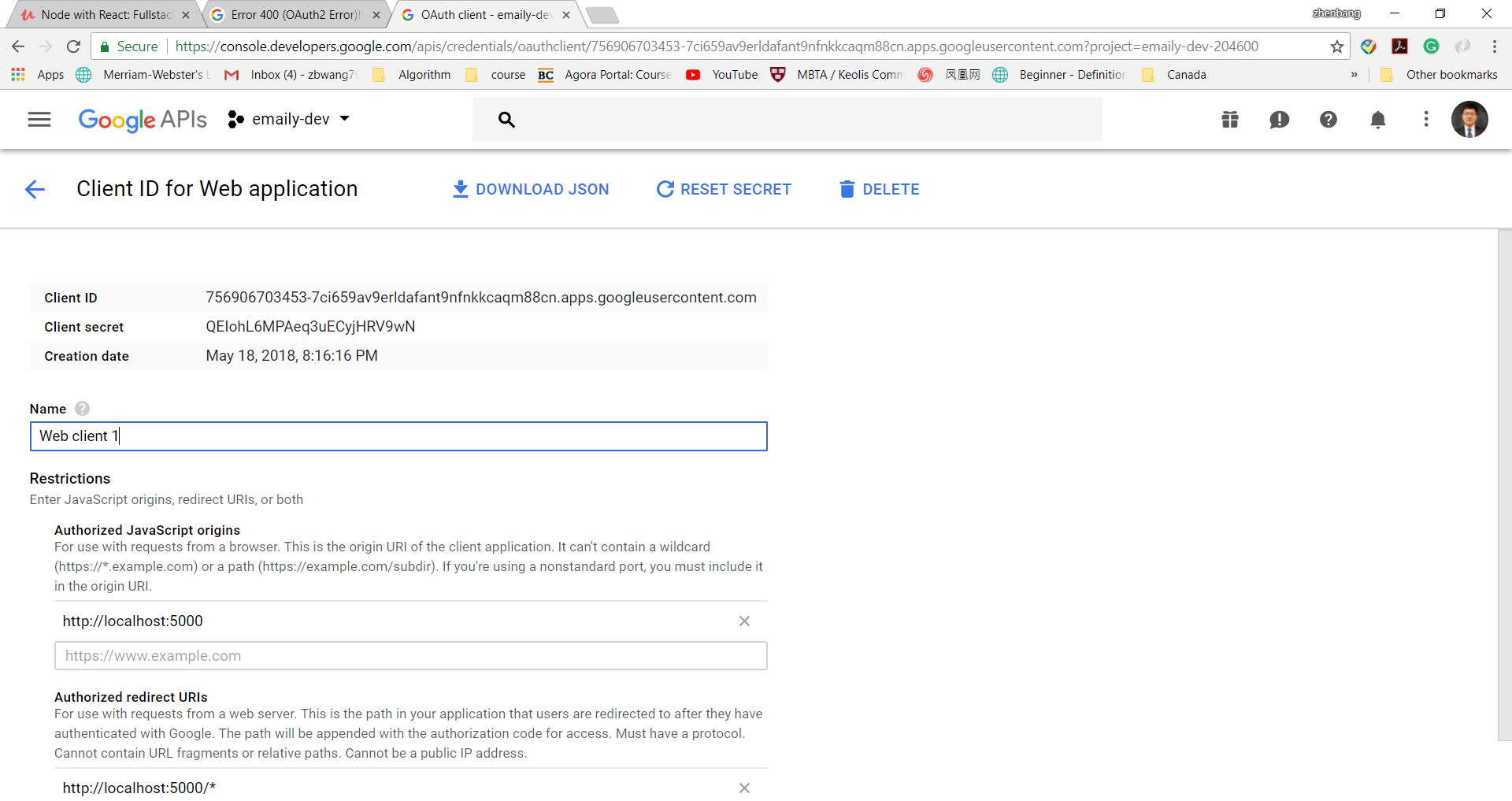


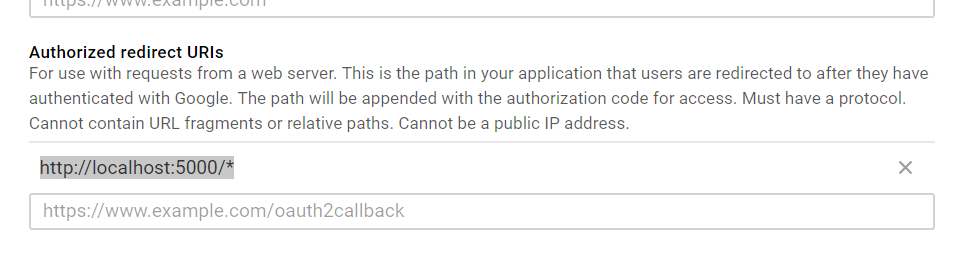
### Method to prevent this

Google error message tell us that **Error: redirect\_uri\_mismatch.** The redirect URI in the request, http://localhost:5000/auth/google/callback, does not match the ones authorized for the OAuth client.

### Go to the link provided by this error message to setup call back link.

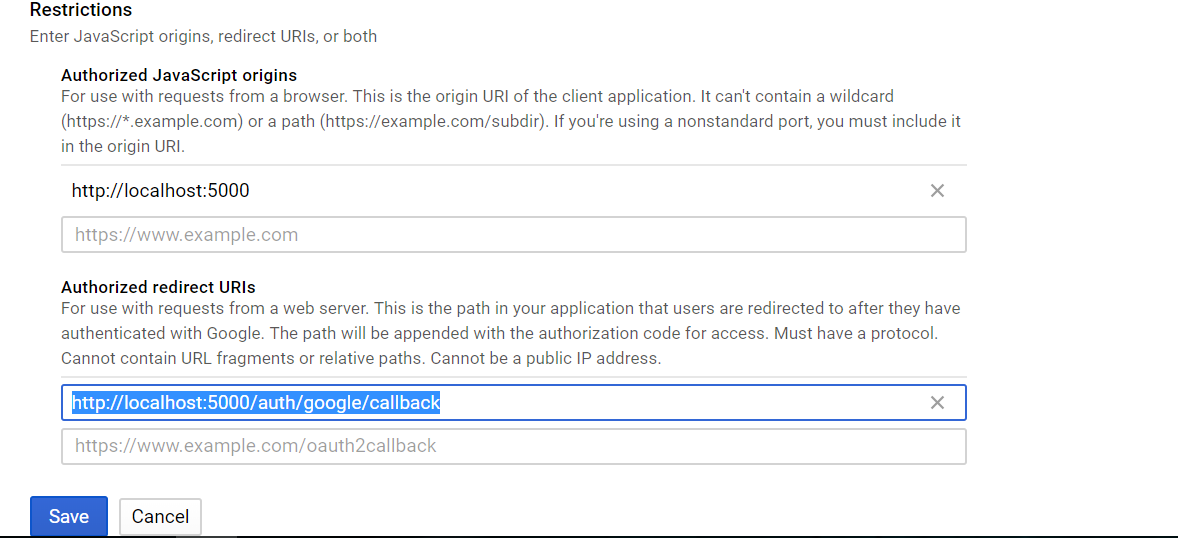
<https://console.developers.google.com/apis/credentials/oauthclient/756906703453-7ci659av9erldafant9nfnkkcaqm88cn.apps.googleusercontent.com?project=emaily-dev-204600>





This URI does not match our call back URI inside node index.js.

### Changed it to our call back URI

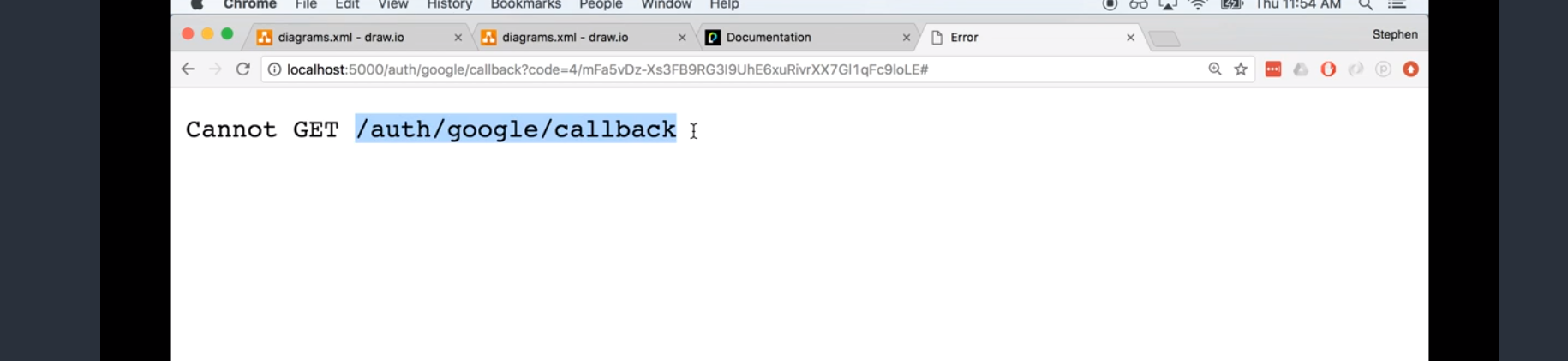


### Test again



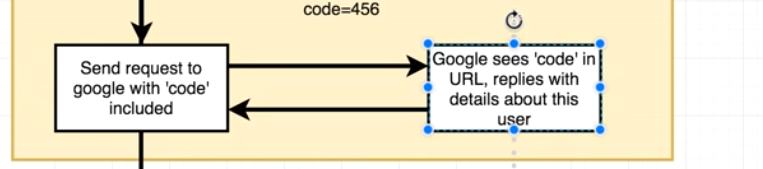
## Server route handler to handle call back

Google already send back to our server, but we need to define route to handle this.

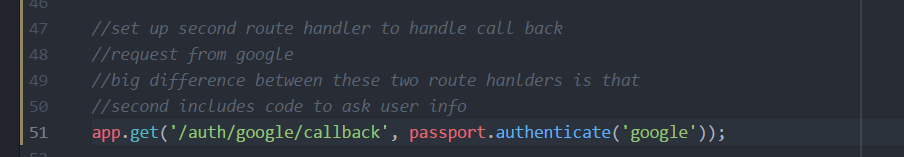


<http://localhost:5000/auth/google/callback?code=4/AAB85y44b-bhWyu4hi1tVQkpIR3oY-1_G0vETRFWxHUtRqSADSPh3sLe_LZx_OQmsoJhfm47GyGAJPB-8Yhe898#>

inside the URL, we can see a code parameter. We can use it later to get user details.



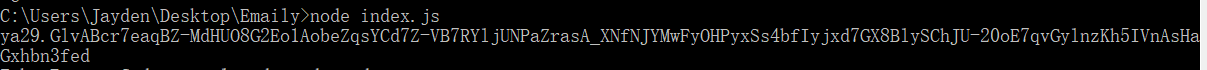
## Add call back handler



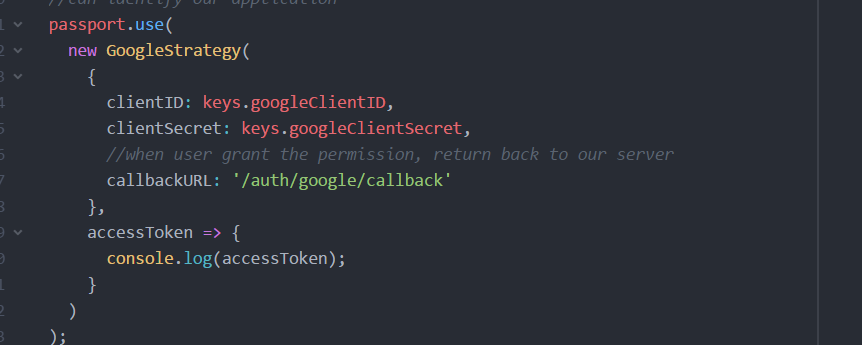
### Test it again

Your page will stuck in because there is no logic to deal with how we are going to use this user info.

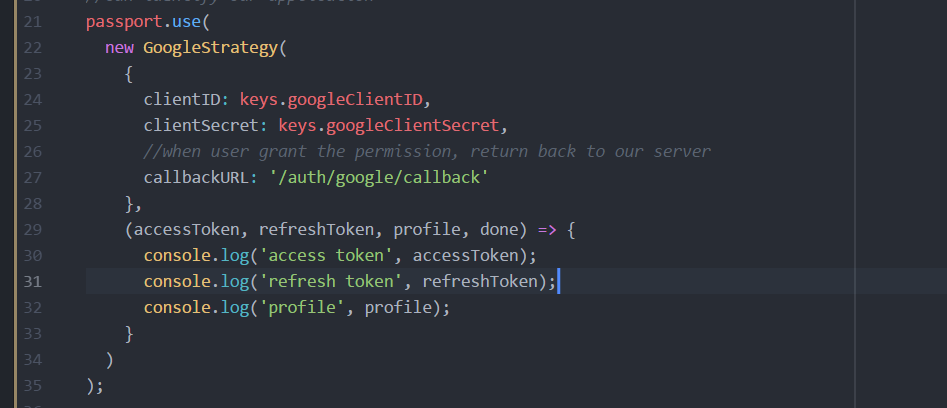
And we can also see the return string in node js server.



Come from the following code.



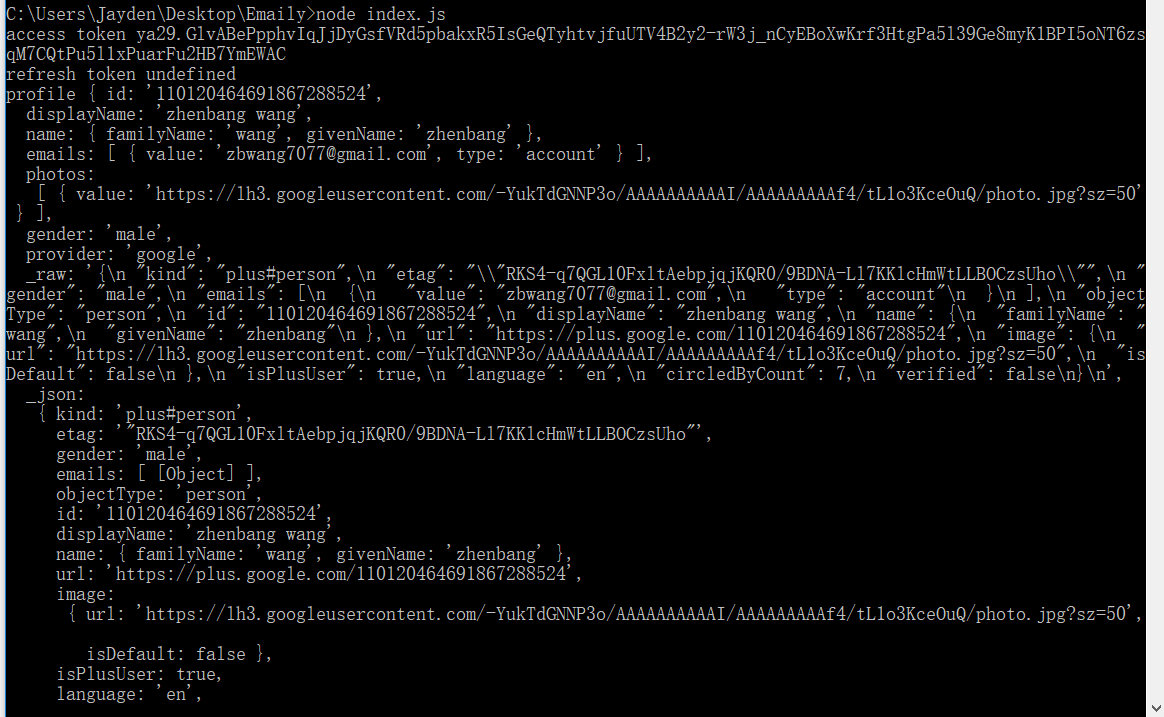
## Log out user info:



### AccessToken

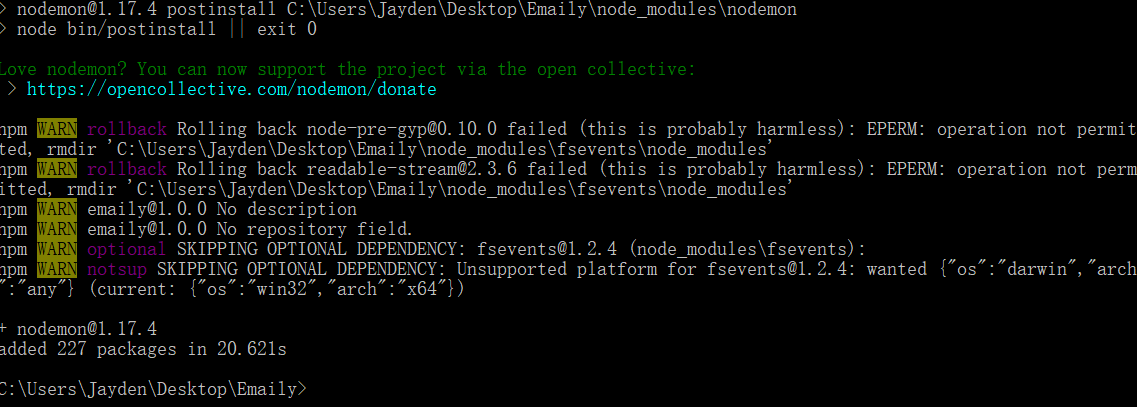
The accessToken was used to access user profile later. We can send request to google to ask for example user contact list using this accessToken.

AccessToken will be expired after a few minutes. So that we need refresh token.

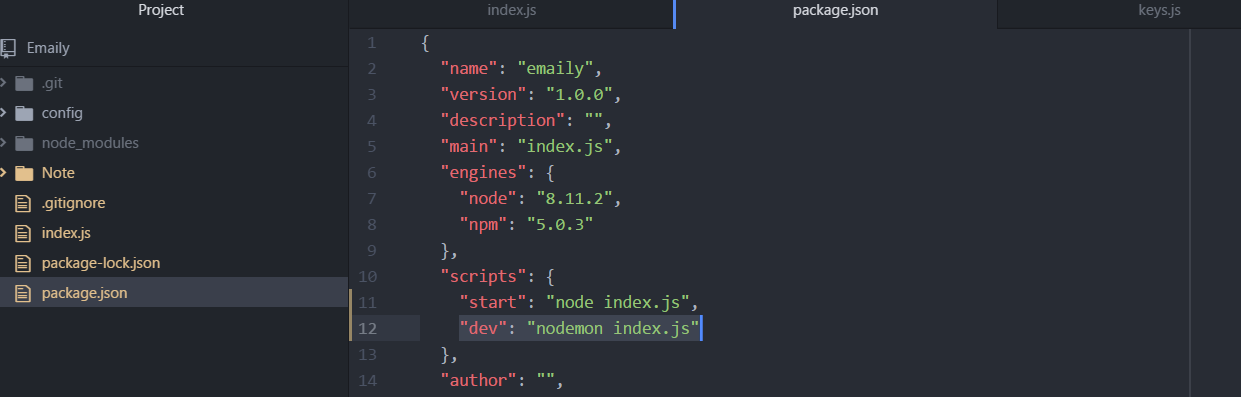


## Automatically catch code changed

### Install nodemon



Add one line inside package.json



This is easy for human to remember how to run this application in dev environment.

### This is the way to start the application.



### You can see node automatically restart when I made some changes

