Search engine for indoor environment data using ElasticSearch and front end search UI using React.

CS410 FINAL PROJECT

NETID: NADIAW2

Table of Contents

Abstract

In the age of sensors, devices and platforms collecting millions of datapoints every second, it comes necessary to be able to sift through all the data to develop insights efficiently Big data offers the solution for analyzing large amount of data and using the technique of Elasticsearch, access to data can be made faster.[1]

I will be creating a web application to use ElasticSearch to search content from a set of documents of environmental datapoints collected by sensors (indoor air, humidity, temp etc). Currently, it is difficult to search for data in a RDMS database and it takes significant time using traditional SQL queries. The project will take the data transfer it to ElasticSearch server. The front end written in React will allow users to search for data. Measurable outcomes are going to be the amount of time it takes to run a query against a traditional RDMS database vs. using ElasticSearch. The planned architecture is shown in Figure1.

Demo App

https://cs410-env-search-app.uc.r.appspot.com/?size=n_20_n

[1] Gujarat, India, Darshita Kalyani, and Dr. Devarshi Mehta, "Paper on Searching and Indexing Using Elasticsearch," *International Journal Of Engineering And Computer Science*, June 30, 2017, https://doi.org/10.18535/ijecs/v6i6.45.

Technical Architecture

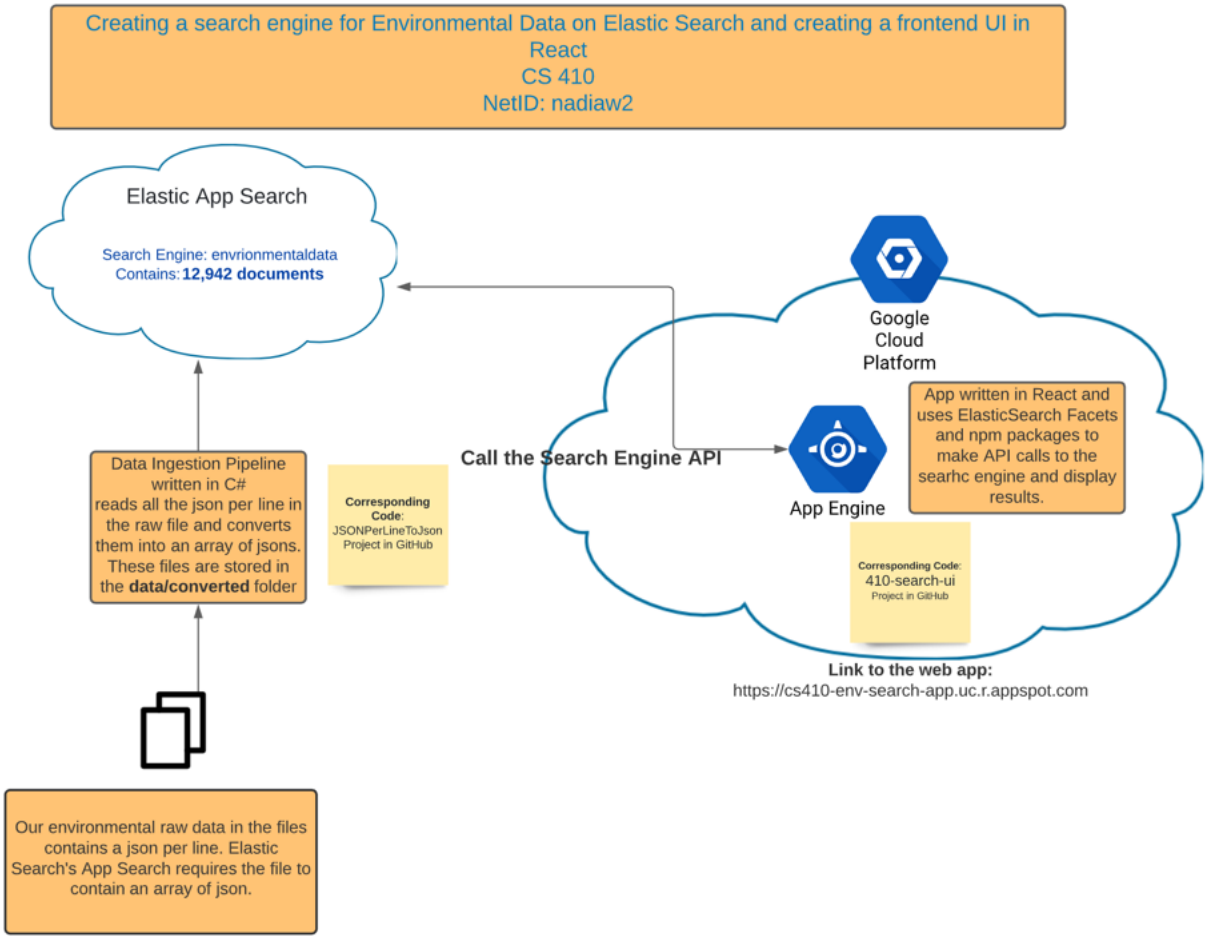
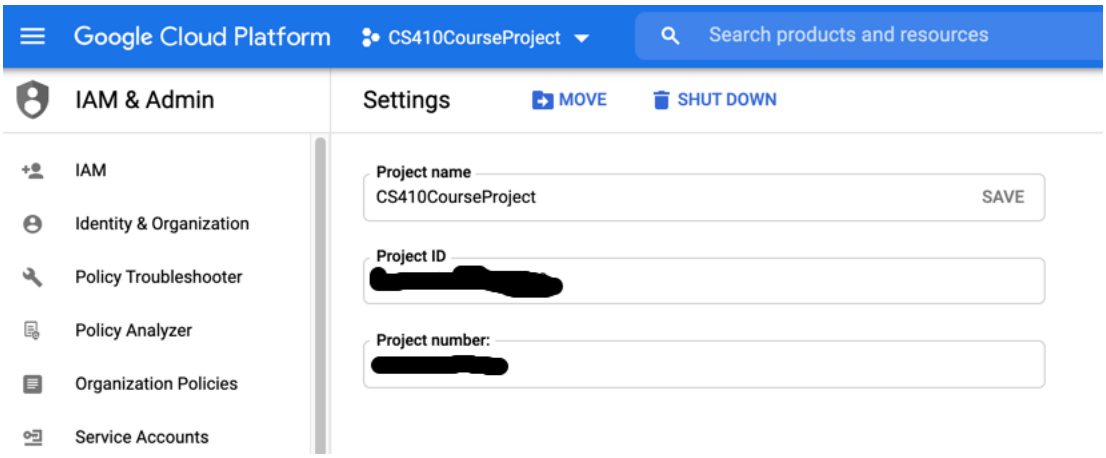


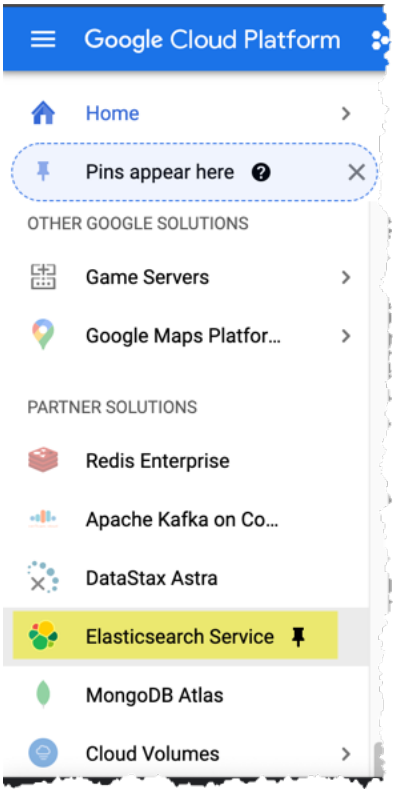
Figure 1

Setting up Elastic Search in Google Cloud.

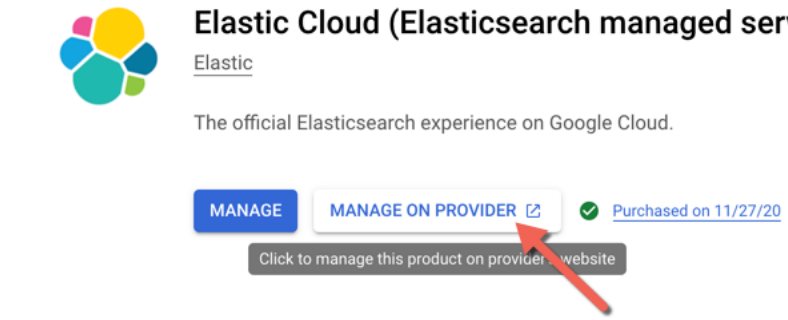
1. Go to https://console.cloud.google.com/ and create a project on Google Cloud.

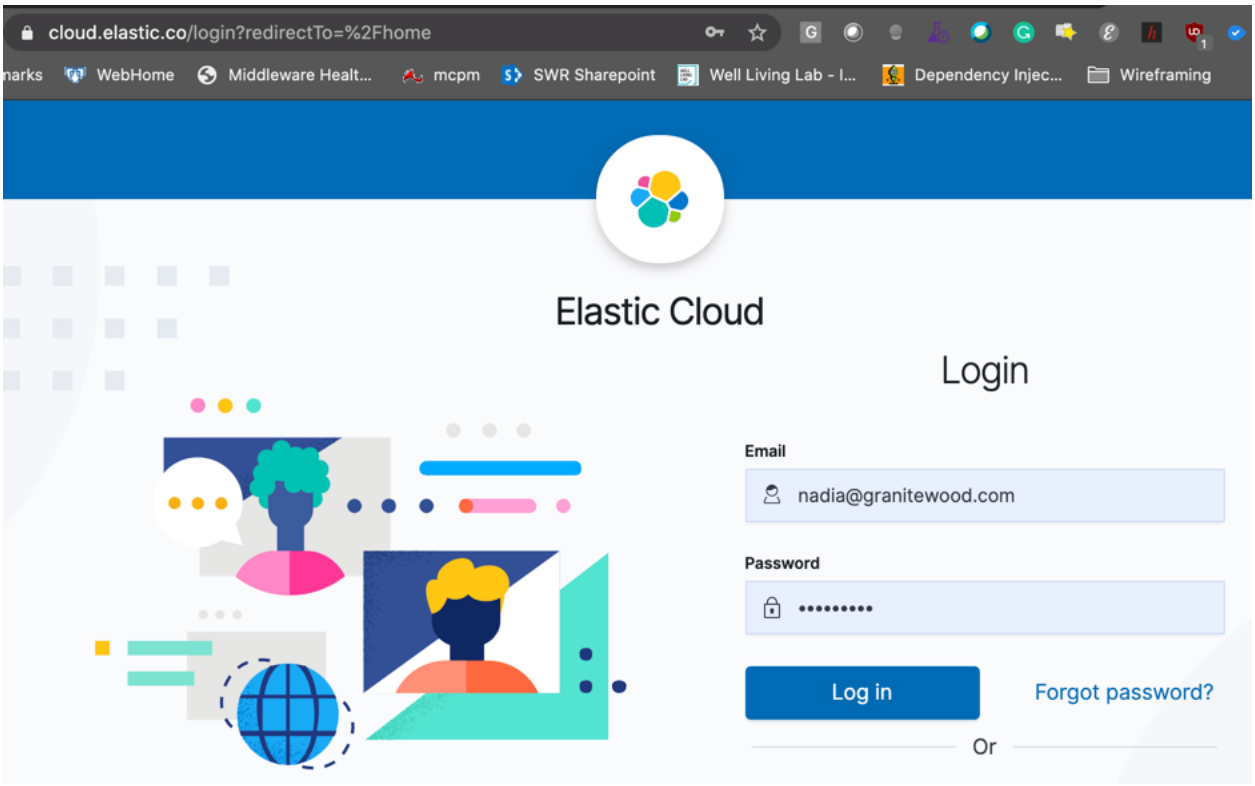


2. Choose Elasticsearch Service. The only reason I chose this is to get some free credit to do my project work. You can create a separate account on Elasticsearch if you want to but the trial only lasts for 14 days.
3. A little bit about, Elasticsearch Service on Google Cloud: The service offers seamless integrated billing through your Google Cloud account for simple management and powerful customization.

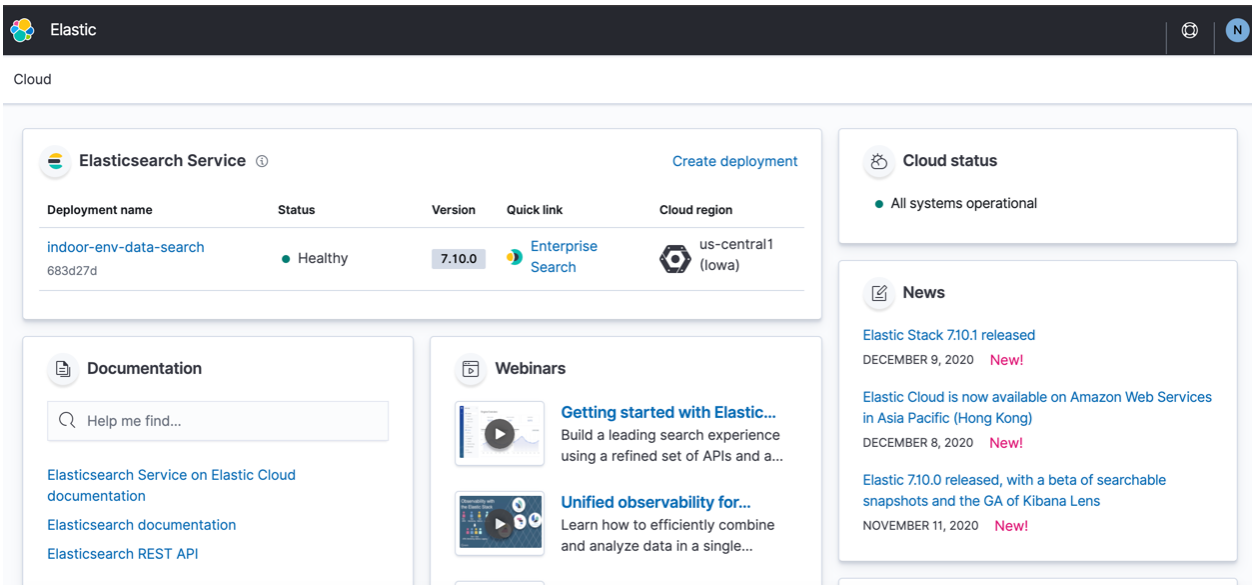


4. Once the service is setup, you can click on manage on provider, to go directly to Elastic Cloud, to manage and create a search deployment on the cloud. https://cloud.elastic.co/

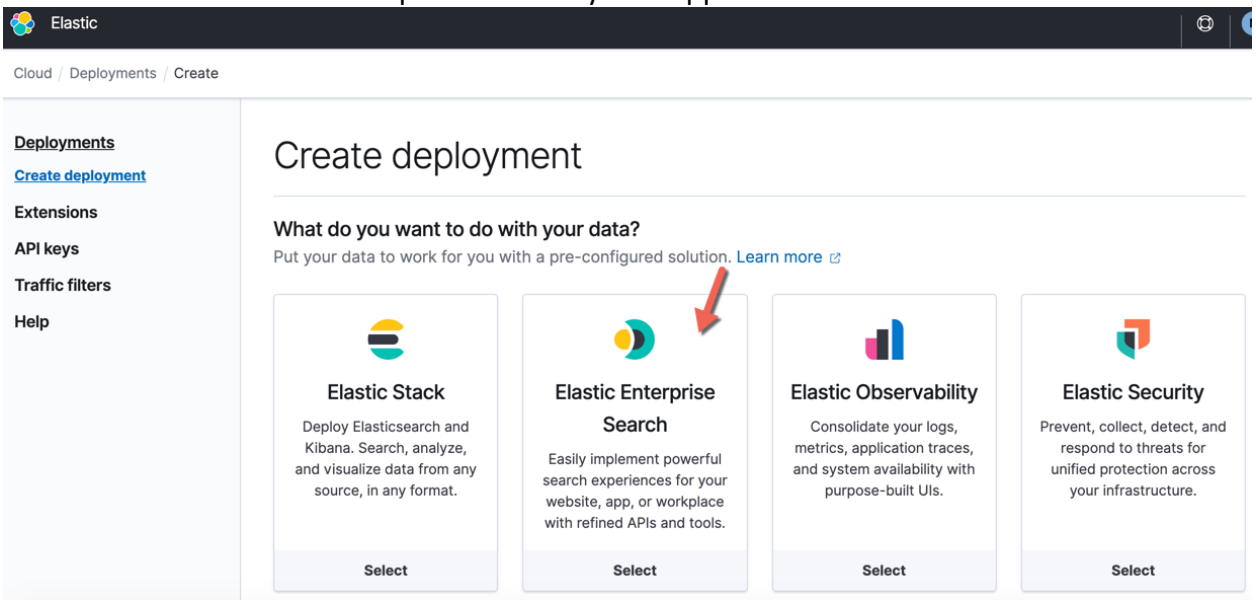




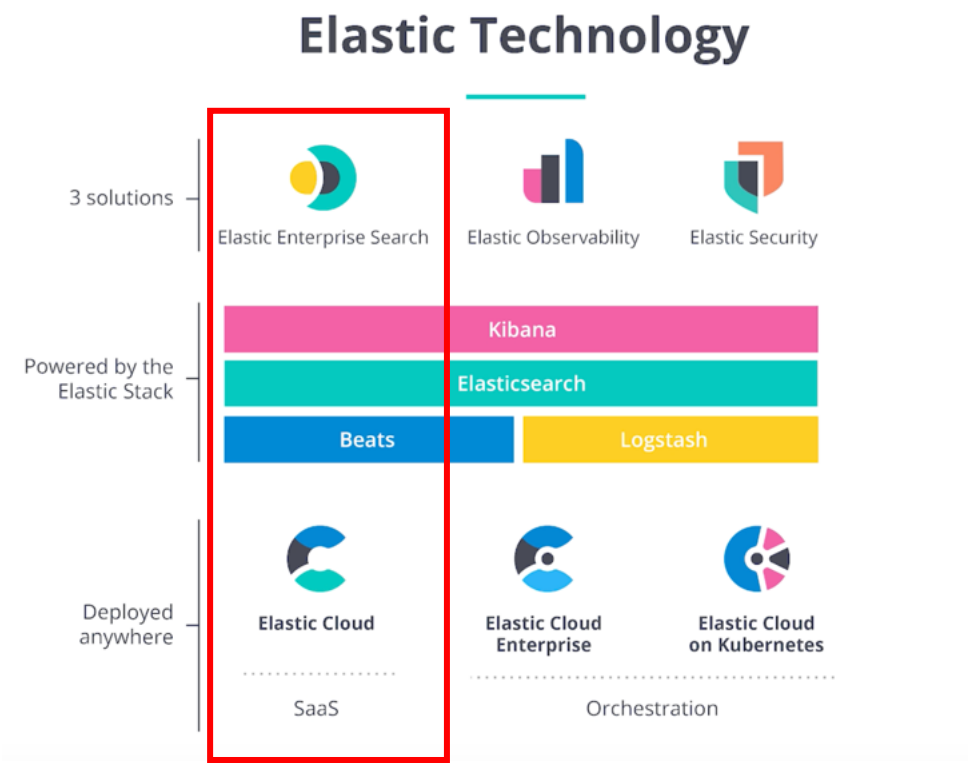
5. Once you login, you will be taken to the Elastic Cloud dashboard. Here you can create your “deployments” .



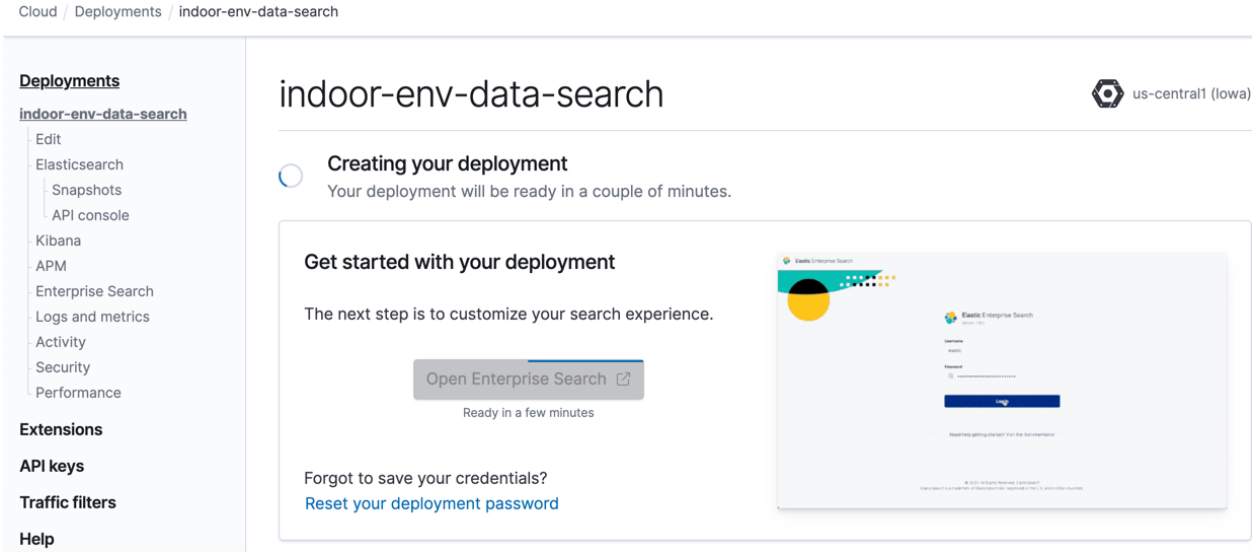
6. When you create a deployment, you are given a choice of selecting from pre-configured environments for your need. In my case, I chose the Elastic Enterprise Search solution to allow me to create a search experience on my web app.

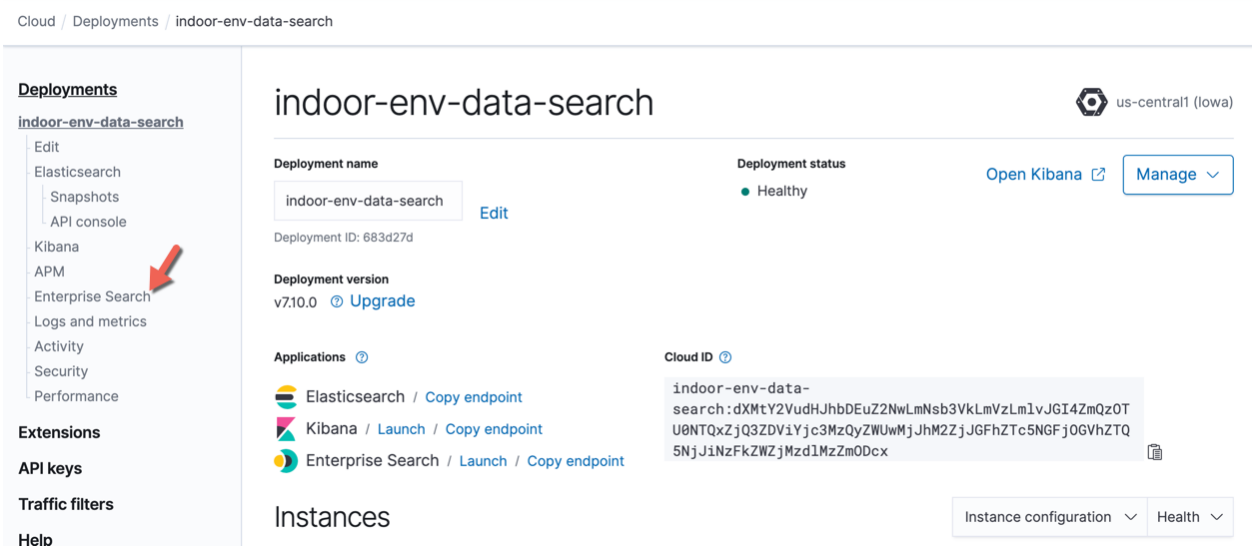


Elastic technology provides the following stack options. For my project I am using the stack outlines in red. This stack gives me the Elastic Cloud, which gives me the ability to make RESTFUL API calls to my search engine.

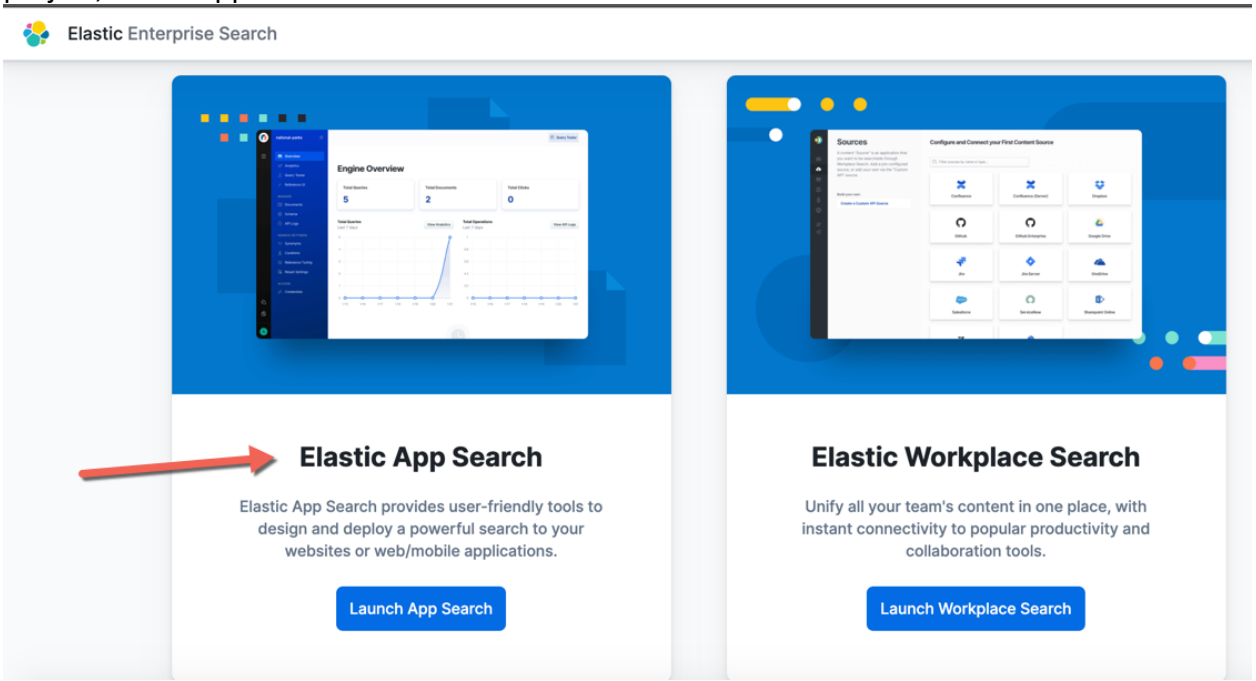


7. Once your deployment is created you will be taken to the deployment dashboard. In this project, we will be focusing on using Enterprise search capability.

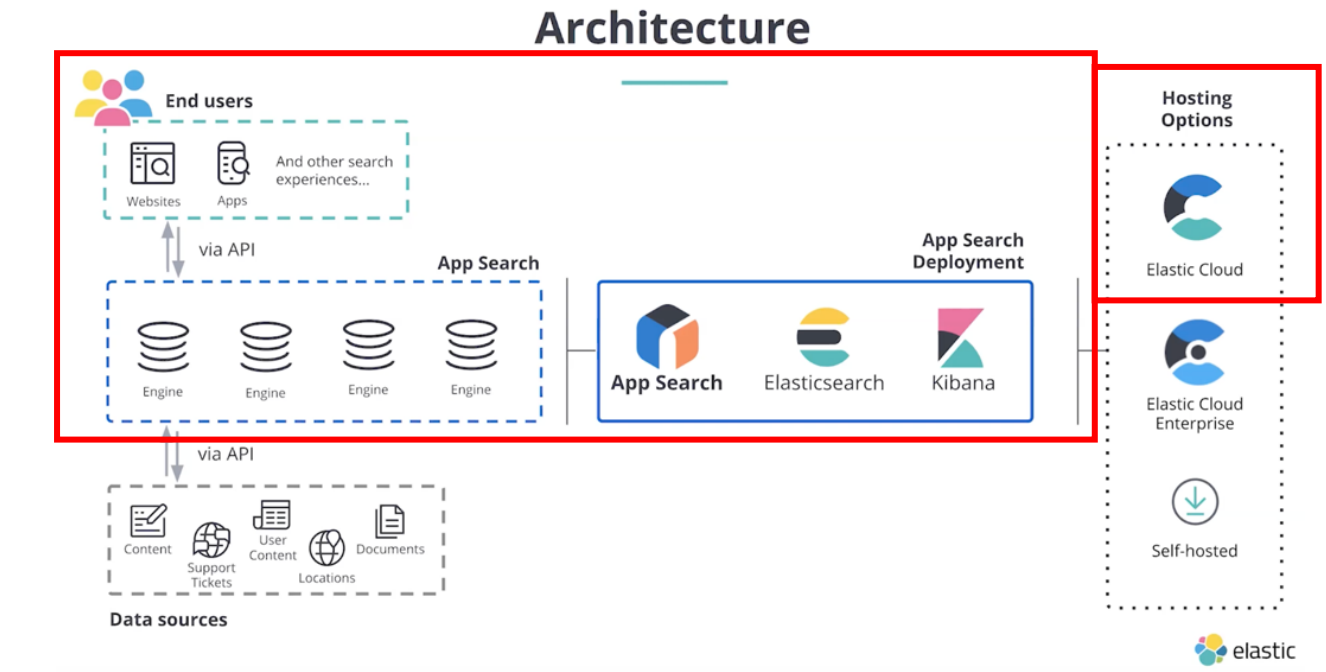


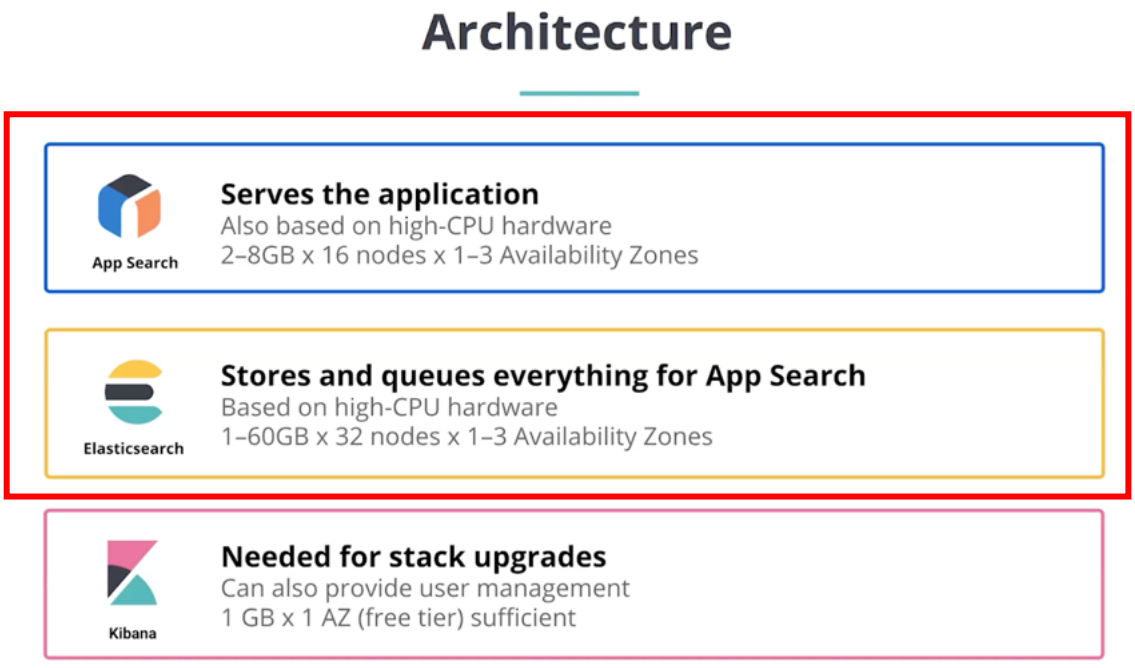


8. Once you launch Enterprise Search, it will give you an option to select a product. For this project, I used App Search.

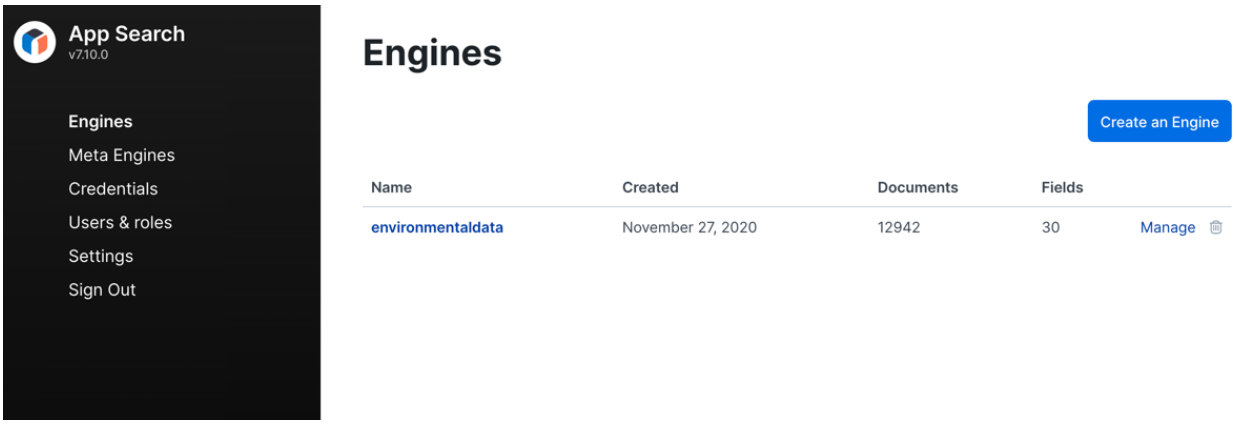


9. An overview of the architecture of App Search stack is below. I have highlighted the architectural components which are being utilized for this project.

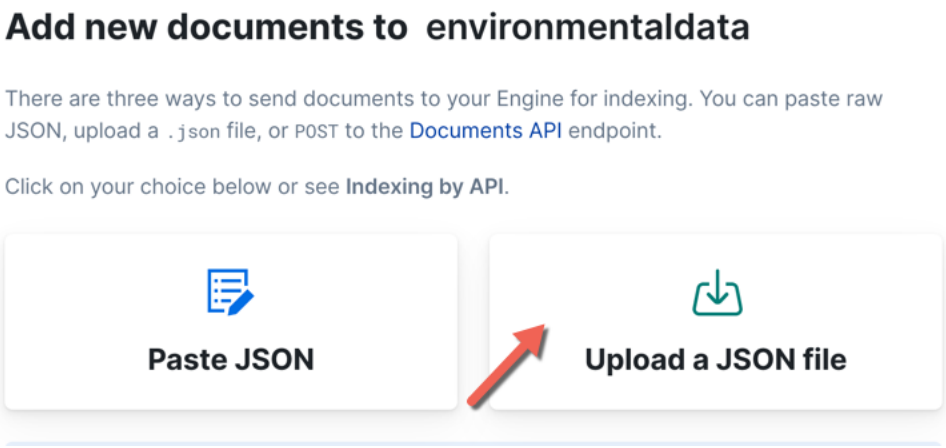




10. Once the App Search is launch, it gives you an option to create a search engine.



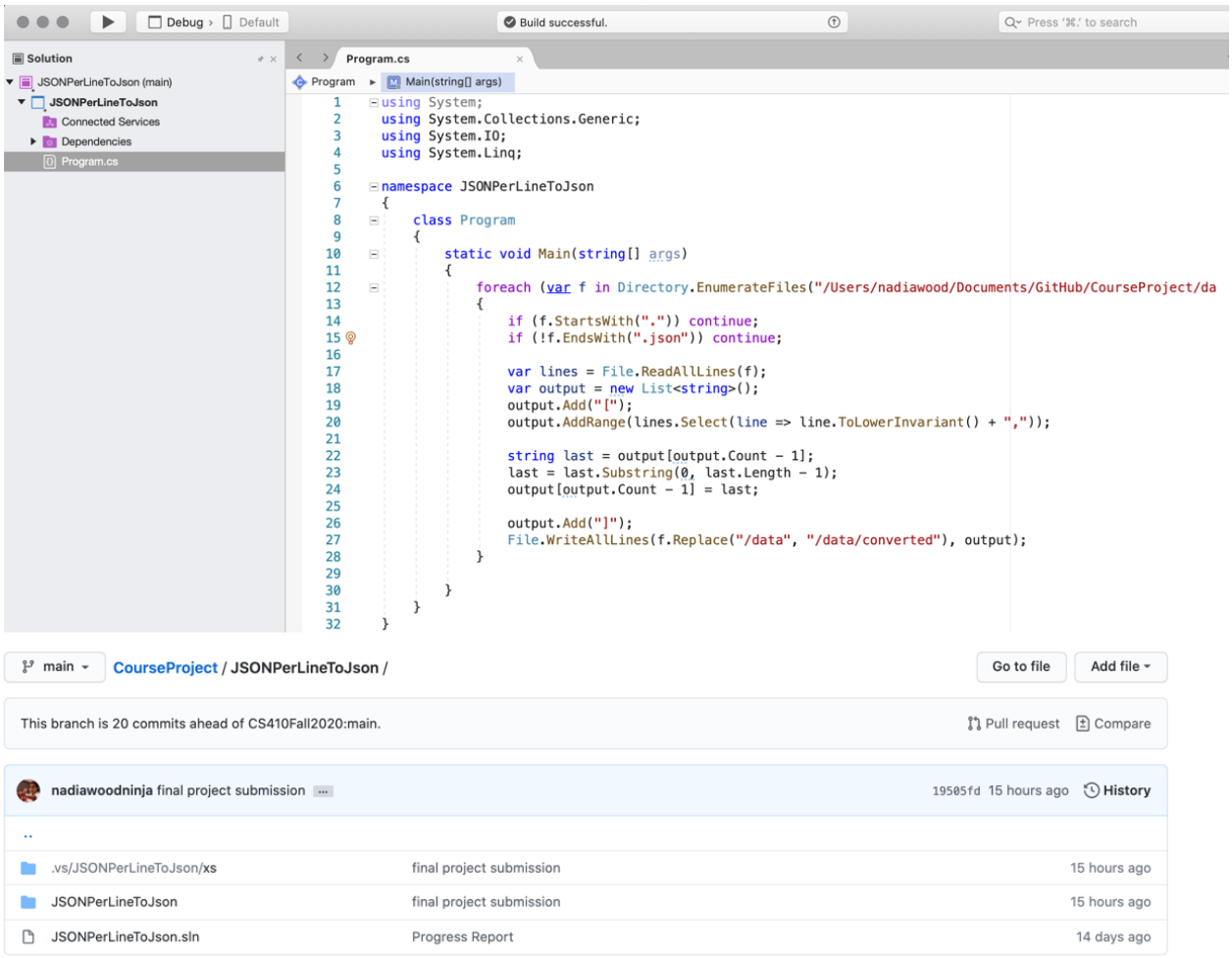
11. For this project, I created an engine called environmentaldata. This search engine allows a user to search through documents which contain sensor data e.g. humidity, temperature, light, battery info etc. Currently the engine contains xxxx documents. The documents were loaded into the engine by uploading JSON file to the engine.



12. The engine requires a specific formatting for the json files to adhere to. The JSON field names have to be all lowercase or be separated by underscore. This created a need to automate the conversion of existing json files to be converted to the required format.

Data Pipeline

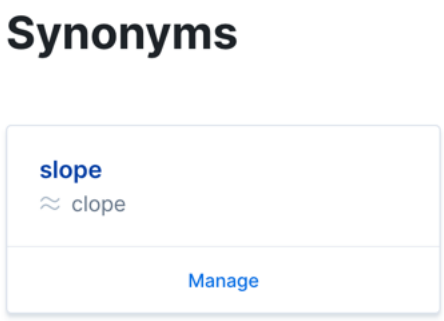
13. **Data Ingestion program in C#**: In order to quickly load json files to the engine, I created a program in C# to convert existing files to a proper json file so that it can be imported into App Search. This code can be run if you have Visual Studio free community version installed. The program requires to have a “data” folder where the files needed to be converted need to stored. The converted files are stored in the “data/converted” folder. I have included some converted file in the repo as well: https://github.com/nadiawoodninja/CourseProject/tree/main/data/converted



Fine Tuning the Engine

14. Once the documents are loaded into then engine, you can index any JSON object. The json object will become a search-optimized document within your Engine. A schema is created for you when you index your data - you do not need to specify any schema or alter your data before uploading. You can alter your schema later to set the appropriate data types.

15. You also have the option to refine search by using features like, Relevance Tuning, Synonyms & Curations. For this project I utilized the Synonym feature as we may have data from different sensors and the same datapoint maybe spelt differently or represented differently.

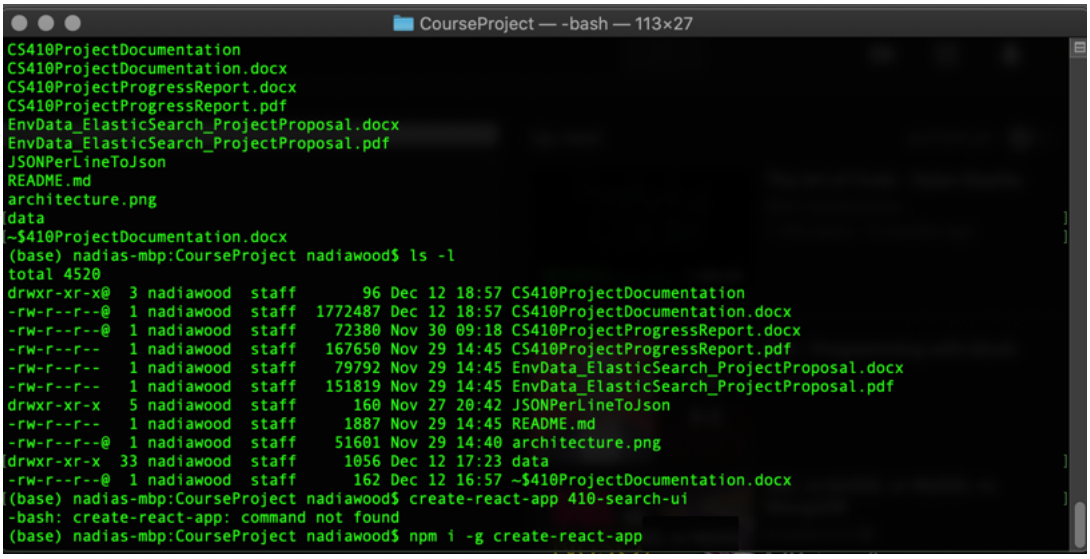


Creating a search UI to search data

Setting a development Environment locally on your computer.

16. Download and install Node.js from https://nodejs.org/en/
17. Once installation is complete run this command. We are going to use this to create a react app.

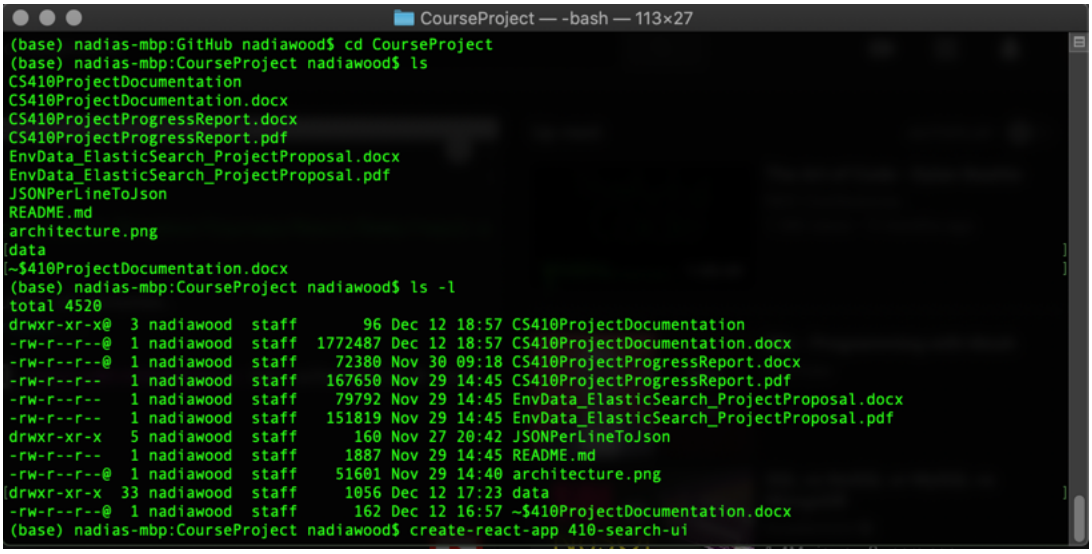
```
npm i -g create-react-app
```



18. Once the package is installed create the react app by running the command below.

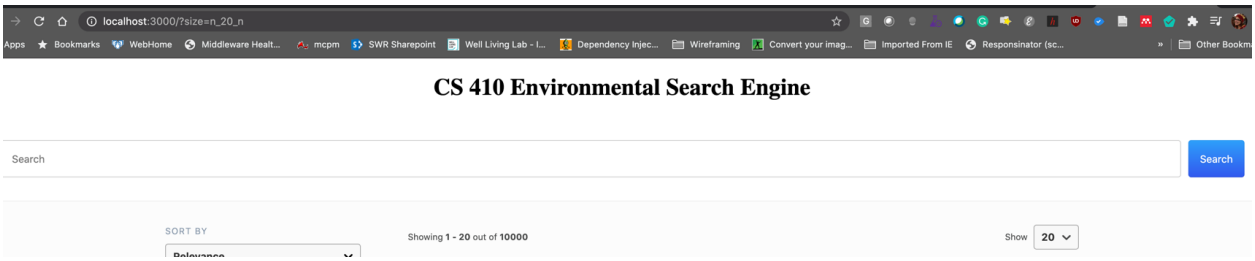
```
create-react-app 410-search-ui
```

This command installs a light weight web server, webpack to bundle our code for deployment and Babel for compiling our JavaScript code.



Once the app is created go to folder 410-search-ui and run this command. This will launch our development server on localhost:3000

```
npm start
```



Creating a UI for search experience by using App Search packages

19. Install **React Search UI** and the **App Search** connector by running these commands

```
npm install --save @elastic/react-search-ui @elastic/search-ui-app-search-connector
```

Creating a search experience

20. I use Atom as my editor for React apps. The app folder contains **src** folder which contains all the source code. **App.js** is the main file where the program starts execution.

21. The src folder also has a config folder which contains engine.json. This file contains all the configuration needed to configure your search UI. In this file you can define your “facets”, the fields which will be displayed on your results page, your sort fields etc.

```
{
  "engineName": "environmentaldata",
  "endpointBase": "https://0189dc168aee4f4c83d3371d52e6b812.ent-search.us-central1.gcp.cloud.es.io",
  "searchKey": "search-3vkosxzxxi64t1jg9gcd9ums",
  "resultFields": [
    "batterycurrentvoltage",
    "coord_phi",
    "coord_theta",
    "dataunits",
    "batteryminimumvoltage",
    "studyname",
    "datatype",
    "datavalue",
    "studyid",
    "vendordata",
    "batterymaximumvoltage",
    "coord_x",
    "coord_y",
    "slope",
    "eventtimestamp",
    "datavaluecalibratedsi",
    "wllid",
    "datavaluecalibratedq",
    "coord_z",
    "dataid",
    "intercept",
    "datasource",
    "location",
    "dataconnection",
    "quadratic_c",
    "placement",
    "quadratic_b",
    "quadratic_a",
    "clope",
    "id"
  ],
  "sortFields": [

```


Figure 2: engine.json

The ability to define these configurations are provided by the packages which were installed above.

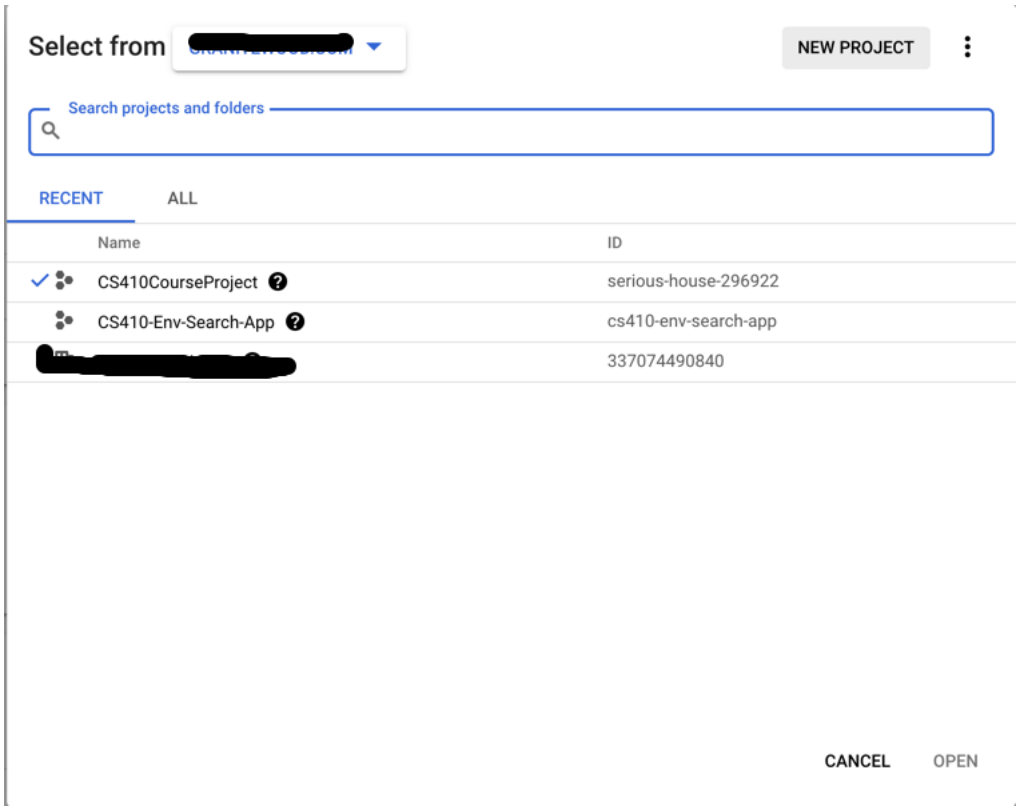
```
//all your imports for the search experience
import React from "react";
import AppSearchAPIConnector from "@elastic/search-ui-app-search-connector";

//import components needed for the Results Page.
import {
  ErrorBoundary,
  Facet,
  SearchProvider,
  SearchBox,
  Results,
  PagingInfo,
  ResultsPerPage,
  Paging,
  Sorting,
  WithSearch
} from "@elastic/react-search-ui";
import { Layout } from "@elastic/react-search-ui-views";

//Import the default css
import "@elastic/react-search-ui-views/lib/styles/styles.css";

//Import any configuration helpers
import {
  buildAutocompleteQueryConfig,
  buildFacetConfigFromConfig,
  buildSearchOptionsFromConfig,
  buildSortOptionsFromConfig,
  getConfig,
  getFacetFields
} from "./config/config-helper";
```

Figure 3: App.js

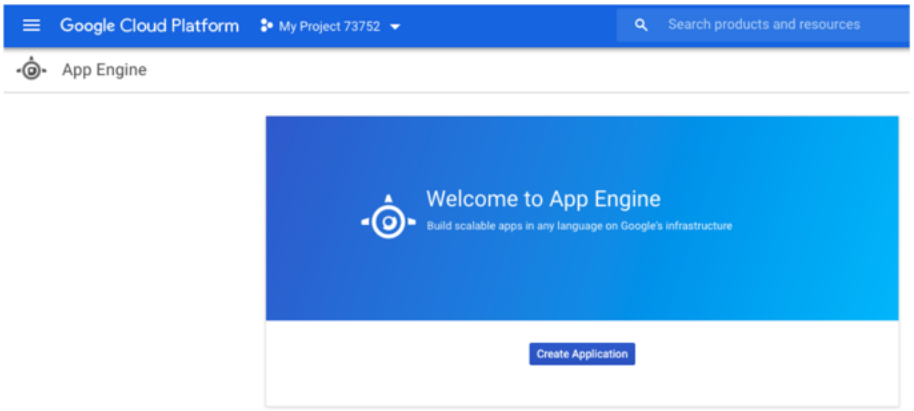
Deploying the React app to Google Cloud Platform

Create the app on the App Engine

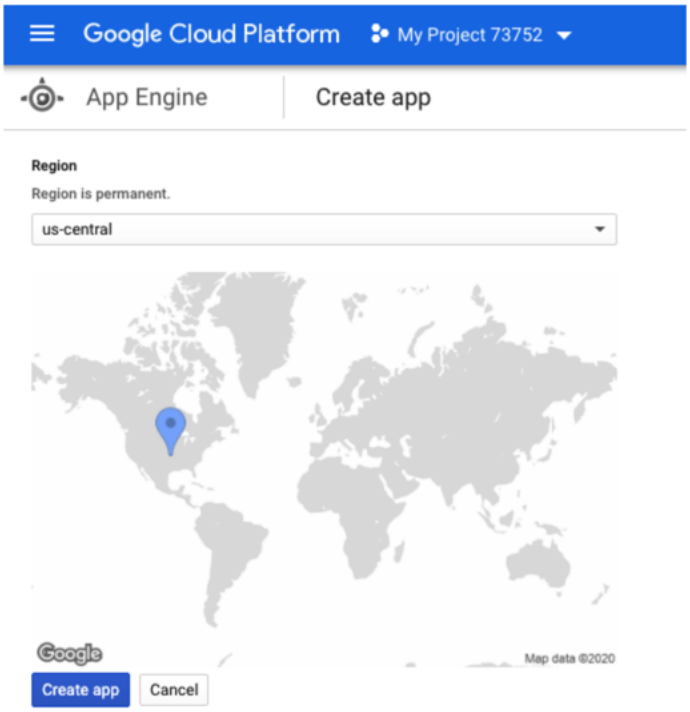
22. Go to [Google's App Engine Console] and create a new project:



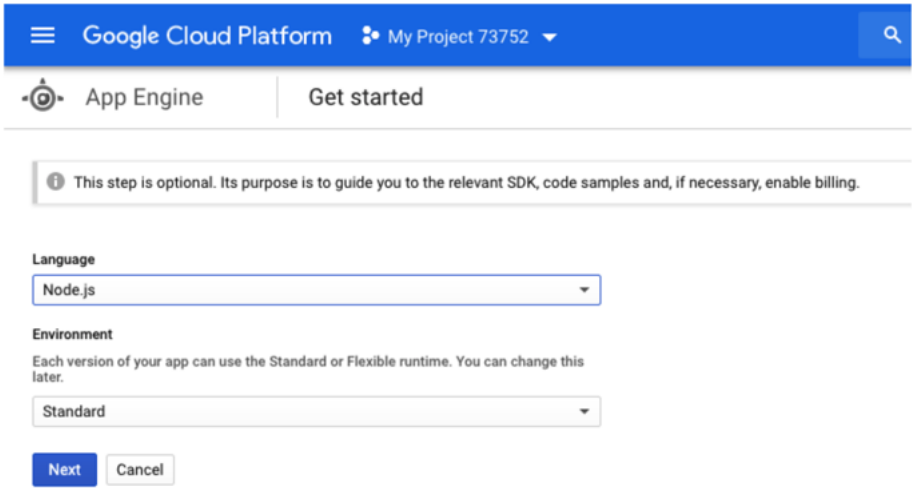
23. Once the project is created, create an App Engine application.



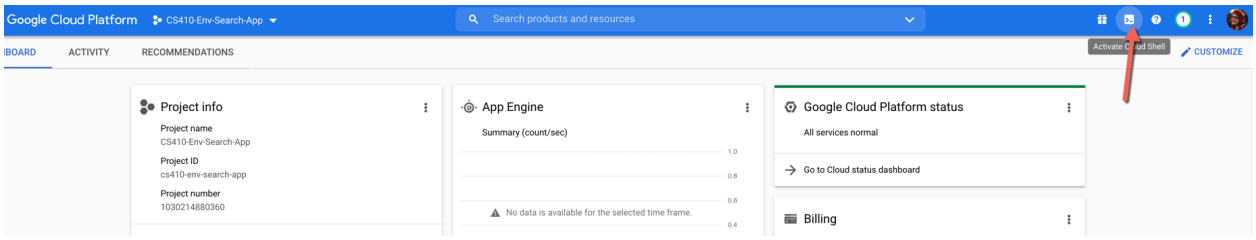
24. Select a region



25. Select Node.js and standard environment



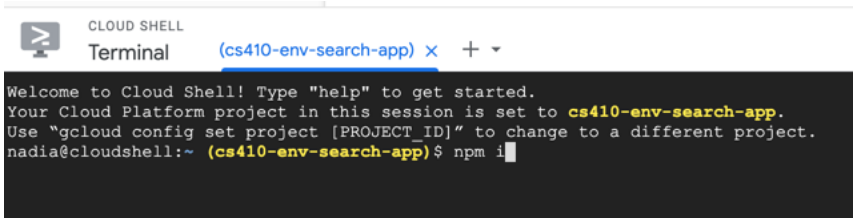
26. Clone our app's source code from GitHub
27. Activate the shell by clicking

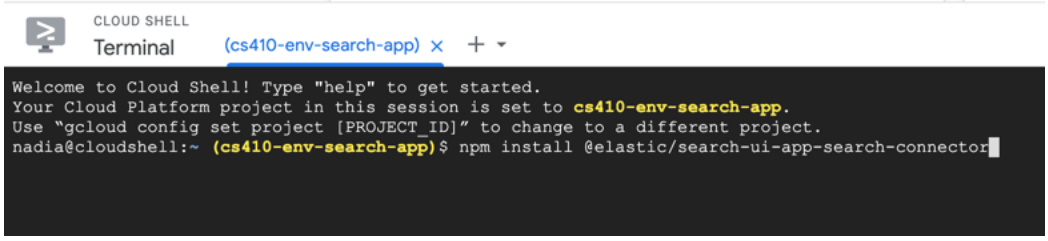


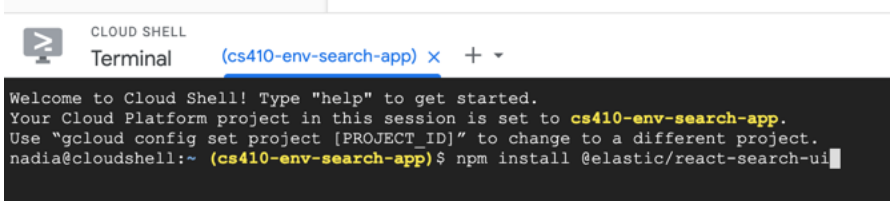
`git clone https://github.com/nadiawoodninja/CourseProject.git`

28. Install npm by running and install other elastic search packages

```
npm i
npm install @elastic/search-ui-app-search-connector
npm install @elastic/react-search-ui
```







29. Build our app for deployment

To do this, simply go into your app's root folder (where your "src" folder is),

```
cd CourseProject
cd 410-search-ui
```

And type the following command:

```
npm i
npm run build
```

This creates a folder named "build" in our root directory.

30. Delete every thing else besides the **build** folder. Get rid of everything else, except for the build folder.

Use these commands to remove files and folders

```
rm <file-to-remove>
rm -r <remove-recursively-like-directories-inside-directories>
```

31. Add an app.yaml and deploy

In the same folder where we have our "build" folder, create a new file named app.yaml. By the end of this step, the only things left should be the "build" folder and "app.yaml". That's all the App Engine will need to run our app.

```
touch app.yaml
nano app.yaml
```

And add the following to its content:

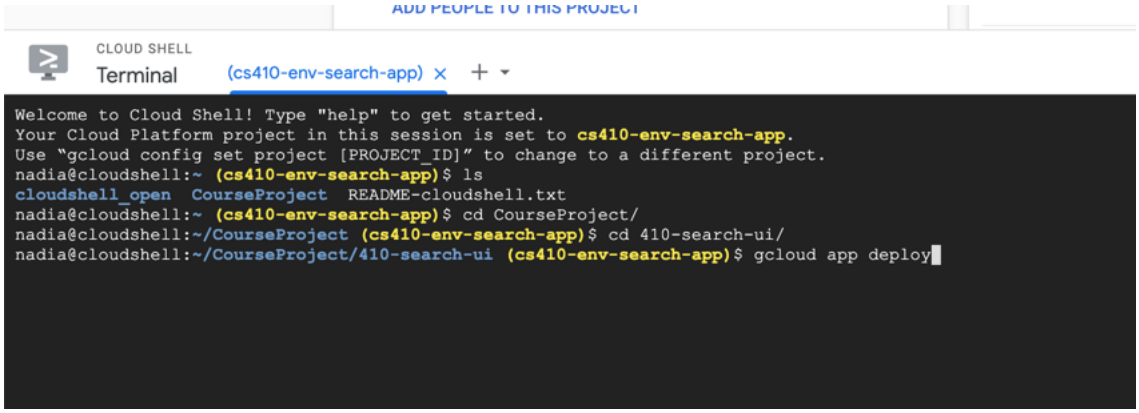
```
runtime: nodejs12
handlers:
# Serve all static files with url ending with a file extension
- url: /(.*\..+)$
  static_files: build/\1
  upload: build/(.*\..+)$
# Catch all handler to index.html
- url: /.*
  static_files: build/index.html
  upload: build/index.html
```

Deploy the app

32. Deploy the app using the following command

```
gcloud app deploy
```

33. The app is running here
https://cs410-env-search-app.uc.r.appspot.com/?size=n_20_n


ADD PEOPLE TO THIS PROJECT
CLOUD SHELL
Terminal
(cs410-env-search-app)
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cs410-env-search-app.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
nadia@cloudshell:~ (cs410-env-search-app)$ ls
cloudshell_open CourseProject README-cloudshell.txt
nadia@cloudshell:~ (cs410-env-search-app)$ cd CourseProject/
nadia@cloudshell:~/CourseProject (cs410-env-search-app)$ cd 410-search-ui/
nadia@cloudshell:~/CourseProject/410-search-ui (cs410-env-search-app)$ gcloud app deploy