CPE Help desk

# user guide - Aravind Karnam Rajendraprasad

Table of Contents

|  |  |
| --- | --- |
| Page no | Topic |
| 1 | TABLE OF CONTENTS |
| 2 | Setup |
| 3 | Block diagram |
| 4 | Source code |
| 5 | API.AI setup |
| 6 | Knowledgebase setup |
| 8 | Cloud hosting |
| 13 | Data Entry |

SETUP:

**Client:**

An embeddable frame provided by api.ai which can be inserted in any web page by adding it to html. It’s the point of contact for all the users

**Natural language processing (NLP):**

In this project this, step is accomplished using Google’s natural language processing engine API.AI which is a free cloud hosted service. Its job is to break down a user’s plain text input into intents and entities.

For example: How is weather in Sacramento tomorrow?

Intent: Weather forecast

Entities: Location -> Sacramento

Time -> tomorrow

Once API.AI figures out the intents and entities of user input, it forward this information to a web hook (server) for response generation. API.AI also supports an additional feature called “small talk”, which is, when user attempts to do every day small talk like “how are you?”, “can you help me” etc., API.AI responds with answers like “I am doing well how are you ” and “sure. I can help you” respectively without calling the web hook.

**Web hook:**

It is a server which hosts both the dash board and a RESTful web service which will create responses to the user’s inputs based on Intents and Entities detected by the Natural language processing engine. For all defined intents, web hook will contact the database with a related query, access the information, construct an answer and give it back to user, and for default intents it will direct the user input to a knowledge base hosted on Microsoft’s QnAmaker service which will in turn respond with the answer to that question. Web hook used for this project is built on a node.js instance.

**Database:**

An in-Server database is used for storing all the information required for response generation. Files stored in JSON format are loaded into server during the start of the process, and parsed into JavaScript objects. An external database could also be used for this purpose.

**Knowledge Base:**

It is a web scraping tool which takes a list of URLs as an input and scrap all question and answer pairs in that web page and use it as a knowledge base to answer questions coming from web hook. Typically, these sorts of systems are used to answer questions that resemble FAQ of an organization.

Here is a block diagram of all the critical infrastructure involved:

Client

query

Web hook

Node.js server

Role: response generation

results

response

response

Knowledge base

ON

Default intent handler

Database

ON

Declared intent handlers

Intents + Entities

Plain text

NLP

(api.ai)

Small talk

query

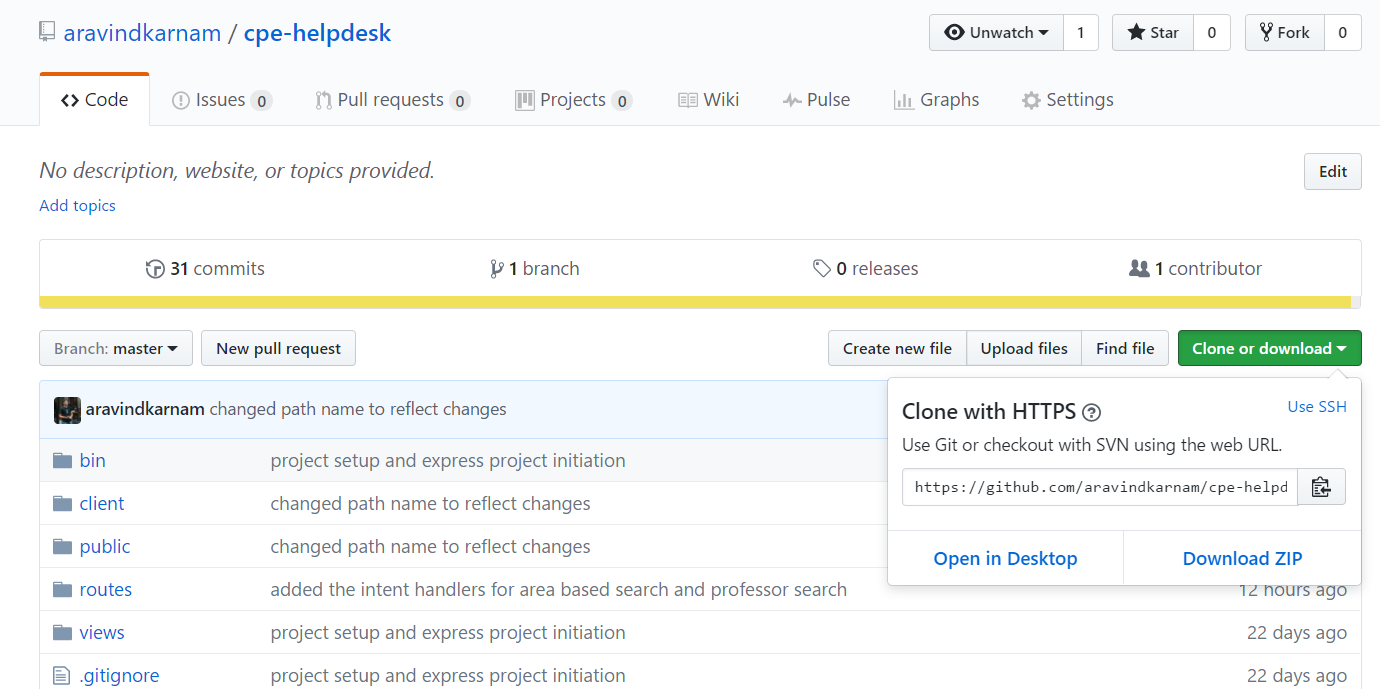
CODE REQUIRED FOR HOSTING THIS BOT:

All the necessary modules for this project are publicly hosted on GitHub in the following repository

<https://github.com/aravindkarnam/cpe_helpdesk_v1.git>

FORKING:

You can obtain a copy of this for cloud hosting through merely forking this repository into your own GitHub account



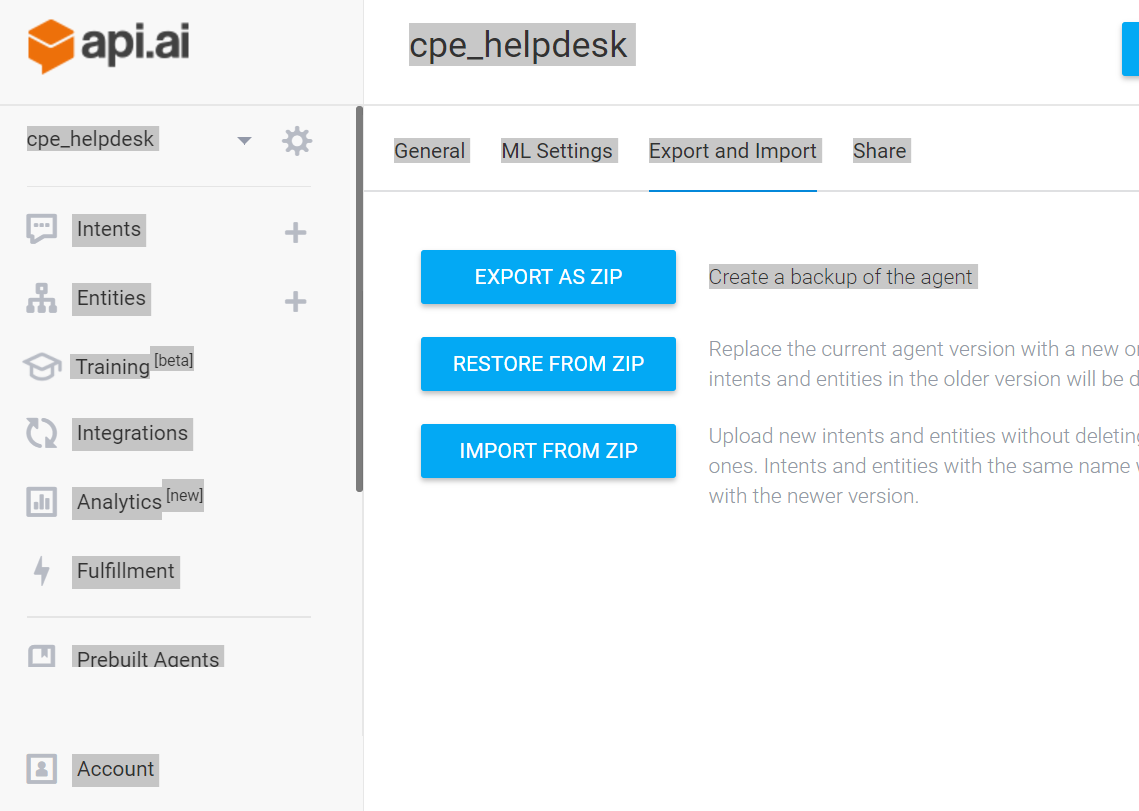
CLONING:

For personal hosting, you can clone this project into your computer by typing the following command into command line tool provided you already have Git version management tool, installed on your PC

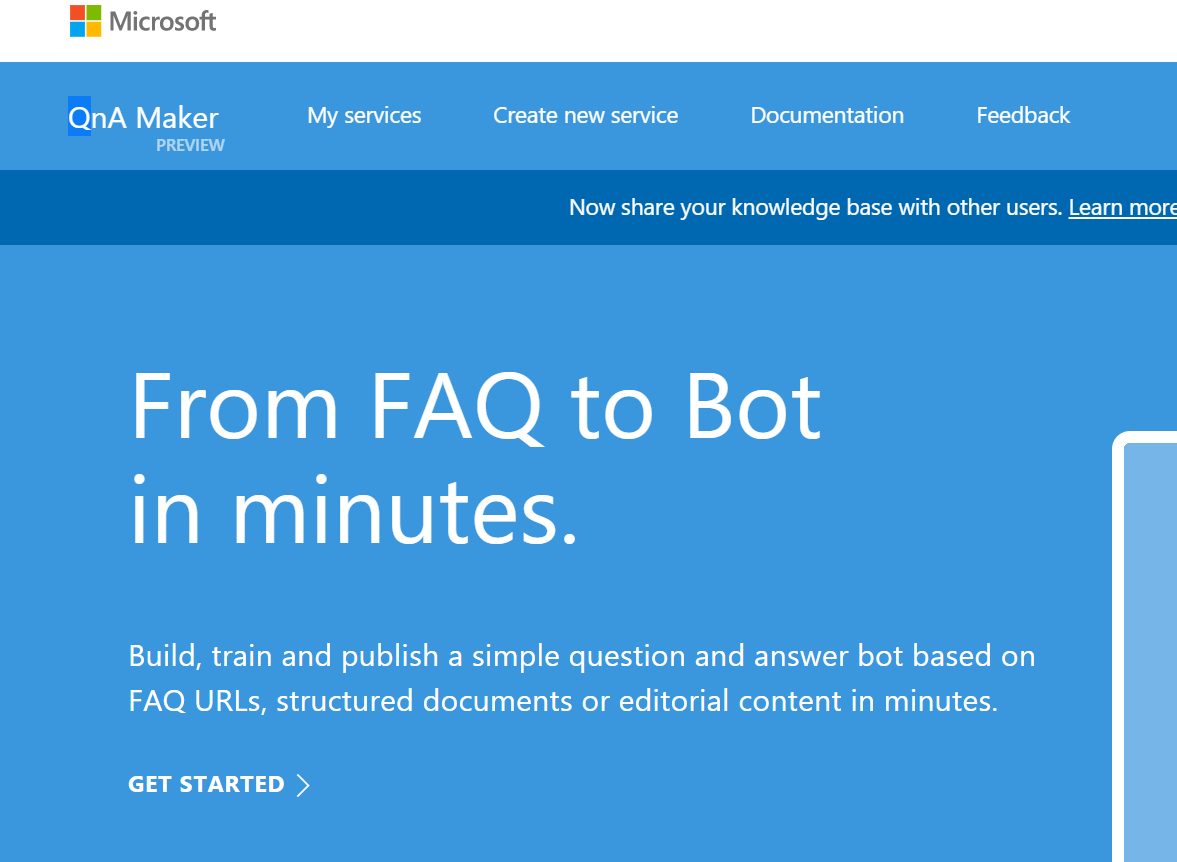
>> git clone <https://github.com/aravindkarnam/cpe_helpdesk_v1.git>

API.AI setup:

API.AI provides a way to export/import projects through a ZIP file. Once you create a new agent click on settings. Use the zip file from repository to deploy your app. You can still edit the app to suit your needs after importing it.

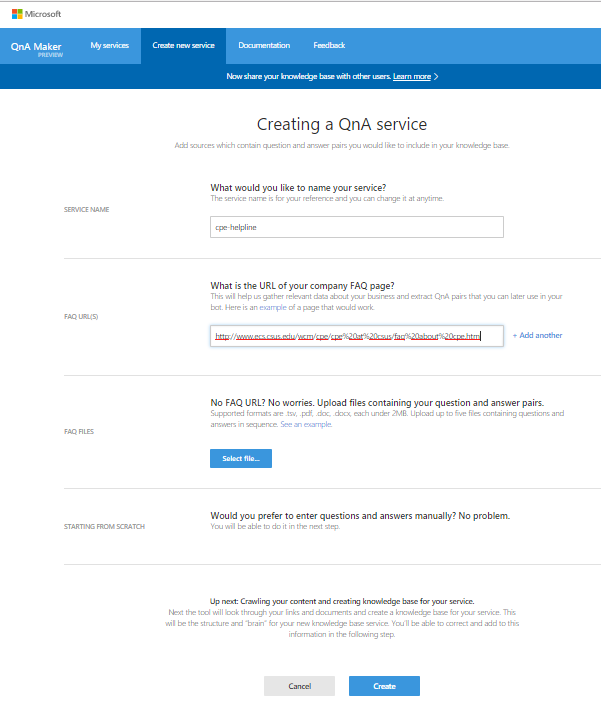


Knowledge base:

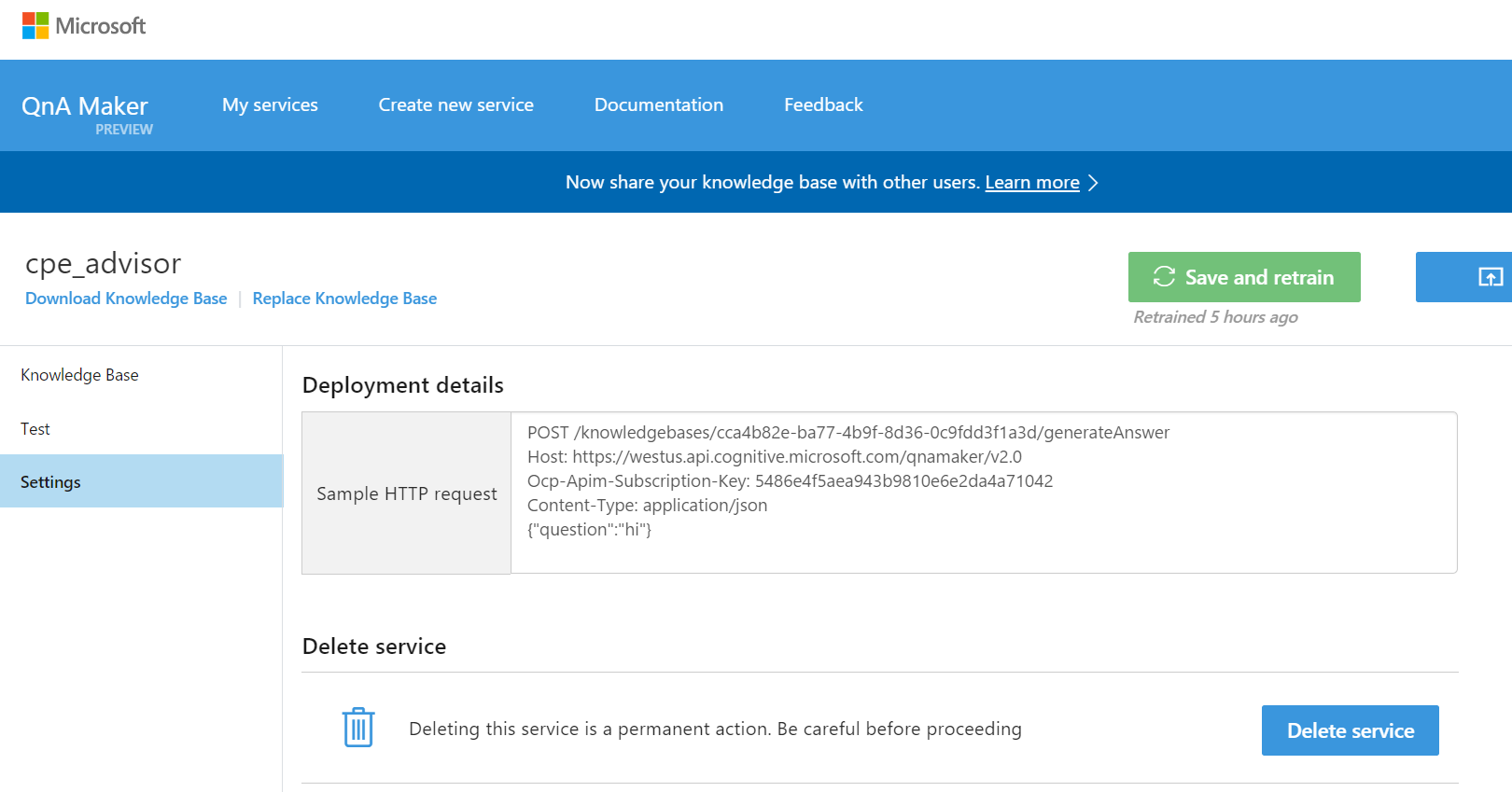
Create a new service in qnamaker.ai after creating your Microsoft account



1. Fill the following fields and create a new service

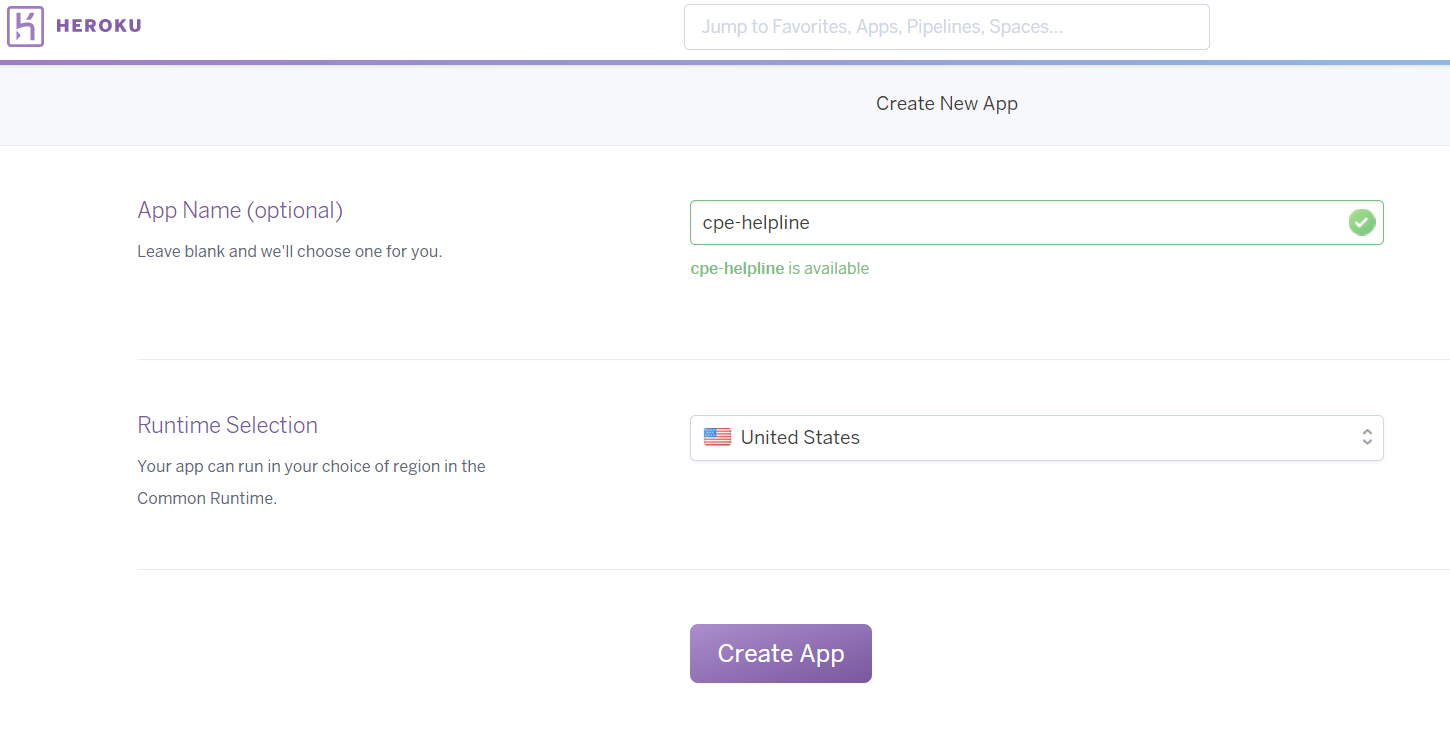


1. Once the new service is created, get the knowledge base Id and access key from the settings tab after scrolling down to the bottom of the page. Store this values to be passed in as variables while setting up server.

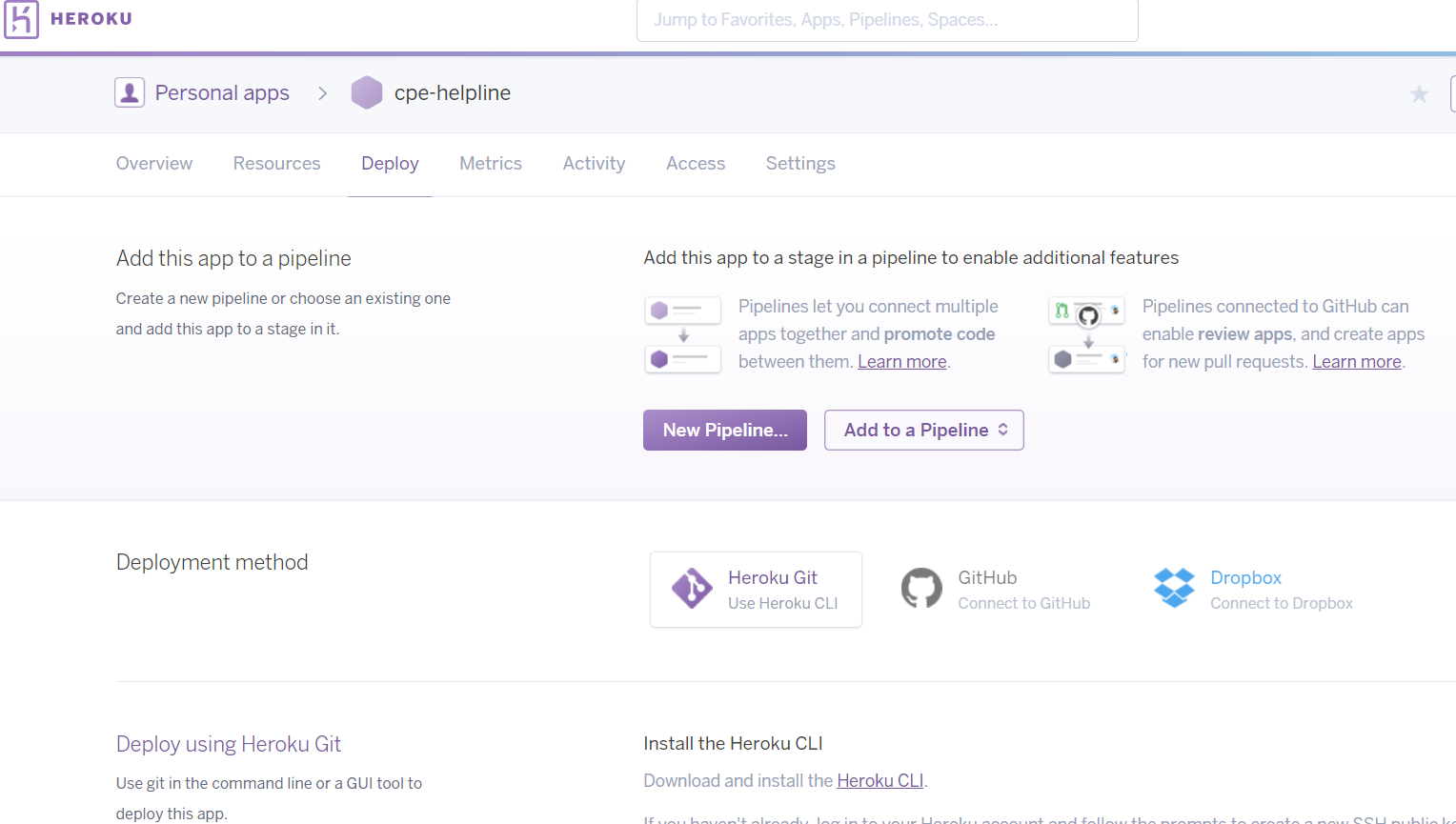


**Cloud Hosting:**

1. Create an new app and give it a name you like.

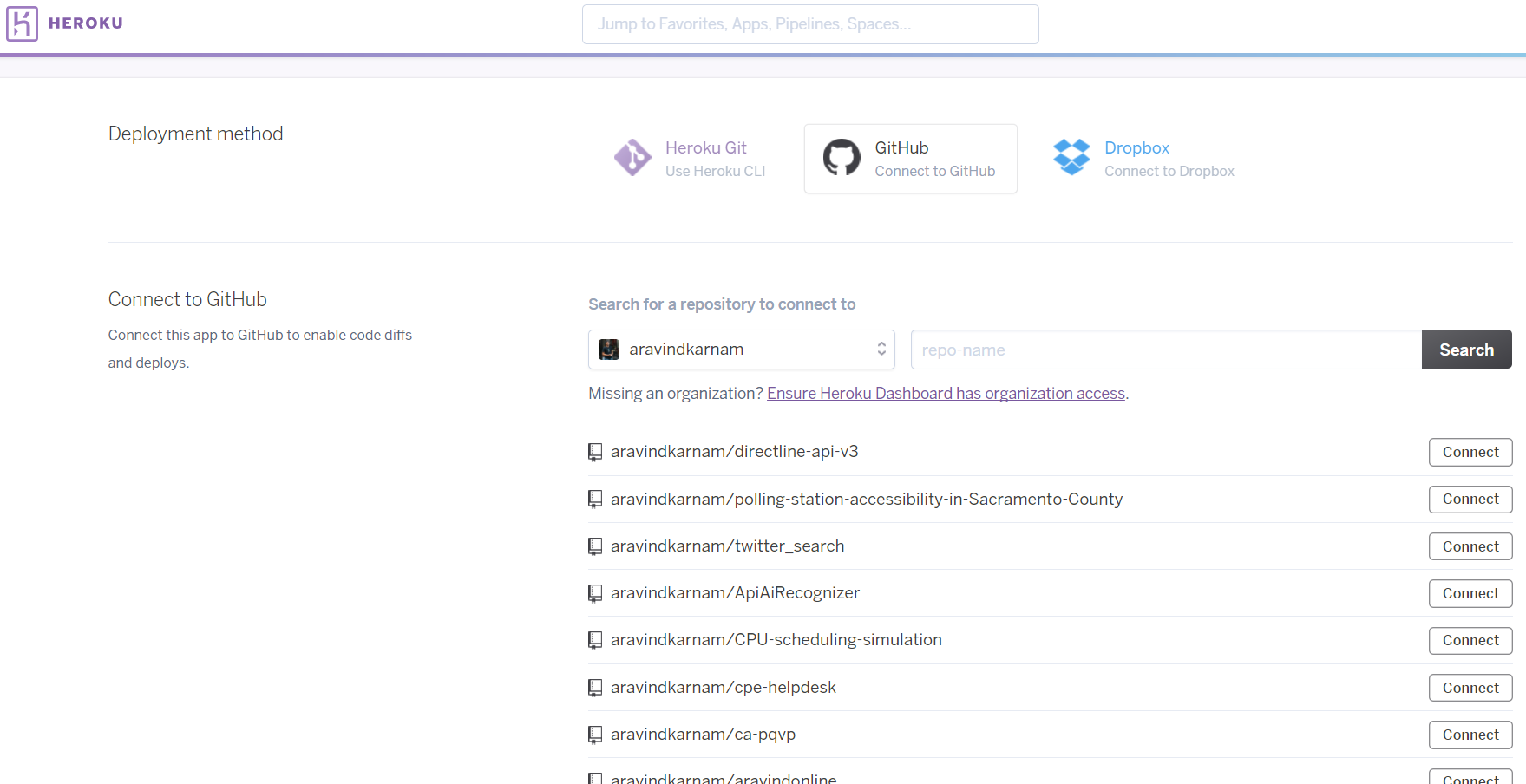


1. On dashboard select the option deploy and under deployment methods choose GitHub and you will be prompted in GitHub to give permissions to Heroku to access to your code.

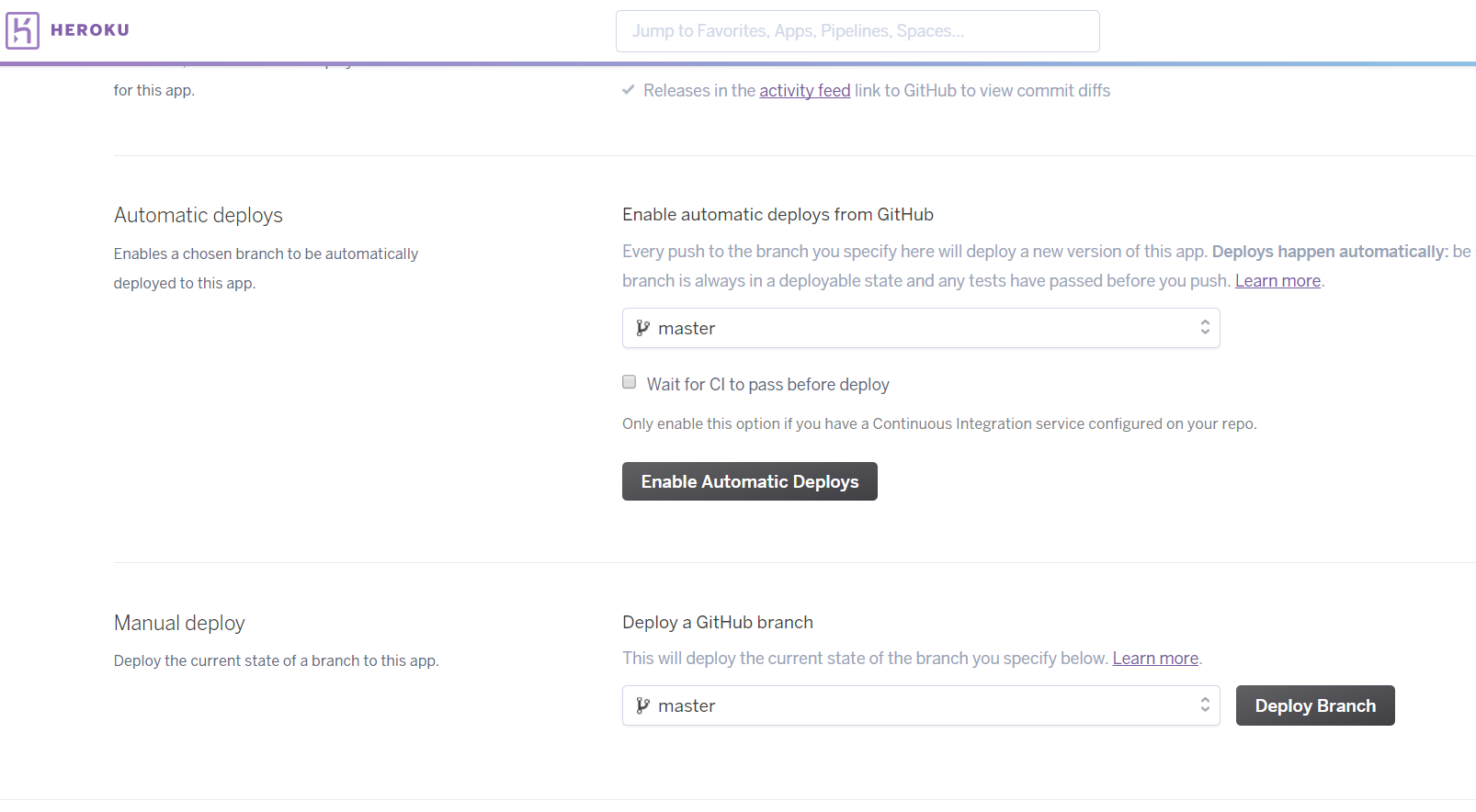




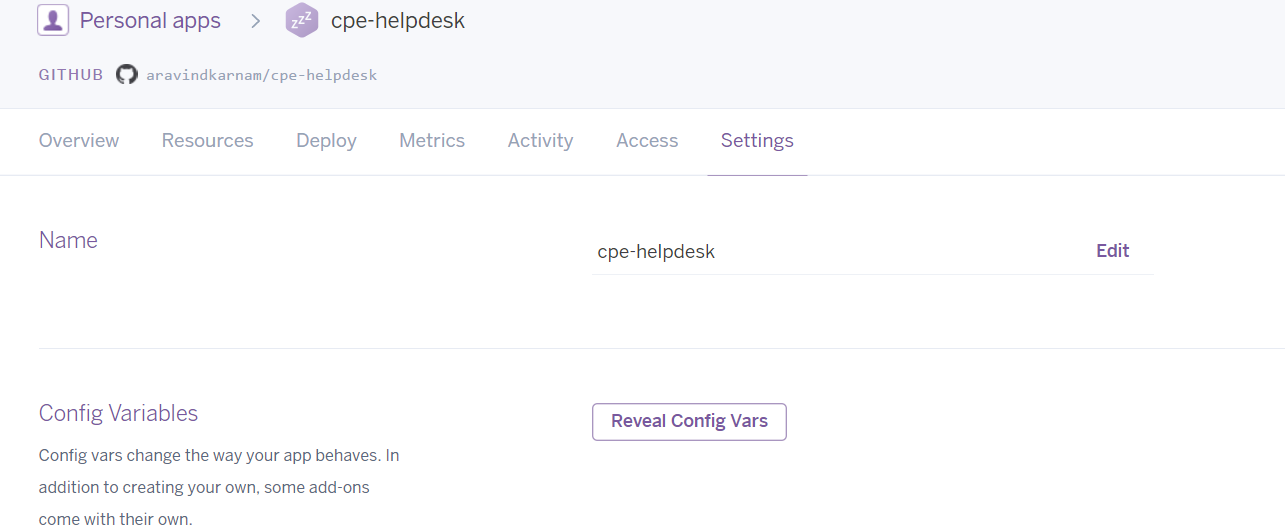
1. Attach the repository that you just forked in GitHub to this app by clicking on connect and this will pair your server instance with code in GitHub

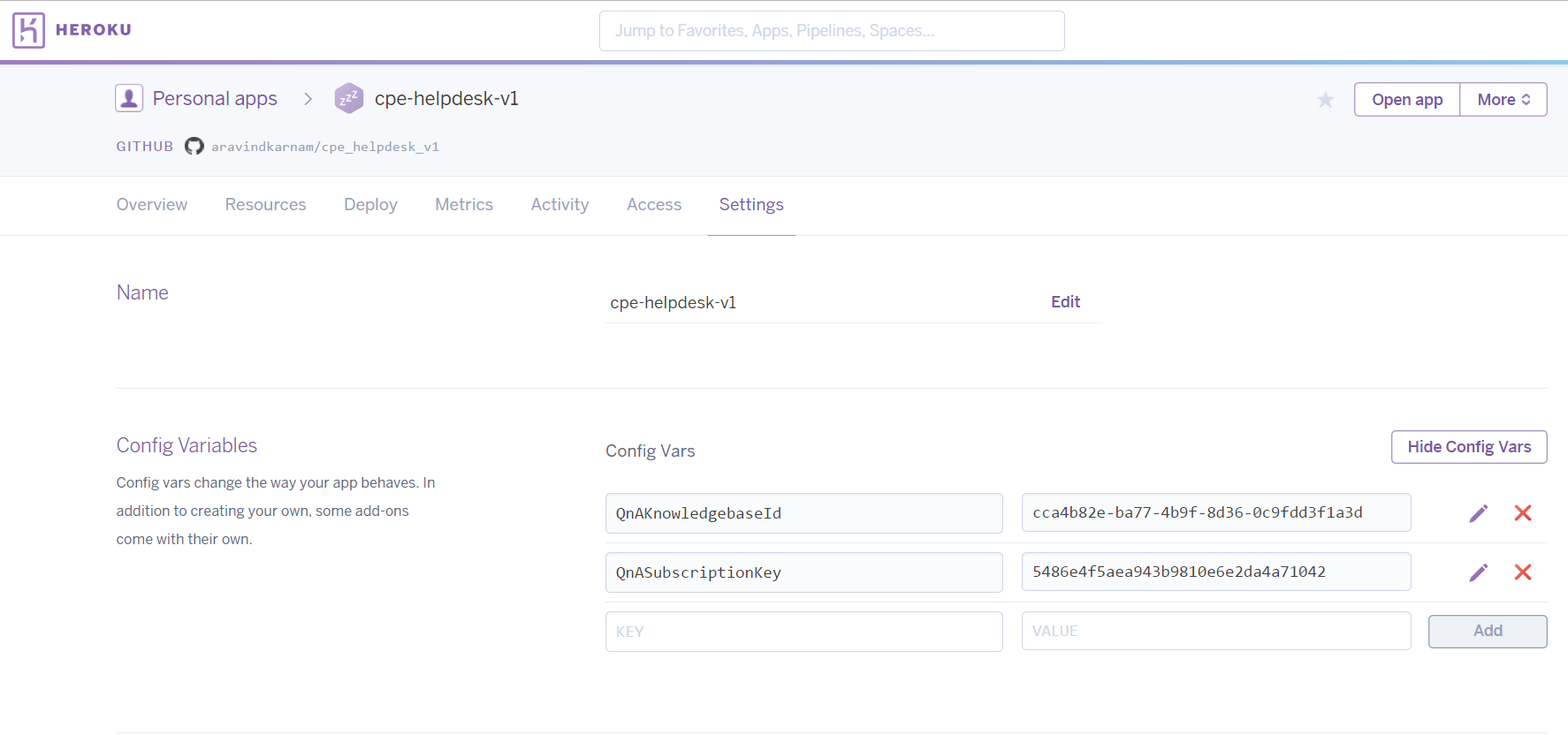


1. Under manual deploy click on deploy branch.

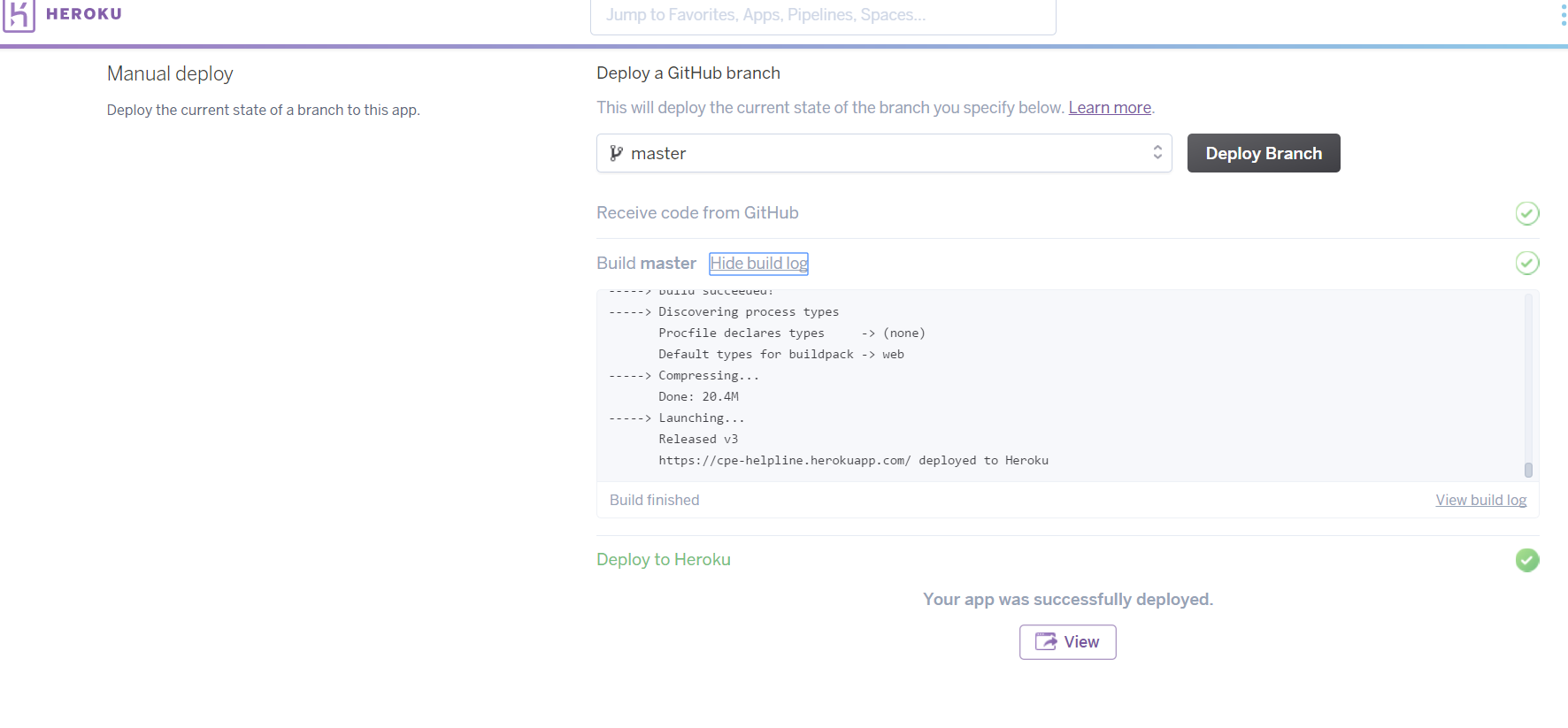


1. Open Settings tab, and click on “reveal config vars” and add all key value pairs as shown below. This help server to connect to the knowledge base.

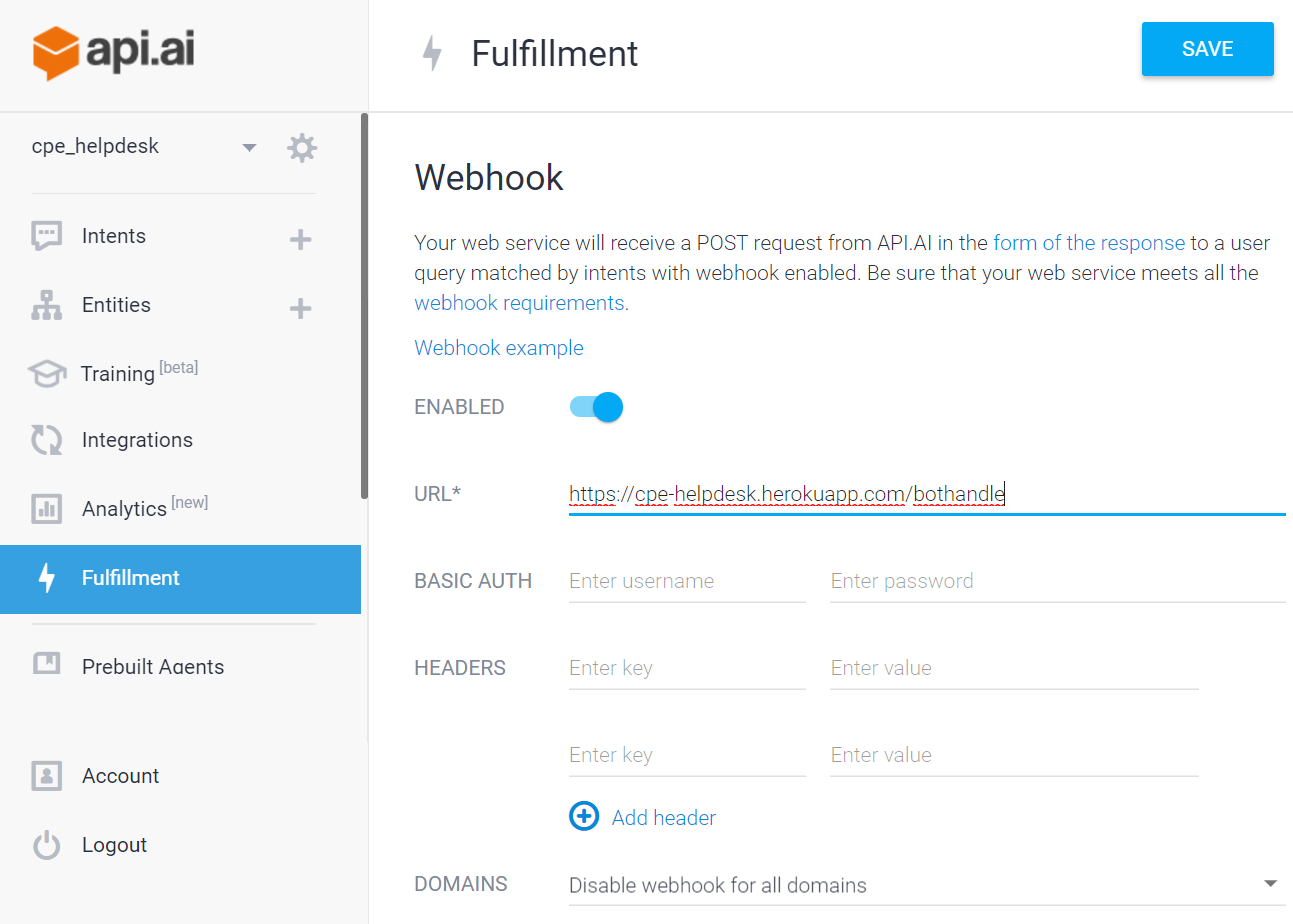
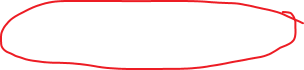




1. Once you see the successful build message, click on view to get to the dash board of your application



1. Copy the URL from the browser and append “/bothandle” at the end of URL, then paste in the fulfilment section of api.ai to finish setup. Don’t forget to click save ☺



Data Entry:

There are three JSON files namely courses.json , professors.json, areas,json. All the required information can be added to this. Follow the format shown below for corresponding files

courses.json:

{

"<course code>": {

"title": "<Course Name>",

"prerequisites": "<list all prerequisites here>"

},

.

.

.

.

"CPE138": {

"title": "Computer networks and internets",

"prerequisites": "CSC 35, CSC 60 and CSC 130"

},

"CPE166": {

"title": "Advanced logic design",

"prerequisites": "ENGR 17 and CPE 64"

},

"CSC139":{

"title": "Operating system principles",

"prerequisites": "CSC 60, CSC 130, CPE 185"

},

"CSC137":{

"title": "Operating system principles",

"prerequisites": "CSC 60, CSC 130, CPE 185"

}

}

areas.json:

{

"<Name of the area>":"<list all related courses here>",

. . . . . . .

"compiler construction": "CSC 151",

"operating systems": "CSC 139 and CSC 60"

}

Professor.json:

[

{

"salutation": "<Dr./Mr./Mrs. etc>",

"lastName": "<last name>",

"firstName": "<first Name>",

"officeHours": "<office hours>",

"email": "<email>",

"courses": "<list all the course they teach here>"

},

.

.

.

.

{

"salutation": "Dr.",

"lastName": "Faroughi",

"firstName": "Nikrouz",

"officeHours": "M W F 4:30 PM to 5:30 PM",

"email": "faroughi@csus.edu",

"courses": "CSC 137"

},

{

"salutation": "Dr.",

"lastName": "Shobaki",

"firstName": "Ghassan",

"officeHours": "M W F 4:30 PM to 5:30 PM",

"email": "shobaki@csus.edu",

"courses": "CSC 139 and CSC 151"

}

]