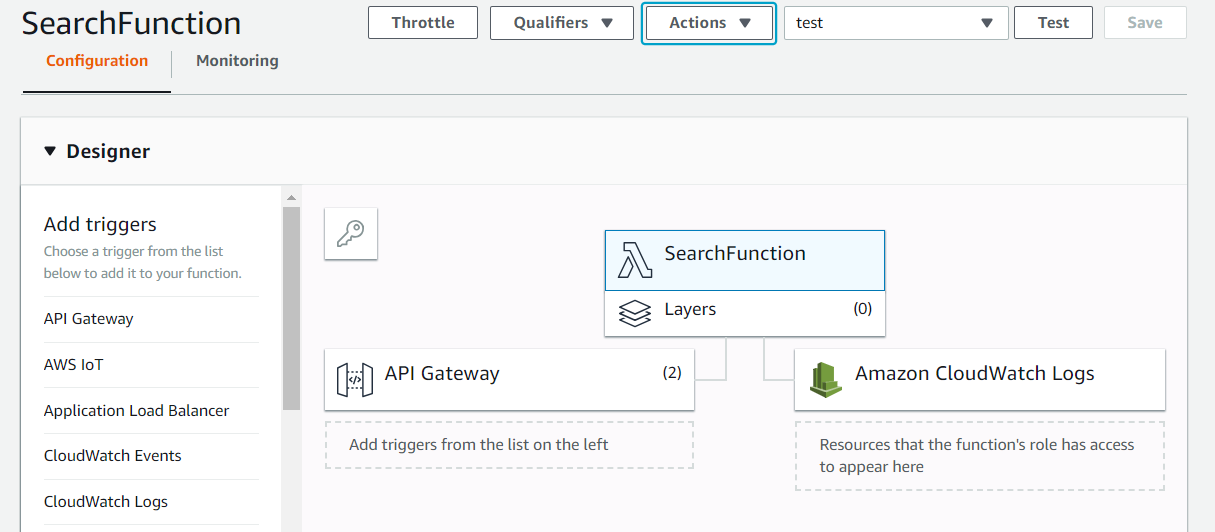
**Overview**

The solution is extensible wherein services are defined in Interface which are extended and implemented in service implementation class. More functions/methods can be added to the service interface and can be implemented in the future as needed. Reusable property file is maintained separately under resources folder to maintain paths and configurations. Also, in the service layer a utility method has been implemented that connects to AWS based on the user credentials and then builds a AWS Cloudsearch Domain on which a search operation can be performed.

The solution will be high performant as it’s designed in a way that the search mechanism is done over cloud using Cloudsearch service, where all the documents are uploaded and most fields are indexed to get the search data/response faster from there. Frequent queries/results can be cached if required. Lambda function can be monitored using X-Ray service on AWS that gives a good insight of the performance and hits coming on the server, in case of high traffic, AWS Lambda dynamically scales function execution in response to increased traffic.

**Solution Architecture and Design**

I used AWS Cloudsearch and AWS Lambda for Java to develop the feature/api. The Lambda function is then backed by API Gateway that exposes an endpoint and a passthrough to Lambda function, which invokes the function call along with the search term. Internally, the call to service and fetches the data back to the client. For testing this out, I have used Postman as a client.



**Technologies/Tools used**

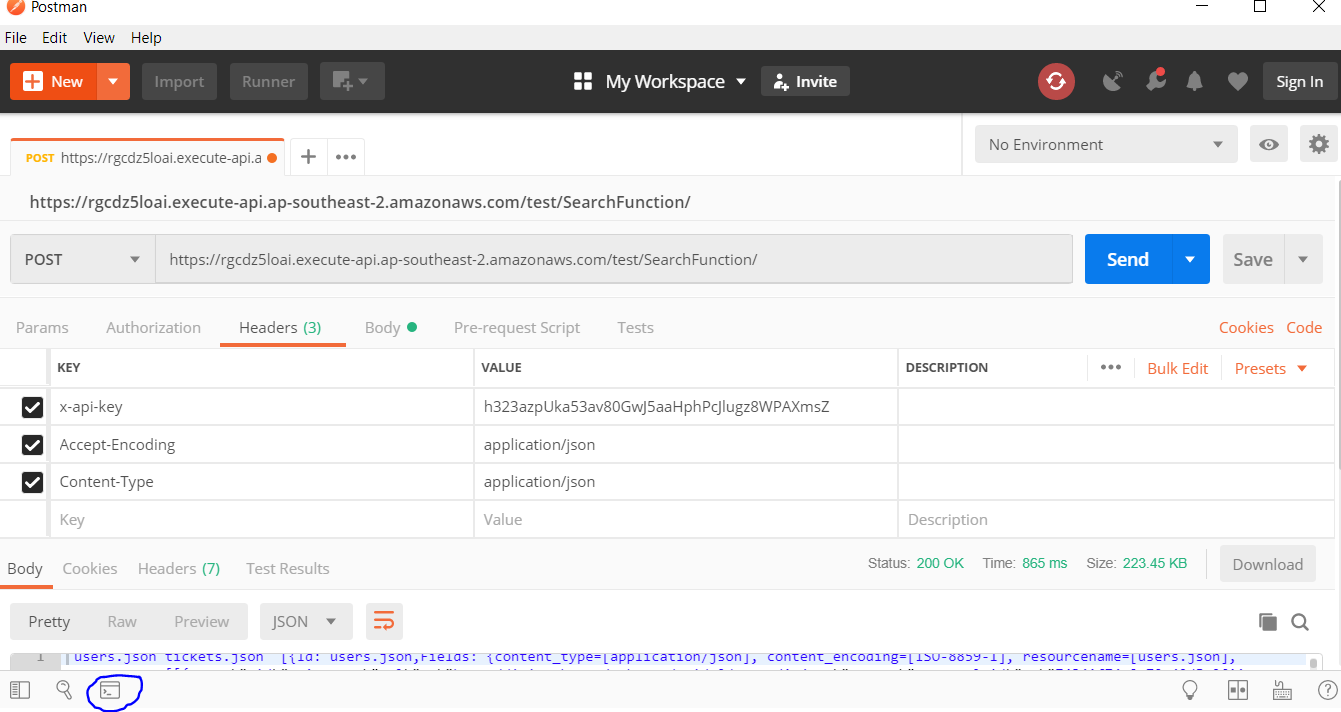
* Java 8
* AWS
* AWS Lambda
* AWS Cloudsearch
* AWS API Gateway
* JSON, REST and Postman

### **Instructions/installation to run the application**

1. Download and Install Postman desktop app or similar
2. Add the following headers as shown in the below screenshot

**Key** **Value**

* X-api-key h323azpUka53av80GwJ5aaHphPcJlugz8WPAXmsZ
* Accept-Encoding application/json
* Content-Type application/json



3. In the Body tab enter the desired text/keyword in double quotes as a search term, for e.g. "Trinidad"

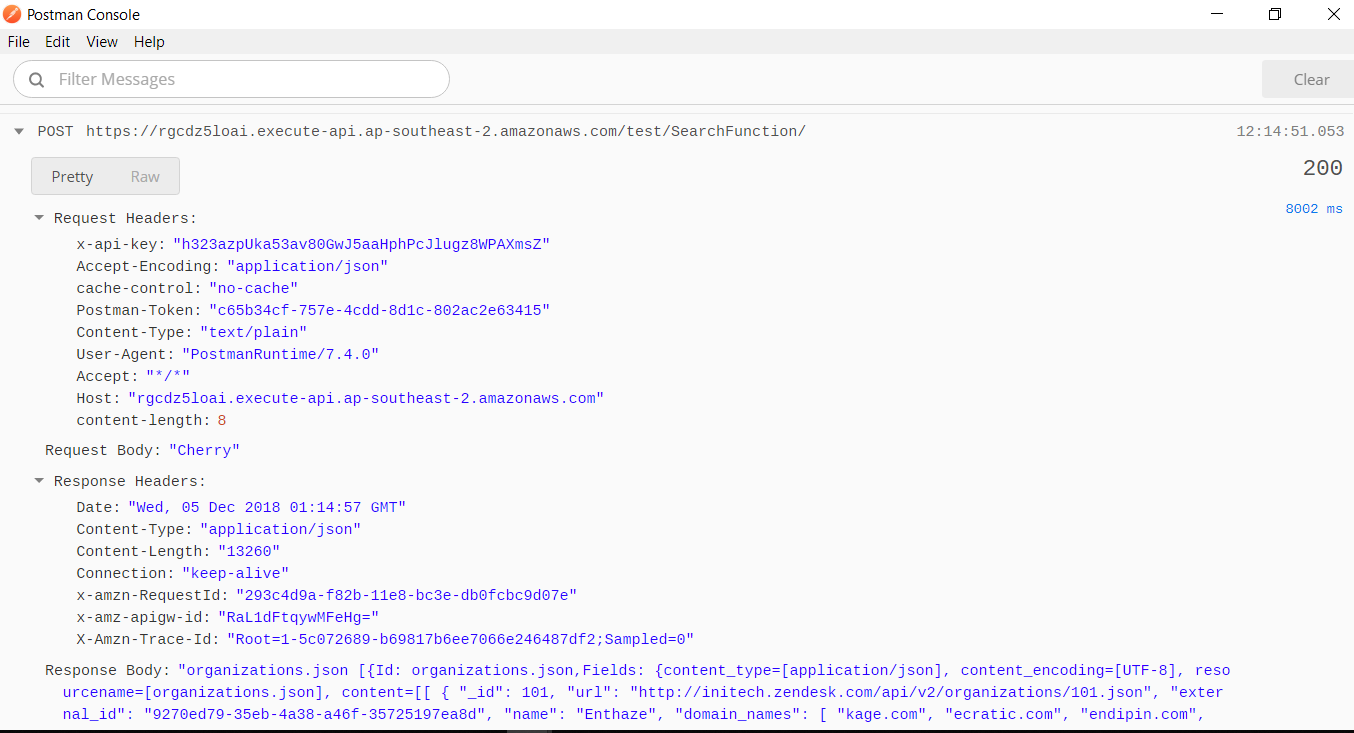
* Click on the Postman Console icon as highlighted at the bottom of the Postman screen above in the blue and open the console
* Go back to the Postman (screen above) and hit the send button and see the results in the Postman console which will record the event and display the result as shown below
* To have a better/pretty json look of the data format, copy the data from the beginning of array [{ till the end }] and paste in any json validator and hit submit button (for e.g. <https://jsonlint.com/>)

4. Output Format -the output/response from the server/Gateway is in the format below:

* Request headers
* Request body (search term)
* Response headers
* Response body (First word is the document name/file which got matched), then it displays the content of the file that matches/contains the term



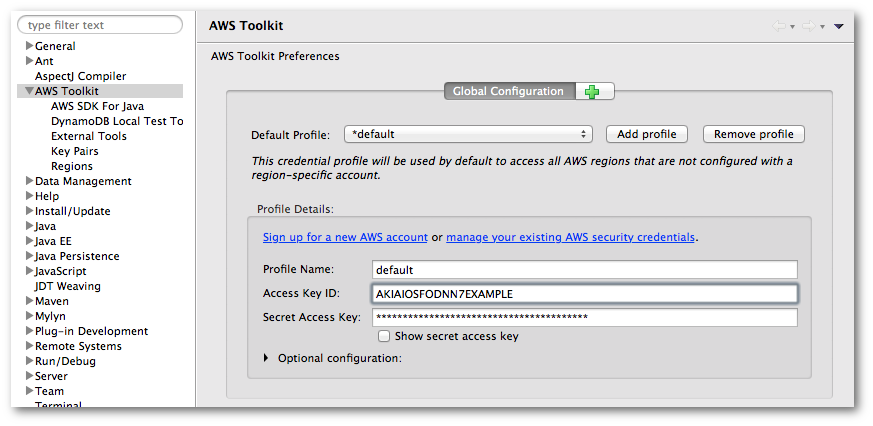
The above screenshot is taken from Postman Console that shows the results matched in both users.json and tickets.json for the given input keyword “Trinidad” which appears in these files/documents. Both files are returned one after the other in the response body.

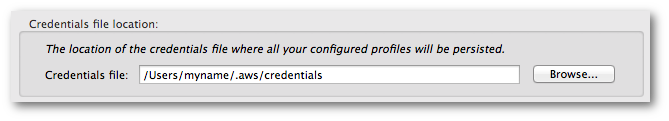


The above screenshot is taken from Postman Console that shows the results matched in organizations.json for the given input keyword “cherry” which appears in this file/document in the tag section.

Run test cases locally

* Download AWS SDK for Java
* Install the Toolkit  
  1. Open Help → Install New Software….  
  2. Enter https://aws.amazon.com/eclipse in the text box labeled “Work with” at the top of the dialog.  
  3. Select the required "AWS Core Management Tools" and other optional items from the list below.  
  4. Click “Next.” Eclipse guides you through the remaining installation steps.  
  Note: The toolkit requires Eclipse 4.4 (Luna) or higher
* Once you have set up the AWS Toolkit for Eclipse you should configure your AWS Credentials.
* To get your access key ID and secret access key  
  - Open the IAM console.  
  - On the navigation menu, choose Users.  
  - Choose your IAM user name (not the check box).  
  - Open the Security credentials tab, and then choose Create access key.  
  - To see the new access key, choose Show. Your credentials resemble the following:  
  Access key ID: AKIAIOSFODNN7EXAMPLE  
  Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY  
    
  - To download the key pair, choose Download .csv file. Store the keys in a secure location.
* **To add your access keys to the AWS Toolkit for Eclipse**- Open Eclipse's Preferences dialog box and click AWS Toolkit in the sidebar.  
  - Type or paste your AWS access key ID in the Access Key ID box.  
  - Type or paste your AWS secret access key in the Secret Access Key box.  
  - Click Apply or OK to store your access key information.  
  - Here's an example of a configured set of default credentials:

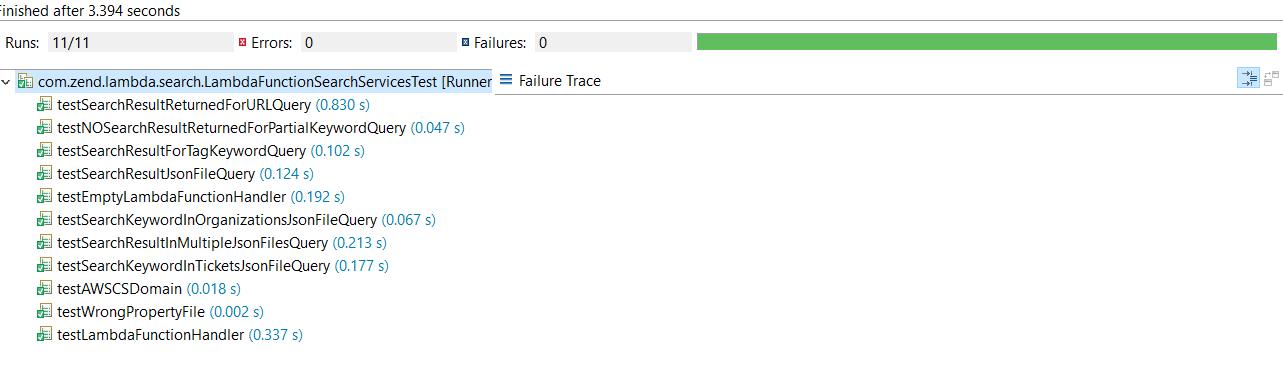


* **To set the AWS credentials file location**In the AWS Toolkit Preferences dialog, locate the Credentials file location section, and enter the pathname of the file where you would like your AWS credentials stored.
* 
* After this setup clone the repository to a location by issuing the command using Gitbash or any terminal-

git clone [git@github.com](mailto:git@github.com):vivek-mishra/SearchApp.git

* Import the project in Eclipse editor
* Build the project to load all the dependencies from pom.xml file
* Open LambdaFunctionSearchServicesTest.java file
* Right click anywhere in this file and click Run as -> Junit Test

This will run all the test cases and can see all the tests are passed as shown below-



**Open aware issues/improvements**

The output is currently unfiltered - it is returning the whole document/file where it matches the term.

**Reference -**

### <https://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/setup-credentials.html>

* <https://aws.amazon.com/eclipse/>
* <https://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/setup-install.html>