# Enterprise Search Using AWS Kendra

***This project report is submitted to***

***Aptus Data Labs, Bangalore***

***in partial fulfillment of the requirements for the award of the certification of***

#### Post Graduate Diploma in

**Data Science (2019-2020)**

##### *Submitted by*

##### Mr. Uday Chandu Bachina (19FTDDS020)

##### Mr. Mohd. Afzal Ramiz (19FTDDS018)&

##### Mr. Sourav Sambit Panda (19FTDDS003)

***Under the Esteemed Supervision of***

##### Mr. Ravindra Swamy

##### Mr. Pramit Kumar Patra

##### Mr. Akshay Barik &

##### Mr. Siddharth

***Duration: From 1st June 2020 To 10th July 2020***

****

**DECLARATION**

I hereby certify that: -

1. The work contained in the project is original and has been done by myself under the supervision of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have conformed to the norms and guidelines given to us by the Project Review Committee of our department.
4. Whenever I have used materials (data, theoretical analysis and text) from other sources, I have given due credit to them by citing them in the text of the project and giving their details in the references.

Date: 12th July 2020

Place: Bhubaneswar

Mr. UdayChandu Bachina (19FTDDS020)

Mr. Mohd. Afzal Ramiz (19FTDDS018)&

Mr. Sourav Sambit Panda (19FTDDS003)

## ACKNOWLEDGEMENT

I would take this opportunity to express my gratitude to all who have made this project possible. I would like to express my deepest appreciation to our team of Aptus Data Labs who provided me the possibility to complete this project. I would like to give a special gratitude to my project supervisor, **Mr. Akshay Barik & Mr. Siddharth,** whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report. I take this opportunity to express my sincere thanks to our CTO of Aptus Data Labs Mr. Ravindra Swamy Sir for providing the necessary facilities. Furthermore, I would also like to acknowledge with much appreciation the crucial role of our friends and parents for encouraging us and helping us complete this project.

**Mr. UdayChandu Bachina (19FTDDS020)**

**Mr. Mohd. Afzal Ramiz (19FTDDS018)&**

**Mr. Sourav Sambit Panda (19FTDDS003)**

## ABSTRACT

Enterprise Search, The practice of identifying and enabling specific content across the enterprise to be indexed, searched, and retrieved.

The ability to locate relevant information is something that users need to be able to do. While this is not a small task, online searching makes it easier to do. Searching is more than just typing something into a search box and getting a result. It’s more about discovering things about a topic that you didn’t necessarily know you were looking for, and browsing as well as discovering new horizons.

A searchable index is created, and other value-added processing, such as metadata extraction and auto-summarization, may take place. These functions group information into logical categories that in turn can be searched and return results to users based on how the particular search engine has categorized them.

Once this index is created, queries can then be accepted. Queries aren’t necessarily questions, as they can also be just terms or phrases that represent whatever you’re looking for, typed into the search box.

At this point, the search engine processes the query by passing over the index, finding the information that matches the particular term or subject entered, and sending that information to some sort of processor, which then sorts the information by relevancy or other measure, clusters it based on the categorization.

**TABLE OF CONTENTS**

[TITLE i](#_Toc4769034)

[DECLARATION ii](#_Toc4769036)

[ACKNOWLEDGEMENT iii](#_Toc4769035)

[ABSTRACT](#_Toc4769034) iv

[TABLE OF CONTENTS](#_Toc4769036) v

[LIST OF FIGURES](#_Toc4769037) vi

[CHAPTER 1 1](#_Toc4769038)

[INTRODUCTION 1](#_Toc4769039)

[1.1 Problem Statement 1](#_Toc4769040)

[1.2 Solution 1](#_Toc4769041)

[CHAPTER 2 3](#_Toc4769042)

[ARCHITECTURE 3](#_Toc4769043)

[2.1 Software Requirements 3](#_Toc4769044)

[2.2 Package Requirements 4](#_Toc4769045)

[2.3 Architecture 5](#_Toc4769046)

[CHAPTER 3 6](#_Toc4769047)

[WORKING FLOW(Step By Step) 6](#_Toc4769048)

[CHAPTER 4 7](#_Toc4769047)

[KNOWN ISSUES/METHODOLOGY 7](#_Toc4769048)

[CHAPTER 5 10](#_Toc4769060)

[MODEL EVALUATION 10](#_Toc4769061)

[CHAPTER 6 11](#_Toc4769047)

[SCOPE OF ENHANCEMENT 11](#_Toc4769048)

[CHAPTER 7 12](#_Toc4769047)

[RESOURCES LINKS 12](#_Toc4769048)

[7.1 Github Links 12](#_Toc4769044)

[CHAPTER 8 13](#_Toc4769083)

[CONCLUSION 14](#_Toc4769084)

[REFERENCES 15](#_Toc4769085)

**Chapter 1**

**Introduction**

**1.1 Problem Statement:**

we have a dataset that resembles a real-world politics content. In the dataset there are 417 unstructured text files which are the sentences about the political parties and their oaths to people.

The final objective of processing the given data set is that retrieving the relevant information for a given query. The query may be any kind of text that related to the data set provided.

**1.2 Solution:**

The required solution is acquired by using AWS KENDRA with the help of some steps.

The given dataset is stored in S3 bucket. From AWS Kendra, create an index with any desired name.

After the creation of index, create a data source specifying the S3 url, thus the initiative of kendra has started.

As soon as the data source is created, sync it, the use of sync is to train the kendra with the set of data.

In the final step, go to the console which had the query search bar. Type any keyword or any sentence to find out the relevant solution. In the background AWS KENDRA uses the machine learning and natural language processing to construct the search model.

**Chapter 2**

**Architecture**

**2.1 Software Requirements:**

The softwares required for building the aws comprehend:

1. Python Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

1. AWS CLI

The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

We have configured the AWS CLI with Access\_Key\_Id ,Secret\_Key,Region and the format of output by the help of a command

> aws configure

**>aws configure**

**AWS Access Key ID [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XJUE]:**

**AWS Secret Access Key [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FtKs]:**

**Default region name [us-west-2]:**

**Default output format [json]:**

Above details are passed correctly to use the services of **AWS(**Amazon Web Services**)**

**2.2 Package Requirements:**

Packages used:

1. boto3

In order to use of boto3 the developer needs a account in AWS. So,by using the credentials of aws account the services are easily accessed.

The methods we used in boto3 are:

1. Client=boto3.client('kendra')

to access the services from aws,we used the client() method.

1. Client.create\_index()

Step1 to work with Kendra is creating an index.

1. client.describe\_index()

display the status of the index whether the index creation is failed or success

or in active state.

1. Client.create\_data\_source()

The next step after creating an index is to create a data source where the S3 bucket url is given.

1. Client.describe\_data\_source()

This describes the created data source with its name, status and its data source connector.

1. Client.list\_data\_sources()

returns the list of data sources present under the index.

1. Client.start\_data\_source\_sync\_job()

Generates the index for each document in the specified data source and processes the data in it.

1. Client.list\_data\_souce\_sync\_jobs()

Returns the list of sync jobs present under the data source with a filter option that is ,succeeded jobs or failed jobs

1. Client.stop\_data\_source\_sync\_jobs()

User can interrupt the ongoing sync job using this method

1. Client.query()

Ask a question to Kendra that returns the related documents to the given query.

1. Client.create\_faq()

User can create a frequent asked query bucket with in the Kendra’s index to retrieve the customized output as a result.

1. Client.delete\_faq()

Deletes the bucket that has created ,user has to provide the faq\_id and index\_id

1. Client.delete\_data\_source()

Deletes the data source that is under an index

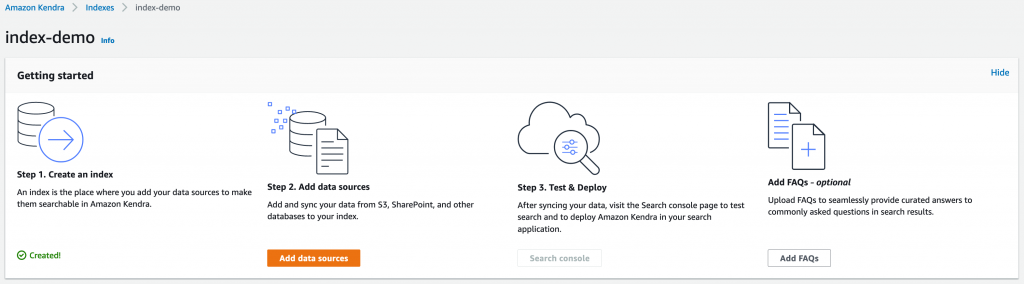
1. Client.delete\_index()

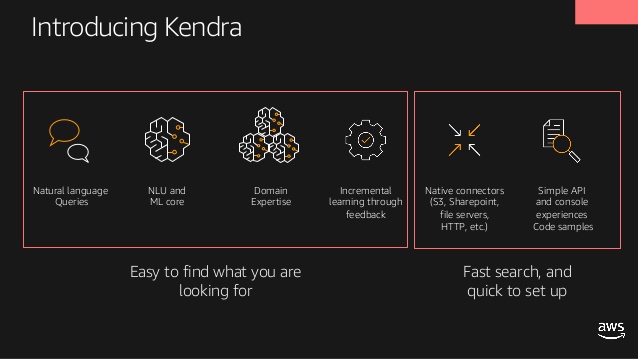
Deletes the main index.

**2.3 Architecture:**

The following figure shows the flow of data for retrieving the result for a given set of documents.

The data from different platforms can be collected and are placed in the S3 storage in AWS and are passed to the AWS Kendra.





In the architecture the following are the important stages in creating the Enterprise Search,

1. creating index

2. creating a data source

3. syncing the job

4. console

5. querying

**Chapter 3**

**Steps to follow to create an Enterprise Search Console**

**Working Flow (Step by Step)**

**Step1: Data Collection**

Collect the required unstructured data on which an search console to be created.

**Step2: Store the data into an S3 bucket in AWS**

The collected data is taken and stored in the s3 bucket in aws. In our case the data is the politics data set which contains 417 unstructured text documents.

**Step3: Creating an index in Kendra**

The following code creates the index with in Kendra .

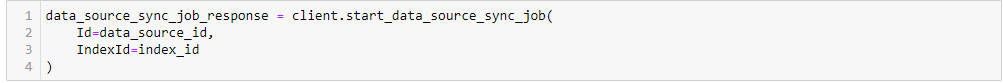


**Step4: Creating the data source**



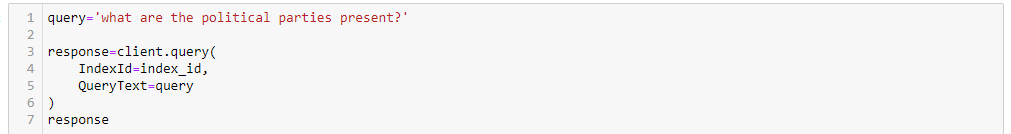
**Step5: Syncing the job**

The 417 text documents are stored into a data source are now going to indexed and any new text documents are added to index can be indexed and crawled with the sync job method.



**Step:6 Querying the console**

To ask anything, search console can be useful to fetch the useful insights from the AWS Kendra.

****

**Chapter 4**

**Known issues and Methodologies**

When you synchronize your Amazon Kendra index with a data source, you may run into issues that prevent the documents from being indexed. Indexing is a two step process. First, there is the data source level process of crawling the data source to find the new and updated documents to index, and to find any documents to remove from the index. Second, there is the document level process where each document is accessed and indexed.

An error can occur in either of these steps. Data source level errors are reported in the console in the Sync run history section of the data source details page. The status of the synchronization job can be Succeeded, Incomplete, or Failed. You can also see the number of documents indexed and deleted during the job. If the status is Failed, a message is shown in the Details column.

Document level errors are reported in Amazon CloudWatch Logs. You can see the errors using the CloudWatch console.

**My synchronization job failed**

A synchronization job typically fails when there is a configuration error in the index or the data source. The error message in the Details column of the data source gives information about what went wrong. The problem is usually that the index or the data source does not have the proper IAM permissions. The error message describes the permissions that are missing. Here are some of the error messages that you can receive:

Failed to create log group for job. Please make sure that the IAM role provided has sufficient permissions.

If your index role does not have permission to use CloudWatch, the data source will not be able to create a CloudWatch log. If you get this error you need to add CloudWatch permissions to the index role.

Failed to access S3 file prefix (bucket name) while trying to crawl your metadata files. Please make sure the IAM Role (role ARN) provided has sufficient permissions.

When you are using an Amazon S3 data source, Amazon Kendra must have permission to access the bucket that contains the documents. You need to add permission for Amazon Kendra to read the bucket to the data source IAM role.

The provided IAM Role (role ARN) could not be assumed. Please make sure Amazon Kendra is a trusted entity that is allowed to assume the role.

Amazon Kendra needs permission to assume the index and data source IAM roles. You need to add a trust policy to the roles with permission for the sts:AssumeRole action.

For the IAM polices that Amazon Kendra needs to index a data source, see IAM access roles for Amazon Kendra.

Q: What is Amazon Kendra?

A: Amazon Kendra is a highly accurate and easy to use enterprise search service that’s powered by machine learning. Kendra enables developers to add search capabilities to their applications so their end users can discover information stored within the vast amount of content spread across their company. This includes data from manuals, research reports, FAQs, HR documentation, customer service guides, and is found across various systems such as S3, SharePoint, Salesforce, Servicenow, RDS databases, One Drive, and many more coming later this year. When you type a question, the service uses machine learning algorithms to understand the context and return the most relevant results, whether that be a precise answer or an entire document. For example, you can ask a question like "How much is the cash reward on the corporate credit card?” and Kendra will map to the relevant documents and return a specific answer like “2%”. Kendra provides sample code so that you can get started quickly and easily integrate highly accurate search into your new or existing applications.

**My synchronization job is incomplete**

Jobs are generally incomplete when they have completed the data source level process but have had some error during the document level process. When a job is incomplete, some of the documents may have been successfully indexed. For an Amazon S3 data source, an incomplete job is typically caused by:.

The metadata for one or more documents was invalid.

When there are documents to submit for indexing, at least one document was not submitted.

When there are documents to submit for deleting from the index, at least one document was not submitted.

To troubleshoot an incomplete synchronization job, look first to your CloudWatch logs.

From the details column, choose View details in CloudWatch.

Review the error messages to see what caused the document to fail.

**My synchronization job succeeded but there are no indexed documents**

Occasionally you will have a index synchronization job run that is marked as Succeeded but there are no new or updated documents indexed when you expect there to be. Here are some reasons:

Check CloudWatch DocumentsSubmittedForIndexingFailed metric to see if there were any documents that failed to synchronize. Check your CloudWatch logs for details.

For an Amazon S3 data source, you may have given Amazon Kendra the wrong bucket name or prefix. Make sure that the bucket that Amazon Kendra is using is the one that contains the documents to index.

If you are re-indexing a document that failed to be indexed in an earlier job, Amazon Kendra won't index it unless you make a change to the document or its associated metadata file.

**General troubleshooting**

Amazon Kendra uses CloudWatch metrics and logs to provide you with insight into synchronizing your data sources. You can use the metrics and logs to determine what went wrong with a synchronization run and to determine what you need to do to fix it.

For general troubleshooting, start with your CloudWatch metrics.

Check the DocumentsCrawled metric to see how many documents your data source checked. For an Amazon S3 bucket, if the number is less than you expect, check to be sure that your data source is pointing to the right bucket.

Check the DocumentsSkippedNoChange metric to see how many documents were skipped because they haven't changed since the last synchronization. If the number does not match what you expect, check to make sure that your repository was updated correctly.

Check the DocumentsSkippedInvalidMetadata metric to see how many documents had invalid metadata. Check your CloudWatch logs to see the specific errors that occurred.

Check the DocumentsSubmittedForIndexingFailed metric to see the number of documents that were sent from the data source to the index but failed to be indexed for some reason. For example, if you use a metadata attribute in an Amazon S3 data source that hasn't been defined as a custom index field the document will not be indexed. Check your CloudWatch logs to see the specific errors that occurred.Check the DocumentsSubmittedForDeletionFailed metric to see how many documents that the data source attempted to remove from the index failed to be deleted from the index. Check your CloudWatch logs to see the specific errors that occurred.You can look at the CloudWatch logs for a particular synchronization run to get details of the errors that occurred during the run. For more information about CloudWatch logs with Amazon Kendra, see Monitoring Amazon Kendra with Amazon CloudWatch Logs.

Q: What types of questions can I ask Amazon Kendra?

A: Amazon Kendra supports the following common types of questions:

Factoid questions (who, what, when, where): “Who is Amazon’s CEO?” or “When is Prime Day 2019?” these questions require fact-based answers that may be returned in the form of a single word or phrase. The precise answer, however, must be explicitly stated in the ingested text contentDescriptive questions: “How do I connect my Echo Plus to my network?” or “How do I obtain tax benefits for lower income families?” where the answer could be a sentence, passage, or an entire document.

Keyword searches: “Health Benefits” or “IT Help Desk”, where the intent and scope are not very clear. Kendra will leverage its deep learning models to return relevant documents in these cases.

Q. How does Amazon Kendra handle security?

A. Amazon Kendra encrypts your data in motion and at rest. You have two choices for encryption keys for data at rest: AWS owned customer master key (CMK), AWS managed CMK in your account; a customer managed CMK. In motion, Amazon Kendra uses the HTTPS protocol to communicate with your client application. API calls to access Amazon Kendra through the network use Transport Layer Security (TLS) that must be supported by the client.

**Chapter 5**

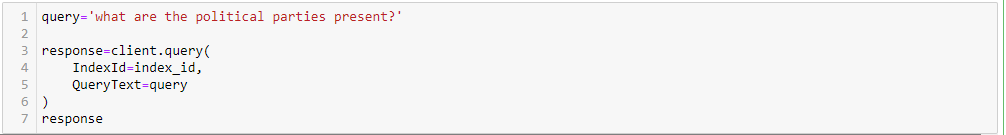
**Model Evaluation**

In Enterprise search there is no model evaluation, instead there is cross checking to the result whether the user has gained the relevant output for a given query or not.

For a given query, the user can filter out what kind of output he/she wants from Kendra. Therefore the available filters are ANSWER, DOCUMET and QUESTION\_ANSWER type.

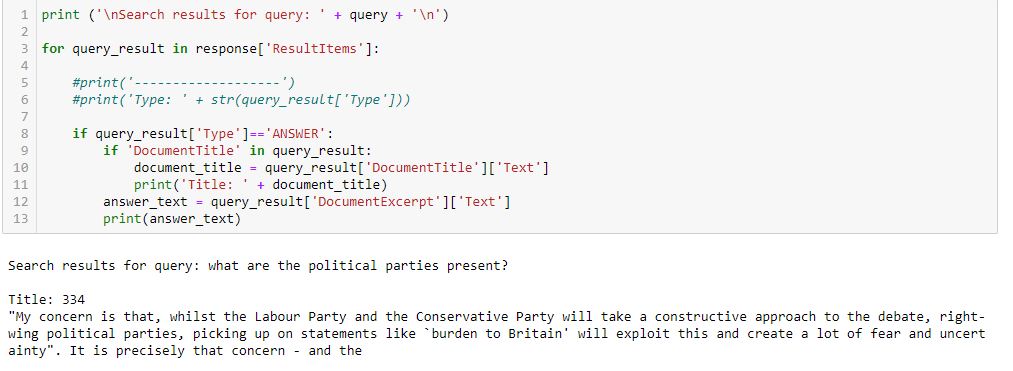
For example:

For a given query like the following,



The results are of in hundreds, in this case the results are 268

Filtering out with the help of filters we can have,



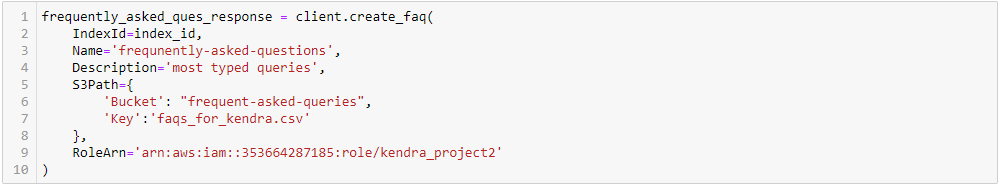
The above result for ANSWER filter. Likewise we can different set of outcomes for different filters

**Chapter 6**

**Scope of Enhancement**

1. Development of Enterprise search doesn’t stop with only creating of search console. In advance this search we can build FAQ’s and also the Kendra can be used in the user own application.

Bulding FAQ bucket in Kendra.



This creates the bucket with in the index of Kendra and whenever the user types of a query from faq, the console displays the same result that whichever the user has given while creation.

1. **Even more enhancing of Kendra is that utilizing the application in the user own space.**

You can add the following components from the search console in your application:

* **Main search page** **–** The main page that contains all the components. This is where you integrate your application with the Amazon Kendra API.
* **Search bar** **–** The component where you enter a search term and that calls the search function.
* **Results** **–**The component that displays the results from Amazon Kendra. It has three components: suggested answers, FAQ results, and recommended documents.
* **Pagination** **–**The component that paginates the response from Amazon Kendra.

Amazon Kendra provides source code that you can deploy in your website. This is offered free of charge under a modified MIT license so you can use it as is or change it for your own needs.

This section contains instructions to deploy Amazon Kendra search to your website. You use a Node.js demo application that runs locally in your machine. This use case is based on a MacOS environment.

To run this demo, you need the following components:

* [Npm (Node.js)](https://docs.npmjs.com/downloading-and-installing-node-js-and-npm)
* [IAM credentials](https://aws.amazon.com/blogs/security/how-to-find-update-access-keys-password-mfa-aws-management-console/) from a user with the appropriate [permissions](https://docs.aws.amazon.com/kendra/latest/dg/security_iam_id-based-policy-examples.html) to use Amazon Kendra

1. Download [amazon\_aws-kendra-sample-app-master.zip](https://aws-ml-blog.s3.amazonaws.com/artifacts/kendra-docs/aws-kendra-sample-app-master.zip) and unzip the file.
2. Open a terminal window and go to the aws-kendra-sample-app-master folder:

cd /{folder path}/aws-kendra-sample-app-master

1. Create a copy of the .env.development.local.example file as .env.development.local:

cp .env.development.local.example .env.development.local

1. Edit the .env.development.local file and add the following connection parameters:
   * **REACT\_APP\_INDEX** – Your Amazon Kendra index ID (you can find this number on the Index home page)
   * **REACT\_APP\_AWS\_ACCESS\_KEY\_ID** – Your account access key
   * **REACT\_APP\_AWS\_SECRET\_ACCESS\_KEY** – Your account secret access key
   * **REACT\_APP\_AWS\_SESSION\_TOKEN** – Leave it blank for this use case
   * **REACT\_APP\_AWS\_DEFAULT\_REGION** – The Region that you used to deploy the Kendra index (for example, us-east-1)
2. Save the changes.
3. Install the Node.js dependencies:

npm install

1. Launch the local development server:

npm start

1. View the demo app at <http://localhost:3000/>. You should see the following screenshot.

**Chapter 7**

**Resources Link**

* **Data available at**<http://mlg.ucd.ie/files/datasets/bbc-fulltext.zip>

**(consider only the politics data folder)**

* <https://aws.amazon.com/kendra/>
* <https://docs.aws.amazon.com/kendra/latest/dg/kendra-dg.pdf>
* <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/kendra.html>
* <https://docs.aws.amazon.com/cli/latest/reference/kendra/index.html>

**Chapter 8**

**Conclusion**

Hence, with the help of AWS KENDRA it is notified that Enterprise Searching is made easy and also the time consumption for searching is really compressed.

Amazon Kendra is a highly accurate and easy to use enterprise search service that's powered by machine learning. Kendra enables developers to add search capabilities to their applications so their end users can discover information stored within the vast amount of content spread across their company.

**References:**

* <https://aws.amazon.com/blogs/machine-learning/enhancing-enterprise-search-with-amazon-kendra/>
* <https://aws.amazon.com/blogs/machine-learning/smarter-faq-bots-with-amazon-kendra/>
* <https://aws.amazon.com/blogs/aws/reinventing-enterprise-search-amazon-kendra-is-now-generally-available/>
* <https://aws.amazon.com/blogs/publicsector/aws-launches-machine-learning-enabled-search-capabilities-covid-19-dataset/>
* <https://aws.amazon.com/blogs/industries/aws-reinvent-2019-healthcare-and-life-sciences-industry-recap/>
* <https://aws.amazon.com/kendra/pricing/>