Building Keyword Searches for Scanned Documents Using Amazon Textract

August 2019

Notices

Contents

Abstract

Exchanging scanned documents is a crucial part of many business transactions today. Invoice processing is just one such example. After an invoice is received, the information it contains is scanned or manually entered into an ERP system and then routed to different groups for validation, approval, and payment. This whitepaper describes building a serverless solution for centrally storing scanned invoices and enabling keyword searches of those invoices using Amazon Textract.

Introduction

A crucial part of many business transactions today is the exchange and processing of scanned documents. One such example is invoice processing, which is an end-to-end process with many different tasks to handle invoices that are received. After an invoice is received, the information it contains is entered into an enterprise resource planning (ERP) system, either manually or by using optical character recognition (OCR) software. Other downstream processing tasks complete the processing of the invoice based on the information extracted, such as transaction date, who ordered it, what were the items, how much did each item cost, and the amount of taxes paid.

Two of the challenges customers face with invoice processing are:

- How to digitally store invoices in a secure, centralized location that not only provides easy access but also is cost-effective
- How to enable keyword-based searching of these invoices in a low-cost manner, while balancing speed with convenience and cost

This whitepaper focuses on building a serverless solution for addressing these two challenges using Amazon S3, Amazon Textract, AWS Lambda, and Amazon Elasticsearch Service. After reading this whitepaper, you should be able to use these AWS services to perform keyword searches of scanned invoices that you have received from your vendors. You also should be able to search for information in these invoices using either the Kibana dashboard, or apps using the Elasticsearch APIs.

Amazon Textract

Amazon Textract[1] is a managed service that automatically extracts text, data, tables, and key-value pairs from a scanned document. No machine language (ML) expertise is needed to use Amazon Textract, which can read many different types of documents and accurately extract text and data from those documents without the need to write code or manually customize and configure templates. It is built on the proven, highly scalable, deep learning technology that Amazon computer vision scientists use daily to analyze billions of images. Amazon Textract is available in the AWS Management Console,[2] the AWS SDK,[3] and the AWS Command Line Interface (CLI).[4]

How Amazon Textract Processes Documents

Amazon Textract can be used to detect text in a document, or to both detect and analyze text to find deeper relationships, such as whether specific text is part of a table or part of a form. Amazon Textract supports both synchronous and asynchronous processing. If you want to process a single page document and need a fast, real-time response with low latency, use synchronous processing. However, if you have a large, multipage document, or if you don't require an immediate response, use asynchronous processing.

Regardless of the processing method, Amazon Textract returns a JSON document that consists of an array of `Block` objects. Each block contains information about a detected item, such as its type, its location, its relationship with other items, and the confidence that Amazon Textract has in the accuracy of the processing.

Detecting Text

When used only to detect text, Amazon Textract returns the lines and words of text that it found in the document, the location of those items on the page, what page they are on, the relationships between the items on a page, and its confidence in the accuracy of the result. This information is returned in multiple `Block` objects. A block of type WORD contains information about a word, a LINE block contains information about a line, and a PAGE contains information about a page. Blocks are related to each other through parent/child relationships.

Analyzing Text

When used to analyze text, Amazon Textract not only detects text in the document but also analyzes that text to determine if the text is part of a table or a form. It also looks for text that can be represented as a key-value pair, such as text that is associated with a check box or radio (option) button.

The analyze processing returns its information in `Block` objects of type TABLE, CELL, KEY_VALUE_SET, SELECTION_ELEMENT, KEY, and VALUE. These blocks are in addition to the PAGE, LINE, and WORD blocks returned by the detect text processing. These blocks are related through parent/child relationships.

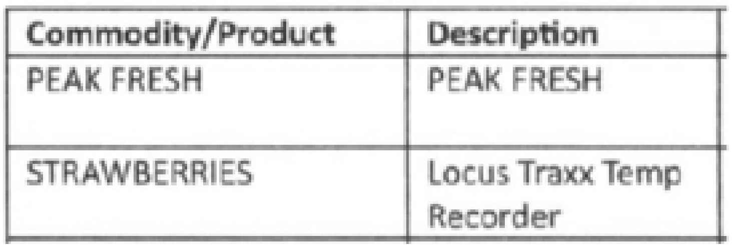
Key-Value Pairs

Amazon Textract can detect linked text items in the document as key-value pairs and returns multiple KEY_VALUE_SET block objects. In the following example, it can identify a key name of `InvoiceNumber` and a value of `IN000001`:

InvoiceNumber: IN000001

Tables

Amazon Textract can detect tables and returns multiple block objects, called TABLE, CELL, and WORD, that represent the table, a cell in the table, and a word in a cell of the table, respectively. For example, in the following image, Amazon Textract detects a table with six cells.



Commodity/Product	Description
PEAK FRESH	PEAK FRESH
STRAWBERRIES	Locus Traxx Temp Recorder

Figure 1: Example Table for Amazon Textract

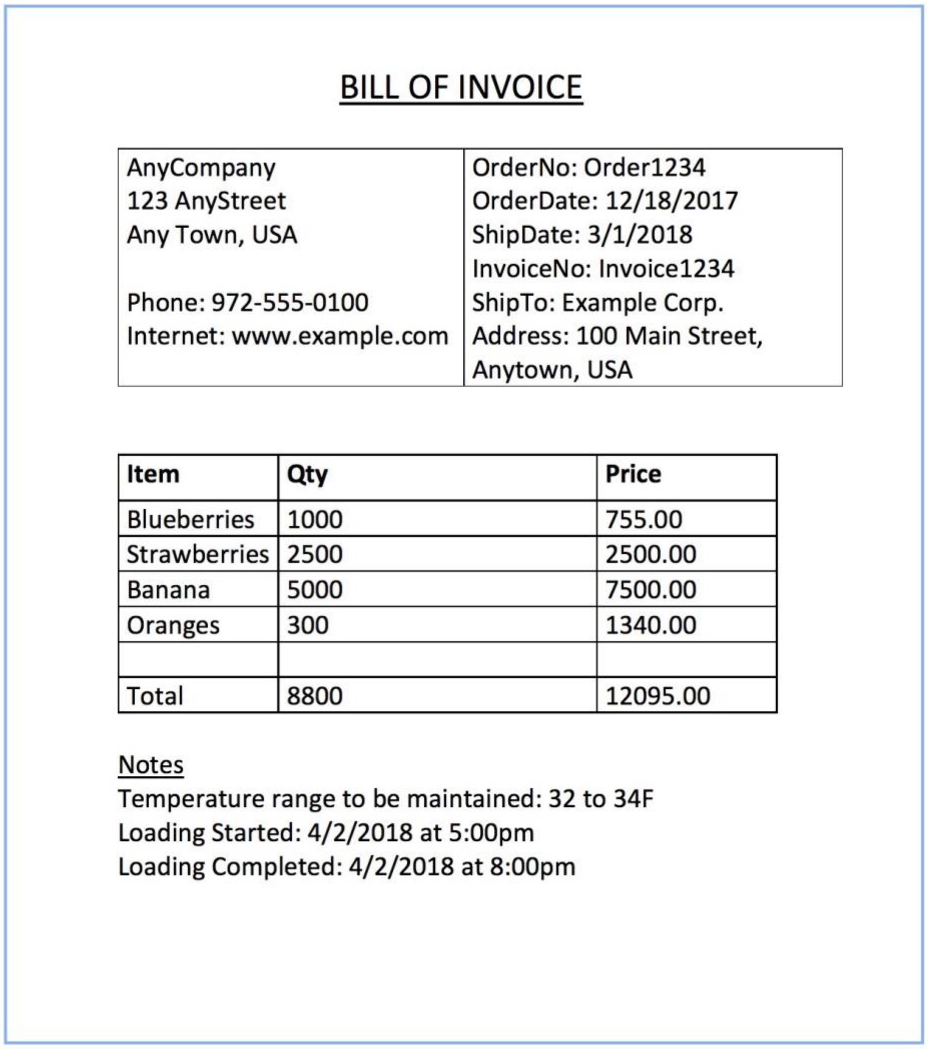
Radio Buttons and Check Boxes

Amazon Textract can detect radio buttons and check boxes in the document and returns a SELECTION_ELEMENT block. A SELECTION ELEMENT block can be associated with either a key-value pair or a table cell.

Invoice Processing Using Amazon Textract

A typical invoice has text, key-value pairs, tables, check boxes, radio buttons, and forms in it. The analyzing text function of Amazon Textract is well suited for extracting this type of information and making it available for the downstream steps associated with invoice processing.

In the following example, Amazon Textract is used to analyze the image of an invoice. Using the sample image in Figure 2, it identifies a table, key-value text pairs, and text.



BILL OF INVOICE

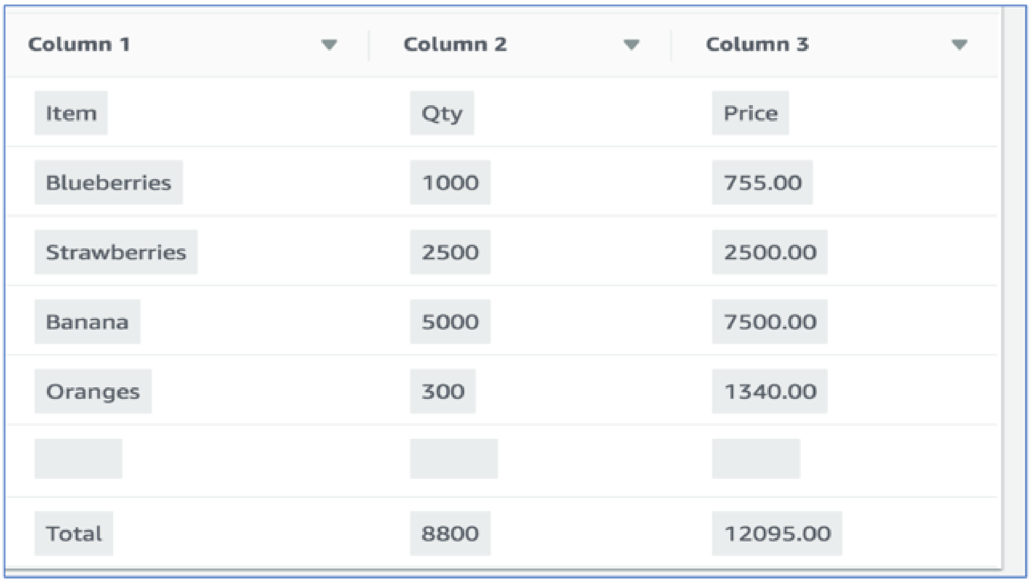
AnyCompany 123 AnyStreet Any Town, USA Phone: 972-555-0100 Internet: www.example.com	OrderNo: Order1234 OrderDate: 12/18/2017 ShipDate: 3/1/2018 InvoiceNo: Invoice1234 ShipTo: Example Corp. Address: 100 Main Street, Anytown, USA

Item	**Qty**	**Price**
Blueberries	1000	755.00
Strawberries	2500	2500.00
Banana	5000	7500.00
Oranges	300	1340.00
Total	8800	12095.00

Notes
Temperature range to be maintained: 32 to 34F
Loading Started: 4/2/2018 at 5:00pm
Loading Completed: 4/2/2018 at 8:00pm

Figure 2: Sample Invoice

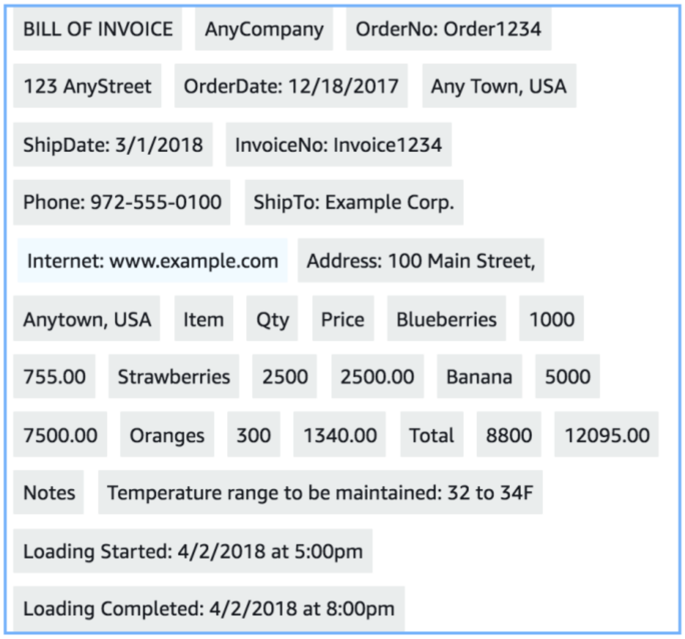
Figure 3 shows the table that was identified.



Column 1	Column 2	Column 3
Item	Qty	Price
Blueberries	1000	755.00
Strawberries	2500	2500.00
Banana	5000	7500.00
Oranges	300	1340.00
Total	8800	12095.00

Figure 3: Table Detected by Amazon Textract

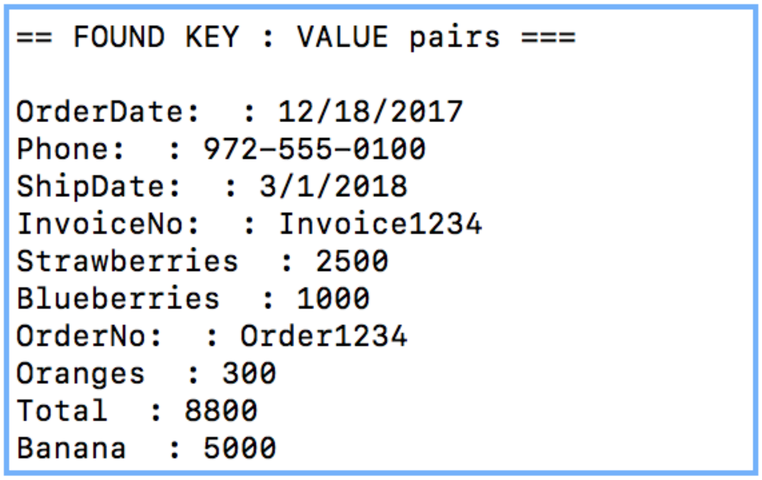
Figure 4 shows the text that was identified.



BILL OF INVOICE | AnyCompany | OrderNo: Order1234
123 AnyStreet | OrderDate: 12/18/2017 | Any Town, USA
ShipDate: 3/1/2018 | InvoiceNo: Invoice1234
Phone: 972-555-0100 | ShipTo: Example Corp.
Internet: www.example.com | Address: 100 Main Street,
Anytown, USA | Item | Qty | Price | Blueberries | 1000
755.00 | Strawberries | 2500 | 2500.00 | Banana | 5000
7500.00 | Oranges | 300 | 1340.00 | Total | 8800 | 12095.00
Notes | Temperature range to be maintained: 32 to 34F
Loading Started: 4/2/2018 at 5:00pm
Loading Completed: 4/2/2018 at 8:00pm

Figure 4: Text Detected by Amazon Textract

Figure 5 shows the key-value pairs that were identified.

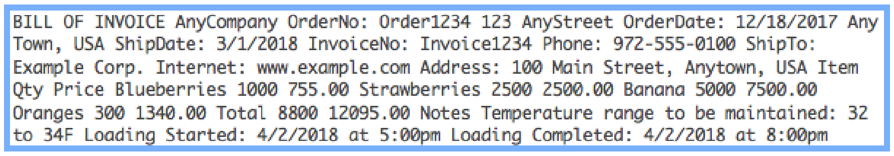


```
== FOUND KEY : VALUE pairs ===

OrderDate:   : 12/18/2017
Phone:   : 972-555-0100
ShipDate:   : 3/1/2018
InvoiceNo:   : Invoice1234
Strawberries   : 2500
Blueberries   : 1000
OrderNo:   : Order1234
Oranges   : 300
Total   : 8800
Banana   : 5000
```

Figure 5: Key-Value Pairs Detected by Amazon Textract

To build the keyword search database for this sample serverless solution, the detect text processing of Amazon Textract is used. Using the example invoice image in Figure 2, the detect text processing returns the keywords shown in Figure 6.



```
BILL OF INVOICE AnyCompany OrderNo: Order1234 123 AnyStreet OrderDate: 12/18/2017 Any
Town, USA ShipDate: 3/1/2018 InvoiceNo: Invoice1234 Phone: 972-555-0100 ShipTo:
Example Corp. Internet: www.example.com Address: 100 Main Street, Anytown, USA Item
Qty Price Blueberries 1000 755.00 Strawberries 2500 2500.00 Banana 5000 7500.00
Oranges 300 1340.00 Total 8800 12095.00 Notes Temperature range to be maintained: 32
to 34F Loading Started: 4/2/2018 at 5:00pm Loading Completed: 4/2/2018 at 8:00pm
```

Figure 6: Keywords Detected by Amazon Textract

Note: Amazon Textract also returns the bounding box and other information that is not needed for this particular search solution. That information has been omitted from the results returned in Figure 6.

Amazon S3

Amazon Simple Storage Service (Amazon S3[5]) is an object storage service built to store and retrieve any amount of data from anywhere. Amazon S3 provides easy to use management features so that you can organize your data and define fine-tuned access controls to meet your specific business, organizational, and compliance requirements. The main benefits of Amazon S3 are that it provides industry-leading performance, scalability, availability, durability, a wide range of cost-effective storage classes, unmatched security, compliance, audit capabilities, management tools for granular data control, and query services for analytics.

For this solution, Amazon S3 will be the centralized location used to store all the uploaded invoices. Because it has limitless scalability, you don’t need to worry about running out of space. Amazon S3 provides a number of ways to upload documents and you can choose the method that works best for your use case. If you are using a web or mobile app, you can use the AWS SDK to upload the scanned invoices directly to an S3 bucket.

Only one S3 bucket is used in this solution, and it’s used to hold all the scanned invoices. Amazon S3 is seamlessly integrated with AWS Lambda using event triggering. This mechanism is used to invoke a Lambda function when an invoice is uploaded. The Lambda function calls Amazon Textract to extract the text and to store keywords in Elasticsearch.

Amazon Elasticsearch Service

Amazon Elasticsearch Service[6] is a fully managed service that makes it easy to deploy, secure, and operate Elasticsearch at scale with zero down time. This service offers open-source Elasticsearch APIs, managed Kibana, and integrations with Logstash and other AWS services, enabling you to securely ingest data from any source and search, analyze, and visualize the data in real time. This solution uses Elasticsearch to store the keywords extracted from the invoices and to search for them using the Kibana dashboard or the Elasticsearch APIs.

AWS Lambda

AWS Lambda[7] lets you run code without provisioning or managing servers. Lambda runs code on a highly available compute infrastructure and performs all of the administration of the underlying platform, including server and operating system maintenance, capacity provisioning, automatic scaling, patching, code monitoring, and logging.

With Lambda, you can run code with zero administration. Lambda functions can be invoked either synchronously or asynchronously. You also can set up your code to automatically be triggered by other AWS services, or called directly from a web or mobile app.

AnalyzeInvoice Lambda Function

For this solution, one Lambda function, called `AnalyzeInvoice,` is used. This function calls Amazon Textract to process the uploaded invoices in the S3 bucket and to store the keywords in Elasticsearch.

Figure 7 shows the `AnalyzeInvoice` Lambda function, which is written in Python.

The `AnalyzeInvoice` Lambda function performs the following actions:

1. Extracts the S3 bucket name and key from the event records.
2. Creates the Amazon Textract client and makes a synchronous call to it, passing the S3 bucket name and key.
3. Extracts the keywords from the result returned by Amazon Textract.
4. Adds the keywords into the Elasticsearch index.

This function uses an environment variable called `ElasticSearchHost` to obtain the host name of the Elasticsearch cluster.

```
import json
import boto3
import os
from urllib.parse import unquote plus
from elasticsearch import Elasticsearch, RequestsHttpConnection

def lambda handler(event, context):
    try:
        #Get the bucket name and objectname
        s3client = boto3.client('s3')
        bucket = event['Records'][0]['s3']['bucket']['name']
        document = unquote plus(event['Records'][0]['s3']['object']['key'])
        documenturl = '{}/{}/{}'.format(s3client.meta.endpoint url, bucket, document)

        #Create boto3 client and call detect document text
        client = boto3.client('textract')
        #process using S3 object
        response = client.detect document text(
            Document={'S3Object': {'Bucket': bucket, 'Name': document}}
        )

        #Get the text blocks
        blocks=response['Blocks']
        text=""
        for item in response["Blocks"]:
            if item["BlockType"] == "LINE":
                text += " "+item["Text"]+" "


        addToESIndex(documenturl,document, text)
        #print(text)
    except Exception as e:
     print(e)
     raise e

def addToESIndex(s3URI, objectName, text):
        lambdaSession = boto3.Session()
        region = lambdaSession.region name
        host=os.environ['ElasticSearchHost']

        es = Elasticsearch(
                hosts = [{'host': host, 'port':443}],
                use ssl = True,
                verify certs = True,
                connection class = RequestsHttpConnection
                )
        document = {
                "name": "{}".format(objectName),
                "s3URI": "{}".format(s3URI),
                "content": text
        }

        es.index(index="textractsearch", doc_type="document", id=objectName, body=document)
```

Figure 7: `AnalyzeInvoice` *Function in Python*

Architecture

The architecture for the serverless invoice processing solution is shown in Figure 8.

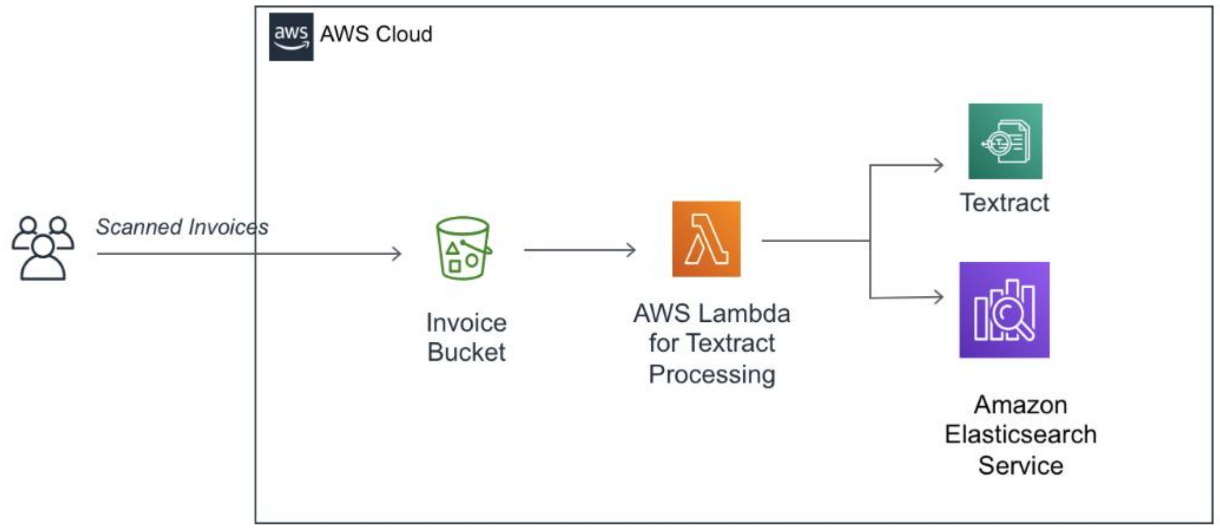

AWS Cloud
Scanned Invoices
Invoice Bucket
AWS Lambda for Textract Processing
Textract
Amazon Elasticsearch Service

Figure 8: Serverless Invoice Processing Using Amazon Textract

Your company vendors will upload the scanned invoices via either Amazon S3 ingestion methods or via your customer application using AWS SDK. It will trigger the Lambda function called `AnalyzeInvoice` that will send the invoice to Amazon Textract for processing, extract the keywords from the Amazon Textract output and stores it in the Elasticsearch index. Here are the steps to deploy this architecture:

1. Create an S3 bucket to store the invoices uploaded from your vendors.

2. Create an Elasticsearch domain and cluster. For instructions and more information, see Creating Amazon ES Domains.[8]

3. Create an AWS Identity and Access Management (IAM) role for the Lambda function that has permission to access the S3 bucket and Amazon Textract. For example:

```
[
 “Version”:”2012-10-17”,
 “Statement”:[
  {
     “Effect”:”Allow”,
     “Action”:[
             “s3:*”,
             “textract:*”
             ],
     “Resource”:”*”
   }
]
```

4. At the time of publication of this whitepaper, the AWS SDK for Python (Boto3)[9] doesn’t include the Amazon Textract API. You must include it with your deployed code by creating a layer in Lambda using the Boto3 package. For details, see Creating an AWS Lambda Function in the *Amazon Textract Developer Guide*.[10]

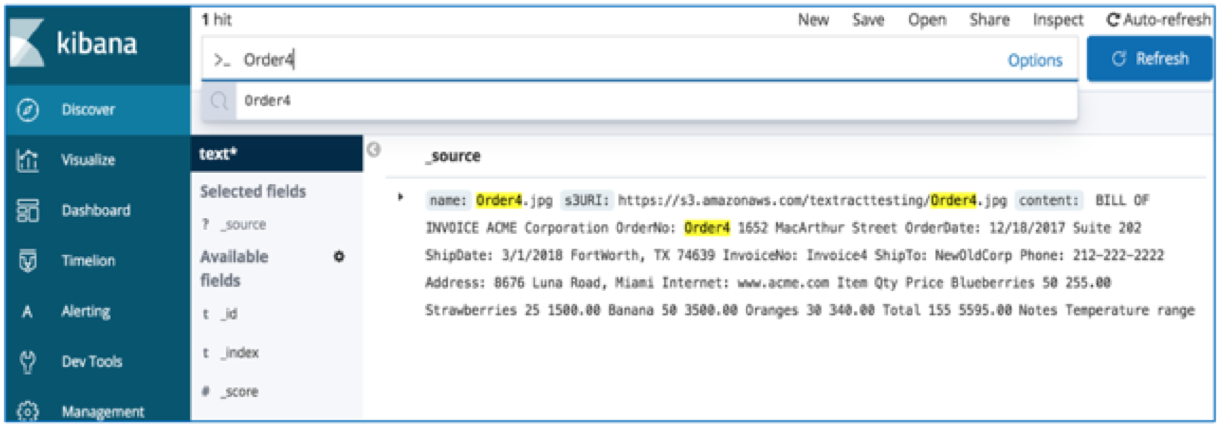
5. Deploy the `AnalyzeInvoice` function shown in Figure 7 with the role created in step 2, and with the layer created in step 4. You might need to modify the code if your Elasticsearch only allows specific IAM users.

6. Create a trigger from the S3 bucket created in step 1 to the `AnalyzeInvoice` Lambda function.

7. Create an environment variable called `ElasticSearchHost` for the `AnalyzeInvoice` function that points to the Elasticsearch host that you configured in step 2.

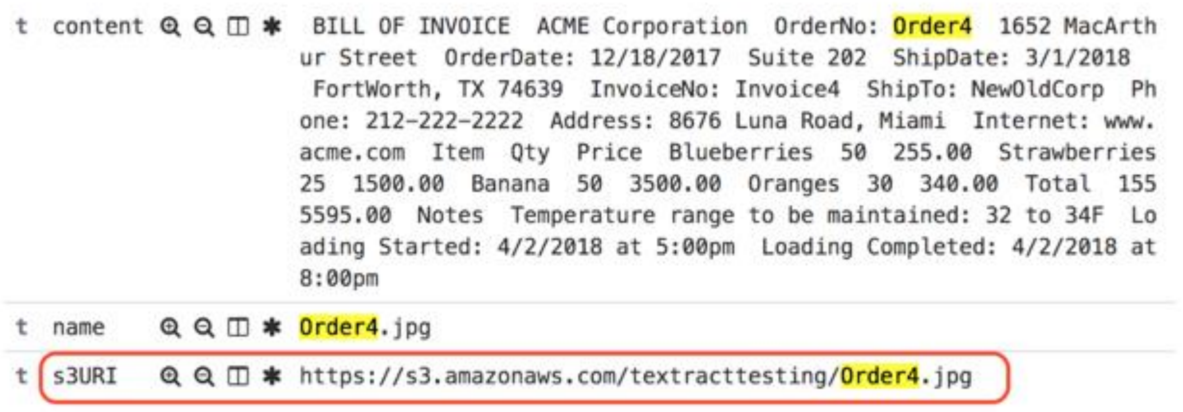
8. Upload an invoice to the S3 bucket and check the Amazon CloudWatch log files for `AnalyzeInvoice` Lambda function entries to verify whether the request was processed successfully.

After successfully deploying the solution, you should be able to search for keywords using the Elasticsearch Kibana dashboard. The search will look similar to this in the Kibana dashboard:



```
name: Order4.jpg s3URI: https://s3.amazonaws.com/textracttesting/Order4.jpg content: BILL OF
INVOICE ACME Corporation OrderNo: Order4 1652 MacArthur Street OrderDate: 12/18/2017 Suite 202
ShipDate: 3/1/2018 FortWorth, TX 74639 InvoiceNo: Invoice4 ShipTo: NewOldCorp Phone: 212-222-2222
Address: 8676 Luna Road, Miami Internet: www.acme.com Item Qty Price Blueberries 50 255.00
Strawberries 25 1500.00 Banana 50 3500.00 Oranges 30 340.00 Total 155 5595.00 Notes Temperature range
```

If you expand the search result, it will show the Amazon S3 URI of the source invoice.



```
t  content   BILL OF INVOICE  ACME Corporation  OrderNo: Order4  1652 MacArth
             ur Street  OrderDate: 12/18/2017  Suite 202  ShipDate: 3/1/2018
              FortWorth, TX 74639  InvoiceNo: Invoice4  ShipTo: NewOldCorp  Ph
             one: 212-222-2222  Address: 8676 Luna Road, Miami  Internet: www.
             acme.com  Item  Qty  Price  Blueberries  50  255.00  Strawberries
             25  1500.00  Banana  50  3500.00  Oranges  30  340.00  Total  155
             5595.00  Notes  Temperature range to be maintained: 32 to 34F  Lo
             ading Started: 4/2/2018 at 5:00pm  Loading Completed: 4/2/2018 at
             8:00pm
t  name      Order4.jpg
t  s3URI     https://s3.amazonaws.com/textracttesting/Order4.jpg
```

Extending the Architecture

You can extend this architecture by adding additional downstream processing steps using AWS Step Functions.[11] For example, you could implement the following use cases:

- Create a document workflow automation using Amazon Textract, Amazon Translate,[12] and Amazon Comprehend.[13]

- Create an electronic discovery (e-discovery) application that involves multimedia processing and building a relationship graph in Amazon Neptune.[14] A number of AI/ML services could be applied, such as Amazon Transcribe,[15] Amazon Rekognition,[16] Amazon Translate, and Amazon Comprehend.

Conclusion

This whitepaper provides a solution for building a keyword search for scanned invoices. You can use this solution as a basis for building keyword searches for any type of scanned document.

Contributors

Contributors to this document include:

- Raja Mani, Solutions Architect, Amazon Web Services

Document Revisions

Date	Description
August 2019	First publication

Notes

[1] https://aws.amazon.com/textract/

[2] https://aws.amazon.com/console/
[3] https://aws.amazon.com/tools/
[4] https://aws.amazon.com/cli/

[5] https://aws.amazon.com/s3/

[6] https://aws.amazon.com/elasticsearch-service/

[7] https://aws.amazon.com/lambda/

[8] https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/es-createupdatedomains.html#es-createdomains

[9] https://aws.amazon.com/sdk-for-python/

[10] https://docs.aws.amazon.com/textract/latest/dg/lambda.html

[11] https://aws.amazon.com/step-functions/

[12] https://aws.amazon.com/translate/

[13] https://aws.amazon.com/comprehend/

[14] https://aws.amazon.com/neptune/

[15] https://aws.amazon.com/transcribe/

[16] https://aws.amazon.com/rekognition/