

SEARCH ENGINE DEPLOYMENT

1. Create a new OpenSearch Domain using the AWS console:

- ❖ Log in to your AWS Management Console.

The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' services like Amazon OpenSearch Service, Lambda, S3, and others. The main area has two main sections: 'Applications' (0) and 'Cost and usage'. The 'Applications' section shows a message to 'Get started by creating an application.' and a 'Create application' button. The 'Cost and usage' section displays current month costs (\$0.65), forecasted month end costs, and a bar chart showing spending from Aug 23 to Jan 24.

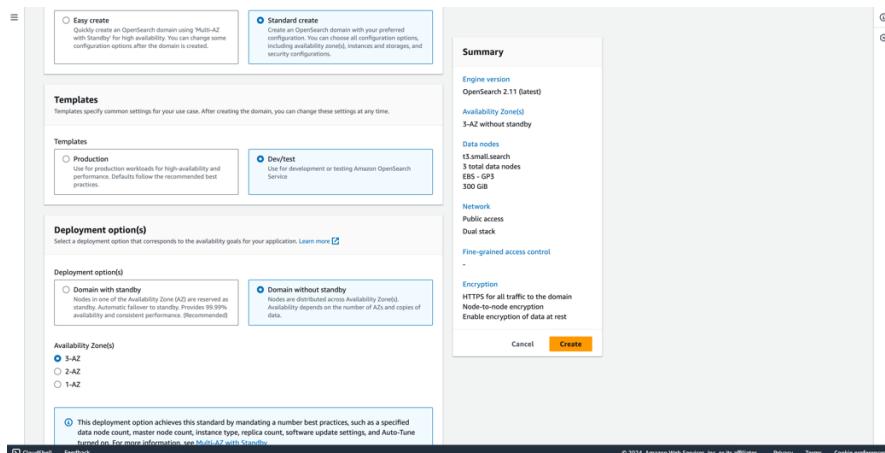
- ❖ Navigate to the OpenSearch Service section.

The screenshot shows the Amazon OpenSearch Service Dashboard. It includes sections for 'Domain health' (0 domains, 0 Red, 0 Yellow, 0 Green), 'Scheduled updates' (No scheduled updates available at this time), and 'Serverless' (with a link to learn more). On the left, there's a navigation sidebar with options like Managed clusters, Serverless, Ingestion, Integrations, and more.

- ❖ Click on "Create domain."

The screenshot shows the 'Create domain' wizard. It starts with a 'Name' field containing 'opensearch'. Below it, there are two options for 'Domain creation method': 'Easy create' (radio button) and 'Standard create' (checkbox selected). The 'Standard create' option allows users to choose configuration options like engine version, availability zones, data nodes, network, and encryption. Other sections include 'Templates' (Production or Dev/test), 'Deployment options', and a summary on the right. A large orange 'Create' button is at the bottom right.

- ❖ Choose a deployment type (e.g., "Development and testing," "Production," etc.) and click "Next."

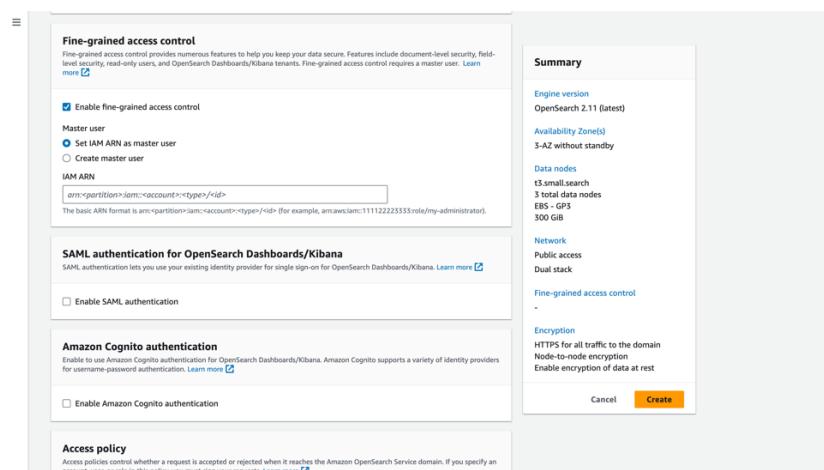


- ❖ Enter a domain name and configure other settings as needed.
- ❖ In the "Data nodes" section, specify the number of data nodes, instance types, and storage settings. Click "Next" when done.
- ❖ Configure other advanced settings if necessary and proceed to the next steps.
- ❖ Finally, review your configuration and click "Create domain."

2. Select “VPC access” in the Network Section:

- ❖ In your OpenSearch domain dashboard, go to the "Network configuration" section.
- ❖ Choose the VPC and subnets where you want to deploy your OpenSearch domain. Configure the security groups accordingly.
- ❖ Save the changes.

3. In the “Fine-grained access control” section:



- ❖ Ensure Fine-grained access control is turned on:
- ❖ In the OpenSearch domain dashboard, navigate to the "Fine-grained access control" section.
- ❖ Ensure that fine-grained access control is enabled. If not, enable it.
- ❖ Create a Master User with a username and password:

The screenshot shows the 'Fine-grained access control' configuration page. It includes fields for enabling fine-grained access control, creating a master user (username: admin, password: password), and SAML authentication options. The 'Create' button is highlighted.

Still in the "Fine-grained access control" section, find the "Master User" tab. Create a new Master User by providing a username and password. Note down these credentials as you'll need them later.

4. In the “Access Policy section,” select “Only use fine-grained access control”:

The screenshot shows the 'Access policy' configuration page. It includes sections for SAML authentication, Amazon Cognito authentication, Access policy (selected), Domain access policy (selected), and Encryption. The 'Only use fine-grained access control' option is selected. The 'Create' button is highlighted.

- ❖ In the OpenSearch domain dashboard, go to the "Access policy" section.
- ❖ Choose "Only use fine-grained access control" to ensure that access to your OpenSearch domain is controlled through fine-grained access policies.
- ❖ Save the changes.

Your OpenSearch domain is being created.

[Amazon OpenSearch Service](#) > [Domains](#) > opensearch

opensearch [Info](#)

General information

Name: opensearch	Domain status: Loading	Version info: OpenSearch 2.11 (latest)	OpenSearch Dashboards URL: (dual stack)
Domain ARN: arn:aws:es:us-east-1:637423408006:domain/opensearch	Preparing to process updates: 0%	Service software version info:	Domain endpoint v2 (dual stack)
		Cluster health info:	Domain endpoint (IPv4)

[Cluster configuration](#) [Security configuration](#) [Cluster health](#) [Instance health](#) [Off-peak window](#) [Auto-Tune](#) [Logs](#) [Indices](#) [Tags](#) [Connections - preview](#) [VPC endpoints](#) [Packages](#)

Cluster configuration

Current configuration		Dry run details
Data nodes	Dedicated master nodes	Network
Availability Zones: 5 AZs without standby	Enabled: No	Access: Public
Instance type: t3.small.search	Warm and cold data storage: UltraWarm data nodes enabled: No	IP address type: IPv4, Dual stack
Number of nodes: 1		

[ClassShell](#) [Feedback](#)

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

By following these steps, you should have successfully created an OpenSearch domain on AWS, configured VPC access, enabled fine-grained access control, and set the access policy to only use fine-grained access control.

IAM ROLE CREATION

1. Create an IAM role for AWS Lambda with permissions for OpenSearch:

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the IAM (Identity and Access Management) service.

IAM > Dashboard

IAM Dashboard

Security recommendations

- Add MFA for root user
- Root user has no active access keys

AWS Account

Account ID: 637423408006
Account Alias: Create
Sign-in URL: https://637423408006signin.aws.amazon.com/console

IAM resources

User groups	Users	Roles	Policies	Identity providers
0	0	5	0	0

What's new

- IAM Access Analyzer now simplifies inspecting unused access to guide you toward least privilege. 7 months ago
- IAM Access Analyzer introduces custom policy checks powered by automated reasoning. 7 months ago
- Announcing AWS IAM Identity Center APIs for visibility into workforce access to AWS. 1 month ago
- New organization-wide IAM condition keys to restrict AWS service-to-service requests. 1 month ago

[View all](#)

Quick Links

My security credentials
Manage your access keys, multi-factor authentication (MFA) and other credentials.

Tools

Policy simulator
The simulator evaluates the policies that you choose and determines the effective permissions for each of the actions that you specify.

Additional Information

[ClassShell](#) [Feedback](#)

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- ❖ In the left sidebar, select "Roles" and then click "Create role."

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with navigation links like Dashboard, Access management, Roles, Policies, Identity providers, and Account settings. The main area displays a table of existing roles, each with a checkbox, a role name, a list of trusted entities, and a last activity timestamp. Below the table, there are sections for 'Roles Anywhere' and 'Temporary credentials'.

- ❖ Choose "AWS service" as the type of trusted entity and select "Lambda."

This screenshot shows the 'Select trusted entity' step in the IAM Role creation wizard. It includes a header 'Step 1 Select trusted entity', a sub-header 'Trusted entity type', and a list of five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this is a 'Use case' section with a dropdown set to 'Lambda'.

- ❖ In the permissions section, attach policies that provide the necessary permissions for your Lambda function to interact with OpenSearch. You might use policies like AmazonESFullAccess or customize permissions based on your specific requirements.



This screenshot shows the 'Add permissions' step in the IAM Role creation wizard. It includes a header 'Step 2 Add permissions', a sub-header 'Permissions policies (1/907)', and a search bar with 'amazones'. A table lists three policies: 'Policy name' (selected), 'AmazonESFullAccess' (selected), and 'AmazonESReadOnlyAccess'. At the bottom, there's a note about setting a permissions boundary.

Name, review, and create

Role details

Role name
Error: a meaningful name to identify this role.

Description
Add a short explanation for this role.

Step 1: Select trusted entities

Trust policy

```

1- {
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Effect": "Allow",
6-       "Action": [
7-         "sts:AssumeRole"
8-       ],
9-       "Principal": [
10-         {
11-           "Service": [
12-             "lambda.amazonaws.com"
13-           ]
14-         }
15-       ]
16-     }
17-   ]
18- }
```

Identity and Access Management (IAM)

Roles (6) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSLambdaRole	AWS Service: opensearchservice[Service]	2 days ago
AWSLambdaVPCAccessRole	AWS Service: ops.apigateway[Service]	-
AWSLambdaBasicExecutionRole	AWS Service: support[Service-Link]	-
AWSLambdaPowerInvokeRole	AWS Service: trustedadvisor[Service]	-
codebuild-opensearch-service-role	AWS Service: codebuild	2 days ago
OpenSearchFullAccess	AWS Service: Lambda	-

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads

Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard

Use your own existing PKI infrastructure or use [AWS Certificate Manager Private Certificate Authority](#) to authenticate identities.

Temporary credentials

Use temporary credentials with ease and benefit from the enhanced security they provide.

- ❖ Complete the role creation, giving it a meaningful name and description.

2. Login to the OpenSearch dashboard:

- ❖ Access the OpenSearch dashboard using the URL provided during the domain creation.

Successfully updated to service software version OpenSearch_2.11_R20231115-P2

[Amazon OpenSearch Service](#) > [Domains](#) > opensearch

opensearch Info

General information

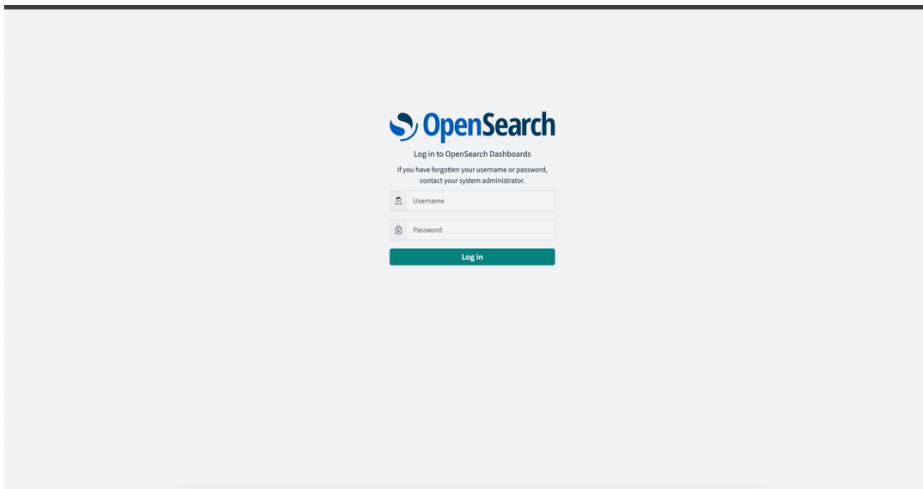
Name opensearch	Domain status Active	Version info OpenSearch 2.11 (latest)	OpenSearch Dashboards URL (dual stack) https://search-opensearch-47a7f1kehzq2ebsoyftz76ri.aos.us-east-1.on.aws
Domain ARN arn:aws:ses:us-east-1:657423408006:domain/opensearch	Cluster health info Green	Service software version info OpenSearch_2.11_R20231115-P2 (latest)	Domain endpoint v2 (dual stack) https://search-opensearch-47a7f1kehzq2ebsoyftz76ri.aos.us-east-1.on.aws

Cluster configuration

Current configuration Dry run details

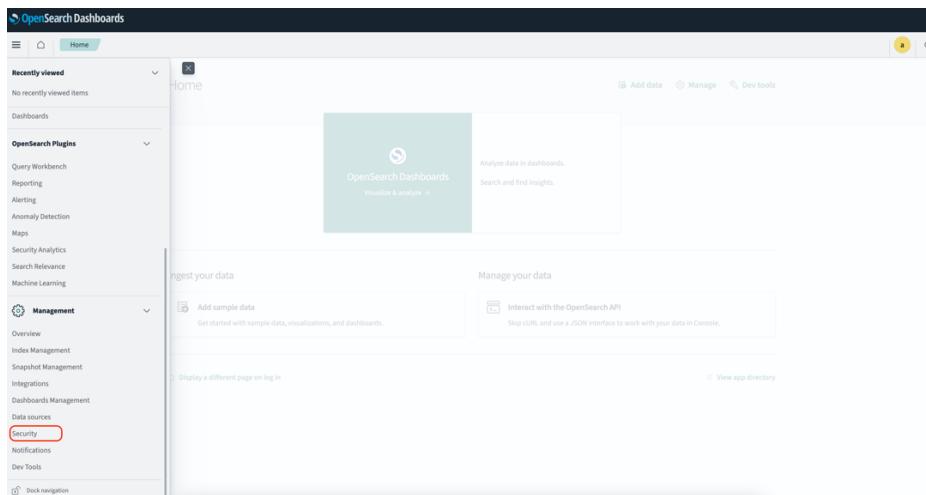
Data nodes **Dedicated master nodes** **Network**

- ❖ Enter the master username and password when prompted.



3. Navigate to Security:

- ❖ In the OpenSearch dashboard, navigate to the "Security" section using the left-hand menu.



4. Select Roles and then choose the all_access role:

- ❖ Within the Security section, select "Roles."

The screenshot shows the 'Get started' section of the OpenSearch Dashboards interface. On the left, a sidebar lists 'Security', 'Get Started', 'Authentication', 'Roles', 'Internal users', 'Permissions', 'Tenants', and 'Audit logs'. The 'Security' tab is selected. The main area has a heading 'Get started' and a sub-section 'Create roles'. It contains two buttons: 'Explore existing roles' (which is highlighted with a red box) and 'Create new role'. Below this is another section 'Map users' with buttons 'Map users to a role' and 'Create internal user'. At the bottom, there are optional sections for 'Audit logs' and 'Purge cache'.

- ❖ Find and click on the all_access role.

The screenshot shows the 'Roles' section of the OpenSearch Dashboards interface. The sidebar on the left is identical to the previous screenshot. The main area has a heading 'Roles' and a sub-section 'Roles (1)'. It shows a table with one row for the 'all_access' role. The table columns are: Role (checkbox), Cluster permissions (dropdown), Index permissions (dropdown), Internal users (dropdown), Backend roles (dropdown), Tenants (dropdown), and Customization (dropdown). The 'all_access' role is highlighted with a red box. The table includes a search bar at the top and a 'Rows per page' dropdown at the bottom right.

5. Choose Mapped users and then Manage mapping:

- ❖ Within the all_access role, select the "Mapped users" tab.

The screenshot shows the OpenSearch Dashboards interface with the 'Security' tab selected. Under 'Roles', the 'all_access' role is chosen. The 'Permissions' section is visible, and the 'Mapped users' tab is currently active. A red box highlights the 'Manage mapping' button in the top right corner of the 'Mapped users' section. Below it, there's a table with one row showing a user named 'admin'. At the bottom left, there's a dropdown for 'Rows per page'.

- ❖ Choose "Manage mapping."

6. Add the Lambda role ARN under Backend roles:

- ❖ Under "Backend roles," click on "Add backend role."

The screenshot shows the 'Map user' screen for the 'all_access' role. The 'Backend roles' section is active, displaying a list with one item: 'arn:aws:iam::637423408006:role/OpenSearchFullAccess'. A red box highlights the 'Add another backend role' button at the bottom left of this section. At the bottom right, there are 'Cancel' and 'Map' buttons.

- ❖ Enter the Amazon Resource Name (ARN) of the Lambda role created in Step 1.
- ❖ Save the changes.

7. Select Map and confirm:

- ❖ After adding the Lambda role ARN, select "Map."

Mapped users (0)
You can map two types of users: users and backend roles. A user can have its own backend role and host for an external authentication and authorization. A backend role directly maps to roles through an external authentication system. Learn more ↗

User type	User
User	admin
Backend role	arn:aws:iam::637423408006:role/OpenSearchFullAccess

Rows per page: 10 < 1 >

- ❖ Confirm that the user or role associated with the Lambda role shows up under "Mapped users."

By completing these steps, you've configured an IAM role for AWS Lambda, granted the necessary permissions for OpenSearch, and mapped the Lambda role to the all_access role in the OpenSearch dashboard. This allows your Lambda function to interact securely with the OpenSearch domain.

LAMBDA FUNCTION (pdftotxt)

1. Lambda Function to Convert PDF to Text (pdftotxt):

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the Lambda service.

Function name	Description	Package type	Runtime	Last modified
pdftotxt	None	None	None	None

- ❖ Click on "Create function."
- ❖ Enter a name for your function (e.g., pdftotxt).

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch
 Start with a simple Hello World example.

Use a blueprint
 Build a Lambda application from sample code and configuration presets for common use cases.

Container image
 Select a container image to deploy for your function.

Basic information

Function name
 Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
 Choose the language to use with your function. Note that the console code editor supports only Node.js, Python, and Ruby.

▼

Architecture Info
 Choose the instance type architecture you want for your function code.
 x86_64
 arm64

Permissions Info
 By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
 Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#) ?
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

- ❖ Choose a runtime (e.g., Python, Node.js, etc.) that supports your PDF to text conversion library.

Successfully created the function pdftotxt. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > pdftotxt

pdftotxt

Throttle Copy ARN Actions ▾

Function overview Info Export to Application Composer Download ▾

Diagram Template

pdftotxt Layers (0)

+ Add trigger + Add destination

Description -

Last modified 15 seconds ago

Function ARN arn:aws:lambda:us-east-1:637423408006:function:pdftotxt

Function URL Info

Code Test Monitor Configuration Aliases Versions

Code source Info Upload from ▾

File Edit Find View Go Tools Window Test Deploy

Go to Anything (N P) Environment

lambda_function Execution results Environment Var

lambda_function.py

```
1 lambda_function:
2     def lambda_handler(event, context):
3         # TODO implement
4         return {
5             'statusCode': 200,
6             'body': json.dumps('Hello from Lambda!')
7         }
```

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Cookie preferences

- ❖ In the "Function code" section, write the code that converts the input PDF file to a text file. Ensure the code includes logic to handle the conversion process.

Successfully updated the function pdftext.

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy Environment

lambda_function.py

```
1 import json
2 import os
3 import logging
4 import pkg_resources
5 import pyPdf
6 from PyPDF2 import PdfReader
7 from io import BytesIO
8 from io import StringIO
9
10
11 logger = logging.getLogger()
12 logger.setLevel(logging.INFO)
13
14
15
16 def lambda_handler(event, context):
17     logger.info(f'Event environment variables are {event["env"]}')
18     logger.info(f'Context environment variables are {context["env"]}')
19     logger.info(f'Extracted event {event}')
20     extract_content(event)
21
22     return {
23         "statusCode": 200,
24         "body": json.dumps("Execution is now complete")
25     }
```

Execution results Environment Var.

Upload from

Environment

Code properties [Info](#)

Package size SHA256 hash Last modified

414.0 byte vpt1Ky6FavafP92hYajJl8AdUJv93etKJkpcOk9Epon January 14, 2024 at 06:48 PM GMT+5:30

CloudWatch Feedback © 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. Release Terms Cookie preferences

- ❖ Add a layer for the dependencies and upload the packages to the layer.

Lambda > Layers > Add layer

Add layer

Function runtime settings

Runtime: Python 3.9 Architecture: x86_64

Choose a layer

Layer source: [Info](#) Choose a layer with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also create a new layer.

AWS layers Choose a layer from a list of layers provided by AWS.

Custom layers Choose a layer from a list of layers created by your AWS account or organization.

Specify an ARN Specify a layer by providing the ARN.

Custom layers Layers created by your AWS account or organization that are compatible with your function's runtime.

pdfText

Version

1

Add

2. Configure Trigger for S3 PUT Event:

- ❖ Add the bucket in S3 (name : storage0909) buckets of AWS.

Storage

Amazon S3

Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Create bucket

How it works

Pricing

With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of your S3 bucket.

Estimate your monthly bill using the [AWS Simple Monthly Calculator](#)

[View pricing details](#)

Resources

User guide
API reference
FAQs
Discussion forums

Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

AWS Region: US East (Virginia) us-east-1

Bucket type: [Info](#)

General purpose Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They're designed to store objects across multiple Availability Zones.

Directory - New Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides fast processing of data within a single Availability Zone.

Bucket name: [Info](#) storage0909

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See notes for bucket naming](#)

Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: `s3://bucketprefix`

Object Ownership: [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLS disabled (recommended) All objects in this bucket are owned by this account. Access to the bucket and its objects is specified using only policies.

ACLS enabled Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

The screenshot shows the AWS S3 console with a green header bar indicating "Successfully created bucket 'storage0909'". Below the header, there's an "Account snapshot" section with a link to "View Storage Lens dashboard". Under "General purpose buckets", there is one entry: "storage0909" (info), created on January 14, 2024, at 18:35:33 (UTC+05:30). The "Create bucket" button is visible in the top right.

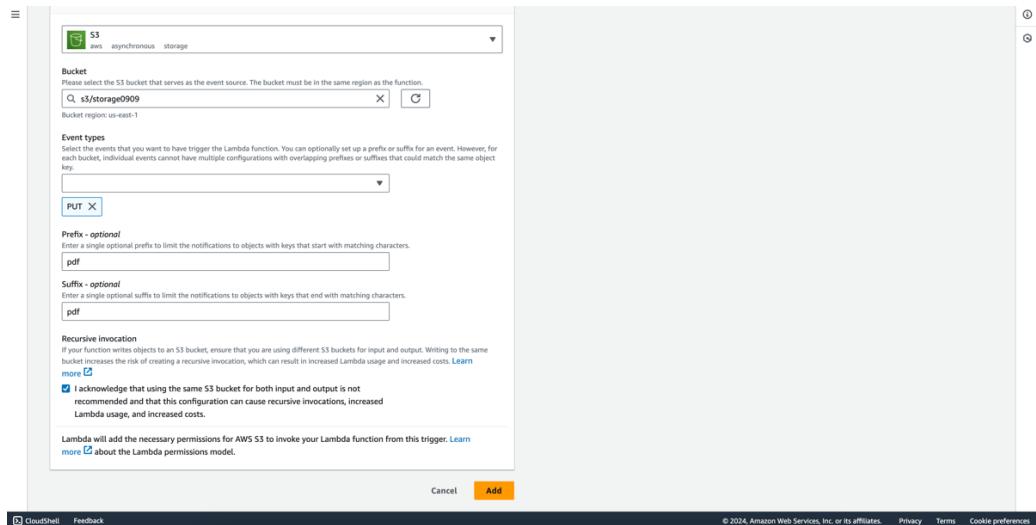
- ❖ In the Lambda function configuration, add a trigger.

The screenshot shows the AWS Lambda function configuration page for "pdftotxt". The "Function overview" tab is selected. On the left, there's a "Diagram" view showing a single function node labeled "pdftotxt" with a "Layers" dependency. A red box highlights the "+ Add trigger" button. On the right, there's a "Description" panel with details like Last modified 3 minutes ago, Function ARN, and Function URL. The "Code source" tab is also visible at the bottom.

- ❖ Choose "S3" as the trigger source.

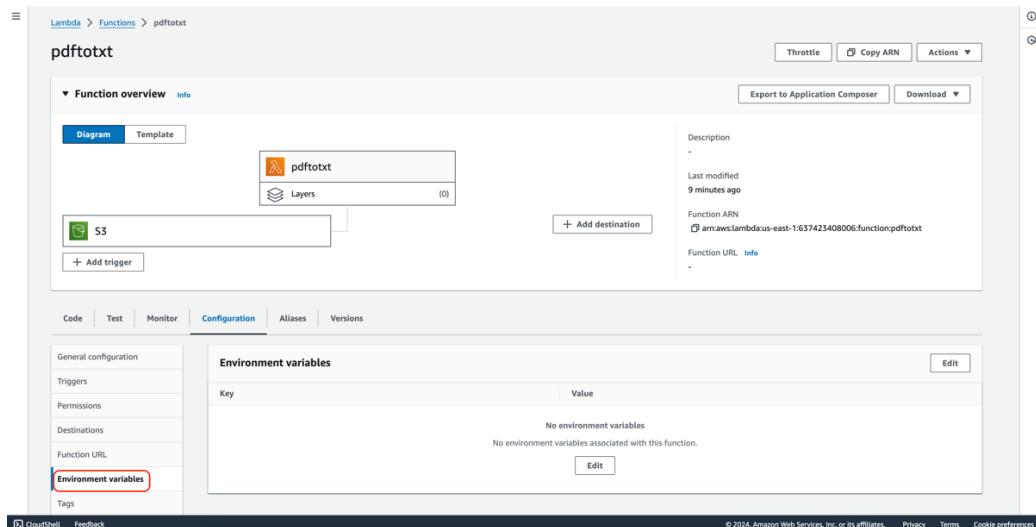
The screenshot shows the "Add trigger" dialog box. The "Trigger configuration" tab is selected. In the "Select a source" dropdown, "s3" is typed and selected from the list. The "Batch/bulk data processing" section is visible below. At the bottom, there are "Cancel" and "Add" buttons.

- ❖ Configure the trigger to listen to the PUT event on the S3 bucket where the sample PDF file is uploaded (content store S3 bucket).



3. Configure Environmental Variable:

- ❖ In the Lambda function configuration, go to the "Configuration" tab.



- ❖ Scroll down to the "Environment variables" section.

The screenshot shows the AWS Lambda Function Overview page for a function named 'pdftotxt'. The function has an S3 trigger and no layers. It shows environment variables TARGET_BUCKET set to 'storage0909'. The ARN is arn:aws:lambda:us-east-1:637423408006:function:pdftotxt.

- ❖ Add a variable named TARGET_BUCKET with the value set to the name of the S3 bucket used as intermediary storage.

4. Assign IAM Role:

- ❖ In the Lambda function configuration, go to the "Execution role" section.

The screenshot shows the AWS IAM Roles page. A policy named 'OpenSearchFullAccess' is attached to a role. The policy grants full access to OpenSearch and AmazonS3.

Policy name	Type	Attached entities
AmazonESFullAccess	AWS managed	1
AmazonOpenSearchServiceFullAccess	AWS managed	1
AmazonS3FullAccess	AWS managed	1

- ❖ Choose an existing role or create a new one that has the necessary permissions to access S3 and any other services required for the conversion process.
- ❖ Ensure that the role allows the Lambda function to read from the source S3 bucket and write to the target S3 bucket.
- ❖ Save the changes.
- ❖ Add the pdf to the S3 Bucket.

Amazon S3 > Buckets > storage0909

storage0909 info

Objects (1) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
Astronomy.pdf	pdf	January 14, 2024, 18:57:41 (UTC+05:30)	26.7 KB	Standard

- ❖ Now configure the test case and click on TEST.
- ❖ Now, check the file in S3 bucket named “storage0909”.

The test event test was successfully saved.

Code Test Monitor Configuration Aliases Versions

Code source Info

File Edit Find View Go Tools Window Test Deploy

lambda_function

Execution results

Creation date is: None

Content is:

```
Astronomy
None
None
In the Ptolemaic model, Venus lies between the Earth and the Sun and hence it must always be lit from behind, so could only show crescent phases whilst its angular size would not alter greatly. In contrast, in the Copernican model Venus orbits the Sun and, when on the side of the Sun, it would show almost full phases whilst, when on its far side but still visible, it would show almost full phases. As its distance from us would change significantly, its angular size (the angle subtended by the planet as seen from the Earth) would likewise change over a large range. Galileo's first set of drawings, made by Galileo with a simple refracting telescope. They are shown in parallel with a set of modern photographs which illustrate not only that Galileo showed the phases, but that he also drew the correct sequence of phases. This sequence shows the phases as the Copernican model predicts: almost full phases when Venus is on the far side of the Sun and a small angular size coupled with thin crescent phases, having a significant change in angular size over a short time period. Galileo's observations, made with the simplest possible astronomical instrument, were able to show which of the two competing models of the Solar System was correct. Using only optical instruments and no sophisticated instruments, astronomers have been able to choose between competing theories of the Universe - a story that will be told in Chapter 9.
```

All done

REPORT RequestId: 00b1b345-b716-49f4-8a05-15cfdfab064f Duration: 1449.83 ms Billed Duration: 1450 ms Memory Size: 128 MB Max Memory Used: 94 MB

Request ID: 00b1b345-b716-49f4-8a05-15cfdfab064f

aws Services Search [Option+S] Global ▾

Amazon S3

Buckets

Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards
Storage Lens groups
AWS Organizations settings

Feature spotlight ?

AWS Marketplace for S3

Amazon S3 > Buckets > storage0909

storage0909 info

Objects (2) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
Astronomy.pdf	pdf	January 14, 2024, 18:57:41 (UTC+05:30)	26.7 KB	Standard
Astronomy.pdf.txt	txt	January 14, 2024, 19:02:26 (UTC+05:30)	1.5 KB	Standard

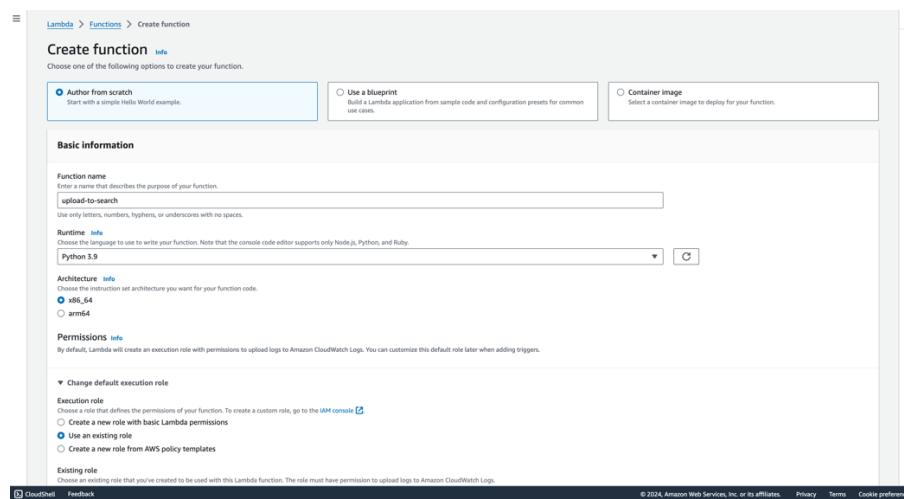
The file has been converted from “pdf” format to “txt” format.

By following these steps, you should have a Lambda function (`pdftotxt`) configured to convert a PDF file to text, triggered by an S3 PUT event, and with the necessary environmental variable and IAM role configured.

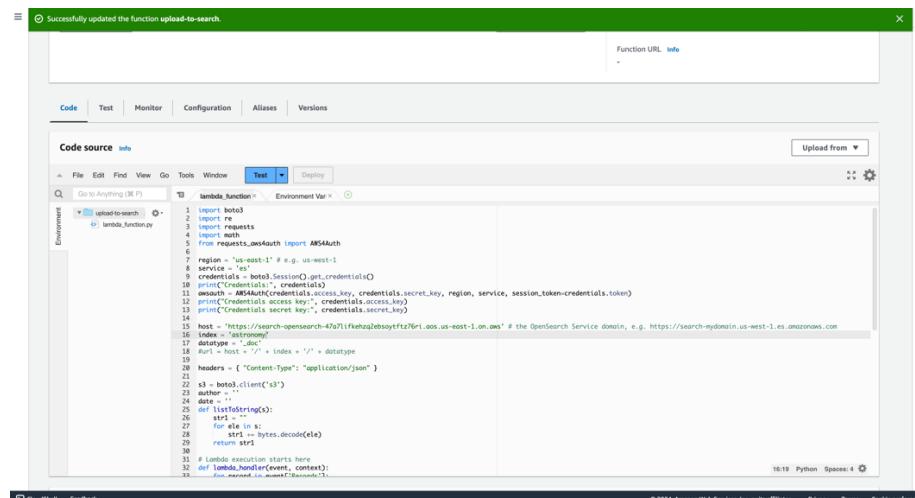
LAMBDA FUNCTION (upload-to-search)

1. Lambda Function to Upload Text to OpenSearch (upload-to-search):

- ❖ Log in to the AWS Management Console.
 - ❖ Navigate to the Lambda service.
 - ❖ Click on "Create function."

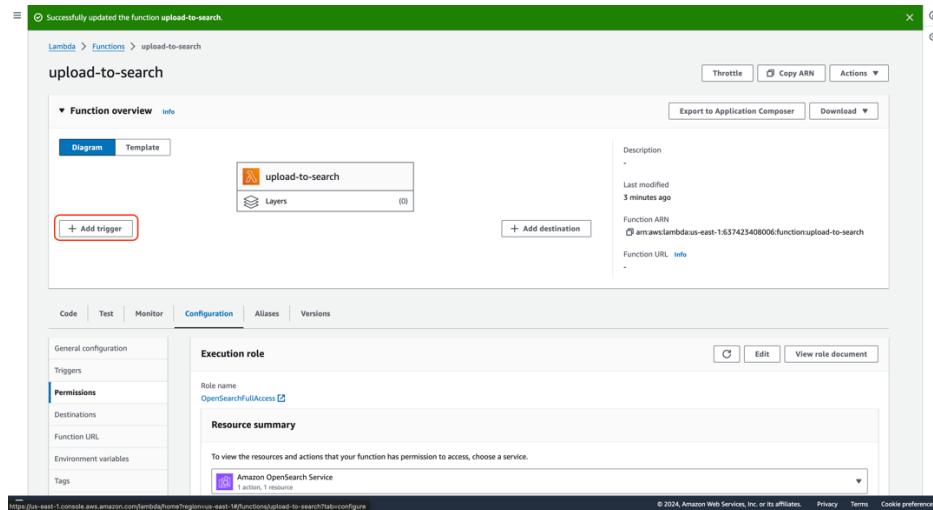


- ❖ Enter a name for your function (e.g., upload-to-search).
 - ❖ Choose the same runtime used in the previous Lambda function (e.g., Python).
 - ❖ In the "Function code" section, upload the sample.py file and ensure that it contains a function named handler that processes the text and uploads it to OpenSearch.

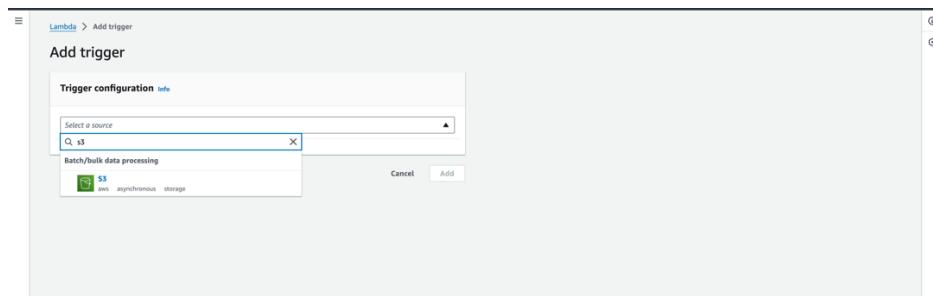


2. Configure Trigger for S3 PUT Event:

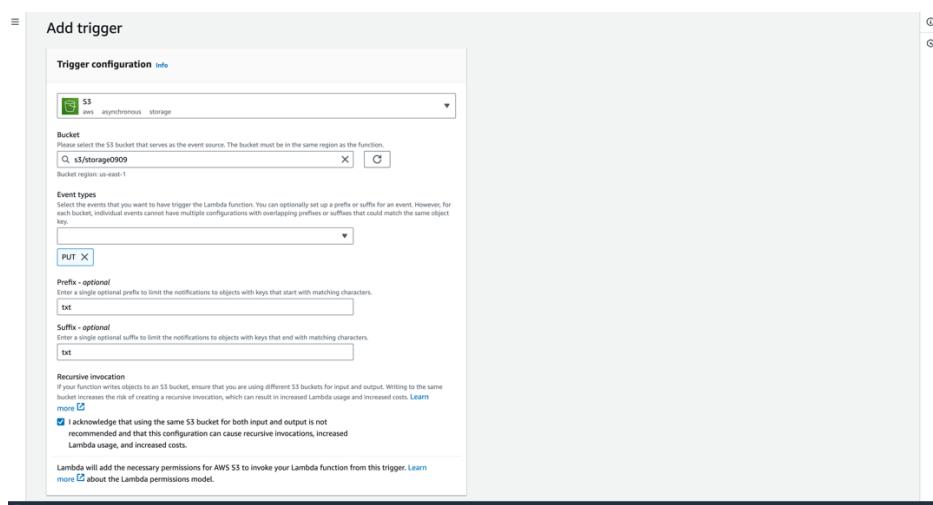
- ❖ In the Lambda function configuration, add a trigger.



- ❖ Choose "S3" as the trigger source.



- ❖ Configure the trigger to listen to the PUT event on the S3 bucket used as intermediary storage.

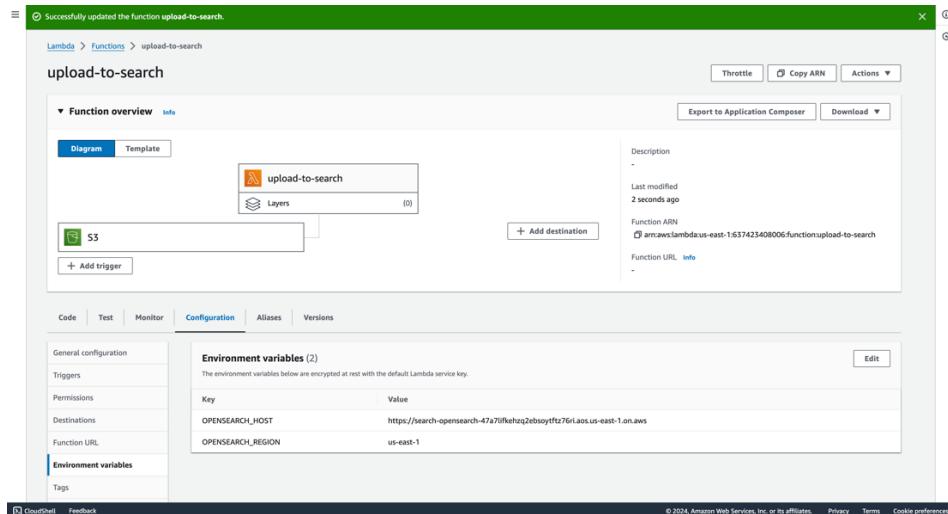


3. Configure Runtime Settings:

- ❖ In the Lambda function configuration, go to the "Configuration" tab.
- ❖ Set the "Handler" field to sample.handler (assuming sample.py contains a handler function).

4. Configure Environmental Variables:

- ❖ In the Lambda function configuration, go to the "Environment variables" section.

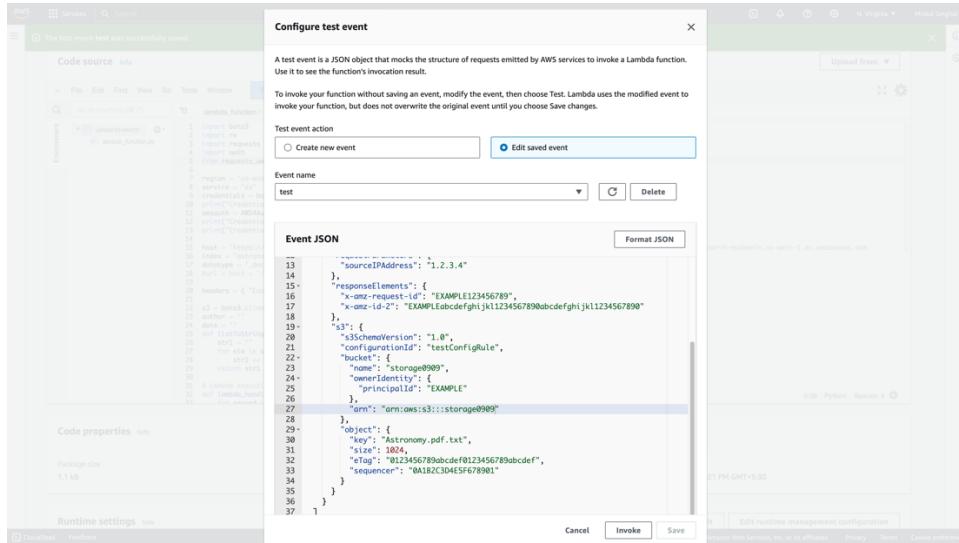


- ❖ Add variables named OPENSEARCH_REGION and OPENSEARCH_HOST with their respective values.

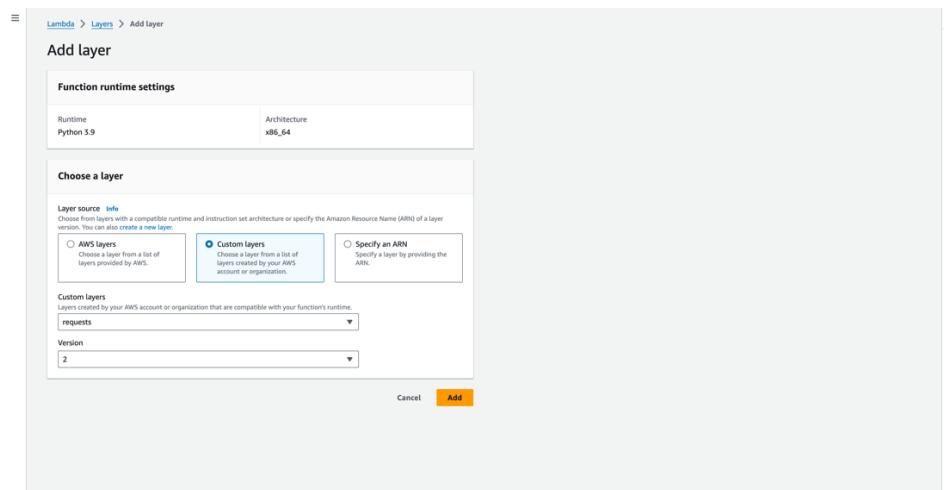
5. Assign IAM Role:

- ❖ In the Lambda function configuration, go to the "Execution role" section.
- ❖ Choose an existing role or create a new one that has the necessary permissions to access S3 and OpenSearch.
- ❖ Ensure that the role allows the Lambda function to read from the S3 bucket used as intermediary storage and interact with OpenSearch.
- ❖ Save the changes.

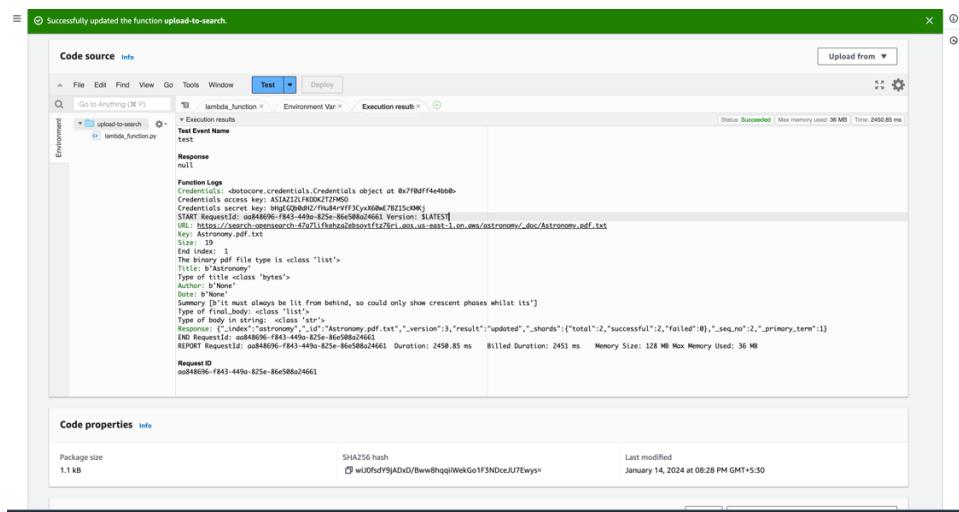
- ❖ Configure the test for the lambda function.



- ❖ Add the dependency layer containing the “requests” dependency.



- ❖ Now execute the function by clicking on Test.



- ❖ An index named “astronomy” will automatically be created in OpenSearch Platform.

The screenshot shows the OpenSearch Dashboards interface under the 'Indices' tab. On the left, there's a sidebar with sections like 'State management policies', 'Indices' (which is selected), 'Policy managed indices', 'Data streams', 'Templates', 'Aliases', 'Rollup jobs', 'Transform jobs', and 'Notification settings'. The main area is titled 'Indices (6)' and contains a table with columns: Index, Health, Managed by policy, Status, Total size, Size of primaries, Total documents, Deleted documents, Primaries, and Replicas. The 'astronomy' index is listed with a green health status, no policy applied, open status, 65.2kb total size, 32.6kb primary size, 1 document, 2 deleted documents, 1 primary, and 1 replica. Other indices listed include '_ql-datasources', '_plugins-ml-config', '.opensearch-observability', '.opendistro_security', and '_kibana_1'. There are buttons for 'Refresh', 'Actions', and 'Create Index' at the top right, and a search bar and 'Show data stream indices' link above the table. A note at the bottom says 'Rows per page: 20'.

`{"_index": "astronomy", "_id": "Astronomy.pdf.txt", "version": 3, "seq": 0, "primary_term": 1, "found": true, "source": {"title": "Astronomy", "Author": "None", "Date": "None", "Body": "In the Ptolemaic model, Venus lies between the Earth and the Sun and hence it must always be lit from behind, so could only show crescent phases whilst its angular size would not alter greatly. In contrast, in the Copernican model Venus orbits the Sun. When on the nearside of the Sun, it would show crescent phases whilst, when on its far side but still visible, it would show almost full phases. As its distance from us would change significantly, its angular size (the angle subtended by the planet as seen from Earth) would also change. Galileo's drawings of Venus, made with a simple telescope, illustrate not only that Galileo showed the phases, but that he also drew the changing angular size correctly. These drawings showed precisely what the Copernican model predicts: almost full phases when Venus is on the far side of the Sun and a small angular size coupled with thin crescent phases, having a significantly larger angular size when it is closest to the Earth. Galileo's observations, made with the simplest possible astronomical instruments, were able to confirm which of the two competing models of the Solar System was correct. In just the same way, but using vastly more sophisticated instruments, astronomers have been able to choose between competing theories of the universe. (See a story that will be told in Chapter 9, 'Summary': 'It must always be lit from behind, so could only show crescent phases whilst its')"}'`

By following these steps, you should have a Lambda function (upload-to-search) configured to process the text file generated by the previous Lambda function (pdftotxt) and upload the text to OpenSearch. The function is triggered by an S3 PUT event, and the necessary runtime settings, environmental variables, and IAM role are configured.

LAMBDA FUNCTION (search-gateway)

1. Lambda Function for Search Gateway (search-gateway):

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the Lambda service.
- ❖ Click on "Create function."

The screenshot shows the 'Create function' wizard. It starts with a summary step: 'Choose one of the following options to create your function.' There are three options: 'Author from scratch' (selected), 'Use a Blueprint' (with a note about common use cases), and 'Container image' (with a note about selecting a container image). The next step is 'Basic information': 'Function name' (set to 'search-gateway'), 'Runtime' (set to 'Python 3.9'), 'Architecture' (set to 'x86_64'), and 'Permissions' (set to 'OpenSearchFullAccess'). Below these are sections for 'Change default execution role' (with options for creating a new role or using an existing one), 'Existing role' (set to 'OpenSearchFullAccess'), and 'CloudWatch Logs' (with a note about CloudWatch Logs permissions). At the bottom, there are links for 'CloudShell', 'Feedback', and 'View the OpenSearchFullAccess role on the IAM console.'

- ❖ Enter a name for your function (e.g., search-gateway).
- ❖ Choose the appropriate runtime (e.g., Python).

The screenshot shows the AWS Lambda Function Configuration page for the 'search-gateway' function. The top bar indicates 'Successfully updated the function search-gateway.' Below the title, there's a summary card with the function name, a 'Layers' section (0), and a timestamp 'Last modified 1 minute ago'. It also shows the ARN 'arn:aws:lambda:us-east-1:637423408006:function:search-gateway' and a 'Function URL' link.

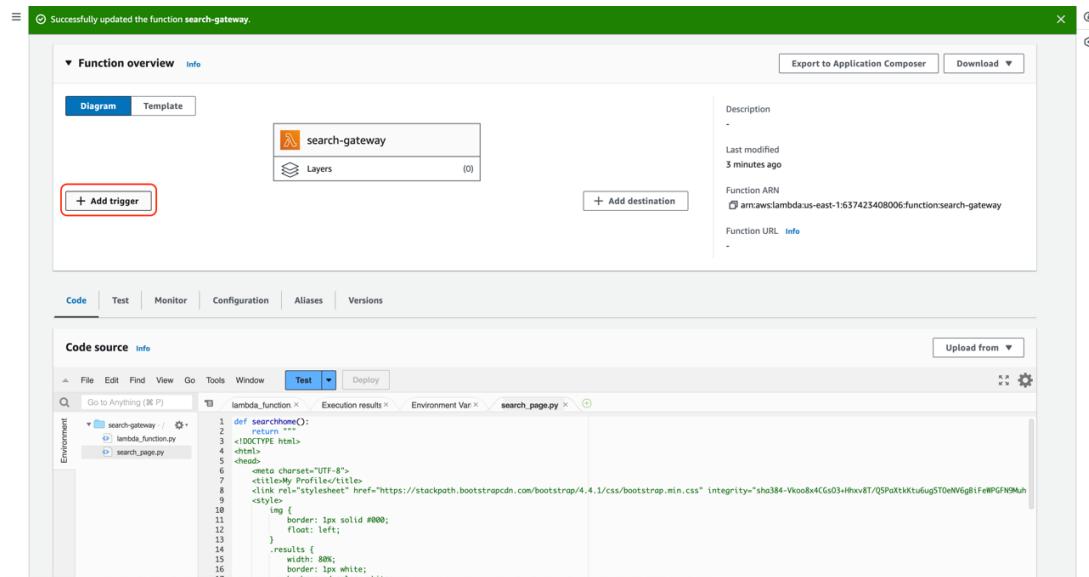
The main area is divided into tabs: Code, Test, Monitor, Configuration, Aliases, and Versions. The Code tab is selected, showing the code editor with the file 'lambda_function.py' open. The code defines a lambda_handler function that imports search_page.json, handles an event and context, sets a status code of 200, and returns a response with headers and a body from search_page.searchHome().

Below the code editor, there's a 'Code properties' section showing the package size (1.9 kB) and SHA256 hash (OQoCpZiWak8M+8rDWfAfn09zqjioNL7rh8TK1MwSYfk=). The last modified date is January 14, 2024, at 08:34 PM GMT+5:30.

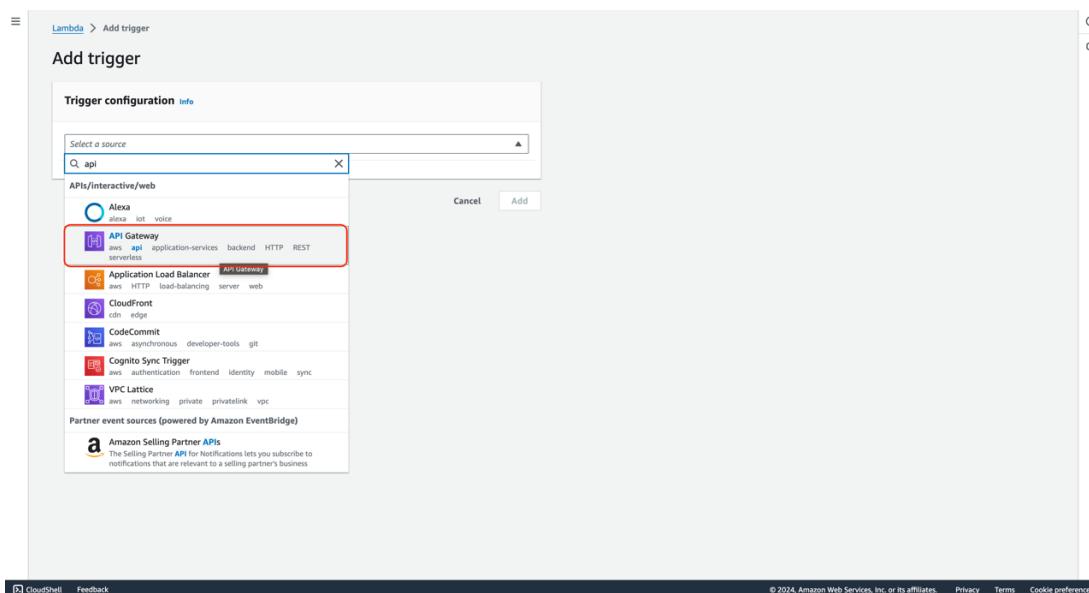
- ❖ In the "Function code" section, upload the sample.py file and ensure that it contains a function named `lambda_handler` to handle incoming HTTP requests and run search queries on OpenSearch.

2. Configure Trigger for API Gateway:

- ❖ In the Lambda function configuration, add a trigger.



- ❖ Choose "API Gateway" as the trigger source.



- ❖ Configure the trigger to receive requests from the API Gateway created as part of this exercise.

Introducing the new API Gateway console experience
We've redesigned the API Gateway console for REST APIs and WebSocket APIs. Let us know what you think.

API Gateway > APIs > Create API > Create

Step 1 Create an API

Step 2 - optional Configure routes

Step 3 - optional Define stages

Step 4 Review and Create

Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations (1) Info

Lambda	AWS Region	Lambda function	Version
arn:aws:lambda:us-east-1:657423408123	us-east-1	arnaws:lambda:us-east-1:657423408123	Learn more. 2.0

Add integration

API name
An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

Cancel Review and Create Next

Introducing the new API Gateway console experience
We've redesigned the API Gateway console for REST APIs and WebSocket APIs. Let us know what you think.

API Gateway > APIs > Create API > Create

Step 1 Create an API

Step 2 - optional Configure routes

Step 3 - optional Define stages

Step 4 Review and Create

Configure routes - optional

Configure routes Info

API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method	Resource path	Integration target
POST	/search-gateway-post	search-gateway
GET	/search-gateway-get	search-gateway

Add route

Cancel Review and Create Previous Next

Introducing the new API Gateway console experience
We've redesigned the API Gateway console for REST APIs and WebSocket APIs. Let us know what you think.

API Gateway > APIs > Create API > Create

Step 1 Create an API

Step 2 - optional Configure routes

Step 3 - optional Define stages

Step 4 Review and Create

Review and Create

API name and integrations

API name

- search-gateway-api

Integrations

- search-gateway (Lambda)

Routes

Routes

- POST /search-gateway-post --> search-gateway (Lambda)
- GET /search-gateway-get --> search-gateway (Lambda)

Stages

Stages

- \$default (Auto-deploy: enabled)

Cancel Previous Create

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'Lambda > Functions > search-gateway'. Below it is the function name 'search-gateway'. On the right side, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. Further down are 'Export to Application Composer' and 'Download' buttons.

The main area features a 'Diagram' tab which displays a flowchart: 'search-gateway' leads to 'Layers (0)' and then to 'API Gateway (2)'. There are buttons for '+ Add destination' and '+ Add trigger'. Below the diagram, tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions' are visible. The 'Configuration' tab is currently active.

In the 'Configuration' section, the left sidebar lists 'General configuration', 'Triggers' (which is selected), 'Permissions', 'Destinations', 'Function URL', 'Environment variables', 'Tags', and 'VPC'. The 'Triggers' section shows two entries:

- Trigger**: API Gateway: search-gateway-api (arn:aws:execute-api:us-east-1:637423408006:5pbxk6ly1m/*/*/search-gateway-post)
- Trigger**: API Gateway: search-gateway-api (arn:aws:execute-api:us-east-1:637423408006:5pbxk6ly1m/*/*/search-gateway-get)

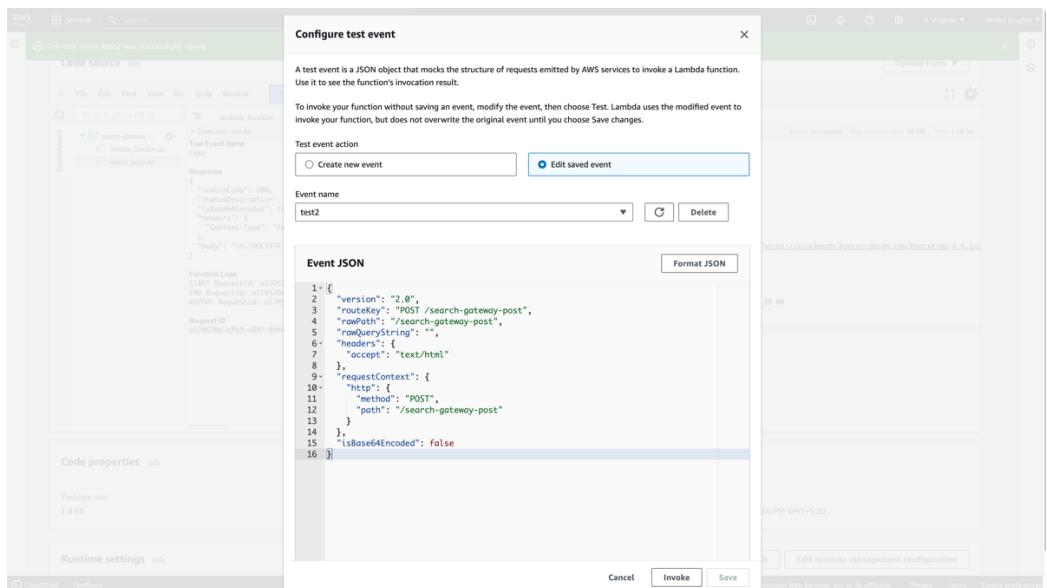
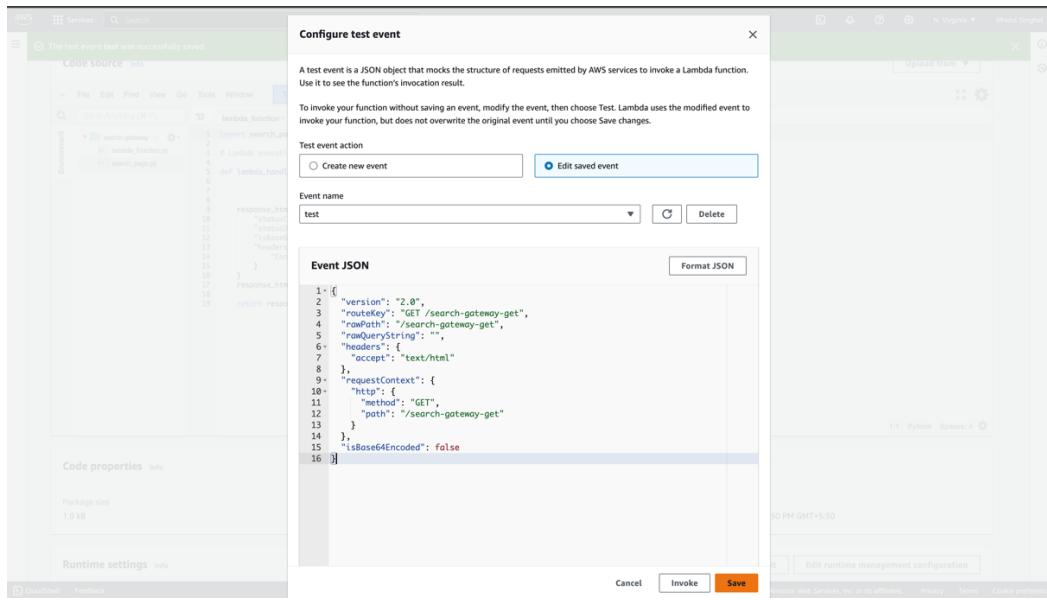
Each trigger entry includes an 'Edit' button and a 'Details' link.

3. Configure Runtime Settings:

- ❖ In the Lambda function configuration, go to the "Configuration" tab.
- ❖ Set the "Handler" field to sample.lambda_handler (assuming sample.py contains a lambda_handler function).

4. Assign IAM Role:

- ❖ In the Lambda function configuration, go to the "Execution role" section.
- ❖ Choose an existing role or create a new one that has the necessary permissions to access OpenSearch and handle API Gateway requests.
- ❖ Ensure that the role allows the Lambda function to interact with OpenSearch and handle incoming API Gateway requests.
- ❖ Save the changes.
- ❖ Configure the test settings.



- ❖ Now run the tests by executing the functions with the test configurations.

The screenshot shows the AWS Lambda function configuration page. The 'Test' tab is selected. A success message at the top states: "The test event test2 was successfully saved." Below this, there's a "Test trigger" button. The main area displays the "Code source" tab with the "Info" sub-tab selected. It shows the file structure: search-gateway / lambda_function.py / search_page.py. The "Execution results" section shows a successful test event named "test2" with the following response:

```

{
    "statusCode": 200,
    "statusDescription": "200 OK",
    "isBase64Encoded": false,
    "headers": {
        "Content-Type": "text/html; charset=utf-8"
    },
    "body": "<html><head><meta charset='UTF-8'><title>My Profile</title><link rel='stylesheet' href='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/c
}

```

Below the response, the "Function Logs" section shows the request and response details.

By following these steps, you should have a Lambda function (search-gateway) configured to process incoming HTTP requests from the API Gateway, run search queries on OpenSearch, and return the search results. The function is triggered by API requests, and the necessary runtime settings and IAM role are configured.

LAMBDA FUNCTION (searchFunction)

1. Lambda Function for Search Function (searchFunction):

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the Lambda service.
- ❖ Click on "Create function."

The screenshot shows the "Create function" wizard. The first step, "Choose one of the following options to create your function.", has three options: "Author from scratch" (selected), "Use a blueprint", and "Container image".

The "Basic information" step follows. It includes fields for "Function name" (set to "searchFunction"), "Runtime" (set to "Python 3.9"), and "Architecture" (set to "x86_64").

The "Permissions" step shows a dropdown for "Change default execution role" with options: "Create a new role with basic Lambda permissions" (selected) and "Create a new role from AWS policy templates".

The "Existing role" step shows a note: "Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs." There are no roles listed in this screenshot.

- ❖ Enter a name for your function (e.g., searchFunction).
- ❖ Choose the appropriate runtime (e.g., Python).

- ❖ In the "Function code" section, write the code to handle the API request, process the search query, and return the search results.

2. Configure Trigger for API Gateway:

- ❖ In the Lambda function configuration, add a trigger.

Lambda > Functions > searchFunction

searchFunction

Throttle Copy ARN Actions ▾

Export to Application Composer Download ▾

Diagram Template

Layers (1)

API Gateway + Add destination

+ Add trigger

Description

Last modified 12 seconds ago

Function ARN arn:aws:lambda:us-east-1:637423408006:function:searchFunction

Function URL Info

Code Test Monitor Configuration Aliases Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

Triggers (1) Info

Find triggers

Trigger API Gateway: search-gateway-api

arn:aws:execute-api:us-east-1:637423408006:Sphxk6lYm/*/*/searchFunction

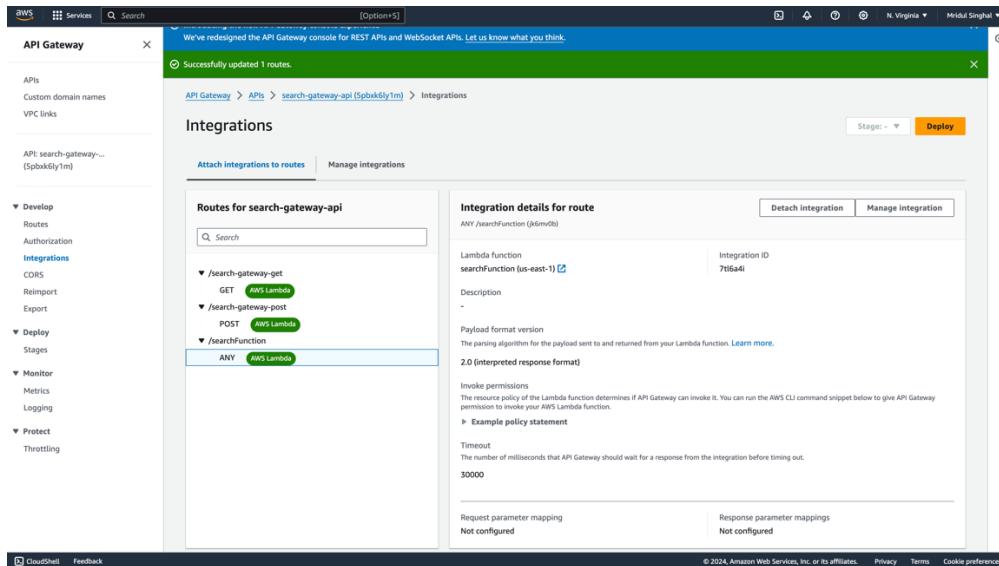
API endpoint: <https://Sphxk6lYm.execute-api.us-east-1.amazonaws.com/searchFunction>

> Details

Fix errors Edit Delete Add trigger < 1 >

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- ❖ Choose "API Gateway" as the trigger source.
 - ❖ Configure the trigger to receive requests from the API Gateway created as part of this exercise.

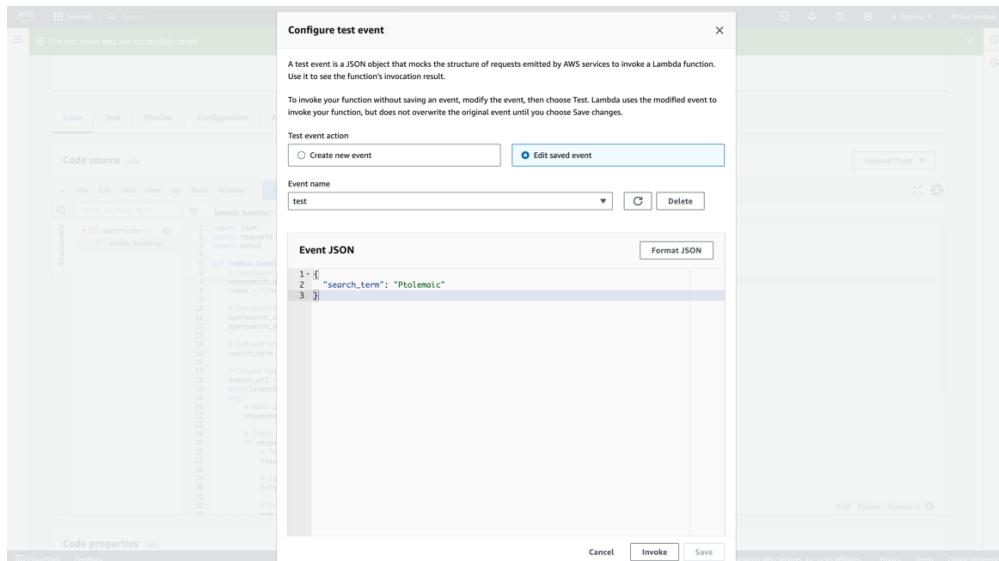


3. Configure Runtime Settings:

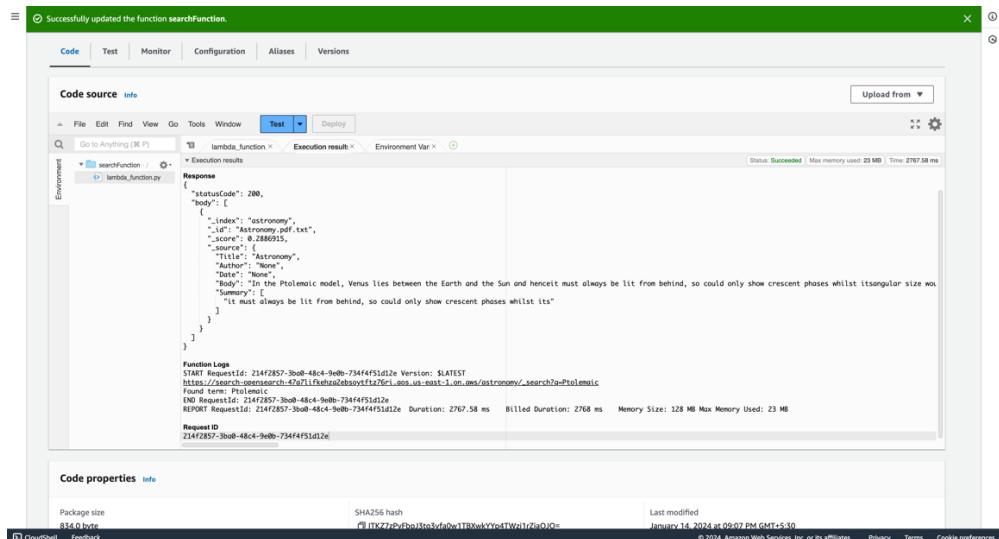
- ❖ In the Lambda function configuration, go to the "Configuration" tab.
- ❖ Set the "Handler" field to searchFunction.lambda_handler (assuming searchFunction.py contains a lambda_handler function).

4. Assign IAM Role:

- ❖ In the Lambda function configuration, go to the "Execution role" section.
- ❖ Choose an existing role or create a new one that has the necessary permissions to access OpenSearch.
- ❖ Ensure that the role allows the Lambda function to interact with OpenSearch and handle incoming API Gateway requests.
- ❖ Save the changes.
- ❖ Configure the test settings of the function.



- ❖ Now run the tests by clicking on the Test button.



By following these steps, you should have a Lambda function (`searchFunction`) configured to receive data entered into the search field through API requests from the API Gateway. The function processes the search query, queries OpenSearch, and returns the search results. The function is triggered by API requests, and the necessary runtime settings and IAM role are configured.

SETTING UP CODE COMMIT

1. Create a CodeCommit Repository using the AWS Console:

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the CodeCommit service.

The screenshot shows the AWS CodeCommit interface. On the left, a sidebar menu includes options like Source, Getting started, and Repositories. The main area displays a table titled 'Repositories' with columns for Name, Description, Last modified, Clone URL, and AWS KMS Key. A search bar is at the top, and a message below the table says 'No results' and 'There are no results to display.' At the bottom right of the main area is a prominent orange 'Create repository' button.

❖ Click on "Create repository."

The screenshot shows the 'Create repository' wizard. The first step, 'Repository settings', is displayed. It includes fields for 'Repository name' (containing 'opensearch'), 'Description - optional' (with a placeholder for tags), and 'Tags' (with an 'Add tag' button). Below these are sections for 'Additional configuration' (including 'AWS KMS key') and 'Enable Amazon CodeGuru Reviewer for Java and Python - optional'. At the bottom are 'Cancel' and 'Create' buttons.

❖ Enter a repository name and configure other settings as needed.

The screenshot shows the AWS CodeCommit interface for creating a new repository named 'opensearch'. The left sidebar contains navigation links for developer tools like CodeCommit, CodeArtifact, CodeBuild, CodeDeploy, CodePipeline, and Settings. The main content area displays a success message: 'Repository successfully created'. It includes sections for 'Connection steps' (HTTPS, SSH, HTTPS (GRPC)), 'Prerequisites' (warning about root account usage), 'Step 1: Prerequisites' (Git client requirements), 'Step 2: Set up the AWS CLI Credential Helper' (AWS CLI version requirement), and 'Additional details' (documentation link). A file list for 'opensearch' is shown with an 'Add file' button. The bottom of the page includes standard AWS footer links for CloudShell, Feedback, and copyright information.

❖ Click "Create repository."

2. Create an IAM User with CodeCommit Credentials:

❖ Navigate to the IAM service in the AWS Console.

The screenshot shows the AWS IAM 'Users' management page. The left sidebar includes links for Identity and Access Management (IAM), Access management (User groups, Roles, Policies, Identity providers, Account settings), Access reports (Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies (SCPs)), and Related consoles (IAM Identity Center, AWS Organizations). The main content area shows a table titled 'Users (0) Info' with a note: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' The table has columns for User name, Path, Group, Last activity, MFA, Password age, Console last sign-in, and Access key ID. A search bar and a 'Create user' button are at the top right. The bottom of the page includes standard AWS footer links for CloudShell, Feedback, and copyright information.

❖ Create a new IAM user with programmatic access.

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

admin

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = _ (hyphen)

Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

❖ Attach the necessary permissions for CodeCommit (e.g., AWSCodeCommitPowerUser policy).

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1171)

Choose one or more policies to attach to your new user.

Policy name	Type	Attached entities
<input type="checkbox"/> AWSCodeCommitFullAccess	AWS managed	0
<input checked="" type="checkbox"/> AWSCodeCommitPowerUser	AWS managed	0
<input type="checkbox"/> AWSCodeCommitReadOnly	AWS managed	0

[Filter by Type](#)

[Create policy](#)

Set permissions boundary - optional

Cancel Previous Next

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access Analyzer

External access

Unused access

Analyzer settings

Credential report

Organization activity

Service control policies (SCPs)

Related consoles

IAM Identity Center

AWS Organizations

IAM > Users > admin

Summary

ARN: arn:aws:iam:637423408006:user/admin

Console access: Disabled

Created: January 15, 2024, 00:22 (UTC+05:30)

Last console sign-in: -

Access key 1: Create access key

Permissions

Permissions policies (1)

Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
<input type="checkbox"/> AWSCodeCommitPowerUser	AWS managed	Directly

[Filter by Type](#)

[Remove](#) [Add permissions](#)

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

[Generate policy](#)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

❖ Note down the access key ID and secret access key generated for this user.

The screenshot shows the AWS IAM Access Keys page. The left sidebar includes sections like Identity and Access Management (IAM), Dashboard, Access management (User groups, Roles, Policies, Identity providers, Account settings), Access reports (Access Analyzer, External access, Unused access, Analyzer settings), Credential report, Organization activity, and Service control policies (SCPs). The main content area shows the 'Access keys (0)' section with a note about best practices. Below it is the 'SSH public keys for AWS CodeCommit (0)' section. The 'HTTPS Git credentials for AWS CodeCommit (0)' section is highlighted with a red box. It contains a table with columns 'User name' and 'Created'. A 'Generate credentials' button is present. The 'Credentials for Amazon Keypairs (for Apache Cassandra) (0)' section is also visible below. The bottom right corner shows copyright information: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

This screenshot shows the 'Generate credentials' dialog box for the HTTPS Git credentials section. The dialog title is 'Generate credentials' and it contains a message: 'Your new credentials are available.' Below this, there is a note: 'Save your user name and password or download the credentials file.' It explains that this is the only time the password can be viewed or downloaded. The dialog shows the user name 'admin-at-637423408006' and a masked password. There are 'Download credentials' and 'Close' buttons at the bottom. The background shows the rest of the IAM Access Keys page with the 'SSH public keys for AWS CodeCommit (0)' and 'Credentials for Amazon Keypairs (for Apache Cassandra) (0)' sections.

- ❖ Copy the credentials and save them somewhere safe.

The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, there's a sidebar with navigation links like Dashboard, Access management, Access reports, and Related consoles. The main content area is titled "SSH public keys for AWS CodeCommit (0)". It has a "Create access key" button. Below it is a table with columns "SSH Key ID", "Uploaded", and "Status". A red box highlights a section titled "HTTPS Git credentials for AWS CodeCommit (1)". This section contains a table with columns "User name", "Created", and "Status". It shows one entry: "admin-at-637423408006" created "Now" and marked as "Active". There are "Actions" and "Generate credentials" buttons. Below this is another section titled "Credentials for Amazon Keyspaces (for Apache Cassandra) (0)" with a "Generate credentials" button. At the bottom, there's a "X.509 Signing certificates (0)" section with a "Create X.509 certificate" button. The footer includes links for CloudShell, Feedback, and copyright information: © 2024, Amazon Web Services, Inc. or its affiliates.

- ❖ The user has been generated for using git on my local machine.

3. Install Git on Your Local Machine:

- ❖ Install Git on your local machine. You can download it from the official Git website: [Git Downloads](#).
- ❖ Optionally, install Gitbash for a convenient terminal with Git-ready functionality.

4. Create an Empty Directory on Your Local Machine:

- ❖ Open a terminal or Gitbash.
- ❖ Navigate to a directory where you want to create your local Git repository.
- ❖ Bash

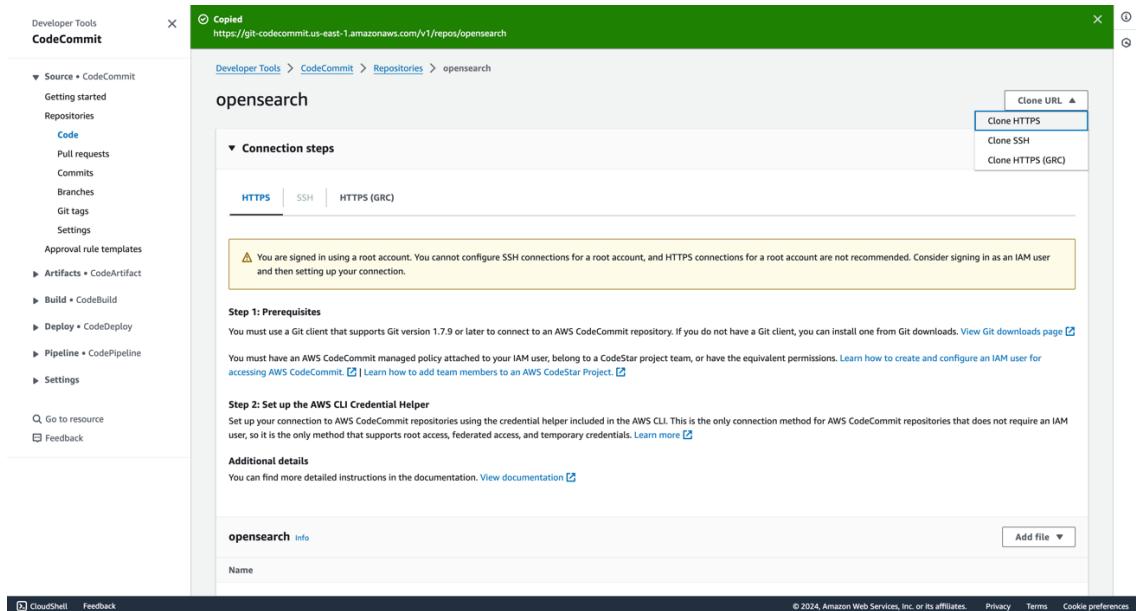
The terminal window shows the following command being run:

```
sh-3.2# mkdir opensearch
sh-3.2# cd opensearch
sh-3.2# ls
sh-3.2#
```

`mkdir opensearch
cd opensearch`

5. Use Git Clone to Set Up Remote Repository:

- ❖ Go to your CodeCommit repository in the AWS Console.



- ❖ Copy the HTTPS URL of the repository.
 - ❖ In your terminal, use the following command to clone the repository:
❖ bash

```
git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/opensearch
```

- ❖ When prompted, enter the CodeCommit credentials generated in Step 2.

```
sh-3.2# mkdir opensearch
sh-3.2# cd opensearch
sh-3.2# git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/opensearch
Cloning into 'opensearch'...
https://git-codecommit.us-east-1.amazonaws.com: admin-at-637423488006
Password for 'https://admin-at-637423488006@git-codecommit.us-east-1.amazonaws.com':
warning: You appear to have cloned an empty repository.
sh-3.2#
```

6. Copy Lambda Function Files and Push to Repository:

- ❖ Copy the Lambda function files (including buildspec and SAM files) into the local directory created in Step 4.
 - ❖ Use the following commands to push the files to the remote repository:

```

sh-3.2$ git add .
sh-3.2$ git commit -m "Initial Commit"
[main (root-commit) 45e6fc] Initial Commit
  Author: System Administrator <system-admin@ip-172-31-10-198.eks-MacBook-Pro.local>
  Committer: System Administrator <system-admin@ip-172-31-10-198.eks-MacBook-Pro.local>
Your name and email were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:
  git config --global --edit

After doing this, you may fix the identity used for this commit with:
  git commit --amend --reset-author

5 files changed, 317 insertions(+)
create mode 100644 pdftotxt/lambdas/function.py
create mode 100644 search-gateway/lambdas/function.py
create mode 100644 search-gateway/search_page.py
create mode 100644 searchFunction/lambdas/function.py
sh-3.2$ git push
Username for 'https://git-codecommit.us-east-1.amazonaws.com': admin-at-637423400806
Password for 'https://admin-at-637423400806@git-codecommit.us-east-1.amazonaws.com':
Enumerating objects: 15, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (11/11), 4.71 KiB | 4.71 MiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Processing objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/opensearch
 * [new branch]      main => main
sh-3.2$ 

```

❖ bash

```

git add .
git commit -m "Initial commit"
git push origin master

```

❖ Enter the CodeCommit credentials if prompted.

The screenshot shows the AWS CodeCommit interface. On the left, there's a sidebar with navigation links for 'Source + CodeCommit', 'Getting started', 'Repositories', 'Code', 'Pull requests', 'Commits', 'Branches', 'Git tags', 'Settings', 'Approval rule templates', 'Artifacts + CodeArtifact', 'Build + CodeBuild', 'Deploy + CodeDeploy', 'Pipeline + CodePipeline', and 'Settings'. The main area shows a repository named 'opensearch'. The repository details page includes sections for 'Info', 'Name', and a list of files: pdftotxt, search-gateway, searchFunction, and UploadToSearch. There are also buttons for 'Notify', 'Reference', 'Create pull request', 'Clone URL', and 'Add file'.

By following these steps, you should have set up a CodeCommit repository, created an IAM user with the necessary permissions, installed Git on your local machine, cloned the remote repository to your local machine, and pushed the Lambda function files into the CodeCommit repository.

CODE BUILD SETUP

1. Create a New CodeBuild Project Using the AWS Console:

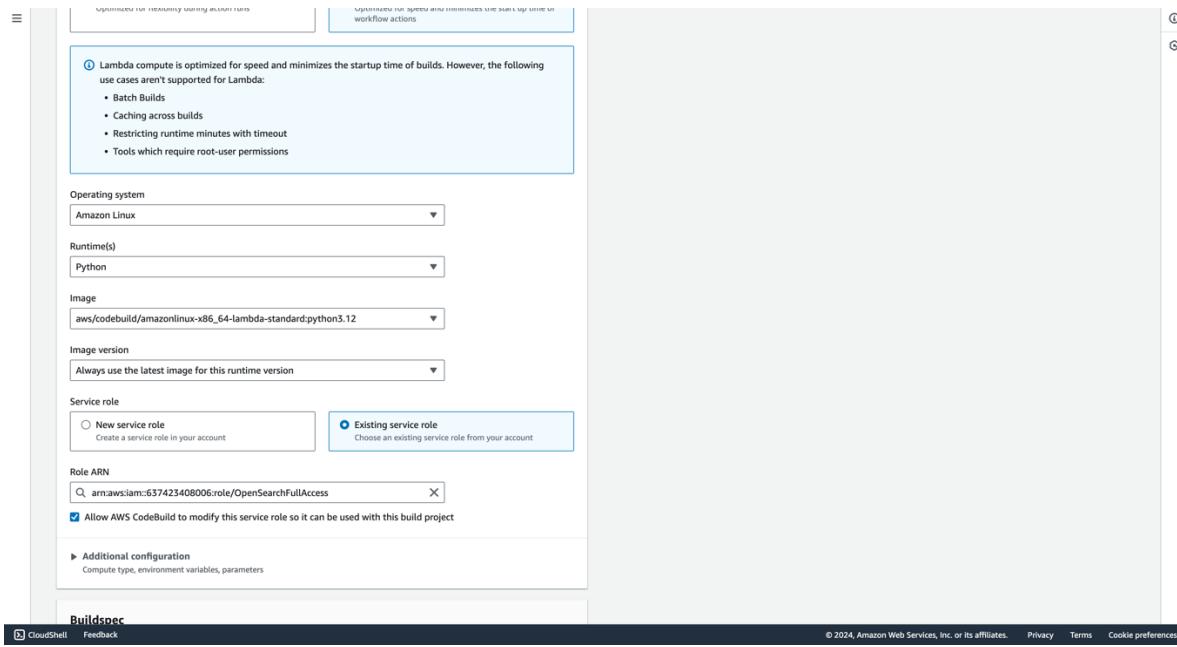
- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the CodeBuild service.

The screenshot shows the AWS Management Console interface for the CodeBuild service. The left sidebar has a tree view with 'CodeBuild' selected. Under 'Build projects', 'Create build project' is highlighted. The main content area is titled 'Build projects' and shows a table with one row labeled 'No results'. Below the table, a message reads 'There are no results to display.'

- ❖ Click on "Create build project."
- ❖ Enter a name for your project.

The screenshot shows the 'Create build project' wizard. The first step is 'Project configuration', where the 'Project name' is set to 'opensearch'. In the 'Source' section, 'Source 1 - Primary' is configured with 'AWS CodeCommit' as the provider and 'opensearch' as the repository. The 'Branch' dropdown is set to 'main'. Other settings like 'Reference type' (Branch), 'Commit ID - optional' (a commit hash), and 'Source version info' are also visible.

- ❖ Configure the source provider as CodeCommit.
- ❖ Choose the repository created previously.
- ❖ Set the other project settings according to your requirements.



- ❖ Click on "Create build project."

2. Use the Latest Standard Ubuntu CodeBuild 6.0 Environment:

- ❖ In the CodeBuild project configuration:

- ❖ Choose the environment image type as "Standard."
- ❖ Select "Ubuntu" as the operating system.
- ❖ Choose the latest CodeBuild 6.0 environment.

3. Create a New Service Role for the Project:

- ❖ In the CodeBuild project configuration:

- ❖ If you don't have an existing service role, choose "New service role" under the "Service role" section.
- ❖ Configure the service role settings according to your requirements.
- ❖ Click on "Update role" or "Create role" to create the service role.

4. Create a New S3 Bucket for Storing Artifacts:

The screenshot shows the Amazon S3 console interface. On the left, there's a sidebar with various navigation options like Buckets, Access Grants, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Storage Lens, Dashboards, Storage Lens groups, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main area is titled 'Amazon S3' and shows an 'Account snapshot' with a link to 'View Storage Lens dashboard'. Below that, there are tabs for 'General purpose buckets' (which is selected) and 'Directory buckets'. Under 'General purpose buckets', there's a heading 'General purpose buckets (1) Info' with a link to 'Learn more'. It says 'Buckets are containers for data stored in S3.' Below this is a table with one row. The table columns are 'Name', 'AWS Region', 'Access', and 'Creation date'. The row shows 'storage0909' as the name, 'US East (N. Virginia) us-east-1' as the region, 'Objects can be public' as the access level, and 'January 14, 2024, 18:35:33 (UTC+05:30)' as the creation date. At the top right of the main area, there are buttons for 'Create bucket' (orange), 'Empty', 'Delete', and others. There's also a search bar labeled 'Find buckets by name'.

- ❖ Navigate to the S3 service in the AWS Console.
- ❖ Click on "Create bucket."
- ❖ Enter a unique name for your bucket.
- ❖ Choose a region for the bucket.
- ❖ Configure other settings as needed.
- ❖ Click on "Create bucket."

5. Configure CodeBuild Project to Use the New S3 Bucket:

- ❖ In the CodeBuild project configuration:
- ❖ Scroll down to the "Artifacts" section.

The screenshot shows the 'Edit Artifacts' configuration page for a CodeBuild project. The 'Artifacts' section is active, showing 'Artifact 1 - Primary'. The 'Type' is set to 'Amazon S3'. The 'Bucket name' is 'storage0909'. The 'Name' field contains 'opensearch-artifacts'. The 'Path - optional' field is empty. The 'Namespace type - optional' is set to 'None'. Under 'Artifacts packaging', the 'None' option is selected, with a note that the artifact files will be uploaded to the bucket. There is also a 'Zip' option. A checkbox for 'Disable artifact encryption' is present, with a note that it disables encryption for publishing static websites or sharing content with others.

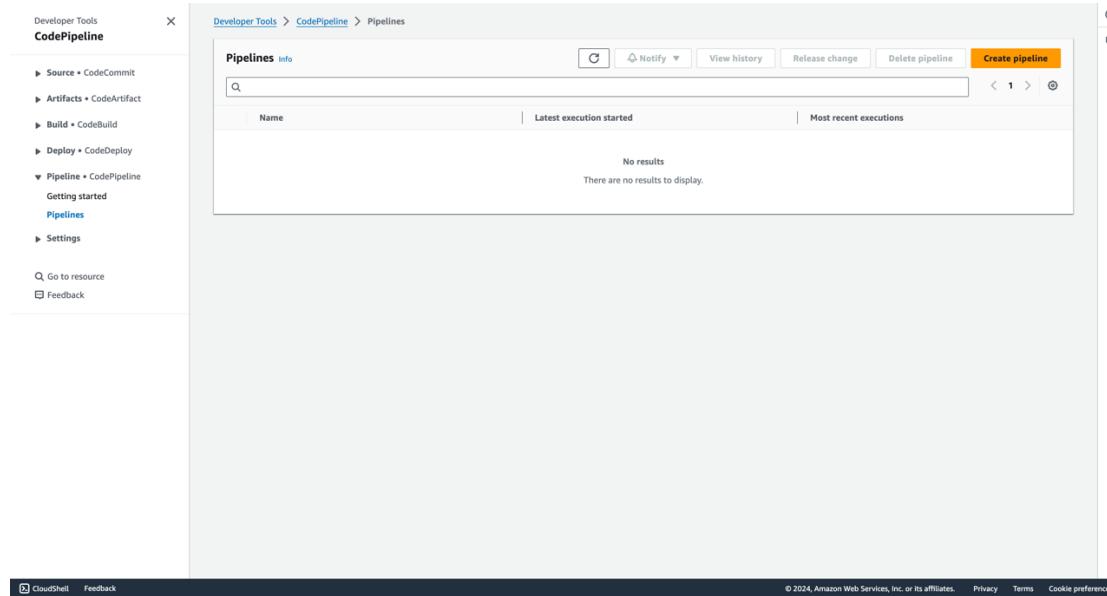
- ❖ Choose "Amazon S3" as the type.
- ❖ Select the region where you created the S3 bucket.
- ❖ Enter the name of the S3 bucket created in Step 4.
- ❖ Set other artifact settings according to your requirements.
- ❖ Click on "Save build project" to save the changes.

By following these steps, you should have created a new CodeBuild project, configured it to use CodeCommit as the source, specified the CodeBuild environment, created a new service role for the project, and set up a new S3 bucket to store artifacts generated during the build process.

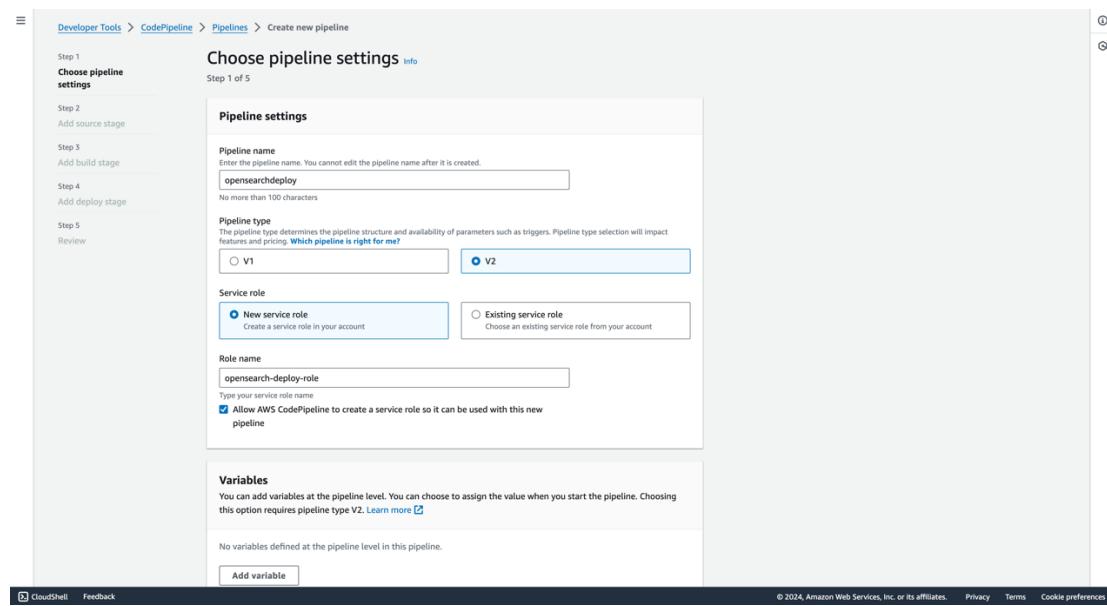
CODE PIPELINE SETUP

1. Create a New CodePipeline Using the AWS Console:

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the CodePipeline service.



- ❖ Click on "Create pipeline."



- ❖ Enter a name for your pipeline.
- ❖ Click on "Next."

2. Select CodeCommit Repository as the Source:

- ❖ In the "Source" configuration:
- ❖ Choose "CodeCommit" as the source provider.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

Add source stage Info

Step 2 of 5

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

Repository name
Choose a repository that you have already created where you have pushed your source code.

opensearch

Branch name
Choose a branch of the repository

main

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

AWS CodePipeline
Use AWS CodePipeline to check periodically for changes

Output artifact format
Choose the output artifact format.

CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

Cancel Previous Next

- ❖ Select the CodeCommit repository created previously.
- ❖ Set other source settings as needed.
- ❖ Click on "Next."

3. Select CodeBuild Project for the Build Stage:

- ❖ In the "Build" configuration:
- ❖ Choose "AWS CodeBuild" as the build provider.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

Add build stage Info

Step 3 of 5

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

Region
US East (N. Virginia)

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

opensearch or Create project

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. Learn more [?]

Add environment variable

Build type

Single build
Triggers a single build.

Batch build
Triggers multiple builds as a single execution.

Cancel Previous Skip build stage Next

- ❖ Select the CodeBuild project created previously.
- ❖ Configure other build settings as needed.
- ❖ Click on "Next."

4. Use CloudFormation for Deployment:

- ❖ In the "Deploy" configuration:
- ❖ Choose "AWS CloudFormation" as the deployment provider.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

Add deploy stage [Info](#)
Step 4 of 5

Deploy - optional

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS CloudFormation

Region
US East (N. Virginia)

Action mode
When you update an existing stack, the update is permanent. When you use a change set, the result provides a diff of the updated stack and the original stack before you choose to execute the change.

Create or update a stack

Stack name
If you are updating an existing stack, choose the stack name.

Template
Specify the template you uploaded to your source location.

Artifact name	File name	Template file path
BuildArtifact	artifact	BuildArtifact::artifact

Template configuration - optional
Specify the configuration file you uploaded to your source location.

Use configuration file

Artifact name	File name	Template configuration file path

Capabilities - optional
Specify whether you want to allow AWS CloudFormation to create IAM resources on your behalf.

CAPABILITY_IAM CAPABILITY_AUTO_EXPAND

Role name
Q arnaws:iam::637423408006:role/OpenSearchFullAccess

Output file name

File generated by this action

► Advanced

Cancel Previous Skip deploy stage **Next**

- ❖ Create a new CloudFormation service role if you don't have one.
- ❖ Ensure the IAM role used for CloudFormation has access to S3 and AWS Lambda.

Action mode
When you update an existing stack, the update is permanent. When you use a change set, the result provides a diff of the updated stack and the original stack before you choose to execute the change.

Create or update a stack

Stack name
If you are updating an existing stack, choose the stack name.

Q

Template
Specify the template you uploaded to your source location.

Artifact name	File name	Template file path
BuildArtifact	artifact	BuildArtifact::artifact

Template configuration - optional
Specify the configuration file you uploaded to your source location.

Use configuration file

Artifact name	File name	Template configuration file path

Capabilities - optional
Specify whether you want to allow AWS CloudFormation to create IAM resources on your behalf.

CAPABILITY_IAM CAPABILITY_AUTO_EXPAND

Role name
Q arnaws:iam::637423408006:role/OpenSearchFullAccess

Output file name

File generated by this action

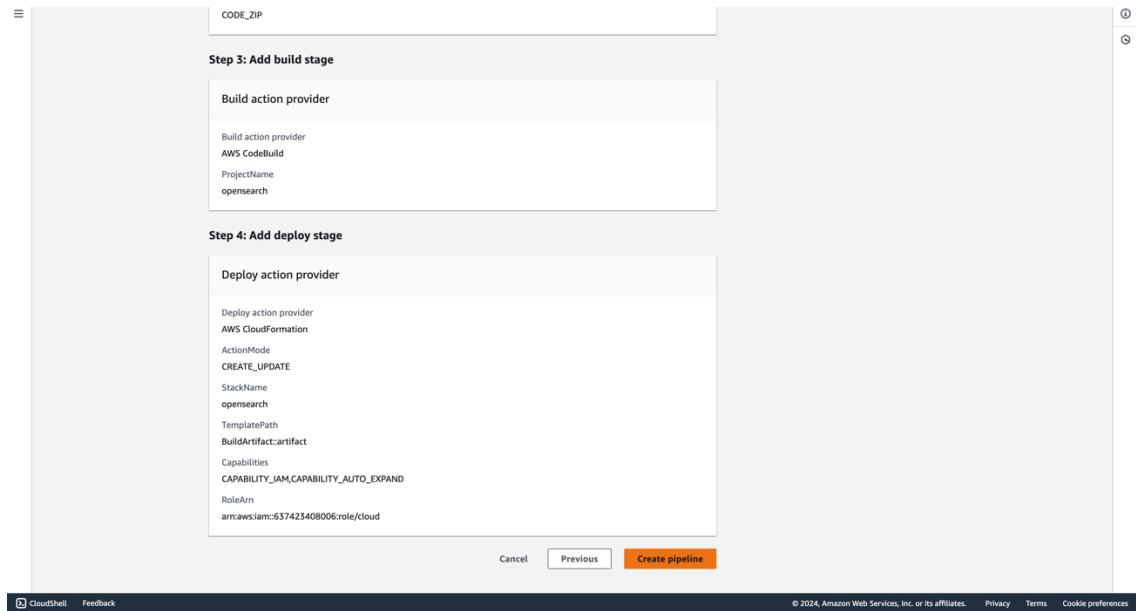
► Advanced

Cancel Previous Skip deploy stage **Next**

- ❖ Add the following capabilities to the CloudFormation role's capabilities:

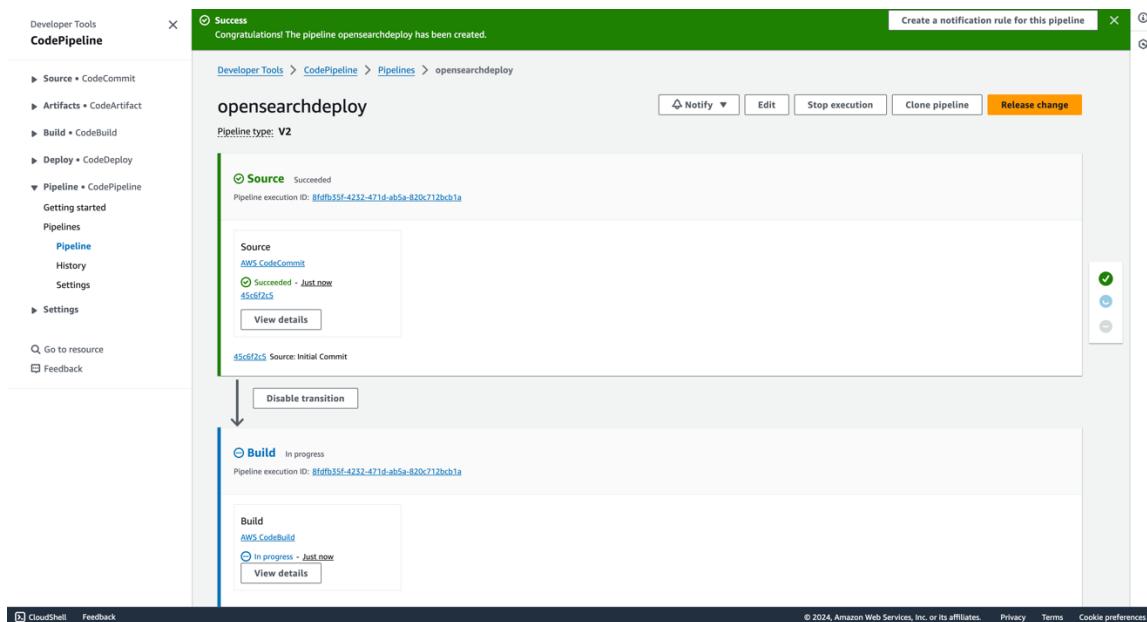
CAPABILITY_IAM
CAPABILITY_AUTO_EXPAND

- ❖ Configure other deployment settings as needed.
- ❖ Click on "Next."



5. Run the Pipeline to Deploy the Lambda Function:

- ❖ Review the pipeline configuration and click on "Create pipeline."



- ❖ After creating the pipeline, click on "Release change" or wait for the pipeline to automatically start.

- ❖ The pipeline will execute the source, build, and deployment stages, deploying the Lambda function.

6. Set Up Separate CodeCommit Repositories, CodeBuild Projects, and Pipelines:

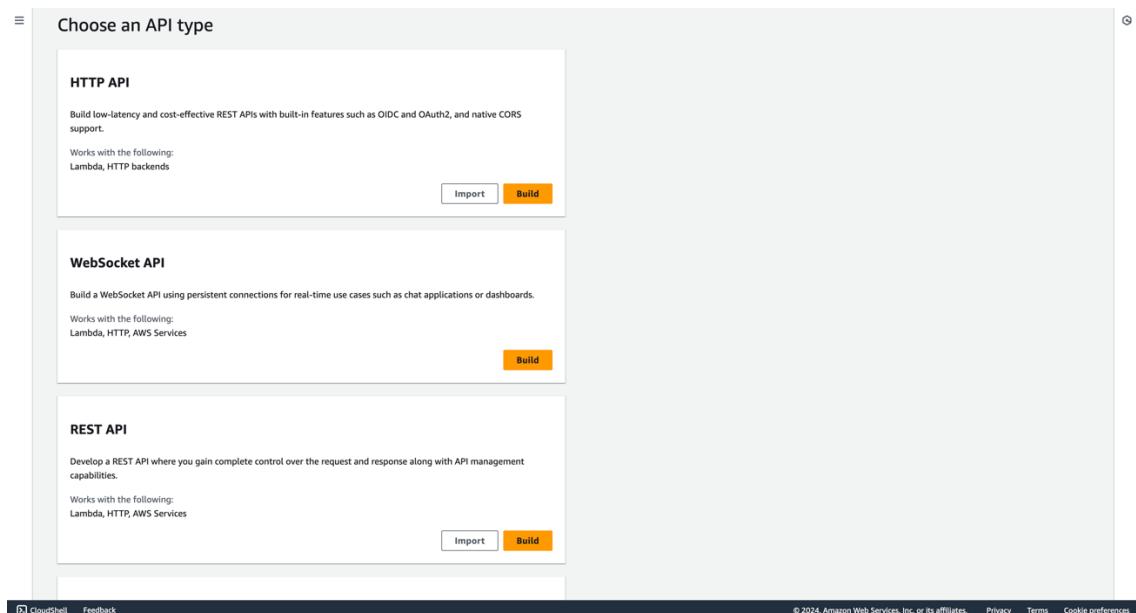
- ❖ Repeat the steps above for each Lambda function:
- ❖ Create a new CodeCommit repository for each Lambda function.
- ❖ Create a new CodeBuild project for each Lambda function.
- ❖ Create a new CodePipeline for each Lambda function, selecting the corresponding CodeCommit repository and CodeBuild project.

By following these steps, you should have set up a CodePipeline that pulls source code from CodeCommit, builds the Lambda function using CodeBuild, and deploys it using CloudFormation. Repeat the process for each Lambda function by creating separate CodeCommit repositories, CodeBuild projects, and pipelines.

API GATEWAY SETUP

1. Create a New HTTP Gateway Using the AWS Console:

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the API Gateway service.
- ❖ Click on "Create API."
- ❖ Choose "HTTP API."



- ❖ Enter a name for your API.
- ❖ Click on "Create API."

2. Add Lambda Functions to Integrations:

- ❖ In your API Gateway configuration:
- ❖ Navigate to the "Integrations" section.

The screenshot shows the AWS API Gateway Integrations page for the 'search-gateway-api' API. On the left, the navigation menu includes 'Develop', 'Routes', 'Authorization', 'Integrations' (which is selected), 'Cors', 'Reimport', 'Export', 'Deploy', 'Stages', 'Monitor', 'Metrics', 'Logging', 'Protect', and 'Throttling'. The main content area displays 'Routes for search-gateway-api' with three entries: '/search-gateway-get' (GET, AWS Lambda), '/search-gateway-post' (POST, AWS Lambda), and '/searchFunction' (ANY, AWS Lambda). A search bar at the top is empty. A message at the bottom right says 'Choose a route.' Below the main content is a URL bar with the address 'https://us-east-1.console.aws.amazon.com/apigateway/main/develop/integrations?region=us-east-1' and a footer with links for 'Cloudshell', 'Feedback', '© 2024, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

- ❖ Add the search-gateway Lambda function as an integration.
- ❖ Add the searchFunction Lambda function as an integration.

3. Add Routes to API Gateway:

- ❖ In your API Gateway configuration:
- ❖ Navigate to the "Routes" section.

The screenshot shows the AWS API Gateway Routes page for the 'search-gateway-api' API. The navigation menu is identical to the previous screenshot. The main content area displays 'Routes for search-gateway-api' with three entries: '/search-gateway-get' (GET), '/search-gateway-post' (POST), and '/searchFunction' (ANY). A 'Create' button is located at the top right of the route list. A message at the bottom right says 'Choose a route.' Below the main content is a URL bar with the address 'https://us-east-1.console.aws.amazon.com/apigateway/main/develop/routes?region=us-east-1' and a footer with links for 'Cloudshell', 'Feedback', '© 2024, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

- ❖ Add the following routes:

- ❖ Route 1:
 - Method: Any
 - Route: /
 - Integration Target: search-gateway
- ❖ Route 2:
 - Method: Any
 - Route: /search
 - Integration Target: searchFunction

4. Deploy the API:

- ❖ After adding the routes, navigate to the "Deployments" section.
- ❖ Click on "Create."

The screenshot shows the AWS API Gateway console. On the left, there's a sidebar with options like APIs, Custom domain names, VPC links, Develop (Routes, Authorization, Integrations), Deploy (Stages), Monitor (Metrics, Logging), and Protect (Throttling). The main area is titled 'Integrations' and shows 'Routes for search-gateway-api'. It lists two routes: '/search-gateway-get' (HTTP GET, AWS Lambda) and '/search-gateway-post' (HTTP POST, AWS Lambda). Below these, there's a link for '/searchFunction' (ANY, AWS Lambda). At the top right, there's a 'Deploy' button, which is highlighted with a red box. The status bar at the bottom indicates 'Choose a route.'

- ❖ Enter a deployment name and click on "Create."
- ❖ After creating the deployment, note the Invoke URL.

5. Test the API:

- ❖ Use the Invoke URL to test the API routes:
- ❖ For the root route /, test various HTTP methods (e.g., GET, POST).
- ❖ For the /search route, test various HTTP methods as well.

By following these steps, you should have created an HTTP Gateway using API Gateway, added Lambda functions as integrations, and configured routes for different methods. Deploy the API and test the routes to ensure that the integration with Lambda functions is working as expected.

FINAL CHECKS

1. Ensure IAM Role Permissions for S3 Access:

- ❖ Navigate to the IAM service in the AWS Console.
- ❖ Locate the IAM role attached to the pdftotxt and upload-to-search Lambda functions.
- ❖ Ensure that the IAM role has the necessary permissions for S3 access:
- ❖ Permissions to read from the S3 bucket where PDF files are uploaded (for pdftotxt).
- ❖ Permissions to write to the S3 bucket used as intermediary storage (for upload-to-search).

2. Ensure IAM Role Permissions for OpenSearch Access:

- ❖ Navigate to the IAM service in the AWS Console.
- ❖ Locate the IAM roles attached to the upload-to-search and searchFunction Lambda functions.
- ❖ For the upload-to-search Lambda function:
 - ❖ Ensure that the IAM role has permissions to interact with OpenSearch.
- ❖ For the searchFunction Lambda function:
 - ❖ Ensure that the IAM role has permissions to interact with OpenSearch.
- ❖ If the IAM role used for searchFunction was created as part of the OpenSearch setup, it should have the necessary permissions.

3. Ensure Triggers are Configured for Lambda Functions:

- ❖ Verify that the triggers for all Lambda functions have been configured as described in their respective sections:
- ❖ For pdftotxt: S3 PUT Event trigger on the content store S3 bucket.
- ❖ For upload-to-search: S3 PUT Event trigger on the S3 bucket used as intermediary storage.
- ❖ For search-gateway: API Gateway trigger.
- ❖ For searchFunction: API Gateway trigger.
- ❖ Ensure that the necessary configurations, such as environmental variables and IAM roles, are correctly set up for each Lambda function based on the provided instructions.

By completing these steps, you should have ensured that the IAM roles attached to the Lambda functions have the required permissions for S3 and OpenSearch access. Additionally, you should have verified that the triggers for each Lambda function are correctly configured according to their specific requirements.

TEST THE DEPLOYMENT

1. Upload the Test PDF File to the Document Store Bucket:

- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the S3 service.
- ❖ Find the document store bucket created previously.
- ❖ Upload the test PDF file provided with this document to the document store bucket.

2. Monitor Lambda Functions in CloudWatch:

- ❖ Navigate to the AWS CloudWatch service.
- ❖ Go to the CloudWatch Logs section.
- ❖ Monitor the CloudWatch log groups associated with each Lambda function (pdftotxt, upload-to-search, search-gateway, and searchFunction).
- ❖ Observe the logs to ensure that each function is triggered and executes successfully.

3. Access the Search Webpage:

- ❖ Use the Invoke URL of the API Gateway created previously.
- ❖ Navigate to the search webpage.

4. Search for a Word in the Sample PDF Document:

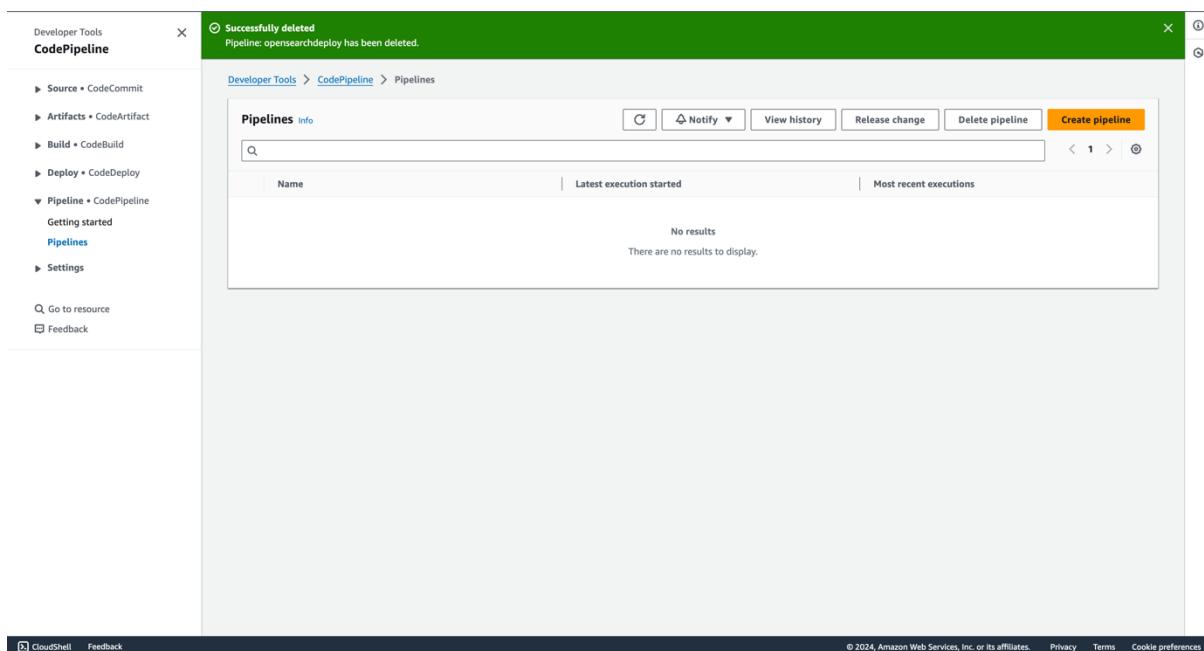
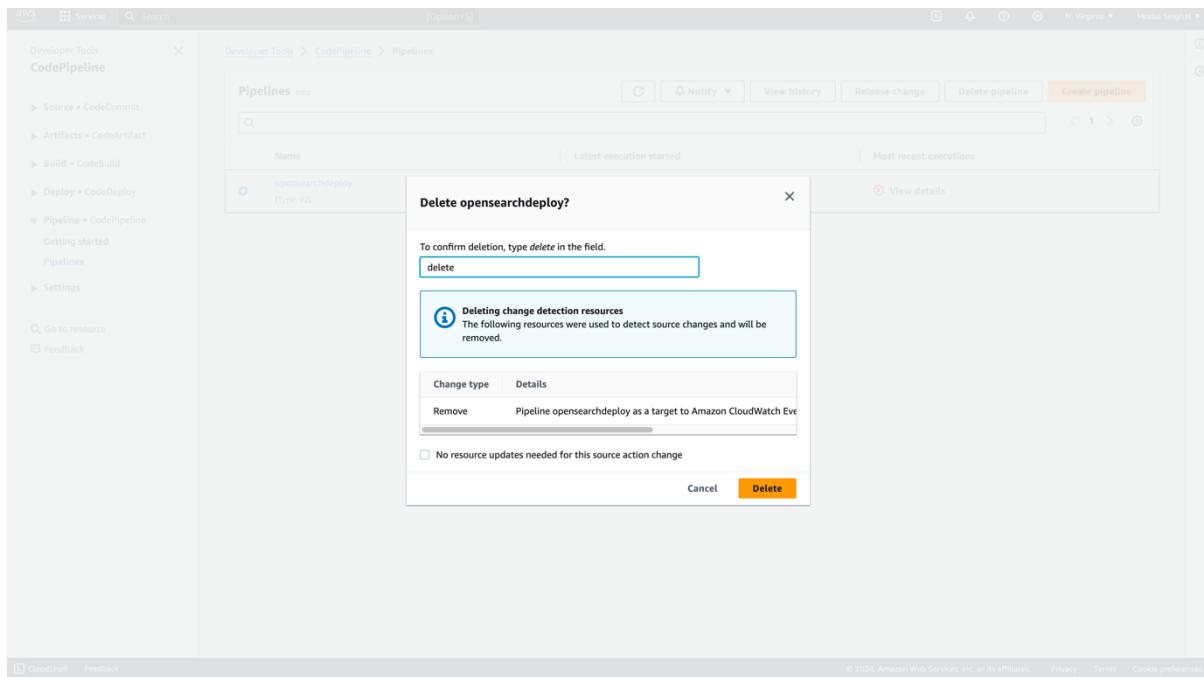
- ❖ On the search webpage, enter a word that is present in the sample PDF document.
- ❖ Submit the search query.
- ❖ Check the output to verify that the search results are correct.

By following these steps, you should be able to test the deployment:

The test PDF file should trigger the Lambda functions in succession as expected. Monitoring the CloudWatch logs will help identify any issues during the execution of each function.

Accessing the search webpage and searching for a word should return correct results if the integration between the API Gateway, Lambda functions, and OpenSearch is set up correctly.

CLEANUP



❖ Pipeline deleted

The screenshot shows the AWS CloudFormation console. On the left, a sidebar menu includes 'Stacks', 'Designer', 'Registry', 'Feedback', and other options. The main area displays a table of stacks with one entry: 'opensearch' (Status: ROLLBACK_COMPLETE, Created time: 2024-01-15 01:37:59 UTC+0530). A modal window titled 'Delete stack?' is open over this entry, containing a warning message: 'Delete stack opensearch permanently? This action cannot be undone.' It also states: 'Deleting this stack will delete all stack resources. Resources will be deleted according to their DeletionPolicy. Learn more'. At the bottom of the modal are 'Cancel' and 'Delete' buttons.

The screenshot shows the AWS CloudFormation console after the stack has been deleted. The main area now displays a table with 'Stacks (0)' and a message: 'No stacks to display'. A large orange 'Create stack' button is prominently displayed. The status bar at the bottom indicates: 'Delete initiated for arn:aws:cloudformation:us-east-1:637423408006:stack/opensearch/bef2b1a0-b317-11ee-81e1-12d02f41ebbd'.

❖ Deleting the stack

The screenshot shows the AWS CodeCommit interface. On the left, a sidebar navigation includes 'Source' (selected), 'CodeCommit', 'Getting started', 'Repositories' (selected), 'Approval rule templates', 'Artifacts', 'Build', 'Deploy', 'Pipeline', and 'Settings'. The main content area shows a 'Repositories' list with one item: 'opensearch'. A modal dialog titled 'Delete opensearch?' is open, containing a warning message: 'Are you sure you want to delete the repository opensearch? This will delete the repository in AWS CodeCommit, including all branches, triggers, comments, pull requests, and history. Deleting the repository cannot be undone.' Below the message, it says 'Users will no longer be able to connect to the repository in AWS CodeCommit, but they will still have access to their local repositories.' A text input field contains the word 'delete'. At the bottom of the modal are 'Cancel' and 'Delete' buttons. The top right of the screen shows 'N. Virginia' and 'Hindi/Singhal'. The bottom right includes links for 'CloudShell', 'Feedback', '© 2024, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Success
opensearch successfully deleted.

❖ Deleting the code commit repository.

The screenshot shows the AWS CodeBuild console. The top navigation bar has a green success banner stating "Success Project arn:aws:codebuild:us-east-1:637423408006:project/opensearch successfully deleted". Below the banner, the breadcrumb trail is "Developer Tools > CodeBuild > Build projects". The main content area is titled "Build projects" with a search bar and a table header: Name, Source provider, Repository, Latest build status, Description, Last Modified. A message below the table says "No results There are no results to display." At the bottom right of the page, there are links for "CloudShell", "Feedback", and copyright information: "© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

❖ Deleted the CodeBuild Project.

The screenshot shows the AWS Lambda console. The left sidebar includes "Dashboard", "Applications", "Functions", "Additional resources" (Code signing configurations, Layers, Replicas), and "Related AWS resources" (Step Functions state machines). The main area is titled "Functions (4/4)" and lists four functions: pdftotxt, upload-to-search, searchFunction, and search-gateway. A modal window titled "Delete 4 functions" is open, containing a warning message: "⚠ Deleting a function permanently removes the function code. The related logs, roles, test event schemas, and triggers are retained in your account." Below the warning, it lists the four selected functions. A text input field contains the word "delete" with the instruction "To confirm deletion, type delete in the field." At the bottom of the modal are "Cancel" and "Delete" buttons. The footer of the page includes "CloudShell", "Feedback", and copyright information: "© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

The screenshot shows the AWS Lambda Functions page. The left sidebar includes links for Dashboard, Applications, Functions (selected), Additional resources (Code signing configurations, Layers, Replicas), and Related AWS resources (Step Functions state machines). The main content area displays a table titled 'Functions (0)' with columns for Function name, Description, Package type, Runtime, and Last modified. A message at the bottom states 'There is no data to display.' The top right of the page shows 'Last fetched 34 seconds ago' and 'Actions' with a 'Create function' button.

❖ Deleted all the four lambda functions.

The screenshot shows the AWS API Gateway APIs page. The left sidebar lists APIs, Custom domain names, VPC links, Usage plans, API keys, Client certificates, and Settings. The main content area shows a table for APIs with one entry: 'search-gateway-api'. The table includes columns for Name, Description, ID, Protocol, API endpoint type, and Created. A modal window titled 'Delete API' is open over the table, asking for confirmation to delete the API. The confirmation text reads: 'Delete the API search-gateway-api permanently? This will delete all child resources and cannot be undone.' A text input field contains the word 'confirm'. At the bottom of the modal are 'Cancel' and 'Delete' buttons.

The screenshot shows the Amazon API Gateway console. At the top, there are two notifications: "Introducing the new API Gateway console experience" and "Successfully deleted API 'search-gateway-api'". Below this, the main header reads "Amazon API Gateway" with the tagline "create, maintain, and secure APIs at any scale". A brief description follows: "Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services." The navigation bar includes "API Gateway > APIs > Create API". The main content area is titled "Choose an API type" and contains two options: "HTTP API" and "WebSocket API". The "HTTP API" section is expanded, showing its description: "Build low-latency and cost-effective REST APIs with built-in features such as OAuth2, and native CORS support." It also mentions "Works with the following: Lambda, HTTP backends" and has "Import" and "Build" buttons. The "WebSocket API" section is collapsed. At the bottom of the page are links for "CloudShell", "Feedback", and copyright information: "© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

❖ Deleted the API Gateway

The screenshot shows the AWS IAM console. The left sidebar includes "Identity and Access Management (IAM)", "Access management" (User groups, Users, Roles, Policies, Identity providers, Account settings), "Access reports" (Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity), and "Service control policies (SCPs)". The main area shows "Users (1/1) info" with one user named "admin". A modal window titled "Delete admin?" is open, asking "Delete admin permanently? This will also delete all its user data, security credentials and inline policies." It lists the user name "admin" and last activity "-". A note states: "Note: Recent activity usually appears within 4 hours. Data is stored for a maximum of 365 days, depending when your region began supporting this feature. [Learn more](#)". Below the note, it says "This action cannot be undone." A text input field contains "admin" with the placeholder "To confirm deletion, enter the user name in the text input field." At the bottom of the modal are "Cancel" and "Delete user" buttons. The footer of the page includes "CloudShell", "Feedback", and copyright information: "© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

❖ Deleted the IAM User

The screenshot shows the AWS IAM Roles page with a modal dialog titled "Delete 5 roles?". The dialog asks if the user wants to permanently delete five roles, noting that this will also delete all their inline policies and any attached instance profiles. The roles listed are:

Role name	Last activity
cloud	-
CodeBuild	28 minutes ago
cwe-role-us-east-1-opensearchdeploy	-
opensearch-deploy-role	27 minutes ago
OpenSearchFullAccess	2 hours ago

Note: Recent activity usually appears within 4 hours. Data is stored for a maximum of 365 days, depending when your region began supporting this feature. [Learn more](#)

This action cannot be undone.

To confirm deletion, enter `delete` in the text input field.

`delete`

On the right, there are sections for "Temporary credentials" and "Access AWS from your non AWS workloads".

❖ Deleted the IAM Roles.

The screenshot shows the AWS Lambda Layers page for the "pdftotxt" layer. A modal dialog titled "Delete pdftotxt version: 1" is displayed, asking if the user wants to permanently remove the associated code and configuration. The dialog states: "Deleting this layer version will permanently remove the associated code and configuration. Invocations of functions using the version will continue working. Are you sure you want to delete this Lambda layer version?"

On the left, the Lambda Layers page shows the "pdftotxt" layer with version 1 details:

Version	Version ARN
1	arn:aws:lambda:us-east-1:637423408006:layer:pdftotxt:1

Below the table, there are tabs for "Versions" and "Functions using this version".

❖ Deleted the Lambda Layer pdftotxt “pypdf” dependency.

The screenshot shows the AWS Lambda console. In the left sidebar, under 'Additional resources', 'Layers' is selected, showing 'requests' as the chosen layer. The main area displays 'Version details' for version 2, which was created 4 days ago and has x86_64 compatible architectures. A modal window titled 'Delete requests version: 2' is open, asking if the user wants to permanently remove the version. The modal includes a 'Cancel' button and a highlighted 'Delete' button. Below the modal, the list of versions shows version 2 with the ARN armawslambdaus-east-1:637423408006:layer:requests:2.

❖ Deleted the layer for requests dependency.

The screenshot shows the Amazon OpenSearch Service console. Under 'Managed clusters', 'Domains' is selected, showing 'opensearch' as the chosen domain. The 'General information' section shows the domain name 'opensearch' and its ARN 'arn:aws:es:us-east-1:637423408006:domain/opensearch'. A modal window titled 'Delete domain?' is open, asking if the user wants to delete the domain. It includes a 'Domain' input field containing 'opensearch' and a confirmation field with 'opensearch'. The modal has 'Cancel' and 'Delete' buttons. To the right, there are sections for 'OpenSearch Dashboards URL (dual stack)', 'Domain endpoint v2 (dual stack)', and 'Domain endpoint (IPv4)'. Below the modal, tabs for 'Cluster configuration', 'Security configuration', 'Cluster health', 'Instance health', 'Off-peak window', 'Auto-Tune', 'Logs', 'Indices', 'Tags', and 'Connections - preview' are visible.

❖ Deleted the OpenSearch.

Amazon S3 > Buckets > storage0909

storage0909 [Info](#)

Objects (4) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Copy](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

<input checked="" type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input checked="" type="checkbox"/>	Astronomy.pdf	pdf	January 14, 2024, 18:57:41 (UTC+05:30)	26.7 KB	Standard
<input checked="" type="checkbox"/>	Astronomy.pdf.txt	txt	January 14, 2024, 19:02:26 (UTC+05:30)	1.5 KB	Standard
<input checked="" type="checkbox"/>	db856c01-be4f-492f-9b4a-e7218baeb324.gz	gz	January 15, 2024, 01:40:21 (UTC+05:30)	130.0 B	Standard
<input checked="" type="checkbox"/>	eed2495d-ceb3-44db-b4f3-8ac54ac63507.gz	gz	January 15, 2024, 01:41:40 (UTC+05:30)	129.0 B	Standard

Amazon S3 > Buckets > storage0909 > Delete objects

Delete objects [Info](#)

⚠ If a folder is selected for deletion, all objects in the folder will be deleted, and any new objects added while the delete action is in progress might also be deleted. If an object is selected for deletion, any new objects with the same name that are uploaded before the delete action is completed will also be deleted.

Deleting the specified objects can't be undone.

[Learn more](#)

Specified objects

Find objects by name

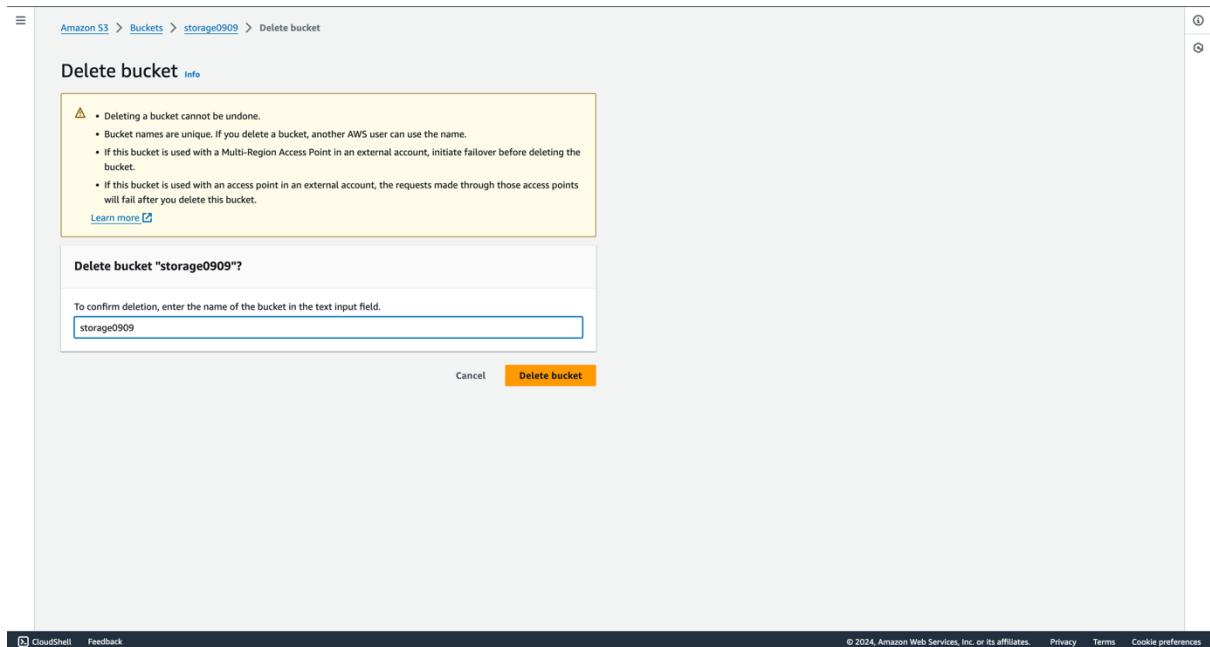
Name	Type	Last modified	Size
Astronomy.pdf	pdf	January 14, 2024, 18:57:41 (UTC+05:30)	26.7 KB
Astronomy.pdf.txt	txt	January 14, 2024, 19:02:26 (UTC+05:30)	1.5 KB
db856c01-be4f-492f-9b4a-e7218baeb324.gz	gz	January 15, 2024, 01:40:21 (UTC+05:30)	130.0 B
eed2495d-ceb3-44db-b4f3-8ac54ac63507.gz	gz	January 15, 2024, 01:41:40 (UTC+05:30)	129.0 B

Permanently delete objects?

To confirm deletion, type **permanently delete** in the text input field.

permanently delete

[Cancel](#) [Delete objects](#)



- ❖ Deleted the S3 bucket also.
- ❖ Full Cleanup Complete.