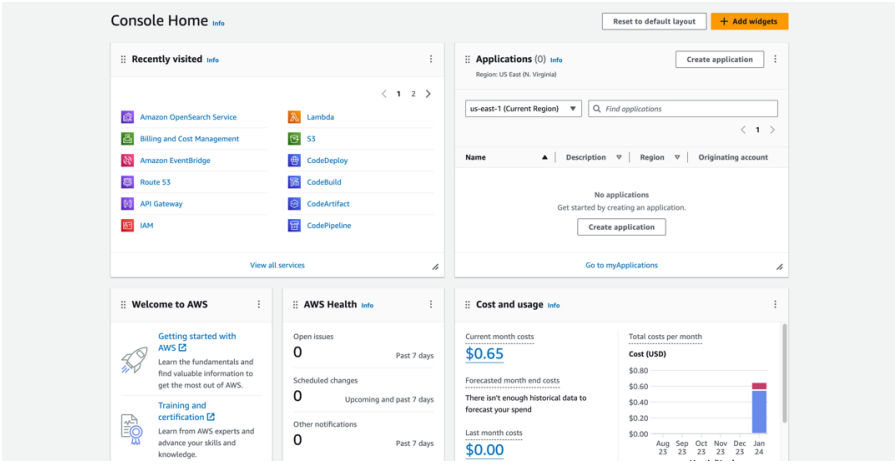
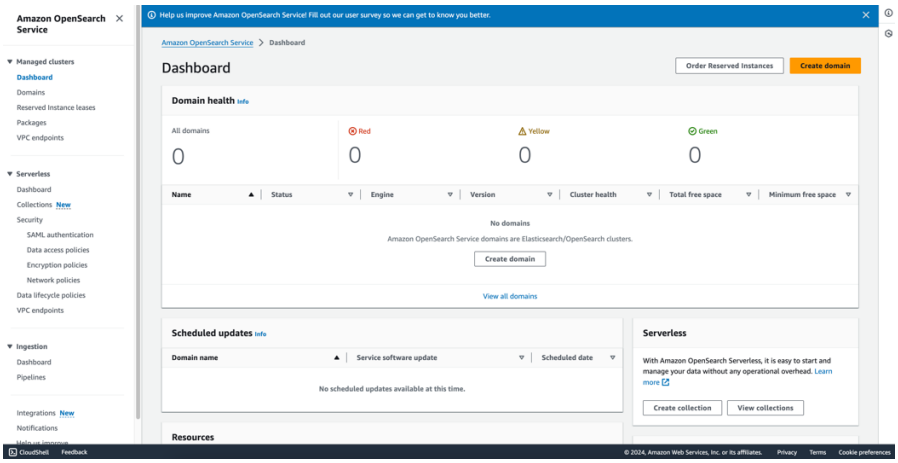
SEARCH ENGINE DEPLOYMENT

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

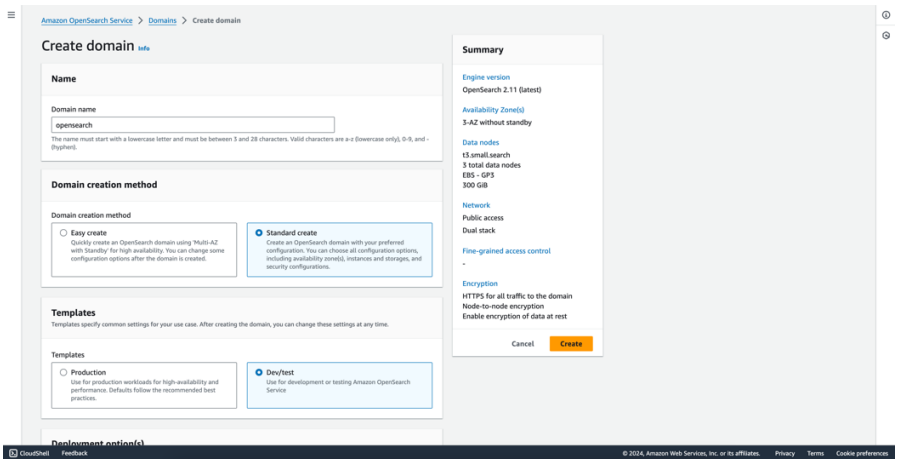
❖ Log in to your AWS Management Console.



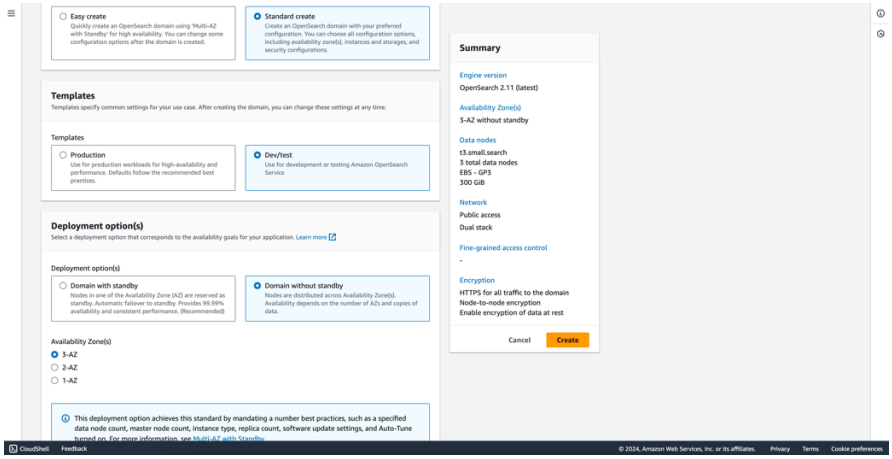
❖ Navigate to the OpenSearch Service section.



❖ Click on "Create domain."



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

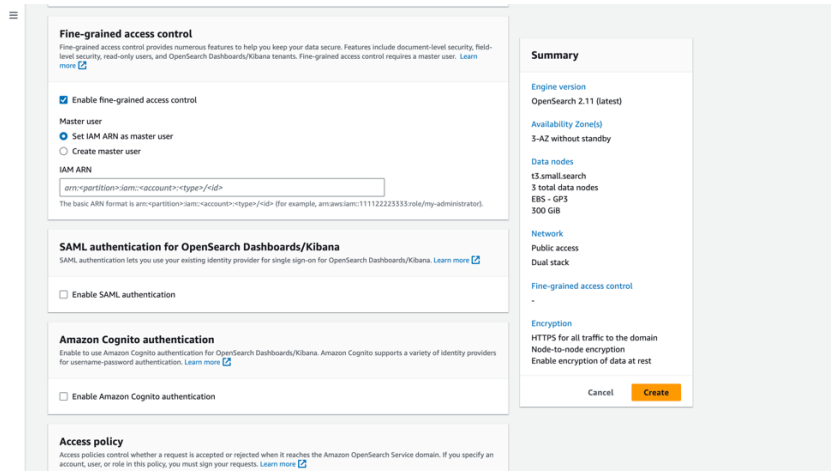


- Enter a domain name and configure other settings as needed.
- In the "Data nodes" section, specify the number of data nodes, instance types, and storage settings. Click "Next" when done.
- Configure other advanced settings if necessary and proceed to the next steps.
- Finally, review your configuration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

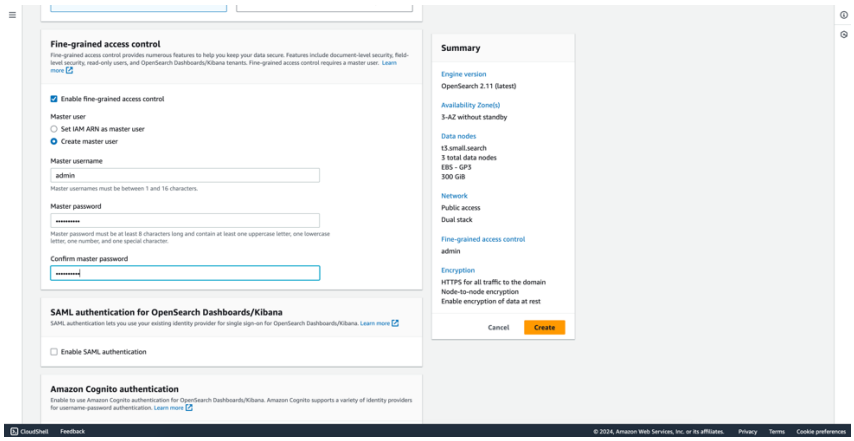
2. Select “VPC access” in the Network Section:

- 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:

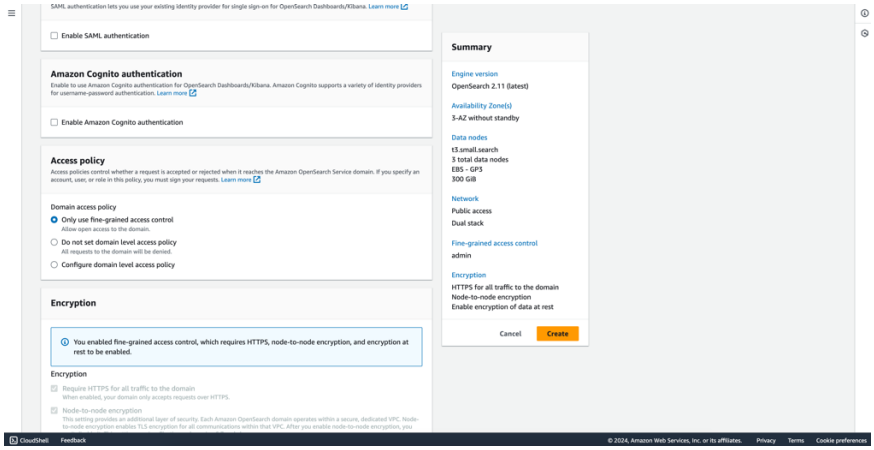


- ❖ Ensure Fine-grained access control is turned on:
- ❖ In the OpenSearch domain dashboard, navigate to the "Fine-grained access control" section.
- ❖ Ensure that fine-grained access control is enabled. If not, enable it.
- ❖ Create a Master User with a username and password:

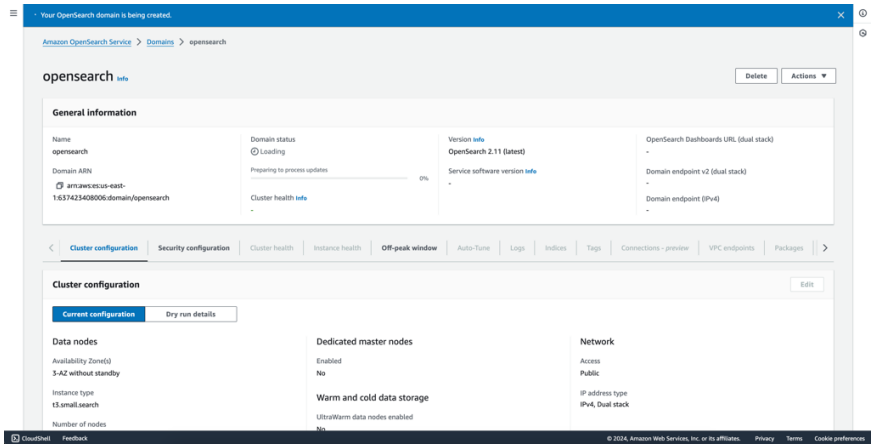


Still in the "Fine-grained access control" section, find the "Master User" tab.
Create a new Master User by providing a username and password. Note down these credentials as you'll need them later.

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



- ❖ 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.

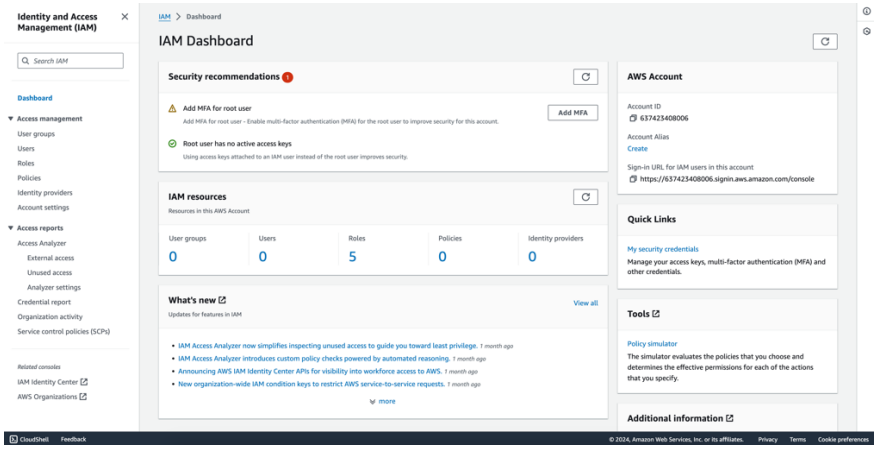


By following these steps, you should have successfully created an OpenSearch domain on AWS, configured VPC access, enabled fine-grained access control, and set the access policy to only use fine-grained access control.

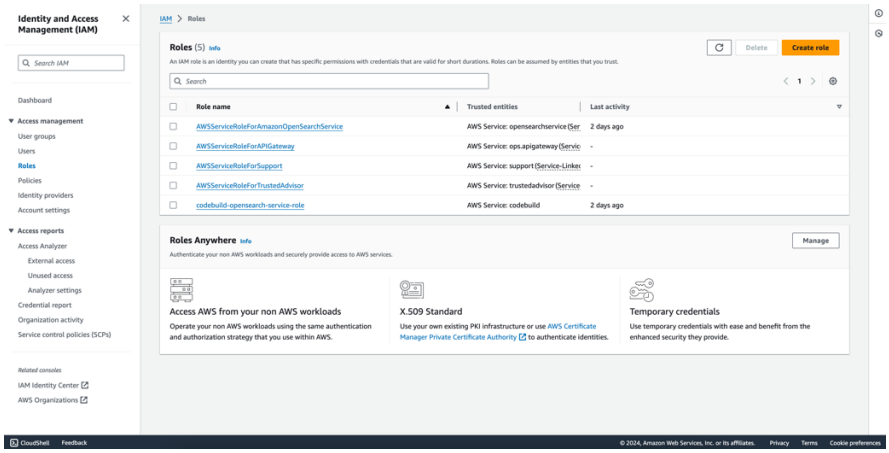
IAM ROLE CREATION

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

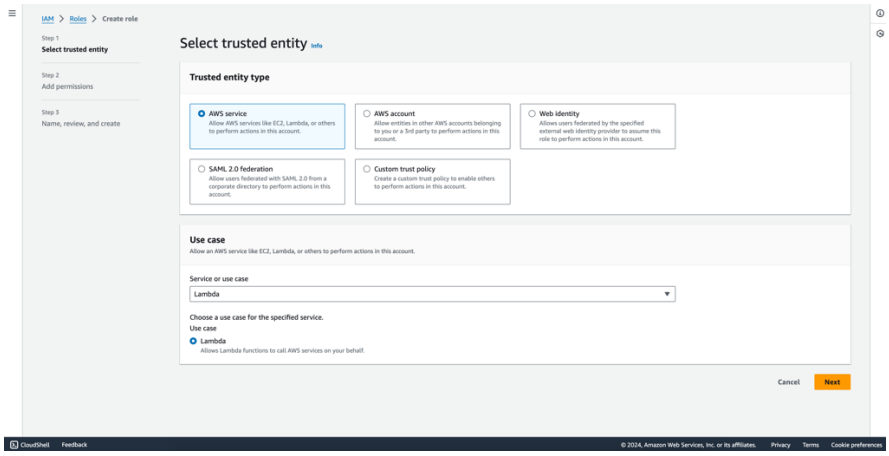
- ❖ Log in to the AWS Management Console.
- ❖ Navigate to the IAM (Identity and Access Management) service.



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

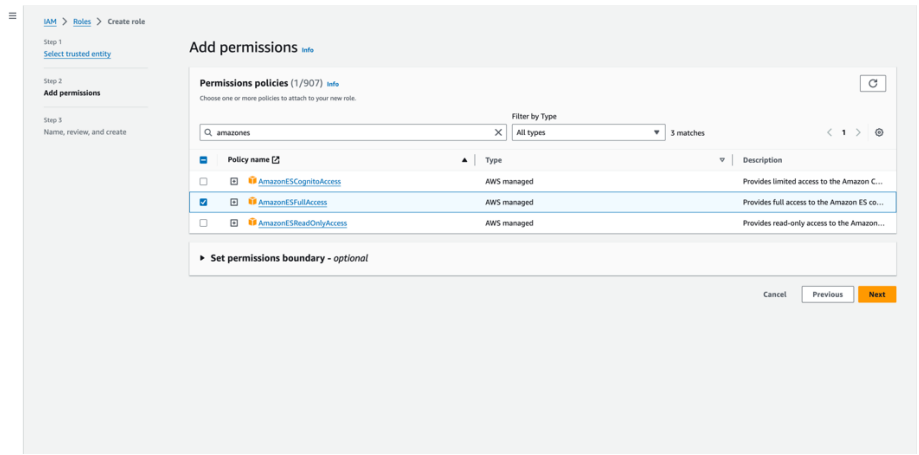


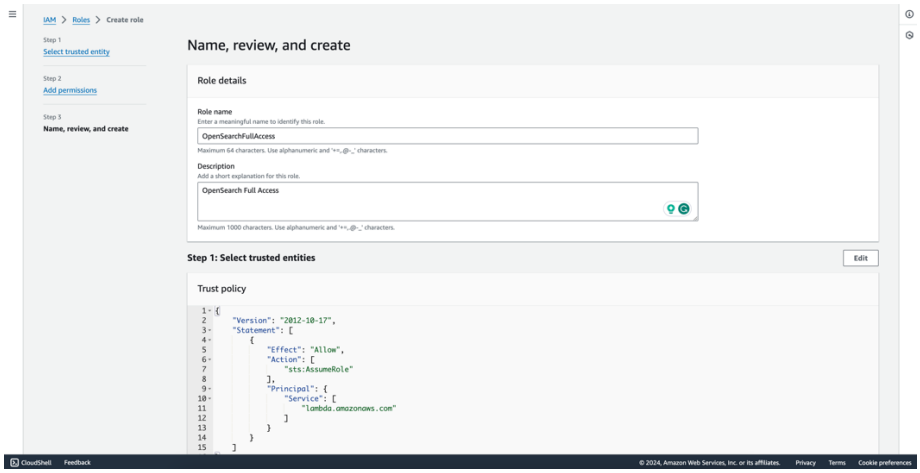
❖ Choose "AWS service" as the type of trusted entity and select "Lambda."

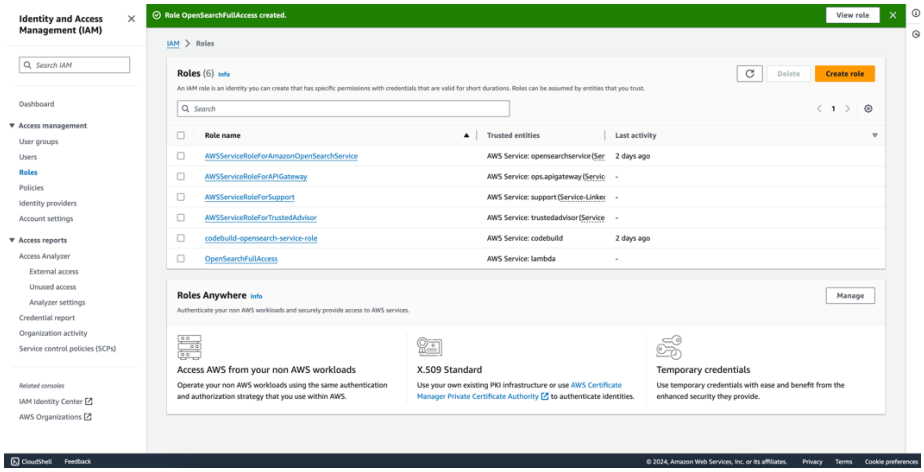


❖ In the permissions section, attach policies that provide the necessary permissions for your Lambda function to interact with OpenSearch. You might use policies like AmazonESFullAccess or customize permissions based on your specific requirements.

❖



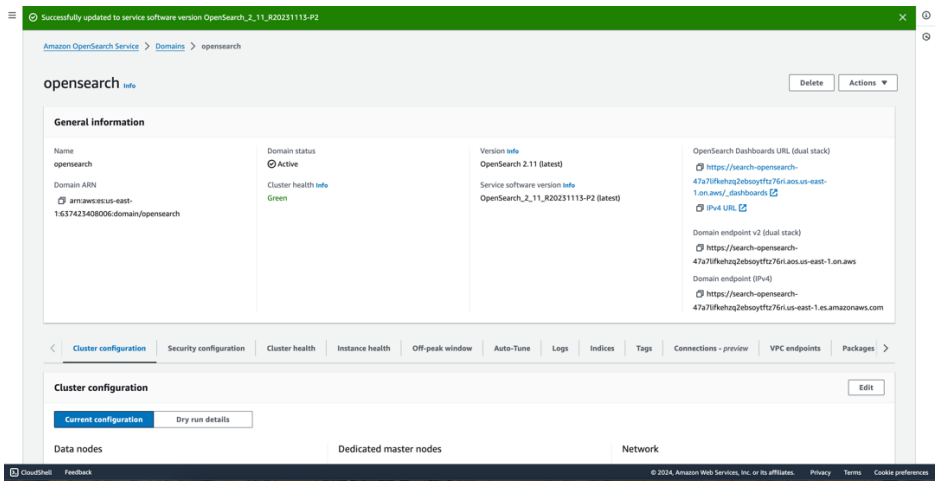




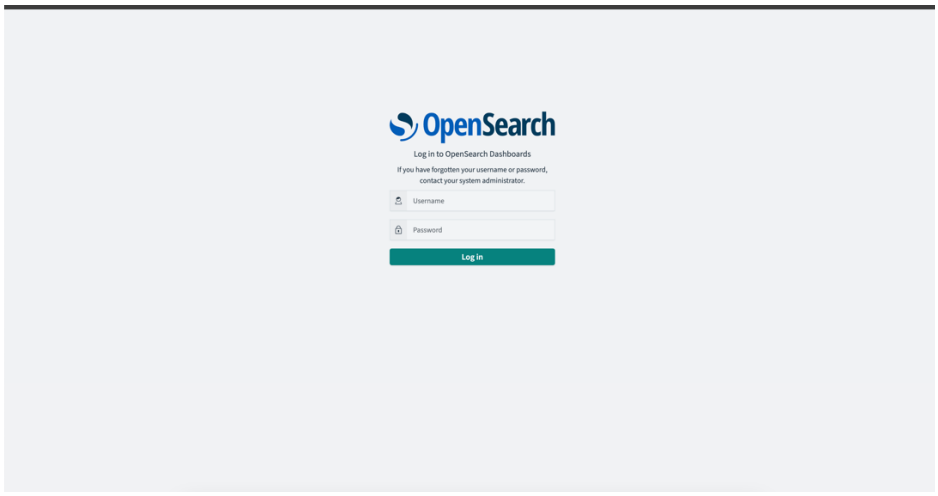
❖ Complete the role creation, giving it a meaningful name and description.

2. Login to the OpenSearch dashboard:

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

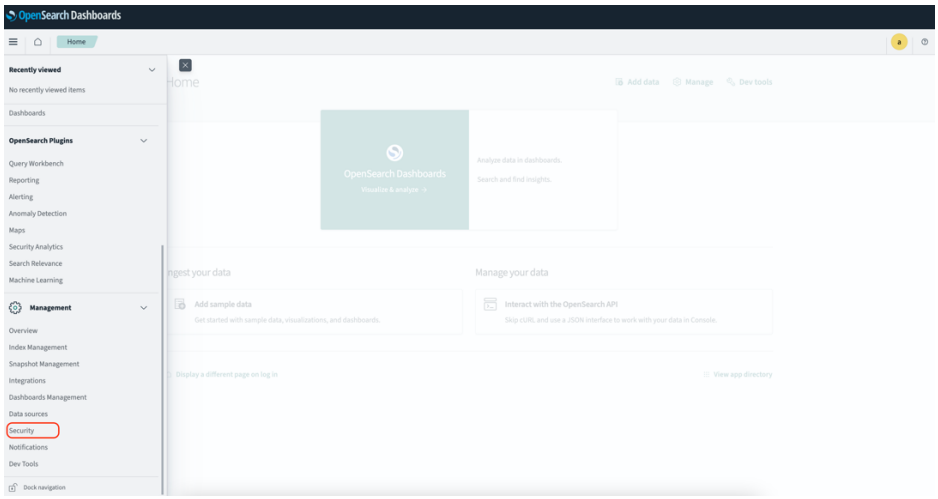


❖ Enter the master username and password when prompted.



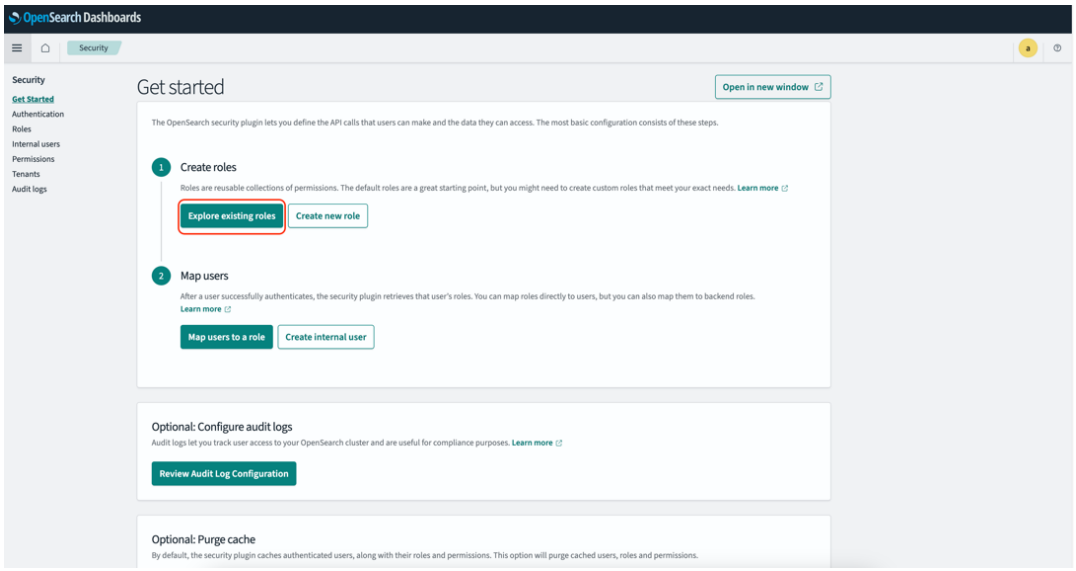
3. Navigate to Security:

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

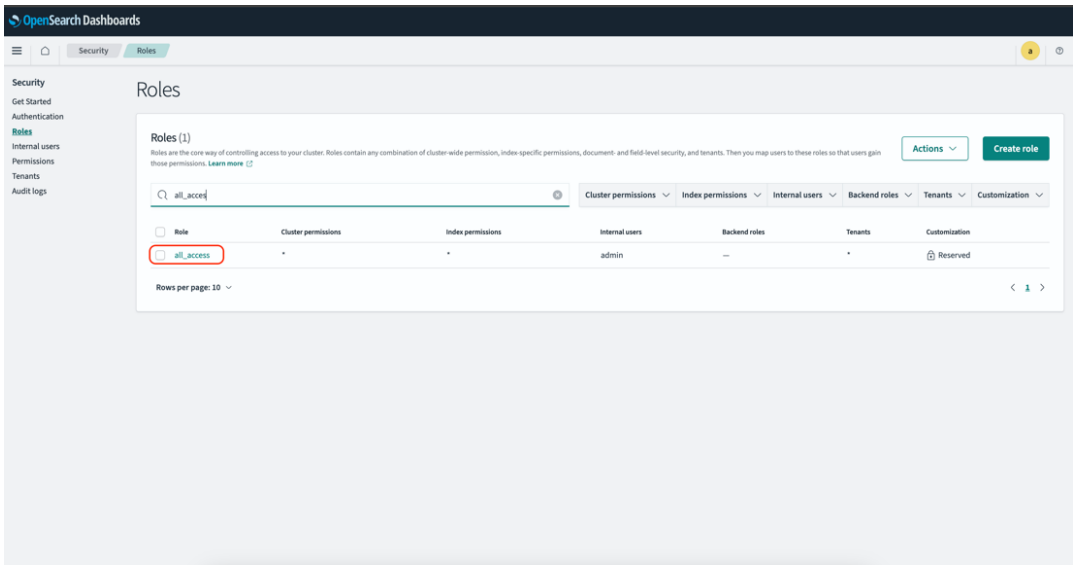


4. Select Roles and then choose the all_access role:

❖ Within the Security section, select "Roles."

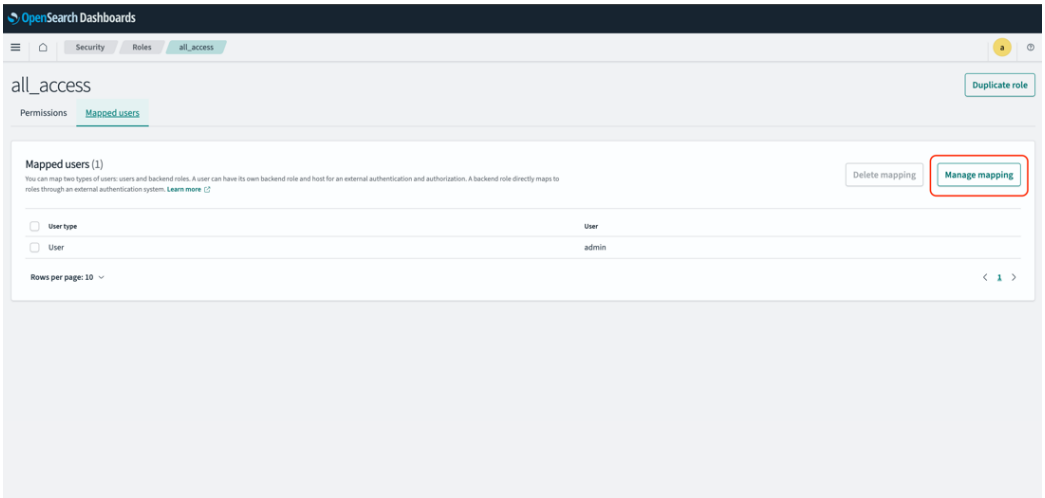


❖ Find and click on the all_access role.



5. Choose Mapped users and then Manage mapping:

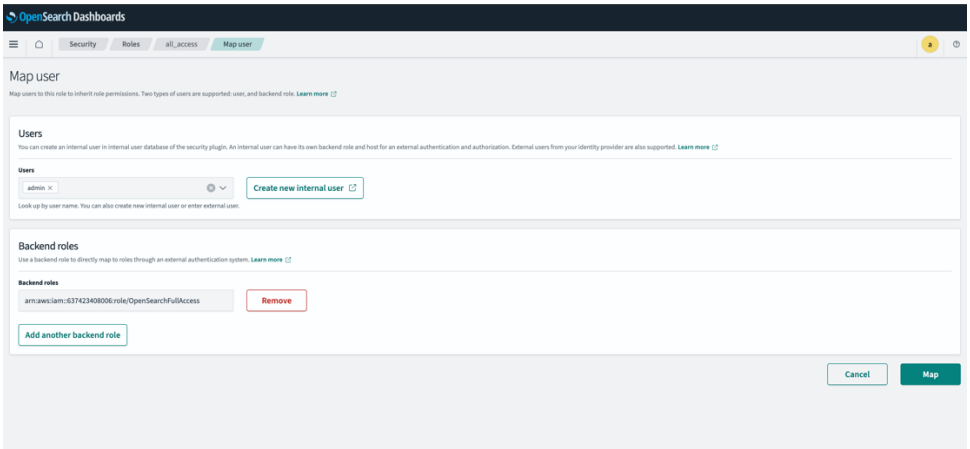
❖ Within the all_access role, select the "Mapped users" tab.



❖ Choose "Manage mapping."

6. Add the Lambda role ARN under Backend roles:

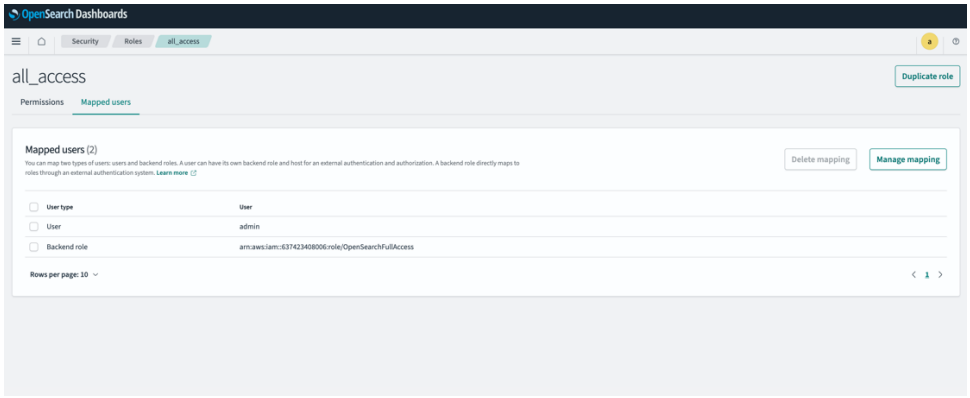
❖ Under "Backend roles," click on "Add backend role."



❖ 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."



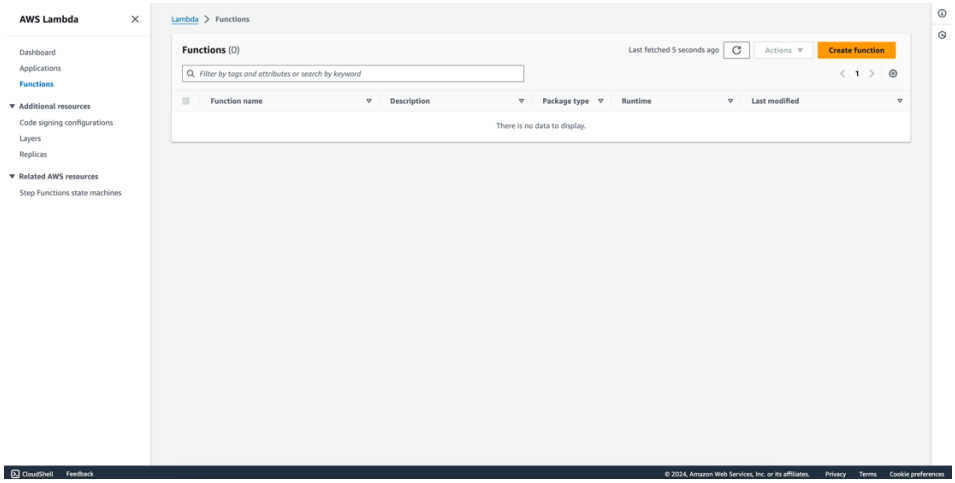
❖ Confirm that the user or role associated with the Lambda role shows up under "Mapped users."

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.

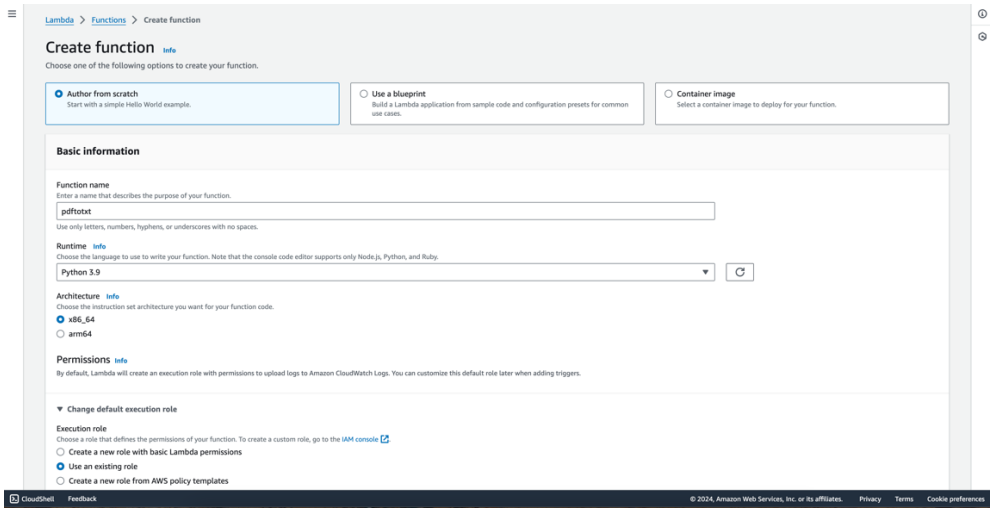
LAMBDA FUNCTION (pdftotxt)

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

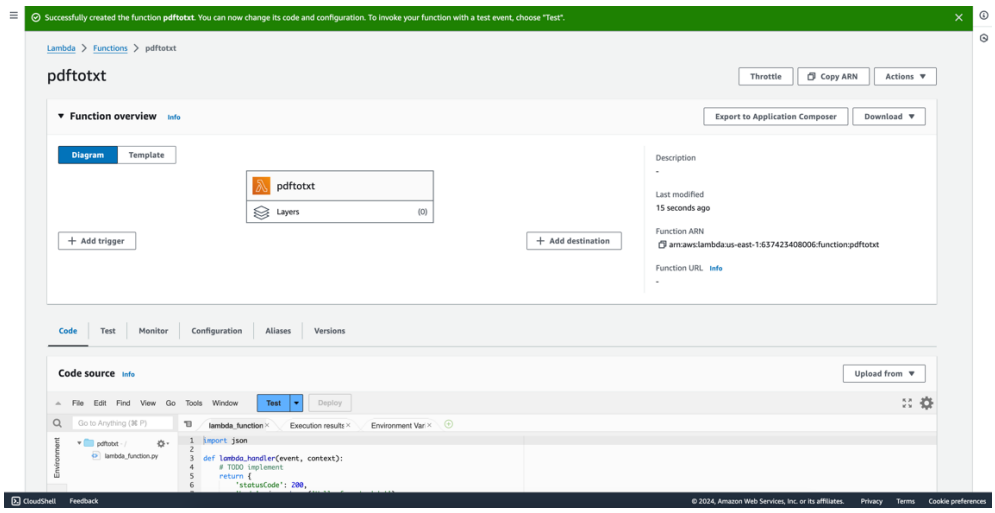
❖ Log in to the AWS Management Console.
❖ Navigate to the Lambda service.



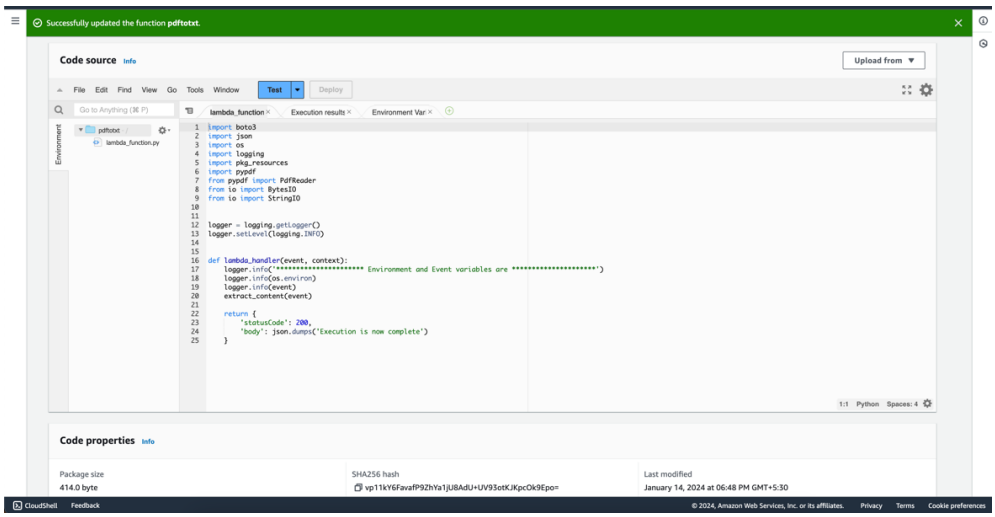
❖ Click on "Create function."
❖ Enter a name for your function (e.g., pdftotxt).



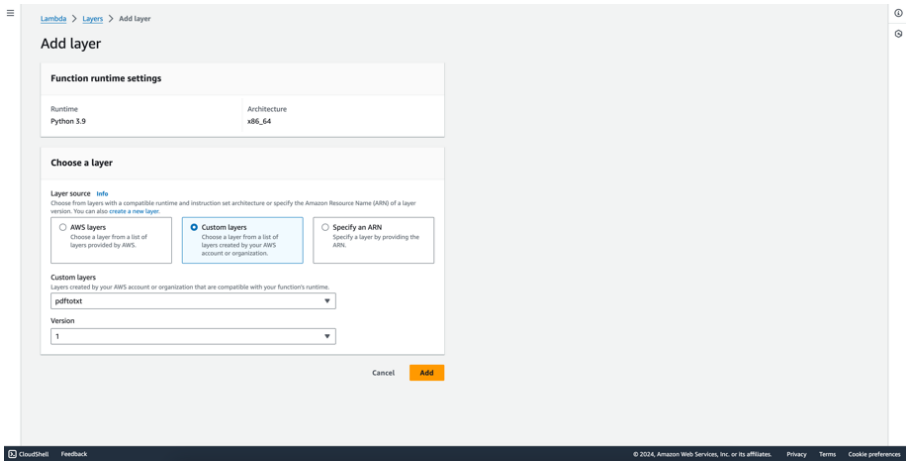
❖ Choose a runtime (e.g., Python, Node.js, etc.) that supports your PDF to text conversion library.



❖ 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.

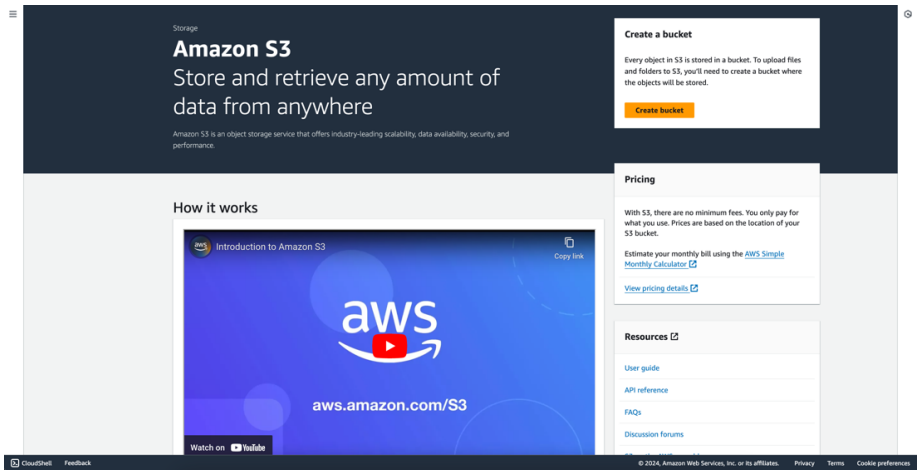


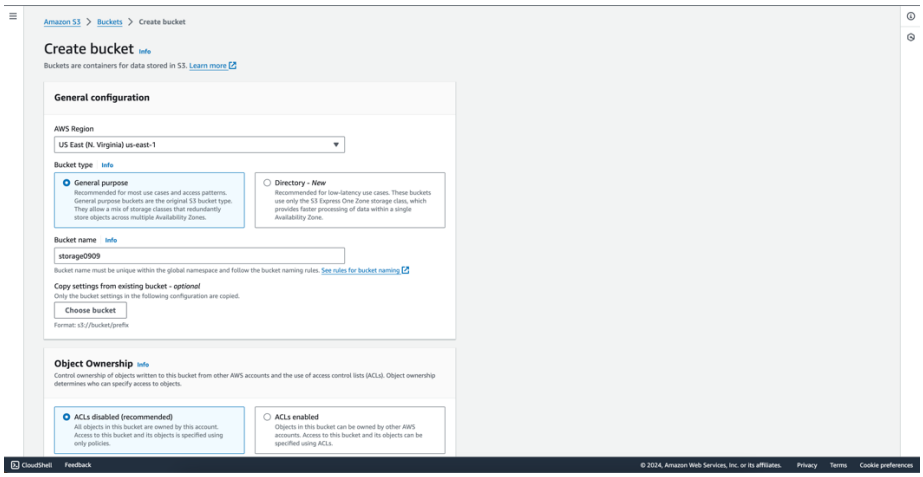
❖ Add a layer for the dependencies and upload the packages to the layer.

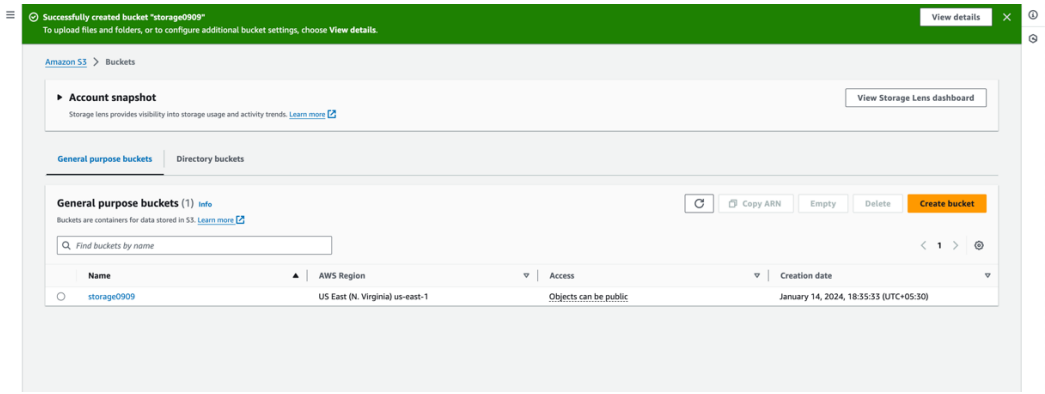


2. Configure Trigger for S3 PUT Event:

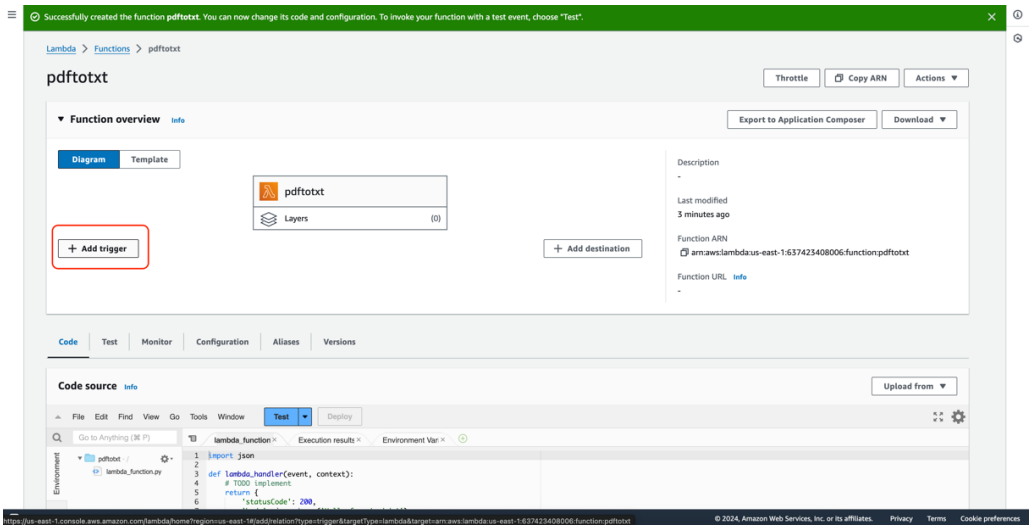
❖ Add the bucket in S3 (name : storage0909) buckets of AWS.



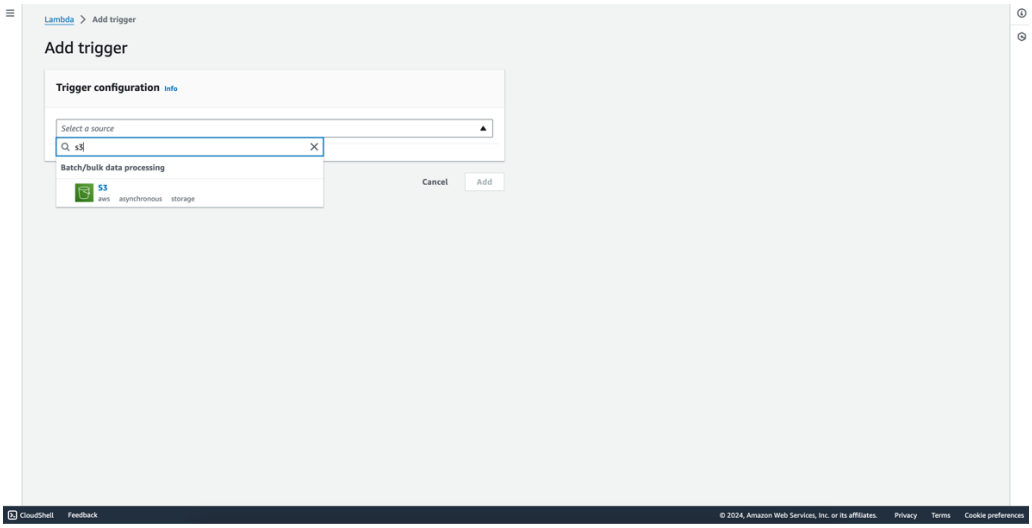




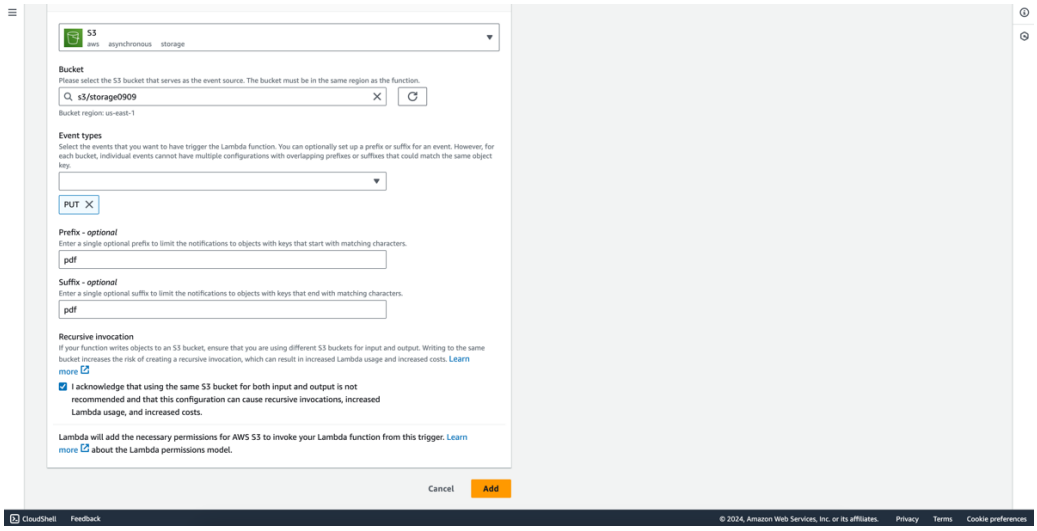
❖ In the Lambda function configuration, add a trigger.



❖ Choose "S3" as the trigger source.

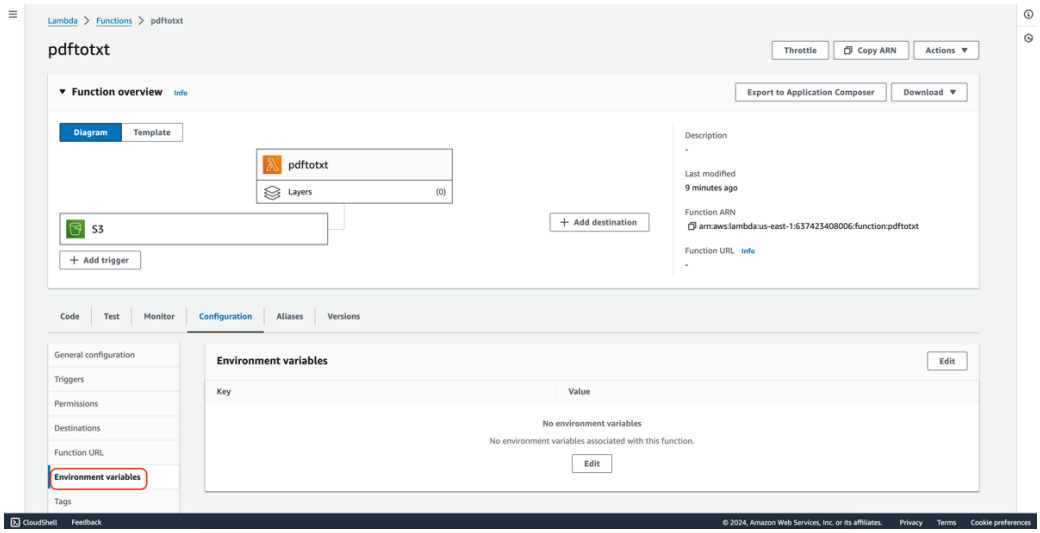


❖ Configure the trigger to listen to the PUT event on the S3 bucket where the sample PDF file is uploaded (content store S3 bucket).

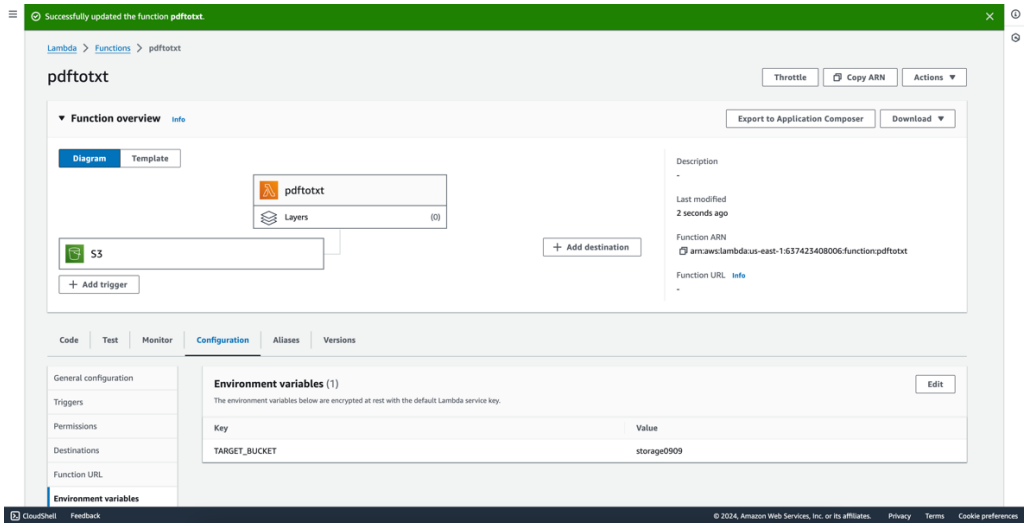


3. Configure Environmental Variable:

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



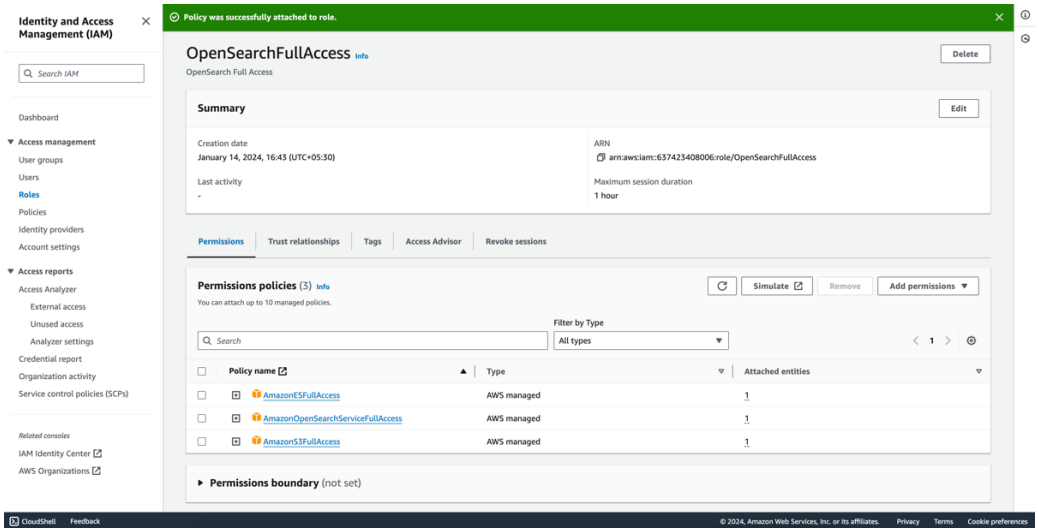
❖ Scroll down to the "Environment variables" section.



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

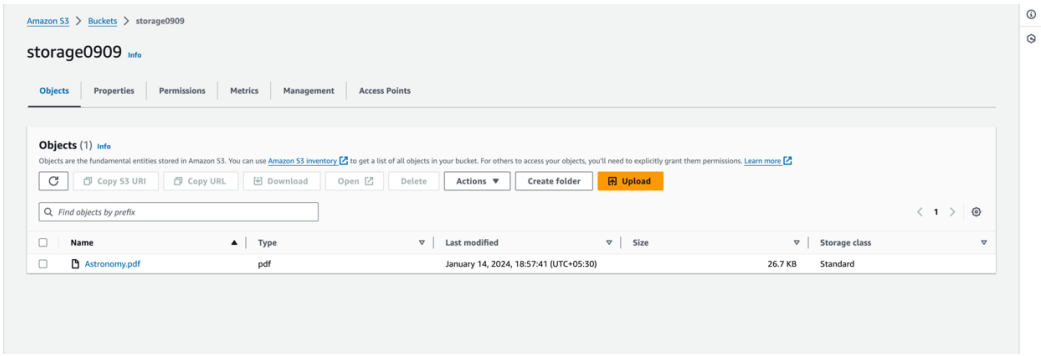
4. Assign IAM Role:

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

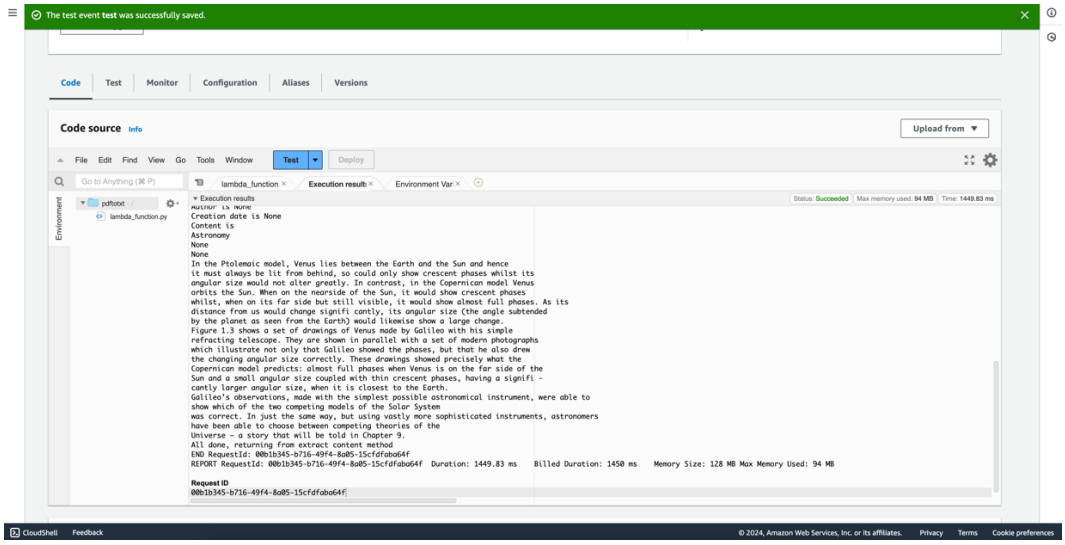


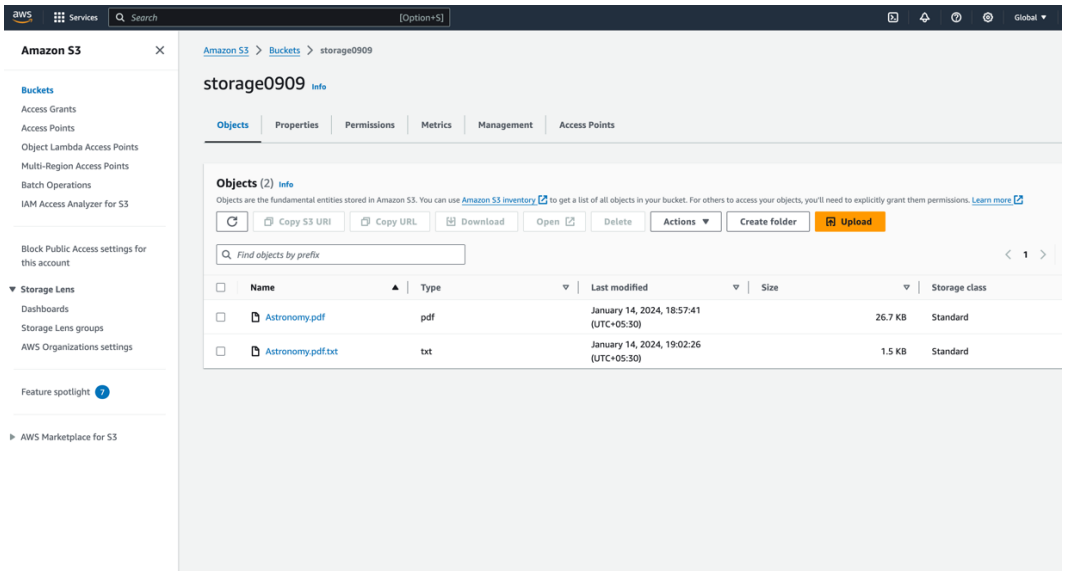
❖ Choose an existing role or create a new one that has the necessary permissions to access S3 and any other services required for the conversion process.
❖ Ensure that the role allows the Lambda function to read from the source S3 bucket and write to the target S3 bucket.
❖ Save the changes.

❖ Add the pdf to the S3 Bucket.



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





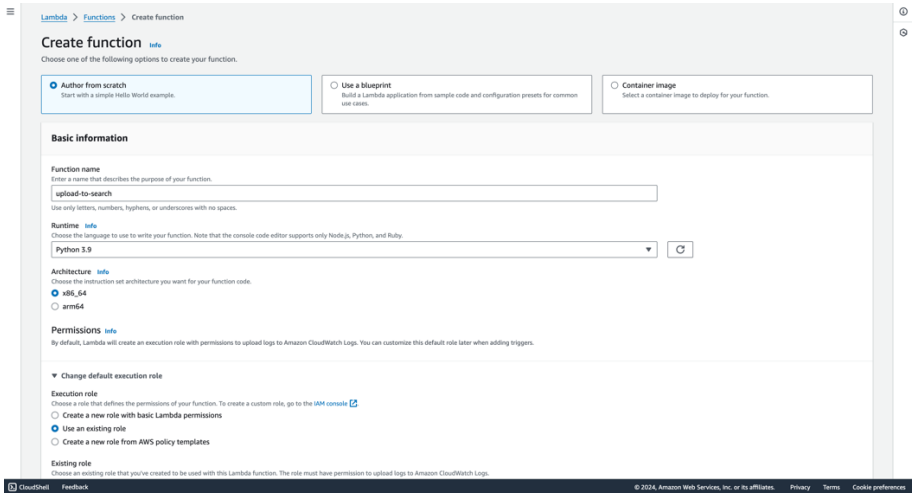
The file has been converted from “pdf” format to “txt” format.

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.

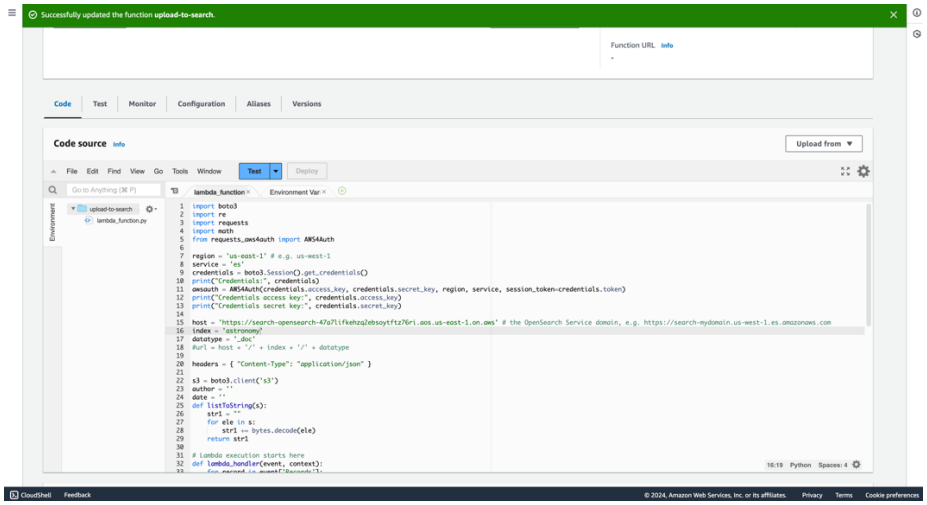
LAMBDA FUNCTION (upload-to-search)

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."

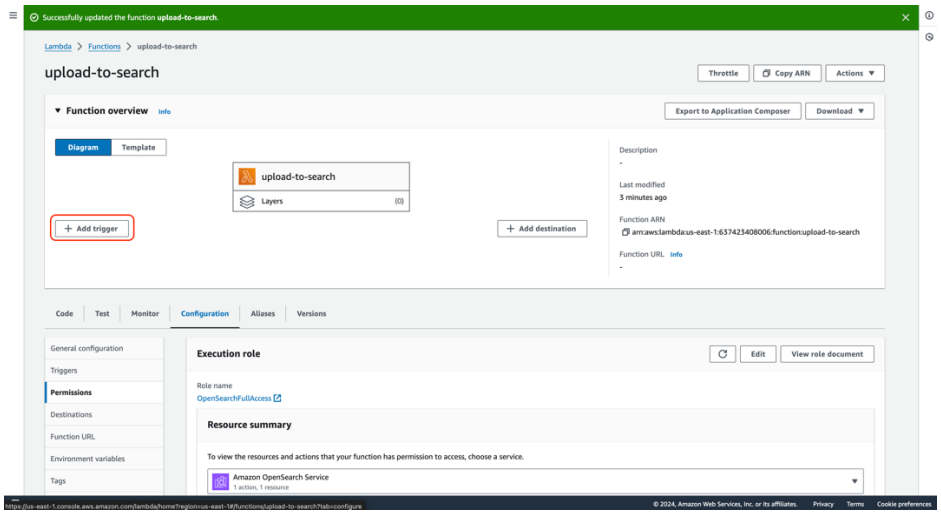


- ❖ Enter a name for your function (e.g., upload-to-search).
- ❖ Choose the same runtime used in the previous Lambda function (e.g., Python).
- ❖ In the "Function code" section, upload the sample.py file and ensure that it contains a function named handler that processes the text and uploads it to OpenSearch.

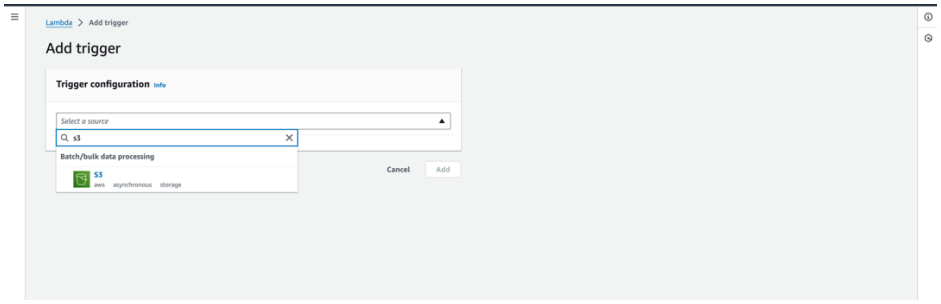


2. Configure Trigger for S3 PUT Event:

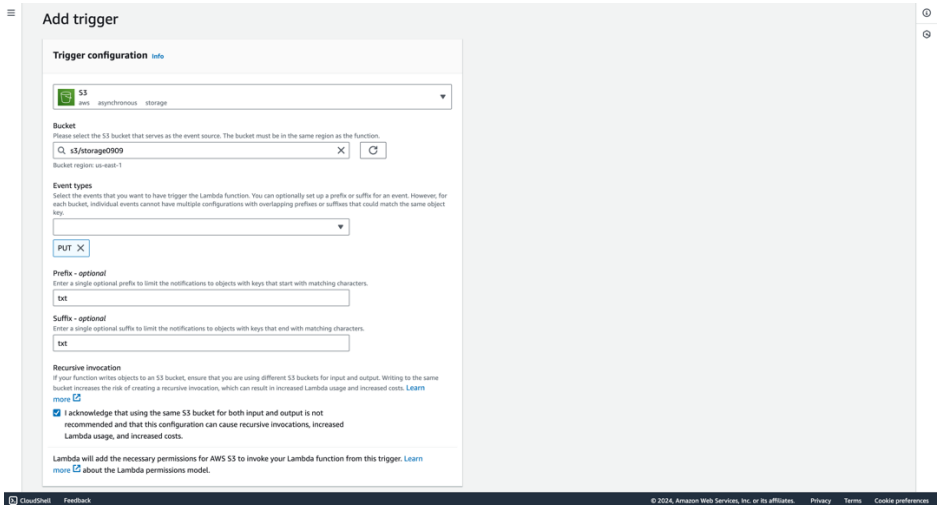
❖ In the Lambda function configuration, add a trigger.



❖ Choose "S3" as the trigger source.



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

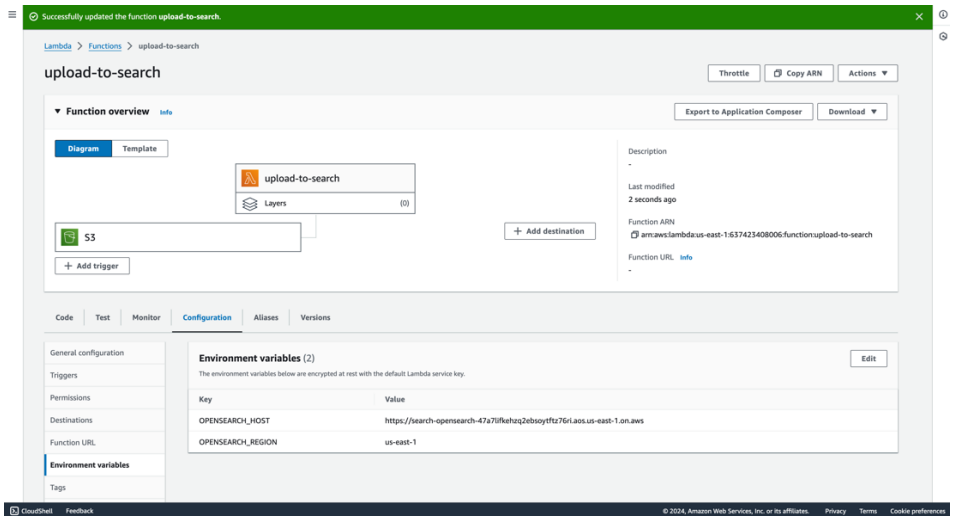


3. Configure Runtime Settings:

- 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.

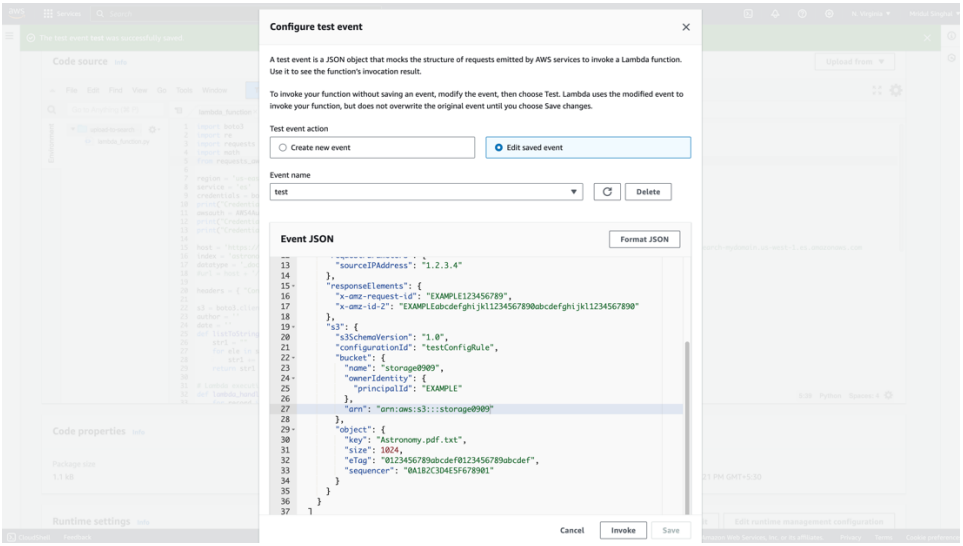


- Add variables named OPENSEARCH_REGION and OPENSEARCH_HOST with their respective values.

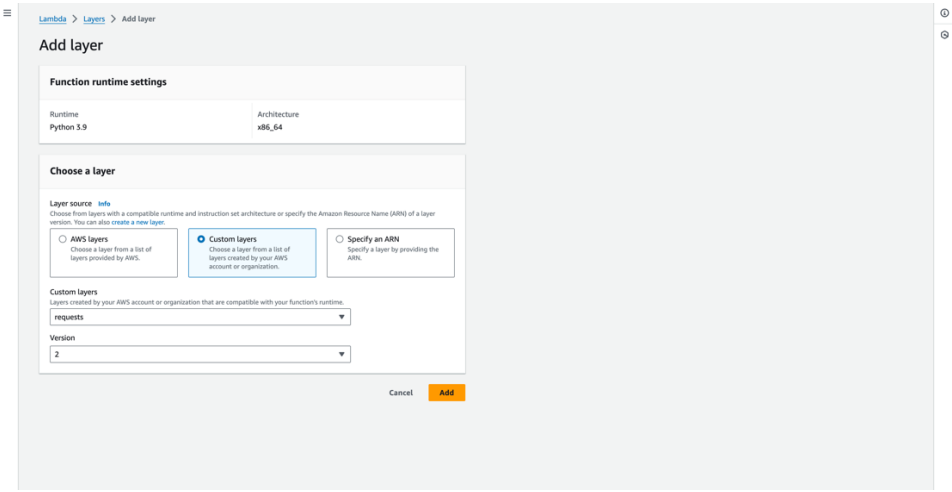
5. Assign IAM Role:

- 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.

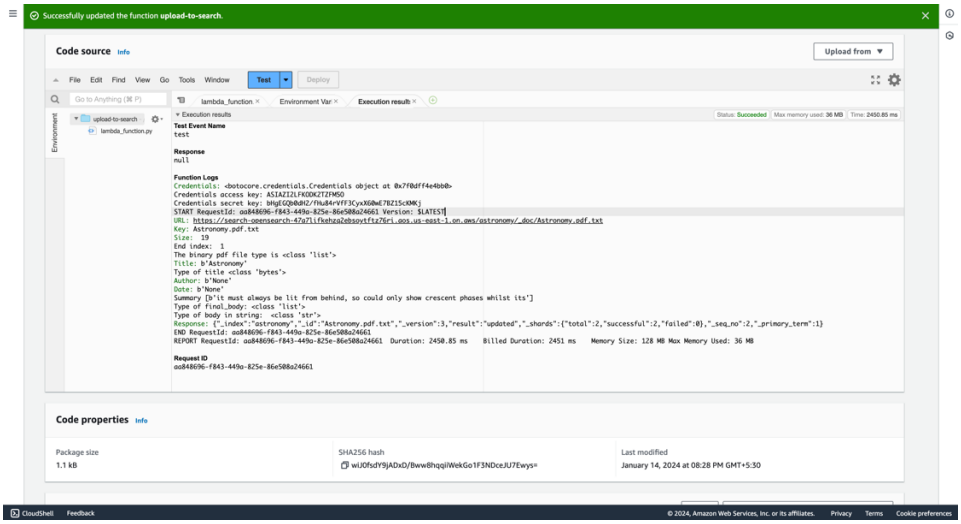
❖ Configure the test for the lambda function.



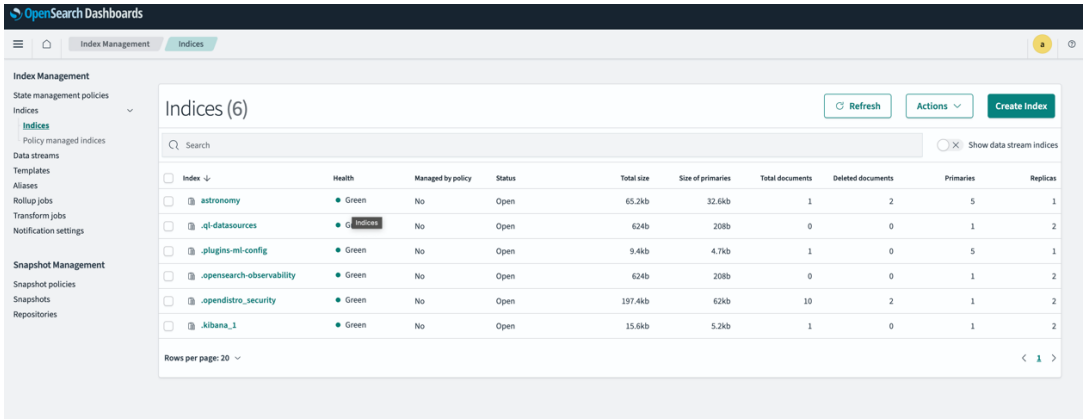
❖ Add the dependency layer containing the “requests” dependency.

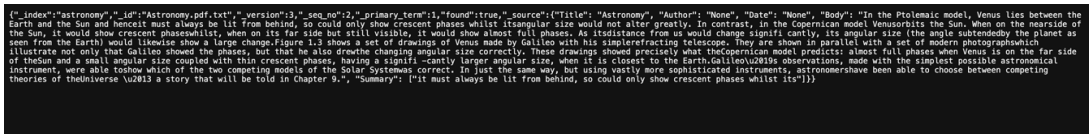


❖ Now execute the function by clicking on Test.



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



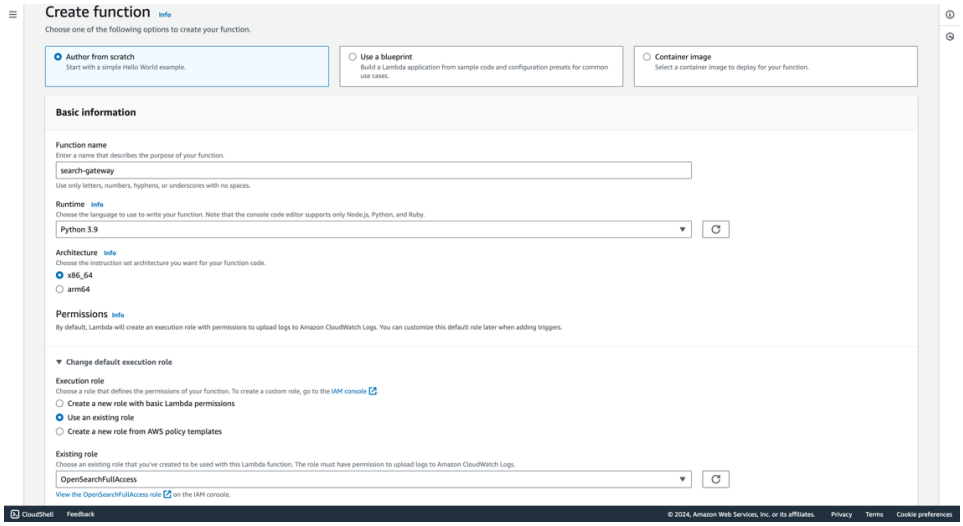


By following these steps, you should have a Lambda function (upload-to-search) configured to process the text file generated by the previous Lambda function (pdftotxt) and upload the text to OpenSearch. The function is triggered by an S3 PUT event, and the necessary runtime settings, environmental variables,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

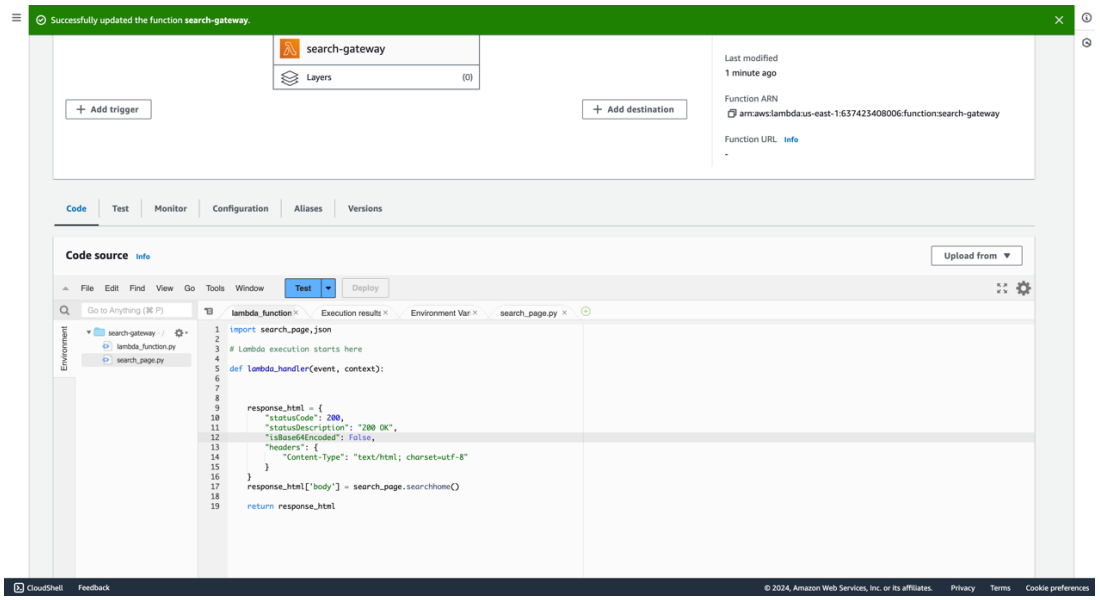
LAMBDA FUNCTION (search-gateway)

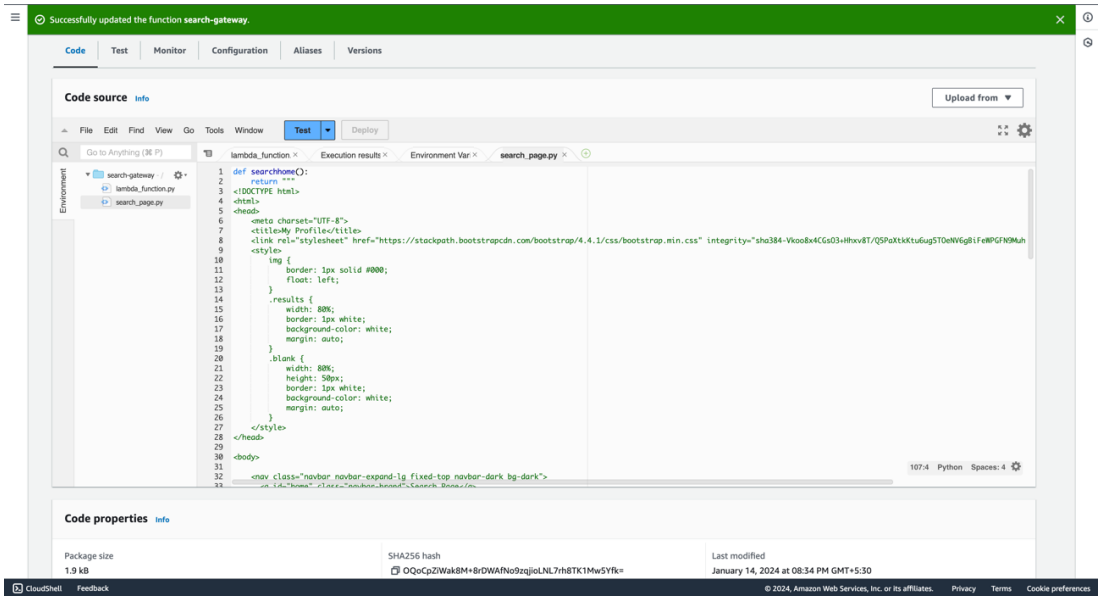
1. Lambda Function for Search Gateway (search-gateway):

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



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

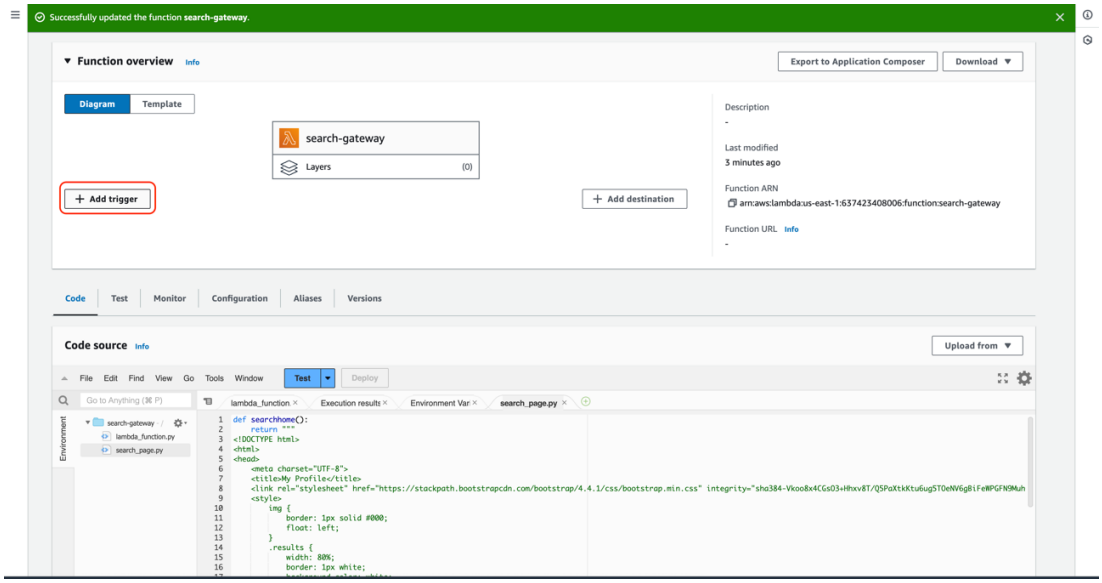




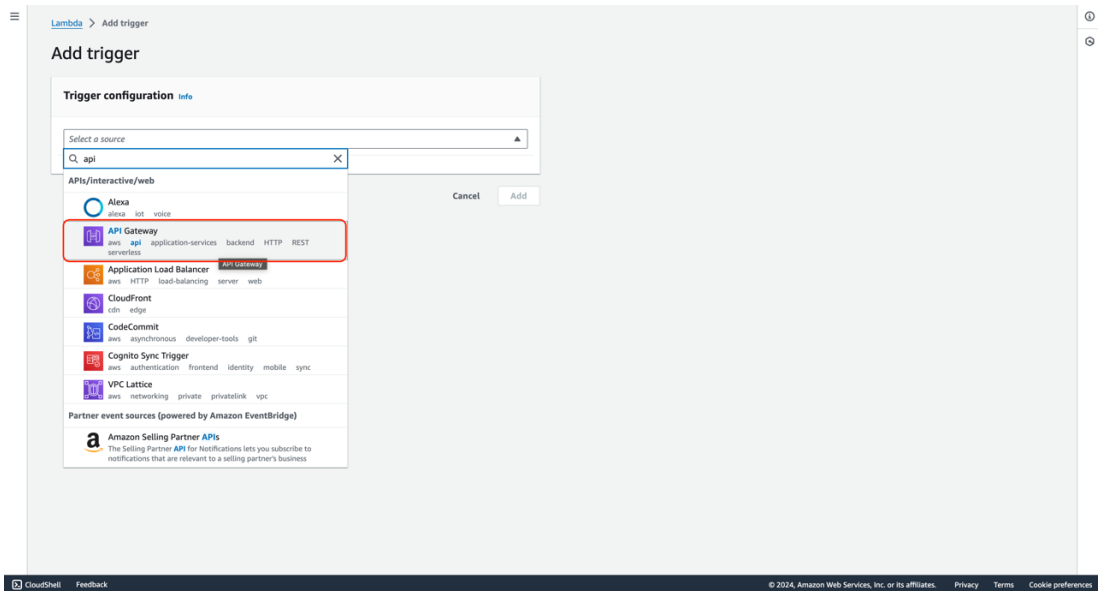
- In the "Function code" section, upload the sample.py file and ensure that it contains a function named lambda_handler to handle incoming HTTP requests and run search queries on OpenSearch.

2. Configure Trigger for API Gateway:

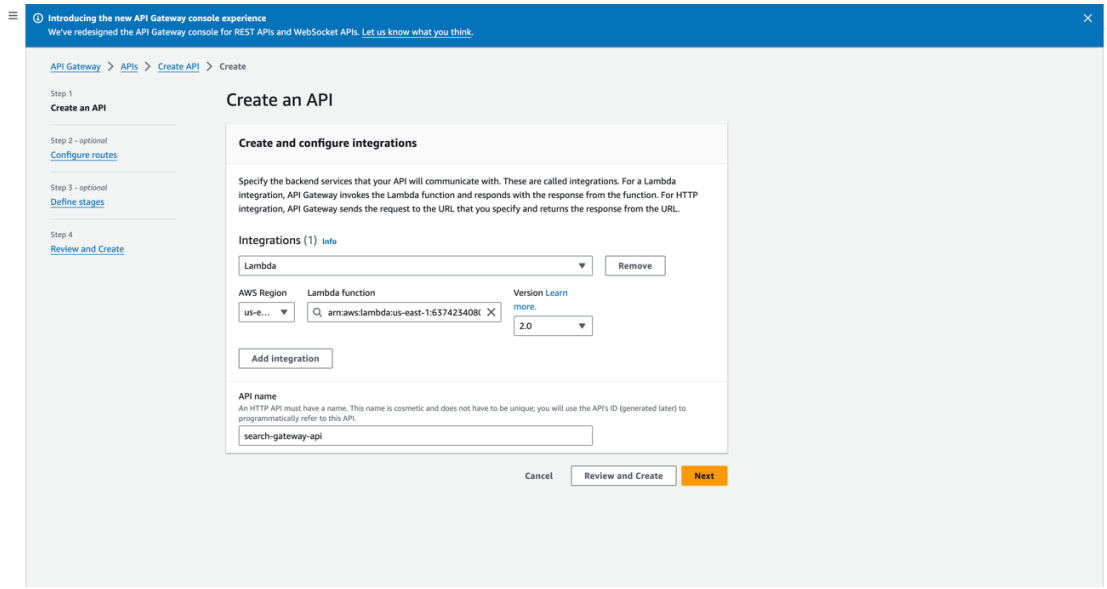
- In the Lambda function configuration, add a trigger.

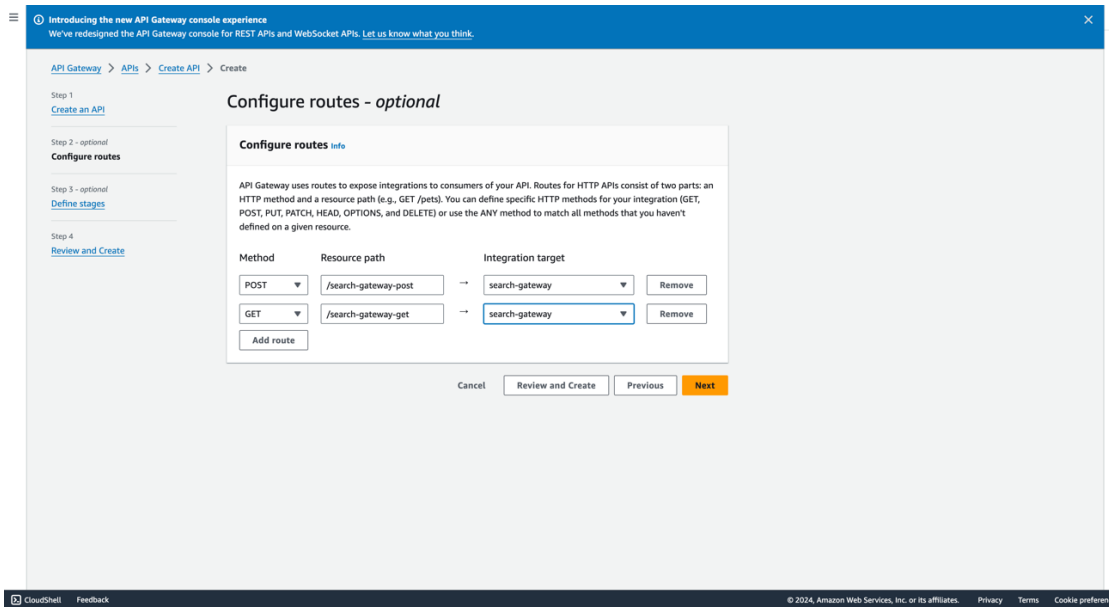


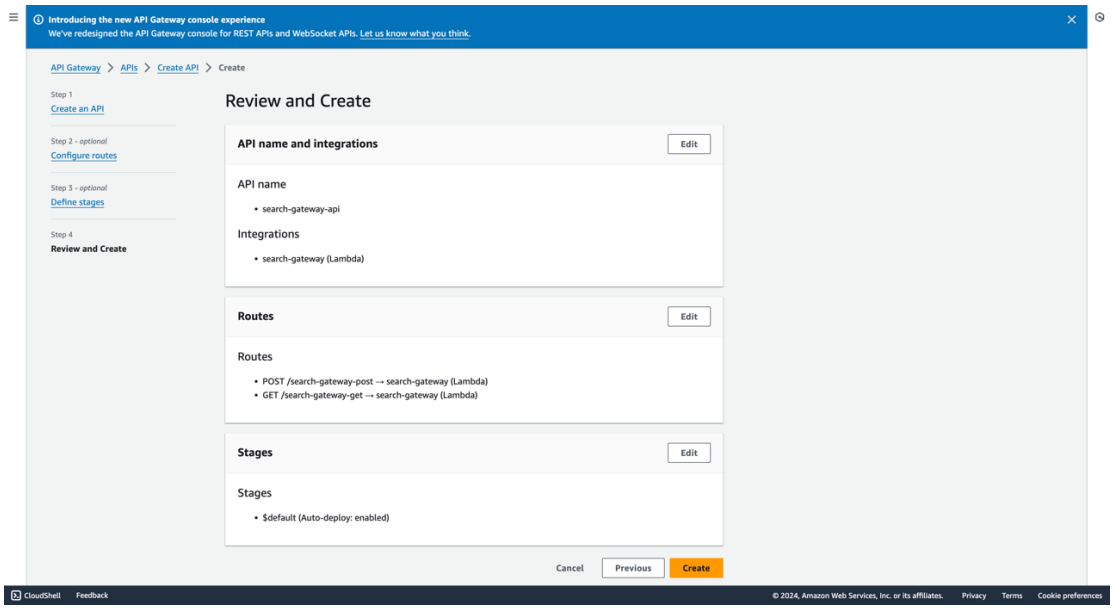
- Choose "API Gateway" as the trigger source.

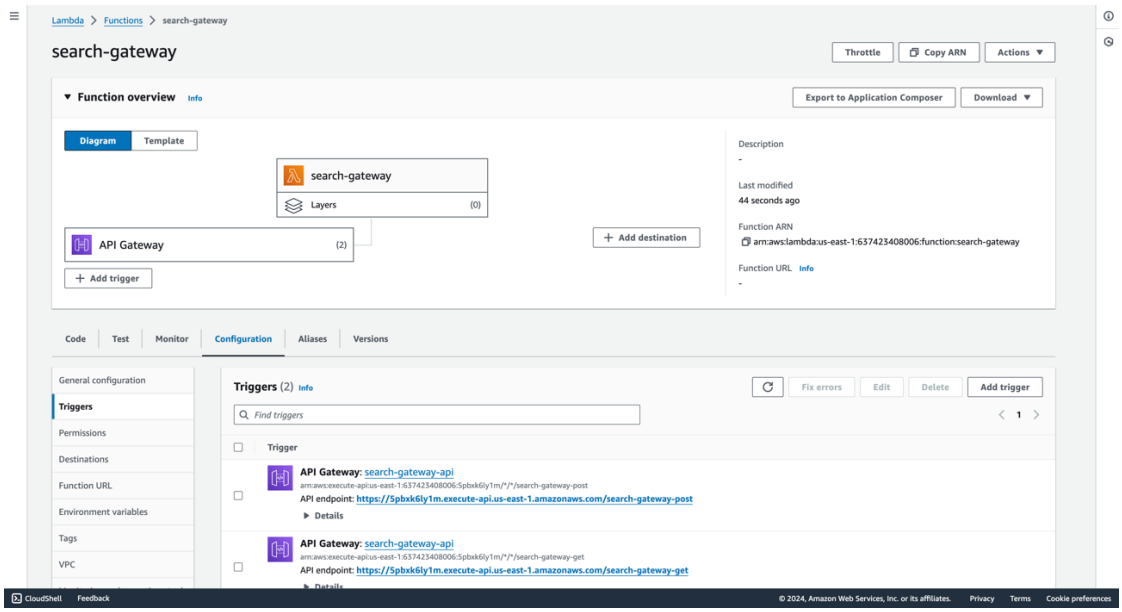


- Configure the trigger to receive requests from the API Gateway created as part of this exercise.


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
Remove
AWS Region
Lambda function
Version Learn more.
us-e...
arn:aws:lambda:us-east-1:637423408(
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.
search-gateway-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
Remove
GET
/search-gateway-get
search-gateway
Remove
Add route
Cancel
Review and Create
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


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
Edit
API name
search-gateway-api
Integrations
search-gateway (Lambda)
Routes
Edit
Routes
POST /search-gateway-post → search-gateway (Lambda)
GET /search-gateway-get → search-gateway (Lambda)
Stages
Edit
Stages
$default (Auto-deploy: enabled)
Cancel
Previous
Create
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

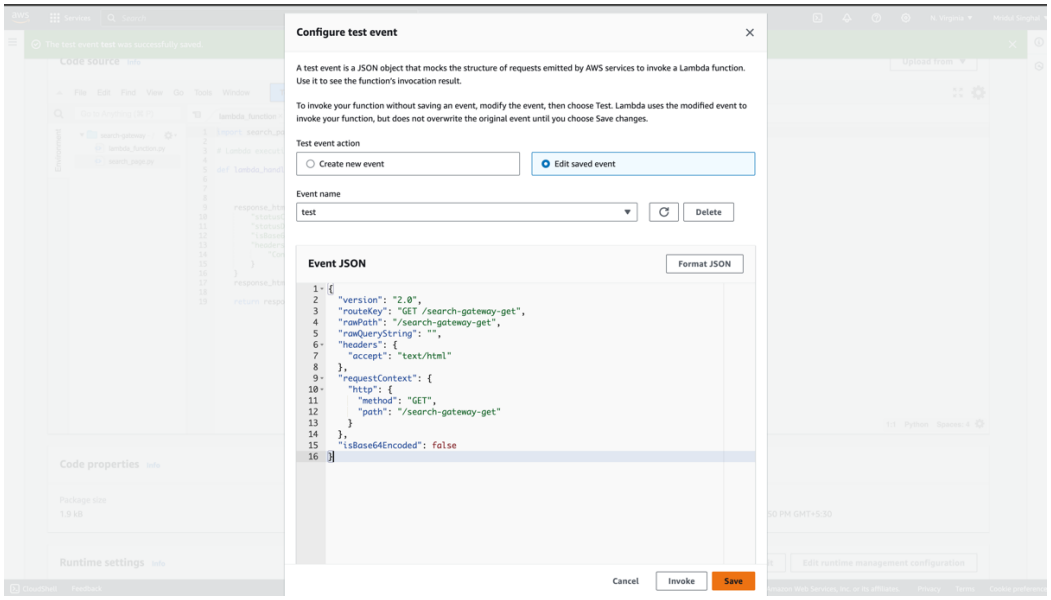


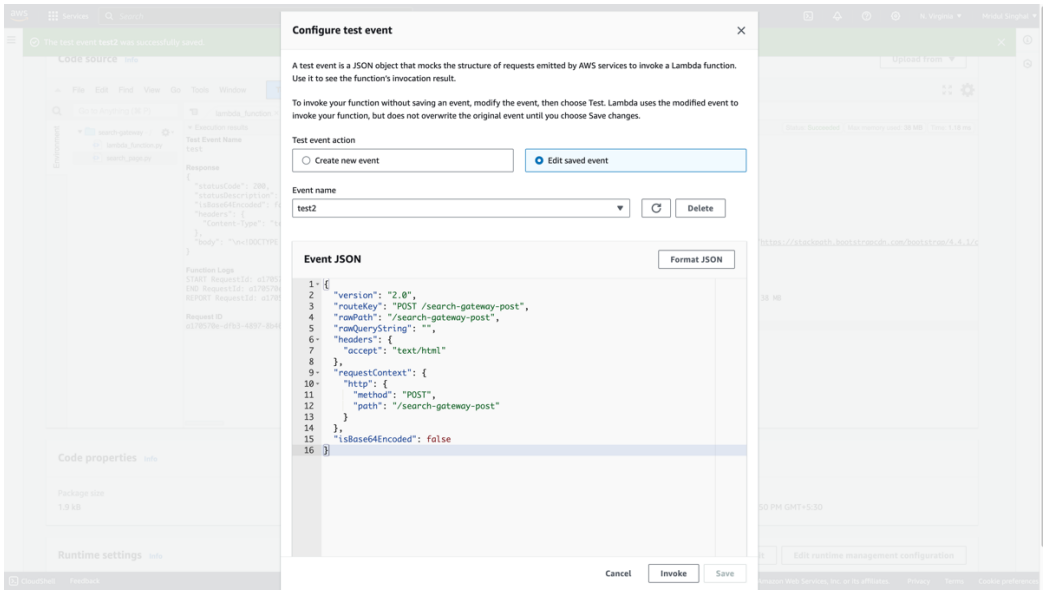
3. Configure Runtime Settings:

- 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).

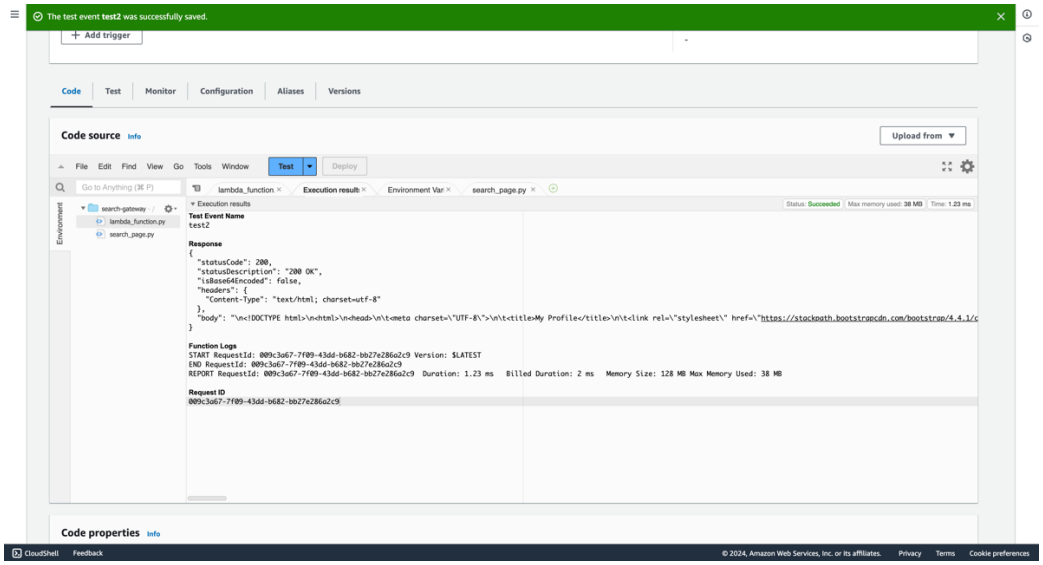
4. Assign IAM Role:

- 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.

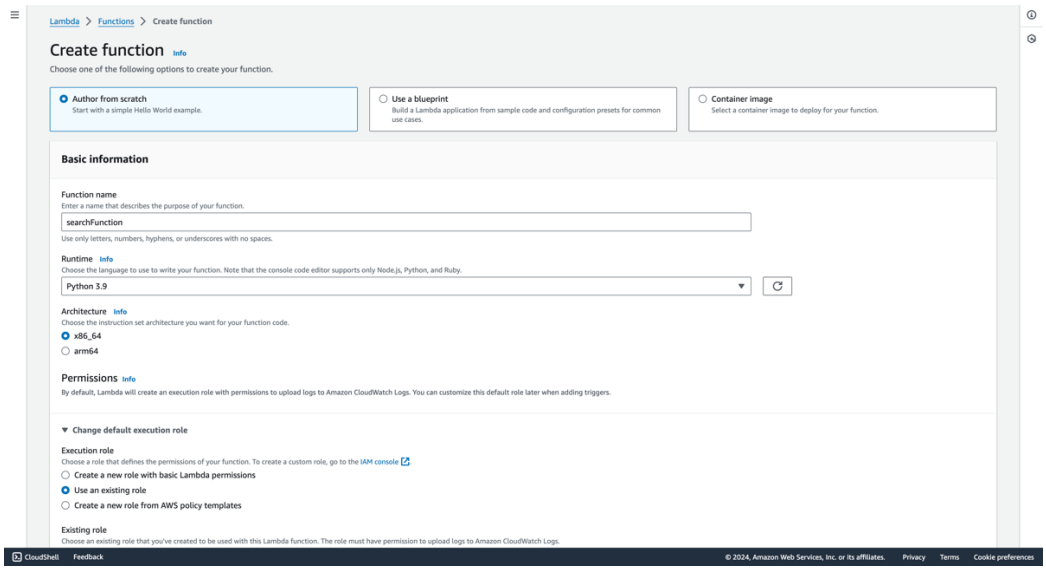


By following these steps, you should have a Lambda function (search-gateway) configured to process incoming HTTP requests from the API Gateway, run search queries on OpenSearch, and return the search results. The function is triggered by API requests, and the necessary runtime settings and IAM role are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

LAMBDA FUNCTION (searchFunction)

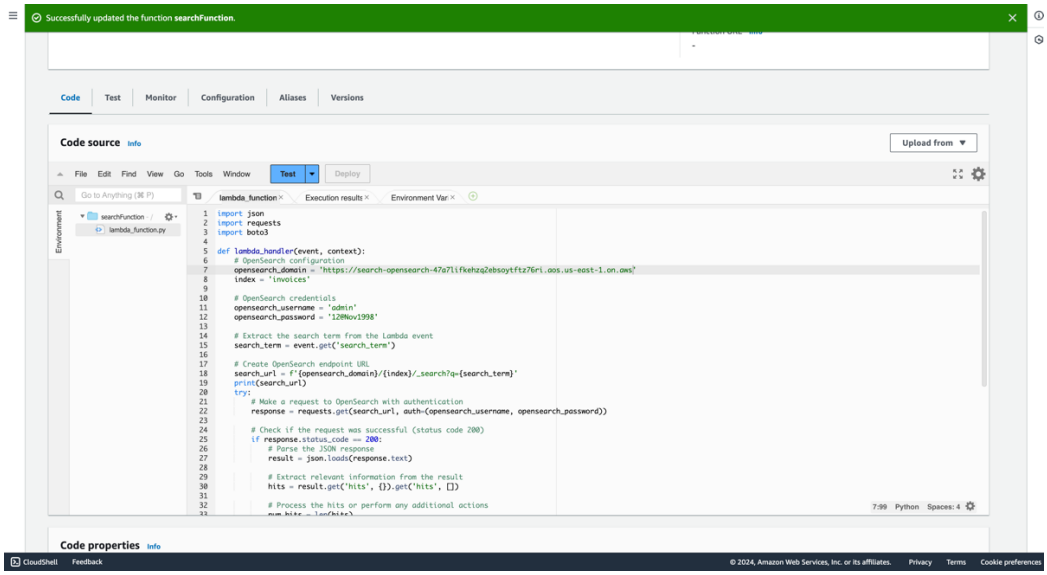
1. Lambda Function for Search Function (searchFunction):

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



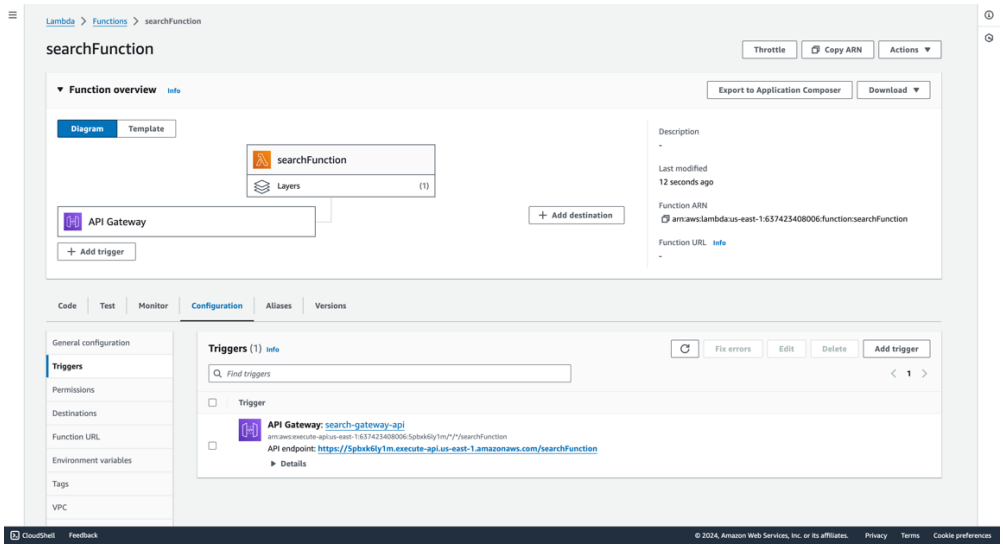
- ❖ Enter a name for your function (e.g., searchFunction).
- ❖ Choose the appropriate runtime (e.g., Python).

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

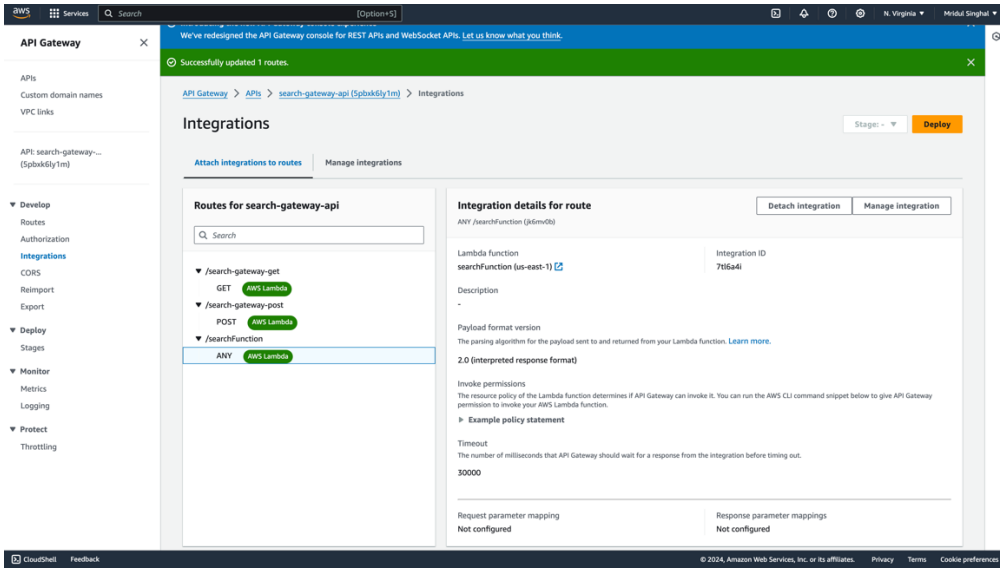


2. Configure Trigger for API Gateway:

❖ In the Lambda function configuration, add a trigger.



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

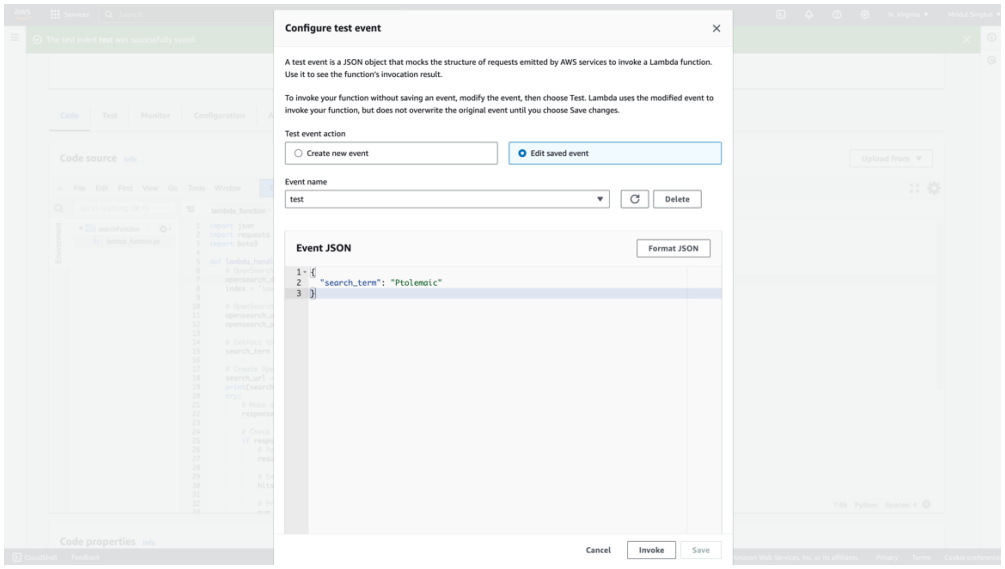


3. Configure Runtime Settings:

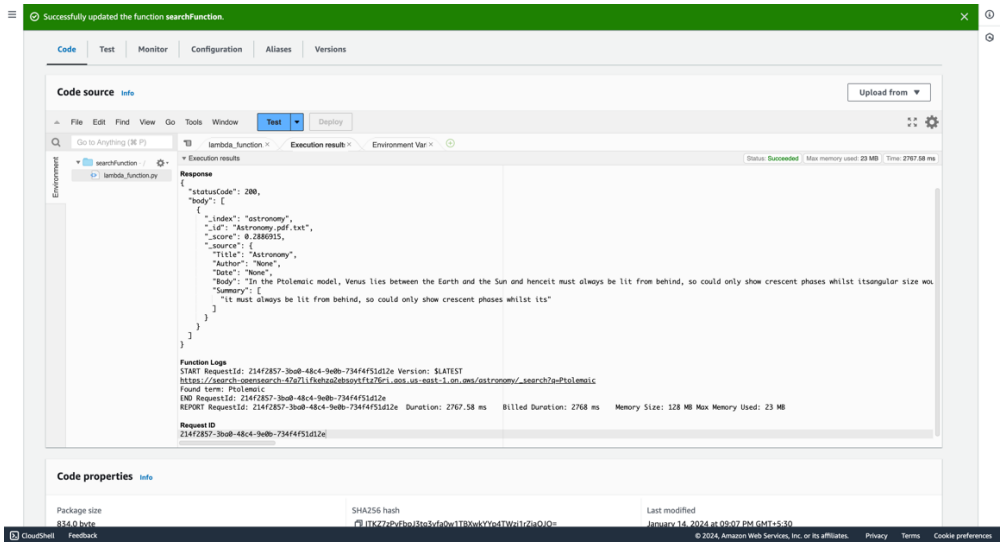
- In the Lambda function configuration, go to the "Configuration" tab.
- Set the "Handler" field to searchFunction.lambda_handler (assuming searchFunction.py contains a lambda_handler function).

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.

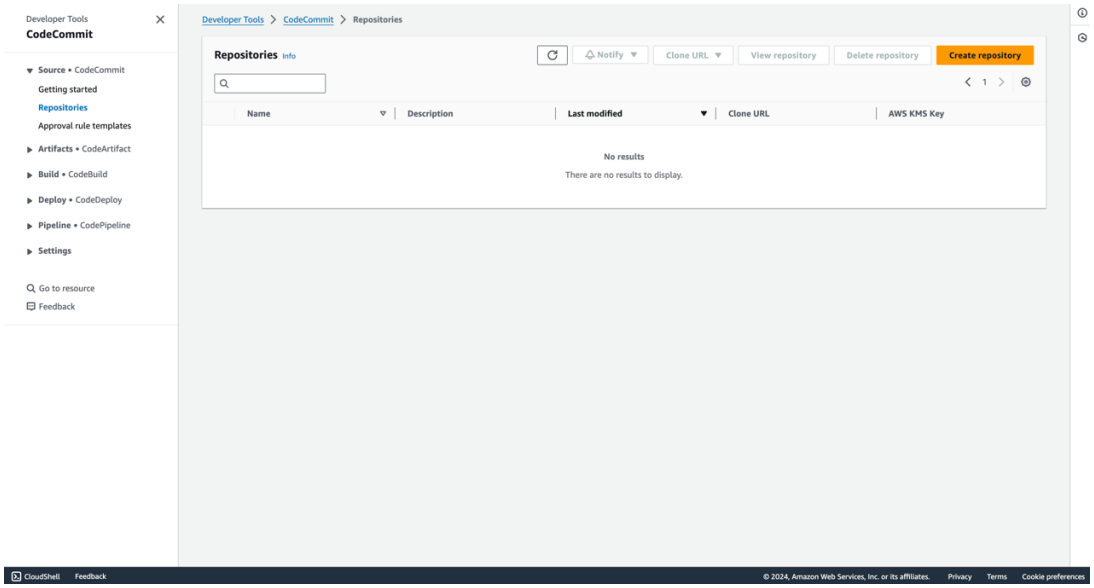


By following these steps, you should have a Lambda function (searchFunction) configured to receive data entered into the search field through API requests from the API Gateway. The function processes the search query, queries OpenSearch, and returns the search results. The function is triggered by API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

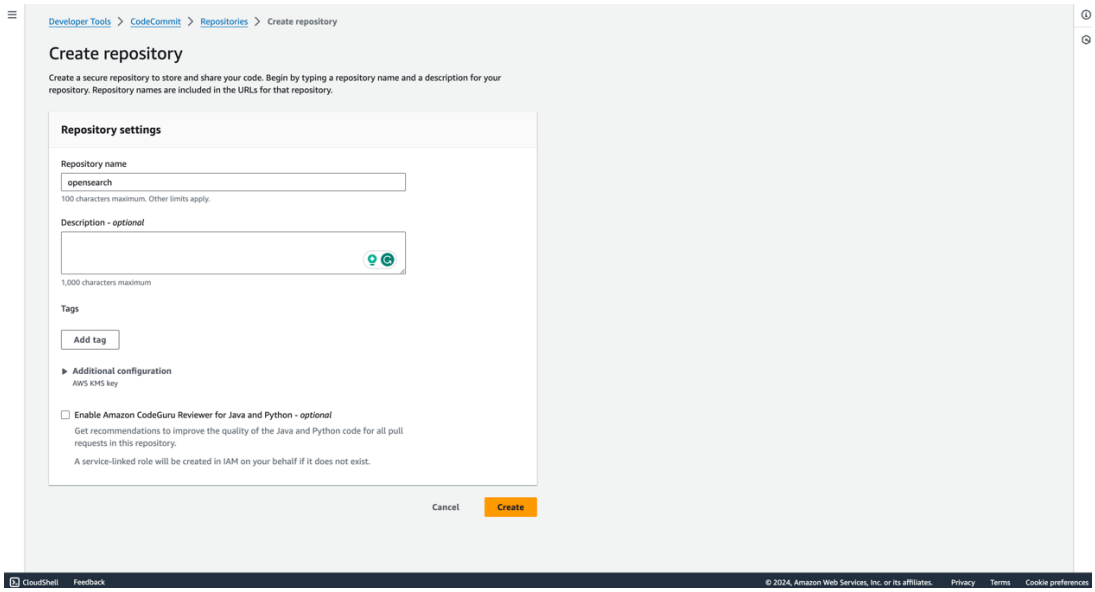
SETTING UP CODE COMMIT

1. Create a CodeCommit Repository using the AWS Console:

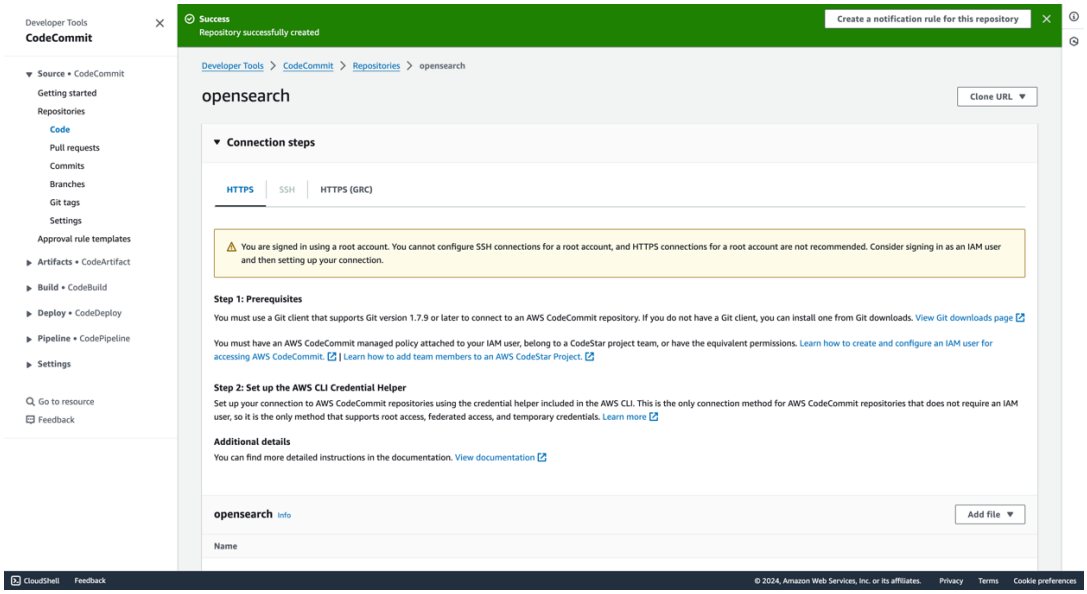
❖ Log in to the AWS Management Console.
❖ Navigate to the CodeCommit service.



❖ Click on "Create repository."



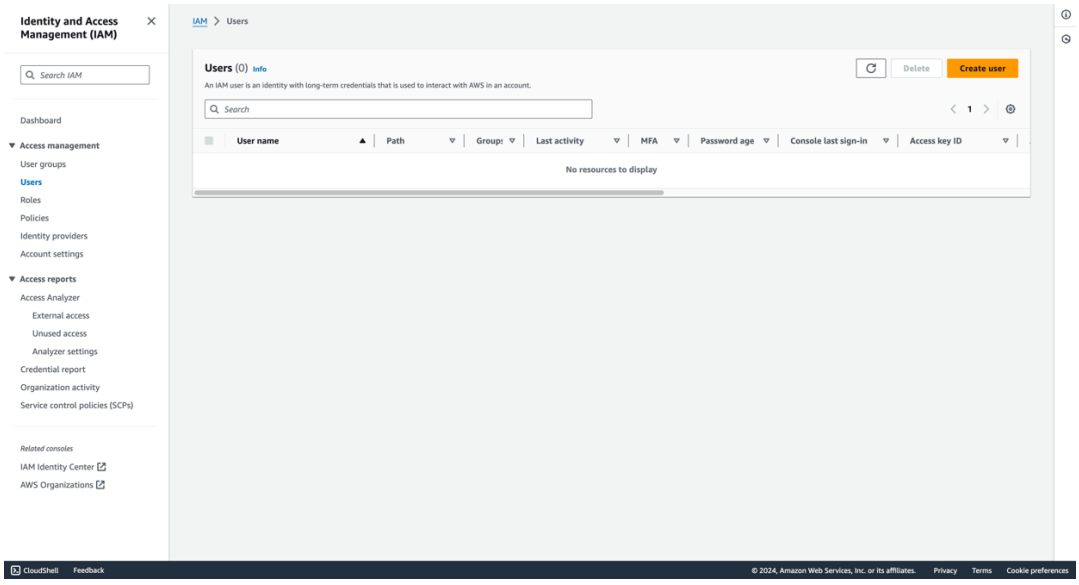
❖ Enter a repository name and configure other settings as needed.



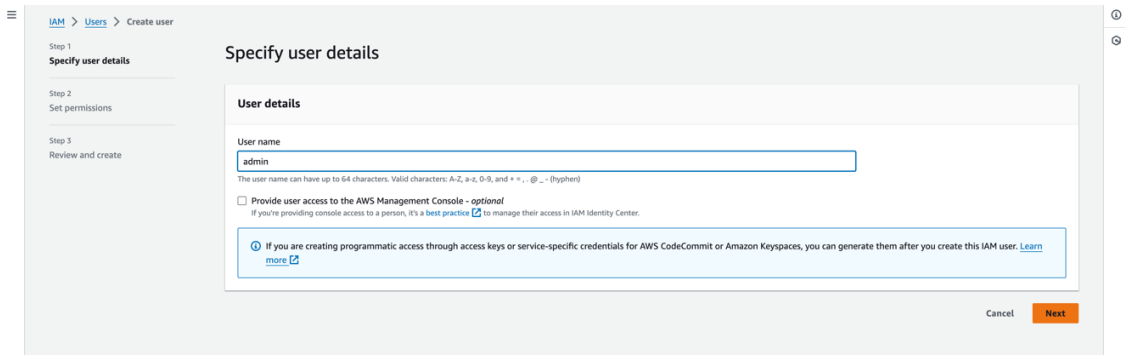
❖ Click "Create repository."

2. Create an IAM User with CodeCommit Credentials:

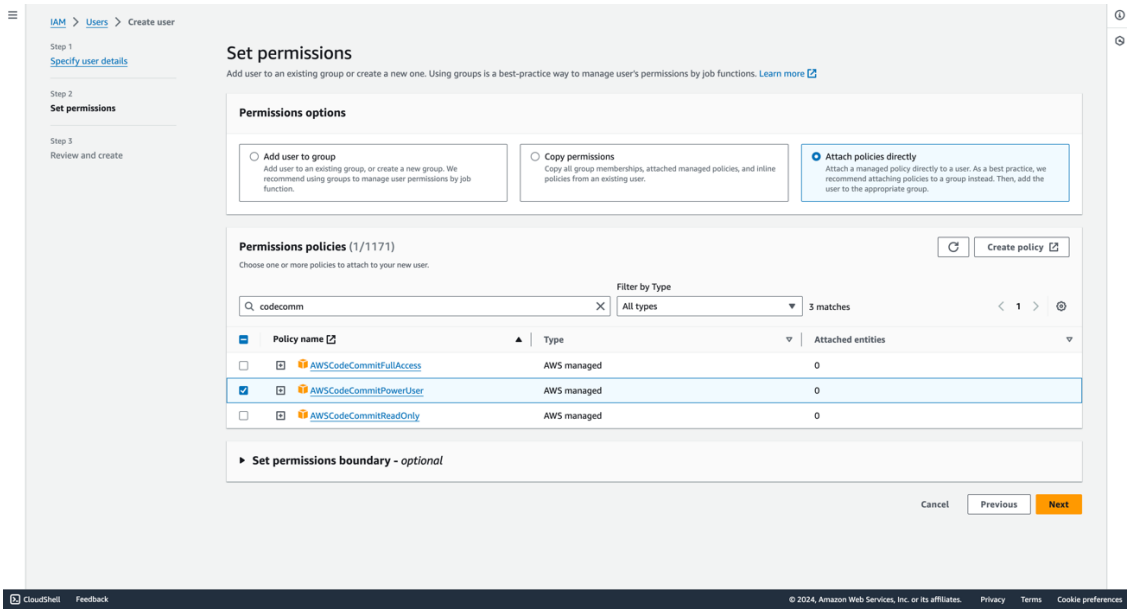
❖ Navigate to the IAM service in the AWS Console.

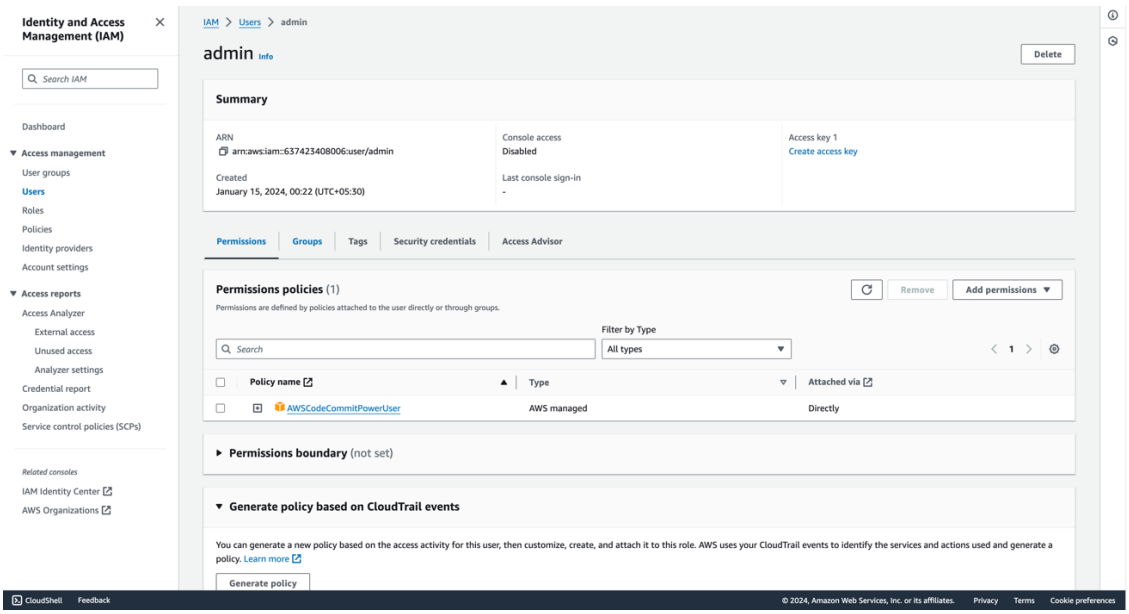


❖ Create a new IAM user with programmatic access.

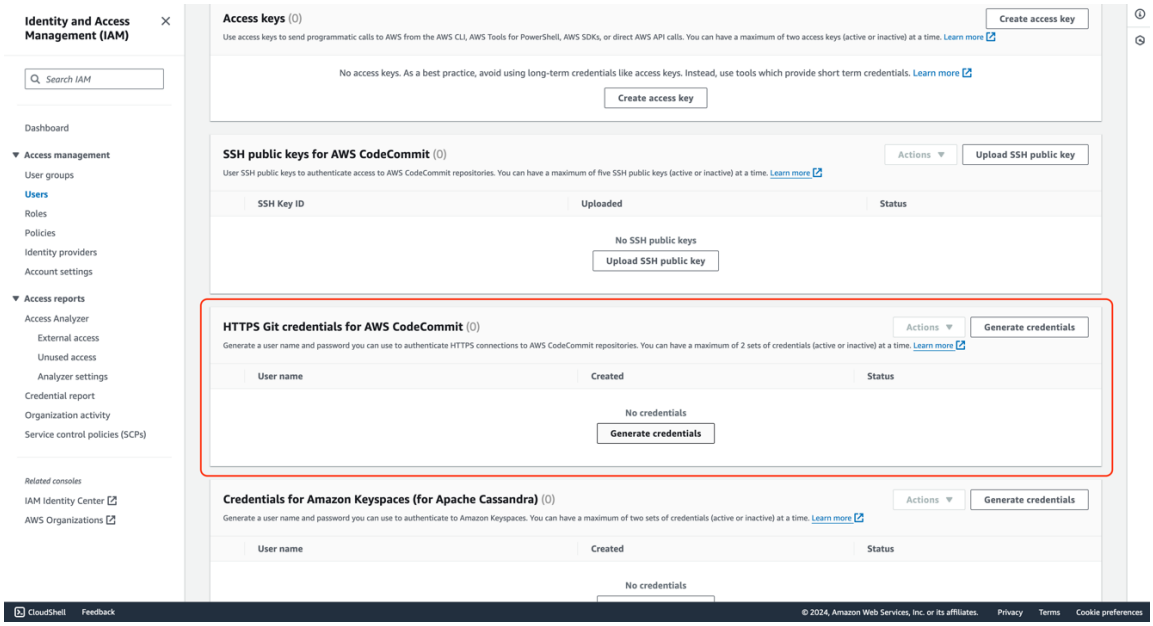


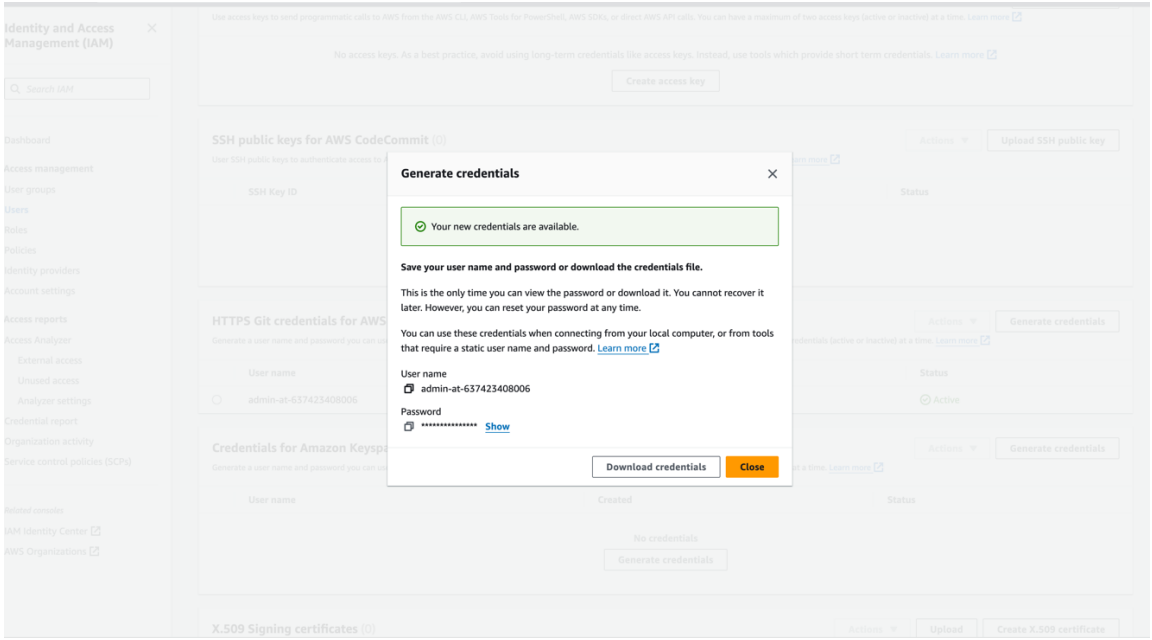
❖ Attach the necessary permissions for CodeCommit (e.g., AWSCodeCommitPowerUser policy).



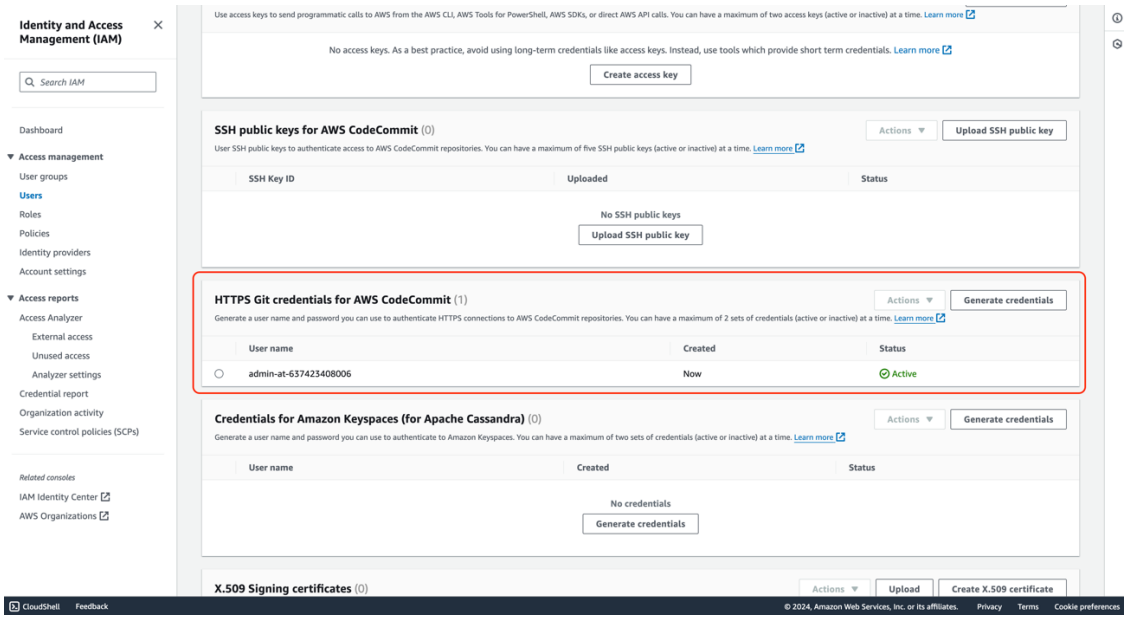


❖ Note down the access key ID and secret access key generated for this user.





❖ Copy the credentials and save them somewhere safe.



❖ 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.

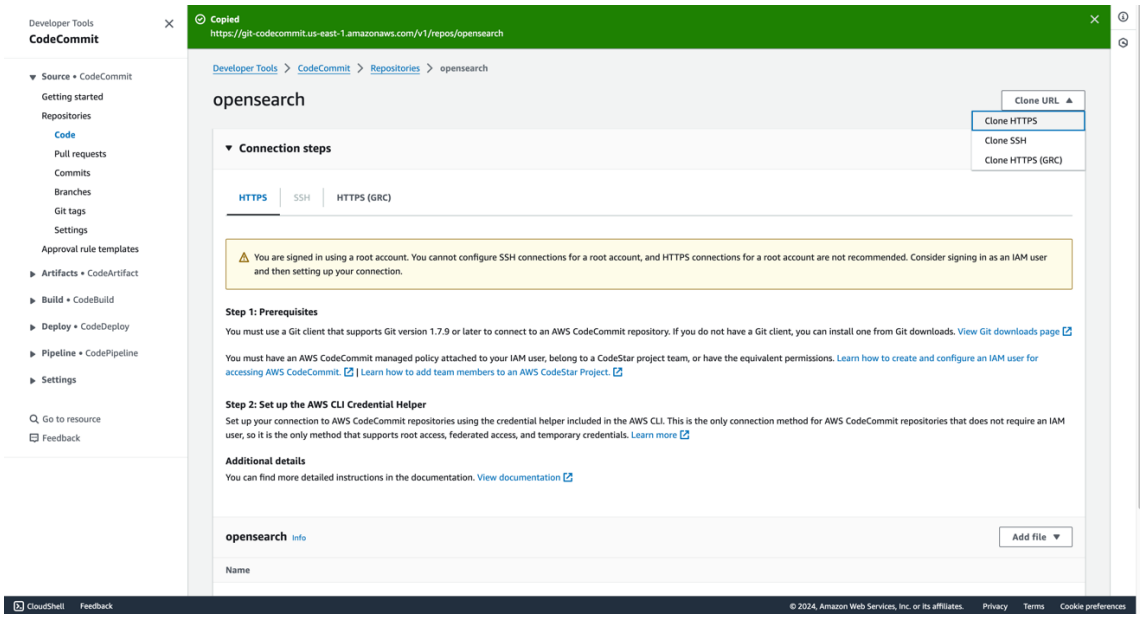
4. Create an Empty Directory on Your Local Machine:

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

mkdir opensearch
cd opensearch

5. Use Git Clone to Set Up Remote Repository:

❖ Go to your CodeCommit repository in the AWS Console.



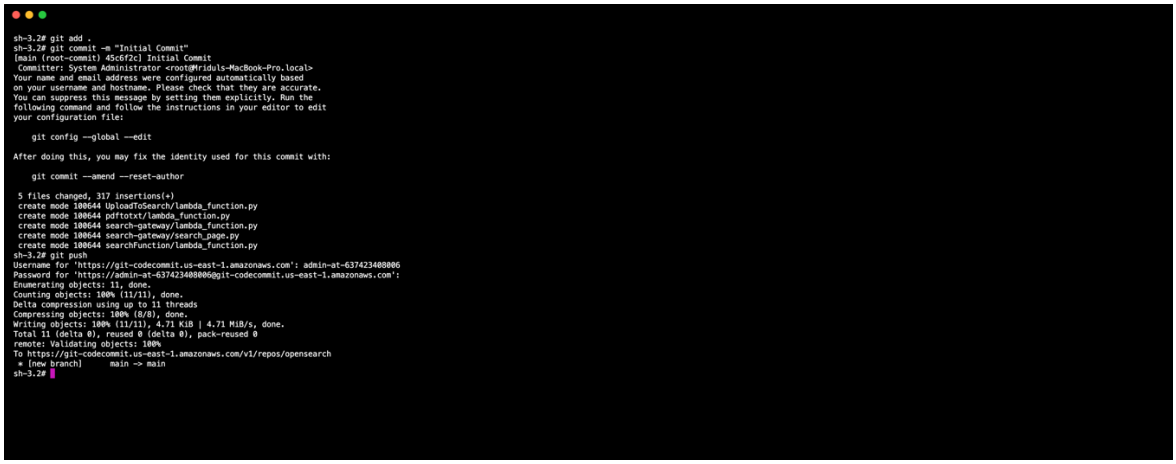
❖ Copy the HTTPS URL of the repository.
❖ In your terminal, use the following command to clone the repository:
❖ bash

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

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

6. Copy Lambda Function Files and Push to Repository:

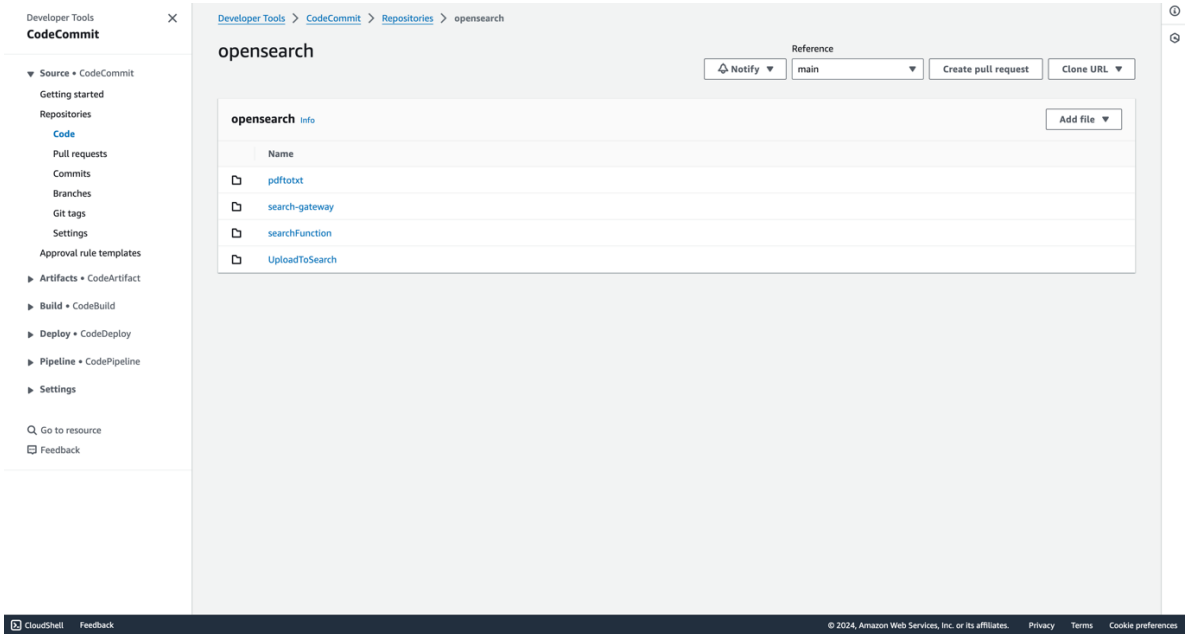
❖ Copy the Lambda function files (including buildspec and SAM files) into the local directory created in Step 4.
❖ Use the following commands to push the files to the remote repository:



❖ bash

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

❖ Enter the CodeCommit credentials if prompted.

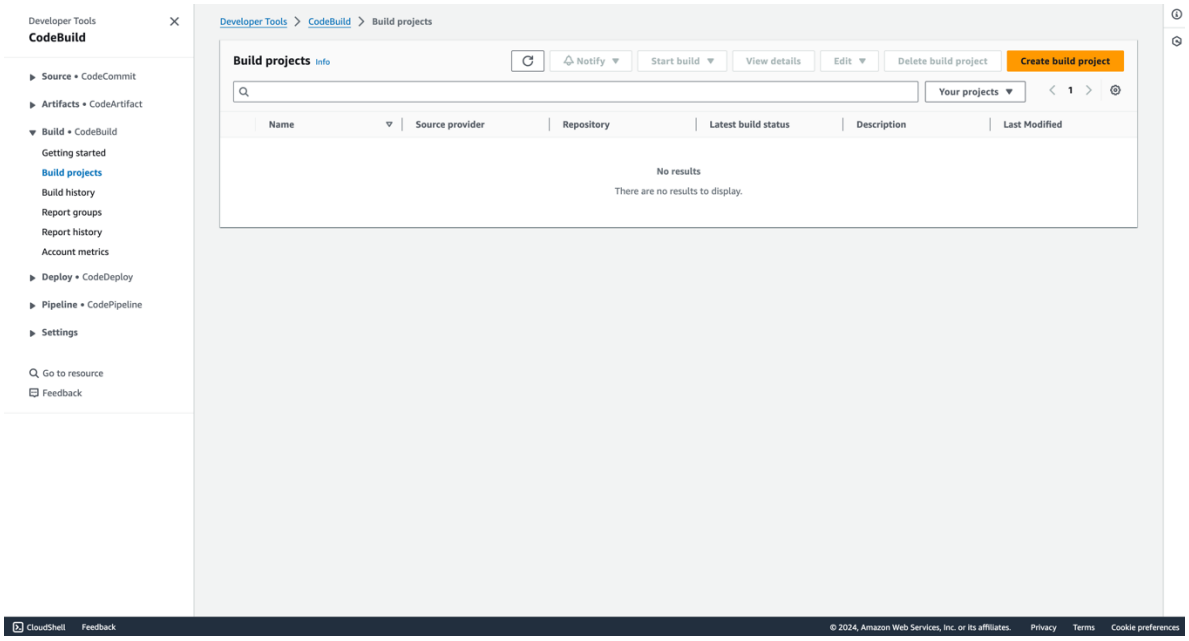


By following these steps, you should have set up a CodeCommit repository, created an IAM user with the necessary permissions, installed Git on your local machine, cloned the remote repository to your local machine, and pushed the Lambda function files into the CodeCommit repository.

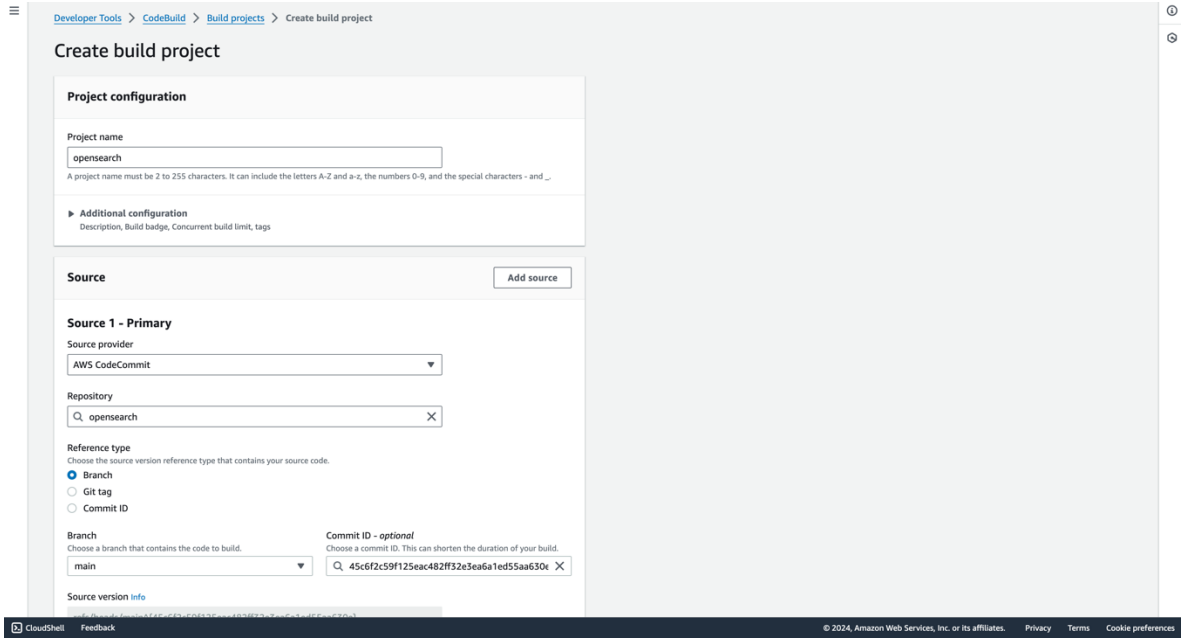
CODE BUILD SETUP

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

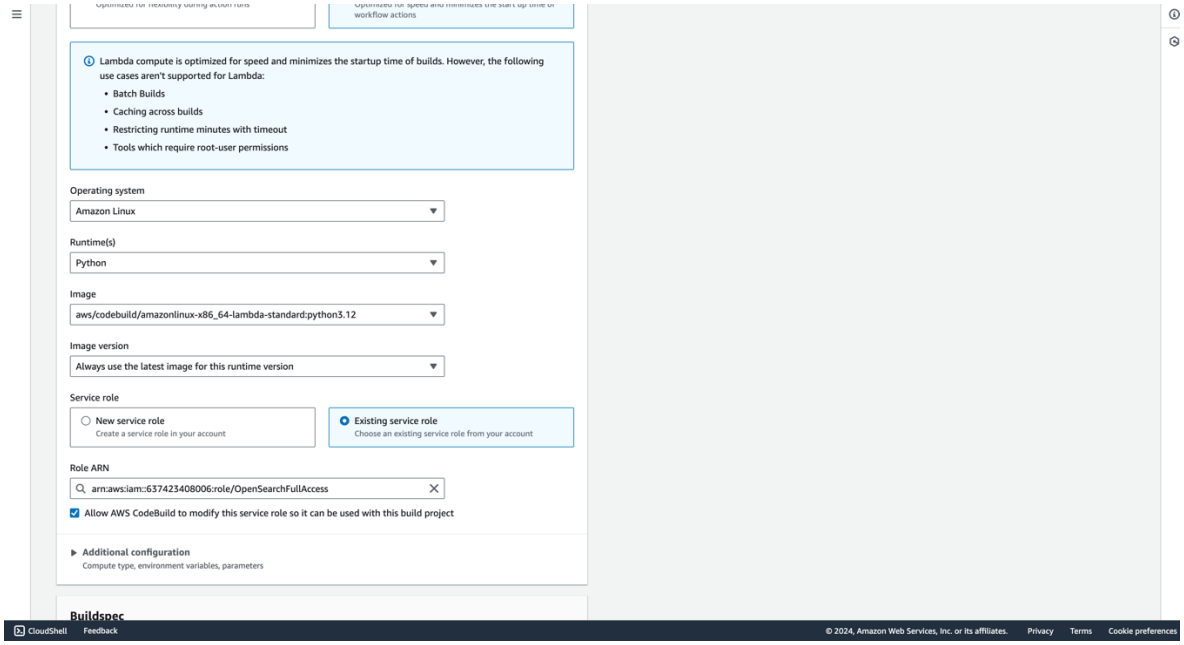
- Log in to the AWS Management Console.
- Navigate to the CodeBuild service.



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



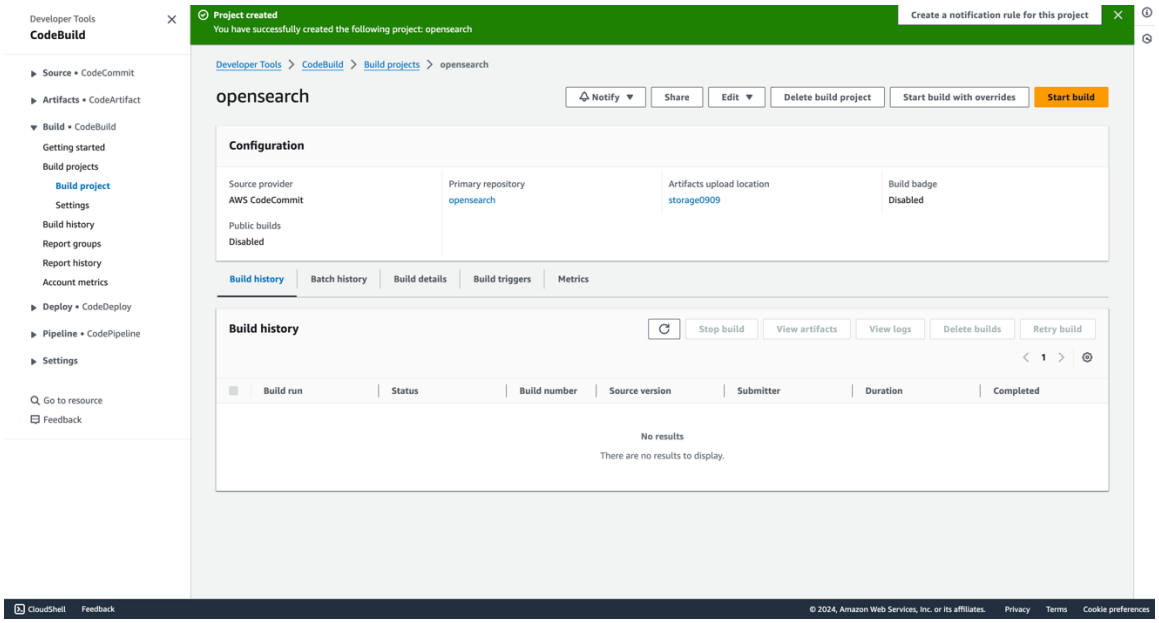
- Configure the source provider as CodeCommit.
- Choose the repository created previously.
- Set the other project settings according to your requirements.



❖ Click on "Create build project."

2. Use the Latest Standard Ubuntu CodeBuild 6.0 Environment:

❖ In the CodeBuild project configuration:



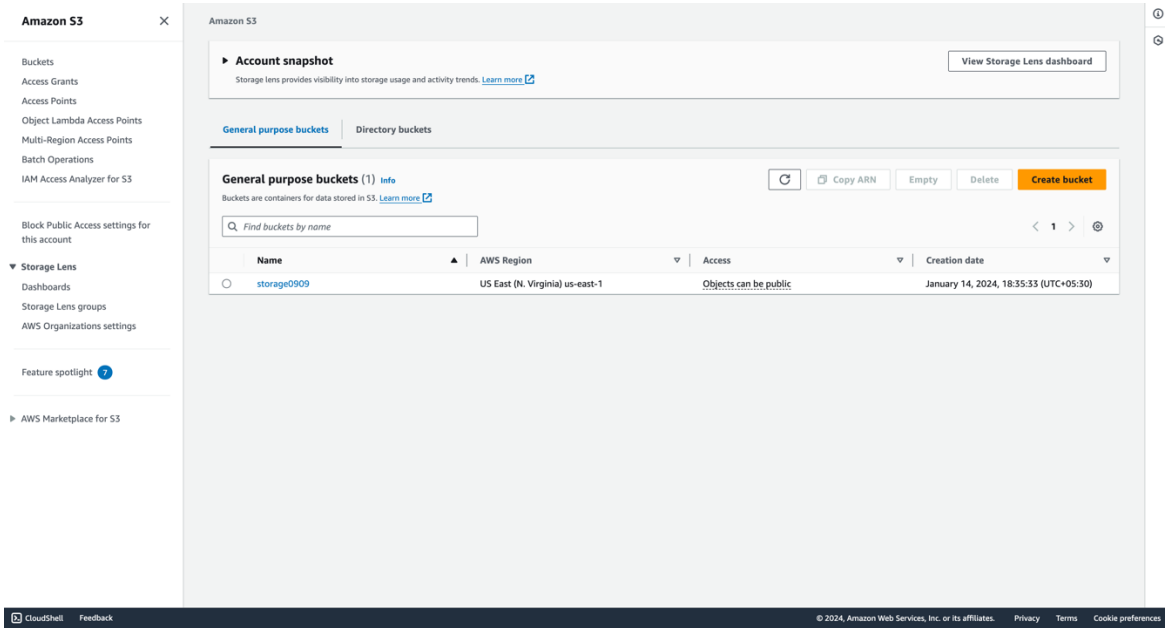
❖ Choose the environment image type as "Standard."
❖ Select "Ubuntu" as the operating system.
❖ Choose the latest CodeBuild 6.0 environment.

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.

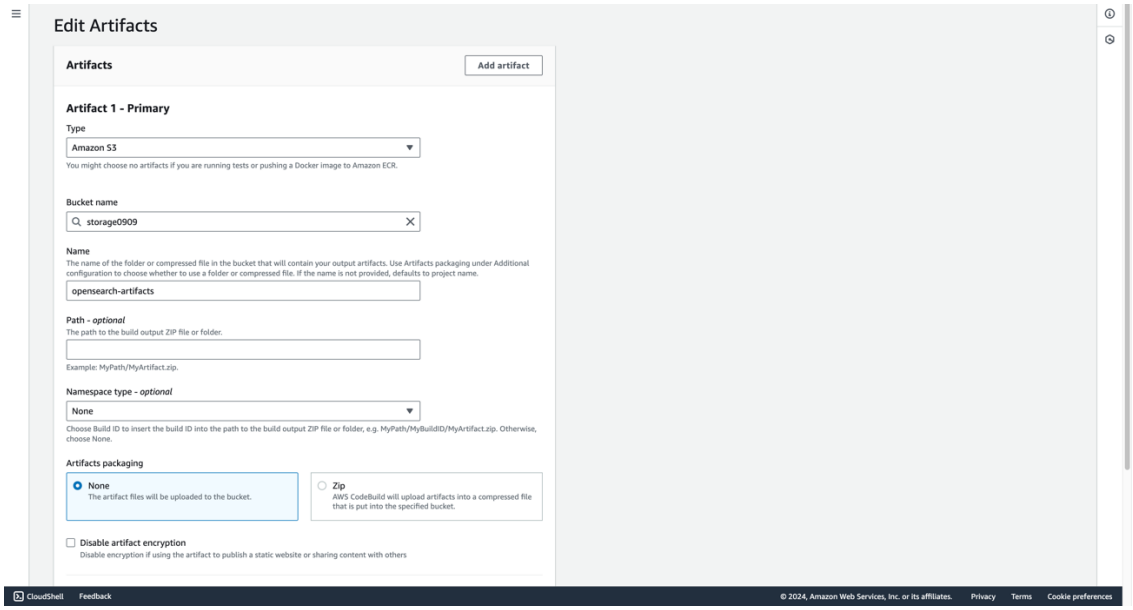
4. Create a New S3 Bucket for Storing Artifacts:



- 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.



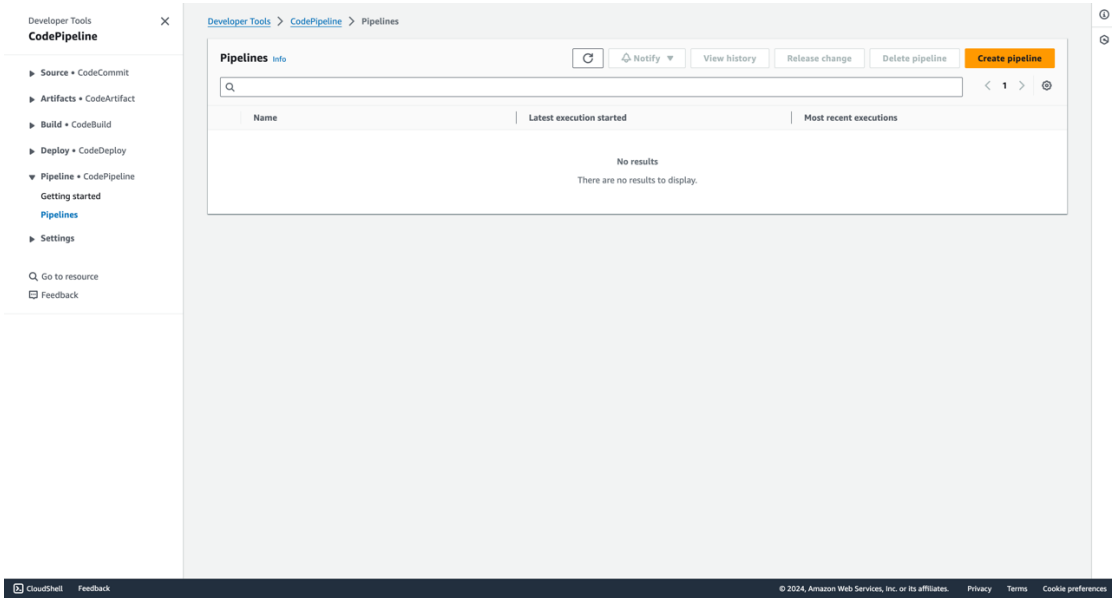
- ❖ Choose "Amazon S3" as the type.
- ❖ Select the region where you created the S3 bucket.
- ❖ Enter the name of the S3 bucket created in Step 4.
- ❖ Set other artifact settings according to your requirements.
- ❖ Click on "Save build project" to save the changes.

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.

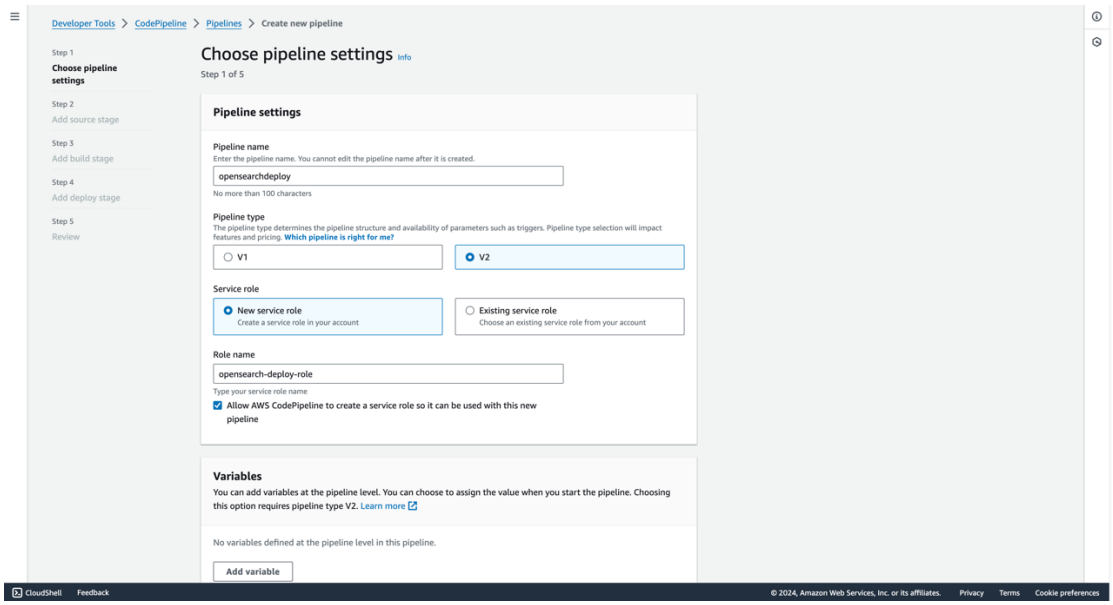
CODE PIPELINE SETUP

1. Create a New CodePipeline Using the AWS Console:

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



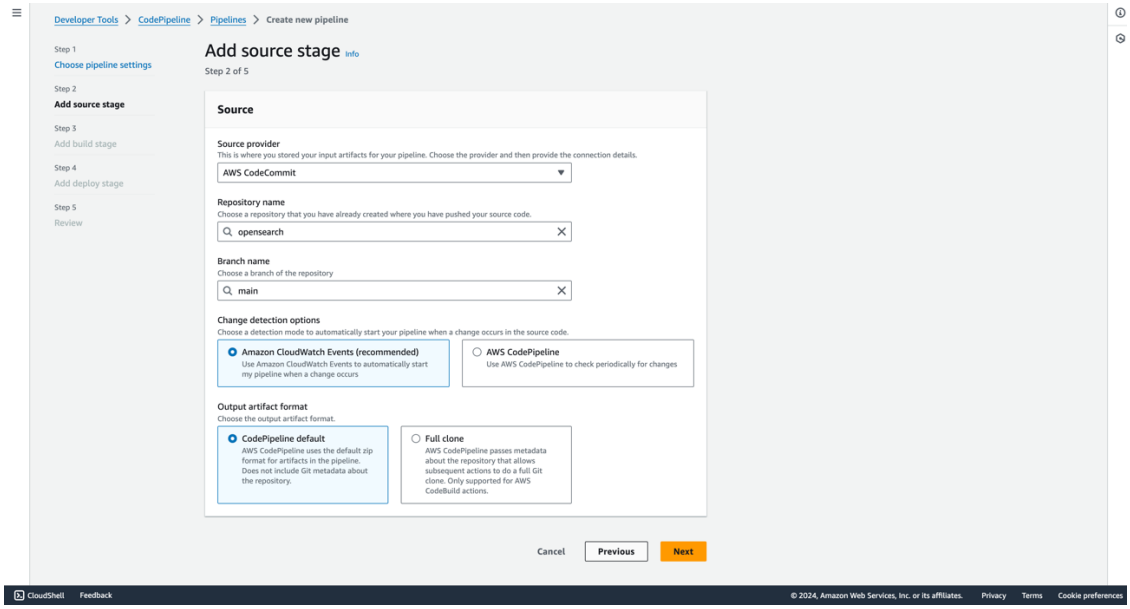
- ❖ Click on "Create pipeline."



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

2. Select CodeCommit Repository as the Source:

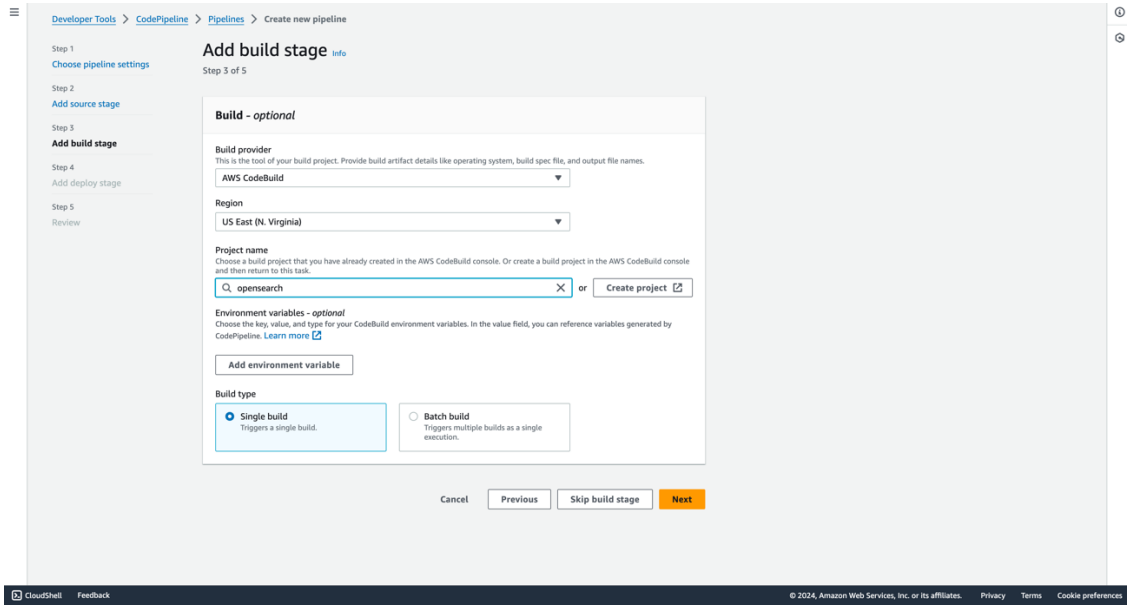
- ❖ In the "Source" configuration:
- ❖ Choose "CodeCommit" as the source provider.



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

3. Select CodeBuild Project for the Build Stage:

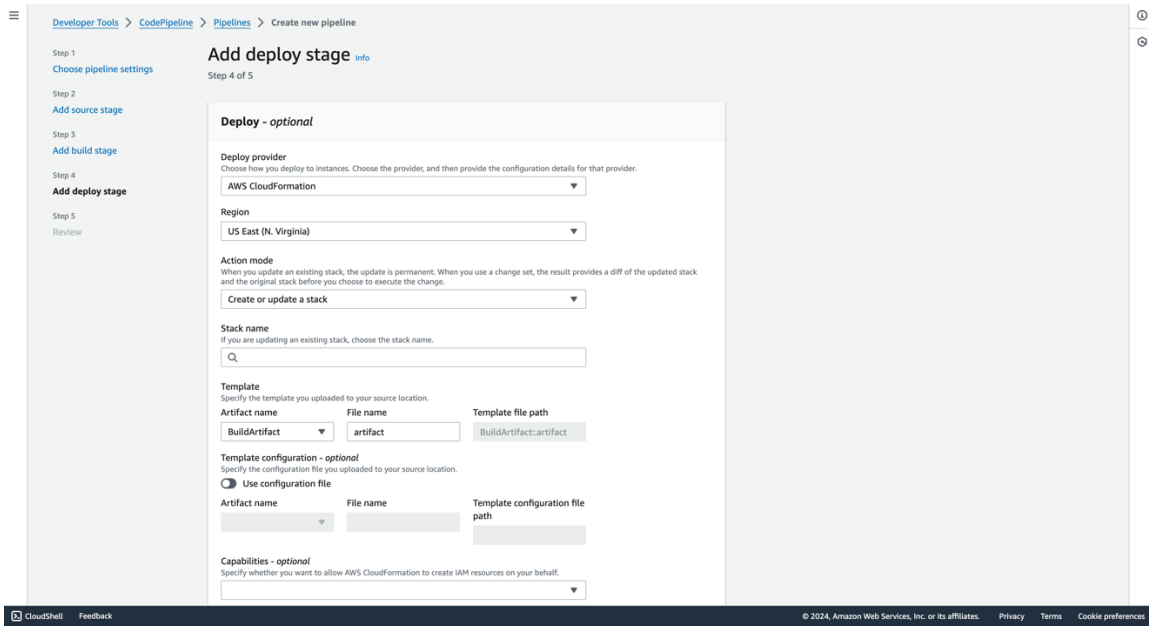
- ❖ In the "Build" configuration:
- ❖ Choose "AWS CodeBuild" as the build provider.



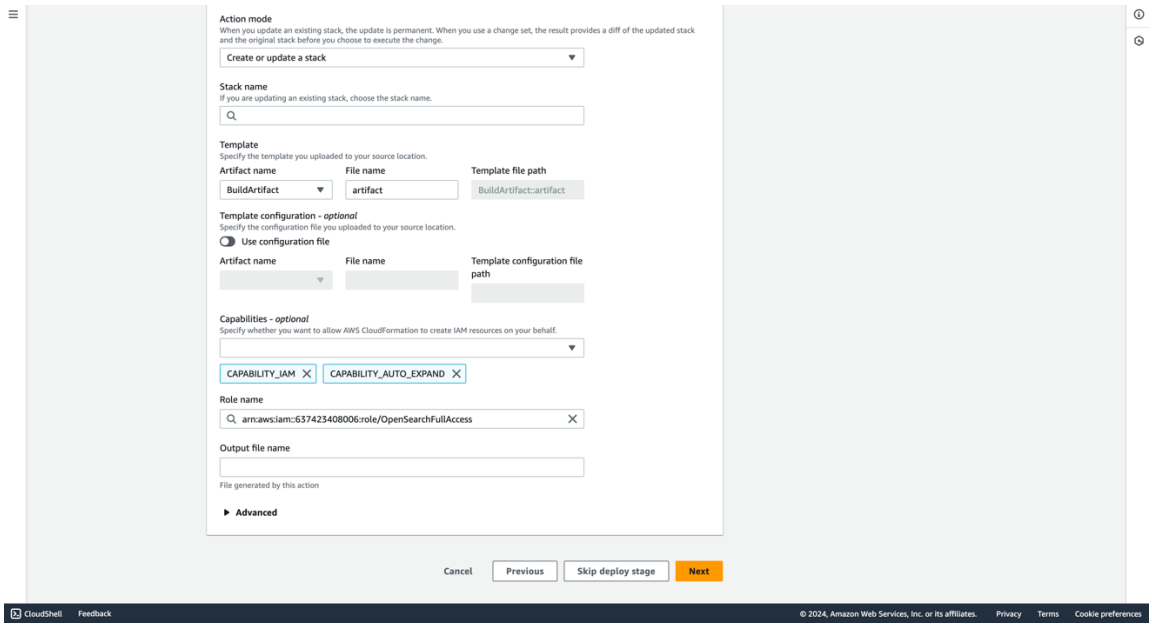
- ❖ Select the CodeBuild project created previously.
- ❖ Configure other build settings as needed.
- ❖ Click on "Next."

4. Use CloudFormation for Deployment:

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



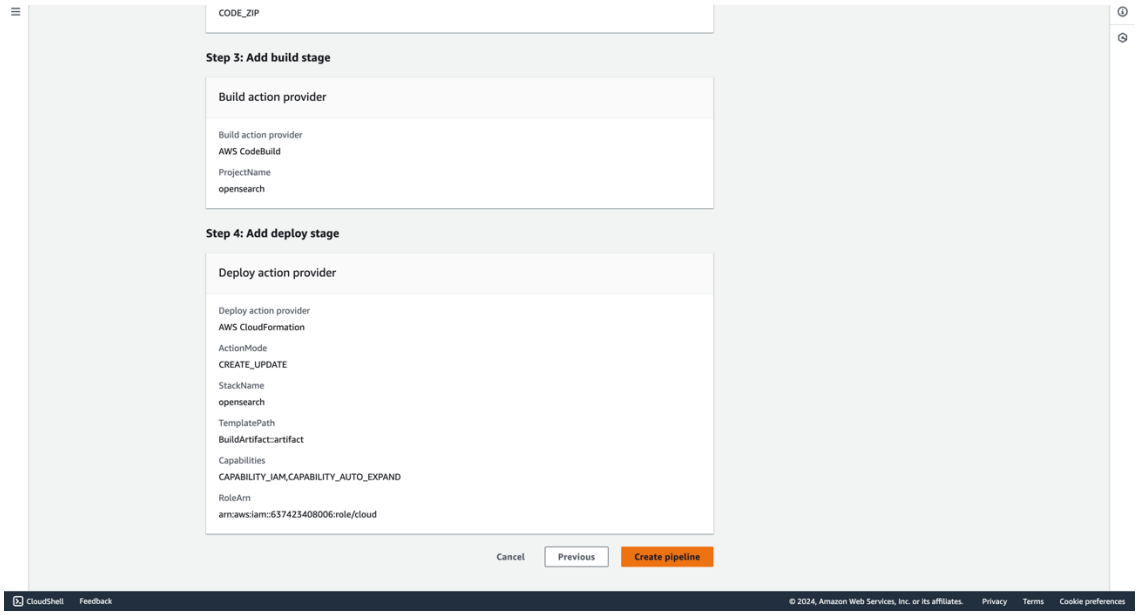
- ❖ Create a new CloudFormation service role if you don't have one.
- ❖ Ensure the IAM role used for CloudFormation has access to S3 and AWS Lambda.



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

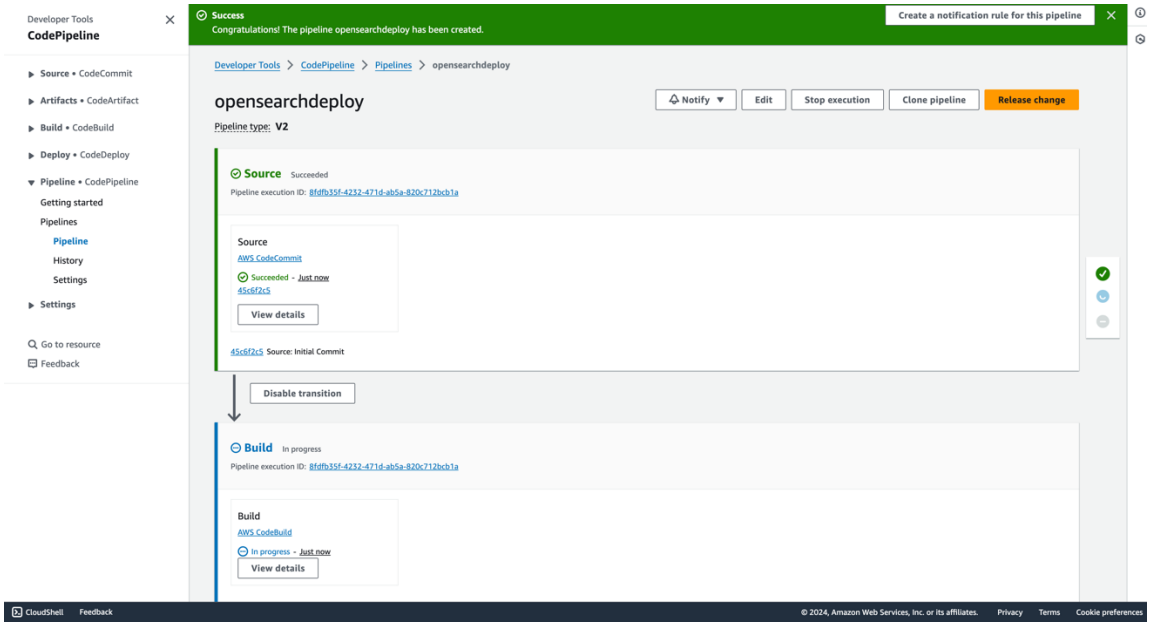
CAPABILITY_IAM
CAPABILITY_AUTO_EXPAND

- ❖ 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:

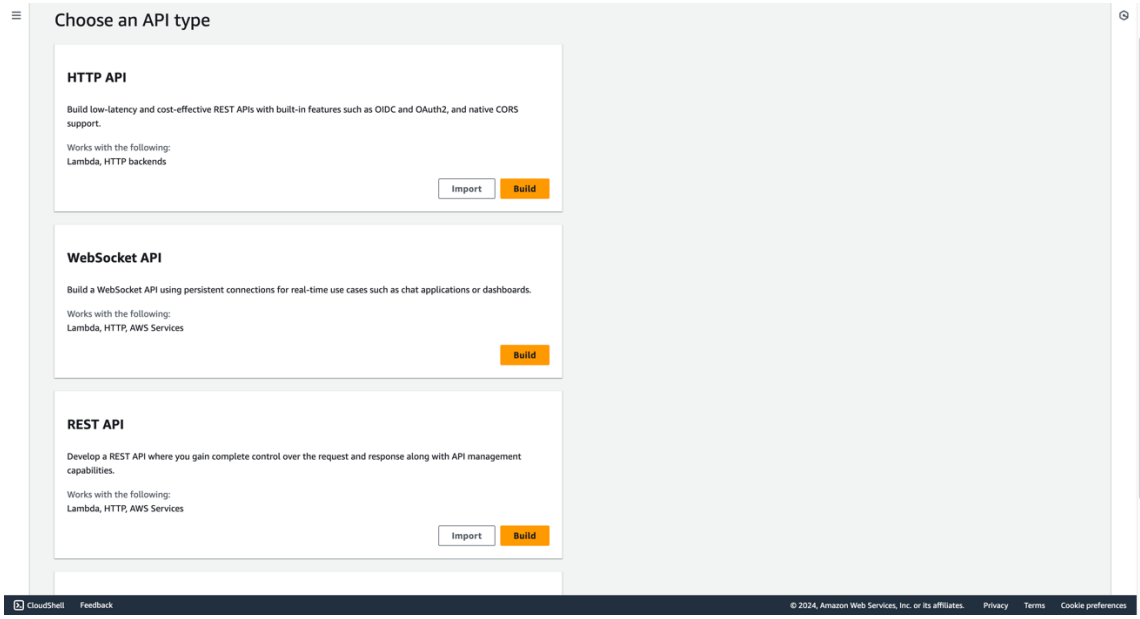
- Repeat the steps above for each Lambda function:
- Create a new CodeCommit repository for each Lambda function.
- Create a new CodeBuild project for each Lambda function.
- Create a new CodePipeline for each Lambda function, selecting the corresponding CodeCommit repository and CodeBuild project.

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:

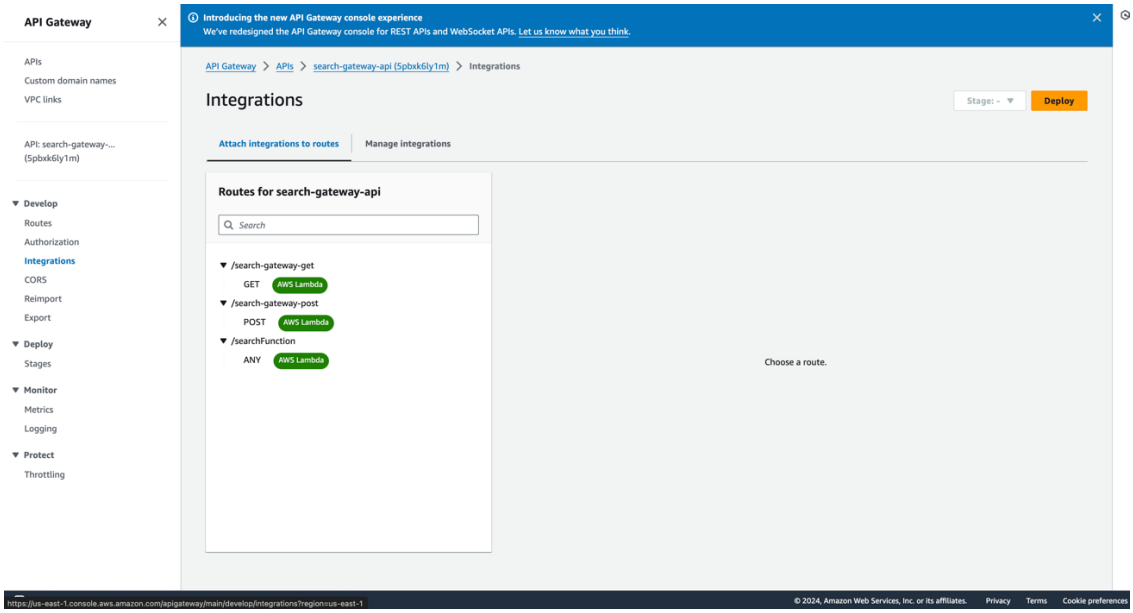
- Log in to the AWS Management Console.
- Navigate to the API Gateway service.
- Click on "Create API."
- Choose "HTTP API."



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

2. Add Lambda Functions to Integrations:

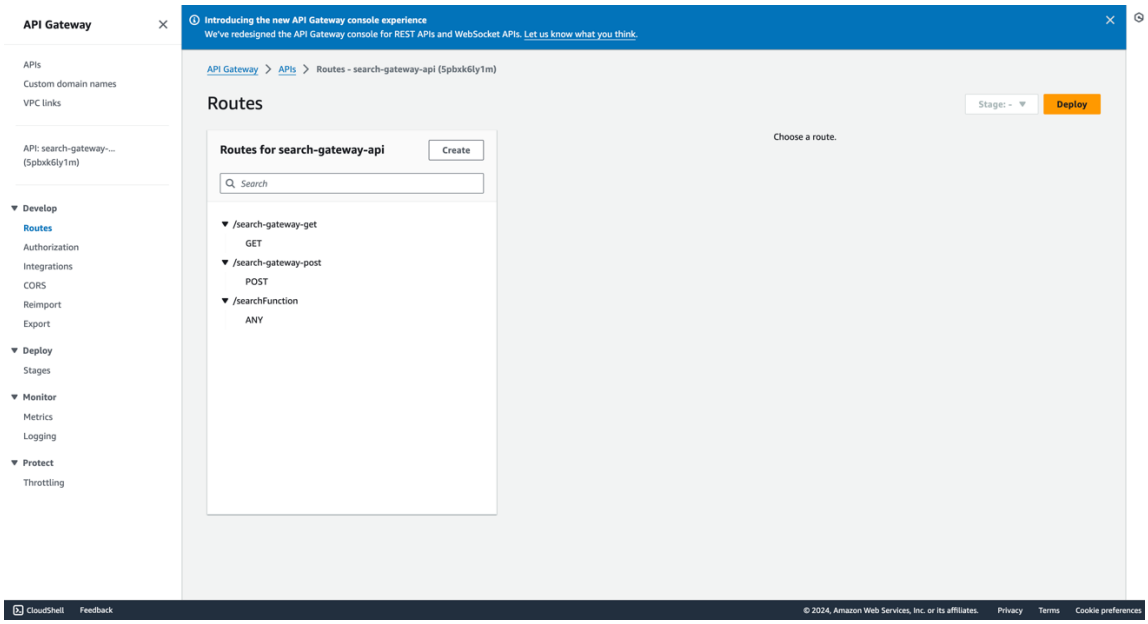
- In your API Gateway configuration:
- Navigate to the "Integrations" section.



- 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.

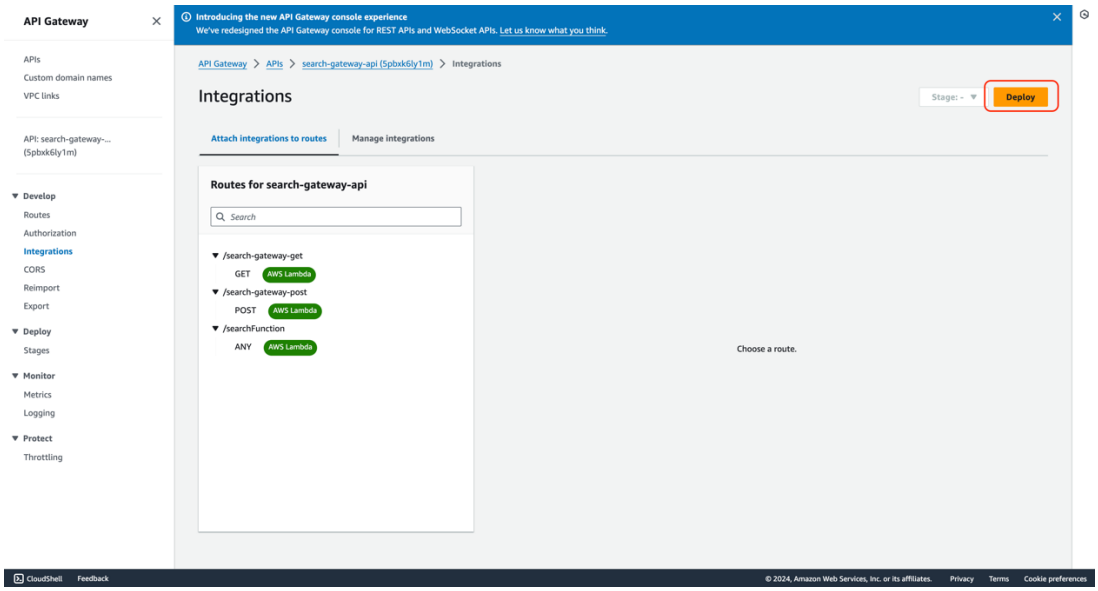


- Add the following routes:

- 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."



- 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.

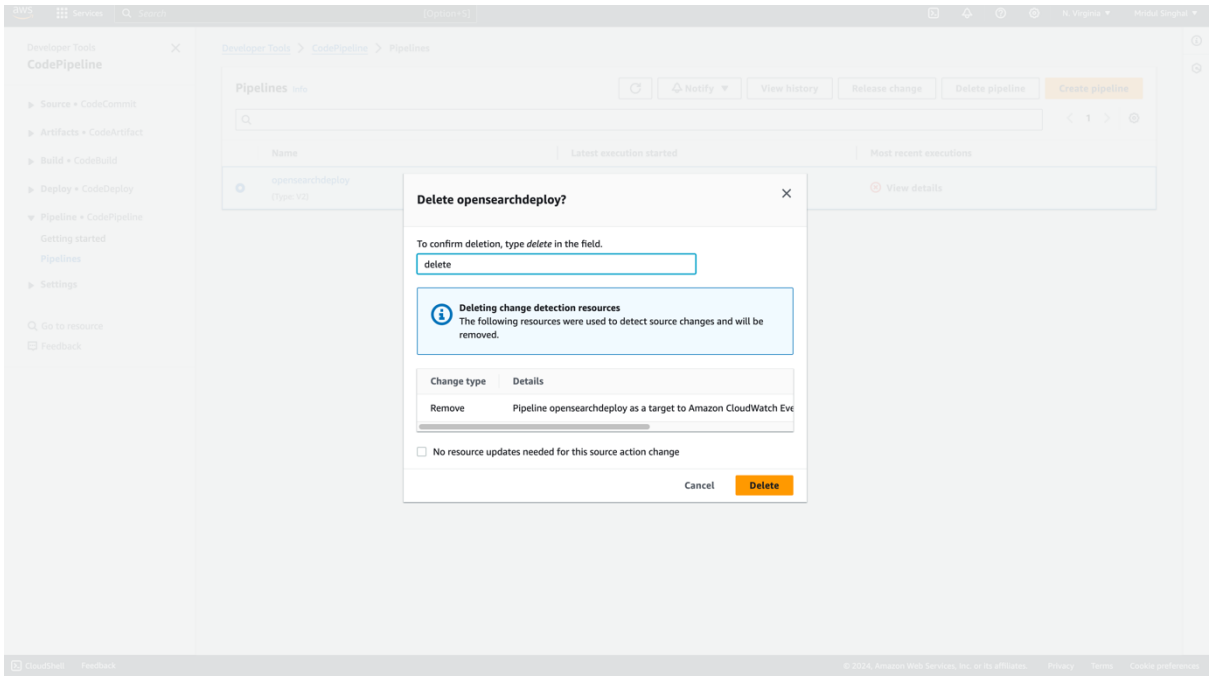
4. Search for a Word in the Sample PDF Document:

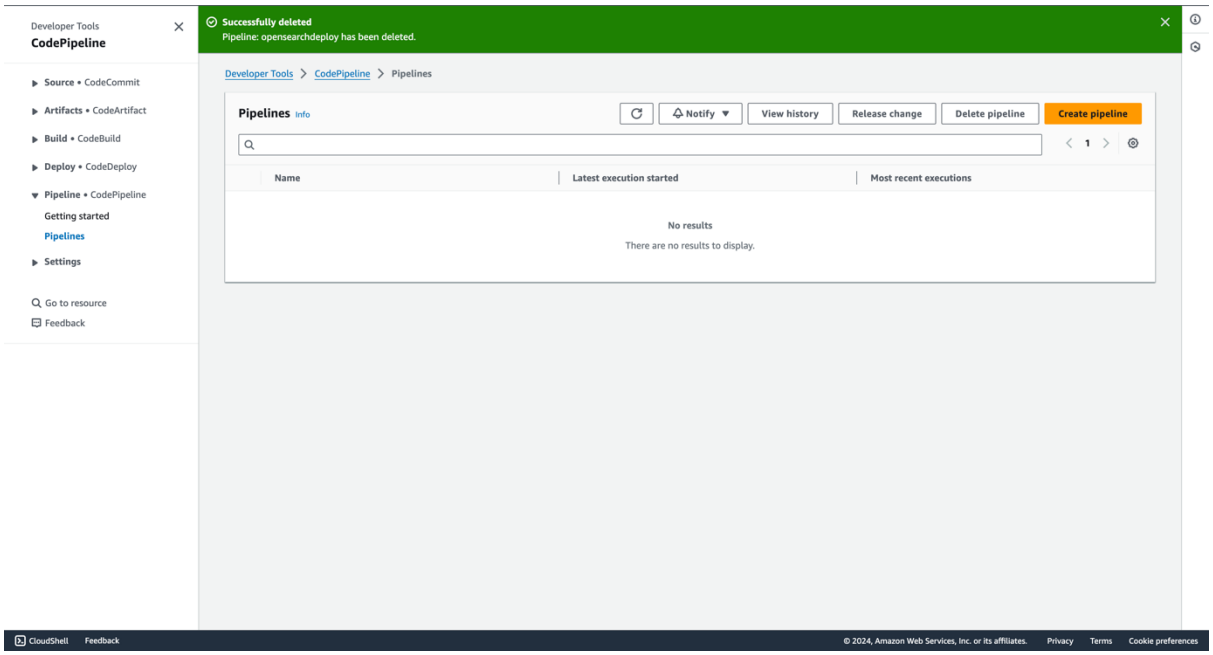
- On the search webpage, enter a word that is present in the sample PDF document.
- Submit the search query.
- Check the output to verify that the search results are correct.

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

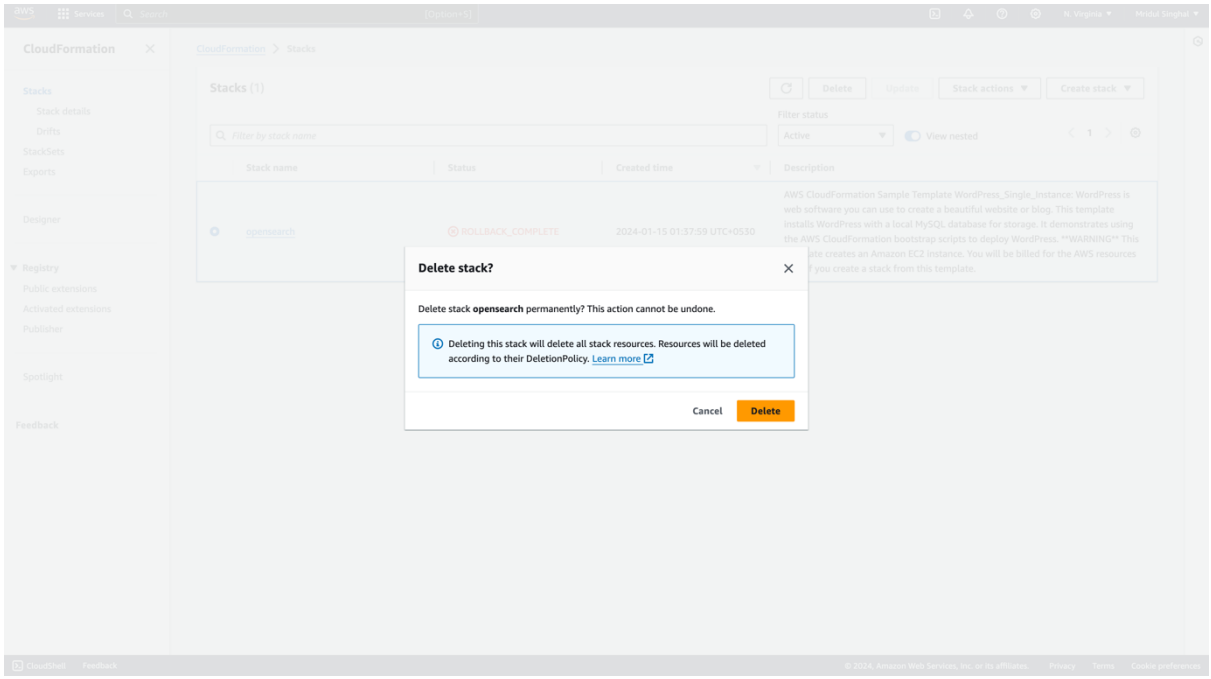
The test PDF file should trigger the Lambda functions in succession as expected.
Monitoring the CloudWatch logs will help identify any issues during the execution of each function.
Accessing the search webpage and searching for a word should return correct results if the integration between the API Gateway, Lambda functions, and OpenSearch is set up correctly.

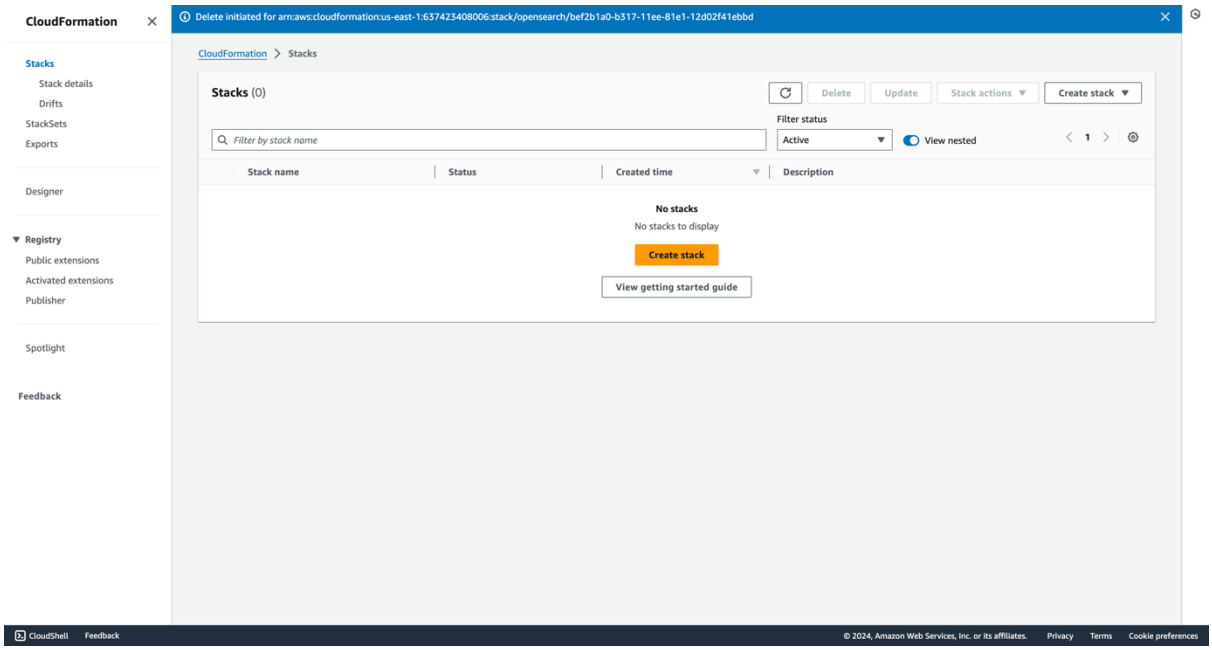
CLEANUP



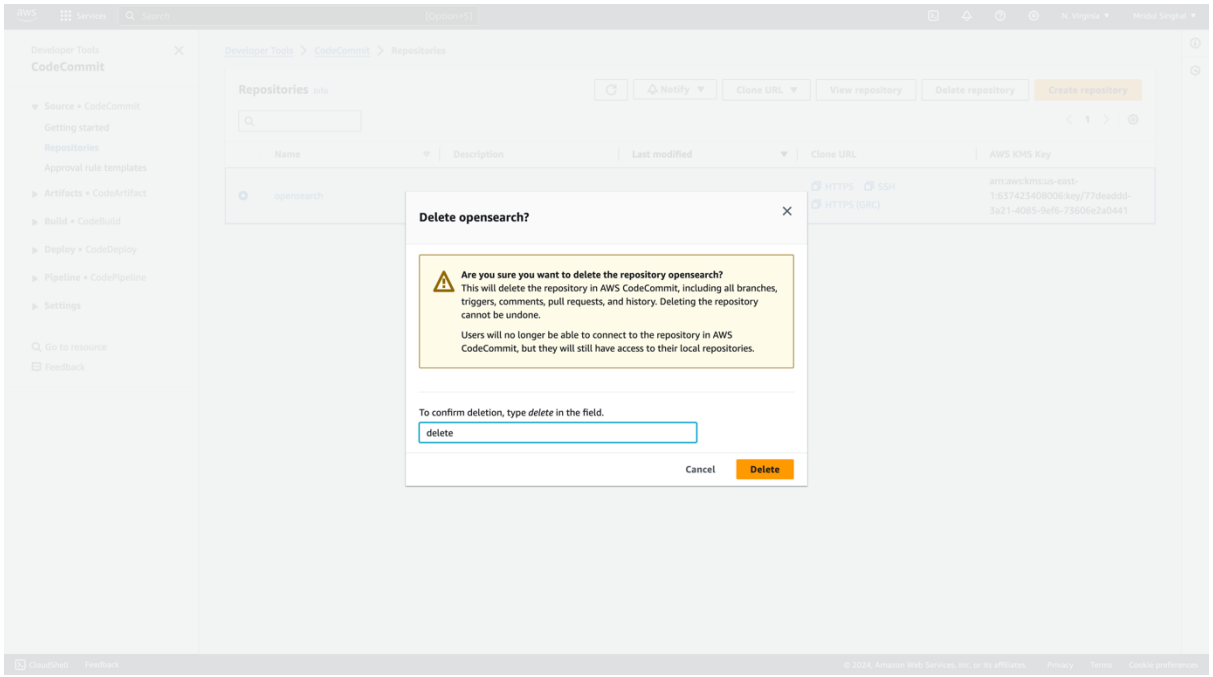


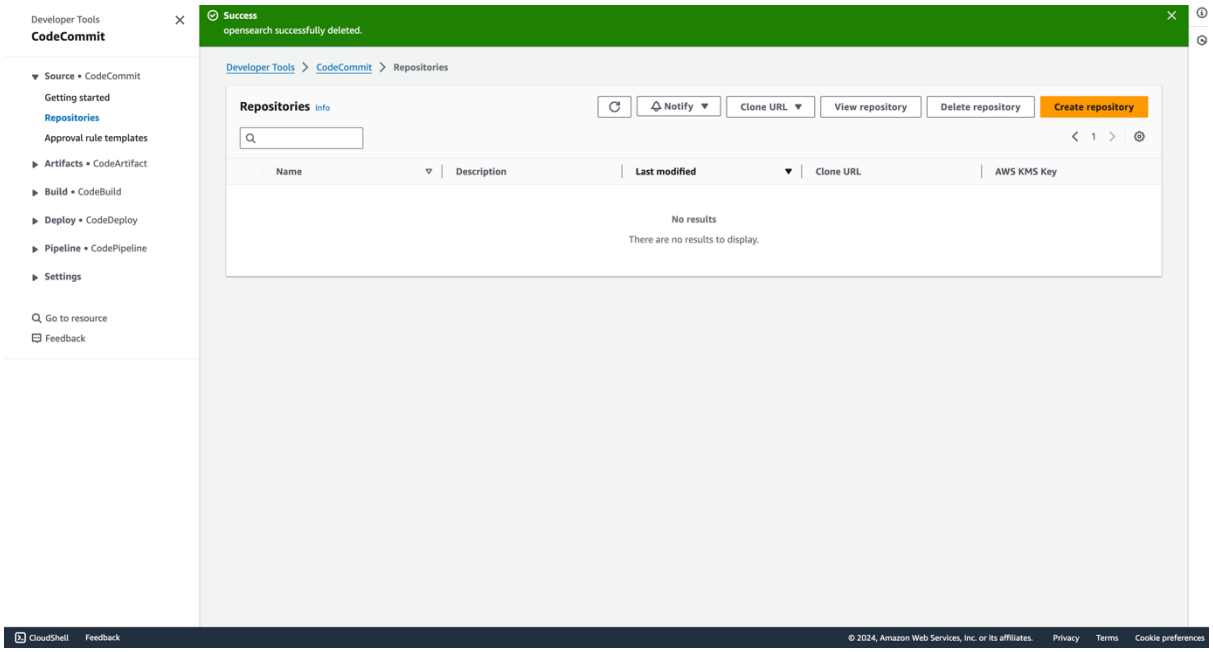
❖ Pipeline deleted



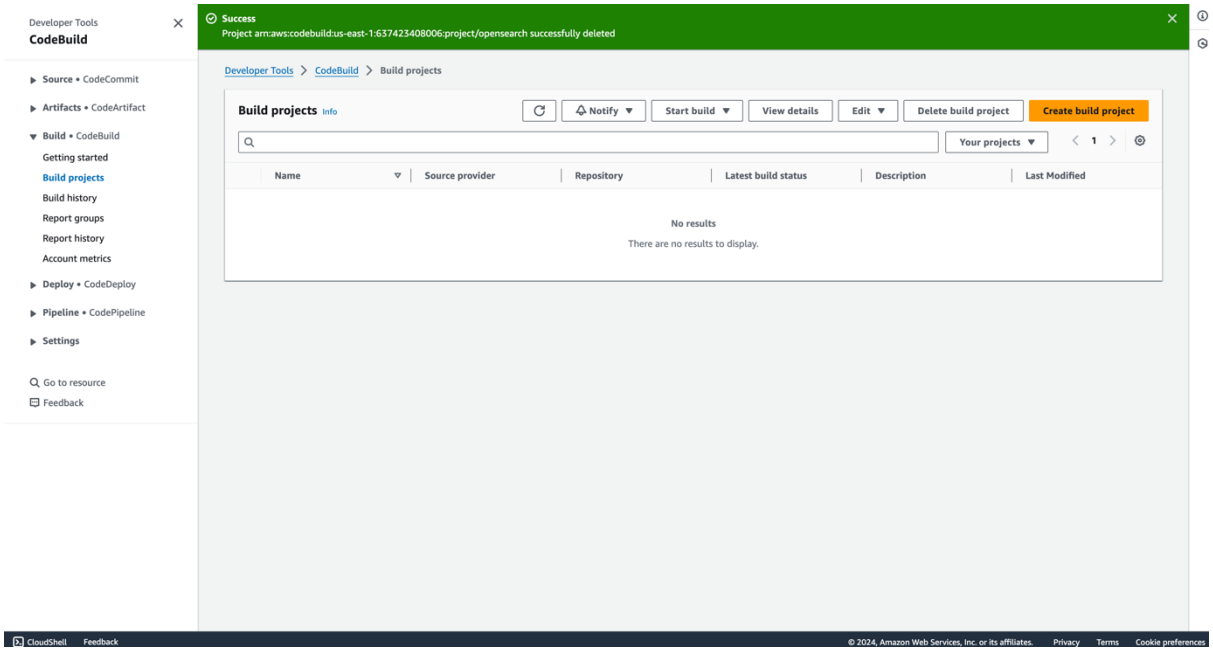


❖ Deleting the stack

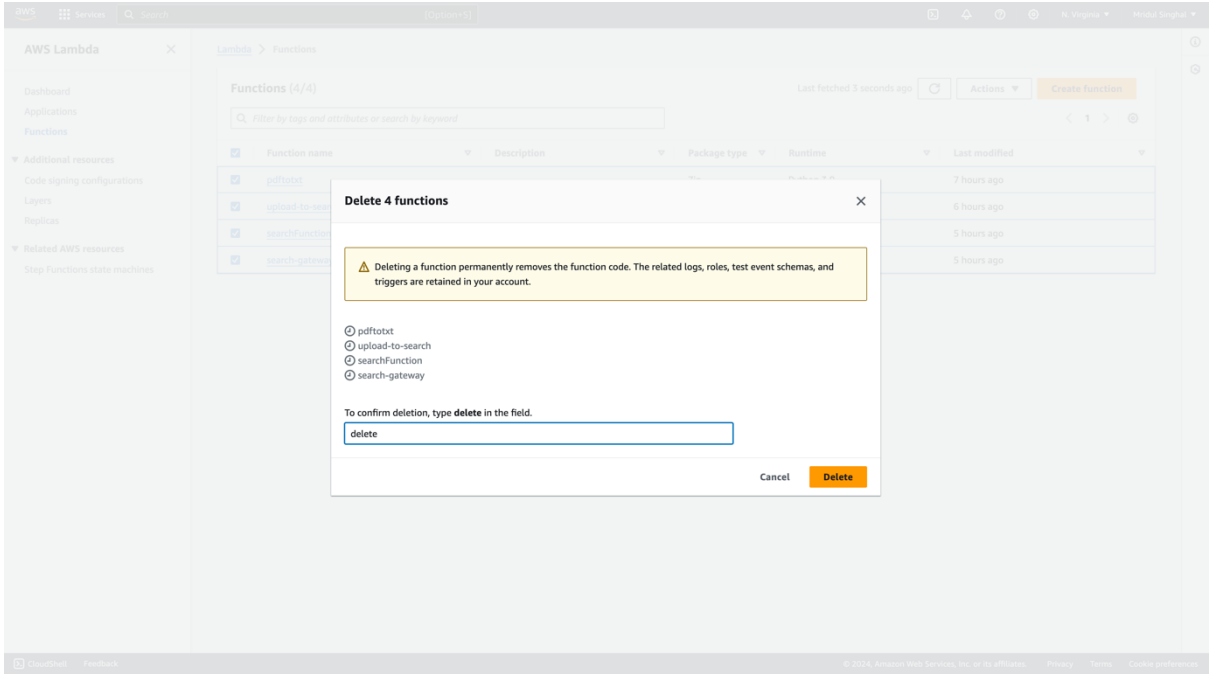


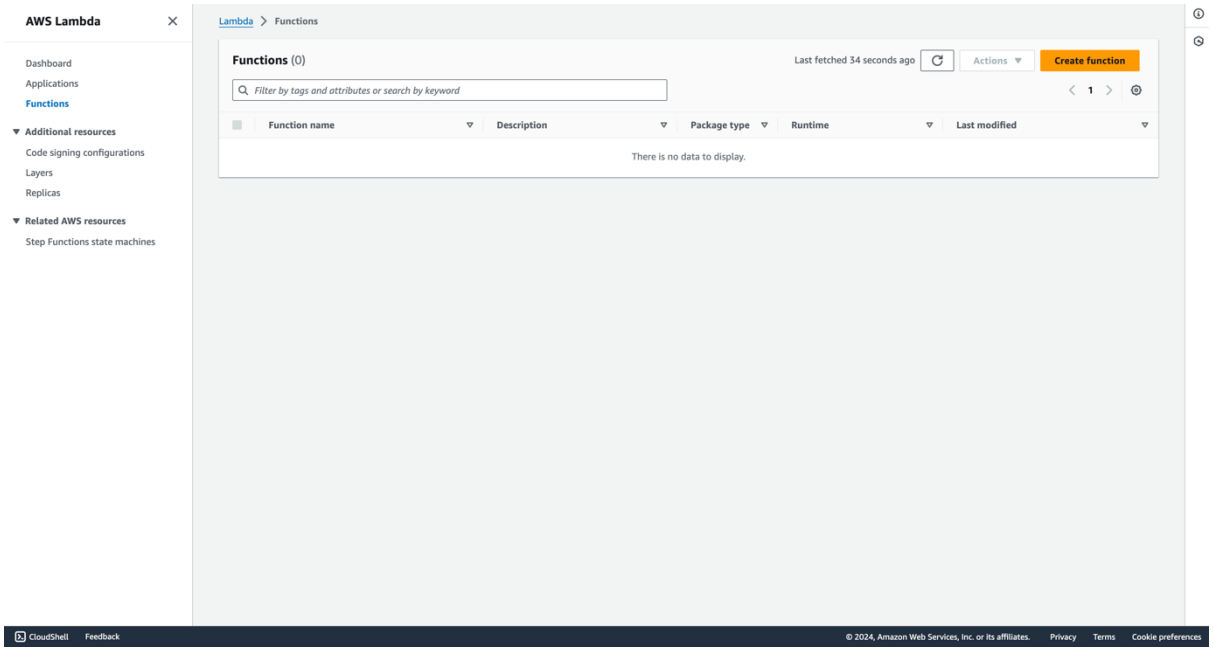


❖ Deleting the code commit repository.

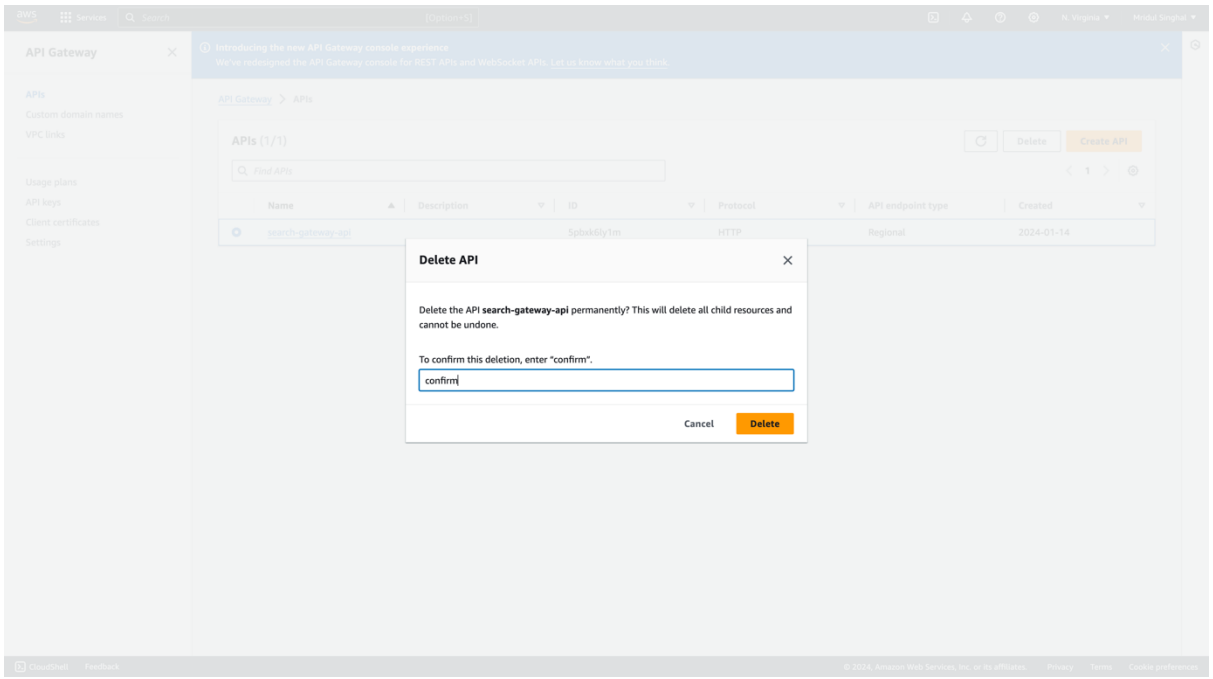


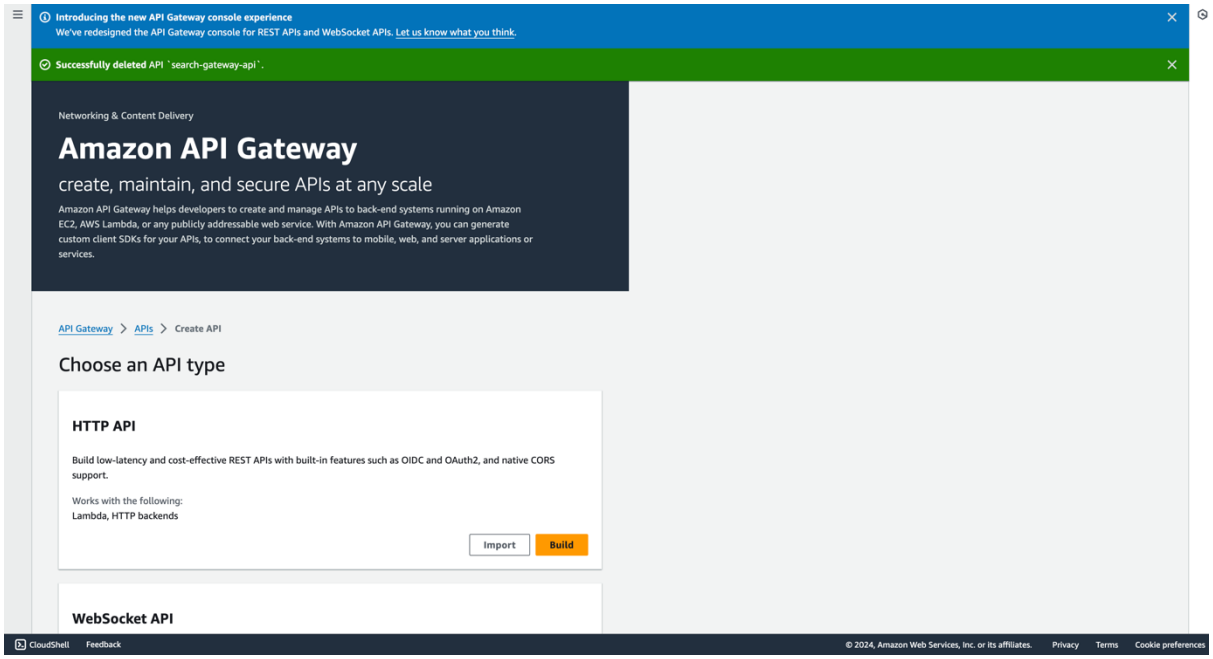
❖ Deleted the CodeBuild Project.



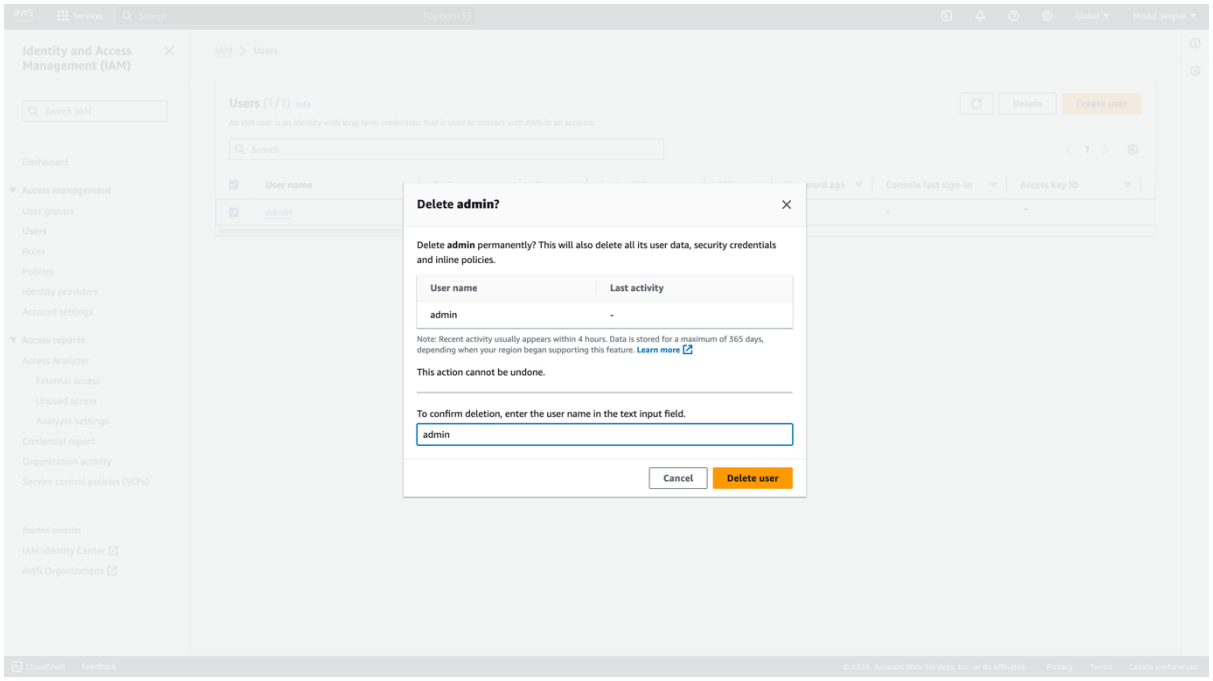


❖ Deleted all the four lambda functions.

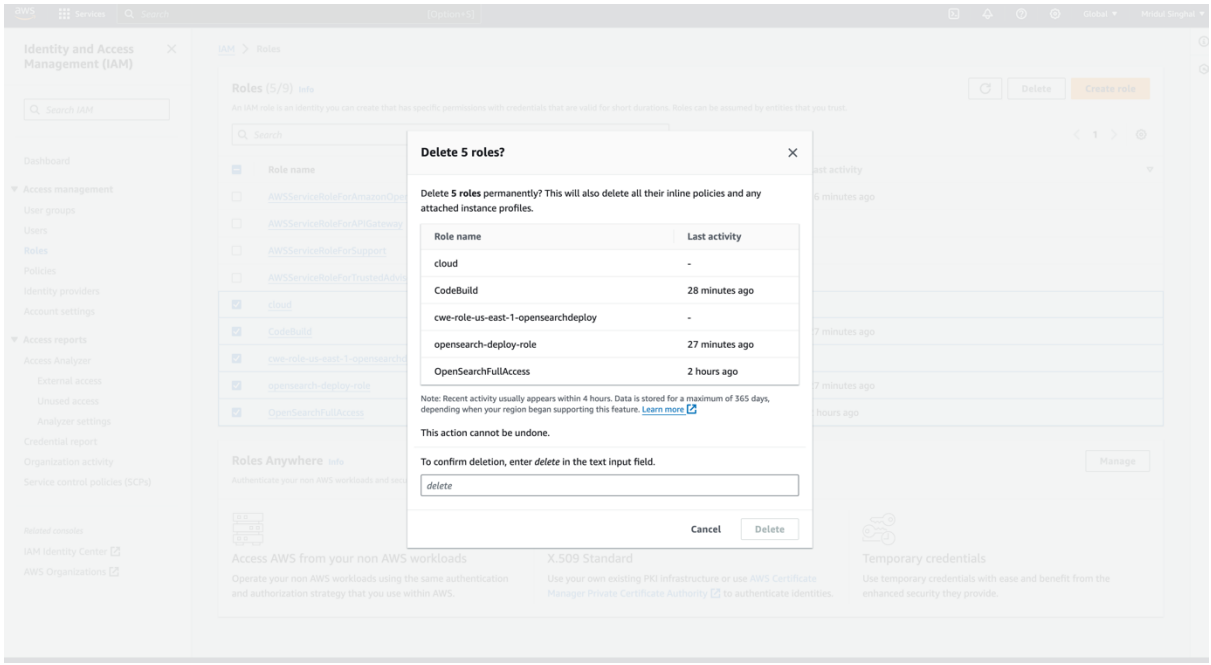




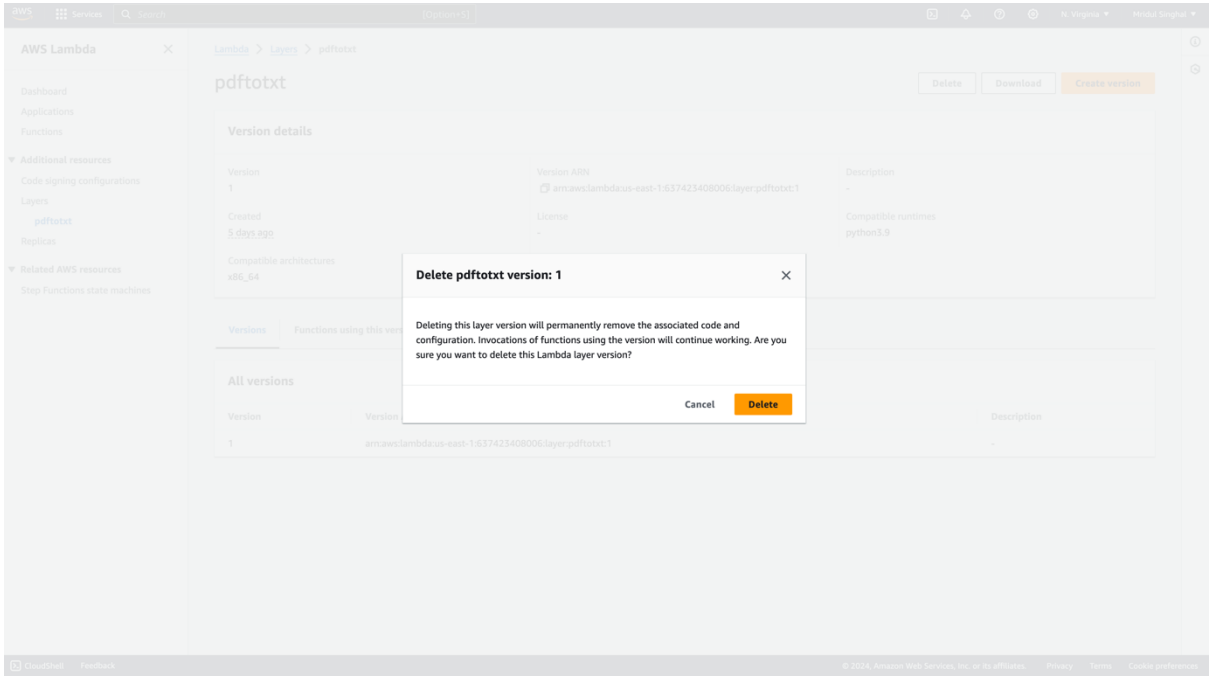
❖ Deleted the API Gateway



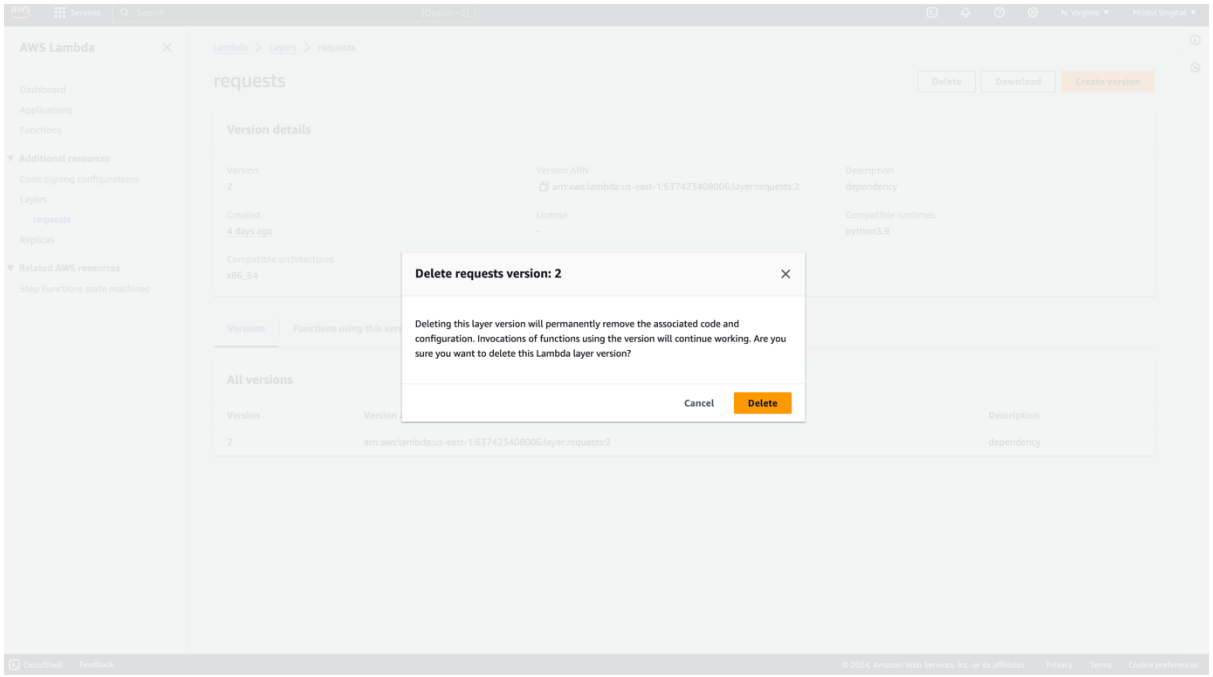
❖ Deleted the IAM User



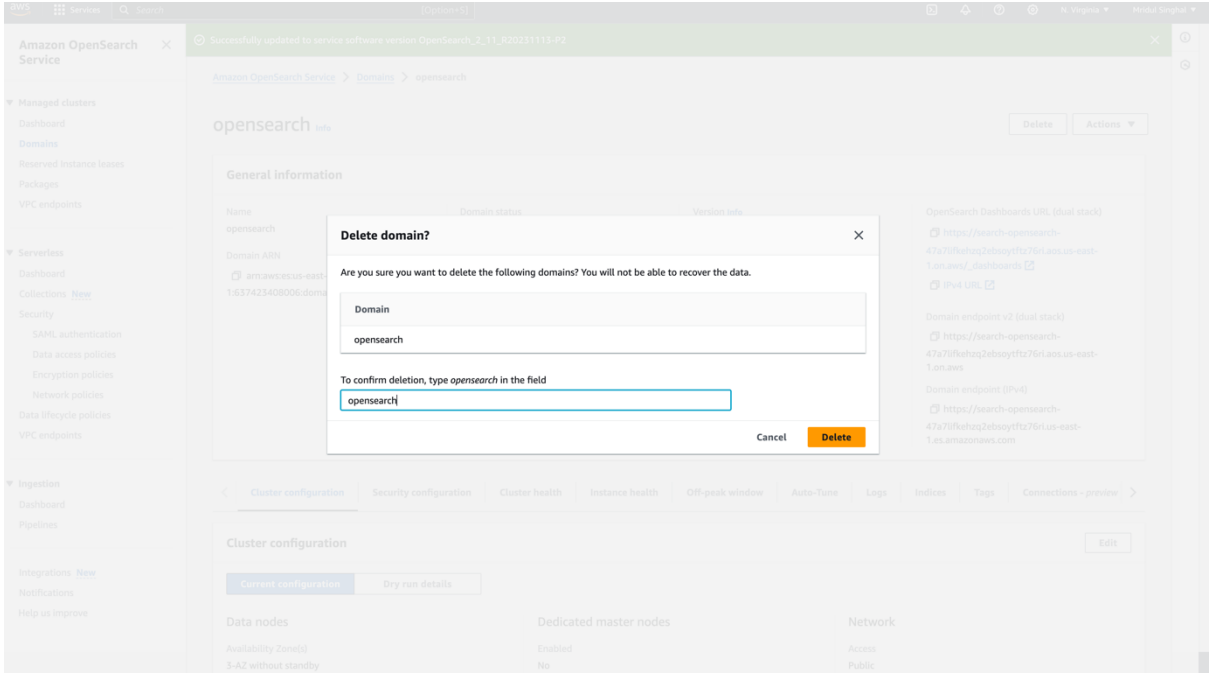
❖ Deleted the IAM Roles.



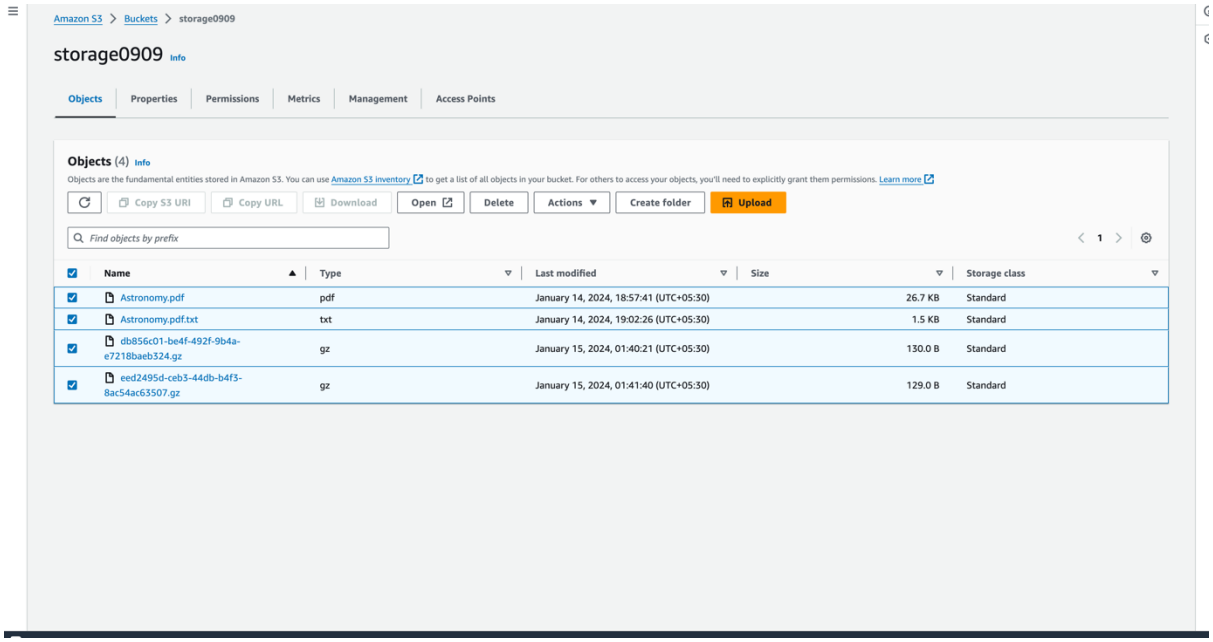
❖ Deleted the Lambda Layer pdftotxt “pypdf” dependency.

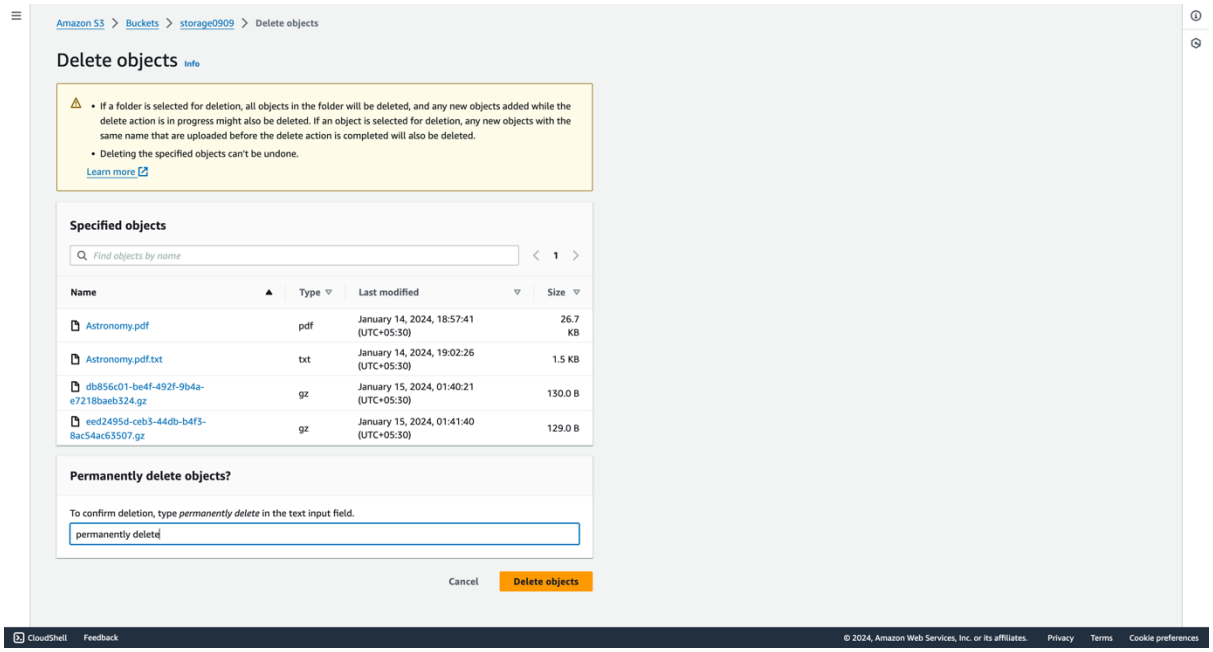


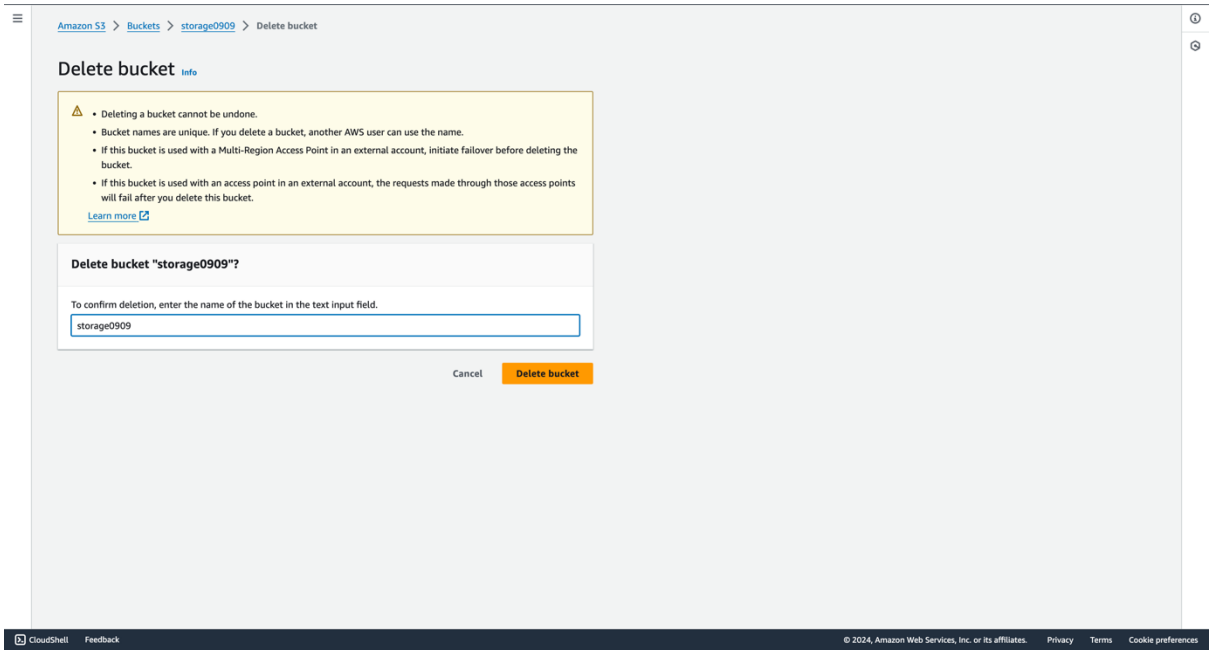
❖ Deleted the layer for requests dependency.



❖ Deleted the OpenSearch.


Amazon S3 > Buckets > storage0909
storage0909 Info
Objects
Properties
Permissions
Metrics
Management
Access Points
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 S3 URI
Copy URL
Download
Open
Delete
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
db856c01-be4f-492f-9b4a-e7218baeb324.gz
gz
January 15, 2024, 01:40:21 (UTC+05:30)
130.0 B
Standard
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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



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