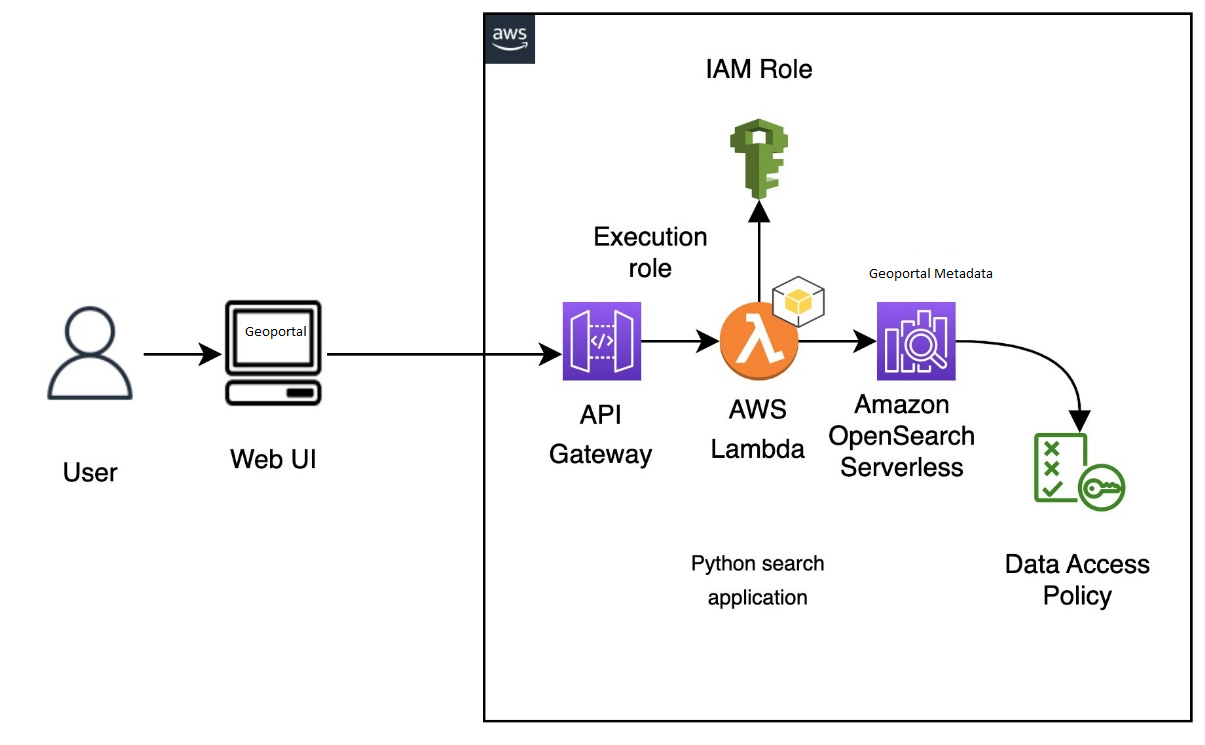
**Configure AWS API Gateway - Lambda - Opensearch for Geoportal Catalog UI**



# 1. Create and deploy the Lambda function

To create your function using the Lambda console

1. Navigate to the Lambda console at <https://console.aws.amazon.com/lambda/home>. On the left navigation pane, choose **Functions**.
2. Select **Create function**.
3. Configure the following fields:
   * Function name: opensearch-function
   * Runtime: Python 3.9
   * Architecture: x86\_64

Keep all other default options and choose **Create function**.

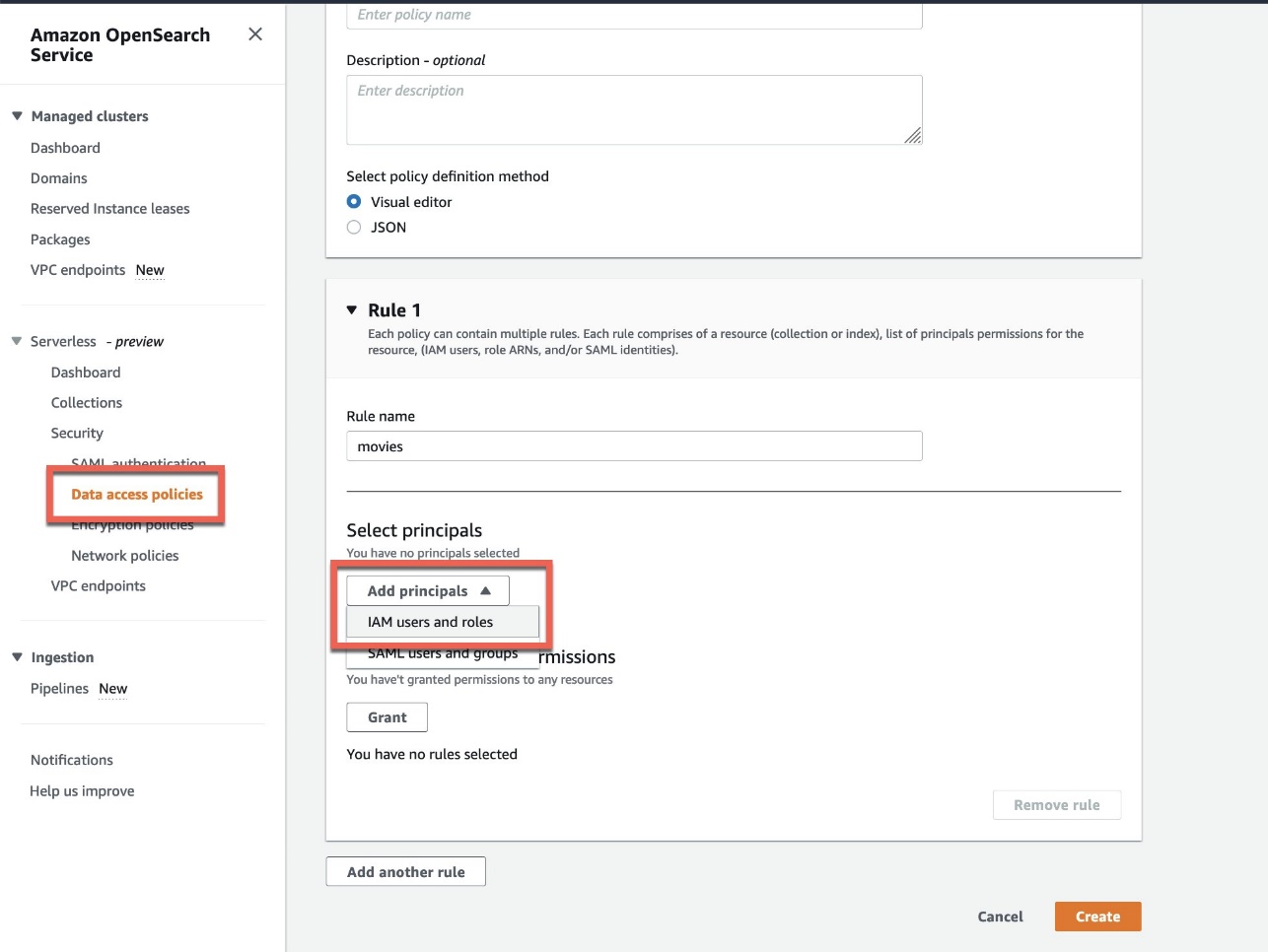
1. In the **Code source** section of the function summary page, choose the **Upload from** dropdown and select **.zip file**. And select geoportal/doc/aws-serverless/deployment-package.zip and **Save**.
2. The *handler* is the method in your function code that processes events. Under **Runtime settings**, choose **Edit** and change the handler name according to the name of the file in your deployment package where the Lambda function is located. Since your file is named opensearch-lambda.py, rename the handler to *opensearch-lambda*.lambda\_handler. For more information, see [Lambda function handler in Python](https://docs.aws.amazon.com/lambda/latest/dg/python-handler.html).

Next, you need to configure the permissions in OpenSearch Serverless’s data access policy to let the Lambda function access the collection.

1. On the Lambda console, navigate to your function.
2. On the Configuration tab, in the Permissions section, under Execution role, copy the value for Role name.  
   A screenshot of a computer

   Description automatically generated
3. Add this role name as one of the principals of your movie-search collection’s data access policy.

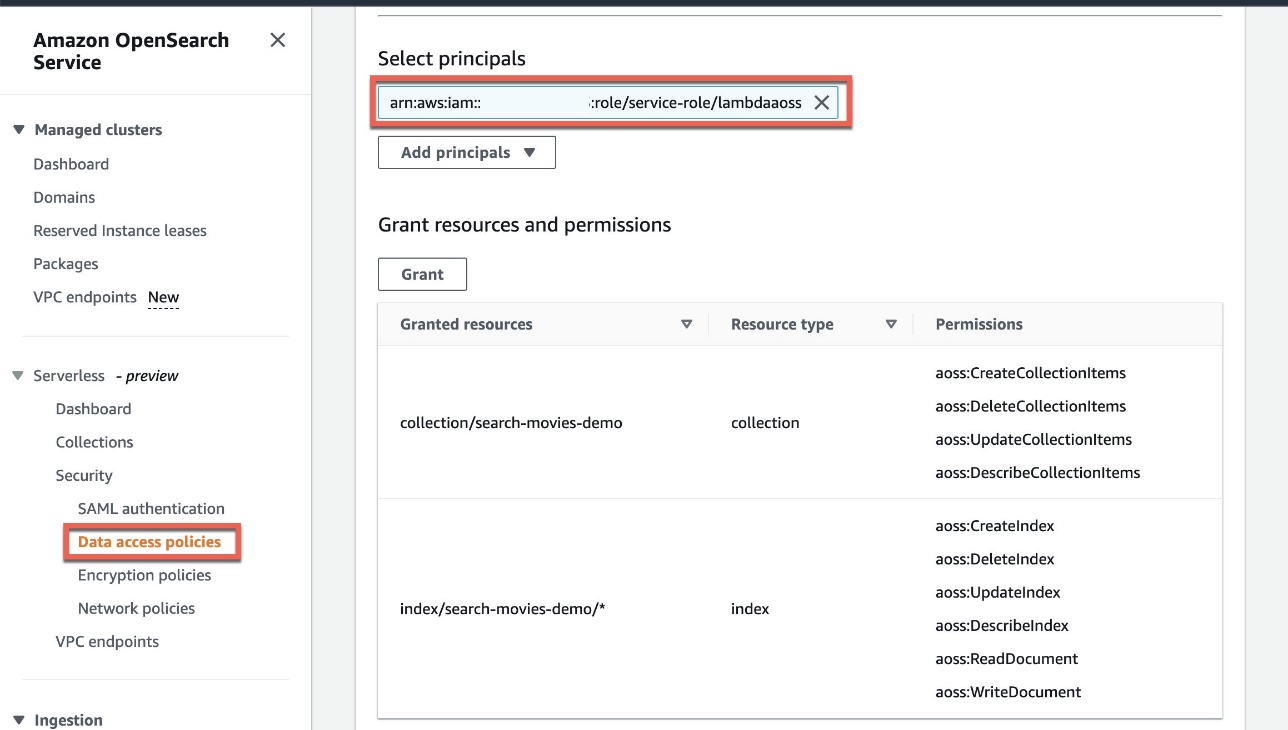
Principals can be [AWS Identity and Access Management](http://aws.amazon.com/iam) (IAM) users, role ARNs, or SAML identities. These principals must be within the current AWS account.



After you add the role name as a principal, you can see the role ARN updated in your rule, as show in the following screenshot.

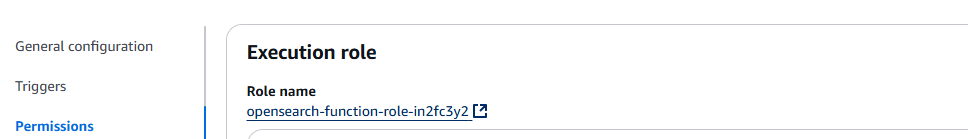
Now you can grant collection and index permissions to this principal.

For more details about data access policies, refer to [Data access control for Amazon OpenSearch Serverless](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html). Skipping this step or not running it correctly will result in permission errors, and your Lambda code won’t be able to query the Geoportal catalog.

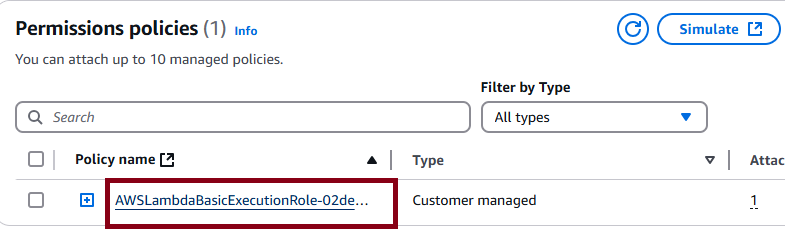


**Edit Role of lambda function**

1. On the Lambda console, navigate to your function.
2. On the Configuration tab, in the Permissions section, Click on Execution role



1. Open Permission Policies

  
Add below permission

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"aoss:DashboardsAccessAll",

"aoss:APIAccessAll"

],

"Resource": "\*"

}

# Create the API in API Gateway

To create your API using the API Gateway console

1. Navigate to the API Gateway console at <https://console.aws.amazon.com/apigateway/home>. On the left navigation pane, choose APIs.
2. Locate REST API (not private) and choose Build.
3. On the following page, locate the Create new API section and make sure New API is selected.
4. Configure the following fields:
   * API name: opensearch-api
   * Description: Public API for searching an Amazon OpenSearch Service domain
   * Endpoint Type: Regional
5. Choose Create API.
6. Choose Actions and Create Method.
7. Select POST in the dropdown and click the checkmark to confirm.
8. Configure the following settings, then choose Save:

| Setting | Value |
| --- | --- |
| Integration type | Lambda function |
| Use Lambda proxy integration | Yes |
| Lambda region | *us-east-1* |
| Lambda function | opensearch-lambda |
| Use default timeout | Yes |

1. Configure the method request

Choose Method Request and configure the following settings:

| Setting | Value |
| --- | --- |
| Authorization | NONE |
| Request Validator | None |
| API Key Required | false |

Under URL Query String Parameters, choose Add query string and configure the following two parameters:

| Setting | Value |
| --- | --- |
| Name | from  size |
| Required | Yes |

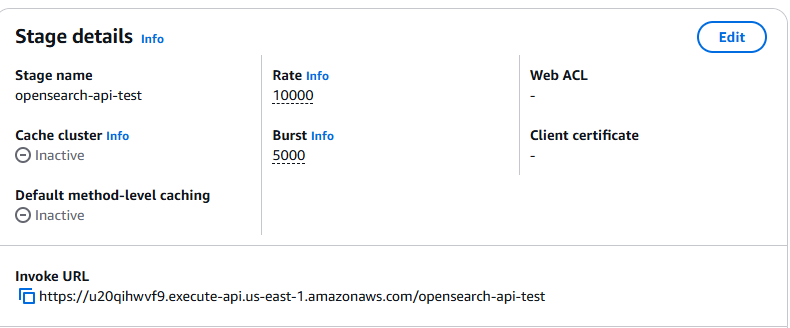
[**Enable CORS**](https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors-console.html) on the POST resource. Under **Advanced**, set **Access-Control-Allow-Credentials** to 'true'.

# Deploy the API and configure a stage

The API Gateway console lets you deploy an API by creating a deployment and associating it with a new or existing stage.

1. Choose **Actions** and **Deploy API**.
2. For **Deployment stage** choose **New Stage** and name the stage opensearch-api-test.
3. Choose **Deploy.**
4. Configure the following settings in the stage editor, then choose **Save Changes**:

| **Setting** | **Value** |
| --- | --- |
| Enable throttling | Yes |
| Rate | 1000 |
| Burst | 500 |



Note down the invoke URL and configure this in app-context.xml

<beans:property name=*"awsAPIGatewayEndpoint"* value=*" "*/> <!-- API gateway endpoint -->