

## Note

The exercises in this course will have an associated charge in your AWS account. In this exercise, you will create the following resources:

- AWS Identity and Access Management (IAM) policy and user (policies and users are AWS account features, offered at no additional charge)
- Amazon Simple Storage Service (Amazon S3) bucket
- Amazon OpenSearch Service cluster
- AWS Lambda function
- Amazon API Gateway application programming interface (API)

**The final exercise task includes instructions to delete all the resources that you create for this exercise.**

Familiarize yourself with [Amazon S3 pricing](#), [Amazon OpenSearch Service](#), [AWS Lambda pricing](#), [Amazon API Gateway pricing](#), and the [AWS Free Tier](#).

# Exercise 1: Creating an Amazon OpenSearch Service Cluster

Your company is measuring the temperature of water in lakes, ponds, and streams around your area. They have deployed sensors to each body of water they are monitoring. The sensors can make HTTPS calls that include the sensor ID and temperature reading on the request payload. The plan is for the sensors to record data every 15 minutes on temperature readings. They will then upload the data to the AWS data lake, through an HTTPS request, for each 15-minute interval.

You are tasked with creating a way to ingest data into a data lake that's hosted on AWS. The data lake uses Amazon S3 as the storage layer and OpenSearch Service for index and search capabilities.

You use Amazon API Gateway to ingest the data that is sent from the sensor, which invokes a Lambda function. This Lambda function takes the payload of the request. First, it writes a file to an S3 bucket. The file name is a combination of the sensor ID and the timestamp. The file contents send the sensorID, the timestamp, and the temperature reading. The Lambda function then loads the record to OpenSearch Service.

This method of ingesting data has a relatively small payload for each individual record. However, potentially hundreds of sensors from the area may post data every 15 minutes, which could produce many records.

After the data is stored in Amazon S3, you could then use other analysis tools for various use cases. For example, you might have other weather-related data that you are ingesting by using another solution. You could combine that data with the sensor data to provide insights. Another example is that other people in the organization might want access to the raw data to do their own visualization or analysis by using the tools they're familiar with.

Data lake architectures often combine many different ingestion methods and analysis methods into one architecture. In this exercise, you focus on one ingestion solution by using API Gateway and Lambda, and using Amazon OpenSearch as the search and indexing solution. However, you could also use Amazon Kinesis for ingestion and AWS Glue for cataloging the data. You will learn about these services in more depth in future lessons. One AWS service isn't any better than another AWS service. Instead, some services best fit certain use cases—and you can decide which services you want to use in your data lake designs.

## Setting up

In this exercise, you ingest mock data into a data lake. Download the following .zip file that contains sample data: [upload-data](#). You will use the file for the Lambda function in the following task.

Before you begin ingesting data into a data lake, you must create an AWS Identity and Access Management (IAM) role. An IAM role defines specific account permissions. In particular, what you can or cannot do in the AWS Cloud.

To complete instructions in this exercise, you must grant full access permissions to Amazon Simple Storage Service and Amazon OpenSearch Services.

1. In the AWS Management Console, choose **Services**, and then open the IAM dashboard by choosing **IAM**.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. Under **Use case**, choose **Lambda** and then choose **Next**.
5. In the **Permissions policies** search box, enter `AmazonS3` and press Enter.
6. From the list of results, select `AmazonS3FullAccess`.
7. Clear the `AmazonS3` filter.
8. In the search box, enter `AmazonES` and press Enter.
9. Select `AmazonESFullAccess` and then choose **Next**.
10. For **Role name**, paste `data-lake-week-2`.
11. Choose **Create role**.

## Task 1: Creating an Amazon OpenSearch Service cluster

You use OpenSearch Service for its cataloging and indexing capabilities. Before you start ingesting documents to your OpenSearch Service domain, you must create a domain. You could also use AWS Glue for cataloging your data. However, you explore that service in a future lesson.

In this task, you create an OpenSearch Service domain.

1. Choose **Services**, and search for and open **Amazon OpenSearch Service**.
2. Choose **Create domain** and configure the following settings.
  - **Domain name:** `water-temp-domain`

- **Deployment type:** *Development and testing*
- **Version:** Choose the latest version of *OpenSearch*
- **Data nodes > Instance type:** *t3.small.search*
- **Network:** *Public access*
- **Fine-grained access control:** Deselect this setting
- **Access policy > Domain access policy:** *Configure domain level access policy*

In the **Elements** section, you can configure a principal that is allowed to access the domain. Use IPv4 address to restrict access to the OpenSearch Service domain.

3. First, find your IPv4 address by using an online lookup service such as [What Is My IP](#) and take note of your **IPv4** address.
4. In the **Elements** section, configure the following settings.
  - **Type:** *IPv4 address*
  - **Principal:** Replace an asterisk ( *\** ) with your *IPv4 address*
  - **Action:** *Allow*
5. Choose the **JSON** tab and note that the policy only allows your IPv4 address to access the OpenSearch Service domain:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "es:*"
      ],
      "Resource": "arn:aws:es:us-east-1:000000000000:domain/water-temp-domain/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "x.x.x.x"
          ]
        }
      }
    }
  ]
}
```

1. Choose **Create**. *The domain-creation process can take up to 15 minutes to complete.*

## Task 2: Creating an Amazon S3 bucket

In this task, you create an object storage bucket for data collected from sensors.

Though you load your data into OpenSearch Service, Amazon S3 serves as the storage layer for your data lake. In this exercise, you use OpenSearch Service for one specific use case. You load the data into both OpenSearch Service and Amazon S3. The idea is that you likely have multiple use cases for the data in a data lake. Each use case uses its own set of appropriate services and tooling. By storing the raw data in Amazon S3, the data is then accessible for many use cases and various services.

To create an Amazon S3 bucket:

1. Choose **Services**, and search for and open **S3**.
2. Choose **Create bucket**.
3. The bucket name must be globally unique and DNS compliant. Name the bucket similar to the following example, using your initials for the FMI, which stands for a *Fill Me In*. When you replace the FMI with your own value, make sure that you also delete the angle brackets (<>).

```
data-lakes-week2-<FMI>
```

**Note:** If bucket name isn't available after you add your initials, add some numbers to the end of the name.

Examples:

```
data-lakes-week2-emr
```

```
data-lakes-week2-emr1220
```

1. Under **Region**, the selected Region should be *US East (N. Virginia) us-east-1*. The bucket Region should be the same Region that your OpenSearch Service cluster is in.
2. Choose **Create bucket**.
3. Take note of the S3 bucket name. You will need to use its name in future steps.

You now have an S3 bucket that is the storage layer for your data lake. In a later task, you modify the bucket access policy so that Lambda can write files to the bucket. You're going to learn about bucket access policies in future lessons.

## Task 3: Creating the AWS Lambda function

AWS Lambda is a serverless compute service. The code that you upload to a Lambda function doesn't run continually. Instead, it runs when an event occurs.

In this task, the code for the Lambda function is already written. However, you need to create and configure the Lambda function.

The Lambda function first captures the data that is on the payload of the incoming request. Next, it uploads a JavaScript Object Notation (JSON) document to Amazon S3. The Lambda function then uploads the document to OpenSearch Service.

To create the Lambda function:

1. Choose **Services**, and search for and open **Lambda**.
2. Choose **Create function** and configure the following settings.
  - **Author from scratch:** Keep this option selected
  - **Function name:** `upload-data`
  - **Runtime:** Select the *latest supported version of Python*
  - **Permissions:** Expand *Change default execution role*
  - **Execution role:** *Use an existing role*
  - **Existing role:** *data-lake-week-2*

### 3. Choose **Create function**.

The Lambda function you create needs access to permissions to sign application programming interface (API) calls that write to Amazon S3 and OpenSearch Service.

1. Scroll to the **Code** tab.
2. Choose **Upload from** and select **.zip file**.
3. Choose **Upload**.
4. Browse to where you saved the `upload-data.zip` file, choose the file, and choose **Open**.
5. Back in the **Upload a .zip file** window, choose **Save**.
6. In the **Code** tab, scroll to **Runtime settings** and choose **Edit**.
7. In the **Handler** box, replace the existing value with `lambda.handler` and choose **Save**.
8. Choose the **Configuration** tab.
9. On the tab menu, choose **Environment variables** and then choose **Edit**.
10. Choose **Add environment variable** and configure the following settings.
  - **Key:** `S3_BUCKET`
  - **Value:** Paste your bucket name

Example:

```
datalakes-week2-emr
```

Environment variables let you adjust your function's behavior without updating code. In a future step, you add another environment variable for the OpenSearch Service domain.

11. Choose **Save**.
12. In the **Configuration** tab menu, choose **Permissions**.
13. Under **Role name**, choose the **data-lake-week-2** link.

**Note:** This action opens the IAM console.

14. Copy the role Amazon Resource Name (ARN), which should look similar to the following example:

```
arn:aws:iam::000000000000:role/data-lake-week-2
```

You need this ARN for the following task.

## Task 4: Modifying the S3 bucket policy and OpenSearch Service cluster for Lambda access

The Lambda function writes data to the S3 bucket. You already associated an IAM role with the Lambda function in the **Setting up** section. However, the role by itself won't allow

access. You must also modify the S3 bucket policy, which determines whether AWS principals are allowed or denied access to the bucket.

1. Return to the **Amazon S3** console.
2. Open the bucket that you created previously in this exercise.
3. Choose the **Permissions** tab, scroll to **Bucket policy**, and choose **Edit**.
4. In the following JSON code, replace the first FMI with the Lambda role ARN:

```
{
  "Version": "2012-10-17",
  "Id": "ExamplePolicy",
  "Statement": [
    {
      "Sid": "ExampleStmt",
      "Effect": "Allow",
      "Principal": {
        "AWS": "<FMI>"
      },
      "Action": "s3:*",
      "Resource": "<FMI>/*"
    }
  ]
}
```

Example:

```
    "Principal": {
      "AWS": "arn:aws:iam::xxxxxxxxxxxx:role/data-lake-week-2"
    },
```

1. Replace the second FMI with the ARN of your bucket:

Example:

```
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::datalakes-week2-emr/*"
  }
```

1. In the **Bucket policy editor**, replace the placeholder policy with the bucket policy that you configured. The final policy should similar to the following example, but *with your account number and bucket name*:

```
{
  "Version": "2012-10-17",
  "Id": "ExamplePolicy",
  "Statement": [
    {
      "Sid": "ExampleStmt",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::xxxxxxxxxxxx:role/data-lake-week-2"
      },
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::datalakes-week2-emr/*"
    }
  ]
}
```

```

    },
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::datalakes-week2-xxx/*"
  }
]
}

```

1. Choose **Save changes**. This policy allows the Lambda function to upload the temperature files to your bucket.
2. Return to the **OpenSearch Service** console.
3. Choose **water-temp-domain**.

The **Domain status** should now say *Active*. If not, wait a few minutes and refresh the page until the status is *Active*.

**Note:** Make sure that you are in the **N. Virginia** Region.

4. Note the **Domain endpoint**, which should look similar to the following example:

```
https://search-water-temp-domain-xxxxxxxxxxxxxxxxxxxxxxxx.us-east-1.es.amazonaws.com
```

1. Choose **Actions** and select **Edit security configuration**.
2. Scroll to **Access policy** and review the policy. Currently, the policy restricts access by IP address. The Lambda function should also have access. The current policy should look similar to the following example:

Example:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:000000000000:domain/water-temp-domain/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "x.x.x.x"
        }
      }
    }
  ]
}

```

1. Add Lambda access to the policy and update values with your account number. The policy must look similar to the following example, but *with your account number and IP address*:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:000000000000:domain/water-temp-domain/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "x.x.x.x"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::000000000000:role/data-lake-week-2"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:000000000000:domain/water-temp-domain/*"
    }
  ]
}
```

1. Choose **Save changes**.

## Task 5: Modifying the Lambda function and creating an API Gateway endpoint

You now have an OpenSearch Service domain and an S3 bucket. You need to modify one final thing for the Lambda function to work.

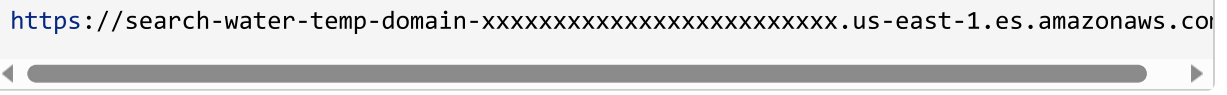
After the Lambda function is configured, you need a way to run the function. Lambda functions run based on events. In this case, the sensors that record the temperature data make HTTPS POST requests, with the data on the payload. You use API Gateway to create an API endpoint that receives the HTTPS POST requests. After API Gateway receives and validates the request, it invokes the Lambda function and passes the information to it. The Lambda function then writes the data to Amazon S3 and OpenSearch Service.

### Step 1: Add a new variable in AWS Lambda

1. Return to the **Lambda** console.
2. Choose the **upload-data** function and then choose the **Configuration** tab.
3. On the tab menu, choose **Environment variables** and then choose **Edit**.
4. Choose **Add environment variable** and configure the following settings.
  - **Key:** `ES_DOMAIN_URL`
  - **Value:** Paste the OpenSearch Service domain endpoint that you noted previously.



It should look similar to the following example:



```
https://search-water-temp-domain-xxxxxxxxxxxxxxxxxxxxxxxx.us-east-1.es.amazonaws.com
```

Ensure that the OpenSearch Service domain endpoint doesn't have a slash (/) at the end.

5. Choose **Save**.

## Step 2: Create a REST API

After you set up the variable for OpenSearch Service, you create a REST API in Amazon API Gateway to receive data from the sensors. In this exercise, you manually enter test data to simulate requests from the sensors.

1. Choose **Services**, and search for and open **API Gateway**.
2. On the **REST API** card, choose **Build**.
3. If needed, close the **Create your first API** dialog box by choosing **OK**.
4. Under **Create new API**, select **New API**.
5. In the **API name** box, paste `sensor-data` and choose **Create API**.
6. In the **Resources** pane, choose **Actions**, and then select **Create Method**.
7. Choose the dropdown menu, select **POST**, and confirm your selection by choosing the checkmark.
8. In the **POST - Setup** pane, keep the **Integration type** setting at **Lambda Function**.
9. In the **Lambda Function** box, enter `upload-data` and select it when it appears.
10. Choose **Save**, and in the **Add Permission to Lambda Function** dialog box, choose **OK**.
11. In the **POST - Method Execution** pane, choose **TEST**.
12. Scroll to the **Request Body** box and paste the following JSON code:

```
{
  "sensorID" : "0025",
  "temperature" : "67"
}
```

1. Choose **Test**. In the **Response Body** section, you should see a response similar to the following example:

```
{
  "Response": "Data Uploaded",
  "sensorID": "0025",
  "temperature": "67"
}
```

This test simulates the POST request that would come in from different IoT sensors.

## Task 6: Checking OpenSearch Service for data

In this task, you use the OpenSearch Service domain to search your data from the browser by submitting a GET request.

You will need your OpenSearch Service domain URL, which you noted previously, to replace the FMI.

1. Open a text editor of your choice, and paste the following text into the editor. Replace the FMI with your OpenSearch Service domain URL.

**Note:** When you replace the FMI with your own value, make sure that you also remove the angle brackets (<>).

```
<FMI>/lambda-s3-index/lambda-type/_search?pretty
```

Example:

```
https://search-water-temp-domain-xxxxxxxxxxxxxxxxxxxxxxxxxxxx.us-east-1.es.amazonaws.com/lam
```

1. Copy the search URL and in a new browser tab, paste the URL.

You should be able to see some test index data that's being returned.

2. You can run a few more tests using different values for **sensorID** and **temperatures** in the API Gateway test feature. For example, you could paste the following example into the **Request Body** box, and then choose **Test**.

```
{  
  "sensorID" : "0026",  
  "temperature" : "70"  
}
```

1. Return to the **Amazon S3** console and open your bucket.
2. You should see several .json files. Download one of the files and review its contents. It should contain the test data that was sent through the API Gateway endpoint, similar to the following example:

```
{"sensorID": "0026", "timestamp": "20201019153601", "temperature": "70"}
```

In the real world, your data wouldn't originate from API Gateway. Instead, it would come from a fleet of sensors and devices. By testing with API Gateway, you can generate a POST request as if it was coming from outside AWS. In a real-world setting, API Gateway would accept the requests from an external device, and then invoke the backend.

By creating the OpenSearch Service domain, you also now have access to Kibana. You can explore Kibana to view and analyze the data that's loaded into the OpenSearch Service domain.

# Cleaning up

In this task, you delete the resources that you created for this exercise.

1. Delete the IAM role.
  - Open the **IAM** console.
  - In the navigation pane, choose **Roles**.
  - In the **Search** box, enter `data-lake-week-2`.
  - Delete the **data-lake-week-2** role and confirm the deletion.
2. Delete the OpenSearch Service domain.
  - Open the OpenSearch Service console
  - Delete **water-temp-domain** and confirm the deletion.
3. Delete the Amazon S3 bucket.
  - Open the **Amazon S3** console.
  - Empty and delete the buckets that you created for this exercise, and confirm their deletion.
4. Delete the Lambda function.
  - Open the **AWS Lambda** console.
  - Delete **upload-data** and confirm the deletion.
5. Delete the API Gateway endpoint.
  - Open the **API Gateway** console.
  - Delete **sensor-data** and confirm the deletion.

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. Corrections, feedback, or other questions? Contact us at <https://support.aws.amazon.com/#/contacts/aws-training>. All trademarks are the property of their owners.