AWS SDK for Python (Boto3)

Contents

[AWS SDK for Python (Boto3) 4](#_Toc94864053)

[Accessing AWS with Python 4](#_Toc94864054)

[Installation 4](#_Toc94864055)

[AWS Common Runtime (CRT) 4](#_Toc94864056)

[AWS CLI – Command Line Interface 5](#_Toc94864057)

[AWS CLI Commands 5](#_Toc94864058)

[Docker for CLI 5](#_Toc94864059)

[Configuration 6](#_Toc94864060)

[Docker Troubleshooting 6](#_Toc94864061)

[Before a Reset of Docker. 6](#_Toc94864062)

[After the Reset: 6](#_Toc94864063)

[https://stackoverflow.com/questions/57460393/the-docker-client-must-be-run-elevated-to-connect 7](#_Toc94864064)

[ARN – Amazon Resource Number 7](#_Toc94864065)

[ARN Format 8](#_Toc94864066)

[Paths in ARNs 8](#_Toc94864067)

[Lambda Function hander 8](#_Toc94864068)

[Event Object 9](#_Toc94864069)

[Example custom event – weather data 9](#_Toc94864070)

[Example service event – Amazon SNS notification 9](#_Toc94864071)

[Lambda Handler example 1 10](#_Toc94864072)

[Example 2: Parsing a response 10](#_Toc94864073)

[AWS resources 12](#_Toc94864074)

[AWS service endpoints 12](#_Toc94864075)

[Managing AWS Regions 12](#_Toc94864076)

[AWS service quotas 12](#_Toc94864077)

[Tagging AWS resources 12](#_Toc94864078)

[Amazon Resource Names (ARNs) 12](#_Toc94864079)

[AWS APIs 12](#_Toc94864080)

[Error retries and exponential backoff in AWS 12](#_Toc94864081)

[Signing AWS API requests 12](#_Toc94864082)

[AWS SDK support for Amazon S3 client-side encryption 12](#_Toc94864083)

[AWS Lambda with other services 12](#_Toc94864084)

[Using Lambda with Amazon SQS 13](#_Toc94864085)

[Test with the S3 trigger 14](#_Toc94864086)

[Lambda event source mappings 15](#_Toc94864087)

[ListEventSourceMappings 15](#_Toc94864088)

[GetEventSourceMapping 16](#_Toc94864089)

[Reference 17](#_Toc94864090)

# AWS SDK for Python (Boto3)

<https://docs.aws.amazon.com/pythonsdk/?id=docs_gateway>

# Accessing AWS with Python

The AWS SDK for Python (Boto3) provides a Python API for AWS infrastructure services. Using the SDK for Python, you can build applications on top of Amazon S3, Amazon EC2, Amazon DynamoDB, and more.

## Installation

1. install Python 3.6 or later; support for Python 3.5 and earlier is deprecated.
2. Install the new version of Boto3. This will also install Botocore, which it requires:

>> python3 -m pip install boto3

# Install Boto3 version 1.0 specifically

>>pip install boto3==1.0.0

# Make sure Boto3 is no older than version 1.15.0

>>pip install boto3>=1.15.0

# Avoid versions of Boto3 newer than version 1.15.3

>>pip install boto3<=1.15.3

1. Verify that the SDK is using the correct version of Python:

>>$ python3 -c "import boto3, sys; print(f'{sys.version} \nboto3: {boto3.\_\_version\_\_}')"

3.8.6 (default, Jan 7 2021, 17:11:21)

[GCC 7.3.1 20180712 (Red Hat 7.3.1-11)]

boto3: 1.16.15

# AWS Common Runtime (CRT)

In addition to the default install of Boto3, you can choose to include the new AWS Common Runtime (CRT). The AWS CRT is a collection of modular packages that serve as a new foundation for AWS SDKs. Each library provides better performance and minimal footprint for the functional area it implements. Using the CRT, SDKs can share the same base code when possible, improving consistency and throughput optimizations across AWS SDKs.

When the AWS CRT is included, Boto3 uses it to incorporate features not otherwise available in the AWS SDK for Python.

At this time, Boto3 uses the AWS CRT's authentication package (aws-c-auth) to add support for the AWS Signature Version 4 (sigv4) signer, which adds authentication to your AWS requests using your security credentials (your AWS access key and secret access key).

Boto3 doesn't use the AWS CRT by default but you can opt into using it by specifying the crt extra feature when installing Boto3:

>> pip install boto3[crt]

To revert to the non-CRT version of Boto3, use this command:

>> pip uninstall awscrt

If you need to re-enable CRT, reinstall boto3[crt] to ensure you get a compatible version of awscrt:

>> pip install boto3[crt]

# ARN – Amazon Resource Number

|  |  |
| --- | --- |
| * Uniquely identify AWS resources. * Require when you need to specify a resource unambiguously across all of AWS, such as in   + IAM policies   + Amazon Relational Database Service (Amazon RDS) tags   + API calls | **Partition** – Resource Location   * aws - AWS Regions * aws-cn - China Regions * aws-us-gov - AWS GovCloud (US) Regions |
| **Service** - service namespace that identifies the AWS product   * AWS S3 * AWS Lambda * AWS SQS | **Region – Regional endpoints**   * Some services, such as IAM, do not support Regions * When you use an endpoint with no Region, AWS routes the Amazon EC2 request to US East (N. Virginia) (us-east-1), which is the default Region for API calls. * **FIPS endpoints** - [Federal Information Processing Standard (FIPS) 140-2](http://aws.amazon.com/compliance/fips/) |
| **account-id** - ID of the AWS account that owns the resource | **resource-id -** resource or a [resource path](https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html#arns-paths)   * Often a human resource |

## ARN Format

arn:*partition*:*service*:*region*:*account-id*:*resource-id*

arn:*partition*:*service*:*region*:*account-id*:*resource-type*/*resource-id*

arn:*partition*:*service*:*region*:*account-id*:*resource-type*:*resource-id*

## Paths in ARNs

Resource ARNs can include a path. For example, in Amazon S3, the resource identifier is an object name that can include slashes (/) to form a path. Similarly, IAM user names and group names can include paths.

Paths can include a wildcard character, namely an asterisk (\*). For example, if you are writing an IAM policy, you can specify all IAM users that have the path product\_1234 using a wildcard as follows:

* arn:aws:iam::123456789012:user/Development/product\_1234/\*
* "Resource":"arn:aws:iam::123456789012:user/\*"
* "Resource":"arn:aws:iam::123456789012:group/\*"
* arn:aws:s3:::my\_corporate\_bucket/\*
* arn:aws:s3:::my\_corporate\_bucket/Development/\*

<https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html>

# Lambda Function hander

|  |  |
| --- | --- |
| * The Lambda function handler is the **method in your function code that processes events**. * When your function is invoked, Lambda runs the handler method. * When the handler exits or returns a response, it becomes available to handle another event. * Lambda invokes your function in an [execution environment](https://docs.aws.amazon.com/lambda/latest/dg/runtimes-context.html). | **Syntax:**  def handler\_name(event, context):  ...  return some\_value |
| * A function handler can be any name; however, the default name in the Lambda console is **lambda\_function.lambda\_handler**. * **lambda\_function** – python\_file.py file name * **lambda\_handler** – python function (def statement) name * This function handler name reflects the function name (lambda\_handler) and the file where the handler code is stored (lambda\_function.py). | **How it works:**   1. When Lambda invokes your function handler, the [Lambda runtime](https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html) passes two arguments to the function handler:  * The first argument is the [event object](https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-concepts.html#gettingstarted-concepts-event).  see [Using AWS Lambda with other services](https://docs.aws.amazon.com/lambda/latest/dg/lambda-services.html). * The second argument is the context object. A context object is passed to your function by Lambda at runtime. This object provides methods and properties that provide information about the invocation, function, and runtime environment. |
| **Returning a value**   * Optionally, a handler can return a value. What happens to the returned value depends on the [invocation type](https://docs.aws.amazon.com/lambda/latest/dg/lambda-invocation.html) and the [service](https://docs.aws.amazon.com/lambda/latest/dg/lambda-services.html) that invoked the function. * If you use an Event an [Asynchronous invocation](https://docs.aws.amazon.com/lambda/latest/dg/invocation-async.html) invocation type, the value is discarded. |  |

# How do I determine if an Event notifications worked in S3?

As mentioned in the comments, the S3 event notification is passed as the event argument to your Lambda handler.

This isn't logged *automatically* anywhere (unlike general invocation information like request ID, memory size, memory usage etc).

To view the event, you need to *manually* log this in your Lambda & then view it in CloudWatch.

Use print(event) to post it to the CloudWatch logs of the function.

Create a lambda function:

**def handler\_name(event, context):**

**print(event)**

Configure this function to be triggered on an S3-upload.

<https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html>

## CloudWatch and Error Checking

Most of the time the problem is a syntax error in the AWS Lambda. Python syntax errors can be found in the CloudWatch logs by log group. Below is an example of a syntax error.

CloudWatch > Log groups > /aws/lambda/sqs\_carrier\_pharmacy\_gap

Text

Description automatically generated

## Event Object

An event is a JSON-formatted document that contains data for a Lambda function to process. The runtime converts the event to an object and passes it to your function code. When you invoke a function, you determine the structure and contents of the event.

### Example custom event – weather data

{

"TemperatureK": 281,

"WindKmh": -3,

"HumidityPct": 0.55,

"PressureHPa": 1020

}

### Example service event – Amazon SNS notification

When an AWS service invokes your function, the service defines the shape of the event.

{

"Records": [

{

"Sns": {

"Timestamp": "2019-01-02T12:45:07.000Z",

"Signature": "tcc6faL2yUC6dgZdmrwh1Y4cGa/ebXEkAi6RibDsvpi+tE/1+82j...65r==",

"MessageId": "95df01b4-ee98-5cb9-9903-4c221d41eb5e",

"Message": "Hello from SNS!",

...

## Lambda Handler example 1

def lambda\_handler(event, context):

message = 'Hello {} {}!'.format(event['first\_name'], event['last\_name'])

return {

'message' : message

}

You can use the following event data to [invoke the function](https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html#get-started-invoke-manually):

{

"first\_name": "John",

"last\_name": "Smith"

}

The response shows the event data passed as input:

{

"message": "Hello John Smith!"

}

## Example 2: Parsing a response

The following example shows a function called lambda\_handler that uses the python3.8 [Lambda runtime](https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html). The function uses event data passed by Lambda at runtime. It parses the [environment variable](https://docs.aws.amazon.com/lambda/latest/dg/configuration-envvars.html) in AWS\_REGION returned in the JSON response.

import os

import json

def lambda\_handler(event, context):

json\_region = os.environ['AWS\_REGION']

return {

"statusCode": 200,

"headers": {

"Content-Type": "application/json"

},

"body": json.dumps({

"Region ": json\_region

})

}

You can use any event data to [invoke the function](https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html#get-started-invoke-manually):

{

"key1": "value1",

"key2": "value2",

"key3": "value3"

}

Lambda runtimes set several environment variables during initialization. For more information on the environment variables returned in the response at runtime, see [Using AWS Lambda environment variables](https://docs.aws.amazon.com/lambda/latest/dg/configuration-envvars.html).

The function in this example depends on a successful response (in 200) from the Invoke API. For more information on the Invoke API status, see the [Invoke](https://docs.aws.amazon.com/lambda/latest/dg/API_Invoke.html) Response Syntax.

<https://docs.aws.amazon.com/lambda/latest/dg/python-handler.html>

# AWS resources

The following pages provide information that helps you work with AWS resources.

## [AWS service endpoints](https://docs.aws.amazon.com/general/latest/gr/rande.html)

## [Managing AWS Regions](https://docs.aws.amazon.com/general/latest/gr/rande-manage.html)

## [AWS service quotas](https://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html)

## [Tagging AWS resources](https://docs.aws.amazon.com/general/latest/gr/aws_tagging.html)

## [Amazon Resource Names (ARNs)](https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html)

# AWS APIs

The following pages provide information that is useful when using an AWS API.

**Contents**

## [Error retries and exponential backoff in AWS](https://docs.aws.amazon.com/general/latest/gr/api-retries.html)

## [Signing AWS API requests](https://docs.aws.amazon.com/general/latest/gr/signing_aws_api_requests.html)

## [AWS SDK support for Amazon S3 client-side encryption](https://docs.aws.amazon.com/general/latest/gr/aws_sdk_cryptography.html)

# AWS Lambda with other services

| **Service** | **Method of invocation** |
| --- | --- |
| [Amazon Alexa](https://docs.aws.amazon.com/lambda/latest/dg/services-alexa.html) | Event-driven; synchronous invocation |
| [Amazon Managed Streaming for Apache Kafka](https://docs.aws.amazon.com/lambda/latest/dg/with-msk.html) | Lambda polling |
| [Self-managed Apache Kafka](https://docs.aws.amazon.com/lambda/latest/dg/with-kafka.html) | Lambda polling |
| [Amazon API Gateway](https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway.html) | Event-driven; synchronous invocation |
| [AWS CloudFormation](https://docs.aws.amazon.com/lambda/latest/dg/services-cloudformation.html) | Event-driven; asynchronous invocation |
| [Amazon CloudFront (Lambda@Edge)](https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html) | Event-driven; synchronous invocation |
| [Amazon EventBridge (CloudWatch Events)](https://docs.aws.amazon.com/lambda/latest/dg/services-cloudwatchevents.html) | Event-driven; asynchronous invocation |
| [Amazon CloudWatch Logs](https://docs.aws.amazon.com/lambda/latest/dg/services-cloudwatchlogs.html) | Event-driven; asynchronous invocation |
| [AWS CodeCommit](https://docs.aws.amazon.com/lambda/latest/dg/services-codecommit.html) | Event-driven; asynchronous invocation |
| [AWS CodePipeline](https://docs.aws.amazon.com/lambda/latest/dg/services-codepipeline.html) | Event-driven; asynchronous invocation |
| [Amazon Cognito](https://docs.aws.amazon.com/lambda/latest/dg/services-cognito.html) | Event-driven; synchronous invocation |
| [AWS Config](https://docs.aws.amazon.com/lambda/latest/dg/services-config.html) | Event-driven; asynchronous invocation |
| [Amazon Connect](https://docs.aws.amazon.com/lambda/latest/dg/services-connect.html) | Event-driven; synchronous invocation |
| [Amazon DynamoDB](https://docs.aws.amazon.com/lambda/latest/dg/with-ddb.html) | Lambda polling |
| [Amazon Elastic File System](https://docs.aws.amazon.com/lambda/latest/dg/services-efs.html) | Special integration |
| [Elastic Load Balancing (Application Load Balancer)](https://docs.aws.amazon.com/lambda/latest/dg/services-alb.html) | Event-driven; synchronous invocation |
| [AWS IoT](https://docs.aws.amazon.com/lambda/latest/dg/services-iot.html) | Event-driven; asynchronous invocation |
| [AWS IoT Events](https://docs.aws.amazon.com/lambda/latest/dg/services-iotevents.html) | Event-driven; asynchronous invocation |
| [Amazon Kinesis](https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html) | Lambda polling |
| [Amazon Kinesis Data Firehose](https://docs.aws.amazon.com/lambda/latest/dg/services-kinesisfirehose.html) | Event-driven; synchronous invocation |
| [Amazon Lex](https://docs.aws.amazon.com/lambda/latest/dg/services-lex.html) | Event-driven; synchronous invocation |
| [Amazon MQ](https://docs.aws.amazon.com/lambda/latest/dg/with-mq.html) | Lambda polling |
| [Amazon Simple Email Service](https://docs.aws.amazon.com/lambda/latest/dg/services-ses.html) | Event-driven; asynchronous invocation |
| [Amazon Simple Notification Service](https://docs.aws.amazon.com/lambda/latest/dg/with-sns.html) | Event-driven; asynchronous invocation |
| [Amazon Simple Queue Service](https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html) | Lambda polling |
| [Amazon Simple Storage Service (Amazon S3)](https://docs.aws.amazon.com/lambda/latest/dg/with-s3.html) | Event-driven; asynchronous invocation |
| [Amazon Simple Storage Service Batch](https://docs.aws.amazon.com/lambda/latest/dg/services-s3-batch.html) | Event-driven; synchronous invocation |
| [Secrets Manager](https://docs.aws.amazon.com/lambda/latest/dg/with-secrets-manager.html) | Event-driven; synchronous invocation |
| [AWS X-Ray](https://docs.aws.amazon.com/lambda/latest/dg/services-xray.html) | Special integration |

# Using Lambda with Amazon SQS

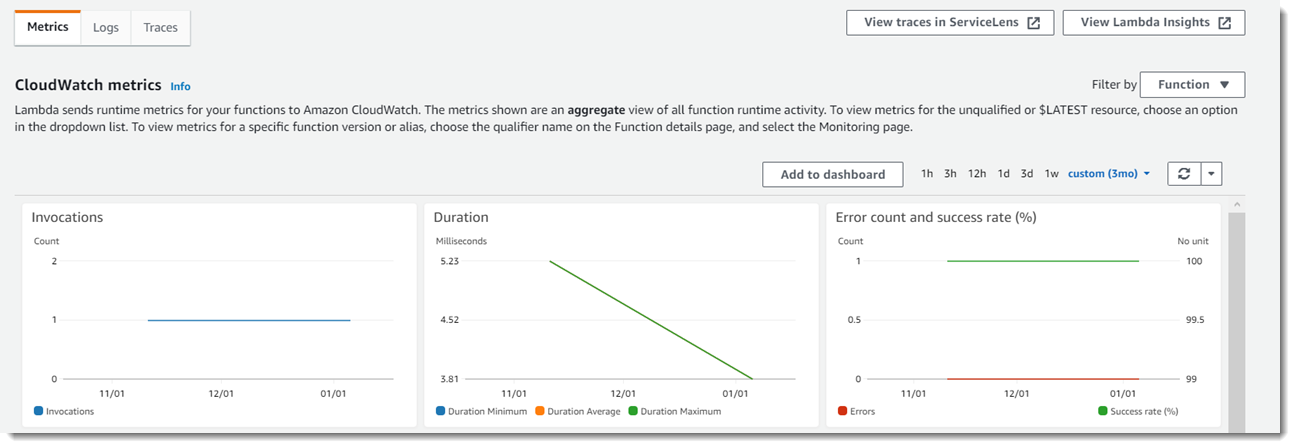
|  |  |
| --- | --- |
| Lambda [event source mappings](https://docs.aws.amazon.com/lambda/latest/dg/invocation-eventsourcemapping.html) support [standard queues](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/standard-queues.html) and [first-in, first-out (FIFO) queues](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html). With Amazon SQS, you can offload tasks from one component of your application by sending them to a queue and processing them asynchronously. | * Lambda polls the queue and invokes your Lambda function synchronously with an event that contains queue messages. * Lambda reads messages in batches and invokes your function once for each batch. * When your function successfully processes a batch, Lambda deletes its messages from the queue. * The following example shows an event for a batch of two messages. |
| Configuring a queue as an event source  **To create a trigger**   1. Open the [Functions page](https://console.aws.amazon.com/lambda/home#/functions) of the Lambda console. 2. Choose the name of a function. 3. Under **Function overview**, choose **Add trigger**. 4. Choose a trigger type. 5. Configure the required options, and then choose **Add**. |  |

<https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html>

# Test with the S3 trigger

Invoke your function when you upload a file to the Amazon S3 source bucket.

To test the Lambda function using the S3 trigger

1. On the [Buckets page](https://console.aws.amazon.com/s3/home) of the Amazon S3 console, choose the name of the source bucket that you created earlier.
2. On the **Upload** page, upload a few .jpg or .png image files to the bucket.
3. Open the [Functions page](https://console.aws.amazon.com/lambda/home#/functions) of the Lambda console.
4. Choose the name of your function (**my-s3-function**).
5. To verify that the function ran once for each file that you uploaded, choose the **Monitor** tab. This page shows graphs for the metrics that Lambda sends to CloudWatch. The count in the **Invocations** graph should match the number of files that you uploaded to the Amazon S3 bucket.

For more information on these graphs, see Monitoring functions on the Lambda console.

1. (Optional) To view the logs in the CloudWatch console, choose View logs in CloudWatch. Choose a log stream to view the logs output for one of the function invocations.

# Lambda event source mappings

An event source mapping uses permissions in the function's [execution role](https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.html) to read and manage items in the event source. Permissions, event structure, settings, and polling behavior vary by event source. For more information, see the linked topic for the service that you use as an event source.

To manage an event source with the [AWS Command Line Interface (AWS CLI)](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html) or an [AWS SDK](http://aws.amazon.com/getting-started/tools-sdks/), you can use the following API operations:

* [CreateEventSourceMapping](https://docs.aws.amazon.com/lambda/latest/dg/API_CreateEventSourceMapping.html)
* [ListEventSourceMappings](https://docs.aws.amazon.com/lambda/latest/dg/API_ListEventSourceMappings.html) - Lists event source mappings. Specify an EventSourceArn to only show event source mappings for a single event source.
* [GetEventSourceMapping](https://docs.aws.amazon.com/lambda/latest/dg/API_GetEventSourceMapping.html) - Returns details about an event source mapping. You can get the identifier of a mapping from the output of ListEventSourceMappings.
* [UpdateEventSourceMapping](https://docs.aws.amazon.com/lambda/latest/dg/API_UpdateEventSourceMapping.html)
* [DeleteEventSourceMapping](https://docs.aws.amazon.com/lambda/latest/dg/API_DeleteEventSourceMapping.html)

## ListEventSourceMappings

GET /2015-03-31/event-source-mappings/?EventSourceArn=*EventSourceArn*&FunctionName=*FunctionName*&Marker=*Marker*&MaxItems=*MaxItems* HTTP/1.1

* Amazon Kinesis - The ARN of the data stream or a stream consumer.
* Amazon DynamoDB Streams - The ARN of the stream.
* Amazon Simple Queue Service - The ARN of the queue.
* Amazon Managed Streaming for Apache Kafka - The ARN of the cluster.

<https://docs.aws.amazon.com/lambda/latest/dg/API_ListEventSourceMappings.html>

Returns details about an event source mapping. You can get the identifier of a mapping from the output of ListEventSourceMappings.

Example:

GET /2022-02-04/event-source-mappings/?EventSourceArn=*EventSourceArn*&FunctionName=*FunctionName*&Marker=*Marker*&MaxItems=*MaxItems* HTTP/1.1

>> aws lambda list-event-source-mappings --function-name my-function --output json

>> aws lambda list-event-source-mappings --function-name warehouse-ingestion-file-added-queue output json

>> aws lambda list-event-source-mappings --function-name warehouse-ingestion-file-added-queue output json

## GetEventSourceMapping

GET /2015-03-31/event-source-mappings/*UUID* HTTP/1.1

>> aws lambda get-event-source-mapping --uuid "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE

<https://awscli.amazonaws.com/v2/documentation/api/2.0.33/reference/lambda/get-event-source-mapping.html>

# CloudWatch and Logging

print(event)

{'Records': [

{'eventVersion': '2.1',

'eventSource': 'aws:s3',

'awsRegion': 'us-west-2',

'eventTime': '2022-11-23T13:22:14.351Z',

'eventName': 'ObjectCreated:Put',

'userIdentity':

{'principalId': 'AWS:AIDA6QAFBHRWYPXNVNFXE'},

'requestParameters':

{'sourceIPAddress': '74.118.237.130'},

'responseElements':

{'x-amz-request-id': 'ZE1KNVEN0K35TD2H',

'x-amz-id-2': 'UE/aZsdZI/T0ll5ZN+F4HHc9G41bRSNQW8uaDLWzMyIYSMGKc3fhUqOPzVZ7avks0bPlHdXXv0eNqZFCJ28V9ueGXx7jE08h'},

's3':

{'s3SchemaVersion': '1.0',

'configurationId': 'Pharmacy SQS file load',

'bucket':

{'name': 'warehouse-ingestion',

'ownerIdentity': {'principalId': 'A34Y2RP4MT6BT1'},

'arn': 'arn:aws:s3:::warehouse-ingestion'},

'object': {'key': 'pharmacy/pharmacy/aetna-fl-hmo/PY-2022/2022-12-12/AetnaHMOMedication20221107.xlsx',

'size': 2364581,

'eTag': 'c951a1e7aaf7f9e05c3faa462123b125',

'versionId': 'SiDM8MERGOBC3WnFQAd6vMn14p6tyQOS', 'sequencer': '00637E1E863C7E419D'}}}]}

# Reference

* [Boto3 documentation](https://boto3.amazonaws.com/v1/documentation/api/latest/index.html)
* <https://github.com/boto/boto3>
* [AWS Common Runtime (CRT) libraries](https://docs.aws.amazon.com/sdkref/latest/guide/common-runtime.html)
* [AWS Command Line Interface](https://aws.amazon.com/cli/) \*\*\*
* [AWS Command Line Interface User Guide](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html)
* [Using the official AWS CLI version 2 Docker image](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-docker.html)
* [AWS Command Line Interface blog](https://aws.amazon.com/blogs/developer/super-charge-your-aws-command-line-experience-with-aws-shell/)
* [AWS Toolkit for Visual Studio Code](https://docs.aws.amazon.com/toolkit-for-vscode/latest/userguide/welcome.html?icmpid=docs_tookitvscode_console)
* AWS Serverless Application Model (AWS SAM)
* [AWS Serverless Application Model Developer Guide](https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html)
* [Creating a new serverless application (local)](https://docs.aws.amazon.com/toolkit-for-vscode/latest/userguide/serverless-apps.html#serverless-apps-create)
* [Installing or updating the latest version of the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)
* [Using the official AWS CLI version 2 Docker image](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-docker.html)
* [Docker Run](https://spin.atomicobject.com/2018/10/04/docker-command-line/#:~:text=Docker%20run%20is%20for%20creating,containers%20that%20are%20already%20running.&text=The%20%2Dit%20runs%20Docker%20interactively,pseudo%2DTTY%20with%20STDIN).)
* <https://github.com/docker/for-win/issues/12523>
* [Amazon Resource Names (ARNs)](https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html)
* [Lambda function handler in Python](https://docs.aws.amazon.com/lambda/latest/dg/python-handler.html)
* [Using Lambda with Amazon SQS](https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html)
* [Using an Amazon S3 trigger to invoke a Lambda function](https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html)
* [AWS CLI Command Reference: lambda](https://awscli.amazonaws.com/v2/documentation/api/2.0.33/reference/lambda/index.html#cli-aws-lambda)
* [AWS CLI Command Reference](https://awscli.amazonaws.com/v2/documentation/api/2.0.33/index.html)