Gain Visibility into the Execution of Your AWS Lambda functions with AWS CloudTrail

by Kyle Somers | on 30 NOV 2017 | in [AWS CloudTrail\*](https://aws.amazon.com/blogs/mt/category/management-tools/aws-cloudtrail/), [Management Tools\*](https://aws.amazon.com/blogs/mt/category/management-tools/) | [Permalink](https://aws.amazon.com/blogs/mt/gain-visibility-into-the-execution-of-your-aws-lambda-functions-with-aws-cloudtrail/) | [Share](https://aws.amazon.com/blogs/mt/gain-visibility-into-the-execution-of-your-aws-lambda-functions-with-aws-cloudtrail/)

Today, we are happy to announce that AWS CloudTrail now supports the Lambda Invoke API as a new data event type with the launch of CloudTrail Lambda data events. Previously, AWS CloudTrail supported management events for AWS Lambda, which allowed you to capture when and by whom a function was created, modified, or deleted. With this new feature, you can now gain visibility into when and by whom an Invoke API call was made and which Lambda function was executed. CloudTrail can deliver these logs to Amazon S3 or Amazon CloudWatch Logs, and for near real-time processing, you can send them to Amazon CloudWatch Events to build event-driven security pipelines for the Lambda functions within your serverless applications.

**AWS CloudTrail Lambda data events**

Serverless computing with AWS Lambda is an increasingly popular business solution. Lambda allows you to run code in the cloud without managing or provisioning servers. You write your code, package it as a Lambda function, and hook it up with one or more event sources that trigger the function in response to events. All Lambda functions are executed using the Lambda Invoke API, whether you manually invoke your Lambda function via the AWS CLI, SDK, or set it up with any of the event source integrations. The AWS CloudTrail Lambda data events feature enables you to capture this invoke activity within your account to see who, when, and how your functions are being executed.

**How can I use AWS CloudTrail Lambda data events?**

AWS CloudTrail Lambda data events can be used to detect and automatically act on invocations of Lambda functions across your AWS account. For example, you can now:

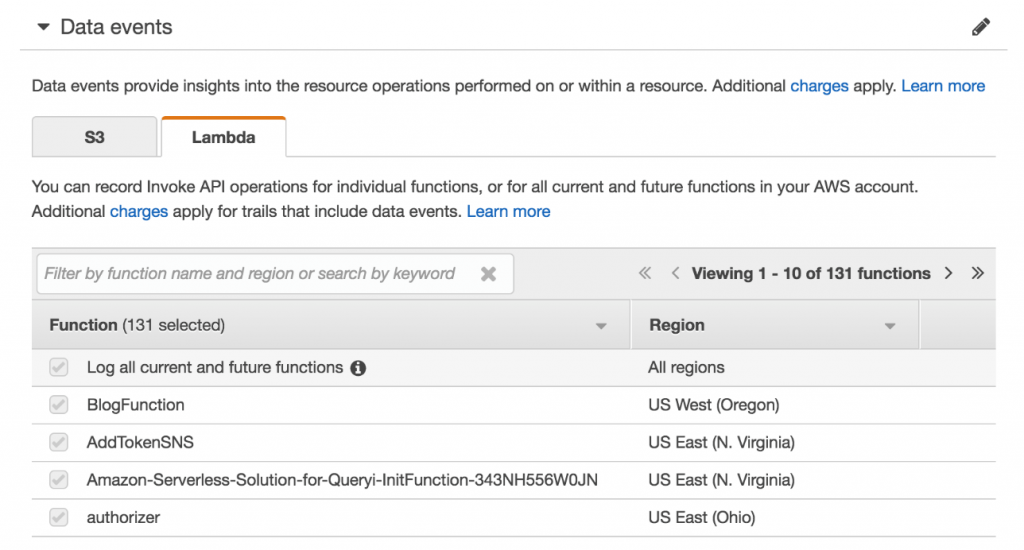
* Meet your IT auditing and compliance requirements. With CloudTrail data events, you can validate that your functions were invoked by permitted users, roles, and services. Customers with regulatory audit and compliance requirements can maintain the same level of visibility and auditability of Lambda function invokes as they do for other AWS service API invocations.
* Perform near real-time and ad-hoc security analysis. Lambda data events are delivered to your log delivery destination just like your existing CloudTrail logs, so it is easy to integrate this new log data type into your security analytics pipelines. You can analyze Lambda function invoke activity in near real time with Amazon CloudWatch Events and Amazon CloudWatch Logs.
* Better understand usage patterns across your AWS Lambda functions. You can perform billing analysis to understand the top user, role, or service callers of Lambda functions. You can also tie costs to specific users and groups for internal chargeback purposes or build reports on function usage across different teams.

Let’s take a look at how this integration works.

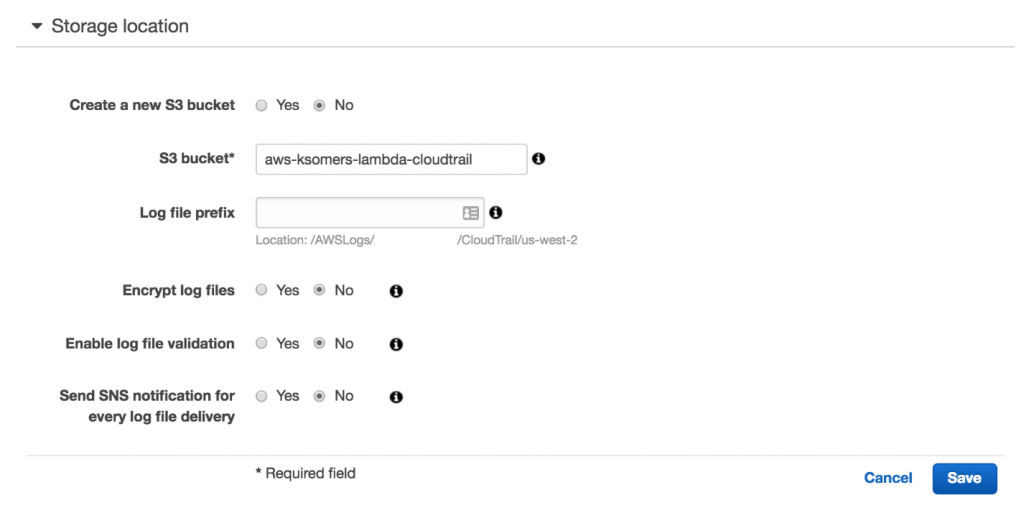
**Setting up Lambda Data Events**

Setting up AWS CloudTrail Lambda data events is easy. You can add Lambda data events to an existing trail that you have already created, or you can create a new separate trail just for Lambda data events – either option is suitable and will depend on how you plan to organize and manage your CloudTrail logs. In my example, I have created a separate trail called LambdaEvents and have configured it to*Log all current and future functions*. This feature is really handy and allows me to “set and forget” without needing to revisit my CloudTrail configuration each time a new function or version is published.

*You might be wondering what that S3 tab is for in the following image. Remember, AWS CloudTrail also supports data events for Amazon S3, which Jeff Barr discusses in his blog post*[*here*](https://aws.amazon.com/blogs/aws/cloudtrail-update-capture-and-process-amazon-s3-object-level-api-activity/)*.*



Once I have configured a new trail called LambdaEvents and enabled it across all AWS Regions for all existing and future functions, I now need to setup a storage location for CloudTrail to store the log files that will be generated. I have previously created my trail to deliver logs to a bucket dedicated to my Lambda data events (aws-ksomers-lambda-cloudtrail). I have left the rest of the settings at the default, but I could optionally enable a few additional things such as log file validation, encryption using AWS KMS, and notifications using Amazon SNS.



Now that CloudTrail Lambda data events is configured, let’s take a look at how data is presented to us with Lambda data events.

In my example, I invoked my BlogFunction from the AWS CLI with my “kyle” IAM user as a synchronous request with Lambda invoke type of “RequestResponse”. Remember that Lambda can be invoked either synchronously via a RequestResponse invocation type, asynchronously via an “Event” invocation type, or used in test mode with a “DryRun” invocation type. That invocation generated a CloudTrail log file in my Amazon S3 bucket. Let’s have a look at the event:

{

"eventVersion": "1.06",

"userIdentity": {

"type": "IAMUser",

"principalId": "AIDAIOR74VCJ2M3NB8U4M",

"arn": "arn:aws:iam::999999999999:user/kyle",

"accountId": "999999999999",

"accessKeyId": "AK2IU7DKE7U2KOI8CCBP",

"userName": "kyle"

},

"eventTime": "2017-11-29T08:47:45Z",

"eventSource": "lambda.amazonaws.com",

"eventName": "Invoke",

"awsRegion": "us-west-2",

"sourceIPAddress": "192.168.0.1",

"userAgent": "aws-cli/1.11.129 Python/2.7.8 Linux/3.1.56-0.6.839hdh3.x86\_64 botocore/1.5.92",

"requestParameters": {

"invocationType": "RequestResponse",

"functionName": "arn:aws:lambda:us-west-2:999999999999:function:BlogFunction:prod",

"clientContext": "ew0KICAiY29udGV4dGtleSIgOiAiY29udGV4dHZhbHVlIg0KfQ==",

"qualifier": "prod"

},

"responseElements": null,

"additionalEventData": {

"functionVersion": "arn:aws:lambda:us-west-2:999999999999:function:BlogFunction:4"

},

"requestID": "eaccb900-8f45-11e7-b60d-179cdf501g92",

"eventID": "0e205f1d-3929-4803-b887-0d2aca122148",

"readOnly": false,

"resources": [{

"accountId": "999999999999",

"type": "AWS::Lambda::Function",

"ARN": "arn:aws:lambda:us-west-2:999999999999:function:BlogFunction"

}],

"eventType": "AwsApiCall",

"managementEvent": false,

"recipientAccountId": "999999999999",

"sharedEventID": "6159da59-ad2f-4e04-9669-cf0a6b6b4827"

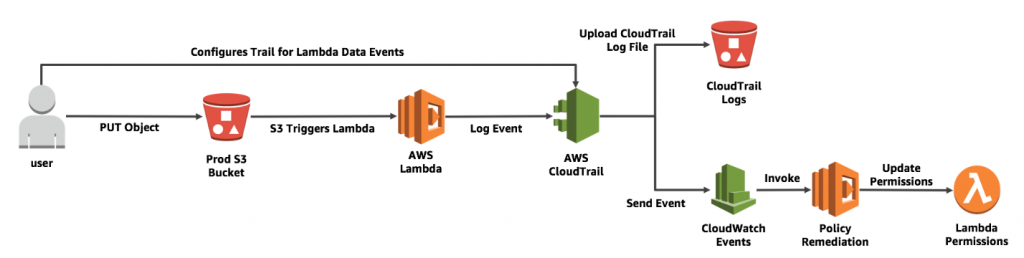
When a Lambda function is invoked, the function being invoked will show up in three locations within the CloudTrail Lambda data event. Here is a breakdown of what each one represents:

1. “requestParameters:functionName” – This field will match the input of the function that was called using the API (qualified version or unqualified). In my example, my function is deployed with a Lambda alias of “prod,” so this is displayed in the ARN.
2. “additionalEventData:functionVersion” – Here you will see the exact qualified function version that was invoked based on the input in the requestParameters of your Lambda invoke. Remember that a Lambda function alias points to an immutable version of a Lambda function. In this case, I invoked my function via the alias, but the ARN that was executed was version 4 of my function. This field will display the version ARN that was invoked.
3. “resources:ARN” – This will show the unqualified function ARN (some refer to it as the “base” ARN).

In the previous example, we looked at the contents of a Lambda Invoke API request of RequestResponse invocation type. For details on the contents of other invocation types such as Event Invocation Types, Cross-Account invocations, invocations from AWS services such as stream-based event sources, check out the documentation [here](http://docs.aws.amazon.com/lambda/latest/dg/API_Invoke.html).

**Lambda data events example**

One interesting way you can use this new feature is to monitor your Lambda functions for unauthorized invocations. Let’s take a look at an example of how you can use Lambda data events to check for unauthorized invocations with one of our classic Lambda examples, the S3 Event Notification Trigger for PUT Object Requests on a Bucket:\



In this architecture, there is an S3 bucket that is configured to invoke a Lambda function when new objects are created in the bucket (or any other event you would like). This function might update an Amazon DynamoDB table, send an email, or any other use case you may have!

With CloudTrail Lambda data events, you can validate that your function is only being invoked by Amazon S3. Every invocation of your function generates a CloudTrail log event which can be delivered to an S3 bucket for long-term retention. You can also configure a CloudWatch Events Rule and run real-time business logic on these log events with Lambda. In this case, if the “*userIdentity:invokedBy*” field in my CloudTrail log event isn’t “*s3.amazonaws.com*”, then my remediation Lambda can update my Lambda Function Policy to properly restrict invocation access to Amazon S3.

Pricing and availability

Once an AWS CloudTrail trail is set up, Amazon S3 charges apply based on your usage, since AWS CloudTrail delivers logs to an S3 bucket. Data events are recorded only for the functions you specify and are charged at $0.10 per 100,000 events, the same as S3 data events. See CloudTrail [pricing page](https://aws.amazon.com/cloudtrail/pricing/) for more details. CloudTrail Lambda data events are available in all AWS public Regions and AWS GovCloud (US).

**What’s coming**

Several AWS partners such as Sumo Logic, Rapid7, Saviynt, Alert Logic, and Evident.io are working on integrations with CloudTrail Lambda Data Events, so keep an eye out for updates!