

Streaming Logs from CloudWatch to Scalyr

In order to send logs to Scalyr in near-real-time, there are two steps involved:

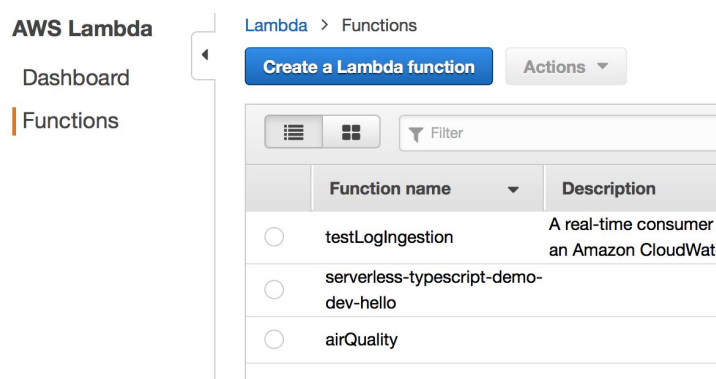
1. Create an AWS Lambda Function that receives CloudWatch log events, transforms them to a Scalyr-compatible message, and posts that to a Scalyr API.
2. Set up your CloudWatch Log Group(s) with a subscription filter that will stream CloudWatch log data to that lambda function.

If you haven't set up a "cloudwatch2scalyr" lambda function yet, both of these steps will be taken care of in the Lambda Function creation wizard as described below.

Create the Lambda Function

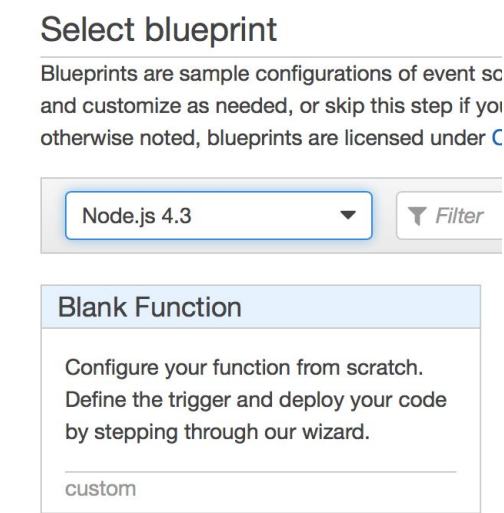
First, go to the Lambda Management Console in AWS. For us-east-1, the link is <https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions?display=list> – if you're in a different region, you can substitute that region for "us-east-1" in the URL above.

Once you navigate to that page, you should see something like this:



Click "Create a Lambda function" - this will start the wizard.

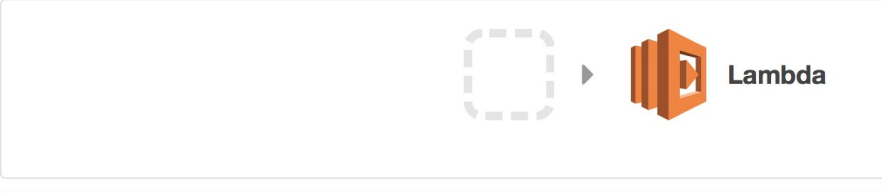
On the next page, you'll need to pick a "runtime" and a function "blueprint". Choose "Node.js 4.3" for the runtime and click on "Blank Function" for the blueprint, as shown below:



On the next page, you’ll configure a trigger to invoke the function. Click on the rounded dashed-line rectangle to do this and then select “CloudWatch Logs” from the dropdown that appears:


Configure triggers

You can choose to add a trigger that will invoke your function.




At this point, you’ll need to select a Log Group and provide a Filter Name (which can be anything, but it is required). Also, check the “Enable trigger” checkbox:

CloudWatch Logs



▶




Lambda

Log Group


/aws/lambda/airQuality

▼




Filter Name

airQualityTrigger



Filter Pattern



Lambda will add the necessary permissions for Amazon CloudWatch Logs to invoke your Lambda function for this log group. For more information, see [about the Lambda permissions model](#).

Enable trigger

☒



On the next screen, you’ll need to supply a function name and an optional description, as well as upload the lambda function code (which is in cloudwatch2scalyr.zip):

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name*

cloudwatch2scalyr

Description

Runtime*

Node.js 4.3

▼

Lambda function code


Provide the code for your function. Use the editor if your code does not require custom libraries (other than the aws-sdk). If you need custom libraries, you can upload your code and libraries as a .ZIP file. [Learn more](#) about deploying Lambda functions.

Code entry type

Upload a .ZIP file

▼

Function package*

 Upload

cloudwatch2scalyr.zip

For files larger than 10 MB, consider uploading via S3.

You'll need to specify your Scalyr "Write Logs" API key as an environment variable on this screen (scroll down a bit). In order to do this, you'll need to create a KMS encryption key if you don't have one in your account already (beyond the scope of this document). Select your encryption key and enter an environment variable:

- Enter "SCALYR_WRITE_LOGS_KEY" in the first column (no quotes)
- Provide your account's actual Scalyr "Write Logs" API Key in the second column.
- Click "Encrypt".

Once you've done this, you should see something like:

You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#). For storing sensitive information, we recommend encrypting values using KMS and the console's encryption helpers.

Enable encryption helpers ☒

Encryption key

scalyrApiKeyKey

Environment variables

SCALYR_WRITE_LOGS_KEY

Decrypt

Code

✕

Key

Value

Encrypt

Code

✕

There are a couple of additional environment variables you can pass to the code - USE_ADD_EVENTS_API and PARSER_NAME. (These should not be encrypted.) In general, USE_ADD_EVENTS_API should be false (and you don't actually need to specify it unless you want it to be true). PARSER_NAME is optional as well and refers to a specific custom parser defined using the Scalyr UI - if not specified, the default parser is used.

USE_ADD_EVENTS_API	false	✕
PARSER_NAME	airQualityParser	✕
Key	Value	✕

The last step on this page is to define a role for your function. In the next section of the page, choose "Create new role from template(s)" from the Role dropdown, give the role a name, and choose "KMS decryption permissions" from the Policy Templates dropdown:

Lambda function handler and role

Handler*

index.handler

i

Role*

Create new role from template(s)

i

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name*

cloudwatch2scalyrRole

i

Policy templates

KMS decryption permissi...

i

Now, click “Next”. On the next page, click “Create Function”. You should see a screen like:

Qualifiers

Test

Actions

Congratulations! Your Lambda function "cloudwatch2scalyr" has been successfully created and configured with /aws/lambda/airQuality as a trigger. You can now click on the "Test" button to input a test event and test your function.

This function contains external libraries. Uploading a new file will override these libraries.

Code

Configuration

Triggers

Monitoring

CloudWatch Logs: [/aws/lambda/airQuality](#)

arn:aws:logs:us-east-1:536275425565:log-group:/aws/lambda/airQuality:*

Filter name: **airQualityTrigger** Filter pattern:

Disable

Delete

+

 Add trigger

►

 View function policy

At this point, you’re done! Try it out by triggering some kind of action that will generate CloudWatch logs. This should cause events to be streamed to Scalyr (it may take a few seconds for them to show up):

ACTIONS

MAR 9 · 5:56 PM → MAR 9 · 6:25 PM

UPDATE

LOGS TO SEARCH

SERVER/HOST (*=wildcard)

/aws/lambda/airQuality

508 MATCHING EVENTS (30 sec/bar)

SHOW GRAPH

MAR 9 · 5:23:11 - 5:25:11 PM - 0 EVENTS

4:58 PM

5:00 PM

5:02 PM

5:04 PM

5:06 PM

5:08 PM

5:10 PM

5:12 PM

5:14 PM

5:16 PM

5:18 PM

5:20 PM

5:22 PM

5:24 PM

logfile

1

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

RSST: '-56',

parser

2

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

Stats: '{"v":"","v1":9.984249943321151,"v2":8.807782945786643,"v3":7.468288927911666,"v4":

serverHost

1

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

2017-03-10T00:17:28.673Z

f0e0f44d-0526-11e7-b5af-a5687fe59bd4

Lat 37.758239300

session

2

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

2017-03-10T00:17:28.673Z

f0e0f44d-0526-11e7-b5af-a5687fe59bd4

Lon -122.507950400

severity

1

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

2017-03-10T00:17:28.673Z

f0e0f44d-0526-11e7-b5af-a5687fe59bd4

PM2.5 9.53

threadId

1

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

2017-03-10T00:17:28.673Z

f0e0f44d-0526-11e7-b5af-a5687fe59bd4

Temp (F) 69

threadName

1

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

2017-03-10T00:17:28.673Z

f0e0f44d-0526-11e7-b5af-a5687fe59bd4

Humidity 44

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

2017-03-10T00:17:28.673Z

f0e0f44d-0526-11e7-b5af-a5687fe59bd4

Distance to Redwood Shore

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

2017-03-10T00:17:28.674Z

f0e0f44d-0526-11e7-b5af-a5687fe59bd4

{ ID: 691,

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

ParentID: null,

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

THINGSPEAK_PRIMARY_ID: '201598',

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

THINGSPEAK_PRIMARY_ID_READ_KEY: 'YBRFGQQLPBX6D0Z',

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

Label: 'Redwood Shores',

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

Lat: '37.543563',

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

Lon: '-122.246045',

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

PM2_5Value: '0.47',

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

State: null,

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

Type: 'PM55003+BME280+PUB',

17:17:46.788

cloudwatch-536275425565

/aws/lambda/airQuality

Hidden: 'false',

As you’ll notice, the log messages that show up in Scalyr are identical to those you would see in CloudWatch. Also note the serverHost field (“cloudwatch-536....”) and logfile field (“/aws/lambda/airQuality”). You can set up a custom parser in Scalyr based on the log file name (it should match your PARSE_NAME environment variable).

A bit about the addEvents transformation code (experimental)

This is an **experimental** feature that allows you to do some parsing in AWS Lambda itself by modifying the code in cloudwatch2scalyr.

The cloudwatch2scalyr.zip file contains one main file (index.js) as well as supporting Node.js libraries. If you wish to pre-parse interesting fields from the logs, you’ll probably be most interested in the *transformToAddEventsMessage* function, which is responsible for translating from CloudWatch-speak to Scalyr-speak. **Note: at this time, we don’t recommend using the addEvents API from Amazon Lambda (use uploadLogs, the default, instead).**

For more information about the format Scalyr expects, see this link:
<https://www.scalyr.com/help/api#addEvents>

Here's the function in question:

```
/**
 * Translates a CloudWatch message into a format appropriate for submitting to the Scalyr addEvents API
 * endpoint.
 *
 * @param cloudWatchMessage Incoming CloudWatch message.
 * @returns {Object} Outgoing Scalyr message.
 */
function transformToAddEventsMessage(cloudWatchMessage) {
  return {
    'token': decryptedScalyrApiKey,
    'session': cloudWatchMessage.logStream,
    'sessionInfo': {
      'serverHost': `cloudwatch-${cloudWatchMessage.owner}`, // TODO change this if you like
      'logfile': cloudWatchMessage.logGroup,
      'parser': parserName
    },
    'events': cloudWatchMessage.logEvents.map((cloudWatchEvent) => {
      return {
        'ts': `${cloudWatchEvent.timestamp}000000`,
        'type': 0,
        'sev': 3,
        'attrs': {
          // TODO make changes here if you want to parse in AWS Lambda before sending to Scalyr
          'message': cloudWatchEvent.message
        }
      }
    });
  })
};
}
```