**Cloudwatch logs ingestion to Splunk -**

There are two mode of log ingestion, pull & push.

Pull ingestion method :-

In this mode, we create a lambda, which has Cloudwatchlogs configured as a source event.

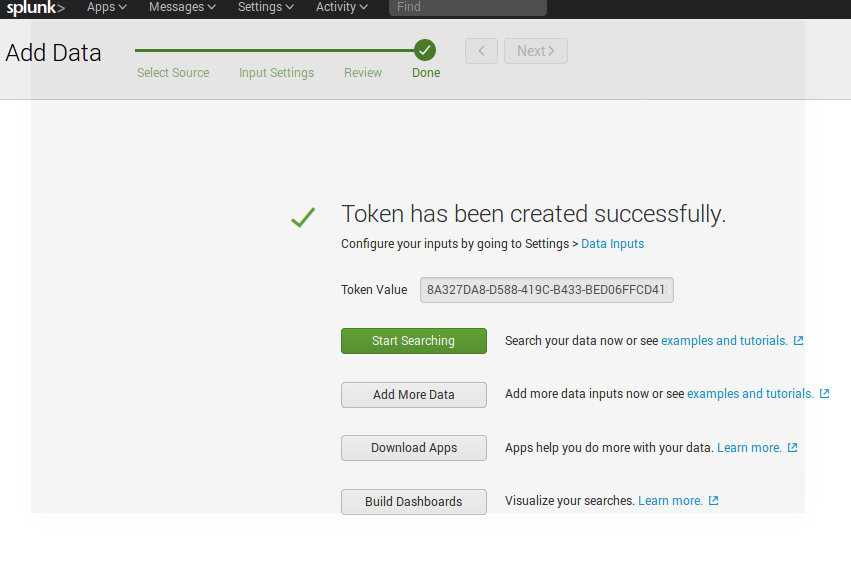
**Splunk Configurations ->**

We need to install AWS app in splunk if it is not already installed. To install this -

* Click on Splunk link on top left corner.
* Click on App button near splunk link
* Click on Browse more apps to find the AWS apps.
* Find AWS app and install it.
* Click on the Splunk button and go back to home page
* Here we can find the newly installed app “Splunk App AWS”

**Generating token: -**

* From the **Settings** menu, click **Data inputs**.
* On the Data inputs page, under **Local inputs**, click **HTTP Event Collector**.
* Click on “New Token” button
* Provide name for token
* Choose source type automatic
* Select main in index
* Click on review button
* And click submit button to create token.
* Now it will provide the token number, copy this token, we will use this in lambda configuration.

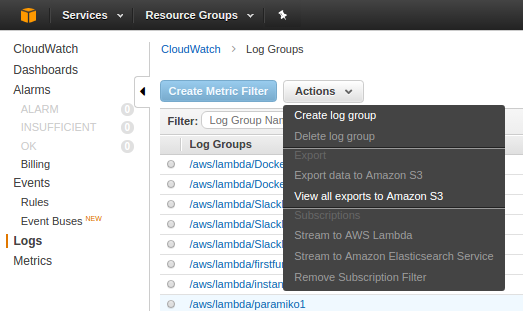


Here are the steps to setup a pipeline to stream cloudwatch logs to splunk-

* **LogGroup**

1. Create loggroup in cloudwatch -

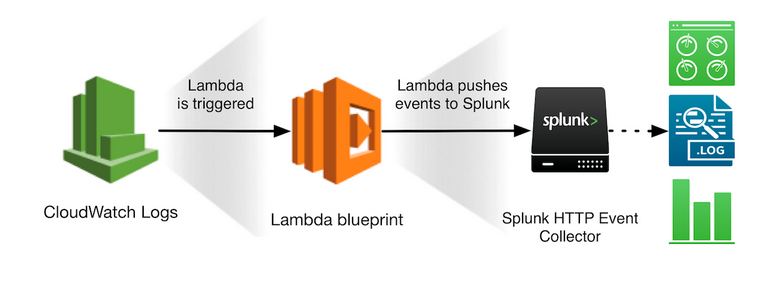
In cloudwatch service, a log-group should to be created and the applications should be configured to send logs to cloudwatch.

****

1. To forward application logs, the awslogs utility should be installed on server. It should be configured with a credentials (iam role or access keys) and should have access on cloudwatch group.

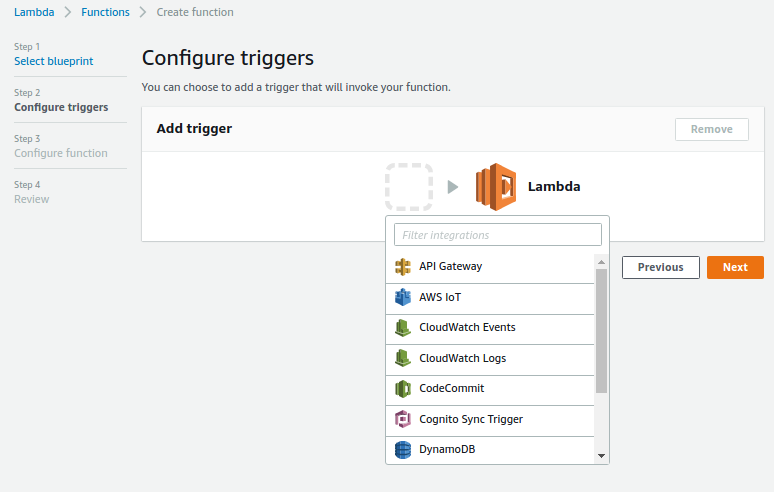
* **Lambda :-**

1. We are going to create a lambda with cloudwatch logs configured as source event. So, the lambda will be triggered by any event happening in cloudwatch logs.

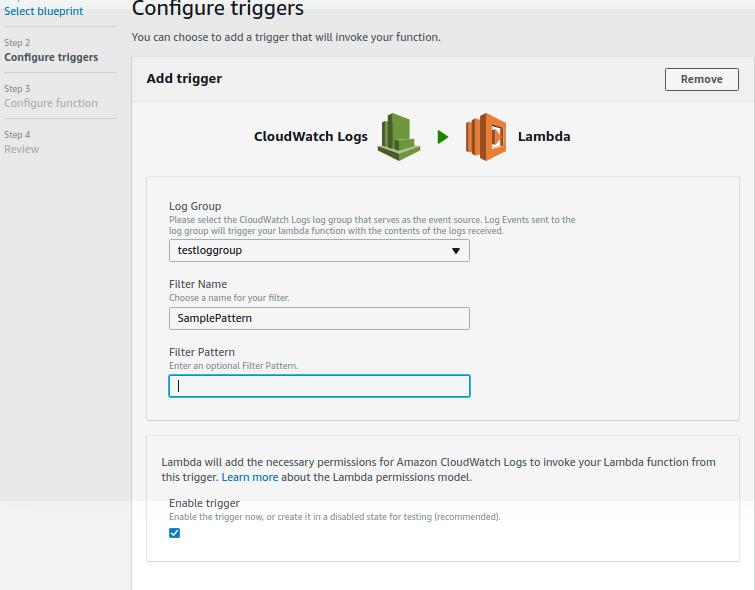


Here are the steps for creating the lambda -

1. Open Lambda service console
2. Click on ‘Create Function’ button to create lambda function.
3. We can start from available blueprints as well, but there is no python based template available for Splunk so we will create is from scratch. So, click on ‘**Author from scratch**’ button.
4. Now we are on ‘Configure Trigger’ step, here we need to select cloudwatch from dropdown that will invoke the lambda.



1. Select the Log-Group in which application logs are getting stored.
2. Provide any name for filter.
3. We’ll leave the filter pattern option blank as for we will send all the logs to cloudwatch for now.
4. Check mark on ‘Enable trigger’ option to allow Cloudwatch logs to invoke lambda function and click on next.



1. In Basic info box, provide name for lambda function, select Python3.6 as RunTime environment.
2. In next box, paste the code which will actually take care of sending logs to splunk.
3. Create two environment variables which are being referenced in lambda. Here are two variable names.
   1. SPLUNK\_HEC\_URL - Splunk HEC url <<https://splunk-servdr:8088/services/collector>>
   2. SPLUNK\_HEC\_TOKEN - Token generated in app - <D58FEA91-F4B7-4510-A7C1-93874C7D89A0>
4. Provide lambda handler name -
5. In Role selection option, we will choose the existing role which has basic execution permission or will create a new one if it is not there.
6. In Advanced settings, increase the timeout to 15sec.
7. Click on next button, review the configuration and click on create function.
8. When the loggroup receives any logs, then it triggers the lambda and lambda will send logs to splunk.
9. To check the logs in Splunk-
   1. Click on splunk button
   2. Click on aws app
   3. Now click on search tag in app windows and choose search
   4. Here type the below query to check logs -

**sourcetype="aws:cloudwatch"**

Here is the code for nodejs platform-

'use strict';

const loggerConfig = {

url: process.env.SPLUNK\_HEC\_URL || 'https://<HOST>:<PORT>/services/collector',

token: process.env.SPLUNK\_HEC\_TOKEN || '<TOKEN>',

};

const SplunkLogger = require('./lib/mysplunklogger');

const zlib = require('zlib');

const logger = new SplunkLogger(loggerConfig);

exports.handler = (event, context, callback) => {

// CloudWatch Logs data is base64 encoded so decode here

const payload = Buffer.from(event.awslogs.data, 'base64');

zlib.gunzip(payload, (err, result) => {

if (err) {

callback(err);

} else {

const parsed = JSON.parse(result.toString('ascii'));

console.log('Event Data:', JSON.stringify(parsed, null, 2));

let count = 0;

if (parsed.logEvents) {

parsed.logEvents.forEach((item) => {

// Send item JSON object (optional 'context' arg used to add Lambda metadata e.g. awsRequestId, functionName)

// Change "item.timestamp" below if time is specified in another field in the event

// Change to "logger.log(item.message, context)" if no time field is present in event

logger.logWithTime(item.timestamp, item.message, context);

count += 1;

});

}

// Send all the events in a single batch to Splunk

logger.flushAsync((error, response) => {

if (error) {

callback(error);

} else {

console.log(`Response from Splunk:\n${response}`);

console.log(`Successfully processed ${count} log event(s).`);

callback(null, count); // Return number of log events

}

});

}

});

};