Getting Data In – AWS Sources

Splunk Add-on for AWS 4.3+ and App 5.1+

Kamilo Amir

Table of Contents

[Collecting Data from AWS an Overview 4](#_Toc517812927)

[Purpose 4](#_Toc517812928)

[Prerequisites 4](#_Toc517812929)

[Module 1 – Syslog and Splunk Universal Forwarders (UF) 5](#_Toc517812930)

[Module 2 – Heavy Weight Forwarders (HWF) 6](#_Toc517812931)

[Module 3 – Serverless 7](#_Toc517812932)

[Lambda Functions 7](#_Toc517812933)

[Kinesis Firehose 9](#_Toc517812934)

[Module 4 – Modular Input vs. Serverless vs. Universal Forwarder 11](#_Toc517812935)

[Modular Input (Classic Inputs) 11](#_Toc517812936)

[Modular Input (SQS Based S3) 12](#_Toc517812937)

[Serverless 13](#_Toc517812938)

[Module 5 – Deploying Heavy Forwarders 15](#_Toc517812939)

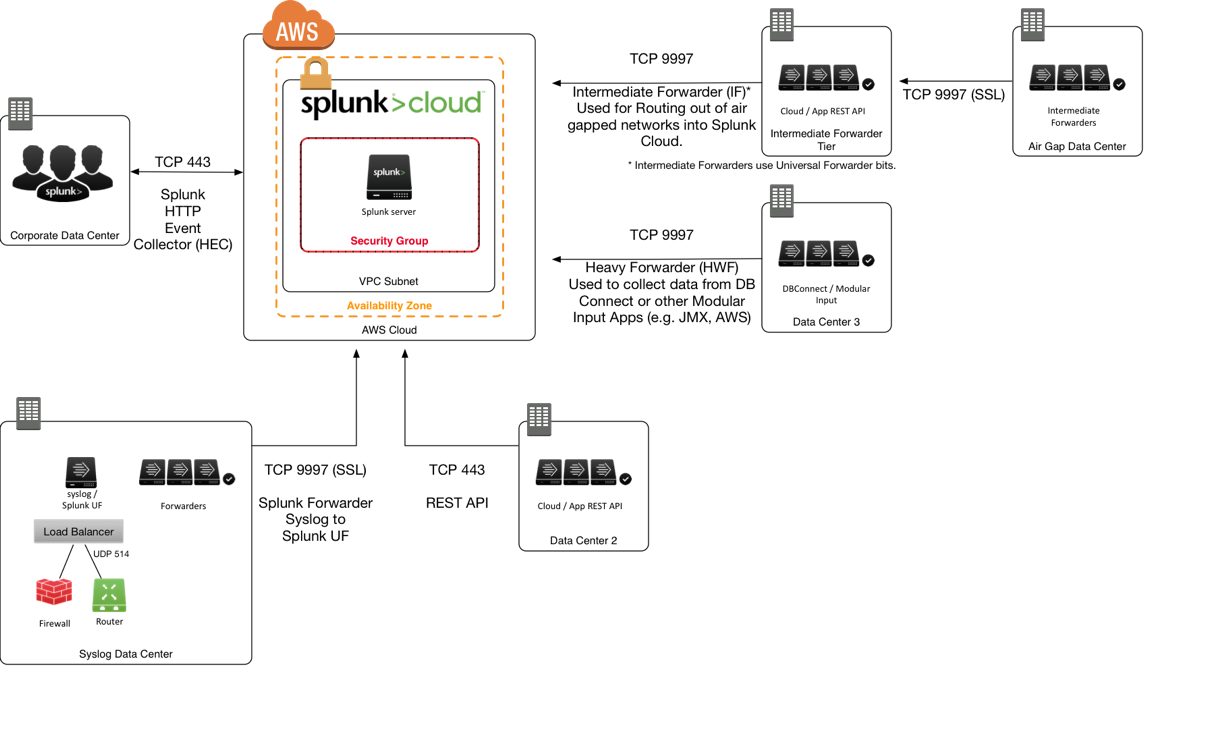
# Collecting Data from AWS an Overview

## Purpose

The Purpose of this document is to help customers determine the best way to collect data out of AWS and into their Splunk deployment:

## Prerequisites

This requires that you have you have full admin rights via the AWS console, admin rights on your Splunk deployment (Indexers, Search Heads and Forwarders) and some base-line experience in AWS and Splunk. For multiple-account deployments, you must have an IAM role with AssumRole enabled in the AWS sub-accounts.



## Module 1 – Syslog and Splunk Universal Forwarders (UF)

Collecting data from EC2 instances and services that can send data into a Syslog server should look to use the Splunk Universal Forwarder. For example, when dealing with network appliances, the standard method to send data into Splunk would be to configure a syslog server ([syslog-](https://syslog-ng.com/)ng or [rsyslog](https://syslog-ng.com/)) to receive data coming from these appliances then use a [Splunk Universal Forwarder](http://docs.splunk.com/Documentation/Forwarder/7.1.1/Forwarder/Abouttheuniversalforwarder) to send the data into your Splunk deployment.

Here is a blog post on how to setup your syslog environment to collect data and use the Splunk UF to send it out: <https://www.splunk.com/blog/2016/03/11/using-syslog-ng-with-splunk.html>

**\*\*NOTE\*\***

It is not advised to send raw UDP Syslog data into Splunk indexers, since UDP data is not load balanced across the indexing tier: <https://docs.splunk.com/Documentation/Splunk/7.1.1/Data/HowSplunkEnterprisehandlessyslogdata>



|  |  |
| --- | --- |
| Pros | Cons |
| Scalable, uses Splunk UF’s for Load balancing | Legacy Architecture requiring knowledge of syslog |
| Cloud Agnostic, will work in any cloud provider | Requires additional servers and updates to software (Splunk Universal Forwarders) |
| Data is encrypted and compressed | Single point of failure if syslog server goes down |

## Module 2 – Heavy Weight Forwarders (HWF)

There are some data sources in AWS which will require that the data get pulled out then forwarded into Splunk. AWS Description (Metadata), S3 Logs, CloudWatch Metrics (as of Splunk Add-on for AWS v.4.4) require that data be pulled from the AWS API.



|  |  |
| --- | --- |
| Pros | Cons |
| Scalable, uses Splunk UF’s for Load balancing | Single point of failure if HWF goes down |
| Cloud Agnostic, will work in any cloud provider | Requires servers to be setup and configured with Splunk to collect data |
| Data can be masked / obfuscated before being indexed | API throttling from AWS if API is called too many times |

## Module 3 – Serverless

Splunk can collect data without the need of a heavy forwarder and can send the data directly into the indexing tier using the [Splunk HTTP Event Collector](http://dev.splunk.com/view/event-collector/SP-CAAAE6M) (HEC).

### Lambda Functions

The first push architecture uses [AWS Lambda Functions](https://aws.amazon.com/lambda/) to feed data from AWS into Splunk HEC. This solution allows you push events from AWS using a trigger from the AWS services. Here is the blog post on how to setup this method: <https://www.splunk.com/blog/2017/02/03/how-to-easily-stream-aws-cloudwatch-logs-to-splunk.html> . This method is the recommended approach for high volume data, but does not require event acknowledgement in Splunk. If there is a failure between Splunk and AWS, then events may be dropped.



|  |  |
| --- | --- |
| Pros | Cons |
| Can send high volumes of data directly into Splunk | Events are not acknowledged in Splunk |
| No hardware is required | Requires some level of understanding of Node.js |
| Rich library of Blueprints to create inputs | Cannot handle some custom data types (non-AWS native events) |

### Kinesis Firehose

The second push architecture leverages the [AWS Kinesis Firehose](https://aws.amazon.com/kinesis/data-firehose/) and pushes data into Splunk’s HEC. The main difference is that this solution adds a level of resilience and acknowledges that Splunk has received the event. Here are the Splunk documents on how to setup this solution: <https://docs.splunk.com/Documentation/AddOns/released/Firehose/ConfigureFirehose>



|  |  |
| --- | --- |
| Pros | Cons |
| Can send high volumes of data directly into Splunk | Requires understanding on setting up AWS Kinesis via CLI |
| No hardware is required | Cost |
| Validates that events are in Splunk otherwise sends data to S3 for future collection | Requires public facing VIP and third party SSL Certificates |

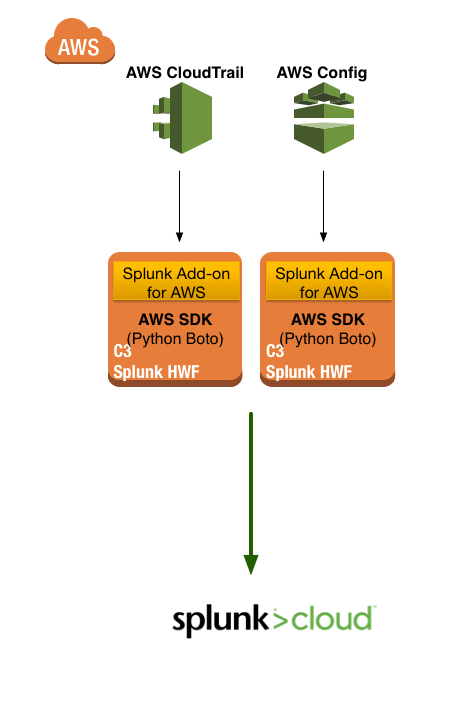
## Module 4 – Modular Input vs. Serverless vs. Universal Forwarder

Customers always ask, “I see there are so many ways to get data into Splunk, but which one should I use?” Like any good engineer, the answer I always give is “It depends.”

### Modular Input (Classic Inputs)

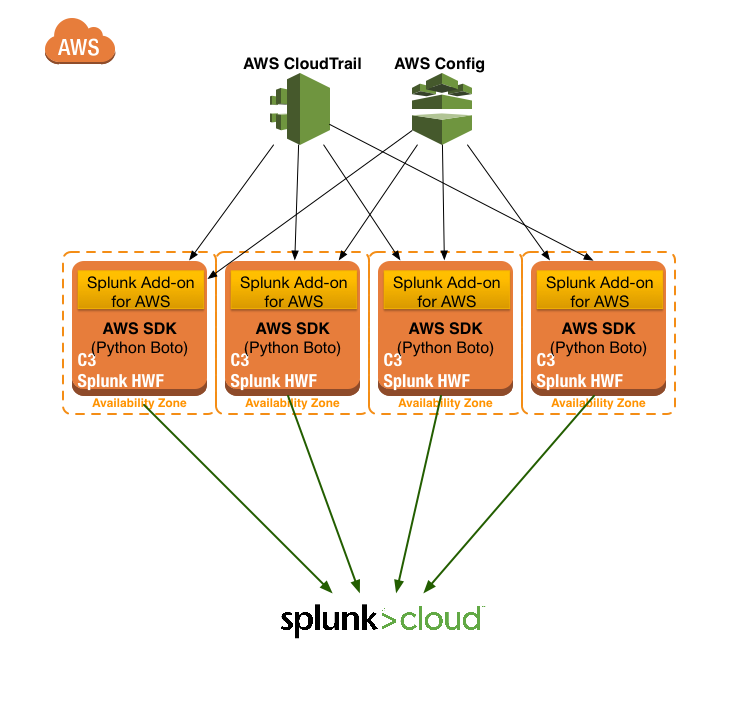
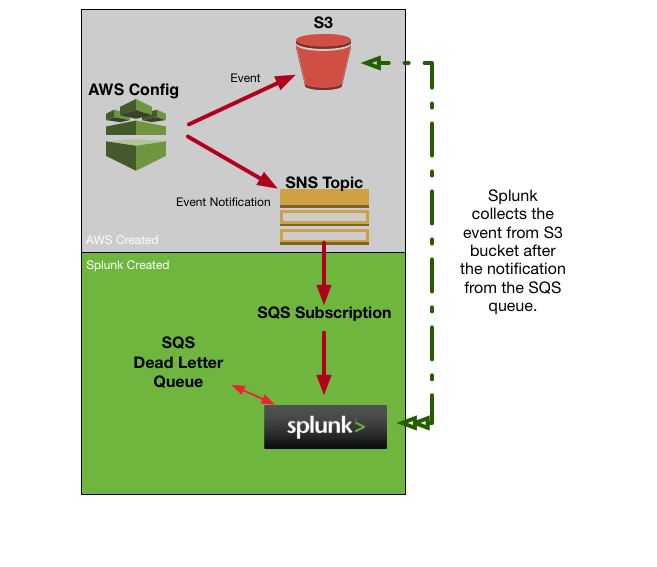
Collecting data using the default API leverages a modular input. This is simply a python script that calls the API from AWS and collects the data. Each input has one, and there are four services which only use the Modular Input to collect data, they are : **AWS Description, AWS Billing, AWS CloudWatch (metrics)** and **S3** related inputs (e.g. **Access, ELB or Cloudfront logs**). This means that you will need a Heavy Forwarder (HWF) to collect these inputs. (See the next section on how to deploy a bank of HWF in AWS.)

I would recommend this input for the inputs mentioned above and for small deployments that are collecting less than 20GB/day from AWS. A single HWF can process multiple AWS accounts with their associated inputs. The [reference hardware](http://docs.splunk.com/Documentation/Splunk/7.1.1/Capacity/Referencehardware) for an indexer would be where I would start for spec’ing out a HWF. Remember, you will NOT need the high performance disk on this HWF since it will not be doing any indexing, just collecting data and forwarding it to your Splunk indexer(s). A good rule of thumb would be 20 AWS accounts per HWF.



### Modular Input (SQS Based S3)

As of Splunk Add-on for AWS version 4.4, the best way to collect Config, CloudTrail and other S3 based inputs is to use the SQS Based S3 approach. This solution allows customers to create a bank of Heavy Forwarders that can collect the data from AWS and not become a single point of success ~~failure~~. The reason is that the state of the event is not kept on the HWF, but rather in the Dead Letter Queue (DLQ) on the SQS. If an event is not in the DLQ, then the HWF will proceed to collect it from the S3 bucket. (More on the topology in the next section.)

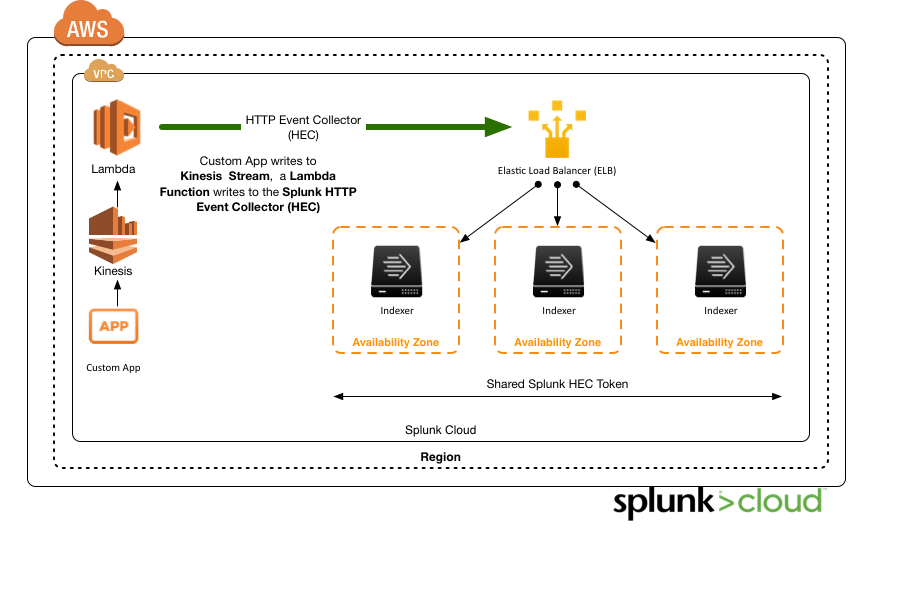
 

### Serverless

There are some data sources from AWS which should not be collected using the Modular input, specifically CloudWatch Logs (e.g. VPCFlow Logs). These logs can quickly cause Amazon to rate limit a customer for accessing the API too frequently, and stop collecting data via the API. The two main approaches here are using Lambda Functions or Kinesis Firehose.

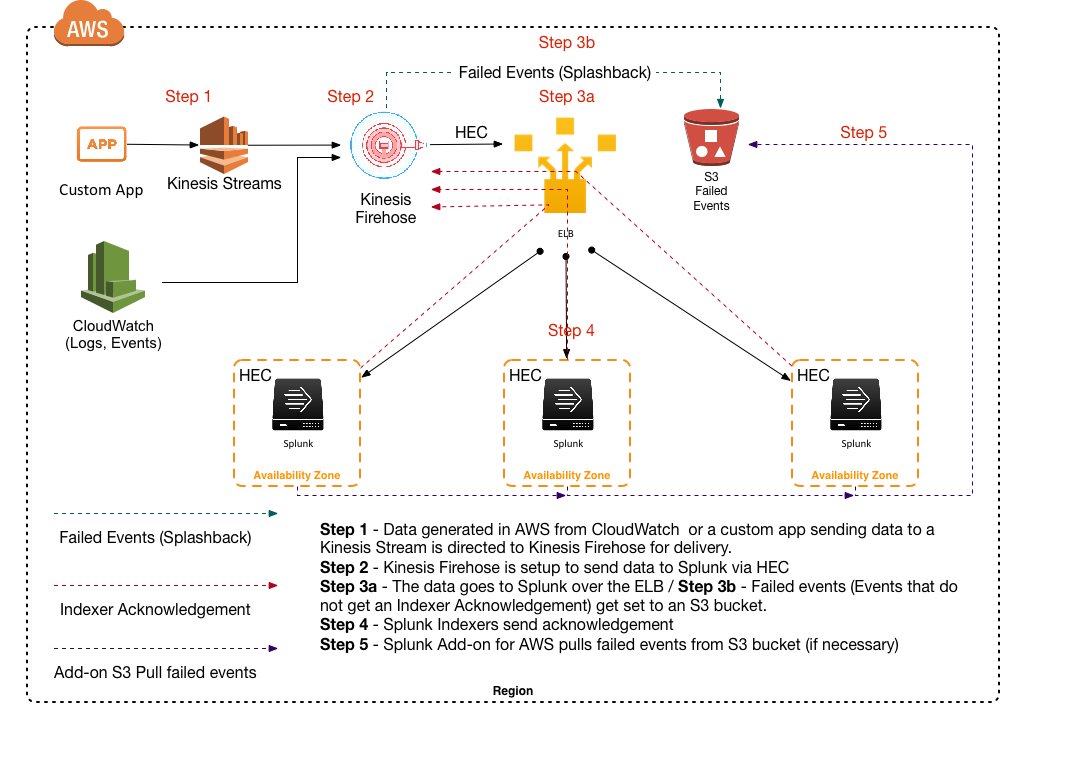
**Lambda Functions** should be used if the data that is being collected is high volume and does not require verification that the event has been indexed in Splunk. Typically, access logs, VPCFlow Logs, Elastic/Application Load Balancer logs can be sent into Splunk using Lambda functions. If there is a failure between Splunk and AWS, then events might not make it into Splunk. Since these events are still being collected in an S3 bucket or a CloudWatch Log Group, it is possible to setup a HWF to pull these events at a later time to recover any lost events.

The two main knocks against Lambda functions are 1) Potential Costs 2) Potential data loss with a third being that they do take some understanding of AWS and Node.js to properly setup. Thankfully the Splunk blueprints for Lambda functions are very extensive and can help non-Node.js developers start using the functions rather quickly.



**Kinesis Firehose** should be used if the events are being sent into Splunk via a **Kinesis Stream**, **CloudTrail**, or **CloudWatch** **Logs** / **Events** and require that Splunk acknowledges it has indexed all the events. While setting up Kinesis Firehose (IAM Roles, permissions, S3 buckets etc.) can be a little difficult, once they are done getting the data into Splunk is very straight forward. The customer will only need the Splunk HEC URL and the Token.

This method of collecting data is great for high volume, high value data and requires ease of use to setup. Generally, customers that are moving large volumes of data through Kinesis will leverage this solution to get their data into Splunk.



## Module 5 – Deploying Heavy Forwarders

The issue with this modular input is that it becomes a single point of success ~~failure~~ which means that if the HWF goes down, the input will not collect data. The best way to guard against this is to setup the HWF in an auto scale group and put the configurations on an EBS volume.

First, make sure to distribute your Heavy Forwarders across multiple Availability Zones (AZ). Next mount the volume with the configurations on an EBS (Elastic Block Storage) volume. Finally, for additional resilience, you can place these instances in an [Auto Scale Group](https://aws.amazon.com/autoscaling/). This way if a HWF fails, it can be brought back up quickly.

