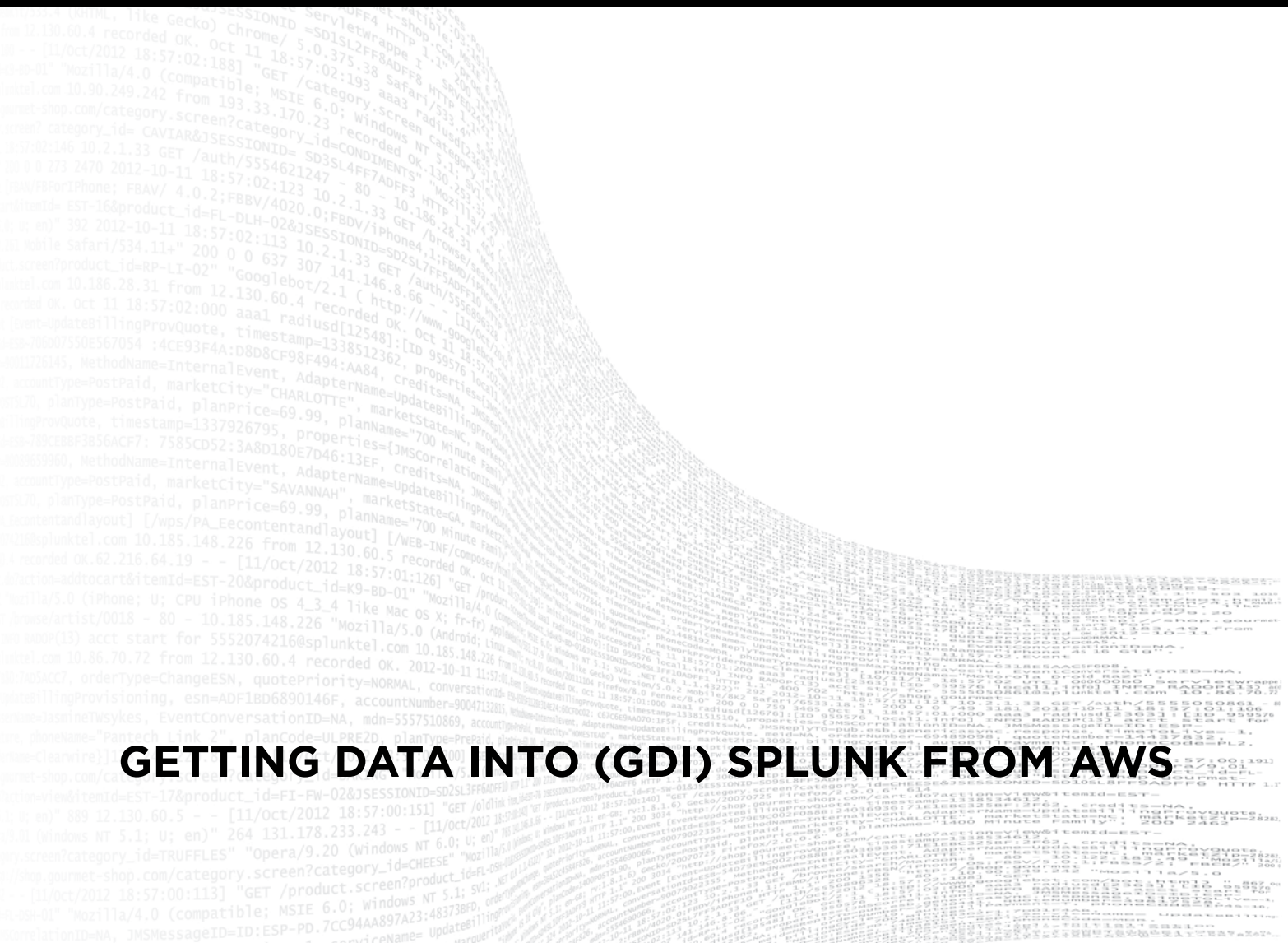
splunk>

WHITE PAPER

GETTING DATA INTO (GDI) SPLUNK FROM AWS



Executive Summary

In a cloud-first world, it is important to be able to access your data no matter the environment. The efficiencies of the cloud should not cost organizations ease of access or control. This white paper will help users determine the best way to collect data out of AWS and into their Splunk deployment via forwarders, heavy forwarders and serverless methods — important capabilities for scaling with the cloud.

Introduction — Scaling for the Cloud

Scaling data collection from AWS into Splunk can be difficult and often requires an understanding of why certain data sources require forwarders while others can be accessed serverlessly. There's complexity in enabling and configuring AWS services (e.g. AWS CloudTrail) and transports (e.g. Amazon Kinesis Data Firehose, SNS, SQS). Then there's managing instances while reducing impact on cost, the issues of security and privacy, management and more.

Fortunately, Splunk can address each of these possible roadblocks. Splunk can collect data using universal/heavy forwarders and send it directly from the endpoints into Splunk. For instance, API data sources (e.g. CloudWatch Metrics, Description and AWS Billing) require a heavy forwarder to pull and forward the data — Splunk can do that. Additionally, high volume data sources that have the ability to send data via an HTTP endpoint can leverage the Splunk HTTP Event Collector (HEC) to collect data serverlessly. Data sources like VPC flow logs, CloudTrail and other CloudWatch Logs can leverage either Lambda Functions or Kinesis Data Firehose to send data into Splunk. Pick the method that best suits you — all encrypt the data on transit and can scale with your deployment.

This white paper will show you how to effectively collect data from AWS instances for Splunk deployments through various methods suited for your particular context and requirements, including:

- Syslog and Splunk Universal Forwarders (UF)
- Serverless with Splunk HTTP Event Collector (HEC)
- Heavy Forwarders (HF)

Prerequisites

To begin, we'll need to address a few prerequisites. The effective use of AWS and Splunk requires that you have baseline experience with AWS and Splunk as well as full admin rights via the AWS console and Splunk deployment (i.e. indexers, search heads and forwarders). In the case of multiple-account deployments, you have an identity and access management (IAM) role with AssumeRole enabled in AWS sub-accounts.

Syslog and Splunk Universal Forwarders (UF)

Splunk Universal Forwarder is perfectly suited for collecting data from EC2 instances and services that can send data into a syslog server. For instance, when dealing with network appliances, the standard method to send data into Splunk would be to configure a syslog server (e.g. syslog-ng or rsyslog) to receive data from those appliances and use Splunk Universal Forwarder to send the data into your Splunk deployment. A dedicated syslog server means no loss of syslogs during downtime and easier distribution of syslog data across multiple indexers. Here's how to set it up. Something to note — it is not advised to send raw UDP Syslog data to Splunk indexers since it is not load balanced across the indexing tier.

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

Serverless with Splunk HTTP Event Collector (HEC)

Heavy forwarders are not always necessary for data collection. You can send data directly into the indexing tier using Splunk HTTP Event Collector using two main methods — AWS Lambda Functions and Amazon Kinesis Data Firehose.

AWS Lambda Functions

Use AWS Lambda Functions to feed data from AWS into Splunk HEC. AWS Lambda allows you to push events from AWS using an AWS services trigger. Moreover, you can run code with provisioning or managing servers—all with continuous scaling. Combining AWS Lambda with Splunk HEC is recommended for high volume data and does not require event acknowledgment in Splunk. It is important to be mindful that if there is a failure between Splunk and AWS, events may be dropped. Set it up today.

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)

Amazon Kinesis Data Firehose

Another push architecture leverages Amazon Kinesis Data Firehose to push data into Splunk HEC easily and reliably for near real-time analytics. The major difference from AWS Lambda being that Amazon Kinesis Data Firehose adds a level of resilience and acknowledges when Splunk receives events. Amazon Kinesis Data Firehose will send the events to an Amazon S3 bucket if it does not receive an acknowledgment from Splunk. Splunk has extensive documentation on how to set up this solution.

Pros	Cons
Can send high volumes of data directly into Splunk	Requires understanding of 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

Modular Input vs. Serverless

Considering the different options, which should you use to get data into Splunk? Well, it depends on the data source, the volume, and even hardware requirements. Here are some of the most commonly employed methods.

MODULAR INPUT

- **Classic Inputs**
  The use of the default API leverages a modular input — a Python script that calls the API from AWS and collects the data. Each input has its script, with four services that explicitly use Modular Input to collect data, including AWS Description (modular input sourcetype), AWS Billing, Amazon CloudWatch (metrics) and Amazon S3 related inputs (e.g. Access, ELB or CloudFront logs).

  Modular Input is recommended for the above, and for small deployments collecting less than 20GB per day from AWS. A single HF can process multiple AWS accounts and their associated inputs. You can start scoping where to start with HF with the indexer reference hardware. Remember, you will not need the high-performance disk on this HF since it won't 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 HF.

- **Amazon Simple Queue Services (SQS) Based Amazon S3**
  With the Splunk Add-on for AWS version 4.4, the best way to collect AWS Config, AWS CloudTrail and other Amazon S3 based inputs is to use the SQS-based Amazon S3 approach. Avoid bottlenecks as this solution creates a bank of HFs that can collect the data from AWS. Event states are not kept on the HFs, but rather in the Dead Letter Queue (DLQ) on the SQS. So if an event is not in the DLQ, then the HFs will process to collect it from the Amazon S3 bucket.

- **Heavy Forwarders (HF)**
  Some data sources in AWS require that data get pulled out and forwarded to Splunk. AWS Description (Metadata), Amazon S3 Logs and CloudWatch Metrics as of Splunk Add-on for AWS v.4.4 require that data be pulled from the AWS API.

  Something to be aware of with HFs is that they can become a single point of failure. The best way to guard against this is to set up the HF in an autoscale group and put the configurations on an Amazon Elastic Block Storage (EBS) volume.

  First, make sure to distribute your HFs across multiple Availability Zones (AZ). Next, mount the volume with the configuration on an Amazon EBS volume. Finally, for additional resilience, you can place these instances in an Auto Scale Group. This way if a HF fails, it can be brought back up quickly.

SERVERLESS

There are some data sources from AWS that should not be collected using Modular input, specifically Amazon CloudWatch Logs (e.g. Amazon VPC flow logs). These logs can cause AWS to rate limit a customer for accessing the API too frequently and stop data collection via API. We can solve this with AWS Lambda functions and Amazon Kinesis Data Firehose. There are other AWS data sources that are easier to collect by serverless push methods over modular inputs pull, Amazon GuardDuty, Amazon Macie, and the sources in Amazon CloudWatch Events.

- **AWS Lambda Functions** should be used for high-volume, event-based data collection that does not require verification that Splunk indexed the event. Typically, access logs, VPC flow logs Amazon Elastic Load Balance and (ELB) Amazon Application Load Balancer (ALB) logs can be sent into Splunk using AWS Lambda functions. If there is a failure between Splunk and AWS, then events might not make it into Splunk. Since these events can still be collected in an Amazon S3 bucket or an Amazon CloudWatch group, it is possible to set up a HF to pull these events at a later time to recover any lost events.

- **Amazon Kinesis Data Firehose** should be used if events are being sent into Splunk via Amazon Kinesis Data Streams, Amazon CloudTrail or Amazon CloudWatch Logs or Events that require acknowledgment from Splunk that events are indexed. Setup can be complicated but once it's running, getting data into Splunk only needs the Splunk HEC URL and the Token. Splunk has developed an open source project **TRUMPET** to simplify the setup of serviceless GDI. Trumpet will enable AWS services, build the transports and Amazon Kinesis Data Firehose to send the data into Splunk.

Conclusion

Remember, access and control does not have to be sacrificed for the cloud, in fact, you can have that and the efficiencies it offers. Consider the above methods as you build out the architecture to get data from your AWS instance into Splunk.

Learn more about how **Splunk solutions** can help you gain visibility into your AWS environment, or get started today with the **Splunk Add-on for Amazon Kinesis Firehose** on Splunkbase.

WP-Splunk-Getting-Data-into-(GDI)-Splunk-From-AWS-101