# Use Amazon Elasticsearch Service to Log and Monitor (Almost) Everything

*December 2016*

# Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

# Contents

# Abstract

Amazon Elasticsearch Service (Amazon ES) makes it easy to deploy, operate, and scale Elasticsearch for log analytics, full text search, application monitoring, and many more use cases. It is a fully managed service that delivers the easy-to-use APIs and real-time capabilities of Elasticsearch along with the availability, scalability, and security required by production workloads.

Our customers have told us that Amazon ES is an excellent service for logging and monitoring because it is fully managed by Amazon Web Services (AWS) and offers compelling value relative to its cost of operation. In this whitepaper, we provide best practices for feeding log data into Elasticsearch and visualizing it with Kibana using a serverless, inbound log management approach.

We show you how to use Amazon CloudWatch Logs and the Amazon CloudWatch Logs agent to manage inbound logs. You can use this approach instead of the more traditional ELK Stack (Elasticsearch-Logstash-Kibana) approach. We identify the strengths and weaknesses of the CloudWatch approach while providing tips and techniques for easy setup and management of the solution.

To get the most out of reading this whitepaper, it's helpful to be familiar with AWS Lambda functions, Amazon Simple Storage Service (S3), and AWS Identity and Access Management (IAM).

# Introduction

AWS cloud implementations differ significantly from on-premises infrastructure. New log sources, the volume of logs, and the dynamic nature of the cloud introduce new logging and monitoring challenges. AWS provides a range of services that help you to meet those challenges. For example, AWS CloudTrail captures all API calls made in an AWS account, Amazon Virtual Private Cloud (Amazon VPC) Flow Logs capture network traffic inside an Amazon VPC, and both containers and EC2 instances can come and go in an elastic fashion in response to Auto Scaling events. Many of these log types have no direct analog in the on-premises data center world.

This whitepaper explains how to use Amazon Elasticsearch Service (Amazon ES) to ingest, index, analyze, and visualize logs produced by AWS services and your applications without increasing the burden of managing or monitoring these systems. Elasticsearch and its dashboard extension, called Kibana,[1] are popular open-source tools because they are simple to use and provide a quick time to value. Additionally, the tools are well supported by a company called Elastic,[2] as well as by an active open-source community.

With the managed Amazon ES service, AWS reduces the effort required to set up and configure a search domain by creating and managing a multi-node Elasticsearch cluster in an automated fashion, replacing failed nodes as needed. The domain is the searchable interface for Amazon ES, and the cluster is the collection of managed compute nodes needed to power the system.

You can easily scale your cluster with a single API call, and configure it to meet your performance requirements by selecting from a range of instance types and storage options, including solid state drive (SSD)-backed EBS volumes. Amazon ES provides high availability using zone awareness, which replicates data between two Availability Zones. By taking advantage of Amazon ES, you can concentrate on getting value from the data that is indexed by your Elasticsearch cluster and not on managing the cluster itself.

You can use AWS tools, settings, and agents to push data into Amazon ES. Then, you can configure Kibana dashboards to make it easy to understand interesting correlations across multiple types of AWS services and application logs. Examples include VPC networking logs, application and system logs, and AWS API calls. Once the data is indexed, you can access it via an extensible, simple, and coherent API using a simple query domain specific language (DSL) without worrying about traditional relational

database concepts such as tables, columns, or SQL statements. As is common with full-text indexing, you can retrieve results based on the closeness of a match to your query. This can be very useful when working with log data to understand and correlate a key problem or failure.

In this whitepaper, we show you how to implement a serverless approach to using Amazon ES and Kibana, and we contrast that approach with other typical configurations for the use of Amazon ES. The advantage of the serverless approach is that AWS manages the underlying compute and networking resources you need, and will manage the durability and reliability of the underlying system. After reading this whitepaper, you should be able to quickly configure Amazon ES and establish log input methods for most, if not all, of the systems in your AWS environment. You also should be able to configure and launch Kibana dashboards that provide visualization for your data, and make API calls to the Elasticsearch cluster.

# What Is Elasticsearch, Exactly?

Amazon Elasticsearch Service (Amazon ES) is a managed service that makes it easy to create a domain and deploy, operate, and scale Elasticsearch clusters in the AWS Cloud. An Amazon ES domain is a service wrapper around an Elasticsearch cluster. A domain contains the engine instances (nodes) that process Amazon ES requests, the indexed data that you want to search, snapshots of the domain, access policies, and metadata.

The first public release of the Elasticsearch engine was issued in early 2010, and since then the Elasticsearch project has become one of the most popular open-source projects on GitHub. Based on Apache Lucene internally for indexing and search, Elasticsearch converts data such as logs that you supply into a JSON-like document structure using key-value pairs to identify the strings and values that are present in the data. In Elasticsearch, a document is roughly analogous to a row in a database, and it has the following characteristics:

- Has a **unique ID**
- Is a collection of **fields** (similar to a column in a database table)
- Is organized by **type** (similar to a table in a database)

In the following example of a document, the document ID is **34171** and the type is **employee**. The fields include **first name**, **last name**, and so on.


**Figure 1: Example of an Elasticsearch document**

Elasticsearch supports a RESTful web services interface. You can use HTTP PUT and GET commands to add data to the Elasticsearch index, which is a logical collection of documents that can be split into shards. Most users and developers use command line tools such as cURL to test these capabilities and execute simple queries, and then develop their applications in the language of their choice.

The following illustration shows an Amazon ES domain that has an index with two shards, Shard A and Shard B.

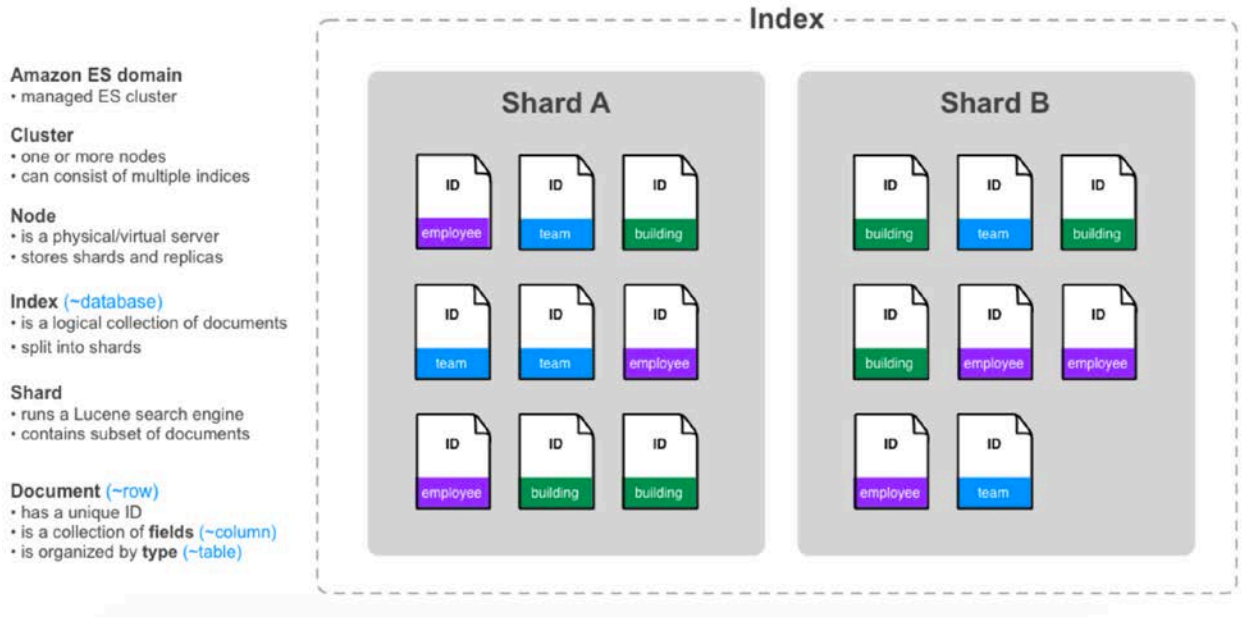


**Figure 2: Elasticsearch terminology**

As mentioned earlier, you can think of an Amazon ES domain as a service wrapper around an Elasticsearch cluster, and the logical API entry point to put and get data from the system. A cluster is a logical grouping of one or more nodes and indices. An index is a logical grouping of documents, each of which has a unique ID. Documents are simply groupings of fields that are organized by type. An index can be further divided into shards. The Lucene search engine in Elasticsearch executes on shards that contain a subset of all documents that are managed by a given cluster.

Conventional relational database systems aren't typically designed to organize unstructured raw data that exists outside a traditional database in the same manner as Elasticsearch. Log data varies from semi-structured (such as web logs) to unstructured (such as application and system logs and related error and informational messages). Elasticsearch does not require a schema for your data and is often orders of magnitude faster than a relational database system when used to organize and search this type of data.

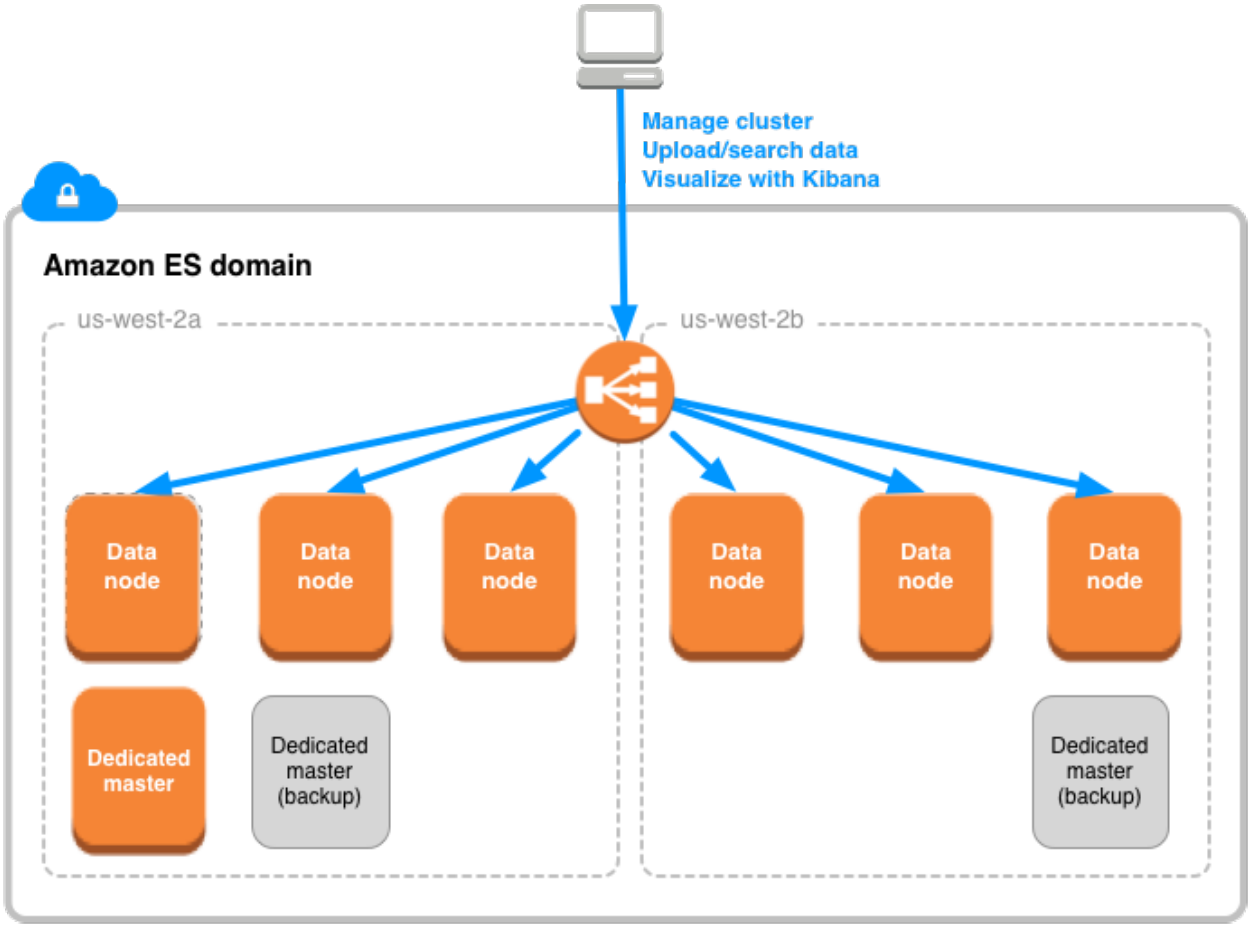


**Figure 3: Amazon ES architecture**

Because Elasticsearch does not store data in a normalized fashion, clusters can grow to 10s or 1,000s of servers and petabytes of data. Searches remain speedy because Elasticsearch stores documents that it creates in close proximity to the metadata that you search via the full-text index.

When you have a large, distributed system running on AWS, there is business value in logging everything. Elasticsearch helps you get as close to this ideal as possible by capturing logs on almost everything, and making the logs easily accessible.

# How Is Elasticsearch Used?

Many users initially start with Elasticsearch for consumption of logs (~50% of initial use cases involve logs), then eventually broaden their usage to include other searchable data. Elasticsearch is also frequently used for marketing and clickstream analytics. Some of the best examples of analytic usage come from the online retailing world, where

several major retailers use Elasticsearch. One example of how they use the data is to follow the clickstream created by their order pipeline to understand buyer behavior and make recommendations either before or after the sale.

Many log applications that target Elasticsearch also start with the use of the Logstash agent and forwarder to transform and enrich their log data (such as geographic information and reformatting) prior to sending to their cluster.[3] The Beats agent is also popular because of its relatively speedy performance and is often used for forwarding log data from instances to an Elasticsearch cluster.[4]

Elasticsearch should not be considered a real-time system, although it can produce analytic value in a relatively short period of time given the performance of its indexing engine. The default index refresh rate is set at one second, but is configurable given the size of your cluster and the rate of log ingestion. Because Elasticsearch and Kibana are open-source software, it is not unusual to see enterprise customers providing Kibana web access across a large subset of desktops in departments that need to understand their customers better.

As a full-text indexing system, you can use Elasticsearch to easily query, correlate, and visualize log data via Kibana, but this capability is not the same as application performance monitoring (APM) such as Ruxit and New Relic. APM tools can map layers of an application system and make it easy to drill into a performance problem in the context of a single application (such as web, app tier, and database) in order to understand the root cause of poor performance or reliability. Elasticsearch log analysis and an APM tools approach are therefore fully complementary.

# What About Commercial Monitoring Tools?

There are many popular commercial logging and monitoring tools available from AWS partners such as Splunk, Sumologic, Loggly, and Datadog. These software-as-a-service (SaaS) and packaged software products provide real value and typically support a high level of commercial feature polish. These packages are generally offered as SaaS offerings that require no installation, or as packaged software that installs very simply, making getting started easy.

You might decide that you have enough spare time to devote to setting up Amazon ES and related log agents, and that the capability it provides meets your requirements. Your decision to pick Amazon ES versus commercial software should include the cost of labor to establish and manage the service, the setup and configuration time for the AWS

services that you are using, and the server and application instance logs that you want to monitor.

Kibana's analytics capabilities continue to improve, but are still relatively limited when compared with commercial, purpose-built monitoring software. Kibana does not presently support any model of user administration, meaning that a web proxy must be used to secure access. Once a user is past the proxy, access is open to all log data presented. Commercial monitoring and logging products such as the ones we mentioned typically have very robust user administration capabilities.

Finally, Elasticsearch and Kibana are not optimized for server and application up/down monitoring. They are best used with free text querying and API interaction as well as dashboard-based viewing of log statistics.

# Why Use Amazon ES?

If you use Amazon ES, you will save considerable effort establishing and configuring a cluster as well as maintaining it over time. Amazon ES automatically finds and replaces failed nodes in a cluster, and you can create or scale up a cluster with a few clicks in the console or a simple API call or command line interface (CLI) command. Amazon ES also automatically configures and provisions a Kibana endpoint, which you can use to begin visualizing your data. You can create Kibana dashboards from scratch or import JSON files describing predefined dashboards, and customize from there. Let's demonstrate how easy it is to provision an Amazon ES cluster.

## Provisioning an Amazon ES Cluster

The following procedure show you how to set up an Amazon ES cluster. You can perform the steps in the AWS console or through the AWS CLI. For this procedure, we assume that you are starting with Amazon Linux or have installed the AWS Command Line Interface (AWS CLI) before using the commands.

1. Use the following command to configure your AWS CLI environment to point to your AWS Region of choice.

```
aws configure
```

2. Create a cluster configuration file in JSON as well as Identity and Access Management (IAM) access policies to allow for testing, and then use the CLI to create an Amazon ES domain, as shown in the following example:

```
cat << EOF > /tmp/elasticsearch_cluster_config.json
{
    "InstanceType":           "m4.large.elasticsearch",
    "InstanceCount":          4,
    "DedicatedMasterEnabled": true,
    "ZoneAwarenessEnabled":   true,
    "DedicatedMasterType":    "m4.large.elasticsearch",
    "DedicatedMasterCount":   3
}
EOF

cat << EOF > /tmp/iam_policy.json
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect":    "Allow",
        "Principal": { "AWS": ["$ESTEST_ACCOUNT_ID"] },
        "Action":    [ "es:*" ],
        "Resource":  "arn:aws:es:us-west-
2:$ESTEST_ACCOUNT_ID:domain/$ESTEST_ELASTICSEARCH_DOMAIN/*"
    }]
}
EOF

aws es create-elasticsearch-domain \
        --domain-name $ESTEST_ELASTICSEARCH_DOMAIN \
        --elasticsearch-cluster-config
file:///tmp/elasticsearch_cluster_config.json \
        --ebs-options
EBSEnabled=true,VolumeType=standard,VolumeSize=100 \
        --snapshot-options AutomatedSnapshotStartHour=5 \
        --access-policies file:///tmp/iam_policy.json
```

3. Next, tag the cluster and domain, as shown in the following example:

```
ESTEST_ELASTICSEARCH_ARN=$(aws es describe-elasticsearch-domain \
                                    --domain-name
$ESTEST_ELASTICSEARCH_DOMAIN \
                                    | jq .DomainStatus.ARN --raw-output)
```

```
aws es add-tags --arn $ESTEST_ELASTICSEARCH_ARN --tag-list
Key=Name,Value=$ESTEST_ELASTICSEARCH_DOMAIN
```

Now you are ready to interact with the cluster through the API.

**Important**: Because Amazon ES runs in an AWS-managed VPC and not in a VPC that you control, you need to secure access to it and the Kibana dashboards that you will use with it. There are two starting points for this:

- IP address restrictions configured with EC2 Security Groups.[5]
- HTTP basic Auth[6] configured through an nginx proxy that sits in front of the Amazon ES endpoint.[7]

Using nginx with SSL/TLS to provide user administration and block all other traffic should be implemented prior to use with production data, as the first two methods are relatively weak security methods.

# Interacting with Amazon ES

It's easy to interact with Amazon ES. Perform the following procedure to get started.

1. First, set several environment variables:

```
ESTEST_REGION=us-west-2
ESTEST_ACCOUNT_ID=<Your account ID>
ESTEST_ELASTICSEARCH_DOMAIN=estest-domain
ES_CLUSTER_DNS=$(aws es describe-elasticsearch-domain \
                      --domain-name $ESTEST_ELASTICSEARCH_DOMAIN \
               | jq .DomainStatus.Endpoint --raw-output)

ES_CLUSTER=http://$ES_CLUSTER_DNS
```

2. Next, you can use cURL to PUT and GET data into the cluster. The following example shows how to add a record into an index:

```
curl -s -XPOST "$ES_CLUSTER/ecorp/employee/1" -d '
{
    "first_name": "John",
    "last_name": "Smith",
```

```
    "age": 32,
    "about": "I like to build cabinets",
    "interests": ["sports", "music"]
}' | jq .
```

3. Now add a second record:

```
curl -s -XPOST "$ES_CLUSTER/ecorp/employee/2" -d '
{
    "first_name": "Jane",
    "last_name": "Smith",
    "age": 23,
    "about": "I like to collect rock albums",
    "interests": ["music"]
}' | jq .
```

4. The following example shows how to use Python code to add a record. This code uses the standard HTTP requests module (requests) to issue the REST API calls:

```
import requests
import sys

url='http://{NAME OF YOUR AMAZON ES DOMAIN}.us-west-2.es.amazonaws.com'
json1 = """
{
    "first_name": "John",
    "last_name": "Smith",
    "age": 32,
    "about": "I like to build cabinets",
    "interests": ["sports", "music"]
}"""
r = requests.put(url + "/ecorp/employee/1", data=json1)
print(r.text)
```

5. You can also manage the cluster via the REST API. The following cluster status example assumes a cluster that allows access from any incoming IP address:

```
curl -s $ES_CLUSTER | jq .
```

Next, you can set up to push log data into Amazon ES.

# Pushing Log Data from EC2 Instances into Amazon ES

While many Elasticsearch users favor the "ELK" (Elasticsearch, Logstash, and Kibana) stack, a serverless approach using Amazon CloudWatch Logs has some distinct advantages. You can consolidate your log feeds, install a single agent to push application and system logs, remove the requirement to run a Logstash cluster on Amazon EC2, and avoid having any additional monitoring or administration requirements related to log management. However, before going serverless you might want to review and consider whether you will need some of the more advanced Logstash transformation capabilities that the CloudWatch Logs agent does not support.

The following process (Figure 4) shows how to set up CloudWatch Logs agent on an Ubuntu EC2 instance to push logs to Amazon ES. AWS Lambda lets you run code without provisioning or managing servers. As logs come in, AWS Lambda runs code to put the log data in the right format and move it into Amazon ES using its API.

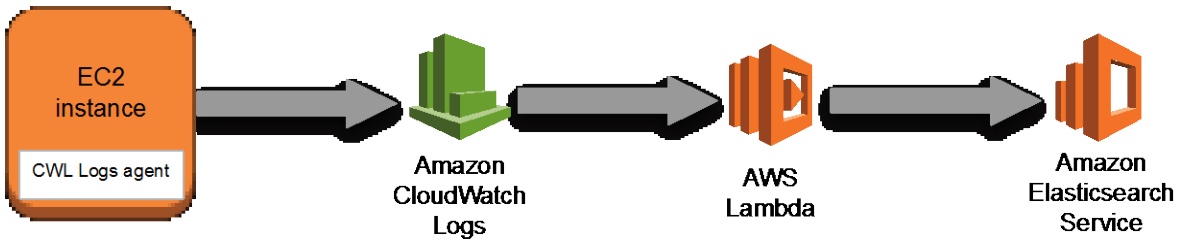


**Figure 4: CloudWatch Logs architecture**

1. First, set several environment variables:

```
ESTEST_INSTANCE_1_KEYPAIR{PATH TO KEYPAIR}/{NAME OF KEYPAIR}.pem
ESTEST_INSTANCE_1_NAME=ESTEST_Instance_1-$(date "+%M%S")
ESTEST_INSTANCE_1_AMI=ami-5189a661 # Ubuntu server

ESTEST_INSTANCE_1_IAM_ROLE_NAME=ESTEST_Instance1-IAM_Role
ESTEST_INSTANCE_1_IAM_POLICY_NAME=ESTEST_Instance1-IAM_Policy
ESTEST_INSTANCE_1_PROFILE_NAME=ESTEST_Instance1-Instance_Profile
```

2. Create an IAM role for EC2 using the AWS CLI. The IAM role grants applications running on the EC2 instance the necessary permissions to write to the CloudWatch Logs group:

```
aws iam create-role \
        --role-name "$ESTEST_INSTANCE_1_IAM_ROLE_NAME" \
        --output text \
        --query 'Role.Arn' \
        --assume-role-policy-document '{
              "Version": "2012-10-17",
              "Statement": [{
                  "Effect": "Allow",
                  "Principal": { "Service": "ec2.amazonaws.com"},
                  "Action": "sts:AssumeRole"
              }]
            }'

ESTEST_INSTANCE_1_IAM_ROLE_ARN=arn:aws:iam::$ESTEST_ACCOUNT_ID:role/$ES
TEST_INSTANCE_1_IAM_ROLE_NAME

aws iam put-role-policy \
        --role-name   "$ESTEST_INSTANCE_1_IAM_ROLE_NAME"   \
        --policy-name "$ESTEST_INSTANCE_1_IAM_POLICY_NAME" \
        --policy-document '{
                  "Version": "2012-10-17",
                  "Statement": [{
                      "Effect": "Allow",
                      "Resource": [ "arn:aws:logs:*:*:*" ],
                      "Action": [
                        "logs:CreateLogGroup",
                        "logs:CreateLogStream",
                        "logs:PutLogEvents",
                        "logs:DescribeLogStreams"
                      ]
                  }]
                }'

aws iam create-instance-profile --instance-profile-name
$ESTEST_INSTANCE_1_PROFILE_NAME

aws iam add-role-to-instance-profile \
    --instance-profile-name $ESTEST_INSTANCE_1_PROFILE_NAME \
    --role-name $ESTEST_INSTANCE_1_IAM_ROLE_NAME
```

3. Create a security group:

```
aws ec2 create-security-group --group-name $ESTEST_INSTANCE_1_NAME --
description "$ESTEST_INSTANCE_1_NAME"
aws ec2 authorize-security-group-ingress --group-name
$ESTEST_INSTANCE_1_NAME --protocol tcp --port 22 --cidr 0.0.0.0/0
```

4. Launch the instance, and tag the instance:

```
ESTEST_INSTANCE_1_ID=$(aws ec2 run-instances \
         --image-id $ESTEST_INSTANCE_1_AMI \
         --count 1 \
         --instance-type m3.medium \
         --key-name jtkeypair_pdx \
         --security-groups $ESTEST_INSTANCE_1_NAME \
         --iam-instance-profile
Name=$ESTEST_INSTANCE_1_IAM_INSTANCE_PROFILE_NAME \
         --region us-west-2 \
         | jq --raw-output .Instances[0].InstanceId) && echo
$ESTEST_INSTANCE_1_ID

aws ec2 create-tags --resources $ESTEST_INSTANCE_1_ID --tags
Key=Name,Value=$ESTEST_INSTANCE_1_NAME
```

5. Log in to the Ubuntu instance that will originate the logs:

```
ESTEST_INSTANCE_1_DNS=$(aws ec2 describe-instances --instance-ids
$ESTEST_INSTANCE_1_ID | jq --raw-output
.Reservations[0].Instances[0].PublicDnsName) && echo
$ESTEST_INSTANCE_1_DNS

ssh -i $ESTEST_INSTANCE_1_KEYPAIR ubuntu@$ESTEST_INSTANCE_1_DNS
```

6. Install the CloudWatch Logs agent:

```
sudo apt-get update
wget https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-
agent-setup.py
sudo python ./awslogs-agent-setup.py --region us-west-2
```

You will be prompted for the location of the application and system logs, datestamp format, and a starting point for the log upload. Your logs will be stored in CloudWatch, and you can stream them into Amazon ES. You can perform the preceding steps for all EC2 instances that you want to connect to CloudWatch, you can use the EC2 **Run** command to install across a fleet of instances, or you can build a boot script to use with auto scaled instances.

To connect a CloudWatch stream to Amazon ES, follow these steps in the AWS documentation using the name of an Amazon ES domain previously created to subscribe your new log group to Amazon ES.[8] Note that there are several log formatting options that you might want to review during the connection process, and you can exclude log information that is not of interest to you. You will be prompted to create an AWS Lambda execution role because AWS uses Lambda to integrate your CloudWatch log group to Amazon ES.[9]

You have now created an Amazon ES domain and configured one or more instances to send data to CloudWatch Logs, which then can be forwarded to Amazon ES via Lambda.

# Pushing Amazon CloudWatch Logs into Amazon ES

The CloudWatch Logs→Lambda→Amazon ES integration makes it easy to send data to Elasticsearch if source data exists in CloudWatch Logs. Figure 5 shows the features and services that you can use to process different types of logs.

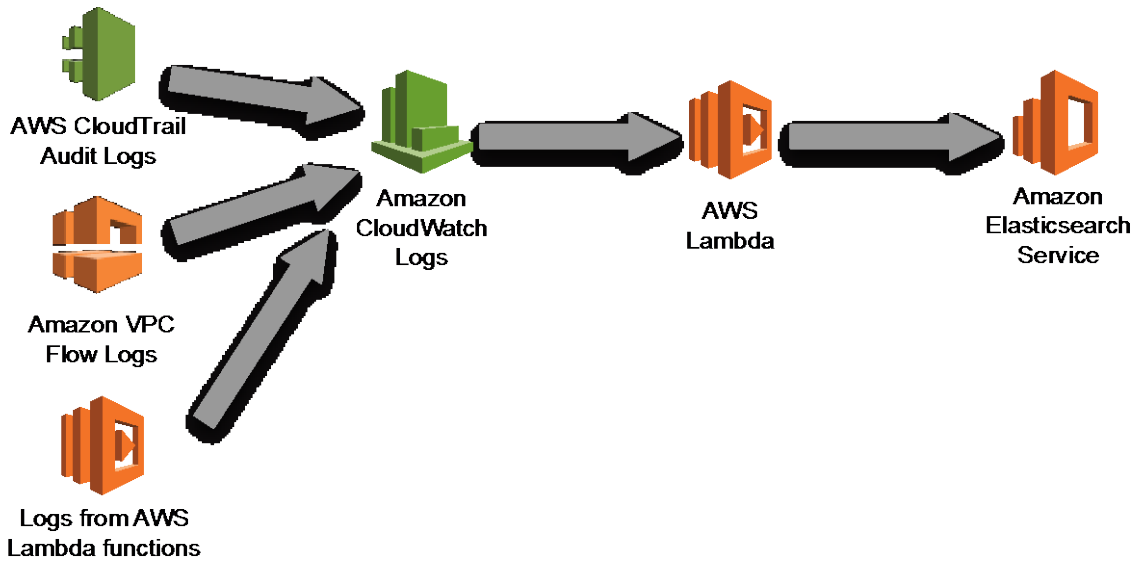


**Figure 5: Pushing CloudWatch Logs into Amazon ES**

- **AWS API activity logs (AWS CloudTrail)**: AWS CloudTrail tracks your activity in AWS, and provides you with an audit trail for API activity in your AWS account. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service.

  You should enable CloudTrail logging for all AWS Regions. CloudTrail logs can be sent to Amazon S3 or to CloudWatch Logs; for the purposes of sending logs to Amazon ES as a final destination, it is easier to send to CloudWatch Logs.

- **Network activity logs (VPC Flow Logs):** VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your Amazon Virtual Private Cloud (Amazon VPC). VPC Flow Log data is stored as CloudWatch Logs.

- **Application logs from AWS Lambda functions:** Application logs from your Lambda code are useful for code instrumentation, profiling, and general troubleshooting. In the code for your AWS Lambda functions, any console output that typically would be sent to standard output is delivered as CloudWatch Logs. For example: console.log() statements for Node.js functions, print() statements for Python functions, and System.out.println() statements for Java functions.

# Using AWS Lambda to Send Logs into Amazon ES

For maximum flexibility, you can use AWS Lambda to send logs directly to your Elasticsearch domain. Custom logic in your Lambda function code can then perform any desired data processing, cleanup, and normalization before sending the log data to Amazon ES. This approach is highly flexible. However, it does require technical understanding of how AWS Signature Version 4 security works.[10]

For security purposes, in order to issue any queries or updates against an Elasticsearch cluster, the request must be signed using AWS Signature Version 4 ("SigV4 signing"). Signature Version 4 is the process to add authentication information to AWS requests. Rather than implementing SigV4 signing on your own, we highly recommend that you adapt existing SigV4 signing code. For the CloudWatch Logs→Lambda→Amazon ES integration described earlier, the Lambda code for implementing SigV4 signing is automatically generated for you. If you inspect the code associated with the auto-generated Lambda function, you can view the SigV4 signing code that is used to authenticate against the Elasticsearch cluster. You can copy the code as a starting point for your Lambda functions that need to interact with the Amazon ES cluster. Another example of code implementing SigV4 signing is described in the AWS blog post *How to Control Access to Your Amazon Elasticsearch Service Domain*.[11]

Many AWS services have built-in integrations with AWS Lambda. In AWS Lambda, an event source is the entity that publishes events, and a Lambda function is the custom code that processes the published events. As of this writing, the following AWS services can serve as event sources for your Lambda function:

- Amazon S3
- Amazon Kinesis Streams
- Amazon DynamoDB
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Email Service (Amazon SES)
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs

- Amazon CloudWatch Events
- Scheduled Events (powered by Amazon CloudWatch Events)
- AWS Config
- Amazon Echo
- Amazon API Gateway

Of course, you can also publish custom events to Lambda from an application.

These AWS event sources can provide data to your Lambda function code, and your Lambda function code can process and send that data to your Amazon ES cluster. For example, log files stored on S3 can be sent to Amazon ES via Lambda.

Streaming data sent to an Amazon Kinesis stream can be forwarded to Amazon ES via Lambda. A Kinesis stream will scale up to handle very high log data rates without any management effort on your part, and AWS will manage the durability of the stream for you. For more information about the data provided by each of these AWS event sources, see the AWS Lambda documentation.[12]

The S3→Lambda→Amazon ES integration pattern is a particularly useful one. As one example, many AWS-powered websites store their web access logs in Amazon S3. If your website uses Amazon CloudFront (for global content delivery), Amazon S3 (for static website hosting), or Elastic Load Balancing (for load balancers in front of your web servers), then you should enable the access logs for each service. There is no extra charge to enable logging, other than the cost of storage for the actual logs in Amazon S3. Once the log files are in Amazon S3, you can process them using Lambda and send them to Amazon ES, where you can analyze your website traffic using Kibana.

# Using Amazon Kinesis Firehose to Load Data into Amazon ES

As another option, you can use Amazon Kinesis Firehose to transform your data and load it to Amazon ES. This approach requires you to install the Amazon Kinesis agent on the EC2 instances that you want to monitor. You don't need to transmit log information to CloudWatch Logs. Because Firehose is a highly scalable managed service, you can transmit log data from hundreds or thousands of instances in a very large installation. You should consider Firehose if you have the following requirements:

- Very high-scale log monitoring installation
- Serverless approach to transforming and loading log data
- Simultaneously store logs in an S3 bucket for compliance or archival purposes, while continuously transmitting to Amazon ES

Amazon Kinesis Firehose is a rich and powerful real-time stream management system that is directly integrated with Amazon ES. The following illustration shows the flow of logs managed by Firehose into Amazon ES.

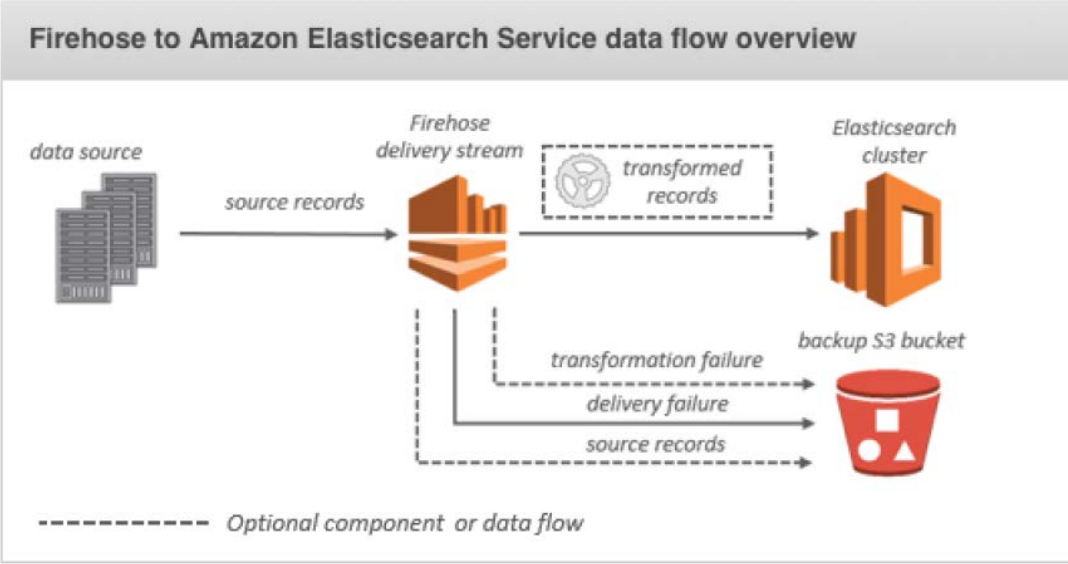


**Figure 4: Overview of Firehose to Amazon ES data flow**

Support for Apache Web logs is built in. To help you evaluate Amazon Kinesis Firehose for log analytics using Amazon ES as a target, see this tutorial.[13]

# Putting It All Together

Figure 7 is an example how the various patterns can be combined for sending logs to Elasticsearch, using AWS CloudWatch Logs, Amazon Kinesis Firehose, and AWS Lambda.

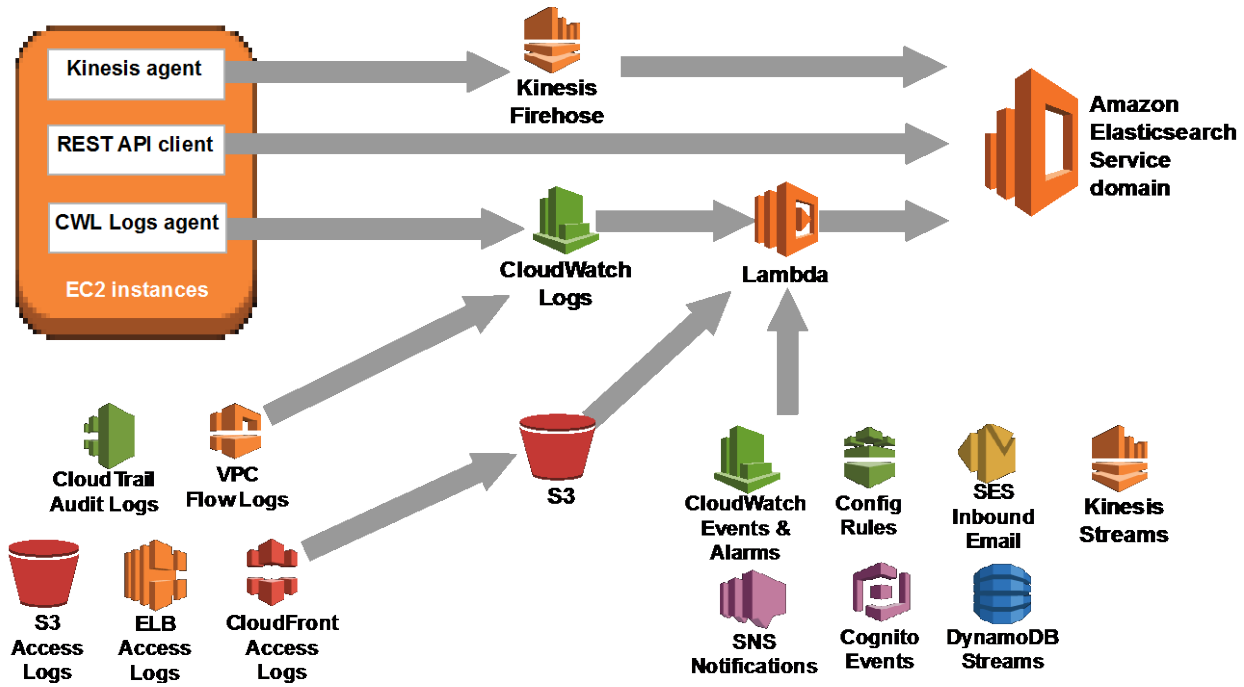


**Figure 7: Serverless Amazon ES monitoring architecture**

# Setting Up Kibana to Visualize Logs

One advantage of Amazon ES is that Kibana is set up and ready to configure after you create your search domain. When you first start Kibana, you are prompted to configure an index pattern.

Community support for Kibana has produced several types of useful Kibana dashboards that are preconfigured. The [main GitHub repository][14] contains dashboards to visualize:

- Amazon Elasticsearch Cluster statistics (KOPF)
- Amazon VPC Flow Logs
- AWS CloudTrail Logs
- AWS Lambda Logs

Remember the requirement to lock down access to Kibana for all users. A best practice is to use HTTPS for all connections and user administration using a proxy like Nginx to limit access and provide adequate security. Your Amazon ES domain and Kibana dashboards should not be accessible without proxy access control.

# Next Steps

Once you have CloudWatch Logs flowing into Amazon ES, make sure you have all of the other types of AWS logs enabled (such as CloudTrail Logs). As you add new log types, you can add or configure additional Kibana dashboards to match the inbound log pattern.

# Contributors

The following individuals and organizations contributed to this document:

- Jim Tran, Principal Enterprise Solutions Architect, AWS
- Lex Crosett, Enterprise Solutions Architect, AWS

# Notes

[1] https://www.elastic.co/guide/en/kibana/current/access.html

[2] http://www.elastic.io/

[3] https://www.elastic.co/guide/en/logstash/current/introduction.html

[4] https://www.elastic.co/products/beats

[5] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html

[6] https://en.wikipedia.org/wiki/Basic_access_authentication

[7] http://nginx.org/

[8] https://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/CWL_ES_Stream.html

[9] https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_ES_Stream.html

[10] http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

[11] https://blogs.aws.amazon.com/security/post/Tx3VP208IBVASUQ/How-to-Control-Access-to-Your-Amazon-Elasticsearch-Service-Domain

[12] http://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html

[13] https://aws.amazon.com/getting-started/projects/build-log-analytics-solution/

[14] https://github.com/awslabs/cloudwatch-logs-subscription-consumer/tree/master/configuration/kibana