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

First published December 2016

Updated July 13, 2021

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. 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.

Amazon ES is a service designed to be useful for logging and monitoring. It is fully managed by Amazon Web Services (AWS) and offers compelling value relative to its cost of operation. This whitepaper provides best practices for feeding log data into Elasticsearch and visualizing it with Kibana using a serverless, inbound log management approach.

It shows how to use Amazon CloudWatch Logs and the unified Amazon CloudWatch Logs agent to manage inbound logs in Amazon Elasticsearch. You can use this approach instead of the more traditional ELK Stack (Elasticsearch-Logstash-Kibana) approach. It also shows you how to move log data into Amazon ES using Amazon Kinesis Data Firehose – and identifies the strengths and weaknesses of using Kinesis versus the simpler 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 (Amazon 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 AWS Auto Scaling events. Many of these log types have no direct analogy 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, are popular open-source tools because they are simple to use and provide a quick time to value. Additionally, the tools are fully supported by AWS Support, 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. AWS currently supports versions of Elasticsearch and Kibana from 1.5 to 7.10.

At the date of this writing, the new 6.x and 7.x versions of Elasticsearch and Kibana offer several new features and improvements, including UltraWarm, real-time anomaly detection, index splitting, weighted average aggregation, higher indexing performance, improved cluster coordination safeguards, and an option to multiplex token filters, support for field aliases, and improved workflow for inspecting the data behind a visualization.

You can create new domains running Elasticsearch 7.10 and also easily upgrade existing 5.6 and 6.x domains with no downtime using in-place version upgrades.

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 among three Availability Zones.

Amazon ES can be scaled up from a default limit of 20 data nodes to 200 data nodes in a single cluster, including up to 3 petabytes of storage by requesting a service limit increase. 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) and piped-processing-language (ppl) 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.

This whitepaper shows you how to provision an Amazon ES Cluster, push log data from Amazon EC2 Instances into Amazon Elasticsearch, push Amazon CloudWatch Logs into Amazon Elasticsearch, use AWS Lambda to send logs into Amazon Elasticsearch, use Amazon Kinesis Firehose to load data into Amazon ES, implement Kubernetes logging with EFK and Amazon Elasticsearch, and configure Alerting for Amazon ES.

What Is Elasticsearch?

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)

In the following example of a document, the document ID is **34171**. The fields include first name, last name, and so on. Note that document types will be deprecated in APIs in Elasticsearch 7.0.0, and completely removed in 8.0.0.

Figure 1 – Example of an Elasticsearch document

Elasticsearch supports a RESTful web services interface. You can use PUT, GET, POST and DELETE commands to interface with an 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 run 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.

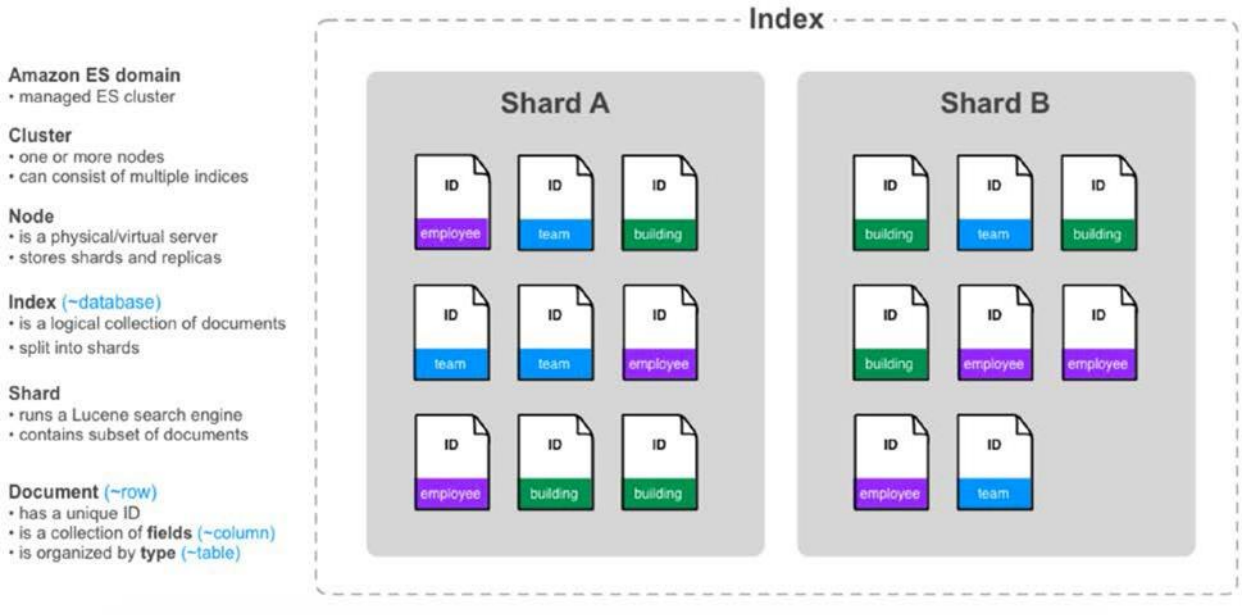

Figure 2 – Elasticsearch terminology

You can think of an Amazon ES domain as a service wrapper around an Elasticsearch cluster, and the logical API entry point to interfaces with 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.

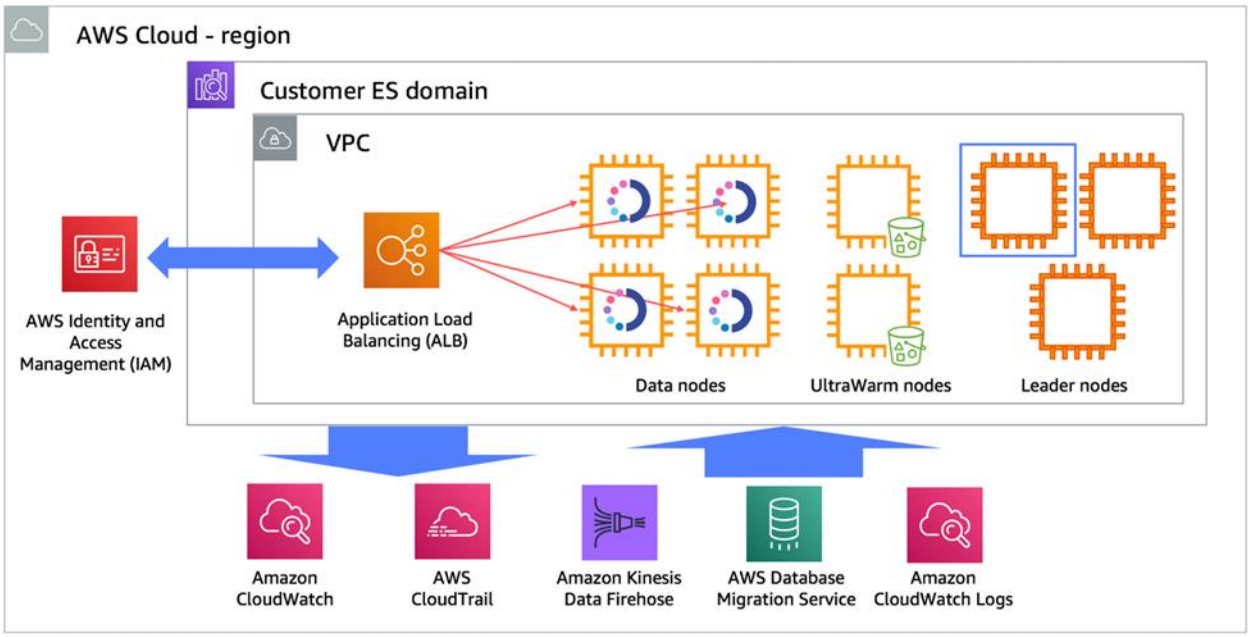

Figure 3 – Amazon ES architecture

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.

Elasticsearch 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.

Amazon Elasticsearch Service (Amazon ES) provides support for cross-cluster search, enabling you to perform searches, aggregations, and visualizations across multiple Amazon ES domains with a single query or from a single Kibana interface. With this feature, you can separate heterogeneous workloads into multiple domains, which provides better resource isolation, and the ability to tune each domain for their specific workloads which can improve availability and reduce costs.

Trace Analytics is a new feature of Amazon Elasticsearch Service that enables developers and IT operators to find and fix performance problems in distributed applications, which leads to faster problem resolution times. Trace Analytics is built using OpenTelemetry, a Cloud Native Computing Foundation (CNCF) project that provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics, which enables customers to leverage Trace Analytics without having to re-instrument their applications. Trace Analytics is powered by the Open Distro for Elasticsearch project, which is open source and freely available for everyone to download and use.

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 generally require no installation or they are software packages that install 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. Commercial monitoring and logging products such as the ones we mentioned typically have very robust user administration capabilities.

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.

It is easy to provision an Amazon ES cluster. You can use the Amazon ES console to set up and configure a domain in minutes. If you prefer programmatic access, you can use the AWS CLI or the AWS SDKs.

Following steps are typically what you need to do to provision an Amazon ES cluster:

- Create a domain
- Size the domain appropriately for your workload
- Control access to your domain using a domain access policy or fine-grained access control
- Index data manually or from other AWS services
- Use Kibana to search your data and create visualizations

Best practices for configuring your Amazon ES domain

When you configure your Amazon ES domain, you choose the instance type and count for data and the dedicated master nodes. Elasticsearch is a distributed service that runs on a cluster of instances or nodes. These node types have different functions and require different sizing. Data nodes store the data in your indexes and process indexing and query requests. Dedicated master nodes don't process these requests; they maintain the cluster state and orchestrate.

Amazon ES supports five instance classes: M, R, I, C, and T. As a best practice, use the latest generation instance type from each instance class. For the latest supported instance classes, see Supported instance types in Amazon Elasticsearch Service. When choosing an instance type for your data nodes, bear in mind that these nodes carry all the data in your indexes (storage) and do all the processing for your requests (CPU). As a best practice, for heavy production workloads, choose the R5 or I3 instance type. If your emphasis is primarily on performance, the R5 typically delivers the best performance for log analytics workloads, and often for search workloads. The I3 instances are strong contenders and may suit your workload better, so you should test both. If your emphasis is on cost, the I3 instances have better cost efficiency at scale, especially if you choose to purchase reserved instances.

For an entry-level instance or a smaller workload, choose the M5s. The C5s are a specialized instance, relevant for heavy query use cases, which require more CPU work than disk or network. Use the T2 or T3 instances for development or QA workloads, but not for production.

When choosing an instance type for your dedicated master nodes, keep in mind that these nodes are primarily CPU-bound, with some RAM and network demand as well. The C5 instances work best as dedicated masters up to about 75 data node clusters. Above that node count, you should choose R5.

For log analytics use cases, you want to control the life cycle of data in your cluster. You can do this with a rolling index pattern. Each day, you create a new index, then archive and delete the oldest index in the cluster.

You define a retention period that controls how many days (indexes) of data you keep in the domain based on your analysis needs. For more information, see Index State Management. You should try to align your shard and instance counts so that your

shards distribute equally across your nodes. You do this by adjusting shard counts or data node counts so that they are evenly divisible.

Elasticsearch Security and Compliance

Security

Amazon ES service is a managed service, this means that AWS is responsible for security of the underlying infrastructure and operating system, patching and management of Elasticsearch software, while you are responsible for setup of service level security controls. This would include areas such as management of authentication and access controls, data encryption in motion and data encryption at rest.

Authentication and access control for Elasticsearch are implemented using a combination of Sigv4 signing and AWS IAM. Integration with Sigv4 will be covered in greater depth during the setup of logging services. For examples of IAM policies that can be used in security access to Amazon Elasticsearch using resource based policies, identity based policies, or IP based policies, review these policy examples.

All Amazon Elasticsearch domains are created in a dedicated VPC. This setup keeps the cluster secure and isolates inter-node network traffic. By default, traffic within this isolated VPC is unencrypted, but you can also enable node-to-node TLS encryption. This feature must be enabled at the time of Elasticsearch cluster creation. To use this feature for an existing cluster, you must create a new cluster and migrate your data. Node-to-node encryption requires Elasticsearch version 6.0 or later.

For enabling data encryption at rest, Amazon ES service natively integrates with AWS Key Management Service (AWS KMS), making it easy to secure data within Elasticsearch indices, automated snapshots, Elasticsearch logs, swap files, and all data in the application directory. This option, along with node-to-node encryption, must be set up during domain creation. Encryption of data at rest requires Elasticsearch 5.1 or later.

Encryption of manual snapshots and encryption of slow logs and error logs must also be configured separately. Manual snapshots can be encrypted using server side encryption in S3. For more details, see Registering a manual snapshot repository. If published to Amazon CloudWatch, slow logs and error logs can be encrypted using the same KMS master key as the ES domain. For more information, see Encrypt log data in CloudWatch Logs using AWS KMS.

Amazon ES Service offers fine-grained access control (FGAC), which adds multiple capabilities to give you tighter control over your data. FGAC features include the ability to use roles to define granular permissions for indices, documents, or fields and to extend Kibana with read-only views and secure multi-tenant support.

Two forms of authentication and authorization are provided by FGAC: a built-in user database, which makes it easy to configure usernames and passwords inside of Elasticsearch, and AWS Identity and Access Management (IAM) integration, which lets you map IAM principals to permissions. Powered by Open Distro for Elasticsearch, which is an Apache 2.0-licensed distribution of Elasticsearch, Fine-grained access control is available on domains running Elasticsearch 6.7 and higher.

Compliance

By choosing to use the Amazon Elasticsearch service, you can greatly reduce your compliance efforts by building compliant applications on top of existing AWS compliance certifications and attestations.

Amazon Elasticsearch Service is HIPAA Eligible. You can use Amazon Elasticsearch Service to store and analyze protected health information (PHI) and build HIPAA compliant applications. To set up, visit AWS Artifact in your HIPAA accounts and agree to the AWS Business Associate Agreement. This BAA can be set up for individual AWS accounts, or for all of the accounts under your AWS Organization supervisory account.

Amazon Elasticsearch Service is also in-scope of AWS Payment Card Industry Data Security Standard (PCI DSS) which allows you to store, process, or transmit cardholder data using the service.

Additionally, Amazon Elasticsearch Service is in-scope for the AWS ISO 9001, 27001, 27017, and 27018 certifications. PCI DSS and ISO are among the most recognized global security standards for attesting to quality and information security management in the cloud.

AWS Config is a service that continuously monitors the configuration of AWS Services for compliance and can automate remediation actions using AWS Config rules. In the case of Amazon Elasticsearch Service, you should consider enabling Config rules such as:

- elasticsearch-in-vpc-only – This checks whether the Amazon Elasticsearch cluster is deployed in a VPC and is NON_Compliant if the ES domain is public,

- elasticsearch-encrypted-at-rest – This will do a check to ensure Amazon Elasticsearch domains have been deployed with encryption at rest enabled and is NON_Compliant if the EncryptionAtRestOptions field is not enabled.

Amazon Elasticsearch Service offers a detailed audit log of all Elasticsearch requests. Audit Logs allows customers to record a trail of all user actions, helping meet compliance regulations, improving the overall security posture and providing evidence for security investigations.

Amazon Elasticsearch Service Audit Logs allows customers to log all of their user activity on their Elasticsearch clusters, including keeping a history of user authentication success and failures, logging all requests to Elasticsearch, modifications to indices, recording incoming search queries and much more.

Audit Logs provides a default configuration that covers a popular set of user actions to be tracked. Administrators can further configure and fine tune the settings to meet their needs. Audit Logs is integrated with Fine Grained Access Control, allowing you the ability to log access or modification requests to sensitive documents or fields, to meet any compliance requirements.

Once configured, Audit Logs will be continuously streamed to CloudWatch Logs and can be further analyzed there. Audit Logs settings can be changed at any time and are automatically updated. Both new and existing Amazon Elasticsearch Service domains (version 6.7+) with Fine Grained Access Control enabled can use the Audit Logs feature.

Multi-Account Log aggregation use case

An important part of every large enterprise AWS deployment is a multi-account strategy that is setup using either AWS Control Tower or AWS Landing Zones.

This creates a core for the centralized governance of accounts, including the aggregation of the logs from all of a customer's accounts into one centralized account where they can be ingested into Elasticsearch to be correlated and monitored in one central location.

It can include logs from services and components such as CloudTrail Logs, CloudWatch Log Groups, VPC Flow Logs, AWS Config Logs, and Amazon GuardDuty Logs.

In the case of CloudWatch logs, these can be streamed directly to Elasticsearch from all accounts in a customers' organization using methods described in Stream Amazon CloudWatch Logs to a Centralized Account for Audit and Analysis.

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

- IP address restrictions configured with EC2 Security Groups.
- HTTP basic Auth configured through an nginx proxy that sits in front of the Amazon ES endpoint. Using nginx with SSL/TLS to provide user administration and block all other traffic should be implemented prior to using this method with production data, as the first two methods are relatively weak security methods.

Beyond these two basic controls, the preferred method for securing access to Kibana is to enable access using AWS Single Sign-On or your own Federation service. This setup will allow for only users within your Microsoft Active Directory access to visualize data stored in Elasticsearch. It uses a standard SAML identity federation approach and a specific Active Directory group can be used to restrict access to an Amazon Elasticsearch domain.

If you do not already have an Active Directory Domain with your users set up, another option would be to use Amazon Elasticsearch Service native integration with Amazon Cognito User Pools to manage access. This approach provides user level access control to Kibana, access to ES domains, and the ability to set policies for groups of users within the Amazon Cognito User Pool.

UltraWarm storage for Amazon ES

UltraWarm provides a cost-effective way to store large amounts of read-only data on Amazon Elasticsearch Service. Standard data nodes use "hot" storage, which takes the form of instance stores or Amazon EBS volumes attached to each node. Hot storage provides the fastest possible performance for indexing and searching new data.

UltraWarm nodes use Amazon S3 and a sophisticated caching solution to improve performance. For indices that you are not actively writing to, query less frequently, and don't need the same performance as hot storage, UltraWarm offers significantly lower costs per GiB of data.

In Elasticsearch, these warm indices behave just like any other index. You can query them using the same APIs or use them to create dashboards in Kibana.

Because UltraWarm uses Amazon S3, it does not incur overhead which was typically from hot storage. When calculating UltraWarm storage requirements, you consider only the size of the primary shards.

The durability of data in S3 removes the need for replicas, and S3 abstracts away any operating system or service considerations. Each UltraWarm node can use 100% of its available storage for primary data.

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 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

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 the steps in the AWS documentation - Streaming CloudWatch Logs data to Amazon Elasticsearch Service - using the name of an Amazon ES domain previously created to subscribe your new log group to Amazon ES. 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. 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. The following figure shows the features and services that you can use to process different types of logs.

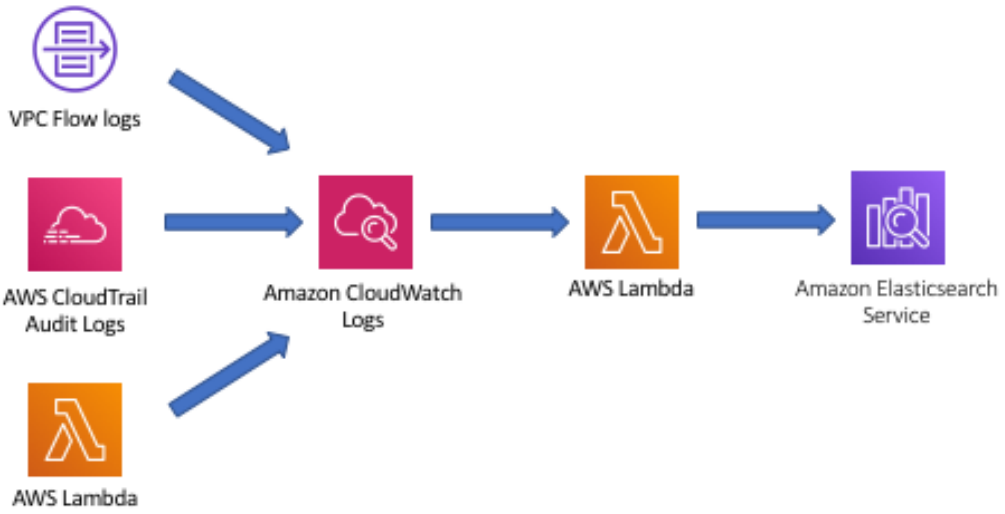

Figure 5 – Pushing CloudWatch Logs into Amazon ES

- **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.

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. Using the AWS SDKs based on your programming language of choice will also take care of the heavy lifting of SigV4 signing making this process much easier.

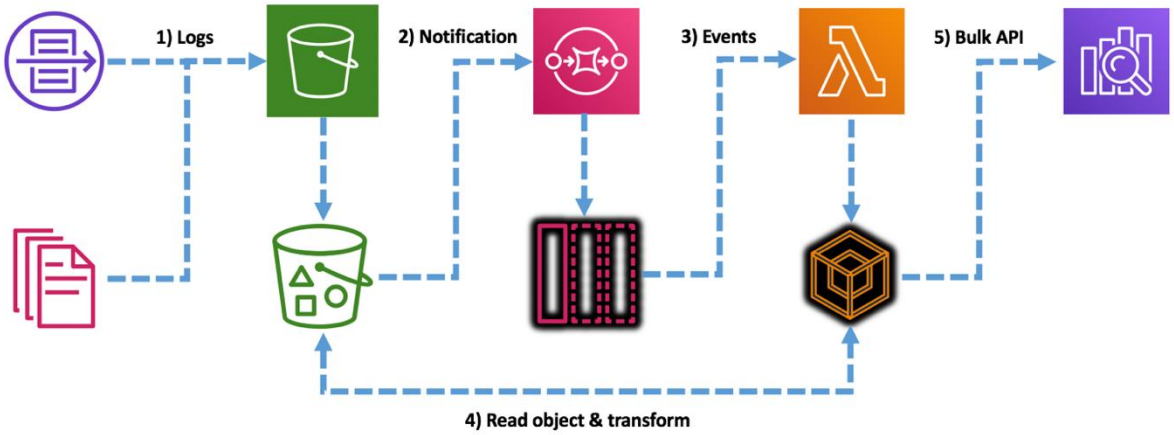

Figure 6 – Overview of Lambda to Amazon ES data flow

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.

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 Data Firehose to load data into Amazon ES

You can use Amazon Kinesis Data 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 Kinesis Data 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 Kinesis Data Firehose if you have the following requirements:

- Large 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 Data 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 Kinesis Data Firehose into Amazon ES.

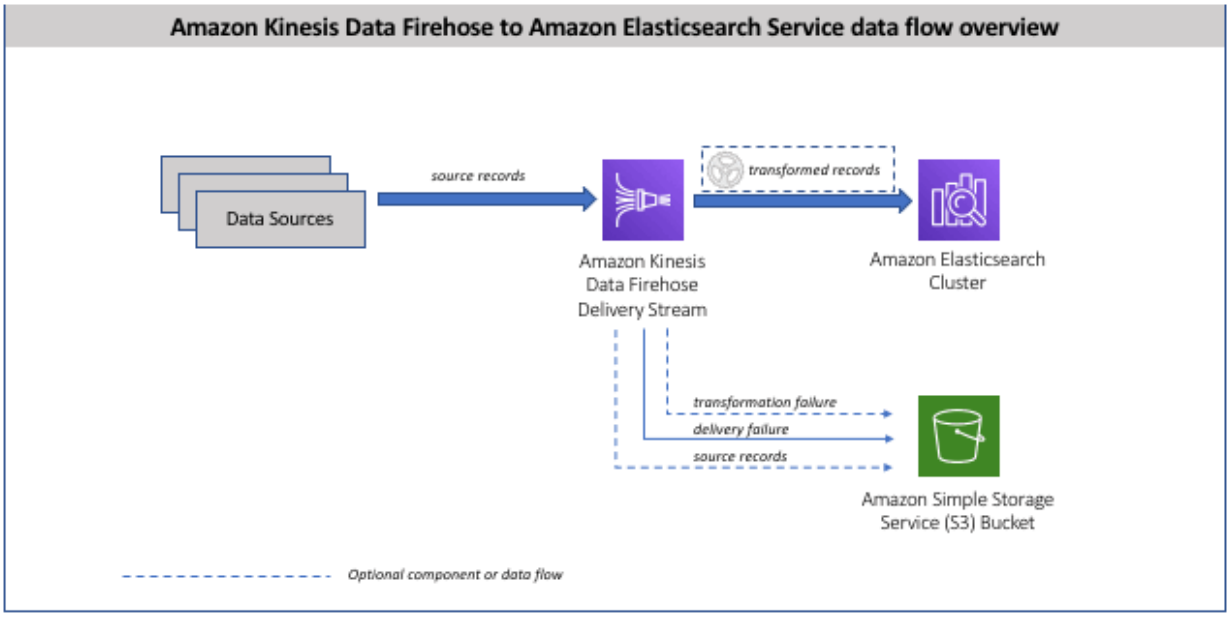

Figure 7 – Overview of Firehose to Amazon ES data flow

Support for Apache Web logs is built in to Amazon Kinesis Data Firehose. To help you evaluate Amazon Kinesis Data Firehose for log analytics using Amazon ES as a target, see the tutorial Build a Log Analytics Solution.

Implement Kubernetes logging with EFK and Amazon ES

The combination of Fluentd unified logging, Elasticsearch RESTFul analytics engine and Kibana for visualizations is known as the EFK stack. Fluentd is configured as a DaemonSet where it collects logs and forwards to Cloudwatch Logs where they can be filtered using a subscription filter and then sent to an ES domain for further querying and visualization. This AWS workshop – Implement Logging with EFK - will walk you through the setup of Kubernetes logging to the EFK stack.

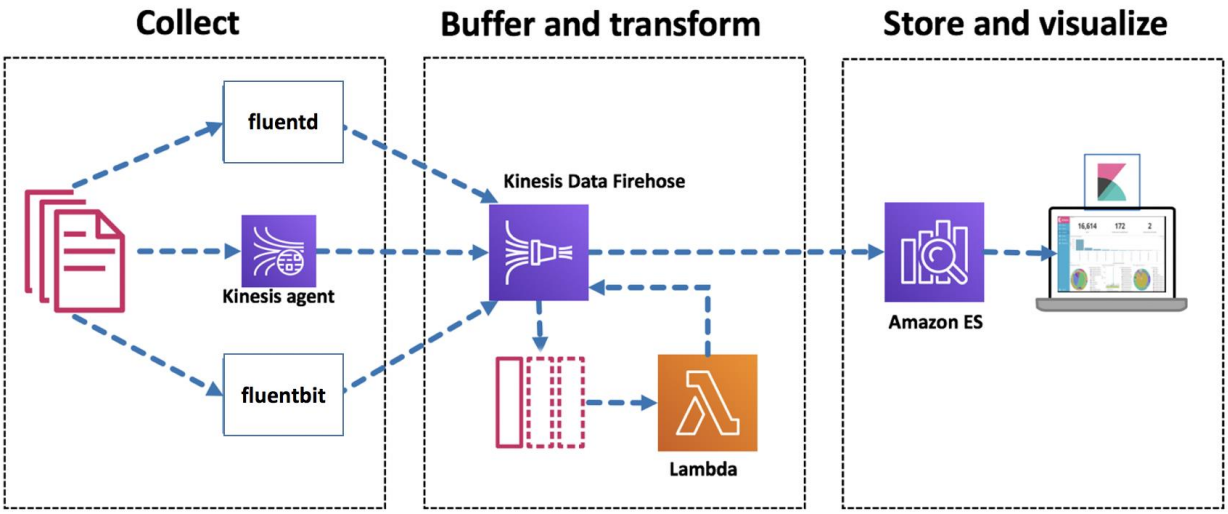

Figure 8 – Setup of Kubernetes logging to EFK Stack

AWS is also supporting Fluent Bit for streaming logs from containerized applications to AWS and partners' solutions for log retention and analytics.

With the Fluent Bit plugin for AWS container images, you can route logs to Amazon CloudWatch and Amazon Kinesis Data Firehose destinations (which includes the Amazon Elasticsearch Service).

The blog post Centralized Container Logging with Fluent Bit contains more information on relative performance of Fluent Bit versus Fluentd and the advantages it offers.

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 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 for this would be to use a corporate LDAP or Active Directory Service to manage access to Kibana.

Alerting for Amazon ES

The Amazon ES alerting feature notifies you when data from one or more Elasticsearch indices meets certain conditions. For example, you might want to receive an email if your application logs more than five HTTP 503 errors in one hour, or you might want to page a developer if no new documents have been indexed in the past 20 minutes. Alerting requires Elasticsearch 6.2 or higher.

Compared to Open Distro for Elasticsearch, the Amazon ES alerting feature has some notable differences. Amazon ES supports Amazon SNS for notifications. This integration with Amazon SNS means that, in addition to standard destinations (Slack, custom webhooks, and Amazon Chime), the alerting feature can send emails, text messages, and even run AWS Lambda functions using SNS topics. The alerting feature supports fine-grained access control. You can mix and match permissions to fit your use cases.

Other configuration options

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.

In addition, you can use the Amazon ES anomaly detection feature to automatically detect anomalies in your log data in near-real time by using the Random Cut Forest (RCF) machine learning algorithm. You can use Trace Analytics to help you visualize this flow of events and identify performance problems.

Conclusion

This whitepaper, explained what Elasticsearch is. It also covered how to use it, compared it with commercial monitoring tools – and explored why you would want to use Amazon Elasticsearch.

In addition, it covered how to configure Amazon Elasticsearch – as well ashow to push logs into it from Amazon EC2, Amazon CloudWatch, AWS Lambda, and Amazon Kinesis Firehose.

Finally it also explained the setup of Kibana for visualization of logs, and alerting for Amazon ES.

Contributors

The following individuals and organizations contributed to this document:

- Jim Tran, Principal Product Manager, AWS
- Pete Buonora, Principal Solutions Architect, AWS
- Changbin Gong, Senior Solutions Architect, AWS
- Naresh Gautam, Senior Analytics Specialist Architect, AWS