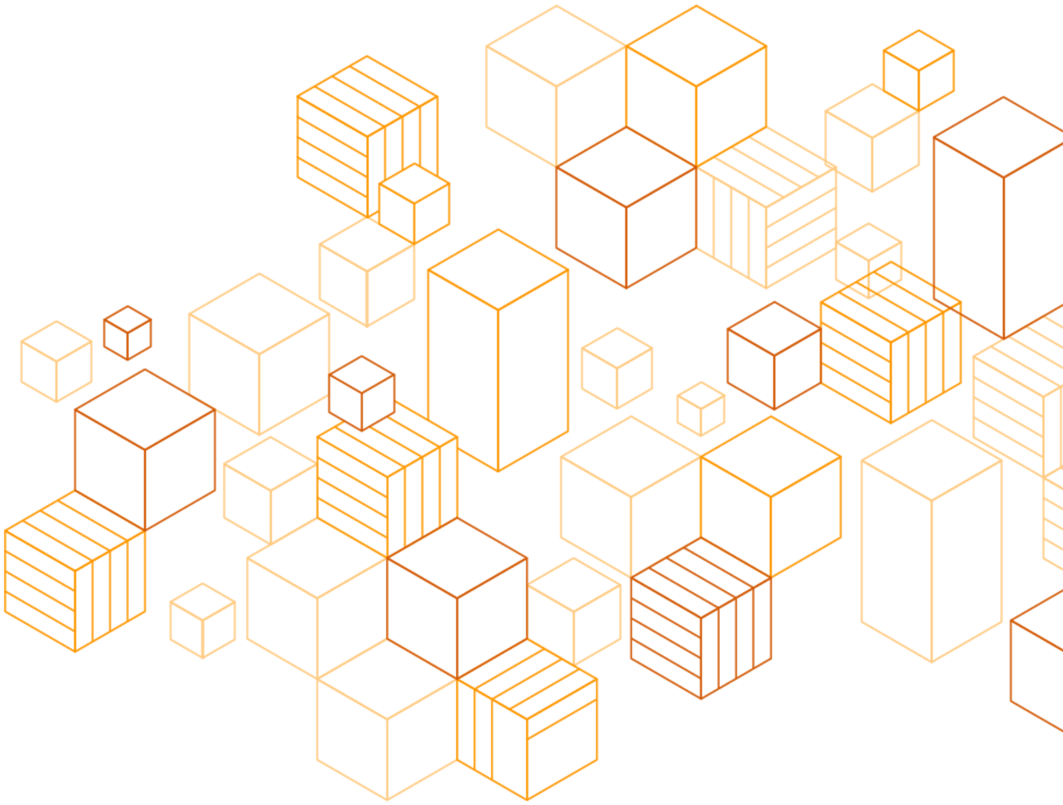
Build a Log Analytics Solution on AWS

July 2020



Notices

Contents

About this Guide

Log analytics is a common big data use case that allows you to analyze log data from websites, mobile devices, servers, sensors, and more for a wide variety of applications such as digital marketing, application monitoring, fraud detection, ad tech, games, and IoT. In this project, you use Amazon Web Services to build an end-to-end log analytics solution that collects, ingests, processes, and loads both batch data and streaming data, and makes the processed data available to your users in analytics systems they are already using and in near real-time. The solution is highly reliable, cost-effective, scales automatically to varying data volumes, and requires almost no IT administration.

Introduction

Amazon Kinesis Data Analytics is the easiest way to process streaming data in real time with standard SQL without having to learn new programming languages or processing frameworks. Amazon Kinesis Data Analytics enables you to create and run SQL queries on streaming data so that you can gain actionable insights and respond to your business and customer needs promptly.

This tutorial walks you through the process of ingesting streaming log data, aggregating that data, and persisting the aggregated data so that it can be analyzed and visualized. You create a complete end-to-end system that integrates several AWS services. You analyze a live stream of Apache access log data and aggregate the total request for each HTTP response type every minute. To visualize this data in near real-time, you use a user interface (UI) tool that charts the results.

Architecture

One of the major benefits to using Amazon Kinesis Data Analytics is that an entire analysis infrastructure can be created with a serverless architecture. The system created in this tutorial implements Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, and Amazon Elasticsearch Service. Each of these services is designed for seamless integration with one another. The architecture is depicted below.

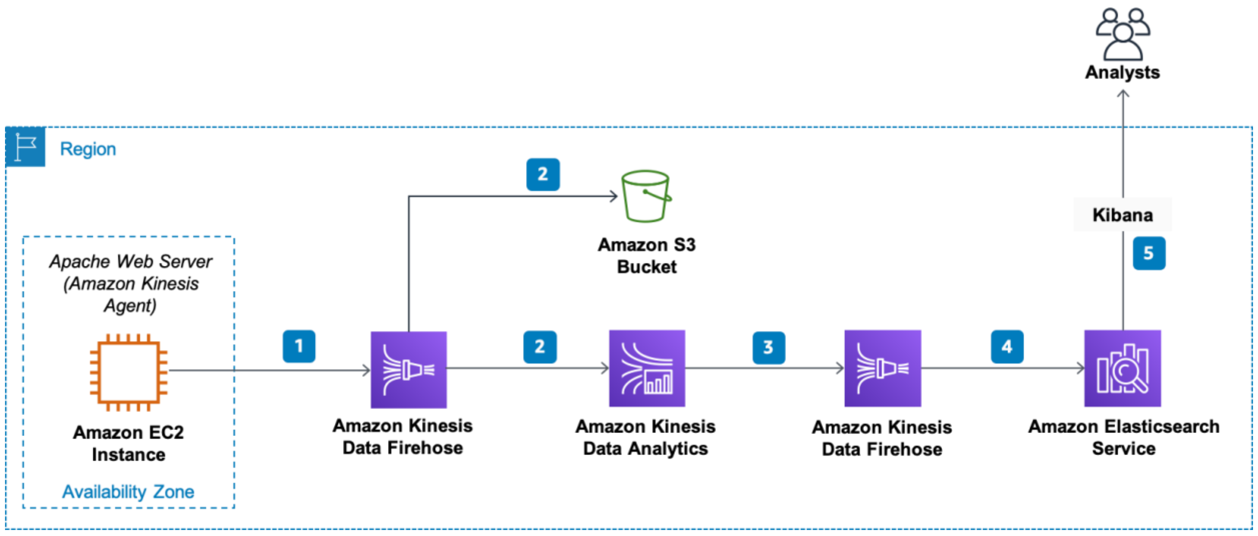

Analysts
Region
2
Amazon S3 Bucket
Kibana
5
Apache Web Server (Amazon Kinesis Agent)
Amazon EC2 Instance
Availability Zone
1
Amazon Kinesis Data Firehose
2
Amazon Kinesis Data Analytics
3
Amazon Kinesis Data Firehose
4
Amazon Elasticsearch Service

Figure 1: Log analytics solution architecture

In this architecture example, the web server is an Amazon Elastic Compute Cloud (Amazon EC2) instance. You install the Amazon Kinesis Agent on this Linux instance.

1. The Kinesis Agent continuously forward log records to an Amazon Kinesis Data Firehose delivery stream.
2. Amazon Kinesis Data Firehose writes each log record to Amazon Simple Storage Service (Amazon S3) for durable storage of the raw log data. Amazon Kinesis Data Analytics continuously runs a SQL statement against the streaming input data.
3. Amazon Kinesis Analytics creates an aggregated data set every minute and outputs that data to a second Kinesis Data Firehose delivery stream.
4. This Firehose delivery stream writes the aggregated data to an Amazon Elasticsearch Service domain.
5. You create a view of the streaming data using Kibana to visualize the output of your system.

Estimate Your Costs

The total cost of analyzing your Apache access logs varies depending on several factors, including the following:

- how many web log records you ingest
- the complexity of your Amazon Kinesis Analytics SQL queries
- the instance size, storage choice, and redundancy chosen for the Amazon ES domain

This tutorial also creates an EC2 instance to generate a sample Apache access log. The instance size you choose and the amount of time that the instance is running affects the cost.

It costs approximately $0.51 to complete the tutorial if you use the default configuration recommended in this guide. This estimate assumes that the infrastructure you create during the tutorial is running for 1 hour. A breakdown of the services used and their associated costs is provided in the following section.

Services Used and Costs

AWS pricing is based on your usage of each individual service. The total combined usage of each service creates your monthly bill. For this tutorial, you are charged for the use of Amazon EC2, Amazon Kinesis Data Firehose, Amazon S3, Amazon Kinesis Data Analytics, and Amazon Elasticsearch Service.

Amazon EC2

Description: Amazon EC2 provides the virtual application servers, known as instances, to run your web application on the platform you choose. Amazon EC2 allows you to configure and scale your compute capacity easily to meet changing requirements and demand. It is integrated into Amazon's computing environment, allowing you to leverage the AWS suite of services.

How Pricing Works: Amazon EC2 pricing is based on four components: the instance type you choose (EC2 comes in 40+ types of instances with options optimized for compute, memory, storage, and more), the AWS Region your instances are based in, the software you run, and the pricing model you select (on-demand instances, reserved capacity, spot, etc.). For more information, see Amazon EC2 pricing.

Example: Assume your log files reside on a single Linux t2.micro EC2 instance in the US East region. With an on-demand pricing model, the monthly charge for your virtual machine is $4.18. For this tutorial, assuming that the log generating instance runs for 1 hour, your EC2 cost is estimated to be **$0.0116** [= ($8.35 per month / 30 days per month / 24 hours per day) * 1 hour].

Amazon Kinesis Data Firehose

Description: Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift, or Amazon Elasticsearch Service. With Kinesis Data Firehose, you do not need to write any applications or manage any resources. You configure your data producers to send data to Firehose and it automatically delivers the data to the destination that you specified.

How Pricing Works: Amazon Kinesis Data Firehose pricing is based on the volume of data ingested into Amazon Kinesis Data Firehose, which is calculated as the number of data records you send to the service, times the size of each record, rounded up to the nearest 5 KB. For example, if your data records are 42 KB each, Amazon Kinesis Data Firehose counts each record as 45 KB of data ingested. In the US East AWS Region, the price for Amazon Kinesis Data Firehose is $0.029 per GB of data ingested and decreases as data total increases. For more information, see Amazon Kinesis Firehose Pricing.

Example: In this tutorial, you create two separate Amazon Kinesis Data Firehose delivery streams. One delivery stream receives the data from your Apache access log producer, and the other delivery stream receives the output from an Amazon Kinesis Data Analytics application. Assume the producer sends 500 records per second, and that each record is less than 5 KB in size (typical for an Apache access log record). The monthly estimate for data ingestion into the Kinesis Data Firehose delivery stream in this example is:

- The price in the US East region is $0.029 per GB of data ingested.
- Record size, rounded up to the nearest 5 KB = 5 KB
- Data ingested (GB per sec) = (500 records/sec * 5 KB/record) / 1,048,576 KB/GB = 0.002384 GB/sec
- Data ingested (GB per month) = 30 days/month * 86,400 sec/day * 0.002384 GB/sec = 6,179.81 GB/month
- Monthly charge: 6,179.81 * $0.029/GB = $179.21

For this tutorial, assume that the system is only ingesting data for 1 hour. The cost specifically for this tutorial would be approximately **$0.25** [= ($179.21 per month / 30 days per month / 24 hours per day) * 1 hour].

The second Kinesis Data Firehose delivery stream is receiving records at a much less frequent rate. Because the Amazon Kinesis Data Analytics application is outputting only a few rows of data every minute, the cost for that delivery stream is correspondingly smaller. Assuming only five records per minute are ingested, and each record is less than 5 KB, the cost for the delivery stream is **$0.00005** for the 1-hour duration assumed for this tutorial.

Amazon S3

Description: Amazon S3 provides secure, durable, and highly-scalable cloud storage for the objects that make up your application. Examples of objects you can store include source code, logs, images, videos, and other artifacts that are created when you deploy your application. Amazon S3 makes it is easy to use object storage with a simple web interface to store and retrieve your files from anywhere on the web, meaning that your website will be reliably available to your visitors.

How Pricing Works: Amazon S3 pricing is based on five components: the type of Amazon S3 storage you use, the region you store your WordPress content (e.g., US East vs. Asia Pacific - Sydney), the amount of data you store, the number of requests you or your users make to store new content or retrieve the content, and the amount of data that is transferred from Amazon S3 to you or your users. For more information, see Amazon S3 Pricing.

Example: Using Standard Storage in the US East Region, if you store 5 GB of content, you pay $0.115 per month. If you created your account in the past 12 months, and you are eligible for the AWS Free Tier, you pay $0.00 per month. For this tutorial, assume that the producer creates 5 GB of data. Over a 1-hour period, the total cost for storing the records in Amazon S3 is **$0.00016/** [= ($0.115 per month / 30 days per month / 24 hours per day) * 1 hour].

Amazon Kinesis Data Analytics

Description: Amazon Kinesis Data Analytics is the easiest way to process and analyze streaming data in real time with ANSI standard SQL. It enables you to read data from Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose, and build stream processing queries that filter, transform, and aggregate the data as it arrives. Amazon Kinesis Data Analytics automatically recognizes standard data formats, parses the data, and suggests a schema, which you can edit using the interactive schema editor. It provides an interactive SQL editor and stream processing templates so you can write sophisticated stream processing queries in just minutes. Amazon Kinesis Data Analytics runs your queries continuously, and writes the processed results to output destinations such as Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose, which can deliver the data to Amazon S3, Amazon Redshift, and Amazon Elasticsearch Service. Amazon Kinesis Data Analytics automatically provisions, deploys, and scales the resources required to run your queries.

How Pricing Works: With Amazon Kinesis Data Analytics, you pay only for what you use. You are charged an hourly rate based on the average number of Kinesis Processing Units (KPUs) used to run your stream processing application.

A single KPU is a unit of stream processing capacity comprised of 4 GB memory, 1 vCPU compute, and corresponding networking capabilities. As the complexity of your queries varies, and the demands on memory and compute vary in response, Amazon Kinesis Data Analytics automatically and elastically scales the number of KPUs required to complete your analysis. There are no resources to provision and no upfront costs or minimum fees associated with Amazon Kinesis Analytics. For more information, see Amazon Kinesis Data Analytics Pricing.

Example: This example assumes that the system is running for 1 hour in the US East Region. The SQL query in this tutorial is basic and does not consume more than one KPU. Given that the price for Amazon Kinesis Data Analytics in US East is $0.11 per KPU-hour, and the tutorial runs for 1 hour, the total cost for the usage of Amazon Kinesis Data Analytics is **$0.11**.

Amazon Elasticsearch Service

Description: Amazon Elasticsearch Service (Amazon ES) is a popular open-source search and analytics engine for big data use cases such as log and click stream analysis. Amazon Elasticsearch Service manages the capacity, scaling, patching, and administration of Elasticsearch clusters for you while giving you direct access to the Elasticsearch API.

How Pricing Works: With Amazon Elasticsearch Service, you pay only for what you use. There are no minimum fees or upfront commitments. You are charged for Elasticsearch Service instance hours, an Amazon Elastic Block Store (Amazon EBS) volume (if you choose this option), and standard data transfer fees. For more information, see Amazon Elasticsearch Service Pricing.

Example: For this tutorial, the total Elasticsearch Service cost can be calculated as follows:

In the **US East (N. Virginia)** Region, an instance type of **m3.medium**.elasticsearch costs $0.094 per hour * 1 hour = **$0.094**.

Tutorial

You can evaluate the simplicity and effectiveness of Amazon Kinesis Data Analytics with this tutorial, which walks you through a simple Amazon Kinesis Data Analytics application.

You perform the following steps in this tutorial:

Step 1: Set Up Prerequisites

Step 2: Create an Amazon Kinesis Data Firehose Delivery Stream

Step 3: Install and Configure the Amazon Kinesis Agent on the EC2 Instance

This tutorial is not meant for production environments and does not discuss options in depth. After you complete the steps, you can find more in-depth information to create your own Amazon Kinesis Data Analytics application in the Additional Resources section.

Step 1: Set Up Prerequisites

Before you begin analyzing your Apache access logs with Amazon Kinesis Data Analytics, make sure you complete the following prerequisites.

This tutorial assumes that the AWS resources have been created in the US East AWS Region (us-east-1).

Create an AWS Account

If you already have an AWS account, you can skip this prerequisite and use your existing account. To create AWS account:

1. Go to http://aws.amazon.com/.
2. Choose **Create an AWS Account** and follow the instructions.

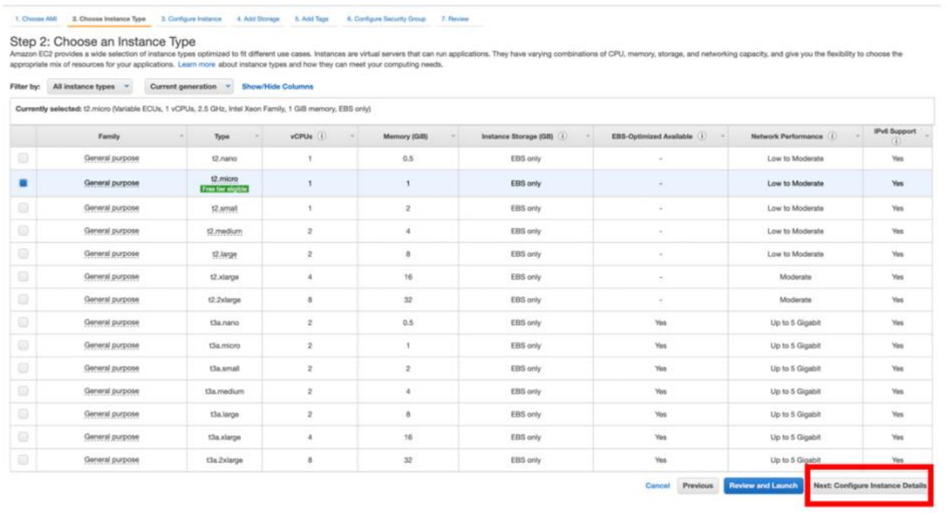
Part of the signup procedure involves receiving a phone call and entering a PIN using the phone keypad.

Start an EC2 Instance

The steps outlined in this tutorial assume that you are using an EC2 instance as the web server and log producer. (For detailed instructions, see Getting started with Amazon EC2 Linux instances.)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/
2. From the console dashboard, choose **Launch Instance**.

3. On the **Choose an Amazon Machine Image (AMI)** page, choose **Amazon Linux AMI**.
4. On the **Choose an Instance Type** page, select the **t2.micro** instance type.



Step 2: Choose an Instance Type

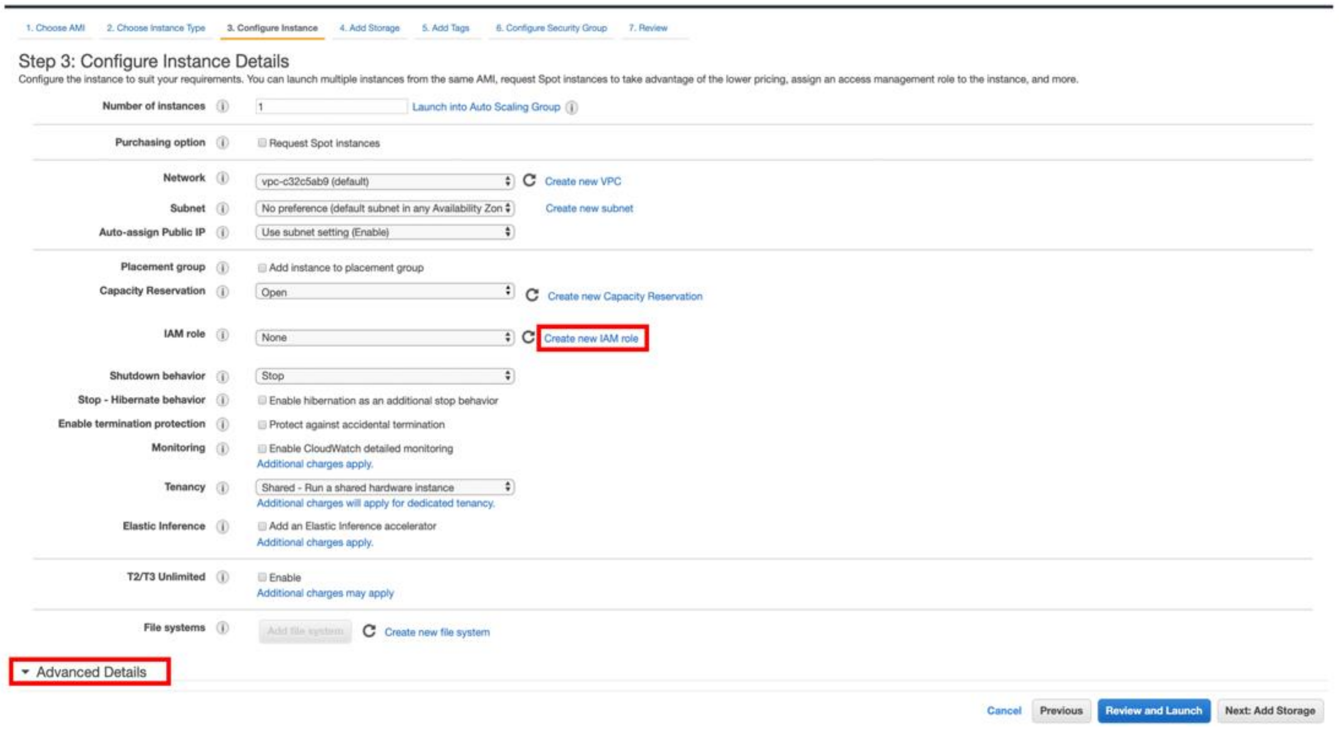
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
☐	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
☐	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.xlarge	4	16	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.2xlarge	8	32	EBS only	Yes	Up to 5 Gigabit	Yes

Figure 2: Choose an instance type

5. Choose **Next: Configure Instance Details**.
6. Choose **Create new IAM role**. A new tab opens to create the role.



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1 Launch into Auto Scaling Group
Purchasing option	Request Spot instances
Network	vpc-c32c5ab9 (default) Create new VPC
Subnet	No preference (default subnet in any Availability Zon Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	Add instance to placement group
Capacity Reservation	Open Create new Capacity Reservation
IAM role	None Create new IAM role
Shutdown behavior	Stop
Stop - Hibernate behavior	Enable hibernation as an additional stop behavior
Enable termination protection	Protect against accidental termination
Monitoring	Enable CloudWatch detailed monitoring Additional charges apply.
Tenancy	Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.
Elastic Inference	Add an Elastic Inference accelerator Additional charges apply.
T2/T3 Unlimited	Enable Additional charges may apply
File systems	Add file system Create new file system

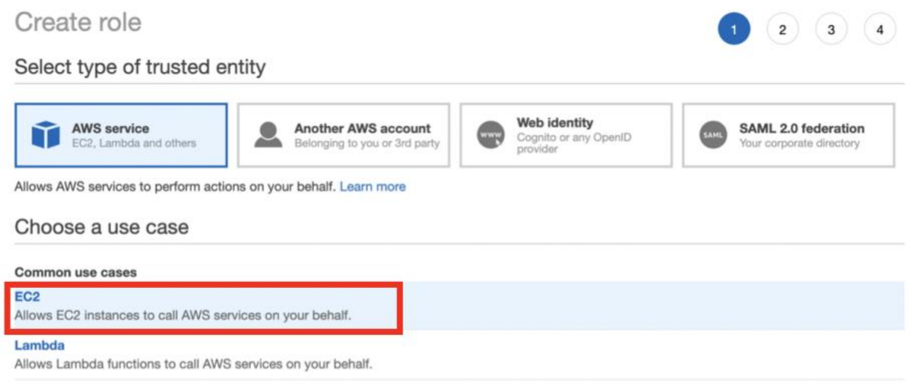
Figure 3: Configure instance details

You want to ensure that your EC2 instance has an AWS Identity and Access Management (IAM) role configured with permission to write to Amazon Kinesis Data Firehose and Amazon CloudWatch. For more information, see [IAM Roles for Amazon EC2.]()

a. Choose **Create role**.

b. For trusted entity, choose **AWS service**.

c. For the use case, choose **EC2**.



Select type of trusted entity

Allows AWS services to perform actions on your behalf. Learn more

Choose a use case

Common use cases

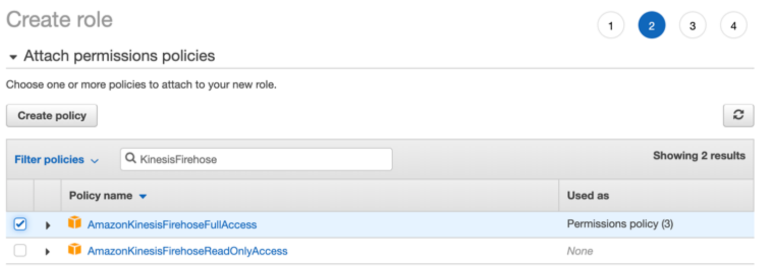
EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

Figure 4: Create new IAM role

d. Choose **Next: Permissions**.

e. In the search bar, type *KinesisFirehose* and select the check box for **AmazonKinesisFirehoseFullAccess**.



Create role

Attach permissions policies

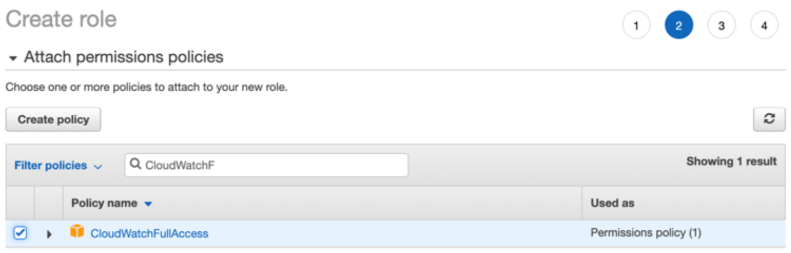
Choose one or more policies to attach to your new role.

Filter policies: KinesisFirehose — Showing 2 results

	Policy name	Used as
✓	AmazonKinesisFirehoseFullAccess	Permissions policy (3)
	AmazonKinesisFirehoseReadOnlyAccess	None

Figure 5: Add AmazonKinesisFirehoseFullAccess policy

7. Clear the search bar and type *CloudWatchFull*. Select the check box for **CloudWatchFullAccess***.*



Create role

Attach permissions policies

Choose one or more policies to attach to your new role.

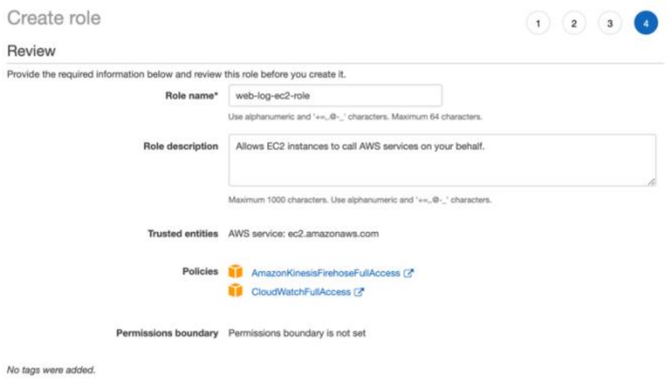
Filter policies: CloudWatchF — Showing 1 result

	Policy name	Used as
✓	CloudWatchFullAccess	Permissions policy (1)

Figure 6: Add CloudWatchFullAccess policy

f. Choose **Next: Tags** to add optional tags.

g. Choose **Next: Review** and for **Role name**, type *web-log-ec2-role.*



Create role

Review

Provide the required information below and review this role before you create it.

Role name*	web-log-ec2-role
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	Allows EC2 instances to call AWS services on your behalf.
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	AWS service: ec2.amazonaws.com
Policies	AmazonKinesisFirehoseFullAccess CloudWatchFullAccess
Permissions boundary	Permissions boundary is not set

No tags were added.

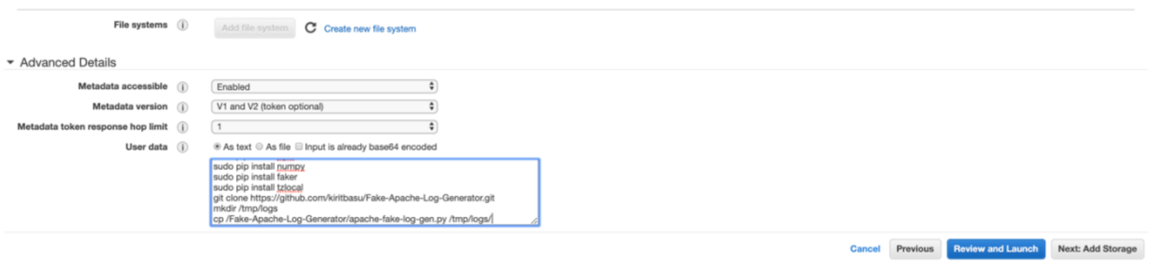
h. Choose **Create role**.

8. Return to the EC2 launch wizard tab (*Figure 3*) and next to the **IAM role**, click the *refresh* icon*.* Then, select **web-log-ec2-role**.

9. Choose **Advanced Details** and fill out the **User data** field:

   To prepare your EC2 instance, copy and paste the following user data script into the **User data** space. Make sure the lines are single-spaced with no extra whitespace in between.

```
#!/bin/bash
sudo yum update -y
sudo yum install git -y
sudo easy_install pip
sudo pip install pytz
sudo pip install numpy
sudo pip install faker
sudo pip install tzlocal
git clone https://github.com/kiritbasu/Fake-Apache-Log-
Generator.git
mkdir /tmp/logs
cp /Fake-Apache-Log-Generator/apache-fake-log-gen.py /tmp/logs/
```

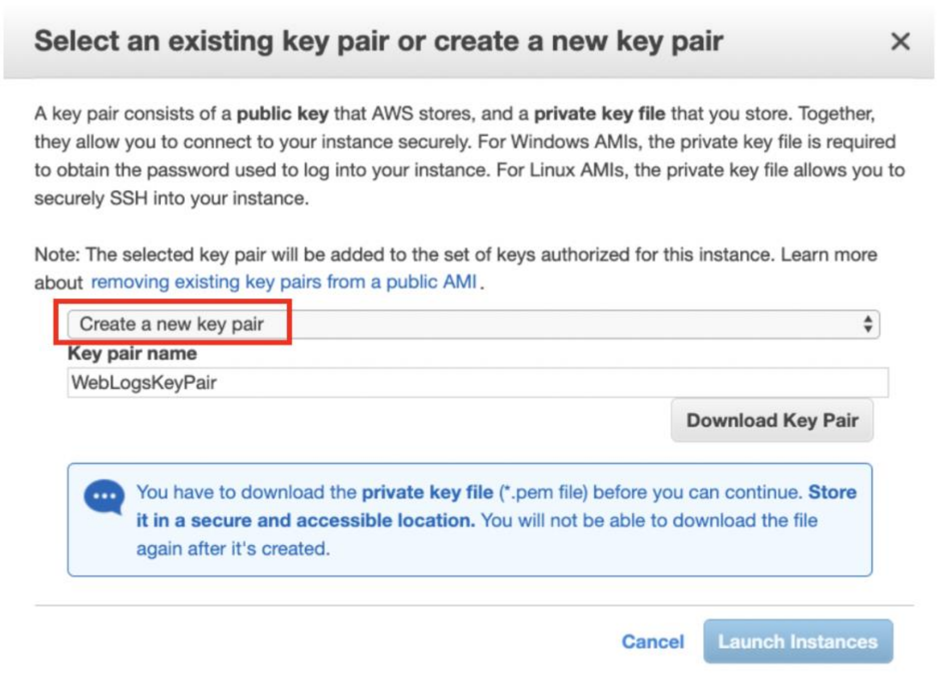


Metadata accessible: Enabled
Metadata version: V1 and V2 (token optional)
Metadata token response hop limit: 1
User data: [✓ As text] As file ☐ Input is already base64 encoded

```
sudo pip install numpy
sudo pip install faker
sudo pip install tzlocal
git clone https://github.com/kiritbasu/Fake-Apache-Log-Generator.git
mkdir /tmp/logs
cp /Fake-Apache-Log-Generator/apache-fake-log-gen.py /tmp/logs/
```

Figure 7: Advanced Details – User data field

10. Choose **Review and Launch**
11. Review the details and choose **Launch.**
12. In the key pair dialog box that appears, choose to create a new key pair or select an existing key pair that you have access to. If you choose to create a new key pair, choose **Download Key Pair** and wait for the file to download.


Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

Key pair name

WebLogsKeyPair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Figure 8: Create a new key pair

13. Choose **Launch Instances**.

The EC2 instance launches with the required dependencies already installed on the machine. The user data script clones Github onto the EC2 instance and the required file is copied into a new directory named `logs` in the `/tmp` folder. Once the EC2 instance is launched, you need to connect to it via SSH.

Prepare Your Log Files

Because Amazon Kinesis Data Analytics can analyze your streaming data in near real time, this tutorial is much more effective when you use a live stream of Apache access log data. If your EC2 instance is not serving HTTP traffic, you need to generate continuous sample log files.

To create a continuous stream of log file data on your EC2 instance, download, install, and run the Fake Apache Log Generator from Github on the EC2 instance. Follow the instructions on the project page and configure the script for infinite log file generation.

Connect to Your Instance

To connect to your instance, follow the steps in Connect to Your Linux Instance.

You can also select the instance and choose **Connect** to view further guidance on how to connect to the instance.

Once you connect into the EC2 instance, run the following line of code to start the Fake Apache Log Generator program. Run this line of code multiple times to create multiple log files within the `/tmp/logs` file.

```
sudo python /tmp/logs/apache-fake-log-gen.py -n 0 -o LOG &
```

Take note of the path to the log file. You need this information later in this tutorial.

Step 2: Create an Amazon Kinesis Data Firehose Delivery Stream

In Step 1, you created log files on your web server. Before they can be analyzed with Amazon Kinesis Data Analytics (Step 6), you must first load the log data into AWS. Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, or Amazon Elasticsearch Service.

In this step, you create an Amazon Kinesis Data Firehose delivery stream to save each log entry in Amazon S3 and to provide the log data to the Amazon Kinesis Data Analytics application that you create later in this tutorial.

To create the Amazon Kinesis Data Firehose delivery stream:

14. Open the Amazon Kinesis console at https://console.aws.amazon.com/kinesis.
15. In the **Get Started** section, choose **Kinesis Data Firehose**, and then choose **Create Delivery Stream**.
16. On the **Name and source** screen:
    a. For **Delivery stream name**, enter *web-log-ingestion-stream.*
    b. For **Choose a source**, select **Direct PUT or other sources**.
    c. Choose **Next**.
17. On the **Process records** screen, keep the default selections and choose **Next.**
18. On the **Choose a destination** screen:
    a. For **Destination**, choose **Amazon S3**.

b. For **S3 bucket**, choose **Create new**.

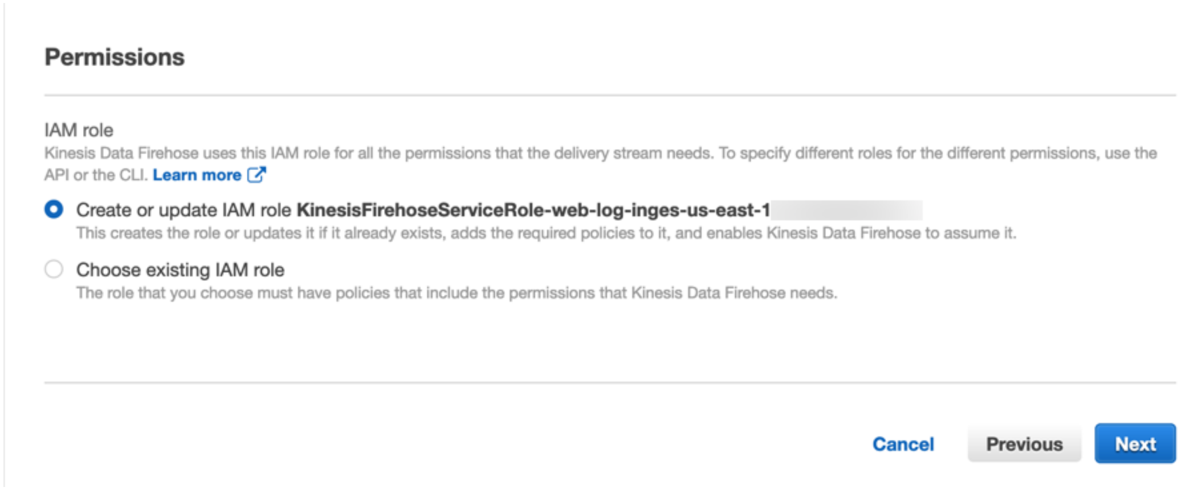
c. In the **Create S3 bucket** window, for **S3 bucket name**, specify a unique name. You do not need to use the name elsewhere in this tutorial. However, Amazon S3 bucket names are required to be globally unique.

d. For **Region**, choose **US East (N. Virginia)**.

e. Choose **Create S3 Bucket**.

19. Choose **Next**.

20. On the **Configure settings** screen, scroll down to **Permissions**, and for **IAM role**, choose **Create or update IAM role**.



Permissions

IAM role
Kinesis Data Firehose uses this IAM role for all the permissions that the delivery stream needs. To specify different roles for the different permissions, use the API or the CLI. Learn more

[✓ Create or update IAM role **KinesisFirehoseServiceRole-web-log-inges-us-east-1**]
This creates the role or updates it if it already exists, adds the required policies to it, and enables Kinesis Data Firehose to assume it.

Choose existing IAM role
The role that you choose must have policies that include the permissions that Kinesis Data Firehose needs.

Figure 9: Permissions – IAM role settings

21. Choose **Next**.

22. Review the details of the Amazon Kinesis Data Firehose delivery stream and choose **Create Delivery Stream**.

Step 3: Install and Configure the Amazon Kinesis Agent on the EC2 Instance

Now that you have an Amazon Kinesis Firehose delivery stream ready to ingest your data, you can configure the EC2 instance to send the data using the Amazon Kinesis Agent software. The agent is a standalone Java software application that offers an easy way to collect and send data to Kinesis Data Firehose. The agent continuously monitors a set of files and sends new data to your delivery stream. It handles file rotation, checkpointing, and retry upon failures. It delivers all of your data in a reliable, timely, and simple manner. It also emits Amazon CloudWatch metrics to help you better monitor and troubleshoot the streaming process.

The Amazon Kinesis Agent can preprocess records from monitored files before sending them to your delivery stream. It has native support for Apache access log files, which you created in Step 1. When configured, the agent parses log files in the Apache Common Log format and convert each line in the file to JSON format before sending the files to your Kinesis Data Firehose delivery stream, which you created in Step 2.

1. To install the agent, copy and paste the following command. For more information, see Download and Install the Agent.

```
sudo yum install –y aws-kinesis-agent
```

2. For detailed instructions on how to configure the agent to process and send log data to your Amazon Kinesis Data Firehose delivery stream, see Configure and Start the Agent.

   To configure the agent for this tutorial, modify the configuration file located at **/etc/aws-kinesis/agent.json** using the following template.

   - Replace `filePattern` with the full-path-to-log-file that represents the path to your log files and a wildcard if you have multiple log files with the same naming convention. For example, it might look similar to: `“/tmp/logs/access_log*”`. The value will be different, depending on your use case.
   - Replace `name-of-delivery-stream` with the name of the Kinesis Data Firehose delivery stream you created in Step 2.
   - The `firehose.endpoint` is `firehose.us-east-1.amazonaws.com` (default).

```
"firehose.endpoint": "firehose.us-east-1.amazonaws.com",
"flows": [
    {
        "filePattern": "/tmp/logs/access_log*",
        "deliveryStream": "name-of-delivery-stream",
        "dataProcessingOptions": [
        {
            "optionName": "LOGTOJSON",
            "LogFormat": "COMMONAPACHELOG"
        }]
    }
]
```

3. Start the agent manually by issuing the following command:

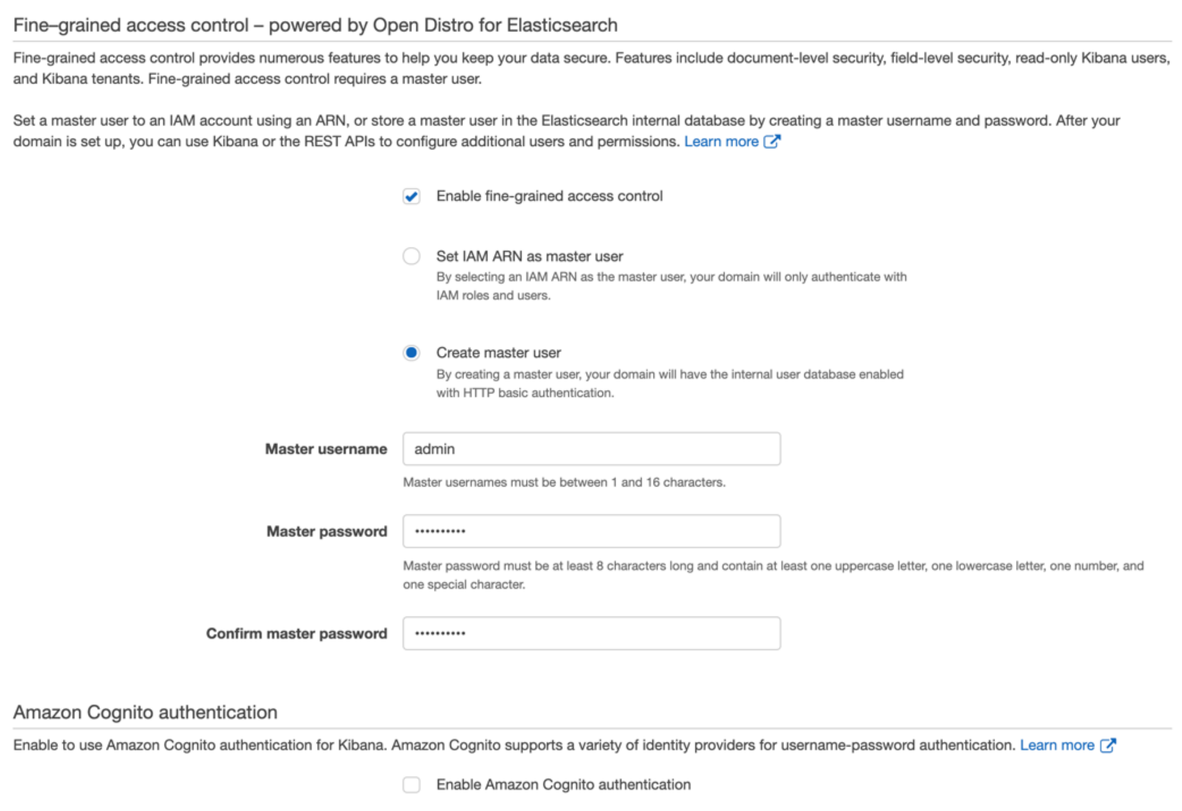
```
sudo service aws-kinesis-agent start
```

Once started, the agent looks for files in the configured location and send the records to the Kinesis Data Firehose delivery stream.

Step 4: Create an Amazon Elasticsearch Service Domain

The data produced by this tutorial is stored in Amazon Elasticsearch Service for later visualization and analysis. To create the Amazon Elasticsearch Service domain:

1. Open the Amazon Elasticsearch Service console at https://console.aws.amazon.com/es.
2. Choose **Create a new domain**.
3. On the **Choose deployment type** page, for **Deployment type**, choose a **Development and testing**. For **Elasticsearch version**, leave it set to the default value.
4. Choose **Next**.
5. On the **Configure domain** page, for **Elasticsearch domain name**, type *web-log-summary*.
6. Leave all settings as their default values and choose **Next.**
7. On the **Configure access and security page**:
   a. For **Network configuration**, choose **Public access**.
   b. Under **Fine-grained access control**, make sure **Enable fine-grained access control** is selected.
   c. Choose **Create master user**, and specify the **Master username** as ***admin*** and set a password.



Fine–grained access control – powered by Open Distro for Elasticsearch

Fine-grained access control provides numerous features to help you keep your data secure. Features include document-level security, field-level security, read-only Kibana users, and Kibana tenants. Fine-grained access control requires a master user.

Set a master user to an IAM account using an ARN, or store a master user in the Elasticsearch internal database by creating a master username and password. After your domain is set up, you can use Kibana or the REST APIs to configure additional users and permissions. Learn more

[✓] Enable fine-grained access control

( ) Set IAM ARN as master user
By selecting an IAM ARN as the master user, your domain will only authenticate with IAM roles and users.

(●) Create master user
By creating a master user, your domain will have the internal user database enabled with HTTP basic authentication.

Master username: admin
Master usernames must be between 1 and 16 characters.

Master password: ••••••••••
Master password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, one number, and one special character.

Confirm master password: ••••••••••

Amazon Cognito authentication

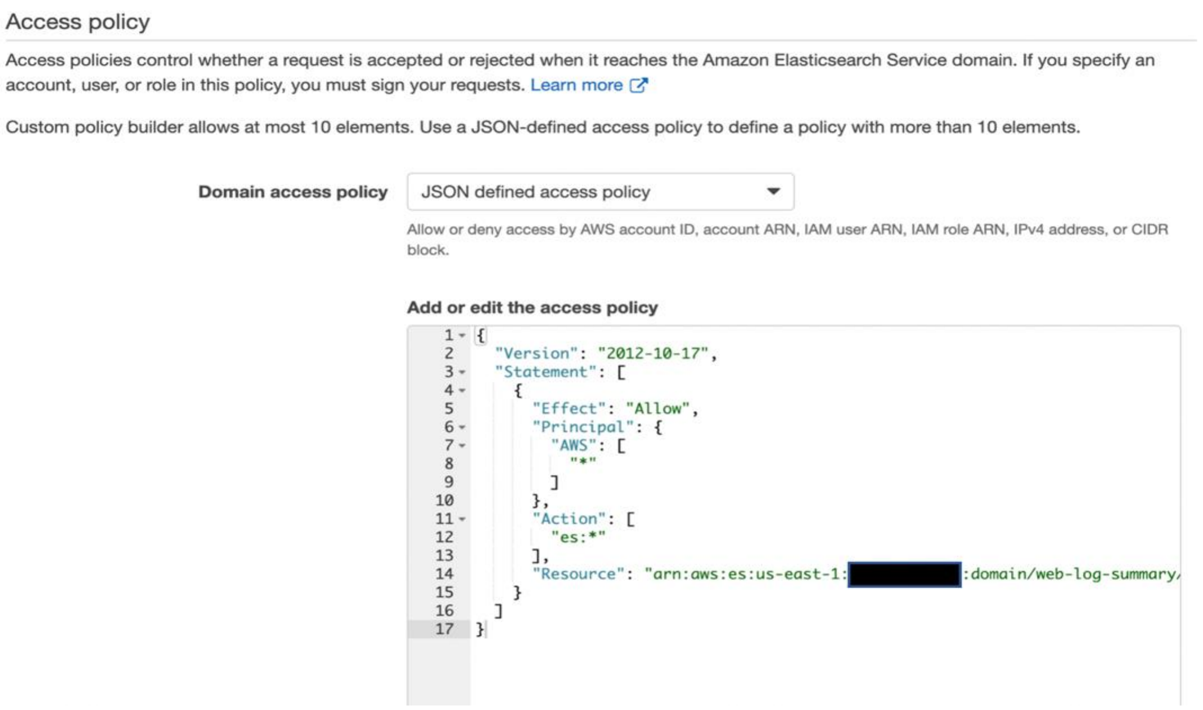
Enable to use Amazon Cognito authentication for Kibana. Amazon Cognito supports a variety of identity providers for username-password authentication. Learn more

[ ] Enable Amazon Cognito authentication

Figure 10: Fine-grained access control options

d. In the **Access policy** section, for **Domain access policy**, choose **JSON defined access policy**. Your JSON policy should look like the one shown in Figure 11.

Note: This is not a recommended setting for production Amazon Elasticsearch Service domains. Make sure to terminate this Amazon Elasticsearch Service domain after completing the tutorial or apply a more restrictive policy.



Access policy

Access policies control whether a request is accepted or rejected when it reaches the Amazon Elasticsearch Service domain. If you specify an account, user, or role in this policy, you must sign your requests. Learn more

Custom policy builder allows at most 10 elements. Use a JSON-defined access policy to define a policy with more than 10 elements.

Domain access policy JSON defined access policy

Allow or deny access by AWS account ID, account ARN, IAM user ARN, IAM role ARN, IPv4 address, or CIDR block.

Add or edit the access policy

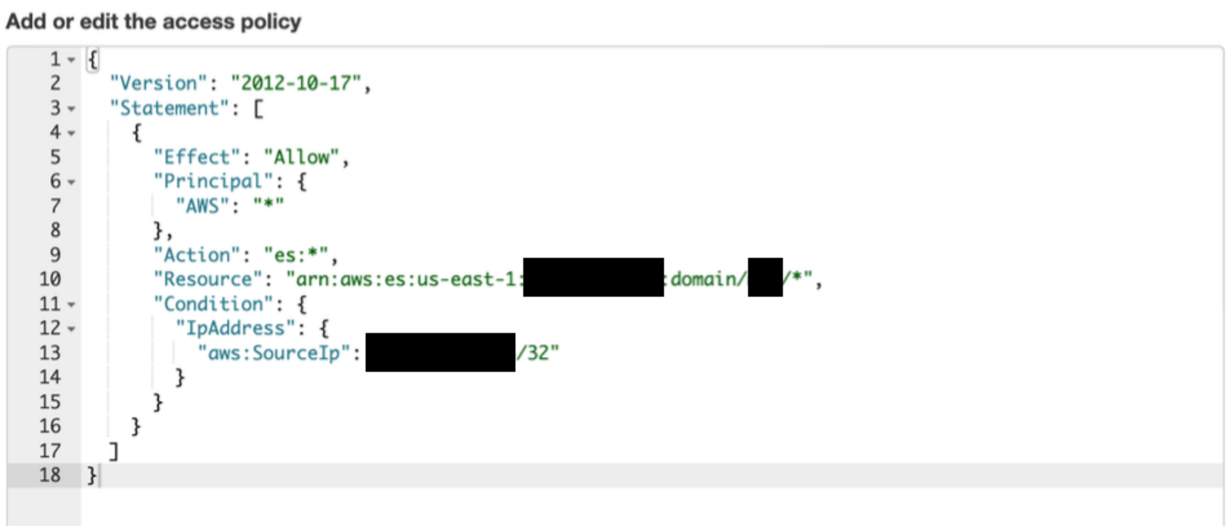
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "*"
        ]
      },
      "Action": [
        "es:*"
      ],
      "Resource": "arn:aws:es:us-east-1:[redacted]:domain/web-log-summary/
    }
  ]
}
```

Figure 11: Access policy settings

8. Leave all other default settings and choose **Next**.
9. Review the details for the Amazon Elasticsearch Service domain and choose **Confirm**.

   It takes approximately 10 minutes for the Amazon Elasticsearch Service domain to be created. While the domain is being created, proceed with the remainder of this guide.

Note: The following example shows a restrictive access policy where the domain is restricted to only allow traffic from a specific IP address.



Add or edit the access policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:[redacted]:domain/[redacted]/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [redacted]/32"
        }
      }
    }
  ]
}
```

Figure 12: Example restrictive access policy

Step 5: Create a Second Amazon Kinesis Data Firehose Delivery Stream

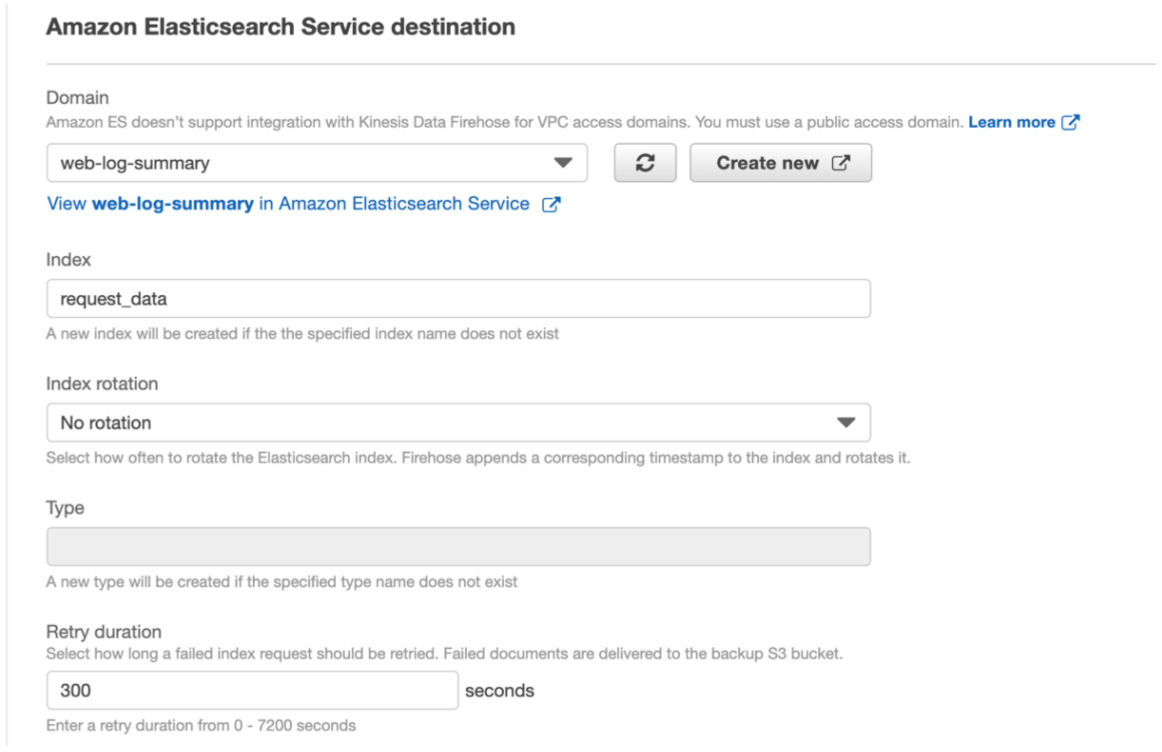
Now that you have somewhere to persist the output of your Amazon Kinesis Data Analytics application, you need a simple way to get your data into your Amazon ES domain. Amazon Kinesis Data Firehose supports Amazon ES as a destination, so create a second Firehose delivery stream:

1. Open the Amazon Kinesis console at [https://console.aws.amazon.com/kinesis.](https://console.aws.amazon.com/kinesis)
2. Choose **Create Delivery Stream**.
3. On the **Name and source** page:
   a. For **Delivery stream name**, enter *web-log-aggregated-data*.
   b. For **Choose a source**, select **Direct PUT or other sources** and choose **Next.**
4. On the **Process records** screen, leave the default values and choose **Next**.
5. On the **Choose a destination** page:
   a. For **Destination**, choose **Amazon Elasticsearch Service**.
   b. For Elasticsearch domain, choose the domain you created in Step 4. (You may need to wait until the Amazon ES domain has finished processing. Click the refresh button periodically to refresh the list of domains).

c. For **Index**, type *request_data.*

d. For **Index rotation**, choose **No rotation** (default).

e. For **Retry duration**, leave the default value of 300 seconds.



Amazon Elasticsearch Service destination

Domain
Amazon ES doesn't support integration with Kinesis Data Firehose for VPC access domains. You must use a public access domain. Learn more

web-log-summary

View web-log-summary in Amazon Elasticsearch Service

Index
request_data
A new index will be created if the the specified index name does not exist

Index rotation
No rotation
Select how often to rotate the Elasticsearch index. Firehose appends a corresponding timestamp to the index and rotates it.

Type
A new type will be created if the specified type name does not exist

Retry duration
Select how long a failed index request should be retried. Failed documents are delivered to the backup S3 bucket.
300 seconds
Enter a retry duration from 0 - 7200 seconds

Figure 13: Amazon Elasticsearch Service destination settings

f. In the **S3 backup** section, for **Backup mode**, choose **Failed Records Only**.

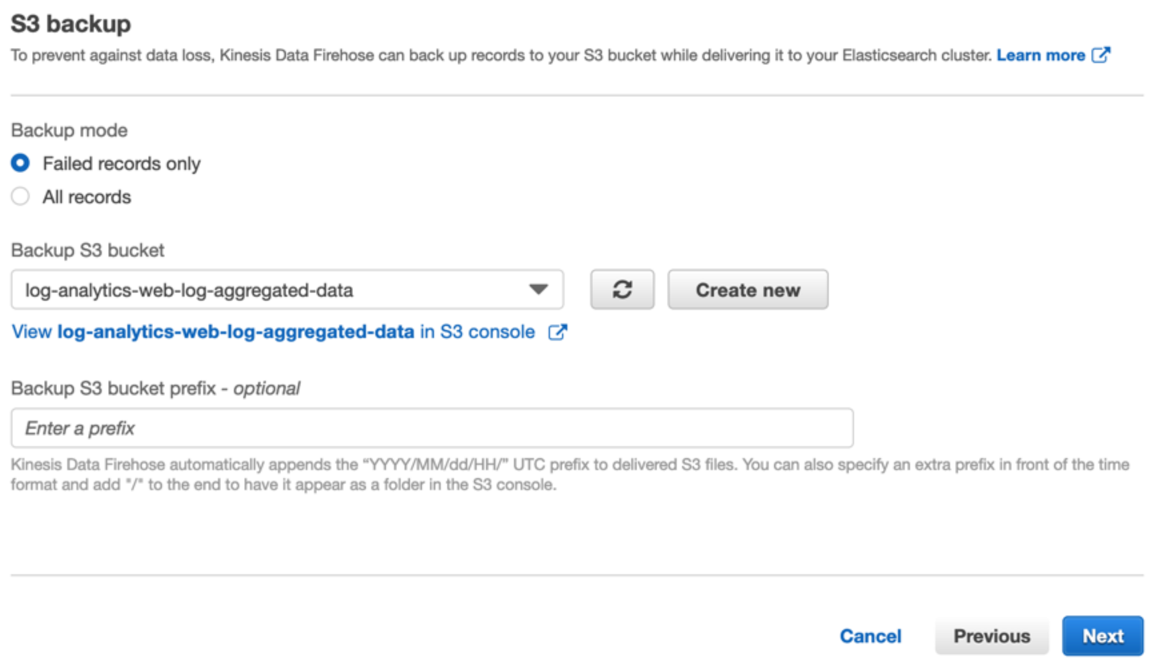
g. For **S3 bucket**, choose **Create new**.

h. In the **Create S3 bucket** window, for **S3 bucket name**, specify a unique name. You do not need to use the name elsewhere in this tutorial. However, Amazon S3 bucket names are required to be globally unique.

i. For **Region**, choose **US East (N. Virginia)**.

j. Choose **Create S3 Bucket**.

6. Choose **Next**.



S3 backup

To prevent against data loss, Kinesis Data Firehose can back up records to your S3 bucket while delivering it to your Elasticsearch cluster. Learn more

Backup mode

● Failed records only
○ All records

Backup S3 bucket

log-analytics-web-log-aggregated-data

View **log-analytics-web-log-aggregated-data** in S3 console

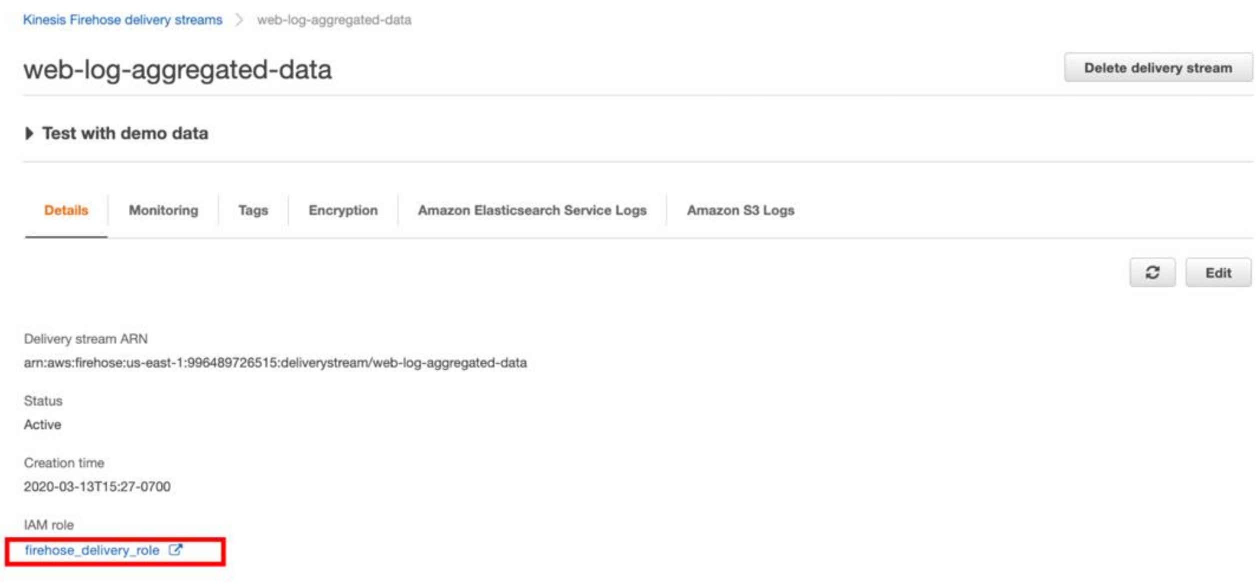
Backup S3 bucket prefix - *optional*

Enter a prefix

Kinesis Data Firehose automatically appends the "YYYY/MM/dd/HH/" UTC prefix to delivered S3 files. You can also specify an extra prefix in front of the time format and add "/" to the end to have it appear as a folder in the S3 console.

Figure 14: S3 backup settings

7. On the **Configure settings** screen, you can leave all fields set to their default values. However, you will need to choose an IAM role so that Amazon Kinesis Firehose can write to your Amazon ES domain on your behalf. For **IAM role**, choose **Create new or choose**.
8. Choose **Next**.
9. Review the details for your Amazon Kinesis Data Firehose delivery stream and choose **Create Delivery Stream**.
10. Add permissions for your Kinesis Data Firehose delivery stream to access your Elasticsearch Service cluster:
    a. Select the newly created **web-log-aggregated-data** stream and choose the **IAM role**.



Kinesis Firehose delivery streams > web-log-aggregated-data

web-log-aggregated-data

Delete delivery stream

Test with demo data

Details | Monitoring | Tags | Encryption | Amazon Elasticsearch Service Logs | Amazon S3 Logs

Edit

Delivery stream ARN
arn:aws:firehose:us-east-1:996489726515:deliverystream/web-log-aggregated-data

Status
Active

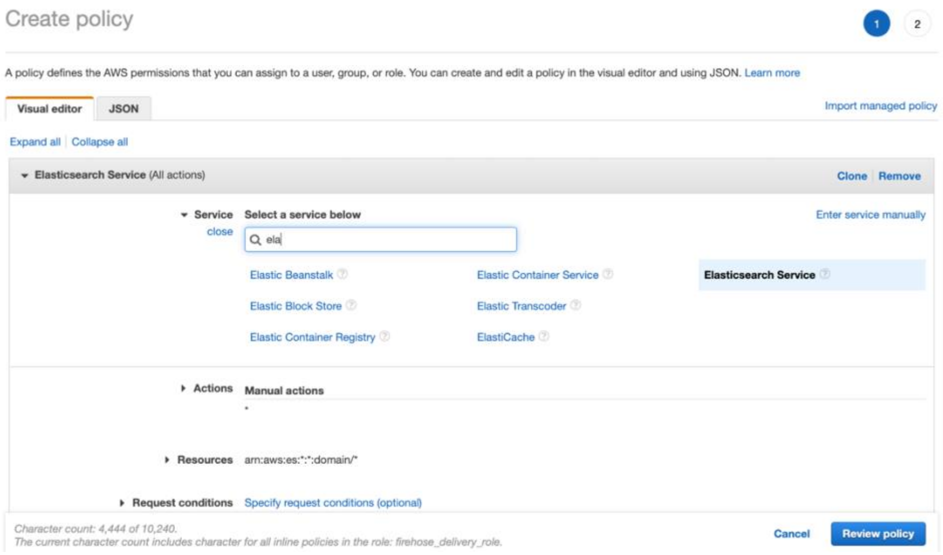
Creation time
2020-03-13T15:27-0700

IAM role
firehose_delivery_role

Figure 15: Web-log-aggregated-data stream IAM role

b. In the IAM window that opens, choose **Add inline policy**.

c. On the **Create policy** page, choose **Choose a service** and search for *Elasticsearch* in the search box.



Create policy

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor | JSON | Import managed policy

Expand all | Collapse all

Elasticsearch Service (All actions) — Clone | Remove

Service: Select a service below — close — Enter service manually

ela

Elastic Beanstalk | Elastic Container Service | Elasticsearch Service
Elastic Block Store | Elastic Transcoder
Elastic Container Registry | ElastiCache

Actions: Manual actions
*

Resources: arn:aws:es:*:*:domain/*

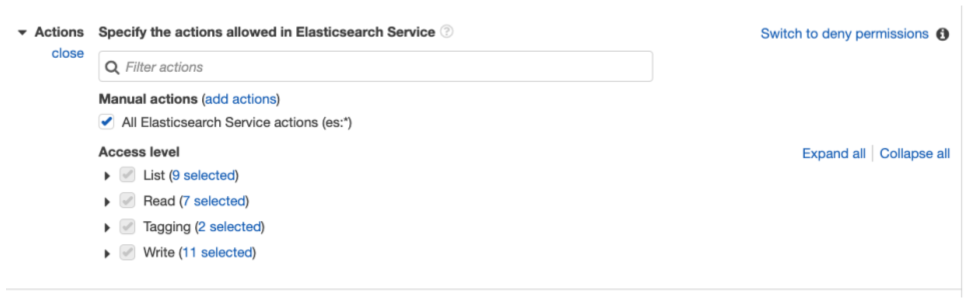
Request conditions: Specify request conditions (optional)

Character count: 4,444 of 10,240.
The current character count includes character for all inline policies in the role: firehose_delivery_role.

Cancel | Review policy

Figure 16: Create policy – choose a service search box

d. Select the check box for **All Elasticsearch Service actions**.


Actions
close
Specify the actions allowed in Elasticsearch Service
Switch to deny permissions
Filter actions
Manual actions (add actions)
Expand all
Collapse all

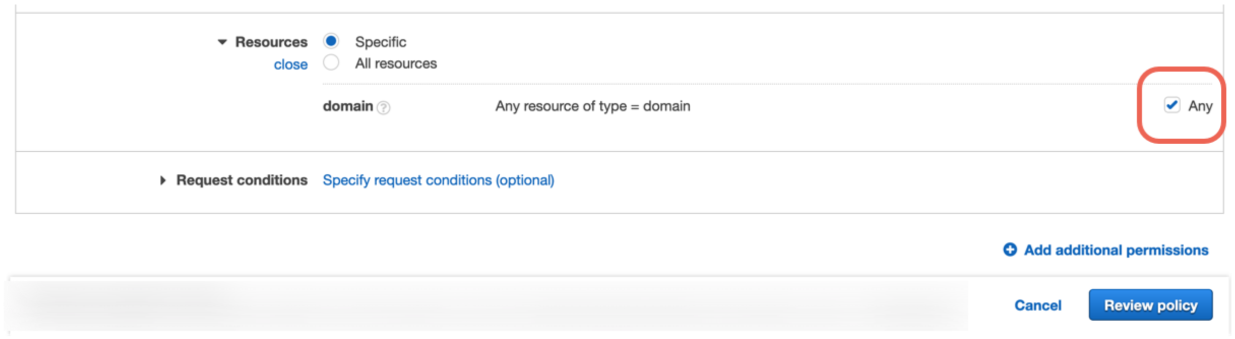
[✓] All Elasticsearch Service actions (es:*)

Access level

- [✓] List (9 selected)
- [✓] Read (7 selected)
- [✓] Tagging (2 selected)
- [✓] Write (11 selected)

Figure 17: Create policy – select option for All Elasticsearch service actions

e. Under **Resources**, select the **Any** check box and choose **Review policy**,


Resources
close
Request conditions
Specify request conditions (optional)
Add additional permissions
Cancel
Review policy

[● Specific] All resources

domain — Any resource of type = domain — [✓ Any]

Figure 18: Create policy – Any resources check box

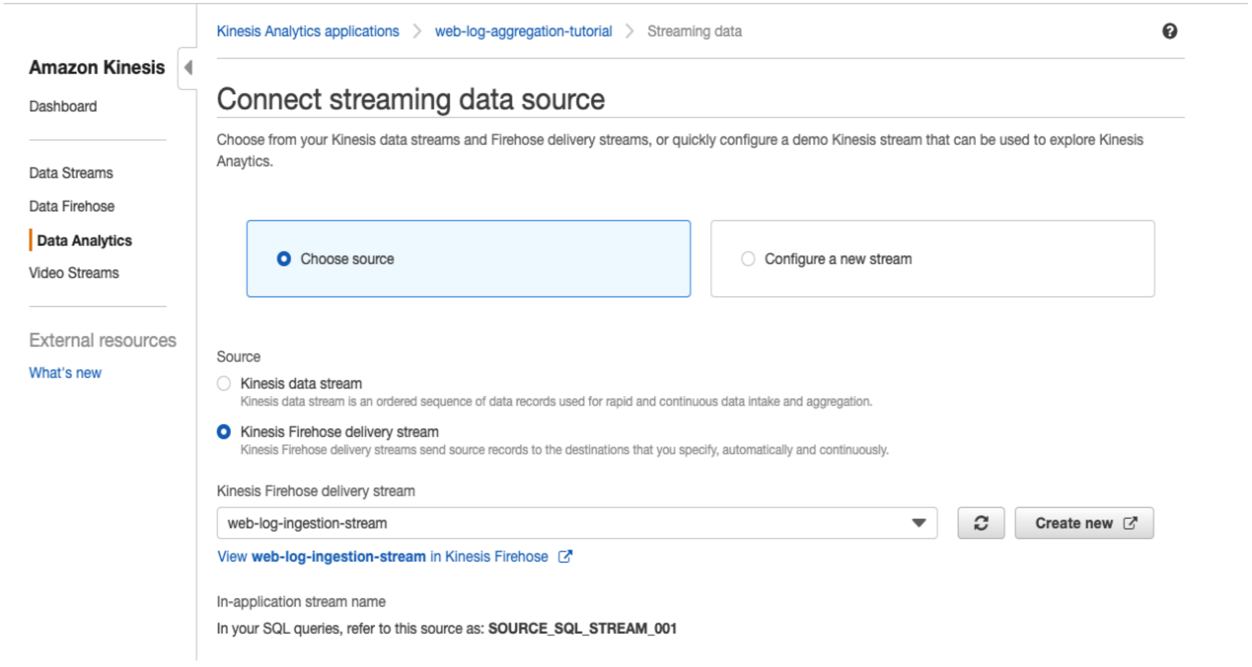
f. On the Review policy page, name the policy *ElasticSearchAccess* and choose **Create policy**.

Step 6: Create an Amazon Kinesis Data Analytics Application

You are now ready to create the Amazon Kinesis Data Analytics application to aggregate data from your streaming web log data and store it in your Amazon ES domain. To create the Amazon Kinesis Data Analytics application:

1. Open the Amazon Kinesis Analytics console at https://console.aws.amazon.com/kinesisanalytics.
2. Choose **Create new application**.
3. For **Application name**, type *web-log-aggregation-tutorial*.
4. Leave the **Runtime** value as the default and choose **Create application**.
5. To configure the source data for the Amazon Kinesis Data Analytics application, choose **Connect streaming data**.

6. Under **Source**, choose **Kinesis Firehose delivery stream** and select **web-log-ingestion-stream** that you created in Step 2.



Connect streaming data source

Choose from your Kinesis data streams and Firehose delivery streams, or quickly configure a demo Kinesis stream that can be used to explore Kinesis Anaytics.

[✓ Choose source] Configure a new stream

Source

Kinesis data stream
Kinesis data stream is an ordered sequence of data records used for rapid and continuous data intake and aggregation.

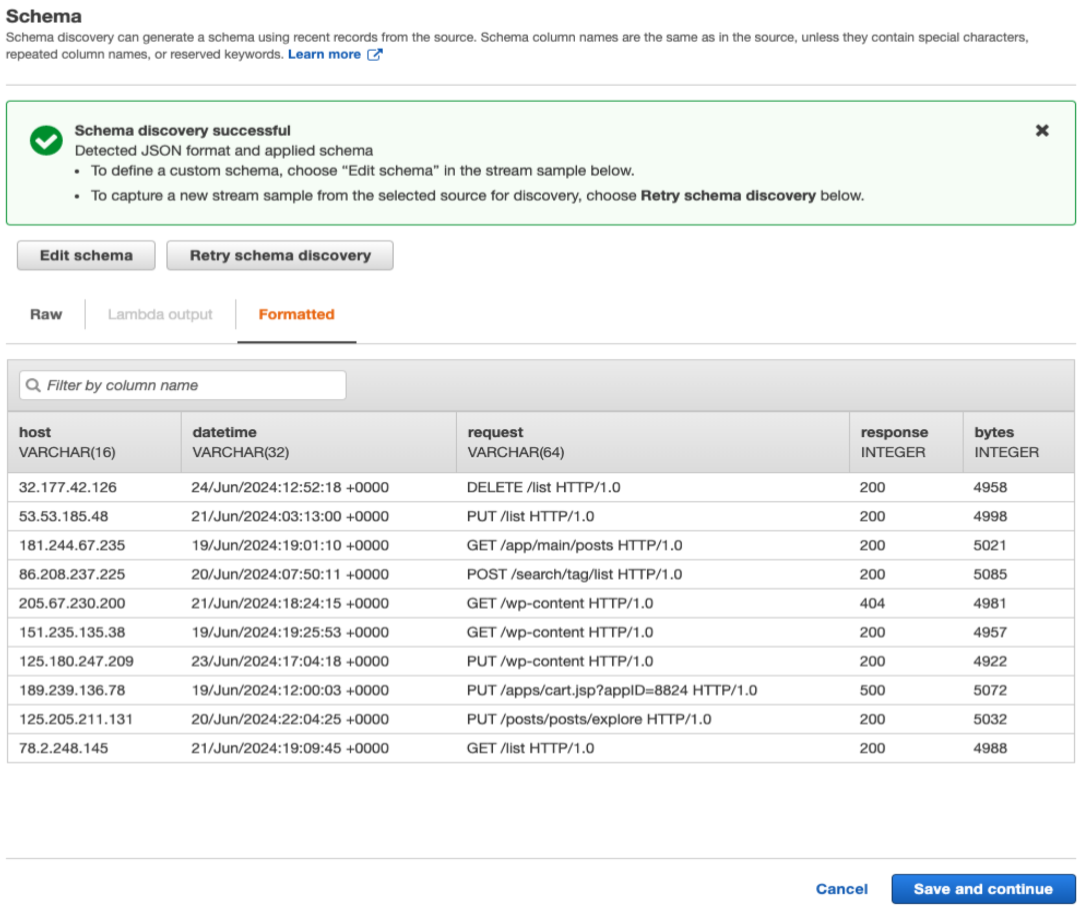
[✓ Kinesis Firehose delivery stream]
Kinesis Firehose delivery streams send source records to the destinations that you specify, automatically and continuously.

Kinesis Firehose delivery stream: web-log-ingestion-stream

In-application stream name
In your SQL queries, refer to this source as: **SOURCE_SQL_STREAM_001**

Figure 19: Specify the Kinesis Firehose delivery stream

7. Scroll down to the **Schema** section and choose **Discover schema**., Amazon Kinesis Analytics analyzes the source data in your Kinesis Data Firehose delivery stream and creates a formatted sample of the input data for your review:



Schema

Schema discovery can generate a schema using recent records from the source. Schema column names are the same as in the source, unless they contain special characters, repeated column names, or reserved keywords. Learn more

Schema discovery successful
Detected JSON format and applied schema
- To define a custom schema, choose "Edit schema" in the stream sample below.
- To capture a new stream sample from the selected source for discovery, choose **Retry schema discovery** below.

host VARCHAR(16)	datetime VARCHAR(32)	request VARCHAR(64)	response INTEGER	bytes INTEGER
32.177.42.126	24/Jun/2024:12:52:18 +0000	DELETE /list HTTP/1.0	200	4958
53.53.185.48	21/Jun/2024:03:13:00 +0000	PUT /list HTTP/1.0	200	4998
181.244.67.235	19/Jun/2024:19:01:10 +0000	GET /app/main/posts HTTP/1.0	200	5021
86.208.237.225	20/Jun/2024:07:50:11 +0000	POST /search/tag/list HTTP/1.0	200	5085
205.67.230.200	21/Jun/2024:18:24:15 +0000	GET /wp-content HTTP/1.0	404	4981
151.235.135.38	19/Jun/2024:19:25:53 +0000	GET /wp-content HTTP/1.0	200	4957
125.180.247.209	23/Jun/2024:17:04:18 +0000	PUT /wp-content HTTP/1.0	200	4922
189.239.136.78	19/Jun/2024:12:00:03 +0000	PUT /apps/cart.jsp?appID=8824 HTTP/1.0	500	5072
125.205.211.131	20/Jun/2024:22:04:25 +0000	PUT /posts/posts/explore HTTP/1.0	200	5032
78.2.248.145	21/Jun/2024:19:09:45 +0000	GET /list HTTP/1.0	200	4988

Figure 20: Schema discovery

8. Leave all values set to their defaults, and choose **Save and continue**. You are taken back to the hub screen for your Amazon Kinesis Data Analytics application.
9. To create the SQL that analyzes the streaming data, choose **Go to SQL editor**.
10. When prompted, choose **Yes, start application**.

    After approximately 60 to 90 seconds, the **Source data** section presents you with a sample of source data that is flowing into your source delivery stream.
11. In the SQL editor, enter the following SQL code:

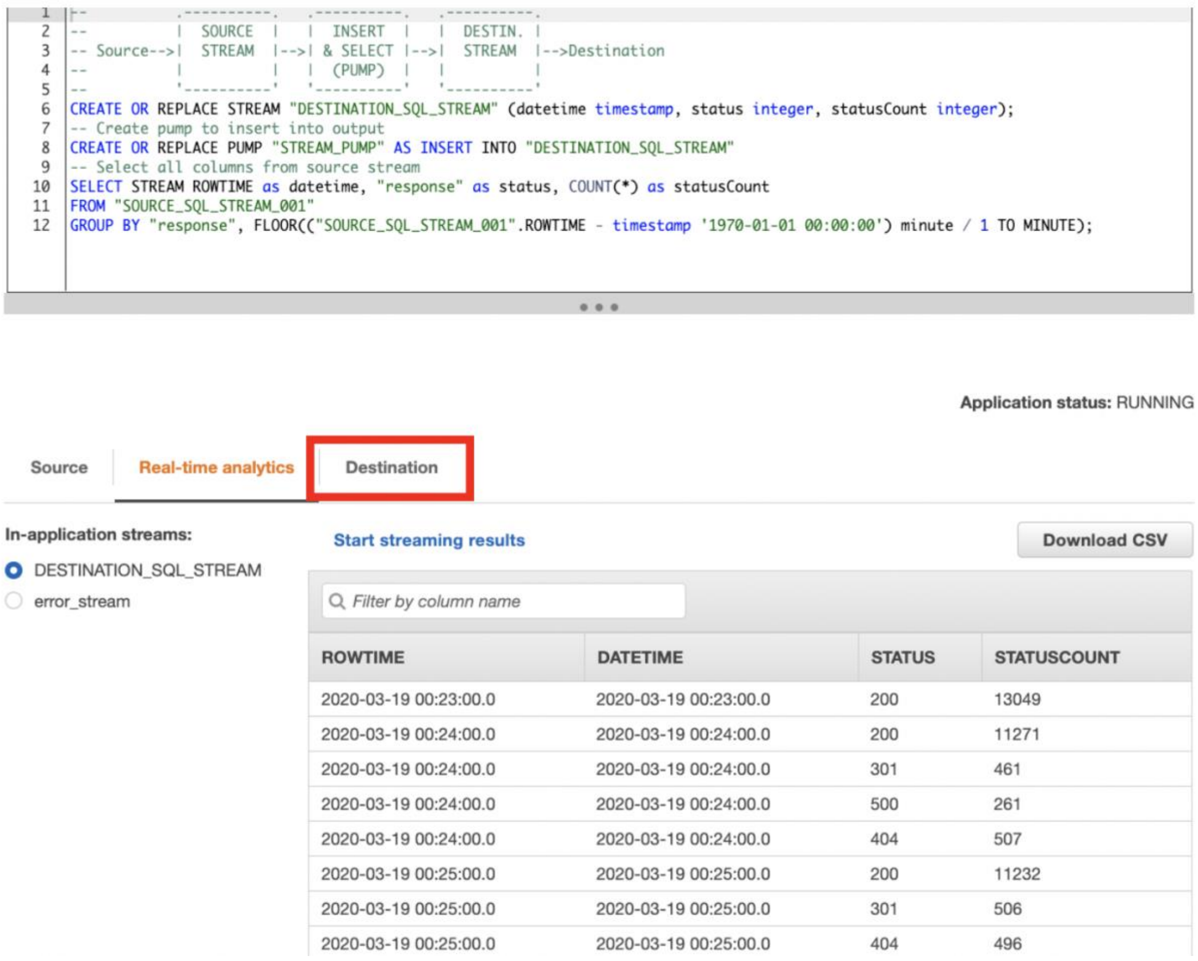
```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM"
    (datetime TIMESTAMP, status INTEGER, statusCount INTEGER);

-- Create pump to insert into output
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
    INSERT INTO "DESTINATION_SQL_STREAM"

-- Select all columns from source stream
SELECT
    STREAM ROWTIME as datetime,
    "response" as status,
    COUNT(*) AS statusCount
        FROM "SOURCE_SQL_STREAM_001"
        GROUP BY
            "response",
            FLOOR(("SOURCE_SQL_STREAM_001".ROWTIME - TIMESTAMP
'1970-0101
    00:00:00') minute / 1 TO MINUTE);
```

The code creates a STREAM and a PUMP:

- A *stream* (in-application) is a continuously updated entity that you can SELECT from and INSERT into (like a TABLE).
- A *pump* is an entity used to continuously 'SELECT...FROM' a source STREAM and INSERT SQL results into an output STREAM.

Finally, an output stream can be used to send results into a destination.

12. Choose **Save and run SQL**. After about 1 minute, Amazon Kinesis Data Analytics displays the output of the query.
13. To save the running output of the query, choose the **Destination** tab and choose **Connect to a destination.**



```
--             .----------.   .----------.   .----------.
--             |  SOURCE  |   |  INSERT  |   |  DESTIN. |
-- Source-->|  STREAM  |-->| & SELECT |-->|  STREAM  |-->Destination
--             |          |   |  (PUMP)  |   |          |
--             '----------'   '----------'   '----------'
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (datetime timestamp, status integer, statusCount integer);
-- Create pump to insert into output
CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
-- Select all columns from source stream
SELECT STREAM ROWTIME as datetime, "response" as status, COUNT(*) as statusCount
FROM "SOURCE_SQL_STREAM_001"
GROUP BY "response", FLOOR(("SOURCE_SQL_STREAM_001".ROWTIME - timestamp '1970-01-01 00:00:00') minute / 1 TO MINUTE);
```

Application status: RUNNING

In-application streams:

- [✓ DESTINATION_SQL_STREAM]
- error_stream

ROWTIME	DATETIME	STATUS	STATUSCOUNT
2020-03-19 00:23:00.0	2020-03-19 00:23:00.0	200	13049
2020-03-19 00:24:00.0	2020-03-19 00:24:00.0	200	11271
2020-03-19 00:24:00.0	2020-03-19 00:24:00.0	301	461
2020-03-19 00:24:00.0	2020-03-19 00:24:00.0	500	261
2020-03-19 00:24:00.0	2020-03-19 00:24:00.0	404	507
2020-03-19 00:25:00.0	2020-03-19 00:25:00.0	200	11232
2020-03-19 00:25:00.0	2020-03-19 00:25:00.0	301	506
2020-03-19 00:25:00.0	2020-03-19 00:25:00.0	404	496

Figure 21: Results of query and Destination tab

14. Under **Destination**, choose **Kinesis Firehose delivery stream** and select the **web-log-aggregated-data** stream that you created in Step 5.
15. For **In-application stream,** choose **DESTINATION_SQL_STREAM**.
16. Leave all other options set to their default values and choose **Save and continue**.

Step 7: View the Aggregated Streaming Data

After approximately 5 minutes, the output of the SQL statement in your Amazon Kinesis Data Analytics application will be written to your Amazon ES domain. Amazon ES has built-in support for Kibana, a tool that allows users to explore and visualize the data stored in an Elasticsearch cluster. To view the output of your Amazon Kinesis Analytics application in Kibana:

1. Open the Amazon ES console at https://console.aws.amazon.com/es.
2. In the **Domain** column, choose the Amazon ES domain called **web-log-summary** that you created in Step 4.
3. On the **Overview** tab, click the link next to **Kibana**.
4. Enter the username and password you created in Step 4.

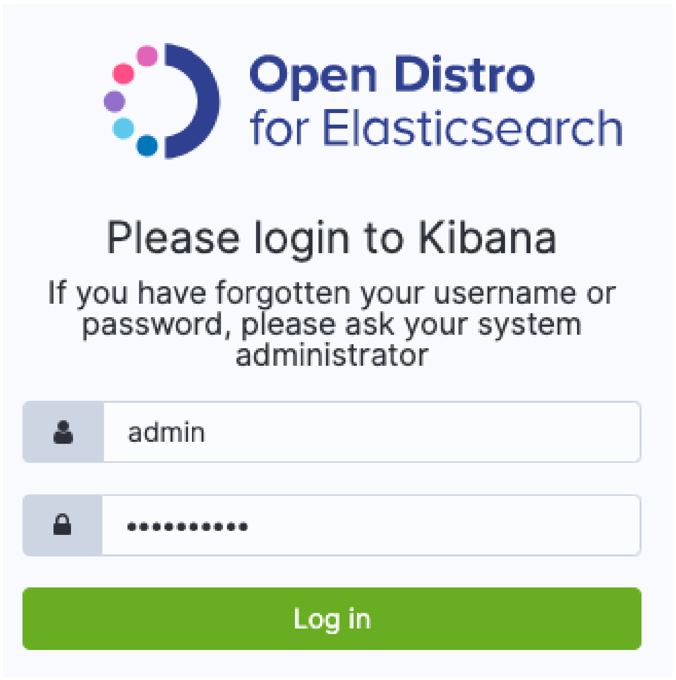

Open Distro
for Elasticsearch
Please login to Kibana
If you have forgotten your username or password, please ask your system administrator
admin
•••••••••••
Log in

Figure 22: Kibana login screen

Because this is the first time you are opening the Kibana application in your Amazon ES domain, you need to configure it. This configuration includes giving your user permissions to access the index in Kibana and giving Kinesis Data Firehose permission to write to the index.

5. In the left toolbar, choose **Security** and then choose **Role Mappings**.

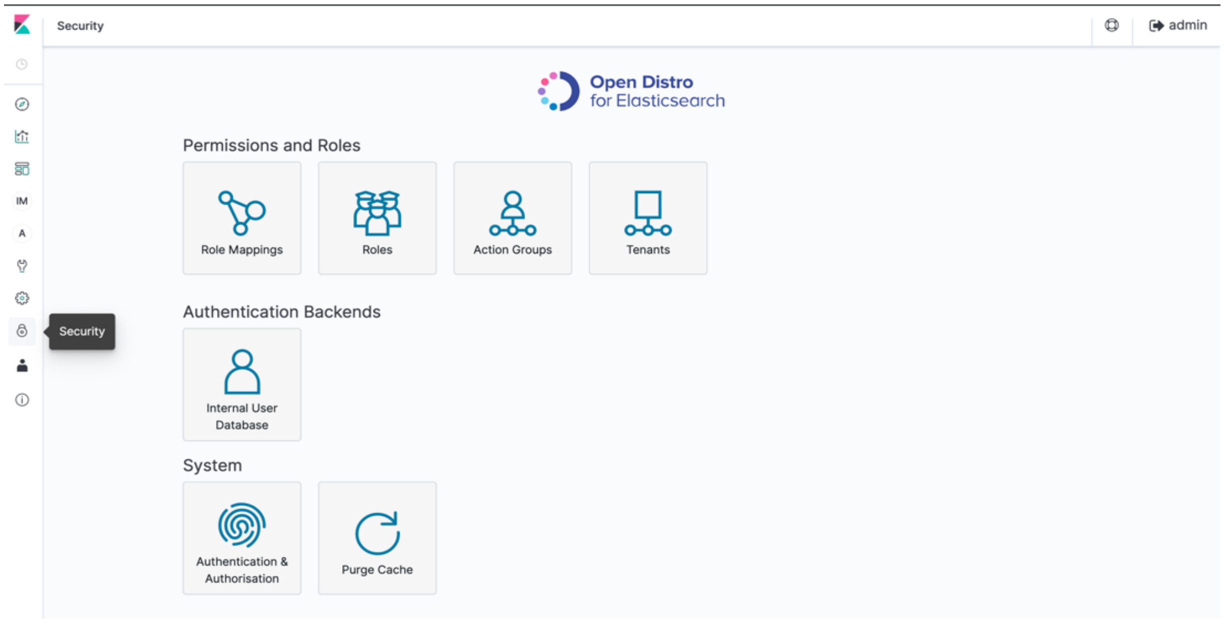
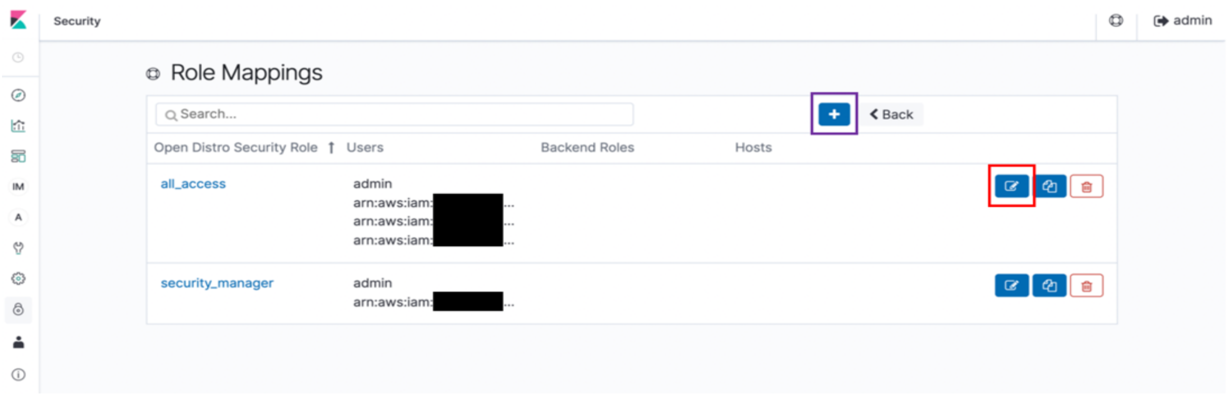


Figure 23: Security options in Kibana

6. Verify an **Open Distro Security Role** named **all_access** role is listed. If it is not listed, choose Add (+ sign) and choose the Role **all_access.** See [Set up Multi-Tenant Kibana Access in Open Distro for Elasticsearch]() on the *AWS Open Source Blog* for more information.

   You need to modify the **Users** associated with this role.

7. Choose **Edit**.



Open Distro Security Role	Users	Backend Roles	Hosts
all_access	admin arn:aws:iam:[redacted]... arn:aws:iam:... arn:aws:iam:...		
security_manager	admin arn:aws:iam:[redacted]...		

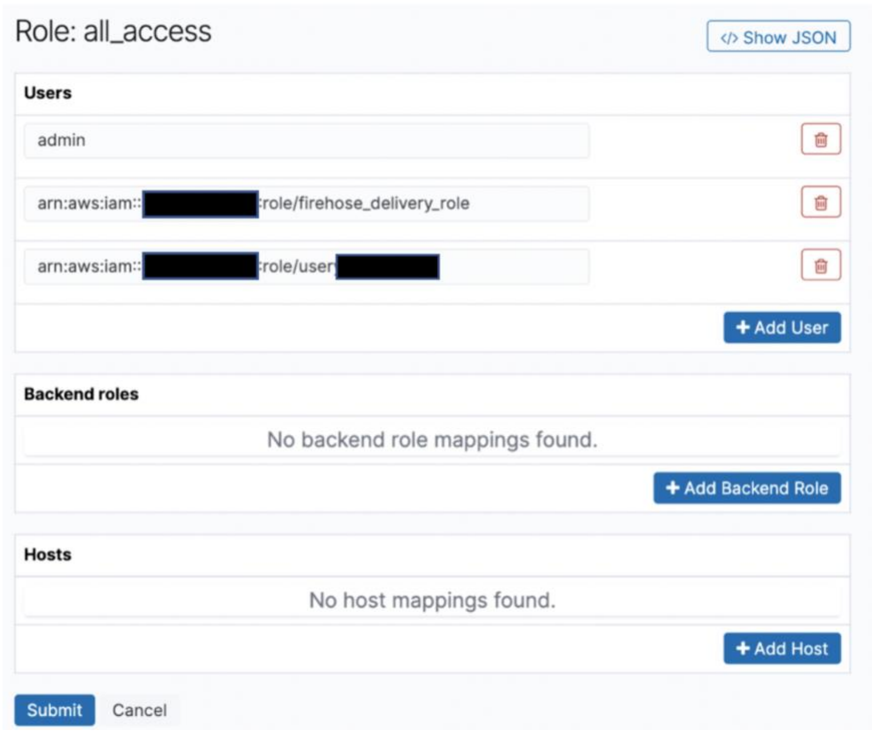
Figure 24: Role Mappings – Add button (in purple box), Edit button (in red box)

   a. Choose **Add User** and type **admin**. This is the user you used to log into Kibana.

b. Choose **Add User** and enter the ARN for your AWS account. The ARN can be found in the IAM console page, under **Users**. Click into your user account and copy the **User ARN**.

c. Repeat this step for the **firehose delivery role** you created earlier**,** found under **Roles** in the IAM Console.

You should now have three entries listed for the **all_access** Kibana Role.



Role: all_access

Users

admin

arn:aws:iam::[redacted]:role/firehose_delivery_role

arn:aws:iam::[redacted]:role/user[redacted]

Backend roles

No backend role mappings found.

Hosts

No host mappings found.

Figure 25: Entries for all_access role

d. Choose **Submit**.

8. Choose the Kibana icon on the top left to return to the Kibana dashboard.

9. Choose **explore on my own.** Then, choose **Connect to your Elasticsearch index.**

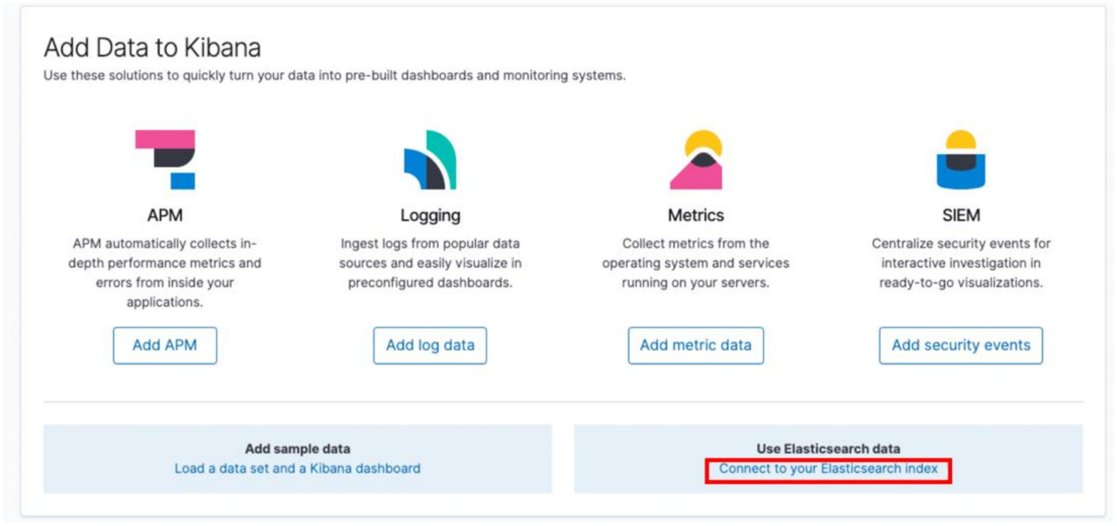

Add Data to Kibana
Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.
APM
APM automatically collects in-depth performance metrics and errors from inside your applications.
Add APM
Logging
Ingest logs from popular data sources and easily visualize in preconfigured dashboards.
Add log data
Metrics
Collect metrics from the operating system and services running on your servers.
Add metric data
SIEM
Centralize security events for interactive investigation in ready-to-go visualizations.
Add security events
Add sample data
Load a data set and a Kibana dashboard
Use Elasticsearch data
Connect to your Elasticsearch index

Figure 26: Kibana dashboard – connect to Elasticsearch index

10. the **Index pattern** field, type *request_data**. This entry uses Elasticsearch index name that you created in Step 5.
11. Choose **Next step**.

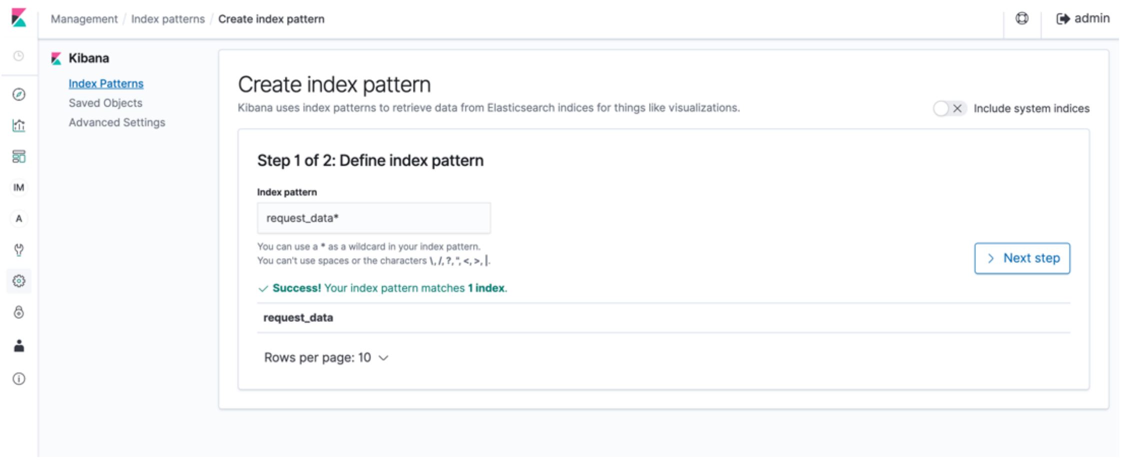

Management / Index patterns / Create index pattern
admin
Kibana
Index Patterns
Saved Objects
Advanced Settings
Create index pattern
Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.
Include system indices
Step 1 of 2: Define index pattern
Index pattern
request_data*
You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.
Next step
Success! Your index pattern matches 1 index.
request_data
Rows per page: 10

Figure 27: Create index pattern

Kibana automatically identifies the DATETIME field in your input data, which contains time data.

12. Choose **Create index pattern**.

To visualize the data in your Elasticsearch index, you will create and configure a line chart that shows how many of each HTTP response type were included in the source web log data per minute.

To create the line chart:

a. In the toolbar, choose **Visualize**, and then choose **Create new visualization**.

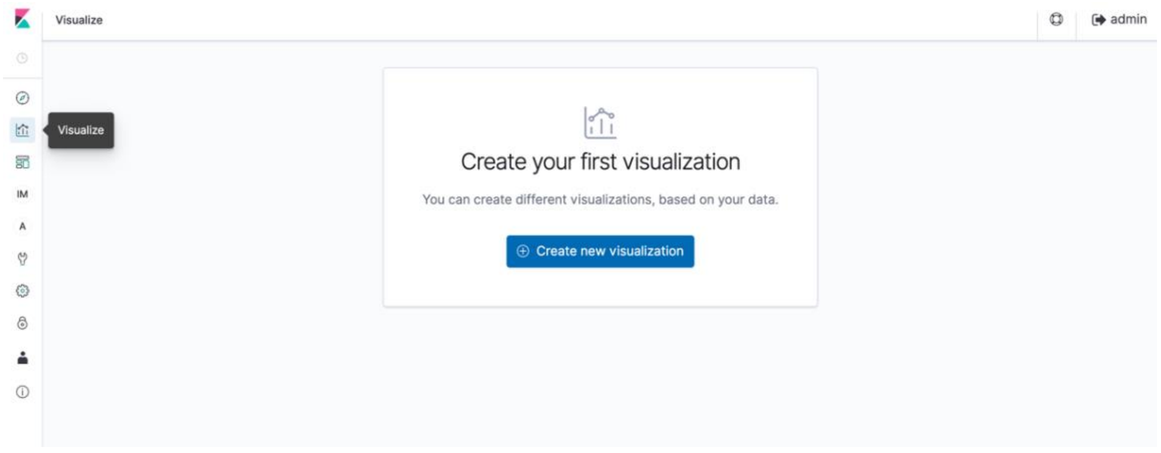

Visualize
admin
Visualize
Create your first visualization
You can create different visualizations, based on your data.
Create new visualization

Figure 28: Create new visualization

b. Choose **Line chart**.

c. For **Choose a search**, select **request_data***.

   To configure your chart, you first need to tell Kibana what data to use for the y-axis:

d. In the **metrics** section, choose the arrow next to **Y-Axis**.

e. Under **Aggregation**, choose **Sum**.

f. Under **Field**, choose **STATUSCOUNT**.

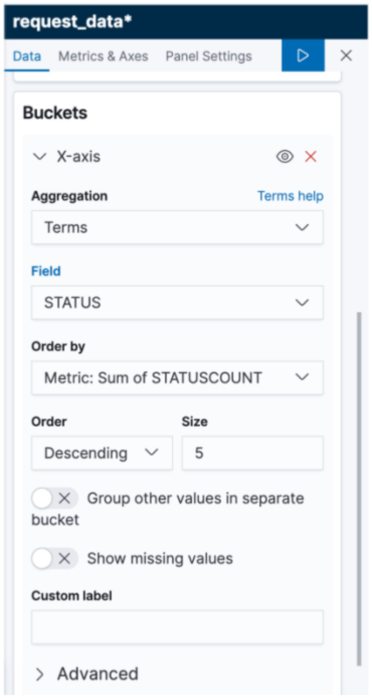
   Now you need to configure the x-axis:

g. In the **buckets** section, select **X-axis** with the **addition button**.

h. Under **Aggregation**, choose **Terms**.

i. Under **Field**, choose **STATUS**.

j. To run the query and view the line chart, choose on the blue “play” button in the top right-hand corner.



Buckets

X-axis

Aggregation: Terms

Field: STATUS

Order by: Metric: Sum of STATUSCOUNT

Order: Descending

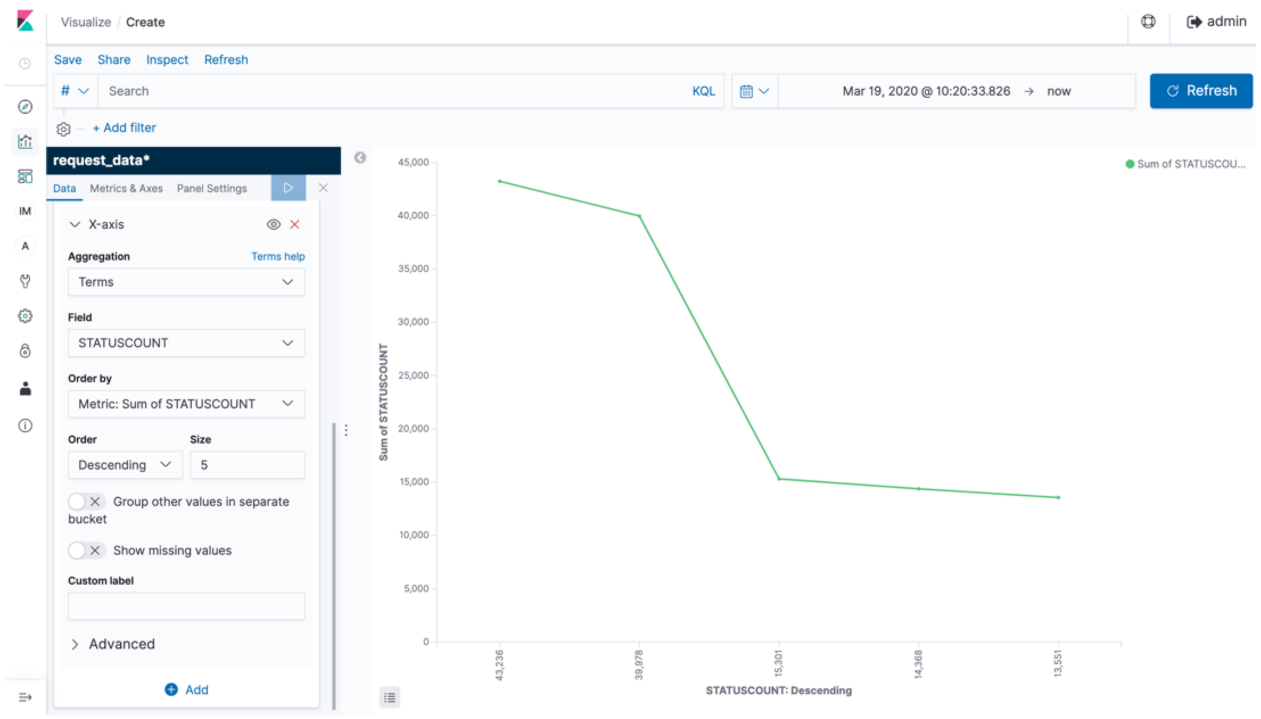
Size: 5

Group other values in separate bucket

Show missing values

Custom label

Figure 29: Configure X-axis values



X-axis

Aggregation: Terms

Field: STATUSCOUNT

Order by: Metric: Sum of STATUSCOUNT

Order: Descending

Size: 5

Group other values in separate bucket

Show missing values

Custom label

Advanced

Figure 30: Query results in Kibana

Step 8: Clean Up

After completing this tutorial, be sure to delete the AWS resources that you created so that you no longer accrue charges.

Terminate the EC2 Instance

If you created a new EC2 instance to generate a continuous stream of Apache access log data, you will need to stop or terminate that instance to avoid further charges.

1. Navigate to the EC2 console at https://console.aws.amazon.com/ec2.
2. Select **Running Instances**, and find the instance you used to generate your Apache access logs.
3. On the **Actions** menu, choose the instance, and then choose **Instance State**, then **Terminate**.
4. Read the warning regarding instance termination, and choose **Yes, Terminate**.

If you used an existing EC2 instance with Apache access logs and you do not plan to stop or terminate the instance, you should stop the Amazon Kinesis Agent so that no additional records are sent to the Kinesis Data Firehose delivery stream. Stop the agent with the following command:

```
sudo service aws-kinesis-agent stop
```

Delete the Amazon Elasticsearch Service Domain

1. Navigate to the Amazon ES console at https://console.aws.amazon.com/es
2. Locate and select the domain **web-log-summary** that you created in Step 4.
3. On the **Actions** menu, choose **Delete domain**.
4. On the **Delete domain** confirmation, select the check box and choose **Delete**.

Delete the Amazon S3 Bucket and Bucket Objects

1. Navigate to the Amazon S3 console at https://console.aws.amazon.com/s3.
2. Locate the S3 bucket that you created in Step 2.
3. Right-click the bucket name and choose **Delete Bucket**.
4. To confirm the deletion, type the bucket name and choose **Delete**.

Note: At this point in the tutorial, you have terminated or stopped any services that accrue charges while ingesting and processing data. Because the data producer has been stopped, you will not incur additional charges for Amazon Kinesis Data Firehose and Amazon Kinesis Data Analytics since data is not being ingested or processed. You can safely leave them in place for later reference or future development. However, if you wish to remove all resources created in this tutorial, continue with the following steps.

Delete the Amazon Kinesis Data Analytics Application and the Amazon Kinesis Data Firehose Delivery Streams

1. Navigate to the Amazon Kinesis console at https://console.aws.amazon.com/kinesis.
2. Choose **Go to Analytics**.
3. Locate and select the name of the Amazon Kinesis Data Analytics application called **web-log-aggregation-tutorial** that you created in Step 6 to view its details.
4. Choose **Application details**.
5. On the **Actions** menu, choose **Delete application**.
6. To confirm the deletion, in the confirmation modal, choose **Delete application**.
7. Navigate to the Amazon Kinesis Data Firehose console at https://console.aws.amazon.com/firehose.
8. Choose the Firehose delivery stream called **web-log-ingestion-stream** that you created in Step 2.
9. On the **Actions** menu, choose **Delete**.
10. To confirm the deletion, enter the name of the delivery stream and choose **Delete**.
11. Repeat items 7 through 10 for the second delivery stream called **web-log-aggregated-data** that you created in Step 5.

Additional Resources

We recommend that you continue to learn more about the concepts introduced in this guide with the following resources:

- For detailed information on Amazon Kinesis Analytics, see Amazon Kinesis Analytics: How It Works.
- For information on how to develop your own Amazon Kinesis Analytics application, with specific information about its SQL extensions, windowed queries, and joining multiple streams, see Streaming SQL Concepts.
- For additional examples, see Example Amazon Kinesis Analytics Applications.