

Building a Bigdata streaming architecture with AWS services and webhook

<https://aws.amazon.com/blogs/architecture/field-notes-building-a-shared-account-structure-using-aws-organizations/>

reference

https://w.amazon.com/bin/view/AWS/Teams/SA/MediaSeries/AWS_Architecture_Blog/

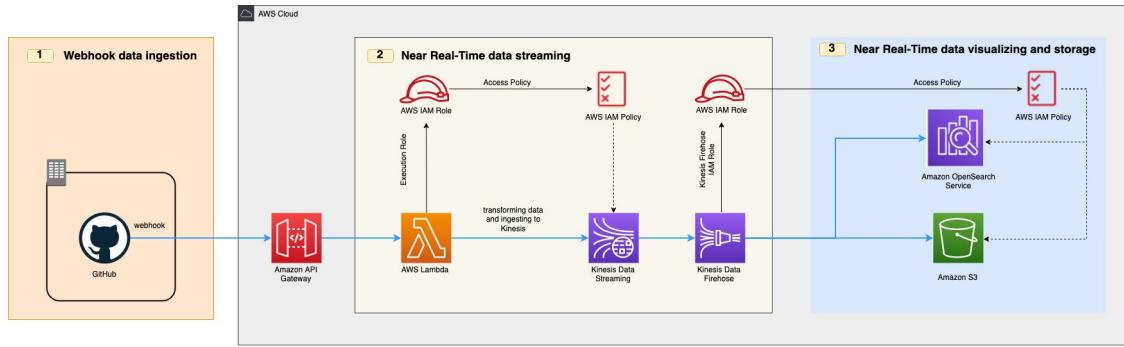
For customers who has 3rd party data present in various services like bitbucket, Rally, JIRA, ServiceNow, GitHub etc. It is very challenging to model a common cloud data streaming architecture. Most of the customer end-up creating a pull based model and face REST API rate limit errors. AWS Kinesis Data Streaming with API Gateway make it possible to build the right architecture to stream the near-real-time events from any data-source using their webhook feature. And Amazon OpenSearch service integration provides near-real-time search feature to their ingested data.

Partners and customers who want to stream their 3rd party data needs to build a new API and link it through the 3rd party service webhook trigger. This is so that all of the events generated on those 3rd party service will be posted to the AWS REST API.

While linking or transferring their 3rd party data to AWS cloud, the customer must check the security settings. It should not compromise any customer security controls and continue to provide full control on data transfer.

Architecture Diagram

The following architecture represents the big data streaming setup which is streaming GitHub webhook events. This architecture can scale for any 3rd party data source which can emit events via webhook events.



GitHub will push each event through webhook to Amazon API Gateway URL. Amazon API Gateway will trigger a lambda function. AWS Lambda function will post the event data to Amazon Kinesis Data Stream which then stream the events using Amazon Kinesis Firehose to Amazon S3 bucket for source of truth data. At the same time Amazon Kinesis Firehose will ingest the same event to the Amazon OpenSearch Service.

Security considerations

The following table highlights both Amazon Kinesis Firehose and AWS Lambda responsibilities and access controls provided by the architecture in the previous section.

For security reason we create 2 IAM roles one for AWS Lambda execution role and another for Amazon Kinesis Firehose role. Each role will have appropriate permissions like AWS Lambda execution role will have permission to send the data to Kinesis Data Stream and Amazon Kinesis Firehose IAM role will have permission mapping in Amazon OpenSearch as a backend role to have access to create index. Also you can create a secret for webhook data encryption and in AWS API gateway create an Authorizer AWS Lambda function to validate the secret key before ingesting git event.

[git data streaming with kinesis\(2\).xml](#)

Steps to create the solution in AWS

1. Create OpenSearch cluster for this demo. Give the domain name **git-data-search-domain** and deployment type as Development and testing with latest version.

Amazon OpenSearch Service > Domains > Create domain

Create domain Info

Name

Domain name
The name must start with a lowercase letter and must be between 3 and 28 characters. Valid characters are a-z (lowercase only), 0-9, and - (hyphen).

Custom endpoint
Each Amazon OpenSearch Service domain has an auto-generated endpoint, but you can also add a custom endpoint using AWS Certificate Manager (ACM). [Learn more](#)

Enable custom endpoint

Deployment type
Deployment types specify common settings for your use case. After creating the domain, you can change these settings at any time.

Deployment type

Production
Domain intended for production workloads spanning multiple AZ and dedicated master.

Development and testing
Domain intended for development or testing use outside of a production environment.

Custom
Choose settings from all available options.

Version
▼

Certain features require specific OpenSearch/Elasticsearch versions. We recommend choosing the latest version.
[Learn more](#)

For Data nodes select 1-AZ (in production you may want to go for 3-AZ), select instance type as **t3.medium.search**, number of nodes is 1.

Data nodes

Select an instance type that corresponds to the compute, memory, and storage needs of your application. Consider the size of your indices, number of shards and replicas, type of queries, and volume of requests. [Learn more](#)

Availability Zone(s)

- 3-AZ
Recommended for production workloads with higher availability requirements.
- 2-AZ
Suitable for production workloads.
- 1-AZ

Instance type

t3.medium.search

t3.medium.search instance type needs EBS storage.

- Include previous generation instance types

⚠️ T3 instance types are suitable only for testing and development purposes. For production workloads, we recommend using latest generation instance types - general purpose, memory optimized, compute optimized, or storage optimized.

Number of nodes

1



The number must be between 1 and 40.

Storage type

Choose a storage type for your data nodes.

EBS



EBS volume type

EBS volumes enable you to independently scale the storage resources of your domain from its compute resources. EBS volumes are most useful for domains with very large data sets, but without the need for large compute resources.

General Purpose (SSD) - gp3



- Include previous generation EBS volume types

EBS storage size per node

30



EBS storage size per node in GiB. Minimum 10 GiB and maximum 200 GiB.

In Network select Public access, in production you may want to select VPC access recommended option.

Network

Choose internet or VPC access. To enable VPC access, we use private IP addresses from your VPC, which provides an inherent layer of security. You control network access within your VPC using security groups. Optionally, you can add an additional layer of security by applying a restrictive access policy. Internet endpoints are publicly accessible. If you select public access, you should secure your domain with an access policy that only allows specific users or IP addresses to access the domain.

Network

- VPC access (recommended)
- Public access

In **Fine-grained access control** select Create master user and enter user name and password as desired.

Fine-grained access control

Fine-grained access control provides numerous features to help you keep your data secure. Features include document-level security, field-level security, read-only users, and OpenSearch Dashboards/Kibana tenants. Fine-grained access control requires a master user. [Learn more](#)

- Enable fine-grained access control

Master user

- Set IAM ARN as master user
- Create master user

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

In Access policy use Only use fine-grained access control

Access policy

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

Domain access policy

- Only use fine-grained access control
Allow open access to the domain.
- Do not set domain level access policy
All requests to the domain will be denied.
- Configure domain level access policy

Rest settings keep as default and select create button.

2. Create S3 bucket where you want to save the GitHub logs. I named the bucket as git-data

Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

git-data

Bucket name must be globally unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1 ▾

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced



Upcoming permission changes to disable ACLs

Starting in April 2027, to disable ACLs when creating buckets by using the S3 console, you will no longer

3. Create **Amazon Kinesis Data Stream** for streaming git events
- Select Kinesis Data Streams and select Create data stream button

Get started

- Kinesis Data Streams
Collect streaming data with a data stream.
- Kinesis Data Firehose
Process and deliver streaming data with data delivery stream.
- Kinesis Data Analytics
Analyze streaming data with data analytics application.

Create data stream

Select on-demand and select create button.

Amazon Kinesis > Data streams > Create data stream

Create data stream Info

Data stream configuration

Data stream name

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens and periods.

Data stream capacity Info

Capacity mode

- On-demand
Use this mode when your data stream's throughput requirements are unpredictable and variable. With on-demand mode, your data stream's capacity scales automatically.
- Provisioned
Use provisioned mode when you can reliably estimate throughput requirements of your data stream. With provisioned mode, your data stream's capacity is fixed.

Total data stream capacity

By default, data streams with on-demand mode scale throughput automatically to accommodate traffic of up to 200 MiB per second and 200,000 records per second for the write capacity. If traffic exceeds capacity, your data stream will throttle. To request capacity increase up to 1GB per second write and 2GB per second read, [submit a support ticket](#).

Write capacity Maximum 200 MiB/second and 200,000 records/second	Read capacity Maximum (per consumer) 400 MiB/second Up to 2 default consumers. Use Enhanced Fan-Out (EFO) for more consumers. EFO supports adding up to 20 consumers, each having a dedicated throughput.
---	---

3. Create Delivery Streams

Select Delivery streams in Amazon Kinesis. Choose Source as Amazon Kinesis Data Streams and Destination as Amazon S3. In **Source settings** select the Kinesis data stream (git-data-stream) that we created in previous step. In **Delivery stream name** give the name as **git-data-delivery-stream**.

Create Kinesis Delivery Streaming, using Kinesis Firehose to stream the git logs to S3 bucket and keep the Buffer interval to 60 seconds. Select create IAM role AWS will create role and assign permissions required to add new files to S3 bucket. Name this as git-data-delivery-stream.

The screenshot shows the 'Create delivery stream' wizard. The top navigation bar includes 'Amazon Kinesis > Delivery streams > Create delivery stream'. The main title is 'Create delivery stream' with an 'Info' link. Below it is a section titled 'Amazon Kinesis Data Firehose: How it works' with a link icon. The next section is 'Choose source and destination', which contains instructions: 'Specify the source and the destination for your delivery stream. You cannot change the source and destination of your delivery stream once it has been created.' It shows 'Source' set to 'Amazon Kinesis Data Streams' and 'Destination' set to 'Amazon S3'. The 'Source settings' section shows the Kinesis data stream ARN: arn:aws:kinesis:us-east-1:605024711850:stream/git-data-stream, with 'Browse' and 'Create' buttons. The 'Delivery stream name' section shows 'Git-Data-Delivery-Stream' in the input field, with a note: 'Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.' The overall interface is clean with a light gray background and white cards for each configuration step.

In **Destination settings** select the S3 bucket (git-data) that we created in previous step.

Destination settings Info

Specify the destination settings for your delivery stream.

S3 bucket

BrowseCreate 

Format: s3://bucket

Dynamic partitioning Info

Dynamic partitioning enables you to create targeted data sets by partitioning streaming S3 data based on partitioning keys. You can partition your source data with inline parsing and/or the specified AWS Lambda function. You can enable dynamic partitioning only when you create a new delivery stream. You cannot enable dynamic partitioning for an existing delivery stream. Enabling dynamic partitioning incurs additional costs per GiB of partitioned data. For more information, see [Kinesis Data Firehose pricing](#).

- Not enabled
 Enabled

S3 bucket prefix - *optional*

By default, Kinesis Data Firehose appends the prefix "YYYY/MM/dd/HH" (in UTC) to the data it delivers to Amazon S3. You can override this default by specifying a custom prefix that includes expressions that are evaluated at runtime.

You can repeat the same keys in your S3 bucket prefix. Maximum S3 bucket prefix characters: 1024.

Expand **Buffer hints, compression and encryption** section and change the **Buffer interval** to 60 seconds. In production based on your data size you can change this time to maximum 900 seconds.

▼ Buffer hints, compression and encryption

The fields below are pre-populated with the recommended default values for S3. Pricing may vary depending on storage and request costs.

S3 buffer hints

Kinesis Data Firehose buffers incoming records before delivering them to your S3 bucket. Record delivery is triggered once the value of either of the specified buffering hints is reached.

Buffer size

The higher buffer size may be lower in cost with higher latency. The lower buffer size will be faster in delivery with higher cost and less latency.

 MiB

Minimum: 1 MiB, maximum: 128 MiB. Recommended: 5 MiB.

Buffer interval

The higher interval allows more time to collect data and the size of data may be bigger. The lower interval sends the data more frequently and may be more advantageous when looking at shorter cycles of data activity.

 seconds

Minimum: 60 seconds, maximum: 900 seconds. Recommended: 300 seconds.

S3 compression and encryption

Kinesis Data Firehose can compress records before delivering them to your S3 bucket. Compressed records can also be encrypted in the S3 bucket using an AWS Key Management Service (KMS) master key.

Compression for data records

Kinesis Data Firehose can compress records before delivering them to your S3 bucket.

- Not enabled
- GZIP
- Snappy
- Zip
- Hadoop-Compatible Snappy

Encryption for data records

Compressed record gets encrypted in the S3 bucket using a KMS master key.

- Not enabled

In **Advance settings** service access select Create or update IAM role. This role will have desired permissions to create git events files in S3 bucket. Finally select Create delivery stream button.

Service access | [Info](#)

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.

Create or update IAM role KinesisFirehoseServiceRole-git-data-deli-us-east-1-1681480280033

Creates a new role or updates an existing one and adds the required policy 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.

Tags | [Info](#)

You can add tags to organize your AWS resources, track costs, and control access.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#)

[Create delivery stream](#)

4. Now we will create **AWS Lambda** to receive the GitHub data and post to Amazon Kinesis Data Stream.

Create a function from AWS Lambda page. Give name as **git-webhook-handler** select **Runtime** as Python 3.9 and select Create function button.

Create function Info

AWS Serverless Application Repository applications have moved to [Create application](#).

Author from scratch

Start with a simple Hello World example.

Use a blueprint

Build a Lambda application from scratch.

Basic information

Function name

Enter a name that describes the purpose of your function.

git-webhook-handler

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.9

Architecture Info

Choose the instruction set architecture you want for your function code.

x86_64

arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ [Change default execution role](#)

Select Configuration tab, select Edit button, change the **Timeout** to 5 mins or anything more than default 3 seconds and select save button.

The screenshot shows the AWS Lambda Configuration page. The top navigation bar includes tabs for Code, Test, Monitor, Configuration (which is selected), Aliases, and Versions. On the left, a sidebar lists General configuration options: Triggers, Permissions, Destinations, Function URL, and Environment variables. The main panel displays the General configuration details:

- Description: -
- Timeout: 5 min 3 sec

Get the code from <https://gist.github.com/rupeshtiwari/1b25f0f57ccf85c25b9a33e689cb58f3> and paste it in the AWS lambda function make sure to change the **StreamName** if you have given different name. This code will stream the git event to Kinesis Data Stream. Next select Deploy button.

```

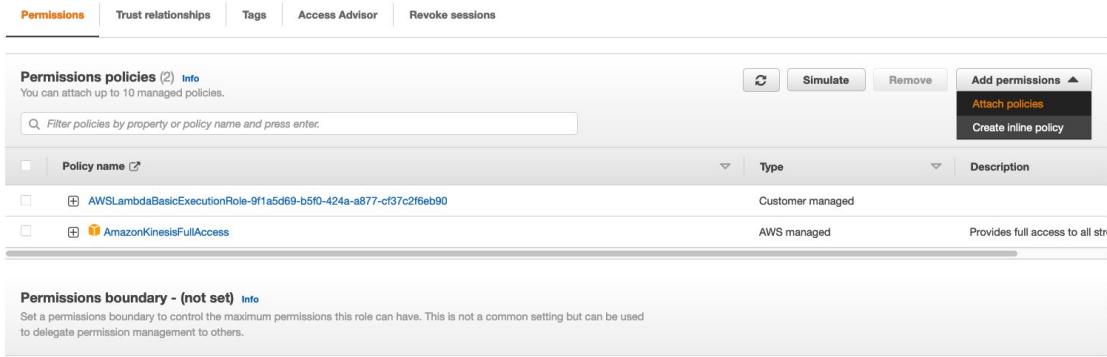
Go to Anything (⌘ P)   lambda_function x   Execution results x   +
└── git-webhook-handle
    └── lambda_function.py
1 import json
2 import boto3
3 import uuid
4
5 def lambda_handler(event, context):
6     client = boto3.client('kinesis')
7     data = {'order': '2 Icecreams'}
8
9     response = client.put_record(
10         StreamName='git-data-stream',
11         Data=json.dumps(data),
12         PartitionKey=str(uuid.uuid4())
13     )
14
15     return response;

```

5. Add permission to AWS Lambda function's execution role to put records on Amazon Kinesis Data Streaming.

Select **Configuration** tab, select **Permissions** and select the role link. Select **Add permissions** options and select **Attach policies** option. Search for AmazonKinesisFullAccess policy in production you may want to just select READ and Write access to Amazon Kinesis for this demo I will give full access. Then select **Add permissions** button.

Go to AWS Lambda function configuration, permissions section and select the role. Select Attach policy and add AmazonKinesisFullAccess policy to the role and save it.



The screenshot shows the 'Permissions' tab of the AWS IAM console. At the top, there are tabs for 'Permissions', 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions'. Below these are sections for 'Permissions policies' and 'Permissions boundary'. In the 'Permissions policies' section, there are two entries: 'AWSLambdaBasicExecutionRole-9f1a5d69-b5f0-424a-a877-cf37c2f6eb90' (Customer managed) and 'AmazonKinesisFullAccess' (AWS managed). A search bar and filter options are also present. At the bottom of this section, there is a prominent 'Add permissions' button with a dropdown menu showing 'Attach policies' and 'Create inline policy'. The 'Permissions boundary - (not set)' section is shown below, with a note about setting a boundary to control maximum permissions.

6. Now we will test the streaming. Navigate to AWS Lambda function, select test button and give the name as **test-data** and select **save button**.

Configure test event

X

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event

Edit saved event

Event name

test-data

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Format JSON

1 [{}]

Cancel

Save

Next select the **test** button and In the **Execution result** tab notice the **ShardId**, **SequenceNumber**, **HTTPStatus** code as 200. Which means we successfully stream the data to Kinesis data stream.

```

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "4963982941315500935913777161061321069153229401648791554",
  "ResponseMetadata": {
    "RequestId": "c09977ea-db8-c072-9c2a-14087a106bed",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "c09977ea-db8-c072-9c2a-14087a106bed",
      "x-amz-trace-id": "0uAlp7AmmT2HIS0U2cmsh272lV1bjd1zsnkQcknWk/oaz57FdK33uy0enjwgAXMuVALmGBESryd+/fZnXhX7qDs0Ywp3k",
      "date": "Fri, 14 Apr 2023 14:22:26 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}

Function Logs
START RequestId: Sece09f7-33ae-4a20-8577-8fc5fd69911e Version: $LATEST
END RequestId: Sece09f7-33ae-4a20-8577-8fc5fd69911e
REPORT RequestId: Sece09f7-33ae-4a20-8577-8fc5fd69911e Duration: 1300.37 ms Billed Duration: 1301 ms Memory Size: 128 MB Max Memory Used: 68 MB Init Duration: 246.90 ms
Request ID
Sece09f7-33ae-4a20-8577-8fc5fd69911e

```

To verify, open **Kinesis Data stream**, select **Data viewer** tab, select the **Shared Id** displayed in function response, select **Trim horizon** and select **Get records**. You will be able to view the ingested data. You can also search by sequence number for the selected event.

Status	Capacity mode	ARN	Creation time
Active	On-demand	arn:aws:kinesis:us-east-1:147228461610:stream/git-data-stream	April 14, 2023 at 09:46 EDT

Applications | Monitoring | Configuration | **Data viewer** | Enhanced fan-out (0)

Shard: shardId-000000000000 Starting position: Trim horizon | Get records

Records (1)

Partition key	Data	Approximate arrival timestamp	Sequence number
10e53dd0-839...	{"order": "2 Icecreams"}	April 14, 2023 at 10:22:26 EDT	4963982941315500935913777161061321069153229

Record data

Sequence number	4963982941315500935913777161061321069153229401648791554
Shard ID	shardId-000000000000

Raw data **JSON** **Copy**

```
{
  "order": "2 Icecreams"
}
```

Close

Navigate to S3 bucket and confirm that the record is also created in the bucket.

Objects **Properties**

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Actions **Create folder** **Upload**

Find objects by prefix

Name	Type	Last modified	Size	Storage class
git-data-delivery-stream-1-2023-04-14-14-22-26-725bdeb3-e703-4f2e-bc40-be1698f37bf2	-	April 14, 2023, 10:23:29 (UTC-04:00)	24.0 B	Standard

Select Query with S3 select and **Run SQL query** to visualize the data.

Query results

Query results are not available after you choose **Close** or navigate away. Choose **Download results** to download a copy of the following query results.

Status

Successfully returned 1 record in 283 ms

Bytes returned: 25 B

```
{"order": "2 Icecreams"}
```

7. Next we will create a **Amazon API Gateway**. Select **REST API** and build button.

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

[Import](#)

[Build](#)

Select New API, give API name as **git-webhook-handler-api** and select **Create API** button. For instructions on how to create and deploy an API by using the API Gateway console, see [Creating a REST API in Amazon API Gateway](#) and [Deploying a REST API in Amazon API Gateway](#), respectively.

Amazon API Gateway APIs > Create

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

REST WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API Clone from existing API Import from Swagger or Open API 3 Example API

Settings

Choose a friendly name and description for your API.

API name* git-webhook-handler-api

Description

Endpoint Type Regional

* Required

Select **Actions** as **Create a resource**, enter Resource Name as **git-data** and select **Create Resource**

New Child Resource

Use this page to create a new child resource for your resource.

Configure as proxy resource

Resource Name* git-data

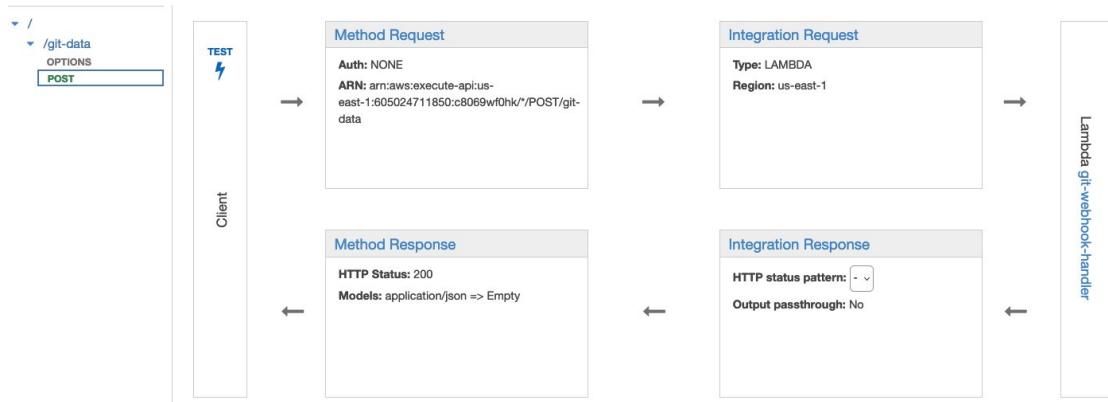
Resource Path* / git-data

You can add path parameters using brackets. For example, the resource path `{username}` represents a path parameter called 'username'. Configuring `/proxy{+}` as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to `/foo`. To handle requests to `/`, add a new ANY method on the `/` resource.

Enable API Gateway CORS

* Required [Cancel](#) [Create Resource](#)

Select **Actions** as **Create Method**, select **POST** and select the tick button. Enter **Lambda Function** name as **git-webhook-handler** and select **Save** button.



Deploy the REST API to **test** stage.

Deploy API



Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage <input type="text" value="New Stage"/>	Stage name* <input type="text" value="test"/>
Stage description <input type="text"/>	
Deployment description <input type="text"/>	

	Cancel	Deploy
--	---------------	---------------

Next select the **Test** button give empty JSON for body and select **Test**

Request Body

1 `{}`

 Test

Validate the data is ingested to Kinesis Successfully.

Request: /git-data

Status: 200

Latency: 1793 ms

Response Body

```
{"ShardId": "shardId-000000000002", "SequenceNumber": "49639829413199610849534838407345601431518270427488059426", "ResponseMetadata": {"RequestId": "c27a58e9-1685-4649-9ec9-33ab91ab2d17", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "c27a58e9-1685-4649-9ec9-33ab91ab2d17", "x-amz-id-2": "fj5g9GZvAgwPYWh5ySC00E0/v8xilmMvGEC/V/juxTVhARQuS0PdmU6YJDVDTj7Sa/P+mXwl7wjavQNoV/RLhToJmaPD8B9", "date": "Fri, 14 Apr 2023 14:53:54 GMT", "content-type": "application/x-amz-json-1.1", "content-length": "110"}, "RetryAttempts": 0}}
```

The screenshot shows the AWS OpenSearch Service Data Viewer interface. At the top, there are tabs for Applications, Monitoring, Configuration, Data viewer (which is selected), and Enhanced fan-out (0). Below the tabs, there's a search bar for 'Shard' containing 'shardId-000000000002'. To the right of the shard ID are fields for 'Starting position' (set to 'Info' and 'Trim horizon') and a 'Get records' button. Under the 'Records (1)' section, it says 'Shard: shardId-000000000002 Starting position: Trim horizon'. A search bar labeled 'Find records' is present. Below the search bar, there are two columns of data: 'Partition key' (1edffff1-ff4b...) and 'Data' ({"order": "2 Icecreams"}), followed by 'Approximate arrival timestamp' (April 14, 2023 at 10:53:54 EDT) and 'Sequence number' (4963982941319961084953483840734560143151...).

1. Lets confirm that Amazon OpenSearch service is created.

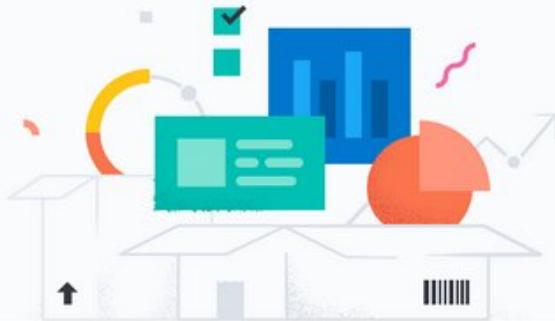
The screenshot shows the AWS OpenSearch Service Domains page. A green header bar indicates 'Successfully updated to service software version R20230308-P2'. Below the header, the URL is 'Amazon OpenSearch Service > Domains > git-logs-search-domain'. The main page title is 'git-logs-search-domain' with an 'Info' link. To the right are 'Delete' and 'Actions' buttons. A 'General information' card displays the following details:

Name	Domain status	Version	OpenSearch Dashboards URL
git-logs-search-domain	Active	OpenSearch 2.5 (latest)	https://search-git-logs-search-domain-dwx5fmmgiksegabm7odltz6di.us-east-1.es.amazonaws.com/_dashboards
Domain ARN	Cluster health	Service software version	Domain endpoint
arn:aws:es:us-east-1:147228461610:domain/git-logs-search-domain	Green	R20230308-P2 (latest)	https://search-git-logs-search-domain-dwx5fmmgiksegabm7odltz6di.us-east-1.es.amazonaws.com

Open the Dashboard for OpenSearch by selecting the URL. Enter the user id and password that you selected in previous step and you should see the welcome page.



Welcome to OpenSearch Dashboards



Start by adding your data

Add data to your cluster from any source, then analyze and visualize it in real time. Use our solutions to add search anywhere, observe your ecosystem, and protect against security threats.

[Add data](#)

[Explore on my own](#)

Select Explore on my own link and then Global and select Confirm button. You see the home page.

8. Create another **Kinesis Data Delivery Stream** to stream the Kinesis git data to **Amazon OpenSearch service**. Select the source as **Amazon Kinesis Data Stream** and destination as **Amazon OpenSearch Service**. In Source settings select the Kinesis data stream that we created previously **git-data-stream**. Enter the name as **git-data-delivery-opensearch-stream**.

Amazon Kinesis > Delivery streams > Create delivery stream

Create delivery stream Info

▶ Amazon Kinesis Data Firehose: How it works

Choose source and destination
Specify the source and the destination for your delivery stream. You cannot change the source and destination of your delivery stream once it has been created.

Source Info
Amazon Kinesis Data Streams

Destination Info
Amazon OpenSearch Service

Source settings

Kinesis data stream
arn:aws:kinesis:us-east-1:605024711850:stream/git-data-stream Browse Create

Format: arn:aws:kinesis:[Region]:[AccountId]:stream/[StreamName]

Delivery stream name

Delivery stream name
git-data-delivery-opensearch-stream

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.

In Destination settings select the OpenSearch that we created earlier. Enter **gitlogs** as **Index**

Destination settings [Info](#)
 Specify the destination settings for your delivery stream.

OpenSearch Service domain

[Browse](#)

[Create domain](#)

Format: arn:aws:es:[Region]:[AccountId]:domain/[DomainName]

Index
 Specify the OpenSearch Service index name to be used when indexing data to your OpenSearch Service cluster.

gitlogs

The index name must be between 1 and 80 characters long.

Index rotation

No rotation

Type
 Specify the OpenSearch Service type name to be used when indexing data to your OpenSearch Service cluster.

Enter a type name

The type name must be between 1 and 100 characters long.

Retry duration
 Specify how long Kinesis Data Firehose retries sending data to OpenSearch Service.

seconds

Minimum: 0 seconds, maximum: 7200 seconds. Recommended: 300 seconds.

Expand **Buffer hints** and enter 60 as **Buffer Interval**.

▼ Buffer hints
 The fields below are pre-populated with the recommended default values for OpenSearch Service. Pricing may vary depending on storage and request costs.

Buffer hints
 Kinesis Data Firehose buffers incoming records before delivering them to your OpenSearch Service domain.

Buffer size
 The higher buffer size may be lower in cost with higher latency. The lower buffer size will be faster in delivery with higher cost and less latency.

MiB

Minimum: 1 MiB, maximum: 100 MiB. Recommended: 5 MiB.

Buffer interval
 The higher interval allows more time to collect data and the size of data may be bigger. The lower interval sends the data more frequently and may be more advantageous when looking at shorter cycles of data activity.

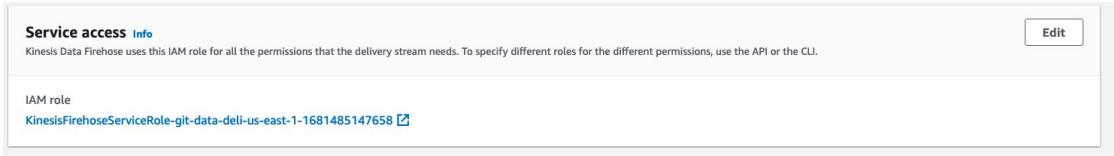
seconds

Minimum: 60 seconds, maximum: 900 seconds. Recommended: 300 seconds.

For **Backup settings** selected **Failed data only** and create a S3 bucket to save the failed data so that you will not lose them.

1. Now we will give Kinesis Firehose IAM role permission to create index on Amazon OpenSearch by mapping IAM Role with Amazon OpenSearch backend role.

Therefore, open Amazon Kinesis delivery stream that we created in previous step (**git-data-delivery-opensearch-stream**). Select configuration tab, in Service access section select IAM role.



Copy the ARN value of this IAM role.

IAM > Roles > KinesisFirehoseServiceRole-git-data-deli-us-east-1-1681485147658

KinesisFirehoseServiceRole-git-data-deli-us-east-1-1681485147658 Delete

Summary Edit

Creation date: April 14, 2023, 11:42 (UTC-04:00)

Last activity: None

Maximum session duration: 1 hour

ARN Copied

arn:aws:iam::147228461610:role/service-role/KinesisFirehoseServiceRole-git-data-deli-us-east-1-1681485147658

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

open Amazon OpenSearch Dashboard and go to security page, navigate to Roles pane, select all_access, go to Mapped users tab, select Manage Mappings and enter IAM role ARN of Amazon Kinesis delivery streaming in the **Backend roles** and select Map button.

OpenSearch Dashboards

☰ Security / Roles / all_access / Map user

Map user

Map users to this role to inherit role permissions. Two types of users are supported: user, and backend role. [Learn more](#)

Users

You can create an internal user in internal user database of the security plugin. An internal user can have its own backend role and host for an external authentication and authorization. External users from your identity provider are also supported. [Learn more](#)

admin Create new internal user

Look up by user name. You can also create new internal user or enter external user.

Backend roles

Use a backend role to directly map to roles through an external authentication system. [Learn more](#)

Backend roles

arn:aws:iam::147228461610:role/service-role/KinesisFirehoseServiceRole-git-data-deli-us-east-1-1681485147658 Remove

Add another backend role

Cancel Map

10. Test data delivery make sure data is delivered to OpenSearch service.

Go to API Gateway and click the test button again to send the data to Amazon Kinesis.

Method Execution /git-data - POST - Method Test

Path: Request: /git-data
Status: 200
Latency: 1876 ms
Response Body:

```
{"ShardId": "shardId-000000000003", "SequenceNumber": "4963982941322191159473369080726467435516308387193159730", "ResponseMetadata": {"RequestId": "f3c6d6db-73dc-593f-af75-1444b358e721", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "f3c6d6db-73dc-593f-af75-1444b358e721", "x-amz-id-2": "bm6c+faA2TTzFB1fsnAKB0SCMDxnp62DyHRSrPB7eyLUZ9v6hrILZMhf4XGUiK2/MQ9W0JUgL6g8TQsHe4gay/u54kIek", "date": "Fri, 14 Apr 2023 15:53:19 GMT", "content-type": "application/x-amz-json-1.1", "content-length": "110"}, "RetryAttempts": 0}}
```

Notice data ingested to ShardId - 3 this time.

Partition key	Data	Approximate arrival timestamp	Sequence number
605b3d67-907...	{"order": "2 icecreams"}	April 14, 2023 at 11:53:19 EDT	4963982941322191159473369080726467435516308

Lets confirm the data got streamed to AmazonOpenSearch after 1 minute. Go to Amazon OpenSearch, select Stack Management, Index Patterns and select Create Indesx pattern button. Enter **gitlogs** as your index name, select **Next step** and **Create index pattern**.

This page lists every field in the **gitlogs** index and the field's associated core type as recorded by OpenSearch. To change a field type, use the OpenSearch Mapping API.

Name	Type	Format	Searchable	Aggregatable	Excluded
<code>_id</code>	string		●	●	ⓘ
<code>_index</code>	string		●	●	ⓘ
<code>_score</code>	number				ⓘ
<code>_source</code>	_source				ⓘ
<code>_type</code>	string				ⓘ
<code>order</code>	string		●		ⓘ
<code>order.keyword</code>	string		●	●	ⓘ

Rows per page: 10 < 1 >

Next click on **Discover** page and you will notice the record entry is present in the dashboard.

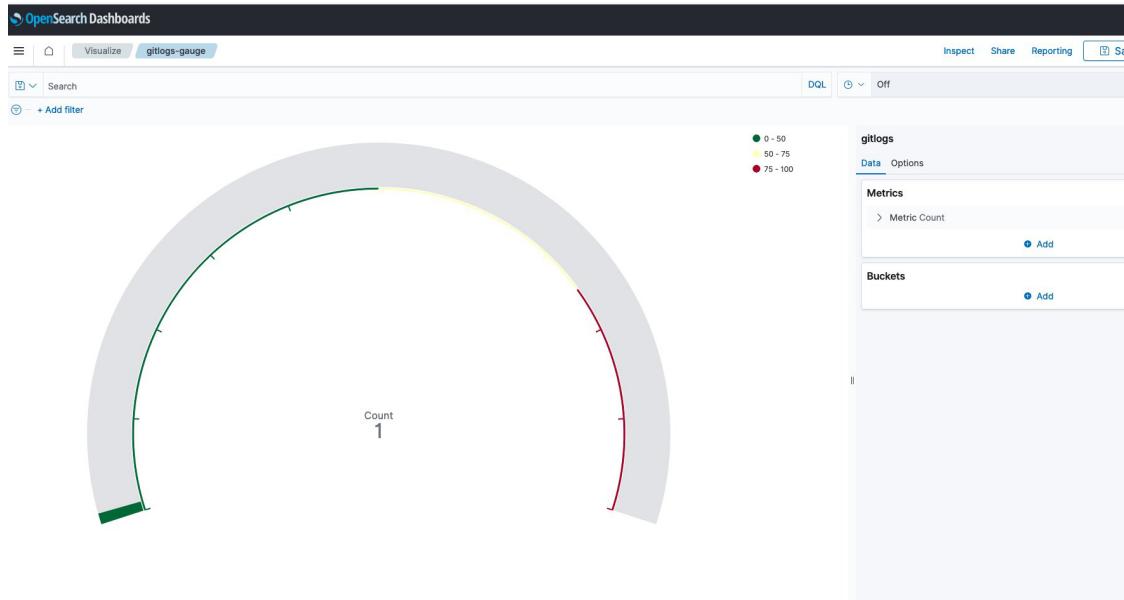
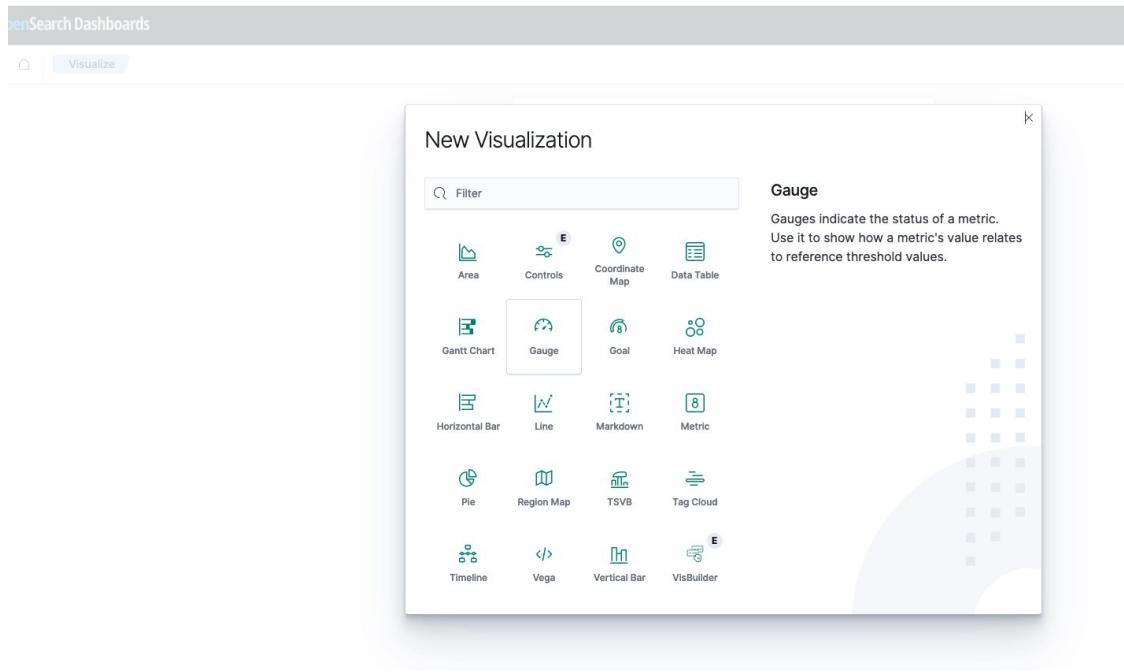
1 hit

```

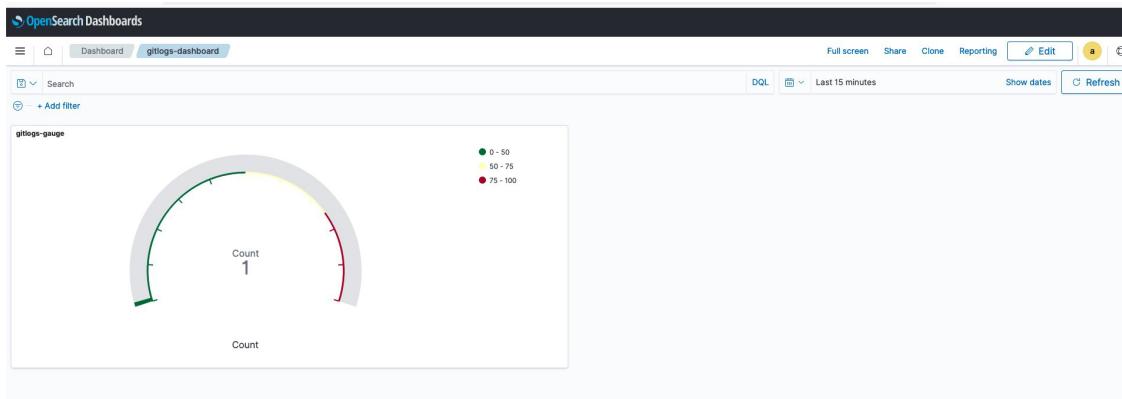
{
  "_index": "gitlogs",
  "_id": "49639829413221911594733369080726467435516308387193159730.0",
  "_type": "gitlogs",
  "_score": 0,
  "_source": {
    "order": "2 Icecreams"
  }
}

```

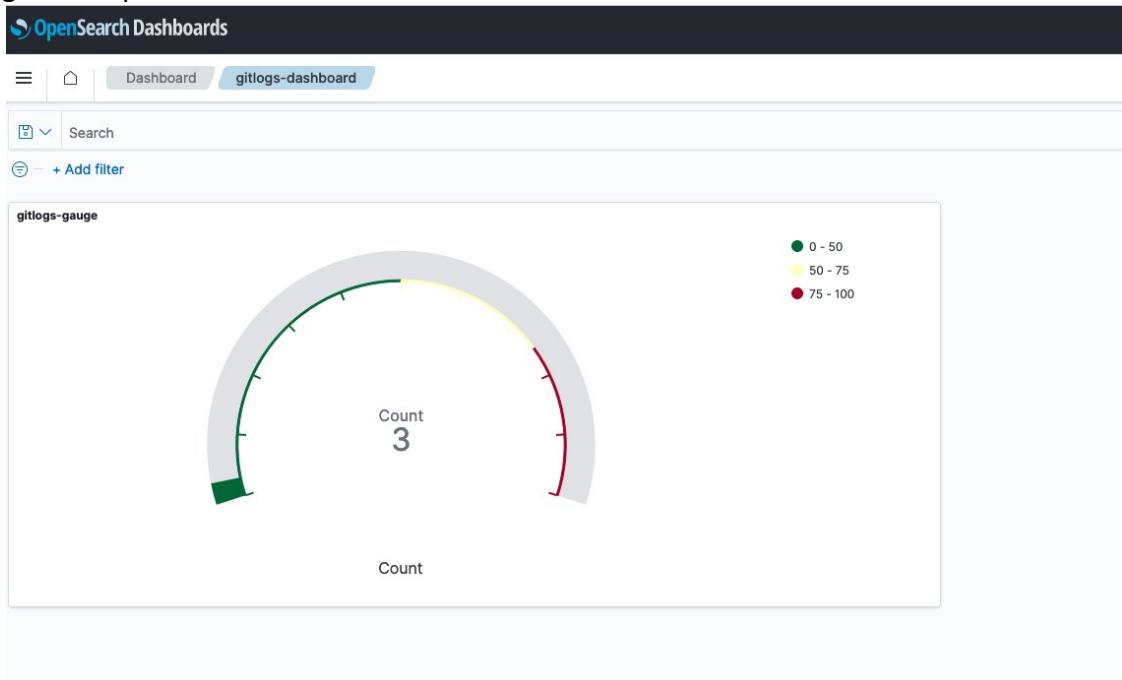
Next select Visualize, Gauge and select gitlogs and Save it, give name as **gitlogs-gauge**.



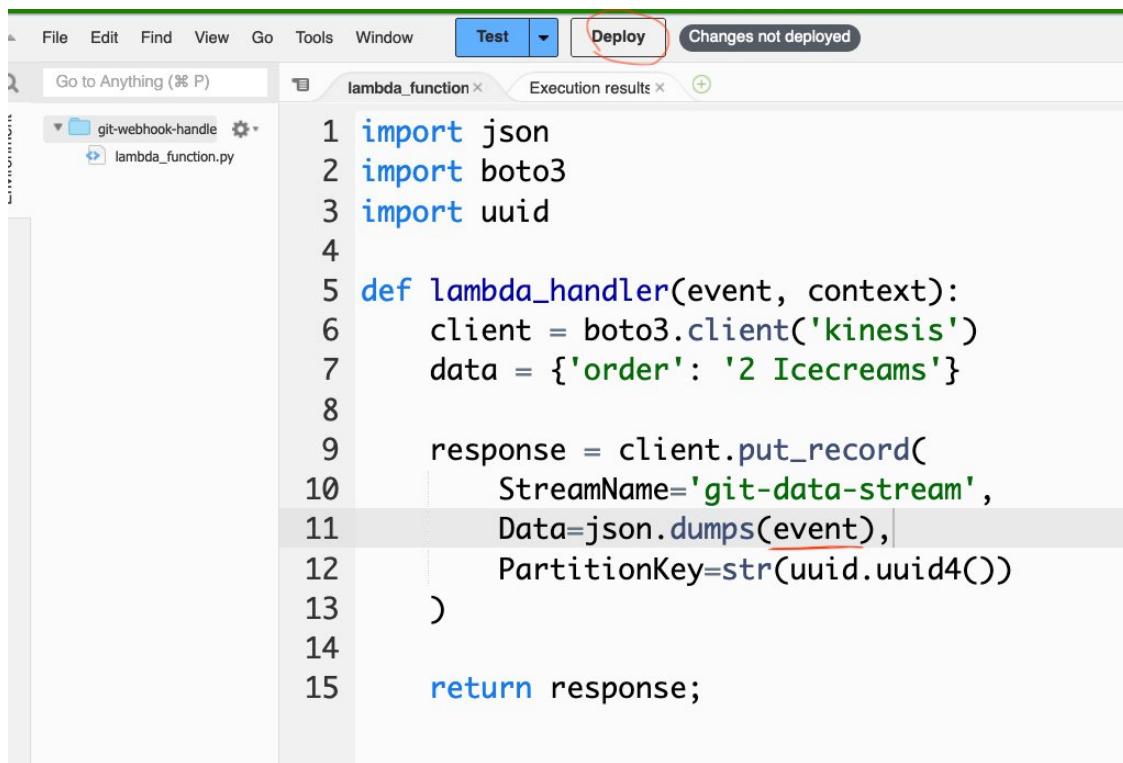
Next select Dashboard, Create new dashboard, Select Add an existing, select gitlogs-gauge, Save it, give name as gitlogs-dashboard.



Now go ahead and test REST API 2 more times to confirm the dashboard count is changed to 3. Navigate to OpenSearch and refresh the dashboard.



11. Next update the AWS Lambda to pass the event object rather hardcoded data. This way we can stream the GitHub event coming via API Gateway. Go to AWS lambda change the code and deploy it.



The screenshot shows a code editor interface with a toolbar at the top. The toolbar includes File, Edit, Find, View, Go, Tools, Window, Test, Deploy (which is circled in red), and Changes not deployed. Below the toolbar, there's a navigation bar with Go to Anything (% P) and tabs for lambda_function and Execution results. On the left, there's a sidebar with a file tree showing a folder named git-webhook-handle containing a file named lambda_function.py. The main area displays the following Python code:

```
1 import json
2 import boto3
3 import uuid
4
5 def lambda_handler(event, context):
6     client = boto3.client('kinesis')
7     data = {'order': '2 Icecreams'}
8
9     response = client.put_record(
10         StreamName='git-data-stream',
11         Data=json.dumps(event),
12         PartitionKey=str(uuid.uuid4())
13     )
14
15     return response;
```

Create new test data

Test event action

Create new event Edit saved event

Event name

test-data-nonempty

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

test-data

Event JSON [Format JSON](#)

```

1 {
2   "order": "20 oranges"
3 }
```

Cancel **Save**

And test this data notice it got ingested successfully to shard 1.

The screenshot shows the AWS Lambda Test Event results page. The test event name is "test-data-nonempty". The response details show the event was successfully processed by shard 1. The response body includes the original event payload and some internal metadata. The function logs at the bottom show the request ID, start time, end time, duration, memory usage, and init duration.

```

{
  "shardId": "shardId-000000000001",
  "sequenceNumber": "49639829413177310104336307834445813850610315414443393042",
  "responseMetadata": {
    "RequestId": "e81d826d-7fd1-0231-b4ae-ff071c58d672",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "e81d826d-7fd1-0231-b4ae-ff071c58d672",
      "x-amz-id-2": "AInjxLPGTzvJmXobefgwT0W97Rg0Fn2h3Rg+ytce566jPe3RyDVej/qM2Mi6mFwR3iGYTCNFmtSIkmI13lZgE14PzjAMk",
      "date": "Fri, 14 Apr 2023 16:11:22 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "retryAttempts": 0
  }
}

Function Logs
START RequestId: 4263d301-8701-44de-a49c-e5ea734d4b0 Version: $LATEST
END RequestId: 4263d301-8701-44de-a49c-e5ea734d4b0
REPORT RequestId: 4263d301-8701-44de-a49c-e5ea734d4b0 Duration: 1353.56 ms Billed Duration: 1354 ms Memory Size: 128 MB Max Memory Used: 68 MB Init Duration: 225.42 ms

Request ID
4263d301-8701-44de-a49c-e5ea734d4b0

```

Data ingested in Kinesis as well.

Record data

X

Sequence number 49639829413177310104336307834445813850610315414443393042
Shard ID shardId-000000000001

Raw data

JSON

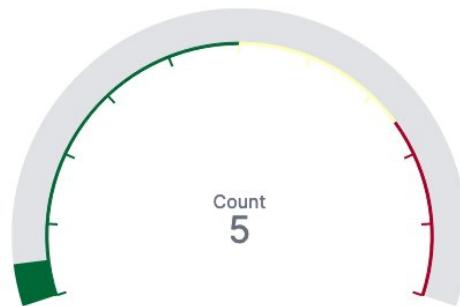
Copy

```
{  
  "order": "20 oranges"  
}
```

OpenSearch count is updated as well

⟳ + Add filter

gitlogs-gauge



- 0 - 50
- 50 - 75
- 75 - 100

Count

		5 hits	
_source			
>	order: 2 Icecreams	_id:	49639829413155009359137777161064947846612494951882162178.0
>	order: 2 Icecreams	_id:	49639829413221911594733369080726467435516308387193159730.0
>	order: 2 Icecreams	_id:	49639829413221911594733369080727676361335967890186174514.0
▼	order: 20 oranges	_id:	49639829413177310104336307834445813850610315414443393042.0
		Expanded document	
		Table	JSON
	t _id	49639829413177310104336307834445813850610315414443393042.0	
	t _index	gitlogs	
	# _score	0	
	t _type	-	
	t order	20 oranges	
>	order: 2 Icecreams	_id:	49639829413199610849534838407346810357338172647672905762.0

12. Next lets update the GitHub Repository Webhook URL to AWS REST API and stream real GitHub events.

Copy the POST URL from REST API

The screenshot shows the AWS Lambda function configuration interface. The top navigation bar includes 'APIs > git-webhook-handler-api (r7peor1q8d) > Stages > test > /git-data > POST'. The left sidebar shows 'Stages' with 'test' expanded, revealing sub-stages like '/' and '/git-data' with a 'POST' method selected. The main panel title is 'test - POST - /git-data'. It displays the 'Invoke URL' as <https://r7peor1q8d.execute-api.us-east-1.amazonaws.com/test/git-data>. Below this, a note says 'Use this page to override the `test stage` settings for the POST to /git-data method.' There are two radio button options under 'Settings': ' Inherit from stage' and ' Override for this method'.

Go to GitHub open existing repository or create a new one then go to the **Settings**, select **Webhooks** , select **Add Webhook**, enter AWS REST API URL, change content type to **application/json** and select Add webhook button.

The screenshot shows the GitHub repository settings page with the 'Webhooks' section selected. The left sidebar lists various settings categories like General, Access, Collaborators, etc., with 'Webhooks' currently active. The main right area is titled 'Webhooks / Add webhook'. It contains fields for 'Payload URL' (set to 'peor1q8d.execute-api.us-east-1.amazonaws.com/test/git-data'), 'Content type' (set to 'application/json'), and a 'Secret' field. Under 'SSL verification', the 'Enable SSL verification' option is selected. In the 'Which events would you like to trigger this webhook?' section, the 'Just the push event.' option is selected. A checked 'Active' checkbox indicates the webhook will deliver event details. A prominent green 'Add webhook' button is at the bottom.

Payload URL *

'peor1q8d.execute-api.us-east-1.amazonaws.com/test/git-data'

Content type

application/json

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

Enable SSL verification Disable (not recommended)

Which events would you like to trigger this webhook?

Just the push event.

Send me everything.

Let me select individual events.

Active

We will deliver event details when this hook is triggered.

Add webhook

Next select the Star button in the GitHub repository. Open Webhook

Recent Deliveries

✓  2484dd10-dae0-11ed-93ab-71efceda4d3d ping 2023-04-14 12:19:36 ...

Request Response **200** Redeliver  Completed in 1.77 seconds.

Headers

```
Content-Length: 534
Content-Type: application/json
Date: Fri, 14 Apr 2023 16:19:36 GMT
X-Amz-Apigw-Id: DYB7lGjNoAMFx8Q=
X-Amzn-Requestid: 6627a460-b306-4c73-b896-5f5f3f734160
X-Amzn-Trace-Id: Root=1-64397d16-362052aa2b9f7f7c407b3927;Sampled=0
```

Body

```
{"ShardId": "shardId-000000000000", "SequenceNumber": "496398294131550093591377771610661567724321}
```

Open Kinesis and confirm git log is arrived.

Record data	
Sequence number	49639829413155009359137777161066156772432171016269070338
Shard ID	shardId-000000000000
Raw data	JSON
	Copy
{ "zen": "Design for failure.", "hook_id": 409875732, "hook": { "type": "Repository", "id": 409875732, "name": "web", "active": true, "events": ["push"], "config": { "content_type": "json", "insecure_ssl": "0", "url": "https://r7peor1q8d.execute-api.us-east-1.amazonaws.com/test/git-data" }, "updated_at": "2023-04-14T16:19:33Z", "created_at": "2023-04-14T16:19:33Z", "url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732", "test_url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732/test", "ping_url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732/pings", "deliveries_url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732/deliveries", "last_response": { "code": null, "status": "unused", "message": null }, "repository": { "id": 623170224, "node_id": "R_kgDOJSTsA", "name": "kinesis-streaming-webhook", "full_name": "rupeshtiwari/kinesis-streaming-webhook", "private": false, "owner": { "login": "rupeshtiwari", "id": 330383, "node_id": "MDQ6VXNlcjMzMDM4Mw==" }, "avatar_url": "https://avatars.githubusercontent.com/u/330383?v=4", "gravatar_id": "" } }	

Open Amazon OpenSearch service and confirm git log is arrived.

OpenSearch Dashboards

Discover

Expanded document

Table JSON

```
{
  "_index": "gitlogs",
  "_id": "4963982941315500935913777161066156772432171016269870338.0",
  "_version": 1,
  "_score": 0,
  "_source": {
    "zen": "Design for failure.",
    "hook_id": 409875732,
    "hook": {
      "type": "Repository",
      "id": 409875732,
      "name": "web",
      "active": true,
      "events": [
        "push"
      ],
      "config": {
        "content_type": "json",
        "insecure_ssl": "0",
        "url": "https://r7peor1q8d.execute-api.us-east-1.amazonaws.com/test/git-data"
      },
      "updated_at": "2023-04-14T16:19:33Z",
      "created_at": "2023-04-14T16:19:33Z",
      "url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732",
      "test_url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732/test",
      "ping_url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732/pings",
      "deliveries_url": "https://api.github.com/repos/rupeshtiwari/kinesis-streaming-webhook/hooks/409875732/deliveries",
      "last_response": {
        "code": null,
        "status": "unused",
        "message": null
      }
    },
    "repository": {
      "id": 623170224,
      "node_id": "R_kgDOJSTSSA",
      "name": "kinesis-streaming-webhook",
      "full_name": "rupeshtiwari/kinesis-streaming-webhook",
      "private": false,
      "owner": {
        "login": "rupeshtiwari",
        "id": 398983
      }
    }
  }
}
```

Search

+ Add filter

gitlogs-gauge

Count 6

Count

● 0 - 50
● 50 - 75
● 75 - 100

Open S3 bucket and verify the git log is arrived

Amazon S3 > Buckets > rt-git-data > 2023/ > 04/ > 14/ > 16/

16/

Objects Properties

Objects (5)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	git-data-delivery-stream-1-2023-04-14-16-19-36-f49f2fa3-d4cb-48e6-b814-3e50ada7c2b8	-	April 14, 2023, 12:20:59 (UTC-04:00)
<input type="checkbox"/>	git-data-delivery-stream-1-2023-04-14-16-11-22-a9546a64-79c7-409a-949f-fd1287a50226	-	April 14, 2023, 12:12:24 (UTC-04:00)
<input type="checkbox"/>	git-data-delivery-stream-1-2023-04-14-16-04-42-430f43af-0b15-4400-ad8e-69f9e782034a	-	April 14, 2023, 12:05:46 (UTC-04:00)
<input type="checkbox"/>	git-data-delivery-stream-1-2023-04-14-16-04-11-7df1bcd-af78-49c6-95ad-4197eb99bcb7	-	April 14, 2023, 12:05:14 (UTC-04:00)
<input type="checkbox"/>	git-data-delivery-stream-1-2023-04-14-16-03-38-dea4bcf8-81d7-4892-af8e-82bc20e75cdf	-	April 14, 2023, 12:04:40 (UTC-04:00)

Query results

Query results are not available after you choose Close or navigate away. Choose [Download results](#) to download a copy of the following query results.

Status
Successfully returned 1 record in 196 ms
Bytes returned: 7836 B

[Raw](#) [Formatted](#)

```
{"zen": "Design for failure.", "hook_id": 409875732, "hook": {"type": "Repository", "id": 409875732, "name": "web", "active": true, "events": ["push"], "config": {"content_
```