
Clickstream Analytics on AWS

Implementation Guide

Clickstream Analytics on AWS: Implementation Guide

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

| | |
|--|----|
| Solution Overview | 1 |
| Features and benefits | 2 |
| Use cases | 2 |
| Terms and concepts | 3 |
| Architecture overview | 5 |
| Architecture diagram | 5 |
| Solution end-to-end architecture | 5 |
| Ingestion module | 6 |
| Data processing module | 7 |
| Data modeling module | 7 |
| Reporting module | 9 |
| AWS Well-Architected pillars | 9 |
| Operational excellence | 9 |
| Security | 10 |
| Reliability | 10 |
| Performance efficiency | 10 |
| Cost optimization | 10 |
| Sustainability | 11 |
| Architecture details | 12 |
| Solution components | 12 |
| Web console | 12 |
| SDKs | 12 |
| Data pipeline | 12 |
| AWS services in this solution | 13 |
| Plan your deployment | 15 |
| Cost | 15 |
| Ingestion module | 15 |
| Data processing & modeling modules | 16 |
| Reporting module | 9 |
| Logging and monitoring | 18 |
| Additional features | 18 |
| Security | 19 |
| IAM Roles | 19 |
| Amazon VPC | 19 |
| Security Groups | 19 |
| Amazon CloudFront | 19 |
| Supported AWS Regions | 20 |
| Deployment | 24 |
| Prerequisites | 24 |
| Deployment in AWS Regions | 24 |
| Deployment in AWS China Regions | 24 |
| Deployment within Amazon VPC | 24 |
| Launch with Cognito User Pool | 24 |
| Deployment Overview | 24 |
| Step 1: Launch the Stack | 25 |
| Step 2: Launch the web console | 26 |
| Launch with OpenID Connect (OIDC) | 27 |
| Prerequisites | 27 |
| Deployment overview | 27 |
| Step 1. Create OIDC client | 27 |
| Step 2. Launch the Stack | 31 |
| Step 3. Update the callback URL of OIDC client | 33 |
| Step 4. Setup DNS Resolver | 34 |
| Step 4. Launch the Web Console | 34 |

| | |
|--|----|
| Launch within VPC | 34 |
| Prerequisites | 34 |
| Deployment Overview | 34 |
| Step 1. Create OIDC client | 35 |
| Step 2. Launch the stack | 35 |
| Step 3. Update the callback URL of OIDC client | 36 |
| Step 4. Launch the web console | 36 |
| Getting started | 37 |
| Step 1: Create a project | 37 |
| Prerequisites | 37 |
| Steps | 37 |
| Next | 38 |
| Step 2: Configure data pipeline | 38 |
| Steps | 38 |
| Next | 39 |
| Step 3: Integrate SDK | 39 |
| Steps | 39 |
| Generate sample data | 40 |
| Next | 41 |
| Step 4: View dashboard | 41 |
| Steps | 41 |
| Next | 41 |
| Pipeline management | 42 |
| Prerequisites | 42 |
| Ingestion | 42 |
| Ingestion endpoint | 43 |
| Data sink – Kafka | 45 |
| Data sink – Kinesis | 45 |
| Data sink – S3 | 46 |
| Data processing | 46 |
| Data schema | 47 |
| Execution parameters | 53 |
| Processing plugin | 53 |
| Data modeling | 55 |
| Preset data views | 55 |
| Reporting | 56 |
| Pipeline maintenance | 57 |
| Monitoring and Alarms | 57 |
| Pipeline modification | 57 |
| Pipeline upgrade | 58 |
| SDK manual | 59 |
| Key features and benefits | 59 |
| Android SDK | 59 |
| Introduction | 59 |
| Integrate the SDK | 59 |
| Data format definition | 61 |
| Preset events and attributes | 63 |
| Swift SDK | 68 |
| Introduction | 68 |
| Integrate the SDK | 68 |
| Data format definition | 71 |
| Preset events and attributes | 72 |
| Dashboards | 79 |
| Overview | 79 |
| View dashboards | 79 |
| Reports | 79 |
| Custom report | 80 |

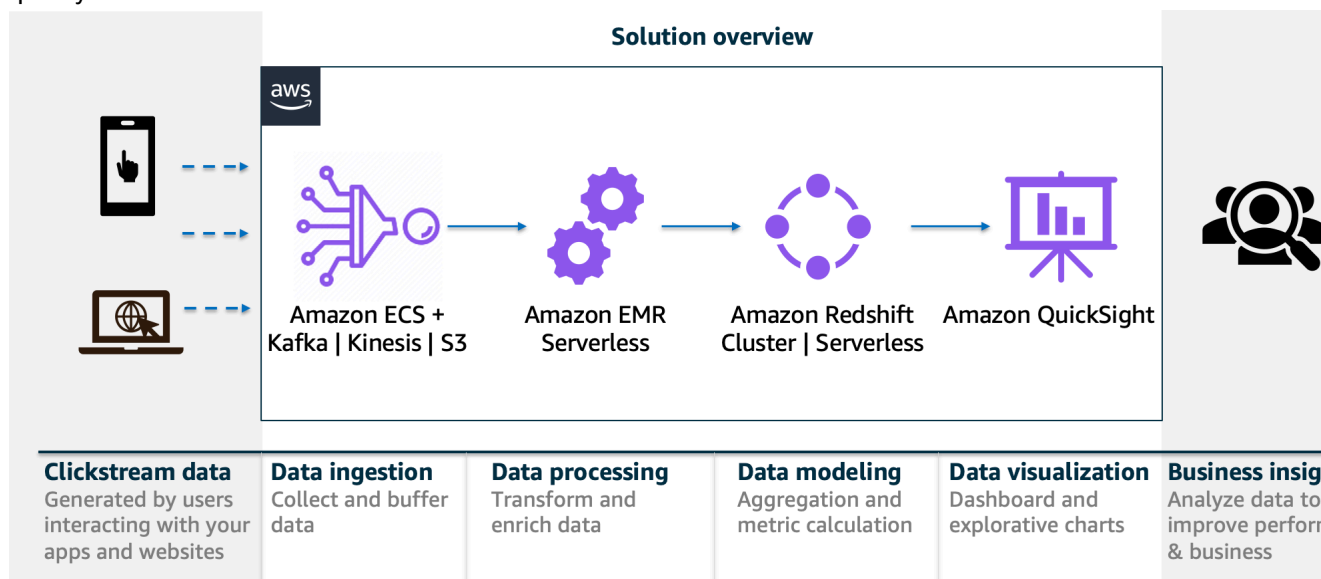
| | |
|--|-----|
| Acquisition report | 80 |
| View the report | 80 |
| Where the data comes from | 80 |
| Dimensions and metrics | 81 |
| Sample dashboard | 82 |
| Engagement report | 84 |
| View the report | 84 |
| Where the data comes from | 84 |
| Dimensions and metrics | 87 |
| Sample dashboard | 88 |
| Activity report | 90 |
| View the report | 90 |
| Where the data comes from | 90 |
| Dimensions and metrics | 92 |
| Sample dashboard | 93 |
| Retention report | 95 |
| View the report | 95 |
| Where the data comes from | 95 |
| Dimensions and metrics | 98 |
| Sample dashboard | 99 |
| Device report | 101 |
| View the report | 101 |
| Where the data comes from | 101 |
| Dimensions and metrics | 103 |
| Sample dashboard | 104 |
| Path explorer | 106 |
| View the report | 106 |
| Where the data comes from | 106 |
| Dimensions and metrics | 107 |
| Sample dashboard | 108 |
| Custom report | 109 |
| Steps | 109 |
| Frequently Asked Questions | 112 |
| General | 112 |
| SDK | 112 |
| Setup and configuration | 112 |
| Pricing | 112 |
| Troubleshooting | 113 |
| Problem: Deployment failure due to "Invalid Logging Configuration: The CloudWatch Logs Resource Policy size was exceeded" | 113 |
| Problem: Cannot delete the CloudFormation stacks created for the Clickstream pipeline | 114 |
| Problem: Can not sink data to MSK cluster, got "InvalidReplicationFactor (Broker: Invalid replication factor)" log in Ingestion Server | 114 |
| Uninstall the solution | 115 |
| Step 1. Delete projects | 115 |
| Step 2. Delete Clickstream Analytics on AWS stack | 115 |
| Additional resources | 116 |
| Upload SSL Certificate to IAM | 116 |
| Developer guide | 117 |
| Source code | 117 |
| Contributors | 118 |
| Revisions | 119 |
| Notices | 120 |

An end-to-end solution to collect, ingest, analyze, and visualize clickstream data inside your web and mobile applications

Publication date: *July 2023*

The Clickstream Analytics on AWS solution allows you to collect, ingest, process and analyze clickstream data from websites and mobile applications to drive your business growth. You can use the solution to spin up an analytics platform that fits your organizational needs, and maintain complete ownership and control over your valuable user behavior data. The solution can be applied to various use cases such as user behavior analysis and marketing analysis to improve website and application's performance.

The solution provides modularized and configurable components of a data pipeline so that you can choose and customize the components to accelerate the building of a Well-Architected data pipeline from weeks to minutes. The purpose-built SDKs and guidance allow you to collect client-side data from different application platforms (for example, Android, iOS, and JavaScript) to AWS. In addition, ready-to-use metrics and visualization templates enable you to derive actionable business insights easily and quickly.



Use this navigation table to quickly find answers to these questions:

| If you want to ... | Read... |
|--|----------------------------------|
| Know the cost for running this solution | Cost (p. 15) |
| Understand the security considerations for this solution | Security (p. 19) |

| If you want to ... | Read... |
|--|---|
| Know which AWS Regions are supported for this solution | Supported AWS Regions (p. 20) |
| Get started with the solution quickly to build an end-to-end data pipeline, send data into the pipeline, and then view the out-of-the-box dashboards | Getting started (p. 37) |
| Learn the concepts related to pipeline, and how to manage a data pipeline throughout its lifecycle | Pipeline management (p. 42) |

This guide is intended for IT architects, developers, DevOps, data analysts, and marketing technology professionals who have practical experience architecting in the AWS Cloud.

Features and benefits

The solution includes the following key features and benefits:

- **Visual data pipeline builder**

You can simply define the data pipeline from a web-based UI console. The solution will take care of the underlying infrastructure creation, required security setup, and data integrations. Each pipeline module is built with various features and designed to be loosely-coupled, making it flexible for you to customize for specific use cases.

- **Purposed-built SDKs**

The SDKs are optimized for collecting data from Android, iOS, and JavaScript platforms, which automatically collect common events (for example, first visit, screen view), support built-in local cache, retry, and verification mechanisms to ensure high completeness of data transmission.

- **Out-of-the-box dashboard**

The solution offers a dozen of built-in visualizations (for example, acquisition, engagement, retention, and user demographic) and exploratory reporting templates (for example, user details, event details), powering various critical business analytics use cases such as user behavior analytics, marketing analytics, and product analytics.

Use cases

Clickstream data play a pivotal role in numerous online business analytics use cases, and the Clickstream Analytics on AWS can be applied to the following:

- **User behavior analytics**

Clickstream data in user behavior analytics provides insights into the sequential and chronological patterns of user interactions on a website or application, helping businesses understand user navigation, preferences, and engagement levels to enhance the overall product experience and drive product innovation.

- **First-party customer data platform (CDP)**

Clickstream data, together with other business data sources (for example, order history, and user profile), allow customers to create a first-party customer data platform that offers a comprehensive

view of their users, enabling businesses to personalize customer experiences, optimize customer journeys, and deliver targeted marketing messages.

- **Marketing analytics**

Clickstream data in marketing analytics offers detailed information about users' click paths and interactions with marketing campaigns, enabling businesses to measure campaign effectiveness, optimize marketing strategies, and enhance conversion rates.

Terms and concepts

This section describes key concepts and defines terminology specific to this solution:

Project

A project in this solution is the top-level entity, like a container, that groups your apps and data pipeline for collecting and processing clickstream data. One project contains one data pipeline, and can have one or more apps registered to it.

Data pipeline

A data pipeline is deployed into one AWS region, which means all the underlining resources are created in one AWS region. A data pipeline in this solution contains four modules:

- **Ingestion server:** a web service that provides an endpoint to collect data through HTTP requests, and sink the data in streaming services or S3.
- **Data processing:** a module that transforms raw data to the solution schema and enriches data with additional dimensions.
- **Data modeling:** a module that aggregates data to calculate metrics for business analytics.
- **Reporting:** a module that creates metrics and out-of-the-box visualizations in QuickSight.

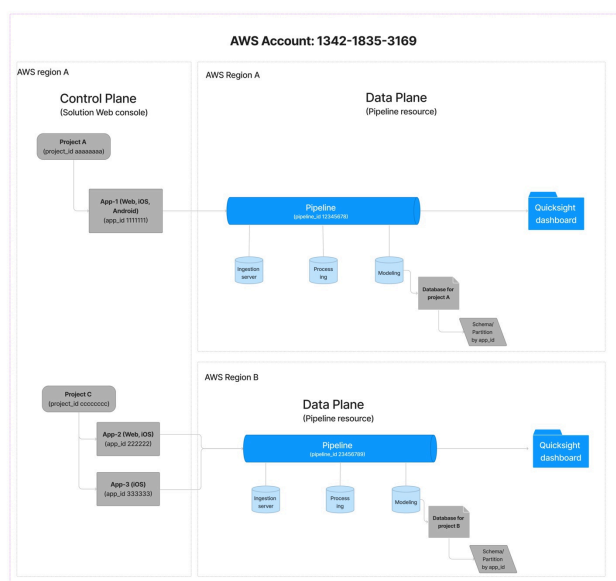
App

An app in this solution can represent an application in your business, which might be built on one or multiple platforms (for example, Android, iOS, and Web).

Dashboard

For each app registered in the dashboard, the solution will create a QuickSight dashboard for each App in the AWS region.

Below is a diagram to help you better understand those concepts and their relationship with each other in the AWS context.



For a general reference of AWS terms, see the [AWS glossary](#) in the *AWS General Reference*.

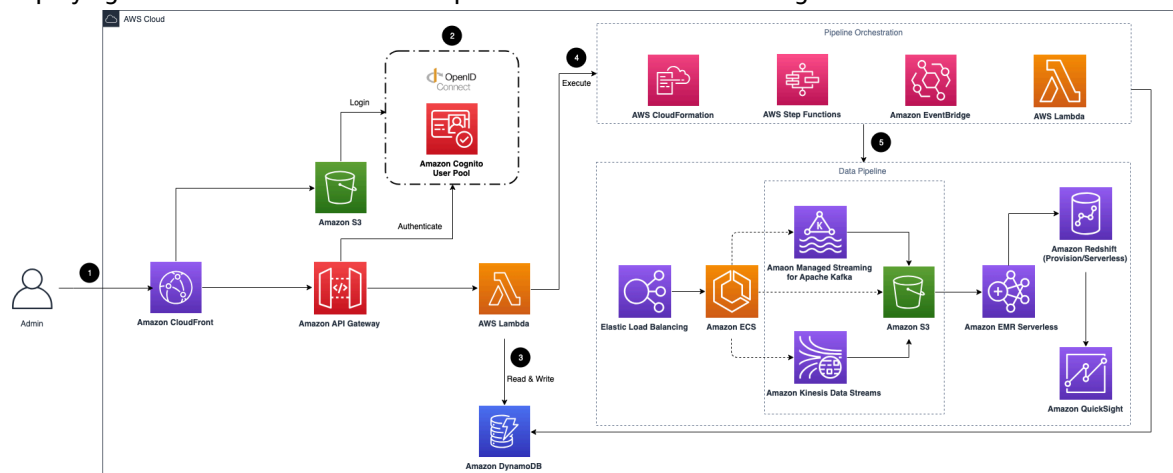
Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution.

Architecture diagram

Solution end-to-end architecture

Deploying this solution with the default parameters builds the following environment in AWS:



Clickstream Analytics on AWS architecture

This solution deploys the AWS CloudFormation template in your AWS Cloud account and completes the following settings.

1. [Amazon CloudFront](#) distributes the frontend web UI assets hosted in the [Amazon S3](#) bucket, and the backend APIs hosted with [Amazon API Gateway](#) and [AWS Lambda](#).
2. The [Amazon Cognito](#) user pool or OpenID Connect (OIDC) is used for authentication.
3. The web UI console uses [Amazon DynamoDB](#) to store persistent data.
4. [AWS Step Functions](#), [AWS CloudFormation](#), AWS Lambda, and [Amazon EventBridge](#) are used for orchestrating the lifecycle management of data pipelines.
5. The data pipeline is provisioned in the region specified by the system operator. It consists of [Application Load Balancer](#), [Amazon ECS](#), [Amazon Managed Streaming for Apache Kafka \(Amazon MSK\)](#), [Amazon Kinesis](#) Data Streams, Amazon S3, [Amazon EMR](#) Serverless, [Amazon Redshift](#), and [Amazon QuickSight](#).

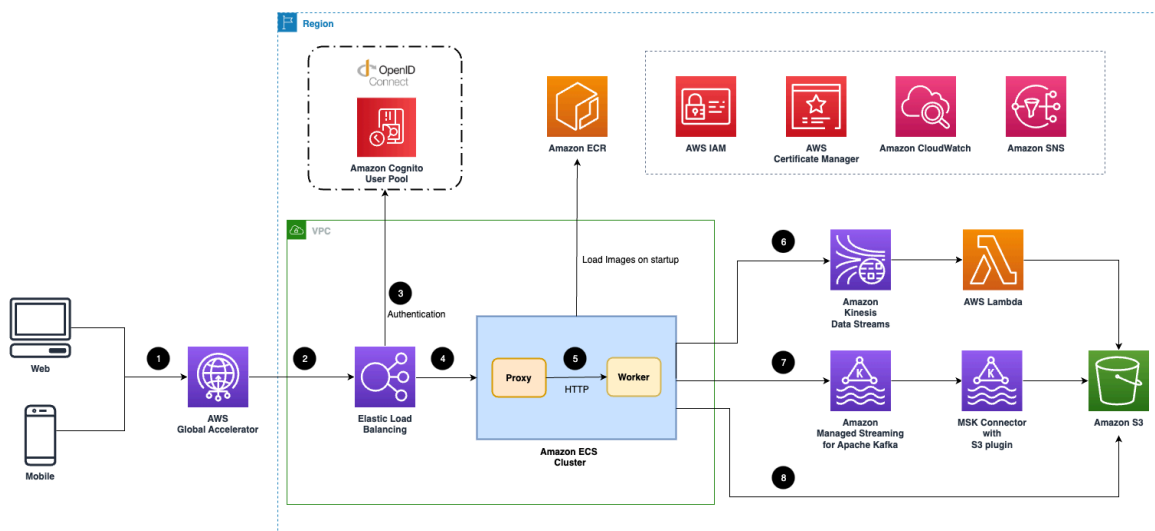
The key functionality of this solution is to build a data pipeline to collect, process, and analyze their clickstream data. The data pipeline consists of four modules:

- Ingestion module

- Data processing module
- Data modeling module
- Reporting module

The following introduces the architecture diagram for each module.

Ingestion module



Ingestion module architecture

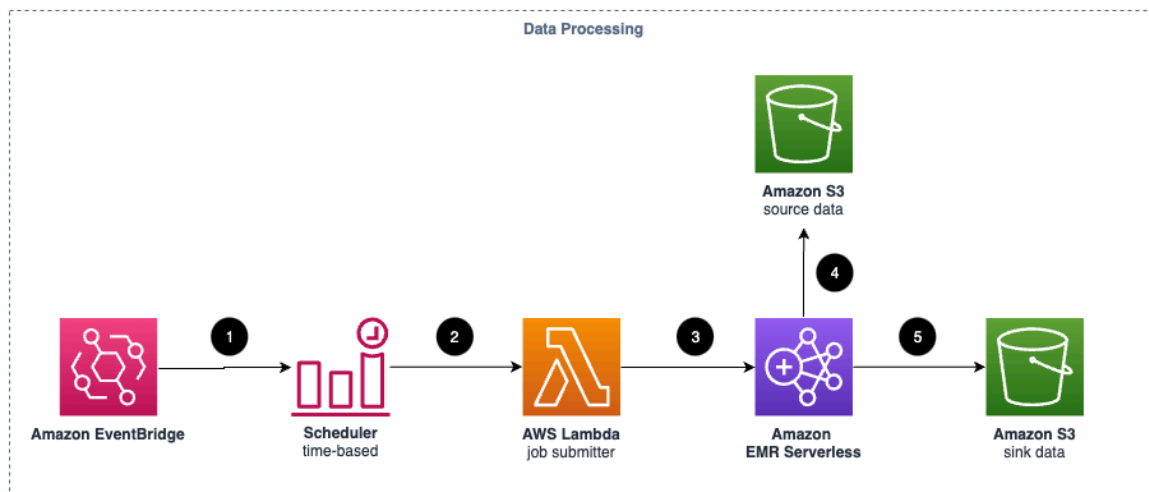
Suppose you create a data pipeline in the solution. This solution deploys the Amazon CloudFormation template in your AWS account and completes the following settings.

Note

The ingestion module supports three types of data sinks. You can only have one type of data sink in a data pipeline.

1. (Optional) The ingestion module creates an AWS global accelerator endpoint to reduce the latency of sending events from your clients (web applications or mobile applications).
2. [Elastic Load Balancing \(ELB\)](#) is used for load balancing ingestion web servers.
3. (Optional) If you enable the authenticating feature, the ALB will communicate with the OIDC provider to authenticate the requests.
4. ALB forwards all authenticated and valid requests to the ingestion servers.
5. Amazon ECS cluster is hosting the ingestion fleet servers. Each server consists of a proxy and a worker service. The proxy is a facade of the HTTP protocol, and the worker will send the events to a data sink based on your choice.
6. If Amazon Kinesis Data Streams is used as a buffer, AWS Lambda consumes the clickstream data in Kinesis Data Streams and then sinks them to Amazon S3 in batches.
7. If Amazon MSK is used as a buffer, MSK Connector is provisioned with an S3 connector plugin that sinks the clickstream data to Amazon S3 in batches.
8. If Amazon S3 is selected as data sink, the ingestion server will buffer a batch of events and sink them to Amazon S3.

Data processing module

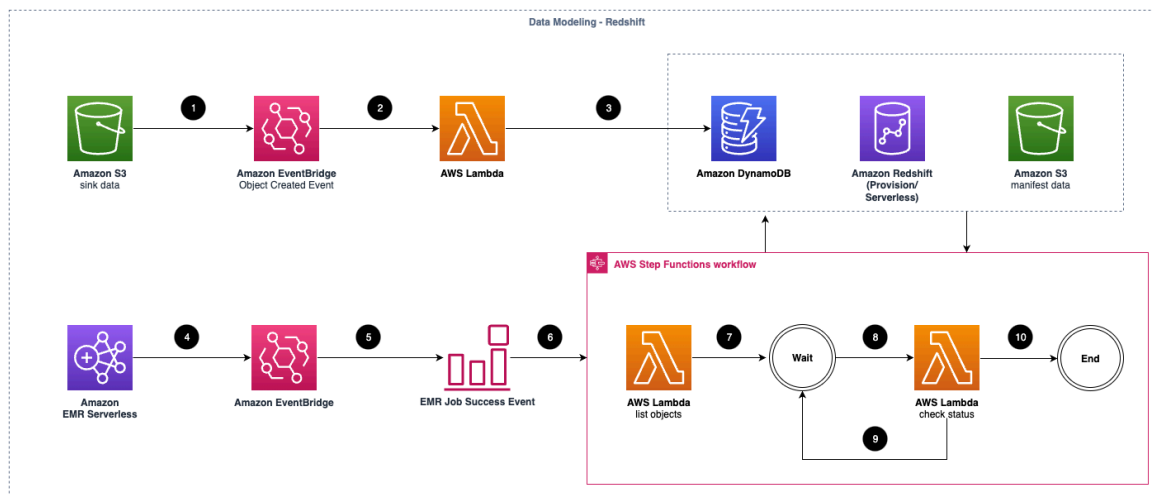


Data processing module architecture

Suppose you create a data pipeline in the solution and enable data processing. This solution deploys the Amazon CloudFormation template in your AWS Cloud account and completes the following settings.

1. Amazon EventBridge is used to trigger the data processing jobs periodically.
2. The configurable time-based scheduler invokes an AWS Lambda function.
3. The Lambda function kicks off an EMR Serverless application based on Spark to process a batch of clickstream events.
4. The EMR Serverless application uses the configurable transformer and enrichment plug-ins to process the clickstream data from the source S3 bucket.
5. After processing the clickstream events, the EMR Serverless application sinks the processed clickstream data to the sink S3 bucket.

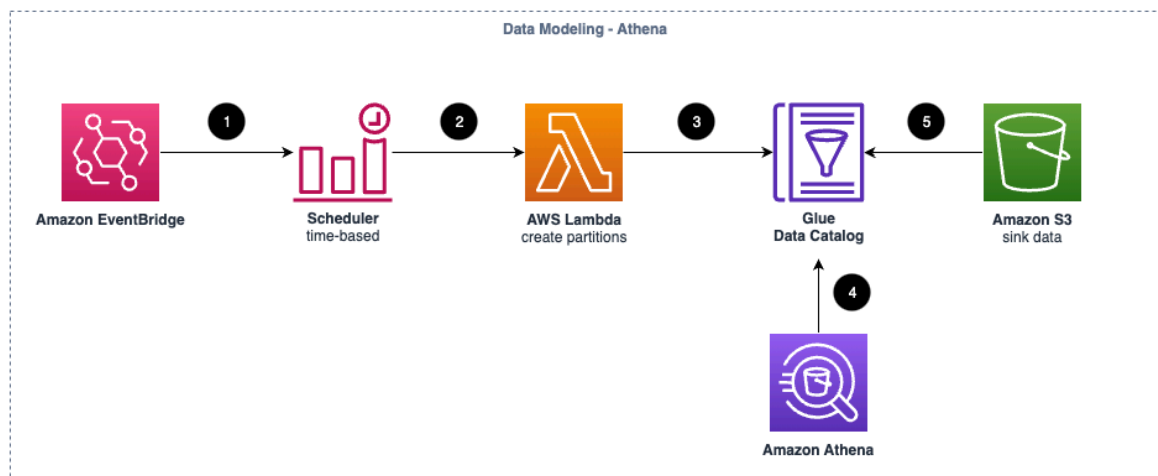
Data modeling module



Data modeling in Redshift architectureE

Suppose you create a data pipeline in the solution and enable data modeling in Amazon Redshift. This solution deploys the Amazon CloudFormation template in your AWS Cloud account and completes the following settings.

1. After the processed clickstream data is written in the Amazon S3 bucket, the Object Created Event is emitted.
2. An Amazon EventBridge rule is created for the event emitted in step 1, and an AWS Lambda function is invoked when the event happens.
3. The Lambda function persists the source event to be loaded in an Amazon DynamoDB table.
4. When data processing job is done, an event is emitted to Amazon EventBridge.
5. The pre-defined event rule of Amazon EventBridge processes the EMR job success event.
6. The rule invokes the AWS Step Functions workflow.
7. The workflow invokes the `list_objects` Lambda function that queries the DynamoDB table to find out the data to be loaded, then creates a manifest file for a batch of event data to optimize the load performance.
8. After a few seconds, the `check_status` Lambda function checks the status of the loading job.
9. If the load is still in progress, the `check_status` Lambda function waits for a few more seconds.
10. After all objects are loaded, the workflow ends.

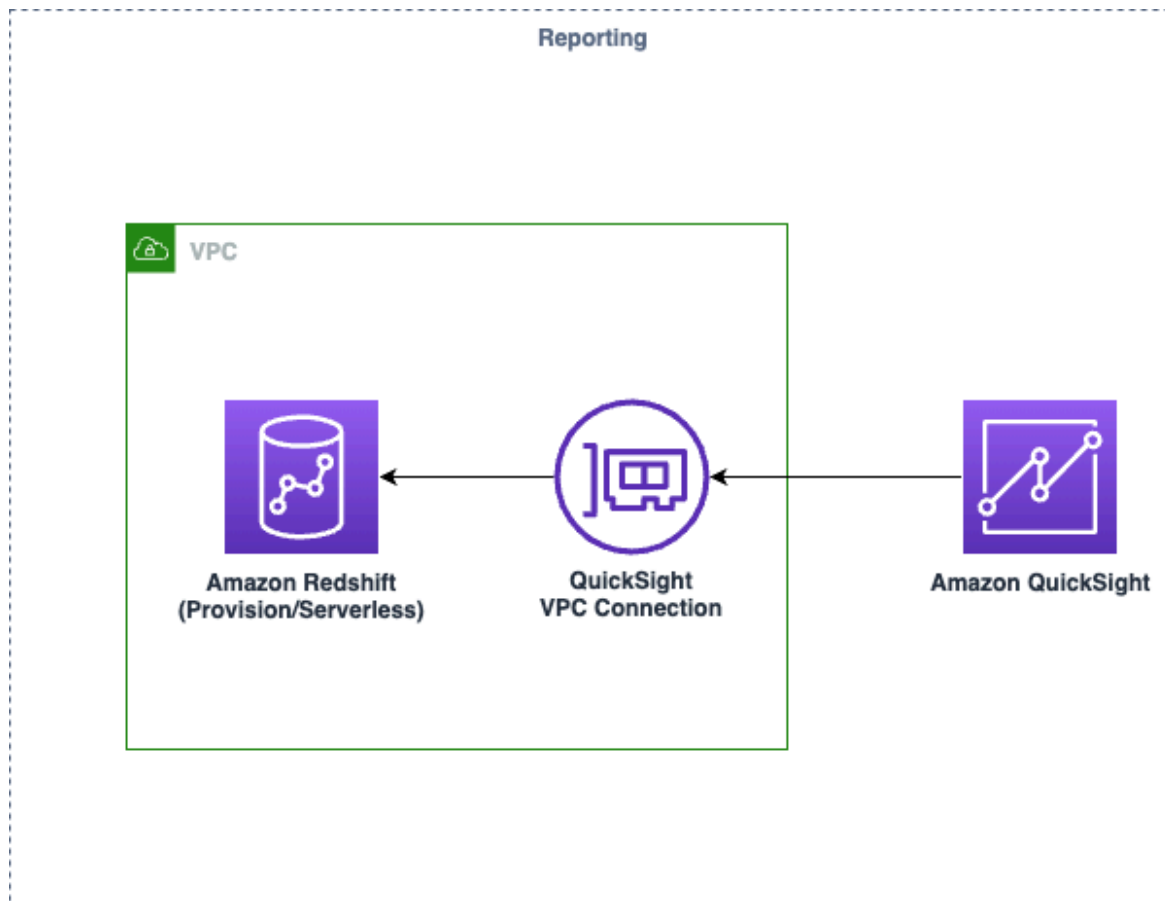


Data modeling in Athena architecture

Suppose you create a data pipeline in the solution and enable data modeling in Amazon Athena. This solution deploys the Amazon CloudFormation template in your AWS Cloud account and completes the following settings.

1. Amazon EventBridge invokes the data load into [Amazon Athena](#) periodically.
2. The configurable time-based scheduler invokes an AWS Lambda function.
3. The AWS Lambda function creates the partitions of the [AWS Glue](#) table for the processed clickstream data.
4. Amazon Athena is used for interactive querying of clickstream events.
5. The processed clickstream data is scanned via the Glue table.

Reporting module



Reporting module architecture

Suppose you create a data pipeline in the solution, enable data modeling in Amazon Redshift, and enable reporting in Amazon QuickSight. This solution deploys the Amazon CloudFormation template in your AWS Cloud account and completes the following settings.

1. VPC connection in Amazon QuickSight is used for securely connecting your Redshift within VPC.
2. The data source, data sets, template, analysis, and dashboard are created in Amazon QuickSight for out-of-the-box analysis and visualization.

AWS Well-Architected pillars

This solution was designed with best practices from the [AWS Well-Architected Framework](#) which helps customers design and operate reliable, secure, efficient, and cost-effective workloads in the cloud.

This section describes how the design principles and best practices of the Well-Architected Framework were applied when building this solution.

Operational excellence

This section describes how the principles and best practices of the [operational excellence pillar](#) were applied when designing this solution.

The Clickstream Analytics on AWS solution pushes metrics, logs and traces to Amazon CloudWatch at various stages to provide observability into the infrastructure, Elastic load balancer, Amazon ECS cluster, Lambda functions, EMR serverless application, Step Function workflow and the rest of the solution components. This solution also creates the CloudWatch dashboard for each [data pipeline \(p. 42\)](#).

Security

This section describes how the principles and best practices of the [security pillar](#) were applied when designing this solution.

- Clickstream Analytics on AWS web console users are authenticated and authorized with Amazon Cognito or OpenID Connect.
- All inter-service communications use AWS IAM roles.
- All roles used by the solution follows least-privilege access. That is, it only contains minimum permissions required so the service can function properly.

Reliability

This section describes how the principles and best practices of the [reliability pillar](#) were applied when designing this solution.

- Using AWS serverless services wherever possible (for example, EMR Serverless, Redshift Serverless, Lambda, Step Functions, Amazon S3, and Amazon SQS) to ensure high availability and recovery from service failure.
- Data ingested by [data pipeline \(p. 42\)](#) is stored in Amazon S3 and Amazon Redshift, so it persists in multiple Availability Zones (AZs) by default.

Performance efficiency

This section describes how the principles and best practices of the [performance efficiency pillar](#) were applied when designing this solution.

- The ability to launch this solution in any Region that supports AWS services in this solution such as: Amazon S3, Amazon ECS, and Elastic load balancer.
- Using Analytics Serverless architectures removes the need for you to run and maintain physical servers for traditional compute activities.
- Automatically testing and deploying this solution daily. Reviewing this solution by solution architects and subject matter experts for areas to experiment and improve.

Cost optimization

This section describes how the principles and best practices of the [cost optimization pillar](#) were applied when designing this solution.

- The solution uses Autoscaling Group so that the compute costs are only related to how much data is ingested and processed.
- The solution uses serverless services such as Amazon S3, Amazon Kinesis Data Streams, Amazon EMR Serverless and Amazon Redshift Serverless so that customers only get charged for what they use.

Sustainability

This section describes how the principles and best practices of the [sustainability pillar](#) were applied when designing this solution.

- The solution's serverless design (using Amazon Kinesis Data Streams, Amazon EMR Serverless, Amazon Redshift Serverless and Amazon QuickSight) and the use of managed services (such as Amazon ECS, Amazon MSK) are aimed at reducing carbon footprint compared to the footprint of continually operating on-premises servers.

Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

Solution components

The Clickstream Analytics on AWS solution has three components: a web console, SDKs, and the data pipeline for ingesting, processing, analyzing and visualizing the Clickstream data.

Web console

This solution provides a simple web console which allows you to create and manage projects with their data pipeline to ingest, process, analyze and visualize the Clickstream data.

SDKs

This solution provides native SDKs to help you easily collect and report in-app events from your applications to your Clickstream pipelines.

- [Android SDK \(p. 59\)](#)
- [Swift SDK \(p. 59\)](#)

Data pipeline

This solution uses the web console to manage the project and its data pipeline. The data pipeline consists of four modules.

Ingestion module

The ingestion module serves as web server for ingesting the Clickstream data. It supports the following features:

- Specify the auto scaling group capability
- Specify warm pool size to scale out faster and save costs
- Support authenticate with OIDC
- Support SSL
- Support enabling AWS Global Accelerator for ELB
- Support different sink buffer, S3, KDS and MSK

Data processing module

The data processing module transforms and enriches the ingested data to solution's data model by the Apache Spark application running in EMR serverless. It supports the following features:

- Specify the batch interval of data processing

- Specify the data refreshness age
- Provider out-of-the-box enrichment plug-ins
 - UA enrichment to parse OS, device, browser information from User Agent string of the HTTP request header
 - IP enrichment to mapping device location information (for example, city, country, region) based on the request source IP
- Support third-party transformer plug-ins
- Support third-party enrichment plug-ins

Data modeling module

The data modeling module loads the processed data into data lakehouse. It supports the following features:

- Support both provisioned Redshift and Redshift Serverless as data warehouse
 - Users can specify the time range for storing data in Redshift
 - Users can specify the interval to update user dimension table
- Support use Athena to query the data in data lake

Reporting module

The reporting module queries the data in Redshift and creates out-of-the-box visualization reports in QuickSight.

AWS services in this solution

The following AWS services are included in this solution:

| AWS service | Description |
|---|---|
| Amazon Elastic Load Balancing | Core. To distribute network traffic to ingestion fleet. |
| Amazon ECS | Core. To run the ingestion module fleet. |
| Amazon EC2 | Core. To provide the underlying computing resources for ingestion fleet. |
| Amazon ECR | Core. To host the container images used by ingestion fleet. |
| Amazon S3 | Core. To store the ingested and processed Clickstream data. And it also stores the service logs and static web assets (frontend user interface). |
| AWS Global Accelerator | Supporting. To improve the availability, performance, and security of the ingestion service in AWS Regions. |
| AWS CloudWatch | Supporting. To monitor the metrics, logs and trace of data pipeline. |

| AWS service | Description |
|---|--|
| Amazon SNS | Supporting. To provide topic and email subscription notifications for the alarms of data pipeline. |
| Amazon Kinesis Data Streams | Supporting. To provide the ingestion buffer. |
| AWS Lambda | Supporting. To integrate with kinds of AWS services. For example, sink ingestion data to S3, manage the lifecycle of AWS resources. |
| Amazon Managed Streaming for Apache Kafka (MSK) | Supporting. To provide the ingestion buffer with Apache Kafka. |
| Amazon EMR Serverless | Supporting. To process the ingested data. |
| Amazon Glue | Supporting. To manage the data catalog of ingested data. |
| Amazon EventBridge | Supporting. To integrate with AWS services with events or schedule. |
| Amazon Redshift | Supporting. To analyze your Clickstream data in data warehouse. |
| Amazon Athena | Supporting. To analyze your Clickstream data in data lake. |
| AWS Step Functions | Supporting. To orchestrate the lifecycle management of project's pipeline. Also it manages the workflow to load data into data warehouse. |
| AWS Secrets Manager | Supporting. To store the credential for OIDC credentials and BI user in Redshift. |
| Amazon QuickSight | Supporting. Visual your analysis reporting of your Clickstream data. |
| Amazon CloudFront | Supporting. To made available the static web assets (frontend user interface) and proxy the backend in the same origin. |
| Amazon Cognito | Supporting. To authenticate users (in AWS Regions). |
| Amazon API Gateway | Supporting. To provide the backend APIs. |
| Amazon DynamoDB | Supporting. To store projects data. |
| AWS CloudFormation | Supporting. To provision the AWS resources for the modules of data pipeline. |

Plan your deployment

This section describes the cost, security, and Region considerations for planning your deployment.

Cost

Important

The following cost estimations are examples and may vary depending on your environment.

You are responsible for the cost of AWS services used when running this solution. Deploying this solution will only create a solution web console in your AWS account, which is completely serverless and typically can be covered within free tier.

The cost for this solution is mostly incurred by the data pipeline. As of this revision, the main factors affecting the solution cost include:

- **Ingestion module**, the cost depends on the size of the ingestion server and the type of the data sink you choose.
- **Data processing and modeling module** (optional), the cost depends on whether you choose to enable this module and its relevant configurations
- **Enabled Dashboards** (optional), the cost depends on whether you choose to enable this module and its relevant configurations
- **Volume of clickstream data**
- **Additional features**

The following are cost estimations for data volumes of 10/100/1000/10000 RPS (request per second) with different data pipeline configurations. Cost estimation is provided by modules. To get a total cost for your use case, sum the cost by modules based on your actual configuration.

Important

As of this revision, the following cost is calculated with On-Demand prices in the us-east-1 Region and measured in USD.

Ingestion module

Ingestion module includes the following cost components:

- Application load balancer
- EC2 for ECS
- Data sink (Kinesis | Kafka | Direct to S3)
- S3 storage

Key assumptions include:

- Request payload: 1KB (compressed, 1:10 ratio)
- MSK configurations (m5.large * 2)

- KDS configuration (on-demand, provision - shard 2)
- 10/100/1000 request per second (RPS)

| RPS | ALB cost | EC2 cost | Buffer type | Buffer cost | S3 cost | Total (USD per month) |
|----------|----------|----------|---|-------------|---------|-----------------------|
| 10 RPS | \$7 | \$122 | Kinesis (On-Demand) | \$36 | \$3 | \$168 |
| | \$7 | \$122 | Kinesis (Provisioned 2 shard) | \$22 | \$3 | \$154 |
| | \$7 | \$122 | MSK (m5.large * 2, connector MCU * 1) | \$417 | \$3 | \$549 |
| | \$7 | \$122 | None | | \$3 | \$132 |
| 100 PRS | \$43 | \$122 | Kinesis(On-demand) | \$86 | \$3 | \$254 |
| | \$43 | \$122 | Kinesis (Provisioned 2 shard) | \$26 | \$3 | \$194 |
| | \$43 | \$122 | MSK (m5.large * 2, connector MCU * 1) | \$417 | \$3 | \$585 |
| | \$43 | \$122 | None | | \$3 | \$168 |
| 1000 RPS | \$396 | \$122 | Kinesis(On-demand) | \$576 | \$14 | \$1108 |
| | \$396 | \$122 | Kinesis (Provisioned 10 shard) | \$146 | \$14 | \$678 |
| | \$396 | \$122 | MSK (m5.large * 2, connector MCU * 2~3) | \$530 | \$14 | \$1062 |
| | \$396 | \$122 | None | | \$14 | \$532 |

Data processing & modeling modules

Data processing & modeling module include the following cost components if you enable:

- EMR Serverless
- Redshift

Key assumptions include:

- 10/100/1000/10000 RPS
- Data processing interval: hourly/6-hourly/daily
- EMR running three built-in plugins to process data

| RPS | EMR schedule interval | EMR cost | Redshift type | Redshift cost | Total (USD) |
|----------|-----------------------|----------|--------------------------|---------------|-------------|
| 10 RPS | Hourly | \$28 | Serverless (8 based RPU) | \$68 | \$96 |
| | 6-hourly | \$10.80 | Serverless (8 based RPU) | \$11 | \$21.80 |
| | Daily | \$9.60 | Serverless (8 based RPU) | \$3 | \$12.60 |
| 100 RPS | Hourly | \$105 | Serverless (8 based RPU) | \$72 | \$177 |
| | 6-hourly | \$99 | Serverless (8 based RPU) | \$17.20 | \$116.20 |
| | Daily | \$140 | Serverless (8 based RPU) | \$16.90 | \$156.90 |
| 1000 RPS | Hourly | \$1362 | Serverless (8 based RPU) | \$172 | \$1534 |
| | 6-hourly | \$678 | Serverless (8 based RPU) | \$176 | \$854 |
| | Daily | \$873 | Serverless (8 based RPU) | \$352 | \$1225 |

Note

For the cost of 1000 PRS Daily, we used the following EMR configuration:

```
{
  "sparkConfig": [
    "spark.emr-serverless.executor.disk=80g",
    "spark.executor.instances=8",
    "spark.dynamicAllocation.initialExecutors=16",
    "spark.executor.memory=80g",
    "spark.executor.cores=16"
  ],
  "inputRePartitions": 2000
}
```

Reporting module

Reporting module includes the following cost components if you choose to enable:

- QuickSight

Key assumptions include

- QuickSight Enterprise subscription
- Exclude Q cost

| Daily data volume/RPS | Cost for authors | Cost for readers | Cost for SPICE | Total (USD) |
|-----------------------|---|--|------------------------|-------------|
| All size | \$48 (Two authors with monthly subscription) | \$18.80 (Ten readers with 22 working days per month, 5% active readers, 50% frequent readers, 25% occasional readers, 20% inactive readers) | 0 10GB capacity | \$66.80 |

Note

The QuickSight cost applies to all your data pipelines, including the visualization managed outside the solution.

Logging and monitoring

The solution utilizes CloudWatch Logs, CloudWatch Metrics and CloudWatch Dashboard to implement logging, monitoring and visualizing features. The total cost is around \$14 per month and may vary based on the volume of logs and the number of metrics being monitored.

Additional features

You will be charged with additional cost only if you choose to enable the following features.

Secrets Manager

- If you enable reporting, the solution creates a secret in Secrets Manager to store the Redshift credentials used by QuickSight visualization. **Cost:** 0.4 USD/month.
- If you enable the authentication feature of the ingestion module, you need to create a secret in Secrets Manager to store the information for OIDC. **Cost:** 0.4 USD/month.

AWS Global Accelerator

It incurs a fixed hourly charge and a per-day volume data transfer cost.

Key assumptions: Ingestion deployment in us-east-1

| RPS | Fixed hourly cost | Data transfer cost | Total (USD) |
|---------|-------------------|--------------------|-------------|
| 10 RPS | \$18 | \$0.30 | \$18.30 |
| 100 RPS | \$18 | \$3 | \$21 |

| RPS | Fixed hourly cost | Data transfer cost | Total (USD) |
|----------|-------------------|--------------------|-------------|
| 1000 RPS | \$18 | \$30 | \$38 |

Application Load Balancer access logs

The charged cost includes the storage cost for Amazon S3, but not for the bandwidth used by Elastic Load Balancing to send log files to Amazon S3. For more information, see <https://aws.amazon.com/s3/pricing/> Amazon S3 pricing.

| RPS | Log size (GB) | S3 cost (USD) |
|----------|---------------|---------------|
| 10 RPS | \$16.50 | \$0.38 |
| 100 RPS | \$165 | \$3.80 |
| 1000 RPS | \$1650 | \$38 |

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This [shared responsibility model](#) reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, see [AWS Cloud Security](#).

IAM Roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions, Amazon API Gateway and Amazon Cognito or OpenID connect access to create regional resources.

Amazon VPC

This solution optionally deploys a web console within your VPC. You can isolate access to the web console via Bastion hosts, VPNs, or Direct Connect. You can create [VPC endpoints](#) to let traffic between your Amazon VPC and AWS services not leave the Amazon network to satisfy the compliance requirements.

Security groups

The security groups created in this solution are designed to control and isolate network traffic between the solution components. We recommend that you review the security groups and further restrict access as needed once the deployment is up and running.

Amazon CloudFront

This solution optionally deploys a web console hosted in an Amazon S3 bucket and Amazon API Gateway. To help reduce latency and improve security, this solution includes an Amazon CloudFront

distribution with an Origin Access Control (OAC), which is a CloudFront user that provides public access to the solution's website bucket contents. For more information, refer to [Restricting access to an Amazon S3 origin](#) in the Amazon CloudFront Developer Guide.

Supported AWS Regions

This solution uses services which may not be currently available in all AWS Regions. Launch this solution in an AWS Region where required services are available. For the most current availability by Region, refer to the [AWS Regional Services List](#).

Clickstream Analytics on AWS provides two types of authentication for its web console, [Amazon Cognito User Pool](#) and [OpenID Connect \(OIDC\) Provider](#). You must choose to launch the solution with OpenID Connect in case one of the following scenarios:

- Amazon Cognito User Pool is not available in your AWS Region.
- You already have an OpenID Connect Provider and want to authenticate against it.

Supported Regions for web console deployment

| Region Name | Launch with Amazon Cognito user pool | Launch with OpenID Connect |
|--------------------------|--------------------------------------|----------------------------|
| US East (N. Virginia) | Yes | Yes |
| US East (Ohio) | Yes | Yes |
| US West (N. California) | Yes | Yes |
| US West (Oregon) | Yes | Yes |
| Africa (Cape Town) | No | Yes |
| Asia Pacific (Hong Kong) | No | Yes |
| Asia Pacific (Jakarta) | No | Yes |
| Asia Pacific (Mumbai) | Yes | Yes |
| Asia Pacific (Osaka) | No | Yes |
| Asia Pacific (Seoul) | Yes | Yes |
| Asia Pacific (Singapore) | Yes | Yes |
| Asia Pacific (Sydney) | Yes | Yes |
| Asia Pacific (Tokyo) | Yes | Yes |
| Canada (Central) | Yes | Yes |
| Europe (Frankfurt) | Yes | Yes |
| Europe (Ireland) | Yes | Yes |
| Europe (London) | Yes | Yes |
| Europe (Milan) | No | Yes |

| Region Name | Launch with Amazon Cognito user pool | Launch with OpenID Connect |
|---|--------------------------------------|----------------------------|
| Europe (Paris) | Yes | Yes |
| Europe (Stockholm) | Yes | Yes |
| Middle East (Bahrain) | No | Yes |
| South America (Sao Paulo) | Yes | Yes |
| China (Beijing) Region Operated by Sinnet | No | Yes |
| China (Ningxia) Region Operated by NWCD | No | Yes |

This solution provides [modular components \(p. 5\)](#) for supporting different data pipeline architecture. The data processing, and reporting modules are optional, that is, you can create a data pipeline without data processing and reporting modules if needed.

Pipeline modules availability

| Region Name | Data ingestion with MSK as buffer | Data ingestion with KDS as buffer | Data ingestion with S3 as buffer | Data processing | Data Modeling with Redshift Serverless | Data Modeling with Provisioned Redshift | Reporting with QuickSight |
|--------------------------|-----------------------------------|-----------------------------------|----------------------------------|-----------------|--|---|---------------------------|
| US East (N. Virginia) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| US East (Ohio) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| US West (N. California) | Yes | Yes | Yes | Yes | Yes | Yes | No |
| US West (Oregon) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Africa (Cape Town) | Yes | Yes | Yes | No | No | No | No |
| Asia Pacific (Hong Kong) | Yes | Yes | Yes | Yes | No | Yes | No |
| Asia Pacific (Mumbai) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Asia Pacific (Osaka) | Yes | Yes | Yes | No | No | No | No |

Clickstream Analytics on AWS Implementation Guide
Supported AWS Regions

| Region Name | Data ingestion with MSK as buffer | Data ingestion with KDS as buffer | Data ingestion with S3 as buffer | Data processing | Data Modeling with Redshift Serverless | Data Modeling with Provisioned Redshift | Reporting with QuickSight |
|--|-----------------------------------|-----------------------------------|----------------------------------|-----------------|--|---|---------------------------|
| Asia Pacific (Seoul) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Asia Pacific (Singapore) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Asia Pacific (Sydney) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Asia Pacific (Tokyo) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Canada (Central) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Europe (Frankfurt) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Europe (Ireland) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Europe (London) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Europe (Milan) | Yes | Yes | Yes | No | No | No | No |
| Europe (Paris) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Europe (Stockholm) | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Middle East (Bahrain) | Yes | Yes | Yes | Yes | No | Yes | No |
| South America (Sao Paulo) | Yes | Yes | Yes | Yes | No | Yes | Yes |
| China (Beijing) Region Operated by Sinnet* | Yes | Yes | Yes | Yes | No | Yes | No |

| Region Name | Data ingestion with MSK as buffer | Data ingestion with KDS as buffer | Data ingestion with S3 as buffer | Data processing | Data Modeling with Redshift Serverless | Data Modeling with Provisioned Redshift | Reporting with QuickSight |
|--|-----------------------------------|-----------------------------------|----------------------------------|-----------------|--|---|---------------------------|
| China (Ningxia) Region Operated by NWCD* | Yes | Yes | Yes | Yes | No | Yes | No |

Note

AWS China Regions don't support using AWS Global Accelerator to accelerate the ingestion endpoint.

Deployment

Before you launch the solution, review the architecture, supported Regions, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Prerequisites

Review all the [considerations \(p. 15\)](#) and make sure you have the following in the target Region you want to deploy the solution:

- At least two vacant S3 buckets.

Deployment in AWS Regions

Clickstream Analytics on AWS provides two ways to authenticate and log into the solution web console. For some AWS Regions where Cognito User Pool is unavailable (for example, Hong Kong), you must launch the solution with OpenID Connect.

- [Launch with Cognito User Pool \(p. 24\)](#) (Recommended)
- [Launch with OpenID Connect \(p. 27\)](#)

For more information about supported Regions, see [Regional deployments \(p. 20\)](#).

Deployment in AWS China Regions

AWS China Regions do not have Cognito User Pool. You must launch the solution with OpenID Connect.

- [Launch with OpenID Connect \(p. 27\)](#)

Deployment within Amazon VPC

Clickstream Analytics on AWS supports being deployed into an Amazon VPC, allowing access to the web console without leaving your VPC network.

- [Launch within VPC \(p. 34\)](#)

Launch with Cognito User Pool

Time to deploy: Approximately 15 minutes

Deployment overview

Use the following steps to deploy this solution on AWS.



[Step 1. Launch the stack \(p. 25\)](#)

[Step 2. Launch the web console \(p. 26\)](#)

Step 1: Launch the Stack

This AWS CloudFormation template automatically deploys the Clickstream Analytics on AWS solution on AWS.

1. Sign in to the [AWS Management Console](#) and select the button to launch the AWS CloudFormation template.

| | Launch in AWS Management Console |
|---------------------------------|--|
| Launch stack |  |
| Launch stack with custom domain |  |

2. The template is launched in the default Region after you log in to the console. To launch the Clickstream Analytics on AWS solution in a different AWS Region, use the Region selector in the console navigation bar.
3. On the **Create stack** page, verify that the correct template URL is shown in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, refer to IAM and AWS STS quotas in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for the template and modify them as necessary.
 - This solution uses the following parameters:

| Parameter | Default | Description |
|------------------|-------------------------------|--|
| Admin User Email | <i><Requires input></i> | Specify the email of the Administrator. This email address will receive a temporary password to access the Clickstream Analytics on AWS web console. You can create more users directly in the provisioned Cognito User Pool after launching the solution. |

Important

By default, this deployment uses TLSv1.0 and TLSv1.1 in CloudFront. However, we recommend that you manually configure CloudFront to use the securer TLSv1.2/TLSv1.3 and apply for a certificate and custom domain to enable this. We highly recommend that you update your TLS configuration and cipher suite selection according to the following recommendations:

- **Transport Layer Security Protocol:** Upgrade to TLSv1.2 or higher
- **Key Exchange:** ECDHE

- **Block Cipher Mode of Operation:** GCM
 - **Authentication:** ECDSA
 - **Encryption Cipher:** AES256
 - **Message Authentication:** SHA(256/384/any hash function except for SHA1)
For example, TLSv1.2_2021 can be used to meet these recommendations.
- If you are launching the solution with custom domain in AWS regions, this solution uses the additional following parameters:

| Parameter | Default | Description |
|------------------|------------------|--|
| Hosted Zone ID | <Requires input> | Choose the public hosted zone ID of Amazon Route 53. |
| Hosted Zone Name | <Requires input> | The domain name of the public hosted zone, for example, example.com. |
| Record Name | <Requires input> | The sub name (as known as record name in R53) of the domain name of console. For example, enter clickstream, if you want to use custom domain clickstream.example.com for the console. |

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Select the box acknowledging that the template creates AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a **CREATE_COMPLETE** status in approximately 15 minutes.

Step 2: Launch the web console

After the stack is successfully created, this solution generates a CloudFront domain name that gives you access to the Clickstream Analytics on AWS web console. Meanwhile, an auto-generated temporary password will be sent to your email address.

1. Sign in to the [AWS CloudFormation console](#).
2. On the **Stacks** page, select the solution's stack.
3. Choose the **Outputs** tab and record the domain name.
4. Open the **ControlPlaneURL** using a web browser, and navigate to a sign-in page.
5. Enter the **Email** and the temporary password.
 - a. Set a new account password.
 - b. (Optional) Verify your email address for account recovery.

After the verification is complete, the system opens the Clickstream Analytics on AWS web console.

Once you have logged into the Clickstream Analytics on AWS console, you can start to [create a project \(p. 37\)](#) for your applications.

Launch with OpenID Connect (OIDC)

Time to deploy: Approximately 30 minutes

Prerequisites

Important

The Clickstream Analytics on AWS console is served via CloudFront distribution which is considered as an Internet information service. If you are deploying the solution in **AWS China Regions**, the domain must have a valid [ICP Recordal](#).

- **A domain.** You will use this domain to access the Clickstream Analytics on AWS console. This is required for AWS China Regions, and is optional for AWS Regions.
- **An SSL certificate in AWS IAM.** The SSL must be associated with the given domain. Follow [this guide \(p. 116\)](#) to upload SSL certificate to IAM. This is required for AWS China Regions, but is not recommended for AWS Regions.
- Make sure to request or import the ACM certificate in the US East (N. Virginia) Region (us-east-1). This is not required for AWS China Regions, and is optional for AWS Regions.

Deployment overview

Use the following steps to deploy this solution on AWS.

[Step 1. Create OIDC client \(p. 27\)](#)

[Step 2. Launch the stack \(p. 31\)](#)

[Step 3. Update the callback URL of OIDC client \(p. 33\)](#)

[Step 4. Set up DNS Resolver \(p. 34\)](#)

[Step 5. Launch the web console \(p. 34\)](#)

Step 1. Create OIDC client

You can use different kinds of OpenID Connect (OIDC) providers. This section introduces Option 1 to Option 4.

- (Option 1) Using Amazon Cognito from another Region as OIDC provider.
- (Option 2) [Authing](#), which is an example of a third-party authentication provider.
- (Option 3) [Keycloak](#), which is a solution maintained by AWS and can serve as an authentication identity provider.
- (Option 4) [ADFS](#), which is a service offered by Microsoft.
- (Option 5) Other third-party authentication platforms such as [Auth0](#).

Follow the steps below to create an OIDC client, and obtain the `client_id` and `issuer`.

(Option 1) Using Amazon Cognito User Pool from another Region

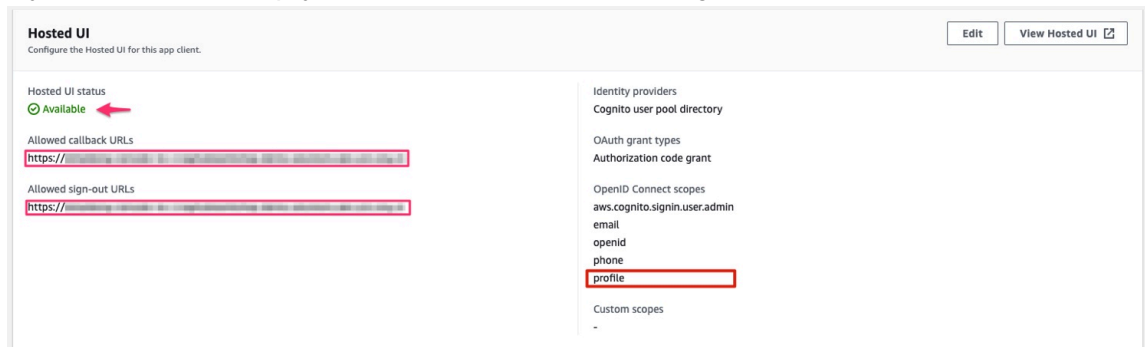
You can leverage the [Amazon Cognito User Pool](#) in a supported AWS Region as the OIDC provider.

1. Go to the [Amazon Cognito console](#) in an AWS Region.
2. Set up the hosted UI with the Amazon Cognito console based on this [guide](#).
 - Choose **Public client** when selecting the **App type**. Make sure NOT to change the selection **Don't generate a client secret** for **Client secret**.
 - Add **Profile** in **OpenID Connect scopes**.
3. Enter the **Callback URL** and **Sign out URL** using your domain name for Clickstream Analytics on AWS console.

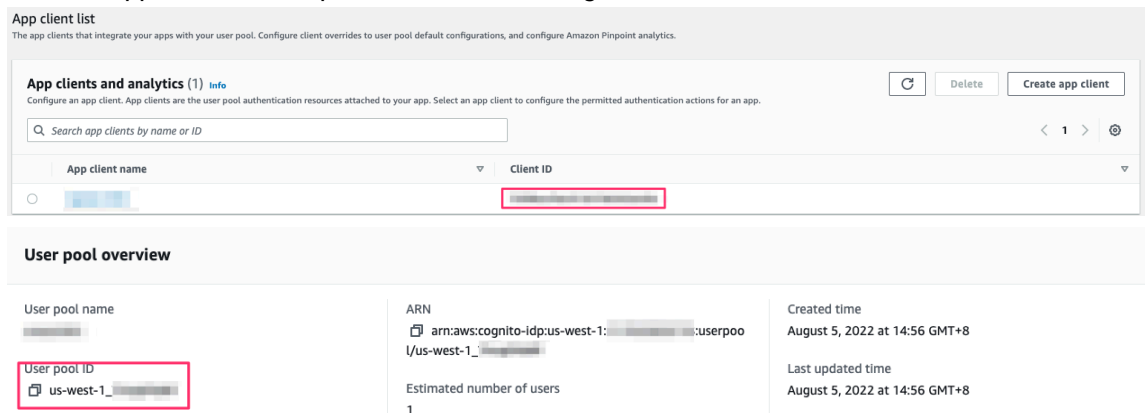
Note

If you don't know the custom domain name of console, you can firstly enter one, for example, `clickstream.example.com`, and then update it following guidelines in [Step 3 Update the callback URL of OIDC client \(p. 33\)](#).

4. If your hosted UI is set up, you should be able to see something like below:



5. Save the App client ID, User pool ID and the AWS Region to a file, which will be used later.



In [Step 2. Launch the stack \(p. 31\)](#), enter the parameters below from your Cognito User Pool.

- **OIDCClientId**: App client ID
- **OIDCProvider**: `https://cognito-idp.${REGION}.amazonaws.com/${USER_POOL_ID}`

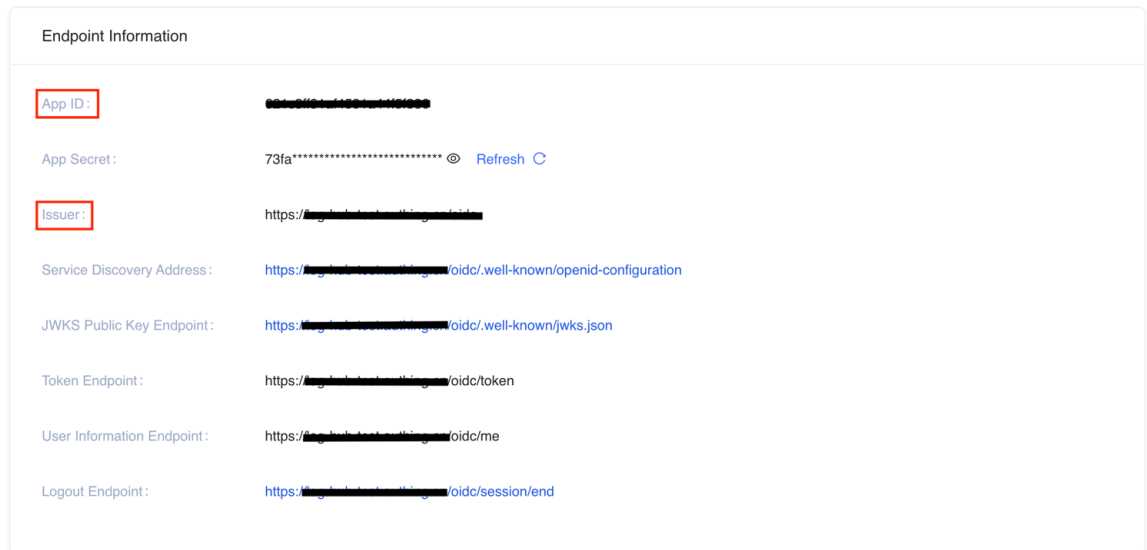
(Option 2) Authing.cn OIDC client

1. Go to the [Authing console](#).
2. Create a user pool if you don't have one.
3. Select the user pool.
4. On the left navigation bar, select **Self-built App** under **Applications**.

5. Choose **Create**.

6. Enter the **Application Name**, and **Subdomain**.

7. Save the App ID (that is, `client_id`) and Issuer to a text file from Endpoint Information, which will be used later.



Endpoint Information

App ID: [red box]

App Secret: 73fa***** Refresh

Issuer: [red box]

Service Discovery Address: https://.../oidc/.well-known/openid-configuration

JWKS Public Key Endpoint: https://.../oidc/.well-known/jwks.json

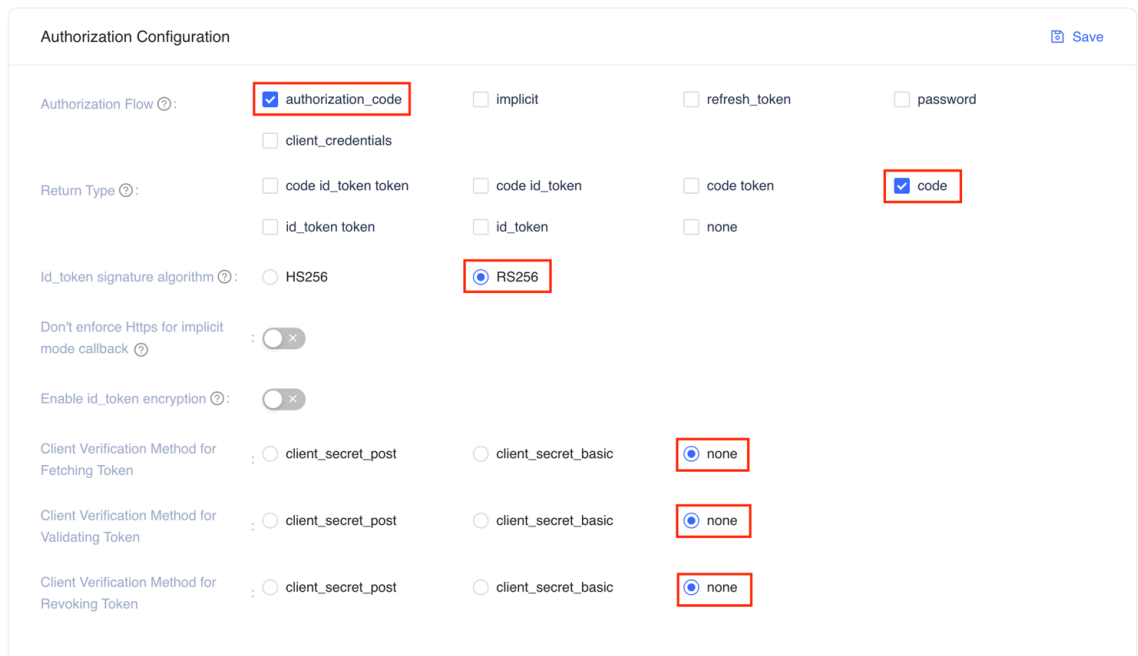
Token Endpoint: https://.../oidc/token

User Information Endpoint: https://.../oidc/me

Logout Endpoint: https://.../oidc/session/end

8. Update the Login Callback URL and Logout Callback URL to your IPC recorded domain name.

9. Set the Authorization Configuration.



Authorization Configuration Save

Authorization Flow: ☒ authorization_code ☐ implicit ☐ refresh_token ☐ password

Return Type: ☐ code_id_token token ☐ code_id_token ☐ code token ☒ code

Id_token signature algorithm: ☐ HS256 ☒ RS256

Don't enforce https for implicit mode callback: ☐

Enable id_token encryption: ☐

Client Verification Method for Fetching Token: ☐ client_secret_post ☐ client_secret_basic ☒ none

Client Verification Method for Validating Token: ☐ client_secret_post ☐ client_secret_basic ☒ none

Client Verification Method for Revoking Token: ☐ client_secret_post ☐ client_secret_basic ☒ none

You have successfully created an Authing self-built application.

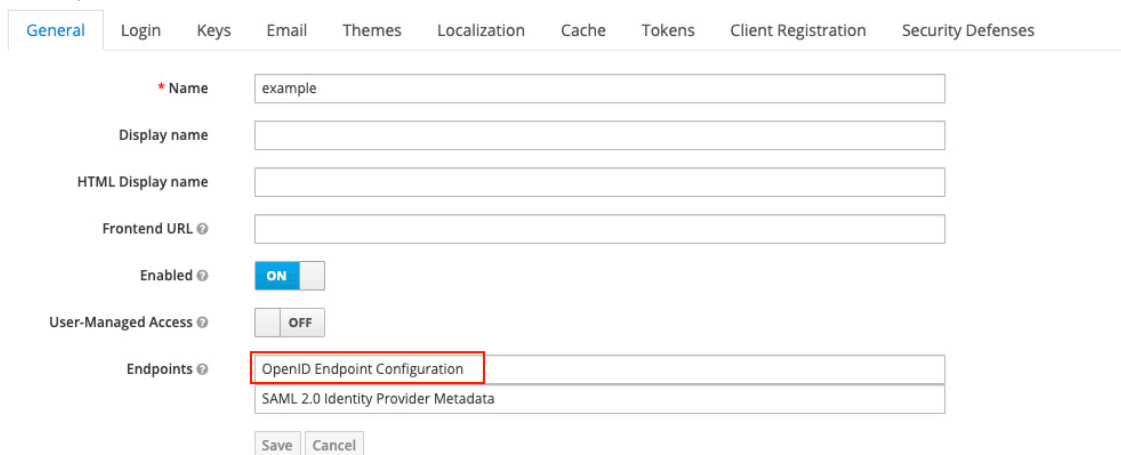
In [Step 2. Launch the stack \(p. 31\)](#), enter the parameters below from your Cognito User Pool.

- **OIDCClientId:** client ID
- **OIDCProvider:** Issuer

(Option 3) Keycloak OIDC client

1. Deploy the Keycloak solution in AWS China Regions following [this guide](#).
2. Sign in to the Keycloak console.
3. On the left navigation bar, select **Add realm**. Skip this step if you already have a realm.
4. Go to the realm setting page. Choose **Endpoints**, and then **OpenID Endpoint Configuration** from the list.

Example 



The screenshot shows the Keycloak console interface. The 'General' tab is active. The 'Name' field is set to 'example'. The 'Endpoints' section is expanded, and 'OpenID Endpoint Configuration' is selected. Other tabs include Login, Keys, Email, Themes, Localization, Cache, Tokens, Client Registration, and Security Defenses. The 'Enabled' toggle is turned ON, and 'User-Managed Access' is turned OFF. The 'Save' and 'Cancel' buttons are at the bottom.

5. In the JSON file that opens up in your browser, record the **issuer** value which will be used later.

```
{"issuer": "https://1.elb.amazonaws.com.cn/auth/realms/example/protocol/openid-connect/auth", "token_endpoint": "https://keyc1-keyc1-159azf1.elb.amazonaws.com.cn/auth/realms/example/protocol/openid-connect/token", "introspection_endpoint": "https://keyc1-keyc1-1.elb.amazonaws.com.cn/auth/realms/example/protocol/openid-connect/token/introspect", "userinfo_endpoint": "https://keyc1-keyc1-1.elb.amazonaws.com.cn/auth/realms/example/protocol/openid-connect/userinfo"}
```

6. Go back to Keycloak console and select **Clients** on the left navigation bar, and choose **Create**.
7. Enter a Client ID, which must contain 24 letters (case-insensitive) or numbers. Record the **Client ID** which will be used later.
8. Change client settings. Enter `http[s]://<Clickstream Analytics on AWS Console domain>/signin` in **Valid Redirect URIs**, and enter `<console domain>` and `+` in **Web Origins**.

Note

If you're not using custom domain for the console, the domain name of console is not available yet. You can enter a fake one, for example, `clickstream.example.com`, and then update it following guidelines in Step 3.

9. In the Advanced Settings, set the **Access Token Lifespan** to at least 5 minutes.
10. Select **Users** on the left navigation bar.
11. Click **Add user** and enter **Username**.
12. After the user is created, select **Credentials**, and enter **Password**.

In [Step 2. Launch the stack \(p. 31\)](#), enter the parameters below from your Keycloak realm.

- **OIDCClientId**: `client id`
- **OIDCProvider**: `https://<KEYCLOAK_DOMAIN_NAME>/auth/realms/<REALM_NAME>`

(Option 4) ADFS OpenID Connect Client

1. Make sure your ADFS is installed. For information about how to install ADFS, refer to [this guide](#).
2. Log in to the ADFS Sign On page. The URL should be >`https://adfs.domain.com/adfs/ls/idpinitiatedSignOn.aspx`, and you need to replace **adfs.domain.com** with your real ADFS domain.
3. Log on your **Domain Controller**, and open **Active Directory Users and Computers**.
4. Create a **Security Group** for Clickstream Analytics on AWS users, and add your planned users to this Security Group.
5. Log on to ADFS server, and open **ADFS Management**.
6. Right click **Application Groups**, choose **Application Group**, and enter the name for the Application Group. Select **Web browser accessing a web application** option under **Client-Server Applications**, and choose **Next**.
7. Record the **Client Identifier** (`client_id`) under **Redirect URI**, enter your Clickstream Analytics on AWS domain (for example, `xx.domain.com`), and choose **Add**, and then choose **Next**.
8. In the **Choose Access Control Policy** window, select **Permit specific group**, select **parameters** under Policy part, add the created Security Group in Step 4, then choose **Next**. You can configure other access control policy based on your requirements.
9. Under Summary window, choose **Next**, and choose **Close**.
10. Open the Windows PowerShell on ADFS Server, and run the following commands to configure ADFS to allow CORS for your planned URL.

```
Set-AdfsResponseHeaders -EnableCORS $true  
Set-AdfsResponseHeaders -CORSTrustedOrigins https://<your-domain>
```

11. Under Windows PowerShell on ADFS server, run the following command to get the Issuer (`issuer`) of ADFS, which is similar to `https://adfs.domain.com/adfs`.

```
Get-ADFSProperties | Select IdTokenIssuer
```

```
PS C:\Users\Administrator.AWS> Get-ADFSProperties | Select IdTokenIssuer  
  
IdTokenIssuer  
-----  
https://sts.aws.azeroth.zone/adfs
```



In [Step 2. Launch the stack \(p. 31\)](#), enter the parameters below from your ADFS server.

- **OIDCClientId**: `client id`
- **OIDCProvider**: Get the server of the issuer from above step 11

Step 2. Launch the Stack

1. Sign in to the [AWS Management Console](#) and use the button below to launch the AWS CloudFormation template.

| | Launch in AWS Management Console |
|-----------------------|--|
| Launch in AWS Regions |  |

| | Launch in AWS Management Console |
|--|--|
| Launch with custom domain in AWS Regions |  |
| Launch in AWS China Regions |  |

- The template is launched in the default region after you log in to the console. To launch the Clickstream Analytics on AWS solution in a different AWS Region, use the Region selector in the console navigation bar.
- On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
- On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, refer to [IAM and AWS STS quotas](#) in the *AWS Identity and Access Management User Guide*.
- Under **Parameters**, review the parameters for the template and modify them as necessary.
 - This solution uses the following parameters:

| Parameter | Default | Description |
|--------------|------------------|---|
| OIDCClientId | <Requires input> | OpenID Connect client Id. |
| OIDCProvider | <Requires input> | OpenID Connector provider issuer. The issuer must begin with https:// |

Important

By default, this deployment uses TLSv1.0 and TLSv1.1 in CloudFront. However, we recommend that you manually configure CloudFront to use the securer TLSv1.2/TLSv1.3 and apply for a certificate and custom domain to enable this. We highly recommend that you update your TLS configuration and cipher suite selection according to the following recommendations:

- **Transport Layer Security Protocol:** Upgrade to TLSv1.2 or higher
- **Key Exchange:** ECDHE
- **Block Cipher Mode of Operation:** GCM
- **Authentication:** ECDSA
- **Encryption Cipher:** AES256
- **Message Authentication:** SHA(256/384/any hash function except for SHA1)
For example, TLSv1.2_2021 can be used to meet these recommendations.
- If you are launching the solution with custom domain in AWS Regions, this solution has the following additional parameters:

| Parameter | Default | Description |
|------------------|------------------|--|
| Hosted Zone ID | <Requires input> | Choose the public hosted zone ID of Amazon Route 53. |
| Hosted Zone Name | <Requires input> | The domain name of the public hosted zone, for example, example.com. |

| Parameter | Default | Description |
|-------------|------------------|--|
| Record Name | <Requires input> | The sub name (as known as record name in R53) of the domain name of console. For example, enter clickstream, if you want to use custom domain clickstream.example.com for the console. |

- If you are launching the solution in AWS China Regions, this solution has the following additional parameters:

| Parameter | Default | Description |
|------------------|------------------|---|
| Domain | <Requires input> | Custom domain for the web console. Do NOT add http(s) prefix. |
| IamCertificateID | <Requires input> | The ID of the SSL certificate in IAM. The ID is composed of 21 characters of capital letters and digits. Use the list-server-certificates command to retrieve the ID. |

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template creates AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a **CREATE_COMPLETE** status in approximately 15 minutes.

Step 3. Update the callback URL of OIDC client

Important

If you don't deploy stack with custom domain, you must complete below steps.

1. Sign in to the [AWS CloudFormation console](#).
2. Select the solution's stack.
3. Choose the **Outputs** tab.
4. Obtain the **ControlPlaneURL** as the endpoint.
5. Update or add the callback URL to your OIDC.
 - For Cognito, add or update the url in **Allowed callback URL** of your client with value `${ControlPlaneURL}/signin`. The url must start with `https://`.
 - For Keycloak, add or update the url in **Valid Redirect URIs** of your client with value `${ControlPlaneURL}/signin`.
 - For Authing.cn, add or update the url in **Login Callback URL** of **Authentication Configuration**.

Step 4. Setup DNS Resolver

Important

If you deploy stack in AWS Regions, you can skip this step.

This solution provisions a CloudFront distribution that gives you access to the Clickstream Analytics on AWS console.

1. Sign in to the [AWS CloudFormation console](#).
2. Select the solution's stack.
3. Choose the **Outputs** tab.
4. Obtain the **ControlPlaneURL** and **CloudFrontDomainName**.
5. Create a CNAME record for **ControlPlaneURL** in DNS resolver, which points to the domain **CloudFrontDomainName** obtained in previous step.

Step 5. Launch the Web Console

Important

You login credentials are managed by the OIDC provider. Before signing in to the Clickstream Analytics on AWS console, make sure you have created at least one user in the OIDC provider's user pool.

1. Use the previous assigned domain name or generated **ControlPlaneURL** in a web browser.
2. Choose **Sign in**, and navigate to OIDC provider.
3. Enter sign-in credentials. You may be asked to change your default password for first-time login, which depends on your OIDC provider's policy.
4. After the verification is complete, the system opens the Clickstream Analytics on AWS web console.

Once you have logged into the Clickstream Analytics on AWS console, you can start to [create a project \(p. 37\)](#) for your applications.

Launch within VPC

Time to deploy: Approximately 30 minutes

Prerequisites

Review all the [considerations \(p. 15\)](#) and make sure you have the following in the target region you want to deploy the solution:

- At least one Amazon VPC.
- At least two private or isolated [subnets](#) across two Availability Zones (AZs).

Deployment Overview

Use the following steps to deploy this solution on AWS.

[Step 1. Create OIDC client \(p. 27\)](#)

[Step 2. Launch the stack \(p. 31\)](#)

[Step 3. Update the callback url of OIDC client \(p. 33\)](#)

[Step 4. Launch the web console \(p. 34\)](#)

Step 1. Create OIDC client



You can use existing OpenID Connector (OIDC) provider or following [this guide \(p. 27\)](#) to create an OIDC client.

Note

This solution deploys the console in VPC without requiring SSL certificate by default. You have to use an OIDC client to support callback url with http protocol.

Step 2. Launch the stack

1. Sign in to the AWS Management Console and use the button below to launch the AWS CloudFormation template.

| | Launch in AWS Management Console |
|-----------------------------|---|
| Launch in AWS Regions |  |
| Launch in AWS China Regions |  |

2. The template is launched in the default Region after you log in to the console. To launch the Clickstream Analytics on AWS solution in a different AWS Region, use the Region selector in the console navigation bar.
3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, refer to [IAM and AWS STS quotas](#) in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for the template and modify them as necessary.

This solution uses the following parameters:

| Parameter | Default | Description |
|----------------|------------------|---|
| VpcId | <Requires input> | Select the VPC in which the solution will be deployed |
| PrivateSubnets | <Requires input> | Select the subnets in which the solution will be deployed. : you must choose two subnets across two AZs at least. |
| OIDCClientId | <Requires input> | OpenID Connect client Id. |
| OIDCProvider | <Requires input> | OpenID Connector provider issuer. The issuer must begin with https:// |

6. Choose **Next**.

7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template creates AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a **CREATE_COMPLETE** status in approximately 10 minutes.

Step 3. Update the callback URL of OIDC client

1. Sign in to the [AWS CloudFormation console](#).
2. Select the solution's stack.
3. Choose the **Outputs** tab.
4. Obtain the **ControlPlaneURL** as the endpoint.
5. Update or add the callback URL `${ControlPlaneURL}/signin` to your OIDC client.
 - For Keycloak, add or update the url in **Valid Redirect URIs** of your client with value `${ControlPlaneURL}/signin`.
 - For Authing.cn, add or update the url in **Login Callback URL** of **Authentication Configuration**.

Step 4. Launch the web console

Important

Your login credentials is managed by the OIDC provider. Before signing in to the Clickstream Analytics on AWS console, make sure you have created at least one user in the OIDC provider's user pool.

1. Because you deploy the solution console in your VPC without public access, you have to setup a network connection to the solution console serving by an internal application load balancer. There are some options for your reference.
 - (Option 1) Use bastion host, for example, [Linux Bastion Hosts on AWS](#) solution
 - (Option 2) Use AWS Client VPN or [AWS Site-to-Site VPN](#)
 - (Option 3) Use [AWS Direct Connect](#)
2. The application load balancer only allows the traffic from specified security group, you can find the security group id from the output named **SourceSecurityGroup** from the stack you deployed in step 2. Then attach the security group to your bastion host or other source to access the solution console.
3. Use the previously assigned domain name or the generated **ControlPlaneURL** in a web browser.
4. Choose **Sign In**, and navigate to OIDC provider.
5. Enter sign-in credentials. You may be asked to change your default password for first-time login, which depends on your OIDC provider's policy.
6. After the verification is complete, the system opens the Clickstream Analytics on AWS web console.

Once you have logged into the Clickstream Analytics on AWS console, you can start to [create a project \(p. 37\)](#) for your applications.

Getting started

After [deploying the solution \(p. 24\)](#), refer to this section to quickly learn how to leverage the Clickstream Analytics on AWS solution to collect and analyze event data from your applications. This chapter shows you how to create a serverless data pipeline to collect data from an Android application, and view the out-of-the-box analytics dashboards.

- [Step 1: Create a project \(p. 37\)](#).

Create a project.

- [Step 2: Configure a data pipeline \(p. 38\)](#).

Configure a data pipeline with serverless infrastructure.

- [Step 3: Integrate SDK \(p. 39\)](#).

Integrate SDK into your application to automatically collect data and send data to the pipeline.

- [Step 4: Access dashboard \(p. 41\)](#).

View the out-of-the-box dashboards based on the data automatically collected from your applications.

Step 1: Create a project

To get started with the Clickstream Analytics on AWS solution, you need to firstly create a project in the solution console. A project is like a container for all the AWS resources provisioned for collecting and analyzing the clickstream data from your apps.

Prerequisites

Make sure you have deployed the Clickstream Analytics on AWS solution. If you haven't, please refer to the [deployment guide \(p. 24\)](#).

Steps

Following below steps to create a project.

1. Log into **Clickstream Analytics on AWS Management Console**.
2. On the **Home** page, choose **Create Project**.
3. In the window that pops up, enter a project name, for example, `quickstart`.
4. (Optional) Customize the project ID that was automatically created by solution. To do so, click the `edit` icon and update the project ID as per your need.
5. Enter a description for your project, for example, `This is a demo project`.
6. Choose **Next**.
7. Enter an email address to receive notification regarding this project, for example, `email@example.com`, and choose **Next**.
8. Specify an environment type for this project. In this example, select `Dev`.
9. Choose **Create**. Wait until the project creation completed, and you will be directed to the **Projects** page.

We have completed all the steps of creating a project.

Next

- [Configure data pipeline \(p. 38\)](#)

Step 2: Configure data pipeline

After you create a project, you need to configure the data pipeline for it. A data pipeline is a set of connected modules that collect and process the clickstream data sent from your applications. A data pipeline contains four modules of ingestion, processing, modeling and reporting. For more information, see [pipeline management \(p. 42\)](#).

Here we provide an example with steps to create a data pipeline with end-to-end serverless infrastructure.

Steps

1. Sign in to **Clickstream Analytics on AWS Management Console**.
2. In the left navigation pane, choose **Projects**, then select the project you just created in **Step 1**, choose **View Details** in the top right corner to navigate to the project homepage.
3. Choose **Configure pipeline**, and it will bring you to the wizard of creating data pipeline for your project.
4. On the **Basic information** page, fill in the form as follows:
 - AWS Region: **us-east-1**
 - VPC: select a VPC that meets the following requirements
 - At least two public subnets across two different AZs (Availability Zone)
 - At least two private subnets across two different AZs
 - One NAT Gateway or Instance
 - Data collection SDK: **Clickstream SDK**
 - Data location: select an S3 bucket. (You can create one bucket, and select it after choosing **Refresh**.)

Note

If you don't have a VPC meet the criteria, you can create a VPC with VPC wizard quickly. For more information, see [Create a VPC](#).

5. Choose **Next**.
6. On the **Configure ingestion** page, fill in the information as follows:
 - Fill in the **Ingestion endpoint settings** form.
 - Public Subnets: Select two public subnets in two different AZs
 - Private Subnets: Select two private subnets in the same AZs as public subnets
 - Ingestion capacity: Keep the default values
 - Enable HTTPS: Uncheck and then **Acknowledge** the security warning
 - Additional settings: Keep the default values
 - Fill in the **Data sink settings** form.
 - Sink type: **Amazon Kinesis Data Stream(KDS)**
 - Provision mode: **On-demand**
 - In **Additional Settings**, change **Sink Maximum Interval** to 60 and **Batch Size** to 1000
 - Choose **Next** to move to step 3.

Important

Using HTTP is not a recommended configuration for production workload. This example configuration is to help you get started quickly.

7. On the **Configure data processing** information, fill in the information as follows:
 - In the **Enable data processing** form, turn on **Enable data processing**
 - In the **Execution parameters** form,
 - Data processing interval:
 - Select **Fixed Rate**
 - Enter **10**
 - Select **Minutes**
 - Event freshness: **35 Days**
 - In the **Enrichment plugins** form, make sure the two plugins of **IP lookup** and **UA parser** are selected.
 - In the form of **Analytics engine**, fill in the form as follow:
 - Select the box for **Redshift**
 - Select the **Redshift Serverless**
 - Keep **Base RPU** as **8**
 - VPC: select the default VPC or the same one you selected previously in the last step
 - Security group: select the default security group
 - Subnet: select **three** subnets across three different AZs
 - Keep **Athena** selection as default
 - Choose **Next**.
8. On the **Reporting** page, fill in the form as follows:
 - If your AWS account has not subscribed to QuickSight, please follow this [guide](#) to subscribe.
 - After your AWS account subscribed to QuickSight Enterprise edition, choose one of QuickSight users for the solution to use to create out-of-the-box dashboards.
 - Choose **Next**.
9. On the **Review and launch** page, review your pipeline configuration details. If everything is configured properly, choose **Create**.

We have completed all the steps of configuring a pipeline for your project. This pipeline will take about 15 minutes to create, and please wait for the pipeline status change to be **Active** in pipeline detail page.

Next

- [Integrate SDK \(p. 39\)](#)

Step 3: Integrate SDK

Once pipeline's status becomes **Active**, it is ready to receive clickstream data. Now you need to register an application to the pipeline, then you can integrate SDK into your application to enable it to send data to the pipeline.

Steps

1. Log into **Clickstream Analytics on AWS Management Console**.
2. In the left navigation pane, choose **Projects**, then select the project (quickstart) you just created in previous steps, click its title, and it will bring you to the project page.
3. Choose **+ Add application** to start adding application to the pipeline.
4. Fill in the form as follows:

- App name: **test-app**
 - App ID: The system will generate one ID based on the name, and you can customize it if needed.
 - Description: **A test app for Clickstream Analytics on AWS solution**
 - Android package name: leave it blank
 - App Bundle ID: leave it blank
5. Choose **Register App & Generate SDK Instruction**, and wait for the registration to be completed.
 6. Select the tab **Android**, and you will see the detailed instruction of adding SDK into your application. You can follow the steps to add SDK.
 7. Choose **Download the config json file** to download the config file, and keep this file open, which will be used later.

It will take about 3 ~ 5 minutes to update the pipeline with the application you just add. When you see the pipeline status become **Active** again, it is ready to receive data from your application.

We have completed all the steps of adding an application to a project.

Generate sample data

You might not have immediate access to integrate SDK with your app. In this case, we provide a Python script to generate sample data to the pipeline you just configured, so that you can view and experience the analytics dashboards.

Important

Python 3.8+ is required.

1. Clone the repository to your local environment.

```
git clone https://github.com/aws-labs/clickstream-analytics-on-aws.git
```

2. After you cloned the repository, change directory into the `examples/standalone-data-generator` project folder.
3. Install the dependencies of the project.

```
pip3 install requests
```

4. Put `amplifyconfiguration.json` into the root of `examples/standalone-data-generator` which you downloaded in **register an app** step. See the `examples/standalone-data-generator/README.md` for more information.
5. Open an terminal at this project folder location. For example, if you are using Visual Studio Code IDE, at the top of **Visual Studio Code**, click **Terminal** -> **New Terminal** to open a terminal.
6. Copy the following command and paste it to the terminal:

```
python3 create_event.py
```

Let's enter the Enter key in terminal to execute the program. If you see the following output, this means that the program execution is completed.

```
job finished, upload 4360476 events, cost: 95100ms
```

This process will take about 10 minutes with default configuration. After job is finished, you can move to next step.

Next

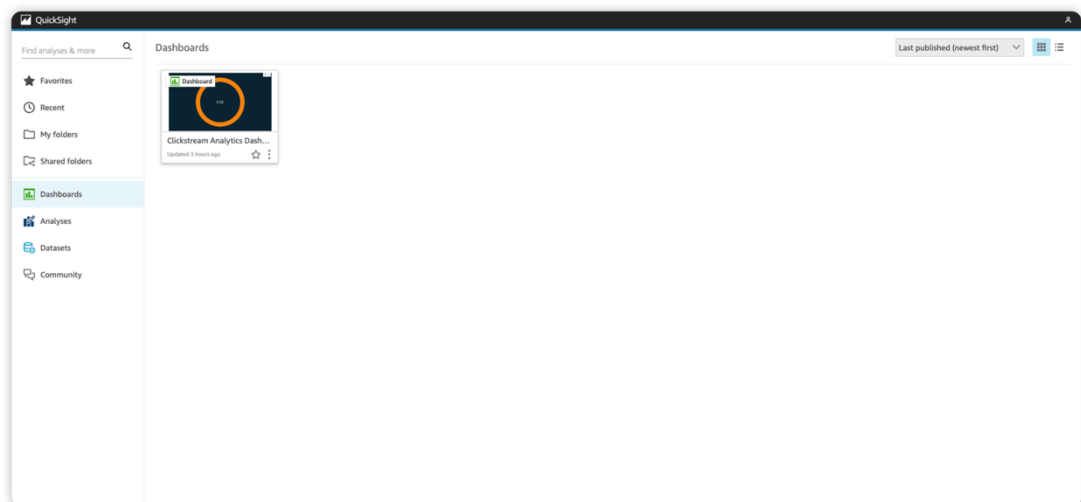
- [View dashboard \(p. 41\)](#)

Step 4: View dashboard

After your application sends data (or the sample data are sent) to the pipeline, you can view the out-of-the-box dashboards that the solution created in QuickSight (normally it takes 15 - 20 minutes to process all of the data).

Steps

1. Log into **Clickstream Analytics on AWS Management Console**.
2. In the left navigation pane, choose **Projects**, then select the project (quickstart) you just created in previous steps, click its title, and it will bring you to the project page.
3. On the project detail page, choose pipeline ID or **View Details** button, and it will bring you to the pipeline detail page.
4. In the pipeline details page, select the **Reporting** tab, and you will see the link to the dashboards created for your app.
5. Choose the dashboard link with the name of your app, and it will bring you to the QuickSight dashboard. You should see a dashboard similar to below in your QuickSight Account.



6. Click the dashboard with name starting with Clickstream.

Congratulations! You have completed the getting started tutorial. You can explore the dashboards or continue to learn more about this solution later.

Next

- [Pipeline management \(p. 42\)](#)

Pipeline management

Data pipeline is the core functionality of this solution. In the Clickstream Analytics on AWS solution, we define a data pipeline as a sequence of integrated AWS services that ingest, process, and model the clickstream data into a destination data warehouse for analytics and visualization. It is also designed to efficiently and reliably collect data from your websites and apps to a S3-based data lake, where it can be further processed, analyzed, and utilized for additional use cases (such as real-time monitoring, and recommendation).

To get a basic understanding of the pipeline, you can refer to [terms and concepts \(p. 3\)](#) for more information.

Prerequisites

You can configure the pipeline in all AWS Regions. For opt-in regions, you need to [enable them](#) first.

Before you start to configure the pipeline in a specific region, make sure you have the following in the target region:

- At least one Amazon VPC.
- At least two public subnets across two AZs in the VPC.
- At least two private (with NAT gateways or instances) subnets across two AZs, or at least two isolated subnets across two AZs in the VPC. If you want to deploy the solution resources in the isolated subnets, you have to create [VPC endpoints](#) for below AWS services,
 - s3, logs, ecr.api, ecr.dkr, ecs, ecs-agent, ecs-telemetry.
 - kinesis-streams if you use KDS as sink buffer in ingestion module.
 - emr-serverless, glue if you enable data processing module.
 - redshift-data, sts, dynamodb, states and lambda if you enable Redshift as analytics engine in data modeling module.
- An Amazon S3 bucket located in the same region.
- If you need to enable Redshift Serverless as analytics engine in data modeling module, you need have subnets across at least three AZs.
- QuickSight Enterprise edition subscription is required if the reporting is enabled.

Ingestion

Ingestion module contains a web service that provides an endpoint to collect data through HTTP/HTTPS requests, which mainly is composed of Amazon Application Load Balancer and Amazon Elastic Container Service. It also supports sinking data into a stream service or S3 directly.

You can create an ingestion module with the following settings:

- [Ingestion endpoint settings \(p. 43\)](#): Create a web service as an ingestion endpoint to collect data sent from your SDKs.
- Data sink settings: Configure how the solution sinks the data for downstream consumption. Currently, the solution supports three types of data sink:
 - [Apache Kafka \(p. 45\)](#)
 - [Amazon S3 \(p. 46\)](#)
 - [Amazon Kinesis Data Stream \(KDS\) \(p. 45\)](#)

Ingestion endpoint

The solution creates a web service as an ingestion endpoint to collect data sent from your SDKs. You can set below configurations for ingestion endpoint.

- **Public subnets:** select at least two existing VPC public subnets, and the Amazon Application Load Balancers (ALBs) will be deployed in these subnets.
- **Private subnets:** select at least two existing VPC private subnets, and the EC2 instances running in ECS will be deployed in these subnets.

Note

The availability zones where the public subnets are located must be consistent with those of the private subnets.

- **Ingestion capacity:** This configuration sets the capacity of the ingestion server, and the ingestion server will automatically scale up or down based on the utilization of the processing CPU.
 - Minimum capacity: The minimum capacity to which the ingestion server will scale down.
 - Maximum capacity: The maximum capacity to which the ingestion server will scale up.
 - Warm pool: Warm pool gives you the ability to decrease latency for your applications that have exceptionally long boot time. For more information, please refer to [Warm pools for Amazon EC2 Auto Scaling](#).
- **Enable HTTPS:** Users can choose HTTPS/HTTP protocol for the Ingestion endpoint.
 - Enable HTTPS: If users choose to enable HTTPS, the ingestion server will provide HTTPS endpoint.
 - Domain name: input a domain name. Once the ingestion server is created, use the custom endpoint to create an alias or CNAME mapping in your Domain Name System (DNS) for the custom endpoint.
 - SSL Certificate: User need to select an ACM certificate corresponding to the domain name that you input. If there is no ACM certificate, please refer [create public certificate](#) to create it.
 - Disable HTTPS: If users choose to disable HTTPS, the ingestion server will provide HTTP endpoint.

Important

Using HTTP protocol is not secure, because data will be sent without any encryption, and there are high risks of data being leaked or tampered during transmission. Please acknowledge the risk to proceed.

- **Additional Settings**
 - Request path: User can input the path of ingestion endpoint to collect data, the default path is "/collect".
 - AWS Global Accelerator: User can choose to create an accelerator to get static IP addresses that act as a global fixed entry point to your ingestion server, which will improve the availability and performance of your ingestion server. Note that additional charges apply.
 - Authentication: User can use OIDC provider to authenticate the request sent to your ingestion server. If you plan to enable it, please create an OIDC client in the OIDC provider then create a secret in AWS Secret Manager with information:
 - issuer
 - token endpoint
 - User endpoint
 - Authorization endpoint
 - App client ID
 - App Client Secret

The format is like:

```
{  
  "issuer": "xxx",
```



```
}
  "userEndpoint": "xxx",
  "authorizationEndpoint": "xxx",
  "tokenEndpoint": "xxx",
  "appClientId": "xxx",
  "appClientSecret": "xxx"
}
```

Note

In the OIDC provider, you need to add `https://<ingestion server endpoint>/oauth2/idpresponse` to "Allowed callback URLs"

- Access logs: ALB supports delivering detailed logs of all requests it receives. If you enable this option, the solution will automatically enable access logs for you and store the logs into the S3 bucket you selected in previous step.

Important

The bucket must have a bucket policy that grants Elastic Load Balancing permission to write the access logs to the bucket. For details, refer to [Step 2: Attach a policy to your S3 bucket](#). Below is an example policy for the bucket in Regions available before August 2022.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<elb-account-id>:root"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::<BUCKET>/clickstream/*"
    }
  ]
}
```

You need to replace `elb-account-id` with the ID of the AWS account for Elastic Load Balancing in your Region:

- US East (N. Virginia) – 127311923021
- US East (Ohio) – 033677994240
- US West (N. California) – 027434742980
- US West (Oregon) – 797873946194
- Africa (Cape Town) – 098369216593
- Asia Pacific (Hong Kong) – 754344448648
- Asia Pacific (Jakarta) – 589379963580
- Asia Pacific (Mumbai) – 718504428378
- Asia Pacific (Osaka) – 383597477331
- Asia Pacific (Seoul) – 600734575887
- Asia Pacific (Singapore) – 114774131450
- Asia Pacific (Sydney) – 783225319266
- Asia Pacific (Tokyo) – 582318560864
- Canada (Central) – 985666609251
- Europe (Frankfurt) – 054676820928
- Europe (Ireland) – 156460612806
- Europe (London) – 652711504416
- Europe (Milan) – 635631232127
- Europe (Paris) – 009996457667

- Europe (Stockholm) – 897822967062
- Middle East (Bahrain) – 076674570225
- South America (São Paulo) – 507241528517
- China (Beijing) – 638102146993
- China (Ningxia) – 037604701340

Data sink – Kafka

This data sink will stream the clickstream data collected by the ingestion endpoint into a topic in a Kafka cluster. Currently, solution support Amazon Managed Streaming for Apache Kafka (Amazon MSK) or a self-hosted Kafka cluster.

Amazon MSK

- **Select an existing Amazon MSK cluster.** Select an MSK cluster from the drop-down list, and the MSK cluster needs to meet the following requirements:
 - MSK cluster and this solution need to be in the same VPC
 - Enable **Unauthenticated access** in Access control methods
 - Enable **Plaintext** in Encryption
 - Set **auto.create.topics.enable** as true in MSK cluster configuration. This configuration sets whether MSK cluster can create topic automatically.
 - The value of **default.replication.factor** cannot be larger than the number of MKS cluster brokers

Note

If there is no MSK cluster, the user needs to create an MSK Cluster following above requirements.

- **Topic:** The user can specify a topic name. By default, the solution will create a topic with “project-id”.

Connector

Enable solution to create Kafka connector and a custom plugin for this connector. This connector will sink the data from Kafka cluster to S3 bucket.

Additional Settings

- **Sink maximum interval:** Specifies the maximum length of time (in seconds) that records should be buffered before streaming to the AWS service.
- **Batch size:** The maximum number of records to deliver in a single batch.

Data sink – Kinesis

This data sink will stream the clickstream data collected by the ingestion endpoint into KDS. The solution will create a KDS in your AWS account based on your specifications.

Provision mode

Two modes are available: **On-demand** and **Provisioned**

- **On-demand:** In this mode, KDS shards are provisioned based on the workshop automatically. On-demand mode is suited for workloads with unpredictable and highly-variable traffic patterns.
- **Provisioned:** In this mode, KDS shards are set at creation. The provisioned mode is suited for predictable traffic with capacity requirements that are easy to forecast. You can also use the provisioned mode if you want fine-grained control over how data is distributed across shards.
- **Shard number:** With the provisioned mode, you must specify the number of shards for the data stream. For more information about shard, please refer to [provisioned mode](#)

Additional settings

- **Sink maximum interval:** With this configuration, you can specify the maximum interval (in seconds) that records should be buffered before streaming to the AWS service.
- **Batch size:** With this configuration, you can specify the maximum number of records to deliver in a single batch.

Data sink - S3

In this option, clickstream data is buffered in the memory of ingestion Server, then sink into a S3 bucket. This option provides the best cost-performance in case real-time data consumption is not required.

Note

Unlike Kafka and KDS data sink, this option buffers data in the ingestion server and responses 200 code to SDK client before sink into S3, so there is chance data could be lost while ingestion server fails and auto-scaled machine is in the process of creation. But it is worth to note that this probability is very low because of the High-availability design of the solution.

- **S3 URI:** You can select an amazon s3 bucket. The data will be sank into this bucket.
- **S3 prefix:** By default, the solution adds prefix of "/YYYY/MM/dd/HH" (in UTC) to the data files when delivered to S3. You can provide additional prefix which will be added before .
- **Buffer size:** Specify the data size to buffer before sending to Amazon S3. The higher buffer size may be lower in cost with higher latency, the lower buffer size will be faster in delivery with higher cost. Min: 1 MiB, Max: 50 MiB
- **Buffer interval:** Specify the max interval (second) for saving buffer to S3. 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. Min: 60 Second, Max: 3600 Second

Data processing

Clickstream Analytics on AWS provides an inbuilt data schema to parse and model the raw event data sent from your web and mobile apps, which makes it easy for you to analyze the data in analytics engines (such as RedShift and Athena).

Data Processing module includes two functionalities:

- **Transformation:** Extract the data from files sank by ingestion module, then parse each event data and transform them to solution data model.
- **Enrichment:** Add additional dimensions/fields to event data.

This chapter includes:

- [Data schema \(p. 47\)](#)
- [Configure execution parameters \(p. 53\)](#)
- [Configure custom plugins \(p. 53\)](#)

Data schema

This article explains the data schema and format in Clickstream Analytics on AWS. This solution uses an **event-based** data model to store and analyze clickstream data, every activity (e.g., click, view) on the clients is modeled as an event with multiple dimensions. Dimensions are common for all events, but customers have the flexibility to use JSON store value for some dimensions (e.g., event parameters, user attributes) to add information that are specific for their business. Those JSON will be stored in special data types which allow customers to unnest the values in the analytics engines.

Database and table

For each project, the solution creates a database with name of <project-id> in Redshift and Athena. In Redshift, each App will have a schema with name of app_id, within which event data are stored in ods_event tables, user-related attributes are stored in dim_user table. In Athena, data from all apps in the project will be stored in ods_event table with partitions of app_id, year, month, and day.

Columns

Each column in the ods_event table represents an event-specific parameter. Note that some parameters are nested within a Super field in Redshift or Array in Athena, and some fields such as items and event_params are repeatable. Table columns are described below.

Event fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|--------------------------|----------------------|--------------------|---|
| event_id | VARCHAR | STRING | Unique ID for the event. |
| event_date | DATE | DATE | The date when the event was logged (YYYYMMDD format in UTC). |
| event_timestamp | BIGINT | BIGINT | The time (in microseconds, UTC) when the event was logged on the client. |
| event_previous_timestamp | BIGINT | BIGINT | The time (in microseconds, UTC) when the event was previously logged on the client. |
| event_name | VARCHAR | STRING | The name of the event. |
| event_value_in_usd | BIGINT | BIGINT | The currency-converted value (in USD) of the event's "value" parameter. |

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|--------------------------|----------------------|--------------------|--|
| event_bundle_sequence_id | BIGINT | BIGINT | The sequential ID of the bundle in which these events were uploaded. |
| ingest_timestamp | BIGINT | BIGINT | Timestamp offset between collection time and upload time in micros. |

Event parameter fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|---------------------------------|----------------------|--------------------|---|
| event_params | SUPER | ARRAY | Event parameters. |
| event_params.key | VARCHAR | | The name of the event parameter. |
| event_params.value | SUPER | ARRAY | A record containing the event parameter's value. |
| event_params.value.string_value | VARCHAR | STRING | If the event parameter is represented by a string, such as a URL or campaign name, it is populated in this field. |
| event_params.value.int_value | BIGINT | INTEGER | If the event parameter is represented by an integer, it is populated in this field. |
| event_params.value.double_value | DOUBLE PRECISION | FLOAT | If the event parameter is represented by a double value, it is populated in this field. |
| event_params.value.float_value | DOUBLE PRECISION | FLOAT | If the event parameter is represented by a floating point value, it is populated in this field. This field is not currently in use. |

User fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|------------|----------------------|--------------------|---|
| user_id | VARCHAR | STRING | The unique ID assigned to a user through setUserId() API. |

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|----------------------------|----------------------|--------------------|--|
| user_pseudo_id | VARCHAR | STRING | The pseudonymous id generated by SDK for the user. |
| user_first_touch_timestamp | BIGINT | BIGINT | The time (in microseconds) at which the user first opened the app or visited the site. |

User property fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|--|----------------------|--------------------|---|
| user_properties | SUPER | ARRAY | Properties of the user. |
| user_properties.key | VARCHAR | STRING | The name of the user property. |
| user_properties.value | SUPER | ARRAY | A record for the user property value. |
| user_properties.value.string_value | VARCHAR | STRING | The string value of the user property. |
| user_properties.value.int_value | BIGINT | BIGINT | The integer value of the user property. |
| user_properties.value.double_value | DOUBLE PRECISION | FLOAT | The double value of the user property. |
| user_properties.value.float_value | DOUBLE PRECISION | FLOAT | This field is currently unused. |
| user_properties.value.set_timestamp_micros | BIGINT | BIGINT | The time (in microseconds) at which the user property was last set. |
| user_ltv | SUPER | ARRAY | The Lifetime Value of the user. |
| user_ltv.revenue | DOUBLE PRECISION | FLOAT | The Lifetime Value (revenue) of the user. |
| user_ltv.currency | DOUBLE PRECISION | FLOAT | The Lifetime Value (currency) of the user. |

Device fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|--------------------------|----------------------|--------------------|------------------------|
| device.mobile_brand_name | VARCHAR | STRING | The device brand name. |

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|---------------------------------|----------------------|--------------------|---|
| device.mobile_model_name | VARCHAR | STRING | The device model name. |
| device.manufacturer | VARCHAR | STRING | The device manufacturer name. |
| device.carrier | VARCHAR | STRING | The device network provider name. |
| device.network_type | VARCHAR | STRING | The network_type of the device, e.g., WIFI, 5G |
| device.operating_system | VARCHAR | STRING | The operating system of the device. |
| device.operating_system_version | VARCHAR | STRING | The OS version. |
| device.vendor_id | VARCHAR | STRING | IDFV (present only if IDFA is not collected). |
| device.advertising_id | VARCHAR | STRING | Advertising ID/IDFA. |
| device.system_language | VARCHAR | STRING | The OS language. |
| device.time_zone_offset_seconds | BIGINT | BIGINT | The offset from GMT in seconds. |
| device.ua_browser | VARCHAR | STRING | The browser in which the user viewed content, derived from User Agent string |
| device.ua_browser_version | VARCHAR | STRING | The version of the browser in which the user viewed content, derive from User Agent |
| device.ua_device | VARCHAR | STRING | The device in which user viewed content, derive from User Agent. |
| device.ua_device_category | VARCHAR | STRING | The device category in which user viewed content, derive from User Agent. |
| device.screen_width | VARCHAR | STRING | The screen width of the device. |
| device.screen_height | VARCHAR | STRING | The screen height of the device. |

Geo fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|-------------------|----------------------|--------------------|--|
| geo.continent | VARCHAR | STRING | The continent from which events were reported, based on IP address. |
| geo.sub_continent | VARCHAR | STRING | The subcontinent from which events were reported, based on IP address. |
| geo.country | VARCHAR | STRING | The country from which events were reported, based on IP address. |
| geo.region | VARCHAR | STRING | The region from which events were reported, based on IP address. |
| geo.metro | VARCHAR | STRING | The metro from which events were reported, based on IP address. |
| geo.city | VARCHAR | STRING | The city from which events were reported, based on IP address. |
| geo.locale | VARCHAR | STRING | The locale information obtained from device. |

Traffic fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|-----------------------|----------------------|--------------------|---|
| traffic_source.name | VARCHAR | STRING | Name of the marketing campaign that acquired the user when the events were reported. |
| traffic_source.medium | VARCHAR | STRING | Name of the medium (paid search, organic search, email, etc.) that acquired the user when the events were reported. |
| traffic_source.source | VARCHAR | STRING | Name of the network source that acquired the user when the event were reported. |

App Info fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|-------------------------|----------------------|--------------------|--|
| app_info.id | VARCHAR | STRING | The package name or bundle ID of the app. |
| app_info.app_id | VARCHAR | STRING | The App ID (created by this solution) associated with the app. |
| app_info.install_source | VARCHAR | STRING | The store that installed the app. |
| app_info.version | VARCHAR | STRING | The app's versionName (Android) or short bundle version. |

Other fields

| Field Name | Data Type - Redshift | Data Type - Athena | Description |
|------------------|----------------------|--------------------|--|
| platform | VARCHAR | STRING | The data stream platform (Web, IOS or Android) from which the event originated. |
| project_id | VARCHAR | STRING | The project id associated with the app. |
| items | SUPER | ARRAY | Key-value records contain information about items associated with the events |
| ecommerce | SUPER | ARRAY | Key-value records contain information about ecommerce-specific attributes associated with the events |
| event_dimensions | SUPER | ARRAY | Key-value records contain information about additional dimensions associated with the events |
| privacy_info | SUPER | ARRAY | Key-value records contain information about user privacy setting associated with the events |

Execution parameters

Execution parameters control the key behaviors of the transformation and enrichment jobs.

Parameters

You can configure the following **Execution parameters** after you turn on **Enable data processing**.

| Parameter | Description | Values |
|--|---|--|
| Data processing interval/Fixed Rate | Specify the interval to batch the data for data processing by fixed rate | 1 hour 12 hours 1 day |
| Data processing interval/Cron Expression | Specify the interval to batch the data for data processing by cron expression | cron(0 * * ? *) cron(0 0,12 * ? *) cron(0 0 * ? *) |
| Event freshness | Specify the days after which the solution will ignore the event data. For example, if you specify 3 days for this parameter, the solution will ignore any event which arrived more than 3 days after the events are triggered | 3 days 5 days 30 days |

Cron expression syntax

Syntax

cron(minutes hours day-of-month month day-of-week year)

For more information, refer to [Cron-based schedules](#).

Processing plugin

There are two types of plugins: **transformer** and **enrichment**. You can choose to have only one **transformer**, and zero or multiple **enrichment**.

Built-in plugins

Below plugins are provided by Clickstream Analytics on AWS.

| Plugin name | Type | Description |
|--------------|------------|---|
| UAEnrichment | enrichment | User-agent enrichment, use ua_parser Java library to enrich User- |

| Plugin name | Type | Description |
|--------------|------------|--|
| | | Agent in the HTTP header to ua_browser,ua_browser_version,ua_os,ua_os |
| IpEnrichment | enrichment | IP address enrichment, use GeoLite2 data by MaxMind to enrich IP to city, continent, country |

The UAEnrichment uses [UA Parser](#) to parse user-agent in Http header

The IpEnrichment plugin uses [GeoLite2-City data](#) created by MaxMind, available from <https://www.maxmind.com>

Custom plugins

You can add custom plugins to transform raw event data or enrich the data for your need.

Note

To add custom plugins, you must develop your own plugins firstly, see [Develop Custom Plugins](#).

To add your plugins, choose **Add Plugin**, which will open a new window, in which you can upload your plugins.

1. Enter the plugin **Name** and **Description**
2. Choose **Plugin Type**
3. **Enrichment**: Plugin to add fields into event data collected by SDK (both Clickstream SDK or third-party SDK)
4. **Transformation**: A plugin used to transform a third-party SDK's raw data into solution built-in schema
5. Upload plugin java JAR file
6. (Optional) Upload the dependency files if any
7. **Main function class**: fill the full class name of your plugin class name, e.g. com.company.sol.CustomTransformer

Develop Custom Plugins

The simplest way to develop custom plugins is making changes based on our example project.

1. Clone/Fork the example project.

```
git clone https://github.com/awslabs/clickstream-analytics-on-aws.git
cd examples/custom-plugins
```

- For enrichment plugin, please refer to the example: custom-enrich/
 - For transformer plugin, please refer to the example: custom-sdk-transformer/
2. Change packages and classes name as you desired.
 3. Implement the method `public Dataset<row> transform(Dataset<row> dataset)` to do transformation or enrichment.
 4. (Optional) Write test code.
 5. Run gradle to package code to jar `./gradlew clean build`.
 6. Get the jar file in build output directory `./build/libs/`.

Data modeling

Once the data pipeline processes the event data, you can load the data into an analytics engine for data modeling, such as Redshift or Athena, where data will be aggregated and organized into different views (such as event, device, session), as well as calculated metrics that are commonly used. Below are the preset data views this solution provides if you choose to enable data modeling module.

Preset data views

| Data model name | Redshift | Athena | Description |
|-----------------------------------|-------------------|--------|--|
| clickstream_ods_event_view | Materialized view | View | A view that contains all event dimensions |
| clickstream_ods_event_attr_view | Materialized view | View | A view that contains all event parameters. |
| clickstream_user_dim_view | Materialized view | View | A view that contains all user dimensions. |
| clickstream_session_view | Materialized view | View | A view that contains all session dimension and relevant metrics, for example, session duration, session views. |
| clickstream_user_attr_view | Materialized view | View | A view that contains all user custom attributes. |
| clickstream_retention_view | Materialized view | View | A view that contains metrics of retentions by dates and return days. |
| clickstream_lifecycle_daily_view | Materialized view | View | A view that contains metrics of user number by lifecycle stages by day, that is, New, Active, Return, Churn. |
| clickstream_lifecycle_weekly_view | Materialized view | View | A view that contains metrics of user number by lifecycle stages by week, that is, New, Active, Return, Churn. |
| clickstream_path_view | Materialized view | View | A view that contains information about user journey within each session. |

You can choose to use Redshift or Athena, or both.

Note: We recommended you select both, that is, using Redshift for hot data modeling and using Athena for all-time data analysis.

You can set below configurations for Redshift.

- **Redshift Mode:** Select Redshift serverless or provisioned mode.
- **Serverless mode**
 - **Base RPU:** RPU stands for Redshift Processing Unit. Amazon Redshift Serverless measures data warehouse capacity in RPUs, which are resources used to handle workloads. The base capacity specifies the base data warehouse capacity Amazon Redshift uses to serve queries and is specified in RPUs. Setting higher base capacity improves query performance, especially for data processing jobs that consume a lot of resources.
 - **VPC:** A virtual private cloud (VPC) based on the Amazon VPC service is your private, logically isolated network in the AWS Cloud.
 - Note**
If you place the cluster within the isolated subnets, the VPC must have VPC endpoints for S3, Logs, Dynamodb, STS, States, Redshift and Redshift-data service.
 - **Security Group:** This VPC security group defines which subnets and IP ranges can access the endpoint of Redshift cluster.
 - **Subnets:** Select at least three existing VPC subnets.
 - Note**
We recommend using private subnets to deploy in accordance with the security best practices.
- **Provisioned mode**
 - **Redshift Cluster:** With a provisioned Amazon Redshift cluster, you build a cluster with node types that meet your cost and performance specifications. You have to set up, tune, and manage Amazon Redshift provisioned clusters.
 - **Database user:** The solution requires permissions to access and create database in Redshift cluster. By default, it grants Redshift Data API with the permissions of the admin user to execute the commands to create DB, tables, and views, as well as loading data.
 - **Data range:** Considering the cost performance issue of having Redshift to save all the data, we recommend that Redshift save hot data and that all data are stored in S3. It is necessary to delete expired data in Redshift on a regular basis.
- **Additional Settings**
 - **User table upsert frequency:** Since all versions of user properties are saved in Redshift. We create a user-scoped custom dimension table dim_users in DWD layer so the BI dashboard can report on the latest user property. The workflow run on schedule to upsert (update and insert) users.
 - **Athena:** Choose Athena to query all data on S3 using the table created in the AWS Glue Data Catalog.

Reporting

Once the data are processed and modeled by the data pipeline, the solution supports creating out-of-the-box dashboards in QuickSight.

Note

To enable this module, your AWS account needs to have subscription in QuickSight. If it hasn't, please follow this [sign up for Amazon QuickSight](#) to create a subscription first.

You need to select a **QuickSight User** for the solution to use to create datasets and dashboards. If you don't have one, click **Create a user** button on the solution console, which will automatically create a user for you.

Pipeline maintenance

This solution provides three features to help you manage and operate the data pipeline after it gets created.

Monitoring and Alarms

The solution collects metrics from each resource in the data pipeline and creates monitoring dashboards in CloudWatch, which provides you a comprehensive view into the pipeline status. It also provides a set of alarms that will notify project owner if anything goes abnormal.

Following are steps to view monitoring dashboards and alarms.

Monitoring dashboards

To view monitoring dashboard for a data pipeline, follows below steps:

1. Go to project detail page.
2. Choose project id or **View Details**, which will direct to the pipeline detail page.
3. Select the "**Monitoring**" tab.
4. In the tab, choose **View in CloudWatch**, which will direct you to the monitoring dashboard.

Alarms

To view alarms for a data pipeline, follows below steps:

1. Go to project detail page.
2. Choose project id or **View Details**, which will direct to the pipeline detail page.
3. Select the "**Alarms**" tab.
4. In the tab, you can view all the alarms. You can also choose **View in CloudWatch**, which will direct you to CloudWatch alarm pages to view alarm details.
5. You can also enable or disable an alarm by selecting the alarm then choosing **Enable** or **Disable**.

Pipeline modification

You are able to modify some configuration the data pipeline after it created, follow below steps to update a pipeline.

1. Go to project detail page.
2. Choose project id or **View Details**, which will direct to the pipeline detail page.
3. In the project details page, choose **Edit**, which will bring you to the pipeline creation wizard page. Note that some configuration are in disable mode, which means they cannot be updated after creation.
4. If needed, update those configuration options which are editable.
5. After editing the configuration, choose **Next** until you reach last page, and choose **Save**.

You will see pipeline is in Updating status.

Pipeline upgrade

As we continue to enhance the solution, there will be new versions of pipeline available for you to use after you update the solution control panel.

1. Go to project detail page.
2. Choose project id or **View Details**, which will direct to the pipeline detail page.
3. In the project details page, choose **Upgrade**. You will be prompted to confirm the upgrade action.
4. Choose **Confirm**, and the pipeline will be in Upgrading status.

SDK manual

Clickstream Analytics on AWS provides purpose-built software development kits (SDKs), which can make it easier for you to report events to the data pipeline created in the solution. Currently, the solution supports the following platforms:

- [Android \(p. 59\)](#)
- [Swift \(p. 68\)](#)

Key features and benefits

- **Automatic data collection.** Clickstream SDKs provide built-in capabilities to automatically collect common events, such as screen view, session, and user engagement, so that you only need to focus on recording business-specific events.
- **Ease of use.** Clickstream SDKs provide multiple APIs and configuration options to simplify the event reporting and attribute setting process.
- **Cross-platform analytics.** Clickstream SDKs are consistent in event data structure, attribute validation rules, and event sending mechanism, so that data can be normalized in the same structure for cross-platform analytics.

Note

All Clickstream SDKs are open source under Apache 2.0 License in [Github](#). You can customize the SDKs if needed. All contributions are welcome.

Android SDK

Introduction

Clickstream Android SDK can help you easily collect and report in-app events from Android devices to AWS. As part of the solution Clickstream Analytics on AWS, this SDK provisions data pipeline to ingest and process event data into AWS services such as Amazon S3, and Amazon Redshift.

The SDK is based on the Amplify for Android SDK Core Library and developed according to the Amplify Android SDK plug-in specification. In addition, the SDK is equipped with features that automatically collect common user events and attributes (for example, screen view, and first open) to simplify data collection for users.

Platform support

Clickstream Android SDK supports Android 4.1 (API level 16) and later.

Integrate the SDK

1 Include the SDK

Add the following dependency to your app module's `build.gradle` file.

```
dependencies {  
    implementation 'software.aws.solution:clickstream:0.5.1'
```

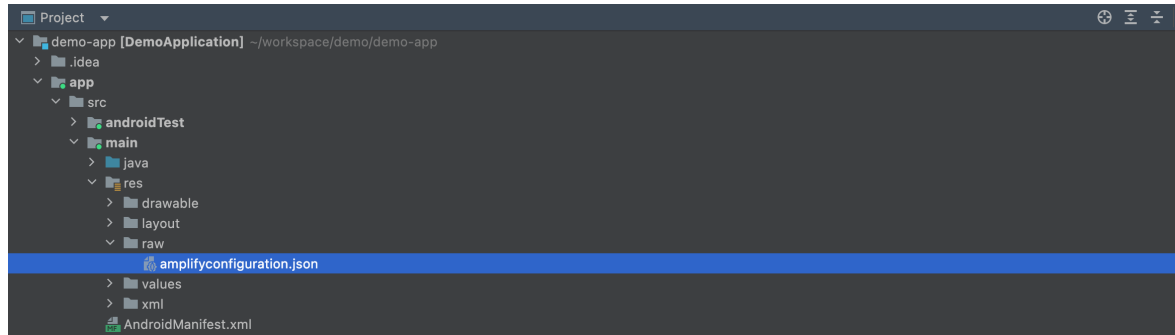


```
}
```

Next, synchronize your project with [the latest version](#).

2 Configure parameters

Find the `res` directory under your project `/app/src/main`, and manually create a `raw` folder in the `res` directory.



Download your `amplifyconfiguration.json` file from your clickstream control plane, and paste it to the `raw` folder. The JSON file is like:

```
{
  "analytics": {
    "plugins": {
      "awsClickstreamPlugin": {
        "appId": "appId",
        "endpoint": "https://example.com/collect",
        "isCompressEvents": true,
        "autoFlushEventsInterval": 10000,
        "isTrackAppExceptionEvents": false
      }
    }
  }
}
```

In the file, your `appId` and `endpoint` are already configured. The explanation for each property is as follows:

- **appId**: the app id of your project in control plane.
- **endpoint**: the endpoint url you will upload the event to AWS server.
- **isCompressEvents**: whether to compress event content when uploading events, and the default value is true
- **autoFlushEventsInterval**: event sending interval, and the default value is 10s
- **isTrackAppExceptionEvents**: whether auto track exception event in app, and the default value is false

3 Initialize the SDK

Initialize the SDK in the application `onCreate()` method.

```
import software.aws.solution.clickstream.ClickstreamAnalytics;

public void onCreate() {
    super.onCreate();
```

```
try{
    ClickstreamAnalytics.init(this);
    Log.i("MyApp", "Initialized ClickstreamAnalytics");
} catch (AmplifyException error){
    Log.e("MyApp", "Could not initialize ClickstreamAnalytics", error);
}
}
```

4 Configure the SDK

After initializing the SDK, you can use the following code to customize it.

Important

This configuration will override the default configuration in `amplifyconfiguration.json` file.

```
import software.aws.solution.clickstream.ClickstreamAnalytics;

// config the SDK after initialize.
ClickstreamAnalytics.getClickStreamConfiguration()
    .withAppId("appId")
    .withEndpoint("https://example.com/collect")
    .withAuthCookie("your authentication cookie")
    .withSendEventsInterval(10000)
    .withSessionTimeoutDuration(1800000)
    .withTrackAppExceptionEvents(false)
    .withLogEvents(true)
    .withCustomDns(CustomOkhttpDns.getInstance())
    .withCompressEvents(true);
```

5 Record event

Add the following code where you need to report an event. For more information, refer to [Github](#).

```
import software.aws.solution.clickstream.ClickstreamAnalytics;
import software.aws.solution.clickstream.ClickstreamEvent;

ClickstreamEvent event = ClickstreamEvent.builder()
    .name("PasswordReset")
    .add("Channel", "SMS")
    .add("Successful", true)
    .add("ProcessDuration", 78.2)
    .add("UserAge", 20)
    .build();
ClickstreamAnalytics.recordEvent(event);

// for record an event directly
ClickstreamAnalytics.recordEvent("button_click");
```

Data format definition

Data types

Clickstream Android SDK supports the following data types:

| Data type | Range | Example |
|-----------|-------------------------|---------|
| int | -214748364 ~ 2147483647 | 12 |

| Data type | Range | Example |
|-----------|--|---------------|
| long | -9223372036854775808 ~ 9223372036854775807 | 26854775808 |
| double | 4.9E-324 ~ 1.7976931348623157E308 | 3.14 |
| boolean | true, false | true |
| String | 1024 characters maximum | "Clickstream" |

Naming rules

1. The event name and attribute name cannot start with a number, and only contain uppercase and lowercase letters, numbers, and underscores. In case of an invalid event name, it will throw `IllegalArgumentException`. In case of an invalid attribute name or user attribute name, it will discard the attribute and record error.
2. Do not use `_` as prefix in an event name or attribute name, because the `_` prefix is reserved for the solution.
3. The event name and attribute name are case sensitive, so `Add_to_cart` and `add_to_cart` will be recognized as two different event names.

Event and attribute limitation

To improve the efficiency of querying and analysis, we apply limitations to event data as follows:

| Item | Recommended | Maximum | Handle strategy if the limit is exceeded |
|---------------------------------|--------------------------|----------------------|---|
| Length of event name | less than 25 characters | 50 characters | throw <code>IllegalArgumentException</code> |
| Length of event attribute name | less than 25 characters | 50 characters | discard, log and record error |
| Length of event attribute value | less than 100 characters | 1024 characters | discard, log and record error |
| Event attribute per event | less than 50 attributes | 500 event attributes | discard, log and record error |
| Number of user attributes | less than 25 attributes | 100 user attributes | discard, log and record error |
| Length of user attribute name | less than 25 characters | 50 characters | discard, log and record error |
| Length of user attribute value | less than 50 characters | 256 characters | discard, log and record error |

Important

- The character limits are the same for single-width character languages (for example, English) and double-width character languages (for example, Chinese).

- The limitation for event attribute per event involves both common attributes and preset attributes.
- If the attribute or user attribute with the same name is added more than twice, the latest value will apply.

Preset events and attributes

Preset events

Automatically collected events:

| Event name | Triggered | Event attributes |
|------------------|--|---|
| _session_start | when users app comes to foreground for the first time and there is no on-going session | _session_id _session_start_timestamp _session_duration |
| _screen_view | when the activity callback onResume() method | _screen_name _screen_id _previous_screen_name _previous_screen_id _entrances _engagement_time_msec |
| _app_exception | when the app crashes | _exception_message _exception_stack |
| _app_update | when the app is updated to a new version and launched again | _previous_app_version |
| _first_open | when the user launches an app the first time after installation | |
| _os_update | when device operating system is updated to a new version | _previous_os_version |
| _user_engagement | when the app is in the foreground for at least one second | _engagement_time_msec |
| _profile_set | when the addUserAttributes() or setUserId() API is called | |

Session definition

In Clickstream Android SDK, we do not limit the total time of a session. As long as the time between the next entry of the app and the last exit time is within the allowable timeout period, the current session is considered to be continuous.

- **_session_start:** When the app starts for the first time, or the app was launched to the foreground and the time between the last exit exceeded `session_time_out` period.
- **_session_duration:** We calculate the `_session_duration` by minus the current event create timestamp and the session's `_session_start_timestamp`. This attribute will be added in every event during the session.
- **session_time_out:** By default, it is 30 minutes, which can be customized through the configuration API.
- **_session_number:** The total number of sessions by distinct session id, and `_session_number` will appear in every event's attribute object.

User engagement definition

In Clickstream Android SDK, we define the `user_engagement` as the app which is in the foreground for at least one second.

- **when to send:** We send the event when the app navigates to background or navigates to another app.
- **engagement_time_msec:** We count the time from when the app comes in the foreground to when the app goes to the background.

Common attributes and reserved attributes

Sample event structure

```
{
  "hashCode": "80452b0",
  "unique_id": "c84ad28d-16a8-4af4-a331-f34cdc7a7a18",
  "event_type": "PasswordReset",
  "event_id": "460daa08-0717-4385-8f2e-acb5bd019ee7",
  "timestamp": 1667877566697,
  "device_id": "f24bec657ea8eff7",
  "platform": "Android",
  "os_version": "10",
  "make": "HUAWEI",
  "brand": "HUAWEI",
  "model": "TAS-AN00",
  "locale": "zh_CN_#Hans",
  "carrier": "CDMA",
  "network_type": "Mobile",
  "screen_height": 2259,
  "screen_width": 1080,
  "zone_offset": 28800000,
  "system_language": "zh",
  "country_code": "CN",
  "sdk_version": "0.2.0",
  "sdk_name": "aws-solution-clickstream-sdk",
  "app_version": "1.0",
  "app_package_name": "com.notepad.app",
  "app_title": "Notepad",
  "app_id": "notepad-4a929eb9",
  "user": {
    "_user_id": {
      "value": "312121",
```

```

        "set_timestamp": 1667877566697
    },
    "_user_name": {
        "value": "carl",
        "set_timestamp": 1667877566697
    },
    "_user_first_touch_timestamp": {
        "value": 1667877267895,
        "set_timestamp": 1667877566697
    }
},
"attributes": {
    "Channel": "SMS",
    "Successful": true,
    "Price": 120.1,
    "ProcessDuration": 791,
    "_session_id": "dc7a7a18-20221108-031926703",
    "_session_start_timestamp": 1667877566703,
    "_session_duration": 391809,
    "_session_number": 1
}
}

```

All user attributes will be stored in user object, and all custom and global attributes in attributes object.

Common attributes

| Attribute name | Description | It is generated... | It is used to or for... |
|----------------|------------------------------|---|--|
| hashCode | the event object's hash code | generated from Integer.toHexString() | distinguish events AnalyticsEvent.hashCode()) |
| app_id | clickstream app id | generated when clickstream app create from solution control plane | identify the events for your apps |
| unique_id | the unique id for user | generated from UUID.randomUUID() to associate the behavior of logging in and not logging in during the SDK first initialization. It will be changed after user relogin to another user who never login, and when user relogin to the previous user in same device. The unique_id will reset to the previous user's unique_id. | identity different users and associate the behavior of logging in and not logging in |
| device_id | the unique id for device | generated from Settings.System.getDeviceId() or Settings.Secure.ANDROID_ID). If Android ID is null or "", we will use UUID instead. | distinguish different devices context.getContentResolver(). |

| Attribute name | Description | It is generated... | It is used to or for... |
|----------------|--|---|-----------------------------------|
| event_type | event name | set by developer or SDK | distinguish different event types |
| event_id | the unique id for event | generated from <code>UUID.randomUUID().toString()</code> when the event is created | distinguish each event |
| timestamp | event create timestamp | generated from <code>System.currentTimeMillis()</code> when the event is created | data analysis needs |
| platform | the platform name | for Android device, it is always "Android" | data analysis needs |
| os_version | the platform version code | generated from <code>Build.VERSION.RELEASE</code> | data analysis needs |
| make | the manufacturer of the device | generated from <code>Build.MANUFACTURER</code> | data analysis needs |
| brand | the brand of the device | generated from <code>Build.BRAND</code> | data analysis needs |
| model | the model of the device | generated from <code>Build.MODEL</code> | data analysis needs |
| carrier | the device network operator name | generated from <code>TelephonyManager.getNetworkOperatorName()</code> , the default is "UNKNOWN" | data analysis needs |
| network_type | the current device network type | generated from <code>android.net.ConnectivityManager</code> , it can be "Mobile", "WIFI" or "UNKNOWN" | data analysis needs |
| screen_height | the absolute height of the available display size in pixels | generated from <code>applicationContext.getResources().displayMetrics.height</code> | data analysis needs |
| screen_width | the absolute width of the available display size in pixels | generated from <code>applicationContext.getResources().displayMetrics.width</code> | data analysis needs |
| zone_offset | the device raw offset from GMT in milliseconds | generated from <code>java.util.Calendar.get(Calendar.ZONE_OFFSET)</code> | data analysis needs |
| locale | the default locale (language, country and variant) for this device of the Java Virtual Machine | generated from <code>java.util.Locale.getDefault()</code> | data analysis needs |

| Attribute name | Description | It is generated... | It is used to or for... |
|------------------|-------------------------------------|---|-------------------------|
| system_language | the device language code | generated from <code>java.util.Locale.getLanguage()</code> , and its default is value "UNKNOWN" | data analysis needs |
| country_code | country/region code for this device | generated from <code>java.util.Locale.getCountry()</code> and its default value is "UNKNOWN" | data analysis needs |
| sdk_version | clickstream SDK version | generated from <code>BuildConfig.VERSION_NAME</code> | data analysis needs |
| sdk_name | clickstream SDK name | it is always "aws-solution-clickstream-sdk" | data analysis needs |
| app_version | the app version name of user's app | generated from <code>android.content.pm.PackageInfo.versionName</code> , and its default value is "UNKNOWN" | data analysis needs |
| app_package_name | the app package name of user's app | generated from <code>android.content.pm.PackageInfo.packageName</code> , and its default value is "UNKNOWN" | data analysis needs |
| app_title | the display name of user's app | generated from <code>android.content.pm.ApplicationLabel(appInfo)</code> | data analysis needs |

Reserved attributes

User attributes

| Attribute name | Description |
|-----------------------------|---|
| _user_id | Reserved for user id that is assigned by app |
| _user_ltv_revenue | Reserved for user lifetime value |
| _user_ltv_currency | Reserved for user lifetime value currency |
| _user_first_touch_timestamp | The time (in microseconds) when the user first opened the app or visited the site, and it is included in every event in user object |

Event attributes

| Attribute name | Description |
|------------------------|---|
| _traffic_source_medium | Reserved for traffic medium. Use this attribute to store the medium that acquired user when events were logged. |

| Attribute name | Description |
|--------------------------|---|
| _traffic_source_name | Reserved for traffic name. Use this attribute to store the marketing campaign that acquired user when events were logged. |
| _traffic_source_source | Reserved for traffic source. Name of the network source that acquired the user when the event were reported. |
| _channel | The channel for app was downloaded |
| _device_vendor_id | Vendor id of the device |
| _device_advertising_id | Advertising id of the device |
| _entrances | Added in _screen_view event. The first _screen_view event in a session has the value 1, and others 0. |
| _session_id | Added in all events. |
| _session_start_timestamp | Added in all events. |
| _session_duration | Added in all events. |
| _session_number | Added in all events. The initial value is 1, and the value is automatically incremented by user device. |

Swift SDK

Introduction

Clickstream Swift SDK can help you easily collect and report in-app events from iOS devices to AWS. As part of the solution Clickstream Analytics on AWS, this SDK provisions data pipeline to ingest and process event data into AWS services such as Amazon S3, and Amazon Redshift.

The SDK is based on the Amplify for Swift Core Library and developed according to the Amplify Swift SDK plug-in specification. In addition, the SDK is equipped with features that automatically collect common user events and attributes (for example, screen view, first open) to simplify data collection for users.

Platform Support

Clickstream Swift SDK supports iOS 13+.

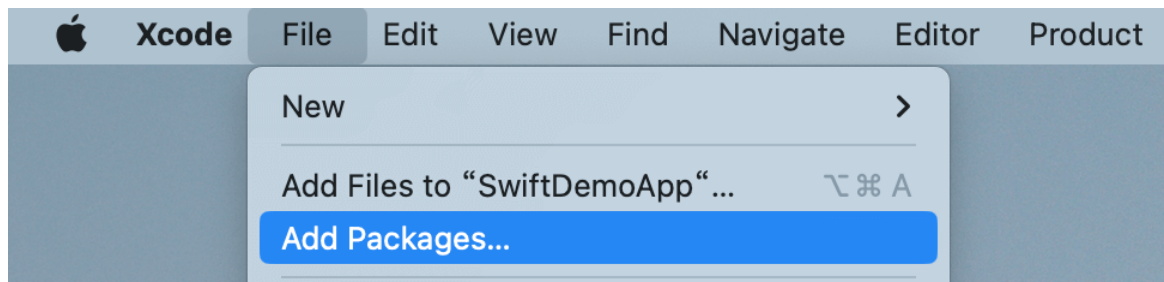
For more information, refer to [API Documentation](#).

Integrate the SDK

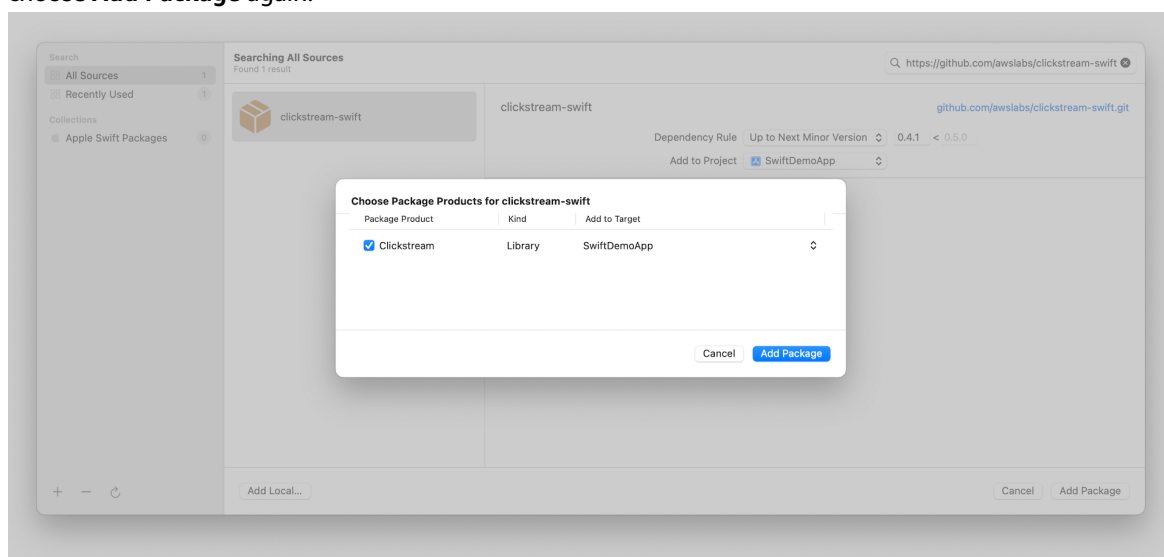
Clickstream requires Xcode 13.4 or higher to build.

1 Add Package

The solution uses Swift Package Manager to distribute Clickstream Swift SDK. Open your project in Xcode and select **File > Add Packages**.

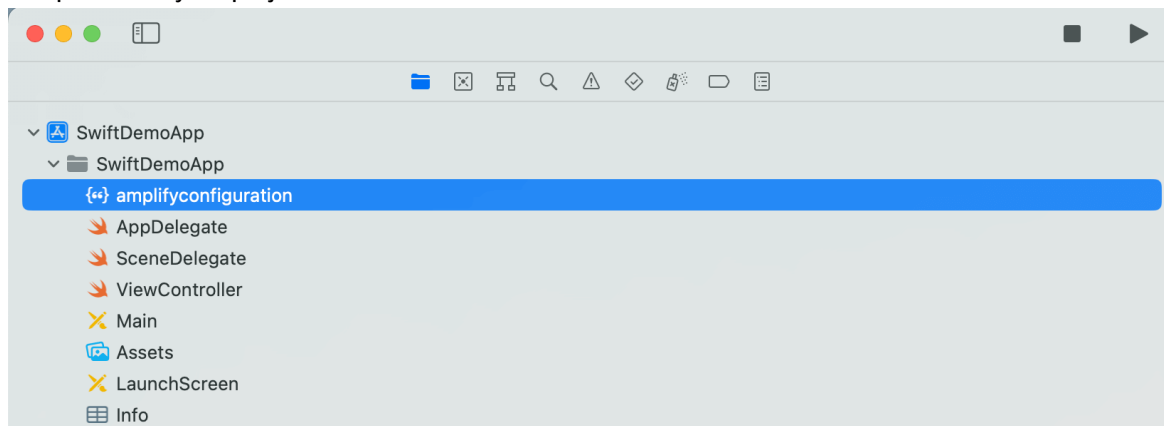


Enter the Clickstream Library for Swift GitHub repo URL (<https://github.com/awslabs/clickstream-swift>) into the search bar, and you'll see the Clickstream Library for Swift repository rules for which version of Clickstream you want Swift Package Manager to install. Choose **Up to Next Major Version**, then choose **Add Package**, make the Clickstream product checked as default, and choose **Add Package** again.



2 Parameter configuration

Download your `amplifyconfiguration.json` file from your Clickstream solution control plane. Copy and paste it to your project root folder:



The JSON file will be as follows:

```
{
  "analytics": {
    "plugins": {
      "awsClickstreamPlugin ": {
        "appId": "appId",
        "endpoint": "https://example.com/collect",
        "isCompressEvents": true,
        "autoFlushEventsInterval": 10000,
        "isTrackAppExceptionEvents": false
      }
    }
  }
}
```

In the file, your `appId` and `endpoint` are already configured. The explanation for each property is as follows:

- **appId:** the app id of your project in control plane
- **endpoint:** the endpoint url you will upload the event to AWS server
- **isCompressEvents:** whether to compress event content when uploading events, and the default value is true
- **autoFlushEventsInterval:** event sending interval, and the default value is 10s
- **isTrackAppExceptionEvents:** whether auto track exception event in app, and the default value is false

3 Initialize the SDK

Once you have configured the parameters, you need to initialize it in `AppDelegate's didFinishLaunchingWithOptions` lifecycle method to use the SDK.

```
import Clickstream
...
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
    [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    do {
        try ClickstreamAnalytics.initSDK()
    } catch {
        assertionFailure("Fail to initialize ClickstreamAnalytics: \(error)")
    }
    return true
}
```

4 Configure the SDK

After initializing the SDK, you can use the following code to customize it.

Important

This configuration will override the default configuration in `amplifyconfiguration.json` file.

```
import Clickstream

// config the sdk after initialize.
do {
    var configuration = try ClickstreamAnalytics.getClickstreamConfiguration()
    configuration.appId = "appId"
```

```
configuration.endpoint = "https://example.com/collect"
configuration.authCookie = "your authentication cookie"
configuration.sessionTimeoutDuration = 1800000
configuration.isTrackAppExceptionEvents = false
configuration.isLogEvents = true
configuration.isCompressEvents = true
configuration.isLogEvents = true
} catch {
    print("Failed to config ClickstreamAnalytics: \(error)")
}
```

5 Record event

Add the following code where you need to report an event.

```
import Clickstream

let attributes: ClickstreamAttribute = [
    "channel": "apple",
    "successful": true,
    "ProcessDuration": 12.33,
    "UserAge": 20,
]
ClickstreamAnalytics.recordEvent(eventName: "testEvent", attributes: attributes)

// for record an event directly
ClickstreamAnalytics.recordEvent(eventName: "button_click")
```

For more information, refer to [Github](#).

For **Objective-c** project, refer to [ClickstreamObjc API Reference](#).

Data format definition

Data type

Clickstream Swift SDK supports the following data types:

| Data type | Range | Example |
|-----------|--|---------------|
| Int | -214748364 ~ 2147483647 | 12 |
| Int64 | -9223372036854775808 ~ 9223372036854775807 | 26854775808 |
| Double | -2.22E-308 ~ 1.79E+308 | 3.14 |
| Boolean | true, false | true |
| String | 1024 characters maximum | "Clickstream" |

Naming rules

1. The event name and attribute name cannot start with a number, and only contain uppercase and lowercase letters, numbers, and underscores. In case of an invalid event name, it will throw precondition failure, In case of an invalid attribute name or user attribute name, it will discard the attribute and record error.

2. Do not use `_` as prefix in an event name or attribute name, because the `_` prefix is reserved for the solution.
3. The event name and attribute name are case sensitive, so `Add_to_cart` and `add_to_cart` will be recognized as two different event names.

Event and attribute limitation

To improve the efficiency of querying and analysis, we apply limitations to event data as follows:

| Item | Recommended | Maximum (hard limit) | Handle strategy if the limit is exceeded |
|---------------------------------|--------------------------|----------------------|---|
| Length of event name | less than 25 characters | 50 characters | throw <code>IllegalArgumentException</code> |
| Length of event attribute name | less than 25 characters | 50 characters | discard, log and record error |
| Length of event attribute value | less than 100 characters | 1024 characters | discard, log and record error |
| Event attribute per event | less than 50 attributes | 500 event attributes | discard, log and record error |
| Number of user attributes | less than 25 attributes | 100 user attributes | discard, log and record error |
| Length of user attribute name | less than 25 characters | 50 characters | discard, log and record error |
| Length of user attribute value | less than 50 characters | 256 characters | discard, log and record error |

Rules:

- The character limits are the same for single-width character languages (for example, English) and double-width character languages (for example, Chinese).
- The limitation of event attribute per event involves both common attributes and preset attributes.
- If the attribute or user attribute with the same name is added more than twice, the latest value will apply.

Preset events and attributes

Preset events

Automatically collected events:

| Event name | Triggered | Event attributes |
|-----------------------------|--|---|
| <code>_session_start</code> | when users app comes to foreground for the first time and there is no on-going session | <code>_session_id</code> <code>_session_start_timestamp</code> <code>_session_duration</code> |

| Event name | Triggered | Event attributes |
|------------------|---|--|
| _screen_view | when the activity callback <code>viewDidAppear()</code> method | <code>_screen_name</code> <code>_screen_id</code> <code>_previous_screen_name</code> <code>_previous_screen_id</code> <code>_entrances</code> <code>_engagement_time_msec</code> |
| _app_exception | when the app crashes | <code>_exception_message</code> <code>_exception_stack</code> |
| _app_update | when the app is updated to a new version and launched again | <code>_previous_app_version</code> |
| _first_open | when the user launches an app the first time after installation | |
| _os_update | when device operating system is updated to a new version | <code>_previous_os_version</code> |
| _user_engagement | when the app is in the foreground for at least one second | <code>_engagement_time_msec</code> |
| _profile_set | when the <code>addUserAttributes()</code> or <code>setUserId()</code> API is called | |

Session definition

In Clickstream Swift SDK, we do not limit the total time of a session. As long as the time between the next entry of the app and the last exit time is within the allowable timeout period, the current session is considered to be continuous.

- **_session_start:** When the app starts for the first time, or the app was launched to the foreground and the time between the last exit exceeded `session_time_out` period.
- **_session_duration:** We calculate the `_session_duration` by minus the current event create timestamp and the session's `_session_start_timestamp`. This attribute will be added in every event during the session.
- **session_time_out:** By default, it is 30 minutes, which can be customized through the configuration API.
- **_session_number:** The total number of sessions by distinct session id, and `_session_number` will be appear in every event's attribute object.

User engagement definition

In Clickstream Swift SDK, we define the `user_engagement` as the app which is in the foreground for at least one second.

- **when to send:** We send the event when the app navigates to background or navigates to another app.

- **engagement_time_msec:** We count the time from when the app comes in the foreground to when the app goes to the background.

Common attributes and reserved attributes

Sample event structure

```
{
  "app_id": "Shopping",
  "app_package_name": "com.compny.app",
  "app_title": "ModerneShopping",
  "app_version": "1.0",
  "brand": "apple",
  "carrier": "UNKNOWN",
  "country_code": "US",
  "device_id": "A536A563-65BD-49BE-A6EC-6F3CE7AC8FBE",
  "device_unique_id": "",
  "event_id": "91DA4BBE-933F-4DFA-A489-8AEFBC7A06D8",
  "event_type": "add_to_cart",
  "hashCode": "63D7991D",
  "locale": "en_US (current)",
  "make": "apple",
  "model": "iPhone 14 Pro",
  "network_type": "WIFI",
  "os_version": "16.4",
  "platform": "iOS",
  "screen_height": 2556,
  "screen_width": 1179,
  "sdk_name": "aws-solution-clickstream-sdk",
  "sdk_version": "0.4.1",
  "system_language": "en",
  "timestamp": 1685082174195,
  "unique_id": "0E6614B7-2D2C-4774-AB2F-B0A9E6C3BFAC",
  "zone_offset": 28800000,
  "user": {
    "_user_city": {
      "set_timestamp": 1685006678437,
      "value": "Shanghai"
    },
    "_user_first_touch_timestamp": {
      "set_timestamp": 1685006678434,
      "value": 1685006678432
    },
    "_user_name": {
      "set_timestamp": 1685006678437,
      "value": "carl"
    }
  },
  "attributes": {
    "_session_duration": 15349,
    "_session_id": "0E6614B7-20230526-062238846",
    "_session_number": 3,
    "_session_start_timestamp": 1685082158847,
    "product_category": "men's clothing",
    "product_id": 1
  }
}
```

All user attributes will be stored in user object, and all custom and global attributes in attributes object.

Common attributes

| Attribute name | Description | It is generated ... | It is used to or for ... |
|------------------|---------------------------------------|--|--|
| hashCode | the AnalyticsEvent Object's hash code | generated from <code>String(format: "%08X", hasher.combine(event.json))</code> | distinguish different events |
| app_id | clickstream app id | generated when clickstream app create from solution control plane | identify the events for your apps |
| unique_id | the unique id for user | generated from <code>UUID().uuidString</code> during the SDK first initialization. It will be changed after user relogin to another user who never login, and when user relogin to the previous user in same device. The <code>unique_id</code> will reset to the previous user's <code>unique_id</code> . | identity different users and associate the behavior of logging in and not logging in |
| device_id | the unique id for device | generated from <code>UIDevice.current.identifierForVendor?.uuidString</code> <code>UUID().uuidString</code> . It will be changed after app is reinstalled. | distinguish different devices |
| device_unique_id | the device advertising id | generated from <code>ASIdentifierManager().advertisingIdentifier</code> | distinguish different devices |
| event_type | event name | set by user or SDK | distinguish different event types |
| event_id | the unique id for event | generated from <code>UUID().uuidString</code> when the event is created | distinguish each event |
| timestamp | event create timestamp | generated from <code>Date().timeIntervalSince1970 * 1000</code> when the event is created | data analysis needs |
| platform | the platform name | for iOS device, it is always "iOS" | data analysis needs |
| os_version | the iOS operating system version | generated from <code>UIDevice.current.systemVersion</code> | data analysis needs |

| Attribute name | Description | It is generated ... | It is used to or for ... |
|-----------------|--|---|--------------------------|
| make | the manufacturer of the device | for iOS device, it is always "apple" | data analysis needs |
| brand | the brand of the device | for iOS device, it is always "apple" | data analysis needs |
| model | the model of the device | generated from mapping of device identifier | data analysis needs |
| carrier | the device network operator name | generated from <code>CTTelephonyNetworkInfo().serviceSubscriberCellular</code> and its default value is "UNKNOWN" | data analysis needs |
| network_type | the current device network type | generated from <code>NWPathMonitor</code> , it can be "Mobile", "WIFI" or "UNKNOWN" | data analysis needs |
| screen_height | the absolute height of the available display size in pixels | generated from <code>UIScreen.main.bounds.size.height * UIScreen.main.scale</code> | data analysis needs |
| screen_width | the absolute width of the available display size in pixels | generated from <code>UIScreen.main.bounds.size.width * UIScreen.main.scale</code> | data analysis needs |
| zone_offset | the device raw offset from GMT in milliseconds | generated from <code>TimeZone.current.secondsFromGMT()*1000</code> | data analysis needs |
| locale | the default locale (language, country and variant) for this device of the Java Virtual Machine | generated from <code>Locale.current</code> | data analysis needs |
| system_language | the device language code | generated from <code>Locale.current.languageCode</code> , and its default is value "UNKNOWN" | data analysis needs |
| country_code | country/region code for this device | generated from <code>Locale.current.regionCode</code> and its default value is "UNKNOWN" | data analysis needs |
| sdk_version | clickstream SDK version | generated from <code>PackageInfo.version</code> | data analysis needs |
| sdk_name | clickstream SDK name | it is always "aws-solution-clickstream-sdk" | data analysis needs |

| Attribute name | Description | It is generated ... | It is used to or for ... |
|------------------|------------------------------------|---|--------------------------|
| app_version | the app version name of user's app | generated from Bundle.main.infoDictionary["CFBundleShortVersionS "" | data analysis needs |
| app_package_name | the app package name of user's app | generated from Bundle.main.infoDictionary["CFBundleIdentifier"] "" | data analysis needs |
| app_title | the display name of user's app | generated from Bundle.main.infoDictionary["CFBundleName"] ?? "" | data analysis needs |

Reserved attributes

User attributes

| Attribute name | Description |
|----------------------------|---|
| _user_id | Reserved for user id that is assigned by app |
| _user_ltv_revenue | Reserved for user lifetime value |
| _user_ltv_currency | Reserved for user lifetime value currency |
| _user_first_touch_timestam | The time (in microseconds) when the user first opened the app or visited the site, and it is included in every event in user object |

Reserved attributes

| Attribute name | Description |
|------------------------|---|
| _traffic_source_medium | Reserved for traffic medium. Use this attribute to store the medium that acquired user when events were logged. |
| _traffic_source_name | Reserved for traffic name. Use this attribute to store the marketing campaign that acquired user when events were logged. |
| _traffic_source_source | Reserved for traffic source. Name of the network source that acquired the user when the event were reported. |
| _channel | The channel for app was downloaded. |
| _device_vendor_id | Vendor id of the device |
| _device_advertising_id | Advertising id of the device |
| _entrances | Added in _screen_view event. The first _screen_view event in a session has the value 1, and others 0. |

| Attribute name | Description |
|--------------------------|---|
| _session_id | Added in all events. |
| _session_start_timestamp | Added in all events. |
| _session_duration | Added in all events. |
| _session_number | Added in all events. The initial value is 1, and the value is automatically incremented by user device. |

Dashboards

Overview

Clickstream Analytics on AWS collects data from your websites and apps to create dashboards that derive insights. You can use dashboards to monitor traffic, investigate data, and understand your users and their activities.

The data will appear in the QuickSight dashboards after being processed by the data pipeline. Depending on your pipeline configuration, the time it takes varies for data to be available in your dashboard. For example, if you set the data processing interval to be 1 day, the dashboard will show data T+1 day (T as reporting date).

View dashboards

Use this procedure to find the dashboard for each application:

1. Go to Clickstream Analytics on AWS Management Console, in the Navigation Bar, choose "Project", then choose the project you want to view dashboards.
2. In the project detail page, choose pipeline ID or View Details, which will bring you to the pipeline detail page.
3. In the pipeline details page, select the Reporting tab. You will see the link to the dashboards created for your app.
4. Choose the dashboard link with the name of your app, which will bring you to the QuickSight dashboard.
5. Choose the dashboard with the name starting with Clickstream.

Reports

The dashboard contains a set of reports.

| Report name | What it is |
|-------------------------------------|--|
| Acquisition (p. 80) | Summarizes key metrics about new users, and provides detail view user profile |
| Engagement (p. 84) | Summarizes key metrics about user engagements and sessions |
| Activity (p. 90) | Summarizes key metrics about events user generates in the app, and provides detail view of event attributes |
| Retention (p. 95) | Summarizes key metrics about active users and user retentions |
| Device (p. 101) | Summarizes key metrics about the devices users are using to access your apps and websites, and provides detail view of each device |

| Report name | What it is |
|--|--|
| Path explorer (p. 106) | Provides charts for you to understand user journey in your apps and websites |

Custom report

If you want to investigate certain pieces of data further, you can write SQL to create views in Redshift or Athena, then add dataset into QuickSight to create visualization. Refer to [this example \(p. 109\)](#) to learn how to create a customize report with Redshift.

Acquisition report

You can use the User acquisition report to get insights into how new users find your website or app for the first time. This report also allows you view the detail user profile.

Note

This article describes the default report. You can customize the report by applying filters or comparisons or by changing the dimensions, metrics, or charts in QuickSight. For more information, refer to [Visualizing data in Amazon QuickSight](#).

View the report

1. Access the dashboard for your application. Refer to [Access dashboard \(p. 79\)](#).
2. In the dashboard, choose the sheet with name of Acquisition.

Where the data comes from

Acquisition report are created based on the QuickSight dataset of Clickstream_User_Acquisition, which connects to the clickstream-user-acquisition view in analytics engine (that is, Redshift or Athena). Below is the SQL command that generates the view.

Redshift SQL:

```
SELECT
    upid.*,
    (case when uid.user_id_count > 0 then 'Registered' else 'Non-registered' end) as
    is_registered from
    (SELECT
        user_pseudo_id
        , user_id
        , event_date as first_visit_date
        , app_info.install_source::varchar as first_visit_install_source
        , device.system_language::varchar as first_visit_device_language
        , platform as first_platform
        , geo.country::varchar as first_visit_country
        , geo.city::varchar as first_visit_city
        , (case when nullif(traffic_source.source::varchar, '') is null then '(direct)'
        else traffic_source.source::varchar end) as first_traffic_source
        , traffic_source.medium::varchar as first_traffic_source_medium
        , traffic_source.name::varchar as first_traffic_source_name
    FROM {{ shema }}.ods_events e
    where e.event_name in ('_first_open', '_first_visit')) upid left outer join
```

```
(select user_pseudo_id, count(distinct user_id) as user_id_count from
{{ schema }}.ods_events ods where event_name not in ('_first_open','_first_visit') group by
1 ) uid on upid.user_pseudo_id=uid.user_pseudo_id
;
```

Athena SQL:

```
with base as (
  select
    *
  from {{ database }}.{{ eventTable }}
  where partition_app = ?
    and partition_year >= ?
    and partition_month >= ?
    and partition_day >= ?
),
clickstream_user_dim_mv_1 as (
  SELECT
    user_pseudo_id
    , user_id
    , event_date as first_visit_date
    , app_info.install_source as first_visit_install_source
    , device.system_language as first_visit_device_language
    , platform as first_platform
    , geo.country as first_visit_country
    , geo.city as first_visit_city
    , (case when nullif(traffic_source.source,'') is null then '(direct)' else
traffic_source.source end) as first_traffic_source_source
    , traffic_source.medium as first_traffic_source_medium
    , traffic_source.name as first_traffic_source_name
  from base
  where event_name in ('_first_open','_first_visit')
),

clickstream_user_dim_mv_2 AS (
  select user_pseudo_id,
    count
    (
      distinct user_id
    ) as user_id_count
  from base ods
  where event_name not in
    (
      '_first_open',
      '_first_visit'
    ) group by 1
)

SELECT upid.*,
(
  case when uid.user_id_count>0 then 'Registered' else 'Non-registered' end
) as is_registered
from clickstream_user_dim_mv_1 as upid left outer join
clickstream_user_dim_mv_2 as uid on upid.user_pseudo_id=uid.user_pseudo_id
```

Dimensions and metrics

The report includes the following dimensions and metrics. You can add more dimensions or metrics by creating calculated field in QuickSight dataset. For more information, refer to [Adding calculated fields](#).

| Field | Type | Description | Data source |
|-----------------------------|-----------|--|-----------------------------|
| user_pseudo_id | Dimension | A SDK-generated unique ID for the user | Query from analytics engine |
| user_id | Dimension | The user ID set via the setUserId API in SDK | Query from analytics engine |
| first_visit_date | Dimension | The date that the user first visited your website or first opened the app | Query from analytics engine |
| first_install_source | Dimension | The installation source when user first opened your app. Blank for web | Query from analytics engine |
| first_visit_device_language | Dimension | The system language of the device user used when they first opened your app or first visited your website. | Query from analytics engine |
| first_visit_device_language | Dimension | The system language of the device user used when they first opened your app or first visited your website. | Query from analytics engine |
| first_platform | Dimension | The platform when user first visited your website or first opened your app | Query from analytics engine |
| first_visit_country | Dimension | The country where user first visited your website or first opened your app. | Query from analytics engine |
| first_visit_city | Dimension | The city where user first visited your website or first opened your app. | Query from analytics engine |
| custom_attr_key | Dimension | The name of the custom attribute key of the user. | Query from analytics engine |
| custom_attr_value | Dimension | The value of the custom attribute key of the user. | Query from analytics engine |
| is_registered | Dimension | Whether user had registered or not | Query from analytics engine |

Sample dashboard

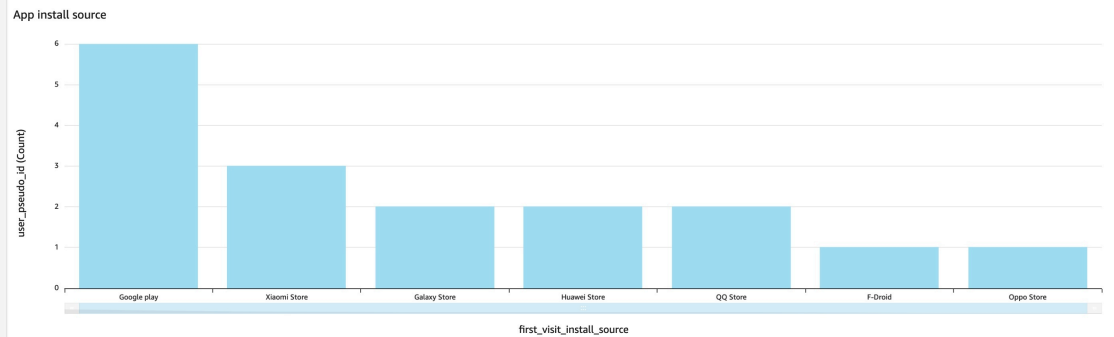
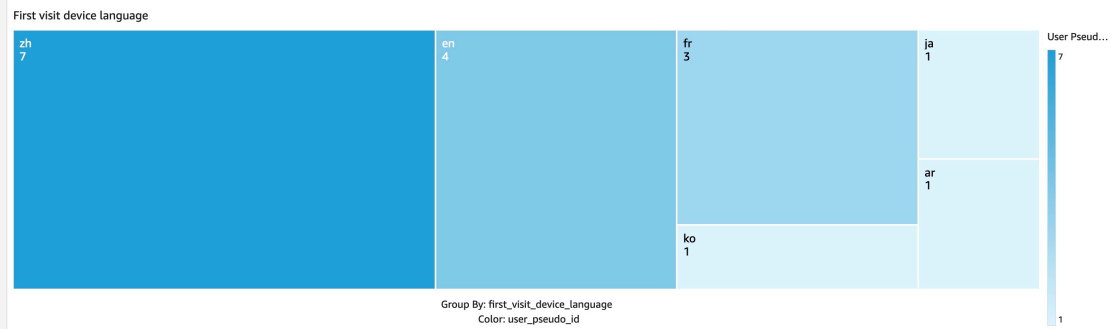
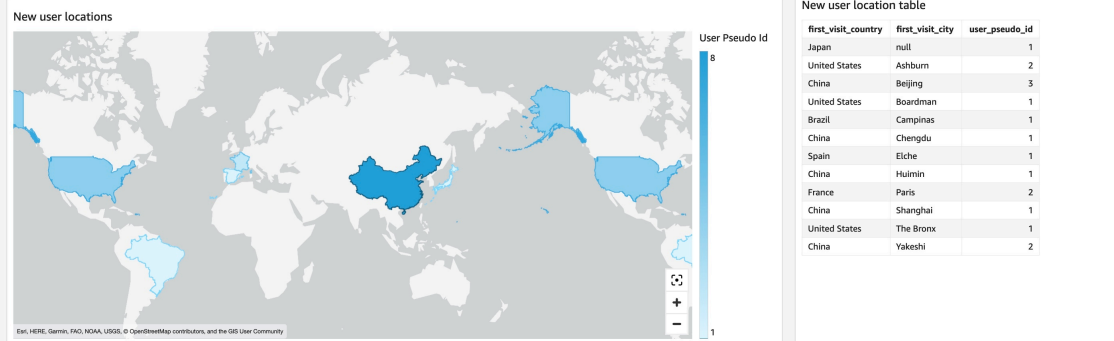
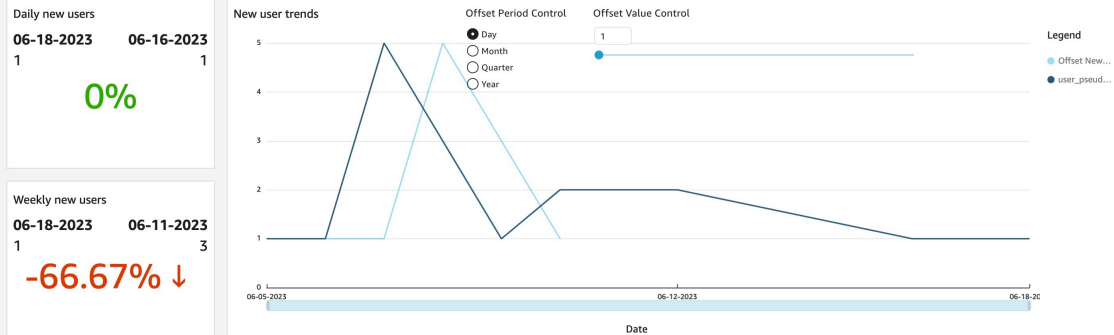
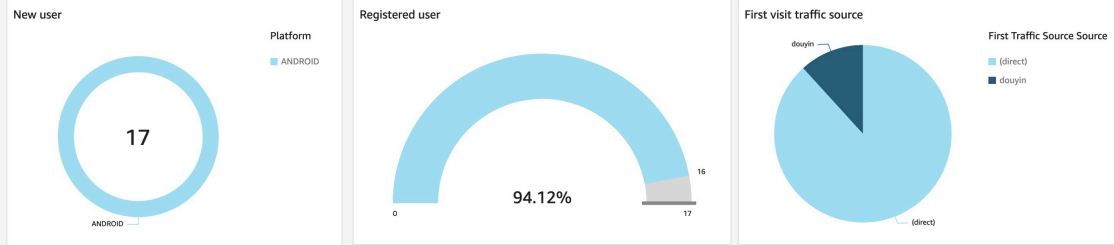
Below image is a sample dashboard for your reference.

Clickstream Analytics on AWS Implementation Guide

Sample dashboard

Acquisition report

User acquisition report provides insights into how new users find your website or app for the first time, and help you understand the user profiles and user growth trends. This report mainly based on the user "first_visit" event data in view of clickstream user dim.



user_pseudo_id equals All

first_traffic_source_source equals All

first_visit_device_language equals All

first_visit_city equals All

custom_attr_key equals All

custom_attr_value equals All

User details

A table contains all user detail information for user to search and filter, useful when you need to focus on one specific user.

| user_pseudo_id | is_registered | first_visit_date | first_visit_device_language | first_visit_install_source | first_traffic_source_source | first_tra... |
|----------------|---------------|---------------------|-----------------------------|----------------------------|-----------------------------|--------------|
| 83 | | 2023-06-18 00:00:00 | zh | Google play | | |

User attributes

Note that user attribute data is updated once per day by default, you might see some users do not have attributes.

| custom_attr_key | custom_attr_value | user_pseud... |
|-----------------|-------------------|---------------|
|-----------------|-------------------|---------------|

Engagement report

You can use the Engagement report to get insights into the engagement level of the users when using your websites and apps. This report measures user engagement by the sessions that users trigger and the web pages and app screens that users visit.

Note

This article describes the default report. You can customize the report by applying filters or comparisons or by changing the dimensions, metrics, or charts in QuickSight. For more information, refer to [Visualizing data in Amazon QuickSight](#).

View the report

1. Access the dashboard for your application. Refer to [Access dashboard \(p. 79\)](#).
2. In the dashboard, choose the sheet with name of Engagement.

Where the data comes from

Engagement report are created based on the QuickSight dataset of Clickstream_Engagement, which connects to the clickstream-session-view view in analytics engines (that is, Redshift or Athena). Below is the SQL command that generates the view.

Redshift SQL:

```
with session as (
  select
    es.session_id::varchar
    ,user_pseudo_id
    ,platform
    -- to_date(event_date, 'YYYYMMDD') as event_date,
    ,max(session_duration) as session_duration
    ,(case when (max(session_duration)>10000 or sum(view) >1) then 1 else 0 end) as
engaged_session
    ,(case when (max(session_duration)>10000 or sum(view) >1) then 0 else 1 end) as
bounced_session
    ,min(session_st) as session_start_timestamp
    ,sum(view) as session_views
    ,sum(engagement_time) as session_engagement_time
  from (
    select
      user_pseudo_id
      ,event_id
      -- ,event_name
      ,platform
      ,(select
        ep.value.string_value as value
        from {{ schema }}.ods_events e, e.event_params ep
        where
          ep.key = '_session_id' and e.event_id = ods.event_id) session_id
      ,(select
        ep.value.int_value::int as value
        from {{ schema }}.ods_events e, e.event_params ep
        where
          ep.key = '_session_duration' and e.event_id = ods.event_id)
session_duration
      ,(select
        ep.value.int_value::bigint as value
        from {{ schema }}.ods_events e, e.event_params ep
        where
```

```
        ep.key = '_session_start_timestamp' and e.event_id = ods.event_id)
session_st
    ,(select
        ep.value.int_value::int as value
        from {{ schema }}.ods_events e, e.event_params ep
        where
            ep.key = '_engagement_time_msec' and event_name = '_user_engagement' and
            e.event_id = ods.event_id) as engagement_time
    ,(case when event_name in ('_screen_view', '_page_view') then 1 else 0 end) as
view
    from {{ schema }}.ods_events ods
    ) es group by 1,2,3), session_f_l_sv as
    ( select session_id, first_sv_event_id, last_sv_event_id, count(event_id) from (
        select
            session_id::varchar
            , event_id
            -- ,event_name
            -- , event_timestamp
            ,first_value(event_id) over(partition by session_id order by event_timestamp asc
rows between unbounded preceding and unbounded following) as first_sv_event_id,
            last_value(event_id) over(partition by session_id order by event_timestamp asc rows
between unbounded preceding and unbounded following) as last_sv_event_id
        from (
            select e.event_name, e.event_id, e.event_timestamp, ep.value.string_value
as session_id
            from {{ schema }}.ods_events e, e.event_params ep
            where e.event_name in ('_screen_view','_page_view')
            and ep.key = '_session_id') ) group by 1,2,3
    ), session_f_sv_view as (
        select * from session_f_l_sv left outer join
        (select e.event_id as event_id, ep.value.string_value as first_sv_view
        from {{ schema }}.ods_events e, e.event_params ep
        where ep.key in ('_screen_name','_page_title')) t on
session_f_l_sv.first_sv_event_id=t.event_id
    ), session_f_l_sv_view as (
        select * from session_f_sv_view left outer join
        (select e.event_id as event_id, ep.value.string_value as last_sv_view
        from {{ schema }}.ods_events e, e.event_params ep
        where ep.key in ('_screen_name','_page_title')) t on
session_f_sv_view.last_sv_event_id=t.event_id
    )
    select
        session.session_id
        ,user_pseudo_id
        ,platform
        ,session_duration
        ,session_views
        ,engaged_session
        ,bounced_session
        ,session_start_timestamp
        ,session_engagement_time
        ,DATE_TRUNC('day', TIMESTAMP 'epoch' + session_start_timestamp/1000 * INTERVAL '1
second') as session_date
        ,DATE_TRUNC('hour', TIMESTAMP 'epoch' + session_start_timestamp/1000 * INTERVAL '1
second') as session_date_hour
        ,first_sv_view::varchar as entry_view
        ,last_sv_view::varchar as exit_view
    from session left outer join session_f_l_sv_view on session.session_id =
session_f_l_sv_view.session_id;
```

Athena SQL:

```
with base as (
    select
        *
```

```

from {{ database }}.{{ eventTable }}
where partition_app = ?
and partition_year >= ?
and partition_month >= ?
and partition_day >= ?
),
clickstream_session_mv_1 as (
SELECT
    es.session_id
    ,user_pseudo_id
    ,platform
    ,max(session_duration) as session_duration
    ,(case when (max(session_duration)>10000 or sum(view) >1) then 1 else 0 end) as
engaged_session
    ,(case when (max(session_duration)>10000 or sum(view) >1) then 0 else 1 end) as
bounced_session
    ,min(session_st) as session_start_timestamp
    ,sum(view) as session_views
    ,sum(engagement_time) as session_engagement_time
FROM
(SELECT
    user_pseudo_id
    ,event_id
    ,platform
    ,(select
        ep.value.string_value as value
        from base e cross join unnest(event_params) as t(ep)
        where
            ep.key = '_session_id' and e.event_id = ods.event_id) session_id
    ,cast((select
        ep.value.int_value as value
        from base e cross join unnest(event_params) as t(ep)
        where
            ep.key = '_session_duration' and e.event_id = ods.event_id) as integer)
session_duration
    ,cast((select
        ep.value.int_value as value
        from base e cross join unnest(event_params) as t(ep)
        where
            ep.key = '_session_start_timestamp' and e.event_id = ods.event_id) as
bigint) session_st
    ,cast((select
        ep.value.int_value as value
        from base e cross join unnest(event_params) as t(ep)
        where
            ep.key = '_engagement_time_msec' and event_name = '_user_engagement' and
e.event_id = ods.event_id) as integer) as engagement_time
    ,cast((case when event_name in ('_screen_view', '_page_view') then 1 else 0
end) as integer) as view
    FROM base ods
    ) AS es
GROUP BY 1,2,3
),
clickstream_session_mv_2 as (
select session_id, first_sv_event_id, last_sv_event_id, count(event_id) from (
    select
        session_id
        , event_id
        ,first_value(event_id) over(partition by session_id order by event_timestamp asc
rows between unbounded preceding and unbounded following) as first_sv_event_id,
        last_value(event_id) over(partition by session_id order by event_timestamp asc rows
between unbounded preceding and unbounded following) as last_sv_event_id
    from (
        select e.event_name, e.event_id, e.event_timestamp, ep.value.string_value
as session_id
        from base e cross join unnest(event_params) as t(ep)

```

```

        where e.event_name in ('_screen_view','_page_view')
        and ep.key = '_session_id')
    ) group by 1,2,3
    ),
    session_f_sv_view as (
        select * from clickstream_session_mv_2 as session_f_l_sv left outer join
        (select e.event_id as event_id, ep.value.string_value as first_sv_view
        from base e cross join unnest(event_params) as t(ep)
        where ep.key in ('_screen_name','_page_title')) t on
        session_f_l_sv.first_sv_event_id=t.event_id
    ),
    session_f_l_sv_view as (
        select * from session_f_sv_view left outer join
        (select e.event_id as event_id, ep.value.string_value as last_sv_view
        from base e cross join unnest(event_params) as t(ep)
        where ep.key in ('_screen_name','_page_title')) t on
        session_f_sv_view.last_sv_event_id=t.event_id
    )
    select
        session.session_id
        ,user_pseudo_id
        ,platform
        ,session_duration
        ,session_views
        ,engaged_session
        ,bounced_session
        ,session_start_timestamp
        ,session_engagement_time
        ,DATE_TRUNC('day', from_unixtime(session_start_timestamp/1000)) as session_date
        ,DATE_TRUNC('hour', from_unixtime(session_start_timestamp/1000)) as
        session_date_hour
        ,first_sv_view as entry_view
        ,last_sv_view as exit_view
    from clickstream_session_mv_1 as session left outer join
    session_f_l_sv_view on session.session_id = session_f_l_sv_view.session_id

```

Dimensions and metrics

The report includes the following dimensions and metrics. You can add more dimensions or metrics by creating calculated field in QuickSight dataset. For more information, refer to [Adding calculated fields](#).

| Field | Type | Description | Data source |
|------------------|-----------|--|-----------------------------|
| session_id | Dimension | A SDK-generated unique ID for the session user triggered when using your websites and apps | Query from analytics engine |
| user_pseudo_id | Dimension | A SDK-generated unique id for the user | Query from analytics engine |
| platform | Dimension | The platform user used during the session | Query from analytics engine |
| session_duration | Dimension | The length of the session in millisecond | Query from analytics engine |
| session_views | Metric | Number of screen view or page view within the session | Query from analytics engine |

| Field | Type | Description | Data source |
|-------------------------------------|-----------|---|--------------------------------|
| engaged_session | Dimension | Whether the session is engaged or not. Engaged session is defined as if the session last more than 10 seconds or have two or more screen views page views | Query from analytics engine |
| session_start_timestamp | Dimension | The start timestamp of the session | Query from analytics engine |
| session_engagement_time | Dimension | The total engagement time of the session in millisecond | Query from analytics engine |
| entry_view | Dimension | The screen name or page title of the first screen or page user viewed in the session | Query from analytics engine |
| exit_view | Dimension | The screen name or page title of the last screen or page user viewed in the session | Query from analytics engine |
| Average engaged session per user | Metric | Average number of session per user in the selected time period | Calculated field in QuickSight |
| Average engagement time per session | Metric | Average engagement time per session in the selected time period | Calculated field in QuickSight |
| Average engagement time per user | Metric | Average engagement time per user in the selected time period | Calculated field in QuickSight |
| Average screen view per user | Metric | Average number of screen views per user in the selected time period | Calculated field in QuickSight |

Sample dashboard

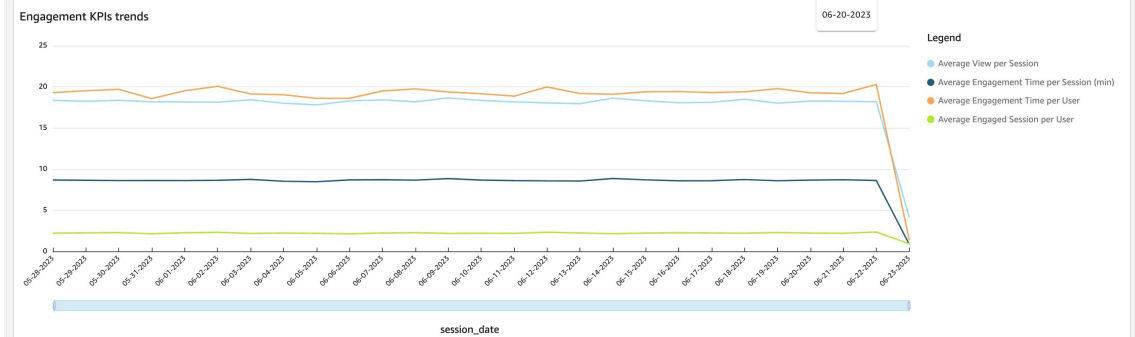
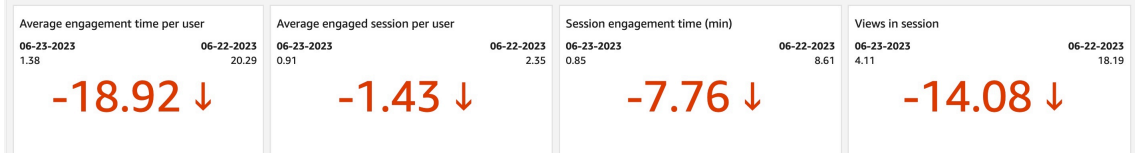
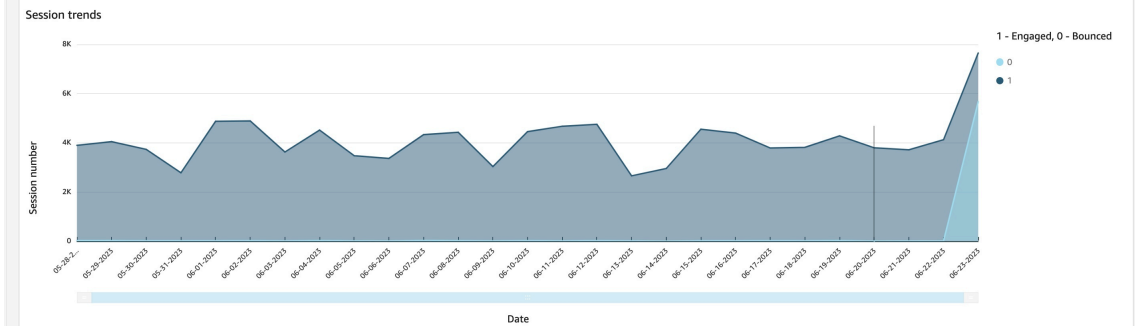
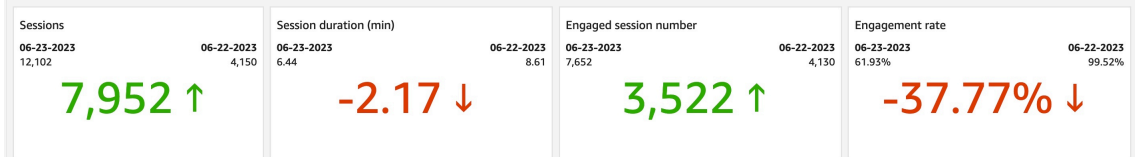
Below image is a sample dashboard for your reference.

Clickstream Analytics on AWS Implementation Guide

Sample dashboard

Engagement report

User engagement report provides metrics based on sessions and views that your users trigger when they are using with your apps to help you understand their engagement level. This report mainly based on the the session data that aggregated at view of clickstream session view.

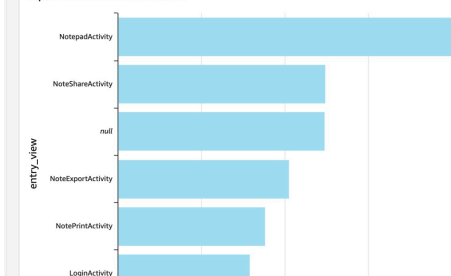


Engagement KPI table

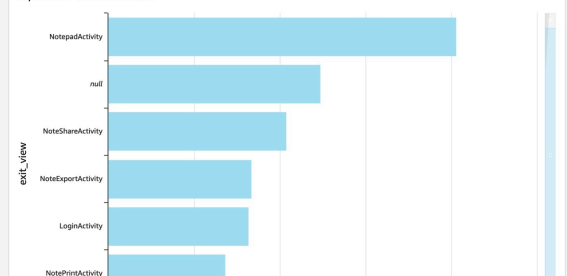
| Date | Session number | Engagement rate | Average Engaged Session per User | Average Engagement Time per User | Average Engagement Time per Session (min) | Average View per Session |
|------------|----------------|-----------------|----------------------------------|----------------------------------|---|--------------------------|
| 05-28-2023 | 3,931 | 99.46% | 2.21 | 19.31 | 8.67 | 18.36 |
| 05-29-2023 | 4,081 | 99.44% | 2.24 | 19.53 | 8.64 | 18.25 |
| 05-30-2023 | 3,760 | 99.55% | 2.28 | 19.7 | 8.61 | 18.36 |
| 05-31-2023 | 2,809 | 99.32% | 2.14 | 18.57 | 8.62 | 18.18 |
| 06-01-2023 | 4,902 | 99.59% | 2.26 | 19.53 | 8.61 | 18.16 |
| 06-02-2023 | 4,919 | 99.53% | 2.31 | 20.06 | 8.64 | 18.13 |
| 06-03-2023 | 3,649 | 99.53% | 2.18 | 19.14 | 8.75 | 18.42 |
| 06-04-2023 | 4,552 | 99.41% | 2.22 | 19.04 | 8.52 | 18.01 |
| 06-05-2023 | 3,496 | 99.51% | 2.19 | 18.61 | 8.46 | 17.81 |
| 06-06-2023 | 3,384 | 99.59% | 2.13 | 18.61 | 8.68 | 18.29 |
| 06-07-2023 | 4,368 | 99.38% | 2.22 | 19.5 | 8.71 | 18.42 |
| 06-08-2023 | 4,458 | 99.46% | 2.27 | 19.76 | 8.66 | 18.19 |
| 06-09-2023 | 3,057 | 99.41% | 2.18 | 19.37 | 8.84 | 18.67 |
| 06-10-2023 | 4,486 | 99.4% | 2.2 | 19.16 | 8.66 | 18.35 |
| 06-11-2023 | 4,694 | 99.64% | 2.19 | 18.88 | 8.6 | 18.17 |

Maximum
Highest day is Jun 23, 2023 with total count of records of 13,328

Top screen as session entrance



Top screen as session exits



Activity report

You can use the Activity report to get insights into the activities the users performed when using your websites and apps. This report measures user activity by the events that users triggered, and let you view the detail attributes of the events.

Note

This article describes the default report. You can customize the report by applying filters or comparisons or by changing the dimensions, metrics, or charts in QuickSight. For more information, refer to [Visualizing data in Amazon QuickSight](#).

View the report

1. Access the dashboard for your application. Refer to [Access dashboard \(p. 79\)](#).
2. In the dashboard, choose the sheet with name of **Activity**.

Where the data comes from

Activity report are created based on the following QuickSight datasets:

- **Clickstream_Events** that connects to the clickstream-ods-events view in analytics engines (that is, Redshift or Athena)
- **Clickstream_Event_Attributes** that connects to the clickstream-event-parameter view in analytics engines

Below is the SQL command that generates the related views.

Redshift SQL:

```
-- clickstream-ods-events view
select
    event_date
    , event_name as event_name
    , event_id as event_id
    , event_bundle_sequence_id::bigint as event_bundle_sequence_id
    , event_previous_timestamp::bigint as event_previous_timestamp
    , event_server_timestamp_offset::bigint as event_server_timestamp_offset
    , event_timestamp as event_timestamp
    , ingest_timestamp as ingest_timestamp
    , event_value_in_usd as event_value_in_usd
    , app_info.app_id::varchar as app_info_app_id
    , app_info.id::varchar as app_info_package_id
    , app_info.install_source::varchar as app_info_install_source
    , app_info.version::varchar as app_info_version
    , device.mobile_brand_name::varchar as device_mobile_brand_name
    , device.mobile_model_name::varchar as device_mobile_model_name
    , device.manufacturer::varchar as device_manufacturer
    , device.screen_width::bigint as device_screen_width
    , device.screen_height::bigint as device_screen_height
    , device.carrier::varchar as device_carrier
    , device.network_type::varchar as device_network_type
    , device.operating_system::varchar as device_operating_system
    , device.operating_system_version::varchar as device_operating_system_version
    , device.ua_browser::varchar
    , device.ua_browser_version::varchar
    , device.ua_os::varchar
    , device.ua_os_version::varchar
```

```
, device.ua_device::varchar
, device.ua_device_category::varchar
, device.system_language::varchar as device_system_language
, device.time_zone_offset_seconds::bigint as device_time_zone_offset_seconds
, geo.continent::varchar as geo_continent
, geo.country::varchar as geo_country
, geo.city::varchar as geo_city
, geo.metro::varchar as geo_metro
, geo.region::varchar as geo_region
, geo.sub_continent::varchar as geo_sub_continent
, geo.locale::varchar as geo_locale
, platform as platform
, project_id as project_id
, traffic_source.name::varchar as traffic_source_name
, traffic_source.medium::varchar as traffic_source_medium
, traffic_source.source::varchar as traffic_source_source
, user_first_touch_timestamp as user_first_touch_timestamp
, user_id as user_id
, user_pseudo_id
from {{ schema }}.ods_events;

-- event parameter view
SELECT
    e.event_id,
    e.event_name,
    e.event_date,
    ep.key::varchar as event_parameter_key,
    coalesce (ep.value.string_value
        , ep.value.int_value
        , ep.value.float_value
        , ep.value.double_value)::varchar as event_parameter_value
FROM {{ schema }}.ods_events e, e.event_params as ep
```

Athena SQL:

```
-- clickstream-ods-events query
select
    event_date
    ,event_name as event_name
    ,event_id as event_id
    ,event_bundle_sequence_id as event_bundle_sequence_id
    ,event_previous_timestamp as event_previous_timestamp
    ,event_server_timestamp_offset as event_server_timestamp_offset
    ,event_timestamp as event_timestamp
    ,ingest_timestamp as ingest_timestamp
    ,event_value_in_usd as event_value_in_usd
    ,app_info.app_id as app_info_app_id
    ,app_info.id as app_info_package_id
    ,app_info.install_source as app_info_install_source
    ,app_info.version as app_info_version
    ,device.mobile_brand_name as device_mobile_brand_name
    ,device.mobile_model_name as device_mobile_model_name
    ,device.manufacturer as device_manufacturer
    ,device.screen_width as device_screen_width
    ,device.screen_height as device_screen_height
    ,device.carrier as device_carrier
    ,device.network_type as device_network_type
    ,device.operating_system as device_operating_system
    ,device.operating_system_version as device_operating_system_version
    ,device.ua_browser
    ,device.ua_browser_version
    ,device.ua_os
    ,device.ua_os_version
    ,device.ua_device
```



```
,device.ua_device_category
,device.system_language as device_system_language
,device.time_zone_offset_seconds as device_time_zone_offset_seconds
,geo.continent as geo_continent
,geo.country as geo_country
,geo.city as geo_city
,geo.metro as geo_metro
,geo.region as geo_region
,geo.sub_continent as geo_sub_continent
,geo.locale as geo_locale
,platform as platform
,project_id as project_id
,traffic_source.name as traffic_source_name
,traffic_source.medium as traffic_source_medium
,traffic_source.source as traffic_source_source
,user_first_touch_timestamp as user_first_touch_timestamp
,user_id as user_id
,user_pseudo_id
from {{ database }}.{{ eventTable }}
where partition_app = ?
and partition_year >= ?
and partition_month >= ?
and partition_day >= ?

-- clickstream-ods-event-parameter query
select
event_id,
event_name,
event_date,
params.key as event_parameter_key,
coalesce (nullif(params.value.string_value, '')
,nullif(cast(params.value.int_value as varchar), ''))
,nullif(cast(params.value.float_value as varchar),'')
,nullif(cast(params.value.double_value as varchar),'')) as event_parameter_value
from {{ database }}.{{ eventTable }} cross join unnest(event_params) as t(params)
where partition_app = ?
and partition_year >= ?
and partition_month >= ?
and partition_day >= ?
```

Dimensions and metrics

The report includes the following dimensions and metrics. You can add more dimensions or metrics by creating calculated field in QuickSight dataset. For more information, refer to [Adding calculated fields](#).

| Field | Type | Description | Data source |
|-----------------|-----------|--|--------------------------------|
| event_id | Dimension | A SDK-generated unique ID for the event user triggered when using your websites and apps | Query from analytics engine |
| event_name | Dimension | The name of the event | Query from analytics engine |
| platform | Dimension | The platform user used during the session | Query from analytics engine |
| Event User Type | Dimension | The type of user performed the event, | Calculated field in QuickSight |

| Field | Type | Description | Data source |
|--------------------------------------|-----------|--|--------------------------------|
| | | that is, new user or existing user | |
| event_date | Metric | The date when the event was logged (YYYYMMDD format in UTC). | Query from analytics engine |
| event_timestamp | Dimension | The time (in microseconds, UTC) when the event was logged on the client. | Query from analytics engine |
| app_info_version | Dimension | The version of the app or website when event was logged | Query from analytics engine |
| event_parameter_key | Dimension | The key of the event parameter | Query from analytics engine |
| event_parameter_value | Dimension | The value of the event parameter | Query from analytics engine |
| User activity number in last 7 days | Metrics | Number of events logged in last 7 days | Calculated field in QuickSight |
| User activity number in last 30 days | Metrics | Number of events logged in last 30 days | Calculated field in QuickSight |

Sample dashboard

Below image is a sample dashboard for your reference.

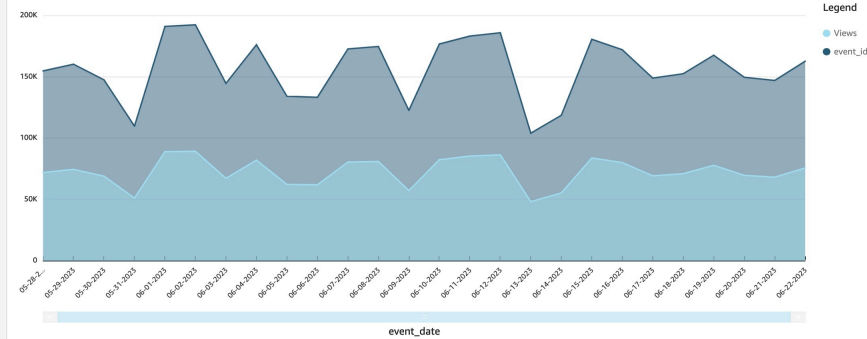
Clickstream Analytics on AWS Implementation Guide

Sample dashboard

Activity report

You can use the Activity report to get insights into the activities the users performed when using your websites and apps. This report measures user activity by the events that users triggered, and let you view the detail attributes of the events. This report mainly bases on data in clickstream ods events view and clickstream ods events parameters view.

Activity trends



Events

06-22-2023 162,671
06-21-2023 146,966

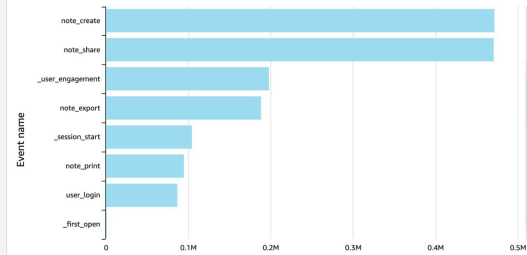
15,705 ↑

Views

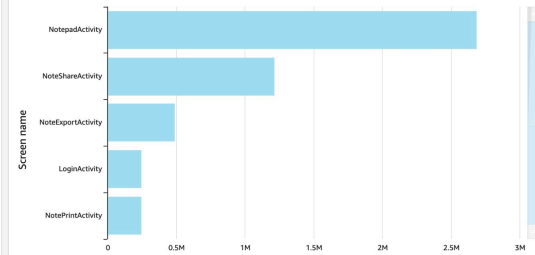
06-22-2023 75,614
06-21-2023 68,098

7,516 ↑

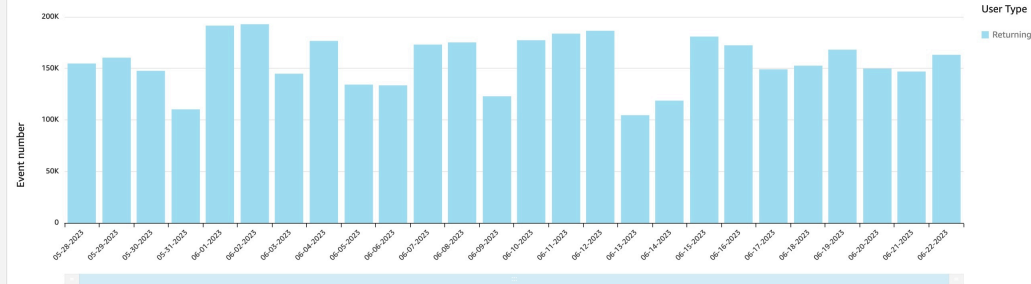
Top events



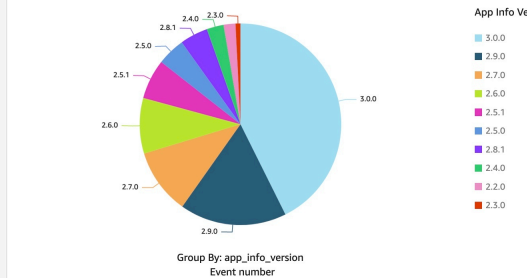
Top screen



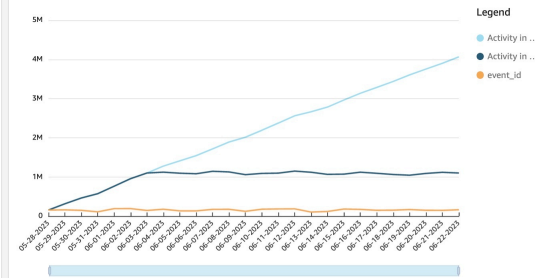
Event number by user type



Events by app version



User activity trends



| | | | | |
|-----------------------------------|--------------------------------|---------------------------------------|------------------------------|-------------------------------------|
| event_id equals All | event_name equals All | user_id equals All | user_pseudo_id equals All | event_parameter_key equals All |
| app_info_package_id equals All | app_info_version equals All | app_info_install_source equals All | User Type equals All | event_parameter_value equals All |

Event details

You can search individual event and view its dimension and parameter details in this table

| event_id | event_name | event_value_in_usd | user_pseudo_id | user_id |
|--------------------------------------|------------------|--------------------|--------------------------------------|-------------------------|
| 00007915-56bb-4242-a1c1-1175fecdb89b | note_create | null | 8926ca06-c8df-46f3-b570-7c0b7b7ae1d5 | 2c1624b8-5ffe-49ef-b33d |
| 00022861-32a0-4541-b98e-8f6f0b17a552 | note_share | null | bebb18d8-7a11-4231-8234-662cfc6c718 | 5ec9a63a-adab-4f70-9b9c |
| 000276cc-670b-4f9a-a11e-02995802b6d0 | note_create | null | 5ab6af1c-e48c-4ae7-8a7d-614d8ee5fb52 | d82de7c7-156d-4e03-add |
| 00035026-4de9-4523-ac2f-474aecdc52d8 | user_login | null | 841419eb-f78a-4864-b5aa-33da648dc27a | c17a73cd-245a-4dd9-997 |
| 00069795-30fb-4ebd-8e09-27dbfda5cdc | add_button_click | null | c4ae68d1-b094-4848-9a8c-b5c6c89aa43c | e4c2ae2f-d3ee-456c-9a12 |
| 0006b0c5-9f0c-404e-9cae-0c4b5e4e1e38 | add_button_click | null | 9008d21d-df01-4e46-a758-3d8accac6970 | e4d7d25b-326e-456e-09a |
| 0008a40c-c35d-4c5c-abc5-baa7303f12da | _screen_view | null | f58479e9-e8cc-4b4c-3f635449ec7c | 39321e85-92be-4bc9-879 |
| 000c1f0a-55cd-41e4-9f19-e6990927bec4 | add_button_click | null | c6647eca-4268-4eef-873f-97585687a8aa | null |
| 000e7a63-f177-4806-a5f7-4f1b21c9c772 | _screen_view | null | 69952b4b-7d25-4bb6-a0ff-3a67b8bbabbd | c8829f15-c12e-47eb-bd11 |
| 000f5d19-665a-49d4-d0a8-832bc9d79f02 | _screen_view | null | 3d20a305-5617-46aa-b050-2d569b08c983 | 622f2c5c-693d-44b8-bdc |
| 000f7a17-cd0f-405c-b902-37db1f31e1a6 | note_share | null | bebb18d8-7a11-4231-8234-662cfc6c718 | 5ec9a63a-adab-4f70-9b9c |
| 00114541-5cde-45f5-bc58-4ee2cc28a80a | _screen_view | null | 0ae3fad8-850b-4f70-878a-d198e4d8a3a | a8d8eae6-637e-46d3-a60 |
| 0012b556-754d-4e92-b43e-dfd19bfef9a3 | add_button_click | null | cbdff6f1-0fad-4dc7-b4bd-2ef5afa9c2de | e9ce707c-e25d-4e25-859- |
| 0013d3df-1244-454a-a57e-77eb56bf12a8 | note_export | null | 58073ee8-34c8-4e78-8ae6-a7dc382cb315 | c57d5b91-92bf-49c8-8660 |

Event attributes

| event_parameter_key | event_parameter_value |
|-----------------------|-----------------------|
| _previous_screen_id | null |
| _previous_screen_name | null |
| _session_id | null |
| _engagement_time_msec | -10000259 |
| _engagement_time_msec | -10000268 |
| _session_duration | -10000268 |
| _session_duration | -10000369 |
| _engagement_time_msec | -10000494 |
| _session_duration | -100006 |
| _session_duration | -10000792 |
| _engagement_time_msec | -10000919 |
| _session_duration | -10000921 |
| _engagement_time_msec | -10000941 |
| _session_duration | -10001371 |
| _engagement_time_msec | -10001544 |

Retention report

You can use the Retention report to get insights into how frequently and for how long users engage with your website or mobile app after their first visit. The report helps you understand how well your app is doing in terms of attracting users back after their first visit.

Note

This article describes the default report. You can customize the report by applying filters or comparisons or by changing the dimensions, metrics, or charts in QuickSight. For more information, refer to [Visualizing data in Amazon QuickSight](#).

View the report

1. Access the dashboard for your application. Refer to [Access dashboard \(p. 79\)](#).
2. In the dashboard, choose the sheet with name of Retention.

Where the data comes from

Retention report are created based on the following QuickSight dataset:

- **Clickstream_Lifecycle_Weekly**, which connects to the clickstream-lifecycle-weekly view in analytics engines (that is, Redshift or Athena).
- **Clickstream_Lifecycle_Daily**, which connects to the clickstream-lifecycle-daily view in analytics engines (that is, Redshift or Athena).
- **Clickstream_Events** that connects to the clickstream-ods-events view in analytics engines

Below is the SQL command that generates the view.

Redshift SQL:

```
-- clickstream-lifecycle-weekly view
with weekly_usage as (
select
    user_pseudo_id,
    -- datediff(week, '1970-01-01', dateadd(ms,event_timestamp, '1970-01-01')) as
    time_period
    DATE_TRUNC('week', dateadd(ms,event_timestamp, '1970-01-01')) as time_period
from {{ app_id }}.ods_events
where event_name = '_session_start' group by 1,2 order by 1,2),
-- detect if lag and lead exists
lag_lead as (
select user_pseudo_id, time_period,
    lag(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
    time_period),
    lead(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
    time_period)
from weekly_usage),
-- caculate lag and lead size
lag_lead_with_diffs as (
select user_pseudo_id, time_period, lag, lead,
    datediff(week,lag,time_period) lag_size,
    datediff(week,time_period,lead) lead_size
    -- time_period-lag lag_size,
    -- lead-time_period lead_size
from lag_lead),
-- case to lifecycle stage
```

```
calculated as (select time_period,
case when lag is null then '1-NEW'
      when lag_size = 1 then '2-ACTIVE'
      when lag_size > 1 then '3-RETURN'
end as this_week_value,
case when (lead_size > 1 OR lead_size IS NULL) then '0-CHURN'
      else NULL
end as next_week_churn,
count(distinct user_pseudo_id)
from lag_lead_with_diffs
group by 1,2,3)
select time_period, this_week_value, sum(count)
from calculated group by 1,2
union
select time_period+7, '0-CHURN', -1*sum(count)
from calculated where next_week_churn is not null group by 1,2
order by 1;

-- clickstream-lifecycle-daily view
with daily_usage as (
select
  user_pseudo_id,
  -- datediff(week, '1970-01-01', dateadd(ms,event_timestamp, '1970-01-01')) as
  time_period
  DATE_TRUNC('day', dateadd(ms,event_timestamp, '1970-01-01')) as time_period
from {{ schema }}.ods_events
where event_name = '_session_start' group by 1,2 order by 1,2),
-- detect if lag and lead exists
lag_lead as (
select user_pseudo_id, time_period,
      lag(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
      time_period),
      lead(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
      time_period)
from daily_usage),
-- caculate lag and lead size
lag_lead_with_diffs as (
select user_pseudo_id, time_period, lag, lead,
      datediff(day,lag,time_period) lag_size,
      datediff(day,time_period,lead) lead_size
      -- time_period-lag lag_size,
      -- lead-time_period lead_size
from lag_lead),
-- case to lifecycle stage
calculated as (select time_period,
case when lag is null then '1-NEW'
      when lag_size = 1 then '2-ACTIVE'
      when lag_size > 1 then '3-RETURN'
end as this_day_value,
case when (lead_size > 1 OR lead_size IS NULL) then '0-CHURN'
      else NULL
end as next_day_churn,
count(distinct user_pseudo_id)
from lag_lead_with_diffs
group by 1,2,3)
select time_period, this_day_value, sum(count)
from calculated group by 1,2
union
select time_period+1, '0-CHURN', -1*sum(count)
from calculated where next_day_churn is not null group by 1,2
order by 1;
```

Athena SQL:

```
-- clickstream-lifecycle-weekly-query
```

```
with weekly_usage as (
select
    user_pseudo_id,
    DATE_TRUNC('week', event_date) as time_period
from {{ database }}.{{ eventTable }}
where partition_app = ?
    and partition_year >= ?
    and partition_month >= ?
    and partition_day >= ?
    and event_name = '_session_start' group by 1,2 order by 1,2
),
lag_lead as (
select user_pseudo_id, time_period,
    lag(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
        time_period) as lag,
    lead(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
        time_period) as lead
from weekly_usage
),
lag_lead_with_diffs as (
select user_pseudo_id, time_period, lag, lead,
    date_diff('week',lag,time_period) lag_size,
    date_diff('week',time_period,lead) lead_size
from lag_lead
),
calculated as (
select time_period,
    case when lag is null then '1-NEW'
    when lag_size = 1 then '2-ACTIVE'
    when lag_size > 1 then '3-RETURN'
    end as this_week_value,
    case when (lead_size > 1 OR lead_size IS NULL) then '0-CHURN'
    else NULL
    end as next_week_churn,
    count(distinct user_pseudo_id) as cnt
from lag_lead_with_diffs
group by 1,2,3
)
select time_period, this_week_value, sum(cnt) as cnt from calculated group by 1,2
union
select date_add('day', 7, time_period), '0-CHURN', -1*sum(cnt) as cnt
from calculated
where next_week_churn is not null
group by 1,2

-- clickstream-lifecycle-daily-query
with daily_usage as (
select
    user_pseudo_id,
    DATE_TRUNC('day', event_date) as time_period
from {{ database }}.{{ eventTable }}
where partition_app = ?
    and partition_year >= ?
    and partition_month >= ?
    and partition_day >= ?
    and event_name = '_session_start' group by 1,2 order by 1,2
),
lag_lead as (
select user_pseudo_id, time_period,
    lag(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
        time_period) as lag,
    lead(time_period,1) over (partition by user_pseudo_id order by user_pseudo_id,
        time_period) as lead
from daily_usage
),
lag_lead_with_diffs as (
```

```
select user_pseudo_id, time_period, lag, lead,
       date_diff('day',lag,time_period) lag_size,
       date_diff('day',time_period,lead) lead_size
from lag_lead
),
calculated as (
select time_period,
       case when lag is null then '1-NEW'
       when lag_size = 1 then '2-ACTIVE'
       when lag_size > 1 then '3-RETURN'
       end as this_day_value,

       case when (lead_size > 1 OR lead_size IS NULL) then '0-CHURN'
       else NULL
       end as next_day_churn,
       count(distinct user_pseudo_id) as cnt
from lag_lead_with_diffs
group by 1,2,3
)
select time_period, this_day_value, sum(cnt) as cnt
from calculated group by 1,2
union
select date_add('day', 1, time_period) as time_period, '0-CHURN', -1*sum(cnt) as cnt
from calculated
where next_day_churn is not null
group by 1,2;
```

Dimensions and metrics

The report includes the following dimensions and metrics. You can add more dimensions or metrics by creating calculated field in QuickSight dataset. For more information, refer to [Adding calculated fields](#).

| Field | Type | Description | Data source |
|---------------------------|-----------|--|--------------------------------|
| Daily Active User (DAU) | Metric | Number of active users per date | QuickSight aggregation |
| Weekly Active User (WAU) | Metric | Number of active users in last 7 days | Calculated field in QuickSight |
| Monthly Active User (MAU) | Metric | Number of active users in last 30 days | Calculated field in QuickSight |
| user_pseudo_id | Dimension | A SDK-generated unique id for the user | Query from analytics engine |
| user_id | Dimension | The user ID set via the setUserId API in SDK | Query from analytics engine |
| DAU/WAU | Metric | DAU/WAU % for user stickiness | Calculated field in QuickSight |
| WAU/MAU | Metric | WAU/MAU % for user stickiness | Calculated field in QuickSight |
| DAU/MAU | Metric | DAU/MAU % for user stickiness | Calculated field in QuickSight |
| Event User Type | Dimension | The type of user performed the event, | Calculated field in QuickSight |

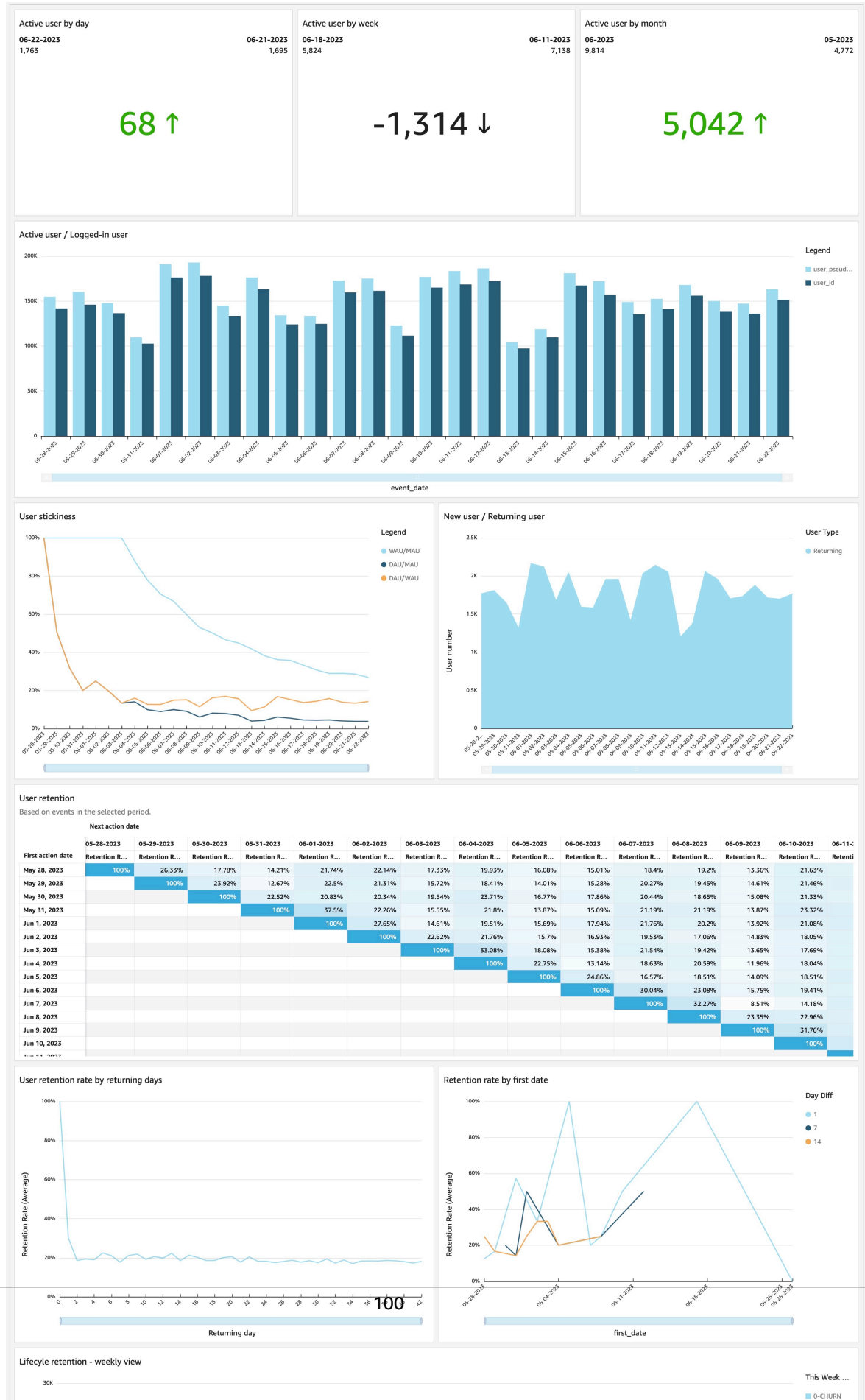
| Field | Type | Description | Data source |
|--------------------------|-----------|---|--------------------------------|
| | | that is, new user or existing user | |
| User first touch date | Metric | The first date that a user use your websites or apps | Calculated field in QuickSight |
| Retention rate | Metric | Distinct active users number / Distinct active user number by User first touch date | Calculated field in QuickSight |
| time_period | Dimension | The week or day for the user lifecycle | Query from analytics engine |
| this_week_value | Dimension | The user lifecycle stage, that is, New, Active, Return, and Churn | Query from analytics engine |
| this_day_value | Dimension | The user lifecycle stage, that is, New, Active, Return, and Churn | Query from analytics engine |
| Daily Active User (DAU) | Metric | Number of active users per date | QuickSight aggregation |
| Weekly Active User (WAU) | Metric | Number of active users in last 7 days | Calculated field in QuickSight |

Sample dashboard

Below image is a sample dashboard for your reference.

Clickstream Analytics on AWS Implementation Guide

Sample dashboard



Device report

You can use the Device report to get insights into the devices that users used when using your apps or websites. The report provides more information for your user profile.

Note

This article describes the default report. You can customize the report by applying filters or comparisons or by changing the dimensions, metrics, or charts in QuickSight. For more information, refer to [Visualizing data in Amazon QuickSight](#).

View the report

1. Access the dashboard for your application. Refer to [Access dashboard \(p. 79\)](#).
2. In the dashboard, choose the sheet with name of Device.

Where the data comes from

Device reports are created based on the following QuickSight dataset:

- **Clickstream_Device**, which connects to the clickstream-device view in analytics engines (that is, Redshift or Athena).

Below is the SQL command that generates the view.

Redshift SQL:

```
-- clickstream-device view
select
device.vendor_id::varchar as device_id
, to_date(event_date,'YYYYMMDD') as event_date
, device.mobile_brand_name::varchar
, device.mobile_model_name::varchar
, device.manufacturer::varchar
, device.screen_width::int
, device.screen_height::int
, device.carrier::varchar
, device.network_type::varchar
, device.operating_system::varchar
, device.operating_system_version::varchar
, device.ua_browser::varchar
, device.ua_browser_version::varchar
, device.ua_os::varchar
, device.ua_os_version::varchar
, device.ua_device::varchar
, device.ua_device_category::varchar
, device.system_language::varchar
, device.time_zone_offset_seconds::int
, device.advertising_id::varchar
, user_pseudo_id
, user_id
, count(event_id) as usage_num
--pleaes update the following schema name with your schema name
from {{ app_id }}.ods_events
group by
device_id
, event_date
, device.mobile_brand_name
```

```
, device.mobile_model_name
, device.manufacturer
, device.screen_width
, device.screen_height
, device.carrier
, device.network_type
, device.operating_system
, device.operating_system_version
, device.ua_browser
, device.ua_browser_version
, device.ua_os
, device.ua_os_version
, device.ua_device
, device.ua_device_category
, device.system_language
, device.time_zone_offset_seconds
, device.advertising_id
, user_pseudo_id
, user_id;
```

Athena SQL:

```
select
  device.vendor_id as device_id
  ,event_date
  ,device.mobile_brand_name
  ,device.mobile_model_name
  ,device.manufacturer
  ,device.screen_width
  ,device.screen_height
  ,device.carrier
  ,device.network_type
  ,device.operating_system
  ,device.operating_system_version
  ,device.ua_browser
  ,device.ua_browser_version
  ,device.ua_os
  ,device.ua_os_version
  ,device.ua_device
  ,device.ua_device_category
  ,device.system_language
  ,device.time_zone_offset_seconds
  ,device.advertising_id
  ,user_pseudo_id
  ,user_id
  ,count(event_id) as usage_num
from {{ database }}.{{ eventTable }}
where partition_app = ?
and partition_year >= ?
and partition_month >= ?
and partition_day >= ?
group by
device.vendor_id
,event_date
,device.mobile_brand_name
,device.mobile_model_name
,device.manufacturer
,device.screen_width
,device.screen_height
,device.carrier
,device.network_type
,device.operating_system
,device.operating_system_version
,device.ua_browser
,device.ua_browser_version
```

```
,device.ua_os
,device.ua_os_version
,device.ua_device
,device.ua_device_category
,device.system_language
,device.time_zone_offset_seconds
,device.advertising_id
,user_pseudo_id
,user_id
```

Dimensions and metrics

The report includes the following dimensions and metrics. You can add more dimensions or metrics by creating calculated field in QuickSight dataset. For more information, refer to [Adding calculated fields](#).

| Field | Type | Description | Data source |
|--------------------------|-----------|---|--------------------------------|
| device_id | Dimension | The unique ID for the device, please refer to SDK Manual for how the device id was obtained | QuickSight aggregation |
| user_pseudo_id | Dimension | A SDK-generated unique id for the user | Query from analytics engine |
| user_id | Dimension | The user ID set via the setUserId API in SDK | Query from analytics engine |
| event_date | Dimension | The event data of when the device information was logged | Query from analytics engine |
| mobile_brand_name | Dimension | The brand name for the device | Query from analytics engine |
| mobile_model_name | Dimension | The model name for the device | Query from analytics engine |
| manufacturer | Dimension | The manufacturer for the device | Query from analytics engine |
| network_type | Dimension | The network type when user logged the events | Query from analytics engine |
| operating_system | Dimension | The operating system of the device | Query from analytics engine |
| operating_system_version | Dimension | The operating system version of the device | Query from analytics engine |
| screen_height | Dimension | The screen height of the device | Query from analytics engine |
| screen_width | Dimension | The screen width of the device | Query from analytics engine |
| Screen Resolution | Dimension | The screen resolution (i.e., screen height x | Calculated field in QuickSight |

| Field | Type | Description | Data source |
|--------------------|-----------|---|-----------------------------|
| | | screen width) of the device | |
| system_language | Dimension | The system language of the solution | Query from analytics engine |
| us_browser | Dimension | The browser derived from user agent | Query from analytics engine |
| us_browser_version | Dimension | The browser version derived from user agent | Query from analytics engine |
| us_os | Dimension | The operating system derived from user agent | Query from analytics engine |
| us_device | Dimension | The device derived from user agent | Query from analytics engine |
| us_device_category | Dimension | The device category derived from user agent | Query from analytics engine |
| usage_num | Metric | Number of event that logged for the device ID | Query from analytics engine |

Sample dashboard

Below image is a sample dashboard for your reference.

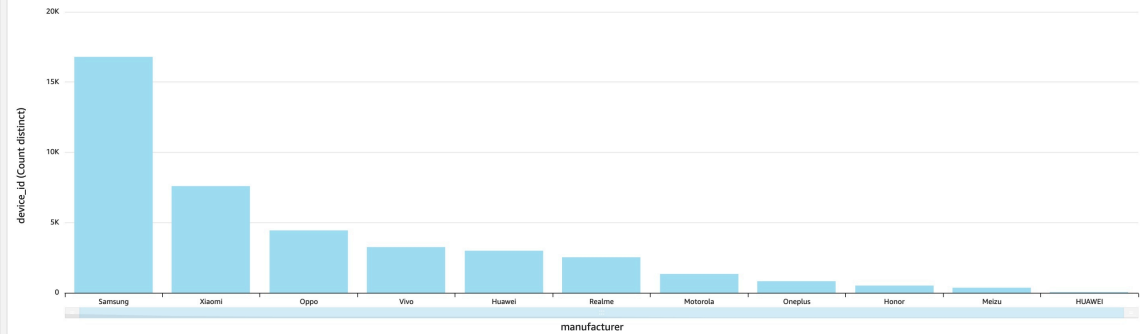
Clickstream Analytics on AWS Implementation Guide

Sample dashboard

Device report

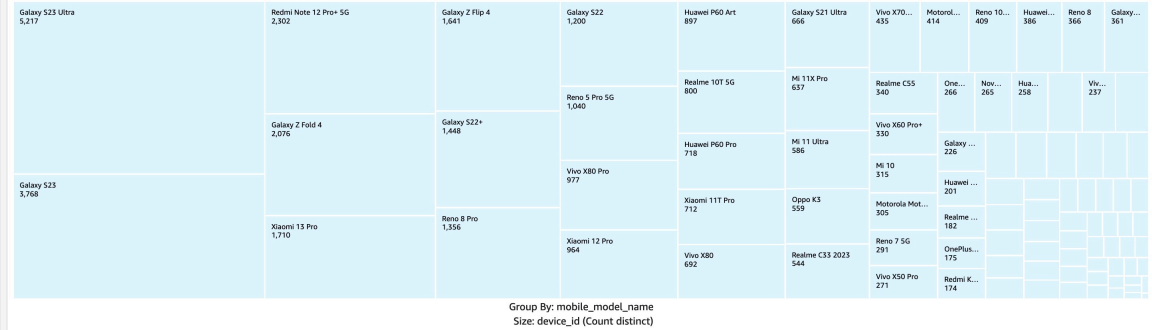
Device report helps you understand what device your users use to access your apps. This report mainly based on data from the view of clickstream_device_view.

Device manufacturer

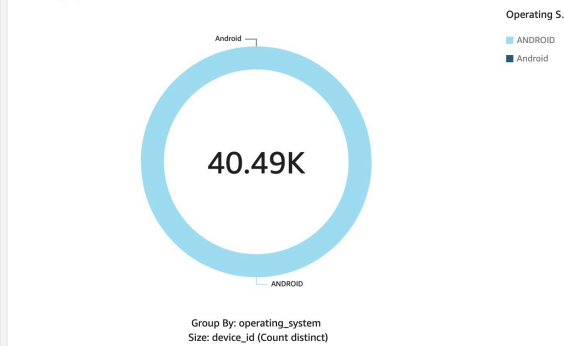


Device model name

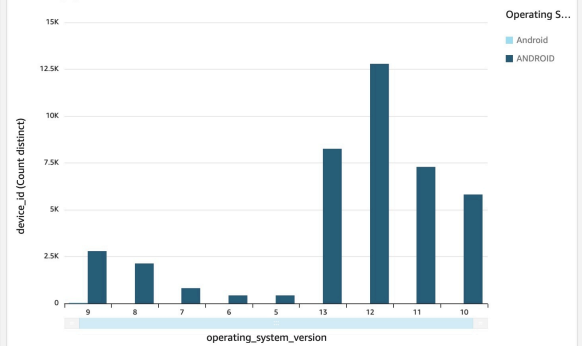
SHOWING TOP 100 IN MOBILE_MODEL_NAME



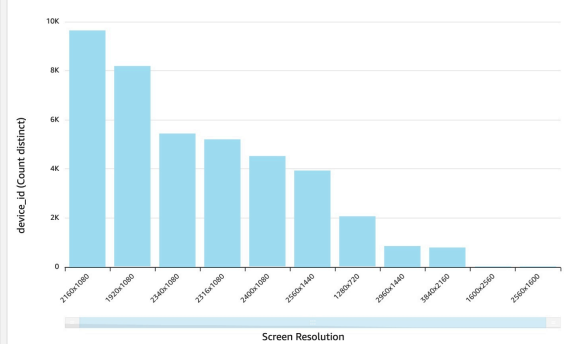
Operating System



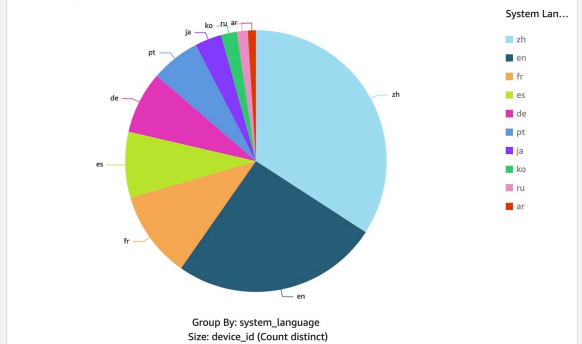
Operating system version



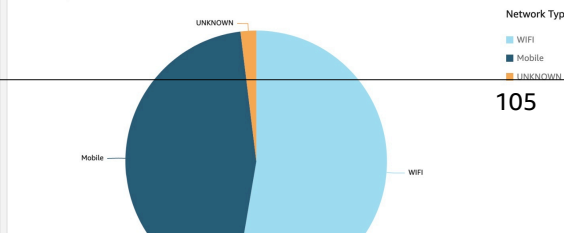
Screen resolution



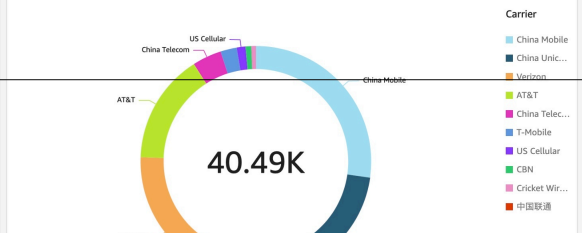
System language



Network type



Carrier



Path explorer

You can use the Path Explorer report to get insights into the user journey when users using your apps or websites, it helps you understand the sequence of events and screen or page transition in your apps.

Note

This article describes the default report. You can customize the report by applying filters or comparisons or by changing the dimensions, metrics, or charts in QuickSight. For more information, refer to [Visualizing data in Amazon QuickSight](#).

View the report

1. Access the dashboard for your application. Refer to [Access dashboard \(p. 79\)](#).
2. In the dashboard, choose the sheet with name of Path explorer.

Where the data comes from

Device reports are created based on the following QuickSight dataset:

- **Clickstream_Path**, which connects to the clickstream-path-mv view in analytics engines (that is, Redshift or Athena)

Below is the SQL command that generates the view.

Redshift SQL:

```
-- clickstream-path view
with event_data as (
select
    user_pseudo_id
    ,event_date
    ,event_id
    ,event_name
    ,event_timestamp
    ,platform
    ,(select
        ep.value.string_value as value
        from {{ schema }}.ods_events e, e.event_params ep
        where
            ep.key = '_session_id' and e.event_id = ods.event_id)::varchar session_id
    ,(select
        ep.value.string_value::varchar as screen_name
        from {{ schema }}.ods_events e, e.event_params ep
        where
            ep.key = '_screen_name' and event_name = '_screen_view' and e.event_id =
            ods.event_id) as current_screen
from notepad_v.ods_events ods), ranked_events as ( select
    *,
    DENSE_RANK() OVER (PARTITION BY user_pseudo_id, session_id ORDER BY event_timestamp
    ASC) event_rank,
    LAG(event_name,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
    event_timestamp ASC) previous_event,
    LEAD(event_name,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
    event_timestamp ASC) next_event,
    LAG(current_screen,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
    event_timestamp ASC) previous_screen,
    LEAD(current_screen,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
    event_timestamp ASC) next_screen
```

```
FROM event_data) select * from ranked_events
```

Athena SQL:

```
with base as (
  select
    *
  from {{ database }}.{{ eventTable }}
  where partition_app = ?
    and partition_year >= ?
    and partition_month >= ?
    and partition_day >= ?
),
event_data as (
  select
    user_pseudo_id
    ,event_date
    ,event_id
    ,event_name
    ,event_timestamp
    ,platform
    ,(select
      ep.value.string_value as value
    from base cross join unnest(event_params) as t(ep)
    where
      ep.key = '_session_id' and event_id = ods.event_id
    ) session_id
    ,(select
      ep.value.string_value as screen_name
    from base cross join unnest(event_params) as t(ep)
    where ep.key = '_screen_name' and event_name = '_screen_view' and event_id =
ods.event_id
    ) as current_screen
    from base ods
),
ranked_events as (
  select
    *,
    DENSE_RANK() OVER (PARTITION BY user_pseudo_id, session_id ORDER BY event_timestamp
ASC) event_rank,
    LAG(event_name,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
event_timestamp ASC) previous_event,
    LEAD(event_name,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
event_timestamp ASC) next_event,
    LAG(current_screen,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
event_timestamp ASC) previous_screen,
    LEAD(current_screen,1) OVER (PARTITION BY user_pseudo_id, session_id ORDER BY
event_timestamp ASC) next_screen
    FROM event_data
  )
  select * from ranked_events
```

Dimensions and metrics

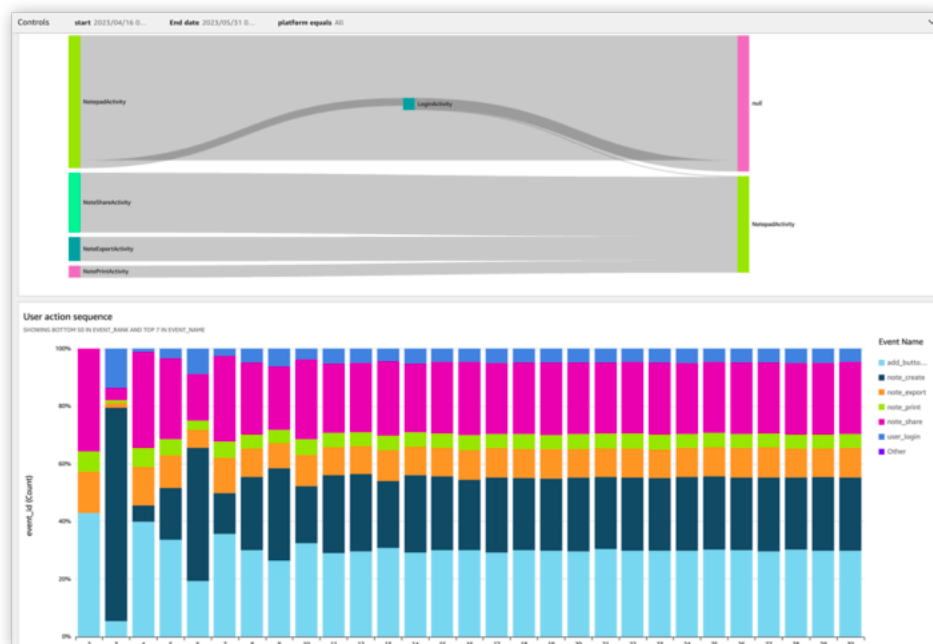
The report includes the following dimensions and metrics. You can add more dimensions or metrics by creating calculated field in QuickSight dataset. For more information, refer to [Adding calculated fields](#).

| Field | Type | Description | Data source |
|----------|-----------|-----------------------------|-----------------------------|
| event_id | Dimension | The unique ID for the event | Query from analytics engine |

| Field | Type | Description | Data source |
|-----------------|-----------|--|-----------------------------|
| user_pseudo_id | Dimension | A SDK-generated unique id for the user | Query from analytics engine |
| event_date | Dimension | The event data of when the device information was logged | Query from analytics engine |
| event_timestamp | Dimension | The timestamp when event happened | Query from analytics engine |
| platform | Dimension | The platform user used when event is logged | Query from analytics engine |
| session_id | Dimension | A SDK-generated unique id for the session user triggered when using your websites and apps | Query from analytics engine |
| current_screen | Dimension | The screen user is on for the event, 'null' for those events are not viewing screen or webpage | Query from analytics engine |
| event_rank | Dimension | The sequence of the event in a session | Query from analytics engine |
| previous_event | Dimension | The event name of previous event | Query from analytics engine |
| next_event | Dimension | The event name of next event | Query from analytics engine |
| previous_screen | Dimension | The screen name of previous screen | Query from analytics engine |
| next_screen | Dimension | The screen name of next screen | Query from analytics engine |
| event_id | Dimension | The unique ID for the event | Query from analytics engine |
| user_pseudo_id | Dimension | A SDK-generated unique id for the user | Query from analytics engine |
| event_date | Dimension | The event data of when the device information was logged | Query from analytics engine |

Sample dashboard

Below image is a sample dashboard for your reference.



Custom report

One of the key benefits of this solution is that you have complete control over the clickstream data collected from your apps and websites. You have complete flexibility to analyze the data for your specific business needs. This article illustrates the steps of creating a custom report with an example of creating funnel analysis by using Redshift Serverless as analytics engine and QuickSight as reporting tools.

Steps

Creating a custom report mainly consists of two parts, the first part is to prepare the dataset in your analytics engine, the second part is to create visualization in QuickSight.

Part 1 - Dataset preparation

1. Open **Redshift Serverless dashboard**.
2. Choose the workgroup starting with `clickstream-<project-id>` created by the solution.
3. Choose Query data. You will be directed to the Redshift Query Editor.
4. In the Editor view on the Redshift Query Editor, right click on the workgroup with name of `clickstream-<project-id>`. In the prompted drop-down, select Edit connection, and you will be asked to provide connection parameters. Follow this guide to use an appropriate method to connect.

Important

You will need read and write permissions for the database (with name as `<project-id>`) to create custom view or table. For example, you can use Admin user to connect to the cluster or workgroup. If you don't know the password for the Admin user, you can reset the admin password in the Redshift Console. For more information, refer to [Security and connections in Amazon Redshift Serverless](#).

5. If it is the first time you access the query editor, you will be prompted to configure the account. Choose **Config account** to open query editor.

6. Add a new SQL editor, and make sure you selected the correct workgroup and schema.
7. Create a new view for funnel analysis. In this example, we used below SQL.

```
CREATE OR REPLACE VIEW notepad.clickstream_funnel_view as
SELECT
platform,
COUNT(DISTINCT step1_id) AS login_users,
COUNT(DISTINCT step2_id) AS add_button_click_users,
COUNT(DISTINCT step3_id) AS note_create_users
FROM (
SELECT
    platform,
    user_pseudo_id AS step1_id,
    event_timestamp AS step1_timestamp,
    step2_id,
    step2_timestamp,
    step3_id,
    step3_timestamp
FROM
    notepad.ods_events
LEFT JOIN (
SELECT
    user_pseudo_id AS step2_id,
    event_timestamp AS step2_timestamp
FROM
    notepad.ods_events
WHERE
    event_name = 'add_button_click' )
ON
    user_pseudo_id = step2_id
    AND event_timestamp < step2_timestamp
LEFT JOIN (
SELECT
    user_pseudo_id AS step3_id,
    event_timestamp AS step3_timestamp
FROM
    notepad.ods_events
WHERE
    event_name= 'note_create' )
ON
    step3_id = step2_id
    AND step2_timestamp < step3_timestamp
WHERE
    event_name = 'user_login' )
group by
platform
```

8. Go to QuickSight console, choose '**Dataset**', and then choose '**New dataset**'.
9. In the New Dataset page, choose **Redshift Manual connect** to add dataset, and fill in the prompted form with the following parameters.
 - **Data source name:** clickstream-funnel-view-<project-id>
 - **Connection type:** select VPC connections / VPC Connection for Clickstream pipeline <project-id>
 - **Database server:** input the endpoint url of the serverless workgroup, which you can find on the workgroup console.
 - **Port:** 5439
 - **Database name:** <project-id>
 - **User name:** name of the user you used to created the custom view in previous steps
 - **Password:** password of the user you used to created the custom view in previous steps
10. Validate the connection, and then choose **Create data source**.
11. Choose the view from Redshift as data source - "**clickstream_funnel_view**", then

- Schema: select notepad
- Tables: clickstream_funnel_view

Note

When prompted to select Import to SPICE or Directly query your data, select Directly query your data for this example.

- Choose **Edit/Preview data** to preview the data. Once you're familiar with the data, choose **PUBLISH & VISUALIZE** at the top-right.

Part 2 - Create visualizations in QuickSight

1. When prompted, select a layout for your visualization.
2. Choose "+Add" at the top-left of the screen then choose "Add visual".
3. Select a Visual type at the bottom-left of the screen, in this example, select **Vertical bar chart**.
4. In the Field wells, select platform as X axis, login_user, add_button_click_users, and note_create_users as Value.

Now you can publish this analysis as dashboard or continue to format it. For more information, see [Visualizing data in Amazon QuickSight](#).

Frequently Asked Questions

General

Q: What is Clickstream Analytics on AWS?

An AWS Solution that enables customers to build clickstream analytic system on AWS easily. This solution automates the data pipeline creation per customers' configurations with a visual pipeline builder, and provides SDKs for web and mobiles apps (including iOS, and Android) to help customers to collect and ingest client-side data into the data pipeline on AWS. After data ingestion, the solution allows customers to further enrich and model the event data for business users to query, and provides built-in visualizations (for example, acquisition, engagement, retention) to help them generate insights faster.

SDK

Q: Could I use other SDK to send data to the pipeline created by this solution?

Yes. The solution supports using third-party SDK to send data to the pipeline. Note that, if you want to enable data processing and modeling module when using a third-party SDK to send data, you need to provide an transformation plugin to map third-party SDK's data structure to solution data schema. Please refer to [Custom plugin \(p. 54\)](#) for more details.

Setup and configuration

Q: How can I create more users for this solution?

If you launched the solution with Cognito User Pool, go to the AWS console, find the user pool created by the solution, and you can create more users. If you launched the solution with OpenID Connect (OIDC), you should add more users in the user pool managed by the OIDC provider. Note that all users have the same privileges.

Pricing

Q: How will I be charged and billed for the use of this solution?

The solution is free to use, and you are responsible for the cost of AWS services used while running this solution. You pay only for what you use, and there are no minimum or setup fees. Refer to the Cost section for detailed cost estimation.

Troubleshooting

The following help you to fix errors or problems that you might encounter when using Clickstream Analytics on AWS.

Problem: Deployment failure due to "Invalid Logging Configuration: The CloudWatch Logs Resource Policy size was exceeded"

If you encounter a deployment failure due to creating CloudWatch log group with an error message like the one below,

Cannot enable logging. Policy document length breaking Cloudwatch Logs Constraints, either < 1 or > 5120 (Service: AmazonApiGatewayV2; Status Code: 400; Error Code: BadRequestException; Request ID: xxx-yyy-zzz; Proxy: null)

Resolution:

[CloudWatch Logs resource policies are limited to 5120 characters](#). The remediation is merging or removing useless policies, then updating the resource policies of CloudWatch logs to reduce the number of policies.

Below is a sample command to reset resource policy of CloudWatch logs:

```
aws logs put-resource-policy --policy-name AWSLogDeliveryWrite20150319 \
--policy-document '
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite2",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "<your AWS account id>"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:logs:<AWS region>:<your AWS account id>:*"
        }
      }
    }
  ]
}
```

Problem: Cannot delete the CloudFormation stacks created for the Clickstream pipeline

If you encounter a failure with an error message like the one below when deleting the CloudFormation stacks created for the Clickstream pipeline,

```
Role arn:aws:iam::<your AWS account id>:role/<stack nam>-  
ClickStreamApiStackActionSta-<random suffix> is invalid or cannot be assumed
```

Resolution:

It results from deleting the web console stack for this solution before the CloudFormation stacks are made for the Clickstream pipeline.

Please create a new IAM role with the identical name mentioned in the above error message and trust the CloudFormation service with sufficient permission to delete those stacks.

Note

You can delete the IAM role after successfully removing those CloudFormation stacks.

Problem: Can not sink data to MSK cluster, got "InvalidReplicationFactor (Broker: Invalid replication factor)" log in Ingestion Server

If you notice that data can not be sunk into S3 through MSK cluster, and the error message in log of Ingestion Server (ECS) worker task is as below:

```
Message production error: InvalidReplicationFactor (Broker: Invalid replication  
factor)
```

Resolution:

This is caused by replication factor larger than available brokers, please edit the MSK cluster configuration, set **default.replication.factor** not larger than the total number of brokers.

Uninstall the solution

You will encounter an IAM role missing error if you delete Clickstream Analytics on AWS main stack before you delete the stacks created for Clickstream projects. Clickstream Analytics on AWS console launches additional CloudFormation stacks for the Clickstream pipelines. We recommend you delete projects before uninstalling the solution.

Step 1. Delete projects

1. Go to the Clickstream Analytics on AWS console.
2. In the left sidebar, choose **Projects**.
3. Select the project to be deleted.
4. Choose the **Delete** button in the upper right corner.
5. Repeat steps 3 and 4 to delete all your projects.

Step 2. Delete Clickstream Analytics on AWS stack

1. Go to the [CloudFormation console](#).
2. Find the CloudFormation stack of the solution.
3. Delete the CloudFormation Stack of the solution.
4. (Optional) Delete the S3 bucket created by the solution.
 - a. Choose the CloudFormation stack of the solution, and select the **Resources** tab.
 - b. In the search bar, enter DataBucket. It shows all resources with the name DataBucket created by the solution. You can find the resource type **AWS::S3::Bucket**, and the **Physical ID** field is the S3 bucket name.
 - c. Go to the S3 console, and find the S3 bucket with the bucket name. **Empty** and **Delete** the S3 bucket.

Additional resources

Upload SSL Certificate to IAM

Upload the SSL certificate by running the AWS CLI command `upload-server-certificate` similar to the following:

```
aws iam upload-server-certificate --path /cloudfront/ \  
--server-certificate-name YourCertificate \  
--certificate-body file://Certificate.pem \  
--certificate-chain file://CertificateChain.pem \  
--private-key file://PrivateKey.pem
```

Replace the file names and Your Certificate with the names for your uploaded files and certificate. You must specify the `file://` prefix in the `certificate-body`, `certificate-chain` and `private-key` parameters in the API request. Otherwise, the request fails with a `MalformedCertificate: Unknown error` message.

Note

You must specify a path using the `--path` option. The path must begin with `/cloudfront` and must include a trailing slash (for example, `/cloudfront/test/`).

After the certificate is uploaded, the AWS command `upload-server-certificate` returns metadata for the uploaded certificate, including the certificate's Amazon Resource Name (ARN), friendly name, identifier (ID), and expiration date.

To view the uploaded certificate, run the AWS CLI command `list-server-certificates`:

```
aws iam list-server-certificates
```

For more information, see [uploading a server certificate](#) to IAM.

Developer guide

This section provides the source code for the solution.

Source code

Visit our GitHub repository to download the [source code](#) for this solution. The Clickstream Analytics on AWS template is generated using the [AWS Cloud Development Kit \(AWS CDK\) \(CDK\)](#). Refer to the [README.md](#) file for additional information.

Contributors

- Chen, Haiyun
- Deng, Mingtong
- Li, Min
- Liu, Yong
- Luo, Robin
- Qiao, Wei
- Qian, Yang
- Que, Mingfei
- Zhu, Mengxin
- Zhu, Xiaowei

Revisions

| Date | Change |
|-----------|------------------|
| July 2023 | Initial release. |

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Clickstream Analytics on AWS is licensed under the terms of the Apache License Version 2.0 available at [The Apache Software Foundation](https://www.apache.org/licenses/LICENSE-2.0).