Real-Time Web Analytics with Kinesis Data Analytics

Implementation Guide

Real-Time Web Analytics with Kinesis Data Analytics: Implementation Guide

Table of Contents

Welcome 1
Overview 2
    Cost 2
    Architecture 2
Components 5
    Beacon Code 5
        Metric Headers 5
    Beacon Web Servers 5
    Amazon Kinesis Data Analytics Application 6
    Amazon DynamoDB 6
    Amendment Strategy 6
    Web Usage Dashboard 6
    Anomaly Detection 7
    Amazon CloudWatch Dashboard 7
Considerations 9
    Regional Deployment 9
Templates 10
Deployment 11
    Launch the Stack 11
Security 14
    Security Group 14
    AWS Systems Manager 14
    HTTPS Websites 14
    Amazon CloudFront 14
Resources 15
Appendix A: Code Components 16
    Beacon Code 16
    SQL Query 16
    JavaScript 17
    HTML Element 17
Appendix B: Customizing the Dashboard 18
    Step 1. Add the Beacon Code to Your Web Server 18
    Step 2: Add the Metric to the DynamoDB Table 18
    Step 3. Modify the Application's SQL Code 19
    Step 4. Update the JavaScript Code 19
    Step 5. Update the Website Assets 20
Appendix C: Existing VPC Template 21
    Prerequisites 21
    Launch the Stack 11
Appendix D: Operational Metrics 24
Source Code 25
Revisions 26
    26

Deploy a framework for analyzing and visualizing website metrics

Publication date: *March 2018 (last update (p. 26): October 2019)*

This implementation guide discusses architectural considerations and configuration steps for deploying Real-Time Web Analytics with Kinesis Data Analytics on the Amazon Web Services (AWS) Cloud. It includes links to an AWS CloudFormation template that launches, configures, and runs the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, developers, and DevOps professionals who have practical experience architecting on the AWS Cloud.

Overview

Tracking website metrics in real-time can provide insight into who visits your website, where they come from, and what content they view. You can track time series data on visitor counts, page views, time spent on site, channels driving traffic, actions visitors take, and other custom metrics. This insight can help you react quickly to changing conditions and emerging situations.

Amazon Web Services (AWS) offers a comprehensive set of powerful, flexible, and simple-to-use services that enable you to extract insights and actionable information in real time. Amazon Kinesis is a platform for streaming data on AWS, offering key capabilities to cost-effectively process streaming data at any scale. Kinesis capabilities include Amazon Kinesis Data Analytics, the easiest way to process streaming data in real time with standard SQL without having to learn new programming languages or processing frameworks.

To help customers more easily leverage Kinesis Data Analytics, AWS offers the Real-Time Web Analytics with Kinesis Data Analytics solution, a reference implementation that automatically provisions the services necessary to collect, process, analyze and visualize website clickstream data in real-time. This solution is designed to provide a framework for analyzing and visualizing metrics, allowing you to focus on adding new metrics rather than managing the underlying infrastructure.

Real-Time Web Analytics with Kinesis Data Analytics creates a web activity monitoring system that includes beacon web servers to log requests from a user's web browser, Amazon Kinesis Data Firehose to capture website clickstream data, Kinesis Data Analytics to compute metrics in real-time, and Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB to durably store metric data. The solution features a dashboard that visualizes your website clickstream activity metrics in real-time.

Cost

You are responsible for the cost of the AWS services used while running this reference deployment. As of the date of publication, the baseline cost for running this solution with default settings in the US East (N. Virginia) Region is approximately **$100 per month**. This cost estimate assumes the solution will record 1 million events per day with an average size of one kilobyte per event. Note that the monthly cost will vary depending on the number of events the solution processes. For 10 million events per day, the cost is approximately **$170 per month**. For 100 million events per day, the cost is approximately **$950 per month**. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution builds the following environment in the AWS Cloud.

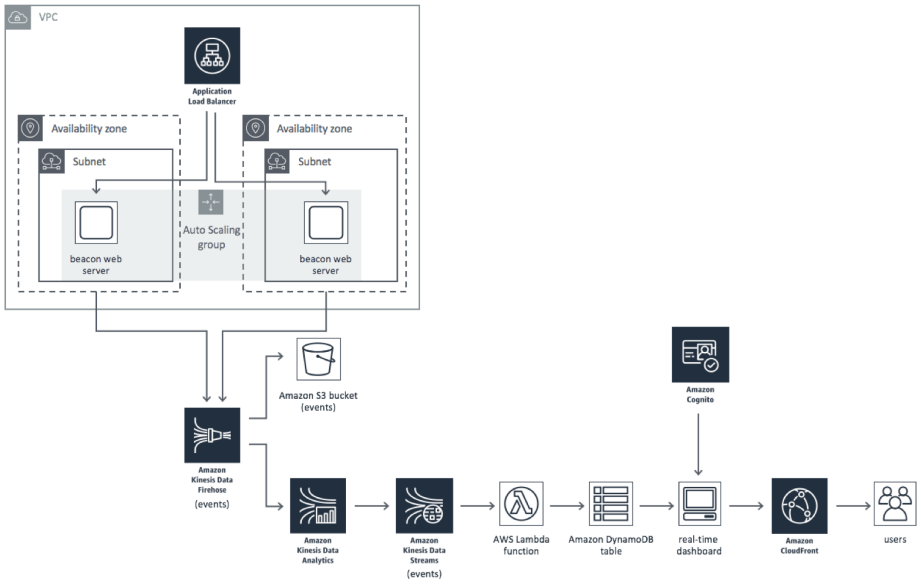

Figure 1: Real-Time Web Analytics with Kinesis Data Analytics architecture

The solution's primary AWS CloudFormation template (`real-time-web-analytics-with-kinesis`) deploys a multi-AZ Amazon Virtual Private Cloud (Amazon VPC) network topology with two public subnets, a multi-AZ Application Load Balancer (ALB), two Amazon Elastic Compute Cloud (Amazon EC2) instances in an Auto Scaling group, an Amazon Kinesis Data Firehose delivery stream, Amazon Simple Storage Service (Amazon S3) buckets, a Kinesis data analytics application, a Kinesis data stream, an AWS Lambda function, Amazon DynamoDB tables, an Amazon Cognito user pool, an optional Amazon CloudWatch dashboard, an Amazon CloudFront distribution, and a real-time web usage dashboard to securely read and display the clickstream metrics stored in the DynamoDB table.

This solution also includes a template that deploys the solution in an existing VPC (`real-time-web-analytics-with-kinesis-existing-vpc`). For more information, see Appendix C (p. 21).

When a user accesses your website, client-side JavaScript code on your web server sends a web analytic beacon to the solution's beacon web servers which log the request. An Application Load Balancer manages the incoming traffic.

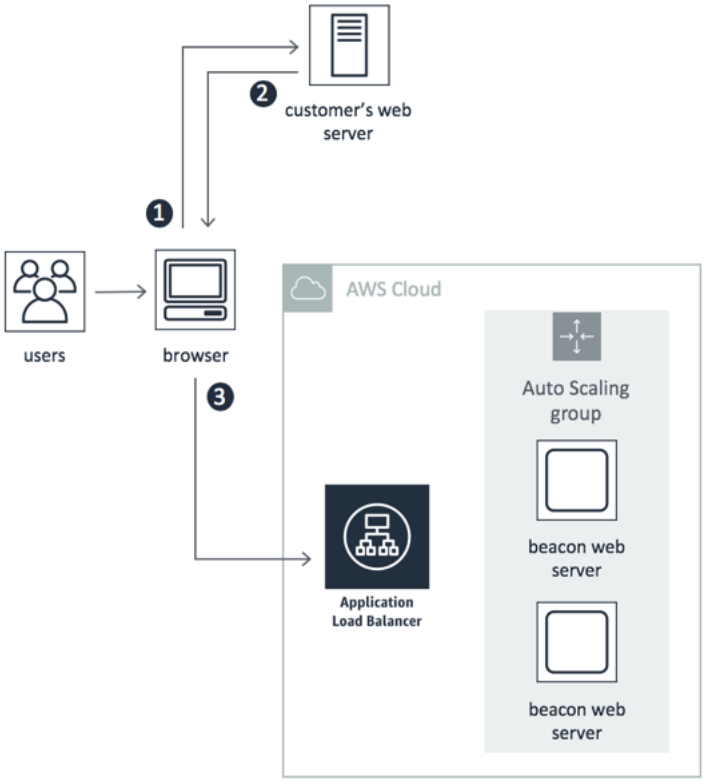

Figure 2: Request process

The beacon web servers send the data to the Kinesis data delivery stream, which archives the events in an Amazon S3 bucket and sends the data to the Kinesis data analytics application for processing. Once the data is processed, it is sent to the Kinesis data stream. A Lambda function (`ProcessMetricsFunction`) reads data from the stream and sends the data in real-time to a DynamoDB table to be stored for up to one week.

Solution Components

Beacon Code

The Real-Time Web Analytics with Kinesis Data Analytics solution includes beacon code that you add to your web server. This code enables your users' web browsers to send requests to the solution's web servers. The requests include header information that is used to calculate the metrics that are displayed as a count over a 10 second time window on the solution's real-time dashboard.

```
var http = new XMLHttpRequest();
var url = beacon_url; //from Outputs section of CloudFormation stack
http.open("POST", beacon_url);
http.setRequestHeader("event","click");
http.setRequestHeader("clientid","user123");
http.setRequestHeader("page", window.location.pathname.split("/").slice(-1));
http.setRequestHeader("referer",document.referrer);
http.setRequestHeader("custom_metric_name","userAgent");
http.setRequestHeader("custom_metric_string_value",navigator.userAgent);
http.send();
```

Figure 3: Sample beacon code

Metric Headers

This solution includes headers you use to collect website usage metrics that are displayed in the dashboard. The following headers are passed to the beacon server.

- `event`: String used to capture the user event (For example, `click`, `pageview`, `playvideo`, `conversion`, `exception`, `login`, `logoff`)
- `page`: Webpage associated with the event (For example, `window.location.pathname` or `window.location.pathname.split("/").slice(-1)`)
- `referrer`: Referring page (For example, `document.referrer`)

The solution also includes headers you can use to include custom metrics on the real-time dashboard. The following headers define the name of the custom metric and the different data types of the values that are passed to the beacon server.

- `custom_metric_name`: Defines the name of your custom metric
- `custom_metric_int_value`: For custom metrics with integer values
- `custom_metric_float_value`: For custom metrics with float values
- `custom_metric_string_value`: For custom metrics with string values

Beacon Web Servers

Real-Time Web Analytics with Kinesis Data Analytics uses two Amazon Elastic Compute Cloud (Amazon EC2) instances for the beacon web servers. You can choose from two preset configurations to support

your anticipated request traffic: 50K requests per minute (t2.medium instance) or 100K requests per minute (m5.large instance). Note that the Auto Scaling group will scale in the preset increment (50K or 100K).

Amazon Kinesis Data Analytics Application

This solution includes an Amazon Kinesis Data Analytics application with SQL statements that compute metrics for the built-in dashboard. The application reads records from the Amazon Kinesis Data Firehose delivery stream, and runs the SQL queries to emit specific website clickstream metrics, which are stored in Amazon DynamoDB. For more information, see Appendix A (p. 16).

Amazon DynamoDB

The Real-Time Web Analytics with Kinesis Data Analytics solution creates an Amazon DynamoDB table: `Metrics`.

The `Metrics` table stores the following information on metrics computed by the Amazon Kinesis Data Analytics application:

- **MetricType:** The name of the computed metric
- **AmendmentStrategy:** The amendment strategy for metric detail items with identical timestamps. For more information, see Amendment Strategy (p. 6).
- **IsSet:** Indicates whether there is more than one data point in a metric detail item. For one data point, set this item to `false`. For more than one data point, set this item to `true`.
- **IsWholeNumber:** Indicates whether the value of this metric is an integer or a float value. For an integer value, set this item to `true`. For a float value, set this item to `false`.

Amendment Strategy

This solution includes an amendment strategy that defines how the solution handles a new record with the same timestamp as a record that has already been received. You can choose from the following three options:

- **Add:** Adds values from the new record to values from the existing record. For example, if a record is received for `event_count` with a value of `logon 10` that has the same timestamp as an existing `event_count` record that has a value of `logon 4`, the values will be added together (`10 + 4`) and the existing record will be updated with the new value: `logon 14`.
- **Replace:** Replaces the existing record with the new record.
- **Replace existing:** New and existing records are merged, with existing values in the existing record overwritten by the new values. For example, if a record is received for `event_count` with values of `logon 10` and `click 2` that has the same timestamp as an existing `event_count` record that has values of `logon 4` and `logoff 2`, the existing record's new values will be `logon 10`, `logoff 2`, and `click 2`.

Web Usage Dashboard

The solution features a simple dashboard that loads data from Amazon DynamoDB into line charts every 10 seconds and bar charts every minute. The dashboard leverages Amazon Cognito for user

authentication and is powered by web assets hosted in an Amazon Simple Storage Service (Amazon S3) bucket. Amazon CloudFront is used to restrict access to the solution's website bucket contents.

The dashboard uses the open-source `chart.js` JavaScript library to draw charts using HTML5. The `index.html` file contains the HTML elements that render the charts in the dashboard. The `dash.js` file in the `js` folder contains the JavaScript that populates the dashboard with metrics. The Kinesis data application contains the SQL queries that compute metrics. For more information, see Appendix A (p. 16).

After you successfully launch the solution, you will receive an email with instructions for logging into the dashboard.

The dashboard can also be customized to include additional metrics. For more information, see Appendix B (p. 18).

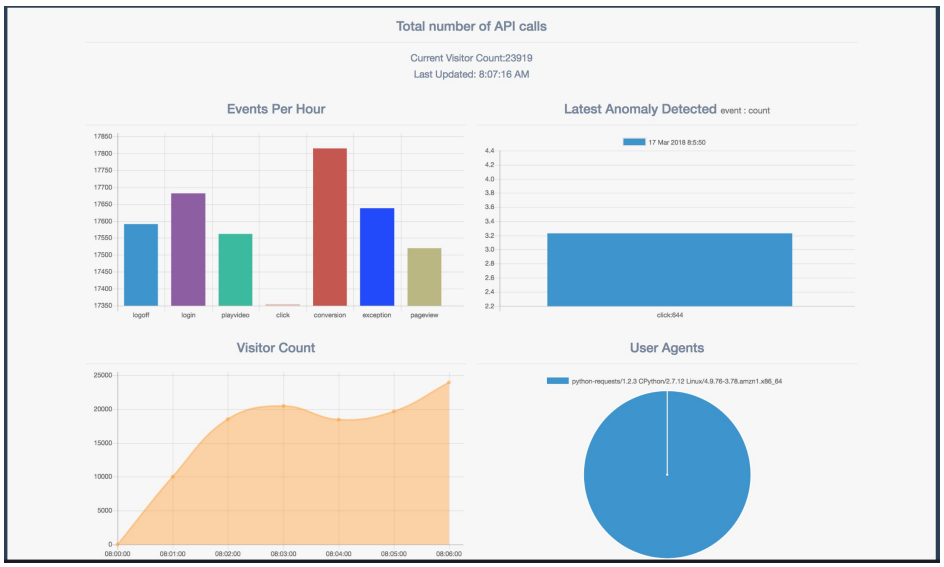

Figure 4: Real-time web usage dashboard

Anomaly Detection

The Real-Time Web Analytics with Kinesis Data Analytics solution leverages the built-in anomaly detection of Amazon Kinesis. This solution calculates an anomaly score based on a comparison of the last 256 events to a random set of the last 100,000 events. When an anomaly score hits the threshold, the events are displayed in the solution's anomaly detection graph. For example, the solution might record 50 logon events every 10 seconds and 0 exception events every 10 seconds. If the solution then records five logon events every 10 seconds and 30 exception events every 10 seconds, the solution will detect the anomaly and display it in the graph.

Amazon CloudWatch Dashboard

This solution provides an optional dashboard you can use to monitor the performance of your beacon web servers. The dashboard displays custom operational Amazon CloudWatch metrics for your beacon web servers, including the number of healthy beacon web servers, the average processed network packets, aggregate requests, 5XX errors, and Amazon DynamoDB throughput capacity and throttling.

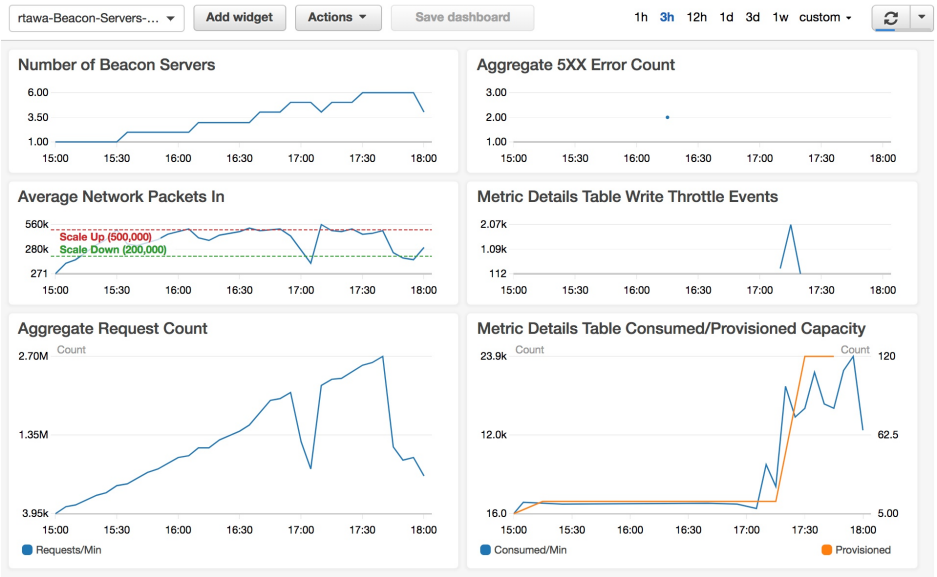

Figure 5: CloudWatch metrics dashboard

Considerations

Regional Deployment

This solution uses the Amazon Kinesis Data Firehose and Amazon Kinesis Data Analytics services, which are currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region where these services are available. For the most current service availability by region, see AWS service offerings by region.

AWS CloudFormation Templates

This solution uses AWS CloudFormation to automate the deployment of the Real-Time Web Analytics with Kinesis Data Analytics solution. It includes the following AWS CloudFormation template, which you can download before deployment:

real-time-web-analytics-with-kinesis.template: Use this template to launch the solution and all associated components. The default configuration deploys a highly available Amazon Virtual Private Cloud (Amazon VPC) network topology, an Application Load Balancer (ALB), two Amazon Elastic Compute Cloud (Amazon EC2) instances in an Auto Scaling group, an Amazon Kinesis Data Firehose delivery stream, Amazon Simple Storage Service (Amazon S3) buckets, a Kinesis data analytics application, a Kinesis data stream, an AWS Lambda function, Amazon DynamoDB tables, an Amazon Cognito user pool, an Amazon CloudFront distribution, and a real-time dashboard, but you can also customize the template based on your specific needs.

real-time-web-analytics-with-kinesis-existing-vpc.template: Use this template to launch the solution and all associated components into an existing VPC. The default configuration deploys two Amazon EC2 instances in an Auto Scaling group, an ALB, a Kinesis data delivery stream, Amazon S3 buckets, a Kinesis data analytics application, a Kinesis data stream, a Lambda function, DynamoDB tables, an Amazon Cognito user pool, an Amazon CloudFront distribution, and a real-time dashboard, but you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy Real-Time Web Analytics with Kinesis Data Analytics into your account.

Time to deploy: Approximately 10 minutes

Launch the Stack

This automated AWS CloudFormation template deploys the Real-Time Web Analytics with Kinesis Data Analytics solution in a new VPC. To launch the solution in an existing VPC, see Appendix C (p. 21). Please make sure that you've verified that you have reviewed the considerations and prerequisites before launching the stack.

> **Note**
> You are responsible for the cost of the AWS services used while running this solution. See the Cost (p. 2) section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button below to launch the `real-time-web-analytics-with-kinesis` AWS CloudFormation template.

   Launch Solution

   You can also download the template as a starting point for your own implementation.
2. The template is launched in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the region selector in the console navigation bar.

   > **Note**
   > This solution uses the Amazon Kinesis Data Firehose and Amazon Kinesis Data Analytics services, which are currently available in specific AWS Regions only. Therefore, you must launch this solution an AWS Region where these services are available. For the most current service availability by region, see AWS service offerings by region.
3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template, and modify them as necessary.This solution uses the following default values.

Parameter	Default	Description
User Name	*<Requires input>*	User name to access the real-time dashboard
User Email Address	*<Requires input>*	Email address of dashboard user. After launch, an email will be sent to this address with dashboard login instructions.

Parameter	Default	Description
CloudWatch Dashboard	*Yes*	Choose whether to deploy the Amazon CloudWatch metrics dashboard (p. 7)
CORS Origin	*	The value that is returned by the Access-Control-Allow-Origin header. A star (*) value will support any origin. We recommend specifying a specific origin (e.g. http://example.com) to restrict cross-site access to your API.
Node Requests/Min	*50K*	The number of requests per minute that each beacon node will support. Choose `50K` or `100K`.
Min Beacon Servers	*2*	The minimum number of beacon web servers. For high availability, specify at least `2`.
Max Beacon Servers	*6*	The maximum number of beacon servers
SSH Key Pair	*<Requires input>*	Public and private key pair, which allows you to connect securely to the beacon web servers. When you created an AWS account, this is the key pair you created in your preferred AWS Region.
Enable SSH?	*false*	Choose whether to allow SSH access to beacon servers. If you select `true` for this parameter, you must specify an **SSH Key Pair** and a CIDR block in the **Enable SSH From** parameter. **Note** This solution's beacon web servers are configured to be managed by AWS Systems Manager. As a result, you can use Run Command to connect to the beacon server instead of SSH.
Enable SSH From	<Optional input>	This IP address CIDR block will have access to the beacon web servers
Beacon Server VPC CIDR	`10.0.0.0/23`	CIDR block for the new VPC for the beacon web servers

Parameter	Default	Description
1st Subnet Network	`10.0.0.0/24`	CIDR block for the new VPC public subnet created in AZ1
2nd Subnet Network	`10.0.1.0/24`	CIDR block for the new VPC public subnet created in AZ2
1st Subnet AZ #	`0`	The Availability Zone number for the first public subnet
2nd Subnet AZ #	`1`	The Availability Zone number for the second public subnet

6. Choose **Next**.
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create** to deploy the stack.

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

   The solution sends an email invitation to join the real-time dashboard.
10. In the email, follow the instructions to sign in to the dashboard.

Note
In addition to the primary AWS Lambda function `ProcessMetricsFunction`, this solution includes the `CustomResourceHelper` Lambda function, which runs only during initial configuration or when resources are updated or deleted.
After launching this solution, you will see these Lambda functions in the AWS console, but only the `ProcessMetricsFunction` function is regularly active. However, do not delete the `CustomResourceHelper` function as it is necessary to manage associated resources.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

Security Group

The security group created in this solution is designed to control and isolate network traffic to the web server Amazon Elastic Compute Cloud (Amazon EC2) instance. The security group allows only traffic from port 80 to the web server.

AWS Systems Manager

The Real-Time Web Analytics with Kinesis Data Analytics beacon web servers are configured to be managed by AWS Systems Manager. By default, AWS Systems Manager pushes the AWS patch baseline to beacon servers once per day. For more information about how to automate beacon server management using Run Command, see AWS Systems Manager Run Command.

HTTPS Websites

This solution deploys an Application Load Balancer to route web client traffic to the beacon servers. For HTTPS websites, we recommend that you leverage HTTPS listeners and create certificates with AWS Certificate Manager (ACM) and deploy it to your load balancer. For more information, see the AWS Certificate Manager User Guide. You will also need to enable port 443 ingress traffic on your ALB security group.

Amazon CloudFront

This solution deploys a static website hosted in an Amazon S3 bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity, which is a special CloudFront user that helps restrict access to the solution's website bucket contents. For more information, see Restricting Access to Amazon S3 Content by Using an Origin Access Identity.

Additional Resources

AWS services

- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Analytics
- AWS Lambda
- Amazon DynamoDB
- Amazon Simple Storage Service
- Amazon Cognito
- Amazon CloudWatch
- AWS CloudFormation
- Amazon CloudFront

Appendix A: Code Components

The Real-Time Web Analytics with Kinesis Data Analytics solution uses four main code components to process and display metrics on the real-time dashboard. Beacon JavaScript code running in the user's web browser sends requests with header information to the solution's beacon web server. The Amazon Kinesis Data Analytics application (`WebMetricsApplication`) runs SQL queries against the in-application streams and emits the results. A JavaScript file (`dash.js`) populates the chart with the results of the queries, and an HTML file (`index.html`) renders the chart on the dashboard in real-time.

The following example shows the beacon code, SQL, JavaScript, and HTML code for the `top_pages` metric.

Beacon Code

The beacon code is used to send header information about pages to the solution's web server, which is then sent to the Kinesis data delivery stream. Note that the `beacon_url` can be found in the AWS CloudFormation stack **Outputs**.

```
var http = new XMLHttpRequest();
var url = beacon_url; //from Outputs section of CloudFormation stack
http.open("POST", beacon_url);
http.setRequestHeader("event","click");
http.setRequestHeader("page","productpage.html");
http.setRequestHeader("clientid","user123");
http.send();
```

The page header values will be displayed in the top pages chart as a count over a ten second window.

SQL Query

The SQL query calculates the top pages, in 10-second intervals, based on page views. The result is stored in an output in-application stream (`DESTINATION_SQL_STREAM`) with the name of the metric (`top_pages`) and the corresponding values.

```
CREATE OR REPLACE PUMP "PAGEVIEWS_PUMP" AS
INSERT INTO "DESTINATION_SQL_STREAM" ( MetricType, EventTimestamp, MetricItem,
 UnitValueInt)
SELECT 'top_pages', UNIX_TIMESTAMP(eventTimestamp), page, page_count FROM (
    SELECT stream
        weblogs."page" as page,
        count(*) as page_count,
        STEP (CHAR_TO_TIMESTAMP('dd/MMM/yyyy:HH:mm:ss z',weblogs."datetime") by INTERVAL
 '10' SECOND) as eventTimestamp
    FROM "WASA_001" weblogs
    GROUP BY
        STEP (weblogs.ROWTIME BY INTERVAL '10' SECOND),
        STEP (CHAR_TO_TIMESTAMP('dd/MMM/yyyy:HH:mm:ss z',weblogs."datetime") by INTERVAL
 '10' SECOND),
        weblogs."page"
```

```
    HAVING count(*) > 1
    ORDER BY STEP (weblogs.ROWTIME BY INTERVAL '10' SECOND), page_count desc
);
```

JavaScript

The JavaScript (in the `dash.js` file) populates the chart with the top pages by page views.

```
switch(mtype) {
    case 'hourly_events' :
        makeBarChart(mtype, items);
        break;
    case 'event_anomaly' :
        makeAmomalyBarChart(mtype, items);
        break;
    case 'agent_count' :
        makePieChart(mtype, items);
        break;
    case 'referral_count' :
    case 'top_pages' :
        makeHorizontalBarChart(mtype,items);
        break;
    case 'visitor_count' :
        document.getElementById(mtype).innerHTML = 'Current Visitor Count:' +
 items[0].UNITVALUEINT;
        makeVisitorLineChart(mtype,items);
        break;
     case 'event_count' :
        makeEventLineChart(mtype,items);
        break;
}
```

HTML Element

The HTML element (in the `index.html` file) renders the `top_pages` chart with the results of the SQL query.

```
<div class="row aws-mb-l">
    <div class="col-xs-10 col-xs-offset-1 col-xs-12">
        <div class="x_title">
            <h3>Pages</h3>
        </div>
        <div class="x_content">
            <canvas id="top_pages" ts="0"></canvas>
        </div>
    </div>
</div>
```

Appendix B: Customizing the Dashboard

The Real-Time Web Analytics with Kinesis Data Analytics solution dashboard displays a default set of metrics, but you can customize the dashboard to include any metrics from your website clickstream. Follow the step-by-step instructions in this section to add a metric that calculates the client-side load time of web pages.

For this exercise, you can add the `page_load_time` header value to the beacon code on your web server.

Step 1. Add the Beacon Code to Your Web Server

Add the beacon code that contains the new header name and data type to your web server. For this example, add the following code to your web server.

```
var loadTime = window.performance.timing.domContentLoadedEventEnd-
 window.performance.timing.navigationStart;
var http = new XMLHttpRequest();
var url = beacon_url; //from Outputs section of CloudFormation stack
http.open("POST", beacon_url);
http.setRequestHeader("custom_metric_name", "page_load_time");
http.setRequestHeader("custom_metric_int_value", loadTime);
http.send();
```

Step 2: Add the Metric to the DynamoDB Table

Use this procedure to add the new metric to the solution's Amazon DynamoDB table. For this exercise, the item you will add represents a metric that will emit the average page load time in milliseconds as an integer.

1. Open the DynamoDB console and navigate to the **Metrics** table.
2. Add a new item to the table with the following fields:
   - For the **MetricType** string attribute, enter a value of `avg_pg_ld`.
   - Add a **LatestEventTimestamp** attribute with a value of `0`.
   - Add an **AmendmentStrategy** string attribute with a value of `replace-existing`.
   - Add an **IsSet** Boolean attribute with a value of `false`.
   - Add an **IsWholeNumber** Boolean attribute with a value of `true`.
3. Select **Save**.

Note that the custom metric header name (`page_load_time`) you defined in step 1 is not the same as the metric type (`avg_pg_ld`). The `page_load_time` custom header contains the number of

milliseconds a page takes to load as calculated on the client. The `avg_pg_ld` metric is used to show the average load time for all instrumented pages over a one-minute period.

Step 3. Modify the Application's SQL Code

Use this procedure to update the application's code with the new SQL statement. For information on using the SQL Editor, see Working with the SQL Editor in the Amazon Kinesis Data Analytics Developer Guide.

For this exercise, the `page_load_time` custom header value will be used to generate an average page load time (`avg_pg_ld`) of all pages over a one-minute period.

1. On the Kinesis Data Analytics application's **SQL Editor** page, select the **Real-time analytics** tab.
2. Add the following SQL statement

```
CREATE OR REPLACE PUMP "PAGELOAD_PUMP" AS
INSERT INTO "DESTINATION_SQL_STREAM" (MetricType, EventTimestamp, MetricItem,
 UnitValueInt)
SELECT 'avg_pg_ld', UNIX_TIMESTAMP(eventTimestamp), MetricItem, average_ms FROM (
    SELECT STREAM
    'All Pages' as MetricItem,
    AVG(weblogs."custom_metric_int_value") as average_ms,
    STEP (CHAR_TO_TIMESTAMP('dd/MMM/yyyy:HH:mm:ss z',weblogs."datetime") by INTERVAL
 '60' SECOND) as eventTimestamp
    FROM "WASA_001" weblogs
    WHERE weblogs."custom_metric_name" = 'page_load_time'
    GROUP BY
        STEP (CHAR_TO_TIMESTAMP('dd/MMM/yyyy:HH:mm:ss z',weblogs."datetime") by
 INTERVAL '60' SECOND),
        STEP (weblogs.ROWTIME BY INTERVAL '60' SECOND)
);
```

   This SQL statement creates a new metric (`page_load_time`) that calculates the client-side load time of web pages.
3. Select **Save and run SQL**.

Step 4. Update the JavaScript Code

The solution creates an Amazon Simple Storage Service (Amazon S3) bucket with a `js` folder that contains a `dash.js` file with the JavaScript code that populates the charts with metrics. To populate the new chart with metrics, download the `dash.js` file and follow the step-by-step instructions to modify the JavaScript.

1. Add the event type to the switch statement of the `getLatestMetrics` function. For this exercise, add the bold JavaScript to the `dash.js` file.

```
switch(mtype) {
    case 'avg_pd_ld' :
        makeBarChart(mtype, items);
        break;
    case 'agent_count' :
        makePieChart(mtype, items);
        break;
```

2. Upload the modified `dash.js` file to the solution's Amazon S3 bucket.

Step 5. Update the Website Assets

In the Amazon S3 bucket with the JavaScript, there is a file (`index.html`)that contains all the HTML elements that render charts on the dashboard. To add a new chart, download the `index.html` file, modify the HTML, and upload the modified file to the Amazon S3 bucket. You can replace the row of an existing chart with the new row, or add the new row to the end of the file.

For this exercise, add the bold HTML element to the `index.html` file.

```
<div class="row aws-mb-l">
    <div class="col-xs-10 col-xs-offset-1 col-xs-12">
        <div class="x_title">
            <h3>Pages</h3>
        </div>
        <div class="x_content">
            <canvas id="top_pages" ts="0"></canvas>
        </div>
    </div>
</div>
<div class="row aws-mb-l">
    <div class="col-xs-10 col-xs-offset-1 col-xs-12">
        <div class="x_title">
            <h3>Average Page Load Time</h3>
        </div>
        <div class="x_content">
            <canvas id="avg_pg_ld" ts="0"></canvas>
        </div>
    </div>
</div>
```

After you upload the modified `index.html` file to the Amazon S3 bucket, open the dashboard in a browser and verify the new chart shows metrics.

Appendix C: Existing VPC Template

The Real-Time Web Analytics with Kinesis Data Analytics solution includes an AWS CloudFormation template that allows you to deploy the solution in an existing virtual private cloud (VPC).

Prerequisites

Before you start, the existing Amazon VPC must have the following:

- Two public subnets in different Availability Zones for the solution's Application Load Balancer (ALB).
- Two subnets that can receive traffic from the solution's ALBs and connect to Amazon Kinesis Data Firehose API endpoints for the solution's beacon web servers.

Launch the Stack

1. Sign in to the AWS Management Console and click the button below to launch the `real-time-web-analytics-with-kinesis-existing-vpc` AWS CloudFormation template.

   

   You can also download the template as a starting point for your own implementation.
2. The template is launched in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the region selector in the console navigation bar.

   Note
   This solution uses the Amazon Kinesis Data Firehose and Amazon Kinesis Data Analytics services, which are currently available in specific AWS Regions only. Therefore, you must launch this solution an AWS Region where these services are available. For the most current service availability by region, see AWS service offerings by region.
3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template, and modify them as necessary.This solution uses the following default values.

Parameter	Default	Description
User Name	*<Requires input>*	User name to access the real-time dashboard
User Email Address	*<Requires input>*	Email address of dashboard user. After launch, an email will be sent to this address with dashboard login instructions.

Parameter	Default	Description
CloudWatch Dashboard	*Yes*	Choose whether to deploy the Amazon CloudWatch metrics dashboard (p. 7)
CORS Origin	*	Specify a name for the new Amazon S3 bucket where the real-time dashboard will be stored. Do not specify an existing bucket.
Node Requests/Min	*50K*	The number of requests per minute that each beacon node will support. Choose `50K` or `100K`.
Min Beacon Servers	*2*	The minimum number of beacon web servers. For high availability, specify at least `2`.
Max Beacon Servers	*6*	The maximum number of beacon servers
SSH Key Pair	*<Requires input>*	Public and private key pair, which allows you to connect securely to the beacon web servers. When you created an AWS account, this is the key pair you created in your preferred AWS Region.
Enable SSH?	*false*	Choose whether to allow SSH access to beacon servers. If you select `true` for this parameter, you must specify an **SSH Key Pair** and a CIDR block in the **Enable SSH From** parameter. **Note** This solution's beacon web servers are configured to be managed by AWS Systems Manager. As a result, you can use Run Command to connect to the beacon server instead of SSH.
Enable SSH From	<Optional input>	This IP address CIDR block will have access to the beacon web servers
Existing VPC	*<Requires input>*	The VPC ID of the existing VPC for the beacon severs

Parameter	Default	Description
ALB Subnet 1	*<Requires input>*	The subnet ID of a public subnet in your existing VPC for the first ALB
ALB Subnet 2	*<Requires input>*	The subnet ID of a public subnet in your existing VPC for the second ALB. Note that this subnet must be in a different AZ than the subnet for the first ALB.
Beacon Server Subnet 1	*<Requires input>*	The subnet ID of a subnet in your existing VPC for the first beacon web server
Beacon Server Subnet 2	*<Requires input>*	The subnet ID of a subnet in your existing VPC for the second beacon web server. Note that this subnet must be in a different AZ than the subnet for the first beacon web server.

6. Choose **Next**.
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create** to deploy the stack.

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

   The solution sends an email invitation to join the real-time dashboard.
10. In the email, follow the instructions to sign in to the dashboard.

Appendix D: Collection of Operational Metrics

This solution includes an option to send operational metrics data to AWS. We use this data to better understand how customers use this solution to improve the services and the products that we offer. When enabled, the following information is collected and sent to AWS each time the AWS Lambda function is invoked:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **Dashboard Views:** The number of times the dashboard is viewed
- **Web Events:** The number of web events processed

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, complete one of the following tasks:

a) Modify the AWS CloudFormation template mapping section as follows:

```
Solution:
   Data:
      SendAnonymousUsageData: "True"
```

to

```
Solution:
   Data:
      SendAnonymousUsageData: "False"
```

OR

b) After the solution has been launched, find the `ProcessMetricsFunction` function in the Lambda console and set the **SEND_ANONYMOUS_DATA** environment variable to `False`.

Source Code

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change
March 2018	Initial release
December 2018	Added information about the Amazon CloudFront distribution for the static website hosted in the Amazon S3 bucket
October 2019	Upgraded the solution to Python 3.7

Notices