Real-Time Analytics with Spark Streaming

AWS Implementation Guide

Vijay Satish

Daniel Pinheiro

February 2017

Last updated: April 2020

Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the Real-Time Analytics with Spark Streaming solution in the Amazon Web Services (AWS) Cloud. It includes links to AWS CloudFormation templates that launch, configure, and run the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects who have practical experience architecting in the AWS Cloud, building big data applications, and are familiar with Apache Spark.

Overview

Many Amazon Web Services (AWS) customers use batch data reports to gain strategic insight into long-term business trends, and a growing number of customers also require streaming data to obtain actionable insights from their data in real time. Batch data is collected over a period of time and processed in batches, and this data can provide snapshots of trends that shape decision-making. Streaming data is generated continuously from thousands of data sources and it can help companies proactively respond to changing conditions.

A *lambda architecture* (not to be confused with the AWS Lambda service) is one way to implement real-time and batch data processing in a single framework. The lambda architecture divides processing into three layers: the batch layer in which new data is appended to the master data set and stored as batch views, the serving layer in which batch views are indexed, and the speed layer in which the real-time data views are produced, continuously updated, and stored for read/write operations.

AWS provides many of the building blocks required to build a secure, flexible, cost-effective lambda architecture in the cloud.[1] These include Amazon Kinesis Data Streams, a platform for processing terabytes of streaming data, Amazon EMR, a service that distributes and processes data across dynamically scalable Amazon Elastic Compute Cloud (Amazon EC2) instances, and Amazon Simple Storage Service (Amazon S3), a secure and durable object store. Customers can combine these AWS services with Apache Spark Streaming, for fault-

[1] For more information on implementing a lambda architecture on AWS, see Lambda Architecture for Batch and Real-Time Processing on AWS.

tolerant stream processing of live-data streams, and Spark SQL, which allows Spark code to execute relational queries, to build a single architecture to process real-time and batch data.

The Real-Time Analytics with Spark Streaming solution is an AWS-provided reference implementation that automatically provisions and configures the AWS services necessary to start processing real-time and batch data in minutes. The solution is designed to work with customers' Spark Streaming applications, and also includes a demo application and data producer to create an example environment. The solution also leverages Apache Zeppelin, a web-based notebook for interactive data analytics, to enable customers to visualize both their real-time and batch data.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of the date of publication, the cost for running this solution with default settings in the US East (N. Virginia) Region is listed in Table 1. This includes charges for Amazon Kinesis Data Streams, Amazon EC2, and Amazon EMR. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Table 1: Cost breakdown

AWS Service	Resource Count and Type	Total Cost/Hour
Amazon Kinesis Data Streams	2 shards	$0.03
Amazon EC2	1 t2.medium instance (2 if using the demo template)	$0.05 $0.10
NAT Gateways	2 (one per availability zone)	$0.09
Amazon EMR	3 r5.xlarge instances	$0.76

Note: The solution creates a customer master key (CMK) in AWS Key Management Service (AWS KMS) that is used to encrypt data at rest. Rotation is automatically enabled, so each new key version raises the cost of the CMK by $1/month (rotation happens once a year).

This pricing does not reflect variable charges incurred from Amazon S3, Amazon CloudWatch, AWS KMS, data transfer fees, or the cost of Amazon DynamoDB. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Apache Spark Streaming, Apache Spark SQL, and Apache Zeppelin are open source. There is no additional cost to use these tools.

Architecture Overview

Deploying this solution with the **default parameters** builds the following environment in the AWS Cloud.

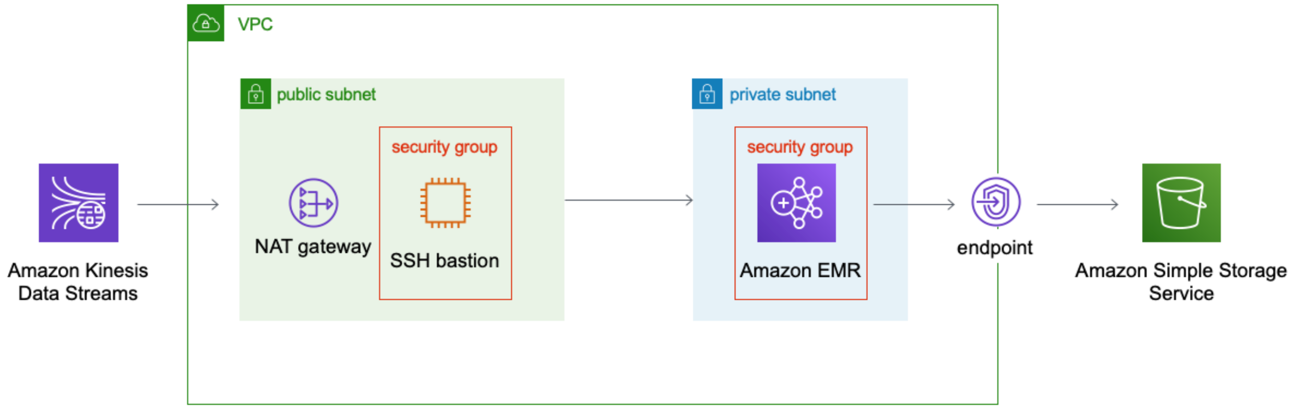

Figure 1: Real-Time Analytics with Spark Streaming default architecture

The AWS CloudFormation template deploys an Amazon Kinesis, an Amazon Virtual Private Cloud (Amazon VPC) network with one public and one private subnet, a NAT gateway, a bastion host, an Amazon EMR cluster, and a VPC endpoint to an Amazon S3 bucket.

The private subnet contains an Amazon EMR cluster with Apache Zeppelin. The public subnet contains a NAT gateway to connect Amazon Kinesis Streams to the Amazon EMR cluster, and a bastion host that provides SSH access to the Amazon EMR cluster.

The Real-Time Analytics solution is designed to allow you to use your own application, but it also includes a demo application that you can deploy for testing purposes. For more information, see Appendix A.

After the Spark Streaming application processes the data, it stores the data in an Amazon S3 bucket.

Implementation Considerations

Application Requirements

The Real-Time Analytics solution requires a working Spark Streaming application written in Java, Scala, or Python. We recommend that you use the latest version of Apache Spark for your application. You can choose to deploy your application as a JAR file or a JSON file (see the next section for details).

Processing Engine

When you deploy the solution, you choose the processing engine for your custom Spark Streaming application: a JAR file or an Apache Zeppelin notebook JSON file.

Application JAR

If you choose to package your custom Spark Streaming application as a JAR file, the solution requires a Spark Submit command to launch your application on the Amazon EMR cluster. You can choose to upload a file with the submit script, or you can enter the command directly in the AWS CloudFormation template parameter when you launch the solution.

Apache Zeppelin

If you choose to use Zeppelin, you will upload your custom Spark Streaming application as a `notebook.json` file from your local machine or a URL you specify. The solution automatically creates the dependencies and configurations to visualize your real-time and batch data.

Demo Application

The solution includes an additional AWS CloudFormation template (`real-time-analytics-spark-streaming-demo.template`) that deploys a demo application for testing purposes.

If you choose to run the demo application, this solution deploys a sample Spark Streaming application on the Amazon EMR cluster, a sample Amazon Kinesis stream, and a sample data producer that sends sample data to your Amazon Kinesis stream. The demo application is packaged as a JAR file. For more information, see Appendix A.

Single Application Deployment

The solution is designed to work with only one Spark Streaming application at a time. If you want to change applications, you must first stop the running application and then deploy the solution again with a new application. This also applies if you deploy the demo application:

you must stop the running demo application before you can deploy the demo application JSON file, a custom JAR file or a custom JSON file.

If you want to upload an application file that uses the same name as an application from a previous deployment (for example, a new version of a JSON template), you must clear the application name from the Amazon DynamoDB table before you deploy solution. See Tracking Amazon Kinesis Streams Application State in the Amazon Kinesis Streams Developer Guide for more information.

EMR Web Interfaces

When you launch an Amazon EMR cluster in a public subnet, the master node of the cluster has a public DNS which allows you to create an SSH tunnel and securely access the Amazon EMR web interfaces. Because this solution deploys the Amazon EMR cluster in a *private* subnet, the master node will not have a public DNS for secure SSH access. To allow you to access the Amazon EMR web interfaces, this solution deploys a bastion host with a public IP address. You must configure dynamic port forwarding to connect to the bastion host. For more information, see View Web Interfaces Hosted on Amazon EMR Clusters in the Amazon EMR Management Guide.

Memory-Optimized Instance Types

We recommend memory-optimized Amazon Elastic Compute Cloud (Amazon EC2) instance types for Apache Spark workloads because Spark attempts to process as much data in memory as possible. By default, this solution deploys an r5.xlarge instance for the Amazon EMR cluster nodes to deliver optimal performance.

Real-Time Data Visualization

For users who choose Zeppelin as their processing engine for this solution, Zeppelin will display visualizations of your real-time data as it flows. These visualizations can be shared or published to external dashboards. For more information, go to the Apache Zeppelin website.

AWS CloudFormation Templates

This solution uses AWS CloudFormation to automate the deployment of Real-Time Analytics with Spark Streaming on the AWS Cloud. It includes the following CloudFormation template, which you can download before deployment:

View template

real-time-analytics-spark-streaming.template: Use this template to launch the Real-Time Analytics solution and all associated components. The default configuration deploys an Amazon VPC network, a bastion host, a

VPC endpoint, an Amazon Kinesis stream, an Amazon EMR cluster, and an Amazon S3 bucket.

View template

real-time-analytics-spark-streaming-demo.template: Use this template to launch the Real-Time Analytics solution with the demo application. This configuration deploys an Amazon VPC network, a bastion host, a VPC endpoint, an Amazon EMR cluster, an Amazon S3 bucket, a demo Spark Streaming application, a sample Amazon Kinesis stream, and a sample data producer.

Automated Deployment

Before you launch the automated deployment, please review the implementation considerations and prerequisites discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the Real-Time Analytics solution into your account.

Time to deploy: Approximately 15-20 minutes

Prerequisites

Review the application requirements and processing options in Implementation Considerations.

- Before you deploy the solution, you must upload your working Spark Streaming application to an Amazon Simple Storage Service (Amazon S3) bucket. If you are using a Spark Submit script to launch your custom application, you must have a *spark-submit.sh* file with the Spark Submit command in an Amazon S3 bucket.
- Remember that you can only deploy one running application with a unique name at a time with this solution (see Single Application Deployment for detailed information).
- Before you deploy the solution, you must enable the Amazon EMR web interfaces to view Apache Zeppelin, the Spark History UI, and the Resource Manager. For more information, see EMR Web Interfaces.
- You must also configure dynamic port forwarding to connect to the bastion host to securely access the Amazon EMR web interfaces. For more information, please see View Web Interfaces Hosted on Amazon EMR Clusters in the Amazon EMR Management Guide.

What We'll Cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Launch the stack

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters.
- Review the other template parameters, and adjust if necessary.

Step 2. Stop a Running Application

- Navigate to the EMR cluster's Resource Manager.
- Stop the running application.

Step 1. Launch the Stack

The automated AWS CloudFormation templates deploy Real-Time Analytics with Spark Streaming on the AWS Cloud using either your own Spark Streaming application or the AWS-provided demo application.

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

1. Log in to the AWS Management Console and click the button to the right to launch the solution or demo application AWS CloudFormation template.
   You can also download the template as a starting point for your own implementation.

   Launch Solution

   Launch Demo Solution

2. The template is launched in the US East (N. Virginia) Region by default. To launch the Real-Time Analytics solution in a different AWS Region, use the region selector in the console navigation bar.
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 Real-Time Analytics 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
Key Name	*<Requires input>*	Public and private key pair, which allows you to connect securely to the bastion host. When you created an AWS account, this is the key pair you created in your preferred AWS Region.
Remote Access CIDR	*<Requires input>*	The IP address range that can be used to SSH to the bastion host
Availability Zones	*<Requires input>*	The list of Availability Zones to use for the Amazon VPC subnets
Number of AZs	`2`	The number of Availability Zones to use in your VPC **Note:** This number must match the number of selections you chose for the **Availability Zone** parameter.
VPC CIDR	`10.0.0.0/16`	The VPC CIDR block
Private Subnet 1A CIDR	`10.0.0.0/19`	The CIDR block for the private subnet located in AZ1
Private Subnet 2A CIDR	`10.0.32.0/19`	The CIDR block for the private subnet located in AZ2
Public Subnet 1 CIDR	`10.0.128.0/20`	The CIDR block for the public DMZ subnet located in AZ1
Public Subnet 2 CIDR	`10.0.144.0/20`	The CIDR block for the public DMZ subnet located in AZ2
Kinesis Stream	`default-data-stream`	The name of the source Amazon Kinesis stream the template will create
Shard Count	`2`	The number of shards for your Amazon Kinesis stream
Master	`r5.xlarge`	EMR master node Amazon EC2 instance type
Core	`r5.xlarge`	EMR core node Amazon EC2 instance type
Artifact Bucket	*<Requires input>*	Amazon S3 bucket where your application artifacts will be stored. For example, `my-artif-bucket`
Submit Mode	`AppJar`	The processing engine of the Spark Streaming application. Choose the appropriate processing engine (`Zeppelin` for JSON templates; `AppJar` for JAR files). **Note:** This parameter will be set to `DemoApp` if you are using the demo template.
Type	None	The submit type of the Spark Streaming application. You can specify a submit script or a submit command. **Note:** Use this parameter only if you choose `AppJar` as your **Submit Mode**. This parameter will not show if you are using the demo template.

Parameter	Default	Description
Script	`<Optional input>`	The Amazon S3 bucket where your script with the Spark submit command is stored. For example, `s3://{bucket_location/spark_submit.sh}` **Note:** Use this parameter only if you choose `AppJar` as your **Submit Mode**. If you choose `Command` as your **Submit Type**, leave this parameter blank. This parameter will not show if you are using the demo template.
Command	`<Optional input>`	A comma-delimited Spark submit command. For example, `--deploy-mode,{cluster/client},--class {className},` --master,{yarn/local[?]},{s3://AppLocation/AppJar}, {Appname},{StreamName},{OutputLoc} **Note:** Use this parameter only if you choose `AppJar` as your **Submit Mode**. If you choose `Script` as your **Submit Type**, leave this parameter blank. This parameter will not show if you are using the demo template.

6. Select **Next.**
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the boxes acknowledging that the template will create AWS Identity and Access Management (IAM) resources and might require the `CAPABILITY_AUTO_EXPAND` capability.
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 15-20 minutes.

> **Note:** This solution includes two AWS Lambda functions that run only during initial configuration or when resources are updated or deleted.
>
> When running this solution, you will see both Lambda functions in the AWS Lambda console. Do not delete the functions as they are necessary to manage associated resources.

Step 2. Stop a Running Application

The solution is designed to work with only one Spark Streaming application at a time. If you want to change applications, you must first stop the running application and then deploy the solution again with a new application.

> **Note:** To view the EMR Resource Manager web interface, you must enable the EMR web interfaces. For more information, see EMR Web Interfaces.

1. On the Amazon EMR console, select the cluster name.
2. On the EMR cluster details page, for **Connections**, choose **Resource Manager**.
3. On the resource manager, select the application **ID**.
4. On the application details page, select **Kill Application**.
5. Select **OK**.

> **Note:** Another option to stop a running application is to use the YARN command line (this approach does not require port forwarding). You must SSH into the Master node (via bastion) and run the following command: `sudo yarn application -kill` *<application-ID>*.

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 Groups

The security groups created in this solution are designed to control and isolate network traffic between the Amazon Kinesis sample data producer, the Amazon EMR cluster and the bastion host. We recommend that you review the security groups and further restrict access as needed once the deployment is up and running.

Security at Rest for Amazon EMR

Security configurations are Amazon EMR templates that can be used to configure data encryption, Kerberos authentication, and Amazon Simple Storage Service (Amazon S3) authorization for EMRFS. The solution creates a security configuration that enables:

- At-rest encryption for EMRFS data in Amazon S3 with Amazon S3-managed encryption keys (SSE – S3)
- At-rest encryption for local disks (Amazon Elastic Block Store root device and storage volumes) with an AWS Key Management Service customer master key (CMK)

However, the solution-created configuration does not enable in-transit encryption because this setting requires either a PEM file or a custom certificate-provider JAR file. You can use a trusted certification authority (CA) to issue certificates. For information about security certificates, see Providing Certificates for Encrypting Data in Transit with Amazon EMR Encryption in the *Amazon EMR Management Guide*.

Additional Resources

AWS services

- AWS CloudFormation
- Amazon DynamoDB
- Amazon EMR
- Amazon CloudWatch
- Amazon Kinesis
- Amazon S3
- Amazon VPC
- AWS KMS

Associated AWS Whitepaper

- Lambda Architecture for Batch and Real-Time Processing on AWS with Spark Streaming and Spark SQL

Appendix A: Demo Application

The Real-Time Analytics with Spark Streaming solution includes a demo application for testing purposes. Deploying this solution with the demo application builds the following environment in the AWS Cloud.

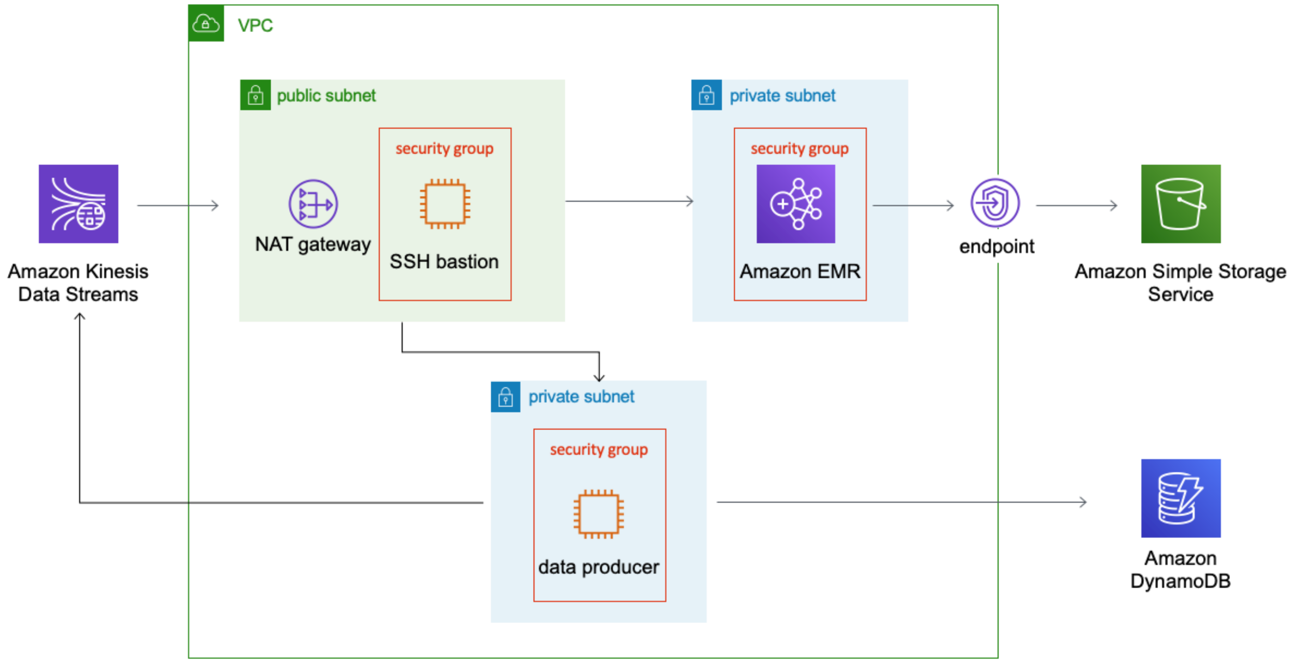

Figure 2: Real-Time Analytics with Spark Streaming architecture with demo application

When you use the demo application, the solution deploys an additional private subnet with a sample data producer and uploads the demo application code to your existing Amazon Simple Storage Service (Amazon S3) bucket. The demo application sends sample data through the NAT gateway to Amazon Kinesis Data Streams. In this architecture, the bastion host provides SSH access to the Amazon EMR cluster and the sample data producer, and the filtered data is stored in Amazon S3.

By default, the data producer will only run for a few seconds (creating one file on Amazon S3). To run the producer manually, take the following steps to replace the necessary values including *<mykeypair>*, *<bastion-host-dns>*, *<kinesis-producer-IP>* address, *<default-data-stream>*, and *<your-AWS-Region>*.

```
# Copy pem file to bastion host using SCP
scp -i "<mykeypair>.pem" <mykeypair>.pem ec2-user@<bastion-host-dns>:<mykeypair>.pem

# Login to bastion host
ssh -i "<mykeypair>.pem" ec2-user@<bastion-host-dns>

# Login to Kinesis Producer
ssh -i "<mykeypair>.pem" ec2-user@<kinesis-producer-IP>

# Run producer for 600 seconds (10 minutes)
```

```
sudo java -jar /home/ec2-user/kinesis-producer.jar <default-data-
stream> <your-AWS-Region> 600
```

Appendix B: Known issues

When deleting the solution stack, the Amazon EMR cluster does not delete the following security groups (since they reference one another): `ElasticMapReduce-Slave-Private`, `ElasticMapReduce-ServiceAccess`, and `ElasticMapReduce-Master-Private`. This will cause the VPC stack deletion to fail, since a VPC cannot be deleted while there are still available security groups.

To remove all inbound and outbound rules for a security group, use the AWS Command Line Interface (AWS CLI) terminal and enter the following commands, which must be executed for each group listed above. Using this method requires jq, a lightweight command-line JSON processor, to be installed.

```
SECURITY_GROUP_ID=<ID-placeholder>
aws ec2 describe-security-groups --group-ids $SECURITY_GROUP_ID --
output json | jq '.SecurityGroups[0].IpPermissions' >
IpPermissions.json

aws ec2 describe-security-groups --group-ids $SECURITY_GROUP_ID --
output json | jq '.SecurityGroups[0].IpPermissionsEgress' >
IpPermissionsEgress.json

aws ec2 revoke-security-group-ingress --group-id $SECURITY_GROUP_ID
--ip-permissions file://IpPermissions.json

aws ec2 revoke-security-group-egress --group-id $SECURITY_GROUP_ID -
-ip-permissions file://IpPermissionsEgress.json
```

Appendix C: Collection of Operational Metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution to improve the services and products that we offer. When enabled, the following information is collected and sent to AWS during the initial stack creation:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each Real-Time Analytics with Spark Streaming deployment

- **Timestamp:** Data-collection timestamp
- **EMR Data:** Type of instances and count of the number of instances
  Example data:

```
{"Master": "1", "InstanceType": "r5.xlarge", "CoreInstance": "2",
"InstanceType": "r5.xlarge", "Region": "us-east-1"}
```

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 the following task.

Modify the AWS CloudFormation template mapping section as follows:

```
AnonymousData:
    SendAnonymousData:
      Data: Yes
```

to

```
AnonymousData:
    SendAnonymousData:
      Data: No
```

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
February 2017	Initial release
March 2017	Corrected demo application architecture diagram
April 2020	Added demo producer application, upgraded the solution to Python 3.8, bug fixes

Notices

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