Amazon Personalize Optimizer Using Amazon Pinpoint Events

Implementation Guide

Amazon Personalize Optimizer Using Amazon Pinpoint Events: Implementation Guide

Table of Contents

Welcome 1
Overview 2
    Cost 2
    Architecture 2
Components 4
    Amazon Personalize Campaigns 4
    Amazon Personalize Event Data 4
    AWS Lambda 4
Considerations 5
    Dependencies 5
    Regional Deployment 5
Template 6
Deployment 7
    Prerequisites 7
        Amazon Personalize Campaign 7
        Amazon Pinpoint Project 8
        Recommender Model in Amazon Pinpoint 8
    What We'll Cover 8
    Step 1. Launch the Stack 9
    Step 2. Update the Processing Method AWS Lambda Function Used in Amazon Pinpoint Recommender Model Configuration 11
    Step 3. Verify Updated Amazon Personalize Campaign 13
Security 15
    Encryption 15
Resources 16
Appendix A: Add Permission Using JSON to the Processing Method AWS Lambda Function 17
Appendix B: Update the Event Scoring Model after Deployment 18
Appendix C: Update Athena Query to Match Amazon Personalize Interaction Schema 19
Appendix D: Operational Metrics 20
Source Code 21
Revisions 22
    22

Create integrations between Amazon Personalize campaigns and Amazon Pinpoint projects

Publication date: *March 2020 (last update (p. 22): June 2020)*

This implementation guide discusses architectural considerations and configuration steps for deploying the Amazon Personalize Optimizer Using Amazon Pinpoint Events solution in the Amazon Web Services (AWS) Cloud. It includes a link 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 architects, developers, DevOps, and marketing technology professionals who have practical experience architecting in the AWS Cloud.

Overview

Today's consumer expects a high level of personalization in order to continue engaging with a company. To address this challenge, customers use curated messaging, rather than a generalized marketing campaign, to lower churn rates, increase consumer interaction, and drive higher conversion rates. Many customers are turning to machine learning to deliver personalized product recommendations or promotions at scale. In order to maintain an effective and relevant model, you need high volumes of recent behavioral data. Without an automated data pipeline, you have to perform inefficient manual retraining or risk using an outdated model. You depend on an engineering team to build and maintain automated pipelines, delaying the time to deploy their models.

To help you leverage your existing Amazon Personalize campaigns and Amazon Pinpoint projects to build a data pipeline easily, AWS offers the Amazon Personalize Optimizer Using Amazon Pinpoint Events solution. This solution empowers you to train and publish models quickly without support from an engineer. You can define the frequency and the type of data used to retrain your models. Using an automated retraining loop frees you to build new models and keep them relevant for your marketers.

This guide provides infrastructure and configuration information for planning and deploying the Amazon Personalize Optimizer Using Amazon Pinpoint Events solution in the AWS Cloud. This automated reference implementation deploys a cost-effective, end-to-end solution for creating a scheduled daily batch to gather select consumer data and then use the data to retrain a personalization model. The personalization model can then be used with Amazon Pinpoint to personalize messaging content.

Cost

You are responsible for the cost of the AWS services used while running this reference deployment. As of the date of publication, the cost for running this solution with default settings in the US East (N. Virginia) Region is approximately **$140 a month**. The cost estimate includes the use of Amazon Pinpoint and AWS Lambda to retrain your Amazon Personalize model using AWS Glue, AWS Lambda, Amazon Athena, Amazon CloudWatch, and Amazon Kinesis. This estimate assumes 100,000 customer endpoints stored in Amazon Pinpoint with up to 1 GB of additional customer data in Amazon Simple Storage Service (Amazon S3).

> **Note**
> This solution requires that you have already deployed an Amazon Personalize campaign and Amazon Pinpoint project in your environment. The cost estimate does not include the costs of running your existing Amazon Personalize, Amazon Pinpoint, and AWS Lambda functions already deployed in your environment.

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

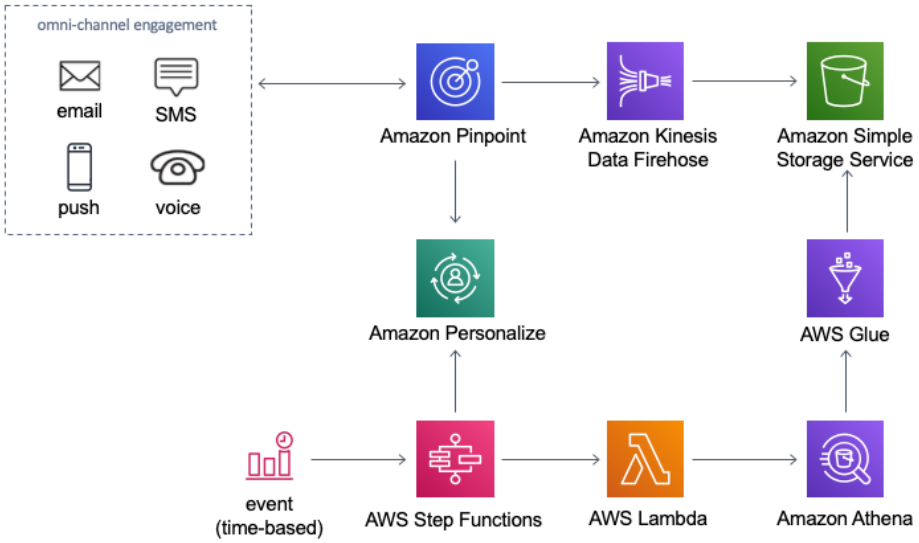

Figure 1: Amazon Personalize Optimizer Using Amazon Pinpoint Events architecture

The AWS CloudFormation template configures the Amazon Pinpoint event stream on an existing Amazon Pinpoint project to use Amazon Kinesis Data Firehose to store event data in Amazon Simple Storage Service (Amazon S3). The Amazon S3 data schema is stored in an AWS Glue Data Catalog enabling data queries.

There is a constant flow of real-time data moving from Amazon Pinpoint through Kinesis Data Firehose and being stored in Amazon S3. When you send a campaign, Amazon Pinpoint connects with Amazon Personalize to retrieve a personalized recommendation based on the Amazon Pinpoint recommender model configuration for each user identified in the campaign.

The AWS CloudFormation template also deploys a daily batch process orchestrated by AWS Step Functions. The process begins when an Amazon CloudWatch time-based event triggers a series of AWS Lambda functions that use an Amazon Athena query to query customer data stored in Amazon S3. The query result is then used to retrain Amazon Personalize by providing new interaction data from the Amazon Pinpoint event data.

Solution Components

Amazon Personalize Campaigns

The Amazon Personalize campaign must be configured with an interactions schema that includes `EVENT_TYPE` and `EVENT_VALUE` which will correspond to Amazon Pinpoint streaming events.

Amazon Personalize Event Data

Amazon Personalize event data is gathered and stored in Amazon S3 (Amazon S3). Event data includes campaign sends, SMS sends, push sends, email sends, email opens, and email clicks. You can configure the event data that is collected and specify additional data points, as needed.

AWS Lambda

This solution uses AWS Lambda functions to perform Amazon Pinpoint exports, query endpoint and event data stored in Amazon S3, import new Amazon Personalize interaction dataset data, create a new Amazon Personalize solution version, and update the Amazon Personalize campaign to use the new solution version.

Design Considerations

Dependencies

This solution requires that you have already deployed an Amazon Personalize campaign and Amazon Pinpoint project in your environment. If you do not already have an existing Amazon Personalize campaign or Amazon Pinpoint project, you can create them. For more information about creating an Amazon Personalize campaign, see Creating a Solution in the *Amazon Personalize Developer Guide*. For instructions to create an Amazon Pinpoint project, see Amazon Pinpoint Project (p. 8) in this guide.

In addition, an Amazon Personalize campaign must be configured as an Amazon Pinpoint recommender model. For information, see Recommender Model in Amazon Pinpoint (p. 8) in this guide.

Regional Deployment

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

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the Amazon Personalize Optimizer Using Amazon Pinpoint Events solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment.

amazon-personalize-optimizer-using-amazon-pinpoint-events.template: Use this template to launch the solution and all associated components. The default configuration deploys Amazon S3, Amazon Pinpoint, Amazon Kinesis, AWS Glue, AWS Step Functions, AWS Lambda, Amazon Athena, Amazon CloudWatch, and AWS Identity and Access Management, but you can also customize the template based on your specific network needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture, configuration, storage security, 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.

Time to deploy: Approximately 3 minutes

Prerequisites

Verify that you have the following setups before deploying the solution:

- A configured and deployed Amazon Personalize campaign
- A configured Amazon Pinpoint project
- A recommender model in Amazon Pinpoint

Amazon Personalize Campaign

A fully deployed Amazon Personalize campaign with the following interaction schema is required.

```
{
    "type": "record",
    "name": "Interactions",
    "namespace": "com.amazonaws.personalize.schema",
    "fields": [
        {
            "name": "USER_ID",
            "type": "string"
        },
        {
            "name": "ITEM_ID",
            "type": "string"
        },
        {
            "name": "EVENT_TYPE",
            "type": "string"
        },
        {
            "name": "EVENT_VALUE",
            "type": "float"
        },
        {
            "name": "TIMESTAMP",
            "type": "long"
        }
    ],
    "version": "1.0"
}
```

If your interaction schema is different, you must update the Athena query so that it generates a CSV file that matches your schema. For instructions to edit the Athena query, see Appendix C (p. 19).

Amazon Pinpoint Project

An Amazon Pinpoint project in the same AWS Region where you plan to deploy this solution must be configured. If one does not already exist, take the following steps to create one.

1. Navigate to the Amazon Pinpoint console.
2. In the **All projects** section, choose **Create a project**.
3. In the **Create a project** dialog box, enter a **Project name** and choose **Create**.
4. On the **Configure features** page, review the URL in the address bar and record the project identifier, which is the hexadecimal string between **/apps/** and **/setup**. For example, 1xxxx8xx055x40x89xxx77830xxx9x27.

   Note
   The project identifier is needed when you launch the solution. Enter the project identifier as the value for the **Amazon Pinpoint Project ID** parameter. If you've used the Amazon Pinpoint API, you may have seen references to *applications*. In Amazon Pinpoint, a *project* is the same as an *application*. This solution uses the term *Project ID* instead of *Application ID*.

The project is automatically saved.

Recommender Model in Amazon Pinpoint

Verify that you have a recommender model available in Amazon Pinpoint. Take the following steps if you need to create one.

1. Navigate to the Amazon Pinpoint console.
2. Choose your Amazon Pinpoint project and then, from the left navigation pane, select **Machine learning models**.
3. Choose **Add recommender model** and set up the model following the instructions in Setting Up a Recommender Model in Amazon Pinpoint in the *Amazon Pinpoint User Guide*.

   Note
   Update the AWS Lambda function that runs the processing method to emit custom events to Amazon Pinpoint. This Lambda function tracks the recommended items to end users using the Amazon Pinpoint campaign. See Step 2 (p. 11) for the instructions to make this update.

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 (p. 9)

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack Name**, **Amazon Personalize Campaign ARN**, **Amazon Personalize Solution ARN**, **Amazon Personalize Interaction Dataset ARN**, and **Amazon Pinpoint Project ID**.
- Review the other template parameters and adjust, if necessary.

Step 2. Update the Processing Method AWS Lambda Function Used in Amazon Pinpoint Recommender Model Configuration (p. 11)

- Update the AWS Lambda function that is currently configured in your Amazon Pinpoint recommender model configuration.

Step 3. Verify Updated Amazon Personalize Campaign (p. 13)

- Verify that a new Amazon Personalize solution version is created and the campaign is updated.

Step 1. Launch the Stack

This automated AWS CloudFormation template deploys the Amazon Personalize Optimizer Using Amazon Pinpoint Events solution in the AWS Cloud.

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 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 Pinpoint and Amazon Personalize services, which are currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region where Amazon Pinpoint and Amazon Personalize are both available. For the most current availability by Region, see AWS service offerings by Region.
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.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following parameters.

Parameter	Default	Description
Amazon Personalize Campaign ARN	*<Requires input>*	The full ARN from the Amazon Personalize campaign that is currently configured in your account.
Amazon Personalize Solution ARN	*<Requires input>*	The full ARN from the Amazon Personalize solution that is used by the campaign.
Amazon Personalize Interaction Dataset ARN	*<Requires input>*	The full ARN from the Amazon Personalize interaction dataset used by the campaign.

Parameter	Default	Description
Pinpoint Project ID	*<Requires input>*	An Amazon Pinpoint Project ID that has the Amazon Personalize recommender configured.
Pinpoint Event Types	`_campaign.opened_notification, _email.open, _email.click, _email.unsubscribe`	Name of the events emitted by the Amazon Pinpoint Event Stream that will be the EVENT_TYPE values when the interaction dataset is generated. For information about updating this parameter post-deployment, see Appendix B (p. 18).
Pinpoint Event Type Interaction Values	`100, 50, 100, -200`	Event values that correspond to the Amazon Pinpoint Event Types that will be the `EVENT_VALUE` values when the interaction dataset is generated. For information about updating this parameter post-deployment, see Appendix B (p. 18).
Interaction History Date Scope	`-1`	Number of days to look back in the query to build out the new interaction dataset. A value of -1 will query all historical event data in Amazon S3.

6. Choose **Next**.
7. On the **Configure stack 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 stack** 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 approximately 3 minutes.

   Note
   In addition to the series of AWS Lambda functions used to query data in Amazon S3, an additional Lambda function, `CustomResourceHelper`, runs only during initial configuration or when resources are updated or deleted.
   When running this solution, you will see all the Lambda functions in the AWS Management Console, but only the ones used to query data in Amazon S3 are regularly active. Do not delete the `CustomResourceHelper` function as it is necessary to manage associated resources.

Step 2. Update the Processing Method AWS Lambda Function Used in Amazon Pinpoint Recommender Model Configuration

Update the AWS Lambda function that is used in the Amazon Pinpoint recommender model configuration to allow the solution to track the item(s) that were recommended by Amazon Personalize when querying for new interaction data. Alter the AWS Lambda function to emit a custom event to Amazon Pinpoint for tracking purposes and to add permission to the role that runs the Lambda function.

1. Navigate to the AWS Lambda console.
2. In the navigation pane, choose **Functions**.
3. Select the AWS Lambda function that is currently configured in the Amazon Pinpoint recommender model configuration settings as the processing method for your campaign.
4. In the **Function code** section, edit the existing code using the examples below to call the Amazon Pinpoint Event API to emit a custom event with `EventType` of `_custom.recommender`.

   Note
   Configure the Amazon Pinpoint event in the sample below to emit the correct `personalize_user_id` value that is being used in your Amazon Personalize integration with Amazon Pinpoint. The integration allows for either the `EndpointID` or the `UserID`. For additional information about Amazon Pinpoint recommender model configuration settings, see Setting Up a Recommender Model in Amazon Pinpoint in the *Amazon Pinpoint User Guide*.

   The following code is an example NodeJS implementation

```
const AWS = require('aws-sdk');
const pinpoint = new AWS.Pinpoint();

exports.handler = async (event) => {

    await emitRecommenderEvents(event);

    //… normal processing ...

    return event.Endpoints;
}

const emitRecommenderEvents = async function(event) {

    const batchEventItems = {};
    Object.keys(event.Endpoints).forEach((endpointId, ind) => {

        batchEventItems[endpointId] = {
            Endpoint:{},
            Events: {}
        };

        const endpoint = event.Endpoints[endpointId];
        endpoint.RecommendationItems.forEach((itemId) => {

            const eventKey = `${endpointId}_rec_${itemId}`;
            batchEventItems[endpointId].Events[eventKey] = {
                EventType: '_custom.recommender',
                Timestamp: new Date().toISOString(),
                Attributes: {
                    personalize_user_id: endpointId || endpoint.User.UserId, // TODO
 Determine UserId or EndpointId
```

```
                    item_id: itemId
                }
            };
        });
    });

    console.log(JSON.stringify(batchEventItems, 1));

    return pinpoint.putEvents({
        ApplicationId: event.ApplicationId,
        EventsRequest: {
            BatchItem: batchEventItems
        }
    }).promise();
};
```

The following code is an example Python implementation.

```
# Example Python Lambda Function
import json
import boto3
import datetime

pinpoint = boto3.client('pinpoint')

def lambda_handler(event, context):

    emit_recommender_events(event)

    # ... normal processing ...

    return event['Endpoints']

def emit_recommender_events(event):

    batchEventItems = {}

    for endpointId in event['Endpoints'] :
        batchEventItems[endpointId] = {
            'Endpoint': {},
            'Events': {}
        }

        endpoint = event['Endpoints'][endpointId]

        for itemId in endpoint['RecommendationItems'] :
            eventKey = endpointId + '_rec_' + itemId

            batchEventItems[endpointId]['Events'][eventKey] = {
                'EventType': '_custom.recommender',
                'Timestamp': datetime.datetime.now().isoformat(),
                'Attributes': {
                    'personalize_user_id': endpointId || endpoint['User']['UserId'], #
 TODO Determine UserId or EndpointId,
                    'campaign_id': event['CampaignId'],
                    'item_id': itemId
                }
            }


    pinpoint.put_events(
        ApplicationId=event['ApplicationId'],
        EventsRequest={
```

```
            'BatchItem': batchEventItems
        }
    )
```

5. Scroll to the **Execution role** section and, under **Existing role**, choose the **View the *<function-name>* role on the IAM console** text link. The IAM console opens in a new browser tab.
6. On the **Summary** page, choose **Attach policies**.
7. On the **Add permissions** page, choose **Create policy**. The **Create policy** page opens in a new browser tab.
8. In the **Visual editor** tab, **Service** option, choose the **Choose a service** text link and in the search field enter **Pinpoint**.
9. Select **Pinpoint** from the search results.
10. For **Actions**, choose the drop-down arrow next to **Write** to expand the list of options and select **PutEvents**.
11. Choose the drop-down arrow next to **Resources** to access the list of options and then select **All resources**.
12. Choose **Review policy**.
13. In the **Name** field, enter a *<policy-name>* and, optionally, enter a **Description**.

    Note
    For the *<policy-name>*, create a name for this role that is meaningful to you, for example, **EmitCustomEventsPolicy**.

14. Choose **Create policy**.
15. Return to the **Add permissions** page, and choose the refresh button to ensure that your newly created policy is added to the list, and search for that policy.
16. Select your policy and choose **Attach policy**.

You return to the IAM policy page and receive a confirmation that the policy was attached. Optionally, you can enter the JSON code to add permission to the role that runs the Lambda function. For instructions, see Appendix A (p. 17)

Step 3. Verify Updated Amazon Personalize Campaign

The AWS Step Functions state machine queries the newly imported interaction dataset, retrains Amazon Personalize to create a new version of the personalized solution, and updates the Amazon Personalize campaign. You can verify that Amazon Personalize was updated with a new version of the personalized solution by taking the following steps.

Note
The state machine is configured to run every morning at 2:00 AM GMT. The state machine does not make changes or updates to Amazon Personalize until new interaction data is received. When new data is available, the state machine queries and retrains Amazon Personalize as needed.

1. Navigate to the Amazon Personalize console.
2. Select your dataset group in the **Dataset groups** list.
3. In the navigation pane, choose **Datasets** then under **Dataset type**, locate **User-item interaction** and choose the corresponding name under **Dataset name**.
4. Identify the latest import job and verify that the **Import job status** is **Active**.

5. In the navigation pane, choose **Solutions and recipes** and select the solution that is in use by the campaign.
6. Identify the latest **Solution version** and verify that the **Solution version status** is **Active**. Note the **Solution version ID**. You will match the ID in the following steps when you navigate to the **Campaigns** page.
7. In the navigation pane, choose **Campaigns**, select your campaign and then select the **Details** tab.
8. Verify that the campaign's **Solution version ID** matches the ID listed in the **Solution** screen.

As you execute Amazon Pinpoint campaigns based on the Amazon Personalize Recommender model integration, this solution will track user interactions and provide new data to Amazon Personalize daily. This allows your recommendations to improve over time as Amazon Personalize learns with the goal of driving higher end user engagements.

This solution can be extended to consume and score custom events from other applications based on the item being recommended. For example, track the recommended item from Amazon Personalize to purchase events to use for retraining Amazon Personalize. You can create custom events and submit them to Amazon Pinpoint as detailed in Step 2 (p. 11).

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.

Encryption

By default, the Amazon Simple Storage Service (Amazon S3) buckets this solution creates are encrypted with S3-SSE AES 256 encryption. The Amazon Kinesis Data Firehose delivery streams are not encrypted. For end-to-end encryption, we recommend restricting access to the solution's delivery streams. For more information, see Controlling Access with Amazon Kinesis Data Firehose.

Additional Resources

AWS services

- AWS CloudFormation
- Amazon Pinpoint
- Amazon Personalize
- AWS Step Functions
- AWS Lambda
- Amazon Simple Storage Service
- Amazon Athena
- AWS Glue
- Amazon Kinesis
- Amazon CloudWatch

Appendix A: Add Permission Using JSON to the Processing Method AWS Lambda Function

You can use JSON to add permission to the role that runs the Processing Method AWS Lambda function by taking the following steps.

1. Scroll to the **Execution role** section and, under **Existing role**, choose the **View the *<function-name>* role on the IAM console** text link. The IAM console opens in a new browser tab.
2. On the **Summary** page, choose **Attach policies**.
3. On the **Add permissions** page, choose **Create policy**. The **Create policy** page opens in a new browser tab.
4. In the **JSON** tab, enter the following code snippet.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EmitCustomEvents",
            "Effect": "Allow",
            "Action": "mobiletargeting:PutEvents",
            "Resource": "*"
        }
    ]
}
```

5. Choose **Review policy**.
6. In the **Name** field, enter a *<policy-name>* and, optionally, enter a **Description**.

   Note
   For the *<policy-name>*, create a name for this role that is meaningful to you, for example, **EmitCustomEventsPolicy**.
7. Choose **Create policy**.
8. Return to the **Add permissions** page, choose the refresh button to ensure that your newly created policy is added to the list, and search for that policy.
9. Select your policy and choose **Attach policy**.

You return to the IAM policy page and receive a confirmation that the policy was attached.

Appendix B: Update the Event Scoring Model after Deployment

During deployment of the AWS CloudFormation template, two parameters, `Pinpoint Event Types` and `Pinpoint Event Type Interaction Values`, are specified that builds an interaction event scoring model. These values correspond to the Amazon Personalize interaction dataset schema attributes `EVENT_TYPE` and `EVENT_VALUE` respectively. These AWS CloudFormation template parameters are converted to a `values.csv` file and stored in an Amazon S3 bucket. The CSV file is used in the Amazon Athena query that generates the Amazon Personalize interactions.

The following example `values.csv` file shows sample values.

```
EVENT_TYPE,EVENT_VALUE
_campaign.opened_notification,100
_email.open,50
_email.click,100
_email.unsubscribe,-200
```

After deployment, you may have business rule changes requiring an update to this event scoring model, including new event types or updates to event values. Modify the `values.csv` file. To locate the `values.csv` file, take the following steps.

1. Navigate to the AWS CloudFormation console.
2. On the **Stacks** page, choose the stack created for this solution.
3. On the stack details page, choose the **Outputs** tab and, under the **Key**Key column, locate `PathToScoringModel`. This key identifies the path to the S3 bucket that stores the `values.csv` file.

Modify this file to add new events generated by Amazon Pinpoint (including events sent to Amazon Pinpoint using the Events API) and update event values.

> **Note**
> You can add or remove rows and change the values of the existing rows. However, you cannot add or remove columns.

Appendix C: Update Athena Query to Match Amazon Personalize Interaction Schema

This solution assumes a particular schema for your Amazon Personalize interaction dataset. If your schema is different, the solution's AWS Lambda function that imports the data will fail to run. To correct the issue, update or rewrite the Amazon Athena query to retrieve the correct values that matches your schema.

The following code snippet shows the `SELECT` statement in the Amazon Athena query that is deployed with this solution. Running this query generates a CSV file with the following fields: `USER_ID`, `ITEM_ID`, `EVENT_TYPE`, `EVENT_VALUE`, and `TIMESTAMP`. These fields must match the interaction dataset schema.

```
SELECT
   r.personalize_user_id as USER_ID,
   r.item_id AS ITEM_ID,
   b.event_type AS EVENT_TYPE,
   v.EVENT_VALUE,
   CAST(to_unixtime(b.arrival_timestamp) AS BIGINT) AS TIMESTAMP
FROM endpoint_export a
INNER JOIN recs r
   ON a.id = r.endpoint_id
INNER JOIN evs b
   ON a.id = b.endpoint_id AND r.campaign_id = b.campaign_id
INNER JOIN event_value v
   ON b.event_type = v.event_type
```

To update the Amazon Athena query, first create a new query. Next, in the AWS Lambda function `QueryAugmentStartLambda`, update the environmental variable `NAMED_QUERY` with the query ID of the Lambda function.

For example, if your interaction schema contains only the required fields: `USER_ID`, `ITEM_ID`, and `TIMESTAMP`, then the query's final `SELECT` statement may be updated as shown in the following code snippet.

```
SELECT
   r.personalize_user_id as USER_ID,
   r.item_id AS ITEM_ID,
   CAST(to_unixtime(b.arrival_timestamp) AS BIGINT) AS TIMESTAMP
FROM endpoint_export a
INNER JOIN recs r
   ON a.id = r.endpoint_id
INNER JOIN evs b
   ON a.id = b.endpoint_id AND r.campaign_id = b.campaign_id
INNER JOIN event_value v
   ON b.event_type = v.event_type
```

Appendix D: 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 and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp

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, modify the AWS CloudFormation template mapping section as follows:

```
MetricsMap:
    Send-Data:
      SendAnonymousData: "Yes"
```

to

```
MetricsMap:
    Send-Data:
      SendAnonymousData: "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	
March 2020	Initial release	
June 2020	Updated the solution's IAM policy to align with Amazon Personalize IAM policy updates	

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.