PINPOINT JOURNEY EVENTS ATTRIBUTION V2

Ioannou Katidis, Pavlos
AWS DUE Specialist Solutions Architect

Pinpoint – Journey Events Attribution

Contents

Disclaimer

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.

Solution Summary

Background

Marketing campaigns come with associated cost. Marketers use KPIs to assess their effectiveness and calculate ROI. While Pinpoint provides certain metrics such as email open/read, it does not allow marketers to attribute any custom events to Campaigns or Journeys. The latter results to a marketing spend without the possibility of ROI calculation and not knowing what works well for customers.

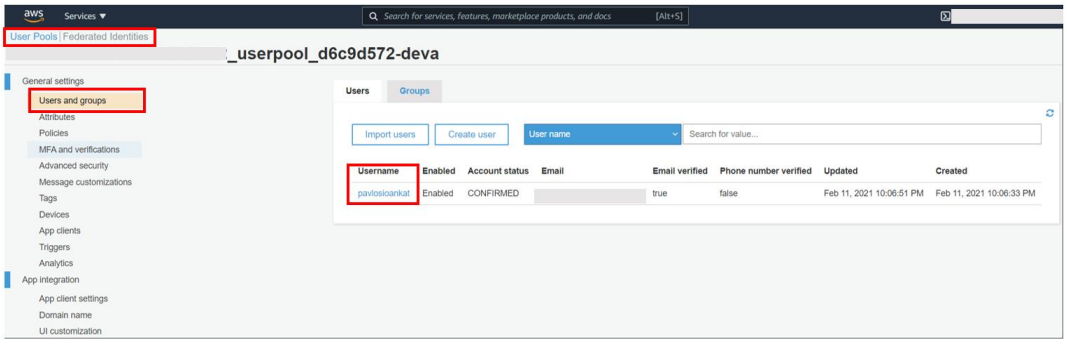
Ideal state

When the marketer is launching an email campaign, they should have a KPI, while the email itself should contain a CTA with a goal aligned to the KPI above such as purchase, subscription or code activation. Email campaign goals are usually on site events and if the customer has clicked or read the email, then that customer's events should be attributed to that email campaign. In the end of an email campaign, the marketer should be able to assess its effectiveness and calculate its ROI.

Solution

The solution is enabling marketers to attribute Pinpoint custom events following a customer's interaction with an email, SMS or custom channel. Additionally marketers are able to define a lookback window on a Pinpoint application level. The solution is utilising Amazon Cognito for its user attributes' storage, Pinpoint Journeys, Lambda, DynamoDB and DynamoDB streams. The solution is applicable only for Pinpoint Journeys and the customer's custom events can be attributed only under one marketing campaign at a time. If a customer interacts with a new marketing campaign while they are already in one, then the new one will overwrite the old and any new custom events will be attributed to the new email campaign.

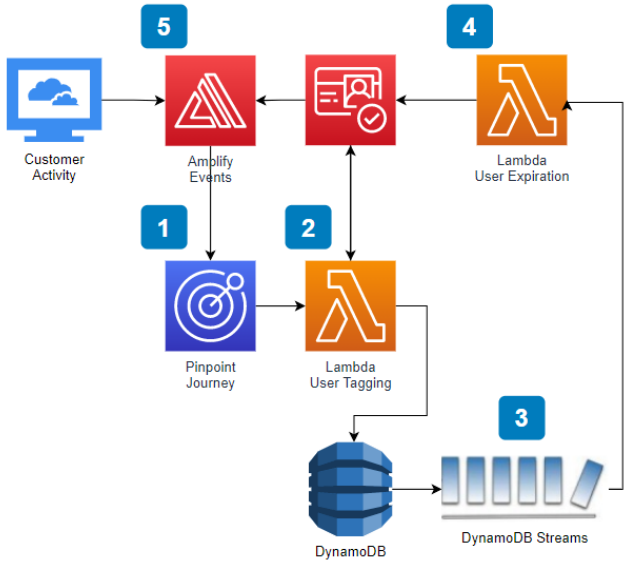
Considerations

1) You will need to install Amplify SDK for sending events to Pinpoint and Cognito for user management
2) All Pinpoint users should have a user attribute with the Cognito username
   a. 

3) Create a custom Cognito user attribute named “campaign”
4) All Pinpoint custom events sent with Amplify should include an event attribute named “campaign” = Cognito user attribute “campaign”
5) The solution is deployed on a Pinpoint application level
6) Applicable only for Pinpoint Journeys and requires a multivariate split step within the journey design
7) Push notification channel requires additional work
8) Supports only one campaign per user at a time and if a user enters to a second campaign then the new campaign will overwrite the old one
9) Expired DynamoDB records based on their TTL attributed are not removed immediately

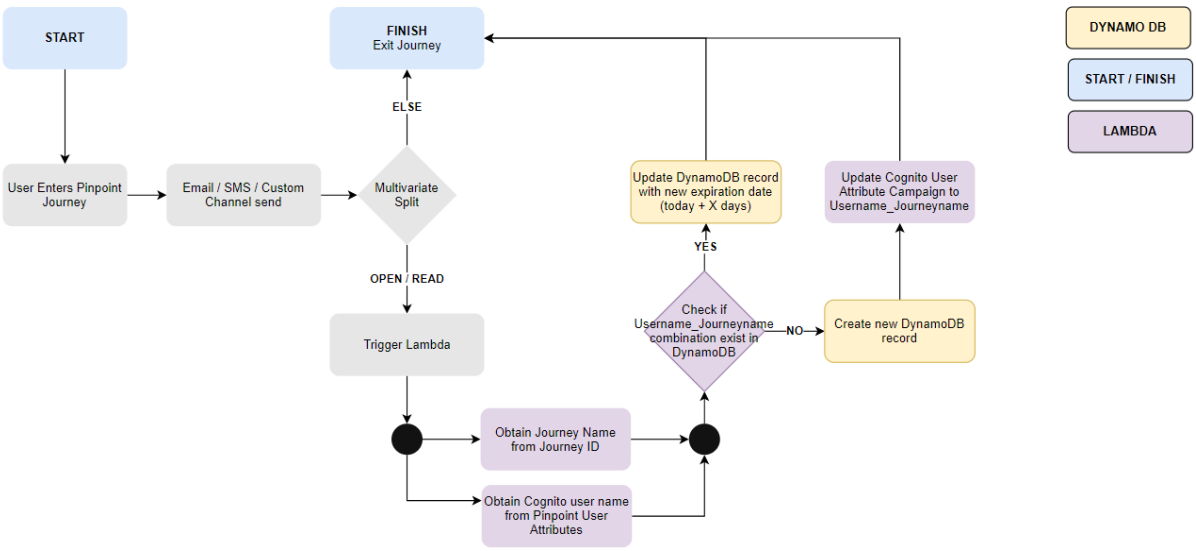
Solution Architecture & Business Logic

Solution Architecture



1. User enters a Pinpoint Journey. Depending the customer interaction with email or SMS, a Lambda will be triggered (step 2)
2. The Lambda creates a combination of Cognito user name and Pinpoint journey name in a DynamoDB table as well as an expiration date, which is defined as (today + X days) for that combination. Furthermore it updates a custom Cognito user attribute named campaign to the Cognito user name and Pinpoint journey name value
3. All DynamoDB records have a TTL attribute that equals the expiration date. When that expiration date is passed then that record is being removed from the table
4. A Lambda processes the DynamoDB streams and when the event = "REMOVE" it updates the custom Cognito attribute Campaign for that user to " "
5. Amplify will record all custom events with an attribute campaign = Cognito Custom Attribute Campaign. This requires extra implementation on client side, which is explained further in this documentation

Business logic for Lambda User Tagging



Business logic for Lambda User Expiration

DynamoDB streams emit a REMOVE event Expiration Lambda, which is processing these records will update the Cognito custom user attribute Campaign to blank.

Steps to implement the solution

Step 1 – Create AWS account & Pinpoint Project

If you have an AWS account and Pinpoint Project setup already please move to step 2

Create an AWS account

Create a Pinpoint project

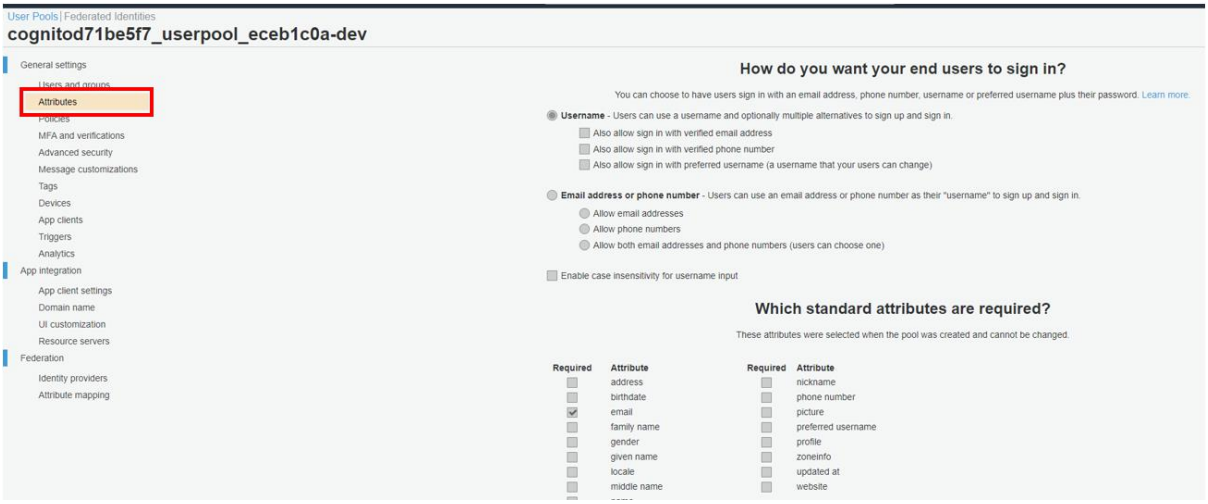
Step 2 – Create S3 bucket for Lambda code and upload the Zip files

Create an S3 bucket in the region that you have your Pinpoint projects and provide it a unique name

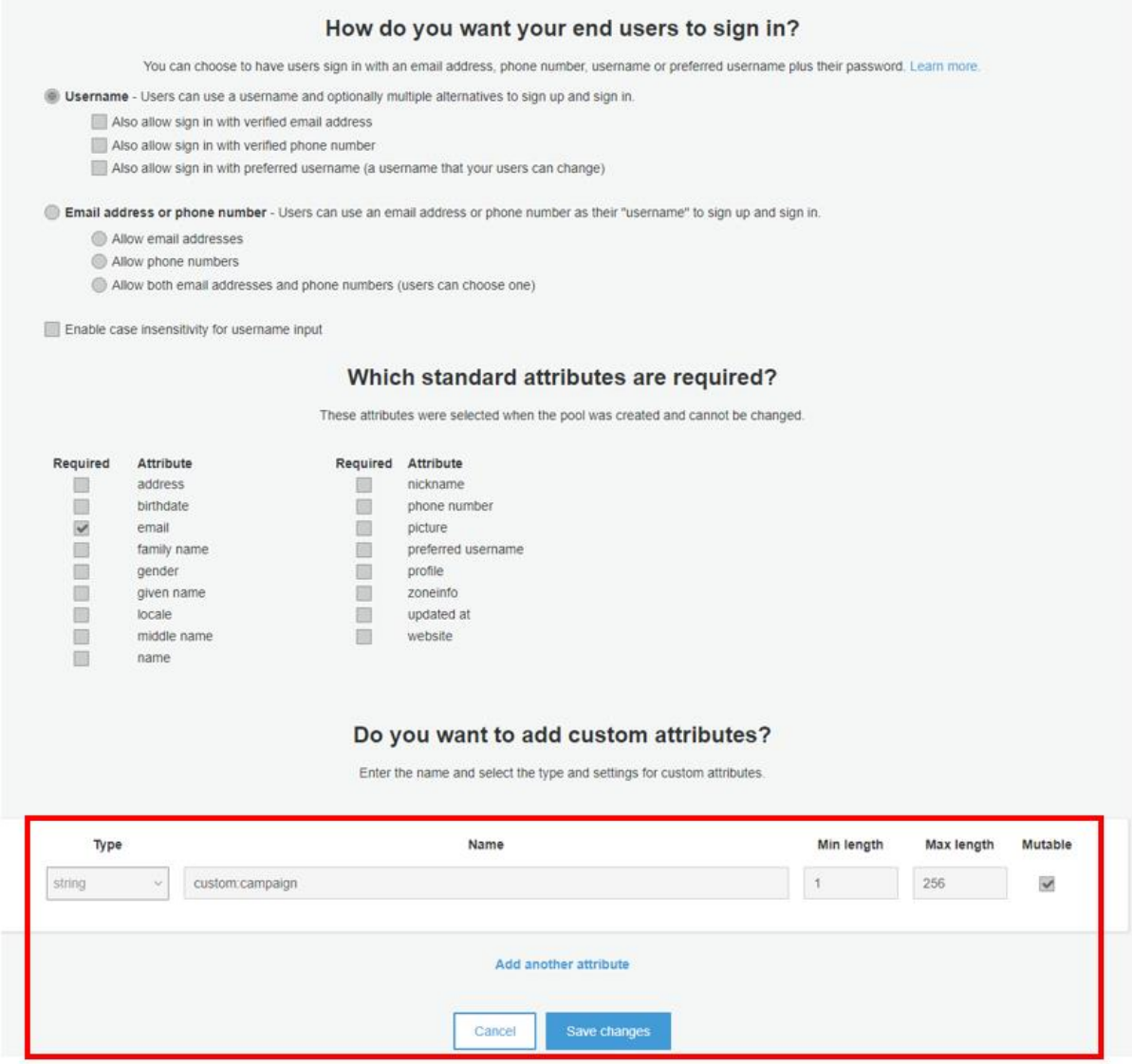
Upload in the root folder the 1 zip file: JourneyEventsAttribution_Tagging.zip

Step 3 – Create a Cognito Custom User Attribute

On the AWS console, navigate to the Amazon Cognito page, select “User Pools”, click on the user pool of the application you want to implement this solution and navigate to “Attibutes” from the list on the right



If you have already existing custom attributes, then click on “Add another attribute” otherwise click on “Add new” and fill it as in the screenshot below

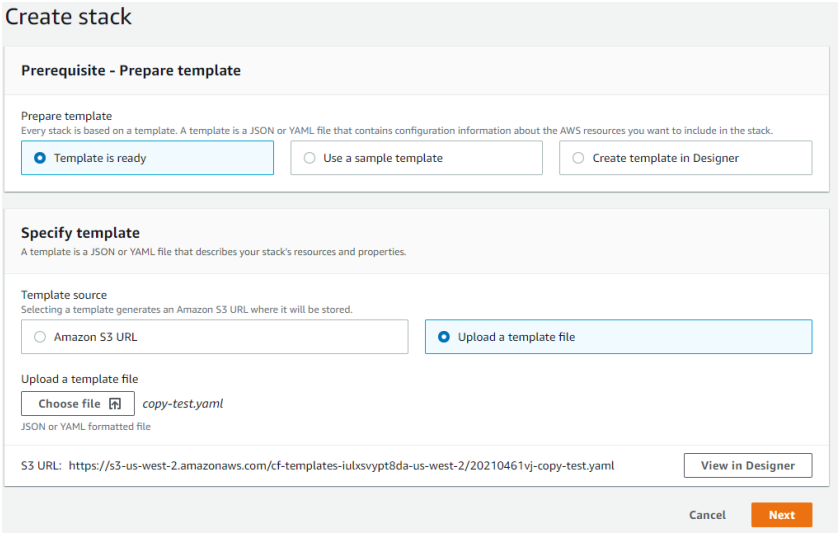


More information about Cognito User Attributes can be found here

Step 4 – Create Cloudformation Stack

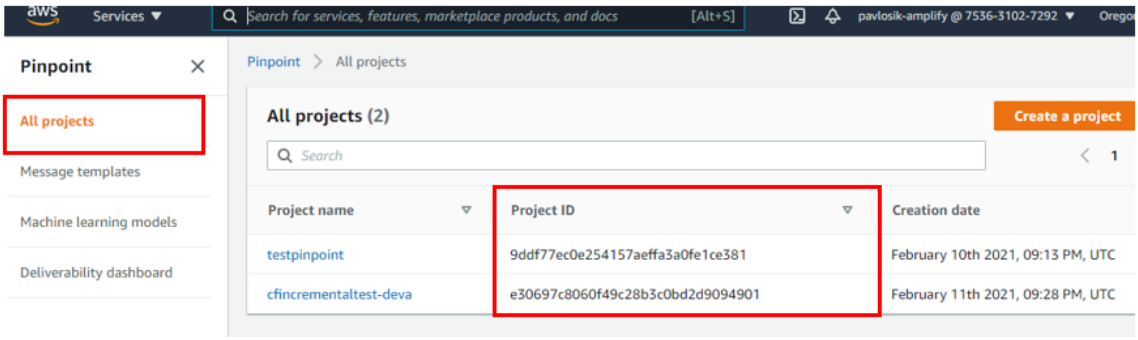
Navigate to Cloudfromation page in AWS console, click up right on “Create stack” and select the option “With new resources (standard)”

Leave the “Prerequisite – Prepare template” to “Template is ready” and for the “Specify template” option, select “Upload a template file”. On the same page, click on “Choose file”, browse to find the file “Pinpoint_Journey-Events-Attribution-V2.yaml” file and select it. Once the file is uploaded, click “Next”



See below information for each of the 6 fields under the section "Specify stack details":

1) **Stack name:** Provide a name of your preference for that Cloudformation stack
2) **CognitoUserPoolID:** Navigate to the AWS console and to the Cognito page. Select User Pools and click on the User Pool you are using for this project. From the list on the left click on General Settings and you should see the Pool Id as the first piece of information
3) **LambdaCodeBucketName:** Type the name of the S3 bucket from step 2
4) **LookbackWindow:** Type the number of days that you would like a customer's events to be attributed to a Journey post their interacted with it
5) **PinpointProjectId:** Copy paste the Pinpoint Project ID, which you can find on the Pinpoint console page under "All projects"
   a.



Once all fields completed, click "Next"

On the "Configure stack options" page, click "Next"

On the "Review [StackName]" page, check the checkbox "I acknowledge that AWS CloudFormation might create IAM resources." And then click on "Create stack"

Step 5 – Amend your client side Amplify code

This solution requires your app to have Amazon Cognito for user management.

For this implementation guide the code snippets used are for React JS web app.

In your code you should have already imported the Auth library

```
import { Auth } from 'aws-amplify';
```

Users should have access to the latest Cognito storage values, thus the BypassCache attribute of Cognito should be set to true

```
Auth.currentAuthenticatedUser({ bypassCache: true });
```

From the Auth.currentAuthenticatedUser you will need to get the Custom User Attribute defined in the previous step

```
const usname = () => Auth.currentAuthenticatedUser({
    bypassCache: true}).then(function(user){
    var campaign = user.attributes.["custom:campaign"];
    var listofitems = {'campaign':campaign};

    return listofitems})
.catch(err => console.log(err));
```

The campaign name obtained from Auth.current AuthenticatedUser should now be passed as a Pinpoint event attribute

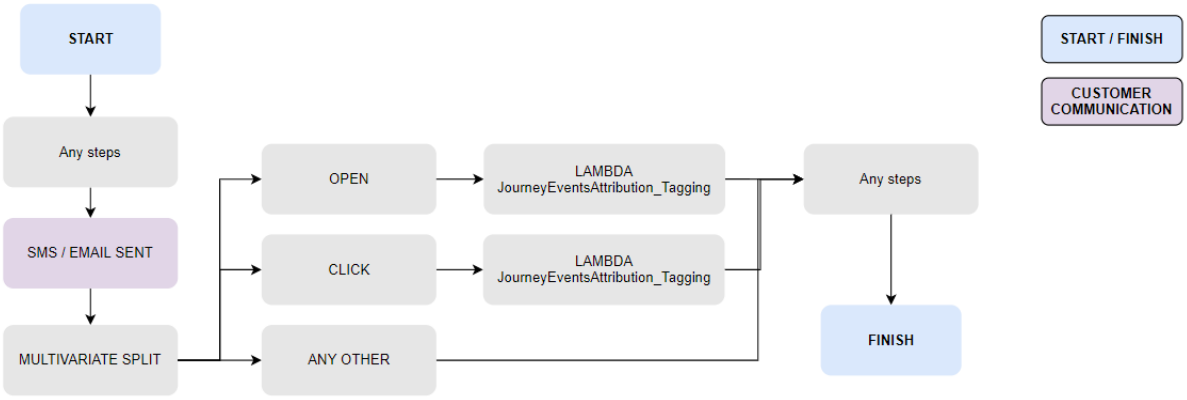
```
    const pinanalyticsred = () => usname().then(
      function(user){

      Analytics.record( {name: 'any-event' , 'Endpoint' : user.userm,
      attributes:{Campaign: user.campaign}});

      }).catch(err => console.log(err));
```

How to use the solution

Once the solution is deployed, your Pinpoint journeys should be structured as shown below. After customer interaction with an email or SMS, you should trigger the Lambda "JourneyEventsAttribution_Tagging". This will update the Cognito User Attribute Campaign to the Journey Name and any events taken post that will be attributed to that Journey. The lookback window you set in the CloudFormation, will define for how long that user's events will be attributed to this Journey.



If the campaign accepts multiple entries of the same user then the lookback window for that user-journey will be updated.

All user-journey records with the expiration date and TTL are stored in a DynamoDB. The DynamoDB TTL attribute ensures that expired records are being removed and through the DynamoDB streams a Lambda processes all records where event = "REMOVE" and updates the respective user, their Cognito User Attribute Campaign to " ".