Getting Started

ML Ops is gaining a lot of popularity. This example showcases a key piece you can use to construct your automation pipeline. As we can see in the following architecture diagram, you will be deploying an AWS Step Funciton Workflow containing AWS Lambda functions that call Amazon S3, Amazon Personalize, and Amazon SNS APIs.

This package contains the source code of a Step Functions pipeline that is able to perform multiple actions within **Amazon Personalize**, including the following:

- Dataset Group creation
- Datasets creation and import
- Solution creation
- Solution version creation
- Campaign creation

Once the steps are completed, the step functions notifies the users of its completion through the use of an SNS topic.

The below diagram describes the architecture of the solution:

Architecture Diagram

The below diagram showcases the StepFunction workflow definition:

stepfunction definition

Prerequisites

Installing AWS SAM

The AWS Serverless Application Model (SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML. During deployment, SAM transforms and expands the SAM syntax into AWS CloudFormation syntax, enabling you to build serverless applications faster.

Install the AWS SAM CLI (https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-install.html). This will install the necessary tools to build, deploy, and locally test your project. In this particular example we will be using AWS SAM to build and deploy only. For additional information please visit our documentation (https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html).

Note: We have pre-installed SAM CLI in this notebooks through a cloudformation life cycle policy config

Let's check what version of SAM we have installed

In [26]:

```
!sam --version
```

```
SAM CLI, version 1.22.0
```

Directory Structure

Let's take a look at directory structure

We have a couple artifacts that we will be using to build our MLOps pipeline.

In [27]:

```
!ls /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops
```

```
deploy.sh  domain  lambdas  LICENSE  poc_data  script.py  shared  template.yaml
```

`ml_ops/domain`

- This directory contains the configuration file and sample data based on the domain. In this example we are going to be using the Retail domain

In [28]:

```
!ls /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/domain
```

```
CPG  Media  Retail
```

`ml_ops/lambdas`

- This directory contains all the code that will be going into the lambda functions, these lambda functions will become a step inside the stepfunctions state machine we will deploy

In [29]:

```
!ls /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas
```

```
create-campaign      event-tracker            s3lambda
create-dataset       import-data              wait-delete-campaign
create-datasetgroup  list-campaigns           wait-delete-dataset
create-solution      list-datasets            wait-delete-datasetgroup
delete-campaign      list-solutions           wait-delete-solution
delete-dataset       list-solution-versions   wait-delete-tracker
delete-datasetgroup  list-trackers            wait-solution-version
delete-solution      notify
delete-tracker       notify-delete
```

`ml_ops/template.yaml`

- This is our SAM template that will deploy the automation into our account, here we are printing just the head

In [30]:

```
!head /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/template.yaml
```

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >

Globals:
  Function:
    Timeout: 300

Resources:
```

Deploying

In order to deploy the project you will need to run the following commands:

In [31]:

```
!cd /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/; sam build
```

```
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/s3lambda
runtime: python3.7 metadata: {} functions: ['S3Lambda']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/notify r
untime: python3.7 metadata: {} functions: ['Notify']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/notify-d
elete runtime: python3.7 metadata: {} functions: ['NotifyDelete']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/create-d
atasetgroup runtime: python3.7 metadata: {} functions: ['CreateDatasetGroup']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/create-d
ataset runtime: python3.7 metadata: {} functions: ['CreateDataset']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/import-d
ata runtime: python3.7 metadata: {} functions: ['ImportData']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/create-s
olution runtime: python3.7 metadata: {} functions: ['CreateSolution']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/wait-sol
ution-version runtime: python3.7 metadata: {} functions: ['WaitSolutionVersion']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/create-c
ampaign runtime: python3.7 metadata: {} functions: ['CreateCampaign']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/delete-c
ampaign runtime: python3.7 metadata: {} functions: ['DeleteCampaign']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/delete-d
ataset runtime: python3.7 metadata: {} functions: ['DeleteDataset']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/delete-d
atasetgroup runtime: python3.7 metadata: {} functions: ['DeleteDatasetGroup']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/delete-s
olution runtime: python3.7 metadata: {} functions: ['DeleteSolution']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/delete-t
racker runtime: python3.7 metadata: {} functions: ['DeleteTracker']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/list-cam
paigns runtime: python3.7 metadata: {} functions: ['ListCampaigns']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/list-dat
asets runtime: python3.7 metadata: {} functions: ['ListDatasets']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/list-sol
ution-versions runtime: python3.7 metadata: {} functions: ['ListSolutionVersions']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/list-sol
utions runtime: python3.7 metadata: {} functions: ['ListSolutions']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/list-tra
ckers runtime: python3.7 metadata: {} functions: ['ListTrackers']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/wait-del
ete-campaign runtime: python3.7 metadata: {} functions: ['WaitDeleteCampaign']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/wait-del
ete-dataset runtime: python3.7 metadata: {} functions: ['WaitDeleteDataset']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/wait-del
ete-datasetgroup runtime: python3.7 metadata: {} functions: ['WaitDeleteDatasetgroup']
```

```
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/wait-del
ete-solution runtime: python3.7 metadata: {} functions: ['WaitDeleteSolution']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/wait-del
ete-tracker runtime: python3.7 metadata: {} functions: ['WaitDeleteTracker']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Building codeuri: /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/lambdas/event-tr
acker runtime: python3.7 metadata: {} functions: ['AttachEventTracker']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Invoke Function: sam local invoke
[*] Deploy: sam deploy --guided
```

In [32]:

```
!cd /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/; sam deploy --template-file template.yaml
 --stack-name notebook-automation --capabilities CAPABILITY_IAM --s3-bucket $(aws cloudformation describe-stack-resources -
-stack-name AmazonPersonalizeImmersionDay --logical-resource-id SAMArtifactsBucket --query "StackResources[0].PhysicalResou
rceId" --output text)
```

```

        Deploying with following values
        ===============================
        Stack name                   : notebook-automation
        Region                       : us-east-1
        Confirm changeset            : False
        Deployment s3 bucket         : amazonpersonalizeimmersionday-samartifactsbucket-1izoghbemfa7p
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {}
        Signing Profiles             : {}

Initiating deployment
=====================

Waiting for changeset to be created..
Error: No changes to deploy. Stack notebook-automation is up to date
```

Uploading data

Let's get the bucket that our cloudformation deployed. We will be uploading our data to this bucket, plus the configuration file to trigger the automation

In [33]:

```
bucket = !aws cloudformation describe-stacks --stack-name notebook-automation --query "Stacks[0].Outputs[?OutputKey=='Input
BucketName'].OutputValue" --output text
bucket_name = bucket[0]
print(bucket_name)
```

```
notebook-automation-inputbucket-1g8dq180xs23x
```

Now that we have the bucket name, lets copy over our Media data so we can explore and upload to S3

In [34]:

```
!cp -R /home/ec2-user/SageMaker/amazon-personalize-immersion-day/automation/ml_ops/domain/Retail ./example
```

In [35]:

```
# Import Dependencies

import boto3
import json
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import time
import requests
import csv
import sys
import botocore
import uuid
from collections import defaultdict
import random
import numpy as np

from packaging import version
from botocore.exceptions import ClientError
from pathlib import Path

%matplotlib inline

# Setup Clients

personalize = boto3.client('personalize')
personalize_runtime = boto3.client('personalize-runtime')
personalize_events = boto3.client('personalize-events')

# We will upload our training data in these files:
raw_items_filename = "example/data/Items/items.csv"                # Do Not Change
raw_users_filename = "example/data/Users/users.csv"                # Do Not Change
raw_interactions_filename = "example/data/Interactions/interactions.csv"  # Do Not Change
items_filename = "items.csv"                # Do Not Change
users_filename = "users.csv"                # Do Not Change
interactions_filename = "interactions.csv"  # Do Not Change
```

In [36]:

```
interactions_df = pd.read_csv(raw_interactions_filename)
interactions_df.head()
```

Out[36]:

	ITEM_ID	USER_ID	EVENT_TYPE	TIMESTAMP	DISCOUNT
0	1def0093-96b2-4cc4-a022-071941f75b92	3156	ProductViewed	1591803788	No
1	1def0093-96b2-4cc4-a022-071941f75b92	3156	ProductViewed	1591803788	No
2	4df77d59-732e-4194-b9aa-7ad3878345e7	332	ProductViewed	1591803812	Yes
3	4df77d59-732e-4194-b9aa-7ad3878345e7	332	ProductViewed	1591803812	Yes
4	31b83eb4-bd8a-4b5a-87ff-f52abe6aa1f4	3981	ProductViewed	1591803830	Yes

There are 2 ways of uploading your datasets to S3:

1. Using the boto3 SDK
2. Using the CLI

In this example we are going to use the CLI command

In [37]:

```
!aws s3 sync ./example/data s3://$bucket_name
```

Starting the State Machine Execution

In order to execute the MLOps pipeline we need to provide a parameters file that will tell our state machine which names and configurations we want in our Amazon Personalize deployment.

We have prepared a parameters.json file, let's explore it

In [38]:

```
with open('example/params.json') as f:
  data = json.load(f)

print(json.dumps(data, indent=4, sort_keys=True))
```

```
{
    "campaigns": {
        "userPersonalizationCampaign": {
            "minProvisionedTPS": 1,
            "name": "userPersonalizationCampaign-APID-Retail-Automation"
        }
    },
    "datasetGroup": {
        "name": "notebook-automation"
    },
    "datasets": {
        "Interactions": {
            "name": "na-interactions-ds",
            "schema": {
                "fields": [
                    {
                        "name": "USER_ID",
                        "type": "string"
                    },
                    {
                        "name": "ITEM_ID",
                        "type": "string"
                    },
                    {
                        "name": "EVENT_TYPE",
                        "type": "string"
                    },
                    {
                        "name": "TIMESTAMP",
                        "type": "long"
                    }
                ],
                "name": "Interactions",
                "namespace": "com.amazonaws.personalize.schema",
                "type": "record",
                "version": "1.0"
            }
        },
        "Items": {
            "name": "na-items-ds",
            "schema": {
                "fields": [
                    {
```

```
                "name": "ITEM_ID",
                "type": "string"
            },
            {
                "categorical": true,
                "name": "CATEGORY",
                "type": "string"
            },
            {
                "categorical": true,
                "name": "STYLE",
                "type": "string"
            }
        ],
        "name": "Items",
        "namespace": "com.amazonaws.personalize.schema",
        "type": "record",
        "version": "1.0"
    }
},
"Users": {
    "name": "na-users-ds",
    "schema": {
        "fields": [
            {
                "name": "USER_ID",
                "type": "string"
            },
            {
                "name": "AGE",
                "type": "int"
            },
            {
                "categorical": true,
                "name": "GENDER",
                "type": "string"
            }
        ],
        "name": "Users",
        "namespace": "com.amazonaws.personalize.schema",
        "type": "record",
        "version": "1.0"
    }
```

```
        }
    },
    "eventTracker": {
        "name": "EventTracker-APID-Retail-Automation"
    },
    "solutions": {
        "personalizedRanking": {
            "name": "na-personalizedRankingCampaign",
            "recipeArn": "arn:aws:personalize:::recipe/aws-personalized-ranking"
        },
        "sims": {
            "name": "na-simsCampaign",
            "recipeArn": "arn:aws:personalize:::recipe/aws-sims"
        },
        "userPersonalization": {
            "name": "na-userPersonalizationCampaign",
            "recipeArn": "arn:aws:personalize:::recipe/aws-user-personalization"
        }
    }
}
```

This parameters file is set up to run at the beginning of this workshop. So let's modify a couple fields to make sure we are not overwritting our previous deployment

```
In [39]:
```

```
# Dataset Groups
data['datasetGroup']['name'] = 'notebook-automation'

# Datasets
data['datasets']['Interactions']['name'] = 'na-interactions-ds'
data['datasets']['Users']['name'] = 'na-users-ds'
data['datasets']['Items']['name'] = 'na-items-ds'

# Solutions

data['solutions']['personalizedRanking']['name'] = 'na-personalizedRankingCampaign'
data['solutions']['sims']['name'] = 'na-simsCampaign'
data['solutions']['userPersonalization']['name'] = 'na-userPersonalizationCampaign'

# Campaigns

data['campaigns']['personalizedRankingCampaign']['name'] = 'na-personalizedRankingCampaign'
data['campaigns']['simsCampaign']['name'] = 'na-simsCampaign'
data['campaigns']['userPersonalizationCampaign']['name'] = 'na-userPersonalizationCampaign'

# Event Tracker

data['eventTracker']['name'] = 'na-eventTracker'

print(json.dumps(data, indent=4, sort_keys=True))
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-39-e8bdecfb79d3> in <module>
     15 # Campaigns
     16 
---> 17 data['campaigns']['personalizedRankingCampaign']['name'] = 'na-personalizedRankingCampaign'
     18 data['campaigns']['simsCampaign']['name'] = 'na-simsCampaign'
     19 data['campaigns']['userPersonalizationCampaign']['name'] = 'na-userPersonalizationCampaign'

KeyError: 'personalizedRankingCampaign'
```

Updating and uploading your parameters file to S3

First let's write the file locally

In [ ]:

```
with open('example/params.json', 'w') as outfile:
    json.dump(data, outfile)
```

Now we can upload this file to S3, we are going to be using the CLI to do so

In [ ]:

```
!aws s3 cp ./example/params.json s3://$bucket_name
```

Validating the deployment

So far we have deployed the automation required lets take a look at the stepfunctions execution

In [ ]:

```
client = boto3.client('stepfunctions')
stateMachineArn = !aws cloudformation describe-stacks --stack-name notebook-automation --query "Stacks[0].Outputs[?OutputKe
y=='DeployStateMachineArn'].OutputValue" --output text
stateMachineArn= stateMachineArn[0]
```

In [ ]:

```
describe_response = client.describe_state_machine(
    stateMachineArn=stateMachineArn
)
print(json.dumps(describe_response, indent=4, sort_keys=True, default=str))
```

In [ ]:

```
executions_response = client.list_executions(
    stateMachineArn=stateMachineArn,
    statusFilter='SUCCEEDED',
    maxResults=2
)
print(json.dumps(executions_response, indent=4, sort_keys=True, default=str))
```

Let's look at the successful execution

Once your step functions are done executing, you can list the executions and describe them

In [ ]:

```
executions_response = client.list_executions(
    stateMachineArn=stateMachineArn,
    statusFilter='SUCCEEDED',
    maxResults=2
)
print(json.dumps(executions_response, indent=4, sort_keys=True, default=str))
```

In [ ]:

```
describe_executions_response = client.describe_execution(
    executionArn=executions_response['executions'][0]['executionArn']
)
print(json.dumps(describe_executions_response, indent=4, sort_keys=True, default=str))
```

Let's look at the input that was delivered to the State Machine

As we can see below, this is the input from our Parameters file we uploaded to S3. This input json was then passed to lambda functions in the state machine to utilize across Amazon Personalize APIs

In [ ]:

```
print(json.dumps(json.loads(describe_executions_response['input']), indent=4, sort_keys=True, default=str))
```

Let's look at the time stamps

As we can see below, this is the input from our Parameters file we uploaded to S3. This input json was then passed to lambda functions in the state machine to utilize across Amazon Personalize APIs

In [25]:

```
print("Start Date:")
print(json.dumps(describe_executions_response['startDate'], indent=4, sort_keys=True, default=str))
print("Stop Date:")
print(json.dumps(describe_executions_response['stopDate'], indent=4, sort_keys=True, default=str))
print("Elapsed Time: ")
elapsed_time = describe_executions_response['stopDate'] - describe_executions_response['startDate']
print(elapsed_time)
```

```
Start Date:

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-25-9b0bccebd776> in <module>
      1 print("Start Date:")
----> 2 print(json.dumps(describe_executions_response['startDate'], indent=4, sort_keys=True, default=str))
      3 print("Stop Date:")
      4 print(json.dumps(describe_executions_response['stopDate'], indent=4, sort_keys=True, default=str))
      5 print("Elapsed Time: ")

NameError: name 'describe_executions_response' is not defined
```

As we see above, the automation around an hour with fourty minutes.

If you are interested in deploying this example in your environment, visit our [Github Samples Page (https://github.com/aws-samples/amazon-personalize-samples/tree/master/next_steps/operations/ml_ops)](https://github.com/aws-samples/amazon-personalize-samples/tree/master/next_steps/operations/ml_ops) to download the latest codebase.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```