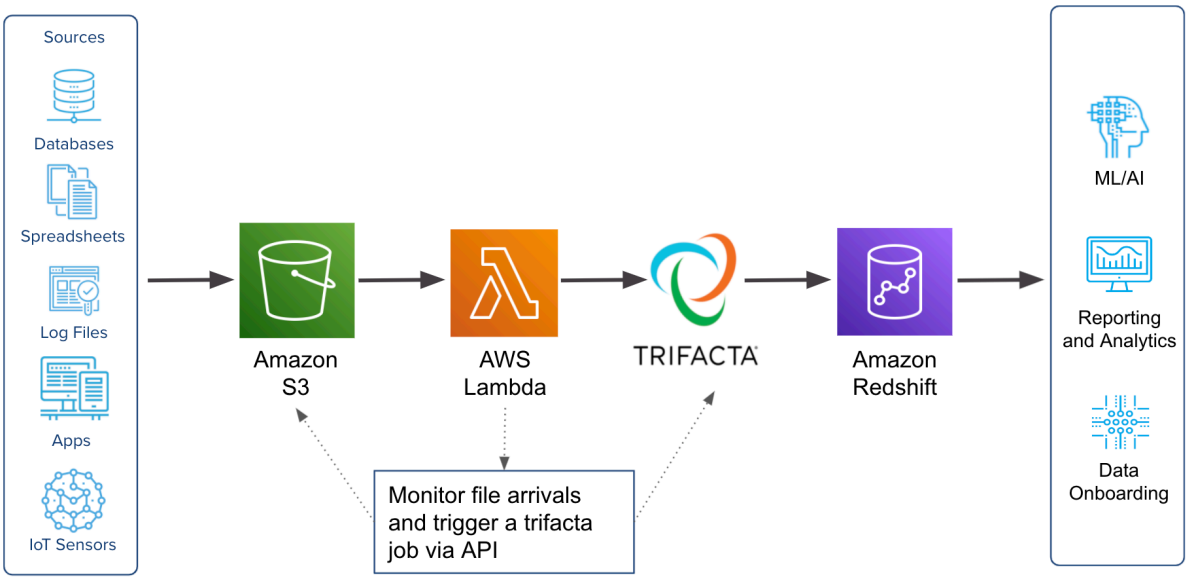
This article is a step-by-step guide to walk you through the process of triggering a Trifacta Job when a new file appears on the S3 base storage. We will be using AWS lambda functions to monitor a file arrival to S3 and call Trifacta APIs to run a Trifacta flow.

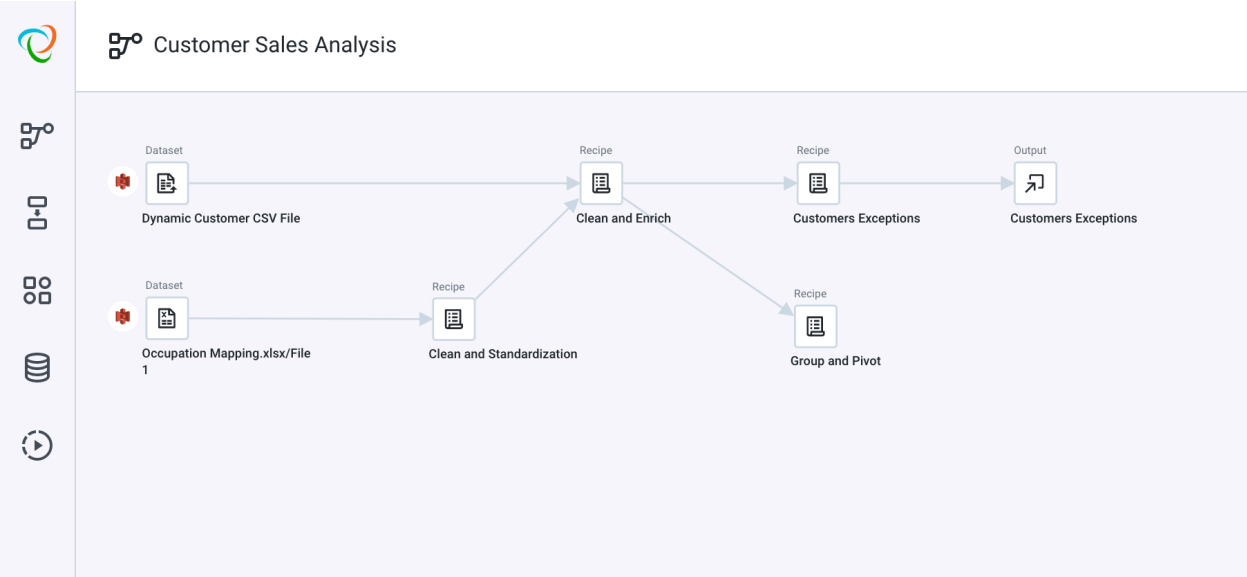


Step-1:

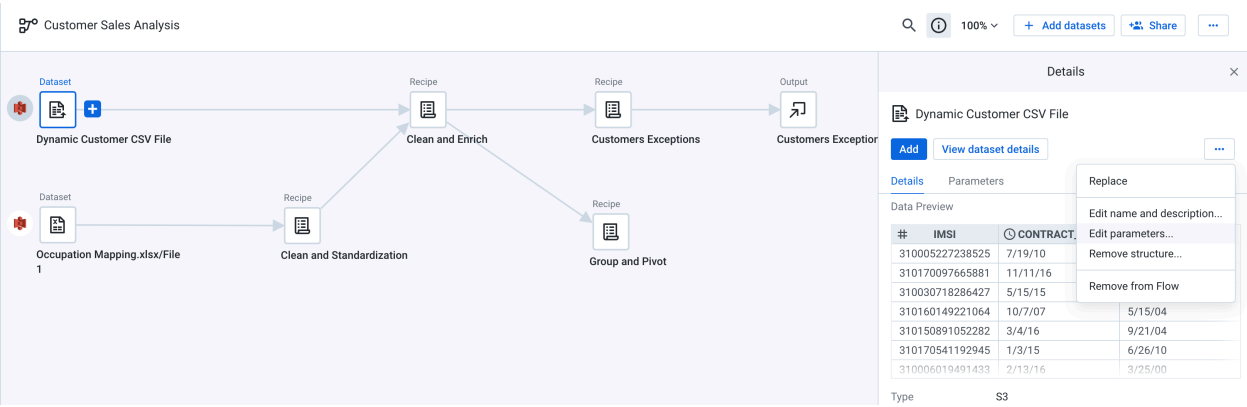
Create a Trifacta flow using an input data defined as a parameter. The available options are Date-time Parameter, Variable or Pattern Parameter. In the next few steps, you will see how we leverage AWS Lambda functions to get the new file-name on the source s3 bucket and invoke the respective Trifacta API call to trigger the required flow

The example flow attached has two input datasets.

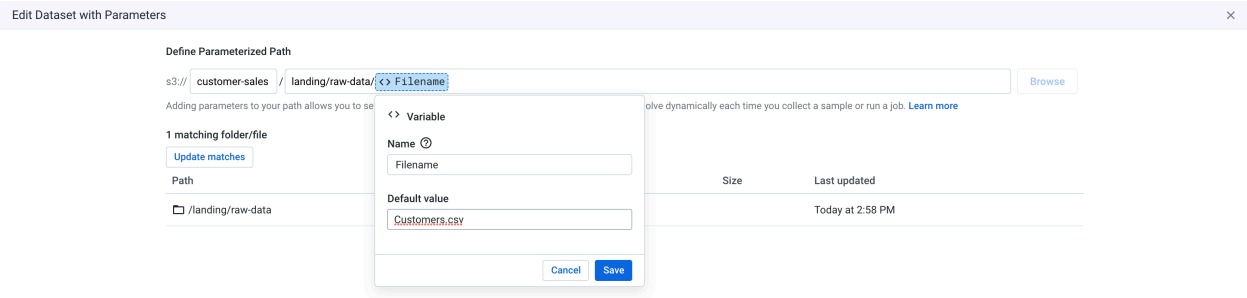
1. A Dynamic Customer CSV File, the source CSV file will be a variable
2. Occupation Mapping spreadsheet, a fixed mapping CSV file, that has information on customer labels and different categories



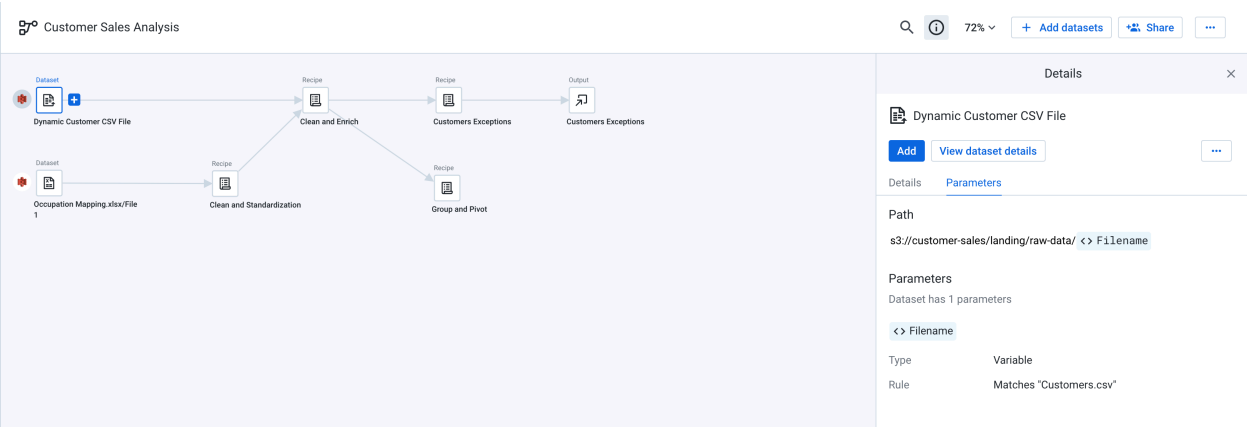
In order to update any parameters of the attached flow, Click on Edit parameters



In this example, we will create a variable called `FileName`. You can dynamically pass key, value references of a variable in a flow when invoking the API.

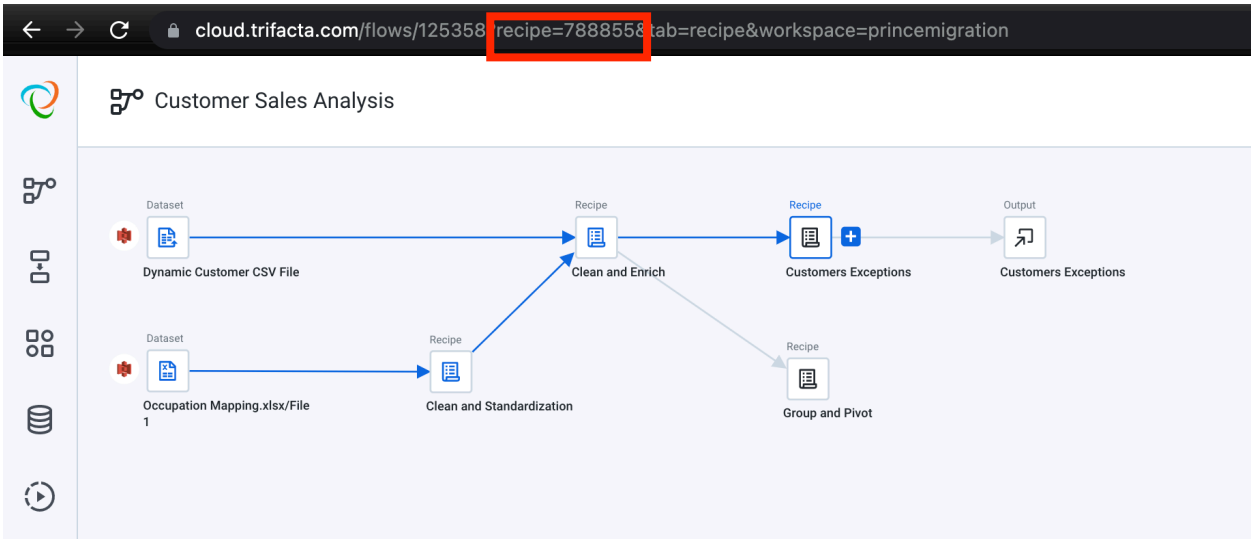


Select the **Parameters** tab in the flow view, to see and edit all flow parameters



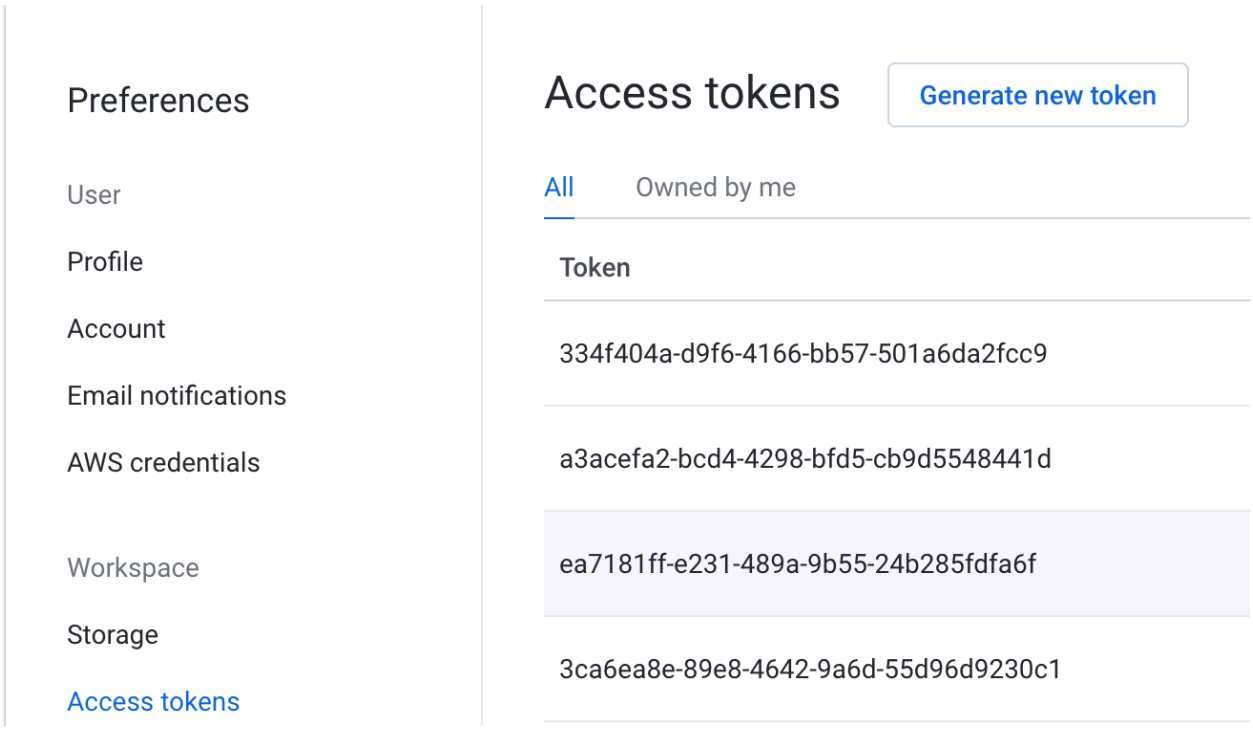
Step-2:

We want to trigger the Customers Exceptions recipe every time a new file lands in the source S3 bucket. Note the recipe ID from corresponding URL, we will need it for Step-5



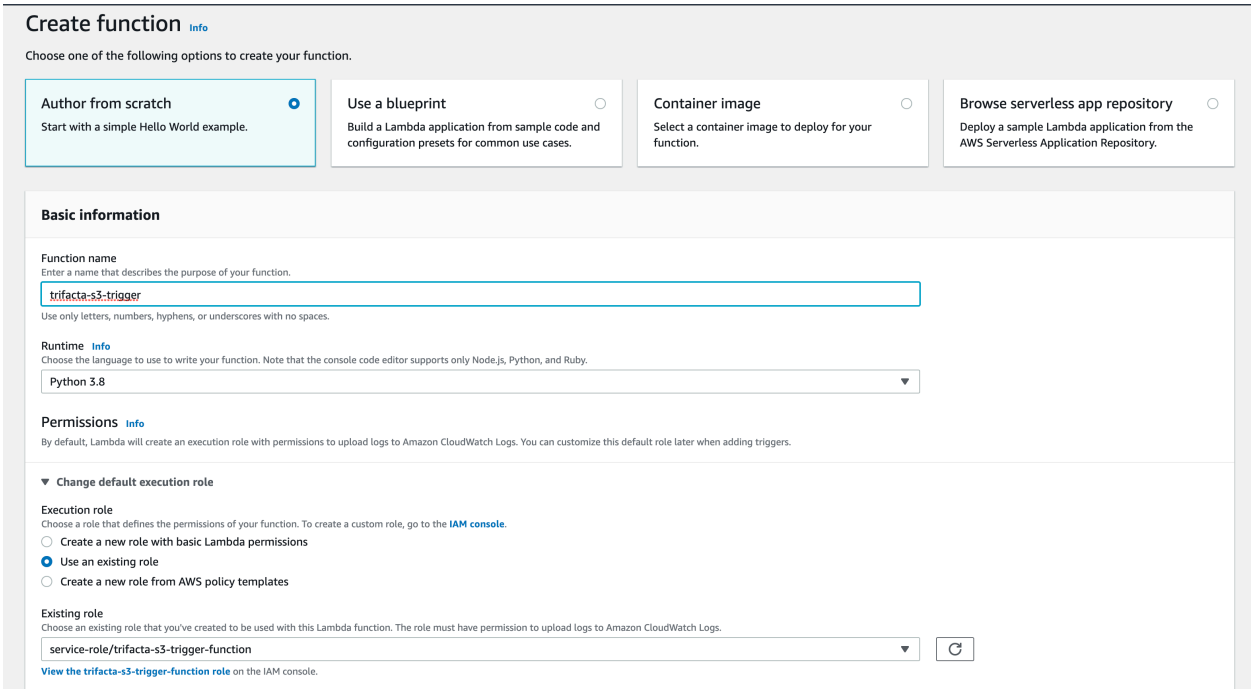
Step-3:

Trifacta requires a valid Access Token to make an API call. To generate an Access Token. Navigate to Preferences -> Access Token. Save the Access Token in a secured location, we will need it for Step-5



Step-4:

Let’s create a Lambda function, Using the attached python code



Step-5:

Update trifacta_auth_token, trifacta_wrangle_dataset_id, trifacta_runjob_endpoint per your environment

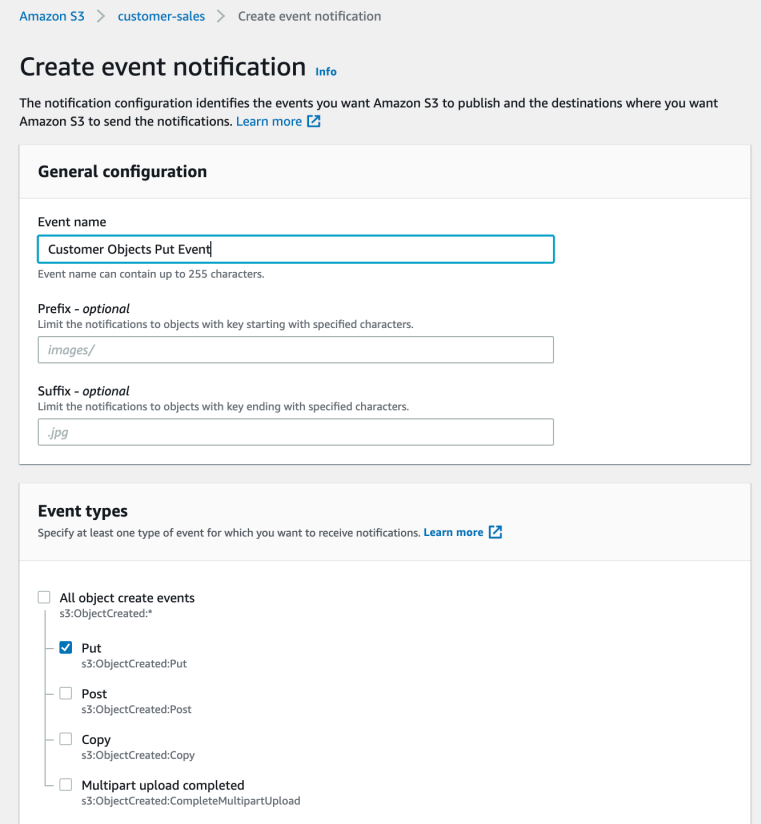
```
import json
import urllib.parse
import boto3
import os
import urllib3
from http.client import responses

s3 = boto3.client('s3')

def lambda_handler(event, context):
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
    trifacta_auth_token = 'eyJ0b2tlbklkIjoiYTNhY2VmYTItYmNkNC00Mjk4LWJmZDUtY2I5ZDU1NDg0NDFkIiwic2VjcmV0IjoiNzIwYTJhMDd
    trifacta_wrangle_dataset_id = 788855
    print('Run Trifacta job on new file: {}'.format(key))
    trifacta_runjob_endpoint = 'https:/[redacted].cloud.trifacta.com/v4/jobGroups'
    trifacta_job_param = {
        "wrangledDataset": {"id": trifacta_wrangle_dataset_id},
        "runParameters": {"overrides": {"data": [{"key": "filename","value": key}]}}
    }
    print('Run Trifacta job param: {}' .format(trifacta_job_param))
    trifacta_headers = {
        "Content-Type":"application/json",
        "Authorization": "Bearer "+trifacta_auth_token
    }
    http = urllib3.PoolManager()
    r = http.request('POST',trifacta_runjob_endpoint, headers=trifacta_headers,  body=json.dumps(trifacta_job_param))
    print('Status Code : {}'.format(r.status))
    print('Result : {}'.format(responses[r.status]))
    return 'End File event'.format(key)
```

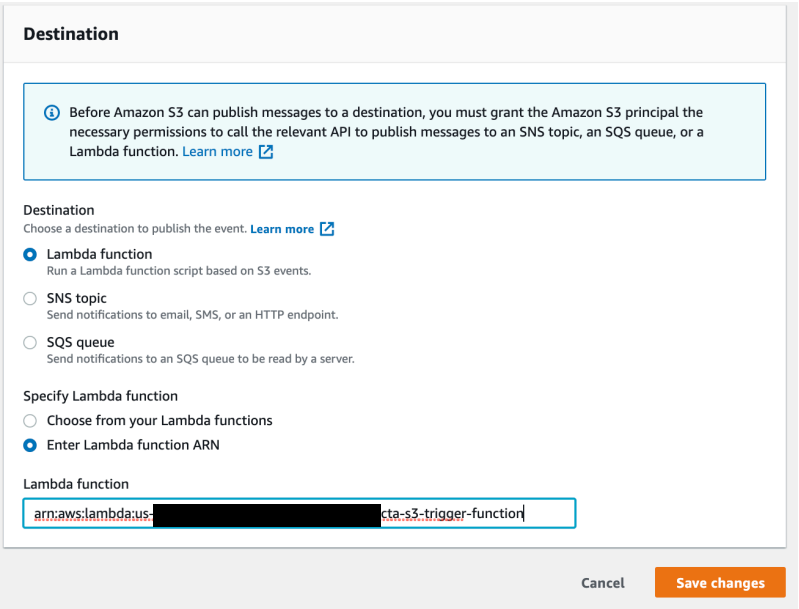
Step-6:

Go to S3 console, Select the Input source bucket and create an event-notification.

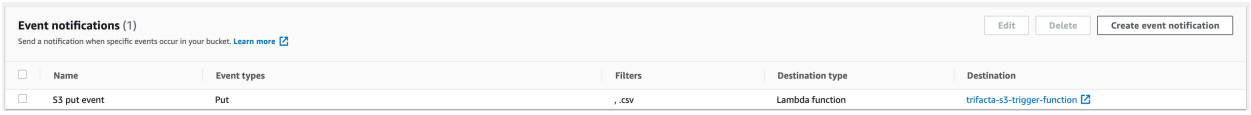


Step-7:

Once the event type is created, select the destination as a Lambda function.



Manage any updates under the Event Notifications section of S3 browser



Step-8:

To test the functionality end-to-end, upload a new file to the input source S3 bucket, and verify a new Trifacta job is triggered every time using the correct input file.