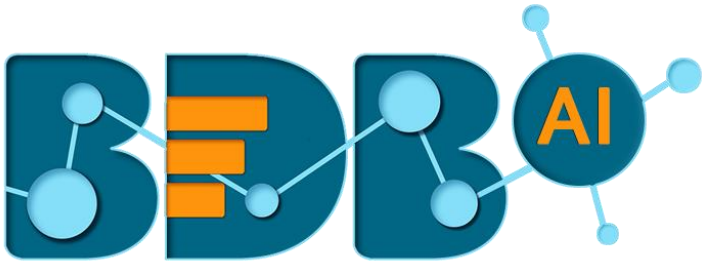


Data Pipeline How to Guide

Custom Python Component

Version: Release 2.0

Contents

1. Custom Python Script -

Python component works as normal python compile. The below given instructions should be followed while writing a Python script in the BDB Data Pipeline:

- The Python script needs to be written inside a valid Python function.

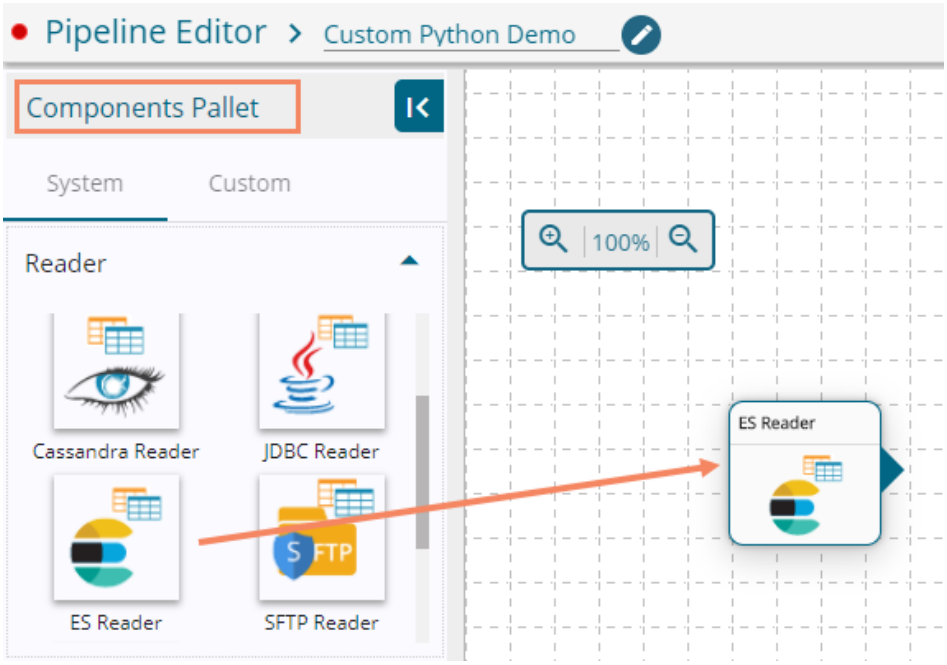
  E.g., The entire code body should be inside the proper indentation of the function (Use 4 spaces per indentation level.)
- The Python script should have at least one main function. Multiple functions are acceptable, and one function can call another function.
  - It should be written above the calling function body (if the called function is an outer function)
  - It should be written above the calling statement (if called function is an inner function)
- Spaces are the preferred indentation method.
- Do not use "type" as the function argument as it is a predefined keyword.
- The code in the core Python distribution should always use UTF-8.
- Single-quoted strings and double-quoted strings are considered the same in Python.
- All the packages used in function need to import explicitly before writing function.
- The Python script should return data in the form of a data frame or list only. The form of data should be defined while writing the function.
- If the user uses some Kafka event data for transformation, then the first argument of the function should be data frame or list.
- If the user needs to use some external library, the user needs to mention the library name in the external libraries field. If the user wants to use multiple external libraries, the library names should be separated by a comma.
- If you need to pass some external input in your main function, then you can use input data filed. Key name should be the same according to the variables name and value that is put as per the requirement.
- We can use that component as reader, transformation, and writer.

2. Steps by Step Process to Use Custom Python Script

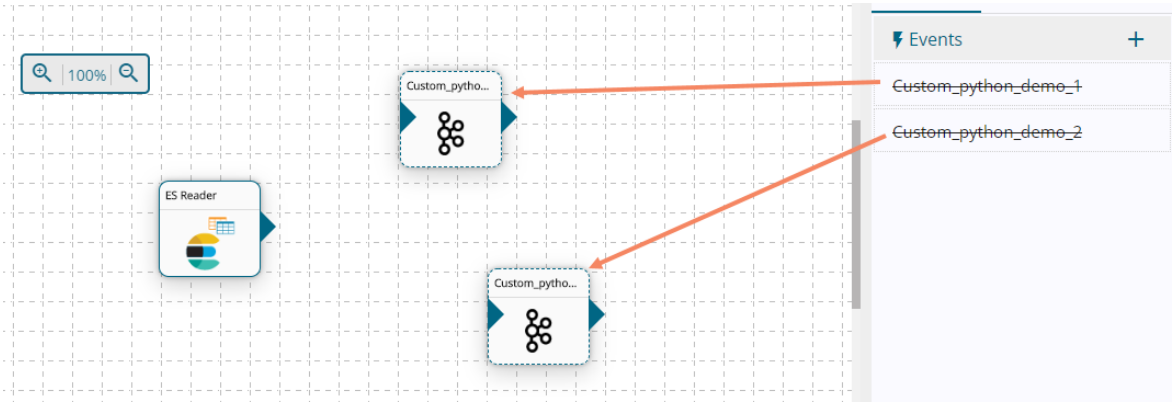
i) Create a new pipeline (Refer the steps mentioned in the **Create and Update Pipeline** document).
Note: the document link of the Create and Update Pipeline is provided at the end.

ii) One Reader is needed to transform data (In this case, the '**ES Reader**' is used).

iii) Drag the ES Reader component to the workspace.

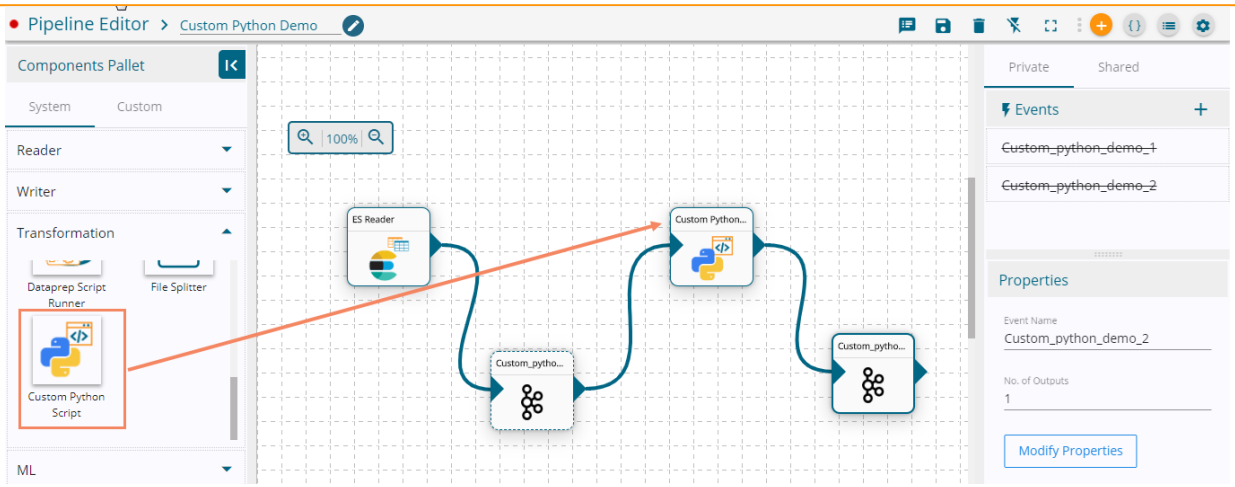


iv) Create two Kafka events and drag them to the workspace.



v) Drag the Custom Python component from the Transformation section of the Component Pallet.

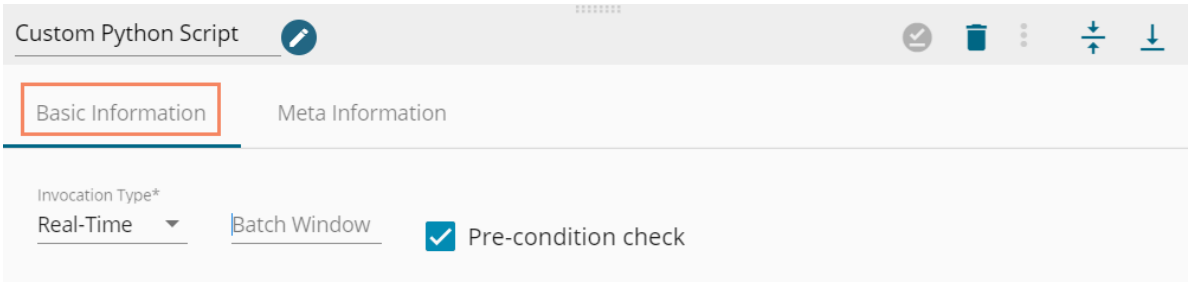
vi) Connect the ES Reader, the Kafka Events and Custom Python component as displayed below:[i]



vii) Click the dragged Custom Python component to get the Configuration fields:

viii) The Basic Information tab opens by default.

a. Select the invocation type (Real-Time/Batch)



ix) Click on the Meta Information tab to open the configuration fields.

a. Component Name: Provide a name for the Python Script component.
Note: The component name should be without space and special characters. Use the underscore symbol to show space in between words.

b. Python Script: Insert the Python script containing at least one function. The function should not have an argument, data frame argument, or custom argument.

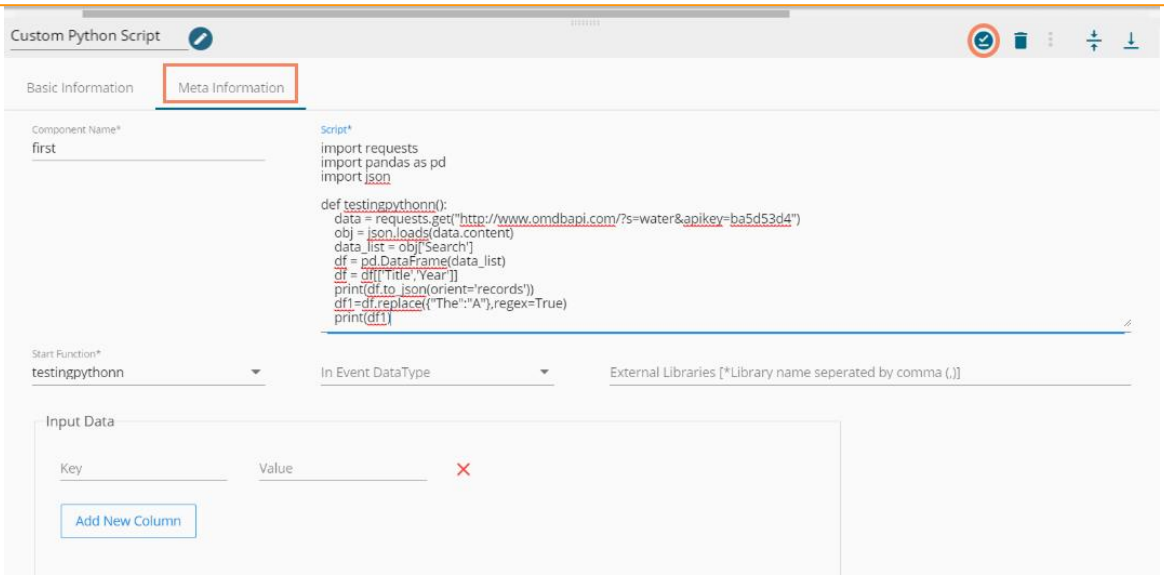
c. Start Function Name: It displays all the function names used in python script in a drop-down menu. Select one function name with which you want to start.

d. In Event Data Type: Provide input data type as a data frame or list.

e. External Library: Provide the external library name in this field. Insert multiple library names separated by commas.

f. Input Data: Use custom argument name as key and provide the required value.

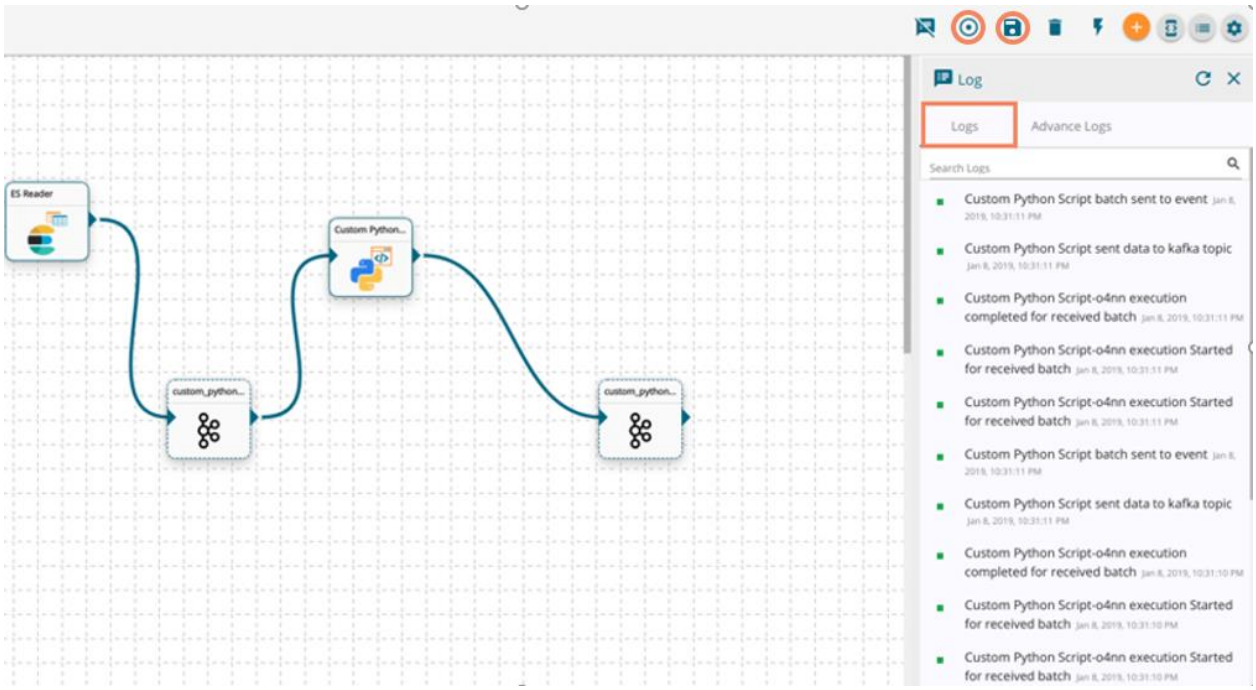
x) Click the '**Save**' icon.



xi) Save the Pipeline workflow. (After getting the success message,

xii) Start the Pipeline workflow.

xiii) Open the Log section to see the logs.



Python Script Examples:

The Custom Python Script transform component supports 3 types of scripts in the Data Pipeline.

1. Without argument (Like Read Data): If you don't have any in Event then you can use no argument function. For Example,

   i. import json

   ii. import requests

iii. import pandas as pd

iv. def getmovies_result():

v. data = requests.get("http://www.omdbapi.com/?s=water&apikey=ba5d53d4")

vi. loaded_json = json.loads(data.content)

vii. data = loaded_json['Search']

viii. df = pd.DataFrame.from_dict(data, orient='columns')

ix. return df

2. With argument (Like Transformation): If you have data frame to execute some operation. Then use first argument as data frame. For Example,

   i. def getcsvdata(df):

   ii. cond1 = df['Unit Price'] > 450

   iii. filter_df = df[cond1]

   iv. return filter_df

3. Custom Argument with Data frame: If there is custom argument with data frame means Input Event then first argument is always data frame variable. For Example,

   i. def getcsvdata(df, range):

   ii. cond1 = df['Unit Price'] > range

   iii. filter_df = df[cond1]

   iv. return filter_df

[i] Document link for the Create and Update Pipeline - https://08009ad7bf1979094b0b-3488c35d3ab28aac7529e703b5435d94.ssl.cf1.rackcdn.com/BDB%20Documentation%20-%203.0/BDB%204.2/Create%20and%20Update%20Pipeline_DP_2.2.pdf